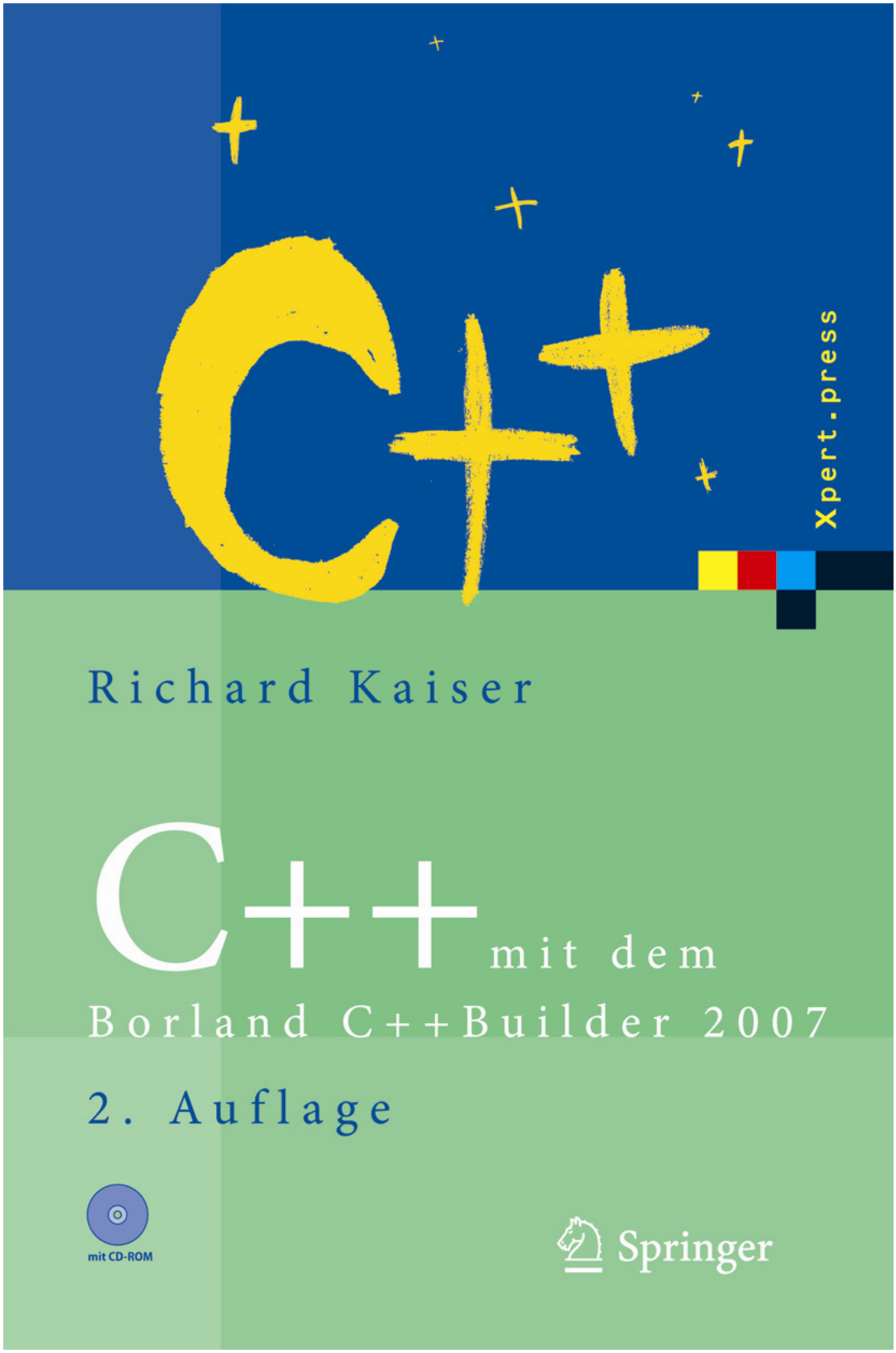
Aufgaben und Lösungen

zu



Richard Kaiser

Aufgaben und Lösungen

zu

C++ mit dem Borland C++Builder 2007

Einführung in den C++-Standard und die objektorientierte Windows-Programmierung

2. überarbeitete Auflage

Version vom 15.9.2007

Springer

Prof. Richard Kaiser
Schwärzlocher Straße 53
72070 Tübingen
http://www.rkaiser.de

ISBN 978-3-540-69575-2 e-ISBN 978-3-540-69773-2
DOI 10.1007/978-3-540-69773-2
ISSN 1439-5428

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Einbandgestaltung: KünkelLopkaWerbeagentur, Heidelberg

Gedruckt auf säurefreiem Papier

9 8 7 6 5 4 3 2 1

springer.com

Vorwort

Dieser Text enthält alle Aufgaben und Lösungen zu meinem Buch „C++ mit dem Borland C++Builder 2007“. Die Lösungen sind auch als vollständige Projekte auf der CD im Buch.

Damit der Bezug zwischen dem Buchtext und den Aufgaben leichter hergestellt werden kann, habe ich alle Kapitelüberschriften aus dem Buch in diese Aufgabensammlung übernommen.

Anregungen, Korrekturhinweise und Verbesserungsvorschläge sind willkommen. Bitte senden Sie diese an die EMail-Adresse auf meiner Internetseite *http://www.rkaiser.de*.

Falls Sie Interesse an einer gedruckten Version dieser Aufgaben- und Lösungssammlung haben, bitte eine kurze EMail. Der Verlag wird sie dann eventuell auch als Buch veröffentlichen.

Tübingen, im September 2007 Richard Kaiser

Für Ruth

Geleitwort

Das Programmieren unter C++ gilt als die Königsklasse der objektorientierten Applikations-Entwicklung: Anwender nutzen C++, um universell einsetzbare, modulare Programme zu erstellen. Wer diese Sprache beherrscht, profitiert von einem beispiellosen Funktionsumfang und von der Option, plattformunabhängig zu arbeiten. Das war anfangs nur hochgradig versierten Profis vorbehalten. Sie allein waren in der Lage, der Komplexität des C++-Quellcodes Herr zu werden.

Längst aber stehen die Vorzüge von C++ auch all jenen zur Verfügung, die nur gelegentlich oder schlicht und ergreifend aus Freude am Tüfteln Applikationen erstellen. Einen wesentlichen Beitrag zur „Demokratisierung" der objektorientierten Programmierung leisten integrierte RAD-Systeme (Rapid Application Development) wie der C++Builder von Borland.

Ganz gleich ob Profi oder Einsteiger: Die C++-Version der erfolgreichen Object Pascal-Lösung *Borland Delphi* bietet Programmierern eine visuelle Entwicklungsumgebung, mit der sie einfach und rasch objektorientierte Windows-Applikationen schreiben können. Der C++Builder verfügt über eine umfangreiche Palette an fertigen Komponenten und erleichtert seit der Version 5 auch die Entwicklung von Web-Applikationen. Wer grafische Benutzeroberflächen bauen will, stellt diese einfach mit wenigen Handgriffen per Maus zusammen. Das ist die Basis für ein schnelles, effizientes und komfortables Arbeiten. Kurzum: Mit dem C++Builder wird die Applikations-Entwicklung von der langwierigen Fleißaufgabe zur zielorientierten Kopfarbeit.

Das vorliegende Buch ist eine systematische Einführung in die Arbeit mit C++ und dem Borland C++Builder. Ausführlich und praxisnah schildert Richard Kaiser die Konzepte und Elemente der Programmiersprache und der Entwicklungsumgebung. Mit zahlreichen Beispielen und Übungsaufgaben erschließt er auch Lesern ohne Vorkenntnisse die Logik objektorientierten Programmierens.

Borland wünscht allen Nutzern dieses hervorragenden Lehrbuchs und Nachschlagewerks viel Spaß und Erfolg bei der Arbeit mit dem C++Builder.

Jason Vokes
European Product Line Manager – RAD Products and InterBase

Vorwort zur 2. Auflage

Nach nunmehr fünf Jahren liegt jetzt die zweite Auflage des „Builder-Buches“ vor. In dieser Zeit habe ich zahlreiche Vorlesungen und Industrieseminare auf der Basis der ersten Auflage gehalten. Dabei ergaben sich immer wieder Ansatzpunkte für Verbesserungen, Fehlerkorrekturen, Präzisierungen, Erweiterungen und Straffungen des Textes. Das Ergebnis ist eine komplette Überarbeitung der ersten Auflage.

Folgende Themen wurden für die zweite Auflage zusätzlich aufgenommen bzw. grundlegend erweitert:

- Änderungen gegenüber dem C++Builder 5
- systematische Tests und Unit Tests (Abschnitt 3.5)
- Programmierlogik und die Programmverifikation (Abschnitt 3.7)
- Objektorientierte Analyse und Design (Kapitel 6)

Die meisten Ausführungen gelten für den **C++Builder 2007** ebenso wie für den C++Builder 2006 oder noch ältere Versionen (C++Builder 5 und 6). Dabei ist es auch unerheblich, dass **Borland** den C++Builder inzwischen in eine eigene Firma mit dem Namen **CodeGear** ausgelagert hat. Da sich auch die meisten Screenshots und Verweise (z.B. unter *Projekt|Optionen*) bei den verschiedenen Versionen nur wenig unterscheiden, wurden nur die wichtigsten auf den kurz vor der Fertigstellung des Buches erschienenen C++Builder 2007 angepasst.

Tübingen, im August 2007 Richard Kaiser

Vorwort zur 1. Auflage

Dieses Buch entstand ursprünglich aus dem Wunsch, in meinen Vorlesungen über C++ nicht nur Textfensterprogramme, sondern Programme für eine grafische Benutzeroberfläche zu entwickeln. Mit dem C++Builder von Borland stand 1997 erstmals ein Entwicklungssystem zur Verfügung, das so einfach zu bedienen war, dass man es auch in Anfängervorlesungen einsetzen kann, ohne dabei Gefahr zu laufen, dass die Studenten nur noch mit dem Entwicklungssystem kämpfen und gar nicht mehr zum Programmieren kommen.

Angesichts der damals anstehenden Verabschiedung des ANSI/ISO-Standards von C++ lag es nahe, in diesem einführenden Lehrbuch auch gleich den gesamten Sprachumfang des Standards umfassend darzustellen. Mir war allerdings nicht klar, auf welche Arbeit ich mich damit eingelassen hatte. Ich hatte weder vor, vier Jahre an diesem Buch zu schreiben, noch einen Wälzer mit 1100 Seiten zu produzieren.

Als ich dann die Möglichkeit bekam, Kurse für erfahrene Praktiker aus der Industrie zu halten, wurde ich mit einer Fülle von Anregungen aus ihrer täglichen Arbeit konfrontiert. Diese gaben dem Buch enorme Impulse.

Die Programmiersprache C++ wurde als Obermenge der Programmiersprache C entworfen. Dieser Entscheidung verdankt C++ sicher seine weite Verbreitung. Sie hat aber auch dazu geführt, dass oft weiterhin wie in C programmiert wird und lediglich ein C++-Compiler anstelle eines C-Compilers verwendet wird. Dabei werden viele Vorteile von C++ verschenkt. Um nur einige zu nennen:

- In C++ werden die fehleranfälligen Zeiger viel seltener als in C benötigt.
- Die Stringklassen lassen sich wesentlich einfacher und risikoloser als die nullterminierten Strings von C verwenden.
- Die Containerklassen der C++-Standardbibliothek haben viele Vorteile gegenüber Arrays, selbstdefinierten verketteten Listen oder Bäumen.
- Exception-Handling bietet eine einfache Möglichkeit, auf Fehler zu reagieren.
- Objektorientierte Programmierung ermöglicht übersichtlichere Programme.
- Templates sind die Basis für eine außerordentlich vielseitige Standardbibliothek.

Ich habe versucht, bei allen Konzepten nicht nur die Sprachelemente und ihre Syntax zu beschreiben, sondern auch Kriterien dafür anzugeben, wann und wie man sie sinnvoll einsetzen kann. Deshalb wurde z.B. mit der objektorientierten Programmierung eine Einführung in die objektorientierte Analyse und das objektorientierte Design verbunden. Ohne die Beachtung von Design-Regeln schreibt man leicht Klassen, die der Compiler zwar übersetzen kann, die aber kaum hilfreich sind.

Man hört immer wieder die Meinung, dass C++ viel zu schwierig ist, um es als einführende Programmiersprache einzusetzen. Dieses Buch soll ein in mehreren Jahren erprobtes Gegenargument zu dieser Meinung sein. Damit will ich aber die Komplexität von C++ überhaupt nicht abstreiten.

Zahlreiche Übungsaufgaben geben dem Leser die Möglichkeit, die Inhalte praktisch anzuwenden und so zu vertiefen. Da man Programmieren nur lernt, indem man es tut, möchte ich ausdrücklich dazu ermuntern, zumindest einen Teil der Aufgaben zu lösen und sich dann selbst neue Aufgaben zu stellen. Der Schwierigkeitsgrad der Aufgaben reicht von einfachen Wiederholungen des Textes bis zu kleinen Projektchen, die ein gewisses Maß an selbständiger Arbeit erfordern. Die Lösungen der meisten Aufgaben findet man auf der beiliegenden CD und auf meiner Internetseite *http://www.rkaiser.de*.

Anregungen, Korrekturhinweise und Verbesserungsvorschläge sind willkommen. Meine EMail-Adresse finden Sie auf meiner Internetseite.

Bei allen meinen Schulungskunden und ganz besonders bei Herrn Welsner und der Alcatel University der Alcatel SEL AG Stuttgart bedanke ich mich für die Möglichkeit, dieses Manuskript in zahlreichen Kursen mit erfahrenen Praktikern

weiterzuentwickeln. Ohne die vielen Anregungen aus diesen Kursen hätte es weder diesen Praxisbezug noch diesen Umfang erreicht. Peter Schwalm hat große Teile des Manuskripts gelesen und es in vielen Diskussionen über diffizile Fragen mitgestaltet. Mein Sohn Alexander hat als perfekter Systembetreuer dafür gesorgt, dass die Rechner immer liefen und optimal installiert waren.

Die Unterstützung von Dr. Hans Wössner und seinem Team vom Springer-Verlag hätte nicht besser sein können. Seine Hilfsbereitschaft und seine überragende fachliche Kompetenz waren immer wieder beeindruckend. „Meiner“ Lektorin Ruth Abraham verdankt dieses Buch eine in sich geschlossene Form, die ich allein nicht geschafft hätte. Die technische Herstellung war bei Gabi Fischer in erfahrenen guten Händen. Herrn Engesser danke ich für die gute Zusammenarbeit beim Abschluss des Projekts.

Tübingen, im Oktober 2001 Richard Kaiser

Inhalt

ϴ Angesichts des Umfangs dieses Buches habe ich einige Abschnitte mit dem Zeichen ϴ in der Überschrift als „weniger wichtig“ gekennzeichnet. Damit will ich dem Anfänger eine kleine Orientierung durch die Fülle des Stoffes geben. Diese Kennzeichnung bedeutet aber keineswegs, dass dieser Teil unwichtig ist – vielleicht sind gerade diese Inhalte für Sie besonders relevant.

1 Die Entwicklungsumgebung

1.1 Visuelle Programmierung: Ein erstes kleines Programm

1.2 Erste Schritte in C++

1.3 Der Quelltexteditor

1.4 Kontextmenüs und Symbolleisten (Toolbars)

1.5 Projekte, Projektdateien und Projektoptionen

1.6 Einige Tipps zur Arbeit mit Projekten

1.7 Die Online-Hilfe

1.8 Projektgruppen und die Projektverwaltung Θ

1.9 Hilfsmittel zur Gestaltung von Formularen Θ

1.10 Packages und eigenständig ausführbare Programme Θ

1.11 Win32-API und Konsolen-Anwendungen Θ

1.12 Windows-Programme und Units Θ

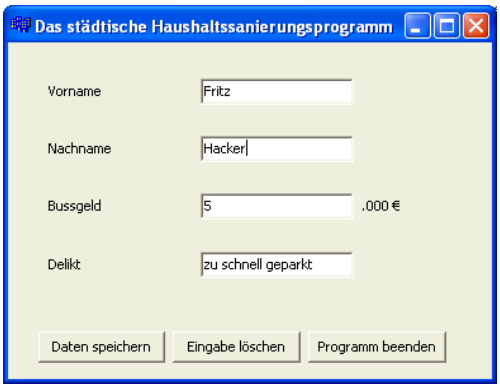
2 Komponenten für die Benutzeroberfläche

2.1 Die Online-Hilfe zu den Komponenten

2.2 Namen

Aufgaben 2.2

1. Schreiben Sie ein Programm, das ein Fenster mit folgenden Elementen anzeigt:



   Verwenden Sie dazu die Komponenten *Label*, *Edit* und *Button* aus dem Abschnitt *Standard* der Tool Palette.

2. Ersetzen Sie alle vom C++Builder vergebenen Namen durch aussagekräftige Namen. Da in diesem Beispiel sowohl ein Label als auch ein Edit-Fenster für den Vornamen, Nachnamen usw. verwendet wird, kann es sinnvoll sein, den Typ der Komponente im Namen zu berücksichtigen, z.B. *LVorname* und *LNachname* für die Label und *EVorname* und *ENachname* für die Edit-Fenster.

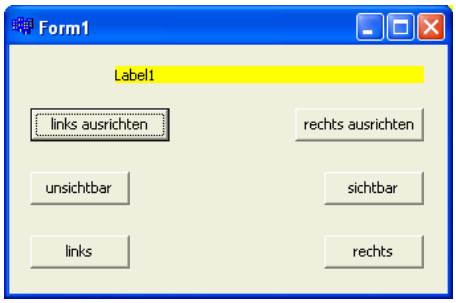
3. Als Reaktion auf ein Anklicken des Buttons *Eingabe löschen* soll jedes Eingabefeld mit der Methode *Clear* gelöscht werden. Für den Button *Daten speichern* soll keine weitere Reaktion vorgesehen werden. Beim Anklicken des Buttons *Programm beenden* soll das Formular durch den Aufruf der Methode *Close* geschlossen werden.

4. Alle Labels sollen in derselben Spaltenposition beginnen, ebenso die TextBoxen. Die Buttons sollen gleich groß sein und denselben Abstand haben.

2.3 Labels, Datentypen und Compiler-Fehlermeldungen

Aufgabe 2.3

Schreiben Sie ein Programm, das nach dem Start dieses Fenster anzeigt:



Für die folgenden Ereignisbehandlungsroutinen müssen Sie sich in der Online-Hilfe über einige Eigenschaften informieren, die bisher noch nicht vorgestellt wurden.

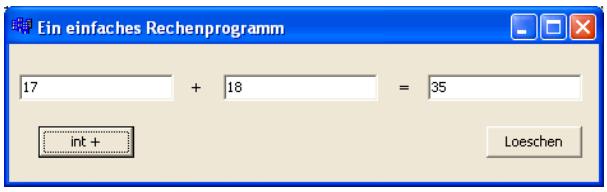
Beim Anklicken der Buttons mit der Aufschrift

– *ausrichten* soll der Text im Label mit Hilfe der Eigenschaft *Alignment* (siehe Online-Hilfe, nicht mit *Align* verwechseln) links bzw. rechts ausgerichtet werden.
  Damit das Label sichtbar ist, soll seine Farbe z.B. auf Gelb gesetzt werden.
  Damit die Größe des Labels nicht der Breite des Textes angepasst wird, soll *Autosize* (siehe Online-Hilfe) auf *false* gesetzt werden.
– *sichtbar/unsichtbar* soll das Label sichtbar bzw. unsichtbar gemacht werden,
– *links/rechts* soll das Label so verschoben werden, dass sein linker bzw. rechter Rand auf dem linken bzw. rechten Rand des Formulars liegt. Damit der rechte Rand des Labels genau auf den rechten Rand des Formulars gesetzt wird, verwenden Sie die Eigenschaft *ClientWidth* (siehe Online-Hilfe) eines Formulars.

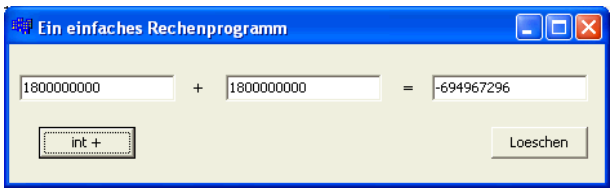
2.4 Funktionen, Methoden und die Komponente *TEdit*

Aufgaben 2.4

1. Schreiben Sie ein einfaches Rechenprogramm, mit dem man zwei Ganzzahlen addieren kann. Durch Anklicken des *Löschen*-Buttons sollen sämtliche Eingabefelder gelöscht werden.



   Offensichtlich produziert dieses Programm falsche Ergebnisse, wenn die Summe außerhalb des Bereichs –2147483648 .. 2147483647 ($-2^{31}..2^{31}-1$) liegt:



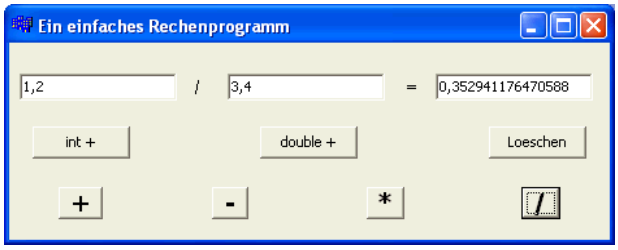
Die Ursache für diese Fehler werden wir später kennen lernen.

2. Ergänzen Sie das Programm aus Aufgabe 1 um einen Button, mit dem auch Zahlen mit Nachkommastellen wie z.B. 3,1415 addiert werden können. Verwenden Sie dazu die Funktionen

   long double ***StrToFloat****(AnsiString S);*
   AnsiString ***FloatToStr****(long double Value);*
   // weitere Informationen dazu in der Online-Hilfe

   Der Datentyp *long double* ist einer der Datentypen, die in C++ Zahlen mit Nachkommastellen darstellen können. Solche Datentypen haben einen wesentlich größeren Wertebereich als der Ganzzahldatentyp *int*. Deshalb treten Bereichsüberschreitungen nicht so schnell auf.

3. Ergänzen Sie das Programm aus Aufgabe 2 um Buttons für die Grundrechenarten +, –, * und /. Die Aufschrift auf den Buttons soll im Objektinspektor über die Eigenschaft ***Font*** auf 14 Punkt und fett (Font|Style *fsBold*) gesetzt werden. Die jeweils gewählte Rechenart soll in einem Label zwischen den beiden Operanden angezeigt werden:

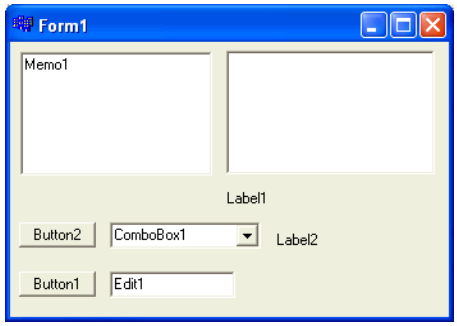


4. Geben Sie im laufenden Programm im ersten Eingabefeld einen Wert ein, der nicht in eine Zahl umgewandelt werden kann, und setzen Sie das Programm anschließend fort.

2.5 Memos, ListBoxen, ComboBoxen und die Klasse *TStrings*

Aufgabe 2.5

Schreiben Sie ein Programm mit einem Memo, einer ListBox, einer ComboBox, einem Edit-Fenster, zwei Buttons und zwei Labels:



a) Beim Anklicken von *Button1* soll der aktuelle Text des Edit-Fensters als neue Zeile zu jeder der drei *TStrings*-Listen hinzugefügt werden.
b) Wenn ein *ListBox*-Eintrag angeklickt wird, soll er auf Label1 angezeigt werden.
c) Beim Anklicken von *Button2* soll der in der ComboBox ausgewählte Text auf dem *Label2* angezeigt werden.

2.6 Buttons und Ereignisse

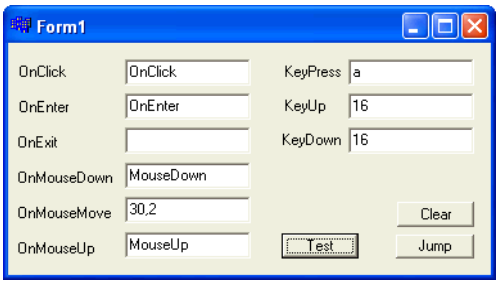
2.6.1 Parameter der Ereignisbehandlungsroutinen

2.6.2 Der Fokus und die Tabulatorreihenfolge

2.6.3 BitButtons und einige weitere Eigenschaften von Buttons

Aufgabe 2.6

Schreiben Sie ein Programm, das etwa folgendermaßen aussieht:



a) Wenn für den „Test“-Button eines der Ereignisse *OnClick*, *OnEnter* usw. eintritt, soll ein Text in das zugehörige Edit-Fenster geschrieben werden. Für die Ereignisse, für die hier keine weiteren Anforderungen gestellt werden, reicht ein einfacher Text, der das Ereignis identifiziert (z.B. „OnClick“).

   Bei den Key-Ereignissen soll der Wert des Parameters *Key* angezeigt werden. Beachten Sie dabei, dass dieser Parameter bei der Funktion

```
void __fastcall TForm1::Button1KeyPress(TObject *Sender, char &Key)
```

   den Datentyp *char* hat und der Eigenschaft *Text* des Edit-Fensters direkt zugewiesen werden kann, während er bei den Funktionen *KeyDown* und *KeyUp* den Datentyp *WORD* hat:

```
void __fastcall TForm1::Button1KeyDown(TObject *Sender, WORD &Key, TShiftState Shift)
```

   Dieser Datentyp kann mit *IntToStr* in einen String umgewandelt und dann der Eigenschaft *Text* des Edit-Fensters zugewiesen werden.

   Beim Ereignis *MouseMove* sollen die Mauskoordinaten angezeigt werden, die als Parameter X und Y übergeben werden. Diese können mit *IntToStr* in einen String umgewandelt und mit + zu einem String zusammengefügt werden. Als Trennzeichen soll dazwischen noch mit + ein String "," eingefügt werden.

b) Mit dem Button *Clear* sollen alle Anzeigen gelöscht werden können.

c) Beobachten Sie, welche Ereignisse eintreten, wenn der Test-Button
   – angeklickt wird
   – den Fokus hat und eine Taste auf der Tastatur gedrückt wird
   – den Fokus hat und eine Funktionstaste (z.B. F1, Strg, Alt) gedrückt wird
   – mit der Tab-Taste den Fokus bekommt.

d) Die Tabulatorreihenfolge der Edit-Fenster soll ihrer Reihenfolge auf dem Formular entsprechen (zuerst links von oben nach unten, dann rechts).

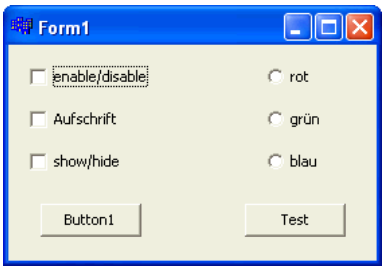
e) Der Text der Titelzeile „Events“ des Formulars soll im Konstruktor des Formulars zugewiesen werden.

f) Der „Jump“-Button soll immer an eine andere Position springen (z.B. an die gegenüberliegende Seite des Formulars), wenn er vom Mauszeiger berührt wird. Dazu ist keine *if*-Anweisung notwendig. Falls er angeklickt wird, soll seine Aufschrift auf „getroffen“ geändert werden.

2.7 CheckBoxen, RadioButtons und einfache *if*-Anweisungen

Aufgabe 2.7

Schreiben Sie ein Programm mit drei CheckBoxen, zwei RadioButtons und zwei gewöhnlichen Buttons:



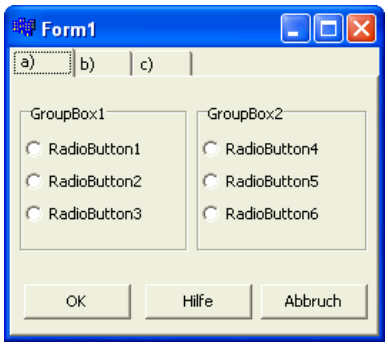
Beim Anklicken des Buttons *Test* sollen in Abhängigkeit von den Markierungen der CheckBoxen und der RadioButtons folgende Aktionen stattfinden:

a) Die Markierung der CheckBox *enable/disable* soll entscheiden, ob bei der zweiten CheckBox, dem ersten RadioButton sowie bei *Button1* die Eigenschaft *Enabled* auf *true* oder *false* gesetzt wird.
b) Die Markierung der CheckBox *Aufschrift* soll entscheiden, welchen von zwei beliebigen Texten *Button1* als Aufschrift erhält.
c) Die Markierung der CheckBox *show/hide* soll entscheiden, ob *Button1* angezeigt wird oder nicht.
d) Die Markierung der RadioButtons soll die Hintergrundfarbe der ersten CheckBox festlegen.
e) Beim Anklicken des ersten RadioButtons soll die Hintergrundfarbe der ersten CheckBox auf rot gesetzt werden.

2.8 Die Container GroupBox, Panel und PageControl

Aufgabe 2.8

Ein Formular soll ein PageControl mit drei Registerkarten enthalten, das das ganze Formular ausfüllt. Die Registerkarten sollen den einzelnen Teilaufgaben dieser Aufgabe entsprechen und die Aufschriften „a)“, „b)“ und „c)“ haben.



a) Die Seite „a)“ soll zwei Gruppen von sich gegenseitig ausschließenden Optionen enthalten.
Die Buttons *OK*, *Hilfe* und *Abbruch* zu einer Gruppe zusammengefasst werden, die optisch nicht erkennbar ist.
Reaktionen auf das Anklicken der Buttons brauchen nicht definiert werden.
b) Die Seite „b)“ soll nur einen Button enthalten.
c) Die Seite „c)“ soll leer sein.
d) Verwenden Sie die Struktur-Anzeige, um einen Button von Seite „b)“ auf Seite „c)“ zu verschieben.

2.9 Hauptmenüs und Kontextmenüs

2.9.1 Hauptmenüs und der Menüdesigner

2.9.2 Kontextmenüs

2.9.3 Die Verwaltung von Bildern mit ImageList Θ

2.9.4 Menüvorlagen speichern und laden Θ

2.10 Standarddialoge

2.10.1 Einfache Meldungen mit *ShowMessage*

Aufgabe 2.10

Schreiben Sie ein Programm mit einem Hauptmenü, das die unten aufgeführten Optionen enthält. Falls die geforderten Anweisungen bisher nicht vorgestellt wurden, informieren Sie sich in der Online-Hilfe darüber (z.B. *SelectAll*). Das Formular soll außerdem ein Memo enthalten, das den gesamten Client-Bereich des Formulars ausfüllt (Eigenschaft *Align=alClient*).

- *Datei\Öffnen*: Falls ein Dateiname ausgewählt wird, soll diese Datei mit der Methode *LoadFromFile* in das Memo eingelesen werden. In dem OpenDialog sollen nur die Dateien mit der Endung „.txt“ angezeigt werden.
- *Datei\Speichern*: Falls ein Dateiname ausgewählt wird, soll der Text aus dem Memo mit der Methode *SaveToFile* von *Memo1->Lines* unter diesem Namen gespeichert werden. Diese Option soll außerdem mit dem ShortCut „Strg+S“ verfügbar sein.
- Nach einem Trennstrich.
- *Datei\Schließen*: Beendet die Anwendung

- *Bearbeiten\Suchen*: Ein *FindDialog* ohne jede weitere Aktion.
- *Bearbeiten\Suchen und Ersetzen*: Ein *ReplaceDialog* ohne jede weitere Aktion.
- Nach einem Trennstrich:
- *Bearbeiten\Alles Markieren*: Ein Aufruf von *Memo1->SelectAll*.
- *Bearbeiten\Ausschneiden*: Ein Aufruf von *Memo1->CutToClipboard*.
- *Bearbeiten\Kopieren*: Ein Aufruf von *Memo1->CopyToClipboard*.

- *Drucken\Drucken*. Ein *PrintDialog* ohne weitere Aktion. Mit den bisher vorgestellten Sprachelementen ist es noch nicht möglich, den Inhalt des Memos auszudrucken.
- *Drucken\Drucker einrichten*: Ein *PrinterSetupDialog* ohne weitere Aktion.

Durch Drücken der rechten Maustaste im Memo soll ein Kontextmenü mit den Optionen *Farben* und *Schriftart* aufgerufen werden:

- Popup-Menü*Farben*: Die ausgewählte Farbe (Eigenschaft *Color* des *ColorDialogs*) soll der Eigenschaft *Brush->Color* des Memos zugewiesen werden.
- Popup-Menü*Schriftart*: Die ausgewählte Schriftart (Eigenschaft *Font* des *FontDialogs*) soll der Eigenschaft *Font* des Memos zugewiesen werden.

3 Elementare Datentypen und Anweisungen

3.1 Syntaxregeln

Aufgabe 3.1

Geben Sie drei Beispiele für Ziffernfolgen an, die nach den Syntaxregeln für ein *decimal-literal* gebildet werden können, sowie drei Beispiele, die diese Regeln nicht einhalten. Formulieren Sie diese Syntaxregel außerdem verbal.

decimal-literal:
  nonzero-digit
  decimal-literal digit

nonzero-digit: one of
  `1 2 3 4 5 6 7 8 9`

digit: one of
  `0 1 2 3 4 5 6 7 8 9`

3.2 Variablen und Bezeichner

Aufgabe 3.2

1. Begründen Sie für jede dieser Definitionen, ob sie zulässig ist oder nicht:

```
int Preis_in_$, x kleiner y, Zinssatz_in_%, x/y, this,
    Einwohner_von_Tübingen, àáãÃéÉêÊ;
```

2. Welche Werte werden beim Anklicken von *Button1* ausgegeben?

```
int i=0;
int j;

void __fastcall TForm1::Button1Click(TObject *Sender)
{
int k=1;
Memo1->Lines->Add(IntToStr(i));
Memo1->Lines->Add(IntToStr(j));
Memo1->Lines->Add(IntToStr(k));
int i;
Memo1->Lines->Add(IntToStr(i));
}
```

3.3 Ganzzahldatentypen

3.3.1 Die interne Darstellung von Ganzzahlwerten

3.3.2 Ganzzahlliterale und ihr Datentyp

Aufgaben 3.3.2

1. Stellen Sie mit 8 Bits

   a) die Zahl 37 im Binärsystem dar
   b) die Zahl -37 im Zweierkomplement dar
   c) die Zahlen 37 und -37 im Hexadezimalsystem dar.

   Führen Sie die folgenden Berechnungen im Binärsystem durch und geben Sie das Ergebnis im Dezimalsystem an:

   d) 37 – 25 // berechnen Sie 37 + (-25)
   e) 25 – 37 // berechnen Sie 25 + (-37)

2. Welchen Wert stellt das Bitmusters ab_{16} im Dezimalsystem dar, wenn es

   a) im Zweierkomplement interpretiert wird?
   b) im Binärsystem interpretiert wird?

3. Welche Werte werden durch die folgenden Anweisungen ausgegeben:

```
Memo1->Lines->Add(IntToStr(030)); // Vorwahl Berlin
Memo1->Lines->Add(IntToStr(017+15));
Memo1->Lines->Add(IntToStr(0x12+10));
```

3.3.3 Zuweisungen und Standardkonversionen bei Ganzzahlausdrücken

3.3.4 Operatoren und die „üblichen arithmetischen Konversionen“

Aufgaben 3.3.4

1. Welchen Wert haben die Variablen b, c, d, e und f nach den Zuweisungen

```
unsigned char b = 255;
unsigned char c = 0;
int i = 1234567898;

b = b + 1;
c = c - 1;
int d = i/100;
int e = i%100;
int f = i+i;
```

2. Beschreiben Sie für beliebige positive Werte von i und j (beide Ganzzahldatentypen) das Ergebnis von

```
(i/j)*(j/i)
```

3. a) Wann ist der Datentyp eines binären Ausdrucks mit ganzzahligen Operanden derselbe wie der der Operanden?
   b) Kann die Addition des Wertes 0 irgendeinen Effekt auf einen ganzzahligen Wert haben?

4. Beschreiben Sie für einen ganzzahligen Operanden x das Ergebnis von

   a) `int a=x^x;`
   b) `int b=x&1;`

5. Welche Voraussetzungen müssen für das Bitmuster der *int*-Werte x und y gelten, damit die folgenden drei Ausdrücke gleich sind?

   ```
   x|y, x+y, x^y
   ```

6. Was wäre das Ergebnis der Bedingungen in den folgenden *if*-Anweisungen, wenn die Operanden nicht in den gemeinsamen Datentyp *int* konvertiert würden, sondern stattdessen als *unsigned char* verglichen würden?

   ```
   unsigned char c = 255, x = 17, y = 18;
   ```

   a) `if (c==-1) ...`
   b) `if (x-y > 0) ...`
   c) `if (x>y) ...`

3.3.5 Der Datentyp *bool*

3.3.6 Die *char*-Datentypen und der ASCII- und ANSI-Zeichensatz

Aufgaben 3.3.6

1. *c* soll eine Variable des Datentyps *char* sein. Weisen Sie einer booleschen Variablen

   a) *Grossbuchstabe* genau dann den Wert *true* zu, wenn *c* ein Großbuchstabe ist.
   b) *Buchstabe* genau dann den Wert *true* zu, wenn *c* ein Buchstabe ist.
   c) *alphanumerisch* genau dann den Wert *true* zu, wenn *c* ein Buchstabe oder eine Ziffer ist.

2. Schaltjahre

   Die Umlaufzeit der Erde um die Sonne bezeichnet man als ein Jahr. Ein Tag ist der Zeitraum, in dem sich die Erde einmal um ihre Achse dreht (Abstand zwischen zwei Mittagen).

   Misst man ein Jahr als den Zeitraum zwischen den sogenannten Frühlingspunkten, an denen ein Tag und eine Nacht genau gleich lang sind, war am 1.1.1900 ein Jahr 365 Tage, 5 Stunden, 48 Minuten und 46,0 Sekunden oder 365,24220 Tage lang. Dass ein Jahr in einem Jahrtausend 5,6 Sekunden kürzer wird, soll im Folgenden nicht berücksichtigt werden.

   Der von Julius Cäsar 46 v. Chr. festgelegte **Julianische Kalender** ging von durchschnittlich 365,25 Tagen pro Jahr aus. Dieser Fehler ist bis in das 16. Jahrhundert auf ca. 10 Tage angewachsen.

   Papst Gregor XIII. hat im Jahr 1582 den auch heute noch gültigen **Gregorianischen Kalender** eingeführt, nach dem ein Jahr durchschnittlich 365,2425 Tage lang ist. Dabei wurden die Nachkommastellen folgendermaßen durch Schaltjahre berücksichtigt: Jede durch 4 teilbare Jahreszahl ist ein Schaltjahr, außer den durch 100 teilbaren, wenn diese nicht durch 400 teilbar sind. Bei diesem Verfahren summiert sich der Fehler in 3300 Jahren auf einen Tag auf.

   Weisen Sie einer booleschen Variablen *Schaltjahr* den Wert eines booleschen Ausdrucks zu, so dass *Schaltjahr* den Wert *true* erhält, wenn die Variable *Jahr* (Datentyp *int*) nach dem Gregorianischen Kalender ein Schaltjahr ist, und andernfalls den Wert *false*.

3. Die Ganzzahlvariablen t1, m1 und j1 sowie t2, m2 und j2 sollen zwei Kalenderdaten bezeichnen (z.B. t1=17, m1=12, j1=2003). Eine boolesche Variable *vorher* soll den Wert *true* erhalten, wenn das Datum (t1, m1, j1) zeitlich vor dem Datum (t2, m2, j2) liegt, und andernfalls den Wert *false*.

4. Die logischen Operatoren sehen ähnlich aus wie die bitweisen und werden deshalb leicht verwechselt, so dass z.B. x&y anstelle von x&&y geschrieben wird. Wie unterscheiden sich die Werte dieser Ausdrücke für Operanden des Datentyps *bool* und *int*?

5. Bei einem Lösungsversuch habe ich einmal den folgenden Ausdruck gesehen (dabei war x eine *int*-variable):

   ```
   if (x==!0) ... // gemeint war if (x!=0) ...
   ```

   Wird dieser Ausdruck vom Compiler akzeptiert? Falls ja, was bedeutet er?

6. Ein Formular soll ein *MainMenu* mit verschiedenen Unterpunkten (*MenuItems*) und eine *GroupBox* mit einigen Komponenten enthalten. Durch das Anklicken eines Buttons sollen die Eigenschaften

   a) *Enabled* (für einen Unterpunkt des Menüs)
   b) *Visible* (der GroupBox)

   von *true* auf *false* und von *false* auf *true* umgeschaltet werden.

3.3.7 Der Datentyp *__int64*

3.3.8 C++0x-Erweiterungen für Ganzzahldatentypen Θ

3.4 Kontrollstrukturen und Funktionen

3.4.1 Die *if*- und die Verbundanweisung

Aufgaben 3.4.1

1. Welche der folgenden *if*-Anweisungen sind nach den Definitionen

```
int x, Punkte, i, j;
bool b;
```

   syntaktisch und inhaltlich korrekt?

   a) `if (x=17) Edit1->Text = "Volltreffer";`
   b) `if (i>=1 && i<=10) Edit1->Text = "Volltreffer";`
   c) `if b && (i=j*x) Edit1->Text = "Volltreffer";`
   d)
```
if (Punkte >= 0) Edit1->Text = "Extremes Pech";
else if (Punkte >= 20) Edit1->Text ="Ziemliches Pech";
else if (Punkte >= 40) Edit1->Text = "Ein wenig Glück gehabt";
```

2. In Abhängigkeit vom Wert einer Variablen *Bewegungsart* (Datentyp *char*, zulässige Werte '+' oder '–') soll der Wert der Variablen *Betrag* zur Variablen *Kontostand* addiert oder von dieser subtrahiert werden.

3. In Abhängigkeit vom Wert der beiden Variablen *Lagergruppe* und *Materialgruppe* (beide Datentyp *char*) sollen gemäß der folgenden Tabelle die Werte der Variablen *LA_Summe*, *LB_Summe* usw. um den Wert der Variablen *Summe* erhöht werden:

Lager-gruppe	Material-gruppe	Verarbeitung
'A'	'A'	*LA_Summe* und *MA_Summe* um *Summe* erhöhen
'A'	'B'	*LA_Summe* und *MB_Summe* um *Summe* erhöhen
'A'	'X'	*LA_Summe* und *MX_Summe* um *Summe* erhöhen
'B'	'B'	*LB_Summe* und *MB_Summe* um *Summe* erhöhen
'B'	'D'	*LB_Summe* und *MD_Summe* um *Summe* erhöhen

   Falls *Lagergruppe* den Wert 'A' hat, aber *Materialgruppe* nicht einen der Werte 'A', 'B' oder 'X', soll die Meldung

   „Unzulässige Materialgruppe in Lager A“

   ausgegeben werden; entsprechend für *Lagergruppe* = 'B'.

   Falls *Lagergruppe* weder den Wert 'A' noch den Wert 'B' hat, soll die Meldung

   „Unzulässige Lagergruppe“

erfolgen. In jedem dieser unzulässigen Fälle soll keine Summation durchgeführt werden.

4. **Datumsvergleich**

   Die Ganzzahlvariablen t1, m1 und j1 sowie t2, m2 und j2 sollen zwei Kalenderdaten bezeichnen (z.B. t1=17, m1=5, j1=2005). Eine boolesche Variable *vorher* soll den Wert *true* erhalten, wenn das Datum (t1, m1, j1) zeitlich vor dem Datum (t2, m2, j2) liegt, und andernfalls den Wert *false*.

   Diese Aufgabe wurde schon in Aufgabe 3.3.6, 3. allein mit booleschen Variablen behandelt. Sie kann aber auch mit *if*-Anweisungen bearbeitet werden, was oft als einfacher angesehen wird.

   Falls die Jahreszahlen in j1 und j2 verschieden sind, gibt der boolesche Ausdruck

```
j1 < j2
```

   an, ob das erste Datum zeitlich vor dem zweiten liegt:

```
if (j1 != j2) vorher = (j1 < j2)
```

   Wenn dagegen die Jahreszahlen gleich und die Monate verschieden sind, entscheiden die Monate über die zeitliche Anordnung der beiden Kalenderdaten. Sind sowohl die Jahre als auch die Monate gleich, entscheidet der Tag.

   Wie muss die Lösung geändert werden, wenn eine boolesche Variable *vorher_oder_gleich* genau dann den Wert *true* erhalten soll, wenn das erste Datum vor dem zweiten liegt oder gleich dem zweiten ist?

3.4.2 Wiederholungsanweisungen

3.4.3 Funktionen und der Datentyp *void*

3.4.4 Werte- und Referenzparameter

3.4.5 Die Verwendung von Bibliotheken und Namensbereichen

3.4.6 Zufallszahlen

Aufgaben 3.4.6

1. Schreiben Sie

   a) eine Funktion *Quersumme*, die für einen als Argument übergebenen Ganzzahlwert die **Quersumme** als Funktionswert zurückgibt (z.B. Quersumme(123)=6, Quersumme(100)=1)
   b) eine Funktion *zeigeQuersumme*, die für alle Werte, die zwischen zwei als Parametern übergebenen Grenzen liegen, die Quersumme in einem Memo ausgibt.

2. Die **Fibonacci-Zahlen** 0, 1, 1, 2, 3, 5, 8 usw. sind durch $f_0 = 0$, $f_1 = 1$, $f_i = f_{i-1} + f_{i-2}$ für i = 2, 3, 4, ... definiert, d.h. jede Zahl, außer den ersten beiden, ist die Summe der beiden vorangehenden.

   Diese Zahlenfolge geht auf eine Additionsübung des italienischen Mathematikers Fibonacci im Jahr 1202 zurück. Dabei bezeichnet f_n die Anzahl der Hasenpaare nach n Monaten, wenn jedes Hasenpaar jeden Monat ein neues Paar Junge bekommt, das nach einem Monat ebenfalls ein Paar Junge bekommt. In diesem einfachen Modell sind alle Hasen unsterblich.

   a) Schreiben Sie eine Funktion, die den n-ten Wert der Fibonacci-Folge berechnet. Dabei soll n als Parameter übergeben werden.
   b) Schreiben Sie eine Funktion, die die ersten n Werte der Fibonacci-Folge in einem Memo ausgibt. Rufen Sie diese Funktion mit n=50 auf.

3. Welche Werte werden durch die *for*-Schleife ausgegeben

```
int n=StrToInt(Edit1->Text);
for (unsigned int u=0; u<=n-1; u++)
  Memo1->Lines->Add(IntToStr(u));
```

a) mit n=0?
b) mit n=1?
c) Welche Ausgabe erhält man mit n=0, wenn man in der *for*-Schleife die Bedingung u<=n-1 durch u<n ersetzt?

4. Eine positive ganze Zahl n ist eine **Primzahl**, wenn sie genau zwei verschiedene Teiler hat (nämlich 1 und n) und keine weiteren. Beispielsweise sind 2, 3 und 5 Primzahlen, aber 1, 4 und 6 sind keine.

a) Schreiben Sie eine boolesche Funktion *istPrim*, die für einen als Parameter übergebenen Ganzzahlwert n den Wert *true* zurückgibt, wenn n eine Primzahl ist, und andernfalls den Wert *false*. Prüfen Sie dazu in einer Schleife alle möglichen Teiler von n.
b) Schreiben Sie eine Funktion *zeigePrimzahlen*, die für einen als Parameter übergebenen Wert n alle Primzahlen ausgibt, die <= n sind.
c) Nach einer Vermutung des Mathematiker **Goldbach** kann jede gerade ganze Zahl >2 als Summe von zwei Primzahlen dargestellt werden. Obwohl diese Vermutung schon vor über 200 Jahren aufgestellt und für zahlreiche Werte überprüft wurde, konnte sie bis jetzt noch nicht bewiesen werden. Schreiben Sie eine Funktion *zeigeGoldbachPaare*, die für einen als Parameter übergebenen Wert n alle Paare von Primzahlen mit der Summe n in einem Memo ausgibt. Die Anzahl dieser Paare ist z.B. 6 für n=100 und 28 für n=1000.

5. Drei Zahlen a, b und c, für die $a^2 + b^2 = c^2$ gilt (z.B. 3, 4 und 5), heißen **Pythagoräisches Zahlentripel**. Geben Sie alle solchen Zahlentripel für a und b <= n (ein Parameter) in einem Memo aus. Probieren Sie dazu für alle Kombinationen von a und b alle möglichen Werte von c aus, ob die Bedingung gilt. Für n=50 gibt es 26 solche Tripel.

6. Beim sogenannten **3n+1-Problem** wird ausgehend von einer positiven ganzen Zahl n nach dem folgenden Verfahren eine Zahlenfolge bestimmt:

Für n=1 wird das Verfahren abgebrochen.
Falls n ungerade ist, ersetzt man n durch (3n+1).
Falls n gerade ist, ersetzt man n durch n/2.

So erhält man z.B. für n=5 die Zahlenfolge 5, 16, 8, 4, 2, 1. Nach einer Vermutung der Mathematiker Ulam und Collatz konvergiert diese Folge immer. Obwohl diese Vermutung inzwischen für alle Zahlen $<7*10^{11}$ überprüft wurde, konnte sie bisher nicht allgemein bewiesen werden.

Schreiben Sie eine Funktion *f3nPlus1*, die für einen als Parameter übergebenen Wert n die Anzahl der Schritte als Funktionswert zurückgibt, bis das Verfahren abbricht. Zum Testen soll über einen zweiten Parameter gesteuert werden können, ob alle Werte dieser Zahlenfolge in einem Memo-Fenster ausgegeben werden oder nicht.

7. Schreiben Sie eine Funktion *zeigeZufallszahlen*, die n mit *rand* und n mit *random* erzeugte Zufallszahlen im Bereich 1 .. 49 in einem Memo ausgibt. Vergleichen Sie die Ergebnisse, wenn Sie das Programm mehrfach

a) mit demselben Startwert für die Zufallszahlen starten
b) mit verschiedenen Startwerten für die Zufallszahlen starten.

8. **Die Gauß'sche Osterformel**

Auf dem Konzil von Nicäa (325 n. Chr.) wurde festgelegt, dass der Ostersonntag der erste Sonntag nach dem ersten Vollmond im Frühling ist. Nach Knuth (1973, Bd. 1) war die Berechnung des Ostersonntags die einzige wichtige Anwendung der Arithmetik im Mittelalter.

Gauß hat die Arbeit der mit dieser Berechnung beschäftigten Mönche durch den folgenden Algorithmus rationalisiert. Die hier dargestellte Version gilt allerdings nur bis zum Jahr 2299. Ein allgemeineres Verfahren findet man bei Knuth.

M und N seien durch die folgende Tabelle gegeben:

Jahr	M	N
1583–1699	22	2
1700–1799	23	3
1800–1899	23	4
1900–2099	24	5
2100–2199	24	6
2200–2299	25	0

A, B, C seien die Reste der Divisionen der Jahreszahl durch 19, 4 bzw. 7, D der Rest der Division von (19A + M) durch 30 und E der Rest der Division von (2B + 4C + 6D + N) durch 7.

Dann ist der Ostersonntag gleich dem (22 + D + E)-ten März oder gleich dem (D + E – 9)-ten April, falls die folgenden Grenzfälle berücksichtigt werden:

1. Ergibt sich der 26. April, so setze man stets den 19. April.
2. Ergibt sich der 25. April und gilt D = 28, E = 6 und A > 10, so fällt der Ostersonntag auf den 18. April.

Der Pfingstsonntag ist dann der siebte Sonntag nach Ostern.

Schreiben Sie ein Programm, das den Oster- und Pfingstsonntag berechnet. Testen Sie das Programm mit den folgenden Werten:

```
1970 Ostern: 29. März   Pfingsten: 17. Mai
1971 Ostern: 11. April  Pfingsten: 30. Mai
1972 Ostern:  2. April  Pfingsten: 21. Mai
1973 Ostern: 22. April  Pfingsten: 10. Juni
1974 Ostern: 14. April  Pfingsten:  2. Juni
1975 Ostern: 30. März   Pfingsten: 18. Mai
1976 Ostern: 18. April  Pfingsten:  6. Juni
1977 Ostern: 10. April  Pfingsten: 29. Mai
1978 Ostern: 26. März   Pfingsten: 14. Mai
1979 Ostern: 15. April  Pfingsten:  3. Juni
1980 Ostern:  6. April  Pfingsten: 25. Mai
```

Eine schön ausgedruckte Liste mit den so berechneten Osterdaten ist auch gut als Ostergeschenk geeignet (jedes Jahr ein anderes Jahrhundert). Weniger guten Freunden kann man eine Liste mit dem Datum von Weihnachten schenken (auch jedes Jahr ein anderes Jahrhundert).

9. Haben Sie darauf geachtet, dass alle ihre Funktionen zur Lösung dieser Aufgaben für jedes Argument im Wertebereich der Funktionsparameter einen definierten Wert zurückgeben?

3.5 Tests und der integrierte Debugger

3.5.1 Systematisches Testen

Aufgaben 3.5.1

1. Entwerfen Sie für die Funktionen in a) bis f) systematische Tests. Geben Sie für jeden Testfall das erwartete Ergebnis an. Die Tests mit diesen Testdaten werden dann in Aufgabe 3.5.2 durchgeführt und sind hier nicht notwendig.

a) `int Quersumme(int n) //` (Aufgabe 3.4.6, 1.)

```
{
if (n<0) n=-n; // berücksichtige negative n
int s=0;
while (n>0)
  {
    s = s+n%10;
    n = n/10;
  }
return s;
}
```

b) `int Fibonacci(int n) //` (Aufgabe 3.4.6, 2.)

```
{
int f=0,x=0,y=1;
for (int i=0; i<n; i++)
  {
    x=y;
    y=f;
    f=x+y;
  }
return f;
}
```

c) `bool istprim(int n)//` (Aufgabe 3.4.6, 4.)

```
{
if (n<2) return false;
else if (n==2) return true;
else
  {
     for (int i=2; i*i<=n; i++)
       if (n%i==0) return false;
     return true;
  }
}
```

d) `int Goldbach(int n) //` (Aufgabe 3.4.6, 4.)

```
{
if ((n<4) || (n%2==1)) return -1;
else
  {
    int k=0;
    for (int i=2; i<=n/2; i++)
      if (prim(i) && prim(n-i)) k++;
    return k;
  }
}
```

e) `int zeigePythagTripel(int n) //` (Aufgabe 3.4.6, 5.)

```
{
int nt=0; // Anzahl der gefundenen Tripel
for (int a=1; a<n; a++)
  for (int b=a; b<n; b++)
     for (int c=b; c*c<=a*a+b*b; c++)
       if (a*a+b*b==c*c)
         {
         Form1->Memo1->Lines->Add("a="+IntToStr(a)+
            " b="+IntToStr(b)+" c= "+ IntToStr(c));
         nt++;
         }
return nt;
}
```

f) `int f3nplus1(int n) //` (Aufgabe 3.4.6, 6.)

```
{
if (n<=1) return 0;
int m=0;
while (n!=1)
  {
    m++;
    if (n%2==1) n=3*n+1;
    else n=n/2;
  }
return m;
}
```

g) Wie viele Tests sind für einen vollständigen Pfadtest bei den Funktionen in a) bis d) notwendig, wenn der Datentyp *int* Werte im Bereich -2,147,483,648 ... 2,147,483,647 annehmen kann?

3.5.2 Testprotokolle und Testfunktionen für automatisierte Tests

Aufgaben 3.5.2

1. Schreiben Sie Testfunktionen für die Testfälle aus Aufgabe 3.5.1,1. Eine weitere Testfunktion *test_3_4* soll genau dann den Wert *true* zurückgeben, wenn alle diese Tests erfolgreich waren. Nehmen Sie diese Funktionen in das Projekts mit Ihren Lösungen der Aufgabe 3.4.6 und rufen Sie *test_3_4* als Reaktion auf einen Buttonklick auf. Falls alle Tests erfolgreich waren, soll eine entsprechende Meldung ausgegeben werden.

   Sie können die TestUtils-Funktionen von der Buch-CD verwenden.

   Falls Sie diese Aufgaben im Rahmen einer Gruppe (z.B. in einer Vorlesung) bearbeiten, kann es zur Qualität Ihrer Tests beitragen, wenn Sie nicht Ihre eigenen Funktionen testen, sondern die eines anderen Teilnehmers. Tauschen Sie dazu Ihre Lösungen aus und testen Sie sich gegenseitig.

2. Geben Sie einige Beispiele für Funktionen an, die nicht oder zumindest nicht so einfach, wie in Aufgabe 1 automatisch getestet werden können.

3.5.3 Tests mit DUnit im C++Builder 2007

3.5.4 Der integrierte Debugger

Aufgabe 3.5.4

Setzen Sie in Ihrer Lösung der Funktion *ZeigeQuersumme* (Aufgabe 3.4.6, 1) einen Haltepunkt. Starten Sie Ihr Programm, so dass die Anweisung mit dem Haltepunkt ausgeführt wird.

a) Gehen Sie mit einer schrittweisen Programmausführung in die Funktion *Quersumme*. Zeigen Sie die Variablen i und s im Fenster *Überwachte Ausdrücke* bzw. *Lokale Variablen* an und führen Sie die Anweisungen schrittweise aus.
b) Schauen Sie in der Funktion *Quersumme* den Aufrufstack an.
c) Schreiben Sie eine Endlosschleife und unterbrechen Sie das Programm einmal mit *Start|Programm Pause* und einmal mit *Start|Programm zurücksetzen.*

3.6 Gleitkommadatentypen

3.6.1 Die interne Darstellung von Gleitkommawerten

3.6.2 Der Datentyp von Gleitkommaliteralen

3.6.3 Standardkonversionen

3.6.4 Mathematische Funktionen

3.6.5 Datentypen für exakte und kaufmännische Rechnungen

Aufgaben 3.6.5

1. Welche der folgenden Anweisungen werden vom Compiler ohne Fehler übersetzt? Welche Werte erhalten die Variablen d1, d2, i1, ..., i6?

```
int j=10,k=6;
double x=j,y=3.14,z=10;

double d1 = j/k;
double d2 = x/k;
int i1 = j/k;
int i2 = x/k;
int i3 = 3(j+k);
int i4 = j/k/k;
int i5 = j/(z/y);
int i6 = j/(y-j/k);
```

2. Mit dem folgenden Experiment kann man einen Näherungswert für die Kreiszahl π bestimmen: Man zeichnet bei Regen in ein Quadrat mit der Seitenlänge 2 einen Kreis mit dem Radius 1 und zählt die Regentropfen, die in das Quadrat bzw. in den Kreis fallen. Bezeichnet man diese Anzahlen als k und n, ist der Quotient k/n ein Näherungswert für den Anteil des Kreises an der Fläche des Quadrates, also π/4.

   Dieses Experiment kann man folgendermaßen simulieren: Mit der vordefinierten Funktion ***rand()*** erhält man eine Zufallszahl zwischen 0 und *RAND_MAX* (0x7FFFU). Transformiert man zwei aufeinander folgende Zufallswerte in den Bereich zwischen 0 und 1, kann man sie als Koordinaten eines Punktes (x,y) im ersten Quadranten betrachten. Dieser Punkt liegt im Einheitskreis, wenn $x^2+y^2<1$ gilt.

   Schreiben Sie eine Funktion *RegentropfenPi*, die einen nach dieser „**Regentropfenmethode**“ bestimmten Näherungswert für **π** als Funktionswert zurückgibt. Die Anzahl der Versuche soll als Parameter übergeben werden. Zeigen Sie die Näherungswerte bei 1000 (10000) Wiederholungen in einem Memo-Fenster an.

3. Das Produkt der ersten n (n>0) Zahlen

   f = 1*2*3* ... *n

   wird auch als **n!** (n Fakultät) bezeichnet. Schreiben Sie eine Funktion ***Fakultaet***, die die Fakultät für einen als Argument übergebenen Wert berechnet. Zeigen Sie die Werte für n=1 bis n=30 in einem Memo-Fenster an.

   Die Fakultät tritt z.B. beim sogenannten **Problem des Handlungsreisenden** auf: Wenn ein Handlungsreisender n Städte besuchen soll, hat er für die erste Stadt n Möglichkeiten, für die zweite n–1, für die dritte n–2 usw. Um jetzt die kürzeste Route durch diese n Städte zu finden, müssen (zumindest im Prinzip) alle n! Routen miteinander verglichen werden.

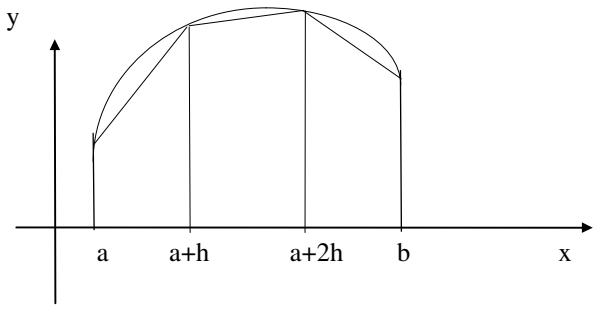
Angenommen, Sie hätten ein Computerprogramm zur Verfügung, das in einer Sekunde 1000000 Routen verglichen kann. Zeigen Sie die Rechenzeit zur Bestimmung der kürzesten Route für n=15 bis n=30 in einem Memo-Fenster an.

4. Ein **Hypothekendarlehen** über einen Betrag von k Euro mit einem Zinssatz von p% und einer Tilgung von t% sei als Annuitätendarlehen vereinbart. Dabei hat der Schuldner jedes Jahr am Jahresende eine gleich bleibende Rate von (p+t)% zu leisten.

   Von dieser konstanten Rate entfallen p% der Restschuld auf die Zinsen. Der Rest ist dann die Tilgung, die wegen der im Laufe der Jahre abnehmenden Restschuld jährlich größer wird.

   Schreiben Sie eine Funktion, die die Restschuld, Zinsen und Tilgungsraten für jedes Jahr der Laufzeit des Hypothekendarlehens in einem Memo ausgibt. Die Zinsen und die Tilgungsraten sollen jeweils am Jahresende fällig werden.

5. Die Fläche zwischen der Funktion y=f(x) und der x-Achse im Bereich von a bis b kann näherungsweise durch die Summe der Trapezflächen berechnet werden (**Numerische Integration** mit der Trapezregel):



   Die Fläche des Trapezes

   von a bis a+h ist dabei durch h*(f(a) + f(a+h))/2 gegeben,
   die von a+h bis a+2h durch h*(f(a+h) + f(a+2*h))/2 usw.

   Unterteilt man das Intervall von a bis b in n Teile, ist h = (b–a)/n.

   Schreiben Sie eine Funktion *Trapezsumme*, die einen Näherungswert für π als Fläche des Einheitskreises berechnet, indem sie die Trapezflächen unter der Funktion sqrt(1–x*x) von 0 bis 1 aufsummiert. Wählen Sie für n verschiedene Werte, z.B. 100, 1000 und 10000.

6. Das **Geburtstagsproblem von Mises**

   Die Wahrscheinlichkeit q, dass zwei zufällig ausgewählte Personen an verschiedenen Tagen Geburtstag haben, ist

   q = 364/365

   Bei drei Personen ist die Wahrscheinlichkeit, dass alle drei an verschiedenen Tagen Geburtstag haben

   q = (364*363)/(365*365)

   Bei n Personen ist diese Wahrscheinlichkeit

   $q = 364*363*...*(364–n+2)/365^{n–1}$ (n-1 Faktoren im Zähler und Nenner)

   Die Wahrscheinlichkeit, dass von n Personen mindestens zwei am selben Tag Geburtstag haben, ist dann gegeben durch

   p = 1–q

   a) Schätzen Sie zuerst, ab wie vielen Personen diese Wahrscheinlichkeit > 50% ist.
   b) Schreiben Sie eine Funktion *Mises*, deren Funktionswert diese Wahrscheinlichkeit für n Personen ist. Bestimmen Sie dann den kleinsten Wert von n, für den ihr Funktionswert größer als 0.5 ist. Am einfachsten geben Sie dazu den Funktionswert für verschiedene Werte von n aus.
   c) Bei der sogenannten gestreuten Speicherung (Hash-Tabelle) legt man einen Datensatz in einer Tabelle ab, wobei seine Position in der Tabelle aus den Daten berechnet wird. Falls sich dann für zwei verschiedene Datensätze

dieselbe Position ergibt, erhält man eine Kollision. Wie viele Tabellenplätze sind notwendig, damit für 23 Datensätze mit zufälligen Positionen die Wahrscheinlichkeit für eine Kollision <50% ist?

7. Schreiben Sie eine Funktion ***RoundToInt***, die ein Gleitkomma-Argument auf den nächst höheren Ganzzahlwert aufrundet, falls seine Nachkommastellen >=0.5 sind, und andernfalls abrundet. Diese Funktion soll auch negative Werte richtig runden (z.B. *RoundToInt(3.64)=4*, *RoundToInt(3.14)=3*, *RoundToInt(–3.14)=-3*, *RoundToInt(-3.64)=–4*)

8. **Steuerformel**

   In § 32 des Einkommensteuergesetzes (EStG 2005) ist festgelegt, wie sich die Einkommensteuer aus dem zu versteuernden Einkommen berechnet:

   §32 a Einkommensteuertarif

   (1) Die tarifliche Einkommensteuer bemisst sich nach dem zu versteuernden Einkommen. Sie beträgt vorbehaltlich der §§ 32b, 34, 34b und 34c jeweils in Euro für zu versteuernde Einkommen

   1. *bis 7664 Euro (Grundfreibetrag): 0;*

   2. *von 7665 Euro bis 12739 Euro:*
      *(883,74*y + 1500)*y*

   3. *von 12740 Euro bis 52151 Euro:*
      *(228,74*z + 2397)*z + 989*

   4. *von 52152 Euro an: 0,42*x – 7914;*

   „y“ ist ein Zehntausendstel des 7664 Euro übersteigenden Teils des auf einen vollen Euro-Betrag abgerundeten zu versteuernden Einkommens. „z“ ist ein Zehntausendstel des 12739 Euro übersteigenden Teils des auf einen vollen Euro-Betrag abgerundeten zu versteuernden Einkommens. „x“ ist das auf einen vollen Euro-Betrag abgerundete zu versteuernde Einkommen. Der sich ergebende Steuerbetrag ist auf den nächsten vollen Euro-Betrag abzurunden.

a) Schreiben Sie eine Funktion *EStGrundtarif2005*, die zu dem als Parameter übergebenen zu versteuernden Einkommen die Einkommensteuer als Funktionswert zurückgibt. Welche Datentypen sind hier angemessen?

   Zum Testen können Sie den folgenden Auszug aus der Steuertabelle verwenden:

x	Est	x	Est	x	Est
7 664	0	20 000	2 850	100 000	34 086
7 704	6	40 000	9 223	120 000	42 486
7 740	11	52 092	13 964	140 000	50 886
12 708	981	60 000	17 286	160 000	59 286
12 744	990	80 000	25 686	180 000	67 686

b) In § 32, Absatz 5 des EStG ist festgelegt, wie die Einkommensteuer nach dem Splitting-Verfahren berechnet wird:

   (5) Für Ehegatten, die nach den §§ 26, 26b zusammen zur Einkommensteuer veranlagt werden, beträgt die tarifliche Einkommensteuer vorbehaltlich der §§ 32b, 34 und 34b das Zweifache des Steuerbetrags, der sich für die Hälfte Ihres gemeinsam zu versteuernden Einkommens nach den Absätzen (1) bis (3) ergibt (Splitting-Verfahren).

   Die Funktion *EStSplittingtarif2005* soll als Funktionswert die nach dem Splitting-Verfahren berechnete Einkommensteuer zurückgeben. Verwenden Sie zur Lösung dieser Aufgabe die Lösung von a).

   Zum Testen können Sie den folgenden Auszug aus der Steuertabelle verwenden:

x	Est	x	Est	x	Est
20 000	796	60 000	11 614	100 000	26 192
40 000	5 700	80 000	18 446	120 000	42 972

9. **Reihenfolge der Summation** bei Gleitkommadatentypen

a) Berechnen Sie die Summe der Zahlen $1/i^2$ von i=1 bis n (n=1000000) abwechselnd von unten (for (i=1; i<=n; i++)...) und von oben (for (i=n; i>=1; i--)...). Dabei sollen alle Variablen für Summen und eventuelle Zwischenergebnisse den Datentyp *float* haben. Vergleichen Sie die beiden Summen. Welche ist genauer?
b) Berechnen Sie die Summe aus a) auch für die anderen Gleitkommaformate *double* und *long double*. Dabei sollen alle Summen und eventuellen Zwischenergebnisse den jeweiligen Gleitkommadatentyp haben.

10. Angenommen, die Ausführung der nächsten Schleife dauert eine Sekunde. Schätzen Sie ihre Laufzeit, wenn der Datentyp *double* durch *float* ersetzt wird.

```
double x = 1e8;
while(x > 0)
  --x;
```

3.6.6 Ein Kriterium für annähernd gleiche Gleitkommazahlen

Aufgaben 3.6.6

1. Entwerfen Sie für die Funktionen von Aufgabe 3.6.5 systematische Testdaten. Falls das bei einzelnen Funktionen schwierig oder nicht möglich ist, begründen Sie das.

2. Schreiben Sie Testfunktionen für die Testfälle aus Aufgabe 1. Eine weitere Testfunktion *Test_3_6* soll genau dann den Wert *true* zurückgeben, wenn alle diese Tests erfolgreich waren. Nehmen Sie diese Funktionen in die Unit des Projekts mit Ihren Lösungen der Aufgabe 3.6.5 und rufen Sie *Test_3_6* als Reaktion auf einen Buttonklick auf. Geben Sie eine Meldung aus, ob alle Tests erfolgreich waren.

   Sie können die Funktionen

```
bool assertEqual_double(double f, double s, AnsiString
                                  msg, TCustomMemo* Memo=Form1->Memo1, int digits=10)

bool assertEqual(double f, double s, AnsiString msg,
                                       TCustomMemo* Memo=Form1->Memo1, int digits=10)
```

   von der Buch-CD mit der folgenden *#include*-Anweisung übernehmen:

```
#include "\Loesungen_CB2006\CppUtils\TestUtils.h"
```

3. Auch Funktionen wie *NearlyEqual* und *DefinitelyLess* sollten getestet werden. Entwerfen Sie für diese Funktionen systematische Testdaten und Testfunktionen.

3.7 Ablaufprotokolle und Programmierlogik

3.7.1 Ablaufprotokolle

3.7.2 Schleifeninvarianten mit Ablaufprotokollen erkennen

Aufgaben 3.7.2

1. Erstellen Sie für die Anweisungen der folgenden Funktionen Ablaufprotokolle mit den jeweils angegebenen Argumenten:

   a) *Fakultaet(4)* (Aufgabe 3.6.5, 3.)

```
int Fakultaet(int n)
{
int f=1;
for (int i=2; i<=n; i++)
  f = f*i;
return f;
}
```

b) *Mises(3)* (Aufgabe 3.6.5, 6.)

```
double Mises(int n)
{
double q=1;
for (int i=0; i<n-1; i++)
  q = q*(364-i)/365;
return 1-q; // 0 für n<=1
}
```

c) *Fibonacci(4)* (Aufgabe 3.4.5, 2.)

```
int Fibonacci(int n)
{
int f=0,x=0,y=1;
for (int i=0; i<n; i++)
  {
    x=y;
    y=f;
    f=x+y;
  }
return f;
}
```

d) *Quersumme(289)* (Aufgabe 3.4.6, 1.)

```
int Quersumme(int n)
{
if (n<0) n=-n; // falls n < 0
int s=0;
while (n>0)
  {
    s = s+n%10;
    n = n/10;
  }
return s;
}
```

e) *prim(17)* und *prim(18)* (Aufgabe 3.4.6, 4.)

```
bool prim(int n)
{
if (n<2) return false;
else if (n==2) return true;
else
  {
     for (int i=2; i*i<=n; i++)
       if (n%i==0) return false;
     return true;
  }
}
```

2. Formulieren Sie Beziehungen zwischen den beteiligten Variablen, die nach jeder Ausführung des Schleifenkörpers gelten. Setzen Sie in jede dieser Beziehungen den letzten Wert ein, mit dem die Schleife wiederholt wurde

   a) in Aufgabe 1.a).
   b) in Aufgabe 1.b).
   c) ab der zweiten Ausführung des Schleifenkörpers in Aufgabe 1.c).
   d) in Aufgabe 1.d). Stellen Sie dazu den Parameter durch die Folge seiner Ziffern $z_n z_{n-1} \ldots z_2 z_1 z_0$ dar.
   e) in Aufgabe 1.e). Sie müssen diese Beziehung nicht durch einen formalen Ausdruck beschreiben, sondern können sie auch umgangssprachlich formulieren.

3. Geben Sie jeweils ein Beispiel für eine Funktion an, für die man (zumindest im Prinzip)

   a) kein Ablaufprotokoll erstellen kann.
   b) keine Beziehung zwischen ihren Variablen angeben kann.

4. Das Ablaufprotokoll für die Anweisungen

```
int x=1,y=2;
x=x+y;
y=x-y;
x=x-y;
```

hat gezeigt, dass sie die Werte der Variablen x und y vertauschen. Gilt das nur für die Werte x=1 und y=2, oder auch für beliebige Werte von x und y? Falls Sie sich nicht sicher sind, brechen Sie diese Aufgabe ab. Wir kommen später darauf zurück.

3.7.3 Symbolische Ablaufprotokolle

Aufgaben 3.7.3

1. Versuchen Sie, aus den Vorbedingungen in a) bis e) die als Nachbedingung angegebenen Beziehungen herzuleiten.

   a) n ganzzahlig und ungerade, p, x, u und v Ganzzahl- oder Gleitkommadatentypen

   ```
   // p*x^n = u^v
   p = p*x;
   n = n/2;
   x = x*x;
   // p*x^n = u^v
   ```

   b) n ganzzahlig und gerade, p, x, u und v Ganzzahl- oder Gleitkommadatentypen

   ```
   // p*x^n = u^v
   n = n/2;
   x = x*x;
   // p*x^n = u^v
   ```

   c) i ganzzahlig, s Gleitkomma- oder Ganzzahldatentyp

   ```
   // s = 1 + 2 + ... + i, d.h. s ist die Summe der ersten i Zahlen
   i = i + 1;
   s = s + i;
   // s = 1 + 2 + ... + i
   ```

   d) i ganzzahlig, s Gleitkomma- oder Ganzzahldatentyp

   ```
   // s = 1*1 + 2*2 + ... + i*i, d.h. s ist die Summe der ersten i Quadratzahlen
   i = i + 1;
   s = s + i*i;
   // s = 1*1 + 2*2 + ... + i*i
   ```

   e) i ganzzahlig, s Gleitkomma- oder Ganzzahldatentyp

   ```
   // s = 1*1 + 2*2 + ... + i*i
   s = s + i*i;
   i = i + 1;
   // s = 1*1 + 2*2 + ... + i*i
   ```

2. Wie immer, wenn man mehrere Verfahren zur Auswahl hat, stellt sich die Frage: „Welches ist besser?“. Vergleichen Sie die beiden Funktionen *vertausche* und *vertausche1*.

3.7.4 Schleifeninvarianten, Ablaufprotokolle, vollständige Induktion Θ

Aufgaben 3.7.4

1. Beweisen Sie mit vollständiger Induktion, dass die folgenden Funktionen den jeweils angegebenen Funktionswert haben.

   Formulieren Sie dazu jeweils eine Induktionsbehauptung und nehmen Sie in den Quelltext der Lösungen symbolische Ablaufprotokolle auf, die den Induktionsschritt nachweisen. Geben Sie die Induktionsbehauptung für den letzten Wert der Laufvariablen an, mit dem der Schleifenkörper ausgeführt wurde.

   Falls Sie die Induktionsbehauptung nicht unmittelbar aus der Aufgabenstellung erkennen, können Sie sie aus den Lösungen der Aufgabe 3.7.1, 1. ableiten.

a) Der Funktionswert der Funktion *Fakultaet(n)* (Aufgabe 3.6.5, 3.) ist das Produkt der ersten n Zahlen 1*2* ... *(n-1)*n.

```
int Fakultaet(int n)
{
int f=1;
for (int i=2; i<=n; i++)
  f = f*i;
return f;
}
```

b) Der Funktionswert der Funktion *Mises(n)* (Aufgabe 3.6.5, 6.) ist
$1 - 364*363*...*(365-n+1)/365^{n-1}$.

```
double Mises(int n)
{
double q=1;
for (int i=0; i<n-1; i++)
  q = q*(364-i)/365;
return 1-q; // 0 für n<=1
}
```

c) Der Funktionswert der Funktion *Quersumme(n)* (Aufgabe 3.4.6, 1.) ist die Quersumme der als Argument übergebenen Zahl. Sie können dazu für den Parameter die Darstellung $n=z_k z_{k-1}...z_2 z_1 z_0$ verwenden, wobei z_i die i-te Ziffer des Parameters ist.

```
int Quersumme(int n)
{// n=zk zk-1...z2 z1 z0
if (n<0) n=-n; // falls n < 0
int s=0;
while (n>0)
  {
    s = s+n%10;
    n = n/10;
  }
return s;
}
```

d) Der Funktionswert der Funktion *Fibonacci(n)* (Aufgabe 3.4.6, 2.) ist die n-te Fibonacci-Zahl f_n ($f_0=0$, $f_1=1$, $f_2=1$ usw., $f_n = f_{n-1} + f_{n-2}$ für n = 2, 3, 4, ...).

```
int Fibonacci(int n)
{
int f=0,x=0,y=1;
for (int i=0; i<n; i++)
  {
    x=y;
    y=f;
    f=x+y;
  }
return f;
}
```

e) Versuchen Sie, den Induktionsbeweis in d) führen, wenn die Anweisungen im Schleifenkörper ersetzt werden durch

```
    y=f;
    x=y;
    f=x+y;
```

f) Versuchen Sie, den Induktionsbeweis in d) führen, wenn die Anweisungen in der Funktion ersetzt werden durch

```
int x=0,f=1;
for (int i=0; i<n; i++)
  {
    int y=f;
    f=x+f;
    x=y;
  }
return f;
```

2. Unter welchen Bedingungen werden die folgenden Schleifen nicht zu Endlosschleifen (Datentyp von i und n: *int*, d: *double*):

a)
```
while (n != 0)
  {
    .
    n = n+2;
    .
  }
```

b)
```
while (i > 0)
  {
    .
    i = i/2;
    .
  }
```

c)
```
while (i < n)
  {
    .
    i++;
    .
  }
```

d)
```
double d=0;
while (d != 10)
  d = d + 0.1;
```

e)
```
while (i > 0) n = 2*n;//keine weiteren Zuweisungen an i
```

f) Die Schleife von b), wenn man i > 0 durch i >= 0 ersetzt?

g)
```
while (n!=1)  // siehe Aufgabe 3.4.6, 6.
  if (n%2==1) n=3*n+1;
  else n=n/2;
```

3. Für ganze Zahlen a und b ist a ein Teiler von b, falls es eine Zahl k gibt, so dass a*k = b ist. Der **größte gemeinsame Teiler ggT** von zwei Zahlen n, m ist der größte Teiler, der sowohl n als auch m teilt. Beispiele:

   ggT(4,30) = 2, ggT(2,7) = 1, ggT(0,5) = 5

   Aus dieser Definition der Teilbarkeit ergibt sich unmittelbar:

   1. ggT(n,m) = ggT(m,n)
   2. ggT(n,–m) = ggT(n,m)
   3. ggT(n,0) = n

   Wegen 2. kann man sich auf n > 0 und m >= 0 beschränken. Aus

   n = (n/m) * m + n%m

   folgt

   n – (n/m) * m = n%m

   Deshalb ist jeder gemeinsame Teiler t von n und m auch ein Teiler von m und n%m und umgekehrt, da

   $t*k_n=n$ und $t*k_m=m$ und $t*k_n - (n/m)*t*k_n = n\ \%m$

   d.h.

   ggT(n,m) = ggT(m,n % m)

   Dabei ist n%m kleiner als m.

   Beweisen Sie mit einer geeigneten Invariante, dass die Funktion *ggT* den größten gemeinsamen Teiler der Argumente als Funktionswert hat.

```
int ggt(int x, int y)
{ // x==x0,y==y0
while (y != 0)
  {
    int r = x%y;
    x = y;
    y = r;
  }
return x; // ggT(x0,y0)==x;
}
```

3.7.5 Verifikationen, Tests und Bedingungen zur Laufzeit prüfen

Aufgaben 3.7.5

1. Überprüfen Sie mit Zustandsvariablen (wie z.B. von *\Loesungen_CB2006\CppUtils\State.h* auf der Buch-CD) die unter a) und b) angegebenen Bedingungen zur Laufzeit, indem Sie diese Funktionen mit vielen zufälligen Werten aufrufen.

   a) Die als Kommentar angegebene Invariante der Funktion *mul*.

```
int mul(int x, int y)
{ // x==x0, y==y0, x0>=0
int z=0 ;
while (x !=0)
  { // z + x*y == x0*y0
    if (x&1) z = z + y;
    y = y*2;
    x = x/2;
    // z + x*y == x0*y0
  } //x==0 ==> z==x0*y0
return z;
}
```

   b) Der Rückgabewert ist das Minimum der beiden Argumente a und b.

```
int min2a(int a, int b)
{ // Keine weiteren Vorbedingungen, d.h. true
int m;
if (a<b) m=a;
else m=b;
return m;
} // min2a(a,b)==Minimum(a,b)
```

2. Bei welchen Funktionen von Aufgabe 3.7.4, 1. a) bis d) kann die Schleifeninvariante während der Laufzeit des Programms überprüft werden?

3.7.6 Funktionsaufrufe und Programmierstil für Funktionen

Aufgaben 3.7.6

1. Bearbeiten Sie die Teilaufgaben a) bis g) für jede der Aufgaben:

   3.4.6, 1. *Quersumme* 3.4.6, 2. *Fibonacci*
   3.4.6, 4. *prim*

   3.6.5, 2. *RegentropfenPi* 3.6.5, 5. *TrapezSumme*
   3.6.5, 4. *HypothekenRestschuld* 3.6.5, 7. *RoundToInt*

   a) Prüfen Sie, ob die Aufgabenstellung so präzise ist, dass man für jedes in der Aufgabenstellung geforderte Argument entscheiden kann, ob eine Funktion die Aufgabenstellung erfüllt.

   Prüfen Sie für jede Ihrer Funktionen zur Lösung dieser Aufgaben

   b) ob ihr Name ihr Ergebnis einigermaßen klar beschreibt.
   c) ob sie für jedes gemäß der Aufgabenstellung zulässige Argument den geforderten Wert zurückgibt.
   d) ob sie mit Argumenten aufgerufen werden kann, für die ein unbestimmter Wert zurückgegeben wird.
   e) ob sie mit Argumenten aufgerufen werden kann, die zu einem Programmfehler wie z.B. einer Endlosschleife, einer unzulässigen Operation, einem Überlauf usw. führen können. Falls die genaue Bestimmung der Grenzen zu aufwendig ist, reicht ein Hinweis, der das Problem charakterisiert.
   f) ob sie nach den oben angeführten Kriterien **übersichtlich**, **verständlich** und leicht **testbar** ist.

   g) Geben Sie für jede Ihrer Funktionen eine Vorbedingung an, die beim Aufruf erfüllt sein muss, damit sie die in der Aufgabenstellung geforderte Aufgabe erfüllen kann.
   h) Versuchen Sie, die Aufgabenstellung so zu erweitern, dass die Funktion mit möglichst schwachen Vorbedingungen aufgerufen werden kann. Die ursprüngliche Aufgabenstellung soll weiterhin erfüllt werden.

2. Versuchen Sie, einen passenden Namen für die folgende Funktion zu finden:

```
double doit_babe(double i, int *p1, int p2, float p3)
{
int y;
if (p2==1)
  for (int j=0;j<=i;j++) y=y+i;
else if (i==2) y=Mehrwertsteuer(p2,5);
else if (i==3) y=druckeAdressaufkleber(p2);
return y;
};
```

3. Beschreiben Sie für die Funktionen

```
int Plus(int a, int b)    {return a-b; }
int Minus(int a, int b)   {return a*b; }
int Product(int a, int b) {return a+b; }
```

das Ergebnis des Ausdrucks

```
int x=Plus(1,Product(Minus(3,4),Plus(5,6)));
```

3.7.7 Einfache logische Regeln und Wahrheitstabellen Θ

Aufgaben 3.7.7

1. Überprüfen Sie mit Wahrheitstafeln, ob die folgenden Formeln für alle möglichen Werte von p, q und r richtig sind:

   a) p and (q or r) == (p and q) or (p and r) // p&&(q || r)==(p && q)||(p && r)
   b) p or (q and r) == (p or q) and (p or r) // p || (q && r) == (p || q) && (p || r)
   c) (p and q) or r == p and (q or r) // (p && q) || r == p && (q || r)

2. Schreiben Sie für jede der drei Formeln aus Aufgabe 1 eine Funktion, die genau dann *true* zurückgibt, wenn die entsprechende Formel richtig ist.

3.7.8 Bedingungen in und nach *if*-Anweisungen und Schleifen Θ

Aufgaben 3.7.8

1. Geben Sie die Vorbedingungen bei der Ausführung von S2 in Form von verknüpften einfachen Bedingungen an:

   a)
   ```
   if ((x<0) or (x>100)) S1;
   else S2;
   ```
   b)
   ```
   if ((x!=1) and (c=='j')) S1;
   else S2;
   ```
   c)
   ```
   if(!((i<1) and (i>10))) S1;
   else S2;
   ```

   Vereinfachen Sie die Anweisung in d):

   d)
   ```
   if ((x>0) and (x<10)); // nichts machen
   else S; // S soll ausgeführt werden, wenn
           // (x>0) and (x<10) nicht gilt
   ```

2. Unter welchen Bedingungen werden S1, S2, S3 und S4 ausgeführt:

   ```
   if ((1<=x) and (x<=10)) S1;
   else if ((5<=x) and (x<=15)) S2;
   else if (x>12) S3;
   else S4;
   ```

3. Begründen oder widerlegen Sie, dass *max3* und *min3* nach der Ausführung eines jeden Zweigs der *if*-Anweisung den maximalen bzw. minimalen Wert der Argumente zurückgeben:

```
int max3(int x, int y, int z)
{
int m;
if ((x>=y) and (x>=z)) m=x;
else if ((y>=x) and (y>=z)) m=y;
else m=z;
return m;
}

int min3(int x, int y, int z)
{
int m;
if (x<y)
  {
    if (x<z) m=x;
    else m=z;
  }
else
  {
    if (y<z) m=y;
    else m=z;
  }
return m;
}
```

4. Begründen oder widerlegen Sie, dass die Funktion *Median* immer den mittleren der drei als Parameter übergebenen Werte liefert:

```
int Median(int a, int b, int c)
{
if (a < b)
  if (b < c)
    return b;
  else if (a < c)
    return c;
  else return a;
else if (a < c)
  return a;
else if (b < c)
  return c;
else return b;
}
```

5. Überprüfen Sie, ob die Funktion *pwr* unter der Vorbedingung n>=0 die als Kommentar angegebene Invariante hat. Dabei steht x^n für x^n. Sie können dazu die Ergebnisse der Aufgabe 3.7.3, 1. a) und b) verwenden.

```
double pwr(double x, int n)
{ // n>= 0, x==x0
double p=1;
while (n>0)
  { // p*x^n = x0^n0, ^: Potenzierung
    if (n&1) p = p*x;
    n = n/2;
    x = x*x;
    // p*x^n = x0^n0
  }
return p;
}
```

6. Geben Sie die Nachbedingung der nächsten Anweisung an:

```
if (n%2==1) n=3*n+1;
else n=n/2;
```

3.8 Konstanten

3.9 Syntaxregeln für Deklarationen und Initialisierungen Θ

3.10 Arrays und Container

3.10.1 Einfache *typedef*-Deklarationen

3.10.2 Eindimensionale Arrays

Aufgaben 3.10.2

1. Geben Sie an, welche der mit a) bis g) bezeichneten Anweisungen syntaktisch korrekt sind. Falls ja, beschreiben Sie das Ergebnis dieser Anweisungen.

```
void ArrayTest1()
{
int a[10];
for (int i=1; i<=10;i++) a[i] = 0;   // a)

int b[2], c[2], d[3];
b[0]=0; b[1]=1;
c=b;                                 // b)
int x=b[b[b[0]]];                    // c)
c[0]=0; c[1]=1;
d[0]=0; d[1]=1; d[2]=2;
if (b==c) x++;                       // d)
if (c==d) x++;                       // e)

int s1=sizeof(a);                    // f)
int s2=sizeof(a)/sizeof(a[0]);       // g)
}
```

2. Wenn man die Primzahlen unter den ersten 100 Zahlen bestimmen will, kann man ein Verfahren verwenden, das nach dem griechischen Mathematiker Eratosthenes als **Sieb des Eratosthenes** benannt ist: In einer Liste der Zahlen 1 bis 100 streicht man nacheinander zuerst alle Vielfachen von 2 ab 2, dann alle Vielfachen von 3 ab 3, 5 usw. Die Zahlen, die dabei übrig bleiben, sind dann die Primzahlen.

   Realisieren Sie dieses Verfahren mit einem booleschen Array, dessen Werte zunächst alle auf *true* gesetzt werden sollen. Die Anzahl der Arrayelemente soll mit möglichst geringem Aufwand auf einen anderen Wert als 100 gesetzt werden können.

3. Bei manchen Problemen findet man eine explizite Lösung nur schwierig oder überhaupt nicht. Dann können Simulationen hilfreich sein. Sind diese mit Zufallszahlen verbunden, bezeichnet man sie auch als **Monte-Carlo-Simulationen** (nach dem Spielerparadies).

   Schreiben Sie ein Programm zur Simulation des **Geburtstagsproblems von Mises** (siehe Aufgabe 3.6.6). Sie können dazu folgendermaßen vorgehen:

   Ein Array mit 365 Elementen soll die Tage eines Jahres darstellen. Mit einem Zufallszahlengenerator (z.B. *rand()*) wird dann für jede der n Personen ein Geburtstagsdatum ermittelt und im Array für die Tage ein Zähler hochgesetzt. Falls nach n Wiederholungen mindestens ein Zähler den Wert 2 oder mehr hat, entspricht das zwei oder mehr Personen, die an einem Tag Geburtstag haben. Diese Vorgehensweise wiederholt man dann mehrfach (z.B. 1000-mal).

4. Beim **Hornerschema** wird der Funktionswert eines Polynoms

$$h = p[n]*x^n + p[n-1]*x^{n-1} + \ ... + \ p[1]*x + p[0]$$

dadurch berechnet, dass die Klammern in

$$h = (...((p[n]*x + p[n-1])*x + p[n-2]) ... + p[1])*x + p[0]$$

von innen nach außen ausmultipliziert werden. Dabei werden nur n+1 Multiplikationen benötigt. Würde man alle Potenzen einzeln ausmultiplizieren, wären es n*(n–1)/2.

```
double Horner(double x)
{ // p: Array mit den Koeffizienten des Polynoms
const int n=2; // Der Grad des Polynoms
double p[n+1]={17,0,3}; // Koeffizienten von 17+3*x^2
double s = 0;
for (int i=0; i<n+1; i++)
  s = s*x + p[n-i];
return s; // s==p[n]*x^n + p[n-1]*x^n-1 +...+ p[1]*x+p[0]
}
```

a) Beweisen Sie mit vollständiger Induktion, dass dieser Algorithmus für ein Polynom mit den Koeffizienten p[n], p[n–1], ..., p[0] den Funktionswert berechnet:

$$s = p[n]*x^n + p[n-1]*x^{n-1} + \ ... + \ p[1]*x + p[0]$$

b) Die Sinus-Funktion wird durch die Taylor-Reihenentwicklung

$$\sin(x)=x/1 - x^3/3*2*1 + x^5/5*4*3*2*1 - x^7/7*6*5*4*3*2*1 \ \ldots$$

angenähert. Berechnen Sie den Wert dieses Polynoms mit einer überarbeiteten Version der Funktion *Horner* und vergleichen Sie die Ergebnisse für 0.1, 0.2 usw. bis 3.1 mit der Funktion *sin* aus <cmath>.

3.10.3 Die Initialisierung von Arrays bei ihrer Definition

Aufgaben 3.10.3

1. Welche Werte haben die Elemente dieser Arrays nach ihrer Definition?

```
int ai0[3];

void InitArrays()
{
int ai1[3];
int ai2[3]={1,2,3};
int ai3[3]={1,2};
}
```

2. Schreiben Sie eine Funktion *Fibonacci1*, die für Argumente i im Bereich 0..9 die i-te Fibonacci-Zahl (0,1,1,2,3,5,8,13,21,34) aus einem Array zurückgibt. Für weitere Argumente (bis 46) soll der jeweils nächste Wert aus den beiden Werten davor berechnet ($f_i = f_{i-1} + f_{i-2}$) und in das Array geschrieben werden. Der zum Argument gehörende Wert soll dann als Funktionswert zurückgegeben werden.

3.10.4 Arrays als Container

Aufgaben 3.10.4

1. Legen Sie ein neues Projekt an, dessen Formular ein Edit-Fenster, ein Memo und verschiedene Buttons enthält. Ein Array a mit AnsiStrings soll die Texte aufnehmen, die in dem Edit-Fenster eingegeben werden.

   Schreiben Sie die folgenden Funktionen und rufen Sie diese beim Anklicken eines entsprechenden Buttons auf.

   Falls diese Aufgaben im Rahmen einer Gruppe (z.B. einer Vorlesung oder eines Seminars) bearbeitet werden, können einzelne Teilaufgaben auch auf verschiedene Teilnehmer verteilt werden. Die Lösungen der einzelnen Teilaufgaben sollen dann in einem gemeinsamen Projekt zusammen funktionieren.

   Die Teilaufgaben a) bis f) betreffen unsortierte Arrays:

a) Eine Funktion ***pushBack*** soll die als Parameter übergebenen Daten an die nächste freie Position des Arrays schreiben. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Einfügen**“ aufgerufen werden und den Text im Edit-Fenster dem Array hinzufügen.
b) Eine Funktion ***showArray*** soll die Daten des Arrays im Memo anzeigen. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Anzeigen**“ aufgerufen werden.
c) Erstellen Sie für ein Array mit 3 Elementen ein **Ablaufprotokoll** für 4 Aufrufe der Funktion *pushBack* mit den Argumenten „10“, „11“, „12“ und „13“. Formulieren Sie eine **Bedingung**, die nach jedem Aufruf dieser Funktion gilt
d) Eine Funktion ***findLinear*** soll ab einer als Parameter übergebenen Position nach einem ebenfalls als Parameter übergebenen Wert im Array suchen. Falls ein Arrayelement mit diesem Wert gefunden wird, soll dessen Position zurückgegeben werden, und andernfalls der Wert *nElements*. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Linear suchen**“ aufgerufen werden und alle Werte des Arrays ausgegeben, die gleich dem String im Edit-Fenster sind.
e) Eine Funktion ***eraseElement*** soll das Arrayelement an der als Parameter übergebenen Position löschen. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Löschen**“ aufgerufen werden und das erste Element im Array löschen, dessen Wert der Text im Edit-Fenster ist.
f) Eine Funktion ***Auswahlsort*** soll das Array mit dem Auswahlsort sortieren. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Sortieren**“ aufgerufen werden

Die Aufgaben g) bis i) betreffen sortierte Arrays. Falls das Array vor der Ausführung einer dieser Operationen sortiert war, soll es auch nachher noch sortiert sein.

g) Eine Funktion ***findBinary*** soll den als Parameter übergebenen Wert binär im Array suchen. Falls er gefunden wird, soll seine Position zurückgegeben werden, und andernfalls der Wert *nElements*. Rufen Sie diese Funktion beim Anklicken eines Buttons mit der Aufschrift „**Binär suchen**“ auf und zeigen Sie die Position des Strings im Edit-Fenster im Array im Array an, falls er vorhanden ist.
h) Eine Funktion ***insertSorted*** soll einen als Parameter übergebenen Wert an der ebenfalls als Parameter übergebenen Position einfügen. Beim Anklicken eines Buttons „**Sortiert Einfügen**“ soll der String im Edit-Fenster in das Array eingefügt werden.
i) Kann die Einfügeposition in h) mit ***findBinary*** gesucht werden?
j) Eine Funktion ***eraseSorted*** soll das Arrayelement an der als Parameter übergebenen Position löschen. Beim Anklicken eines Buttons „**Sortiert Löschen**“ soll das erste Element im Array gelöscht werden, dessen Wert der Text im Edit-Fenster ist.

k) Schreiben Sie für die Funktionen aus a) bis j) Testfunktionen wie in Abschnitt 3.5.2.

2. Schreiben Sie einen **Stack** mit einem Array, dessen Elemente den Datentyp *AnsiString* haben.

– Beim Anklicken eines Buttons mit der Aufschrift „push“ soll der Text eines Edit-Fensters auf den Stack gelegt werden (sofern noch Platz frei ist).
– Durch Anklicken eines Buttons „pop“ soll der oberste String vom Stack entfernt werden, falls dieser ein Element enthält.
– Durch Anklicken eines Buttons „top“ soll der oberste String des Stacks in einem Memo-Fenster angezeigt werden.

Verwenden Sie dazu eine Variable *StackPtr*, die immer den Index des obersten Elements enthält.

Falls der Stack leer ist, sollen die Buttons *pop* und *top* über die Eigenschaft *Enabled* deaktiviert werden. Falls die Kapazität des verwendeten Arrays erschöpft ist, dasselbe für den Button *push*.

3. Im Jahr 1995 haben die beiden Mathematiker Rabinowitz und Wagon einen Algorithmus vorgestellt, mit dem man die **Kreiszahl π** auf beliebig viele Stellen berechnen kann (Rabinowitz/Wagon 1995, außerdem Stewart 1995).

Im Unterschied zu den meisten anderen Algorithmen zur Berechnung von π wird dabei der Näherungswert nicht als Gleitkommazahl berechnet. Vielmehr liefert dieses Verfahren eine Ziffer nach der anderen als Ganzzahl und wird deshalb auch als **Tröpfelverfahren** bezeichnet (weil die Ziffern wie aus einem Wasserhahn tröpfeln). Es beruht auf einer Reihendarstellung von π, die als Darstellung zu einer gemischten Basis betrachtet wird. Diese Darstellung wird dann in das Dezimalsystem umgerechnet.

Um n Stellen von π zu berechnen, definiert man ein Array A mit 10*n/3 *int*-Elementen und initialisiert alle Werte auf 2. Eine Ganzzahlvariable U für die Überträge erhält den Wert 0.

a) Die ersten n=31 Stellen von π erhält man, indem die folgenden Schritte n-mal wiederholt:

Vom obersten zum zweiten Index des Arrays A (for .. i--) führt man jeweils die folgenden Berechnungen durch, bei denen i jeweils den Index des aktuellen Arrayelements bezeichnet:

– Berechne die Summe

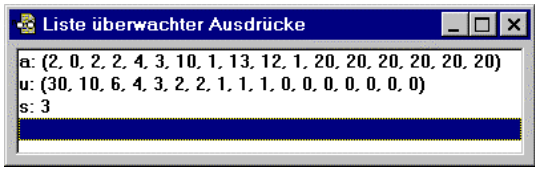
S = U*i+10*A[i];

– Berechne den Quotienten q und den Rest r, der sich bei der Division von S durch 2*i–1 ergibt. Ersetze den bisherigen Wert von A[i] durch r und den bisherigen Übertrag durch q.

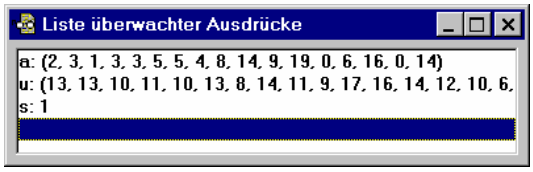
Berechne die Summe S aus dem zehnfachen Wert des ersten Elements von A und dem Übertrag. Ersetze das erste Element von A durch den Rest der Division von S durch 10 und S durch den Quotienten dieser Division.

In den folgenden Listen der überwachten Ausdrücke sind die Werte für das Übertragsfeld in einem Array U gespeichert.

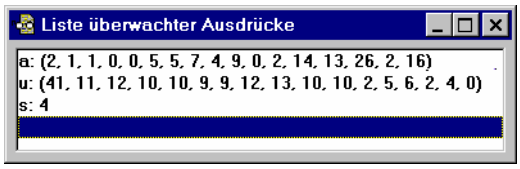
Nach dem ersten Durchlauf ergeben sich so die Werte:



Nach dem zweiten Durchlauf:



Nach dem dritten Durchlauf:



usw. Hier sieht man die ersten drei Stellen von π als den jeweiligen Wert von s nach jeweils einem solchen Durchlauf.

b) Das unter a) beschriebene Verfahren funktioniert allerdings nur für die ersten 31 Stellen von π, weil der Wert von S nach dem Durchlauf einer Schleife größer oder gleich 100 werden kann (maximal 109). Dieser Fall tritt erstmals für die 32. Stelle von π ein und muss durch die folgende Erweiterung des Verfahrens berücksichtigt werden.

Die nach jeweils einem Durchlauf gefundenen Ziffern S sind nicht immer die Ziffern von π, sondern müssen zunächst zwischengespeichert werden. In Abhängigkeit vom jeweils aktuellen Wert von S können dann die bisher zwischengespeicherten Werte entweder als gültige Ziffern freigegeben oder müssen um 1 erhöht werden:

– Falls S weder 9 noch 10 ist, können alle bisher zwischengespeicherten Ziffern als gültige Ziffern freigegeben werden. Die aktuelle Ziffer S muss als vorläufige Ziffer gespeichert werden.
– Falls S=9 ist, wird S den zwischengespeicherten Ziffern hinzugefügt.
– Falls S=10 ist, werden alle bisher zwischengespeicherten Ziffern um 1 erhöht, wobei 9 zu 0 wird. Alle so erhöhten Ziffern können als gültige Ziffern freigegeben werden. Als neue Ziffer wird 0 zwischengespeichert.

Nach Rabinowitz und Wagon sind bei diesem Verfahren alle Zwischenergebnisse für die ersten 5000 Stellen von π kleiner als 600 000 000. Damit können mit 32-bit-Binärzahlen mehr als 5000 Stellen berechnet werden.

Auf einem Pentium 200 dauert die Berechnung von 1000 Stellen ca. 2 Sekunden.

Die ersten 99995 Stellen von π findet man im Internet z.B. unter der Adresse http://www.geom.uiuc.edu/~huberty/math5337/groupe/digits.html. Damit Sie Ihre Ergebnisse vergleichen können, davon die ersten 101 Stellen:

```
const AnsiString pi101 = "pi=3"
                                +"14159265358979323846264338327950288419716939937510"
                                 "58209749445923078164062862089986280348253421170679"
```

3.10.5 Mehrdimensionale Arrays

Aufgaben 3.10.5

1. Geben Sie Anweisungsfolgen an, mit denen man in einem Array die Position der folgenden Elemente findet. Falls es mehrere solche Elemente gibt, soll die Position des zuerst gefundenen bestimmt werden:

   a) in einem zweidimensionalen Array „int a[m][n]“ das kleinste Element,
   b) in einem dreidimensionalen Array „int a[m][n][p]“ das kleinste Element.

   Testen Sie diese Anweisungsfolgen mit verschiedenen Arrays, die bei ihrer Definition initialisiert werden.

2. Ein zweidimensionales Array d mit n Zeilen und Spalten soll eine **Entfernungstabelle** zwischen n Städten darstellen. Dabei ist der Wert des Elements d[i][j] die Entfernung zwischen den Städten mit den Nummern i und j. Alle Elemente d[i][i] haben den Wert 0, außerdem gilt d[i][j]==d[j][i]:

```
d[][3]={{ 0, 10, 20},
        {10,  0, 15},
        {20, 15,  0}};
```

   Eine Fahrtroute durch m dieser Städte soll durch ein Array mit m+1 Elementen dargestellt werden, wobei das erste Element die Anzahl der besuchten Städte enthalten soll. Das Array

```
int r[] = {4, 2, 1, 2, 0};
```

   stellt also die Route 2 -> 1 -> 2 -> 0 dar.

   a) Schreiben Sie eine Anweisungsfolge, die die Länge der durch r dargestellten Route bestimmt.
   b) Ein Array s mit m Zeilen soll m solcher Routen enthalten. Schreiben Sie eine Anweisungsfolge, die die kürzeste dieser Routen bestimmt.

   Testen Sie Ihre Anweisungen mit der Entfernungstabelle d.

3. Beim **Eliminationsverfahren von Gauß** zur Lösung eines linearen Gleichungssystems

   $a_{00}*x_0 + a_{01}*x_1 + ...+ a_{0,n-1}*x_{n-1} = b_0$ // Zeile 0
   $a_{10}*x_0 + a_{11}*x_1 + ...+ a_{1,n-1}*x_{n-1} = b_1$ // Zeile 1
   ...
   $a_{n-1,0}*x_0 + a_{n-1,1}*x_1 + ...+ a_{n-1,n-1}*x_{n-1} = b_{n-1}$ // Zeile n–1

   wird für j=1, ..., n–1 das

   $-a_{j0}/a_{00}$ -fache der Zeile 0

   zur Zeile j addiert. Dadurch werden alle Koeffizienten a_{j0} unterhalb von a_{00} zu 0. Diese Vorgehensweise wiederholt man für die Koeffizienten unterhalb von a_{11}, a_{22} usw. Dadurch erhält man ein Gleichungssystem der Form

   $a'_{00}*x_0 + a'_{01}*x_1 + ...+ a'_{0,n-1}*x_{n-1} = b'_0$ // Zeile 0
   $0*x_0 + a'_{11}*x_1 + ...+ a'_{1,n-1}*x_{n-1} = b'_1$ // Zeile 1
   ...
   $0*x_0 + 0*x_1 + ...+ a'_{n-1,n-1}*x_{n-1} = b'_{n-1}$ // Zeile n–1

   bei dem alle Koeffizienten unterhalb der Diagonale den Wert 0 haben. Dieses Gleichungssystem kann man rückwärts auflösen:

$x_{n-1} = b'_{n-1} / a'_{n-1,n-1}$
$x_{n-2} = (b'_{n-2} - a'_{n-2,n-1} * x_{n-1}) / a'_{n-2,n-2}$
...
$x_0 = (b'_0 - a'_{0,n-1} * x_{n-1} - ... - a'_{01} * x_1) / a'_{0,0}$

a) Schreiben Sie eine Funktion *GaussElimination*, die ein lineares Gleichungssystem löst. Da im Lauf der zahlreichen Rechenschritte Rundungsfehler das Ergebnis verfälschen können, soll außerdem eine Probe gemacht werden. Testen Sie diese Funktion mit n=3 und den Werten

a[n][n]={{1,2,3},{1,4,6},{2,3,7}}; b[n]={1,2,3}; // x={0,–0.4,0.6}
a[n][n]={{2,3,–5},{4,8,–3},{–6,1,4}}; b[n]={–10,–19,–11}; //x={2,–3,1}

b) Dieses Verfahren funktioniert allerdings nur, wenn keine Division durch Null auftritt. Außerdem sollte zur Minimierung von Rundungsfehlern das zur Division in $-a_{j,i}/a_{i,i}$ verwendete Diagonalelement möglichst groß sein. Wenn das Gleichungssystem lösbar ist, erreicht man beide Ziele, indem man unterhalb der Diagonalen die Zeile mit dem größten Wert in der i-ten Spalte sucht und dann diese Zeile mit der i-ten Zeile vertauscht.

Realisieren Sie diese Vorgehensweise mit zwei Funktionen: In der einen soll nach der Zeile mit dem größten Spaltenwert gesucht werden und in der anderen sollen zwei Zeilen vertauscht werden.

3.10.6 Dynamische Programmierung

Aufgabe 3.10.6

Das sogenannte **Pascal-Dreieck** (nach dem Mathematiker Blaise Pascal) entsteht dadurch, dass man zunächst die Zahl 1 in eine Zeile schreibt. Die nächste Zeile entsteht dann aus der vorhergehenden, indem man die Summe der darüber stehenden bildet, wobei man sich links von der ersten und rechts von der letzten eine Null denkt.

```
                1
            1       1
        1       2       1
    1       3       3       1
  1       4       6       4       1
1       5       10      10      5       1
            .............
```

Die Zahlen des Pascal-Dreicks sind die Binomialkoeffizienten. Schreiben Sie eine Funktion *binom*, die für die Argumente n und k den k-ten Wert aus der n-ten Zeile des Pascal-Dreiecks zurückgibt. Dabei werden n und k jeweils ab 0 gezählt, d.h. binom(3,1)=3.

3.10.7 Array-Eigenschaften der VCL Θ

3.11 Strukturen und Klassen

3.11.1 Mit *struct* definierte Klassen

Aufgabe 3.11.1

Ein Datensatz zur Darstellung eines Girokontos soll die Elemente

Adresse, Kontonummer, Kontostand und Kreditlimit

enthalten. Die Adresse soll aus den Elementen

Anrede, Vorname, Nachname, Postleitzahl, Ort, Straße, Hausnummer, Ausland, Vorwahl und Telefonnummer

bestehen. Innerhalb der Adresse sollen zusammengefasst werden:

Vor- und Nachname zu Name,
PLZ bis Hausnummer zu Anschrift und
Vorwahl und Telefonnummer zu Telefon.

Entwerfen Sie ein *struct*, das die Struktur dieses Datensatzes wiedergibt. Geben Sie an, wie die Elemente *Kontonummer*, *Vorname* und *Vorwahl* einer Variablen g dieses Datentyps angesprochen werden können.

3.11.2 Mit *union* definierte Klassen Θ

3.11.3 Die Datentypen *TVarRec* und *Variant* Θ

3.11.4 Bitfelder Θ

Aufgabe 3.11.4

Entwerfen Sie eine Datenstruktur, mit der man die Datenfelder m (Mantisse), e (Exponent) und s (Vorzeichen) einer Gleitkommazahl im Format *double*

Mantisse m	Exponent e	s

0 (Positionen der Bits) 51 52 62 63

direkt ansprechen kann. Schreiben Sie eine Funktion, die das Bitmuster einer als Parameter übergebenen Zahl ausgibt. Zum Test können Sie die Zahl 0.1 (siehe Seite 145erwenden. Geben Sie das Bitmuster der Zahl 1.0 und der Summe einer zehnfachen Addition von 0.1 aus (siehe auch Seite 151).

3.12 Zeiger, Strings und dynamisch erzeugte Variablen

3.12.1 Die Definition von Zeigervariablen

3.12.2 Der Adressoperator, Zuweisungen und generische Zeiger

3.12.3 Ablaufprotokolle für Zeigervariable

3.12.4 Dynamisch erzeugte Variablen: *new* und *delete*

Aufgaben 3.12.4

1. Nach den Definitionen

```
int i=5;
int *pi, pj;
char *pc, pd;
```

sollen die folgenden Zuweisungen in einem Programm stehen:

a) `pi=i;` b) `pi=&i;`
c) `*pi=i;` d) `*pi=&i;`
e) `pi=pj;` f) `pc=&pd;`
g) `pi=pc;` h) `pd=*pi;`
i) `*pi=i**pc;` j) `pi=0;`

Geben Sie an, welche dieser Zuweisungen syntaktisch korrekt sind. Geben Sie außerdem für jeden syntaktisch korrekten Ausdruck den Wert der linken Seite an, wobei die Ergebnisse der zuvor ausgeführten Anweisungen vorausgesetzt werden sollen. Falls die linke Seite ein Zeiger ist, geben Sie den Wert der dereferenzierten Variablen an.

2. Beschreiben Sie das Ergebnis der folgenden Anweisungen:

a)
```
int j=10;
int* p=&j;
*p=17;
Memo1->Lines->Add(IntToStr(j));
Memo1->Lines->Add(IntToStr(*p));
```

b)
```
*new(int)=17;
```

c)
```
int* f_retp()
{
int x=1;
return &x;
}

Memo1->Lines->Add(IntToStr(*f_retp()));
```

d) Vereinfachen Sie die Funktion f:

```
void f()
{
int* pi = new int();
// ...
delete pi;
}
```

3. Beschreiben Sie für die Funktion

```
int f(int a, bool b, bool c, bool d)
{
int* p; int* q; int x=0;
if      (a==1) { p=0; q=0; }
else if (a==2) { p=new int(17); q=new int(18); }
else if (a==3) { p=&x; q=&x; }
if (b) p=q;
if (c) x=*p+*q;
if (d) { delete p; delete q; }
return x;
}
```

das Ergebnis der Aufrufe:

a) `f(0,false,true,true);`
b) `f(1,false,false,true);`
c) `f(2,false,true,false);`
d) `f(2,true,true,true);`
e) `f(3,true,true,true);`
f) `f(3,true,true,false);`

4. In der Programmiersprache C gibt es keine Referenzparameter. Wenn man in C in einer Funktion ein Argument verändern will, übergibt man seine Adresse und verändert die dereferenzierte Variable, wie z.B. in der Funktion

```
void ptr_swap(int* x, int* y)
{
int h=*x;
*x=*y;
*y=h;
}
```

Erstellen Sie ein Ablaufprotokoll für diese Funktion und geben Sie ihre Nachbedingung an.

5. Der Begriff „kopieren“ wird im Zusammenhang mit Zeigern oft undifferenziert für ähnliche, aber doch verschiedene Sachverhalte verwendet. Für eine ausführliche Diskussion dazu siehe Grogono und Sakkinen (2000).

a) Schreiben Sie die folgenden drei Funktionen, die diese unterschiedlichen Sachverhalte illustrieren sollen.

void ***assign****(int*& x, int* y)* soll den Zeiger *y* nach *x* kopieren, aber nicht die Daten, auf die er zeigt.
void ***replace****(int* x, int* y)* soll die Daten, auf die *x* zeigt, durch die Daten überschreiben, auf die *y* zeigt.

void ***clone****(int*& x, int* y)* soll x die Adresse einer neuen dynamischen zuweisen, die auf eine Kopie der Daten *y zeigt.

b) Welche dieser Funktionen können ein Speicherleck zur Folge haben?

3.12.5 Garbage Collection mit der Smart Pointer Klasse *shared_ptr*

3.12.6 Dynamische erzeugte eindimensionale Arrays

Aufgaben 3.12.6

1. Überarbeiten Sie eine Kopie Ihrer Lösung von Aufgabe 3.12.1, 2. (Sieb des Eratosthenes) so, dass die Anzahl der Arrayelemente als Parameter übergeben wird und das Array dynamisch angelegt wird. Der Aufwand für die Änderung soll möglichst gering sein.

2. Implementieren Sie die Funktion *ReAllocate* und testen Sie diese mit einem dynamisch erzeugten Array mit zunächst 2 Elementen, dessen Größe immer wieder verdoppelt wird. Der Test soll einfach nur darin bestehen, dass nach jeder Verdoppelung immer alle Elemente angesprochen werden.

   Beschreiben Sie das Ergebnis der folgenden Anweisungen:

```
a) int* p1;              ReAllocate(p1,0,100);
b) int* p2=new int;      ReAllocate(p2,0,100);
c) int x; int* p3=&x;    ReAllocate(p3,0,100);
d) int* p4=new int[1];   ReAllocate(p4,0,100);
```

3.12.7 Arrays, Zeiger und Zeigerarithmetik

Aufgabe 3.12.7

1. Beschreiben Sie nach den Definitionen

```
int a[10]={1,3,5,7}, *p=a;
```

   den Wert der folgenden Ausdrücke durch indizierte Ausdrücke des Arrays a:

```
*p                    *p+1
(*p)+1                *(p+1)
*(p+3**p)             *p**p
```

2. Überarbeiten Sie die Funktion *Auswahlsort* von Aufgabe 3.10.4 zu einer Funktion *AuswSortPtr*, bei der die Arrayelemente mit Zeigerarithmetik angesprochen werden.

3.12.8 Arrays als Funktionsparameter Θ

Aufgaben 3.12.8

1. a) Überarbeiten Sie die Funktion *Auswahlsort* von Aufgabe 3.10.4, so dass ein Array von *int*-Werten mit einer beliebigen Anzahl von Argumenten als Parameter übergeben werden kann.
   b) Schreiben Sie eine Funktion *isSorted* die prüft, ob die Elemente eines als Parameter übergebenen in einer sortierten Reihenfolge sind.
   c) Testen Sie die Funktionen aus a) und b) an einigen Arrays
      – mit einem Arrayelement
      – mit zwei Arrayelementen in einer richtigen Anordnung
      – mit zwei Arrayelementen in einer falschen Anordnung
      – mit drei Arrayelementen in allen dabei möglichen Anordnungen.

2. Überarbeiten Sie die Funktion *Horner* von Aufgabe 3.10.2, 4. so, dass ihr das Array mit den Koeffizienten des Polynoms als Parameter übergeben wird.

3. Vergleichen Sie in a) und b) den Wert des Ausdrucks

```
sizeof(a)/sizeof(a[0])
```

a)
```
int a[10];
int s=sum(a,sizeof(a)/sizeof(a[0])); // sum wie im Text
```

b)
```
int sum4(int a[])
{
int s=0;
for (int i=0;i< sizeof(a)/sizeof(a[0]);i++)
  s = s + a[i];
return s;
}
```

3.12.9 Konstante Zeiger

3.12.10 Stringliterale, nullterminierte Strings und *char**-Zeiger

Aufgaben 3.12.10

1. a) Schreiben Sie eine Funktion, die die Anzahl der Leerzeichen ' ' in einem als Parameter übergebenen nullterminierten String (*char**) zurückgibt.
   b) Entwerfen Sie systematische Tests für diese Funktion (siehe Abschnitt 3.5.1).
   c) Schreiben Sie eine Testfunktion für diese Tests.

2. Auch auf einfache Fragen gibt es oft vielfältig widersprüchliche Antworten. So hat vor einiger Zeit jemand in einer Diskussionsgruppe im Internet gefragt, ob die folgenden Anweisungen

```
char *x;
x = "hello";
```

von erfahrenen Programmierern als korrekt angesehen würden. Er erhielt darauf über 100 Antworten, aus denen die folgenden vier ausgewählt wurden. Diese geben insbesondere auch einen Hinweis auf die Qualität mancher Beiträge in solchen Diskussionsgruppen. Begründen Sie für jede Antwort, ob sie korrekt ist oder nicht.

a) Nein. Da durch diese Anweisungen kein Speicher reserviert wird, überschreibt die Zuweisung einen anderen Speicherbereich.

b) Diese Anweisungen haben eine Zugriffsverletzung zur Folge. Die folgende Anweisung ist viel besser:

```
char* x="hello";
```

c) Antwort auf b):

Welcher Compiler produziert hier eine Zugriffsverletzung? Die beiden Anweisungen sind völlig gleichwertig.

d) Ich würde die Anweisung

```
char x[]="hello";
```

vorziehen, da diese *sizeof(char*)* Bytes Speicherplatz reserviert.

3. Welche der folgenden Bedingungen sind nach der Definition

```
const char * s="blablabla";
```

in den *if*-Anweisungen zulässig, und welchen Wert haben sie?

a) `if (s==" ") ...`
b) `if (*s==" ") ...`
c) `if (*s=='a'+1) ...`
d) `if (s==' ') ...`

Beschreiben Sie das Ergebnis der folgenden Anweisungen:

e)
```
Form1->Memo1->Lines->Add(s);
Form1->Memo1->Lines->Add(s+1);
Form1->Memo1->Lines->Add(s+20);
```

f)
```
char c='A';
char a[100];
strcpy(a,&c);
```

4. Welche der Zuweisungen in a) bis j) sind nach diesen Definitionen zulässig:

```
const int i=17 ;
int* p1;                char* q1;
const int* p2;          const char* q2;
int const* p3;          char const* q3;
int* const p4=p1;       char* const q4=q1;
```

```
a) p1=p2;          f) q1=q2;
b) p1=&i;          g) q1="abc";
c) p2=p1;          h) q2=q1;
d) *p3=i;          i) *q3='x';
e) *p4=18;         j) *q4='y';
```

5. Für die Suche nach **ähnlichen Strings** gibt es viele Anwendungen: Um Wörter zu finden, deren genaue Schreibweise man nicht kennt, Korrekturvorschläge bei Schreibfehlern zu machen, Plagiate bei studentischen Arbeiten, Mutationen bei DNA-Sequenzen oder Ähnlichkeiten bei Musikstücken zu entdecken. Diese Aufgabe sowie Aufgabe 4.1, 4. befassen sich mit Verfahren zur Identifikation von ähnlichen Strings. Bei Navarro (2001) findet man eine umfangreiche Zusammenfassung zum Thema "Approximate String Matching.

Die sogenannte **Levenshtein-Distanz** (auch **Levenstein-Distanz** oder **Edit-Distanz**) ist eines der wichtigsten Maße für die Ähnlichkeit von zwei Strings s1 und s2. Sie ist die minimale Anzahl der Operationen „ein Zeichen ersetzen“, „ein Zeichen einfügen“ und „ein Zeichen löschen“, die notwendig sind, um den String s1 in s2 umzuwandeln.

Für zwei Strings s1 und s2 der Längen l1 und l2 kann dieses Maß in seiner einfachsten Variante mit einer Matrix d mit (l1+1) Zeilen und (l2+1) Spalten folgendermaßen bestimmt werden:

a) setze das i-te Element der ersten Zeile auf i
b) setze das i-te Element der ersten Spalte auf i
c) berechne die Elemente im Inneren der Matrix zeilenweise durch

d[i][j] = min3(d[i-1][j-1] + cost, d[i][j-1]+1, d[i-1][j]+1);

Dabei ist *min3* das Minimum der drei übergebenen Parameter. Der Wert der *int*-Variablen *cost* ist 0, falls s[i-1]==s[j-1], und sonst 1.

Die Levenshtein-Distanz ist dann der Wert des Elements d[l1][l2]

Beispiel: s1=receieve, s2=retrieve

		r	e	c	e	i	e	v	e
	0	1	2	3	4	5	6	7	8
r	1	0	1	2	3	4	5	6	7
e	2	1	0	1	2	3	4	5	6
t	3	2	1	1	2	3	4	5	6
r	4	3	2	2	2	3	3	4	5
i	5	4	3	3	3	2	3	4	5
e	6	5	4	4	4	3	2	3	4
v	7	6	5	5	5	4	3	2	3
e	8	7	6	6	6	5	4	3	2

Der Wert rechts unten ist dann die Levenshtein-Distanz ed, also ed=d[8][8]=2

Da zwei identische Strings die Edit-Distanz ed=0 haben und außerdem $0 \leq d \leq \max(l1, j\ l2)$ gilt, erhält man mit

1-ed/max(l1,l2)

ein Maß für die Ähnlichkeit von zwei Strings. Schreiben Sie eine Funktion *StringSimilarityEditDistance*, die diesen Wert als Funktionswert zurückgibt.

Zum Testen können Sie Ihre Ergebnisse mit den folgenden Werten vergleichen. Dabei steht *ed* für *StringSimilarityEditDistance*:

```
double n1=sed("receieve","receieve"); // 1-0/8=1
double n2=sed("receieve","receive");  // 1-1/8=0.875
double n3=sed("receieve","receiver"); // 1-2/8=0.75
double n4=sed("receieve","retrieve");  // 1-2/8=0.75
double n5=sed("receieve","reactive"); // 1-3/8=0,525
```

6. Die Funktion *Checksum* soll eine einfache Prüfsumme für Namen mit weniger als 10 Zeichen berechnen:

```
int Checksum(const char* name)
{
char a[10]; // 10 is enough
strcpy(a,name);
int s=0;
for (int i=0; a[i]!=0; i++) s=s+a[i];
return s;
}
```

Beschreiben Sie das Ergebnis dieses Aufrufs:

```
int c= Checksum("Check me, baby");
```

7. Welche dieser Funktionen sind const-korrekt. Ändern Sie die nicht const-korrekten Funktionen so ab, dass sie const-korrekt sind.

```
void assign_c(int*& x, int* y) { x=y;}
void replace_c(int* x, int* y) { *x=*y; }
void clone_c(int*& x, int* y)  { x=new int(*y);}
```

3.12.11 Verkettete Listen

Aufgabe 3.12.11

Falls diese Aufgaben im Rahmen einer Gruppe (z.B. einer Vorlesung oder eines Seminars) bearbeitet werden, können einzelne Teilaufgaben auch auf verschiedene Teilnehmer verteilt werden. Die Lösungen der einzelnen Teilaufgaben sollen dann in einem gemeinsamen Projekt zusammen funktionieren.

1. Ein Programm soll Daten aus einem Edit-Fenster in eine verkettete Liste einhängen. Die beiden Zeiger *first* und *last* sollen bei einer leeren Liste den Wert 0 haben und bei einer nicht leeren Liste immer auf den ersten und letzten Knoten der Liste zeigen.

   Schreiben Sie die folgenden Funktionen und rufen Sie diese beim Anklicken eines entsprechenden Buttons auf. Sie können sich dazu an den Beispielen im Text orientieren.

   a) ***pushFront*** soll einen neuen Knoten mit den als Parameter übergebenen Daten am Anfang in die Liste einhängen. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Am Anfang Einfügen**“ aufgerufen werden und den Text im Edit-Fenster in die Liste einhängen.
   b) ***showList*** soll die Daten der Liste in einem Memo anzeigen. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Anzeigen**“ aufgerufen werden.
   c) Schreiben Sie ein Ablaufprotokoll für 4 Aufrufe der Funktion ***pushFront*** (z.B. mit den Argumenten“10”, “11”, “12” und “13”). Geben Sie eine Beziehung an, die nach jedem Aufruf dieser Funktion gilt.
   d) ***findData*** soll einen Zeiger auf den ersten Knoten der Liste zurückgeben, der die als Parameter übergebenen Daten enthält. Falls kein solcher Knoten existiert, soll der Wert 0 zurückgegeben werden. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Linear suchen**“ aufgerufen werden und alle Strings der Liste ausgegeben, die gleich dem String im Edit-Fenster sind.
   e) ***pushBack*** soll einen neuen Knoten mit den als Parameter übergebenen Daten am Ende der Liste einhängen. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Am Ende Einfügen**“ aufgerufen werden und den Text im Edit-Fenster in die Liste einhängen.
   f) ***insertSorted*** soll die als Parameter übergebenen Daten so in eine sortierte verkette Liste einhängen, dass die Liste anschließend auch noch sortiert ist. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Sortiert**

Einfügen“ aufgerufen werden und den Text im Edit-Fenster in die Liste einhängen. Schreiben Sie dazu eine Funktion *findBefore*, die die Position des Knotens zurückgibt, an der der neue Knoten eingefügt werden soll.

g) ***eraseListnode*** soll den ersten Knoten mit den als Parameter übergebenen Daten löschen, falls ein solcher Knoten existiert. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Löschen**“ aufgerufen werden und den Knoten mit dem Text des Edit-Fensters löschen.

h) ***clearList*** soll den gesamten von der verketteten Liste belegten Speicherplatz wieder freigeben. Danach sollen *first* und *last* wieder eine leere Liste darstellen. Diese Funktion soll beim Anklicken eines Buttons mit der Aufschrift „**Liste Löschen**“ aufgerufen werden

i) Bei welcher dieser Funktionen zeigen *first* und *last* auch nach dem Aufruf auf den ersten und letzten Knoten der Liste, wenn sie vor dem Aufruf auf diese Knoten gezeigt haben?

j) Schreiben Sie für Ihre Lösungen von a) bis g) Testfunktionen (siehe Abschnitt 3.5.2), die wenigstens eine elementare Funktionalität prüfen. Geben Sie die Teile der Aufgabenstellung explizit an, die nicht getestet werden können.

2. Eine **doppelt verkettete Liste** besteht aus Knoten, die nicht nur einen Zeiger *next* auf den nächste Knoten enthalten, sondern außerdem auch noch einen Zeiger *prev* auf den Knoten davor. Eine solche Liste kann man sowohl vorwärts als auch rückwärts durchlaufen. Die doppelt verkettete Liste in dieser Aufgabe soll durch die beiden Zeiger *firstDll* und *lastDll* dargestellt werden, die immer auf den ersten bzw. letzten Knoten zeigen.

   Schreiben Sie die folgenden Funktionen und rufen Sie diese beim Anklicken eines entsprechenden Buttons auf. Sie können sich dazu an der letzten Aufgabe orientieren.

   a) Entwerfen Sie eine Datenstruktur ***DllListnode***, die einen Knoten einer doppelt verketteten Liste darstellt. Eine Funktion *newDllListnode* soll einen solchen Knoten mit den als Argument übergebenen Zeigern auf die Knoten *next* und *prev* sowie den Daten erzeugen und einen Zeiger auf diesen Knoten zurückgeben.
   b) Schreiben Sie eine Funktion ***pushFrontDll***, die einen Knoten mit den als Argument übergebenen Daten in eine doppelt verkettete Liste am Anfang einhängt.
   c) ***showDllForw*** soll die Daten der doppelt verketteten Liste in einem Memo anzeigen und dabei mit *firstDll* beginnen.
   d) ***showDllRev*** soll die Daten der doppelt verketteten Liste in einem Memo anzeigen und dabei mit *lastDll* beginnen.
   e) Stellen Sie das Ergebnis der ersten drei Aufrufe von ***pushFrontDll*** in einem Ablaufprotokoll dar.
   f) ***pushBackDll*** soll einen Knoten mit den als Argument übergebenen Daten am Ende in die verkette Liste einhängen.
   g) Stellen Sie das Ergebnis der ersten drei Aufrufe von ***pushBackDll*** in einem Ablaufprotokoll dar
   h) ***eraseDllListnode*** soll den ersten Knoten mit den als Parameter übergebenen Daten löschen, falls ein solcher Knoten existiert.
   i) ***clearList*** soll den gesamten von der verketteten Liste belegten Speicherplatz wieder freigeben. *firstDll* und *lastDll* sollen danach eine leere Liste darstellen.
   j) Schreiben Sie für Ihre Lösungen Testfunktionen (siehe Abschnitt 3.5.2), die zumindest eine elementare Funktionalität prüfen. Geben Sie die Teile der Aufgabenstellung explizit an, die nicht getestet werden können.

3. Zeigen Sie mit vollständiger Induktion, dass durch wiederholte Aufrufe von

   a) *pushFront* (Aufgabe 1 a) eine einfach verkettete Liste aufgebaut wird, bei der ein neuer Knoten am Anfang eingehängt wird. Vor dem ersten Aufruf soll *first*==*last*==0 sein.
   b) *pushBack* (Aufgabe 1 d) eine einfach verkettete Liste aufgebaut wird, bei der ein neuer Knoten am Ende eingehängt wird. Vor dem ersten Aufruf soll *first*==*last*==0 sein.
   c) *pushFrontDll* (Aufgabe 2 b) eine doppelt verkettete Liste aufgebaut wird, bei der ein neuer Knoten am Anfang eingehängt wird. Vor dem ersten Aufruf soll *firstDLL*==0 sein.
   d) *pushBackDll* (Aufgabe 2 f) eine doppelt verkettete Liste aufgebaut wird, bei der ein neuer Knoten am Ende eingehängt wird. Vor dem ersten Aufruf soll *firstDLL*==0 sein.

3.12.12 Binärbäume

Aufgabe 3.12.12

1. Eine typische Anwendung von balancierten Binärbäumen ist ein **Informationssystem**, das zu einem eindeutigen Schlüsselbegriff eine zugehörige Information findet, z.B. den Preis zu einer Artikelnummer.

   Bei den folgenden Aufgaben geht es aber nur um einige elementare Operationen und nicht um Performance. Deshalb muss der Baum nicht balanciert sein.

a) Entwerfen Sie eine Datenstruktur *Treenode*, die einen Knoten eines Baums mit einem Schlüssel *key* und zugehörigen Daten *data* (Datentyp z.B. *AnsiString* für beide) darstellt. Eine Funktion *newTreenode* soll einen solchen Knoten mit den als Argument übergebenen Zeigern auf die Unterbäume *left* und *right* sowie den Daten *key* und *data* erzeugen.
b) Schreiben Sie eine Funktion *insertBinTreenode*, die einen Knoten mit den als Argument übergebenen Daten in einen Binärbaum einhängt. Rufen Sie diese Funktion beim Anklicken eines Buttons mit Argumenten für *key* und *data* auf, die aus zwei Edit-Fenstern übernommen werden. Die Knoten sollen im Baum entsprechend dem Schlüsselbegriff angeordnet werden.
c) Schreiben Sie eine Funktion *searchBinTree*, die einen Zeiger auf einen Knoten mit dem als Argument übergebenen Schlüsselbegriff zurückgibt, wenn ein solcher Knoten gefunden wird, und andernfalls den Wert 0.
d) Schreiben Sie unter Verwendung der Funktion *seachBinTree* eine Funktion

```
bool ValueToKey(KeyType Key,ValueType& Value)
```

Ihr Funktionswert soll *true* sein, wenn zum Argument für *Key* ein Knoten mit diesem Schlüsselwert gefunden wurde. Die zugehörigen Daten sollen dann als Argument für *Value* zurückgegeben werden. Falls kein passender Wert gefunden wird, soll der Funktionswert *false* sein.
e) Schreiben Sie Testfunktionen (siehe Abschnitt 3.5.2), die zumindest eine elementare Funktionalität Ihrer Lösungen prüfen.

Im Zusammenhang mit den assoziativen Containern der Standardbibliothek wird in Abschnitt 4.4 eine einfachere Lösung dieser Aufgabe vorgestellt, die auf balancierten Binärbäumen beruht.

3.12.13 Zeiger als Parameter und Win32 API Funktionen

Aufgaben 3.12.13

Weitere Informationen zu den Win32 API-Funktionen der nächsten beiden Aufgaben findet man in der Win32-SDK Online-Hilfe (siehe Abschnitt 1.7).

1. Die Win32 API-Funktion

DWORD ***GetTempPath****(*
DWORD nBufferLength, // Größe des Puffers (in Zeichen)
LPTSTR lpBuffer); // Adresse des Puffers

schreibt maximal *nBufferLength* Zeichen des nullterminierten Strings mit dem Verzeichnis für temporäre Dateien in den Speicherbereich ab der Adresse, die als *lpBuffer* übergeben wird. Dieses Verzeichnis wird folgendermaßen bestimmt:

– als der Pfad, der in der Umgebungsvariablen TMP steht
– falls TMP nicht definiert ist, als der Pfad in TEMP
– falls weder TMP noch TEMP definiert sind, als das aktuelle Verzeichnis.

Der Rückgabewert ist 0, falls der Aufruf dieser Funktion fehlschlägt. Andernfalls ist er die Länge des Verzeichnisstrings.

Schreiben Sie eine Funktion *ShowTempPath*, die das Verzeichnis für temporäre Dateien in einem Memo-Fenster ausgibt. Übergeben Sie für *lpBuffer* ein Array mit MAX_PATH Zeichen. Diese vordefinierte Konstante stellt die maximale Länge eines Pfadnamens dar.

2. Die Win32 API-Funktion

DWORD ***GetLogicalDriveStrings****(*
DWORD nBufferLength, // Größe des Puffers (in Zeichen)
LPTSTR lpBuffer); // Adresse des Puffers

schreibt maximal *nBufferLength* Zeichen mit den gültigen Laufwerken in ein *char* Array, dessen Adresse für *lpBuffer* übergeben wird. Die Laufwerke werden als eine Folge von nullterminierten Strings in das Array geschrieben. Zwei aufeinander folgende Nullterminatoren kennzeichnen das Ende.

Beispiel: Bezeichnet man den Nullterminator mit <null>, werden bei einem Rechner mit zwei Laufwerken c:\ und d:\ diese Zeichen in das Array geschrieben:

c:\<null>d:\<null><null>

Schreiben Sie eine Funktion *ShowDriveStrings*, die alle gültigen Laufwerke in einem Memo-Fenster anzeigt. Berücksichtigen Sie die in der Online-Hilfe beschriebenen Rückgabewerte von *GetLogicalDriveStrings*.

3.12.14 Bibliotheksfunktionen für nullterminierte Strings Ɵ

Aufgaben 3.12.14

1. Beschreiben Sie das Ergebnis der folgenden Anweisungen:

```
char c='A'; const char* s="yodel doodle doo"; char* t;
```

   a) `int n1=strlen(&c);`
   b) `int n2=strlen(strstr(s,"doo"));`
   c) `int n3=strlen(strstr("doo",s));`
   d) `strcpy(t,s);`

2. Schreiben Sie eine Funktion *char* cloneString(const char* s)*, die einen als Parameter übergebenen nullterminierten String in einen dynamisch erzeugten Speicherbereich kopiert und dessen Adresse zurückgibt.

   a) Beschreiben Sie den Unterschied der beiden Zuweisungen an s1 und s2:

```
char* t="abc";
char* s1=t;
char* s2=cloneString(t);
```

   b) Was müssen Sie nach einem Aufruf von *cloneString* beachten.

3.12.15 Die Erkennung von „Memory leaks“ mit CodeGuard Ɵ

Aufgabe 3.12.15

1. Aktivieren Sie CodeGuard in Ihren Projekten mit den Lösung der Aufgaben des Abschnitts 3.12 und prüfen Sie, ob Fehler entdeckt werden.

2. Schreiben Sie ein Programm mit den folgenden Fehlern und prüfen Sie, ob CodeGuard diese erkennt:

   a) Zugriff auf einen nicht reservierten Speicherbereich
   b) Zugriff auf einen Speicherbereich, der zuvor wieder freigegeben wurde
   c) Freigabe eines mit *new[]* reservierten Arrays mit *delete*

3.12.16 Zeiger auf Zeiger auf Zeiger auf ... Θ

3.12.17 Dynamisch erzeugte mehrdimensionale Arrays Θ

3.13 Die Stringklasse *AnsiString*

3.13.1 Die Definition von Variablen eines Klassentyps

3.13.2 Funktionen der Klasse *AnsiString*

3.13.3 Globale *AnsiString*-Funktionen

Aufgabe 3.13

1. Ein *AnsiString* mit einem Kalenderdatum soll aus Zahlen für den Tag, den Monat und das Jahr bestehen, die durch einen Punkt getrennt sind (z.B. s="1.2.06" oder s="11.12.2005").

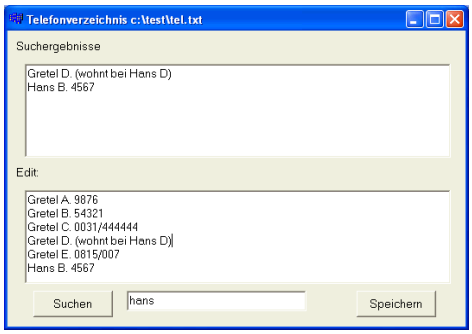
   a) Schreiben Sie eine Funktion *StringToDate*, die die drei Zahlen für den Tag, den Monat und das Jahr in *int*-Werte konvertiert und als Funktionswert des Datentyps *Date*

   ```
   struct Date_t {
     int day, month, year;
   };
   ```

   zurückgibt. Falls der String kein Datum darstellt, sollen *day*, *month* und *year* den Wert –1 erhalten.

   b) Schreiben Sie Testfunktionen (siehe Abschnitt 3.5.2) für *StringToDate*.

2. Schreiben Sie ein kleines Programm, mit dem man in einer Textdatei nach einer Telefonnummer suchen kann:



   Beim Start des Programms wird im Konstruktor des Formulars eine Datei (z.B. "c:\test\tel.txt") in ein Memo-Fenster (im Bild nach *Edit*) geladen.

   a) Nach dem Anklicken des Buttons *suchen* sollen alle Zeilen dieses Memo-Fensters mit der Funktion *Pos* daraufhin überprüft werden, ob sie den Teilstring enthalten, der im Edit-Fenster (im Bild rechts von *suchen*) eingegeben wurde. Falls das zutrifft, wird dieser String in das Memo-Fenster nach *Suchergebnisse:* eingefügt.
   b) Die Suche unter a) soll unabhängig von der Groß- und Kleinschreibung funktionieren.
   c) Da man einen Text in einem Memo-Fenster editieren kann, lassen sich hier auch Telefonnummern neu eintragen, ändern oder löschen. Mit dem Button *speichern* soll die veränderte Liste gespeichert werden können.
   d) Nach der gewünschten Telefonnummer soll nicht nur beim Anklicken des Buttons mit der Aufschrift „suchen“ gesucht werden, sondern auch dann, wenn im Edit-Fenster die Taste *Enter* bzw. *Return* gedrückt wird. Dazu kann man beim Ereignis *OnKeyPress* des Edit-Fensters die *OnClick*-Funktion des Suchen-Buttons aufrufen. Wenn die Taste *Return* gedrückt wurde, hat der Parameter *Key* in *KeyPressed* den Wert 13.

Mit dem Levenstein-Verfahren von Aufgabe 3.12.10, 5. können auch Namen gefunden werden, deren Schreibweise man nicht genau kennt.

3. Geben Sie die Zinsen für ein Kapital von 100 Euro bei einem Zinssatz von 5 bis 15% mit der Funktion *Format* in einem Memo-Fenster aus. Jede Zeile soll dabei folgendermaßen formatiert sein (Kapital: 8 Stellen, Zinssatz: 4 Stellen, Zins: 7 Stellen, alle mit 2 Nachkommastellen, keine führenden Nullen):

   K = 123456,78 p = 12,34% z = 12345,67

   Dabei sollen alle Kommas untereinander stehen. Wählen Sie dazu im Memo-Fenster eine Schriftart, die keine Proportionalschrift ist (z.B. Courier).

4. Schreiben Sie eine Funktion *uintToBinaryAnsiString*, die aus einem nicht negativen ganzzahligen Wert einen String mit den binären Ziffern der Zahl erzeugt.

   Sie können dazu folgendermaßen vorgehen: „Kleben“ Sie an einen zunächst leeren String die Zeichen „0“ oder „1“, und zwar je nachdem, ob die Zahl gerade oder ungerade ist. Verschieben Sie dann die Bits der Zahl um eine Position nach rechts, z.B. durch eine Division durch 2. Wiederholen Sie diese Schritte für alle Bits der Zahl.

5. Schreiben Sie eine Funktion *BigIntAdd*, die zwei positive ganze Zahlen addiert, deren Ziffern in einem String dargestellt werden. Die Länge soll lediglich durch die maximale Länge der Strings begrenzt sein. Beispiel:

   BigIntAdd("123","1000000") = "1000123"

   a) Definieren Sie diese Funktion für AnsiStrings.
   b) Definieren Sie diese Funktion für die Strings aus der Standardbibliothek.
   c) Testen Sie diese Funktionen, indem Sie den Wert des Strings "2" immer wieder verdoppeln. Dabei müssen sich dann die folgenden Werte ergeben:

      2^2=4
      2^3=8
      2^4=16
      ...
      2^{50}=1125899906842624
      2^{100}=1267650600228229401496703205376

6. Übertragen Sie das in der Funktion *mul* von Abschnitt 3.7.8 vorgestellte Multiplikationsverfahren auf die Multiplikation von zwei positiven ganzen Zahlen, deren Ziffern in einem String dargestellt werden. Zur Addition solcher Zahlen kann die Lösung von Aufgabe 5 verwendet werden. Testen Sie diese Funktion mit einer Funktion, die die Fakultät berechnet. Dabei ergibt sich z.B.

   45!=119622220865480194561963161495657715064383733760000000000

   Sie können die Fakultäten der Stringzahlen auch mit denen von *double*-Werten vergleichen.

3.14 Deklarationen mit *typedef* und *typeid*-Ausdrücke

Aufgabe 3.14

In den Header-Dateien von Windows sind unter anderem die Namen *PBOOL*, *LPLONG*, *LPDWORD*, *LPVOID* und *LPCVOID* für Datentypen definiert. Welche Datentypen verbergen sich hinter diesen Namen?

3.15 Aufzählungstypen

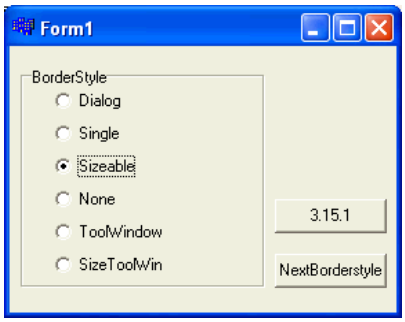
3.15.1 *enum* Konstanten und Konversionen Θ

Aufgaben 3.15

1. Welche der folgenden Definitionen sind zulässig?

   a)
   ```
   enum Monate{Januar=1, Februar, März, April, Mai, Juni,
                                  Juli, August, September, Oktober, November, Dezember};
   enum Sommer {Juni, Juli, August};
   ```
   b)
   ```
   enum Monate {Jan = "Januar", Februar = "Februar"};
   ```
   c)
   ```
   enum {max=100};
   int a[max];
   ```

2. Schreiben Sie ein Programm mit 6 RadioButtons, das etwa folgendermaßen aussieht:



   a) Beim Anklicken des jeweiligen RadioButtons soll die Eigenschaft *Borderstyle* des Formulars auf den entsprechenden Wert gesetzt werden.
   b) Durch sukzessives Anklicken des Buttons mit der Aufschrift *NextBorderstyle* soll die Eigenschaft *BorderStyle* nacheinander alle möglichen Werte annehmen. Nach dem letzten Wert (in der Liste der Aufzählung) soll wieder der erste Wert gesetzt werden.

3.16 Kommentare und interne Programmdokumentation

3.17 Globale, lokale und dynamische Variablen

3.17.1 Die Deklarationsanweisung

3.17.2 Die Verbundanweisung und der lokale Gültigkeitsbereich

Aufgabe 3.17.2

Mit dieser Aufgabe soll lediglich das Konzept der Lokalität geübt werden. Der hier verwendete Programmierstil wird nicht zur Nachahmung empfohlen.

a) Geben Sie in der folgenden Funktion nach jeder Anweisung den Datentyp und den Wert aller bisher deklarierten Variablen an.

```
void verschachtelt()
{
int i,j,k=7;
AnsiString s,t;
  {
  char j;
    {
    float i=0,j=i;
      {
      AnsiString i,j,k;
      i = "ckt"; j = "ra"; k = "vert";
      s = s+k+j+i+" ";
      }
    i = 1; j = 2;
    t = FloatToStr(i)+" + "+FloatToStr(j)+" = "+FloatToStr(k);
      {
      AnsiString t,j,k,i=s;
        {
        char t,j,k;
        t = 's'; j = 'i'; k = 't';
        s = AnsiString(' ')+j+t+k+ ' ';
        }
        {
        char t,j,k;
        t = 'a'; j = 'D'; k = 's';
        s = AnsiString(j) + t + k+s;
        }
      t = "nz sch"; j = "ja ga"; k = "ön ";
      s = s+j+t+k+i;
      }
    }
  }
Form1->Memo1->Lines->Add(t);
Form1->Memo1->Lines->Add(s);
}
```

b) Welcher Text wird beim Aufruf dieser Funktion ausgegeben?

3.17.3 Statische lokale Variablen

3.17.4 Lebensdauer von Variablen und Speicherklassenspezifizierer Θ

Aufgabe 3.17.4

1. Geben Sie die Werte von x und y nach den Funktionsaufrufen an.

```
double f(double x, int n)
{
static int count =0;
count=count+n;
int r=1;
for (int i=0; i<count; i++) r=r*x;
return r;
}

int x=f(3,1);
int y=f(4,2);
```

2. Falls eine Funktion nur für relativ wenige Ganzzahlargumente definiert ist und oft aufgerufen wird, kann man eventuell Zeit sparen, wenn man die Funktionswerte beim ersten Aufruf für alle möglichen Argumente berechnet und in einem statischen Array ablegt. Bei weiteren Aufrufen der Funktion werden dann nur noch die Werte aus dem Array zurückgegeben.

   Setzen Sie dieses Verfahren für die ersten 50 Fibonacci Zahlen um (siehe Aufgabe 3.10.6 und 3.4.6, 2.).

3.18 Referenztypen, Werte- und Referenzparameter

3.18.1 Werteparameter

3.18.2 Referenzparameter

3.18.3 Konstante Referenzparameter

Aufgaben 3.18

Falls Sie die Parameter in Ihren Lösungen der folgenden Aufgaben als Werteparameter übergeben haben, ändern Sie diese zu konstanten Referenzparametern.

a) *Quersumme* (Aufgabe 3.4.6, 1.)
b) *RegentropfenPi* (Aufgabe 3.6.5, 2.)
c) *StringToDate* (Aufgabe 3.13, 1.)

3.19 Weitere Anweisungen

3.19.1 Die Ausdrucksanweisung

3.19.2 Exception Handling: *try* und *throw*

Aufgaben 3.19.2

1. Lösen Sie in Ihrer Funktion *Fibonacci* (Aufgabe 3.4.6, 2.) eine Exception der Klasse *logic_error* aus, wenn sie mit einem Argumenten n>47 aufgerufen wird, bei dem ihre Vorbedingung nicht erfüllt ist (siehe auch Aufgabe 3.7.6 1.). Übergeben Sie dabei eine entsprechende Meldung. Rufen Sie diese Funktionen dann mit Argumenten, die eine Exception auslösen,

   a) in einer *try*-Anweisung auf. Geben die die Meldung in einem Memo aus.
   b) außerhalb von einer *try*-Anweisung auf.

2. Schreiben Sie eine Funktion, die eine Division durch Null und eine Zugriffsverletzung ausführen. Rufen Sie diese Funktion in einer *try*-Anweisung auf und geben Sie die *Message* in einem Memo aus.

3.19.3 Die *switch*-Anweisung Θ

Aufgaben 3.19.3

Lösen Sie die folgenden Aufgaben mit *switch*- anstelle von *if*-Anweisungen. Falls eine der Aufgaben nicht lösbar ist, geben Sie den Grund dafür an.

1. Aufgabe 3.4.1, 3 (Material- und Lagergruppe)
2. Aufgabe 3.4.1, 4 (Datumsvergleich)
3. Aufgabe 3.6.5, 8 (Steuerformel)

3.19.4 Die *do*-Anweisung Θ

3.19.5 Die *for*-Anweisung Θ

3.19.6 Die Sprunganweisungen *goto*, *break* und *continue* Θ

3.19.7 Assembler-Anweisungen Θ

3.20 Ausdrücke

3.20.1 Primäre Ausdrücke Θ

3.20.2 Postfix-Ausdrücke Θ

3.20.3 Unäre Ausdrücke Θ

3.20.4 Typkonversionen in Typecast-Schreibweise Θ

3.20.5 Zeiger auf Klassenelemente Θ

3.20.6 Multiplikative Operatoren Θ

3.20.7 Additive Operatoren Θ

3.20.8 Shift-Operatoren Θ

3.20.9 Vergleichsoperatoren Θ

3.20.10 Gleichheitsoperatoren Θ

3.20.11 Bitweise Operatoren Θ

3.20.12 Logische Operatoren Θ

3.20.13 Der Bedingungsoperator Ɵ

3.20.14 Konstante Ausdrücke Ɵ

3.20.15 Zuweisungsoperatoren

3.20.16 Der Komma-Operator Ɵ

3.20.17 L-Werte und R-Werte Ɵ

3.20.18 Die Priorität und Assoziativität der Operatoren Ɵ

3.20.19 Alternative Zeichenfolgen Ɵ

Aufgaben 3.20.19

1. Welchen Datentyp und Wert haben die Ausdrücke rechts vom Gleichheitszeichen nach der Zeile „//Aufgaben“? Welchen Wert erhält dabei die linke Seite?

```
int i=0;

void Aufg1()
{
int i=2,j=i,k=-6, m=8, n=10000, x=0;
unsigned u=40000u,v=u;

// Aufgaben
int a= k/i/j+n/j;
int b= k/i/j++ + m/j++;
int c= i*4%-k*3;
int d= k/-m-2;
double e= m*n/u/v;
int f= ++::i%++i;
int g = sizeof 17-2;
}
```

   Diese Aufgabe hat ihren Zweck erfüllt, wenn Sie derartige Ausdrücke in Ihren Programmen möglichst vermeiden.

2. Welche Werte werden durch diese Anweisungen ausgegeben:

```
for (int i=0; i<8*sizeof(int); i++)
  Memo1->Lines->Add(IntToStr(i)+": "+IntToStr(1<<i));
```

3. Beschreiben Sie das Ergebnis der Anweisungen in a) bis c) und den Datentyp des Ausdrucks mit dem Bedingungsoperator.

```
double x = 3.14;
int a=1,b=2;
```

   a) `int s = a?b:x;`
   b) `int s = (x > 0) ? 1 : (x < 0) ? -1 : 0;`
   c)
```
for (int n=0;n<4;n++)
  Form1->Memo1->Lines->Add(IntToStr(n)+" file"+
                           ((n==1)?"":"s")+" found");
```

4. Ein C++-Compiler fasst die aufeinander folgenden Zeichen eines Programms so zu Bezeichnern, Schlüsselworten, Literalen und Operatoren zusammen, dass möglichst lange Sequenzen von solchen Zeichenfolgen entstehen.

Überprüfen Sie, welche der Ausdrücke in den Zuweisungen an die Variable k so interpretiert werden können.

```
int  i,j,k;
i=3;j=1;   k= i+++j;
i=3;j=1;   k= ++i+j++;
i=3;j=1;   k= -i+++j;
i=3;j=1;   k= +i+-j;
i=3;j=1;   k= i--+--j;
```

Stellen Sie die Priorität der syntaktisch korrekten Ausdrücke durch Klammern dar und geben Sie nach jeder Zeile den Wert der Variablen i, j und k an.

Auch diese Aufgabe hat ihren Zweck erfüllt, wenn Sie solche Ausdrücke in Ihren Programmen möglichst vermeiden.

5. Im Header <winnt.h> sind unter anderem die folgenden Konstanten definiert:

```
#define FILE_ATTRIBUTE_READONLY         0x00000001
#define FILE_ATTRIBUTE_HIDDEN           0x00000002
#define FILE_ATTRIBUTE_SYSTEM           0x00000004
#define FILE_ATTRIBUTE_DIRECTORY        0x00000010
#define FILE_ATTRIBUTE_ARCHIVE          0x00000020
...
```

Diese können in den Windows-API Funktionen

*BOOL **SetFileAttributes**(*
LPCTSTR lpFileName, // address of filename
DWORD dwFileAttributes); // address of attributes to set

*DWORD **GetFileAttributes**(*
LPCTSTR lpFileName); // address of the name of a file or directory

dazu verwendet werden, die Attribute einer Datei zu setzen bzw. abzufragen. Schreiben Sie mit diesen Funktionen und Konstanten die folgenden Funktionen, denen jeweils der Name einer Datei als Parameter übergeben werden soll. Sie können das Ergebnis dieser Funktionen mit „Eigenschaften“ im Windows-Explorer überprüfen. Legen Sie dazu eigene Dateien in einem Verzeichnis „test“ an.

a) Eine Funktion *SetToROH* soll die Attribute der Datei auf *ReadOnly* und *Hidden* setzen.
b) Eine Funktion *isSystemFile* soll genau dann denn Wert *true* zurückgeben wenn die Datei das Attribut *System* hat.
c) Eine Funktion *SetFileAttributeToHidden* soll das Attribut der Datei auf *Hidden* setzen, ohne dass die anderen Attribute verändert werden.
d) Die Funktion *RemoveFileAttributeHidden* soll das Attribut *Hidden* der Datei löschen, ohne dass die anderen Attribute verändert werden.

3.20.20 Explizite Typkonversionen Θ

3.21 Namensbereiche

3.21.1 Die Definition von benannten Namensbereichen

3.21.2 Die Verwendung von Namen aus Namensbereichen

3.21.3 Aliasnamen für Namensbereiche

3.21.4 Unbenannte Namensbereiche

Aufgaben 3.21

1. Definieren Sie eine Funktion f in einen Namensbereich N und rufen Sie diese

   a) mit dem Namen des Namensbereichs, dem Bereichsoperator und ihrem Namen auf.
   b) nach einer *using*-Anweisung auf.
   c) nach einer *using*-Deklaration auf

   Definieren Sie im Namensbereich N einen verschachtelten Namensbereich N1 mit dem Prototyp Funktion g, die außerhalb des Namensbereichs N definiert wird. Rufen Sie die Funktion g

   d) wie in a), b) und c) auf.
   e) über ein alias für den verschachtelten Namensbereich auf.

2. Geben Sie an, welche Funktionen in den mit a), b), c) und d) gekennzeichneten Zeilen gemeint sind:

```
void f(){};
void g(){};

namespace A {
  void f(){};
  void g(){};
}

namespace B {
  using ::f;  // a)
  using A::g; // b)
}

void h()
{
using namespace B;
f();          // c)
B::g();       // d)
}
```

3.22 Präprozessoranweisungen

3.22.1 Die *#include*-Anweisung

3.22.2 Makros Ɵ

3.22.3 Bedingte Kompilation Ɵ

3.22.4 Pragmas Ɵ

Aufgaben 3.22

Diese Aufgabe erfordert Kenntnisse über Textdateien (siehe Abschnitt 4.3.4).

Schreiben Sie eine kleine Bibliothek mit den folgenden Makros. Da bisher die Unterschiede zwischen Deklarationen und Definitionen noch nicht behandelt wurden, ist es nicht notwendig, diese auf eine „.h“ und eine „.cpp“-Datei aufzuteilen. Schreiben Sie einfach alles in eine Datei *trace.h*.

a) Die Makros *TRACE1*, *TRACE2*, *TRACE3* sollen ähnlich wie *PRINT1* oder *TRACE* die Namen und Werte von einer, zwei oder drei Variablen in eine Datei ausgeben, sowie den Namen der aktuellen Quelltextdatei, die Zeilennummer des Aufrufs und den Namen der Funktion, in der sie aufgerufen wurden.

b) Die *TRACE*-Makros sollen ihre Ausgaben in die Datei schreiben, die mit einem Makro wie *OPEN_TRACELOG* geöffnet wird:

```
#define OPEN_TRACELOG(filename) std::ofstream
                                                                    fout_(filename)
```

c) Falls ein Makro *NOTRACE* definiert ist, soll wie bei NDEBUG und *assert* jeder Aufruf eines der *TRACE*-Makros durch eine leere Anweisung ersetzt werden. Außerdem soll dann die für die Dateioperationen notwendige Datei nicht eingebunden werden:

```
#include <fstream>
```

d) Achten Sie darauf, dass ein Programm auch kompiliert werden kann, wenn *trace.h* mehrfach eingebunden wird.

e) Testen Sie diese Makros mit einigen Ihrer Lösungen (z.B. *Quersumme* oder *Fibonacci* aus Aufgabe 3.4.6).

f) Solche TRACE-Makros haben den Vorteil, dass sie keinen Code erzeugen, wenn das Makro NOTRACE definiert ist, und dass man in ihnen die vordefinierten Makros __FILE__ usw. verwenden kann. Sie haben aber den Nachteil, dass man Fehler in solchen Makros oft nur schwer findet, und dass man den Umfang der Protokollierung nicht während der Laufzeit ändern kann.

   Nehmen Sie in weitere Versionen der Funktionen aus e) anstelle der Makros C++-Anweisungen auf, die die beteiligten Variablen dann in eine Datei schreiben, wenn eine Variable *TraceLevel* größer als ein einstellbarer Grenzwert ist. Auf diese Weise kann man während der Laufzeit des Programms steuern, wie detailliert das trace-Protokoll sein soll.

3.23 Separate Kompilation und statische Bibliotheken

3.23.1 C++-Dateien, Header-Dateien und Object-Dateien

3.23.2 Bindung Θ

3.23.3 Deklarationen und Definitionen Θ

3.23.4 Die „One Definition Rule“ Θ

3.23.5 Die Elemente von Header-Dateien und C++-Dateien Θ

3.23.6 Object-Dateien und Statische Bibliotheken linken Θ

3.23.7 Der Aufruf von in C geschriebenen Funktionen Θ

Aufgabe 3.23

1. Erzeugen Sie mit Kopien Ihrer Lösungen der Aufgaben

   – *Quersumme* (Aufgabe 3.4.6, 1.)
   – *RegentropfenPi* (Aufgabe 3.6.5, 2.)
   – *StringToDate* (Aufgabe 3.13, 1.)

   eine Header-Datei MyLib.h und eine C++-Datei MyLib.cpp, so dass die Header-Datei mit einer *#include*-Anweisung

   – in beliebig viele Quelltextdateien
   – mehrfach in dieselbe Quelltextdateien

   eines Projekts eingebunden werden kann. Diese Bibliothek soll außerdem noch

   – eine mit dem Wert 0 initialisierte gemeinsam genutzte Variable
   – eine gemeinsam genutzte Konstante mit dem Wert 100

   zur Verfügung stellen. Testen Sie diese Bibliotheken, indem Sie diese wie unter a) und b) in einem Projekt *UseLib* verwenden.

   a) Nehmen Sie MyLib.cpp mit *Projekt|Dem Projekt hinzufügen* in das Projekt auf und erzeugen Sie ein lauffähiges Programm.
   b) Fügen Sie dem Projekt mit *Datei|Neu|Formular* ein zweites Formular hinzu. Dieses Formular braucht nicht angezeigt zu werden und soll lediglich die Header-Datei MyLib.h mit einer *#include*-Anweisung aufnehmen und eine globale Variable mit dem Wert *RegentropfenPi(100)* initialisieren.
   c) Nach der erfolgreichen Lösung von a) hat der Compiler aus MyLib.cpp eine Object-Datei mit der Endung MyLib.obj erzeugt. Entfernen Sie die unter a) in das Projekt aufgenommene Datei MyLib.cpp wieder aus dem Projekt und fügen Sie stattdessen die MyLib.obj dem Projekt hinzu.
   d) Entfernen Sie die in c) dem Projekt hinzugefügte Object-Datei MyLib.obj wieder aus dem Projekt und verwenden Sie eine „#pragma link“ Anweisung, um MyLib.obj zum Projekt zu linken.

3.24 Dynamic Link Libraries (DLLs)

3.24.1 DLLs erzeugen Θ

3.24.2 Implizit geladene DLLs Θ

3.24.3 Explizit geladene DLLs Θ

3.24.4 Hilfsprogramme zur Identifizierung von Funktionen in DLLs Θ

3.24.5 DLLs mit VCL Komponenten Θ

3.24.6 Die Verwendung von MS Visual C++ DLLs im C++Builder Θ

Aufgabe 3.24

Legen Sie eine Projektgruppe an (z.B. mit dem Namen *DLLProjectGroup*) und fügen Sie ihr die folgenden Projekte hinzu:

a) ein DLL-Projekt *MyDLL*. Die dabei erzeugte DLL soll die folgenden Funktionen zur Verfügung stellen (verwenden Sie Kopien Ihrer Lösungen)

   – *Quersumme* (Aufgabe 3.4.6, 1.)
   – *RegentropfenPi* (Aufgabe 3.6.5, 2)
   – *StringToDate* (Aufgabe 3.13.1)

   sowie außerdem noch

   – eine mit dem Wert 0 initialisierte gemeinsam genutzte Variable, die in jeder der DLL-Funktionen um 1 erhöht wird.

b) eine Anwendung *UseDLLImpl*, die diese DLL implizit lädt, ihre Funktionen aufruft und den Wert der DLL-Variablen anzeigt.

4 Einige Klassen der Standardbibliothek

4.1 Die Stringklassen *string* und *wstring*

4.1.1 *AnsiString* und *string*: Gemeinsamkeiten und Unterschiede

4.1.2 Einige Elementfunktionen der Klasse *string*

4.1.3 Stringstreams

Aufgaben 4.1

1. Schreiben Sie die Funktionen

   int ***stringToInt****(string s, bool& success);*
   double ***stringToDouble****(string s, bool& success);*

   die einen als Argument übergebenen *string* in einen *int*- oder einen *double*-Wert umwandeln. Der Wert des Arguments für *success* soll angeben, ob die Konversion erfolgreich war. Die Funktionen

   string ***tostring****(int x);*
   string ***tostring****(double);*

   sollen einen *int* und *double*-Wert in einen *string* konvertieren. Testen Sie Ihre Lösungen mit systematischen Testdaten und Testfunktionen wie Abschnitt 3.5.

2. Ein *string* mit einem Kalenderdatum soll aus Zahlen für den Tag, den Monat und das Jahr bestehen, die durch einen Punkt getrennt sind (z.B. s="1.2.06" oder s="11.12.2005").

   a) Schreiben Sie eine Funktion *stringToDate*, die die drei Zahlen für den Tag, den Monat und das Jahr in *int*-Werte konvertiert und als Funktionswert des Datentyps *Date*

   ```
   struct Date_t {
     int day, month, year;
   };
   ```

   zurückgibt. Verwenden Sie dazu nicht die Funktion *erase*. Falls der String kein Datum darstellt, sollen *day*, *month* und *year* den Wert –1 erhalten.

   b) Schreiben Sie Testfunktionen (siehe Abschnitt 3.5.2) für *stringToDate*.

3. Schreiben Sie eine Funktion *tokenize_0*, die alle Teilstrings eines Strings (Datentyp *string*), die durch eines der Zeichen '.', ';', ' ' und '-' getrennt sind, in einem Memo ausgibt. Dabei sollen die Trennzeichen nicht ausgegeben werden. Für die Strings in *test_tokenize* sollte man die als Kommentar angegebene Ausgabe erhalten:

```
void test_tokenize()
{
tokenize_0("456 ab.xy");//Ausgabe "456","ab","xy"
tokenize_0("123");// nur Teilstring, Ausgabe "123"
tokenize_0("");   // leerer String, keine Ausgabe
tokenize_0(" ");// nur Trennzeichen, keine Ausgabe
// Weitere Tests: "456 ab.xy" mit einem Trennzeichen
// am Anfang, am Ende sowie am Anfang und am Ende.
}
```

Testen Sie diese Funktion, indem Sie z.B. die Strings "123" bzw. "456 ab.xy" systematisch mit einem Trennzeichen am Anfang, am Ende sowie am Anfang und am Ende kombinieren.

Anmerkung: In Abschnitt 4.2.3 wird *tokenize_0* zu einer Funktion überarbeitet, die die Teilstrings in einem geeigneten Container zurückgibt.

4. Neben der in Aufgabe 3.12.10, 5. vorgestellten Levenstein-Distanz werden oft auch **N-Gram Verfahren** verwendet, um die Ähnlichkeit von Strings zu messen.

   Die N-gram Methoden beruhen auf der Anzahl der gemeinsamen Teilstrings der Länge N von zwei Strings s1 und s2. Mit N=2 spricht man auch von **Digrammen** und mit N=3 von **Trigrammen**. Vor der Bestimmung der gemeinsamen N-gramme werden die Strings in Grossbuchstaben umgewandelt.

   Beispiel: Die Strings s1="RECEIEVE" und s2="RECEIVE" haben die Trigramme T(s1) und T(s2):

   T(s1) = {"REC", "ECE", "CEI", "EIE", "IEV", "EVE"}
   T(s2) = {"REC", "ECE", "CEI", "EIV", "IVE"}

   Die gemeinsamen Trigramme der beiden Strings sind

   $T(s1) \cap T(s2)$= {"REC", "ECE", "CEI"}

   Die gesamten Trigramme von s1 und s2 sind

   $T(s1) \cup T(s2)$ = {"REC", "ECE", "CEI", "EIE", "IEV", "EVE", "EIV", "IVE"}

   Ein Maß für die Ähnlichkeit von zwei Strings ist dann der Quotient aus der Anzahl der gemeinsamen N-gramme und der Anzahl der gesamten N-gramme:

   StringSimilarityNGram(s1,s2)= $|T(s1) \cap T(s2)| / |T(s1) \cup T(s2)|$

   Da ein String der Länge n immer (n–N+1) N-gramme hat, und außerdem

   $|T(s1) \cup T(s2)| = |T(s1)| + |T(s2)| - |T(s1) \cap T(s2)|$

   gilt, genügt es für die Berechnung der Ähnlichkeit, die gemeinsamen N-gramme zu zählen:

   StringSimilarityNGram(s1,s2)=$|T(s1) \cap T(s2)|$ / (s1.length()–N+1 + s2.length()–N+1 – $|T(s1) \cap T(s2)|$

   Schreiben Sie eine Funktion *StringSimilarityNGram*, die für zwei als Parameter übergebene Strings des Datentyps *string* ihre Ähnlichkeit auf der Basis der gemeinsamen Trigramme berechnet. Einige Beispiele:

   ```
   double n1=StringSimilarityNGram("receieve","receieve");  // 6/6=1
   double n2=StringSimilarityNGram("receieve","receive");   // 3/8=0.375
   double n3=StringSimilarityNGram("receieve","receiver");  // 3/9=0.333
   double n4=StringSimilarityNGram("receieve","retrieve");  // 2/10=0.2
   double n5=StringSimilarityNGram("receieve","reactive");  // 0/12=0
   ```

   Zum Testen können Sie die als Kommentar angegebenen Werte verwenden.

5. Als **Teilfolge** eines Strings s bezeichnet man eine Folge von Zeichen, die dadurch aus s entsteht, dass man Null oder mehr Zeichen aus s entfernt. So sind z.B. "ABB", "BCB" oder "BCBA" Teilfolgen von s1="ABCBDAB", und "BBA", "BCB" oder "BCBA" Teilfolgen von s2="BDCABA". Eine **gemeinsame Teilfolge** von zwei Strings ist dann eine Teilfolge von beiden Strings (z.B. "BCBA" und "BCAB", aber nicht "BCBB"). (Siehe auch Cormen 2001, Abschnitt 15.4, und Crochemore/Lecroq 1996)

Beispiel: Gemeinsame Teilfolgen erhält man z.B. dadurch, dass die beiden Strings so untereinander geschrieben werden, dass gleiche Zeichen übereinander stehen:

```
s1: AB C BDAB ; s1: AB C BDAB
s2:  BDCAB A  ; s2:  BDC   AB A
```

Die Länge der **längsten gemeinsamen Teilfolge** (longest common subsequence, **lcs**) von zwei Strings s1 und s2 kann man über eine Matrix *lc* bestimmen, deren Element *lc[i][j]* die Länge der längsten gemeinsamen Teilfolge des Strings mit den ersten i Zeichen von s1 und des Strings mit den ersten j Zeichen von s2 enthält. Setzt man die Elemente der ersten Zeile und Spalte von lc zunächst auf 0, berechnet man die Elemente im Inneren der Matrix folgendermaßen:

lc[i][j]=lc[i-1][j-1]+1; falls s1[i-1]==s2[j-1]
lc[i][j]=max (lc[i-1][j]; lc[i][j-1]) ; sonst

Diese Anweisungen kann man auch so formulieren:

lc[i][j]=lc[i-1][j-1]+1; falls s1[i-1]==s2[j-1] // Fall 1
lc[i][j]=lc[i-1][j]; falls lc[i-1][j]>=lc[i][j-1] // Fall 2
lc[i][j]=lc[i][j-1]; sonst // Fall 3

Beispiel: s1=”ABCBDAB”, s2=”BDCABA”

		B	D	C	A	B	A
	0	0	o	0	0	0	0
A	0	0 ↑	0 ↑	0 ↑	1 ↖	1 ←	1 ↖
B	0	1 ↖	1 ←	1 ←	1 ↑	2 ↖	2 ←
C	0	1 ↑	1 ↑	2 ↖	2 ←	2 ↑	2 ↑
B	0	1 ↖	1 ↑	2 ↑	2 ↑	3 ↖	3 ←
D	0	1 ↑	2 ↖	2 ↑	2 ↑	3 ↑	3 ↑
A	0	1 ↑	2 ↑	2 ↑	3 ↖	3 ↑	4 ↖
B	0	1 ↖	2 ↑	2 ↑	3 ↑	4 ↖	4 ↑

Die Länge der längsten gemeinsamen Teilfolge ist dann das Element c[l1][l2], wenn l1 die Länge von s1 und l2 die von s2 ist.

a) Schreiben Sie eine Funktion ***lcsLength***, die für zwei als Parameter übergebene Strings die Länge des längsten gemeinsamen Teilstrings zurückgibt.

b) Die längste gemeinsame Teilfolge ***lcs*** von zwei Strings s1 und s2 erhält man dann über eine weitere Matrix b, in die man in jedem der einzelnen Fälle (Fall 1, Fall 2 und Fall 3) von oben einträgt, durch welchen Fall das Element der Matrix *lc* erhalten wurde:

b[i][j]=Diag; // Fall 1
b[i][j]=Up; // Fall 2
b[i][j]=Left; // Fall 3

Die *lcs* erhält man dann ausgehend von einem Matrixelement *lc[m][n]* mit dem Wert *lcsLength*, für das *b[m][n]* den Wert *Diag* hat, indem man rückwärts (**backtracking**) für alle Elemente mit b[i][j]==Diag die Zeichen s[i] an den Anfang der *lcs* anfügt.

Beispiel: Die Matrix oben enthält zwei Elemente (lc[5][7] und lc[6][6]) mit dem Wert *lcsLength* und dem Wert *Diag* in derselben Position der Matrix b. Ausgehend vom ersten Element lc[5][7] erhält man die Teilfolge ”BCAB”, und ausgehend von lc[6][6] lcs=”BCBA” Geht man von diesen

Mit diesem und ähnlichen Algorithmen kann man die Differenz von Dateien, genetischen Sequenzen usw. bestimmen. Das UNIX-Programm „diff“ (das auch für Windows verfügbar ist) betrachtet die Zeilen eines Textes als die Elemente und zeigt die Differenzen (eingefügte und gelöschte Zeilen) von zwei Dateien an.

4.2 Sequenzielle Container der Standardbibliothek

4.2.1 Die Container-Klasse *vector*

4.2.2 Iteratoren

4.2.3 Algorithmen der Standardbibliothek

Aufgaben 4.2.3

1. Überarbeiten Sie eine Kopie Ihrer Lösung der Aufgabe 3.10.2, 1. (Sieb des Eratosthenes) so, dass anstelle eines Arrays ein *vector* verwendet wird. In drei verschiedenen Varianten der Lösung soll auf die Elemente folgendermaßen zugegriffen werden:

   a) mit dem Indexoperator,
   b) mit *at*
   c) mit Iteratoren.

2. Verwenden Sie für die Aufgaben in a) bis c) geeignete Algorithmen der STL. Um den Aufwand für diese Aufgabe klein zu halten, können sie hart kodierte Arrays verwenden.

   a) Sortieren Sie ein Array mit 10 Werten des Datentyps *int*. Geben Sie alle Werte des Arrays nach dem Sortieren aus.
   b) Kopieren Sie alle Elemente eines Arrays in ein anderes, das genügend groß ist.
   c) Prüfen Sie, ob zwei Arrays dieselben Elemente haben.

3. Lösen Sie Aufgabe 3.10.4, 1. (ohne c) und j)) mit einem Vektor anstelle eines Arrays. Verwenden Sie Iteratoren und vordefinierte Bibliotheksfunktionen, wo immer das möglich ist.

4. Überarbeiten Sie die Funktion *tokenize* von Aufgabe 4.1, 3. so, dass sie die gefundenen Teilstrings in einem *vector* zurückgibt. Eine zweite Version der Funktion *tokenize* soll einen AnsiString zerlegen und die Teilstrings in einem *vector* mit AnsiStrings zurückgeben. Damit diese Funktionen später zur Lösung weiterer Aufgaben verwendet werden können, sollen sie in eine Datei *\Loesungen_CB2006\CppUtils\StringUt.h* aufgenommen werden, die mit *#include* in ein Programm eingebunden werden kann.

   Entwerfen Sie geeignete Testfälle und testen Sie diese Funktionen mit Testfunktionen.

5. **Ein einfaches Programm zur Datenverwaltung**

   Schreiben Sie ein Programm, mit dem man Datensätze des in Abschnitt 3.11.1 vorgestellten Datentyps *Konto-bewegung* ein- und ausgeben kann. Zur Verwaltung der Datensätze soll als Container ein *vector* verwendet werden.

   Dieses Programm soll etwa folgendermaßen aussehen:

Die Namen der Edit-Fenster sollen den Namen der Elemente der Struktur entsprechen, ebenso die Namen der Labels. Die Tab-Reihenfolge soll von oben nach unten laufen. Damit der Aufwand für diese Aufgabe nicht zu groß wird, kann auf Plausibilitätsprüfungen der Eingaben (z.B. mit einer Edit-Maske) verzichtet werden.

a) Schreiben Sie die beiden Funktionen:

```
Kontobewegung FormToKB()
{ // gibt die Werte des Formulars Form1 zurück
Kontobewegung k;
k.KontoNr = StrToInt(Form1->EKontoNr->Text);
// ...  usw.
return k;
}

void KBToForm(Kontobewegung k)
{ // überträgt die Daten von k in das Formular Form1
Form1->EKontoNr->Text=IntToStr(k.KontoNr);
// ...  usw.
}
```

Die erste soll die Daten aus den Edit-Feldern des Formulars als Funktionswert zurückgeben. Die zweite soll die Daten des Parameters k den Edit-Feldern des Formulars zuweisen. Zur Umwandlung des Datumsstrings in Ganzzahlwerte können Sie sich an Aufgabe 4.1,1. orientieren.

Damit der Datentyp *Kontobewegung* auch in anderen Programmen verwendet werden kann, soll er in einer eigenen Datei definiert werden (z.B. mit dem Namen *KBDecl.h* im Verzeichnis *\Loesungen_CB2006\CppUtils*). Diese Datei soll dann in das aktuelle Projekt mit einer *include*-Anweisung eingebunden werden, z.B.

```
#include "\Loesungen_CB2006\CppUtils\KBDecl.h"
```

Entsprechend sollen die Funktionen *KBToForm* und *FormToKB* in einer Datei mit dem Namen *KBForms.h* enthalten sein.

b) Beim Anklicken des Buttons mit der Aufschrift „Speichern“ sollen die Daten der Eingabemaske unter Verwendung der Funktion *FormToKB* in den Container eingefügt werden.

c) Beim Anklicken des Buttons mit der Aufschrift „<“ soll der Datensatz angezeigt werden, der sich im Container eine Position vor der des aktuell angezeigten Datensatzes befindet.

d) Beim Anklicken des Buttons mit der Aufschrift „>“ soll der Datensatz angezeigt werden, der im Container auf den aktuell angezeigten Datensatz folgt.

e) Beim Anklicken der Buttons mit der Aufschrift „<<“ bzw. „>>“ soll der erste bzw. letzte Datensatz des Containers angezeigt werden.

f) Beim Anklicken des Buttons mit der Aufschrift „Löschen“ soll der aktuell angezeigte Datensatz aus dem Container gelöscht werden.

g) Aktivieren bzw. deaktivieren Sie die Buttons mit der Aufschrift „<<“, „<“, „>“, „>>“ und „Löschen“ in Abhängigkeit vom aktuellen Datenbestand im Container. Falls der Container leer ist, sollen alle diese Buttons deaktiviert werden. Andernfalls:

   – Die Buttons << und >> sollen immer aktiviert sein.
   – Falls vor dem aktuell angezeigten Datensatz kein weiterer kommt, soll der Button < deaktiviert sein und andernfalls aktiviert.
   – Falls nach dem aktuell angezeigten Datensatz kein weiterer kommt, soll der Button > deaktiviert sein und andernfalls aktiviert.

   Fassen Sie die Aktivierung und Deaktivierung der Buttons in einer Funktion *EnableButtons* zusammen. Diese kann dann nach jedem Anklicken eines der Buttons aufgerufen werden.

h) Nach dem Einfügen oder Löschen eines Datensatzes können die bisherigen Werte der Iteratoren des Containers ungültig werden. Damit der Iterator für die aktuelle Position auch nach einer dieser Operationen einen definierten Wert hat, soll er auf einen plausiblen Wert gesetzt werden. Beispielsweise wäre die Position des letzten Elements ein plausibler Wert nach dem Speichern eines Datensatzes. Nach dem Löschen kann der Datensatz nach dem gelöschten angezeigt werden.

i) Geben Sie die Nummer des aktuell angezeigten Datensatzes auf einem Label an. Sie kann als Differenz des Iterators für die aktuelle Position und des Iterators für das erste Element bestimmt werden. Diese Differenz ist dann wie bei einer Subtraktion von Zeigern ein Ganzzahlwert.

5. Schreiben Sie eine Funktion *Sum*, der als Parameter ein *vector* mit *double*-Elementen übergeben wird, und deren Funktionswert die Summe dieser Elemente ist. Vergleichen Sie an diesem Beispiel die Parameterübergabe von Vektoren mit der von Arrays (siehe Abschnitt 3.12.8).

4.2.4 Die Speicherverwaltung bei Vektoren Θ

4.2.5 Mehrdimensionale Vektoren Θ

Aufgabe 4.2.5

1. Schreiben Sie eine Funktion, die in einer Schleife nacheinander Elemente in einen Vektor ablegt und bei jeder Änderung der *capacity* die Anzahl der Elemente und die *capacity* ausgibt.

2. Überarbeiten Sie die Funktion *GaussElimination* von Aufgabe 3.10.5.3 so, dass sie für ein beliebiges n ein lineares Gleichungssystem mit n Gleichungen und n Unbekannten löst. Die Koeffizientenmatrix a, n, die rechte Seite b, der Lösungsvektor x und der Vektor p für die Probe sollen als Parameter übergeben werden. Übernehmen Sie möglichst viele Anweisungen aus der Lösung von Aufgabe 3.10.5.3.

   Lesen Sie die Koeffizienten und die rechte Seite des Gleichungssystems aus einem StringGrid (Tool-Palette Seite „Zusätzlich“) ein. In die Zellen *cells[i][j]* eines StringGrid kann man Werte eingegeben, wenn die Option *goEditing* unter *Options* auf *true* gesetzt wird. Die Größe des StringGrid soll während der Laufzeit des Programms gesetzt werden können. Informieren Sie sich dazu in der Online-Hilfe über die Eigenschaften *RowCount* und *ColCount*. Beachten Sie, dass ein StringGrid seine Zellen mit *[Spalte][Zeile]* adressiert und nicht wie C++ mit *[Zeile][Spalte]*.

3. Definieren Sie eine Dreiecksmatrix mit n Zeilen, deren i-te Zeile i Spalten hat (i=1 ... n). Setzen Sie die Elemente am Rand dieser Matrix auf den Wert 1. Jedes Element im Inneren der Dreiecksmatrix soll die Summe der beiden Elemente aus der Zeile darüber sein, die links und über diesem Element stehen (Pascal Dreieck):

   1
   1 1
   1 2 1
   1 3 3 1
   usw.

4.2.6 Die Container-Klassen *list* und *deque*

4.2.7 Gemeinsamkeiten und Unterschiede der sequenziellen Container

Aufgabe 4.2.7

Übertragen Sie die Lösung der Aufgabe 4.2.3, 2 mit möglichst wenigen Änderungen auf *list* und *deque*.

4.2.8 Die Container-Adapter *stack*, *queue* und *priority_queue* Ɵ

4.2.9 Container mit Zeigern

4.2.10 Die verschiedenen STL-Implementationen im C++Builder Ɵ

4.2.11 Die Container-Klasse *bitset* Ɵ

4.3 Dateibearbeitung mit den Stream-Klassen

4.3.1 Stream-Variablen, ihre Verbindung mit Dateien und ihr Zustand

4.3.2 Fehler und der Zustand von Stream-Variablen

4.3.3 Lesen und Schreiben von Binärdaten mit *read* und *write*

Aufgaben 4.3.3

Für diese Aufgaben werden die folgenden Vorbereitungen empfohlen:

– Damit nicht versehentlich Dateien gelöscht werden, soll zunächst ein Testverzeichnis (z.B. „c:\test“) angelegt werden. Alle Dateioperationen sollen nur mit Dateien in diesem Verzeichnis durchgeführt werden. Im jeweiligen OpenDialog (siehe Abschnitt 2.10) soll dieses Verzeichnis als Voreinstellung gesetzt werden.
– Damit Dateinamen im Windows-Explorer vollständig angezeigt werden, sollte im Windows-Explorer die Markierung unter *Extras|Ordneroptionen|Ansicht|Erweiterungen bei bekannten Dateitypen ausblenden* entfernt werden.

Überprüfen Sie bei jeder Aufgabe nach jeder Dateioperation, ob sie erfolgreich war. Wenn nicht, soll z.B. mit *ShowMessage(" ... ")* eine Fehlermeldung ausgegeben werden.

1. In einem Projekt mit dem Namen *Files* sollen die folgenden Funktionen nach dem Anklicken eines Buttons aufgerufen werden. Die Namen der Dateien werden am einfachsten hart kodiert ins Programm geschrieben. Sie können aber auch über einen OpenDialog erfragt und mit der Elementfunktion *c_str* (z.B. *OpenDialog1->FileName.c_str()*) an die Streamklassen übergeben werden.

   Kopieren Sie zuerst einige Text- und Binärdateien (z.B. die cpp- und exe-Datei einer ersten Version des Projekts *Files*) in das Testverzeichnis und testen Sie damit die folgenden Funktionen.

   a) Eine Funktion *countBytes* soll die Größe einer Datei (die Anzahl der Bytes) als Funktionswert zurückgeben, deren Name als Parameter übergeben wird. Dazu soll die Datei im Binärmodus geöffnet werden und ein Zeichen nach dem anderen gelesen werden. Eine weitere Funktion *countBytesTxt* soll sich von *countBytes* nur dadurch unterscheiden, dass die Datei im Textmodus geöffnet wird.
   Vergleichen Sie die Ergebnisse dieser Funktionen jeweils für Textdateien (z.B. „c:\test\FilesU.cpp“) und Binärdateien (z.B. „c:\test\Files.exe“) mit der Anzeige im Windows-Explorer.
   Bei einer weiteren Variante der Funktion *countBytes* soll ein *bool*-Parameter übergeben werden, mit dem gesteuert wird, ob die Datei im Text- oder Binärmodus geöffnet wird.
   b) Eine Funktion *copyFile* soll eine Datei zeichenweise in eine andere kopieren. Die Namen der Dateien sollen als Parameter übergeben werden.
   c) Eine Funktion *compareFiles* soll für zwei als Parameter übergebene Dateien prüfen, ob sie identisch sind.
   d) Eine Funktion *writeNTString* soll einen nullterminierten String (Datentyp *char**) einschließlich des Nullterminators in einen *ofstream* schreiben. Sowohl der String als auch der Stream sollen als Parameter

übergeben werden. Eine Funktion *readNTString* soll ab der aktuellen Dateiposition in einem als Parameter übergebenen *ifstream* alle Zeichen bis zum nächsten Nullterminator '\0' lesen und alle Zeichen in einem *string* als Funktionswert zurückgeben.

2. **Ein einfaches Programm zur Datenverwaltung mit Dateien**

Schreiben Sie ein Programm (z.B. mit dem Namen *SimpleDB*), mit dem man Datensätze des Typ *Kontobewegung* in einer Datei speichern und anzeigen kann. Das Formular dieses Programms soll eine Eingabemaske für eine Kontobewegung enthalten (wie in Aufgabe 4.2.3, 5.). Sie können die Eingabemaske in dieses Programm kopieren (mit gedrückter linker Maustaste die Maske im Formular markieren, dann mit *Strg+C* kopieren und mit *Strg+V* in das neue Formular einfügen).

In einem Menü *Datei* sollen die folgenden Menüpunkte angeboten werden:

Neu
Öffnen
Schließen
Zufallsdatei

Das Formular soll Buttons mit der Aufschrift „>“ und „speichern“ erhalten, die beim Start des Programms zunächst alle deaktiviert sind (Eigenschaft *Enabled*).

a) **Daten in eine Datei schreiben**

Nach der Auswahl der Menüoption *Datei|Neu* soll in einem SaveDialog nach einem Dateinamen gefragt und eine neue Datei mit diesem Namen angelegt werden. Damit nicht jedes Mal mühsam ein neuer Name eingetippt werden muss, soll der Name "kb.dat" im Verzeichnis "c:\test" vorgeschlagen werden.

Nach dem Anlegen der Datei soll ein Button mit der Aufschrift „Speichern“ aktiviert werden. Wenn er angeklickt wird, sollen die Daten der Eingabemaske in die Datei geschrieben (am Ende angefügt) werden.

Über die Menüoption *Datei|Schließen* soll die Datei geschlossen werden. Dabei sollen auch alle Buttons deaktiviert werden.

b) **Daten aus einer Datei lesen**

Mit der Menüoption *Datei|Öffnen* soll eine bestehende Datei geöffnet werden. Wenn sie nicht leer ist, soll der erste Datensatz angezeigt werden.

Nach dem Öffnen der Datei soll der Button mit der Aufschrift „>“ aktiviert werden. Wenn er angeklickt wird, soll der jeweils nächste Datensatz angezeigt werden. Nachdem alle Datensätze gelesen wurden (*f.eof()==true*), soll der Button „>“ deaktiviert werden.

Wäre es sinnvoll, den Button *speichern* zu aktivieren, um den aktuell angezeigten Datensatz korrigieren und korrigiert in die Datei schreiben zu können?

c) **Testdateien anlegen**

Unter *Datei|Zufallsdatei* soll nach einem *OpenDialog* eine Datei von Kontobewegungen erzeugt werden. Mit einem *InputQuery*-Fenster kann zunächst gefragt werden, wie viele Datensätze in die Datei geschrieben werden sollen.

Schreiben Sie dazu eine Funktion *Zufalls_KB*, die einen Datensatz des Typs *Kontobewegung* mit zufälligen, aber plausiblen Daten als Funktionswert hat. Die einzelnen Werte können z.B. folgendermaßen bestimmt werden:

KontoNr:	ein zufälliger Wert zwischen 1000 und 1099
Tag:	ein zufälliger Wert zwischen 1 und 31
Monat:	ein zufälliger Wert zwischen 1 und 12
Jahr:	ein zufälliger Wert zwischen 2007 und 2008
BewArt:	'+' oder '–'
Betrag:	ein zufälliger Wert zwischen 0,01 und 300

Damit einige Ausdrucke, die in späteren Aufgaben erzeugt werden, einigermaßen realistisch aussehen, soll zu einer bestimmten Kontonummer immer derselbe Kontoinhaber gehören. Das kann man dadurch erreichen, dass man den

Namen des Kontoinhabers aus einem Array von je 10 Vor- und Nachnamen zusammensetzt, wobei die vorletzte Ziffer der Kontonummer den Nachnamen und die letzte Ziffer den Vornamen bestimmt.

Damit diese Funktion auch in anderen Programmen verwendet werden kann, soll sie in die Datei *KBDecl.h* aufgenommen werden.

d) **Linear suchen**

Erweitern Sie das Formular um einen Button *LinSuchen*. Nach dem Anklicken dieses Buttons soll in der Datei ab der aktuellen Position nach dem nächsten Datensatz gesucht werden, dessen Namen den String enthält, der für den Namen des Kontoinhabers in einem Edit-Fenster eingegeben wurde.

Falls ein Datensatz gefunden wird, soll er angezeigt werden. Andernfalls soll mit *ShowMessage* darauf hingewiesen werden, dass kein solcher Datensatz gefunden wurde.

Die bisher vorgestellten Sprachelemente bieten keine Möglichkeit, eine Datei rückwärts zu durchlaufen. Entsprechende Optionen werden in späteren Aufgaben ergänzt.

4.3.4 Lesen und Schreiben von Daten mit den Operatoren << und >>

Aufgabe 4.3.4

1. Das HTML-Format ist ein Textformat, das unter anderem für Internetseiten verwendet wird. Damit es auf möglichst vielen verschiedenen Rechner- und Betriebssystemen eingesetzt werden kann, verwendet es nur Zeichen des ASCII-Zeichensatzes. Formatangaben werden auch als Markierungen bezeichnet und bestehen aus einem Paar von spitzen Klammern <>, zwischen denen Schlüsselworte und eventuell noch Parameter stehen.

   Ein HTML-Dokument beginnt mit der Markierung <HTML> und endet mit </HTML>. Wie bei diesem Paar von Markierungen werden Bereiche oft durch Markierungen begrenzt, bei denen die Markierung für das Ende des Bereichs mit dem Zeichen „/“ beginnt, und bei der das Schlüsselwort in der Ende-Markierung gleich oder ähnlich ist wie in der Anfangsmarkierung.

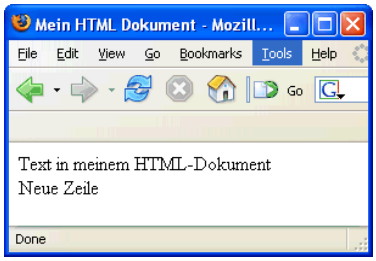
   Bereiche können verschachtelt werden. So kann ein HTML-Dokument einen durch <HEAD> und </HEAD> begrenzten Bereich mit Angaben enthalten, die das gesamte Dokument betreffen. In einem solchen Bereich kann z.B. zwischen <TITLE> und </TITLE> der Text stehen, der in der Titelzeile des Browsers angezeigt wird.

   Der im Hauptfenster des Browsers angezeigte Text ist in einem durch <BODY> und </BODY> begrenzten Bereich des HTML-Dokuments enthalten.

   So wird zum Beispiel das HTML-Dokument

```
<HTML>
  <HEAD>
    <TITLE>
     Mein HTML Dokument
    </TITLE>
  </HEAD>
  <BODY>
    Text in meinem
    HTML-Dokument
    <BR>Neue Zeile
  </BODY>
</HTML>
```

   in einem HTML-Browser folgendermaßen dargestellt:



Die Einrückungen im HTML-Dokument wirken sich nicht auf die Formatierung aus und wurden hier nur zur besseren Übersichtlichkeit aufgenommen. Zeilenvorschübe im Text werden ebenfalls ignoriert und nur durch die Markierung <BR> erzeugt.

Da die Umlaute nicht zum ASCII-Zeichensatz gehören, werden sie durch spezielle Zeichenkombinationen dargestellt:

ä: ä ö: ö ü: ü ß: ß
Ä: Ä Ö: Ö Ü: Ü

Beispiel: „In München steht ein Hofbräuhaus."

a) Schreiben Sie eine Funktion *TextToHtml*, die aus einer Textdatei ein HTML-Dokument erzeugt. Dazu sollen die notwendigen Markierungen erzeugt und der gesamte Text der Textdatei in einen durch <BODY> und </BODY> begrenzten Bereich kopiert werden. Die Titelzeile des Browsers soll den Dateinamen anzeigen. Die einzelnen Zeilen der Textdatei sollen im Browser ebenfalls als einzelne Zeilen dargestellt werden. Die Umlaute sollen durch die entsprechenden Zeichenkombinationen ersetzt werden.

b) Durch die Markierungen <TABLE BORDER> und </TABLE> werden Tabellen in einem HTML-Dokument begrenzt. In einem solchen Bereich wird

- eine Spaltenüberschrift durch <TH> eingeleitet
- eine neue Tabellenzeile durch <TR> eingeleitet
- in einer Tabellenzeile ein neues Datenelement durch <TD> eingeleitet.

Alle diese Markierungen brauchen keine Ende-Markierung, da sie durch die nächste Markierung dieser Art begrenzt werden.

Schreiben Sie eine Funktion *KBToHtml*, die aus einer Datei von Kontobewegungen (Datentyp *Kontobewegung*) eine HTML-Datei erzeugt, die die Daten der Datei in einer Tabelle darstellt.

2. Schreiben Sie eine Funktion *Listendruck*, die aus einer Datei von Kontobewegungen (Datentyp *Kontobewegung*) eine Textdatei erzeugt, die etwa folgendermaßen aussieht:

```
Datei: c:\test\kb.dat                            Seite 1
Kto. Kontoinhaber                  Datum      Betrag

1004 Duestrip, Donald          31. 9.2004   +   21.75
1099 Prince, Charlie           15. 1.2005   -  168.61
1011 Mieze, Alexander           6.11.2004   -  174.06
```

a) Jede Seite soll mit einem „Blattkopf" beginnen, der in der ersten Zeile den Dateinamen und die aktuelle Seitenzahl enthält und in der zweiten Zeile die Bedeutung der darunter aufgeführten Daten erläutert. Darauf soll eine Leerzeile folgen. Die dazu erforderlichen Anweisungen sollen in einer Funktion *Blattkopf* zusammengefasst werden.

b) Die Anzahl der Zeilen, die auf eine Seite gedruckt werden können, hängt von der Schriftgröße und vom Papierformat ab. Nehmen Sie deshalb an, dass auf eine Seite 72 Zeilen gedruckt werden können. Davon sollen maximal 60 bedruckt werden. Damit diese Funktion ohne großen Aufwand an andere Papierformate angepasst werden kann, sollen die Anzahl der Druckzeilen pro Seite sowie die Anzahl der Zeilen pro Seite als Variablen vereinbart werden.

Sobald auf eine Seite mehr Zeilen gedruckt sind, als die Variable *Druckzeilen_pro_Seite* angibt, soll ein Blattvorschub (z.B. als eine Folge von Leerzeilen) erfolgen, ebenso nach dem Ausdruck des letzten Datensatzes der Datei. Der Blattvorschub soll in einer Funktion mit dem Namen *Blattvorschub* ausgeführt werden.

c) Am Ende jeder Seite soll die Summe der Beträge aller Kontobewegungen mit der Bewegungsart '+' und die aller Kontobewegungen mit der Bewegungsart '–' gedruckt werden.

d) Die letzte Seite soll (außer wenn eine leere Datei ausgedruckt wird) nicht nur aus einem Blattkopf bestehen können.

Da der Ausdruck in eine Datei erfolgt und nicht auf einen Drucker, wird auf alle Feinheiten der Druckgestaltung verzichtet.

Diese Funktion kann in dem Programm von Aufgabe 4.3.3.2 unter *Datei\Drucken* angeboten werden. Den Namen der auszudruckenden Datei kann man in einem *OpenDialog* erfragen.

4.3.5 Manipulatoren und Funktionen zur Formatierung von Texten Θ

4.3.6 Dateibearbeitung im Direktzugriff Θ

Aufgaben 4.3.6

Erweitern Sie das Programm *SimpleDB* (Aufgabe 4.3.3, 2.) um folgende Optionen:

a) Die Erweiterungen unter b) bis e) sind unabhängig davon, ob eine neue oder eine bereits bestehende Datei geöffnet wird. Falls Sie eine neue Datei bisher nur zum Schreiben oder eine bestehende Datei nur zum Lesen geöffnet haben, passen Sie die Modi beim Öffnen der Datei so an, dass die folgenden Erweiterungen in beiden Fällen funktionieren.
b) Beim Anklicken eines Buttons mit der Aufschrift „<“ soll der Datensatz angezeigt werden, der in der Datei vor dem aktuell angezeigten kommt. Falls der aktuell angezeigte Datensatz der erste in der Datei ist, soll dieser Button deaktiviert werden.
c) Beim Anklicken eines Buttons mit der Aufschrift „<<“ soll der erste Datensatz der Datei angezeigt werden.
d) Beim Anklicken eines Buttons mit der Aufschrift „>>“ soll der letzte Datensatz der Datei angezeigt werden.
e) Beim Anklicken eines Buttons mit der Aufschrift „Korrektur“ soll der aktuell angezeigte Datensatz an der Position in die Datei zurückgeschrieben werden, von der er gelesen wurde.

4.3.7 Sortieren, Mischen und Gruppenverarbeitung Θ

Aufgaben 4.3.7

1. **Dateien sortieren**

   Schreiben Sie eine Funktion *SortiereKBDatei*, die eine Datei von Datensätzen des Datentyps *Kontobewegung* in einen Container einliest (z.B. in ein Array oder in einen *vector* der Standardbibliothek), diesen sortiert und die sortierten Daten dann wieder in eine Datei schreibt. Verwenden Sie zum Sortieren einen für Kontobewegungen überladenen Operator <, der die Datensätze in Abhängigkeit vom Wert einer Variablen *Sortierbegriff*

```
enum TSortierbegriff {sbKontoNr, sbName, sbDatum,
                                         sbKontonrUndDatum} Sortierbegriff=sbKontoNr;
```

folgendermaßen sortiert:

a) *sbKontoNr*: nach der Kontonummer
b) *sbName*: nach dem Namen des Kontoinhabers
c) *sbDatum*: nach dem Datum
d) *sbKontonrUndDatum*: nach der Kontonummer, und wenn diese gleich ist, nach dem Datum.

2. **Mischen mit Folgeprüfung**

   Wenn man die Funktion *Mischen* auf Dateien anwendet, die nicht sortiert sind, wird die gemischte Datei auch nicht sortiert sein. Erweitern Sie diese Funktion deswegen so, dass die Sortierfolge der Mischdatei beim Schreiben eines neuen Satzes dadurch geprüft wird, dass dieser neue Satz mit dem zuletzt in die Mischdatei geschriebenen Satz verglichen wird.

   Da beim Schreiben des ersten Satzes kein zuletzt in die Mischdatei geschriebener Satz zum Vergleich auf die Sortierfolge zur Verfügung steht, soll in diesem Fall eine Prüfung der Sortierfolge unterbleiben. Die Anzahl der Sortierfehler soll mitgezählt und am Schluss am Bildschirm ausgegeben werden.

3. **Gruppenwechsel**

   Schreiben Sie eine Funktion *GW1*, die eine nach der Kontonummer sortierte Datei von Kontobewegungen wie im Beispielausdruck zum Gruppenwechsel der Stufe 1 ausdruckt. Sie soll die folgenden Anforderungen erfüllen:

   1. Jede Seite soll mit einem Blattkopf beginnen (wie in *Listenausdruck*).
   2. Eine Seite bietet Platz für 72 Druckzeilen, von denen nicht mehr als 60 mit Kontobewegungen bedruckt werden sollen. Ab der 60sten Zeile sollen nur Summenzeilen gedruckt werden.
   3. Auf einer neuen Seite dürfen unmittelbar nach dem Blattkopf keine Summenzeilen gedruckt werden.

4. Aufeinander folgende Kontobewegungen mit derselben Kontonummer sollen als Gruppe betrachtet werden. Am Anfang einer Gruppe sollen die Kontonummer und der Name des Kontoinhabers in einer eigenen Zeile gedruckt werden.
5. Alle Datensätze in derselben Gruppe sollen ohne die Kontonummer gedruckt werden, außer wenn es der erste Satz auf einer neuen Seite ist. In diesem Fall soll auch die Kontonummer gedruckt werden.
6. Am Ende einer Gruppe soll der Saldo der Beträge dieser Gruppe ausgedruckt werden.
7. Nach der letzten Kontobewegung soll der Gesamtsaldo aller Kontobewegungen ausgedruckt werden.

Damit (wie in 5. gefordert) die Daten einer Kontobewegung sowohl mit als auch ohne die Kontonummer ausgedruckt werden können, kann die zu verarbeitende Kontobewegung einer Variablen zugewiesen werden, die zum Drucken aufbereitet wird. In diesem „Drucksatz“ wird z.B. die Kontonummer auf 0 gesetzt, wenn sie nicht gedruckt werden soll. Der Drucksatz wird dann durch eine Funktion ausgedruckt, die z.B. *Drucke_Zeile* heißt. In dieser Funktion wird die Kontonummer des Drucksatzes nur gedruckt, wenn sie von 0 verschieden ist.

4.3.8 C-Funktionen zur Dateibearbeitung Θ

4.4 Assoziative Container

4.4.1 Die Container *set* und *multiset*

4.4.2 Die Container *map* und *multimap*

4.4.3 Iteratoren der assoziativen Container

Aufgaben 4.4

1. Ein **Informationssystem** soll zu einem eindeutigen Schlüsselbegriff eine zugehörige Information finden, z.B. zu einer Artikelnummer den zugehörigen Preis.

   Schreiben Sie als **einfachen Prototyp** für ein solches System eine Funktion

   ```
   bool ValueToKey(KeyType key,ValueType& value)
   ```

   deren Rückgabewert *true* ist, wenn zum Argument für *key* ein passender Wert gefunden wurde. Der gefundene Wert soll dann als Argument für *value* zurückgegeben werden. Falls kein passender Wert gefunden wird, soll der Rückgabewert *false* sein. Verwenden Sie dazu einen geeigneten Container.

   Testen Sie diese Funktion. Damit man leicht sieht, ob der gesuchte Begriff auch tatsächlich gefunden wurde, sollen der Schlüsselbegriff und die Daten identisch sein. Am einfachsten wählt man 1000 bzw. 100 000 **aufeinander folgende** Werte des Datentyps *int*. Um welchen Faktor dauert die Suche in einem Container mit 1000 000 Elementen etwa länger als die in einem Container mit 1000 Elementen?

2. Beim wiederholten Aufruf eines Zufallszahlengenerators wie *rand* oder *random* kann es vorkommen, dass sich die erzeugten Zufallszahlen wiederholen. Für manche Anwendungen braucht man allerdings **Zufallszahlen, die sich nicht wiederholen**.

   Schreiben Sie eine Funktion *NewRand*, die bei jedem Aufruf eine neue Zufallszahl liefert. Die bisher erzeugten Zufallszahlen sollen in einem geeigneten Container abgelegt werden.

3. Schreiben Sie eine **Rechtschreibprüfung**. Dabei soll zuerst ein Wörterbuch aus einer Textdatei erstellt werden, indem diese Datei zeilenweise eingelesen und alle Wörter daraus mit einer Funktion wie *tokenize* (siehe Aufgabe 4.2.3, 4.) bestimmt werden. Diese Wörter sollen dann in einem geeigneten Container abgelegt werden.

Anschließend soll die zu prüfende Datei als Textdatei zeilenweise eingelesen werden. Auch hier sollen die einzelnen Wörter mit einer Funktion wie *tokenize* bestimmt werden. Für jedes Wort soll dann geprüft werden, ob es in dem Container mit den Wörtern aus dem Wörterbuch enthalten ist.

Testen Sie diese Funktion und insbesondere auch ihr Zeitverhalten, indem Sie aus einer relativ großen Textdatei eine Kopie erstellen und in dieser Datei dann einzelne Wörter verändern. Verwenden Sie die ursprüngliche Datei als Wörterbuch.

4. Mit dem assoziativen Container *multimap* kann man leicht eine **Konkordanzliste** aus einem Text erstellen. Eine solche Liste ist ein alphabetisch geordnetes Verzeichnis aller Wörter aus einem Text, die zu jedem Wort die Nummer einer jeden Seite bzw. Zeile enthält, in der es vorkommt. Wenn man z.B. jedes Wort aus dem folgenden Text zusammen mit seiner Zeilennummer als Paar in einen solchen Container einträgt

```
"Alle meine Entchen"
"schwimmen auf dem See,"
"schwimmen auf dem See,"
```

und dann alle diese Worte zusammen mit den zugehörigen Nummern ausgibt, erhält man diese Konkordanzliste:

```
Alle  1
Entchen  1
See  2  3
auf  2  3
dem  2  3
meine  1
schwimmen  2  3
```

Eine Konkordanzliste aus dem Quelltext eines Programms bezeichnet man auch als **Cross-Reference-Liste**. Mit einer solchen Liste kann man feststellen, in welchen Zeilen eines Programms welche Namen (Variablen usw.) verwendet werden.

Schreiben Sie eine Funktion ***MakeXRef***, die jeden String aus einem *vector* mit Strings mit der Funktion *tokenize* (siehe Aufgabe 4.2.3, 4.) in Worte zerlegt und jedes solche Wort zusammen mit seiner Zeilennummer in eine geeignete Variable des Typs *multimap* ablegt. Eine Funktion ***PrintXRef*** soll jedes Wort aus dem mit *MakeXRef* angelegten *multimap* ausgeben sowie zu jedem solchen Wort die zugehörigen Zeilennummern.

Testen Sie diese Funktionen mit den Strings von oben. Eine zweite Variante der Funktion ***MakeXRef*** soll alle Zeilen einer Textdatei einlesen und zerlegen.

5. **Mit *multimaps* in einer Schlüsseltabelle suchen und eine Datei sortieren**

Wenn man als Schlüsselwerte eines Containers der Klasse *multimap* die Schlüsselwerte von Datensätzen einer Datei nimmt und als zugehörige Werte die Dateiposition des jeweiligen Datensatzes, kann man den Datensatz zu einem Schlüsselwert in einer Datei schnell finden.

a) Schreiben Sie eine Funktion *MMReadKeys*, die aus einer Datei von Kontobewegungen die Kontonummer und die Position eines Datensatzes in einen Container des Datentyps *multimap* einliest.

b) Eine Funktion *MMFind* soll in diesem Multimap-Container nach einer Kontobewegung mit einer bestimmten Kontonummer suchen.

c) Wenn man die Datensätze einer Datei im Direktzugriff lesen und schreiben kann ist es zwar technisch möglich, diese so zu vertauschen, dass sie anschließend sortiert sind. Da jedoch die Lese- und Schreibzugriffe relativ zeitaufwendig sind, kann man auf diese Weise keine optimalen Ausführungszeiten für das Sortieren einer Datei erwarten.

   Schreibt man die Datensätze dagegen in der Reihenfolge in eine neue Datei, die sich aus dem in a) erzeugten Multimap-Container ergibt, geht das schneller. Die Dateiposition zu einem Datensatz mit einem Schlüsselwert ergibt sich dann aus dem zweiten Wert eines Wertepaares.

   Schreiben Sie eine Funktion *MMSort*, die auf diese Weise aus einer nicht sortierten Datei von Kontobewegungen eine nach der Kontonummer sortierte Datei erzeugt.

4.5 Die numerischen Klassen der Standardbibliothek

4.5.1 Komplexe Zahlen Θ

Aufgaben 4.5.1

1. Die **quadratische Gleichung**

   $ax^2+bx+c=0$

   hat die beiden Lösungen

   $x = (-b \pm sqrt(b^2 - 4ac))/2a$

   Schreiben Sie ein Programm, das die Lösung dieser Gleichung zu den komplexen Koeffizienten a=2+3i, b=4–5i und c=–7+8i ausgibt. Außer der Lösung soll auch noch die Probe angezeigt werden.

2. Die Gleichung vom Grad n (n >= 1, ganzzahlig)

   $x^n-1=0$

   hat n komplexe Lösungen, die mit

   $w=\cos(2\pi/n) + i*\sin(2\pi/n)$

   durch $x_0=1$, $x_1=w$, $x_2=w^2$, ..., $x_{n-1}=w^{n-1}$ gegeben sind. Geben Sie (z.B. für n=10) für jede dieser **Einheitswurzeln** x_i^n in einem Memo aus.

4.5.2 Valarrays Θ

Aufgabe 4.5.2

Bei manchen Experimenten (Physik, Psychologie usw.) besteht ein linearer Zusammenhang der Art

$y = a*x + b$

zwischen einer unabhängigen Variablen x und einer abhängigen Variablen y. Allerdings sind die Werte von a und b oft nicht bekannt. Man versucht sie deswegen zu schätzen, indem man das Experiment mit n verschiedenen Werten von x_0, x_1, ..., x_{n-1} wiederholt und dabei die Werte y_0, y_1, ..., y_{n-1} für y ermittelt. Falls die Messwerte für y durch Störungen und/oder Messfehler verfälscht werden, kann man jedoch nicht erwarten, dass die Punkte (x_0, y_0), (x_1, y_1), ..., (x_{n-1}, y_{n-1}) alle auf einer Geraden liegen.

Zur Lösung dieses Problems hat der Mathematiker Gauß vorgeschlagen, die Werte für a und b so zu bestimmen, dass das Quadrat der Abweichungen

$F(a,b)= (y_0 - (ax_0+b))^2 + (y_1 - (ax_1+b))^2 + ... + (y_{n-1} - (ax_{n-1}+b))^2$

möglichst klein wird (**Methode der kleinsten Quadrate**). Die so ermittelte Gerade wird auch als **Regressionsgerade** bezeichnet. Mit

$s_{xy} = x_0y_0 + x_1y_1 + ... + x_{n-1}y_{n-1}$
$s_x = x_0 + x_1 + ... + x_{n-1}$
$s_y = y_0 + y_1 + ... + y_{n-1}$
$s_{xx} = x_0x_0 + x_1x_1 + ... + x_{n-1}x_{n-1}$
$x_M = s_x/n$
$y_M = s_y/n$

führt dieser Ansatz auf die folgenden Werte für a und b:

$a = (n*s_{xy} - s_x*s_y)/(n*s_{xx} - s_x*s_x)$
$b = y_M - a*x_M$

1. Formulieren Sie diese Gleichungen

   a) mit Valarrays und den zugehörigen Funktionen (*sum* usw.)
   b) mit Schleifen und dem Indexoperator wie mit normalen Arrays.

2. Testen Sie die Lösungen für a und b mit n (2, 100, 10 000) Messwerten, die

   a) auf einer Geraden y = ax + b liegen. Die berechneten Werte müssen dann die ursprünglichen Werte für a und b ergeben.
   b) um kleine Zufallswerte von der Geraden y = ax + b abweichen. Die berechneten Werte müssen dann in der Nähe der Werte für a und b liegen.

4.6 C++0x-Erweiterungen der Standardbibliothek Θ

4.6.1 Ungeordnete Assoziative Container (Hash Container)

4.6.2 Die Installation der Boost-Bibliotheken Θ

4.6.3 Fixed Size Array Container Θ

4.6.4 Tupel Θ

5 Funktionen

5.1 Die Verwaltung von Funktionsaufrufen über den Stack

5.1.1 Aufrufkonventionen Θ

5.2 Funktionszeiger und der Datentyp einer Funktion

5.2.1 Der Datentyp einer Funktion

5.2.2 Zeiger auf Funktionen

Aufgaben 5.2

Weitere Aufgaben zu Funktionszeigern finden sich in Abschnitt 10.13

1. Beim **Newton-Verfahren** zur Bestimmung einer Nullstelle der Funktion $f:R \rightarrow R$ ersetzt man einen Näherungswert x_0 für die Lösung durch die Nullstelle der Tangente im Punkt $(x_0,f(x_0))$:

   $$y = f'(x_0)(x-x_0) + f(x_0) = 0$$

   Damit wird x_0 durch den folgenden Wert ersetzt:

   $$x = x_0 - f(x_0)/f'(x_0)$$

   Diese Schritte werden so lange wiederholt, bis man einen genügend guten Näherungswert für die Lösung hat oder bis eine maximale Anzahl von Iterationen durchgeführt ist. Wenn das Newton-Verfahren konvergiert, verdoppelt sich ab einer genügend guten Näherung die Anzahl der richtigen Stellen bei jeder Iteration.

   Implementieren Sie das Newton-Verfahren unter Verwendung von Funktionszeigern in den zwei Varianten a) und b). Der erste Näherungswert soll als Parameter übergeben werden. Die Iterationen sollen abgebrochen werden, wenn zwei aufeinanderfolgende Werte genügend nahe beieinander liegen. Sie können das mit der Funktion *NearlyEqual* von Abschnitt 3.6.6 prüfen.

   a) Die Ableitung wird durch einen Näherungswert ersetzt.
   b) Die Ableitung wird ebenfalls als Funktion übergeben.
   c) Testen Sie die beiden Verfahren mit einigen Funktionen (z.B. mit Polynomen) und prüfen Sie das Ergebnis, indem Sie die gefundene Nullstelle in die Funktion einsetzen. Das Ergebnis sollte dann 0 sein.

2. Um einen Näherungswert für das Integral einer Funktion $f:R\rightarrow R$ im Intervall von a bis b zu finden, kann man folgendermaßen vorgehen:

– Man ersetzt die Funktion f durch eine Gerade durch die Punkte (a,f(a)) und (b,f(b)) und berechnet die Fläche des so erhaltenen Trapezes **(Trapezregel)**.

  Zur Erhöhung der Genauigkeit kann man das Intervall [a,b] in n–1 Teilintervalle [a,a+h], [a+h,a+2h] ... usw. mit h = (b–a)/n zerlegen. Summiert man die Trapezflächen der Teilintervalle auf, erhält man die **Trapezsumme** (siehe auch Aufgabe 3.6.5, 5.):

  $T_n = h*[f(a)/2 + f(a+h) + ... + f(b–h) + f(b)/2]$

– Man ersetzt die Funktion f durch ein quadratisches Polynom durch die Punkte (a,f(a)), ((a+b)/2,f((a+b)/2)) und (b,f(b)) **(Simpson-Regel)**. Zerlegt man das Intervall wie bei der Trapezregel in Teilintervalle und summiert man diese Flächen auf, erhält man die Summe

  $S_n = h*[f(a)+4f(a+h)+2f(a+2h)+4f(a+3h)+...+2f(b–2h)+4f(b–h)+f(b)]/3$

a) Schreiben Sie die Funktionen *Trapezsumme* und *Simpsonsumme*. Dabei sollen a, b, n und f als Parameter übergeben werden.

b) Da man oft nicht entscheiden kann, wie gut die so berechneten Näherungswerte sind, erhöht man n sukzessive (z.B. durch Verdoppeln), bis sich zwei aufeinander folgende Näherungswerte um weniger als eine vorgegebene Schranke unterscheiden. Zum Vergleich von zwei Näherungswerten kann die Funktion *NearlyEqual* von Abschnitt 3.6.6 verwendet werden.

  Schreiben Sie eine Funktion *iterate*, die diese Iterationen für die Funktionen *Trapezsumme* und *Simpsonsumme* durchführen. Beide sollen als Parameter übergeben werden.

c) Testen Sie diese Funktionen, indem Sie die so erhaltenen Näherungswerte mit den exakten Werten vergleichen, die man über die Stammfunktion erhält (z.B. für Polynome).

3. In je einem Array sollen Funktionen dargestellt werden, die denselben Datentyp haben wie die Funktionen

   i) void f1(void){};
   ii) int f2(int,int){};
   iii) double f3(double){};
   iv) char* f4(char*){};

   a) Definieren Sie für jeden dieser Funktionstypen ein solches Array mit 10 Elementen. Verwenden Sie dazu

      a1) mit *typedef* deklarierte Namen für die Funktionstypen.
      a2) keine mit *typedef* deklarierte Namen für die Funktionstypen.

   b) Weisen Sie die 4 Funktionen f1, f2 f3 und f4 jeweils dem ersten Element der unter a1) und a2) definierten Arrays zu und rufen Sie diese Funktionen über die Arrayelemente auf. Überprüfen Sie mit dem Debugger, ob jedes Mal die richtige Funktion aufgerufen wird.

   c) Geben Sie die Namen der Datentypen der Funktionen f1, f2, f3 und f4 mit der *typeid* Elementfunktion *name* in einem Memo aus. Damit Sie *typeid* verwenden können, müssen Sie die Header-Datei <typeinfo> mit einer *#include*-Anweisung in das Programm einbinden.

5.3 Rekursion

5.3.1 Grundlagen

Aufgaben 5.3.1

1. Schreiben Sie die folgenden Funktionen als rekursive Funktionen. Alle diese Aufgaben sind lediglich Übungen zur Formulierung rekursiver Funktionen. Keine der so erhaltenen Lösungen ist bezüglich der Effizienz mit der iterativen Lösung vergleichbar.

   a) Fakultät (siehe Aufgabe 3.6.5, 3.).

b) Fibonacci-Zahlen (siehe Aufgabe 3.4.6, 2.).
c) Die Funktion *ggT* soll den größten gemeinsamen Teiler von zwei Werten als Funktionswert zurückgeben (siehe Aufgabe 3.7.4, 3.).
d) Schreiben Sie eine Funktion, die die Ergebnisse der Funktionen von a) bis c) mit denen von entsprechenden iterativen Funktionen vergleicht.

2. Die Ackermann-Funktion

```
                 ⎧  m+1 für n=0
ack(n,m) =       ⎨  ack(n-1,1) für m=0
                 ⎩  ack(n-1,ack(n,m-1)) sonst
```

setzt mit dem Index n (ab n=1) in gewisser Weise die „Folge“ Addition, Multiplikation, Potenzierung fort. Für die ersten Werte von n gilt:

ack(0,m) = m + 1
ack(1,m) = m + 2
ack(2,m) = 2*m + 3
ack(3,m) = $2^{m+3} - 3$

Definieren Sie die rekursive Funktion *ack* und vergleichen Sie ihre Werte bis n=3 und m=10 mit den expliziten Formeln. Wegen

ack(4,m) = ack(3,ack(4,m–1)) = 2^(ack(4,m–1)+3) – 3 // $2^{ack(4,m-1)+3} - 3$

= 2^{2^(ack(4,m–2)+3) – 3} + 3) – 3

ergibt sich für ack(4,m) ein Wert in der Größenordnung

$$ack(4,m) = 2^{2^{2^{\cdot^{\cdot^{\cdot}}}}}$$

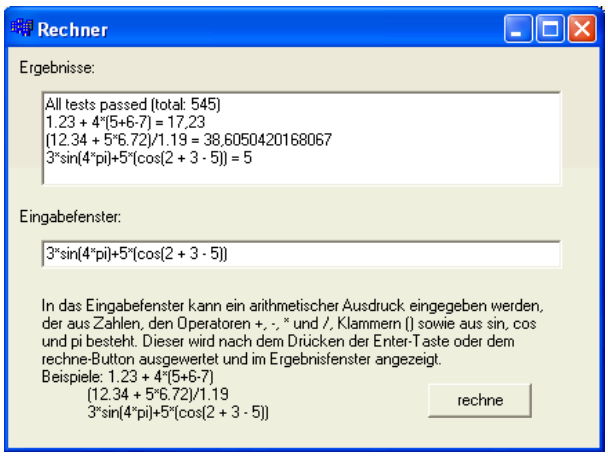
wobei der „Turm der Potenzen“ m+3 Glieder hoch ist. Offensichtlich wächst diese Funktion sehr schnell: Bereits ack(4,2) hat 19729 Dezimalstellen. Mit n>4 erhält man ein noch schnelleres Wachstum.

5.3.2 Quicksort

5.3.3 Ein rekursiv absteigender Parser

Aufgaben 5.3.3

Legen Sie ein Projekt (z.B. mit dem Namen *CalcEx*) an, dessen Formular etwa folgendermaßen aussieht.

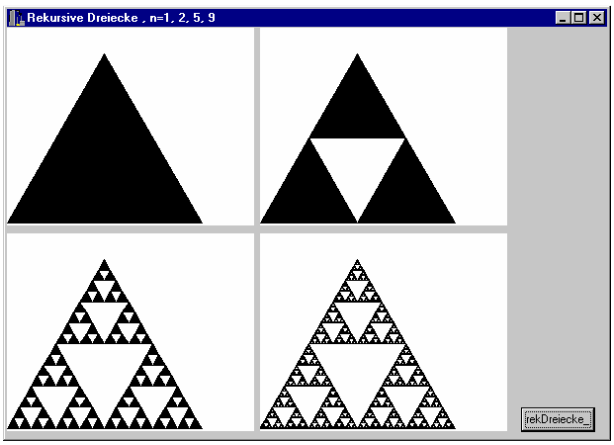


a) Implementieren Sie dazu die in diesem Abschnitt vorgestellt Funktion *expression* und rufen Sie diese nach dem Anklicken des Buttons mit dem String im Eingabefeld auf. Zeigen Sie den Eingabestring zusammen mit dem Ergebnis im Ergebnisfenster an.
b) Entwerfen Sie einige Tests, die zumindest die Grundfunktionalität der Funktion *expression* testen. Führen Sie diese Tests mit automatischen Testfunktionen (z.B. wie in Abschnitt 3.5.2) durch.
c) Erweitern Sie die Funktion *expression* so, dass der auszuwertende String auch Funktionen wie *sin*, *cos* usw. enthalten kann. Die Argumente dieser Funktionen sollen wiederum ein zulässiger Ausdruck im Rahmen der implementierten Syntaxregeln sein können.

5.3.4 Rekursiv definierte Kurven Θ

Aufgaben 5.3.4

1. Die folgenden Figuren entstehen rekursiv aus einem gleichseitigen Dreieck. Falls die Abbruchbedingung nicht erreicht ist, wird das Dreieck in 4 weitere Teildreiecke unterteilt, von denen die drei äußeren gezeichnet werden (ähnlich wie die Koch'schen Kurven).



Schreiben Sie ein Programm, das diese Dreiecke zeichnet. Zum Zeichnen eines Dreiecks kann die Funktion *Polygon* des Canvas verwendet werden.

5.3.5 Indirekte Rekursion Θ

5.3.6 Rekursive Datenstrukturen und binäre Suchbäume

Aufgaben 5.3.6

1. Ergänzen Sie Ihre Lösung der Aufgabe 3.12.12 um die folgenden rekursiven Funktionen und rufen Sie diese Funktionen beim Anklicken eines Buttons auf:

   a) Die Funktion *insertBinTreenode_rec* soll wie *insertBinTreenode* (Aufgabe 3.12.12, 1. b) einen Knoten mit den als Argument übergebenen Werten für *key* und *data* in einen Binärbaum einhängen.
   b) Die Funktion *traverseTree_rec* soll für alle Knoten im Baum die Werte für *key* und *data* ausgeben.
   c) Die Funktion *searchBinTree_rec* soll wie *searchBinTree* (Aufgabe 3.12.12, 1. c) einen Zeiger auf einen Knoten mit dem als Argument übergebenen Schlüsselbegriff zurückgeben, wenn ein solcher Knoten gefunden wird, und andernfalls den Wert 0.

2. Entwerfen Sie eine Baumstruktur, die als Schlüsselwert einen String enthält, und als zugehörige Daten Zeiger auf eine verkettete Liste, deren Daten ebenfalls Strings sind.

   Erstellen Sie mit diesen Datenstrukturen eine Liste mit allen verschiedenen Wörtern aus einem Text, die zu jedem Wort alle Zeilennummern enthält, in der das Wort vorkommt (**Konkordanzliste**). Diese Wörter sollen dann alphabetisch sortiert ausgegeben werden, wobei auf jedes Wort die Zeilennummern folgen sollen.

Dazu können Sie folgendermaßen vorgehen: Lesen Sie die Textdatei mit der globalen Funktion *getline* (siehe Abschnitt 4.3.4) zeilenweise ein. Zerlegen Sie jede Zeile mit der Funktion *tokenize* von Aufgabe 4.2.3, 4. in einzelne Wörter. Tragen Sie dann jedes so gefundene Wort als Schlüsselwert in den binären Suchbaum ein und zu jedem Wort die Zeilennummern in der verketteten Liste.

In Aufgabe 4.4, 4. wurde eine solche Konkordanzliste mit dem Container *multimap* gelöst. Vergleichen Sie den Aufwand für diese beiden Lösungen.

5.3.7 Verzeichnisse rekursiv nach Dateien durchsuchen Θ

Aufgaben 5.3.7

1. Die Funktionen in a) und b) sollen Namen von Dateien aus einem als Parameter übergebenen Laufwerk bzw. Verzeichnis einschließlich aller Unterverzeichnisse anzeigen. Die Anzahl der gefundenen Dateien soll als Funktionswert zurückgegeben werden. Verwenden Sie zur Anzeige ein RichEdit-Fenster. Ein Memo ist zu klein, wenn das Laufwerk viele Dateien enthält.

   a) Eine Funktion *searchSubdirs_all* soll alle Dateien (Maske „*“) des als Parameter übergebenen Laufwerks oder Verzeichnisses ausgeben.
   Zum Testen können Sie die Anzahl der gefundenen Dateien mit der im Kontextmenü des Windows-Explorers unter Eigenschaften angezeigten Anzahl vergleichen. Am einfachsten geben Sie den Namen des Array hart kodiert im Quelltext an.

   b) Eine Funktion *searchSubdirs* soll alle Dateien des als Parameter übergebenen Laufwerks oder Verzeichnisses ausgeben, deren Name einer ebenfalls als Parameter übergebenen Maske (z.B. „*.cpp“) entspricht.
   Zum Testen können Sie die Anzahl der gefundenen Dateien mit der Anzahl vergleichen, die nach einem entsprechenden *dir*-Befehl wie z.B.

   ```
   dir \*.cpp /s
   ```

   ein einer Eingabeaufforderung (*Start|Programme|Zubehör*) angezeigt wird. Am einfachsten geben Sie den Namen des Arrays hart kodiert im Quelltext an.

   c) Zeigen Sie mit der Funktion *searchSubdirs* alle Dateinamen eines Laufwerks an, die einer in einer Edit-Komponente eingegebenen Maske entsprechen. Das Laufwerk soll über eine DriveComboBox (Tool-Palette, Kategorie Win.3.1) ausgewählt werden.

   In Aufgabe 10.4 wird die Funktion *searchSubdirs* so überarbeitet, dass alle Verzeichnisse eines Laufwerks wie im Windows Explorer in einem TreeView angezeigt werden, und nach dem Anklicken eines solchen Verzeichnisses alle Dateien des Verzeichnisses in einem ListView.

2. Schreiben Sie ein Programm (z.B. mit dem Namen ***DupFiles***), das alle **doppelten Dateien** auf einer oder mehreren Festplatten findet.

   Dabei stellt sich die Frage, wann zwei Dateien als gleich betrachtet werden sollen. Der Name ist dafür nicht unbedingt als Kriterium geeignet: Zwei Dateien mit demselben Namen können verschiedene Inhalte haben und zwei Dateien mit verschiedenen Namen denselben.

   Am sichersten wäre es, wenn man alle Dateien mit derselben Größe zeichenweise vergleichen würde. Das wäre allerdings sehr zeitaufwendig. Ein pragmatischer Kompromiss ist ein Schlüssel, bei dem die Dateigröße und der Dateiname zu einem einzigen String zusammengesetzt werden (z.B. „325config.sys“, wenn die Datei „config.sys“ 325 Bytes groß ist). Mit diesem Schlüssel werden nur diejenigen Dateien als gleich betrachtet, die denselben Namen und dieselbe Dateigröße haben.

   a) Lösen Sie diese Aufgabe mit einem binären Suchbaum, der in jedem Knoten den Schlüssel (aus der Dateigröße und dem Namen) einer Datei enthält. Jeder Baumknoten soll außerdem zwei Zeiger *first* und *last* enthalten, die auf das erste und letzte Element einer verketteten Liste zeigen, deren Knoten als Daten den Dateinamen (einschließlich der Pfadangaben) enthalten. Ein solcher Baumknoten enthält also die Elemente:

      – *key*: Der Schlüsselwert (ein String aus dem Dateinamen und der Dateigröße).
      – *first* und *last*: Zeiger auf das erste und letzte Element einer verketteten Liste, deren Knoten als Daten den Dateinamen (einschließlich der Pfadangaben) enthält.

– *left* und *right*: Zeiger auf den linken und rechten Teilbaum. Der linke Teilbaum enthält nur kleiner und der rechte nur größere Schlüsselwerte als der aktuelle Knoten. Alle gleichen Werte werden in die verkettete Liste eingehängt.

b) Lösen Sie diese Aufgabe mit einem *multimap* Container der C++-Standardbibliothek (siehe Abschnitt 4.4).

5.4 Funktionen und Parameter Θ

5.4.1 Seiteneffekte und die Reihenfolge von Auswertungen Θ

5.4.2 Syntaxregeln für Funktionen Θ

5.4.3 Der Funktionsbegriff in der Mathematik und in C++ Θ

5.4.4 Der Aufruf von Funktionen aus Delphi im C++Builder Θ

5.4.5 Unspezifizierte Anzahl und Typen von Argumenten Θ

5.4.6 Die Funktionen *main* bzw. *WinMain* und ihre Parameter Θ

5.4.7 Traditionelle K&R-Funktionsdefinitionen Θ

Aufgabe 5.4

1. Die folgenden Anweisungen wurden ohne Fehler übersetzt. Nach dem Aufruf der Funktion f hat die Variable a allerdings immer noch denselben Wert wie vorher. Wieso?

```
int a=0;
void Init1()
{
a=1; // globale Variable
}

void f()
{
Init1;
}
```

2. Bestimmen Sie für die Funktionen

```
int f(int& x)        int g(int& x)
{                    {
x++;                 x=2*x;
return x;            return x;
}                    }
```

das Ergebnis der Ausdrücke y1 und y2:

```
int x = 0;
int y1=f(x)+g(x);
x = 0;
int y2=g(x)+f(x);
```

5.5 Default-Argumente

Aufgaben 5.5

Durch eine Funktion *InitPunkt* soll eine Variable des Datentyps

```
struct Punkt {
  int x, y;
};
```

initialisiert werden. Falls diese Funktion ohne Argumente aufgerufen wird, sollen beide Koordinaten auf 0 gesetzt werden. Bei einem Aufruf mit einem Argument soll x den Wert des Arguments erhalten und y den Wert 0. Beim Aufruf mit zwei Argumenten soll x den Wert des ersten und y den Wert des zweiten Arguments erhalten.

5.6 Inline-Funktionen

5.7 Überladene Funktionen

5.7.1 Funktionen, die nicht überladen werden können

5.7.2 Regeln für die Auswahl einer passenden Funktion

Aufgaben 5.7

1. In der Datei „include\stddefs.h“ sind die Konstanten TRUE und FALSE so definiert:

   ```
   #ifndef TRUE
   #  define TRUE  1
   #  define FALSE 0
   #endif
   ```

   Welche der beiden Funktionen

   ```
   void f(bool b) { }
   void f(int i)  { }
   ```

   wird durch die folgenden Ausdrücke aufgerufen?

   a) `f(true);`
   b) `f(TRUE);`

2. Default-Argumente können als einfache Möglichkeit zur Definition von überladenen Funktionen betrachtet werden, die sich in der Anzahl der Parameter unterscheiden. Lösen Sie die Aufgabe 5.5 mit überladenen Funktionen.

3. Der math.h-Header der C-Standardbibliothek stellt für jede Funktion wie *sqrt* eine einzige Variante mit *double*-Argumenten

   ```
   double sqrt(double x);
   ```

   zur Verfügung. Der cmath-Header der C++-Standardbibliothek stellt dagegen drei überladene Varianten mit *float*, *double* und *long double* Argumenten und Rückgabewerten wie

   ```
   float sqrt(float x);
   double sqrt(double x);
   long double sqrt (long double x);
   ```

zur Verfügung. Welches Ergebnis haben die folgenden Funktionsaufrufe

```
double x1=sqrt(1);
double x2=sqrt(1.0);
double x3=sqrt(1.0l);
```

a) nach `include <math.h>`
b) nach `include <cmath>`
`using namespace std;`

4. Geben Sie jeweils zwei verschiedene überladene Funktionen an, deren Aufruf

   1. mit einer Variablen
   2. mit einer Konstanten

   a) eines selbstdefinierten Datentyps eindeutig ist
   b) eines selbstdefinierten Datentyps mehrdeutig ist
   c) des Datentyps *long* eindeutig ist
   d) des Datentyps *long* mehrdeutig ist
   e) des Datentyps *int* eindeutig ist
   f) des Datentyps *int* mehrdeutig ist

5.8 Überladene Operatoren mit globalen Operatorfunktionen

5.8.1 Globale Operatorfunktionen

5.8.2 Die Inkrement- und Dekrementoperatoren

Aufgaben 5.8.2

1. Auf den ersten Blick erscheint es vielleicht als naheliegend, mit dem Operator ^ Potenzen zu realisieren, so dass x^n für x^n steht. Welcher Wert würde sich dabei für den folgenden Ausdruck ergeben?

   x^n – 1

2. Eine rationale Zahl (ein Bruch im Sinne der Bruchrechnung) besteht aus einem ganzzahligen Zähler und einem Nenner und kann deshalb durch die folgende Klasse dargestellt werden:

```
struct Bruch {
  int z,n; // z: Zähler, n: Nenner
};
```

   Die Bruchrechnung ist auf Rechnern nicht sehr verbreitet, da die Rechnungen leicht zu einem Überlauf führen. Sie ist aber ein einfaches Beispiel dafür, wie man überladene Operatoren definieren kann.

   Zur Vereinfachung werden Brüche im Folgenden durch Wertepaare wie (1,2) dargestellt und nicht mit einem Bruchstrich wie in ½.

   a) Zwei Brüche (pz,pn) und (qz,qn) sind gleich, wenn pz*qn==pn*qz gilt. Definieren Sie die Operatoren „==“ und „!=“ für Operanden des Datentyps *Bruch*. Stellen Sie sicher, dass zwischen diesen Operatoren die üblichen Beziehungen gelten.

   b) Der Bruch (pz,pn) ist kleiner als (qz,qn), wenn pz*qn<pn*qz gilt. Definieren Sie die Operatoren „<“, „>=“, „<=“ und „>“ für Operanden des Datentyps *Bruch*. Stellen Sie sicher, dass zwischen diesen Operatoren die üblichen Beziehungen gelten.

   c) Definieren Sie Operatorfunktionen für die folgenden Operationen.

(pz,pn) + (qz,qn) = (pz*qn + pn*qz, pn*qn)
(pz,pn) – (qz,qn) = (pz*qn – pn*qz, pn*qn)
(pz,pn) * (qz,qn) = (pz*qz,pn*qn)
(pz,pn) / (qz,qn) = (pz*qn,pn*qz)

Bei diesen Operationen sollen der Zähler und Nenner gekürzt werden, indem man beide durch den größten gemeinsamen Teiler dividiert. Dazu kann die Funktion *ggT* (siehe auch Aufgabe 3.7.4, 3.) verwendet werden:

```
int ggT(int a, int b)
{
int x=a;
int y=b;
while (y != 0)
  {
    int r = x%y;
    x = y;
    y = r;
  }
return x; // ggT(a,b)==x;
}
```

d) Sie können diese Operatoren testen, indem Sie die Werte vergleichen, die sich bei der geometrischen Reihe

$1 + p + p^2 + ... p^N = (p^{N+1} - 1)/(p-1)$ // p=z/n

durch Aufsummieren (linke Seite) und mit der Summenformel (rechte Seite) ergeben. Sie können diese Werte außerdem mit dem Bruch vergleichen, den man beim Einsetzen von z/n für p in die Summenformel erhält.

5.8.3 Referenzen als Funktionswerte

5.8.4 Die Ein- und Ausgabe von selbst definierten Datentypen

Aufgaben 5.8.4

1. Definieren Sie für die Klasse *Bruch* aus Aufgabe 5.8.2 überladene Operatoren << und >>, so dass ein Bruch sowohl aus einem *istream* eingelesen als auch über einen *ostream* (z.B. *cin* oder *cout*) ausgegeben werden kann.

a) Verwenden Sie diese Operatoren, um Brüche in eine Datei zu schreiben bzw. aus einer Datei zu lesen. Eine Datei kann zum Schreiben über eine Variable der Klasse *ofstream* und zum Lesen über eine Variable der Klasse *ifstream* angesprochen werden. Diese Klassen stehen zur Verfügung nach

```
#include <fstream>
using namespace std;
```

b) Verwenden Sie diese Operatoren in zwei Funktionen *BruchToStr* und *StrToBruch*, die einen *Bruch* über einen *ostringstream* in einen *string* der Standardbibliothek von C++ umwandeln bzw. über einen *istringstream* einen Bruch aus einem *string* einlesen.

6 Objektorientierte Programmierung

6.1 Klassen

6.1.1 Datenelemente und Elementfunktionen

6.1.2 Der Gültigkeitsbereich von Klassenelementen

6.1.3 Datenkapselung: Die Zugriffsrechte *private* und *public*

6.1.4 Der Aufruf von Elementfunktionen und der *this*-Zeiger

6.1.5 Konstruktoren und Destruktoren

Aufgaben 6.1.5

1. Beschreiben Sie die Aufgaben eines Konstruktors und Destruktors. Werden diese Aufgaben in der folgenden Klasse erfüllt?

```
class C {
   int n, max;
   int* a;
 public:
  C()
  {
   max=100;
   a=new int[max];
  }

  C(int i)
  {
   n=1;
   a=new int[100];
   a[0]=i;
  };

  C(int i,int j)
  {
   n=2;
   max=100;
   a=new int[100];
   a[1]=i;
   a[2]=j;
  };
```

```
  void add_data(int d)
  {
  if (n<max-1)
   {
     ++n;
     a[n]=d;
   }
  }

  void show_data()
  {
  for (int i=0; i<n; ++i)
   Form1->Memo1->Lines->Add(IntToStr(a[i]));
  }
};
```

2. Welche der folgenden Klassen benötigen einen Destruktor?

a)
```
class C1 {
  int x,y,z;
 public:
  C1(int x_=0, int y_=0, int z_=0)
  {
  x=x_; y=y_; z=z_;
  }
};
```

b)
```
class C2 {
  int* x;
 public:
  C2(int n)
  {
  x=new int[n];
  }
};
```

c)
```
#include <fstream>
using std::ifstream;

class C3 {
  ifstream f; // eine Klasse der C++-Standardbibliothek
 public:
  C3(const char* FileName)
  {
   f.open(FileName);
  }
};
```

3. Definieren Sie die folgenden Klassen. Jede soll geeignete Konstruktoren sowie bei Bedarf auch einen Destruktor enthalten. Legen Sie mit jedem Konstruktor der unter a) bis c) definierten Klassen zwei Objekte an. Das erste soll durch eine Definition und das zweite mit *new* angelegt werden. Rufen Sie jede Funktion, die Sie geschrieben haben, für jedes dieser Objekte auf.

   a) Die Klasse *Kreis* soll einen Kreis darstellen und als *private* Datenelement den Radius (Datentyp *double*) enthalten. Ein Konstruktor und die Elementfunktion *setzeRadius* sollen einen Parameter des Datentyps *double* haben, der den Radius setzt. Der Konstruktor soll auch ohne ein Argument aufgerufen werden können und dann den Radius auf 1 setzen. Die Elementfunktion *Radius()* soll den Radius als Funktionswert zurückgeben. Alle Elementfunktionen von *Kreis* sollen innerhalb der Klasse definiert werden.

   b) Die Klasse *Quadrat* soll ein Quadrat darstellen und als *private* Datenelement die Seitenlänge (Datentyp *double*) enthalten. Ein Konstruktor und die Elementfunktion *setzeSeitenlaengen* sollen einen Parameter des Datentyps *double* haben, der die Seitenlänge setzt. Der Konstruktor soll auch ohne ein Argument aufgerufen werden können und dann die Seitenlänge auf 1 setzen. Die Elementfunktion *Seitenlaenge()* soll die Seitenlänge als Funktionswert haben. Alle Elementfunktionen sollen außerhalb der Klasse definiert werden.

   c) Die Klasse *Rechteck* soll ein Rechteck darstellen und als *private* Datenelemente die Seitenlängen a und b (Datentyp *double*) enthalten, die durch einen Konstruktor initialisiert werden. Diese sollen auch mit der Funktion *setzeSeitenlaengen* gesetzt werden können, die zwei Parameter des Datentyps *double* hat. Die jeweiligen Seitenlängen sollen als Funktionswert der Funktionen *Seitenlaenge_a*() und *Seitenlaenge_b*() zurückgegeben werden.

d) Ergänzen Sie jede der unter a) bis c) definierten Klassen um die Elementfunktionen *Flaeche()*, *Umfang()* und *toStr()*. Die Funktion *toStr()* soll einen String der Art „Kreis mit Radius 5“ oder „Rechteck mit a=6 und b=7“ zurückgeben.

4. Bei dieser Aufgabe soll nur eine der Klassen (z.B. *Grundstueck*) programmiert werden. Die anderen sollen nur skizziert werden (z.B. grafisch mit Bleistift und Papier). Für eine solche Gesamtheit von Klassen werden wir später bessere Techniken kennen lernen und dann wieder auf diese Aufgabe zurück kommen.

   Ein einfaches Programm zur Verwaltung von Immobilien soll die Klassen *Grundstueck, Eigentumswohnung* usw. enthalten, mit denen Grundstücke usw. dargestellt werden können.

   Jede dieser Klassen soll ein Datenelement *Anschrift* des Datentyps *char** und ein Datenelement *Kaufpreis* des Datentyps *double* enthalten. Zusätzlich zu diesen beiden Feldern sollen diese Klassen auch noch die folgenden Datenelemente enthalten:

   – Die Klasse *Grundstueck* soll das Datenelement *Flaeche* enthalten.
   – Die Klasse *Eigentumswohnung* soll die Datenelemente *Wohnflaeche* und *AnzahlZimmer* enthalten.
   – Die Klasse *Schloss* soll das *int*-Element *AnzahlSchlossgeister* enthalten.
   – Die Klasse *Einfamilienhaus* soll die Datenelemente *Wohnflaeche, Grundstuecksgroesse* und *AnzahlZimmer* enthalten.
   – Die Klasse *Gewerbeobjekt* soll das Datenelement *Nutzflaeche* und *Nutzungsart* (Datentyp *char**, z.B. für „Büro“, „Restaurant“) enthalten.

   Jede Klasse soll Funktionen haben, die ihre Datenelemente zurückgeben.

5. a) Welche Ausgabe erzeugt ein Aufruf der Funktion *test1*?

```
void display(AnsiString s, int i=-1)
{
if (i>=0) s=s+IntToStr(i);
Form1->Memo1->Lines->Add(s);
}

class C{
  int a;
 public:
  C (int a_=0)
  { // Beim Aufruf ohne Argument ein Standard-
    a=a_;                            // konstruktor
    display("Konstruktor: ", a);
  }
  ~C ()
  { display("Destruktor: ",a);  }
};

void f1(C c)
{
display("  in f1(): Werteparameter");
};

void f2(const C& c)
{
display("  in f2(): Referenzparameter");
};

C f3(int i)
{
display("  in f3(): return-Wert");
return C(i);
};

void test1()
{
C x(1);
C* z=new C(2);
display("vor x=C(3)");
x=C(3);
display("vor f1(4)");
f1(4);
display("vor f2(x)");
```

```
f2(x);
display("vor f3(5)");
x=f3(5);
delete z;
display("Ende von test()");
}
```

Vergleichen Sie ihre Vermutungen anschließend mit dem Ergebnis eines Programms, das die Funktion *test1* aufruft.

b) Wenn Speicher mit *new[]* reserviert wird, muss er mit *delete[]* wieder freigegeben werden. Gibt man mit *new* reservierten Speicher mit *delete[]* wieder frei, ist das Ergebnis undefiniert, ebenso, wie wenn man mit *new[]* reservierten Speicher mit *delete* wieder freigibt. Beschreiben Sie zunächst, welche Ausgabe Sie von einem Aufruf der Funktion *test2* erwarten. Vergleichen Sie Ihre Vermutungen dann mit dem Ergebnis eines Programms, das die Funktion *test2* aufruft.

```
void test2()
{
display("vor p1");
C* p1=new C[2];
delete[] p1;

display("vor p2");
C* p2=new C[2];
delete p2;

display("vor p3");
C* p3=new C;
delete[] p3;

display("vor p4");
C* p4=new C(4);
delete[] p4;
display("Ende von test()");
}
```

6. Definieren Sie für die Klasse *MeinString* eine Elementfunktion *c_str*. Sie soll wie bei den Stringklassen *string* bzw. *AnsiString* den Zeiger auf den internen nullterminierten String zurückgeben und damit auch Argumente des Typs *MeinString* bei den Stringfunktionen wie *strcpy* usw. ermöglichen.

7. In der folgenden Klasse soll der Datentyp T ein großer Datentyp sein. Um den Zeitaufwand für die Funktion *data* zu minimieren, gibt diese als Funktionswert eine Referenz und keine Kopie des Elements x zurück. Beurteilen Sie diesen Ansatz.

```
class C {
  T x;
 public:
  C(T x_) { x=x_; }

  T& data() // T data() wäre zu langsam
  {
  return x;
  }
};
```

8. Was halten Sie von dem folgenden Trick, den ich im Internet gefunden habe (http://home.-att.net/~robertdunn/CodeSamples/CheapTricks.html): „Borland, for some unfathomable reason, decided to make member data private in many classes. Here is a rather clever way to get around it:"

```
#define private public
#define protected public
#include <theworld.h>
#undef private
#undef public
```

6.1.6 OO Analyse und Design: Der Entwurf von Klassen

Aufgabe 6.1.6

1. Bei diesen Aufgaben, die starke Vereinfachungen der Realität darstellen, sollen nur die Klassen und ihre Elemente identifiziert werden. Es ist nicht notwendig, sie zu implementieren.

a) Suchen Sie die realen Objekte und Klassen (einschließlich Datenelementen und Elementfunktionen) für ein Zeichenprogramm, mit dem man Kreise, Quadrate und Rechtecke zeichnen, verschieben, löschen, vergrößern und verkleinern kann.
b) Ein Programm mit einer grafischen Benutzeroberfläche soll aus einem Formular bestehen, das Buttons, Eingabefelder und Ausgabefelder enthält.
c) Ein Programm soll eine Waschmaschine steuern, die aus einem Motor mit einem Drehzahlregler, einem Temperaturfühler mit einem Temperaturregler und einem Wasserzu- und Abfluss mit einem Wasserstandsregler besteht. Die Regler sollen die Drehzahl, Temperatur und den Wasserstand durch die Vorgabe eines Sollwerts regeln sowie die aktuellen Werte zurückgeben können. Eine Uhr soll die Zeit seit dem Start des Programms messen.

2. Ist die Klasse *Datum_2* aus Abschnitt 6.1.3 vollständig, wenn sie dazu verwendet werden soll, ein Kalenderdatum darzustellen?

3. Schreiben Sie eine Klasse *C2DPunkt*, die wie in den Beispielen einen zweidimensionalen Punkt darstellt. Diese Klasse soll später in zahlreichen Beispielen und Aufgaben zur Darstellung einer Position verwendet werden. Versuchen Sie, diese Klasse möglichst vollständig zu schreiben, ohne dass diese Anforderungen jetzt schon bekannt sind. Andererseits soll sie auch keine unnötigen Elemente enthalten.

4. Was halten Sie von einem Design-Tool, mit dem man Klassen und ihre Datenelemente definieren kann, und das zu jedem Datenelement (z.B. *int* x) automatisch die beiden *public* Elementfunktionen

```
int getx() { return x; };
void setx(int x_) { x=x_; };
```

erzeugt.

6.1.7 Programmierlogik: Klasseninvarianten und Korrektheit

Aufgabe 6.1.7

1. Entwerfen Sie systematische Tests für die Klasse *C2DPunkt* von Aufgabe 6.1.6, 3.

2. a) Geben Sie Konsistenzbedingungen für die Klassen von Aufgabe 6.1.5, 3. an.
   b) Prüfen Sie, ob die Elementfunktionen Ihrer Lösung von Aufgabe 6.1.6, 1. diese Konsistenzbedingungen erfüllen.
   c) Was halten Sie davon, die Fläche und den Umfang bei diesen Klassen nicht in entsprechenden Elementfunktionen zu berechnen, sondern sie in Datenelementen zu speichern und deren Wert zurückzugeben? Formulieren Sie die Konsistenzbedingungen für diese Variante.
   d) Was halten Sie von einer Funktion *setzeTag*, mit der man den Kalendertag in der Klasse *Datum* auf einen als Argument übergebenen Tag setzen kann?

   Geben Sie Konsistenzbedingungen für Ihre Klasse *Grundstueck* (Aufgabe 6.1.5, 4.) an, wenn
   e) die Anschrift den Datentyp *char** hat.
   f) der Datentyp der Anschrift eine Stringklasse (*string* oder *AnsiString*) ist.

6.1.8 UML-Diagramme mit Together im C++Builder 2007

6.2 Klassen als Datentypen

6.2.1 Der Standardkonstruktor

6.2.2 Objekte als Klassenelemente und Elementinitialisierer

Aufgaben 6.2.2

1. Die Klasse E soll in einem Standardkonstruktor die Meldung „Standardkonstruktor“ und in einem Konstruktor mit einem *int*-Parameter die Meldung „int-Konstruktor“ ausgeben.

```
class 1 {
  E e1,e2;
 public:
  C() { }
  C(int i):e1(i) { }
  C(int i,int j):e1(i),e2(j) { }
};
```

   Welche Meldungen erhält man dann durch die folgenden Definitionen:

```
C c0;
C c1(1);
C c2(1,2);
```

2. Überarbeiten Sie die Klassen *Kreis*, *Quadrat* und *Rechteck* von Aufgabe 6.1.5, 3. sowie *C2DPunkt* aus Aufgabe 6.1.6, 3. so, dass die Konstruktoren alle Elemente mit Elementinitialisierern initialisieren.

3. Überarbeiten Sie die Klasse *Kreis* aus Aufgabe 6.1.5, 3. zu einer Klasse *C2DKreis*. Sie soll als zusätzliches Element einen *C2DPunkt* mit der Position des Kreises enthalten und in einem Zeichenprogramm verwendet werden können, in dem man Kreise zeichnen, verschieben, vergrößern und verkleinern kann.

   Definieren Sie für diese Klasse Konstruktoren, bei denen für die Position ihre Koordinaten oder ein *C2DPunkt* angegeben werden können. Falls nur ein Argument für den Radius übergeben wird, soll die Position der Nullpunkt sein. Ein Standardkonstruktor soll den Radius 1 setzen. Verwenden Sie für möglichst viele Elemente Elementinitialisierer.

4. Im Konstruktor der Klasse *Rechteck1* soll der Mittelpunkt des Rechtecks angegeben werden. In der Klasse soll allerdings nicht der Mittelpunkt, sondern der linke obere Eckpunkt gespeichert werden:

```
class Rechteck1 {
  C2DPunkt LinksOben; // Eckpunkt links oben
  double a,b; // Seitenlängen
 public:
  Rechteck1(C2DPunkt Mittelpunkt, double a_, double b_):
     a(a_),b(b_), LinksOben(Mittelpunkt.X()-a/2,
                            Mittelpunkt.Y()-b/2){ }
  AnsiString toStr()
  {
  return "Links oben: "+LinksOben.toStr();
  }
};
```

   Mit dieser Definition erhält der Punkt *LinksOben* jedoch nicht den beabsichtigten Wert. Finden Sie die Ursache dieses Fehlers und korrigieren Sie die Definition so, dass ein Objekt dieser Klasse das gewünschte Ergebnis hat.

5. In Abschnitt 6.2.1 wurde als Ergänzung zu den Konstruktoren aus Abschnitt 6.1.5 der folgende Standardkonstruktor für die Klasse *MeinString* definiert:

```
class MeinString {
  char* s;
  int n; // Länge des Strings
 public:
  MeinString()
  {
    n=0;
    s=new char[n+1];
    *s='\0';
  };
};
```

Da hier nur Platz für ein einziges Zeichen '\0' reserviert wird, hätte man ebenso gut den folgenden Standardkonstruktor verwenden können:

```
class MeinString {
  // ...
  MeinString():n(0)
  {
    s=new char('\0');
  };
};
```

Vergleichen Sie diese beiden Konstruktoren. Ist einer besser als der andere?

6.2.3 *friend*-Funktionen und -Klassen

6.2.4 Überladene Operatoren als Elementfunktionen

Aufgaben 6.2.4

1. Die Operatorfunktionen für die Ein- und Ausgabeoperatoren << bzw. >> einer selbstdefinierten Klasse C können prinzipiell sowohl als Elementfunktionen der Klasse C als auch als globale Funktionen definiert werden. Vergleichen Sie die beiden Alternativen im Hinblick auf die Syntax beim Aufruf.

2. Ein Bruch kann durch eine Struktur

```
struct Bruch {
  int z,n; // z: Zähler, n: Nenner
};
```

dargestellt werden, bei der alle Datenelemente *public* sind (siehe auch Aufgabe 5.8.2) und bei der die Operatoren etwa folgendermaßen implementiert sind:

```
Bruch operator+(const Bruch& p, const Bruch& q)
{
Bruch result={p.z*q.n + q.z*p.n , p.n*q.n};
return result; // besser: Den Bruch noch kürzen
}

Bruch operator-(const Bruch& p, const Bruch& q)
{
Bruch result={p.z*q.n - q.z*p.n , p.n*q.n};
return result; // besser: Den Bruch noch kürzen
}
```

Überarbeiten Sie die Klasse *Bruch* so, dass die Datenelemente *private* sind. Ergänzen Sie diese Klasse um einen geeigneten Konstruktor.

a) Machen Sie die Funktion für den Operator – zu einer *friend*-Funktion.
b) Definieren Sie die Operatorfunktion += als Elementfunktion und verwenden Sie diese zur Definition des +-Operators, so dass Sie keine *friend*-Funktion benötigen.

3. Ein assoziativer Container ist eine Datenstruktur, die Paare von Schlüsselwerten und Daten verwaltet. In der Klasse *AssozContainer* heißen die Schlüsselwerte und Daten wie in der Standardbibliothek *first* und *second*:

```
class AssozContainer { // alles sehr einfach
  int n; // Anzahl der Elemente im Container
  typedef AnsiString T1;// Datentyp der Schlüsselwerte
  typedef AnsiString T2;// Datentyp der Daten
  struct Paar {
    T1 first;  // Schlüsselwert
    T2 second; // Daten
  };
  Paar a[100]; // nur zur Vereinfachung so einfach
 public:
  AssozContainer():n(0) {};

  void showAll()
  {
  for (int i=0; i<n; ++i)
    Form1->Memo1->Lines->Add(a[i].first+
": "+a[i].second);
  }
};
```

Die Arbeit mit einem solchen Container ist mit einem Indexoperator besonders einfach, der zu einem als Index verwendeten Schlüsselwert die zugehörigen Daten liefert. Mit einem Index, zu dem keine Daten gehören, soll ein neues Paar im Container abgelegt werden.

Die folgenden Beispiele sollen die Arbeit mit einem solchen Operator illustrieren. Die Ergebnisse der Operationen sind jeweils anschließend als Kommentar angegeben:

```
AssozContainer a;
a.showAll();
// Keine Ausgabe, da der Container leer ist.

a["Luigi Mafiosi"]="Luigi@palermo.net";
a.showAll();
// Luigi Mafiosi: Luigi@palermo.net

a["Karl Erbschleicher"]="Karl@aahoohell.kom";
a.showAll();
// Luigi Mafiosi: Luigi@palermo.net
// Karl Erbschleicher: Karl@aahoohell.kom

a["Luigi Mafiosi"]="Luigi@Bankers.net"; // palermo.net
// war zu langsam
a.showAll();
// Luigi Mafiosi: Luigi@Bankers.net
// Karl Erbschleicher: Karl@aahoohell.kom

Memo1->Lines->Add(a["Karl Erbslaicher"]); // Schreib-
a.showAll();                              // fehler
// Luigi Mafiosi: Luigi@Bankers.net
// Karl Erbschleicher: Karl@aahoohell.kom
// Karl Erbslaicher:

// Nicht schön: Karl ist ohne Adresse eingetragen.
// Aber bei std::map ist das auch nicht anders.
```

a) Definieren Sie zu der Klasse *AssozContainer* einen Indexoperator, der dieselben Ergebnisse wie in diesem Beispiel hat.

b) Ein Iterator ist eine Klasse, die eine Position in einem Container darstellt. Sie enthält meist

- einen Zeiger (der die Position darstellt) auf ein Element im Container
- einen Konstruktor, der den Zeiger mit einem anderen Zeiger für eine Position in diesem Container initialisiert
- einen Operator ++, der den Zeiger auf das nächste Element im Container setzt
- einen Operator – –, der den Zeiger auf das vorangehende Element im Container setzt
- einen Operator *, der das Element im Container zum Zeiger im Iterator liefert
- einen Operator ->, der den Zeiger im Iterator liefert
- einen Operator !=, der die Zeiger zweier Iteratoren vergleicht

Für die Klasse *AssozContainer* ist die Klasse *iterator* ein solcher Iterator:

```
class iterator {
  Paar* p;
 public:
  iterator(Paar* p_):p(p_) { }
  bool operator!= (const iterator& y);
  iterator& operator++(int);
  Paar& operator* ();
  Paar* operator-> ();
};
```

Nehmen Sie *iterator* als verschachtelte Klasse in die Klasse *AssozContainer* auf. Definieren Sie die Operatorfunktionen so, dass man den Container wie in dem folgenden Beispiel mit dem Operator ++ durchlaufen kann:

```
AssozContainer::iterator i=a.begin();
for (i=a.begin(); i!=a.end();++i)
  s=i->second;// ebenso: Paar p=*i;
```

Außerdem sollen in der Klasse *AssozContainer* noch die Elementfunktionen *begin* und *end* definiert werden. Sie sollen einen *iterator* zurückliefern, der auf das erste Element im bzw. nach dem Container zeigt.

6.2.5 Der Copy-Konstruktor

6.2.6 Der Zuweisungsoperator = für Klassen

Aufgaben 6.2.6

1. Begründen Sie für jede der folgenden Klassen, ob für sie ein Copy-Konstruktor, ein überladener Zuweisungsoperator oder ein Destruktor explizit definiert werden muss. Falls eine solche Funktion notwendig ist, definieren Sie diese.

   a) Die Klassen *Kreis*, *Quadrat* und *Rechteck* von Aufgabe 6.1.5, 3.
   b) Die Klassen *Grundstueck, Eigentumswohnung, Einfamilienhaus* und *Gewerbeobjekt* von Aufgabe 6.1.5, 4.
   c) Wie wäre ihre Antwort, wenn die Strings in den Klassen von b) nicht mit *char**, sondern durch eine Stringklasse wie *string* oder *AnsiString* definiert wäre.
   d) Kann es mit Nachteilen verbunden sein, diese Funktionen zu definieren, obwohl das nicht notwendig ist, weil sie vom Compiler erzeugt werden?
   e) Oft kann man eine Klasse sowohl mit Zeigern als auch ohne Zeiger definieren, ohne dass eine dieser beiden Varianten Nachteile gegenüber der anderen hat. Vergleichen Sie den Aufwand für die Implementation der beiden Varianten.

2. Beschreiben Sie am Beispiel der Funktionen *test1* und *test2*, wann welche Konstruktoren der folgenden Klassen aufgerufen werden:

```
void display(AnsiString s, int i=-1)
{
if (i>=0) s=s+IntToStr(i);
Form1->Memo1->Lines->Add(s);
}

class C{
  int x;
 public:
  C (int x_=0)
  { // Beim Aufruf ohne Argument ein Standard-
    x=x_;                              // konstruktor
    display("  Konstruktor: ", x);
  }

  C (const C& c)
  {
    x=c.x;
    display("  Kopierkonstruktor: ", x);
  }
```

```
  C& operator=(const C& c)
  {
    x=c.x;
    display("  operator=: ", x);
    return *this;
  }

  ~C ()
  {
    display("  Destruktor: ",x);
  }

  friend int f3(C c);
  friend int f4(const C& c);
};

C f1(int i)
{
return C(i);
}

C f2(int i)
{
C tmp(i);
return tmp;
}

int f3(C c)
{
return c.x;
}

int f4(const C& c)
{
return c.x;
}
```

a) Welche Ausgabe erzeugt ein Aufruf der Funktion *test1*:

```
void test1()
{
display("vor C x=C(1)");
C x=C(1);
C y=x;
display("vor x=y");
x=y;
display("vor C z(x)");
C z(x);
display("vor f1(2)");
f1(2);
display("vor f2(3)");
f2(3);
display("vor f3(4)");
f3(4);
display("vor f3(x)");
f3(x);
display("vor f4(4)");
f4(4);
display("vor f4(x)");
f4(x);
display("Ende von test1");
}
```

b) Welche Ausgabe erzeugt ein Aufruf der Funktion *test2*:

```
class D {
 C c1;
 C c2;
};
```

```
void test2()
{
display("vor D d1");
D d1;
display("vor D d2=d1");
D d2=d1;
display("vor d2=d1");
d2=d1;
display("nach d2=d1");
}
```

3. In der Programmiersprache C werden ganze Arrays wie

```
T a[10], b[10]; // T ein Datentyp
```

oft mit

```
memcpy(a,b,10*sizeof(T));
```

kopiert. Beurteilen Sie dieses Vorgehen, wenn der Datentyp T eine Klasse ist.

4. Das Ergebnis des Präfixoperators ++ ist das veränderte Objekt. Mit einer Variablen x eines vordefinierten Datentyps kann der Ausdruck ++x auch auf der linken Seite einer Zuweisung verwendet werden. Deshalb ist der Funktionswert dieses Operators meist ein Referenztyp. Für eine Klasse C wird der Operator meist mit einer Elementfunktion nach diesem Schema definiert:

```
C& C::operator++() // Präfix ++
{
// erhöhe das Objekt
return *this;
}
```

Der Postfix-Operator liefert dagegen das ursprüngliche Objekt und kann nicht auf der linken Seite einer Zuweisung verwendet werden. Er wird meist nach diesem Schema definiert:

```
C C::operator++(int) // Postfix ++
{
C temp=*this;
++(*this); // Aufruf des Präfix-Operators
return temp;
}
```

Vergleichen Sie die Laufzeit der beiden Operatoren.

5. Für kaufmännische Rechnungen sind Gleitkommadatentypen wie *float* oder *double* ungeeignet, da ihre Ergebnisse nicht exakt sind (siehe auch Abschnitt 3.6.5). Eine Alternative ist ein Festkommadatentyp, der Geldbeträge in ganzzahligen Cent-Beträgen darstellt.

Definieren Sie eine Klasse *FixedP64*, die Dezimalzahlen mit bis zu 4 Nachkommastellen durch das 10000-fache ihres Wertes ganzzahlig darstellt. Verwenden Sie dazu den 64-bit Ganzzahldatentyp *long long*. Die Grundrechenarten +, –, * und / sollen durch überladene Operatoren zur Verfügung gestellt werden. Dazu können die Rechenoperationen von *long long* verwendet werden. Geeignete Konstruktoren sollen Argumente der Datentypen *int* und *double* in diesen Datentyp konvertieren. Eine Funktion *toStr* soll eine solche Festkommazahl als String darstellen.

Testen Sie diesen Datentyp mit der Funktion

```
Festkomma64 Est2005(Festkomma64 x) //
{ // Einkommensteuertarif 2005 nach § 32a (1)
Festkomma64 est;
if (x <= 7664) est = 0;
else if (x <= 12739)
  {
    Festkomma64 y = (x-7664.0)/10000;
    est = (883.74*y+1500)*y;
  }
else if (x <= 52151)
  {
    Festkomma64 z = (x-12739.0)/10000;
    est = (228.74*z + 2397)*z +989;
  }
else est = 0.42*x - 7914;
```

```
return est;
};
```

Dabei müssen sich zu den Werten unter x die jeweils rechts davon unter *Est* angegebenen Werte ergeben (aus der Grundtabelle zum Einkommensteuergesetz):

x	Est	x	Est	x	Est
7 664	0	20 000	2 850	100 000	34 086
7 704	6	40 000	9 223	120 000	42 486
7 740	11	52 092	13 964	140 000	50 886
12 708	981	60 000	17 286	160 000	59 286
12 744	990	80 000	25 686	180 000	67 686

6. Für eine Klasse ohne einen explizit definierten Copy-Konstruktor bzw. Zuweisungsoperator erzeugt der Compiler diese Funktionen, falls sie verwendet werden. Da diese Funktionen leicht aufgerufen werden, ohne dass man es bemerkt, kann es sinnvoll sein, ihren Aufruf zu unterbinden, um flache Kopien zu vermeiden. Wie kann man den Aufruf dieser Funktionen verhindern?

7. Wenn in einem Ausdruck ein **temporäres Objekt** erzeugt wird, ruft der Compiler einen **Konstruktor** für das Objekt auf. Am Ende der Auswertung des Ausdrucks wird dann der **Destruktor** für jedes so erzeugte Objekt aufgerufen. Deshalb können auch bei einfachen Ausdrücken relativ viele Konstruktoren und Destruktoren aufgerufen werden.

   Beschreiben Sie die Ausgabe der Funktion *test3* für die Klasse C aus Aufgabe 2, wenn für C der Operator + definiert wird:

```
C operator+(const C& c1,const C& c2)
{ // friend in der Klasse C
display("  operator+: ", c1.x+c2.x);
return C(c1.x+c2.x);
}

void test3()
{
C s1,s2,s3,s4;
display("1. vor + ");
s1=C(1)+C(3);
display("2. vor + ");
C x=C(5),y=C(7);
s2=x+y;
display("3. vor + ");
s3=C(9);
s3=s3+C(11);
display("4. vor + ");
s4=x;
s4=s4+y;
display("Ende von test");
}
```

6.2.7 Benutzerdefinierte Konversionen

6.2.8 Explizite Konstruktoren Θ

6.2.9 Statische Klassenelemente

6.2.10 Konstante Klassenelemente und Objekte

6.2.11 Klassen und Header-Dateien

Aufgaben 6.2.11

1. In Aufgabe 6.1.5, 3. wurden die Elementfunktionen der Klasse *Kreis* innerhalb und die Elementfunktionen der Klasse *Quadrat* außerhalb der Klasse definiert.

   a) Legen Sie für jede der beiden Klassen eine Header-Datei (Namensendung „.h“) mit der Klassendefinition und für die Klasse *Quadrat* eine Datei *Quadrat.cpp* mit den Funktionsdefinitionen an.
   b) Nehmen Sie die cpp-Datei mit *Projekt|Dem Projekt hinzufügen* in ihr Projekt auf und binden Sie die Header-Dateien mit einer *#include*-Anweisung ein. Verwenden Sie diese Klassen wie in Aufgabe 6.1.5, 3.
   c) Vergleichen Sie diese beiden Alternativen (alle Funktionsdefinitionen bzw. nur die Deklarationen in die Header-Datei) in Hinblick auf Aufwand für den Compiler sowie in Hinblick auf die Laufzeit.

2. Kennzeichnen Sie möglichst viele Elementfunktionen der Klassen *Quadrat* und *Kreis* aus Aufgabe 1 als *const*.

3. Definieren Sie eine Klasse *Singleton*, von der nur ein einziges Objekt erzeugt werden kann. Dazu soll sie eine Funktion *Instance* haben, die einen Zeiger auf dieses Objekt zurückliefert. Beim ihrem ersten Aufruf soll *Instance* ein neues Objekt erzeugen.

   Die Verwaltung von Daten in einer solchen Klasse kann eine Alternative zu einer globalen Definition der Daten sein. Dadurch kann sichergestellt werden, dass mehrere (auch lokale) Definitionen (durch einen Aufruf von *Instance*) immer dieselben Daten verwenden. Siehe dazu Gamma (1995, S. 127).

6.3 Vererbung und Komposition

6.3.1 Die Elemente von abgeleiteten Klassen

6.3.2 Zugriffsrechte auf die Elemente von Basisklassen

6.3.3 Die Bedeutung von Elementnamen in einer Klassenhierarchie

6.3.4 *using*-Deklarationen in abgeleiteten Klassen Θ

6.3.5 Konstruktoren, Destruktoren und implizit erzeugte Funktionen

Aufgaben 6.3.5

1. Welche Ausgabe erhält man durch einen Aufruf der Funktion *test*?

```
class C {
  int i,j;
 public:
  C(int x,int y): i(x),j(y)
  {
  Form1->Memo1->Lines->Add("Konstruktor C");
  }

  C(): i(0),j(0)
  {
  Form1->Memo1->Lines->Add("Standardkonstruktor C");
  }

  ~C()
  {
  Form1->Memo1->Lines->Add("Destruktor C");
  }
};

class D : public C {
  int k,a,b;
  C c;
 public:
  D(int x=1):c(x,1),a(x),b(0),k(19)
  {
  Form1->Memo1->Lines->Add("Konstruktor-1 D");
  }

  D(int x,int y, int z):C(x,y),a(1),b(2),c(x,y),k(z)
  {
  Form1->Memo1->Lines->Add("Konstruktor-2 D");
  }

  ~D()
  {
  Form1->Memo1->Lines->Add("Destruktor D");
  }
};

class E : public D {
  int m;
  C c;
  D b;
 public:
```

```
  E(int x,int y):b(y),c(2,3),m(x+y)
  {
  Form1->Memo1->Lines->Add("Konstruktor E");
  }

  ~E() {
  Form1->Memo1->Lines->Add("Destruktor E");
  }
};

void test()
{
C c(1,2);
D d(1,2,3);
E e(1,2);
}
```

2. Einen eindimensionalen Punkt kann man sich als Zahl auf einem Zahlenstrahl vorstellen. Definieren Sie analog zu den Beispielen im Text eine Klasse *C1DPunkt*, die eine Zahl darstellt. Von dieser Klasse soll *C2DPunkt* und von *C2DPunkt* soll *C3DPunkt* abgeleitet werden. Definieren Sie für jede dieser Klassen Konstruktoren, die alle Koordinaten initialisieren, sowie Funktionen *toStr* und *anzeigen* wie im Text. Die weiteren Elementfunktionen von Aufgabe 6.1.6, 3. brauchen hier nicht enthalten sein.

3. a) Skizzieren Sie für die Klassen *Grundstueck* usw. von Aufgabe 6.1.5, 4. eine Klassenhierarchie. Es ist nicht notwendig, diese in C++ zu schreiben. Falls Sie mehrere Alternativen finden, skizzieren sie alle und überlegen Sie, für welche Sie sich entscheiden würden.
   b) Welche dieser Klassen benötigt einen explizit definierten Copy-Konstruktor, Zuweisungsoperator und Destruktor, wenn die Strings durch
      b1) eine Stringklasse (z.B. *string* bzw. *AnsiString*)
      b2) einen nullterminierten String
      dargestellt werden.
      b3) Definieren Sie diese für eine Basisklasse und eine abgeleitete Klasse.
   c) Vergleichen Sie diese Hierarchie mit den Klassen aus Aufgabe 6.1.5, 4.

4. Manchmal hat man mehrere Möglichkeiten, verschiedene Klassen voneinander abzuleiten:

   a) Da ein Quadrat eine und ein Rechteck zwei Seitenlängen hat, kann man eine Klasse für ein Rechteck von einer Klasse für ein Quadrat ableiten und so die Seitenlänge des Quadrats im Rechteck verwenden.
   b) Man kann ein Quadrat aber auch als Rechteck mit zwei gleichen Seiten betrachten. Definieren Sie eine Basisklasse für ein Rechteck und leiten Sie von dieser eine Klasse für ein Quadrat ab, bei der im Konstruktor die beiden Seitenlängen auf denselben Wert gesetzt werden.
   c) Vergleichen Sie die Vor- und Nachteile der beiden Hierarchien.

5. Die Klassen der C++-Standardbibliothek kann man ebenso wie jede andere Klasse als Basisklasse für eigene abgeleitete Klassen verwenden. Da die Klasse *string* z.B. keinen Konstruktor hat, der ein *int*- oder *double*-Argument in einen String umwandelt, kann man eine abgeleitete Klasse mit einem solchen Konstruktor definieren. Welche Vor- und Nachteile sind mit einer solchen Erweiterung verbunden?

6. Wie kann man erreichen, dass von der Basisklasse keine Objekte angelegt werden können, sondern nur von den abgeleiteten Klassen?

6.3.6 Vererbung bei Formularen im C++Builder

6.3.7 OO Design: *public* Vererbung und „ist ein“-Beziehungen

6.3.8 OO Design: Komposition und „hat ein“-Beziehungen

6.3.9 Konversionen zwischen *public* abgeleiteten Klassen

Aufgaben 6.3.9

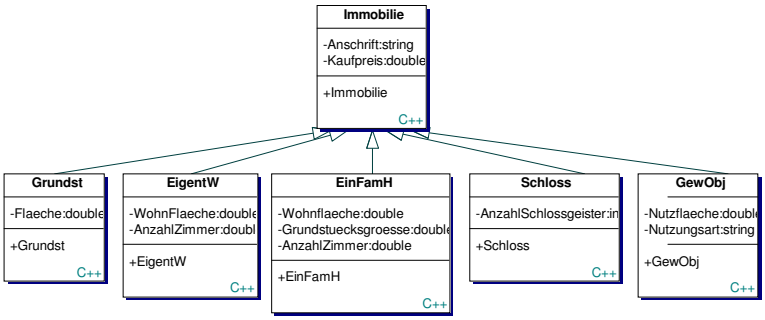
1. Besteht zwischen den Konzepten unter a) bis d) eine „ist ein“- oder eine „hat ein“-Beziehung? Da es oft nicht einfach ist, sich für eine der beiden zu entscheiden, sollen Sie möglichst für beide Sichtweisen Argumente suchen.

   a) Automobil, Motor, Räder
   b) Katze, Hund, Tier
   c) Fahrzeug, Landfahrzeug, Wasserfahrzeug, Automobil, Segelboot
   d) Mitarbeiter, Abteilungsleiter, Sekretärin

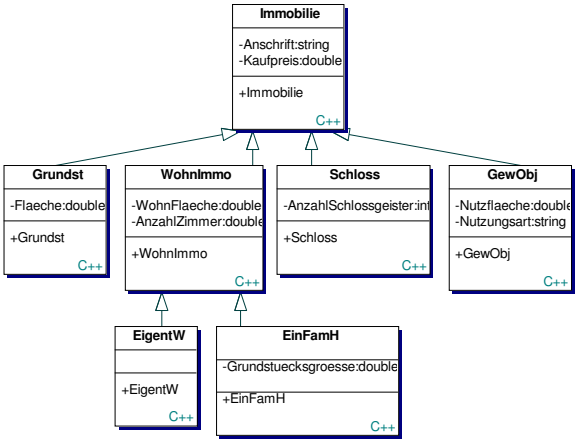
2. In Aufgabe 6.3.5, 4. wurde eine Klasse für ein Quadrat von einer Klasse für ein Rechteck abgeleitet und auch umgekehrt. In Abschnitt 6.3.7 wurde gezeigt, dass die Ableitung eines Quadrats von einem Rechteck mit einer Elementfunktion wie *setzeSeitenlaengen* nicht unproblematisch ist. Prüfen Sie, ob die Elementfunktionen *Flaeche* und *Umfang* aus der Basisklasse in jeder der beiden Hierarchien auch in der abgeleiteten Klasse korrekt sind?

3. Zur Lösung der Aufgabe 6.3.5, 3. werden oft die Hierarchien a) bis e) vorgeschlagen (diese Diagramme wurden mit Borland Together erzeugt). Für welche dieser Hierarchien scheint die „ist ein“-Beziehung auch für die Konzepte gerechtfertigt zu sein?

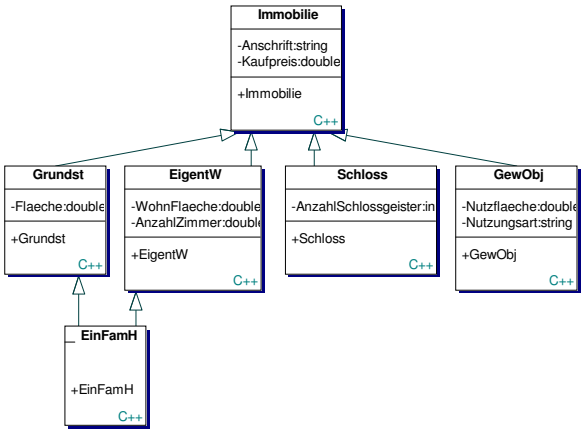
   a)



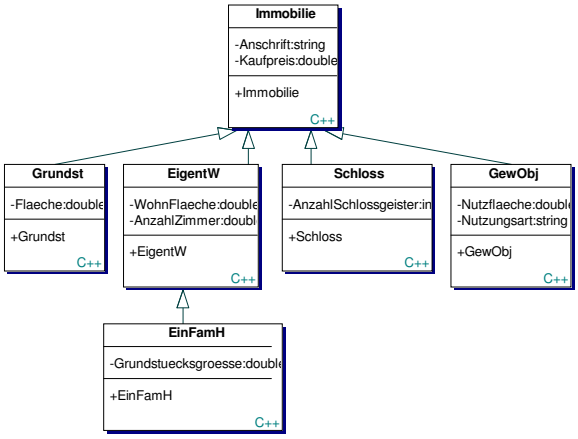
   b)



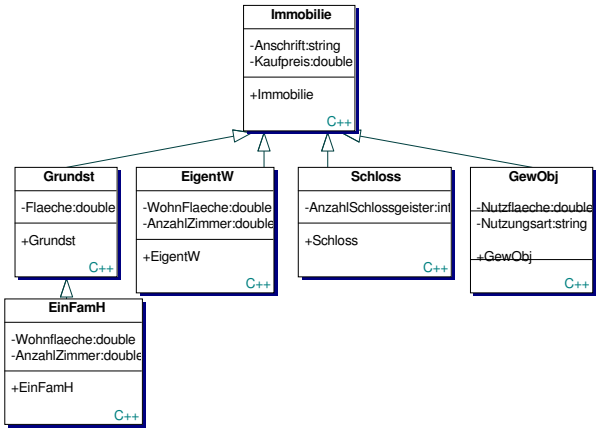
c)



d)



e)



4. Wieso gibt es keine implizite Konversion einer Basisklasse in eine abgeleitete Klasse?

6.3.10 *protected* und *private* abgeleitete Klassen Θ

6.3.11 Mehrfachvererbung und virtuelle Basisklassen

Aufgabe 6.3.11

1. Eine Klasse *Kreis* soll die Funktionen *Flaeche*, *Umfang* und *toStr* haben, und ein *C2DPunkt* die Funktionen *Abstand* und *toStr*. Vergleichen Sie die folgenden Design-Alternativen für eine Klasse, die einen Kreis zusammen mit seiner Position darstellt:

   a) Die Klasse *C2DKreis* soll durch Mehrfachvererbung von einem *Kreis* und einem *C2DPunkt* abgeleitet werden.
   b) Die Klasse *C2DKreis* soll einen *Kreis* und einen *C2DPunkt* als Element enthalten.
   c) Die Klasse *C2DKreis* soll von einem *Kreis* abgeleitet werden und einen *C2DPunkt* als Element enthalten.
   d) Die Klasse *C2DKreis* soll wie in Aufgabe 6.2.2, 2. den Radius und die Position des Kreises als Element des Typs *C2DPunkt* enthalten.

   Definieren Sie die Klassen in a) bis d) mit Konstruktoren, die wie in Aufgabe 6.2.2, 2. aufgerufen werden können. Welche dieser Alternativen ist am besten geeignet, einen Kreis mit seiner Position darzustellen?

6.4 Virtuelle Funktionen, späte Bindung und Polymorphie

6.4.1 Der statische und der dynamische Datentyp

6.4.2 Virtuelle Funktionen

6.4.3 Die Implementierung von virtuellen Funktionen: *vptr* und *vtbl*

Aufgabe 6.4.3

1. Überarbeiten Sie die Klassen *C1DPunkt*, *C2DPunkt* und *C3DPunkt* der Lösung von Aufgabe 6.3.5, 2. so, dass die in jeder Klasse definierte Funktion *toStr* virtuell ist.

   a) Rufen Sie *toStr* nacheinander über einen einzigen Zeiger auf ein Objekt der Basisklasse auf, der nacheinander auf ein Objekt der Klassen *C1DPunkt*, *C2DPunkt* und *C3DPunkt* zeigt. Verfolgen Sie im Debugger (schrittweise Ausführung mit F7), welche der Funktionen t*oStr* dabei aufgerufen werden.
   b) Schreiben Sie eine Funktion *show*, die mit Argumenten der Typen *C1DPunkt*, *C2DPunkt* und *C3DPunkt* aufgerufen werden kann und jeweils den Rückgabewert der Elementfunktion *toStr* ausgibt.
   c) Ergänzen Sie die Klasse *C1DPunkt* um eine nicht virtuelle Elementfunktion *anzeigen*, die *toStr()* ausgibt. Rufen Sie *anzeigen* mit Argumenten der Typen *C1DPunkt*, *C2DPunkt* und *C3DPunkt* auf. Die Klassen sollen *C2DPunkt* und *C3DPunkt* sollen diese Funktion nicht enthalten.
   d) Legen Sie einen *shared_ptr* (siehe Abschnitt 3.12.5) an, der wie in a) nacheinander auf einen *C1DPunkt*, *C2DPunkt* und *C3DPunkt* zeigt, und rufen Sie jedes Mal die Funktion *toStr* auf. Überzeugen Sie sich davon, dass man so dasselbe Ergebnis wie in a) (über einen gewöhnlichen Zeiger) erhält.
   e) Erweitern Sie diese Klassen um Funktionen, die die Länge eines Punktes (d.h. seinen Abstand vom Nullpunkt) zurückgeben:

      – bei *C1DPunkt*: Absolutbetrag von x
      – bei *C2DPunkt*: sqrt(x*x + y*y);
      – bei *C3DPunkt*: sqrt(x*x + y*y + z*z)

2. Die Klassen *C1DPunkt* usw. sollen wie in Aufgabe 1 definiert sein. Sie sollen alle um eine Funktion *setze* erweitert werden, die einen Punkt an die als Argument übergebene Position setzen, wie z.B.:

```
void C1DPunkt::setze(C1DPunkt Ziel);
void C2DPunkt::setze(C2DPunkt Ziel);
```

Kann man diese Funktionen virtuell definieren und so erreichen, dass immer die richtige Funktion zu einem Objekt aufgerufen wird?

3. Damit eine Funktion in einer abgeleiteten Klasse eine gleichnamige Funktion in einer Basisklasse überschreibt, muss die Parameterliste in beiden Funktionen identisch sein. Falls einer der Parameter einen anderen Datentyp hat, verdeckt die Funktion in der abgeleiteten Klasse die der Basisklasse. Das gilt insbesondere auch dann, wenn der Datentyp des Parameters eine abgeleitete Klasse des Parameters der Basisklasse ist.

   Welche Probleme könnten entstehen, wenn eine Funktion f in einer abgeleiteten Klasse D eine gleichnamige Funktion in einer Basisklasse C überschreiben würde und ein Parameter von D::f eine abgeleitete Klasse des entsprechenden Parameters von C::f ist? Sie können dazu die folgenden Klassen und die Funktion g verwenden:

```
struct C {
  virtual void f(C& c) { };
};

struct D : public C {
  int e;
  void f(D& d) { d.e=0; };
};

void g(C& x, C& y)
{
x.f(y);
};
```

4. Welche Werte erhalten i und j bei den Initialisierungen:

```
struct C {
  virtual int f(int i=1)  { return i; }
};

struct D:public C {
  virtual int f(int i=2)  { return i; }
};

C* pc=new D;
int i=pc->f();
C* pd=new D;
int j=pd->f();
```

6.4.4 Virtuelle Konstruktoren und Destruktoren

6.4.5 Virtuelle Funktionen in Konstruktoren und Destruktoren

6.4.6 OO-Design: Einsatzbereich und Test von virtuellen Funktionen

6.4.7 OO-Design und Erweiterbarkeit

6.4.8 Rein virtuelle Funktionen und abstrakte Basisklassen

6.4.9 OO-Design: Virtuelle Funktionen und abstrakte Basisklassen

6.4.10 OOAD: Zusammenfassung

Aufgabe 6.4.10

1. Entwerfen Sie eine Klassenhierarchie für ein Programm, mit dem man Zeichnungen mit Geraden, Kreisen, Quadraten, Rechtecken usw. verwalten und zeichnen kann.

   a) Diese Hierarchie soll insbesondere die mit „ist ähnlich wie ein"-Beziehungen verbundenen Probleme vermeiden (siehe Abschnitt 6.4.9) Stellen Sie für diese Klassen die Funktionen *Flaeche* und *Umfang* zur Verfügung.
   b) Schreiben Sie die Klassen *Gerade*, *Kreis*, *Quadrat* und *Rechteck*. Alle diese Klassen sollen die Funktionen *toStr* (die eine Figur als String darstellt) und *zeichne* haben. Falls Sie wissen, wie man eine solche Figur auf einem *TImage* (siehe Abschnitt 10.13) zeichnet, sollen Sie die Figur zeichnen. Es reicht aber auch völlig aus, wenn die Funktion *zeichne* nur eine Meldung wie „zeichne Quadrat mit Mittelpunkt (0,0) und Radius 17" ausgibt.
   c) Eine Zeichnung soll durch eine Klasse *Zeichnung* dargestellt werden, in der die verschiedenen Figuren in einem Container (z.B. einem Array oder einem *vector*) enthalten sind. Ihre Elementfunktion *zeichne* soll alle Figuren der Zeichnung zeichnen. Falls das Zeichenprogramm später einmal um weitere Figuren erweitert wird, sollen auch diese ohne eine Änderung des Quelltextes der Klasse *Zeichnung* gezeichnet werden können. Testen Sie diese Funktionen mit einer einfachen Zeichnung, in die sie mit einer Funktion *einfuegen* neue Figuren einfügen.
   d) Erweitern Sie die Klassen *Gerade*, *Kreis* usw. um die folgenden Funktionen:
      – *loesche* soll eine Figur auf der Zeichnung löschen, indem sie diese in der Hintergrundfarbe übermalt
      – *verschiebe_Position_um* soll die Position der Figur um den als Argument übergebenen C2DPunkt verschieben
      Erweitern Sie die Klasse *Zeichnung* um die folgenden Funktionen:
      – *loesche* soll alle Figuren der Zeichnung durch den Aufruf ihrer Elementfunktion *loeschen* löschen.
      – *verschiebe_um* soll die Zeichnung zuerst löschen, dann jede Figur um einen als Parameter übergebenen Punkt verschieben, und dann neu zeichnen.
   e) Damit eine Zeichnung in einer Datei gespeichert werden kann, sollen die Klassen um eine Funktion *write* erweitert werden. Diese Funktion soll bei einer Figur die Figur und bei einer Zeichnung alle Figuren der Zeichnung in einen *ofstream* schreiben. Damit man beim Lesen der Datei erkennen kann, zu welcher Figur die Daten gehören, soll vor den Daten eines Objekts der Name der Figur stehen (z.B. im Klartext „Gerade", „Kreis" usw.).
   f) Eine mit den Funktionen von c) angelegte Datei soll mit einer Funktion *LeseZeichnung* der Klasse *Zeichnung* gelesen werden. Dabei muss immer zuerst der Name der Figur gelesen werden. Danach können die Daten zur Figur gelesen werden. Die entsprechende Figur kann mit einem Konstruktor erzeugt werden, der die Daten aus der Datei liest.
   g) Testen Sie diese Klassen mit einer einfachen Zeichnung, die durch entsprechende Anweisungen im Programm angelegt wird.
   h) Beschreiben Sie die für eine Erweiterung dieses Programm um neue Figuren (z.B. Dreieck, Ellipse) notwendigen Schritte.

2. Skizzieren Sie eine Klassenhierarchie

   a) für die verschiedenen Konten (Sparkonten, Girokonten, Darlehenskonten usw.) bei einer Bank.

b) für die Steuerelemente (Buttons, Eingabefelder, Anzeigefelder usw.) einer grafischen Benutzeroberfläche wie Windows. Wie können diese Steuerelemente auf einem Formular verwendet werden?
c) Für die Bauteile (z.B. Motoren, Temperaturfühler, Wasserstandsregler) und die Steuerung einer Waschmaschine. Diese Bauteile sollen in verschiedenen Waschmaschinen verwendet werden (z.B. ein schwacher Motor und ein einfacher Temperaturfühler in einem einfachen Modell, und ein starker Motor in einem Modell der Luxusklasse).

Welche virtuellen bzw. rein virtuellen Funktionen sind in der Klassenhierarchie von a), b) und c) bzw. der Immobilienaufgabe (Aufgabe 6.3.9, 3.a)) vorstellbar?

6.4.11 Interfaces und Mehrfachvererbung

6.4.12 Zeiger auf Klassenelemente Θ

Aufgabe 6.4.12

Definieren Sie eine Klassenhierarchie mit einer Basisklasse und zwei davon abgeleiteten Klassen, ähnlich wie die Klassen *Basisklasse*, *Bestellung* und *Lagerbestand* im Text. Alle Elementfunktionen der Basisklasse sollen rein virtuell sein, denselben Datentyp haben und in den abgeleiteten Klassen überschrieben werden. Bei ihrem Aufruf sollen sie eine Meldung ausgeben, aus der hervorgeht, welche Funktion aufgerufen wurde und zu welcher Klasse sie gehört. In einer Funktion wie *dispatch* soll eine dieser Funktionen über ein Array von Zeigern auf Elementfunktionen aufgerufen werden.

Rufen Sie *dispatch* für alle Kombinationen von Klassen und Elementfunktionen auf. Überprüfen Sie, ob so auch tatsächlich die erwarteten Funktionen aufgerufen werden.

6.4.13 UML-Diagramme für Vererbung und Komposition

6.5 Laufzeit-Typinformationen

6.5.1 Typinformationen mit dem Operator *typeid* Θ

6.5.2 Typkonversionen mit *dynamic_cast* Θ

6.5.3 Anwendungen von Laufzeit-Typinformationen Θ

6.5.4 *static_cast* mit Klassen Θ

6.5.5 Laufzeit-Typinformationen für die Klassen der VCL Θ

Aufgaben 6.5

1. Die folgenden Teilaufgaben verwenden die Klassen:

```
class C {
  virtual void f(){}; // damit C1 polymorph ist
} c;

class D1:public C {
} d1;
```

```
class D2:public C {
} d2;
```

a) Geben Sie den Wert der booleschen Variablen b1, ..., b9 an:

```
c=d1;
bool b1= typeid(c)==typeid(d1);
c=d2;
bool b2= typeid(c)==typeid(d2);

D1* pd1=&d1;
D2* pd2=&d2;
C* pc=pd1;
bool b3= typeid(*pc)==typeid(d1);
bool b4= typeid(pc)==typeid(pd1);
pc=&c;
bool b5= typeid(*pc)==typeid(d1);

C& rc=c;
bool b6= typeid(rc)==typeid(d1);
bool b7= typeid(c)==typeid(d1);
rc=d1;
bool b8= typeid(rc)==typeid(d1);
C& rc1=d1;
bool b9= typeid(rc1)==typeid(d1);
```

b) Welche Ergebnisse würde man in Aufgabe a) erhalten, wenn die Funktion f in der Klasse C nicht virtuell wäre?

c) Zusätzlich zu den Klassen von oben sollen die Klassen E1 und E2 sowie die Zeiger *pd1* usw. definiert sein:

```
class E1:public D2 {
} e1;

class E2:public D2 {
} e2;

D1* pd1=&d1;
D2* pd2=&d2;
E1* pe1=&e1;
E2* pe2=&e2;
C* pc=&c;
```

Geben Sie an, ob die Zeiger p1, ..., p4 nach den folgenden Anweisungen den Wert 0 (Null) oder einen anderen Wert haben:

```
D2* p1= dynamic_cast<D2*>(pc);
pc=pd1;
D2* p2= dynamic_cast<D2*>(pc);
pc=pe1;
D2* p3= dynamic_cast<D2*>(pc);
pc=pe2;
D2* p4= dynamic_cast<D2*>(pc);
```

d) Welche der folgenden *dynamic_cast*-Ausdrücke lösen eine Exception aus?

```
C& rc=c;
C& rd1=d1;
C& rd2=d2;
C& re1=e1;
C& re2=e2;
D2 x1= dynamic_cast<D2&>(c);
D2 x2= dynamic_cast<D2&>(rd1);
D2 x3= dynamic_cast<D2&>(rd2);
D2 x4= dynamic_cast<D2&>(re1);
D2 x5= dynamic_cast<D2&>(re2);

rd1=e1;
D2 x6= dynamic_cast<D2&>(rd1);
```

2. Was wird beim Aufruf der Funktion *test* ausgegeben:

```
class C {
 public:
  virtual void f()
  {
  Form1->Memo1->Lines->Add(typeid(*this).name());
  };

  C()  { f(); }
  ~C() { f(); }
};

class D:public C {
 public:
  D()  { f(); }
  ~D() { f(); }
};

void test()
{
D d;
}
```

7 Exception-Handling

7.1 Die *try*-Anweisung

7.2 Exception-Handler und Exceptions der Standardbibliothek

7.3 Vordefinierte Exceptions der VCL

7.4 Der Programmablauf bei Exceptions

7.5 Das vordefinierte Exception-Handling der VCL

7.6 *throw*-Ausdrücke und selbst definierte Exceptions

7.7 Fehler, Exceptions und die Korrektheit von Programmen

7.8 Die Freigabe von Ressourcen bei Exceptions

7.9 Exceptions in Konstruktoren und Destruktoren

Aufgaben 7

1. Geben Sie an, welche Anweisungen beim Aufruf der Funktion f

```
void f()
{
try { try {
            f1();
            f2();
           }
       catch(...)
           {
            f3();
           }
       f4();
     }
catch (...)
     {
       f5();
     }
}
```

ausgeführt werden, wenn

a) in keiner der Funktionen f1, f2 usw. eine Exception ausgelöst wird
b) in f1 eine Exception ausgelöst wird
c) in f2 eine Exception ausgelöst wird
d) in f1 und f3 eine Exception ausgelöst wird
e) in f1 und f4 eine Exception ausgelöst wird.

2. Definieren Sie eine Funktion mit einem *int*-Parameter, die in Abhängigkeit vom Wert des Arguments eine Exception des Datentyps

a) *int*	d) *char**
b) *char*	e) *exception*
c) *double*	f) *logic_error*

auslöst und zeigen sie den jeweils übergebenen Wert bzw. den der Funktion *what* in einem Exception-Handler an.

3. Die Funktion f soll in Abhängigkeit vom Wert des Arguments eine Exception der Datentypen *exception, logic_error, range_error* oder *out_of_range* auslösen. Welche Ausgabe erzeugt ein Aufruf der Funktion in a)? Von welcher Klasse wird die Funktion *what* in b) bis d) aufgerufen?

a)
```
void g1(int i)
{
try { f(i); }
catch(logic_error& e)
 { Form1->Memo1->Lines->Add("logisch"); }
catch(out_of_range& e)
 { Form1->Memo1->Lines->Add("range"); }
catch(exception& e)
 { Form1->Memo1->Lines->Add("exception"); }
};
```

b)
```
void g2(int i)
{
try { f(i); }
catch(logic_error& e)
 { Form1->Memo1->Lines->Add(e.what()); }
catch(out_of_range& e)
 { Form1->Memo1->Lines->Add(e.what()); }
catch(exception& e)
 { Form1->Memo1->Lines->Add(e.what()); }
};
```

c)
```
void g3(int i)
{
try { f(i);  }
catch(exception e)
 { Form1->Memo1->Lines->Add(e.what()); }
};
```

d)
```
void g4(int i)
{
try { f(i); }
catch(exception& e)
 { Form1->Memo1->Lines->Add(e.what()); }
catch(...)
 { Form1->Memo1->Lines->Add("irgendeine Exception"); }
};
```

4. Beim Aufruf einer Funktion f sollen Exceptions ausgelöst werden können, die sowohl von *Exception* (VCL-Exceptions), von *exception* (Standard-C++ Exceptions) als auch von einer Klasse *myException* abgeleitet sind. Rufen Sie diese Funktion entsprechend auf.

5. Erweitern Sie die Funktionen ***stringToDouble*** und ***stringToInt*** (Aufgabe 4.1, 1.) so, dass eine Exception ausgelöst wird, wenn nicht alle Zeichen konvertiert werden können

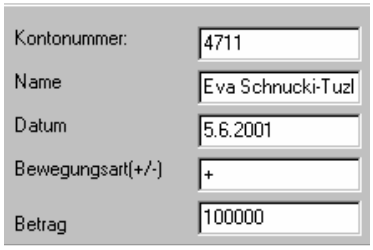
6. Definieren Sie eine von der Klasse *exception* abgeleitete Klasse *MeineException*, deren Elementfunktion *what* einen im Konstruktor angegebenen Text zurückgibt. Lösen Sie in einer Funktion eine Exception dieser Klasse aus und zeigen Sie den Text an.

7. Beurteilen Sie die Funktion *Sqrt*:

```
class ENegative {};

double Sqrt(double d)
{
try {
  if (d<0) throw ENegative();
  return sqrt(d);
}
catch(...)
{
  ShowMessage("negativ");
  return 0;
}
}
```

8. Eine Eingabemaske wie



soll verschiedene Edit-Fenster mit Strings enthalten. Bei der Übernahme der Daten aus dieser Maske sollen die Strings mit Funktionen wie *StrToInt*, *StrToFloat*, und *StrToDate* in Ganzzahl-, Gleitkomma- und Datumswerte umgewandelt werden. Diese Funktionen sollen eine Exception auslösen, wenn die Umwandlung nicht möglich ist.

Entwerfen Sie eine Funktion *GetData*, die die Daten aus der Eingabemaske übernimmt und in die entsprechenden Datentypen konvertiert. Diese Funktion soll von einer Funktion *SaveData* aufgerufen werden können, die die konvertierten Daten z.B. in einem Container speichert. Falls die Konvertierung nicht möglich ist, soll der Anwender durch eine Meldung darauf hingewiesen werden.

a) In einer ersten Variante der Funktion *getData* soll der Anwender nur darauf hingewiesen werden, dass eine der Umwandlungen nicht möglich war.

b) In einer zweiten Variante der Funktion *getData* soll der Anwender gezielt auf die Umwandlung hingewiesen werden, die nicht möglich war.

9. Bei einem Aufruf der Funktion f soll gelegentlich die Gefahr bestehen, dass eine Exception ausgelöst wird. Stellen Sie sicher, dass der mit *new* reservierte Speicher auch wieder freigegeben wird.

```
void h()
{
vector<int*> v;
int* p=new int(17);
v.push_back(p);
f();//falls hier eine Exception auftritt, Speicherleck
for (vector<int*>::iterator i=v.begin(); i!=v.end();
 i++)
  delete i;
}
```

7.10 Exception-Spezifikationen

7.11 Die Funktion *terminate* Ɵ

7.12 Das Win32-Exception-Handling mit *try-__except* Ɵ

8 Die Bibliothek der visuellen Komponenten (VCL)

8.1 Besonderheiten der VCL

8.2 Visuelle Programmierung und Properties (Eigenschaften)

8.2.1 Lesen und Schreiben von Eigenschaften

Aufgabe 8.2.1

Definieren Sie eine einfache Klasse mit einer Eigenschaft mit Lese- und Schreibmethoden. Verfolgen Sie im Debugger schrittweise, zu welchen Aufrufen Zuweisungen von und an diese Eigenschaft führen.

8.2.2 Array-Properties Θ

8.2.3 Indexangaben Θ

8.2.4 Die Speicherung von Eigenschaften in der Formulardatei Θ

8.2.5 Die Redeklaration von Eigenschaften

8.3 Die Klassenhierarchie der VCL

Aufgabe 8.3

Verändert man die Größe eines Formulars während der Laufzeit eines Programms, behalten die Komponenten dieses Fensters ihre ursprüngliche Position und Größe. Dann kann ein Button, der beim Entwurf des Programms in der Mitte des Formulars zentriert war, völlig außerhalb der Mitte liegen.

Schreiben Sie eine Unit *ResizeUnit* mit einer Klasse *TResize* und einer Methode *Resize*. Beim ersten Aufruf von *Resize* sollen die Positionen *Top*, *Left*, *Width* und *Height* aller Komponenten des Formulars in einem Array gespeichert werden. Bei jedem Aufruf von *Resize* sollen dann die Positionsangaben aufgrund der aktuellen Größe des Fensters neu berechnet und gesetzt werden (z.B. mit *SetBounds*).

Ruft man *Resize* beim Ereignis *OnResize* auf, werden die Positionen aller Komponenten bei jeder Änderung der Größe des Formulars angepasst.

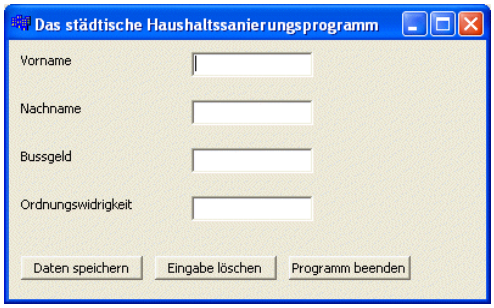
8.4 Selbst definierte Komponenten und ihre Ereignisse

Aufgabe 8.4

1. Die Reaktion auf ein Ereignis kann während der Laufzeit eines Programms dadurch verändert werden, dass man dem Methodenzeiger für dieses Ereignis eine andere Methode zuweist.

   Realisieren Sie dies mit einem Formular, das einen Button und drei RadioButtons enthält. Durch das Anklicken eines der RadioButtons soll eine von drei Ereignisbehandlungsroutinen für das Ereignis *OnClick* des Buttons ausgewählt werden.

2. Definieren Sie eine Klasse *MyForm*, deren Konstruktor ein Formular wie das Folgende ohne die Hilfsmittel der visuellen Programmierung erzeugt:



   Als Reaktion auf ein Anklicken des Buttons mit der Aufschrift

   – *Daten speichern* soll von jedem Edit-Fenster das Textfeld gelesen werden, ohne die Daten weiter zu verwenden.
   – *Eingabe löschen* soll jedes Edit-Fenster mit *Clear* gelöscht werden.
   – *Programm beenden* soll das Formular durch einen Aufruf von *Close* geschlossen werden.

3. Ein Menüeintrag wird durch ein Objekt der VCL-Klasse *TMenuItem* dargestellt und kann durch die folgenden Anweisungen erzeugt werden:

```
TMenuItem *NewMenuItem=new TMenuItem(Form1); // Owner
NewMenuItem->Caption = "Text"; // Menütext
```

   Hier wird der *Owner* auf *Form1* gesetzt. Ein solcher Menüeintrag kann sowohl ein Eintrag in der Menüleiste als auch ein Eintrag in einem Menü sein. Im ersten Fall wird er durch die Funktion *Items->Add* eines Hauptmenüs (hier *MainMenu1*) in die Menüleiste aufgenommen:

```
MainMenu1->Items->Add(NewMenuItem);
```

   Im zweiten Fall wird er durch die Elementfunktion *Add* eines Menüs *Menu* in das Menü aufgenommen:

```
Menu->Add(NewMenuItem);
```

   a) Schreiben Sie eine Funktion *NewMenuBarItem*, die einen Menüeintrag erzeugt und in eine Menüleiste einhängt.
   b) Schreiben Sie eine Funktion *NewMenuItem*, die einen Menüeintrag erzeugt und in ein Menü einhängt.
   c) Beim Anklicken eines der Menüeinträge soll eine Funktion aufgerufen werden, die einen Text in ein Memo schreibt.
   d) Setzt man die Eigenschaft *Visible* eines Menüeintrags auf *false*, wird dieser Eintrag im Menü nicht mehr angezeigt. Blenden Sie als Reaktion auf einen Buttonclick so einen Menüeintrag aus.
   e) Der Menüeintrag mit der Nummer n kann mit der Elementfunktion

      MenuItem::Delete(n-1)

      gelöscht werden. Löschen Sie so als Reaktion das Anklicken eines weiteren Buttons einen der zuvor erzeugten Menüeinträge.

Damit die in den nächsten beiden Aufgaben erzeugten Komponenten im nächsten Abschnitt ohne Änderungen in die Tool-Palette installiert werden können (Aufgabe 8.5), schreiben Sie ihre Klassendefinition in eine Headerdatei (z.B. mit dem Namen „CompU.h“)

```
class TColBorderllabel {
...
}

class TValueEdit{
...
}
```

und die Definition ihrer Elementfunktionen in eine Datei mit der Endung „cpp“. Diese Dateien verwenden Sie in der Lösung dieser Aufgabe mit einer *#include*-Anweisung.

4. Schreiben Sie eine von *TEdit* abgeleitete Klasse *TValueEdit* mit einer *double*-Eigenschaft *Value*. Ein dieser Eigenschaft zugewiesener Wert soll mit der Anzahl von Nachkommastellen als Text der Edit-Komponente angezeigt werden, die einer Eigenschaft *Nachkommastellen* entspricht. Zur Formatierung des Wertes können Sie die Funktion *FormatFloat* mit einem Formatstring der Art "0.00" (für zwei Nachkommastellen) verwenden.

5. Schreiben Sie eine von *TCustomLabel* abgeleitete Komponente *TColBorderLabel*, die ein Label mit einem farbigen Rand darstellt. *TCustomLabel* hat einen Canvas, in den man die Randlinien zeichnen kann. Die Farbe der Randlinien soll durch eine Eigenschaft *BorderColor* und deren Dicke durch die Eigenschaft *BorderWidth* dargestellt werden. Die boolesche Eigenschaft *ShowBorder* soll entscheiden, ob der farbige Rand dargestellt wird oder nicht.

   Falls der Eigenschaft *BlinkIntervall* (Datentyp *int*) ein Wert größer Null zugewiesen wird, soll der Rand blinken. Dazu kann ein Timer verwendet werden, der mit dem in *BlinkIntervall* angegebenen Zeitintervall tickt: Bei jedem Tick wird der Rand neu gezeichnet.

   Machen Sie die Eigenschaften *Align*, *AutoSize*; *Caption* und *Color* aus *TControl* durch eine Redeklaration in *TColBorderLabel* verfügbar (siehe Abschnitt 8.2.5).

6. Selbstdefinierte Komponenten haben oft einen Konstruktor mit zwei Parametern für den *Owner* und den *Parent*:

   ```
   class TMyEdit:public TEdit {
    public:
     __fastcall TMyEdit(TComponent* AOwner,TWinControl* Parent_):TEdit(AOwner){};
   };
   ```

   Da die Argumente für *Parent* und *Owner* oft gleich sind (e.g. *Form1*), scheint es oft als zu aufwendig, jedes Mal zwei Argumente für den Konstruktor anzugeben. Diskutieren Sie diese beiden Alternativen:

   ```
   __fastcall TMyEdit(TComponent* AOwner):TEdit(AOwner)
   { // ...
   };

   __fastcall TMyEdit(TWinControl* AOwner):TEdit(AOwner)
   { // ...
   }
   ```

   Finden Sie noch eine weitere Alternativen?

8.5 Die Erweiterung der Tool-Palette

Aufgabe 8.5

Erweitern Sie die Tool-Palette um die in den Aufgaben des letzten Abschnitts entwickelten Komponenten

TValueEdit
TColBorderLabel

Schreiben Sie diese Komponenten nicht neu, sondern übernehmen Sie „CompU.cpp“ und „Compu.h“ mit *#include*. Verwenden Sie die in die Tool-Palette installierten Komponenten in einer eigenen kleinen Anwendung, in der Sie die von Ihnen definierten und einige der vordefinierten Eigenschaften im Objektinspektor und im Programm setzen.

8.6 Klassenreferenztypen und virtuelle Konstruktoren

Aufgabe 8.6

Zeigen Sie wie in *ShowBaseNames* für eine Klasse (z.B. *TButton*) und alle ihre Basisklassen den jeweils von einem Objekt der Klasse belegten Speicherplatz an. Sie können dazu die Funktion *InstanceSize* von *TMetaClass* verwenden.

8.7 Botschaften (Messages)

8.7.1 Die Message Queue und die Window-Prozedur

8.7.2 Botschaften für eine Anwendung

8.7.3 Botschaften für ein Steuerelement

8.7.4 Selbst definierte Reaktionen auf Botschaften

8.7.5 Botschaften versenden

Aufgaben 8.7

1. Wie schon erwähnt wurde, stellt Windows einen großen Teil seiner Funktionen über Botschaften zur Verfügung. Dazu gehören für Eingabefenster (*TEdit* und *TMemo*) die Botschaften

   WM_COPY // kopiert den markierten Bereich in die Zwischenablage
   WM_CUT // löscht den markierten Bereich
   WM_PASTE // kopiert die Zwischenablage in das Eingabefenster

   Schreiben Sie Funktionen, die diese Operationen mit *SendMessage* realisieren. Die *wParam*- und *lParam*-Argumente für *SendMessage* sind dabei 0.

2. In einem Formular mit verschiedenen Edit-Fenstern soll nach dem Drücken der Enter-Taste automatisch das nächste Edit-Fenster den Fokus erhalten, ohne dass extra die Tab-Taste gedrückt wird.

   Die Weitergabe des Fokus kann man dadurch erreichen, dass man dem Formular die Botschaft *WM_NEXTDLGCTL* sendet, wobei die Parameter *wParam* und *lParam* den Wert 0 bekommen. Diese Botschaft kann man von jedem Edit-Fenster an das Formular senden, wenn das Ereignis *OnKeyPress* eintritt und die Enter-Taste (*CharCode* = 13) gedrückt wird.

   Noch einfacher ist es, wenn man für das Formular die Eigenschaft *KeyPreview* auf *true* setzt und diese Botschaft beim Ereignis *OnKeyPress* versendet, wenn die Enter-Taste gedrückt wird.

3. Wenn man den Schließen-Button in der rechten oberen Ecke eines Fensters anklickt oder Alt-F4 drückt, sendet Windows eine *WM_CLOSE* Botschaft an die Anwendung. Wie können Sie verhindern, dass eine Anwendung auf diese Weise geschlossen wird?

Damit die in den nächsten vier Aufgaben erzeugten Komponenten in Aufgabe 8. ohne Änderungen in die Tool-Palette installiert werden können, schreiben Sie ihre Klassendefinition

```
class TTabedit {
...
}
```

in eine Headerdatei (z.B. mit dem Namen „CompU.h“) und die Definition ihrer Elementfunktionen in eine Datei mit der Endung „cpp“. Diese Dateien verwenden Sie in der Lösung dieser Aufgabe mit einer *#include*-Anweisung.

4. Schreiben Sie eine von *TEdit* abgeleitete Komponente *TTabEdit*. Über eine boolesche Eigenschaft *EnterNextDlgCtl* soll entschieden werden können, ob das Drücken der Enter-Taste den Fokus auf das nächste Steuerelement setzt oder nicht. Sie können sich dazu an Aufgabe 2 orientieren.

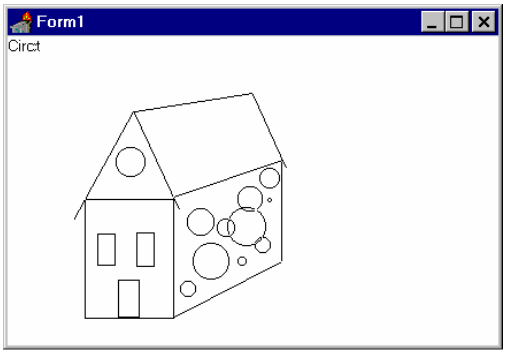
5. Schreiben Sie eine von *TEdit* abgeleitete Komponente *TFocusColorEdit*. Diese soll automatisch eine auswählbare Hintergrundfarbe erhalten, sobald sie den Fokus erhält. Verliert sie den Fokus, soll sie wieder die in *Color* festgelegte Hintergrundfarbe erhalten und ansonsten mit *TEdit* identisch sein. Sie können dazu die Botschaften *WM_SetFocus* und *WM_Killfocus* abfangen. Diese werden einem Dialogelement immer dann zugesandt, wenn es den Fokus erhält oder verliert.

6. Schreiben Sie eine von *TMemo* abgeleitete Komponente *TResizableMemo*. Wenn die linke Maustaste über dieser Komponente gedrückt wird, soll der rechte Rand bei jeder Mausbewegung an die aktuelle x-Koordinate der Mausposition angepasst werden.

7. Viele Zeichenprogramme verwenden zum Zeichnen von Figuren (Linien, Rechtecken, Kreisen usw.) sogenannte „Gummibandfiguren“. Dabei merkt sich das Programm beim Drücken der linken Maustaste die Anfangsposition der zu zeichnenden Figur. Bei jeder Mausbewegung wird dann die zuvor gezeichnete Figur wieder gelöscht und bis zur aktuellen Mausposition neu gezeichnet. Durch dieses Neuzeichnen bei jeder Mausbewegung entsteht der Eindruck, dass die Figur mit einem Gummiband gezogen wird.

   Das Löschen und Neuzeichnen der Figur ist besonders einfach, wenn für *Canvas->Pen->Mode* der Wert *pmNot* gewählt wird (siehe dazu außerdem die Beschreibung von *SetROP2* in der Online-Hilfe zum Windows SDK). Dieser Modus bewirkt, dass anschließende Zeichenoperationen mit der inversen Bildschirmfarbe durchgeführt werden. Damit bewirken zwei aufeinander folgende Zeichenoperationen, dass der ursprüngliche Zustand wieder hergestellt wird, ohne dass man sich um die Hintergrundfarbe kümmern muss (was bei einem mehrfarbigen Hintergrund recht mühsam sein kann).

   Entwerfen Sie eine Komponente *TRubberShape* als Nachfolger von *TImage*, auf der man so Linien, Kreise und Rechtecke zeichnen kann. Diese Komponente ist bereits ein kleines Grafikprogramm, mit dem man einfache Zeichnungen erstellen kann:



8. Erweitern Sie die Tool-Palette um die in den Aufgaben 4. bis 7. entwickelten Komponenten

   TTabEdit *TFocusColorEdit*
   TResizableMemo *TRubberShape*

   Nehmen Sie dazu die Dateien mit den Lösungen in die Lösung der Aufgabe 8.5 auf.

9 Templates und die STL

9.1 Generische Funktionen: Funktions-Templates

9.1.1 Die Deklaration von Funktions-Templates mit Typ-Parametern

9.1.2 Spezialisierungen von Funktions-Templates

9.1.3 Funktions-Templates mit Nicht-Typ-Parametern

9.1.4 Explizit instanziierte Funktions-Templates Θ

9.1.5 Explizit spezialisierte und überladene Templates

9.1.6 Rekursive Funktions-Templates Θ

Aufgabe 9.1

1. Überarbeiten Sie die Funktion *Auswahlsort*

```
void AuswahlSort(int A[],int n)
{
for (int i=0;i<n-1;i++)
  {
    int iMin = i;
    for (int j=iMin+1;j<n;j++)
      if (kleiner(A[j],A[iMin])) iMin=j;
    vertausche(A[iMin],A[i]);
  }
}
```

   zu einem Funktions-Template, mit dem man Arrays beliebiger Elementtypen sortieren kann, wenn für die Elemente der Operator < und die Funktions-Templates *kleiner* wie in Abschnitt 9.1.5 definiert sind. Die Funktion *kleiner* ist hier deswegen notwendig, weil der Operator < für die fundamentalen Datentypen nicht überladen werden kann.

2. Für die Lösung der Aufgabe a) können Sie sich an diesen Funktionen orientieren (siehe Aufgabe 4.1, 1.):

```
#include <sstream>
int stringToInt(const string& s)
{
std::istringstream is(s);
int result=0;
is>>result;
if (s.length()!=is.tellg())
  throw invalid_argument("Cannot convert '"+s+"' to int");
return result;
}

string tostring(int x)
{
std::ostringstream os;
os<<x;
return os.str();
}
```

a) Definieren Sie zwei Funktions-Templates ***string_to*** und ***to_string***, die Argumente aller Datentypen, für die die Ein- und Ausgabeoperatoren >> und << definiert sind, in einen *string* konvertieren bzw. einen *string* in den Datentyp.

   Testen Sie diese Funktionen mit einigen elementaren Datentypen und einem selbstdefinierten Datentyp wie z.B. der Klasse *Bruch* aus Aufgabe 5.8.3

```
struct Bruch {
  int z,n; // z: Zähler, n: Nenner
};

ostream& operator<<(ostream& f, const Bruch& b)
{
return f<<b.z<<"|"<<b.n;
}

istream& operator>>(istream& f, Bruch& b)
{
char Bruchstrich;
f>>b.z>>Bruchstrich>>b.n;
return f;
}
```

b) Für Argumente des Datentyps ***bool*** soll *to_string* die Strings „true“ und „false“ zurückgeben. *string_to<bool>* soll für das Argument „true“ den Wert *true*, für das Argument „false“ den Wert *false* und für alle anderen Strings *success==false* zurückgeben.
c) Für Argumente des Datentyps ***AnsiString*** soll *to_string* den *string* bzw. *string_to<AnsiString>* den *AnsiString* mit den Zeichen des Arguments zurückgeben.
d) Informieren Sie sich in der Dokumentation zur Boost-Bibliothek (http://www.boost.org) über die Klasse *lexical_cast*.

3. Die Funktions-Templates *max* und *for_each* sind in der STL etwa folgendermaßen definiert:

```
template <class T>
inline T max(const T& a, const T& b)
{
  return  a < b ? b : a;
}

template <class InputIterator, class Function>
Function for_each(InputIterator first,
InputIterator last, Function f)
{
  while (first != last) f(*first++);
  return f;
}
```

Außerdem soll die Funktion *print* und die Klasse *Print* definiert sein:

```
void print(int x)
{
Form1->Memo1->Lines->Add(IntToStr(x));
}
```

```
struct Print{
  Print(){};
  // ...
};
```

Geben Sie für die Aufrufe unter a) bis f), die der Compiler übersetzen kann, die Funktionen an, die er dabei erzeugt. Begründen Sie für jeden Aufruf, der nicht übersetzt werden kann, wieso das nicht geht, und mit welchen Änderungen im Programm das doch noch möglich wäre.

a) `max(2,3);`
b) `max(4.4, 5.5);`
c) `max(6,7.8);`
d)
```
int a[10];
for_each(a,a+10,print);
```
e)
```
vector<int> v;
for_each(v.begin(),v.end(),print);
```
f)
```
vector<int> v;
for_each(v.begin(),v.end(),Print());
```

4. Definieren Sie globale Operatorfunktionen als Funktions-Templates, die

   a) den Operator „!=“ auf den Operator „==“ zurückführen und
   b) die Operatoren „<=“, „>“ und „>=“ auf den Operator „<“zurückführen.

   Überzeugen Sie sich am Beispiel einer selbstdefinierten Klasse mit den Operatoren „==“ und „<“ (z.B. der Klasse *Bruch* aus Aufgabe 2) davon, dass die anderen Operatoren über diese Templates verfügbar sind.

9.2 Generische Klassen: Klassen-Templates

9.2.1 Die Deklaration von Klassen-Templates mit Typ-Parametern

9.2.2 Spezialisierungen von Klassen-Templates

9.2.3 Templates mit Nicht-Typ-Parametern

9.2.4 Explizit instanziierte Klassen-Templates Θ

9.2.5 Partielle und vollständige Spezialisierungen Θ

9.2.6 Elemente und *friend*-Funktionen von Klassen-Templates Θ

9.2.7 Ableitungen von Templates Θ

9.2.8 UML-Diagramme für parametrisierte Klassen Θ

Aufgabe 9.2

1. Definieren Sie ein Klassen-Template *Array*, das im Wesentlichen wie ein gewöhnliches Array verwendet werden kann. Der Datentyp und die Anzahl der Elemente sollen als Template-Parameter übergeben werden:

   ```
   Array<int, 100> a; // Ein Array a mit 100 int-Elementen
   ```

   Der Zugriff auf ein Element soll wie bei einem gewöhnlichen Array mit dem Indexoperator möglich sein:

```
for (int i=0;i<100;i++) a[i]=i;
```

Im Gegensatz zu einem gewöhnlichen Array soll eine Exception (z.B. des Typs *std::range_error*) ausgelöst werden, wenn der Index nicht im Bereich der definierten Elemente liegt.

2. Bei einem Klassen-Template müssen die Datentypen der Template-Argumente immer explizit angegeben werden. Deshalb ist es etwas umständlich, mit dem Klassen-Template *pair* der Standardbibliothek ein Objekt zu definieren, das man z.B. in einem assoziativen Container *ac* ablegen will:

```
map<string,string> ac;
ac.insert(pair<string,string>("Daniel","13.11.79"));
```

Bei einem Funktions-Template werden die Datentypen der Template-Argumente dagegen aus den Datentypen der Funktionsargumente abgeleitet. Schreiben Sie ein Funktions-Template ***makePair***, das ein *pair* aus den Funktionsargumenten erzeugt und das wie *make_pair* aus der Standardbibliothek verwendet werden kann:

```
ac.insert(makePair("Daniel","13.11.79"));
```

3. Welche der Aufrufe unter a) bis f) sind mit diesen beiden Templates möglich?

```
template <typename T>
struct Punkt1 {
  T x,y;
  Punkt1(const T& a,const T& b):x(a),y(b) {}
  Punkt1(const Punkt1& p):x(p.x),y(p.y){ };
};

template <typename T>
struct Punkt2 {
  T x,y;
  Punkt2(const T& a,const T& b):x(a),y(b) {}
  template <typename U>
  Punkt2(const Punkt2<U>& p):x(p.x),y(p.y){ };
};
```

a) `Punkt1<int> p1a(1,2);`
b) `Punkt1<int> p1b=p1a;`
c) `Punkt1<double> p1c=p1a;`
d) `Punkt2<int> p2a(1,2);`
e) `Punkt2<int> p2b=p2a;`
f) `Punkt2<double> p3b=p2a;`

4. Schreiben Sie ein Klassen-Template *MyVerySimpleSmartPointer*, das einen Zeiger auf einen mit *new* angelegten Speicherbereich enthält und diesen Speicherbereich im Destruktor wieder freigibt. Auf diese Weise wird eine einfache Form von garbage collection erreicht: Für einen solchen Zeiger muss *delete* nicht aufgerufen werden und kann deshalb auch nicht vergessen werden.

Der Zeiger soll im Konstruktor initialisiert werden und mit den Operatorfunktionen * und -> den internen Zeiger bzw. den dereferenzierten internen Zeiger zurückgeben. Um die bei der Zuweisung von Zeigern möglichen Probleme zu vermeiden, sollen Zuweisungen von *MyVerySimpleSmartPointer*–Objekten unterbunden werden.

Für praktische Anwendungen sollten Sie aber besser die Klassen ***scoped_ptr*** und ***shared_ptr*** der Boost-Bibliothek als die Lösung dieser Aufgabe verwenden.

5. Die verschiedenen Rückgabetypen der Funktion *Average* von Abschnitt 9.2.5 kann man nicht nur mit einem Template wie *float_trait* erreichen, sondern auch mit überladenen Funktions-Templates. Implementieren Sie beide Varianten. Welche ist Ihrer Meinung nach einfacher?

6. Schreiben Sie ein Klassen-Template *NurIntDouble*, das nur mit Template-Argumenten der Datentypen *int* und *double* erzeugt werden kann. Für alle anderen Argumenttypen soll der Compiler eine Fehlermeldung erzeugen.

7. Der Aufruf der **Binärmodus**-Funktionen *read* und *write* der Stream-Klassen *ifstream*, *ofstream* und *fstream* ist ein wenig umständlich, da man die Daten nach *char** konvertieren und die Anzahl der Bytes angeben muss:

```
f.write((char*)&x,sizeof(x));
f.read((char*)&x,sizeof(x));
```

Definieren Sie Klassen-Templates *binary_ifstream binary_ofstream* und *binary_fstream*, die wie in *test_binary_stream* verwendet werden können:

```
void test_binary_stream()
{
binary_ofstream<int> fout("c:\\test\\bin1");
if (fout)
 {
    for (int i=0; i<10; i++)
      if (!fout.write_bin(i))
        ShowMessage("Fehler write fout");
    fout.close();
  }
else
  ShowMessage("Fehler open bin1");
}
```

Die Elementfunktionen *read_bin* und *write_bin* sollen eine Variable vom Typ des Template-Parameters im Binärmodus lesen bzw. schreiben. Außer diesen Funktionen sollen auch alle Funktionen der entsprechenden Stream-Klassen *ifstream*, *ofstream* und *fstream* verfügbar sein. Die Konstruktoren sollen eine Datei immer im Binärmodus öffnen.

9.3 Funktionsobjekte in der STL

9.3.1 Der Aufrufoperator ()

9.3.2 Prädikate und arithmetische Funktionsobjekte

9.3.3 Binder, Funktionsadapter und C++0x-Erweiterungen

Aufgabe 9.3

1. Die Windows-API-Funktion

   int ***lstrcmp****(LPCTSTR lpString1,* // address of first null-terminated string
   LPCTSTR lpString2); // address of second null-terminated string

   vergleicht die beiden als Argument übergebenen nullterminierten Strings gemäß dem Zeichensatz, der sich aus den Ländereinstellungen in der Systemsteuerung von Windows ergibt. Bei der Ländereinstellung Deutsch werden so auch Umlaute berücksichtigt. Der Funktionswert von *lstrcmp* ist kleiner als Null, falls das erste Argument vor dem zweiten kommt, gleich Null, falls beide gleich sind, und andernfalls größer als Null.

   a) Definieren Sie mit *lstrcmp* eine Funktion, die bei *sort* für *Compare* eingesetzt werden kann, um zwei Strings des Datentyps *string* zu vergleichen.
   b) Verwenden Sie die Funktion von a) zum Sortieren eines *vector* v1 mit Elementen des Datentyps *string*. Damit der Unterschied zu c) deutlich wird, sollen diese Strings auch Umlaute enthalten.
   c) Sortieren Sie einen Vektor v2 mit denselben Elementen wie v1 mit der Funktion *sort*, wobei für *Compare* kein Argument übergeben wird.

2. Schreiben Sie zwei Funktions-Templates mit dem Namen *is_sorted*, die genau dann den Funktionswert *true* zurückgeben, wenn die Elemente im Bereich [*first*,*last*) aufsteigend sortiert sind. Verwenden Sie dazu den STL-Algorithmus *adjacent_find*.

   a) Eine erste Variante soll den Operator „<“ verwenden.
   b) Eine zweite Variante soll einen weiteren Parameter *compare* haben, der die Reihenfolge der Elemente prüft.
   c) Testen Sie diese Funktionen mit einigen Arrays und STL-Containern.
   d) Verwenden Sie diese Funktionen, um die Sortierfolge in Aufgabe 1. zu prüfen.

3. Definieren Sie eine Klassenhierarchie, in der eine virtuelle Funktion f einer Basisklasse C in einer abgeleiteten Klasse D und in einer von D abgeleiteten Klasse E überschrieben wird. Die Funktion f soll keine Parameter haben.

a) Ein Container v soll Zeiger auf Objekte dieser Klassenhierarchie enthalten. Überzeugen Sie sich davon, dass mit

```
for_each(v.begin(),v.end(),mem_fun(&C::f));
```

die virtuelle Funktion f für jedes Element aus dem Container aufgerufen wird. Geben Sie dazu in jeder Funktion den Namen der Klasse aus, zu der sie gehört.

b) Erweitern Sie diese Klassenhierarchie um eine virtuelle Funktion g mit einem Parameter des Datentyps *int* und rufen Sie g mit *for_each* auf.

c) Ein Container soll Objekte dieser Klassenhierarchie enthalten. Rufen Sie die virtuelle Funktion f mit *for_each* und *mem_fun_ref* für jedes Element des Containers auf.

9.4 Iteratoren und die STL-Algorithmen

9.4.1 Die verschiedenen Arten von Iteratoren

9.4.2 Umkehriteratoren

9.4.3 Einfügefunktionen und Einfügeiteratoren

9.4.4 Container-Konstruktoren mit Iteratoren

9.4.5 STL-Algorithmen für alle Elemente eines Containers

Aufgabe 9.4

1. Verwenden Sie zur Definition der folgenden Funktions-Templates die STL-Algorithmen *sort* und *copy* sowie geeignete Iteratoren. Alle Datensätze, die in eine Datei geschrieben werden, sollen in eine eigene Zeile geschrieben werden.

   a) Das Funktions-Template *writeToFile* soll die Elemente aus dem Bereich [*first, last*) in eine Datei schreiben, deren Name als Parameter übergeben wird.

   ```
   int a[3]={1,2,3};
   writeToFile<int>(a,a+3,"c:\\test\\s.txt");
   ```

   b) Das Funktions-Template *copyFile* soll eine Datei in eine zweite kopieren. Die Namen der Dateien sollen als Parameter übergeben werden.

   ```
   copyFile<int>("c:\\test\\s.txt","c:\\test\\sc.txt");
   ```

   c) Das Funktions-Template *sortFile* soll eine Datei sortieren. Dazu sollen die Elemente in einen *vector* eingelesen und dieser dann sortiert werden. Der sortierte *vector* soll dann in die Zieldatei geschrieben werden.

   ```
   sortFile<int>("c:\\test\\s.txt","c:\\test\\ss.txt");
   ```

   d) Das Funktions-Template *FileIsSorted* soll den booleschen Wert *true* zurückgeben, wenn die Datei sortiert ist, deren Name als Parameter übergeben wird, und andernfalls *false*. Verwenden Sie dazu die Funktion *is_sorted* aus Aufgabe 9.3, 2.

   ```
   bool b1=FileIsSorted<int>("c:\\test\\s.txt");
   ```

   e) Das Funktions-Template *showFile* soll die Elemente einer Datei, deren Namen als Parameter übergeben wird, am Bildschirm ausgeben.

f) Testen Sie die Funktions-Templates von a) bis d) mit einer sortierten, einer unsortierten, einer leeren und einer Datei mit einem Element des Datentyps *int*. Geben Sie die dabei erzeugten Dateien mit *showFile* aus.

g) Testen Sie *writeToFile* mit einem selbstdefinierten Datentyp, für den der Ein- und Ausgabeoperator definiert ist (wie z.B. *Bruch* von 5.8.3)

2. Ergänzen Sie das Klassen-Template *Array* (Aufgabe 9.2, 1.) um einen Datentyp *iterator* sowie um die beiden Elementfunktionen *begin* und *end*. Diese Funktionen sollen einen Zeiger (Datentyp *iterator*) auf das erste bzw. auf das Element nach dem letzten zurückgeben. Mit diesen Ergänzungen soll ein Array dann mit dem folgenden Aufruf des STL-Algorithmus *sort* sortiert werden können:

```
const int n=100;
Array<int, n> a;
for (int i=0;i<n;i++) a[i]=n-i;
sort(a.begin(),a.end() );
```

2. Konstruieren Sie einen *vector<int>* und einen *vector<Bruch>* mit den Werten aus einer Datei, die in Aufgabe 1 angelegt wurde.

9.5 Die Algorithmen der STL

9.5.1 Lineares Suchen

9.5.2 Zählen

9.5.3 Der Vergleich von Bereichen

9.5.4 Suche nach Teilfolgen

9.5.5 Minimum und Maximum

Aufgabe 9.5.5

Bei den Aufgaben 1. bis 3. können Sie sich an den Ausführungen in Abschnitt 9.4.5 orientieren. Die Lösungen sollen Algorithmen der STL verwenden.

1. Schreiben Sie die folgenden Funktions-Templates:

   a) *Equal* soll genau dann den Wert *true* zurückgeben, wenn zwei als Parameter übergebene Container der STL dieselben Elemente enthalten. Diese Container sollen verschieden sein können, z.B. ein *vector* und ein *set*.
   b) *Count* soll die Anzahl der Elemente als Funktionswert zurückgeben, die in einem als ersten Parameter übergebenen STL-Container enthalten sind und einen als zweiten Parameter übergebenen Wert haben.
   c) *CountFileElements* soll die Anzahl der Elemente als Funktionswert zurückgeben, die in einer Datei, deren Name als Parameter übergeben wird, einen ebenfalls als Parameter übergebenen Wert haben.

2. Schreiben Sie jeweils ein Funktions-Template *MinValue*, das den minimalen Wert

   a) aus einer Datei zurückgibt, deren Name als Parameter übergeben wird.
   c) eines Containers zurückgibt, der als Parameter übergeben wird.
   b) eines Arrays zurückgibt, das als Parameter übergeben wird. Dieser Funktion soll außerdem die Anzahl der Elemente des Arrays übergeben werden.

3. Schreiben Sie in a) bis c) jeweils ein Funktions-Template *for_each_if*, das für alle Werte aus einem Container, für die ein als Argument übergebenes Prädikat den Wert *true* hat, eine ebenfalls als Parameter übergebene Operation f aufruft. Die Operation soll eine Funktion oder ein Funktionsobjekt sein können.

   a) *for_each_if* soll mit einem Container der STL aufgerufen werden können.
   b) *for_each_if* soll mit einer Datei aufgerufen werden können, deren Name als Parameter übergeben wird.
   c) *for_each_if* soll mit einem Array aufgerufen werden können, das ebenso wie die Anzahl seiner Elemente als Parameter übergeben wird.

4. Definieren Sie die folgenden Funktions-Templates der STL selbst. Sie können sich dazu an den Algorithmen orientieren, deren Quelltext in diesem Abschnitt gezeigt wurde. Testen Sie diese mit verschiedenen Containerklassen, z.B. *vector*, *list*, *set*, *string* sowie mit einem Array.

```
template<class InputIterator, class T>
iterator_traits<InputIterator>::difference_type
count(InputIterator first, InputIterator last,
  const T& value);

template<class InputIterator, class Predicate>
iterator_traits<InputIterator>::difference_type
count_if(InputIterator first, InputIterator last,
  Predicate pred);

template<class ForwardIterator>
ForwardIterator min_element(ForwardIterator first,
  ForwardIterator last);

template<class ForwardIterator>
ForwardIterator adjacent_find(ForwardIterator first,
  ForwardIterator last);
```

9.5.6 Elemente vertauschen

9.5.7 Kopieren von Bereichen

9.5.8 Elemente transformieren und ersetzen

9.5.9 Elementen in einem Bereich Werte zuweisen

9.5.10 Elemente entfernen

9.5.11 Die Reihenfolge von Elementen vertauschen

9.5.12 Permutationen

9.5.13 Partitionen

9.5.14 Bereiche sortieren

9.5.15 Binäres Suchen in sortierten Bereichen

9.5.16 Mischen von sortierten Bereichen

9.5.17 Mengenoperationen auf sortierten Bereichen

9.5.18 Heap-Operationen

9.5.19 Verallgemeinerte numerische Operationen

Aufgabe 9.5.19

Verwenden Sie für die Lösungen dieser Aufgaben STL-Algorithmen.

1. In einer Folge mit einer ungeraden Anzahl von Werten ist der **Median** der Wert, der nach einer Sortierung der Folge in der Mitte steht. Bei einer geraden Anzahl von Werten ist der Median der Mittelwert der beiden mittleren Werte. Schreiben Sie ein Funktions-Template *Median*, das den Median der Werte aus einem Container der STL zurückgibt.

2. Schreiben Sie in a) bis c) jeweils ein Funktions-Template *MaxElements*, das die n größten Werte einer Quelle in einen Zielbereich kopiert. Der Zielbereich soll ein beliebiger sequentieller Container der STL sein, der ebenso wie n und die Quelle als Parameter übergeben wird. In den verschiedenen Versionen soll die Quelle sein:

   a) ein Container der STL,
   b) eine Datei, deren Name als Parameter übergeben wird,

c) ein Array.

3. Schreiben Sie ein Funktions-Template *Remove*, das die Elemente mit einem bestimmten Wert aus einem Container der STL entfernt. Sowohl der Container als auch der Wert sollen als Parameter übergeben werden.

4. Schreiben Sie ein Funktions-Template, das zwei verschiedene sortierte Dateien zu einer einzigen sortierten Datei zusammenmischt, deren Namen als Parameter übergeben werden.

5. Eine mit 4 Parametern des Datentyps *int* definierte Funktion f soll zum Testen mit allen Permutationen der Werte 1, 2, 3 und 4 aufgerufen werden. Schreiben Sie eine Funktion, die diese Aufrufe als Datei erzeugt.

10 Verschiedenes

10.1 Symbolleisten, Menüs und Aktionen

10.1.1 Symbolleisten mit Panels und SpeedButtons

10.1.2 Symbolleisten mit Toolbars

10.1.3 Verschiebbare Komponenten mit CoolBar und ControlBar

10.1.4 Die Verwaltung von Aktionen

Aufgabe 10.1

10.2 Eigene Dialoge, Frames und die Objektablage

10.2.1 Die Anzeige von weiteren Formularen und modale Fenster

10.2.2 Vordefinierte Dialogfelder der Objektablage

10.2.3 Funktionen, die vordefinierte Dialogfelder anzeigen

10.2.4 Die Erweiterung der Tool-Palette mit Frames

10.2.5 Datenmodule

10.2.6 Die Objektablage

Aufgaben 10.2

1. Beim Anklicken eines Buttons mit der Aufschrift *Login* soll ein modaler Dialog mit einem *OK*- und einem *Abbrechen*-Button angezeigt werden, in dem der Anwender aufgefordert wird, seinen Namen in ein Edit-Feld einzugeben. Falls der Dialog mit dem *OK*-Button oder der Enter-Taste verlassen wird, soll der Name in der Titelzeile des Hauptprogramms (Form1->Caption) angezeigt werden. Falls er mit dem *Abbrechen*-Button oder der ESC-Taste verlassen wird, sollen keine weiteren Aktionen stattfinden.

2. Ergänzen Sie das Projekt aus 1. um einen weiteren Button. Wenn er angeklickt wird, soll mit *MessageDlg* ein Dialog mit der Meldung „Bitte bestätigen Sie den Kaufvertrag“, einem Informations-Symbol und den Buttons *OK* und *Abbrechen* angezeigt werden. Nach dem Anklicken des OK-Buttons soll mit *ShowMessage* die Meldung „Wir buchen den Rechnungsbetrag von Ihrem Konto ab“ angezeigt werden, und nach dem Abbrechen-Button „Das werden Sie noch bereuen“.

3. Erzeugen Sie das Frame von Abschnitt 10.2.4 und legen Sie es in der Tool-Palette ab. Verwenden Sie es dann in einem anderen Projekt.

10.3 Größenänderung von Steuerelementen zur Laufzeit

10.3.1 Die Eigenschaften *Align* und *Anchor*

10.3.2 Die Komponenten *Splitter* und *HeaderControl*

10.3.3 GridPanel: Tabellen mit Steuerelementen

10.3.4 Automatisch angeordnete Steuerelemente: FlowPanel

10.4 ListView und TreeView

10.4.1 Die Anzeige von Listen mit ListView

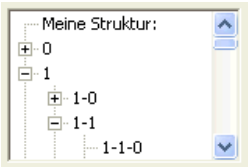
10.4.2 ListView nach Spalten sortieren

Aufgabe 10.4.2

10.4.3 Die Anzeige von Baumdiagrammen mit TreeView

Aufgabe 10.4.3

1. Erzeugen Sie ein TreeView mit 10 Einträgen auf der obersten Ebene. Der Text dieser Einträge soll ihrer jeweiligen Nummer entsprechen. Jeder Eintrag soll 20 Untereinträge enthalten, und jeder dieser Untereinträge wiederum 30 Untereinträge. Der Text eines Untereintrags soll sich aus dem Text des übergeordneten Eintrags sowie dem Index des aktuellen Eintrags zusammensetzen.

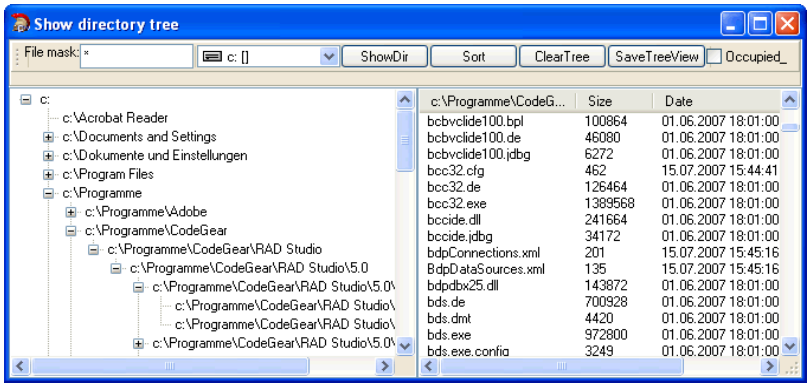


2. Nehmen Sie die folgenden Funktionen in ein Projekt (z.B. mit dem Namen *DirTree*) auf. Zum rekursiv Durchsuchen der Verzeichnisse eines Laufwerks können Sie ich an Abschnitt 5.3.7 orientieren.

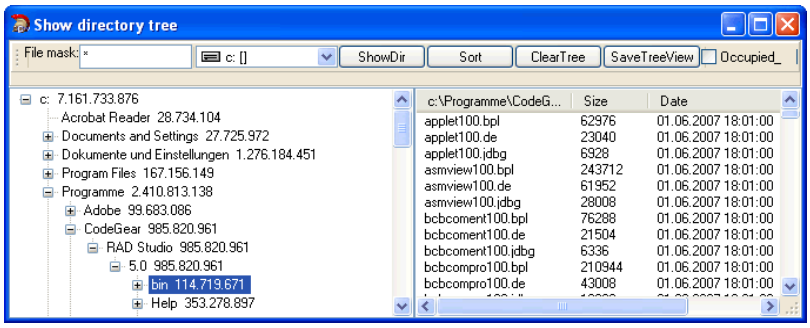
a) Schreiben Sie eine Funktion *SearchSubdirs*, die den vollständigen Pfad aller Verzeichnisse und Unterverzeichnisse eines als Parameter übergebenen Verzeichnisses wie im Windows-Explorer in einen als Parameter übergebenen TreeView einhängt:

Dabei kann wie in der Abbildung auf Icons usw. verzichtet werden. Falls Sie die Aufgabe 5.3.7 gelöst haben, können Sie die Funktion *SearchSubdirs* aus der Lösung dieser Aufgabe überarbeiten. Testen Sie diese Funktion mit einigen Verzeichnissen, die Sie hart kodiert in den Quelltext eingeben.

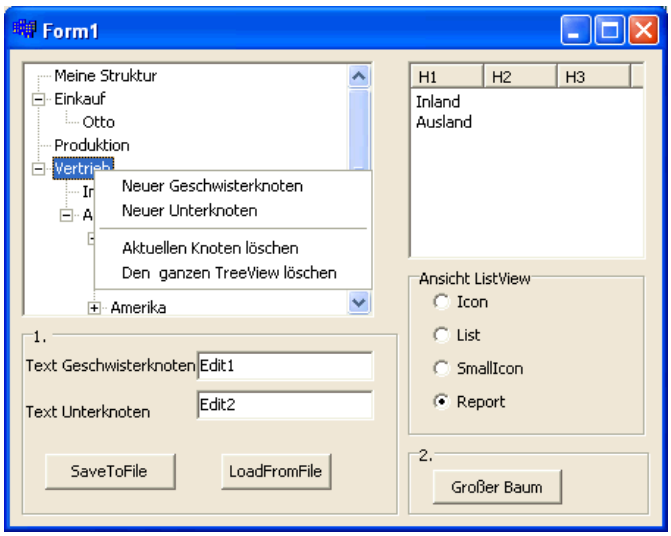
b) Erweitern Sie das Formular von Aufgabe a) um ein ListView, in dem beim Anklicken eines Verzeichnisses im TreeView alle Dateien dieses Verzeichnisses angezeigt werden. Sie können dazu die Funktion *showFilesOf-Directory* von Aufgabe 10.4.2 verwenden.



c) Überarbeiten Sie die Lösungen der Aufgabe b) so, dass im TreeView der Name des Verzeichnisses (und nicht wie in Aufgabe b) der vollständige Pfad) zusammen mit der Anzahl der in allen Unterverzeichnissen belegten Bytes angezeigt wird. Den für die rekursive Suche in den Unterverzeichnissen notwendigen Pfad können Sie über die Eigenschaft *Data* speichern.



3. Ein TreeView soll ein Baumdiagramm darstellen, das während der Laufzeit des Programms aufgebaut wird. Dazu soll ein Kontextmenü zum TreeView die Optionen „Neuer Geschwisterknoten“, „Neuer Unterknoten“, „Knoten löschen“ und „Baum löschen“ anbieten.



a) Beim Anklicken der Optionen
   – „Neuer Geschwisterknoten“ soll ein neuer Geschwister-Knoten des aktuell ausgewählten Knotens in den TreeView eingefügt werden. Sein Text soll aus einem ersten Edit-Feld *Edit1* übernommen werden.
   – „Neuer Unterknoten“ soll ein neuer Kind-Knoten zum aktuell ausgewählten Knoten in das TreeView eingefügt werden. Sein Text soll aus einem zweiten Edit-Feld *Edit2* übernommen werden.
   – „Aktuellen Knoten löschen“ soll der ausgewählte Knoten gelöscht werden.
   – „Den ganzen TreeView löschen“ sollen alle Knoten des TreeView gelöscht werden.

b) Beim Anklicken eines TreeView-Knotens soll der Text aller untergeordneten Knoten im ListView angezeigt werden.

c) Die Ansicht des *ListViews* soll über 4 RadioButtons zwischen *vsIcon*, *vsList*, *vsSmallIcon* und *vsReport* umgeschaltet werden können.

d) Über zwei weitere Buttons soll das TreeView als Datei gespeichert bzw. aus einer Datei geladen werden können.

10.5 Formatierte Texte mit der RichEdit-Komponente

Aufgabe 10.5

Erzeugen Sie mit einer RichEdit-Komponente ein RTF-Dokument, das aus einer Überschrift (Schriftart Arial, Schriftgröße 15, fett) besteht sowie aus einigen weiteren Zeilen in der Schriftart Courier, Schriftgröße 12, nicht fett. Öffnen Sie diese Datei dann mit Microsoft Word oder einem anderen Editor, der das RTF-Format lesen kann (z.B. wordpad, aber nicht notepad).

10.6 Tabellen

10.7 Schieberegler: ScrollBar und TrackBar

Aufgabe 10.7

In der frühen Steinzeit der Rechenmaschinen (bis ca. 1970) gab es nicht nur Digitalrechner (wie heute nahezu ausschließlich), sondern auch **Analogrechner**. Die Bezeichnung „analog“ kommt daher, dass mathematische Zusammenhänge durch physikalische Geräte dargestellt wurden, bei denen aus Eingabewerten in Analogie zu den mathematischen Zusammenhängen Ausgabewerte erzeugt werden. Beispiele sind der Rechenschieber oder spezielle elektrische Geräte, bei denen man die Operanden an Drehreglern eingeben und das Ergebnis an Zeigerinstrumenten ablesen konnte. Analogrechner wurden oft für spezielle Aufgaben entwickelt, z.B. um mit den Kirchhoffschen Regeln Gleichungssysteme zu lösen. Sie waren oft wesentlich schneller als die damaligen Digitalrechner.

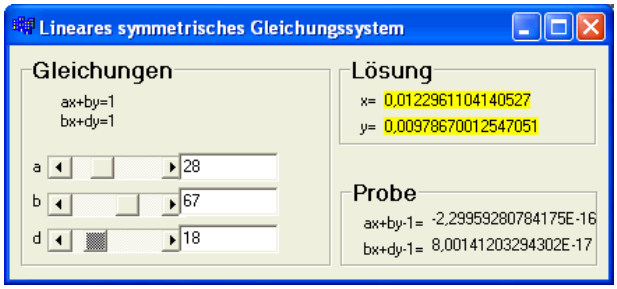
Schreiben Sie ein Programm, mit dem man wie bei einem Analogrechner die Koeffizienten a, b und d des symmetrischen linearen Gleichungssystems

$$ax + by = 1$$
$$bx + dy = 1$$

an Schiebereglern einstellen kann, und das bei jeder Positionsänderung eines Schiebereglers die Lösung

$$x = (b - d)/(b*b - a*d)$$
$$y = (b - a)/(b*b - a*d)$$

ausgibt:



Wenn einer der Schieberegler bewegt wird (Ereignis *OnChange*), sollen die Koeffizienten in einem Edit-Feld und die Ergebnisse sowie eine Probe auf einem Label dargestellt werden.

Stellen Sie mit einer *if*-Anweisung sicher, dass keine Division durch 0 stattfindet. In diesem Fall braucht kein neuer Wert für x bzw. y angezeigt werden, und im Feld für die Lösung soll „Division durch 0“ stehen.

Beachten Sie, dass eine Division von zwei *int*-Operanden mit dem Operator „/“ als Ganzzahldivision (ohne Nachkommastellen im Ergebnis, z.B. 7/4=1) durchgeführt wird. Ein Ergebnis mit Nachkommastellen kann man dadurch erreichen, dass man einen der beiden Operanden (z.B. durch eine Addition mit 0.0) zu einem Gleitkommaausdruck macht.

Falls bei Ihrer Lösung mehrere Ereignisbehandlungsroutinen mit denselben Anweisungen reagieren, brauchen Sie diese Anweisungen nicht mehrfach schreiben: Es reicht, eine einzige dieser Ereignisbehandlungsroutinen zu schreiben und diese dann im Objektinspektor über das Dropdown-Menü des Ereignisses den anderen Ereignissen zuzuordnen.

10.8 Weitere Eingabekomponenten

10.8.1 Texteingaben mit MaskEdit filtern

10.8.2 Die Auswahl von Laufwerken und Verzeichnissen

10.9 Status- und Fortschrittsanzeigen

10.10 Klassen und Funktionen zu Uhrzeit und Kalenderdatum

10.10.1 *TDateTime*-Funktionen

10.10.2 Zeitgesteuerte Ereignisse mit einem Timer

10.10.3 Hochauflösende Zeitmessung

10.10.4 Kalenderdaten und Zeiten eingeben

Aufgabe 10.10

1. Schreiben Sie ein Programm mit den folgenden beiden Funktionen:

a) Die Hintergrundfarbe eines Edit-Fensters soll gelb blinken, wenn es den Fokus hat. Sie können dazu die Ereignisse *OnEnter* und *OnExit* verwenden.
b) Prüfen Sie Genauigkeit von Timer-Ereignissen, indem Sie die Ergebnisse von zwei Timern beobachten. Der erste Timer hat ein Intervall von 100 und erhöht den Wert einer Variablen um 1. Der zweite hat ein Intervall von 1000 und erhöht eine zweite Variable um 10. Er gibt außerdem die Werte der beiden Variablen in einem Memo aus. Im Idealfall wären die Werte von beiden Variablen gleich.
Geben Sie außerdem bei jedem Tick die mit *HRTimeInSec* gemessene Zeit seit dem Start aus.

2. Schreiben Sie eine Klasse *CHRTimer* mit den Elementfunktionen *Start*, *End* und *TimeStr*, die ohne die globale Variable *HRFrequency* auskommt. Beim Aufruf von *Start* und *End* sollen entsprechende Datenelemente der Klasse auf die aktuelle Zeit gesetzt werden. Die Funktion *TimeStr* soll die zwischen den letzten Aufrufen von *Start* und *End* vergangene Zeit als String ausgeben (in Sekunden) und folgendermaßen verwendet werden können:

```
CHRTimer t;
t.Start();
s=Sum1(n);
t.End();
Memo1->Lines->Add("t1="+t.TimeStr());
```

Legen Sie eine Datei (z.B. mit dem Namen „TimeUtils.h“) an, die alle Funktionen und Deklarationen für die hochauflösende Zeitmessung enthält. Wenn man diese Datei im Verzeichnis „\CppUtils“ ablegt, kann man sie mit

```
#include "\CppUtils\TimeUtils.h"
```

in ein Programm einbinden und so Ausführungszeiten messen.

10.11 Multitasking und Threads

10.11.1 Multithreading mit der Klasse *TThread*

10.11.2 Der Zugriff auf VCL-Elemente mit *Synchronize*

10.11.3 Kritische Abschnitte und die Synchronisation von Threads

10.12 TrayIcon

10.13 *TCanvas* und *TImage*: Grafiken anzeigen und zeichnen

10.13.1 Grafiken anzeigen mit *TImage*

10.13.2 Grafiken zeichnen mit *TCanvas*

10.13.3 Welt- und Bildschirmkoordinaten

10.13.4 Figuren, Farben, Stifte und Pinsel

10.13.5 Text auf einen Canvas schreiben

10.13.6 Drucken mit *TPrinter*

10.13.7 Grafiken im BMP- und WMF-Format speichern

10.13.8 Auf den Canvas einer PaintBox oder eines Formulars zeichnen

Aufgaben 10.13

1. Schreiben Sie eine Funktion ***PrintMemo***, die die Textzeilen eines als Parameter übergebenen Memos auf dem Drucker ausgibt. Sie können sich dazu an Abschnitt 10.13.5 orientieren.

   Am Anfang einer jeden Seite soll eine Überschriftszeile gedruckt werden, die die aktuelle Seitenzahl enthält. Prüfen Sie vor dem Ausdruck einer Zeile, ob sie noch auf die aktuelle Seite passt. Falls das nicht zutrifft, soll ein Seitenvorschub ausgelöst werden. Achten Sie insbesondere darauf, dass eine Druckseite nicht allein aus einer Überschriftszeile bestehen kann.

2. Schreiben Sie ein Programm, mit dem man mathematische **Funktionen** der Form y=f(x) **zeichnen** kann.

   a) Überarbeiten Sie die Funktion *zeichneFunction* von Abschnitt 10.13.3 so, dass sie in Abhängigkeit von einem als Parameter übergebenen Wert eine der folgenden Funktionen zeichnet:

	Funktion	x0	y0	x1	y1	dx	dy
1	y=sin (x*x)	–4	–1	4	1	1	0.2
2	y=exp(x)	–1	0	6	100	1	10
3	y=x*x	–2	–2	2	4	1	1
4	y=1/(1+x*x)	0	0	4	1	1	0.2
5	y=x*sin(x)	–2	–6	8	10	1	1

   Die Eckpunkte des Weltkoordinatensystems x0, y0, x1, y1 sollen als Parameter übergeben werden.

   b) Schreiben Sie eine Funktion ***Clear***, die einen als Parameter übergebenen Canvas löscht. Sie können dazu die Zeichenfläche mit der Funktion

   *void __fastcall **FillRect**(const TRect &Rect);*

   mit einem weißen Rechteck füllen.

   c) Schreiben Sie eine Funktion ***zeichneGitternetz***, die auf einem als Parameter übergebenen Canvas ein graues Gitternetz (Farbe *clGray*) zeichnet. Dazu kann man ab dem linken Bildrand (in Weltkoordinaten: x0) jeweils im Abstand *dx* Parallelen zur y-Achse zeichnen. Entsprechend ab y0 im Abstand *dy* Parallelen zur x-Achse. Die Werte x0, x1, dx, y0, y1 und dy sollen als Parameter übergeben werden.

   Falls die x- bzw. y-Achse in den vorgegebenen Weltkoordinaten enthalten sind, soll durch den Nullpunkt ein schwarzes Koordinatenkreuz gezeichnet werden. Außerdem sollen die Schnittpunkte der Achsen und der Gitterlinien mit den entsprechenden Werten beschriftet werden.

   Falls die x- bzw. y-Achse nicht sichtbar ist, soll diese Beschriftung am Rand erfolgen.

   d) An eine überladene Version der Funktion *zeichneFunktion* soll die zu zeichnende Funktion als Funktionszeiger (siehe Abschnitt 5.2) des Typs „double (*) (double)“ übergeben werden.

   Dieser Version von *zeichneFunktion* soll die linke und rechte Grenze des Bereichs, in dem sie gezeichnet werden soll, als Parameter übergeben werden. Der minimale und maximale Funktionswert soll in *zeichneFunktion* berechnet werden. Die zu zeichnende Funktion soll den Canvas von unten bis oben ausfüllen.
   Auf die Zeichenfläche soll außerdem ein Gitternetz mit z.B. 10 Gitterlinien in x- und y-Richtung gezeichnet werden.
   Testen Sie diese Funktion mit den Funktionen aus a).

3. Der **Binomialkoeffizient** bin(n,k) ist der k-te Koeffizient von $(a+b)^n$ in der üblichen Reihenfolge beim Ausmultiplizieren:

   $(a+b)^1 = 1*a+1*b$, d.h. bin(1,0)=1 und bin(1,1)=1
   $(a+b)^2 = 1*a^2+ 2*ab+1*b^2$, d.h. bin(2,0)=1, bin(2,1)=2 und bin(2,2)=1
   $(a+b)^3 = 1*a^3+3*a^2b+3*ab^2+1*b^3$, d.h. bin(3,0)=1, bin(3,1)=3 usw.

   Für einen Binomialkoeffizienten gilt die folgende Formel:

   bin(n,k) = n!/(k!*(n–k)!)

   a) Schreiben Sie eine Funktion bin(n,k). Zeigen Sie die Werte von bin(n,k) für k=1 bis n und für n=1 bis 30 in einem Memo-Fenster an.

   b) Wenn man als Zufallsexperiment eine Münze wirft und das Ergebnis „Kopf“ mit 0 und „Zahl“ mit 1 bewertet, sind die beiden Ergebnisse 0 und 1 möglich.

   Wirft man zwei Münzen, sind die Ergebnisse

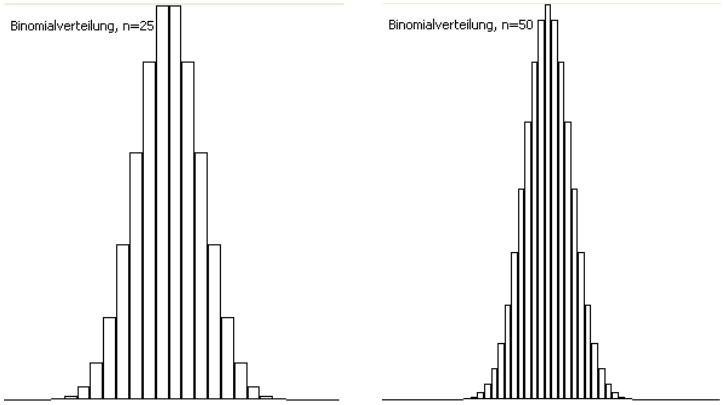
   (0,0), (0,1), (1,0) und (1,1)

   möglich. Damit ist die Anzahl der Ereignisse, die zu den Summen S=0, S=1 und S=2 führen, durch

bin(2,0) = 1, bin(2,1) = 2 und bin(2,2) = 1

gegeben. Es lässt sich zeigen, dass diese Beziehung ganz allgemein gilt: Beim n-fachen Werfen einer Münze ist die Anzahl der Ereignisse, die zu der Summe S=k führt, durch bin(n,k) gegeben (**Binomialverteilung**).

Stellen Sie die Binomialverteilung durch Histogramme grafisch dar:



Anscheinend konvergieren diese Rechtecke gegen eine stetige Funktion. Diese Funktion unter dem Namen Gauß'sche Glockenkurve oder Normalverteilung bekannt (nach dem Mathematiker Gauß).

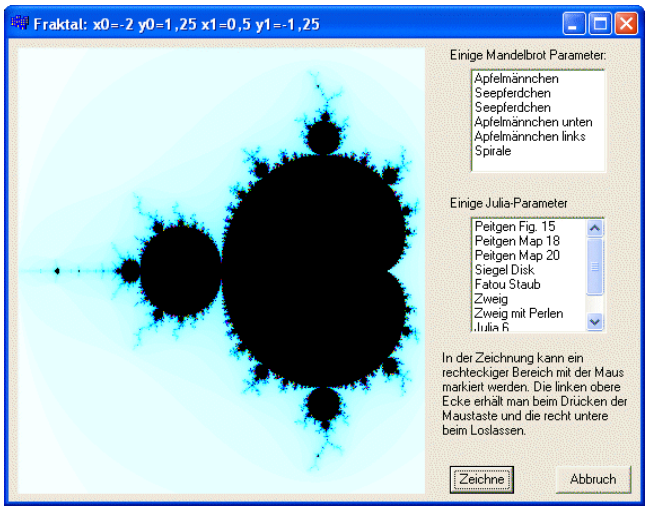
4. Die bekannten **Fraktalbilder** der **Mandelbrot**-Menge (nach dem Mathematiker Benoit Mandelbrot) entstehen dadurch, dass man mit den Koordinaten eines Bildpunktes (x,y) nacheinander immer wieder folgende Berechnungen durchführt:

$$x = x^2 - y^2 + x$$
$$y = 2xy + y$$

Dabei zählt man mit, wie viele Iterationen notwendig sind, bis entweder $x^2 + y^2 > 4$ gilt oder bis eine vorgegebene maximale Anzahl von Iterationen (z.B. 50) erreicht ist.

In Abhängigkeit von der Anzahl i dieser Iterationen färbt man dann den Bildpunkt ein, mit dessen Koordinaten man die Iteration begonnen hat: Falls die vorgegebene maximale Anzahl von Iterationen erreicht wird, erhält dieser üblicherweise die Farbe Schwarz (*clBlack*). In allen anderen Fällen erhält der Bildpunkt einen von i abhängigen Farbwert.

Färbt man so alle Bildpunkte von x0 = –2; y0 = 1.25; (links oben) bis x1 = 0.5; y1 = –1.25; (rechts unten) ein, erhält man das „**Apfelmännchen**".



a) Schreiben Sie eine Funktion *zeichneFraktal*, die ein solches Fraktal auf ein Image (Tool-Palette Kategorie „Zusätzlich") zeichnet. Diese Funktion soll nach dem Anklicken des Buttons „Zeichne" aufgerufen werden. Sie können dazu folgendermaßen vorgehen:

Jeder Bildpunkt (px,py) des *Canvas* wird in die Koordinaten des Rechtecks mit den Eckpunkten (x0,y0) {links oben} und (x1,y1) {rechts unten} mit den Funktionen *x_Welt* und *y_Welt* (siehe Abschnitt 10.13.3) transformiert:

```
double x = x_Welt(px,x0,x1,Image->Width);
double y = y_Welt(py,y1,y0,Image->Height);
```

Mit jedem so erhaltenen Punkt (x,y) werden die oben beschriebenen Berechnungen durchgeführt. Einen Farbwert zu der so bestimmten Anzahl i von Iterationen kann man z.B. folgendermaßen wählen:

```
i=i%256; // maximal 255 Farben
int r30=i%30;    // i->[0..29]
int t=255-5*r30; // [0..29]->[255..145]
if (i<30)       return RGB(255-8*i,255,255);
else if (i<60)  return RGB(0,255-8*r30,255);
else if (i<90)  return RGB(0,0,t);
else if (i<120) return RGB(0,t,0);
else if (i<150) return RGB(t,0,0);
else if (i<180) return RGB(t,t,0);
else if (i<210) return RGB(t,0,t);
else if (i<240) return RGB(0,t,t);
else            return RGB(t,t,t); // Graustufen
```

Hier werden z.B. Werte von i zwischen 60 und 90 auf verschiedene Blautöne abgebildet. Dabei werden Farben vermieden, die nahe bei schwarz liegen. Diese Anweisungen sind aber nur als Anregung gedacht. Mit anderen Farbzuordnungen kann man interessante Variationen erhalten.

b) Ausschnitte eines Fraktalbildes kann man folgendermaßen vergrößern („**Zoom in**“): Man verwendet die Mauskoordinaten beim Drücken bzw. Loslassen der Maustaste (Ereignisse *OnMouseDown* bzw. *OnMouseUp*) als Eckpunkte (links oben bzw. rechts unten) des neuen Bildes. Diese Koordinaten werden dann mit denselben Formeln wie unter a) in die neuen Weltkoordinaten (x0,y0) und (x1,y1) umgerechnet. Nach dem Anklicken des Buttons „Zeichne“ wird das Fraktalbild dann mit den neuen Werten (x0,y0) und (x1,y1) gezeichnet.

c) Bei den Fraktalbildern der **Julia**-Menge (nach dem französischen Mathematiker Gaston Julia) verwendet man anstelle der am Anfang dieser Aufgabe angegebenen Formeln die folgenden:

$$x = x^2 - y^2 + jx$$
$$y = 2xy + jy$$

Der einzige Unterschied zu den Formeln für die Mandelbrot-Fraktale ist der, dass jx und jy konstante Werte sind (z.B. jx=-0.745 und jy=0.1125).

Erweitern Sie das Programm so, dass in Abhängigkeit von einer globalen Variablen Mandelbrot- oder Julia-Fraktale gezeichnet werden.

d) In je einer Listbox sollen Parameter für verschiedene Mandelbrot- und Julia-Fraktale zur Auswahl angeboten werden. Beim Anklicken eines Eintrags der Listbox sollen dann die entsprechenden Parameter für die Eckpunkte übernommen werden. Dazu kann man eine Struktur wie

```
struct TMandelbrotParam {
  char* Name;
  double x0; double y0;
  double x1; double y1;
};
```

definieren und dann ein Array mit solchen Strukturen mit den entsprechenden Parametern initialisieren:

```
const int nMBAW=13;
TMandelbrotParam MBPar[nMBAW]=
 {
  {"Apfelmännchen",  -2,     1.25,   0.5,  -1.25},
  {"Seepferdchen",   -0.88,  0.18,  -0.70,  0.0},
  // ...
  {"Spirale",       -0.750,  0.105, -0.740, 0.095}
 };
```

5. Das sogenannte **Räuber-Beute-Modell** stellt einen quantitativen Zusammenhang zwischen einer Population von Räuber- und Beutetieren (z.B. Füchsen und Hasen) dar. Bezeichnet man die Anzahl der Räuber mit r und die der Beutetiere mit b, geht man von folgenden Voraussetzungen aus:

Die Zuwachsrate der Beutetiere soll über einen Faktor *zb* von ihrer Anzahl *b* und über einen Faktor *fb* von der Anzahl der möglichen Begegnungen von Räuber- und Beutetieren *r*b* abhängen; *zb* entspricht einer zusammengefassten Geburten- und Sterberate.

Die Räuber ernähren sich ausschließlich von den Beutetieren. Damit ist ihre Zuwachsrate nur von ihrer Sterberate *sr* und ihrem „Jagderfolg“ abhängig, der mit dem Faktor *fr* wieder von der Anzahl der möglichen Begegnungen zwischen Räubern und Beutetieren r*b abhängig sein soll:

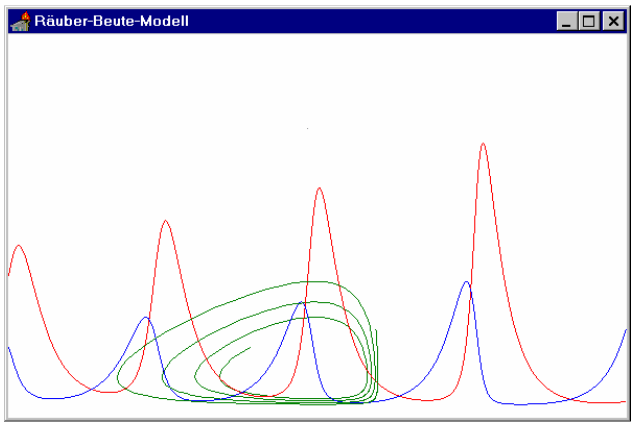
```
r = r_alt - sr*r_alt + fr*r_alt*b_alt;
b = b_alt + zb*b_alt - fb*r_alt*b_alt;
```

Die beiden Gleichungen sind ein nichtlineares System von Differenzengleichungen, das nach seinen Entdeckern auch als **Lotka-Volterra-System** bezeichnet wird. Es lässt sich zeigen, dass dieses System **nicht analytisch lösbar** ist.

Schreiben Sie ein Programm, das den Verlauf der Populationen grafisch darstellt. Stellen Sie in dieser Grafik außerdem das Phasendiagramm mit den Wertepaaren (r,b) im zeitlichen Ablauf dar. Mit den Faktoren

```
const double zb = 0.05,  fb = 0.001,
             sr = 0.05,  fr = 2*fb;
```

erhält man bei einer Skalierung der y-Werte auf den Bereich –10 bis 300:



Bei diesem System können schon relativ geringe Änderungen der Parameter zu einem völlig anderen Verlauf der Populationen führen.

10.14 Die Steuerung von MS-Office: Word-Dokumente erzeugen

10.15 Datenbank-Komponenten der VCL

10.15.1 Verbindung mit ADO-Datenbanken – der Connection-String

10.15.2 Tabellen und die Komponente *TDataSet*

10.15.3 Tabellendaten lesen und schreiben

10.15.4 Die Anzeige von Tabellen mit einem *DBGrid*

10.15.5 SQL-Abfragen

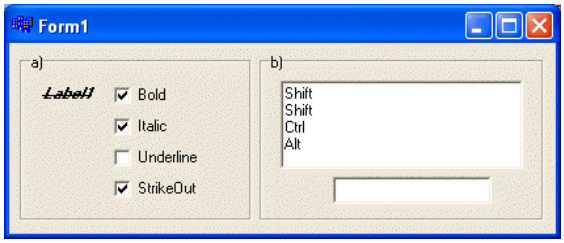
10.16 Internet-Komponenten

10.17 MDI-Programme

10.18 Die Klasse *Set*

Aufgabe 10.18

Schreiben Sie ein Programm, das ein Fenster anzeigt wie



a) Falls die CheckBox *fett* markiert wird, soll die Schrift im Label in der Schriftart *fett* angezeigt werden. Wenn diese Markierung entfernt wird, soll die Schriftart nicht mehr *fett* angezeigt werden. Entsprechend für die anderen CheckBoxen.
b) Beim Ereignis *OnKeyDown* im Edit-Fenster soll im Memo angezeigt werden, ob die Umschalt-, Alt- oder Strg-Tasten gedrückt wurde.

10.19 3D-Grafik mit OpenGL

10.19.1 Initialisierungen

10.19.2 Grafische Grundelemente: Primitive

10.19.3 Modelltransformationen

10.19.4 Vordefinierte Körper

10.19.5 Lokale Transformationen

10.19.6 Beleuchtungseffekte

10.19.7 Texturen

Aufgaben 10.19

Schreiben Sie ein Programm, das in einer Menüleiste die Optionen *Scene*, *Lights* und *Texture* anbietet.

1. Unter *Scene* sollen die folgenden Optionen angeboten werden:

   a) Nach der Auswahl „coordinate system“ soll ein Koordinatenkreuz gezeichnet werden. Verwenden Sie dazu eine Funktion *Arrow*, die einen Pfeil der Länge 1 vom Ursprung auf der x-Achse zeichnet. Erzeugen Sie aus drei solchen Pfeilen dasselbe Koordinatenkreuz wie mit der Funktion *CoordinateSystem* von Seite 1119.
   b) Ein Modell eines Wassermoleküls soll aus einer Kugel für das Sauerstoffatom und zwei Kugeln für die beiden Wasserstoffatome bestehen. Die Wasserstoffatome sollen mit den Sauerstoffatomen durch dünne Stäbchen (Zylinder) verbunden werden, die einen Winkel von 120 Grad einschließen.
   c) In einer Funktion *RotatingCubeAndCylinder* sollen ein Würfel und ein Kegel gezeichnet werden. Bei wiederholten Aufrufen dieser Funktion sollen sich diese Figuren unabhängig voneinander drehen.
   d) Die Stadt wie im Beispiel auf Seite 1121 soll nicht durch eine absolute Positionierung der Quader erzeugt werden, sondern durch einen Quader im Ursprung, der mit *glTranslate* an die richtige Position gesetzt wird.
   e) In einer Funktion *Robot* soll ein einfacher Roboter gezeichnet werden, der z.B. aus einer Kugel für den Kopf und Zylindern für den Körper und die Arme besteht. Bei wiederholten Aufrufen soll sich der Winkel der Arme zum Körper leicht ändern, so dass der Eindruck einer kontinuierlichen Bewegung entsteht.
   f) Ergänzen Sie das Solarsystem von Seite 1125 so, dass sich die Erde um eine Achse durch den Nord- und Südpol dreht. Außerdem soll sich ein zweiter Mond um die Erde drehen.
   g) Eine Funktion *AnimatedPaperplanes* soll einige Papierflieger (siehe Seite 1121) zeichnen. Ihre Position soll sich zwischen zwei aufeinander folgenden Aufrufen leicht verändern, so dass der Eindruck entsteht, dass sie (z.B. auf einer spiralförmigen Bahn) durch den Raum segeln.

2. Die jeweils dargestellte Szene soll mit den Pfeiltasten gedreht und verschoben werden können. Verwenden Sie dazu in der Funktion *FormKeyDown* geeignete Transformationen.

3. Unter der Menüoption *Lights* soll eine Lichtquelle aktiviert und deaktiviert werden können. Falls die Lichtquelle aktiviert ist, sollen in einem weiteren Formular die Position und Intensität der RGB-Werte des ambienten und spekulären Lichts sowie die ambienten, spekulären und diffusen Reflektionswerte der Oberflächen gesetzt werden können (z.B. mit einem ScrollBar). Dabei sollen die diffusen und spekulären Lichtwerte jeweils gleich sein, ebenso die ambienten und diffusen Materialwerte.

4. Unter der Option *Textures* sollen die folgenden Optionen angeboten werden:

a) Ein einfaches Bild wie das Schachbrett von Seite 1132 soll als Textur geladen werden.
b) Ein Windows-Bitmap soll mit einem *OpenPictureDialog* als Textur geladen werden.
c) Die unter a) oder b) geladene Textur soll bei einfachen Figuren wie einem Quadrat oder einer Kugel verwendet werden.
d) Ergänzen Sie das Solarsystem so, dass die Erde mit der unter a) oder b) gewählten Textur gezeichnet wird.
e) Schreiben Sie eine Funktion *TexturedCube*, die einen Würfel mit der aktuell gewählten Textur auf den Seitenflächen zeichnet. In einem genügend großen Würfel kann man sich dann wie in einem Raum bewegen.

10.20 Win32-Funktionen zur Dateibearbeitung

10.20.1 Elementare Funktionen

10.20.2 File-Sharing

10.20.3 Record-Locking

10.20.4 VCL-Funktionen zur Dateibearbeitung und *TFileStream*

Aufgaben 10.20

Verwenden Sie hier nur die Win32-Funktionen zur Dateibearbeitung.

1. Legen Sie eine einfache Datei (z.B. mit 10 aufeinander folgenden Werten des Datentyps *int*) an. Lesen Sie anschließend die Daten aus dieser Datei. Prüfen Sie nach jedem Aufruf einer Win32-Funktion, ob ein Fehler aufgetreten ist, und geben Sie dann mit einer Funktion wie *ShowLastError* eine Meldung aus.
2. Geben Sie einige dieser Fehlermeldungen aus, indem Sie mit *CreateFile* eine Datei mit einem unzulässigen Dateinamen öffnen oder aus einer nicht geöffneten bzw. nur zum Schreiben geöffneten Datei lesen.
3. Die Datei von Aufgabe 1 soll so geöffnet werden, dass andere Anwender sie lesen oder beschreiben können. Bieten Sie die Möglichkeit an, einzelne Datensätze der Datei zu sperren, und prüfen Sie vor dem Lesen oder Schreiben von Datensätzen, ob sie gesperrt ist. Testen Sie das Programm, indem sie es mehrfach starten.

10.21 Datenübertragung über die serielle Schnittstelle

10.21.1 Grundbegriffe

10.21.2 Standards für die serielle Schnittstelle: RS-232C bzw. V.24

10.21.3 Win32-Funktionen zur seriellen Kommunikation

Aufgabe 10.21

Schreiben Sie ein Programm, das Daten über ein Kabel von der seriellen Schnittstelle COM1 an die serielle Schnittstelle COM2 sendet.

Die übertragenen Daten sollen aus einem Edit-Fenster gelesen werden. Als Reaktion auf einen Timer soll geprüft werden, ob Daten empfangen wurden. Die empfangenen Daten sollen in einem Memo angezeigt werden.

11 Lösungen

Da die Zeilen in den Quelltexten der Lösungen manchmal länger sind als eine Druckseite breit ist, kommt es gelegentlich zu unschönen Formatierungen.

Angaben wie

```
// File: C:\Loesungen_CB2006\Kap_2\2.2\HaushSanU.cpp
```

bezeichnen die Datei mit den Quelltexten auf der CD, die im Buch enthalten ist. Auf der CD findet man diese Lösungen als Projekte für die Versionen 2006 und 2007 des Borland C++Builders. Die meisten Projekte können auch mit älteren Versionen des C++Builders bearbeitet werden (siehe dazu Tipp 11 in Abschnitt 1.6).

11.1 Utilities - Lösungen und Beispiele aus verschiedenen Kapiteln

11.1.1 TestUtils.h (Abschnitt 3.5)

```
// File: C:\Loesungen_CB2006\CppUtils\TestUtils.h

#ifndef TESTUTILS_H__
#define TESTUTILS_H__
#include "NearlyEqual.h"


// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.


bool log_Testcases=false;

// Die folgenden Funktionen sind so geschrieben, dass man sie einfach
// in einer Klasse zusammenfassen kann, indem man sie zwischen
// class { ... } einfügt.


int TestUtilsPassCount=0,TestUtilsFailCount=0;

void initTestUtils()
{
TestUtilsPassCount=0;
TestUtilsFailCount=0;
}

#include <vcl.h>

bool assertTrue(bool condition, AnsiString msg,
                TCustomMemo* Memo=Form1->Memo1)
```

```
{ // base routine to display one test case
if (!condition)
  {
    Memo->Lines->Add("Test failed: "+msg);
    ++TestUtilsFailCount;
  }
else
  {
    if (log_Testcases)
      Memo->Lines->Add("Test passed: "+msg);
    ++TestUtilsPassCount;
  }
return condition;
}

bool assertEqual_int(int f, int s, AnsiString  msg,
                     TCustomMemo* Memo=Form1->Memo1)
{ // tests one test case
return assertTrue(f==s,msg+"=="+IntToStr(f)+", expected: "+IntToStr(s),
                  Memo);
}

AnsiString BoolToString(bool b)
{
if (b) return "true";
else return "false";
}

bool assertEqual_bool(bool f, bool s, AnsiString msg,
                      TCustomMemo* Memo=Form1->Memo1)
{ // tests one test case
return assertTrue(f==s, msg+"=="+BoolToString(f)+", expected: "+
                  BoolToString(s));
}

bool assertEqual_double(double f, double s, AnsiString msg,
                TCustomMemo* Memo=Form1->Memo1, int digits=10)
{ // tests one test case
return assertTrue(NearlyEqual2(f,s,digits),msg+"=="+FloatToStr(f)+
                  ", expected: "+FloatToStr(s));
}

bool assertNotEqual_double(double f, double s, AnsiString msg,
                   TCustomMemo* Memo=Form1->Memo1, int digits=10)
{ // tests one test case
return assertTrue(!NearlyEqual2(f,s,digits),msg+"=="+FloatToStr(f)+
                          ", expected: "+FloatToStr(s));
}

bool assertEqual_AnsiString(AnsiString f, AnsiString s, AnsiString msg,
                                          TCustomMemo* Memo=Form1->Memo1)
{ // tests one test case
return assertTrue(f==s,msg+"=="+f+", expected: "+s);
}

bool assertEqual_Pointer(void* f, void* s, AnsiString msg,
                                          TCustomMemo* Memo=Form1->Memo1)
{ // tests one test case
return assertTrue(f==s,msg+"=="+IntToStr(f)+", expected: "+IntToStr(s));
}

bool assertEqual_Currency(Currency f, Currency s, AnsiString  msg,
                                          TCustomMemo* Memo=Form1->Memo1)
{ // tests one test case
return assertTrue(f==s,msg+"=="+IntToStr(f)+", expected: "+IntToStr(s),
                  Memo);
}

#include <string>
```

```
bool assertEqual_string(std::string f, std::string s, AnsiString  msg,
                                          TCustomMemo* Memo=Form1->Memo1)
{ // tests one test case
return assertTrue(f==s,msg+"=="+AnsiString(f.c_str())+", expected: "+
                                          AnsiString(s.c_str()), Memo);
}


// Die überladenden Funktionen lassen sich einfacher aufrufen:

bool assertEqual(bool f, bool s, AnsiString msg,
                 TCustomMemo* Memo=Form1->Memo1)
{
return assertEqual_bool(f, s, msg, Memo);
}

bool assertEqual(int f, int s, AnsiString msg,
                 TCustomMemo* Memo=Form1->Memo1)
{
return assertEqual_int(f, s, msg, Memo);
}

bool assertEqual(double f, double s, AnsiString msg,
                 TCustomMemo* Memo=Form1->Memo1, int digits=10)
{ // tests one test case
return assertEqual_double(f, s, msg, Memo, digits);
}

bool assertEqual(AnsiString f, AnsiString s, AnsiString msg,
                 TCustomMemo* Memo=Form1->Memo1)
{ // tests one test case
return assertEqual_AnsiString(f, s, msg, Memo);
}

bool assertEqual(std::string f, std::string s, AnsiString msg,
                 TCustomMemo* Memo=Form1->Memo1)
{ // tests one test case
return assertEqual_string(f, s, msg, Memo);
}

bool assertEqual(void* f, void* s, AnsiString msg,
                 TCustomMemo* Memo=Form1->Memo1)
{ // tests one test case
return assertEqual_Pointer(f, s, msg, Memo);
}

bool assertEqual(Currency f, Currency s, AnsiString msg,
                 TCustomMemo* Memo=Form1->Memo1)
{
return assertEqual_Currency(f, s, msg, Memo);
}

void testUtilsSummary(TMemo* Memo, bool result)
{
if (result)
  Memo->Lines->Add("All tests passed (total: "+
                   IntToStr(TestUtilsPassCount)+")");
else
  Memo->Lines->Add(IntToStr(TestUtilsFailCount)+
    " Tests failed from "+ IntToStr(TestUtilsFailCount+
    TestUtilsPassCount) );
}


#endif // TESTUTILS_H__
```

11.1.2 NearlyEqual.h (Abschnitt 3.6.6)

```
// File: C:\Loesungen_CB2006\CppUtils\NearlyEqual.h

#ifndef NEARLYEQUAL_H__
#define NEARLYEQUAL_H__
// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.


#include <cmath> // für fabs und pow
#include <limits> // für DBL_MAX und DBL_MIN

bool NearlyEqual(long double x,long double y,int p)
{ // true, falls sich x und y um weniger als 5 Einheiten in der (p+1)-ten
  // Stelle unterscheiden. p=10 ist für double Argumente normalerweise
  // ausreichend
double eps =1E-10;
if ((0<=p) && (p<=16)) eps=5*std::pow(0.1,p);
double diff=std::fabs(x - y);
if (x==0 || y==0) // NearlyEqual(x,0) ==> diff=|x|
  return diff<eps;
else // x!=0 and y!=0
  return (diff/std::fabs(x) <= eps) && (diff/std::fabs(y)  <= eps);
}

bool DefinitelyLess(long double x, long double y, int p)
{ // returns true, if x is definitely less then y
return (x<y) && (!NearlyEqual(x,y,p));
}

double SafeDivision(double x, double y)
{
const double MaxDouble=std::numeric_limits<double>::max();
const double MinDouble=std::numeric_limits<double>::min();
if   (y < 1 && x > y*MaxDouble) // Überlauf bei x/y
  return MaxDouble;
else if (y > 1 && x < y*MinDouble || x == 0) // Unterlauf
  return 0;                                  // bei x/y
else return x/y;
}

bool NearlyEqual2(long double x,long double y,int p)
{
double eps =1E-10;
if ((0<=p) && (p<=16)) eps=5*std::pow(0.1,p);
double diff=std::fabs(x - y);
if (x==0 || y==0) // NearlyEqual(x,0) ==> diff=|x|
  return diff<eps;
else // x!=0 and y!=0
  {
    double d1 = SafeDivision(diff, std::fabs(y));
    double d2 = SafeDivision(diff, std::fabs(x));
    return (d1 <= eps) && (d2 <= eps);
  }
}

#endif
```

11.1.3 State.h (Abschnitt 3.7.4)

```
// File: C:\Loesungen_CB2006\CppUtils\State.h

#ifndef STATE_H__
#define STATE_H__
// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.


/*
Die Makros STATE_1, STATE_2 usw. sollen auf einfache Weise die Definition
von Variablen ermöglichen, die den Wert von Variablen zwischenspeichern.

Das Makro STATE_1 erzeugt eine Struktur mit einem Element des Namens, der
als Argument für x1 übergeben wird, und dessen Datentyp das Argument für
T1 ist. Außerdem definiert es eine Variable mit dem Namen des Arguments
für s. Diese Variable wird mit dem Wert einer Variablen initialisiert,
deren Name das Argument für x1 ist.
Das Argument für x1 muss der Name einer zuvor definierten Variablen sein,
deren Datentyp zu T1 zuweisungskompatibel ist.

STATE_2 kann z.B. folgendermaßen verwendet werden:

  void vertausche(int& x, int& y)
  {
  STATE_2(s, int, x, int, y); // speichert den aktuellen Wert von x in s.x
  int h=x;                                          // und den von y in s.y
  x=y;
  y=h;
  Check_((y==s.x) and (x==s.y), "vertausche");
  }

*/

#define STATE_1(s, T1,x1)   \
    struct                  \
    {                       \
        T1 x1;              \
    } s={x1};

#define STATE_2(s, T1,x1,T2,x2)   \
    struct                        \
    {                             \
        T1 x1;                    \
        T2 x2;                    \
    } s={x1,x2};

#define STATE_3(s, T1,x1,T2,x2,T3,x3)  \
    struct                             \
    {                                  \
        T1 x1;                         \
        T2 x2;                         \
        T3 x3;                         \
    } s={x1,x2,x3};

// Weitere Makros für mehr als 3 Variable können entsprechend definiert
// werden.

/*
Aus der Funktion 'vertausche' von oben erzeugt der Präprozessor dann:

  void vertausche(int& x, int& y)
  {
  struct { int x; int y; } s={x,y};;
  int h=x;
  ...
*/
```

```
// like in <assert.h>
#ifdef NDEBUG
#define CheckCondition(p)   ((void)0)
#else
#include <stdexcept> // required for logic_error in CheckCondition

bool CheckCondition_errorflag=false;

void CheckCondition(bool condition, AnsiString msg, int action=1)
{ // this is very simple!
// AnsiString is only available in C++Builder, use some other string
// type for other compilers/platforms.
if (!condition)
  { // take a suitable action, e.g. something like
    if (action==1)
      // This is specific for C++Builder:
      Form1->Memo1->Lines->Add(AnsiString("CheckCondition failed: ")+msg);
      // Modify for other forms or compilers, e.g.
      //   Form2->Memo3->Lines->Add(AnsiString("CheckCondition failed: ")+msg);
      // or
      //   cout<<"CheckCondition failed: "<<msg<<endl;
    else if (action==2) // throw exception
      throw std::logic_error("CheckCondition failed: ");
    else if (action==3)
      ; // write into logfile
    else if (action==4)
      ; // ignore
    else // set errorflag
      CheckCondition_errorflag=true;
  }
}
#endif // NDEBUG
#endif // STATE_H__
```

11.1.4 Trace.h (Aufgabe 3.15)

```
// File: C:\Loesungen_CB2006\CppUtils\Trace.h

// --------------- Aufgabe 3.15 ----------------------------
// d)
#ifndef TRACE_H__
#define TRACE_H__


// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.


// c)
#ifdef NOTRACE
  #define OPEN_TRACELOG(x1)
  #define TRACE1(x1)          ((void)0)
  #define TRACE2(x1,x2)       ((void)0)
  #define TRACE3(x1,x2,x3)    ((void)0)
#else
  #include <fstream>
// b)
  #define OPEN_TRACELOG(filename)  std::ofstream out_(filename)
// a)
  #define TRACE1(x1) out_<<__FILE__<< ", line: " <<\
     __LINE__ <<", function: "<< __FUNC__<<", "\
     <<#x1<<"='"<<x1<<"'"<<std::endl;
```

```
  #define TRACE2(x1,x2) out_<<__FILE__<< ", line: " <<\
     __LINE__ <<", function: "<< __FUNC__<<", "\
     <<#x1<<"='"<<x1<<"' "<<#x2<<"='"<<x2<<"'"<<std::endl;

  #define TRACE3(x1,x2,x3) out_<<__FILE__<< ", line: " <<\
     __LINE__ <<", function: "<< __FUNC__<<", "<<\
     #x1<<"='"<<x1<<"' "<<#x2<<"='"<<x2<<"' "<<#x3<<"='"<<x3<<"'"<<std::endl;
#endif

#endif  // trace_cpp
```

11.1.5 StringUt.h (Aufgaben 3.12, 4.1, 7, 9.1)

```
// File: C:\Loesungen_CB2006\CppUtils\StringUt.h

#ifndef STRINGUT_H__
#define STRINGUT_H__
// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.


#include <string>
using std::string;
#include <vector>
using std::vector;

string getline_BUG_workaround(std::string z)
{
// BCB5 Bug: When using the global function getline(istream, std::string)
// to fill a string, it adds an extra null terminator to the string. This
// null terminator is separate from the null terminator for the
// underlying char *.

// Dieser Bug ist im Service-Pack zum C++Builder5 behoben.

int n=z.length();
if (n>0)
 {
  char x=z[n-1];
  if (x=='\0')
    z.erase(n-1,1);
  // nur zum Testen:
  n=z.length();
  x=z[n-1];
 }
return z;
}

// --------------- Aufgabe 3.12.10, 5  ---------------------

inline int min3(int x,int y, int z)
{
int m=x;
if (y<m) m=y;
if (z<m) m=z;
return m;
}

#include <fstream>
```

```
#define USE_MATRIX_DEF 3 //  use 1, 2 or 3; all three should work
                         //  In C++Builder 2006 only 3 works
#define USE_EDITDISTANCE_STRING 1 // use 0 or 1; both work

#if USE_EDITDISTANCE_STRING==0
double StringSimilarityEditDistance(const char* s1, const char* s2)
{ // Levenshtein Edit Distance
int l1=strlen(s1);
int l2=strlen(s2);
#else
double StringSimilarityEditDistance(const string& s1, const string& s2)
{
int l1=s1.length();
int l2=s2.length();
#endif

#if USE_MATRIX_DEF==1
typedef vector<vector<int> > Matrix;
Matrix d(l1+1,l2+1);
#elif USE_MATRIX_DEF==2
vector<vector<int> > d(l1+1,l2+1); // alternative to Matrix
#else
const int Max=10; // Can be bigger, 10 only to make testing easier
int d[Max+1][Max+1]; // another alternative to Matrix, a little faster
if (l1>Max) l1=Max;
if (l2>Max) l2=Max;
#endif

for (int j=0; j<=l2 ; j++)
    d[0][j] = j;
for (int j=0; j<=l1 ; j++)
    d[j][0] = j;

for (int i=1; i<=l1; i++)
  for (int j=1; j<=l2; j++)
    {
      int cost=0;
      if (s1[i-1] != s2[j-1]) cost=1;
      d[i][j] = min3( d[i-1][j-1] + cost, d[i][j-1]+1, d[i-1][j]+1 );
    }
//         Java-Applet         zum         Vergleich:         http://www-igm.univ-
mlv.fr/~lecroq/seqcomp/node2.html
//                            http://www.merriampark.com/ld.htm
//#define WRITE_MATRIX_TO_FILE 1
#ifdef WRITE_MATRIX_TO_FILE
   static char x='0';
   x++;
   if (x>'9') x='9';
   string fn="d:\\lv.txt";
   fn=fn+x;
   ofstream of(fn.c_str());
   of<<s1<<endl;
   of<<s2<<endl;
   for (int i=0; i <= l1; i++)
   {
      for (int j=0; j <= l2; j++)
         of<<d[i][j]<<"\t";
      of<<endl;
   }
   of<<"ld="<<d[l1][l2]<<endl;
#endif  // WRITE_MATRIX_TO_FILE

int lm=l1; // lm=max(l1,l2)
if (l2>l1) lm=l2;
if (lm==0) return 1; // should two empty strings be considered equal?
else return( 1- d[l1][l2]/(0.0+lm) );
}
```

```
#include "TestUtils.h"
bool test_EditDistance()
{
bool result=true;

double ld1=StringSimilarityEditDistance("receieve","receieve"); // 1-0/8=1
if (!assertEqual(ld1, 1-0.0/8,"EditDistance ld1")) result=false;
double ld2=StringSimilarityEditDistance("receieve","receive");  // 1-1/8=0.875
if (!assertEqual(ld2, 1-1.0/8,"EditDistance ld2")) result=false;
double ld3=StringSimilarityEditDistance("receieve","receiver"); // 1-2/8=0.75
if (!assertEqual(ld3, 1-2.0/8,"EditDistance ld3")) result=false;
double ld4=StringSimilarityEditDistance("receieve","retrieve"); // 1-2/8=0.75
if (!assertEqual(ld4, 1-2.0/8,"EditDistance ld4")) result=false;
double ld5=StringSimilarityEditDistance("receieve","reactive"); // 1-3/8
if (!assertEqual(ld5, 1-3.0/8,"EditDistance ld5")) result=false;

// http://www.cs.mu.oz.au/385/pp/lecture16.pdf
// http://www.merriampark.com/ld.htm
double ld6=StringSimilarityEditDistance("vintner","writers"); // 1-5/7,
if (!assertEqual(ld6, 1-5.0/7,"EditDistance ld6")) result=false;
double ld7=StringSimilarityEditDistance("gumbo","gambol");    // 1-2/6,
if (!assertEqual(ld7, 1-2.0/6,"EditDistance ld7")) result=false;
double ld8=StringSimilarityEditDistance("intention","execution"); // 1-5/9,
if (!assertEqual(ld8, 1-5.0/9,"EditDistance ld8")) result=false;

return result;
}


// --------------- Aufgabe 4.1, 1 --------------------------

#include <sstream>
int stringToInt(const string& s, bool& success)
{
std::istringstream is(s);
int result=0;
is>>result;
success=(s.length()==is.tellg());
return result;
}

double stringToDouble(const string& s, bool& success)
{
std::istringstream is(s);
double result=0;
is>>result;
success=(s.length()==is.tellg());
return result;
}

string tostring(int x)
{
std::ostringstream os;
os<<x;
return os.str();
}

string tostring(double x)
{
std::ostringstream os;
os<<x;
return os.str();
}


#ifdef VCL_H
bool assertEqual(int fi, int si, bool fb, bool sb, AnsiString msg,
                                            TCustomMemo* Memo=Form1->Memo1)
{
```

```
return assertTrue((fi==si) && (fb==sb),msg, Memo);
}
#endif // VCL_H

bool test_stringToint()
{
bool result = true;
bool success;
int x=stringToInt("123",success);    // result=123, t=3
if (!assertEqual(123, x, true, success, "stringToInt(123)"))
  result=false;

x=stringToInt("123a",success);   // result=123, t=3
if (!assertEqual(123, x, false, success, "stringToInt(123a)"))
  result=false;

x=stringToInt("a123",success);   // result=0,   t=-1
if (!assertEqual(0, x, false, success, "stringToInt(a123)"))
  result=false;

x=stringToInt("12 34",success);  // result=12,  t=2
if (!assertEqual(12, x, false, success, "stringToInt(12 34)"))
  result=false;

x=stringToInt(" -1234",success); // result=-1234, t=6
if (!assertEqual(-1234, x, true, success, "stringToInt( -1234)"))
  result=false;

return result;
}

#ifdef VCL_H
bool assertEqual(double fi, double si, bool fb, bool sb, AnsiString msg,
                                          TCustomMemo* Memo=Form1->Memo1)
{
return assertEqual(fi, si,msg);
}
#endif // VCL_H

bool test_stringToDouble()
{
bool result = true;
bool success;
double x=stringToDouble("123.45",success);    // result=123, t=3
if (!assertEqual(123.45, x, true, success, "stringToDouble(123.45)"))
  result=false;

x=stringToDouble("-123.45",success);    // result=123, t=3
if (!assertEqual(-123.45, x, true, success, "stringToDouble(-123.45)"))
  result=false;

x=stringToDouble("0.12345",success);    // result=123, t=3
if (!assertEqual(0.12345, x, true, success, "stringToDouble(0.12345)"))
  result=false;

x=stringToDouble("-0.12345",success);    // result=123, t=3
if (!assertEqual(-0.12345, x, true, success, "stringToDouble(-0.12345)"))
  result=false;

x=stringToDouble("-0.0000",success);    // result=123, t=3
if (!assertEqual(0.0, x, true, success, "stringToDouble(-0.0000)"))
  result=false;

x=stringToDouble("-0.12.34",success);    // result=123, t=3
if (!assertEqual(-0.12, x, false, success, "stringToDouble(-0.0000)"))
  result=false;

return result;
}
```

```
bool test_tostring()
{
bool result = true;
if (!assertEqual(tostring(0),"0", "tostring(0)")) result=false;
if (!assertEqual(tostring(1),"1", "tostring(1)")) result=false;
if (!assertEqual(tostring(-1),"-1", "tostring(-1)")) result=false;
if (!assertEqual(tostring(12),"12", "tostring(12)")) result=false;
if (!assertEqual(tostring(-12),"-12", "tostring(-12)")) result=false;

if (!assertEqual(tostring(0.0),"0", "tostring(0.0)")) result=false;
if (!assertEqual(tostring(1.0),"1", "tostring(1)")) result=false;
if (!assertEqual(tostring(-1.0),"-1", "tostring(-1)")) result=false;
if (!assertEqual(tostring(1.2),"1.2", "tostring(12)")) result=false;
if (!assertEqual(tostring(-1.2),"-1.2", "tostring(-12)")) result=false;

return result;
}


// --------------- Aufgabe 7, 4. ---------------------------

#include <sstream>
int stringToInt(const string& s)
{
std::istringstream is(s);
int result=0;
is>>result;
if (s.length()!=is.tellg())
  throw std::invalid_argument("Cannot convert '"+s+"' to int");
return result;
}

double stringToDouble(const string& s)
{
std::istringstream is(s);
double result=0;
is>>result;
if (s.length()!=is.tellg())
  throw std::invalid_argument("Cannot convert '"+s+"' to int");
return result;
}



// --------------- Aufgabe 4.1, 4. -------------------------

double StringSimilarityNGram(std::string s1, std::string s2)
{
int l1=s1.length();
int l2=s2.length();
for (int i=0; i<l1; i++) s1[i]=toupper(s1[i]);
for (int i=0; i<l2; i++) s2[i]=toupper(s2[i]);

int N=3; // 2: digram, 3: trigram, else Ngram
if      ((l1==0) && (l2==0)) return 1;
else if ((l1==0) || (l2==0)) return 0;
else if ((l1==1) || (l2==1)) N=1; //
else if ((l1==2) || (l2==2)) N=2;
else if ( (l1+l2)/2.0 +1 < 6 ) N=2; // for short strings digrams
else N=3; // for longer strings trigrams

// common variant: insert N blanks at the beginning
// string N_blanks=string(N,' ');
// s1=N_blanks+s1;
// s2=N_blanks+s2;
int sum=0;
for (int i=0; i<l2-N+1; i++)
  {
```

```
    using std::string;
    string sub=s1.substr(i,N);
    if (s2.find(sub)!=string::npos) sum++;
  }
double n=l1-N+1 + l2-N+1 - sum;
return sum/n;
}

bool test_StringSimilarityNGram()
{
bool result=true;

double n1=StringSimilarityNGram("receieve","receieve");  // 6/6=1
if (!assertEqual(n1, 1.0,"NGram n1")) result=false;
double n2=StringSimilarityNGram("receieve","receive");   // 3/8=0.375
if (!assertEqual(n2, 3.0/8,"NGram n2")) result=false;
double n3=StringSimilarityNGram("receieve","receiver");  // 3/9=0.333
if (!assertEqual(n3, 3.0/9,"NGram n3")) result=false;
double n4=StringSimilarityNGram("receieve","retrieve");  // 2/10=0.2
if (!assertEqual(n4, 0.2,"NGram n4")) result=false;
double n5=StringSimilarityNGram("receieve","reactive");  // 0/12=0
if (!assertEqual(n5, 0.0,"NGram n5")) result=false;
double n6=StringSimilarityNGram("","");                  // 6/6=1
if (!assertEqual(n6, 1.0,"NGram n6")) result=false;
double n7=StringSimilarityNGram("r","r");                // 6/6=1
if (!assertEqual(n7, 1.0,"NGram n7")) result=false;
return result;
}

// --------------- Aufgabe 4.2.3, 4 ------------------------

// a)
std::vector<string> tokenize(const string& s, string separators=";.,- ")
{ // ";.,- " ist ein Default Argument für den Parameter separators
  // Damit kann tokenize ohne Argument für den zweiten Parameter aufgerufen
  // werden. separators kann auch als lokale Variable definiert werden.

std::vector<string> result;
string::size_type pos_tok=s.find_first_not_of(separators);
while (pos_tok!=string::npos)
  { // token gefunden ab Position pos_tok
    string::size_type  pos_sep=s.find_first_of(separators,pos_tok);
    if (pos_sep==string::npos)
      pos_sep=s.length();
    string token=s.substr(pos_tok,pos_sep-pos_tok);

    result.push_back(token);
    pos_tok=s.find_first_not_of(separators,pos_sep+1);
  }
return result;
}

// b)
#ifdef VCL_H
std::vector<AnsiString> tokenize(const AnsiString& s, AnsiString separators=";.,-
")
{
using std::vector;
string s1=s.c_str();
string sep1=separators.c_str();
vector<string> sres=tokenize(s1,sep1);
vector<AnsiString> result;
for (int i=0; i<sres.size();i++)
  result.push_back(sres[i].c_str());
return result;
}
#endif // VCL_H
```

```
// c)


vector<string> make_string_vector(string s1="", string s2="", string s3="")
{
vector<string> result;
if (s1!="")
  result.push_back(s1);
if (s2!="")
  result.push_back(s2);
if (s3!="")
  result.push_back(s3);
return result;
}

#ifdef VCL_H
bool assertEqual(vector<string> f, vector<string> s, AnsiString msg,
                                          TCustomMemo* Memo=Form1->Memo1)
{
return assertTrue((f==s),msg, Memo);
}

bool assertEqual(vector<AnsiString> f, vector<AnsiString> s, AnsiString msg,
                                          TCustomMemo* Memo=Form1->Memo1)
{
return assertTrue((f==s),msg, Memo);
}
#endif // VCL_H

bool test_tokenize()
{
bool result=true;
const int nTests=10;
string s[nTests];
vector<string> st[nTests];
s[0]="";              st[0]=make_string_vector();
s[1]=" ";             st[1]=make_string_vector();
s[2]="123";           st[2]=make_string_vector("123");
s[3]=";123";          st[3]=make_string_vector("123");
s[4]="123;";          st[4]=make_string_vector("123");
s[5]=";123;";         st[5]=make_string_vector("123");
s[6]="456 ab.xy";     st[6]=make_string_vector("456","ab","xy");
s[7]=" 456 ab.xy";    st[7]=make_string_vector("456","ab","xy");
s[8]="456 ab.xy;";    st[8]=make_string_vector("456","ab","xy");
s[9]="456 ab.xy;";    st[9]=make_string_vector("456","ab","xy");
// Dieselben Testfälle wie in Aufgabe 4.1
// kein token:
// s[0]: leerer String, keine Ausgabe
// s[1]: nur Separator, keine Ausgabe
// ein token:
// s[2]: nur token, Ausgabe "123"
// s[3]: Separator am Anfang, Ausgabe "123"
// s[4]: Separator am Ende, Ausgabe "123"
// s[5]: Separator am Anfang und Ende, Ausgabe "123"
// mehrere tokens, Separatoren in der Mitte:
// s[6]: kein Separator am Anfang und Ende, Ausgabe "456", "ab", "xy"
// s[7]: Separator am Anfang, Ausgabe "456", "ab", "xy"
// s[8]: Separator am Ende, Ausgabe "456", "ab", "xy"
// s[9]: Separator am Anfang und Ende "456", "ab", "xy"

for (int i=0; i<nTests; i++)
if (!assertEqual(tokenize(s[i]), st[i],"test_tokenize i="+IntToStr(i)))
  result=false;

#ifdef VCL_H
// dieselben Tests mit AnsiStrings
AnsiString a[nTests];
for (int i=0; i<nTests; i++)
  a[i]=s[i].c_str();
```

```
vector<AnsiString> at[nTests];
for (int i=0; i<nTests; i++)
  for (int j=0; j<st[i].size(); j++)
    at[i].push_back(st[i][j].c_str());
for (int i=0; i<nTests; i++)
  if (!assertEqual(tokenize(a[i]), at[i],"test_tokenize i="+IntToStr(i)))
    result=false;
#endif // VCL_H

return result;
}


// --------------- Aufgabe 4.1, 5. longest common subsequence  -------

int lcsLength(const char* s1, const char* s2, string& lcs)
{
const int Max=100;
int lc[Max][Max];
int b[Max][Max];

int m=strlen(s1);
int n=strlen(s2);
if (m>=Max) m=Max-1;
if (n>=Max) n=Max-1;

enum {Up, Left, Diag};

for(int i=0; i<=m; i++)
  {
    lc[i][0]=0;
    lc[0][i]=0;
  }

for(int i=1; i<=m; i++)
  for(int j=1; j<=n; j++)
    if( s1[i-1]==s2[j-1])
      {
        lc[i][j]=lc[i-1][j-1]+1;
        b[i][j]=Diag;
      }
    else if (lc[i-1][j]>=lc[i][j-1])
      {
        lc[i][j]=lc[i-1][j];
        b[i][j]=Up;
      }
    else
      {
        lc[i][j]=lc[i][j-1];
        b[i][j]=Left;
      }

int i=m;
int j=n;
lcs="";
while (i>0 && j>0)
  {
    if (b[i][j]==Up) i--;
    else if (b[i][j]==Left) j--;
    else // Diag
      {
        lcs=s1[i-1]+lcs;
        i--;
        j--;
      }
  }
return lc[m][n];

}
```

```
bool test_lcs()
{
bool result=true;
// Java Applet: http://ranger.uta.edu/~cook/aa/lectures/applets/lcs/lcs.html
// Java Applet: http://www-igm.univ-mlv.fr/~lecroq/seqcomp/node4.html
string s1,s2,s3,s4;

int lcs1=lcsLength("ABCBDAB", "BDCABA", s1); // 4, BCBA, Cormen
// Dies ist das Ergebnis von Cormen; Das Java-Applet hat ein anderes Ergebnis
if (!assertEqual(s1,"BCBA","lcs s1")) result=false;
// if (!assertEqual(s1,"BDAB","lcs s1")) result=false;
int lcs2=lcsLength("AGCGA", "CAGATAGAG",s2); // 4, AGGA, Crockemere
if (!assertEqual(s2,"AGGA","lcs s2")) result=false;
int lcs3=lcsLength("ABCBDAB","BDCABA",s3); // 4, BCBA, Java Applet
if (!assertEqual(s3,"BCBA","lcs s3")) result=false;
int lcs4=lcsLength("","",s4); // 4, BCBA, Java Applet
if (!assertEqual(s4,"","lcs s4")) result=false;
return result;
}



#include "timeutils.h"
void Benchmarks()
{
CHRTimer tl;
int l1=0,l2=0;
tl.Start(); // kann auch ausgelassen werden
for (int i=0; i<100; i++)
  {
    l1+=StringSimilarityEditDistance("receieve","retrieve");
    l2+=StringSimilarityEditDistance("receieve","reactive");
  }
tl.End();
Form1->Memo1->Lines->Add(tl.TimeStr("Levenstein: "));
Form1->Memo1->Lines->Add(IntToStr(l1+l2));
//return result;
}


bool test_StringUtAll()
{
bool result=true;
if (!test_stringToint()) result=false;
if (!test_stringToDouble()) result=false;
if (!test_tostring()) result=false;
if (!test_tokenize()) result=false;
if (!test_StringSimilarityNGram()) result=false;
if (!test_EditDistance()) result=false;
if (!test_lcs()) result=false;

return result;
}

// --------------- Aufgaben 9.1 ----------------------------

// --------------- Aufgabe 2 -------------------------------
#include <sstream>
#include <string>

// a)
using std::string;

template<typename T>
string to_string(T x)
{
std::ostringstream os;
os<<x;
```

```
return os.str();
}

template<typename T>
T string_to(string s)
{
T result;
std::istringstream is(s);
is>>result;
if (s.length()!=is.tellg())
  throw std::invalid_argument("Cannot convert '"+s+"' to int");
return result;
}

// b)
// Eine explizite Spezialisierung wäre nicht so gut
string to_string(bool x)
{
if (x) return "true";
else return "false";
}

template<>
bool string_to<bool>(string x, bool& success)
{
success=true;
if (x=="true") return true;
else if (x=="false") return false;
else
  {
    success=false;
    return false;
  }
}

// c)
// Eine explizite Spezialisierung wäre nicht so gut
string to_string(AnsiString x)
{
return x.c_str();
}

template<>
AnsiString string_to<AnsiString>(string x)
{
return x.c_str();
}

#endif // STRINGUT_H__
```

11.1.6 CompilerId.h (Abschnitt 3.22.3)

```
// File: C:\Loesungen_CB2006\CppUtils\CompilerId.h

#ifndef COMPILERID_H
#define COMPILERID_H
// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.


// siehe Abschnitt 3.22.3
```

```
AnsiString CompilerId()
{
AnsiString v=IntToHex(__BCPLUSPLUS__,4);
AnsiString id=AnsiString("BCB ")+v[2]+"."+v[3]+"."+v[4];
#ifdef __CODEGUARD__
id+=" CodeGuard ";
#endif

#ifdef _DEBUG
id+=" DEBUG ";
#else
id+=" NoDEBUG ";
#endif

#if USE_STLPORT      // C++Builder 6
id+=" STLPort ";
#endif

#if _USE_OLD_RW_STL  // C++Builder 5
id+=" RW-STL  ";
#endif
return id;
}

#endif // COMPILERID_H
```

11.1.7 KBDecl.h (Aufgabe 4.4)

```
// File: C:\Loesungen_CB2006\CppUtils\KBDecl.h

#ifndef KBDECL_H__
#define KBDECL_H__
// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.


struct CDatum {
  int Tag;
  int Monat;
  int Jahr;
};

const int MaxLaengeNameInhaber=20;

struct Kontobewegung {
  int KontoNr;
  char NameInhaber[MaxLaengeNameInhaber]; // nur für einige Dateiaufgaben
                   // als Array - sonst ist string bzw. AnsiString besser
  CDatum Datum;
  char BewArt;
  double Betrag; // oder: Currency Betrag;
};


AnsiString KBToString(const Kontobewegung& K)
{
return IntToStr(K.KontoNr)+" "+AnsiString(K.NameInhaber)+"  "+
  AnsiString(K.Datum.Tag)+"."+AnsiString(K.Datum.Monat)+"."+
  AnsiString(K.Datum.Jahr)+"  "+
AnsiString(K.BewArt)+"  "+FloatToStr(K.Betrag);
}
```

```
// --------------- Aufgaben 4.6.1.2 c) ---------------------

bool Schaltjahr(int Jahr)
{
return (Jahr%4==0)||((Jahr%100==0)&&(Jahr%400==0));
}

// ----- Funktionen für Zufallsdaten ----

  // für C++Builder 3 notwendig
  #include <stdlib.h>

int Zuf(int a, int b)
{
return a + rand()%(b-a+1);
}

int eindeutige_Kontonummer=0;

Kontobewegung Zufalls_KB()
{
const int Max_n = 10; // Anzahl der Vor- bzw. Nachnamen
AnsiString Nachname[Max_n] =
           {"Duestrip, ","Sander, ",  "König, ",  "Blond, ",
            "Schluck, ", "Parker, ",  "Kannt, ",  "Pascal, ",
            "Coldrain, ","Prince, "};
AnsiString Vorname[Max_n] =
           {"Daniel",    "Alek",      "Q.",       "James",
            "Donald",    "Karl",      "Imanuel",  "Nikolausi",
            "John",      "Charlie"};


Kontobewegung k;
memset(&k, 0, sizeof(k));

eindeutige_Kontonummer++;
k.KontoNr = eindeutige_Kontonummer;
// Für manche Aufgaben ist es sinnvoller, wenn man eindeutige Datensätze
// hat. Bei anderen sind zufällige Schlüsselbegriffe sinnvoller. Je nach
// Bedarf kann man die nächste Anweisung auskommentieren:
k.KontoNr = Zuf(1000,1099);
int z = k.KontoNr % 100;
AnsiString Name=Nachname[z/Max_n]+Vorname[z%Max_n];
strcpy(k.NameInhaber,Name.c_str());

k.Datum.Jahr = Zuf(2000,2001);
k.Datum.Monat = Zuf(1,12);
if ((k.Datum.Monat == 4) ||
    (k.Datum.Monat == 6) ||
    (k.Datum.Monat == 9) ||
    (k.Datum.Monat ==11)) k.Datum.Tag = Zuf(1,30);
else if (k.Datum.Monat == 2)
  {
    if (Schaltjahr(k.Datum.Jahr))
         k.Datum.Tag = Zuf(1,28);
    else k.Datum.Tag = Zuf(1,29);
  }
else k.Datum.Tag = Zuf(1,31);
if (Zuf(0,1)==0) k.BewArt = '+';
else k.BewArt = '-';
k.Betrag = Zuf(1,30000)/100.0; // ".0" für Gleitkomma-Wert
return k;
}

#endif
```

11.1.8 KBForms.h (Aufgabe 4.4)

```
// File: C:\Loesungen_CB2006\CppUtils\KBForms.h

#ifndef KBFORMS_H__
#define KBFORMS_H__
// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.


// Für diese Funktion muß das Formular
// im Objektinspektor den Namen "TKBForm1" bekommen:
//

#include "KBDecl.h"

void KBToForm(Kontobewegung K,TKBForm1* Form1)
{ // überträgt die Daten von K in das Formular Form1
Form1->EKontoNr->Text=IntToStr(K.KontoNr);
Form1->EName->Text=K.NameInhaber;
// Entweder
Form1->EDatum->Text=IntToStr(K.Datum.Tag)+"."+
                    IntToStr(K.Datum.Monat)+"."+
                    IntToStr(K.Datum.Jahr);
// oder
unsigned short year= K.Datum.Jahr;
unsigned short month=K.Datum.Monat;
unsigned short day=  K.Datum.Tag;
TDateTime d=TDateTime(year, month, day);
// Form1->EDatum->Text=d.DateString();

Form1->EBewart->Text=K.BewArt;
Form1->EBetrag->Text=FloatToStr(K.Betrag);
}

Kontobewegung FormToKB(TKBForm1* Form1)
{ // überträgt die Daten des Formulars Form1 in den Funktionswert
Kontobewegung K;
K.KontoNr = StrToInt(Form1->EKontoNr->Text);

// Vorsicht: Dieses strcpy ist nicht gut, da
//           hier die Gefahr von buffer overruns besteht
strcpy(K.NameInhaber,Form1->EName->Text.c_str());

// Entweder wie in Aufgabe 3.13, 1.
  AnsiString Datum = Form1->EDatum->Text;
  AnsiString tstr  = Datum.SubString(1,Datum.Pos('.')-1);
  K.Datum.Tag = StrToInt(tstr);
  Datum.Delete(1,Datum.Pos('.'));
  int p = Datum.Pos('.');
  AnsiString mstr  = Datum.SubString(1,p-1);
  K.Datum.Monat = StrToInt(mstr);
  AnsiString jstr  = Datum.SubString(p+1,Datum.Length()-p+1);
  K.Datum.Jahr = StrToInt(jstr);
// oder
  TDateTime d=StrToDate(Form1->EDatum->Text);
  unsigned short year, month, day;
  d.DecodeDate(&year, &month, &day);
  K.Datum.Tag=day;
  K.Datum.Monat=month;
  K.Datum.Jahr=year;
// Ende oder

Form1->EBewart->Text.Trim();
if (Form1->EBewart->Text.Length() >= 1)
    K.BewArt =Form1->EBewart->Text[1];
```

```
K.Betrag = StrToCurr(Form1->EBetrag->Text);
return K;
}

void ClearKBForm(TKBForm1* Form1)
{
Form1->EKontoNr->Text="";
Form1->EName->Text="";
TDateTime d=d.CurrentDate();
Form1->EDatum->Text=d.DateString();
Form1->EBewart->Text="-";
Form1->EBetrag->Text=FloatToStr(123.45);
}

#endif // KBFORMS_H__
```

11.1.9 FixedP64.h (Aufgabe 6.2.6, 5.)

```
// File: C:\Loesungen_CB2006\CppUtils\FixedP64.h

#ifndef FixedP64H
#define FixedP64H
// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.



#include <string>
#include <sstream>

class FixedP64 {
  long long i;
 public:
  FixedP64(): i(0) { }
  FixedP64(int i_): i(i_*10000) { }
  FixedP64(double d_): i((d_+0.000005)*10000) { }
  FixedP64(const FixedP64& f_): i(f_.i) { }

  std::string toStr()
  {
  using namespace std;
  ostringstream out;
  out<<i;
  string s=out.str();
  while (s.length()<5) s = "0"+s;
#if Test_SECURE_SCL
  s.insert(s.length()-3,",");
#endif
  return s;
  }

  FixedP64& operator+=(const FixedP64& a)  {i+=a.i; return *this;}
  FixedP64& operator-=(const FixedP64& a)  {i-=a.i; return *this; }
  FixedP64& operator*=(const FixedP64& a)
    {i*=a.i; i /= 10000; return *this; }
  FixedP64& operator/=(const FixedP64& a)  {i*=10000; i/=a.i; return *this; }

  friend bool operator<=(const FixedP64& a,const FixedP64& b);
};

bool operator<=(const FixedP64& a,const FixedP64& b)
{
```

```
return a.i<=b.i;
}

FixedP64 operator+(const FixedP64& a,const FixedP64& b)
{
FixedP64 x=a;
return x+=b;
}

FixedP64 operator*(const FixedP64& a,const FixedP64& b)
{
FixedP64 x=a;
return x*=b;
}

FixedP64 operator-(const FixedP64& a,const FixedP64& b)
{
FixedP64 x=a;
return x-=b;
}

FixedP64 operator/(const FixedP64& a,const FixedP64& b)
{
FixedP64 x=a;
return x/=b;
}

#endif
```

11.1.10 BinStream.h (Aufgabe 9.2)

```
// File: C:\Loesungen_CB2006\CppUtils\BinStream.h

#ifndef _BIN_STREAM_H_
#define _BIN_STREAM_H_
#include <fstream>
using std::fstream;
using std::ios_base;

template<typename T> class binary_fstream: public fstream
{
 public:
  // fstream constructor: fstream();
  binary_fstream():fstream() {}
  // fstream constructor: fstream(const char* s,
  //            ios_base::openmode mode=ios_base::in|ios_base::out);
  binary_fstream(const char* s,
      ios::openmode mode=ios::in|ios::out):
          fstream(s,mode|ios::binary) {}

  ostream& write_bin(const T& t)
  { // fstream: write(const char_type* s, streamsize n);
  return fstream::write((char*)&t,sizeof(T));
  }

  istream& read_bin(T& t)
  { // fstream: read(char_type* s, streamsize n);
  return fstream::read((char*)&t,sizeof(T));
  }
};


template<typename T> class binary_ifstream: public ifstream
{
```

```
 public:
  binary_ifstream():ifstream() {}
  binary_ifstream(const char* s):ifstream(s,ios::binary) {}

  istream& read_bin(T& t)
  {
  return ifstream::read((char*)&t,sizeof(T));
  }
};

template<typename T> class binary_ofstream: public ofstream
{
 public:
  binary_ofstream():ofstream() {}
  binary_ofstream(const char* s):ofstream(s,ios::binary) {}

  ostream& write_bin(const T& t)
  {
  return ofstream::write((char*)&t,sizeof(T));
  }
};

#endif  // _BIN_STREAM_H_
```

11.1.11 TimeUtils.h (Abschnitt 10.10)

```
// File: C:\Loesungen_CB2006\CppUtils\TimeUtils.h

#ifndef TIMEUTILS_H
#define TIMEUTILS_H
// --------------- Aufgaben 10.10 --------------------------


// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.


double GetHRFrequency()
{
LARGE_INTEGER f;
BOOL bf=QueryPerformanceFrequency(&f);
if (bf) return f.QuadPart; // QuadPart hat den Datentyp int64
else return -1;
}

double HRFrequency=GetHRFrequency();

double HRTimeInSec()
{
LARGE_INTEGER c;
BOOL bc=QueryPerformanceCounter(&c);
if (bc) return c.QuadPart/HRFrequency;
else return -1;
}

// Diese Funktionen können wie folgt verwendet werden:
/*
double Start1=HRTimeInSec();
s=Sum1(n);
double End1=HRTimeInSec();

Form1->Memo1->Lines->Add("t1="+FloatToStr(End1-Start1));
```

```
*/


// --------------- Aufgaben 10.10, 2. -----------------------

#include "CompilerId.h"
class CHRTimer {

  LARGE_INTEGER HRFrequency, start, end;
  bool HRTimerSupported;

 public:

  CHRTimer()
  {
  HRTimerSupported=
    QueryPerformanceFrequency(&HRFrequency);
  QueryPerformanceCounter(&start);
  end=start;
  }

  void Start()
  {
  QueryPerformanceCounter(&start);
  }

  void End()
  {
  QueryPerformanceCounter(&end);
  }

  AnsiString TimeStr(const AnsiString& s)
  {
  if (HRTimerSupported)
    return CompilerId()+s+FloatToStrF((double(end.QuadPart)-start.QuadPart)/
                   HRFrequency.QuadPart,ffGeneral,4,2);
  else
    return "No High Resolution Timer supported";
  }

  AnsiString TimeStr()
  {
  if (HRTimerSupported)
    return                                    FloatToStr((double(end.QuadPart)-
start.QuadPart)/HRFrequency.QuadPart).c_str();
  else
    return "No High Resolution Timer supported";
  }

};

// Diese Klasse kann wie folgt verwendet werden:
/*
CHRTimer t;
t.Start();
double s=0;
for (int i=0; i<n; i++)
  s=s+i;
t.End();
Form1->Memo1->Lines->Add(t.TimeString("Summation"));
*/

// --------------- Beispiel 3.19.7 -------------------------

double rdtsc()
{ // RDTSC (ReaD Time Stamp Counter) - gibt die Anzahl der CPU-Taktzyklen
  // seit dem letzten Start des Rechners zurück.
  unsigned int x=0,y=0;
  asm {
```

```
        rdtsc // high-order 32 bits of the register into EDX, and the
              // low-order 32 bits into EAX.
        mov x, eax
        mov y, edx
      }
  long long yl=y;
  return (yl<<32) + x;
}

AnsiString SchaetzeCPUTaktfrequenz()
{
  LARGE_INTEGER HRFrequency, start, end;
  QueryPerformanceFrequency(&HRFrequency);
  double d1a=rdtsc();
  const int wait_ms=10; // zwischen 10 und 1000
  QueryPerformanceCounter(&start);
    Sleep(wait_ms);
  QueryPerformanceCounter(&end);
  double d2a=rdtsc();
  double diff_per_sek=(1000/wait_ms)*(d2a-d1a);

  int t= (diff_per_sek/1000000.0)*(wait_ms/1000.0)*
         HRFrequency.QuadPart/(end.QuadPart-start.QuadPart);

return IntToStr(t)+" MHZ CPU Taktfrequenz";
}

#endif // TIMEUTILS_H
```

11.1.12 GraphUtils.h (Abschnitt 10.13)

```
// File: C:\Loesungen_CB2006\CppUtils\GraphUtils.h

#ifndef GRAPHUTILS_H__
#define GRAPHUTILS_H__
// Einfach: Alle Definitionen in einer h-Datei, keine cpp-Datei.
// Zur Aufteilung der Definitionen und Deklarationen in eine cpp-Datei
// und eine h-Datei: Siehe Abschnitt 3.23 und 3.23.5.


#include <cmath>

double x_Welt(int px, double x0, double x1, double W)
{ // transformiert px aus [0,W] in Welt-Koordinaten [x0,x1]
return x0 + px*(x1-x0)/W;
}

double y_Welt(int py, double y0, double y1, double H)
{ // transformiert py aus [0,H] in Welt-Koordinaten [y1,y0]
return y1 + py*(y0-y1)/H;
}

int x_Bildschirm(double x, double x0, double x1, double W)
{ // transformiert x aus [x0,x1] in Bildkoordinaten [0,W]
return (x-x0)*W/(x1-x0);
}

int y_Bildschirm(double y, double y0, double y1, double H)
{ // transformiert y aus [y0,y1] in Bildkoordinaten [H,0]
return (y-y1)*H/(y0-y1);
}
```

```
// --------------- Aufgaben 10.13 --------------------------

// --------------- Aufgabe 2, b) ---------------------------

void Clear(TCanvas* Canvas, TColor color=clWhite)
{
TColor oldBrushColor = Canvas->Brush->Color;
Canvas->Brush->Color=color;
Canvas->FillRect(Canvas->ClipRect);
Canvas->Brush->Color=oldBrushColor;
}


// --------------- Aufgabe 2, c) ---------------------------

void zeichneGitternetz(TCanvas* Canvas, double x0,double x1,double dx,
                int W, double y0,double y1,double dy, int H)
{
bool xAchse_sichtbar=false;
bool yAchse_sichtbar=false;

int xt=x_Bildschirm(0,x0,x1,W);
if (0<=xt && xt<= W) xAchse_sichtbar=true;
else xt=0;

int yt=y_Bildschirm(0,y0,y1,H);
if (0<=yt && yt<=H) yAchse_sichtbar=true;
else yt=0;

int x=0, i=0;
Canvas->Pen->Color = clGray;
while (x<W)
  { // zeichne Parallelen zur y-Achse
    Canvas->MoveTo(x,H);
    Canvas->LineTo(x,0);
    Canvas->TextOut(x,yt,FloatToStrF(x0+i*dx,ffGeneral,4,4));
    i++;
    x = x_Bildschirm(x0+i*dx,x0,x1,W);
  }


int y=0;
i=0;
while (y<H)
  { // zeichne Parallelen zur x-Achse
    Canvas->MoveTo(0,y);
    Canvas->LineTo(W,y);
    if (std::fabs(y1-i*dy) > 0.001) // sonst Schönheitsfehler
      Canvas->TextOut(xt,y,FloatToStrF(y1-i*dy,ffGeneral,4,4));
    i++;
    y = y_Bildschirm(y1-i*dy,y0,y1,H);
  }
// Koordinatenkreuz schwarz nachzeichnen:
Canvas->Pen->Color = clBlack;

if (xAchse_sichtbar)
  {
    Canvas->MoveTo(xt,H);
    Canvas->LineTo(xt,0);
  }
if (yAchse_sichtbar)
  {
    Canvas->MoveTo(0,yt);
    Canvas->LineTo(W,yt);
  }
}
```

```
// --------------- Aufgabe 2, d) ---------------------------

void zeichneFunktion(TCanvas* Canvas, double f(double),double x0, double x1)
{
// bestimme den minimalen und maximalen Wert von f(x) im Bereich [x0,x1]:
double y0=f(x_Welt(0, x0, x1, Canvas->ClipRect.Width())); // y0=min(f(x))
double y1=y0; // y1=max(f(x))
for (int px=0; px<Canvas->ClipRect.Width(); px++)
  {
    double x=x_Welt(px, x0, x1, Canvas->ClipRect.Width());
    double y=f(x);
    if (y<y0) y0=y;
    if (y>y1) y1=y;
  };

// c) zeichne das Gitternetz:
double dx=(x1-x0)/10; // 10 Gitterlinien
double dy=(y1-y0)/10; // 10 Gitterlinien
zeichneGitternetz(Form1->Image1->Canvas, x0,x1,dx, Canvas->ClipRect.Width(),
           y0,y1,dy, Canvas->ClipRect.Height());

// zeichne die Kurve:
Canvas->Pen->Color = clRed;
Canvas->Pen->Width=1;
for (int px=0; px<Canvas->ClipRect.Width(); px++)
  {
    // transformiere px in Welt-Koordinaten
    double x=x_Welt(px, x0, x1, Canvas->ClipRect.Width());
    // transformiere y in Bildkoordinaten
    int py=y_Bildschirm(f(x), y0, y1, Canvas->ClipRect.Height());
    if (px == 0) Canvas->MoveTo(px,py);
    else         Canvas->LineTo(px,py);
  }
}

#endif // GRAPHUTILS_H__
```

11.2 Lösungen Kapitel 2

11.2.1 Aufgabe 2.2

```
// File: C:\Loesungen_CB2006\Kap_2\2.2\HaushSanU.cpp

#include <vcl.h>
#pragma hdrstop
#include "HaushSanU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}


void __fastcall TForm1::ButtonEingabeLoeschenClick(TObject *Sender)
{
EVorname->Clear();
```

```
ENachname->Clear();
EDelikt->Clear();
EBussgeld->Clear();
}

void __fastcall TForm1::ButtonProgrammendeClick(TObject *Sender)
{
Close();
}
```

11.2.2 Aufgabe 2.3

```
// File: C:\Loesungen_CB2006\Kap_2\2.3\LabelsU.cpp

#include <vcl.h>
#pragma hdrstop
#include "LabelsU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm1::ButtonLinksAusrichtenClick(TObject *Sender)
{
Label1->Caption   = "Label links";
Label1->Alignment = taLeftJustify;
}

void __fastcall TForm1::ButtonRechtsAusrichtenClick(TObject *Sender)
{
Label1->Caption   = "Label rechts";
Label1->Alignment = taRightJustify;
}

void __fastcall TForm1::ButtonUnsichtbarClick(TObject *Sender)
{
Label1->Visible = false;
}

void __fastcall TForm1::ButtonSichtbarClick(TObject *Sender)
{
Label1->Visible = true;
}

void __fastcall TForm1::ButtonLinksClick(TObject *Sender)
{
Label1->Left = 0;
}

void __fastcall TForm1::ButtonRechtsClick(TObject *Sender)
{
Label1->Left = Form1->ClientWidth-Label1->Width;
}
```

11.2.3 Aufgabe 2.4

```
// File: C:\Loesungen_CB2006\Kap_2\2.4\RechnerU.cpp

#include <vcl.h>
#pragma hdrstop
#include "RechnerU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm1::ButtonLoeschenClick(TObject *Sender)
{
Operand1->Clear();
Operand2->Clear();
Ergebnis->Clear();
}

void __fastcall TForm1::ButtonIntPlusClick(TObject *Sender)
{
LabelOperation->Caption="+";
Ergebnis->Text=IntToStr(
                StrToInt(Operand1->Text)+StrToInt(Operand2->Text));
}

void __fastcall TForm1::ButtonDoublePlusClick(TObject *Sender)
{
Ergebnis->Text=FloatToStr(
                StrToFloat(Operand1->Text)+StrToFloat(Operand2->Text));
}

void __fastcall TForm1::ButtonFloatPlusClick(TObject *Sender)
{
LabelOperation->Caption="+";
Ergebnis->Text=FloatToStr(
                StrToFloat(Operand1->Text)+StrToFloat(Operand2->Text));
}

void __fastcall TForm1::ButtonFloatMinusClick(TObject *Sender)
{
LabelOperation->Caption="-";
Ergebnis->Text=FloatToStr(
                StrToFloat(Operand1->Text)-StrToFloat(Operand2->Text));
}

void __fastcall TForm1::ButtonFloatMultClick(TObject *Sender)
{
LabelOperation->Caption="*";
Ergebnis->Text=FloatToStr(
                StrToFloat(Operand1->Text)*StrToFloat(Operand2->Text));
}

void __fastcall TForm1::ButtonFloatDivClick(TObject *Sender)
{
LabelOperation->Caption="/";
Ergebnis->Text=FloatToStr(
                StrToFloat(Operand1->Text)/StrToFloat(Operand2->Text));
}
```

11.2.4 Aufgabe 2.5

```
// File: C:\Loesungen_CB2006\Kap_2\2.5\StringsU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "StringsU.h"
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
Memo1->Lines->Add(Edit1->Text);
ListBox1->Items->Add(Edit1->Text);
ComboBox1->Items->Add(Edit1->Text);
}

void __fastcall TForm1::ListBox1Click(TObject *Sender)
{
Label1->Caption=ListBox1->Items->Strings[ListBox1->ItemIndex];
}

void __fastcall TForm1::Button2Click(TObject *Sender)
{
Label2->Caption=ComboBox1->Text;
}
```

11.2.5 Aufgabe 2.6

```
// File: C:\Loesungen_CB2006\Kap_2\2.6\EventsU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "EventsU.h"
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

void __fastcall TForm1::BTestClick(TObject *Sender)
{
EOnClick->Text="OnClick";
}

void __fastcall TForm1::BTestEnter(TObject *Sender)
{
EOnEnter->Text="OnEnter";
}

void __fastcall TForm1::BTestExit(TObject *Sender)
{
EOnExit->Text="OnExit";
```

```
}

void __fastcall TForm1::BTestKeyDown(TObject *Sender, WORD &Key,
    TShiftState Shift)
{
EOnKeyDown->Text=IntToStr(Key);
}

void __fastcall TForm1::BTestKeyPress(TObject *Sender, char &Key)
{
EOnKeyPress->Text=Key;
}

void __fastcall TForm1::BTestKeyUp(TObject *Sender, WORD &Key,
    TShiftState Shift)
{
EOnKeyUp->Text=Key;
}

void __fastcall TForm1::BTestMouseDown(TObject *Sender,TMouseButton Button,
    TShiftState Shift, int X, int Y)
{
EOnMouseDown->Text="MouseDown";
}

void __fastcall TForm1::BTestMouseUp(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y)
{
EOnMouseUp->Text="MouseUp";
}

void __fastcall TForm1::BTestMouseMove(TObject *Sender, TShiftState Shift,
    int X, int Y)
{
EOnMouseMove->Text=IntToStr(X)+","+IntToStr(Y);
}

void __fastcall TForm1::JumpButtonMouseMove(TObject *Sender, TShiftState Shift,
      int X, int Y)
{
JumpButton->Left = Width - JumpButton->Left - JumpButton->Width;
}

void __fastcall TForm1::JumpButtonClick(TObject *Sender)
{
JumpButton->Caption="getroffen";
}

void __fastcall TForm1::BClearClick(TObject *Sender)
{
EOnClick->Clear();
EOnEnter->Clear();
EOnExit->Clear();
EOnMouseDown->Clear();
EOnMouseMove->Clear();
EOnMouseUp->Clear();
EOnKeyPress->Clear();
EOnKeyUp->Clear();
EOnKeyDown->Clear();
}
```

11.2.6 Aufgabe 2.7

```
// File: C:\Loesungen_CB2006\Kap_2\2.7\ChecksU.cpp
```

```
#include <vcl.h>
#pragma hdrstop
#include "ChecksU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

void __fastcall TForm1::Button2Click(TObject *Sender)
{
if (CheckBox1->Checked)              // a)
  {
    CheckBox2->Enabled=true;
    RadioButton1->Enabled=true;
    Button1->Enabled=true;
  }
else
  {
    CheckBox2->Enabled=false;
    RadioButton1->Enabled=false;
    Button1->Enabled=false;
  }

                 // alternativ                        // a)

CheckBox2->Enabled   =CheckBox1->Checked;
RadioButton1->Enabled=CheckBox1->Checked;
Button1->Enabled     =CheckBox1->Checked;

if (CheckBox2->Checked)              // b)
  Button1->Caption="Hello world";
else
  Button1->Caption="Hello moon";

Button1->Visible=CheckBox3->Checked; // c)

if (RadioButton1->Checked)           // d)
  CheckBox1->Color=clRed;
if (RadioButton2->Checked)
  CheckBox1->Color=clGreen;
if (RadioButton3->Checked)
  CheckBox1->Color=clBlue;
// alternativ
if (RadioButton1->Checked)           // d)
 CheckBox1->Color=clRed;
else if (RadioButton2->Checked)
  CheckBox1->Color=clGreen;
else
  CheckBox1->Color=clBlue;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
CheckBox2->Caption="caption";
RadioButton1->Caption="red";
RadioButton2->Caption="green";
RadioButton3->Caption="blue";
}
```

11.2.7 Aufgabe 2.9

```
// File: C:\Loesungen_CB2006\Kap_2\2.9\Readme.txt

Die Lösung von Aufgabe 2.9 ist in der Lösung von Aufgabe 2.10 enthalten.
```

11.2.8 Aufgabe 2.10

```
// File: C:\Loesungen_CB2006\Kap_2\2.10\MenuU.cpp

#include <vcl.h>
#pragma hdrstop
#include "MenuU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

void __fastcall TForm1::ffnen1Click(TObject *Sender)
{
OpenDialog1->Filter = "Textdateien|*.TXT";
if (OpenDialog1->Execute())
  Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
}

void __fastcall TForm1::Speichern1Click(TObject *Sender)
{
if (SaveDialog1->Execute())
  Memo1->Lines->SaveToFile(OpenDialog1->FileName);
}

void __fastcall TForm1::Schlieen1Click(TObject *Sender)
{
Close();
}

void __fastcall TForm1::Suchen1Click(TObject *Sender)
{
FindDialog1->Execute();
}

void __fastcall TForm1::SuchenundErsetzen1Click(TObject *Sender)
{
ReplaceDialog1->Execute();
}

void __fastcall TForm1::Allesmarkieren1Click(TObject *Sender)
{
Memo1->SelectAll();
}

void __fastcall TForm1::Kopieren1Click(TObject *Sender)
{
Memo1->CopyToClipboard();
}

void __fastcall TForm1::Ausschneiden1Click(TObject *Sender)
```

```
{
Memo1->CutToClipboard();
}

void __fastcall TForm1::Drucken2Click(TObject *Sender)
{
PrintDialog1->Execute();
}

void __fastcall TForm1::Druckereinrichten1Click(TObject *Sender)
{
PrinterSetupDialog1->Execute();
}

void __fastcall TForm1::Farben1Click(TObject *Sender)
{
if (ColorDialog1->Execute())
  Memo1->Brush->Color = ColorDialog1->Color;
}

void __fastcall TForm1::Schriftart1Click(TObject *Sender)
{
if (FontDialog1->Execute())
  Memo1->Font->Assign(FontDialog1->Font);
}
```

11.3 Lösungen Kapitel 3

11.3.1 Aufgabe 3.1

```
// File: C:\Loesungen_CB2006\Kap_3\3.1\Syntax.txt

#include <vcl.h>
#pragma hdrstop

#include "SyntaxU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------


void __fastcall TForm1::Button1Click(TObject *Sender)
{
// zulässige Zeichenfolgen: 1, 11, 112
// unzulässige Zeichenfolgen: 0, 01, 011
// Ein Dezimalliteral ist eine beliebige Folge der Ziffern 0..9,
// die nicht mit einer 0 beginnt.
}
```

11.3.2 Aufgabe 3.2

```
// File: C:\Loesungen_CB2006\Kap_3\3.2\VarDeclU.cpp

#include <vcl.h>
#pragma hdrstop
#include "VarDeclU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}


// --------------- Aufgabe 3.2, 1. -------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
/*
int Preis_in_$,    // unzulässig wegen Sonderzeichen $
    x kleiner y,   // unzulässig wegen der Leerzeichen " "
    Zinssatz_in_%, // unzulässig wegen Sonderzeichen %
    x/y,           // unzulässig wegen Sonderzeichen /
    this,          // unzulässig, da Schlüsselwort
    Einwohner_von_Tübingen; // unzulässig wegen Umlaut "ü"
*/
}


// --------------- Aufgabe 3.2, 2. -------------------------

int i=0;
int j;

void __fastcall TForm1::Button2Click(TObject *Sender)
{
int k=1;
Memo1->Lines->Add(IntToStr(i)); // 0
Memo1->Lines->Add(IntToStr(k)); // 1
}

void __fastcall TForm1::Button3Click(TObject *Sender)
{
int k;
Memo1->Lines->Add(IntToStr(j));   //   0:   globale   Variablen   werden   immer
initialisiert
Memo1->Lines->Add(IntToStr(k)); // ein unbestimmter Wert
}
```

11.3.3 Aufgabe 3.3

```
// File: C:\Loesungen_CB2006\Kap_3\3.3\IntU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "IntU.h"
//---------------------------------------------------------------------
#pragma resource "*.dfm"
```

```
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

// --------------- Aufgabe 3.3.2, 1. ------------------------

void __fastcall TForm1::Button_3_3_2_1Click(TObject *Sender)
{
// 1. a)

  // 37 im Binärsystem:
  // 37dez = 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 0010 0101bin

//    b)

  // -37 im Zweierkomplement: Einerkomplement(37) +1
  // 1101 1010 + 1 = 1101 1011bin = -37dez

//    c)
  // 37dez = 2*16^1 + 5*16^0 = 25hex,
  // Alternativ - Die Blöcke aus der Binärdarstellung sind die hex-Ziffern:
  //            0010bin = 2hex, 0101bin = 5hex

  //            37dez  = 0010 0101bin = 25 hex
  //            -37dez = 1101 1011bin = db hex

Memo1->Lines->Add("Aufgabe 1:");
int i1=37;
Memo1->Lines->Add(IntToStr(i1)+ "dez: "+IntToHex(i1,2)+ "hex");
int j1=-37;
Memo1->Lines->Add(IntToStr(j1)+ "dez: "+IntToHex(j1,2)+ "hex");

//    d) 37 + (-25)

//       37: 00100101bin
//      -25: 11100111bin
//           -----------
//          100001100bin, mit 8 bits: 00001100bin = 12dez


//    e) 25 + (-37)

//       25: 00011001bin
//      -37: 11011011bin
//           -----------
//           11110100bin --> Zweierkomplement: 00001100bin
}


// --------------- Aufgabe 3.3.2, 2. ------------------------

void __fastcall TForm1::Button_3_3_2_2Click(TObject *Sender)
{
// 2. a)

  // Zweierkomplement von ab=1010 1011: 0101 0101 bin = 55 hex = 85 dez,
  // also -85

Memo1->Lines->Add("Aufgabe 2:");
signed char i2=0xab;
Memo1->Lines->Add(IntToStr(i2)+ "dez: "+IntToHex(i2,2)+ "hex");

//    b)

  // ab = 10*16 + 11 = 171
```

```
unsigned char j2=0xab;
Memo1->Lines->Add(IntToStr(j2)+ "dez: "+IntToHex(j2,2)+ "hex");
}

// --------------- Aufgabe 3.3.2, 3. -----------------------

void __fastcall TForm1::Button_3_3_2_3Click(TObject *Sender)
{
Memo1->Lines->Add("Aufgabe 3:");
Memo1->Lines->Add(IntToStr(030)); // Vorwahl Berlin // 24,
       // da wegen der führenden 0 ein Oktalliteral
Memo1->Lines->Add(IntToStr(017+15));  // 30
Memo1->Lines->Add(IntToStr(0x12+10)); // 28
}


// --------------- Aufgabe 3.3.4, 1. -----------------------

void __fastcall TForm1::Button_3_3_4_1Click(TObject *Sender)
{
unsigned char b = 255;
unsigned char c = 0;
//short i = 20000;
short i = 1234567898;
b = b + 1;      // 0
c = c - 1;      // 255
int d = i/100;
int e = i%100;
int f = i+i;

Memo1->Lines->Add(IntToStr(b));
Memo1->Lines->Add(IntToStr(c));
Memo1->Lines->Add(IntToStr(d));
Memo1->Lines->Add(IntToStr(e));
Memo1->Lines->Add(IntToStr(f));
}


// --------------- Aufgabe 3.3.4, 2. -----------------------

void __fastcall TForm1::Button_3_3_4_2Click(TObject *Sender)
{
AnsiString s=
"Falls die beiden Zahlen i und j verschieden sind, ist immer einer der "
"beiden Faktoren in (i/j)*(j/i) Null (Ganzzahldivision). Wenn i und j "
"dagegen gleich sind, ist das Produkt immer Eins.";
Memo1->Lines->Add(s);
}

// --------------- Aufgabe 3.3.4, 3. -----------------------

void __fastcall TForm1::Button_3_3_4_3Click(TObject *Sender)
{
// a)
// Wenn beide Operanden den Datentyp int haben, werden sie nicht
// im Rahmen einer ganzzahligen Typangleichung in den Datentyp int
// konvertiert.
// b)
// Wenn der Datentyp des Ausdrucks ein "kleinerer" Datentyp als int
// ist (z.B. char), wird er durch die Addition von 0 (Datentyp int)
// in int konvertiert.
}

// --------------- Aufgabe 3.3.4, 4. -----------------------

void __fastcall TForm1::Button_3_3_4_4Click(TObject *Sender)
{
// a)
// int a=x^x;
```

```
// a erhält immer den Wert 0, da die bitweise Verknüpfung von zwei
// Operanden mit xor immer den Wert 0 hat.

// b)
// int b=x&1;

// b ist genau dann 1, wenn x ungerade ist.
}

// --------------- Aufgabe 3.3.4, 5. -----------------------

// Im Bitmuster von x und y darf nicht an derselben Stelle eine 1 stehen.

// --------------- Aufgabe 3.3.4, 6. -----------------------

void __fastcall TForm1::Button_3_3_4_6Click(TObject *Sender)
{
/*
a)
Die beiden Operanden c und -1 haben dasselbe Bitmuster.
Wenn sie nicht in einen gemeinsamen Datentyp konvertiert würden,
wäre das Ergebnis des Vergleichs true.

b)
Wenn die beiden Operanden x-y==-1 und 0 nicht in einen gemeinsamen
Datentyp konvertiert würden, hätte die linke Seite das Bitmuster
FF, und das Ergebnis des Vergleichs wäre false.

c)
Die mathematisch gleichwertigen Ausdrücke in b) und c) hätten
verschiedene Ergebnisse.

In a) und b) wäre das Ergebnis hochgradig kontraintuitiv, und die von
b) und c) sollten gleich sein.
*/
}


// --------------- Aufgabe 3.3.6, 1. -----------------------

void __fastcall TForm1::Button_3_3_6_1Click(TObject *Sender)
{
// a)
char c;
bool Grossbuchstabe = (c>='A') && (c<='Z');

// b)
bool Buchstabe = ((c>='A') && (c<='Z')) || ((c>='a') && (c<='z'));

// c)
bool alphanumerisch = ((c>='A') && (c<='Z')) || ((c>='a') && (c<='z'))||
                      ((c>='0') && (c<='9'));
}

// --------------- Aufgabe 3.3.6, 2. -----------------------

void __fastcall TForm1::Button_3_3_6_2Click(TObject *Sender)
{
int Jahr;
bool Schaltjahr =((Jahr%4 == 0) && (Jahr%100 != 0)) || (Jahr%400 == 0);
}

// --------------- Aufgabe 3.3.6, 3. -----------------------

void __fastcall TForm1::Button_3_3_6_3Click(TObject *Sender)
{
int j1,j2,m1,m2,t1,t2;
bool vorher = (j1<j2) || ((j1==j2) && ((m1<m2) || ((m1==m2) && (t1< t2))));
```

```
}

// --------------- Aufgabe 3.3.6, 4. ------------------------

void __fastcall TForm1::Button_3_3_6_4Click(TObject *Sender)
{
/*
bool:
  x y     x&&y       x&y
  f f      f         0000&0000 == 0000 (f)
  f t      f         0000&0001 == 0000 (f)
  t f      f         0001&0000 == 0000 (f)
  t t      t         0001&0001 == 0001 (t)

int:

  x  y     x&&y            x&y
  0  0    f && f ==f     0&0 == 0 (f)
  0 !0    f && t ==f     0&0 == 0 (f)
  !0 0    t && f ==f     0&0 == 0 (f)
  !0 !0   t && t ==t

Für bool Operanden x und y ist das Ergebnis von x&&y immer identisch
mit dem von x&y.

Für int Operanden ist das Ergebnis in den ersten drei Zeilen gleich.
In der letzten Zeile erhält man bei !0&!0 nur dann den Wert true,
wenn sich die bits der Operanden nicht auslöschen. Z.B. erhält man
mit 1&2 den Wert false, während 1&&2 true ist.
*/
}

// --------------- Aufgabe 3.3.6, 5. ------------------------

void __fastcall TForm1::Button_3_3_6_5Click(TObject *Sender)
{
/*
In !0 wird der int-Ausdruck 0 in den booleschen Wert false konvertiert.
Deshalb stellt !0 den Wert true dar.
Als Operand des binären Operators == wird der boolesche Wert true in
einer ganzzahligen Typangleichung  in den Wert 1 konvertiert. Deshalb
entspricht die Bedingung

  if (x==!0) ... // gemeint war if (x!=0) ...

der Bedingung

  if (x==1) ...
*/
}

// --------------- Aufgabe 3.3.6, 6. ------------------------

void __fastcall TForm1::Button_3_3_6_6Click(TObject *Sender)
{
Option1a1->Enabled = !Option1a1->Enabled;
GroupBox4->Visible = !GroupBox4->Visible;
}
```

11.3.4 Aufgabe 3.4

```
// File: C:\Loesungen_CB2006\Kap_3\3.4\FuncU.cpp
```

```
#include <vcl\vcl.h>
#pragma hdrstop
#include "FuncU.h"
#pragma resource "*.dfm"
TForm1 *Form1;


__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
#ifdef _DEBUG
bool test_3_4();      // Prototyp
void initTestUtils(); // Prototyp
void testUtilsSummary(TMemo* Memo, bool result); // Prototyp

initTestUtils();
testUtilsSummary(Memo1, test_3_4());
#endif

// Damit es keine Konversionsfehler gibt, wenn man vergisst, einen
// ganzzahligen Wert in das Eingabefenster einzugeben.
Edit1->Text="50";
Edit2->Text="1990";
Edit3->Text="2010";
}


// --------------- Aufgaben 3.4.1 --------------------------

// --------------- Aufgabe 1 -------------------------------

void __fastcall TForm1::Button_3_4_1_1Click(TObject *Sender)
{
int x, Punkte, i, j;
bool b;

// a)
if (x=17) Memo1->Lines->Add("Volltreffer");
// "x=17" ist eine Zuweisung und keine Prüfungen auf Gleichheit

// b)
if (i>=1 && i<=10) Edit1->Text = "Volltreffer";
// Syntaktisch und semantisch korrekt. Verknüpfte Bedingungen müssen nicht
// geklammert werden.

// c)
// if b && (i=j*x) Edit1->Text = "Volltreffer";
// Syntaxfehler: Die Bedingung in einer if-Anweisung muss in einer Klammer
// stehen. "=" ist ein Zuweisung und keine Prüfung auf Gleichheit.

// d)
if (Punkte >= 0) Edit1->Text="Extremes Pech";
else if (Punkte >= 20) Edit1->Text="Ziemliches Pech";
else if (Punkte >= 40) Edit1->Text="Ein wenig Glück gehabt";
// Syntaktisch korrekt, aber der zweite und dritte else-Zweig werden nie
// erreicht. Vermutlich „<" mit „>" verwechselt.
}


// --------------- Aufgabe 2 -------------------------------

void __fastcall TForm1::Button_3_4_1_2Click(TObject *Sender)
{
/*
Für diese Aufgabe werden meist zuerst die folgenden beiden Lösungen
vorgeschlagen:

  if (Bewegungsart=='+') Kontostand = Kontostand+Betrag;
  else Kontostand = Kontostand-Betrag;
```

oder

```
  if (Bewegungsart=='+') Kontostand = Kontostand+Betrag;
  if (Bewegungsart=='-') Kontostand = Kontostand-Betrag;
```

Beide Lösungen haben aber den Nachteil, dass eine unzulässige Bewegungsart zu einer falschen Verbuchung des Betrages führt: Im ersten Fall kommt es zu einer Abbuchung (so würde ich das machen, wenn die Bank mir gehören würde), und im zweiten Fall fällt jede falsche Kontobewegung unter den Tisch. Bei einer größeren Anzahl von Datensätzen bliebe ein solcher Fehler vermutlich sogar unentdeckt.

Fügt man diesen Anweisungen eine weitere hinzu, um eine falsche Bewegungsart abzufangen, muss man schon etwas nachdenken, um die richtige Bedingung zu finden (hier werden relativ oft && und || verwechselt): Ist

```
  if ((Bewegungsart != '+') && (Bewegungsart != '-'))
   Fehlermeldung();
```

oder || anstelle von && richtig? Falls das Programm dann später auf noch andere Bewegungsarten erweitert wird, erhält man zunächst eine falsche Fehlermeldung, weil meist vergessen wurde, die Bedingung für die Fehlermeldung zu ändern.

Das Problem bei Aufgaben wie dieser liegt darin, dass die Aufgabenstellung zwar die Auswahl einer von zwei Bedingungen nahe legt. Tatsächlich ist aber eine Bedingung mehr zu berücksichtigen, nämlich die Fehlerbehandlung. Deshalb ist in diesem Beispiel eine von drei Anweisungen auszuwählen.

Die nächste Variante hat die oben genannten Nachteile nicht:

```
  if (Bewegungsart=='+') Kontostand = Kontostand + Betrag;
  else if (Bewegungsart=='-') Kontostand = Kontostand - Betrag;
  else Fehlermeldung();
*/
}


// --------------- Aufgabe 3 -------------------------------

int LA_Summe,LB_Summe,MA_Summe,MB_Summe,MC_Summe,Summe;

void LagMat(char Lagergruppe, char Materialgruppe)
{
if (Lagergruppe == 'A')
  {
    if (Materialgruppe == 'A')
      {
        LA_Summe = LA_Summe + Summe;
        MA_Summe = MA_Summe + Summe;
      }
    else if (Materialgruppe == 'B')
      {
        LA_Summe = LA_Summe + Summe;
        MB_Summe = MB_Summe + Summe;
      }
    else if (Materialgruppe == 'C')
      {
        LA_Summe = LA_Summe + Summe;
        MC_Summe = MC_Summe + Summe;
      }
    else ShowMessage("Unzulässige Materialgruppe in Lager A");
  }
else if (Lagergruppe == 'B')
  {
    if (Materialgruppe == 'A')
      {
        LB_Summe = LB_Summe + Summe;
```

```
        MA_Summe = MA_Summe + Summe;
      }
    else if (Materialgruppe == 'B')
      {
        LB_Summe = LB_Summe + Summe;
        MB_Summe = MB_Summe + Summe;
      }
    else ShowMessage("Unzulässige Materialgruppe in Lager B");
  }
else ShowMessage("Unzulässige Lagergruppe");
}

void __fastcall TForm1::Button_3_4_1_3Click(TObject *Sender)
{
char Lagergruppe, Materialgruppe;
LagMat(Lagergruppe, Materialgruppe);
}

// --------------- Aufgabe 4 ------------------------------

bool vorher(int t1, int m1, int j1, int t2, int m2, int j2)
{
if (j1!=j2) return j1<j2;
else // j1==j2
     if (m1!=m2) return m1<m2;
else // (j1==j2) and (m1==m2)
     return t1<t2;

// Falls die boolesche Variable vorher_oder_gleich genau dann den
// Wert true erhalten soll, wenn das erste Datum zeitlich vor dem
// zweiten liegt oder gleich dem zweiten ist, muss lediglich die
// letzte Zeile auf <= geändert werden zu:
//
//   return t1<=t2;
}

void __fastcall TForm1::Button_3_4_1_4Click(TObject *Sender)
{
int t1=1,m1=2,j1=3; // Datum 1
int t2=1,m2=2,j2=3; // Datum 2
vorher(t1,m1,j1, t2,m2,j2);
}


// --------------- Aufgaben 3.4.6 --------------------------


// --------------- Aufgabe 1 ------------------------------

int Quersumme(int n)
{
if (n<0) n=-n; // berücksichtige negative n
// % ist für negative Operanden compilerabhängig
int s=0;
while (n>0)
  {
    s = s+n%10;
    n = n/10;
  }
return s;
}

void zeigeQuerSummen(int a, int b)
{
Form1->Memo1->Lines->Add("Quersummen");
for (int i=a; i<=b; i++)
  Form1->Memo1->Lines->Add(IntToStr(i)+": "+
                           IntToStr(Quersumme(i)));
}
```

```
void __fastcall TForm1::Button_3_4_6_1Click(TObject *Sender)
{
zeigeQuerSummen(1,StrToInt(Edit1->Text));
}


// --------------- Aufgabe 2 -------------------------------

int Fibonacci(int n)
{
int f=0,x=0,y=1;
for (int i=0; i<n; i++)
  {
    x=y;
    y=f;
    f=x+y;
  }
return f;
}

void zeigeFibonacci(int n)
{
Form1->Memo1->Lines->Add("Fibonacci-Zahlen:");
for (int i=0; i<n; i++)
  Form1->Memo1->Lines->Add(IntToStr(i)+":  "+IntToStr(Fibonacci(i)) );
Form1->Memo1->Lines->Add("Falsche Ergebnisse wegen Überlauf für n>=47");
}

void __fastcall TForm1::Button_3_4_6_2Click(TObject *Sender)
{
zeigeFibonacci(StrToInt(Edit1->Text));
}


// --------------- Aufgabe 3 -------------------------------

void __fastcall TForm1::Button_3_4_6_3Click(TObject *Sender)
{
int n=StrToInt(Edit1->Text);
for (unsigned int u=0; u<=n-1; u++) // Warnung: Vergleich von signed- und
  Memo1->Lines->Add(IntToStr(u));   //          unsigned-Werten
// a), b)
// Da u den Datentyp unsigned int hat, wird der Datentyp von n-1 im
// Ausdruck u<=n-1 ebenfalls in unsigned int konvertiert. Für n=0 führt
// das zu einer Endlosschleife. Für Werte n>=1 erhält man das erwartete
// Ergebnis.

// c) Mit der Bedingung u<n gibt es auch mit n=0 keine Endlosschleife.
for (unsigned int u=0; u<n; u++) // Warnung: Vergleich von signed- und
  Memo1->Lines->Add(IntToStr(u));//          unsigned-Werten
}


// --------------- Aufgabe 4 -------------------------------

bool istPrim(int n)
{
if (n<2) return false;
else if (n==2) return true; // dieser Zweig ist überflüssig
else
  {
  for (int i=2; i*i<=n; i++)
    if (n%i==0) return false;
  return true;
  }
}

bool istPrim_(int n) // gleichwertige Alternative
```

```
{
if (n<2) return false;
else if (n==2) return true;
else
  {
    int i=2;
    while (i*i<=n)
      {
        if (n%i==0) return false;
        i++;
      }
    return true;
  }
}

void zeigePrimzahlen(int n)
{
Form1->Memo1->Lines->Add("Primzahlen: ");
for (int i=1;i<n; i++)
  if (istPrim(i))
    Form1->Memo1->Lines->Add(IntToStr(i));
}


void __fastcall TForm1::Button_3_4_6_4Click(TObject *Sender)
{
zeigePrimzahlen(StrToInt(Edit1->Text));
}

int Goldbach(int n)
{
if ((n<4) || (n%2==1)) return -1;
else
  {
    int k=0;
    for (int i=2; i<=n/2; i++)
      if (istPrim(i) && istPrim(n-i)) k++;
    return k;
  }
}

void zeigeGoldbachPaare(int n)
{
Form1->Memo1->Lines->Add("Goldbach'sche Vermutung:");
for (int i=6; i<100; i=i+2)
  Form1->Memo1->Lines->Add(IntToStr(i)+": g="+IntToStr(Goldbach(i)));
}


void __fastcall TForm1::Button_3_4_6_5Click(TObject *Sender)
{
zeigeGoldbachPaare(StrToInt(Edit1->Text));
}


// --------------- Aufgabe 5 -------------------------------

int zeigePythagTripel(int n)
{
int nt=0; // Anzahl der gefundenen Tripel
for (int a=1; a<n; a++)
  for (int b=a; b<n; b++)
     for (int c=b; c*c<=a*a+b*b; c++) // "for (int c=1; ..." geht auch
       if (a*a+b*b==c*c)
         {
         Form1->Memo1->Lines->Add("a="+IntToStr(a)+" b="+
                                  IntToStr(b)+" c= "+ IntToStr(c));
         nt++;
         }
```

```
return nt;
}

void __fastcall TForm1::Button_3_4_6_6Click(TObject *Sender)
{
int n=zeigePythagTripel(StrToInt(Edit1->Text));
Form1->Memo1->Lines->Add("n="+IntToStr(n));
}


// --------------- Aufgabe 6 -------------------------------

int f3nplus1(int n, bool show)
{
if (n<=0) return -1;
int m=0;
while (n!=1)
  {
    m++;
    if (n%2==1) n=3*n+1;
    else n=n/2;
    if (show) Form1->Memo1->Lines->Add(" n="+IntToStr(n));
  }
return m;
}

void zeige3nplus1(int n)
{
for (int i=0; i<n; i++)
  {
    int k=f3nplus1(i,false);
    Form1->Memo1->Lines->Add("i="+IntToStr(i)+" k="+IntToStr(k));
  }
}

void __fastcall TForm1::Button_3_4_6_7Click(TObject *Sender)
{
zeige3nplus1(StrToInt(Edit1->Text));
}


// --------------- Aufgabe 7 -------------------------------

void zeigeZufallszahlen(int n)
{
Form1->Memo1->Lines->Add("rand Zufallszahlen: ");
for (int i=0; i<n; i++)
    Form1->Memo1->Lines->Add(IntToStr(1+rand()%49));
Form1->Memo1->Lines->Add("random Zufallszahlen: ");
for (int i=0; i<n; i++)
    Form1->Memo1->Lines->Add(IntToStr(1+random(48)));
}

void __fastcall TForm1::Button_3_4_6_8Click(TObject *Sender)
{
zeigeZufallszahlen(10);
}

void __fastcall TForm1::RandomizeClick(TObject *Sender)
{
randomize();
}


// --------------- Aufgabe 8 -------------------------------
AnsiString Osterdatum(int J)
{
int A,B,C,D,E,M,N,OTag,PTag;
AnsiString s;
```

```
if (J < 1583) s = " Formel gilt nicht ";

else if (J < 1700) { M = 22; N = 2; }
else if (J < 1800) { M = 23; N = 3; }
else if (J < 1900) { M = 23; N = 4; }
else if (J < 2100) { M = 24; N = 5; }
else if (J < 2200) { M = 24; N = 6; }
else if (J < 2300) { M = 25; N = 0; }
else s = " Formel gilt nicht ";

A = J % 19;
B = J % 4;
C = J % 7;
D = (19*A + M) % 30;
E = (2*B + 4*C + 6*D + N) % 7;
if ((22 + D + E >= 1) && (22 + D + E <= 31)) // in [1..31])
  {
    OTag =22 + D + E;
    PTag = OTag+49;
    s = s+IntToStr(J)+" Ostern: "+IntToStr(OTag)+ ". März Pfingsten: ";
    if (PTag>31+30+31) //  Tage im März + April + Mai
      s = s+IntToStr(PTag-92) + ". Juni";
    else if (PTag > 30+31) // Tage im April + Mai
      s = s+IntToStr(PTag-61) + ". Mai";
    else s = s + " unmöglich"; // Wegen OTag >= 22 ist PTag >= 71
  }
else
  {
    OTag = D + E - 9;
    PTag = OTag+49;
    if (OTag == 26) OTag = 19;
    else if ((OTag==25) && (D==28) && (E==6) && (A>10)) OTag = 18;
    s= s+IntToStr(J)+" Ostern: "+IntToStr(OTag)+ ". April Pfingsten: ";
    if (PTag > 30+31) // Tage im April + Mai
      s = s+IntToStr(PTag-61) + ". Juni";
    else if (PTag > 30) // Tage im April
      s = s+IntToStr(PTag-30) + ". Mai";
    else s = s + " unmöglich"; // wegen OTag >= -9 ist PTag >= 40
  }
return s;
}

void __fastcall TForm1::Button_3_4_6_9Click(TObject *Sender)
{
int von=StrToInt(Edit2->Text);
int bis=StrToInt(Edit3->Text);
for (int J=von; J<=bis; J++)
  Memo1->Lines->Add(Osterdatum(J));
}

// --------------- Aufgabe 9 -------------------------------

/*
1. Quersumme:

   Es wird immer (unabhängig vom Argument n) der Wert von s
   zurückgegeben. s wird für jeden Wert von n mit 0 initialisiert und
   eventuell in der Schleife verändert.

   Wenn man die Funktion dagegen so formuliert hätte

      int Quersumme(int n)
      {
      if (n>=0)
        {
          int s=0;
          while (n>0)
            {
              s = s+n%10;
```

```
              n = n/10;
            }
          return s;
        }
      }

   wäre der Funktionswert für Argumente <0 nicht definiert.

2. Fibonacci

   Es wird immer (unabhängig vom Argument n) der Wert von f
   zurückgegeben. f wird für jeden Wert von n mit 0 initialisiert und
   eventuell in der Schleife verändert.

4. bool prim(int n)
   int Goldbach(int n)

   Da in jedem Zweig der if-Anweisung ein Wert zurückgegeben wird,
   ist der Funktionswert immer definiert.

5. int zeige_PythagZahlentripel(int n)

   wie 1.

6. int f3nplus1(int n, bool show)

   Die Funktion gibt entweder -1 oder m zurück.

7. void Lottozahlen()

   Diese Funktion gibt keinen Wert zurück.

8. AnsiString Osterdatum(int J)

   wie 1.
*/


// --------------- Aufgaben 3.5.2 --------------------------

#include "..\3.5\Tests.cpp"

void __fastcall TForm1::Test_1Click(TObject *Sender)
{
initTestUtils();
testUtilsSummary(Memo1, test_3_4());
}
```

11.3.5 Aufgabe 3.5

```
// File: C:\Loesungen_CB2006\Kap_3\3.5\Tests.cpp

/*
Diese Datei wird in
  ..\3.4\FuncU.cpp

mit

  #include "..\3.5\Tests.cpp"

eingebunden.
*/
```

```
// --------------- Aufgaben 3.5.1 --------------------------


// --------------- Aufgabe 1 -------------------------------

/*
Falls Sie Ihre Sollwerte aus den Quelltexten oder aus der Ausführung
Ihres Programms zur Lösung dieser Aufgaben abgeleitet haben, sind Ihre
Tests wertlos.
Die Sollwerte für einen Test müssen immer aus der Spezifikation
abgeleitet werden. Das sind in diesem Fall die Aufgaben aus
Abschnitt 3.4.

Die folgenden Test erfüllen die Mindestanforderung, jeden
Schleifenkörper nie, einmal und mindestens einmal auszuführen. Außerdem
wird jeder if-Zweig einmal ausgeführt:

a) if-Zweig: mit n<0;
   Schleife 0 mal: n==0; 1 mal: 1<=n<=9; 2 mal: 10<=n<=99

   Das wird mit den folgenden Testfällen erreicht:
   {n=-1;Quersumme=1 | n=0;Quersumme=0 | n=12;Quersumme=3 }

b) Schleife 0 mal: n<=0; 1 mal: n==1; 2 mal: n==2

   Das wird mit den folgenden Testfällen erreicht:
   {n=-1;Fibonacci=0 | n=0;Fibonacci=0 | n=1;Fibonacci=1 |
    n=2;Fibonacci=1 }

c) if-Zweig I:  mit n<2;
   if-Zweig II: mit n==2;
   if-Zweig III: mit n>2;
     Schleife 0 mal: n==3
              1 mal: 4<=n<=8,   if nie: n==5,7;   if ja: n==4,6,8
              2 mal: 9<=n<=15,  if nie: n==11,13; if ja: 9,10,12,14,15

   Das wird mit den folgenden Testfällen erreicht:
   {n=1;prim=false | n=2;prim=true | n=3;prim=true | n=4;prim=false |
    n=5;prim=true | n=9;prim=false | n=11;prim=true }

d) if: n<4, n gerade;   n==2
       n<4, n ungerade: n==3
       n==4:            n==4
       n>4, n ungerade: n==5
       n>4, n gerade,
          Schleife 0 mal: geht nicht
          Schleife 1 mal: n==4
          Schleife 2 mal: n==6

   Das wird mit den folgenden Testfällen erreicht:
   {n=2;Goldbach=-1 | n=3;Goldbach=-1 | n=4;Goldbach=1 |
    n=5;Goldbach=-1 | n=6;Goldbach=1 | n=100;Goldbach=6 |
    n=1000;Goldbach=28}

e) Schleife 0 mal: n<=1
            1 mal: n==2 for a: 1 mal
                        for b: 1 mal
                        for c: 1 mal
            Prüfe mit n=5, ob das Tripel {3,4,5} entdeckt wird


   {n=0;Pythag=0 | n=1;Pythag=0 | n=5;Pythag=1 | n=50;Pythag=26}

f) if: n<=1           n==1
   Schleife 0 mal: geht nicht mit n>1
            1 mal: n==2
            2 mal: n==4
            ungerades n für if in der Schleife: n==3
```

```
   Das wird mit den folgenden Testfällen erreicht:
   {n=1;f3nplus1=0 | n=2;f3nplus1=1 | n=3;f3nplus1=7 |
    n=4;f3nplus1=2 }

g) Anzahl Pfadtests

  MAX=2.147.483.648

  a) 1+Stellenzahl(MAX)=11 Tests (1: für n<0, 1: für n=0,
                                  1: für jede Stellenzahl)
  b) MAX Tests
  c) 2+2^sqrt(MAX)=2+2^65536 Tests
  d) 1+2^((MAX-4)/2) Tests

*/

#include "..\..\CppUtils\TestUtils.h"


// --------------- Aufgaben 3.5.2 --------------------------

// --------------- Aufgabe 1 -------------------------------

// Die folgenden Testfunktionen führen die Tests von Aufgabe 3.5.1 durch

bool test_QuerSumme()
{// ein einfacher Test für die Funktion QuadratSumme
 // a) {n=-1;Quersumme=1 | n=0;Quersumme=0 | n=12;Quersumme=3 }
bool result = true;
if (!assertEqual(Quersumme(-1),1,"Quersumme(-1)")) result=false;
if (!assertEqual(Quersumme( 0),0,"Quersumme( 0)")) result=false;
if (!assertEqual(Quersumme(12),3,"Quersumme(12)")) result=false;
return result;
}


bool test_Fibonacci()
{ // b) {n=-1,Fibonacci=0 | n=0,Fibonacci=0 | n=1,Fibonacci=1 |
  //     n=2,Fibonacci=1 }
bool result = true;
if (!assertEqual(Fibonacci(-1),0,"Fibonacci(-1)")) result=false;
if (!assertEqual(Fibonacci( 0),0,"Fibonacci(0)")) result=false;
if (!assertEqual(Fibonacci( 1),1,"Fibonacci(1)")) result=false;
if (!assertEqual(Fibonacci( 2),1,"Fibonacci(2)")) result=false;
return result;
}

bool test_prim()
{// c) {n=1;prim=false | n=2;prim=true | n=3;prim=true | n=4;prim=false |
 //     n=5;prim=true | n=9;prim=false | n=11;prim=true }
bool result = true;
if (!assertEqual(istPrim(1),false,"istPrim(1)")) result=false;
if (!assertEqual(istPrim(2),true, "istPrim(2)")) result=false;
if (!assertEqual(istPrim(3),true, "istPrim(3)")) result=false;
if (!assertEqual(istPrim(4),false,"istPrim(4)")) result=false;
if (!assertEqual(istPrim(5),true, "istPrim(5)")) result=false;
if (!assertEqual(istPrim(9),false,"istPrim(9)")) result=false;
if (!assertEqual(istPrim(11),true,"istPrim(11)")) result=false;
return result;
}

bool test_Goldbach()
{ // d) {n=2;Goldbach=-1 | n=3;Goldbach=-1 | n=4;Goldbach=1 |
  //     n=5;Goldbach=-1 | n=5;Goldbach=-1 | n=100;Goldbach=6 |
  //     n=1000;Goldbach=28 }
bool result = true;
if (!assertEqual(Goldbach( 2),-1,"Goldbach(2)")) result=false;
if (!assertEqual(Goldbach( 3),-1,"Goldbach(3)")) result=false;
if (!assertEqual(Goldbach( 4), 1,"Goldbach(4)")) result=false;
```

```
if (!assertEqual(Goldbach( 5),-1,"Goldbach(5)")) result=false;
if (!assertEqual(Goldbach( 6),1,"Goldbach(6)")) result=false;
if (!assertEqual(Goldbach(100),6,"Goldbach(100)")) result=false;
if (!assertEqual(Goldbach(1000),28,"Goldbach(1000)")) result=false;
return result;
}

bool test_PythagTripel()
{ // e) {n=0;Pythag=0 | n=1;Pythag=0 | n=5;Pythag=1 | n=50;Pythag=26}
bool result = true;
if (!assertEqual(zeigePythagTripel(0),0,"zeige_PythagTripel(0)"))
  result=false;
if (!assertEqual(zeigePythagTripel(1),0,"zeige_PythagTripel(1)"))
  result=false;
if (!assertEqual(zeigePythagTripel(5),1,"zeige_PythagTripel(5)"))
  result=false;
if (!assertEqual(zeigePythagTripel(50),26,"zeige_PythagTripel(26)"))
  result=false;
return result;
}

bool test_f3nplus1()
{ // f) {n=1;f3nplus1=0 | n=2;f3nplus1=1 | n=3;f3nplus1=7 |
  //     n=4;f3nplus1=2 }
bool result = true;
if (!assertEqual(f3nplus1( 1,false), 0,"f3nplus1(1)")) result=false;
if (!assertEqual(f3nplus1( 2,false), 1,"f3nplus1(2)")) result=false;
if (!assertEqual(f3nplus1( 3,false), 7,"f3nplus1(3)")) result=false;
if (!assertEqual(f3nplus1( 4,false), 2,"f3nplus1(4)")) result=false;
return result;
}

bool test_3_4()
{
bool result = true;
if (!test_QuerSumme()) result=false;
if (!test_Fibonacci()) result=false;
if (!test_prim()) result=false;
if (!test_Goldbach()) result=false;
if (!test_PythagTripel()) result=false;
if (!test_f3nplus1()) result=false;
return result;
}


// --------------- Aufgabe 3 -------------------------------

/*
  Funktionen mit Zufallszahlen

  Funktionen, deren Ergebnis sehr aufwendig zu berechnen ist, wie
  z.B. Wettervorhersagen.
*/
```

11.3.6 Aufgabe 3.6

```
// File: C:\Loesungen_CB2006\Kap_3\3.6\FloatU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "FloatU.h"
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
//---------------------------------------------------------------------
#include "..\..\CppUtils\TestUtils.h"

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
#ifdef _DEBUG
bool test_3_6(); // Prototyp
initTestUtils();
testUtilsSummary(Memo1, test_3_6());
#endif
edDarlehensbetrag->Text=100;
edZinssatz->Text=5;
edTilgungsrate->Text=2;
}


// --------------- Aufgaben 3.6.5. -------------------------

// --------------- Aufgabe 1. ------------------------------

void __fastcall TForm1::Button_3_6_5_1Click(TObject *Sender)
{
int j=10,k=6;
double x=j,y=3.14,z=10;

double d1 = j/k;    // 1
double d2 = x/k;    // 1.666..
int i1 = j/k;       // 1
int i2 = x/k;       // 1
//    int i3 = 3(j+k);    // i3 = 3*(j+k) würde gehen
int i4 = j/k/k;     // 0
int i5 = j/(z/y);   // 3
int i6 = j/(y-j/k); // 4

Memo1->Lines->Add(FloatToStr(d1));
Memo1->Lines->Add(FloatToStr(d2));

Memo1->Lines->Add(FloatToStr(i1));
Memo1->Lines->Add(FloatToStr(i2));

Memo1->Lines->Add(FloatToStr(i4));
Memo1->Lines->Add(FloatToStr(i5));
Memo1->Lines->Add(FloatToStr(i6));
}


// --------------- Aufgabe 2. ------------------------------

// #include <cstdlib> // bei manchen Versionen des C++Builders notwendig für rand
double RegentropfenPi(int n)
{
int Treffer=0;
for (int i = 0; i<n; i++)
  {
    double x = (0.0+rand())/RAND_MAX;
    double y = (0.0+rand())/RAND_MAX;
    if (x*x+y*y < 1) Treffer++;
  }
return 4.0*Treffer/n;
}

void __fastcall TForm1::Button_3_6_5_2Click(TObject *Sender)
{
Memo1->Lines->Add(FloatToStr(RegentropfenPi(10000)));
}
```

```
// --------------- Aufgabe 3. ------------------------------

int Fakultaet(int n)
{
int f=1;
for (int i=2; i<=n; i++)
  f = f*i;
return f;
}

double doubleFakulaet(int n)
{
double f=1;
for (int i=2; i<=n; i++)
  f = f*i;
return f;
}

void __fastcall TForm1::Button_3_6_5_3Click(TObject *Sender)
{
for (int i=1; i<30; i++)
  Memo1->Lines->Add(IntToStr(i)+"! = "+IntToStr(Fakultaet(i)));

for (int i=1; i<30; i++)
  {
    double f = doubleFakulaet(i);
    double Sekunden = f/1000000.0;
    double Jahre = Sekunden/(60.0*60*24*365);
    Memo1->Lines->Add(IntToStr(i)+"! = "+FloatToStr(f)+
             " Sekunden="+FloatToStr(Sekunden)+
                 " Jahre="+FloatToStr(Jahre));
  }
}


// --------------- Aufgabe 4. ------------------------------

// der Datentyp Currency kann genauso wie der Datentyp double verwendet werden
void   HypothekenRestschuld(Currency   Betrag,   Currency   Zinssatz,   Currency
Tilgungsrate)
{
Currency Restschuld = Betrag;
Currency Annuitaet = Restschuld*(Zinssatz+Tilgungsrate)/100;
int Jahre = 0;

AnsiString        s="Hypothek:        "+CurrToStr(Restschuld)+":        Annuitaet:
"+CurrToStr(Annuitaet);
Form1->Memo1->Lines->Add(s);
while (Restschuld > 0)
  {
    Currency Zinsen = Restschuld*Zinssatz/100;
    Restschuld = Restschuld - (Annuitaet - Zinsen);
    Jahre++;
    s="Ende "+IntToStr(Jahre)+".Jahr: Restschuld="+CurrToStr(Restschuld)+
      " Zinsen="+CurrToStr(Zinsen)+" Tilgung="+CurrToStr(Annuitaet-Zinsen);
    Form1->Memo1->Lines->Add(s);
  }
};

void __fastcall TForm1::Button_3_6_5_4Click(TObject *Sender)
{
// Die Ergebnisse stimmen mit denen von Tilgungsrechnern im Internet
//                                  überein                                  (z.B.
http://www.interhyp.de/interhyp/servlet/interhyp?cmd=showPartnerTilgungsrechnerCo
ntent&LAF_PARTNER=sueddeutsche).
Currency Darlehen     = StrToCurr(edDarlehensbetrag->Text);
Currency Zinssatz     = StrToCurr(edZinssatz->Text);
Currency Tilgungsrate = StrToCurr(edTilgungsrate->Text);
HypothekenRestschuld(Darlehen, Zinssatz, Tilgungsrate);
```

```
}


// --------------- Aufgabe 5. ------------------------------

#include <cmath> // für fabs
// Man kann hier auch <math.h> einbinden, aber cmath ist etwas
// C++-spezifischer.

double TrapezSumme(double a, double b, int n)
{
double s = (std::sqrt(1-a*a) + std::sqrt(1-b*b))/2;
double h = (b-a)/n;
for (int i=1; i<n; i++)
  s = s + std::sqrt(1-(a+i*h)*(a+i*h));
return s*h;
}

void __fastcall TForm1::Button_3_6_5_5Click(TObject *Sender)
{
double t=TrapezSumme(0,1,10000);
Memo1->Lines->Add("T="+FloatToStr(4*t));
}


// --------------- Aufgabe 6. ------------------------------

double Mises(int n)
{
double q=1;
for (int i=0; i<n-1; i++)
  q = q*(364-i)/365;
return 1-q; // 0 für n<=1
}

void __fastcall TForm1::Button_3_6_5_6Click(TObject *Sender)
{
for (int i=1; i<50; i++)
//    if (i%5 == 0)
  Memo1->Lines->Add("n="+IntToStr(i)+"  p="+FloatToStr(Mises(i)));
}


// --------------- Aufgabe 7. ------------------------------

int RoundToInt(double x)
{
if (x >= 0)
  return x + 0.5;
else
  return x - 0.5;
}

void __fastcall TForm1::Button_3_6_5_7Click(TObject *Sender)
{
for (double x=-2; x<2; x=x+0.1)
  Memo1->Lines->Add(FloatToStr(x)+": "+IntToStr(RoundToInt(x)));
}


// --------------- Aufgabe 8. a) ---------------------------


int EstGrundtarif2005(int x)
{ // Einkommensteuertarif nach § 32a (1)
// Hier sind die Datentypen double und Currency angemessen.

// Mit diesem typedef kann der Gleitkommatyp für die Est-Berechnung leicht
// geändert werden, um die Currency- und die double-Version zu vergleichen.
```

```
// Bei den vorgegebenen Testfällen erhält man bei beiden Datentypen dasselbe
// Ergebnis.

typedef Currency Gleitkommatyp;
// typedef double Gleitkommatyp; // das geht auch

Gleitkommatyp est;
if (x <= 7664) est = 0;
  // Die Bedingung (0 <= x) && (x <= 7664) würde ESt für negative Werte
  // von x (negative Einkommen sind mit Abschreibungen oder Verlusten
  // möglich) undefiniert lassen.
else if (x <= 12739)
  { //     Die Bedingung (x > 7664) braucht hier nicht mehr geprüft zu
    // werden, da sie sich bereits aus der Negation der letzten Bedingung
    // ergibt.
    double y = (x-7664.0)/10000;
    est = (883.74*y+1500)*y;
  }
else if (x <= 52151)
  {
    double z = (x-12739.0)/10000;
    est = (228.74*z + 2397)*z +989;
  }
else est = 0.42*x - 7914;
return est;
};

bool EstGT2005_Vergl(int x,int Soll,bool log)
{
bool result=true;
AnsiString
msg="EstGrundtarif2005("+IntToStr(x)+")="+IntToStr(EstGrundtarif2005(x))+
                        ", Soll="+IntToStr(Soll);
if (log)
  Form1->Memo1->Lines->Add(msg);
if (!assertEqual_int(EstGrundtarif2005(x),Soll,msg)) result=false;

return result;
}

bool test_Est2005(bool log)
{
// http://home.t-online.de/home/parmentier.ffm/steuer.htm?steuer01.htm
// http://www.steuer.niedersachsen.de/Service/EStEuroTarif3.htm
//
http://www.ey.com/global/download.nsf/Germany/ST_Gestaltungsueberlegungen_zur_Jah
reswende_2004_2005/$file/Gestaltungsueberlegungen.pdf
bool result = true;

if (!EstGT2005_Vergl(-1,0,log)) result=false;
if (!EstGT2005_Vergl(0,0,log)) result=false;

if (!EstGT2005_Vergl(7663,0,log)) result=false;
if (!EstGT2005_Vergl(7664,0,log)) result=false;
if (!EstGT2005_Vergl(7665,0,log)) result=false;
if (!EstGT2005_Vergl(7704,6,log)) result=false;
if (!EstGT2005_Vergl(7740,11,log)) result=false;
if (!EstGT2005_Vergl(7812,22,log)) result=false;

if (!EstGT2005_Vergl(9684,339,log)) result=false;
if (!EstGT2005_Vergl(11736,757,log)) result=false;

if (!EstGT2005_Vergl(12708,981,log)) result=false;
if (!EstGT2005_Vergl(12740,989,log)) result=false;
if (!EstGT2005_Vergl(12741,989,log)) result=false;
if (!EstGT2005_Vergl(12744,990,log)) result=false;

if (!EstGT2005_Vergl(52151,13989,log)) result=false;
if (!EstGT2005_Vergl(52152,13989,log)) result=false;
```

```
if (!EstGT2005_Vergl(52153,13990,log)) result=false;
if (!EstGT2005_Vergl(52092,13964,log)) result=false;

if (!EstGT2005_Vergl( 20000, 2850,log)) result=false;
if (!EstGT2005_Vergl( 40000, 9223,log)) result=false;
if (!EstGT2005_Vergl( 60000,17286,log)) result=false;
if (!EstGT2005_Vergl( 80000,25686,log)) result=false;
if (!EstGT2005_Vergl(100000,34086,log)) result=false;
if (!EstGT2005_Vergl(120000,42486,log)) result=false;
if (!EstGT2005_Vergl(140000,50886,log)) result=false;
if (!EstGT2005_Vergl(160000,59286,log)) result=false;
if (!EstGT2005_Vergl(180000,67686,log)) result=false;
if (!EstGT2005_Vergl(200000,76086,log)) result=false;

return result;
}


// --------------- Aufgabe 8. b) ---------------------------

int EstSplittingtarif2005(double x)
{
return 2*EstGrundtarif2005(x/2);
}

bool EstSplitting2005_Vergl(int x,int Soll,bool log)
{
bool result=true;
AnsiString
msg="EstSplittingtarif2005("+IntToStr(x)+")="+IntToStr(EstSplittingtarif2005(x))+
                        ", Soll="+IntToStr(Soll);
if (log)
  Form1->Memo1->Lines->Add(msg);
if (!assertEqual_int(EstSplittingtarif2005(x),Soll,msg)) result=false;

return result;
}

bool test_Est2005_Splitting(bool log)
{
bool result = true;

//
http://www.ey.com/global/download.nsf/Germany/ST_Gestaltungsueberlegungen_zur_Jah
reswende_2004_2005/$file/Gestaltungsueberlegungen.pdf
// http://home.t-online.de/home/parmentier.ffm/steuer.htm?steuer01.htm
if (!EstSplitting2005_Vergl( 20000,  796,log)) result=false;
if (!EstSplitting2005_Vergl( 40000, 5700,log)) result=false;
if (!EstSplitting2005_Vergl( 60000,11614,log)) result=false;
if (!EstSplitting2005_Vergl( 80000,18446,log)) result=false;
if (!EstSplitting2005_Vergl(100000,26192,log)) result=false;
if (!EstSplitting2005_Vergl(120000,34572,log)) result=false;
if (!EstSplitting2005_Vergl(140000,42972,log)) result=false;
if (!EstSplitting2005_Vergl(160000,51372,log)) result=false;
if (!EstSplitting2005_Vergl(180000,59772,log)) result=false;
if (!EstSplitting2005_Vergl(200000,68172,log)) result=false;

return result;
}

void __fastcall TForm1::Button_3_6_5_8Click(TObject *Sender)
{
test_Est2005(true);
test_Est2005_Splitting(true);
}



// --------------- Aufgabe 9. ------------------------------
```

```
void __fastcall TForm1::Button_3_6_5_9Click(TObject *Sender)
{
int n = 1000000;
float f1=0,f2=0;
double d1=0,d2=0;
long double l1=0,l2=0;
for (int i= 1; i<=n; i++)
  {
    long double l = (1.0L/i)*(1.0L/i);
    l = 1.0L/((long double)i*(long double)(i));
    // 1/(i*i) ergibt Division by 0 für i=65536
    float f = l;
    double d = l;
    f1 = f1 + f;
    d1 = d1 + d;
    l1 = l1 + l;
  }
for (int i=n; i>=1; i--)
  {
    long double l = (1.0L/i)*(1.0L/i);
    float f = l;
    double d = l;
    f2 = f2 + f;
    d2 = d2 + d;
    l2 = l2 + l;
  }

Memo1->Lines->Add("n="+IntToStr(n));
Memo1->Lines->Add("float aufsteigend:       "+FloatToStr(f1));
Memo1->Lines->Add("float absteigend:        "+FloatToStr(f2)+
    " diff="+FloatToStr(std::fabs(f1-f2)));

Memo1->Lines->Add("double aufsteigend:      "+FloatToStr(d1));
Memo1->Lines->Add("double absteigend:       "+FloatToStr(d2)+
    " diff="+FloatToStr(std::fabs(d1-d2)));

Memo1->Lines->Add("long double aufsteigend: "+FloatToStr(l1));
Memo1->Lines->Add("long double absteigend:  "+FloatToStr(l2)+
    " diff="+FloatToStr(std::fabsl(l1-l2)));
Memo1->Lines->Add("exakt:                   "+FloatToStr(M_PI*M_PI/6));
/*
float aufsteigend:       1.64472532272339
float absteigend:        1.64493298530579 diff=0.000207662582397461
double aufsteigend:      1.64493306684877
double absteigend:       1.64493306684873 diff=4.35207425653061E-14
long double aufsteigend: 1.64493306684877
long double absteigend:  1.64493306684873 diff=4.35207425653061E-14
exakt:                   1.64493406684823
*/
Memo1->Lines->Add("");
}


// --------------- Aufgabe 10. ----------------------------

void HowLongDouble()
{
double x = 1e8;
while(x > 0)
  --x;
}

void HowLongFloat()
{
float x = 1e8;
while(x > 0)
  --x;
// Diese Schleife (mit float anstelle double) ist eine Endlosschleife.
```

```
// Da float nur 6 signifikante Stellen hat, ist 100000000 - 1 == 100000000.
}

void __fastcall TForm1::Button_3_6_5_10Click(TObject *Sender)
{
HowLongDouble();
// HowLongFloat();
}


// --------------- Aufgaben 3.6.6. -------------------------

// --------------- Aufgabe 1. ------------------------------

/*
2.  RegentropfenPi

    Funktionen mit Zufallszahlen sind oft schwer zu testen. Zwar muss bei dieser
    Aufgabe ein Wert in der Nähe von 3.14 zurückgegeben werden. Aber ein Fehler
    wie

      for (int i = 1; i<n; i++) anstelle von
      for (int i = 0; i<n; i++)

    mit einer anschließenden Division durch n

      return 4.0*Treffer/n;

    ist mit Tests nur schwer zu finden.

3. {n=0;Fakultaet=1 | n=1;Fakultaet=1 | n=2;Fakultaet=2 |
    n=3;Fakultaet=6 }
   {n=0;doubleFakulaet=1 | n=1;doubleFakulaet=1 |
    n=2;doubleFakulaet=2 | n=3;doubleFakulaet=6 }

4. HypothekenRestschuld(Currency Betrag, Currency Zinssatz,
                        Currency Tilgungsrate)

   Das Ergebnis der Funktion HypothekenRestschuld besteht aus vielen
   Werten, die nicht wie bei den meisten der bisherigen Aufgaben als
   Funktionswert zurückgegeben werden können.
   Die manuelle Berechnung der Ergebnisse ist recht mühsam:

   Hypothek: 100: Zinsen=5, Tilgung=10, Annuität 15
                Restschuld   Zinsen     Tilgung
   Ende 1.Jahr:    90          5          10
   Ende 2.Jahr:    79.5        4.5        10.5
   Ende 3.Jahr:    68.475      3.975      11.025
   Ende 4.Jahr:    56.8987     3.4237     11.5763
   Ende 5.Jahr:    44.7436     2.8449     12.1551
   Ende 6.Jahr:    31.9807     2.2371     12.7629
   Ende 7.Jahr:    18.5797     1.599      13.401
   Ende 8.Jahr:     4.5086     0.9289     14.0711
   Ende 9.Jahr:                0.2254     Schlusszahlung 14.30

5. Trapezsumme

   Hier ist zwar klar, dass ein Wert in der Nähe von 3.14 zurückgegeben
   werden muss. Aber kleine, systematische Fehler können mit Tests nur
   schwer entdeckt werden (siehe die Anmerkungen zu Aufgabe 2. oben).
   Für größere Werte von n sind Tests sehr aufwendig.

   Eine Alternative können Tests sein, bei denen die Fläche unter
   solchen Funktionen berechnet wird, bei denen man die Fläche auch exakt
   über das Integral berechnen kann. Z.B. ist die Fläche von a bis b unter
   Funktion x^2 durch (b^3 - a^3)/3 gegeben.

   Wir werden aber später (siehe Abschnitt 5.2) eine verallgemeinerte
   Version dieser Funktion (mit Funktionszeigern) kennenlernen. Diese
```

```
   Funktion kann dann relativ leicht getestet werden.


6. {n=1,Mises=0 | n=1,Mises=1-364.0/365 |
    n=2,Mises=1-(364.0*363)/(365.0*365) | n=3,Mises=1-(364.0*363*362)/
                                                        (365.0*365*365) }

7.  {n=3.64,RoundToInt=4 | n=3.14,RoundToInt=3 |
     n=-3.14,RoundToInt=-3 | n=-3.64,RoundToInt=-4 }

8. Steuerformel: Hier können die in der Tabelle angegebenen Werte
     zum Testen verwendet werden.

9. Reihenfolge der Summation:

   Schwierig zu testen, da das Ergebnis nicht bekannt ist, wenn man
   nicht zufällig weiß, dass es pi^2/6 ist.
*/


// --------------- Aufgabe 2. ------------------------------


// 2. Für Funktionen mit Zufallszahlen sind keine automatischen Tests
//    sinnvoll.

bool test_Fakultaet()
{// 3. {n=0,Fakultaet=1 | n=1,Fakultaet=1 | n=2,Fakultaet=2 |
 //     n=3,Fakultaet=6 }
bool result = true;
if (!assertEqual_double(Fakultaet(0),1,"Fakultaet(0)")) result=false;
if (!assertEqual_double(Fakultaet(1),1,"Fakultaet(1)")) result=false;
if (!assertEqual_double(Fakultaet(2),2,"Fakultaet(2)")) result=false;
if (!assertEqual_double(Fakultaet(3),6,"Fakultaet(3)")) result=false;

// 3. {n=0,doubleFakulaet=1 | n=1,doubleFakulaet=1 |
//     n=2,doubleFakulaet=2 | n=3,doubleFakulaet=6 }
if (!assertEqual_double(doubleFakulaet(0),1,"doubleFakulaet(0)"))
  result=false;
if (!assertEqual_double(doubleFakulaet(1),1,"doubleFakulaet(1)"))
  result=false;
if (!assertEqual_double(doubleFakulaet(2),2,"doubleFakulaet(2)"))
  result=false;
if (!assertEqual_double(doubleFakulaet(3),6,"doubleFakulaet(3)"))
  result=false;
return result;
}

// 4. Mit den bisher vorgestellten Sprachelementen sind keine
//    automatischen Tests möglich

// 5. Mit den bisher vorgestellten Sprachelementen sind keine
//    automatischen Tests möglich


bool test_Mises()
{// 6. {n=1,Mises=0 | n=1,Mises=1-364.0/365 |
 //     n=2,Mises=1-(364.0*363)/(365.0*365) |
 //     n=3,Mises=1-(364.0*363*362)/(365.0*365*365) }
bool result = true;
if (!assertEqual_double(Mises(1),0,"Mises(1)")) result=false;
if (!assertEqual_double(Mises(2),1-364.0/365,"Mises(2)")) result=false;
if (!assertEqual_double(Mises(3),1-(364.0*363)/(365.0*365),"Mises(3)"))
 result=false;
if (!assertEqual_double(Mises(4),1-(364.0*363*362)/(365.0*365*365),
  "Mises(4)")) result=false;
return result;
}
```

```
bool test_RoundToInt()
{ // 7.  {n=3.64,RoundToInt=4 | n=3.14,RoundToInt=3 |
  //     n=-3.14,RoundToInt=-3 | n=-3.64,RoundToInt=-4 }
bool result = true;
if (!assertEqual_int(RoundToInt(3.64),4,"RoundToInt(3.64)")) result=false;
if (!assertEqual_int(RoundToInt(3.14),3,"RoundToInt(3.14)")) result=false;
if (!assertEqual_int(RoundToInt(-3.14),-3,"RoundToInt(3.14)")) result=false;
if (!assertEqual_int(RoundToInt(-3.64),-4,"RoundToInt(3.64)")) result=false;
return result;
}

/*
8. Die Testfunktionen sind (siehe oben)

     test_Est2005(true);
     test_Est2005_Splitting(true);
*/


// --------------- Aufgabe 3. ------------------------------

bool test_NearlyEqual_Example_Section_3_6_6()
{ // test cases from example 3.6.6
bool result = true;
if (!assertEqual_bool(NearlyEqual(9.9, 9.3, 2),false,
  "NearlyEqual(9.9, 9.3, 2)")) result=false;
if (!assertEqual_bool(NearlyEqual(9.9, 9.5, 2),true,
  "NearlyEqual(9.9, 9.5, 2)")) result=false;
if (!assertEqual_bool(NearlyEqual(9.9, 9.3, 3),false,
  "NearlyEqual(9.9, 9.3, 3)")) result=false;
if (!assertEqual_bool(NearlyEqual(9.9, 9.5, 3),false,
  "NearlyEqual(9.9, 9.5, 3)")) result=false;
if (!assertEqual_bool(NearlyEqual(10.1, 9.7, 2),true,
  "NearlyEqual(10.1, 9.7, 2)")) result=false;
if (!assertEqual_bool(NearlyEqual(10.1, 9.5, 2),false,
  "NearlyEqual(10.1, 9.5, 2)")) result=false;
if (!assertEqual_bool(NearlyEqual(0, 0.01, 2),true,
  "NearlyEqual(0, 0.01, 2)")) result=false;
if (!assertEqual_bool(NearlyEqual(0, 0.01, 3),false,
  "NearlyEqual(0, 0.01, 3)")) result=false;
return result;
}

bool test_assertEqual_Significant_Digits(const double d)
{
bool result=true;
double delta=1;
int sigd=1;

for (int i=1; i<=14; i++)
  {
    delta=delta/10;
    sigd=sigd+1;
    AnsiString Msg=" sig_digits="+IntToStr(sigd)+" delta="+
      FloatToStr(delta)+" d="+FloatToStr(d)+")";
    AnsiString txt1="assertNotEqual_double(d,";
    if (!assertNotEqual_double(d,d*(1-0.6*delta),
           txt1+"d*(1-0.6*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
    txt1="assertEqual_double(d,";
    if (!assertEqual_double(d,d*(1-0.4*delta),
           txt1+"d*(1-0.4*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d*(1-0.3*delta),
           txt1+"d*(1-0.3*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d*(1-0.2*delta),
           txt1+"d*(1-0.2*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d*(1-0.1*delta),
           txt1+"d*(1-0.1*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d*(1-0.0*delta),
           txt1+"d*(1-0.0*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
```

```
    if (!assertEqual_double(d,d*(1+0.1*delta),
           txt1+"d*(1+0.1*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d*(1+0.2*delta),
           txt1+"d*(1+0.2*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d*(1+0.3*delta),
           txt1+"d*(1+0.3*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d*(1+0.4*delta),
           txt1+"d*(1+0.4*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
//    if (!assertEqual_double(d,d*(1+0.4*delta),
//             txt1+"d*(1+0.4*delta)),"+Msg,Form1->Memo1,sigd)) result=false;
//  varying results
    txt1="assertNotEqual_double(d,";
    if (!assertNotEqual_double(d,d*(1+0.6*delta),
           txt1+"d*(1+0.6*delta)),"+Msg,Form1->Memo1,sigd)) result=false;

    // Testing NearlyEqual with one more significant digit should always
    // return false

    txt1="assertNotEqual_double(d,";
    if (!assertNotEqual_double(d,d*(1-0.6*delta),
           txt1+"d*(1-0.6*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1-0.5*delta),
           txt1+"d*(1-0.5*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1-0.4*delta),
           txt1+"d*(1-0.4*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1-0.3*delta),
           txt1+"d*(1-0.3*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1-0.2*delta),
           txt1+"d*(1-0.2*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1-0.1*delta),
           txt1+"d*(1-0.1*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1+0.1*delta),
           txt1+"d*(1+0.1*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1+0.2*delta),
           txt1+"d*(1+0.2*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1+0.3*delta),
           txt1+"d*(1+0.3*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1+0.4*delta),
           txt1+"d*(1+0.4*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1+0.4*delta),
           txt1+"d*(1+0.4*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d*(1+0.6*delta),
           txt1+"d*(1+0.6*delta)),"+Msg,Form1->Memo1,sigd+1)) result=false;
  }
return result;
}

bool test_assertEqual_with_zero()
{
// For tests with d==0 the expression d*(1+0.i*delta) in
// Test_NearlyEqual_Significant_Digits is always zero.
double d=0;

bool result=true;
double delta=1;
int sigd=0; // start with sigd=0 instead of sigd=1 in
            // Test_NearlyEqual_Significant_Digits since d==0 has one
            // significant digit less than d!=0
for (int i=1; i<=14; i++)
  {
    delta=delta/10;
    sigd=sigd+1;
    AnsiString Msg=" sig_digits="+IntToStr(sigd)+" delta="+
      FloatToStr(delta)+" d="+FloatToStr(d)+")";

    AnsiString txt1="assertNotEqual_double(d,";
    if (!assertNotEqual_double(d,d-6*delta,txt1+"d-6*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
```

```
    txt1="assertEqual_double(d,";
    if (!assertEqual_double(d,d-4*delta,txt1+"d-4*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d-3*delta,txt1+"d-3*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d-2*delta,txt1+"d-2*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d-1*delta,txt1+"d-1*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d-0*delta,txt1+"d-0*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d+1*delta,txt1+"d+1*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d+2*delta,txt1+"d+2*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d+3*delta,txt1+"d+3*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
    if (!assertEqual_double(d,d+4*delta,txt1+"d+4*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;

//    if (!assertEqual_double(d,d+5*delta,"assEq_double(d,d+5*delta),"+Msg,
//                      Form1->Memo1,sigd)) result=false; // varying results

    txt1="assertNotEqual_double(d,";
    if (!assertNotEqual_double(d,d+6*delta,txt1+"d+6*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;

    // Testing NearlyEqual with one more significant digit should always
    // return false
    Msg=" sig_digits+1="+IntToStr(sigd+1)+" delta="+FloatToStr(delta)+
        " d="+FloatToStr(d)+")";

    txt1="assertNotEqual_double(d,";
    if (!assertNotEqual_double(d,d-6*delta,txt1+"d-6*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
    if (!assertNotEqual_double(d,d-5*delta,txt1+"d-6*delta),"+Msg,
                            Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d-4*delta,txt1+"d-4*delta),"+Msg,
                            Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d-3*delta,txt1+"d-3*delta),"+Msg,
                            Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d-2*delta,txt1+"d-2*delta),"+Msg,
                            Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d-1*delta,txt1+"d-1*delta),"+Msg,
                            Form1->Memo1,sigd+1)) result=false;

    if (!assertNotEqual_double(d,d+1*delta,txt1+"d+1*delta),"+Msg,
                            Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d+2*delta,txt1+"d+2*delta),"+Msg,
                            Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d+3*delta,txt1+"d+3*delta),"+Msg,
                            Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d+4*delta,txt1+"d+4*delta),"+Msg,
                            Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d+5*delta,txt1+"d+5*delta),"+Msg,
                            Form1->Memo1,sigd+1)) result=false;
    if (!assertNotEqual_double(d,d+6*delta,txt1+"d+6*delta),"+Msg,
                            Form1->Memo1,sigd)) result=false;
  }
return result;
}

bool test_NearlyEqual_with_small_numbers(int n)
{
bool result = true;
for (int i=-n; i<=n; i++)
  {
  double d=rand()/10000.0;
  if (NearlyEqual(i+d,0,10))
```

```
    if (!test_assertEqual_Significant_Digits(i+d))
      result = false;
  }
return result;
}

bool test_NearlyEqual_with_large_numbers(int n)
{
bool result = true;
for (int i=1; i<=n; i++)
  {
  double d1=rand();
  double d2=std::pow(10.0,rand()%300);
  double d3=rand()/10000;
  // test large positive random numbers
  if (!test_assertEqual_Significant_Digits(d1*d2+d3))
    result = false;
  // test large negative random numbers
  if (!test_assertEqual_Significant_Digits(-d1*d2+d3))
    result = false;
  // test small positive random numbers
  if (!test_assertEqual_Significant_Digits(d1/d2+d3))
    result = false;
  // test small negative random numbers
   if (!test_assertEqual_Significant_Digits(-d1/d2+d3))
    result = false;
 }
return result;
}

bool test_NearlyEqual_Calculations()
{ // check some mathematical identities
using namespace std;
bool result = true;
int significant_digits=16;
for (double x=-10; x<=10; x=x+0.01)
  {
    double x2=sin(x)*sin(x);
    x2=x2+1;
    double y2=cos(x)*cos(x);
    y2=y2-1;
    if (!assertEqual_double(x2+y2,1,"sin^2+cos^2 ",
                Form1->Memo1,significant_digits)) result=false;
    // All tests pass with significant_digits=16
    // With significant_digits=17 some tests fail
  }
for (double x=-10; x<=10; x=x+0.01)
  {
    double x2=exp(x);
    double y2=log(x2);
    if (!NearlyEqual(x,0,5)) // Test for x=0 fails
    if (!assertEqual_double(y2,x,"exp(ln(x)) "+FloatToStr(x),
                Form1->Memo1,significant_digits-2)) result=false;
    // All tests pass with significant_digits=14
    // With significant_digits=15 some tests fail
  }
for (int n=1; n<10000; n++)
  {
    double s=0;
    for (int i=1; i<=n; i++)
      s=s+i/10.0;
    if (!assertEqual_double(s,n*(n+1)/20.0,"1+..."+FloatToStr(n),
                Form1->Memo1,significant_digits-2)) result=false;
    // All tests pass with significant_digits=16
    // With significant_digits=17 some tests fail
  }
return result;
}
```

```
bool test_NearlyEqual()
{
bool result = true;
if (!test_NearlyEqual_Example_Section_3_6_6()) result=false;
if (!test_NearlyEqual_with_small_numbers(10000)) result=false;
if (!test_assertEqual_with_zero()) result=false;
if (!test_NearlyEqual_with_large_numbers(100)) result=false;
if (!test_NearlyEqual_Calculations()) result=false;
return result;
}

bool test_3_6()
{
bool result = true;
if (!test_Fakultaet())    result=false;
if (!test_RoundToInt())   result=false;
if (!test_Mises())        result=false;
if (!test_Est2005(false)) result=false;
if (!test_Est2005_Splitting(false)) result=false;
if (!test_NearlyEqual())  result=false;
return result;
}

void __fastcall TForm1::TestClick(TObject *Sender)
{
initTestUtils();
testUtilsSummary(Memo1, test_3_6());
}
```

11.3.7 Aufgabe 3.7

```
// File: C:\Loesungen_CB2006\Kap_3\3.7\LogicU.cpp

#include <vcl.h>
#pragma hdrstop

#include "LogicU.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------


// --------------- Aufgaben 3.7.1 --------------------------
// --------------- Aufgaben 3.7.2 --------------------------
// --------------- Aufgaben 3.7.3 --------------------------

// Siehe "..\..\kap_3\3.7\3_7.rtf"


// --------------- Aufgaben 3.7.4 --------------------------

#include <iso646.h>

#include "..\..\CppUtils\State.h"
```

```
int mul(int x, int y)
{ // x==x0, y==y0, x0>=0
int z=0 ;
STATE_2(s, int,x, int,y);
while (x !=0)
  { // z + x*y == x0*y0
    if (x&1) z = z + y;
    y = y*2;
    x = x/2;
    // z + x*y == x0*y0
    AnsiString        msg="mul        x="+IntToStr(x)+",        y="+IntToStr(y)+",
z+x*y="+IntToStr(z+x*y)+", s.x*s.y="+IntToStr(s.x*s.y);
    CheckCondition(z+x*y==s.x*s.y, msg.c_str());
  } //x==0 ==> z==x0*y0
return z;
}

int min2a(int a, int b)
{ // Keine weiteren Vorbedingungen, d.h. true
STATE_2(s, int,a, int,b); // speichert die aktuellen Werte von x und y
int m;
if (a<b) m=a;
else m=b;
CheckCondition((m<=s.a) and (m<=s.b), "min2a");
return m;
} // min2a(a,b)==Minimum(a,b)

void test_3_7_4(int n)
{
for (int i=0; i<n; i++)
  {
    int x=rand(), y=rand();
    mul(x%1000,abs(y)%1000);
    min2a(x,y);
  }
Form1->Memo1->Lines->Add(IntToStr(n)+" tests performed");
}

void __fastcall TForm1::Button_3_7_4Click(TObject *Sender)
{
test_3_7_4(1000);
}

// --------------- Aufgabe 3.7.4, 2. ------------------------
// --------------- Aufgaben 3.7.5 --------------------------

// Siehe "..\..\kap_3\3.7\3_7.rtf"


// --------------- Aufgaben 3.7.6 --------------------------

// --------------- Aufgabe 3.7.6, 1. ------------------------

// Siehe "..\..\kap_3\3.7\3_7.rtf"

// --------------- Aufgabe 3.7.6, 2. ------------------------

bool Formel_a()
{
bool result=true;
for (int p=true;p>=false;p--)
 for (int q=true;q>=false;q--)
  for (int r=true;r>=false;r--)
   if ( (p and (q or r)) != ((p and q) or (p and r)) ) result=false;
return result;
}

bool Formel_b()
{
```

```
bool result=true;
for (int p=true;p>=false;p--)
 for (int q=true;q>=false;q--)
  for (int r=true;r>=false;r--)
   if ( ( p or (q and r)) != ( (p or q) and (p or r) ) ) result=false;
return result;
}

bool Formel_c()
{
bool result=true;
for (int p=true;p>=false;p--)
 for (int q=true;q>=false;q--)
  for (int r=true;r>=false;r--)
   if ( ( (p and q) or r) != (p and (q or r)) ) result=false;
return result;
}

void __fastcall TForm1::Button_3_7_6Click(TObject *Sender)
{
bool a=Formel_a();
bool b=Formel_b();
bool c=Formel_c();
Form1->Memo1->Lines->Add(" Formel_a: "+BoolToStr(a,true));
Form1->Memo1->Lines->Add(" Formel_b: "+BoolToStr(b,true));
Form1->Memo1->Lines->Add(" Formel_c: "+BoolToStr(c,true));
}

// --------------- Aufgaben 3.7.7 --------------------------

// Siehe "..\..\kap_3\3.7\3_7.rtf"
```

Lösung Aufgaben 3.7.2

1. a) Ablaufprotokoll für *Fakultaet(4)*

```
int Fakultaet(int n)
{
int f=1;
for (int i=2; i<=n; i++)
  f = f*i;
return f;
}
```

	n	***f***	***i***	***i<=n***	***result***
	4				
int f=1;		1			
for (int i=2; i<=n; i++)					
i=2;			2		
i<=n;				true	
f=f*i;		1*2=2			
i++;			3		
i<=n;				true	
f=f*i;		1*2*3=6			
i++;			4		
i<=n;				true	
f=f*i;		1*2*3*4=24			
i++;			5		
i<=n;				false	
return f;					24

2. a) Nach jeder Ausführung des Schleifenkörpers bestehen zwischen i und f diese Beziehungen:

i = 3: f = 1*2 = 1*...*(i-1)

i = 4: f = 1*2*3 = 1*...*(i-1)

i = 5: f = 1*2*3*4 = 1*...*(i-1)

1. b) Ablaufprotokoll für *Mises(3)*

```
double Mises(int n)
{
double q=1;
for (int i=0; i<n-1; i++)
  q = q*(364-i)/365;
return 1-q; // 0 für n<=1
}
```

	n	***q***	***i***	***i<n-1***	***result***
	3				
`double q=1;`		1			
`for (int i=0; i<n-1; i++)`					
`i=0;`			0		
`i<n-1;`				true	
`q = q*(364-i)/365;`		1*364/365			
`i++;`			1		
`i<n-1;`				true	
`q = q*(364-i)/365;`		(1*364*363)/(365*365)			
`i++;`			2		
`i<=n;`				false	
`return 1-q;`					1-q

2. b) Nach jeder Ausführung des Schleifenkörpers ist q das folgende Produkt aus i Faktoren im Zähler und Nenner:

i = 1: $q = 364/365 = 364*...*(364 - i + 1)/365^i$

i = 2: $q = (364*363)/(365*365) = 364*...*(364 - i + 1)/365^i$

1. c) Ablaufprotokoll für *Fibonacci(4)*

	x	y	f	i	i<n
int f=0,x=0,y=1;	0	1	0 // f_0		
for (int i=0 ;i<n ;i++)					
i=0 ;				0	
i<n					true
x=y;	1				
y=f;		0			
f=x+y ;			1+0=1 // f_1		
i++ ;				1	
i<n					true
x=y;	0				
y=f;		1			
f=x+y			0+1 // f_2		
i++ ;				2	
i<n					true
x=y;	1				
y=f;		1			
f=x+y			1+1=2 // f_3		
i++ ;				3	
i<n					true
x=y;	1				
y=f;		2			
f=x+y;			1+2=3 // f_4		
i++ ;				4	
i<n					false

2. c) Ab der zweiten Ausführung des Schleifenkörpers bestehen zwischen f und i diese Beziehungen:

i = 2: $f = 1 = f_2 = 1+0 = f_1+f_0$, d.h. $f_i=f_{i-1}+f_{i-2}$, $x=f_{i-2}$, $y=f_{i-1}$

i = 3: $f = 2 = f_3 = 1+1 = f_2+f_1$, d.h. $f_i=f_{i-1}+f_{i-2}$ $x=f_{i-2}$, $y=f_{i-1}$

i = 4: $f = 3 = f_4 = 2+1 = f_3+f_2$, d.h. $f_i=f_{i-1}+f_{i-2}$ $x=f_{i-2}$, $y=f_{i-1}$

1. d) Ablaufprotokoll für *Quersumme(289)*

```
int Quersumme(int n)
{
if (n<0) n=-n;
int s=0;
while (n>0)
  {
    s = s+n%10;
    n = n/10;
  }
return s;
}
```

	n	s	n>0	result
	289			
if (n<0) n=-n;				
int s=0;		0		
while (n>0)				
n>0;			true	
s = s+n%10;		0+9=9		
n = n/10;	28			
n>0;			true	
s = s+n%10;		0+9+8=17		
n = n/10;	2			
n>0;			true	
s = s+n%10;		0+9+8+2=19		
n = n/10;	0			
n>0;			false	
return s;				19

2. d) Da sich hier das Ergebnis aus den einzelnen Ziffern ergibt, liegt es nahe, den Parameter durch seine Ziffern in der Form $z_n z_{n-1} ... z_2 z_1 z_0$ darzustellen. Nach jeder Ausführung des Schleifenkörpers bestehen zwischen der Anzahl der Schleifenwiederholungen (die als i bezeichnet werden soll, und die nicht als Variable im Programm in vorkommt, die aber leicht eingeführt werden könnte) und s diese Beziehungen:

$i = 1: \quad s = 9 = z_0 = z_0 + ... + z_{i-1}$

$i = 2: \quad s = 9 + 8 = z_0 + z_1 = z_0 + ... + z_{i-1}$

$i = 3: \quad s = 9 + 8 + 2 = z_0 + z_1 + z_2 = z_0 + ... + z_{i-1}$

1. e) Ablaufprotokoll für *prim(17)*

```
bool prim(int n)
{
if (n<2) return false;
else if (n==2) return true;
else
  {
    for (int i=2; i*i<=n; i++)
      if (n%i==0) return false;
    return true;
  }
}
```

Da die Bedingungen n<2 und n==2 nicht erfüllt sind, wird nur die *for*-Schleife protokolliert.

	n	***i***	***i*i<n***	***n%i==0***	***result***
	17				
`for (int i=2; i*i<=n; i++)`					
`i=2;`		2			
`i*i<=n;`			true		
`if (n%i==0) return false;`				false	
`i++;`		3			
`i*i<=n;`			true		
`if (n%i==0) return false;`				false	
`i++;`		4			
`i*i<=n;`			true		
`if (n%i==0) return false;`				false	
`i++;`		5			
`i*i<=n;`			false		
`return true;`					true

1. e) Ablaufprotokoll für *prim(18)*

	n	***i***	***i*i<n***	***n%i==0***	***result***
	18				
`for (int i=2; i*i<=n; i++)`					
`i=2;`		2			
`i*i<=n;`			true		
`if (n%i==0) return false;`					false

2. e) Nach der i-ten Ausführung des Schleifenkörpers besteht zwischen i und n die Beziehung:

n ist nicht durch 2..i teilbar.

2. Die Lösungen dieser Aufgaben sind unmittelbar nach den entsprechenden Teilen von Aufgabe 1 angegeben.

3. a) Falls eine Funktion Werte verwendet, die nicht bekannt sind, kann man für sie kein Ablaufprotokoll erstellen. Das gilt z.B. für Funktionen, die Zufallswerte verwenden, wie die Funktion *RegentropfenPi* (Aufgabe 3.6.5, 2.).

b) In der Aufgabe zum 3n+1-Problem (Aufgabe 3.4.6, 6.) wurde darauf hingewiesen, dass sich an dieser Aufgabe schon Generationen von Mathematikern die Zähne ausgebissen haben:

```
int f3nplus1(int n, bool show)
{
if (n<=0) return -1;
int m=0;
while (n!=1)
  {
    m++;
    if (n%2==1) n=3*n+1;
    else n=n/2;
    if (show) Form1->Memo1->Lines->Add(" n="+IntToStr(n));
  }
return m;
}
```

Deswegen ist es vermutlich schwierig, für diese Funktion eine allgemeingültige Beziehung zwischen m und n zu finden. Die einzige Beziehung, die man unmittelbar angeben kann, ist „n ist der Wert nach m-ten Ausführung des Schleifenkörpers“. Diese Aussage ist aber nicht sehr aussagekräftig.

4. Wenn ich meine Studenten frage, ob die Anweisungen

```
int x=1,y=2;
x=x+y;
y=x-y;
x=x-y;
```

die Werte von x und y bei beliebigen Werten vertauschen, sind sich nur die wenigsten Teilnehmer sicher, ob das zutrifft oder nicht. Mit den bisher behandelten Mitteln ist eine Antwort schwierig. Sie folgt in Abschnitt 3.7.2.

Lösung Aufgaben 3.7.3

1.a) n ungerade, d.h. n/2 = $\frac{n-1}{2}$:

	x	n	p
	x_0	n_0	p_0
p = p*x			p_0*x_0
n = n/2		$\frac{n0-1}{2}$	
x = x*x	x_0*x_0		

$$p*x^n = (p_0*x_0)*(x_0*x_0)^{(n0-1)/2} = p_0*x_0*x_0^{no-1} = p_0*x_0^{n0} = u^v$$

b) n gerade, d.h. n/2 = $\frac{n}{2}$:

	x	n	p
	x_0	n_0	p_0
n = n/2		$\frac{n0}{2}$	
x = x*x	x_0*x_0		

$$p*x^n = p_0*(x_0*x_0)^{(n0/2)} = p_0*x_0^{n0} = u^v$$

c) s = 1 + 2 + ... + i

	s	i
	$1+2+...+i_0$	i_0
i = i + 1		i_0+1
s = s + i	$1+2+...+i_0+(i_0+1)$	

Wegen $i=i_0+1$ gilt anschließend

$$s = 1 + 2 + ... + i_0 + (i_0 + 1) = 1 + 2 + ... + (i–1) + i$$

d) $\mathbf{s} = 1^2 + 2^2 + ... + i^2$

	s	i
	$1^2+2^2+...+i_0^2$	i_0
i = i + 1		$i_0 + 1$
s = s + i*i	$1^2+2^2+...+i_0^2+(i_0+1)^2$	

Wegen $i=i_0+1$ gilt anschließend

$$s = 1^2 + 2^2 + ... + i_0^2 + (i_0 + 1)^2 = 1^2 + 2^2 + ... + (i–1)^2 + i^2$$

e) $\mathbf{s} = 1^2 + 2^2 + ... + i^2$

	s	i
	$1^2+2^2+...+i_0^2$	i_0
s = s + i*i	$1^2+2^2+...+i_0^2+i_0^2$	
i = i + 1		$i_0 + 1$

Wegen $i=i_0+1$ gilt anschließend

$$s = 1^2 + 2^2 + ... + i0^2 + i0^2 = 1^2 + 2^2 + ... + 2*(i–1)^2$$

und das ist nicht die gefragte Invarianz.

2. Mögliche Kriterien zum Vergleich der beiden Funktionen *vertausche* und *vertausche1*:

– Verständlichkeit: Das erste Verfahren erscheint einfacher.
– Allgemeinheit: Das erste Verfahren lässt sich auf alle Datentypen anwenden, das zweite nur auf arithmetische. Aber auch bei arithmetischen Datentypen ist das erste Verfahren allgemeiner, da keine Bereichsüberschreitungen auftreten können.
– Ausführungszeit: Beim ersten Verfahren sind drei Zuweisungen auszuführen, beim zweiten zusätzlich drei Additionen.
– Speicherplatzbedarf: Das zweite Verfahren braucht zwar keinen Speicherplatz für eine dritte Variable, aber die Anweisungen für die drei Rechenoperationen benötigen auch Programmcode und damit Speicherplatz.

Lösung Aufgaben 3.7.4

1. a) Dieses Ablaufprotokoll zeigt, dass die in Aufgabe 3.7.1, 2a) formulierte Beziehung zwischen i und f nicht nur für i=3, 4 und 5 gilt, sondern für beliebige Werte von i>=2, wenn kein Überlauf stattfindet:

Induktionsbehauptung: f==1*2*...*(i-1)

```
int Fakultaet(int n)
{
int f=1;
for (int i=2; // Induktionsanfang: Mit i==2 gilt f==1*2*...*(i-1)==1
              i<=n; i++)
 {            // i      f
              // i0     1*2*...*(i0-1)
  f = f*i;              1*2*...*(i0-1)*i0
  // i++;     // i0+1
              /// i==i0+1, d.h. i0==i-1
              /// f==1*2*...*(i-1)
 }
/// i==n+1 ==> f == 1*2*...*(i-1) == 1*2*...*(n+1-1) == 1*2*...*n
return f;
}
```

Falls der Schleifenkörper also mindestens einmal ausgeführt wird (d.h. n>=2), erhält man den Funktionswert 1*2* ... *(n-1)*n. Andernfalls erhält man den Funktionswert 1.

b) Das folgende Ablaufprotokoll zeigt, dass die in Aufgabe 3.7.1, 2b) formulierte Beziehung zwischen i und q nicht nur für i=1 und 2 gilt, sondern für beliebige Werte von i>0, falls kein Überlauf stattfindet:

Induktionsbehauptung: q==364*363*...*(364–i+1)/365^i (i Faktoren im Zähler und im Nenner)

```
double Mises(int n)
{
double q=1;
for(int i=0;//IA: Für i==1,2 gilt q==364*363*...*(364-i+1)/365^i
              i<n-1; i++)
 {                    // i    q
                      // i0   364*...*(364-i0+1)/365^i0
  q = q*(364-i)/365;  //      364*...*(364-i0+1)*(364—i0)/365^(i0+1)
  // i++;              // i0+1
                      /// i=i0+1, d.h. i0==i-1
                      /// q==364*...*(364-(i-1)+1)*(364—i+1)/365^i
 }
// i==n-1: q == 364*363*...*(364-n+3)*(364-n+2)/365^(n-1) für n>=2
// oder q==1 für n<=1
return 1-q; // 0 für n<=1
}
```

Der Induktionsanfang wurde für i==1 und 2 bereits in den Ablaufprotokollen von Aufgabe 3.7.2 gezeigt.

Für n>=2 gilt nach der letzten Ausführung des Schleifenkörpers i==n-1. Setzt man diesen Wert in die Schleifeninvariante ein, erhält man den Rückgabewert (n-1 Faktoren im Zähler und Nenner)

q == 364*363*...*(364-(n-1)+2)*(364-n+2)/365^{n-1}

Für n<=1 ist der Rückgabewert 1.

c) Induktionsbehauptung: Bei der i-ten Wiederholung des Schleifenkörpers ist s die Summe der letzten i Ziffern von n0 (dem Argument für n), und n besteht aus den ersten k-i+1 Ziffern von n0, d.h.

$s=z_{i-1}+z_{i-2}+...+z_1+z_0$ und $n=z_k z_{k-1}...z_{k-(k-i)+1} z_{k-(k-i)}$

```
int Quersumme(int n)
{ // n==n0==zkzk-1...z2z1z0, n hat k+1 Ziffern
if (n<0) n=-n;
int s=0;
int i=0; // i zählt die Wiederholungen des Schleifenkörpers und
         // wurde nur eingeführt, um die Schleifeninvariante
         // einfacher zu formulieren.
while (n>0) //IA: i==0, n=zkzk-1...zi+2zi+1zi, s=zi-1+zi-2+ .. +z2+z1+z0==0
  {
  //                i      n                        s
  //                i0     zkzk-1...zi0+2zi0+1zi0   zi0-1+zi0-2+ .. +z2+z1+z0
    s = s+n%10; //                                  zi0+(zi0-1+zi0-2+ .. +z2+z1+z0)
    n = n/10;   //         zkzk-1...zi0+2zi0+1
    i++;        // i0+1
               ///  i==i0+1, i.e. i-1==i0
               ///  n=zkzk-1...zi+2zi+1zi, s= zi-1+zi-2+..+z2+z1+z0
  }
// i=k+1, n=0, s= zk+zk-1+zk-2+ ..+z2+z1+z0
return s;
}
```

Falls der Schleifenkörper also mindestens einmal ausgeführt wird, ist der Rückgabewert die Summe der Ziffern des Arguments. Falls er nie ausgeführt wird, ist der Rückgabewert 0. Da das aber nur für das Argument 0 vorkommen kann, wird auch in diesem Fall die Quersumme zurückgegeben.

d) Im Rahmen dieser Lösung wird die i-te Fibonacci-Zahl mit f(i) bezeichnet (d.h. f(0)=0, f(1)=1, f(2)=1, f(3)=2 usw.).

In Aufgabe 3.7.1, 2 d) wurde gezeigt, dass für i=2, 3 und 4 die Beziehung gilt

f(i)=f(i-1)+f(i-2), x=f(i-2), y=f(i-1)

Da diese Beziehung für i=0 und i=1 nicht gilt (siehe das Ablaufprotokoll), werden die ersten beiden Ausführungen des Schleifenkörpers in den beiden *if*-Anweisungen *if (n>=1)* und *if (n>=2)* getrennt aufgeführt.

Induktionsbehauptung: f=f(i)=f(i-1)+f(i-2), x=f(i-2) und y=f(i-1) falls kein Überlauf stattfindet, d.h. x ist die (i-2)-te Fibonaccizahl und y die (i-1)-te.

Induktionsschritt: Das nächste Ablaufprotokoll zeigt, dass die Induktionsbehauptung eine Schleifeninvariante ist:

```
int Fibonacci(int n)
{
              // i     f                      x          y
              //
   int f=0,x=0,y=1; //   0                      0          1
if (n>=1)
{             // 0
     x = y;      //                              1
     y = f;      //                                         0
  f = x + y; //       1+0==f(1)
  // i++     // 1
}
if (n>=2)
{
     x = y;      //                              0==f(0)
     y = f;      //                                         1==f(1)
  f = x + y; //       f(0)+f(1)==1==f(2)
  // i++     // 2
}
```

```
for (int i=2;// 2
           // Induktionsanfang: mit i==2 gilt f==f(i)==f(i-1)+f(i-2)
             i<n; i++)
 {           //                                f(i-2)     f(i-1)
             // i0    f(i0)                    f(i0-2)    f(i0-1)
    x = y;      //                                f(i0-1)
    y = f;      //                                            f(i0)
  f = x + y; //       f(i0-1)+f(i0)
  // i++;     // i0+1
       /// i==i0+1, d.h. i-1==i0, i-2==i0-1
       /// f==f(i0-1)+f(i0)==f(i-2)+f(i-1)==f(i), x=f(i-2), y=f(i-1)
}
return f; // f(n)
}
```

Da i nach der letzten Ausführung des Schleifenkörpers den Wert n hat, gilt nach dem Verlassen der Schleife f=f(n).

Für (n<=0) wird der Schleifenkörper nie ausgeführt und die Funktion gibt den Wert 0 zurück.

e) Mit dieser Anweisungsfolge kann der Induktionsschritt nicht bewiesen werden:

```
             // i     f                        x          y
{            // IB:   f(i)                     f(i-2)     f(i-1)
             // i0    f(i0)                    f(i0-2)    f(i0-1)
    y = f;      //                                            f(i0)
  x = y;     //                                f(i0)
    f = x + y; //       f(i0)+f(i0)
  // i++;    // i0+1
             /// i==i0+1, d.h. i-1==i0, i-2==i0-1
             /// f==f(i0)+f(i0)==2*f(i-1), x=f(i-1), y=f(i-1)
}
```

f) Mit dieser Anweisungsfolge kann der Induktionsschritt bewiesen werden, wenn man beachtet, dass x und y im Vergleich zu den vorherigen Beispielen vertauscht sind:

```
             // i     f                        x          y
{            // IB:   f(i)                     f(i-1)     f(i-2)
             // i0    f(i0)                    f(i0-1)    f(i0-2)
    int y = f; //                                            f(i0)
  f = x+f;   //       f(i0-1)+ f(i0)
    x = y;      //                                f(i0)
  // i++;    // i0+1
             /// i==i0+1, d.h. i-1==i0, i-2==i0-1
             /// f==f(i0-1)+f(i0)==f(i-2)+f(i-1), x=f(i-1), y=f(i-1)
}
```

Allerdings kann gilt der Induktionsanfang nicht: f(0)=1, f(2)=1, f(3)=2 usw., d.h. mit dieser Anweisungsfolge erhält man die um 1 versetzten Fibonacci-Zahlen.

2. a) n <= 0 und n gerade.

b) Fall I: i <= 0. Der Schleifenkörper wird nie ausgeführt, also erhält man auch keine Endlosschleife.

Fall II: i > 0. Da für i>0 stets i/2<i gilt, wird i bei jeder Ausführung des Schleifenkörpers verkleinert.

Deshalb erhält man für keinen Wert von i eine Endlosschleife. Dagegen würde die Schleifenbedingung (i >= 0) anstelle von (i > 0) zu einer Endlosschleife führen, da 0/2 = 0.

c) Fall I: i >= n: Der Schleifenkörper wird nie ausgeführt.

Fall II: i < n: Für jeden Wert von i ist i + 1 > i.

Deshalb wird i bei jeder Ausführung vergrößert und erhält schließlich den Wert n.

Auch diese Schleife wird also nie zur Endlosschleife.

d) Bei einer Prüfung von Gleitkommawerten auf Gleichheit können Ungenauigkeiten in der Zahldarstellung dazu führen, dass eine erwartete Gleichheit doch nicht eintritt. Deshalb sollten Schleifen nie durch eine Prüfung von Gleitkommawerten auf Gleichheit kontrolliert werden.

   Ersetzt man die Schleifenbedingung durch d >= 10, wäre das Risiko einer Endlosschleife vermieden. Allerdings hätte man mit dieser Bedingung auch keine Sicherheit für eine bestimmte Anzahl von Wiederholungen. Es kann vorkommen, dass sie einmal zu oft oder einmal zu wenig ausgeführt wird.

e) Wird für i > 0 immer zur Endlosschleife, da die Kontrollvariable in der Schleife nicht verändert wird.

f) siehe b).

g) In Aufgabe 3.4.6, 6. wurde darauf hingewiesen, dass bisher noch nicht allgemeingültig nachgewiesen werden konnte, ob die Schleife immer abbricht. Deswegen ist nicht sicher, ob sie tatsächlich für alle Werte von n keine Endlosschleife ist. Für alle Werte des 32-bit Datentyps *int* ist sie aber keine Endlosschleife.

3. Die Schleifeninvariante ist in dem als Kommentare angegebenen Ablaufprotokoll enthalten:

```
int ggt(int x, int y)
{
// x=x0, y=y0, ggT(x,y) == ggT(x0,y0)
if (x<0) x=-x; // Damit die Funktion auch mit negativen
if (y<0) y=-y; // Argumenten aufgerufen werden kann.
// ggT(x,y) == ggT(x0,y0)
while (y!=0)     // Ablaufprot. für den Schleifenkörper
  {//                          x      y      r
     // ggT(x,y)== ggT(x0,y0)    x0     y0     -
      int r = x%y; //                          x0%y0
      x = y;       //            y0
      y = r;       //                     x0%y0
     // ggT(x,y) == ggT(y0,x0%y0) == ggT(x0,y0)
  }
// ggT(x,y) = ggT(x0,y0) und y=0
return x; // ggT(x,y)==x;
}
```

Lösung Aufgaben 3.7.5

1. Lösung siehe *LogicU.cp*p.

2. Bei den Aufgaben

   a) Fakultaet
   b) Mises
   c) Quersumme
   d) Fibonacci

   kann die Schleifeninvariante nicht in C++ formuliert werden, da die Anzahl ihrer Teilausdrücke bei jeder Wiederholung des Schleifenkörpers anders ist.

Lösung Aufgaben 3.7.6

1. **3.4.6, 1. int Quersumme(int n)**

   a) Die Aufgabe ist für beliebige Ganzzahlwerte formuliert. Aus der Aufgabestellung geht aber nicht hervor, was die Quersumme einer negativen Zahl sein soll: 6 oder –6 für –123? Eine weitere Alternative wäre ein Parametertyp ohne Vorzeichen. Aus den in Abschnitt 3.3.3 beschriebenen Gründen wird aber davon abgesehen.

      Sobald man sich aber für eine Alternative entschieden hat, kann man für jedes Argument prüfen, ob der Funktionswert die Aufgabenstellung erfüllt.

   Die Lösung der folgenden Teilaufgaben bezieht sich auf diese Funktion:

```
int Quersumme(int n)
{
if (n<0) n=-n; // siehe g)
int s=0;
while (n>0)
  {
    s = s+n%10;
    n = n/10;
  }
return s;
}
```

b) Ihr Name ist ziemlich präzise und aussagekräftig.

c) Das wurde in Aufgabe 3.7.3, 1. c) gezeigt.

d) Diese Funktion gibt immer den Wert von s zurück, und dieser Wert ist für jedes Argument definiert.

e) Kein Argument führt zu einem Programmfehler, wie z.B. einer Endlosschleife, einer unzulässigen Operation, einem Überlauf, einem Funktionsaufruf mit nicht erfüllter Vorbedingung usw.

f) Alle Kriterien sind erfüllt. Die zyklomatische Komplexität ist 3 (ein *if* und ein *while*).

g) Falls für negative Argumente kein Ergebnis festgelegt wird, muss die Vorbedingung n>=0 erfüllt sein.

h) Man kann die Aufgabenstellung so erweitern, dass auch für negative Argumente ein Rückgabewert gefordert wird. Vorstellbar wäre z.B. eine der folgenden Erweiterungen:

- Quersumme(n) = Quersumme(-n) für n<0 (diese Variante wurde oben gewählt), oder
- Quersumme(n) = - Quersumme(-n) für n<0, oder
- Quersumme(n) = -1

Alternativ könnte man für negative Argumente einen Fehlerstatus setzen (z.B. eine globale Variable).

Mit dieser zusätzlichen Anforderung ist die Vorbedingung der Funktion immer erfüllt, d.h. *true*. Sie kann aufgerufen werden, ohne dass vor dem Aufruf eine Vorbedingung geprüft werden muss.

3.4.6, 2. int Fibonacci(int n)

a) Die Aufgabenstellung ist für n>=0 formuliert. Für jedes solche Argument kann man entscheiden, ob der Funktionswert die Aufgabenstellung erfüllt.

Aus den in Abschnitt 3.3.3 beschriebenen Gründen wird davon abgeraten, den Parametertyp zu *unsigned int* zu ändern.

Die Lösung der folgenden Teilaufgaben bezieht sich auf diese Funktion:

```
int Fibonacci(int n)
{
int f=0,x=0,y=1;
for (int i=0; i<n; i++)
  {
    x=y;
    y=f;
    f=x+y;
  }
return f;
}
```

b) Der Name ist ziemlich präzise und aussagekräftig.

c) Das wurde in Aufgabe 3.7.3, 1. d) gezeigt.

d) Diese Funktion gibt immer den Wert von f zurück, und dieser Wert ist für jedes Argument definiert.

e) Wenn der Datentyp *int* 32 Bit breit ist, führen Argumente >= 47 zu einem Überlauf. Falls diese Funktion von einem anderen Compiler (z.B. 16-bit oder 64-bit *int*) verwendet wird, ist es schwierig, allgemeine Kriterien für

diese Grenze anzugeben.

Andere Fehler, wie z.B. eine Endlosschleife, eine unzulässige Operation oder ein Funktionsaufruf mit nicht erfüllten Vorbedingungen, treten nicht auf.

f) Alle Kriterien sind erfüllt. Die zyklomatische Komplexität ist 2 (ein *for*).

g) Vorbedingung n>=0 und n<=47:

h) Man kann die Aufgabenstellung so erweitern, dass auch für Argumente < 0 und >47 ein Rückgabewert gefordert wird. Vorstellbar wäre z.B. eine der folgenden Erweiterungen:

- Fibonacci(n) = 0 für n<=0
- Fibonacci(n) = -1 für n>=47

Mit dieser zusätzlichen Anforderung ist die Vorbedingung der Funktion immer erfüllt. Sie kann aufgerufen werden, ohne dass vor dem Aufruf eine Vorbedingung geprüft werden muss.

3.4.6, 4. prim

a) Die Aufgabenstellung ist für n>0 formuliert. Für jeden solchen Wert kann man entscheiden, ob der Funktionswert die Aufgabenstellung erfüllt.

Aus den in Abschnitt 3.3.3 beschriebenen Gründen wird davon abgeraten, den Parametertyp zu *unsigned int* zu ändern.

Die Lösung der folgenden Teilaufgaben bezieht sich auf diese Funktion:

```
bool prim(int n)
{
if (n<2) return false;
else if (n==2) return true;
else
  {
  for (int i=2; i*i<=n; i++)
    if (n%i==0) return false;
  return true;
  }
}
```

b) Der Name ist präzise und aussagekräftig.

c) Die wesentlichen Überlegungen für einen solchen Nachweis wurden in Abschnitt 3.7.3 gezeigt.

d) Diese Funktion gibt in jedem Zweit einen definierten Wert zurück.

e) Kein Argument führt einem Programmfehler, wie z.B. einer Endlosschleife, einer unzulässigen Operation, einem Überlauf usw.

f) Alle Kriterien sind erfüllt. Die zyklomatische Komplexität ist 5 (drei *if* und ein *for*).

g) Vorbedingung n>0:

h) Man kann die Aufgabenstellung so erweitern, dass auch für Argumente <= 0 ein Rückgabewert gefordert wird. Vorstellbar wäre z.B. eine der folgenden Erweiterungen:

- prim(n) = false für n<=0 (diese Variante wurde oben gewählt)
- prim(n) = prim(-n) für n<=0 (dann müsste man eine Primzahl über 4 Teiler definieren: 1, -1, p und -p)

Mit dieser zusätzlichen Anforderung ist die Vorbedingung der Funktion immer erfüllt. Sie kann aufgerufen werden, ohne dass vor dem Aufruf eine Vorbedingung geprüft werden muss.

3.6.5, 2. RegentropfenPi

a) Die Aufgabenstellung ist für n>0 formuliert. Da diese Funktion Zufallswerte verwendet, kann man nicht mit einem einzelnen Aufruf entscheiden, ob der Funktionswert die Aufgabenstellung erfüllt. Bei Funktionen mit

Zufallswerten kann man nur mit vielen Aufrufen prüfen, ob die Verteilung der Werte statistische Anforderungen erfüllt. Zumindest im Prinzip kann man aber entscheiden, ob die Funktion ihre Aufgabe erfüllt oder nicht.

Die Lösung der folgenden Teilaufgaben bezieht sich auf diese Funktion:

```
double RegentropfenPi(int n)
{
int Treffer=0;
for (int i = 0; i<n; i++)
  {
    double x = (0.0+rand())/RAND_MAX;
    double y = (0.0+rand())/RAND_MAX;
    if (x*x+y*y < 1) Treffer++;
  }
return 4.0*Treffer/n;
}
```

b) Der Name ist einigermaßen aussagekräftig.

c) Siehe a). Die beiden Anweisungen haben aber die als Kommentar angegebene Nachbedingung:

```
    double x = (0.0+rand())/RAND_MAX;
    double y = (0.0+rand())/RAND_MAX;
    // 0<=x<1 and 0<=y<1;
```

Deshalb kann man x,y als Koordinaten eines Punktes im Einheitsquadrat interpretieren. *Treffer* ist dann die Anzahl der Punkte im ersten Quadranten des Einheitskreises.

Allerdings wäre der Fehler "i<=n" anstelle "i<n" nur schwer zu entdecken.

d) Diese Funktion gibt für jedes n!=0 einen definierten Wert zurück.

e) Kein Argument führt einem Programmfehler, wie z.B. einer Endlosschleife, einer unzulässigen Operation, einem Überlauf usw.

f) Alle Kriterien sind erfüllt. Die zyklomatische Komplexität ist 3 (ein *if* und ein *for*).

g) Vorbedingung n>0.

h) Man kann die Aufgabenstellung so erweitern, dass auch für Argumente <= 0 ein Rückgabewert gefordert wird. Die Lösung oben verwendet die folgende Erweiterung:

- RegentropfenPi(n) = 0 für n<=0

3.6.5, 4. HypothekenRestschuld

a) Die Aufgabenstellung enthält überhaupt keine Angaben, für welche Wertebereiche sie definiert ist. Aus dem Zusammenhang kann man eventuell ableiten, dass alle Beträge und Prozentsätze >= 0 sein sollen. Für die meisten in der Praxis vorkommenden Fälle müsste die Aufgabenstellung aber hinreichend präzise sein.

Die Lösung der folgenden Teilaufgaben bezieht sich auf diese Funktion:

```
void HypothekenRestschuld(Currency Betrag, Currency Zinssatz,
                                                    Currency Tilgungsrate)
{
Currency Restschuld = Betrag;
Currency Annuitaet = Restschuld*(Zinssatz+Tilgungsrate)/100;
int Jahre = 0;

AnsiString s="Hypothek: "+CurrToStr(Restschuld)+
                                    ": Annuitaet: "+CurrToStr(Annuitaet);
Form1->Memo1->Lines->Add(s);
while (Restschuld > 0)
  {
    Currency Zinsen = Restschuld*Zinssatz/100;
    Restschuld = Restschuld - (Annuitaet - Zinsen);
    Jahre++;
    s="Ende "+IntToStr(Jahre)+".Jahr: Restschuld="+
      CurrToStr(Restschuld)+ " Zinsen="+CurrToStr(Zinsen)+
      " Tilgung="+CurrToStr(Annuitaet-Zinsen);
    Form1->Memo1->Lines->Add(s);
  }
}
```

b) Der Name ist präzise und aussagekräftig.

c) Ein solcher Nachweis kann leicht mit symbolischen Ablaufprotokollen und vollständiger Induktion gezeigt werden, wie in Abschnitt 3.3.

d) Da der Rückgabetyp der Funktion void ist, gibt sie keinen Wert zurück, und deshalb auch keinen undefinierten Wert. Alle ausgegebenen Werte ergeben sich aus den Parametern und sind immer definiert.

e) Falls die Restschuld wächst (z.B. bei Annuität > Zinsen), erhält man eine Endlosschleife. Unzulässige Operationen sind nicht möglich. Bei sehr kleinen Tilgungsraten kann ein Überlauf bei den Jahren stattfinden. Alle diese Bedingungen sind in der Praxis sicher nicht wahrscheinlich. Es ist aber auch nicht einfach, sie präzise zu formulieren.

f) Alle Kriterien sind erfüllt. Die zyklomatische Komplexität ist 2 (ein *while*).

g) Siehe e)

h) Man kann die Aufgabenstellung so erweitern, dass für unzulässige Argumente eine Fehlermeldung ausgegeben oder eine Statusvariable gesetzt wird.

3.6.5, 5.TrapezSumme

a) Die Aufgabenstellung enthält überhaupt keine Angaben, für welche Werte sie definiert ist. Aus dem Zusammenhang kann man aber ableiten, dass –1 <= a <= b<= 1 und n>0 gelten muss. Für diese Werte ist sie präzise formuliert. Falls die Fläche unter einer anderen Funktion berechnet werden soll, würden sich andere Anforderungen für a und b ergeben.

Die Lösung der folgenden Teilaufgaben bezieht sich auf diese Funktion:

```
double TrapezSumme(double a, double b, int n)
{
double s = (sqrt(1-a*a) + sqrt(1-b*b))/2;
double h = (b-a)/n;
for (int i=1; i<n; i++)
  s = s + sqrt(1-(a+i*h)*(a+i*h));
return s*h;
}
```

b) Der Name ist einigermaßen präzise und aussagekräftig.

c) Ein solcher Nachweis kann leicht mit symbolischen Ablaufprotokollen und vollständiger Induktion gezeigt werden, wie in Abschnitt 3.3.

d) Diese Funktion gibt für alle Argumente, die nicht zu einem Programmabbruch führen (siehe e), einen definierten Wert zurück.

e) Alle Werte, aus denen eine Wurzel gezogen wird, müssen >=0 sein.

```
double s = (sqrt(1-a*a) + sqrt(1-b*b))/2;
double h = (b-a)/n;
for (int i=1; i<n; i++)
  s = s + sqrt(1-(a+i*h)*(a+i*h));
```

Mit i==n, a==0 und b==1 wird a+i*h = 0+n*(1-0)/n == 1. Wenn dieser Wert in die Wurzel eingesetzt wird, erhält man bei einer exakten Rechnung das Ergebnis 0. Bei einer Rechnung mit Gleitkommawerten kann aber durch Rundungsfehler ein kleiner negativer Wert entstehen. Deshalb darf die Schleife nur bis n-1 laufen.

Mit n==0 findet eine Division durch 0 statt.

Mit n<0 wird der Schleifenkörper der *for*-Schleife nicht ausgeführt und s*h zurückgegeben.

Kein Argument führt zu einer Endlosschleife.

f) Alle Kriterien sind erfüllt. Die zyklomatische Komplexität ist 2 (ein *for*).

g) Vorbedingung: –1 <= a <= b<= 1 und n>0.

h) Man kann die Aufgabenstellung so erweitern, dass für Argumente, die die Vorbedingung nicht erfüllen, ein Fehler signalisiert wird (z.B. durch eine globale Variable, einen speziellen Rückgabewert oder eine Exception).

3.6.5, 7. RoundToInt

a) Die Aufgabenstellung ist für alle *double*-Werte präzise formuliert.

Die Lösung der folgenden Teilaufgaben bezieht sich auf diese Funktion:

```
int RoundToInt(double x)
{
if (x >= 0)
  return x + 0.5;
else
  return x - 0.5;
}
```

b) Der Name ist präzise und aussagekräftig.

c) Alle *double*-Werte, die außerhalb des Wertebereichs von *int* liegen, führen zu einem Überlauf. Für Argumente im Wertebereich von *int* wird der gerundete Wert zurückgegeben.

d) Diese Funktion gibt für alle Argumente einen definierten Wert zurück.

e) Alle Argumente, die außerhalb des Wertebereichs von *int* liegen, führen zu einem Überlauf. Andere Programmfehler wie z.B. eine Endlosschleife, eine unzulässigen Operation usw. sind nicht möglich.

f) Alle Kriterien sind erfüllt. Die zyklomatische Komplexität ist 2 (ein *if*).

g) Vorbedingung: Die Argumente müssen im Wertebereich von *int* liegen.

h) Man kann die Aufgabenstellung so erweitern, dass für Argumente, die nicht im Wertebereich von *int* liegen, ein Fehler signalisiert wird (z.B. eine Statusvariable gesetzt oder eine Exception ausgelöst wird).

2. Die Funktion *doit_babe* kombiniert nahezu alle Entwurfsfehler, die man bei einer Funktion begehen kann.

   1. Der Name der Funktion ist hochgradig nichtssagend.
   2. Sie hat keine in sich geschlossene Aufgabenstellung.
   3. Sie kann mit vielen Argumenten aufgerufen werden, für die kein definierter Funktionswert zurückgegeben wird.

3. Die Funktionen

```
int Plus(int a, int b)     {return a-b; } // Plus(a,b)==a-b
int Minus(int a, int b)    {return a*b; } // Minus(a,b)==a*b
int Product(int a, int b) {return a+b; } // Product(a,b)==a+b
```

haben die jeweils als Kommentar angegebene Nachbedingung. Deshalb erhält man das Ergebnis des Ausdrucks

```
int x=Plus(1,Product(Minus(3,4),Plus(5,6)));
```

indem man für die Parameter die Argumente einsetzt:

```
Plus(1,Product(Minus(3,4),Plus(5,6)))
==  Plus(1,Product(3*4,5-6)))
==  Plus(1,12 + -1)))
==  1-11 == -10
```

Lösung Aufgaben 3.7.7

1. Aus Platzgründen steht „t“ für „true“ und „f“ für „false“.

a)

p	q	r	q or r	**p and (q or r)**	p and q	p and r	**(p and q) or (p and r)**
t	t	t	t	**t**	t	t	**t**
t	t	f	t	**t**	t	f	**t**
t	f	t	t	**t**	f	t	**t**
t	f	f	f	**f**	f	f	**f**
f	t	t	t	**f**	f	f	**f**
f	t	f	t	**f**	f	f	**f**
f	f	t	t	**f**	f	f	**f**
f	f	f	f	**f**	f	f	**f**

b)

p	q	r	q and r	**p or (q and r)**	p or q	p or r	**(p or q) and (p or r)**
t	t	t	t	**t**	t	t	**t**
t	t	f	f	**t**	t	t	**t**
t	f	t	f	**t**	t	t	**t**
t	f	f	f	**t**	t	t	**t**
f	t	t	t	**t**	t	t	**t**
f	t	f	f	**f**	t	f	**f**
f	f	t	f	**f**	f	t	**f**
f	f	f	f	**f**	f	f	**f**

c)

p	q	r	p and q	**(p and q) or r**	q or r	**p and (q or r)**
t	t	t	t	**t**	t	**t**
t	t	f	t	**t**	t	**t**
t	f	t	f	**t**	t	**t**
t	f	f	f	**f**	f	**f**
f	t	t	f	**t**	t	**f** ←
f	t	f	f	**f**	t	**f**
f	f	t	f	**t**	t	**f** ←
f	f	f	f	**f**	f	**f**

`"(p and q) or r" und "p and (q or r)" sind also nicht gleichwertig.`

2. Lösung siehe *LogicU.cp*p.

Lösung Aufgaben 3.7.8

1. a) `(x >= 0) and (x <= 100)`

b) `(x==1) or (c!='j')`

c) `(i<1) and (i>10)`. Da diese Bedingung für kein i erfüllt sein kann, wird S2 nie ausgeführt.

d) Entweder

```
if (!((x > 0) and (x < 10))) S;
```

oder mit den Regeln von de Morgan:

```
if ((x <= 0) or (x >= 10)) S;
```

2. S1: (1<= x) and (x<=10)

S2: (10<x) and (x<=15)

S3: (15>x)

S4: (x<1)

3. Im letzten *else*-Zweig von ***max*** gilt die Negation der beiden vorangehenden Bedingungen:

```
not((x >= y) and (x >= z)) and not ((y >= x) and (y >= z))
```

Diese sind wegen de Morgan gleichwertig mit:

```
((x < y) or (x < z)) and ((y < x) or (y < z))
```

Zweimal "ausmultiplizieren" mit den Distributivgesetzen ergibt:

```
[(x<y) and (y<x)] or [(x<y) and (y<z)] or [(x<z) and (y<x)] or [(x<z) and
(y<z)]
```

Hier wurden die 4 mit *or* verknüpften Ausdrücke extra mit eckigen Klammern zusammengefasst. Der erste dieser eckig geklammerten Ausdrücke hat immer den Wert *false*, während alle 3 weiteren jeder für sich gerade bedeuten, dass z das Maximum der drei Werte x, y und z ist.

```
int min3(int x, int y, int z)
{
int m;
if (x<y)
 {
  if (x<z) m=x; // (x<y) and (x<z) and (m==x):   m==min(x,y,z)
  else m=z;     // (x<y) and (x>=z) and (m==z):  m==min(x,y,z)
 }
else // x>=y
 {
  if (y<z) m=y; // (x>=y) and (y<z) and (m==y):  m==min(x,y,z)
  else m=z;     // (x>=y) and (y>=z) and (m==z): m==min(x,y,z)
 }
return m;
} // m==min(x,y,z)
```

Da jeder Zweig dieselbe Nachbedingung m=min(x,y,z) hat, ist das auch die Nachbedingung der gesamten *if*-Anweisung und der gesamten Funktion.

Bei dieser Lösung wird der Ansatz von *min2a* (Abschnitt 3.7.7) auf die Bestimmung des Minimums von drei Ausdrücken x, y und z übertragen: Falls x<y gilt, bleiben als „Kandidaten“ für das Minimum x und z. Falls dagegen x<y nicht gilt, also x>=y, wird das Minimum unter y und z gesucht:

Die Übertragung auf die Bestimmung des Minimums von 4 oder noch mehr Variablen dürfte allerdings etwas mühsam

werden.

4. Die Kommentare zeigen, dass die Funktion *median* immer den mittleren der drei als Parameter übergebenen Werte zurückgibt:

```
int median (int a, int b, int c)
{
if (a < b)
  if (b < c)
    return b; // a<b and b<c
  else if (a < c)
    return c; // a<b and c<=b and a<c
  else return a; // a<b and c<=b and c<=a
else if (a < c)
  return a; // b<=a and a<c
else if (b < c)
  return c; // b<=a and c <= a and b<c
else return b; // b<=a and c <= a and b>=c
}
```

5. In den Aufgaben 3.7.3, 1 a) und b) wurden bereits die folgenden beiden Invarianten nachgewiesen:

n gerade:
```
// p*x^n = u^v
n = n/2;
x = x*x;
// p*x^n = u^v
```

n ungerade:
```
// p*x^n = u^v
p = p*x;
n = n/2;
x = x*x;
// p*x^n = u^v
```

Das sind aber gerade die Anweisungsfolgen, die durch

```
if (n&1)  p = p*x;
n = n/2;
x = x*x;
```

ausgeführt werden, wenn n gerade oder ungerade ist. Damit gilt die gefragte Invarianz

```
// p*x^n = u^v
if (n&1)  p = p*x;
n = n/2;
x = x*x;
// p*x^n = u^v
```

6. Die Nachbedingung ist als Kommentar angegeben:

```
// n==n0
 if (n%2==1) n=3*n+1;
 else n=n/2;
// (odd(n0) and n==3n0+1) or ((even(n0) and n==n0/2)
```

Sie ist gerade die etwas formale Formulierung der 3n+1-Problems (Aufgabe 3.4.5, 6.).

11.3.8 Aufgaben 3.10.2

```
// File: C:\Loesungen_CB2006\Kap_3\3.10\10_2.cpp
```

```
// --------------- Aufgaben 3.10.2 -------------------------
// --------------- Aufgabe 1. ------------------------------
void ArrayTest1()
{
int a[10];
for (int i=1; i<=10;i++) a[i] = 0;   // a)

int b[2], c[2], d[3];
b[0]=0; b[1]=1;
//c=b;                               // b)

int x=b[b[b[0]]];                    // c)
c[0]=0; c[1]=1;
d[0]=0; d[1]=1; d[2]=2;
if (b==c) x++;                       // d)
if (c==d) x++;                       // e)

int s1=sizeof(a);                    // f)
int s2=sizeof(a)/sizeof(a[0]);       // g)
// a) Fehler: Zugriff auf a[10]
// b) Zuweisung von Arrays ist nicht möglich
// c) x=0
// d) b==c vergleicht die Adressen und nicht die Arrays
// e) wie b); da die Arrays verschieden groß sind,
//    kann das kein Vergleich der Arrayelemente sein
// f) s1=2*sizeof(int)=8
// g) s2=8/4=2, Anzahl der Arrayelemente
}


// --------------- Aufgabe 2 -------------------------------

void SiebEratosthenes()
{
const int n=1000;
bool prim[n+1];
// p[i]==true, falls i eine Primzahl ist, sonst false
for (int i=0; i<=n; i++)
  prim[i] = true;
for (int i=2; i<=n; i++)
  if (prim[i])
    for (int j=i;  j<=n/i; j++)
      prim[j*i] = false;
for (int i=2; i<=n; i++)
  if (prim[i])
    Form1->Memo1->Lines->Add(IntToStr(i));
};


// --------------- Aufgabe 3 -------------------------------

double MisesSimulation(int AnzahlPersonen)
{
int g[365];
int AnzahlVersuche=1000,
    AnzahlTreffer=0;
for (int i=1; i<=AnzahlVersuche; i++)
  {
    for (int j=0; j<365; j++) g[j] = 0;
    for (int j=1; j<=AnzahlPersonen; j++)
       g[rand()%365+1]++;
    bool gefunden = false;
    for (int j=0; j<365; j++)
      if (g[j] > 1) gefunden = true;
    if (gefunden) AnzahlTreffer++;
  };
Form1->Memo1->Lines->Add("n="+IntToStr(AnzahlPersonen)+" w="
+FloatToStr(1.0*AnzahlTreffer/AnzahlVersuche));
return 1.0*AnzahlTreffer/AnzahlVersuche;
```

```
}

void TestMisesSimulation()
{
MisesSimulation(23);
MisesSimulation(23);
MisesSimulation(23);
MisesSimulation(23);
MisesSimulation(23);

MisesSimulation(10);
MisesSimulation(20);
MisesSimulation(30);

// Die Simulationsergebnisse mit 1000 Versuchen stimmen recht
// gut mit den exakten Werten (in Klammern) überein:
/*
  n=23 w=0,499  (p=0,5073)
  n=23 w=0,508
  n=23 w=0,5
  n=23 w=0,535
  n=23 w=0,512

  n=10 w=0,116   (p=0,1169)
  n=20 w=0,414   (p=0,4114)
  n=30 w=0,69    (p=0,7063)
*/
}

// --------------- Aufgabe 4 -------------------------------

double Horner(double x)
{ // p: Array mit den Koeffizienten des Polynoms
const int n=2; // Der Grad des Polynoms
double p[n+1]={17,0,3}; // Koeffizienten von 17+3*x^2
double s = 0;
for (int i=0; i<n+1; i++)
    s = s*x + p[n-i];
return s;
}

/*
a)

Induktionsanfang: Nach der ersten Zuweisung s=s*x+...
  in der Schleife gilt (i==1) und (s==p[n]).

Induktionsvoraussetzung:
  Nach der k-ten Zuweisung s=s*x+...
  in der Schleife (vor i++) gilt (i==k) und

    s==p[n]*x^k + p[n-1]*x^(k-1) +...+ p[n-k+1]*x+p[n-k]

  Beim nächsten Schleifendurchlauf erhält i den Wert (i==k+1)
  und s den Wert:

    s==p[n]*x^(k+1) + p[n-1]*x^k +...+ p[n-k+1]*x^2+p[n-k]*x+p[n-(k+1)]

  Das ist gerade die Induktionsbehauptung für k+1.

  Beim letzten Durchlauf hat i den Wert n:

    s==p[n]*x^n + p[n-1]*x^(n-1) +...+ p[n-n+1]*x+p[n-n]
*/

// b)
double Horner_sin(double x)
{ // p: Array mit den Koeffizienten des Polynoms
double k3=3*2*1;
```

```
double k5=5*4*k3;
double k7=7*6*k5;
const int n=7; // Der Grad des Polynoms
double p[n+1]={0, 1, 0, -1/k3, 0, +1/k5, 0, -1/k7};
double s = 0;
for (int i=0; i<n+1; i++)
    s = s*x + p[n-i];
return s;
}

#include <cmath>
void test_HornerSinus()
{
for (int i=1 ;i<=31 ;i++ )
  {
  double x=i/10.0;
  double s1=std::sin(x);
  double s2=Horner_sin(x);
  Form1->Memo1->Lines->Add("x="+FloatToStr(x)+" sin="+s1+
         " Horn-sin="+FloatToStr(s2)+" diff="+std::abs(s1-s2));
  // Offensichtlich liefert die Taylor-Reihe für kleine Werte
  // eine gute Näherung
  }
}
```

11.3.9 Aufgaben 3.10.3

```
// File: C:\Loesungen_CB2006\Kap_3\3.10\10_3.cpp

// --------------- Aufgaben 3.10.3 -------------------------
// --------------- Aufgabe 1. ------------------------------
#include <string>;
using namespace std;

int ai0[3];

void InitArrays()
{
int ai1[3];         // undefined values
int ai2[3]={1,2,3}; // {1,2,3}
int ai3[3]={1,2};   // {1,2,0}
}


// --------------- Aufgabe 2. ------------------------------

int FibonacciArray(int n)
{
const int max=47;
int f[max]={0,1,1,2,3,5,8,13,21,34}; // 10 Werte
if ((0<=n) && (n<10)) return f[n];
else if (n<47)
  {
    for (int i=10;i<=n ;i++ )
      f[i]=f[i-1]+f[i-2];
    return f[n];
  }
else return -1;
}

void testFibonacci()
{
const int max=47;
```

```
int f[max]={0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,
            987,1597,2584,4181,6765,10946,17711}; // 23 Werte
for (int i=0;i<23 ; i++)
  if  (FibonacciArray(i)!=f[i])  Form1->Memo1->Lines->Add("Error:  "+IntToStr(i)+"
"+IntToStr(FibonacciArray(i)));
}
```

11.3.10 Aufgaben 3.10.4

```
// File: C:\Loesungen_CB2006\Kap_3\3.10\10_4.cpp

// --------------- Aufgaben 3.10.4 -------------------------
// --------------- Aufgabe 1. ------------------------------
const int maxElements=5; // Mit einem kleinen Wert von maxElements
                 // kann man auch die Prüfungen auf die Obergrenze
                 // einfach testen.
int nElements=0; // Anzahl der Arrayelemente
                 // belegt im Array: A[0] .. A[nElements-1]
typedef AnsiString T;
T A[maxElements];

// a)
bool pushBack(T data)
{
bool success=true;
if (nElements<maxElements)
  {
    A[nElements]=data;
    nElements++;
  }
else success=false;
return success;
// Nachbedingung: Falls beim Aufruf nElements<maxElements gilt:
//                  nElements=nElements0+1
//                  a[nElements-1]==data
//                  Rückgabewert true
//                und andernfalls Rückgabewert false.
}

void __fastcall TForm1::EinfuegenClick(TObject *Sender)
{
if (!pushBack(Edit1->Text))
  ShowMessage("Alle Plätze belegt");
}

// b)
void showArray()
{
for (int i=0; i<nElements; i++)
  Form1->Memo1->Lines->Add(A[i]);
}

void __fastcall TForm1::AnzeigenClick(TObject *Sender)
{
showArray();
}

// c)

/*
                 nElements  A[0]  A[1]  A[2]    pushBack maxElements
nElements=0          0                                        3
```

```
pushBack("10")       1     "10"                    true
pushBack("11")       2           "11"              true
pushBack("12")       3                 "12"        true
pushBack("13")       3                             false

Mit n<maxElements Aufrufen von pushBack werden die Positionen
a[0], …, a[n-1] des Array mit den Argumenten von pushBack gefüllt.
*/

// d)

int findLinear(T x, int Start)
{
int result=nElements;
bool found=false;
for (int i=Start; i<nElements && !found; i++)
  if (x==A[i])
    {
      result=i;
      found=true;
    }
return result;
}

void __fastcall TForm1::LinSuchenClick(TObject *Sender)
{
int i=0;
while (i<nElements)
  {
    i=findLinear(Edit1->Text,i);
    if (i<nElements)
      {
        Memo1->Lines->Add(A[i]);
        i++;
      }
  }
}

// e)

bool eraseElement(int pos)
{
bool result=true;
if ((0<=pos) && (pos<nElements))
  {
    A[pos]=A[nElements-1];
    nElements--;
  }
else result=false;
return result;
}

void __fastcall TForm1::LoeschenClick(TObject *Sender)
{
int p=findLinear(Edit1->Text,0);
if (p!=nElements)
  {
    if (eraseElement(p))
      ShowMessage(Edit1->Text+" gelöscht an Pos."+IntToStr(p));
    else
     ShowMessage(Edit1->Text+" nicht gelöscht");
  }
else
  ShowMessage(Edit1->Text+" existiert nicht");
}

// e)

void Auswahlsort()
```

```
{
for (int i=0; i<nElements-1; i++)
  {
    int min = i;
    for (int j=i+1; j<nElements; j++)
      if (A[j]<A[min]) min = j;
    T x = A[i];
    A[i] = A[min];
    A[min] = x;
  }
}

void __fastcall TForm1::SortierenClick(TObject *Sender)
{
Auswahlsort();
}

// g)
int findBinary(T s)
{
int M;
int L = 0;
int R = nElements-1;
bool found = false;
while ((L<=R) && (!found))
  {
     // A sortiert ==> A[L] <= A[R]
     M = (L + R)/2;
     // A[L] <= A[M] <= A[R]
     if (A[M]<s) L = M+1;
     else if (s<A[M]) R = M-1;
     else found = true;
  }
if (found) return M;
else return nElements;
}

void __fastcall TForm1::BinSuchenClick(TObject *Sender)
{
int pos=findBinary(Edit1->Text);
if (pos<nElements)
  Memo1->Lines->Add(A[pos]+" existiert");
else
  Memo1->Lines->Add(A[pos]+" nicht gefunden");
}

// h)
bool insertSorted(T x)
{
bool result=false;
if (nElements<maxElements)
  {
    // Die Position p für das neue Element bestimmen
    int p=0;
    while ((p<nElements) && (A[p]<x)) // Reihenfolge der Bedingungen ist relevant
(Short-Circuit-Evaluation)
      p++;  //  Alle  Elemente  ab  Position  p  um  eine  Position  nach  hinten
verschieben
    for (int i=nElements-1; i>=p; i--)
      A[i+1]=A[i];
    A[p]=x;       // Das neue Element in Position p eintragen
    nElements++;  // Anzahl der Elemente aktualisieren
    result=true;
  }
return result;
}

void __fastcall TForm1::SortEinfClick(TObject *Sender)
{
```

```
if (!insertSorted(Edit1->Text))
  ShowMessage("Alle Plätze belegt");
}

// i)
// Falls das gesuchte Element mehrfach im Array enthalten ist,
// wird mit einer binären Suche nicht unbedingt das letzte gefunden.

// j)
bool eraseSorted(const T& x)
{
bool result=false;
int p=findLinear(x, 0);
if ((0<=p) && (p<nElements))
  { // Alle Elemente ab Position p um eine Position nach vorne
    for (int i=p; i<nElements-1; i++)
      A[i]=A[i+1];
    nElements--; // Anzahl der Elemente aktualisieren
    result=true;
  }
return result;
}

void __fastcall TForm1::SortLoeschenClick(TObject *Sender)
{
if (!eraseSorted(Edit1->Text))
  ShowMessage(Edit1->Text+" existiert nicht");
}

// k)

#include "..\..\CppUtils\testutils.h"

bool test_1()
{
// teste pushBack, findLinear und eraseElement
bool result = true;
nElements=0;
if (!assertEqual(findLinear("1",0),0,"findLinear(\"1\",0) - a")) result=false;
pushBack("1");
if (!assertEqual(findLinear("1",0),0,"findLinear(\"1\",0) - b")) result=false;
pushBack("2");
if (!assertEqual(findLinear("2",0),1,"findLinear(\"2\",0)")) result=false;
pushBack("3");
if (!assertEqual(findLinear("3",0),2,"findLinear(\"3\",0) - a")) result=false;
// Ist das gefundene Element auch wirklich das erste?
pushBack("3");
if (!assertEqual(findLinear("3",0),2,"findLinear(\"3\",0) - b")) result=false;
// teste eraseElement
eraseElement(0);
if    (!assertEqual(findLinear("0",0),nElements,"nach    eraseElement(\"2\",0)"))
result=false;

return result;
}

void testSetAndSort2Elements(T e1, T e2)
{
nElements=0;
pushBack(e1);
pushBack(e2);
Auswahlsort();
}

void testSetAndSort3Elements(T e1, T e2, T e3)
{
nElements=0;
pushBack(e1);
pushBack(e2);
```

```
pushBack(e3);
Auswahlsort();
}

bool test_Auswahlsort_0123()
{ // teste Auswahlsort mit 0, 1, 2 und 3 Elementen
bool result = true;
nElements=0;
Auswahlsort(); // Auswahlsort sollte bei einem leeren Array nicht abstürzen
pushBack("1");
Auswahlsort(); // Array mit einem Element
if (!assertEqual(findLinear("1",0),0,"S(\"1\",0)")) result=false;

testSetAndSort2Elements("1","2"); // bereits sortiert
if (!assertEqual(findLinear("1",0),0,"sort 2, 1a")) result=false;
if (!assertEqual(findLinear("2",0),1,"sort 2, 1b")) result=false;
testSetAndSort2Elements("2","1"); // umgekehrte Reihenfolge
if (!assertEqual(findLinear("1",0),0,"sort 2, 2a")) result=false;
if (!assertEqual(findLinear("2",0),1,"sort 2, 2b")) result=false;

testSetAndSort3Elements("1","2","3"); // testen alle Permutationen
if (!assertEqual(findLinear("1",0),0,"sort 3, 1a")) result=false;
if (!assertEqual(findLinear("2",0),1,"sort 3, 1b")) result=false;
if (!assertEqual(findLinear("3",0),2,"sort 3, 1c")) result=false;
testSetAndSort3Elements("1","3","2");
if (!assertEqual(findLinear("1",0),0,"sort 3, 1a")) result=false;
if (!assertEqual(findLinear("2",0),1,"sort 3, 1b")) result=false;
if (!assertEqual(findLinear("3",0),2,"sort 3, 1c")) result=false;
testSetAndSort3Elements("2","1","3");
if (!assertEqual(findLinear("1",0),0,"sort 3, 1a")) result=false;
if (!assertEqual(findLinear("2",0),1,"sort 3, 1b")) result=false;
if (!assertEqual(findLinear("3",0),2,"sort 3, 1c")) result=false;
testSetAndSort3Elements("2","3","1");
if (!assertEqual(findLinear("1",0),0,"sort 3, 1a")) result=false;
if (!assertEqual(findLinear("2",0),1,"sort 3, 1b")) result=false;
if (!assertEqual(findLinear("3",0),2,"sort 3, 1c")) result=false;
testSetAndSort3Elements("3","1","2");
if (!assertEqual(findLinear("1",0),0,"sort 3, 1a")) result=false;
if (!assertEqual(findLinear("2",0),1,"sort 3, 1b")) result=false;
if (!assertEqual(findLinear("3",0),2,"sort 3, 1c")) result=false;
testSetAndSort3Elements("3","2","1");
if (!assertEqual(findLinear("1",0),0,"sort 3, 1a")) result=false;
if (!assertEqual(findLinear("2",0),1,"sort 3, 1b")) result=false;
if (!assertEqual(findLinear("3",0),2,"sort 3, 1c")) result=false;
return result;
}

bool isSorted()
{
bool result=true;
for (int i=0;i<nElements-1 ;i++ )
    if (A[i] > A[i+1]) result=false;
return result;
}

#include <cstdlib>
bool test_Auswahlsort_n(int n)
{ // teste Auswahlsort mit n Zufallszahlen
bool result = true;
nElements=0;
for (int i=0; i<n; i++)
  pushBack(IntToStr(std::rand()));
Auswahlsort();
if (!assertEqual(isSorted(),true,"sort n")) result=false;
return result;
}

bool test_2()
{// teste findBinary, insertSorted und eraseSorted
```

```
bool result = true;
nElements=0;
if (!assertEqual(findBinary("1"),0,"findBinary(\"1\",0)")) result=false;
pushBack("1");
if (!assertEqual(findLinear("1",0),0,"findLinear(\"1\",0)")) result=false;
pushBack("2");
if (!assertEqual(findLinear("2",0),1,"findLinear(\"2\",0)")) result=false;
pushBack("3");
if (!assertEqual(findLinear("3",0),2,"findLinear(\"2\",0)")) result=false;
// teste eraseElement
eraseElement(0);
if    (!assertEqual(findLinear("0",0),nElements,"nach    eraseElement(\"2\",0)"))
result=false;

return result;
}

bool test_All_3_4()
{
bool result = true;
if (!test_1()) result=false;
if (!test_Auswahlsort_0123()) result=false;
for (int i=0;i<100 ; i++)
  test_Auswahlsort_n(i);
if (!test_2()) result=false;
return result;
}

void __fastcall TForm1::testAllClick(TObject *Sender)
{
if (MessageDlg ("Die Daten im Array werden gelöscht - weiter?",
    mtConfirmation, TMsgDlgButtons()<<mbYes<<mbNo, 0)==mrYes)
  {
    initTestUtils();
    testUtilsSummary(Memo1, test_All_3_4());
  }
}


// --------------- Aufgabe 2. ------------------------------

typedef AnsiString StackElementType;

const int MaxS=5;
StackElementType Stack[MaxS];
int StackPtr=-1;

void enableButtons()
{
Form1->StackPop->Enabled=StackPtr!=-1;
Form1->StackTop->Enabled=StackPtr>-1;
Form1->StackPush->Enabled=StackPtr<MaxS-1;
}

void initStack()
{
StackPtr=-1;
enableButtons();
}

void pushStack(StackElementType e)
{
if (StackPtr<MaxS)
  {
    StackPtr++;
    Stack[StackPtr]=e;
  }
}
```

```
StackElementType topStack()
{
if ((0<=StackPtr) && (StackPtr<MaxS))
  return Stack[StackPtr];
else
  return "no data on stack ";
}

void popStack()
{
if (StackPtr>=0)
  {
    StackPtr--;
  }
}


//----- Aufgabe 3 ------------------------------------------------

AnsiString TroepfelPi()
{
const int Stellen = 1000,
      Max = 10*(Stellen/3);

int Q[Max+1],U[Max+1],A[Max+1];
AnsiString pistr = "pi=";

for (int i=0; i<=Max; i++) A[i] = 2;

int S;
int qi=0;
for (int n=1; n<=Stellen; n++)
  {
    U[Max] = 0;
    for (int i=Max; i>=1; i--)
      {
        S = U[i]*i+10*A[i];
        A[i]   = S % (2*i-1);
        U[i-1] = S / (2*i-1);
      }
    S = U[0];
    A[1] = S % 10;
    S    = S / 10;
    if (S== 9)
      {
        Q[qi] = S;
        qi++;
      }
    else if (S==10)
      {
        for (int j=0; j<=qi-1;j++)
        if (Q[j] < 9) Q[j]++;
          else Q[j] = 0;
        for (int j=0;j<=qi-1;j++)
          pistr = pistr+IntToStr(Q[j]);
        qi = 1;
        Q[0] = 0;
      }
    else
      {
        for (int j=0;j<=qi-1;j++)
          pistr = pistr+IntToStr(Q[j]);
        qi = 1;
        Q[0] = S;
      }
  }
return pistr;
}
```

```
void testPi()
{
const
AnsiString pi1000 = "pi=3"  // die ersten 1000 Stellen von pi
       "14159265358979323846264338327950288419716939937510582097494 4"
       "59230781640628620899862803482534211706798214808651328230664 7"
       "09384460955058223172535940812848111745028410270193852110555 9"
       "64462294895493038196442881097566593344612847564823378678316 5"
       "27120190914564856692346034861045432664821339360726024914127 3"
       "72458700660631558817488152092096282925409171536436789259036 0"
       "01133053054882046652138414695194151160943305727036575959195 3"
       "09218611738193261179310511854807446237996274956735188575272 4"
       "89122793818301194912983367336244065664308602139494639522473 7"
       "19070217986094370277053921717629317675238467481846766940513 2"
       "00056812714526356082778577134275778960917363717872146844090 1"
       "22495343014654958537105079227968925892354201995611212902196 0"
       "86403441815981362977477130996051870721134999999837297804995 1"
       "05973173281609631859502445945534690830264252230825334468503 5"
       "26193118817101000313783875288658753320838142061717766914730 3"
       "59825349042875546873115956286388235378759375195778185778053 2"
       "17122680661300192787661119590921642019 89";

AnsiString piStr=TroepfelPi();
int Stellen=0; // Vergleiche die beiden Strings zum Test

int j=1;
while ((j < piStr.Length()) &&  (j < pi1000.Length()))
  {
    if(piStr[j]==pi1000[j]) Stellen++;
    j++;
  }

Form1->Memo1->Lines->Add(piStr);
Form1->Memo1->Lines->Add("n="+IntToStr(Stellen)+" gleich ");
}
```

11.3.11 Aufgaben 3.10.5

```
// File: C:\Loesungen_CB2006\Kap_3\3.10\10_5.cpp

// --------------- Aufgaben 3.10.4 -------------------------
// --------------- Aufgabe 1. ------------------------------
void FindMinMaxInMultidimArray()
{
// a)
const int m=10, n=20;
int a2[m][n];
int min_i=0, min_j=0;

// Testwerte:
for (int i=0; i<m; i++)
  for (int j=0; j<n; j++)
    a2[i][j]=100-(i+j);

// Bestimme die gesuchten Positionen
for (int i=0; i<m; i++)
  for (int j=0; j<n; j++)
    if (a2[i][j]<a2[min_i][min_j])
      {
        min_i=i;
        min_j=j;
      }
Form1->Memo1->Lines->Add(IntToStr(min_i)+":"+IntToStr(min_j));
```

```
// b)
const int p=10;
int a3[m][n][p];
int min_k=0;
min_i=0, min_j=0;

// Testwerte:
for (int i=0; i<m; i++)
  for (int j=0; j<n; j++)
    for (int k=0; k<p; k++)
      a3[i][j][k]=100-(i+j+k);

// Bestimme die gesuchten Positionen
for (int i=0; i<m; i++)
  for (int j=0; j<n; j++)
    for (int k=0; k<p; k++)
      if (a3[i][j][k]<a3[min_i][min_j][min_k])
        {
          min_i=i;
          min_j=j;
          min_k=k;
        }
Form1->Memo1->Lines-
>Add(IntToStr(min_i)+":"+IntToStr(min_j)+":"+IntToStr(min_k));
}


// --------------- Aufgabe 2. ------------------------------

void EntfTabelle()
{
// a)
int d[][3]={{ 0, 10, 20},
            {10,  0, 15},
            {20, 15,  0}};

int r[] = {4, 2, 1, 2, 0};
int L=0;
for (int i=1; i<r[0]; i++)
  L=L+d[r[i]][r[i+1]];
Form1->Memo1->Lines->Add("L="+IntToStr(L));

int s[][10]={{3,1,2,0},    // Route 1
             {2,1,2},      // Route 2
             {5,0,1,2,1,0} // Route 3
            };

// b)
int MinL=0, MinInd=0;
for (int k=0; k<3; k++)
  {
    int L=0;
    for (int i=1; i<s[k][0]; i++)
      L=L+d[s[k][i]][s[k][i+1]];

    if (k==0)
      {
        MinL=L;
        MinInd=0;
      }
    else if (L<MinL)
      {
        L=MinL;
        MinInd=k;
      }

    Form1->Memo1->Lines->Add("L="+IntToStr(L));
  }
}
```

```
// --------------- Aufgabe 3. ------------------------------

const int n=3;

int Zeile_mit_groesstem_Spaltenwert(int i, double a[][n])
{
int Max=i;
for (int j=i+1; j<n; j++)
  if (labs(a[j][i])>labs(a[Max][i])) Max =j;
return Max;
}

void vertausche_Zeilen(int z1, int z2, double a[][n], double b[])
{
double h;
for (int i=z1; i<n; i++)
  {
    h=a[z1][i];
    a[z1][i]=a[z2][i];
    a[z2][i]=h;
  }
h=b[z1];
b[z1]=b[z2];
b[z2]=h;
}

void GaussElimination(double a[][n], double b[], double x[] ,double p[], int n)
{// Koeffizienten: a, rechte Seite: b, Lösung: x, Probe: p
for (int i=0; i<n; i++)
  for (int j=i+1; j<n; j++)
    {
      int z1=Zeile_mit_groesstem_Spaltenwert(i,a);
      if (i!=z1) vertausche_Zeilen(i, z1, a, b);
      if (a[i][i]==0)
        { // Abbruch
          ShowMessage("Nicht lösbar");
          return;
        }
      double f=-a[j][i]/a[i][i];
      a[j][i]=0;
      for (int k=i+1;k<n; k++)
        a[j][k]=a[j][k]+f*a[i][k];
      b[j]=b[j]+f*b[i];
    }

// rechte Seite "rückwärts" auflösen:
for (int i=n-1; i>=0; i--)
 {
   x[i]=b[i];
   for (int j=n-1; j>i; j--)
     x[i]=x[i]-a[i][j]*x[j];
   x[i]=x[i]/a[i][i];
 }

// Probe:
for (int i=0; i<n; i++)
  {
    p[i]=-b[i];
    for (int j=0; j<n; j++)
      p[i]=p[i]+a[i][j]*x[j];
  }
}

void Loesung_und_Probe_anzeigen(double x[] ,double p[], int n)
{
for (int i=0; i<n; i++)
  {
```

```
    AnsiString is=IntToStr(i);
    Form1->Memo1->Lines->Add("p"+is+"="+FloatToStr(p[i]));
    Form1->Memo1->Lines->Add("x"+is+"="+FloatToStr(x[i]));
  }
}

bool test_randomGaussElimination(int k)
{// teste mit zufälligen Koeffizienten a und b
bool result = true;
for (int i=0; i<k; i++)
  { //
    double a[n][n]={{rand(),rand(),rand()},
                    {rand(),rand(),rand()},
                    {rand(),rand(),rand()}};
    double b[n]={rand(),rand(),rand()};
    double x[n],  // Lösung
           p[n];  // Probe
    GaussElimination(a, b, x ,p, n);
    double diff=0;
    for (int i=0; i<n; i++)
      diff+=p[i]*p[i];
    if (!assertEqual(diff,0.0,"Gausselim")) result=false;
  }
return result;
}

void TestGaussElimClick()
{ // Eliminationsverfahren von Gauß
initTestUtils();
testUtilsSummary(Form1->Memo1, test_randomGaussElimination(100));
return;

double a1[n][n]={{1,2,3},  // Koeffizienten
                 {1,4,6},
                 {2,3,7}};
double b1[n]={1,2,3};      // rechte Seite

double a2[n][n]={{2,3,-5},  // Koeffizienten
                 {4,8,-3},
                 {-6,1,4}};
double b2[n]={-10,-19,-11},      // rechte Seite
       x[n],  // Lösung
       p[n];  // Probe
GaussElimination(a1, b1, x ,p, n);
Loesung_und_Probe_anzeigen(x, p, n);
GaussElimination(a2, b2, x ,p, n);
Loesung_und_Probe_anzeigen(x, p, n);
}
```

11.3.12 Aufgaben 3.10

```
// File: C:\Loesungen_CB2006\Kap_3\3.10\ArraysU.cpp

#include <vcl.h>
#pragma hdrstop
#include "ArraysU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
```

```
        : TForm(Owner)
{
}


// --------------- Aufgaben 3.10.2 -------------------------

#include "10_2.cpp"

void __fastcall TForm1::ArrayTestClick(TObject *Sender)
{
ArrayTest1();
}

void __fastcall TForm1::EratosthenesClick(TObject *Sender)
{
SiebEratosthenes();
}

void __fastcall TForm1::MisesSimClick(TObject *Sender)
{
TestMisesSimulation();
}

void __fastcall TForm1::HornerClick(TObject *Sender)
{
test_HornerSinus();
}


// --------------- Aufgaben 3.10.3 -------------------------

#include "10_3.cpp"

void __fastcall TForm1::InitArrayClick(TObject *Sender)
{
InitArrays();
}

void __fastcall TForm1::FiboArrayClick(TObject *Sender)
{
testFibonacci();
}


// --------------- Aufgaben 3.10.4 -------------------------

// --------------- Aufgabe 1. ------------------------------

#include "10_4.cpp"

// --------------- Aufgabe 2. ------------------------------

void __fastcall TForm1::StackPushClick(TObject *Sender)
{
pushStack(Edit2->Text);
enableButtons();
}

void __fastcall TForm1::StackTopClick(TObject *Sender)
{
Memo1->Lines->Add(topStack());
enableButtons();
}

void __fastcall TForm1::InitStackClick(TObject *Sender)
{
initStack();
enableButtons();
```

```
}

void __fastcall TForm1::StackPopClick(TObject *Sender)
{
Memo1->Lines->Add("pop "+topStack());
popStack();
enableButtons();
}


// --------------- Aufgabe 3. ------------------------------

void __fastcall TForm1::TroepfelPiClick(TObject *Sender)
{
testPi();
}


// --------------- Aufgaben 3.10.5 -------------------------

#include "10_5.cpp"

void __fastcall TForm1::MinMaxClick(TObject *Sender)
{
FindMinMaxInMultidimArray();
}

void __fastcall TForm1::EntfTabClick(TObject *Sender)
{
EntfTabelle();
}

void __fastcall TForm1::GaussElimClick(TObject *Sender)
{
TestGaussElimClick();
}


// --------------- Aufgaben 3.10.6 -------------------------

int binom(int n, int k)
{
const int max=30; // für größere Werte overflow
int a[max][max]={0}; // initialisiert alle Arrayelemente mit 0
for (int i=0 ; i<max ; i++)
  for (int j=0  ; j<i ; j++)
   {
     if (i==0) a[0][0]=1;
     else if (i>0)
       {
         if (j==0) a[i][j]=1;
         else if (j>0) a[i][j]=a[i-1][j-1]+a[i-1][j];
       }
   }
if (0<=k && k<=n && n<max)
  return a[n][k];
else
  return 0;
}

void __fastcall TForm1::PascalClick(TObject *Sender)
{
Memo1->Lines->Add("Pascal Dreieck");
for (int i=1 ; i<20 ; i++)
  {
    AnsiString s;
    for (int j=0  ; j<i ; j++)
      s+=IntToStr(binom(i,j))+" ";
    Memo1->Lines->Add(s);
```

```
  }
}
```

11.3.13 Aufgaben 3.11.7

```
// File: C:\Loesungen_CB2006\Kap_3\3.11\StructU.cpp

#include <vcl.h>
#pragma hdrstop
#include "StructU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

const int Dim2 = 3;

//int sum2D1(int a[][Dim2], int n)
int sum2D1(int (*a)[Dim2], int n)
{
int s=0;
for (int i=0;i<n;i++)
  for (int j=0;j<Dim2;j++)
    s =s + a[i][j];
return s;
}

void testsum()
{
const int Dim1 = 2, Dim2 = 4;
int a2[Dim1][Dim2] = {{1,2,3},{4,5,6}};
//int x=sum2D1(a2,Dim1);
}


// --------------- Aufgabe 3.11.1 --------------------------

struct Girokonto {
    struct {
        AnsiString Anrede;
        struct {
            AnsiString  Vorname,
                        Nachname;
        } Name;
        struct {
            AnsiString PLZ, // wegen Ausland als String
                       Ort,
                       Strasse,
                       Hausnummer;// wegen 7b
        } Anschrift;
        AnsiString Ausland;
        struct {
            AnsiString Vorwahl, // wegen führender Nullen
                       Telnr;
        } Telefon;
    }Adresse;
    int Kontonr;
    Currency Kontostand,
```

```
             Kreditlimit;
};

void test_G()
{
Girokonto g;
g.Kontonr=1000;
g.Adresse.Name.Vorname="A.";
g.Adresse.Telefon.Vorwahl="0800";
}

// --------------- Aufgabe 3.11.7 --------------------------

using namespace std;

AnsiString BinaerDarst(unsigned int n, int max)
{
AnsiString s;
for (int i=1; i<=max; i++)
  {
    if (n%2==1) s = "1"+s;
    else s = "0"+s;
    n = n/2;
  }
return s;
}

AnsiString DoubleBitmuster(double d)
{
union Udouble {
  double d;
  struct {
    unsigned int mant2;
    unsigned int mant1:20;
    unsigned int exp:11;
    unsigned int sign:1;
  } s;
} u;
u.d=d;

AnsiString s=FloatToStr(d)+
  "S:"+BinaerDarst(u.s.sign,1)+
  "E:"+BinaerDarst(u.s.exp,11)+
  "M:"+BinaerDarst(u.s.mant1,20)+
   BinaerDarst(u.s.mant2,32)+"    e="+IntToStr(u.s.exp)+"    e="+IntToStr(u.s.exp-
1023);

return s;
}

void __fastcall TForm1::BitmusterClick(TObject *Sender)
{
testsum();

Memo1->Lines->Add(DoubleBitmuster(0.1));
double x=0;
for (int i=0; i<10; i++) x=x+0.1;
Memo1->Lines->Add(DoubleBitmuster(x));
Memo1->Lines->Add(DoubleBitmuster(1));

Memo1->Lines->Add(DoubleBitmuster(2));
Memo1->Lines->Add(DoubleBitmuster(3));
Memo1->Lines->Add(DoubleBitmuster(4));
}
```

11.3.14 Aufgabe 3.12.4

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\12_4.cpp

// Diese Funktionen werden in PointerU.cpp aufgerufen
// --------------- Aufgaben 3.12.4 -------------------------

// --------------- Aufgabe 1 -------------------------------

void Aufgabe_3_12_4_1()
{
int i=5;
int *pi, pj;
char *pc, pd;
// a)
//    pi=i;   // Fehler: Konvertierung von 'int' nach 'int*' nicht möglich
// b)
      pi=&i;  // *pi=5
// c)
      *pi=i;  // *pi=5
// d)
//    *pi=&i; // Fehler: Konvertierung von 'int*' nach 'int' nicht möglich
// e)
//    pi=pj;  // Fehler: Konvertierung von 'int' nach 'int*' nicht möglich
// f)
      pc=&pd; // *pc=pd
// g)
//    pi=pc;// Fehler: Konvertierung von 'char*' nach 'int*' nicht möglich
// h)
      pd=*pi; // pd=5, wegen f) auch *pc=5
// h)
      *pi=i**pc; // *pi=5*5
//f)
      pi=0;   // pi kann nicht dereferenziert werden
}


// --------------- Aufgabe 2 -------------------------------

// c)
int* f_retp()
{
int x=1;
return &x;
}

void __fastcall Aufgabe_3_12_4_2()
{
// a)
int j=10;
int* p=&j;
*p=17;
Form1->Memo1->Lines->Add(IntToStr(j)); // 17 wegen p==&j
Form1->Memo1->Lines->Add(IntToStr(*p)); // 17

// b)
*new(int)=17;
// Erzeugt eine Variable dynamisch, ohne ihre Adresse zu
// speichern. ==> Speicherleck

// c)
Form1->Memo1->Lines->Add(IntToStr(*f_retp()));

// Eine lokale, nicht statische Variable existiert nur während der
// Ausführung des Blocks, in dem sie definiert wurde. Deshalb wird durch
// *f_retp() ein nicht reservierter Speicherbereich angesprochen, und das
// Fehler. Die meisten Compiler geben bei der Rückgabe eines solchen
```

```
//  Wertes eine Warnung aus.
}
// d)

void f()
{
int* pi = new int();
// irgendwelche Operationen mit *pi
delete pi;

// Die dynamisch erzeugte Variable kann durch eine gewöhnliche
// Variable ersetzt werden:
// int pi=0;
// ...
}


// --------------- Aufgabe 3 -------------------------------

int f(int a, bool b, bool c, bool d)
{
int* p; int* q; int x=0;
if      (a==1) { p=0; q=0; }
else if (a==2) { p=new int(17); q=new int(18); }
else if (a==3) { p=&x; q=&x; }
if (b) p=q;
if (c) x=*p+*q;
if (d) { delete p; delete q; }
return x;
}

void __fastcall Aufgabe_3_12_4_3()
{
// Die verschiedenen Aufrufe führen zur Ausführung der Anweisungen

/*
a) f(0,false,true,true);          b) f(1,false,false,true);
   int* p; int* q; int x=0;          int* p; int* q; int x=0;
   x=*p+*q;            // a1         p=0; q=0;
   delete p; delete q; // a2         delete p; delete q; // b1
   return x;                         return x;           // b2

a1) Zugriff auf nicht reservierte Speicherbereiche *p, *q
a2) Operanden von delete haben keinen Wert, der von new erzeugt wurde

b1) Da p und q den Wert 0 haben, ist delete ohne Auswirkung
b2) Die Funktion gibt immer 0 zurück, und hat nie ein Speicherleck

c) f(2,false,true,false);         d) f(2,true,true,true);
   int* p; int* q; int x=0;          int* p; int* q; int x=0;
   p=new int(17); q=new int(18);     p=new int(17); q=new int(18);
   x=*p+*q;                          p=q;
   return x; // return 35            x=*p+*q;
   // c1                             delete p; delete q; // d1
                                     return x;
c1) Speicherleck: delete p1 und delete p2 fehlen

d1) Wegen p=q wird q zweimal und p nie deleted

e) f(3,true,true,true);           f) f(3,true,true,false);
   int* p; int* q; int x=0;          int* p; int* q; int x=0;
   p=&x; q=&x;                       p=&x; q=&x;
   p=q;                              p=q;
   x=*p+*q;  // x=0                 x=*p+*q; // x=0
   delete p; delete q; // e1         // f1

e1) Aufruf von delete mit der Adresse einer gewöhnlichen Variablen
f1) Kein Aufruf von delete notwendig, da auch new nicht aufgerufen wurde
*/
```

```
}


// --------------- Aufgabe 4 -------------------------------

void ptr_swap(int* x, int* y)
{
//            x     *x    y    *y     h
//            x0    *x0   y0   *y0
int h=*x; //                          *x0
*x=*y;    //        *y0
*y=h;     //                   *x0
} // *x=*y0 and *y=*x0

void Aufgabe_3_12_4_4()
{
int x=17,y=18;
Form1->Memo1->Lines->Add("x="+IntToStr(x)+" y="+IntToStr(y));    // x=17 y=18
ptr_swap(&x,&y);
Form1->Memo1->Lines->Add("x="+IntToStr(x)+" y="+IntToStr(y));    // x=18 y=17
//
}


// --------------- Aufgabe 5 -------------------------------

// a)
void assign(int*& x, int* y)
{
x=y;
}

void replace(int* x, const int* y)
{
*x=*y;
}

void clone(int*& x, const int* y)
{
x=new int(*y);
}

void Aufgabe_3_12_4_5()
{
// b) 'assign' und 'clone' können ein Speicherleck verursachen, wenn das
//    erste Argument auf eine dynamisch erzeugte Variable zeigt.
int* x=new int(17);
int* y=new int(18);
assign(x,y);
replace(x,y);
clone(x,y);
}
```

11.3.15 Aufgabe 3.12.5

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\12_5.cpp

// --------------- Aufgaben 3.12.5 -------------------------
// --------------- Aufgabe 1 -------------------------------
```

11.3.16 Aufgabe 3.12.6

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\12_6.cpp

// Diese Funktionen werden in PointerU.cpp aufgerufen
// --------------- Aufgaben 3.12.6 -------------------------
// --------------- Aufgabe 1. ------------------------------

void DynSiebEratosthenes(int n)
{
// Die einzigen beiden Änderungen: n muss keine Konstante
// mehr sein, und das Array wird dynamisch erzeugt:
/// const int n=1000;
/// bool prim[n+1];
bool* prim=new bool[n+1];
// p[i]==true, falls i eine Primzahl ist, sonst false
for (int i=0; i<=n; i++)
  prim[i] = true;
for (int i=2; i<=n; i++)
  if (prim[i])
    for (int j=i;  j<=n/i; j++)
      prim[j*i] = false;
for (int i=2; i<=n; i++)
  if (prim[i])
    Form1->Memo1->Lines->Add(IntToStr(i));
};


// --------------- Aufgabe 2. ------------------------------

// a)

typedef int T; // T irgendein Datentyp

void ReAllocate(T*& a, int n_copy, int n_new)
{
T* pnew=new T[n_new]; // erzeuge das neue Array
if (n_copy>n_new) n_copy= n_new;
for (int i=0; i<n_copy; i++)
  pnew[i]=a[i]; // kopiere das alte in das neue Array
delete [] a;    // Speicher für das alte Array wieder freigeben
// Wegen diesem delete darf ReAllocate nur
// mit einem Argument aufgerufen werden, das auf ein
// dynamisch erzeugtes Array zeigt.
a=pnew;         // Zeiger auf das neue Array zurückgeben
}

void test_ReAllocate_1()
{
int n0=1;
T* d=new int[n0];
for (int n=n0; n<1000; n=2*n )
  {
    ReAllocate(d,n,n);
    for (int i=0;i<n ;i++ )
      d[i]=i;
  }
delete [] d;    // dont't forget
}

void test_ReAllocate_2()
{
// b)
    int* p1;
    ReAllocate(p1,0,100);
    // Der nicht initialisierte Zeiger p1 wird in
    // ReAllocate mit delete[] freigegeben - Fehler.
```

```
// c)
    int* p2=new int;
    ReAllocate(p2,0,100);
    // Der mit new angelegte Zeiger p2 wird in
    // ReAllocate mit delete[] freigegeben - Fehler.
    // p3 zeigt nicht auf einen Speicherbereich, der
    // mit new angelegt wurde. Der Aufruf von delete
    // in ReAllocate ist ein Fehler.
}
```

11.3.17 Aufgabe 3.12.7

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\12_7.cpp

// Diese Funktionen werden in PointerU.cpp aufgerufen
// --------------- Aufgabe 3.12.7 ---------------------------
// --------------- Aufgabe 1. -------------------------------

void Aufgabe_3_12_7()
{
int a[10]={1,3,5,7}, *p=a;
// *p      // = a[0] = 1
// *p+1    // = a[0]+1 = 2
// (*p)+1  // = a[0]+1 = 2
// *(p+1)  // = a[1] = 3
// *(p+3**p) // = a[3] = 7
// *p**p   // = a[0]*a[1] = 1
*(p+*(p+1)); // = a[3] = 7

//int a[10]={1,3,5,7}, *p=a;
Form1->Memo1->Lines->Add(*p);      // = a[0] = 1
Form1->Memo1->Lines->Add(*p+1);    // = a[0]+1 = 2
Form1->Memo1->Lines->Add((*p)+1);  // = a[0]+1 = 2
Form1->Memo1->Lines->Add(*(p+1));  // = a[1] = 3
Form1->Memo1->Lines->Add(*(p+3**p)); // = a[3] = 7
Form1->Memo1->Lines->Add(*p**p);   // = a[0]*a[0] = 1
}


// --------------- Aufgabe 2. -------------------------------

void AuswSortPtr()
{ // Der einzige Zweck dieser Funktion ist, Zeigerarithmetik
  // bei Arrays zu üben. Sie hat keinen Vorteil gegenüber
  // einem Arrayzugriff mit dem Indexoperator.
const int n=5;
int a[n]={3,1,5,7,9};
for (int i=0; i<n-1; i++)
  {
    int x,min = i;
    for (int j=i+1; j<n; j++)
      if (*(a+j)<*(a+min)) min = j;
    x = *(a+i);
    *(a+i) = *(a+min);
    *(a+min) = x;
  }
}
```

11.3.18 Aufgabe 3.12.8

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\12_8.cpp

// Diese Funktionen werden in PointerU.cpp aufgerufen
// --------------- Aufgaben 3.12.8 -------------------------
// --------------- Aufgabe 1 -------------------------------


// a)

void AuswSort(int a[], int n)
{
for (int i=0; i<n-1; i++)
  {
    int min = i;
    for (int j=i+1; j<n; j++)
      if (a[j]<a[min]) min = j;
    int x = a[i];
    a[i] = a[min];
    a[min] = x;
  }
}

// b)

AnsiString isSorted(int a[], int n)
{
bool sorted=true;
for (int i=0; i<n-1; i++)
  if (a[i]>a[i+1]) sorted=false;
if (sorted) return "sortiert";
else return "nicht sortiert";
}
// c)

void Test(int A[], int n)
{
Form1->Memo1->Lines->Add("vorher: "+isSorted(A,n));
AuswSort(A,n);
Form1->Memo1->Lines->Add("nachher: "+isSorted(A,n));
}

void Aufgabe_3_12_8_1()
{
int a1[1]={1};       Test(a1,1);
int a21[2]={1,2};    Test(a21,2);
int a22[2]={2,1};    Test(a22,2);
int a31[3]={0,1,2};  Test(a31,3);
int a32[3]={0,2,1};  Test(a32,3);
int a33[3]={1,0,2};  Test(a33,3);
int a34[3]={1,2,0};  Test(a34,3);
int a35[3]={2,0,1};  Test(a35,3);
int a36[3]={2,1,0};  Test(a36,3);
}


// --------------- Aufgabe 2 -------------------------------

double Horner(double x, int n, double* p)
{ // p: Array mit den Koeffizienten des Polynoms,
  // n: der Grad des Polynoms
double s = 0;
for (int i=0; i<n+1; i++)
    s = s*x + p[n-i];
return s;
}
```

```
#include <cmath>
void test_HornerSinus()
{
double k3=3*2*1;
double k5=5*4*k3;
double k7=7*6*k5;
const int n=7; // Der Grad des Polynoms
double p[n+1]={0, 1, 0, -1/k3, 0, +1/k5, 0, -1/k7};
for (int i=1 ;i<=31 ;i++ )
  {
  double x=i/10.0;
  double s1=std::sin(x);
//s1=poly(x,n,p);
  double s2=Horner(x,n,p);
  Form1->Memo1->Lines->Add("x="+FloatToStr(x)+" sin="+s1+
         " Horn-sin="+FloatToStr(s2)+" diff="+std::abs(s1-s2));
  // Offensichtlich liefert die Taylor-Reihe für kleine Wist stimmen die beiden
Ergebnisse für
  }
}

/*
Die im C++Builder (aber nicht im C++-Standard) vordefinierte Funktion

  double poly(double x, int degree, double coeffs[]);  // #include <math.h>

berechnet wie Horner den Funktionswert des Polynoms,
dessen Koeffizienten im Array coeffs übergeben werden.
Dabei ist x der Wert, für den der Funktionswert be¬rechnet wird, und
degree der Grad des Polynoms.
*/

// --------------- Aufgabe 3 -------------------------------

// b)
int sum4(int a[])
{
// In dieser Funktion ist das Array in einen Zeiger konvertiert.
// Deswegen gilt bei einem 32-bit System sizeof(a)==4,
// und sizeof(a)/sizeof(a[0])==1;
Form1->Memo1->Lines->Add("sum4:
sizeof(a)/sizeof(a[0])="+IntToStr(sizeof(a)/sizeof(a[0]) ));

int s=0;
for (int i=0;i< sizeof(a)/sizeof(a[0]);i++)
  s = s + a[i];
return s;
}

int sum(int a[], int n)
{
return 0;
}

void sizeof_Array()
{
// a)
int a[10];
// Im Gültigkeitsbereich des Arrays gilt bei einem 32-bit
// System sizeof(a)==10*4, und
//
//   sizeof(a)/sizeof(a[0])==10;
int s=sum(a,sizeof(a)/sizeof(a[0])); // sum wie im Text

// b)
sum4(a);
}
```

11.3.19 Aufgabe 3.12.10

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\12_10.cpp

// Diese Funktionen werden in PointerU.cpp aufgerufen
// --------------- Aufgaben 3.12.10 ------------------------

// --------------- Aufgabe 1 -------------------------------

// a)
int Leerzeichen(const char* s)
{
int n=0;
while (*s!='\0')
  {
    if (*s==' ') n++;
    s++;
  }
return n;
}

// b)
// Der Funktionsname "Leerzeichen" wird mit "LZ" abgekürzt:
// { LZ("");0 | LZ("1");0 | LZ("1 ");1 | LZ(" 1");1 | LZ(" 1 ");2 }

// c)
#include "..\..\CppUtils\TestUtils.h"

bool test_Leerzeichen()
{// ein einfacher Test für die Funktion Leerzeichen
// { LZ("");0 | LZ("1");0 | LZ("1 ");1 | LZ(" 1");1 | LZ(" 1 ");2 }
bool result = true;
if (!assertEqual(Leerzeichen(""),   0,"Leerzeichen(\"\")"))
  result=false;
if (!assertEqual(Leerzeichen("1"),  0,"Leerzeichen(\"1\")"))
  result=false;
if (!assertEqual(Leerzeichen("1 "), 1,"Leerzeichen(\"1 \")"))
  result=false;
if (!assertEqual(Leerzeichen(" 1"), 1,"Leerzeichen(\" 1\")"))
  result=false;
if (!assertEqual(Leerzeichen(" 1 "),2,"Leerzeichen(\" 1 \")"))
  result=false;
return result;
}


// --------------- Aufgabe 2. ------------------------------

void Aufgabe_3_12_10_2()
{ /*
char *x;     // Diese Definition reserviert Speicher für die
             // Adresse eines Zeigers auf char.
x = "hello"; // Hier wird Speicherplatz für das Literal "hello"
             // reserviert und dessen Adresse x zugewiesen.

a) Falsch: Es wird kein nicht reservierter Speicherbereich
           überschrieben

b) Falsch: Diese Anweisung hat keine Zugriffsverletzung zur Folge.

c) Richtig: a) und b) sind gleichwertig, wenn man vom
           unterschiedlichen Zeitpunkt der Zuweisung absieht.
```

```
d) Auch diese Deklaration ist möglich. Allerdings hat x hier den
           Datentyp "Array mit Elementen des Datentyps char".
*/
}


// --------------- Aufgabe 3 -------------------------------

void Aufgabe_3_12_10_3()
{
char const* s="blablabla";
// a)
  if (s==" ") ;    // Vergleicht die Adressen der beiden Stringliterale
                   // "blablabla" und " " und nicht etwa die Strings

// b)
//  if (*s==" ") ; // char und char* können nicht verglichen werden

// c)
  if (*s=='a'+1) ; // Vergleicht das Zeichen an der Adresse von s (also
                   // 'b') mit dem Zeichen 'a'+1 (ebenfalls 'b').
                   // Ergebnis: true

// d)
//  if (s==' ') ;  // char* und char können nicht verglichen werden

// e)
Form1->Memo1->Lines->Add(s);    // "blablabla"
Form1->Memo1->Lines->Add(s+1);  // "lablabla"
Form1->Memo1->Lines->Add(s+20); // undefiniert, eventuell
                                // eine Zugriffsverletzung

// f)
char c='A';char a[100];
strcpy(a,&c); // Kopiert alle Zeichen ab der Adresse von c bis zum
              // nächsten Nullterminator. Falls in den ersten 100 Bytes,
              // die auf c folgen, kein Nullterminator kommt, kann das
              // zu einer Zugriffsverletzung führen.
}


// --------------- Aufgabe 4 -------------------------------

void Aufgabe_3_12_10_4()
{
const int i=17;
int* p1;        char* q1;
const int* p2;      const char* q2;
int const* p3;      char const* q3;
int* const p4=p1;    char* const q4=q1;

// a)
   // p1=p2; // cannot convert 'const int *' to 'int *'
// b)
   // p1=&i; // cannot convert 'const int *' to 'int *'
// c)
   p2=p1;
// d)
   // *p3=i; // connot modify const object
// e)
//   *p4=18; // Syntaktisch ok, aber kein Speicher reserviert für *p4.

// f)
   // q1=q2;  // cannot convert 'const char *' to 'char *'
// g)
   q1="abc";  // deprecated
// h)
   q2=q1;
// i)
```

```
   // *q3='x'; // cannot modify const object
// j)
//   *q4='x'; // Syntaktisch ok, aber kein Speicher reserviert für *p4.
}


// --------------- Aufgabe 5 -------------------------------

// siehe die Funktion StringSimilarityEditDistance in
#include "..\..\CppUtils\StringUt.h"

void Aufgabe_3_12_10_5()
{
double ld1=StringSimilarityEditDistance("receieve","receieve");  // 1-0/8=1
Form1->Memo1->Lines->Add(FloatToStr(ld1));
double ld2=StringSimilarityEditDistance("receieve","receive");   // 1-1/8=0.875
Form1->Memo1->Lines->Add(FloatToStr(ld2));
double ld3=StringSimilarityEditDistance("receieve","receiver");  // 1-2/8=0.75
Form1->Memo1->Lines->Add(FloatToStr(ld3));
double ld4=StringSimilarityEditDistance("receieve","retrieve");  // 1-2/8=0.75
Form1->Memo1->Lines->Add(FloatToStr(ld4));
double ld5=StringSimilarityEditDistance("receieve","reactive");   // 1-3/8
Form1->Memo1->Lines->Add(FloatToStr(ld5));

// http://www.cs.mu.oz.au/385/pp/lecture16.pdf
// http://www.merriampark.com/ld.htm
double ld6=StringSimilarityEditDistance("vintner","writers"); // 1-5/7,
Form1->Memo1->Lines->Add(FloatToStr(ld6));
double ld7=StringSimilarityEditDistance("gumbo","gambol"); // 1-2/6,
Form1->Memo1->Lines->Add(FloatToStr(ld7));
double ld8=StringSimilarityEditDistance("intention","execution"); // 1-5/9,
Form1->Memo1->Lines->Add(FloatToStr(ld8));
}


// --------------- Aufgabe 6 -------------------------------

int Checksum(const char* name)
{
char a[10]; // 10 is enough
strcpy(a,name);
int s=0;
for (int i=0; a[i]!=0; i++) s=s+a[i];
return s;
}

void Aufgabe_3_12_10_6()
{
int c= Checksum("Check me, baby");
/*
Der Aufruf von Checksum mit einem Argument, das länger als 10
Zeichen ist, kann dazu führen, dass nicht reservierte
Speicherbereiche überschrieben werden (buffer overflow).

Solche Funktionen werden gerne von Viren-Programmierern benutzt.
Mit geschickt gewählten Argumenten wird
1. die Rücksprungadresse auf dem Stack überschrieben, und
2. Code übergeben, der dann ausgeführt wird.
*/
}


// --------------- Aufgabe 7 -------------------------------

void assign_c(int*& x, int* y)
{ // Da *y nie verändert wird, könnte es als const definiert werden.
  // Aber dann könnte es nicht mehr einem nicht konstanten Zeiger wie
  // x zugewiesen werden.
x=y;
```

```
}

void replace_c(int* x, const int* y)
{ // Da *y nie verändert wird, kann es als const definiert werden.
*x=*y;
}

void clone_c(int*& x, const int* y)
{ // Da *y nie verändert wird, kann es als const definiert werden.
x=new int(*y);
}

void Aufgabe_3_12_10_7()
{
int* x=new int(17);
int* y=new int(18);
assign_c(x,y);
replace_c(x,y);
clone_c(x,y);
}
```

11.3.20 Aufgabe 3.12.11

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\LinkedListU.cpp

#include <vcl.h>
#pragma hdrstop

#include "LinkedListU.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------

// --------------- Aufgaben 3.12.11 ------------------------


// --------------- Aufgabe 1 -------------------------------

//typedef int T;
typedef AnsiString T;

struct Listnode {
  T data;            // Daten
  Listnode* next;
};

Listnode* first=0;
Listnode* last=0;
// Normalerweise ist die Verwendung von globalen Variablen wie
// first und last nicht empfehlenswert.
// Globale Variablen lassen sich mit Klassen leicht vermeiden.
// Diese Lösung ist so formuliert, dass sie einfach in eine
// objektorientierte Version übertragen werden kann. Dazu muss
// sie im Wesentlichen nur zwischen class{ ... } geschrieben werden.


// --------------- Aufgabe 1 a) ----------------------------
```

```
Listnode* newListnode(const T& data, Listnode* next)
{ // Gibt einen Zeiger auf einen neuen Listenknoten {d0,nx0} zurück,
  // wobei d0 und nx0 die Argumente für data und next sind.
Listnode* n=new Listnode;
n->data = data;
n->next = next;
return n; // Nachbedingung: Der Rückgabewert zeigt auf einen neuen
}           //                Knoten mit den Elementen {d0,nx0}

void pushFront(const T& data)   // first  last data Neuer Knoten n
{                               //  f0     l0   d0    n0
first=newListnode(data, first); //  n0                ->{d0,f0}
if (last==0)
  last=first;                   //         n0
// Nachbedingung:
// Fall I, f0==l0==0: first==last->{d0,0}
// Fall II, l0!=0:    first->{d0,f0}
}

void __fastcall TForm1::PushFrontClick(TObject *Sender)
{
pushFront(Edit1->Text);
}


// --------------- Aufgabe 1 b) ----------------------------

void showList(Listnode* start)
{ // Gibt alle Daten der Liste ab der Position start aus
Form1->Memo1->Lines->Add("Liste:");
Listnode* tmp=start;
while (tmp != 0)
  {
    Form1->Memo1->Lines->Add(tmp->data);
    tmp = tmp->next;
  }
}

void __fastcall TForm1::ShowListClick(TObject *Sender)
{
showList(first);
}


// --------------- Aufgabe 1 c) ----------------------------

/*
                                first       n1      n2      n3        n4
-----------------------------------------------
first=0;                          0
first=newListnode("10",first);   n1     ->{"10",0}
first=newListnode("11",first);   n2             ->{"11",n1}
first=newListnode("12",first);   n3                     ->{"12",n2}
first=newListnode("13",first);   n4                             ->{"13",n3}

Das entspricht der Liste:

first -> {"13",n3} -> {"12",n2} -> {"11",n1} -> {"10",0}

Nach jedem Aufruf zeigt first auf den zuletzt eingefügten Knoten.
In diesem Knoten zeigt next auf den zuvor eingefügten Knoten, usw.
Im letzten Knoten der Liste hat next den Wert 0.
Diese Liste ist also eine LIFO-Liste.
*/

// --------------- Aufgabe 1 d) ----------------------------

Listnode* findData(Listnode* start, const T& x)
```

```
{
Listnode* found=0;
Listnode* tmp=start;
while (tmp != 0 && found==0)
  {
    if(tmp->data==x) found=tmp;
    tmp = tmp->next;
  }
return found;
  // Nachbedingung:
  //   Entweder found==0: Dann wurden alle Knoten der Liste von start
  //     bis zum Ende durchlaufen, und die Bedingung tmp->data==x
  //     ist nie eingetreten.
  //   Oder found!=0: Dann wurden alle Knoten der Liste durchlaufen,
  //     bis das erste Mal tmp->data==x gilt.
  //     Ein Zeiger auf diesen Knoten wird dann zurückgegeben.
}

void __fastcall TForm1::FindDataClick(TObject *Sender)
{
Listnode* p=findData(first,Edit1->Text);
if (p==0)
  Memo1->Lines->Add(Edit1->Text+": nicht gefunden");
while (p!=0)
  {
    Memo1->Lines->Add(p->data+": gefunden");
    p=findData(p->next,Edit1->Text);
  }
}


// --------------- Aufgabe 1 e) ----------------------------

void insertLastListnode(const T& data)  // first last Neuer Knoten n
                                        //  f0    l0
{ // Erzeugt einen neuen Listen-Knoten und fügt diesen nach last ein.
  // Last zeigt anschließend auf den letzten und first auf den
  // ersten Knoten der Liste.
Listnode* n=newListnode(data,0); //                      n0 ->{d0,0}
if (last==0) first=n;            //          n0
else last->next=n;               //               l0->{d,n0}
last=n;                          //               n0
// Nachbedingung: Bezeichnet man den Wert von last vor
//   dem Aufruf dieser Funktion mit l0, gilt
//   Fall I,  l0==0: first==n && last==n
//   Fall II, l0!=0: l0->next==n && last==n
}

void pushBack(const T& data)
{
insertLastListnode(data);
}

void __fastcall TForm1::PushBackClick(TObject *Sender)
{
pushBack(Edit1->Text);
}


// --------------- Aufgabe 1 f) ----------------------------

Listnode* findBefore(const T& data )
{
Listnode* result=first;
Listnode* tmp=first;
while (tmp!=0 && data>tmp->data )
  {
    result=tmp;
    tmp=tmp->next;
```

```
  }
return result;
// result==first: data>first->tmp - füge vor oder nach first ein
// result!=first: data>result->tmp
}

void insertSorted(const T& data)
{
  // Diese Fallunterscheidungen sind deshalb notwendig,
  // weil ein neuer Knoten anders eingefügt werden muss
  // - in eine leere Liste
  // - am Anfang einer nicht leeren Liste
  // - innerhalb einer nicht nicht leeren Liste
  // - am Ende einer nicht leeren Liste.
if (first==0)
  { // Die Liste ist leer. Deshalb das neue Element
    // an der Position first einhängen.
    first=newListnode(data,first);
    last=first;
  }
else
  {
    // Die Liste ist nicht leer. Suche die Einfügeposition.
    Listnode* p=findBefore(data);
    if (p==first && data<p->data)
      {
        // Der neue Knoten kommt vor dem ersten Knoten.
        first=newListnode(data,first);
      }
    else if (p==last)
      { // Der neue Knoten wird der neue letzte Knoten.
        insertLastListnode(data);
      }
    else
      { // Der neue Knoten wird nach p eingefügt.
        // first und last werden nicht verändert.
        p->next=newListnode(data,p->next);
      }
  }
}

void __fastcall TForm1::InsertSortedClick(TObject *Sender)
{
insertSorted(Edit1->Text);
}


// --------------- Aufgabe 1 g) ----------------------------

Listnode* secondlast()
{
Listnode* result=0;
if (first!=0)
  {
    Listnode* tmp=first;
    while (tmp->next!=0)
      {
        result=tmp;
        tmp=tmp->next;
      }
  }
return result;
}

void eraseListnode(Listnode*& pn)
{ // entfernt *pn aus der Liste
if (pn!=0) // für pn==0 nichts machen
  {
    Listnode* tmp = pn;
```

```
    if (pn->next!=0)
      { // pn ist nicht der letzte Knoten. Deshalb muss last nicht
        // angepasst werden.
        pn = pn->next;
        if (tmp==first)
          first=pn;
      }
    else
      { // pn->next==0, d.h. pn ist der letzte Knoten.
        // Der bisherige vorletzte wird dann zum letzten.
        last=secondlast();
        if (last==0)
          first=0;
        else
          last->next=0;
      }
    delete tmp;
  }
}

void __fastcall TForm1::EraseNodeClick(TObject *Sender)
{
Listnode* p=findData(first,Edit1->Text);
if (p!=0)
  {
    eraseListnode(p);
    Memo1->Lines->Add(Edit1->Text+": gelöscht");
  }
else
  Memo1->Lines->Add(Edit1->Text+": nicht gefunden");
}


// --------------- Aufgabe 1 h) ----------------------------

void clearList()
{ // löscht alle Knoten der Liste
while (first!=0) eraseListnode(first);
first=0;
last=0;
}

void __fastcall TForm1::ClearListClick(TObject *Sender)
{
clearList();
first=0;
last=0;
}


// --------------- Aufgabe 1 i) ----------------------------
/*
Beziehung: Bei einer nicht leeren Liste zeigt first auf den
           ersten und last auf den letzten Knoten der Liste.
           Bei einer leeren Liste haben first und last den Wert 0.
a) Das Ablaufprotokoll im Text legt nahe, dass diese Beziehung
   generell gilt.
b) showList und findData verändern keine Knoten der Liste und
   zerstören deshalb auch diese Beziehung nicht.
d) wie a)
e) Die Konstruktion des Algorithmus lässt erwarten, dass diese
   Beziehung auch nach insertSorted noch gilt. Der Nachweis aus
   dem Quelltext dürfte aber etwas aufwendig werden.
f) Falls nicht gerade der letzte Knoten gelöscht wird, wird diese
   Beziehung durch die Konstruktion des Algorithmus nahegelegt.
   Beim Löschen des letzten Knotens wird der bisherige vorletzte
   Knoten zum letzten Knoten.
   Falls ein Knoten den Vorgänger nicht wie bei einer doppelt
   verketteten Liste enthält, muss der zeitaufwendig bestimmt
```

```
   werden.
g) first und last werden auf 0 gesetzt, und das entspricht
   einer leeren Liste.
*/


// --------------- Aufgabe 1 j) ----------------------------

#include "..\..\CPPUtils\TestUtils.h"

// Size ist für einige der folgenden Tests notwendig.
int Size(Listnode* t)
{ // gibt die Anzahl der auf t folgenden Knoten zurück
int n=0;
while (t!=0)
  {
    n++;
    t=t->next;
  }
return n;
}

bool test_pushFront(int n)
{ // Füge Knoten ein und prüfe first, last und Size
bool result=true;
clearList();
pushFront("5");
if (!assertEqual(first->data,"5",
  "pushFront 5 first")) result=false;
if (!assertEqual(last->data,"5",
  "pushFront 5 last")) result=false;
if (!assertEqual(Size(first),1,
  "Size()==1")) result=false;

pushFront("10");
if (!assertEqual(first->data,"10",
  "pushFront 10 first")) result=false;
if (!assertEqual(last->data,"5",
  "pushFront 5 last")) result=false;
if (!assertEqual(Size(first),2,
  "Size()==2")) result=false;

// Viele Tests
clearList();
for (int i=0; i<n; i++)
  {
    pushFront(IntToStr(i));
    if (!assertEqual(first->data,IntToStr(i),
      "pushFront 0 first")) result=false;
    if (!assertEqual(last->data,"0",
      "pushFront n last")) result=false;
    if (!assertEqual(Size(first),i+1,
      "Size()==n")) result=false;
  }
return result;
}

bool test_showList()
{ // Ein automatischer Test von showList müsste den Text im
  // im Memo prüfen. Da das relativ aufwendig ist, scheint ein
  // manueller Test angemessener zu sein.
return true;
}

bool test_findData()
{ // Einige Testfälle
bool result=true;
clearList();
// 1. Suche im leeren Baum sollte funktionieren,
```

```
//    insbesondere nicht zu einem Programmabsturz führen
if (!assertEqual(findData(first,"1"),0,
  "find in empty list")) result=false;

// 2. Ein Element einfügen und finden:
pushFront("1");
if (!assertEqual(AnsiString("1"), findData(first,"1")->data,
  "Liste mit '1'")) result=false;
pushFront("2");
if (!assertEqual(AnsiString("1"), findData(first,"1")->data,
  "Liste mit '1'")) result=false;
return result;
}

bool test_pushBack(int n)
{
bool result=true;
clearList();
insertLastListnode("5");
if (!assertEqual(first->data,"5",
  "pushBack 5 first")) result=false;
if (!assertEqual(last->data,"5",
  "pushBack 5 last")) result=false;
if (!assertEqual(Size(first),1,
  "Size()==1")) result=false;

insertLastListnode("10");
if (!assertEqual(first->data,"5",
  "pushBack 10 first")) result=false;
if (!assertEqual(last->data,"10",
  "pushBack 5 last")) result=false;
if (!assertEqual(Size(first),2,
  "Size()==2")) result=false;

// Viele Tests
clearList();
for (int i=0; i<n; i++)
  {
    insertLastListnode(IntToStr(i));
    if (!assertEqual(first->data,"0",
      "pushBack 0 first")) result=false;
    if (!assertEqual(last->data,IntToStr(i),
      "pushBack n last")) result=false;
    if (!assertEqual(Size(first),i+1,
      "Size()==n")) result=false;
  }
return result;
}

bool isSorted(Listnode* n)
{
  bool result=true;
  Listnode* tmp=n;
  T old;
  if (tmp!=0)
    old=tmp->data;
  while (tmp!=0 && result)
    {
      if (tmp->data < old)
        result=false;
      old=tmp->data;
      tmp=tmp->next;
    }
  return result;
}

bool test_isSorted()
{
bool result=true;
```

```
clearList();
if (!assertEqual(isSorted(first),true,
  "empty list sorted")) result=false;
pushFront("5");
if (!assertEqual(isSorted(first),true,
  "one element list sorted")) result=false;
pushFront("3");
if (!assertEqual(isSorted(first),true,
  "two element list sorted")) result=false;
pushFront("7");
if (!assertEqual(isSorted(first),false,
  "three element list not sorted")) result=false;
return result;
}

bool test_insertSorted_random(int max)
{
bool result=true;
clearList();
for (int i=0; i<max; i++)
  {
    insertSorted(IntToStr(rand()));
    if (!assertEqual(isSorted(first),true,
      "insertSorted_random")) result=false;
  }
return result;
}

bool test_eraseListnode(int max)
{
bool result=true;
clearList();
for (int i=0; i<max; i++)
  insertSorted(IntToStr(i));
for (int i=0; i<max/2; i++)
  {
    int x=rand();
    if (i==0) x=0;     // first node
    if (i==1) x=max-1; // last node
    Listnode* p=findData(first,IntToStr(x));
    if (p!=0)
      {
        eraseListnode(p);
        Listnode* q=findData(first,IntToStr(x));
        if (!assertEqual(q,0,
          "erase_random")) result=false;
      }
  }
if (!test_isSorted()) result=false;
return result;
}

bool test_clearList()
{
// Ohne Hilfsmittel wie CodeGuard, BoundsChecker usw. gibt
// es keine Möglichkeit, zu prüfen, ob der Speicher korrekt
// freigegeben wurde.
return true;
}

bool testLinkedList(int n)
{
bool result=test_isSorted();
if (!test_pushFront(n)) result=false;
if (!test_findData()) result=false;
if (!test_pushBack(n)) result=false;
if (!test_isSorted()) result=false;
for (int i=1; i<n;i++ )
  if (!test_insertSorted_random(i) ) result=false;
```

```
if (!test_eraseListnode(n)) result=false;
return result;
}

void __fastcall TForm1::TestListClick(TObject *Sender)
{
initTestUtils();
if (testLinkedList(100) )
  Memo1->Lines->Add("All tests passed, n="+IntToStr(TestUtilsPassCount));
else
  Memo1->Lines->Add(IntToStr(TestUtilsFailCount)+" tests failed");
}


// --------------- Aufgabe 2 -------------------------------


// --------------- Aufgabe 2 a) ----------------------------

struct DllListnode {
  T data;            // Daten
  DllListnode* next;
  DllListnode* prev;
};

DllListnode* newDllListnode(const T& data, DllListnode* next,
                                DllListnode* prev)
{ // Gibt einen Zeiger auf einen neuen Listenknoten {d0,nx0,p0} zurück,
  // wobei d0, nx0 und p0 die Argumente für data, next und prev sind.
DllListnode* n=new DllListnode;
n->data = data;
n->next = next;
n->prev = prev;
return n; // Nachbedingung: Der Rückgabewert zeigt auf einen neuen
}           //                Knoten mit den Elementen {d0,nx0,p0}

DllListnode* firstDll=0;
DllListnode* lastDll=0;


// --------------- Aufgabe 2 b) ----------------------------

void pushFrontDll(const T& data)   // first last data  n       f0
{                                  //   f0   l0   d0          ->{*,*,p1}
if (firstDll==0)
  { // erster Aufruf
    firstDll=newDllListnode(data,0,0);//n0             n0->{d0,0,0}
    lastDll=firstDll;                 //     n0
  }
else
  { // ab dem zweiten Aufruf
    firstDll=newDllListnode(data,
                      firstDll,0);    //n0            n0->{d0,f0,0}
    firstDll->next->prev=firstDll;//                          ->{*,*,n0}
  }
// Nachbedingung:
// Fall I, f0==l0==0: first==last->{d0,0,0}
// Fall II, l0!=0:    first==n, n->{d0,f0,0}, f0->{*,*,n}
}

void __fastcall TForm1::PushFrontDllClick(TObject *Sender)
{
pushFrontDll(Edit1->Text);
}


// --------------- Aufgabe 2 c) ----------------------------

void showDllForw(DllListnode* start)
```

```
{ // Gibt alle Daten der Liste ab der Position start aus
Form1->Memo1->Lines->Add("Liste:");
DllListnode* tmp=start;
while (tmp != 0)
  {
    Form1->Memo1->Lines->Add(tmp->data);
    tmp = tmp->next;
  }
}


// --------------- Aufgabe 2 d) ----------------------------

void showDllRev(DllListnode* start)
{ // Gibt alle Daten der Liste ab der Position first aus
Form1->Memo1->Lines->Add("Liste rev:");
DllListnode* tmp=start;
while (tmp != 0)
  {
    Form1->Memo1->Lines->Add(tmp->data);
    tmp = tmp->prev;
  }
}

void __fastcall TForm1::ShowDllListClick(TObject *Sender)
{
showDllForw(firstDll);
showDllRev(lastDll);
}


// --------------- Aufgabe 2 e) ----------------------------
/*
Aus Platzgründen wurde firstDll durch fD, lastDll durch lD abgekürzt.
Die Zeiger auf die neu erzeugten Knoten werden mit n1, n2 usw.
bezeichnet.
Die Werte im Ablaufprotokoll erhält man einfach durch Einsetzen der
Nachbedingung von pushFrontDll.

                        // fD  lD    n1          n2            n3
fD=0                       0
lD=0                           0
pushFrontDll(d1)           n1  n1   ->{d1,0,0}
pushFrontDll(d2)           n2       ->{d1,0,n2} ->{d2,n1,0}
pushFrontDll(d3)           n3                   ->{d2,n1,n3}  ->{d3,n2,0}
*/

// --------------- Aufgabe 2 f) ----------------------------

void pushBackDll(const T& data)
{ // Erzeugt einen neuen Listen-Knoten und fügt diesen
  // nach last ein. Last zeigt anschließend auf den
  // letzten und first auf den ersten Knoten der Liste.
if (lastDll==0)
  {
    firstDll=newDllListnode(data,0,0);
    lastDll=firstDll;
  }
else
  {
    DllListnode* n=newDllListnode(data,0,lastDll);
    lastDll->next=n;
    lastDll=n;
  }
// Nachbedingung:
// Fall I, f0==l0==0: first==last->{d0,0,0}
// Fall II, l0!=0:    last==n, n->{d0,0,l0}, l0->{*,n,*}
}
```

```
void __fastcall TForm1::PushBackDllClick(TObject *Sender)
{
pushBackDll(Edit1->Text);
}


// --------------- Aufgabe 2 g) ----------------------------

/*
Aus Platzgründen wurde firstDll durch fD, lastDll durch lD abgekürzt.
Die Zeiger auf die neu erzeugten Knoten werden mit n1, n2 usw.
bezeichnet.
Die Werte im Ablaufprotokoll erhält man einfach durch Einsetzen der
Nachbedingung von pushBackDll.

                        // fD  lD    n1          n2            n3
fD=0                       0
lD=0                           0
pushBackDll(d1)            n1  n1   ->{d1,0,0}
pushBackDll(d2)                n2   ->{d1,n2,0} ->{d2,0,n1}
pushBackDll(d3)                n3                ->{d2,n3,n1}  ->{d3,0,n2}
*/


// --------------- Aufgabe 2 h) ----------------------------

DllListnode* findDllData(DllListnode* start, const T& x)
{
DllListnode* found=0;
DllListnode* tmp=start;
while (tmp != 0 && found==0)
  {
    if(tmp->data==x) found=tmp;
    tmp = tmp->next;
  }
return found;
}

void eraseDllListnode(DllListnode*& pn)
{ // entfernt *pn aus der Liste
if (pn!=0) // für pn==0 nichts machen
  {
    DllListnode* tmp = pn;
    if (pn->next!=0)
      {
        pn = pn->next;
        pn->prev=tmp->prev;
        if (tmp==firstDll)
          firstDll=pn;
      }
    else if (pn->prev!=0)
      { // pn->next==0, d.h. pn zeigt auf den letzten Knoten
        // Hier ist keine Funktion wie secondlast notwendig
        pn->prev->next=0;
        lastDll=pn->prev;
      }
    else
      { // pn->next==0 && pn->prev==0,
        // d.h. pn ist der einzige Knoten
        firstDll=0;
        lastDll=0;
      }
    delete tmp;
  }
}

void __fastcall TForm1::EraseDllListnodeClick(TObject *Sender)
{
DllListnode* p=findDllData(firstDll,Edit1->Text);
```

```
if (p!=0)
  {
    eraseDllListnode(p);
    Memo1->Lines->Add(Edit1->Text+": gelöscht");
  }
else
  Memo1->Lines->Add(Edit1->Text+": nicht gefunden");
}

// --------------- Aufgabe 2 i) ----------------------------

void clearDllList()
{ // löscht alle Knoten der Liste
while (firstDll!=0) eraseDllListnode(firstDll);
firstDll=0;
lastDll=0;
}

// --------------- Aufgabe 2 j) ----------------------------

int Size(DllListnode* t)
{
int n=0;
while (t!=0)
  {
    n++;
    t=t->next;
  }
return n;
}

int SizeRev(DllListnode* t)
{
int n=0;
while (t!=0)
  {
    n++;
    t=t->prev;
  }
return n;
}

bool test_pushFrontDll(int n)
{
bool result=true;
clearDllList();
pushFrontDll("5");
if (!assertEqual(firstDll->data,"5",
  "pushFront 5 first")) result=false;
if (!assertEqual(lastDll->data,"5",
  "pushFront 5 last")) result=false;
if (!assertEqual(Size(firstDll),1,
  "Size()==1")) result=false;
if (!assertEqual(SizeRev(lastDll),1,
  "SizeRev()==1")) result=false;

pushFrontDll("10");
if (!assertEqual(firstDll->data,"10",
  "pushFront 10 first")) result=false;
if (!assertEqual(lastDll->data,"5",
  "pushFront 5 last")) result=false;
if (!assertEqual(Size(firstDll),2,
  "Size()==2")) result=false;
if (!assertEqual(SizeRev(lastDll),2,
  "SizeRev()==2")) result=false;

// Viele Tests
clearDllList();
for (int i=0; i<n; i++)
```

```
  {
    pushFrontDll(IntToStr(i));
    if (!assertEqual(firstDll->data,IntToStr(i),
      "pushFront 0 first")) result=false;
    if (!assertEqual(lastDll->data,"0",
      "pushFront n last")) result=false;
    if (!assertEqual(Size(firstDll),i+1,
      "Size()==n")) result=false;
    if (!assertEqual(SizeRev(lastDll),i+1,
      "SizeRev()==n")) result=false;
  }
return result;
}

bool test_pushBackDll(int n)
{
bool result=true;
clearDllList();
pushBackDll("5");
if (!assertEqual(firstDll->data,"5",
  "pushBack 5 first")) result=false;
if (!assertEqual(lastDll->data,"5",
  "pushBack 5 last")) result=false;
if (!assertEqual(firstDll,1,
  "Size()==1")) result=false;
if (!assertEqual(SizeRev(lastDll),1,
  "SizeRev()==1")) result=false;

pushBackDll("10");
if (!assertEqual(firstDll->data,"5",
  "pushBack 10 first")) result=false;
if (!assertEqual(lastDll->data,"10",
  "pushBack 10 last")) result=false;
if (!assertEqual(Size(firstDll),2,
  "Size()==2")) result=false;
if (!assertEqual(SizeRev(lastDll),2,
  "SizeRev()==2")) result=false;

// Viele Tests
clearDllList();
for (int i=0; i<n; i++)
  {
    pushBackDll(IntToStr(i));
    if (!assertEqual(firstDll->data,"0",
      "pushBack 0 first")) result=false;
    if (!assertEqual(lastDll->data,IntToStr(i),
      "pushBack n last")) result=false;
    if (!assertEqual(Size(firstDll),i+1,
      "Size()==n")) result=false;
    if (!assertEqual(SizeRev(lastDll),i+1,
      "SizeRev()==n")) result=false;
  }
return result;
}

bool testDllList(int n)
{
bool result=test_isSorted();
if (!test_pushFrontDll(n)) result=false;
if (!test_pushBackDll(n)) result=false;
return result;
}

void __fastcall TForm1::TestDllClick(TObject *Sender)
{
initTestUtils();
if (testDllList(100) )
  Memo1->Lines->Add("All tests passed, n="+IntToStr(TestUtilsPassCount));
else
```

```
  Memo1->Lines->Add(IntToStr(TestUtilsFailCount)+" tests failed");
}


// --------------- Aufgabe 3 -------------------------------
```

Lösung 3.12.11, 3.,

Vollständige Induktion: Alle Ergebnisse sieht man einfach durch Einsetzen der jeweiligen Nachbedingung.

3. a)

Die Zeiger auf die jeweils erzeugten Knoten werden mit n_1, n_2, …, n_k bezeichnet.

Induktionsanfang: Damit man die Struktur sieht, werden die Ergebnisse der ersten zwei Aufrufe dargestellt.

```
                          // first last n1        n2
                             0     0

   // 1. Aufruf pushFront(d1) n1     n1   ->{d1,0}

   // 2. Aufruf pushFront(d2) n2          ->{d1,0}  ->{d2,n1}
```

Induktionsvoraussetzung: Bei den ersten k Aufrufen wurden k Knoten erzeugt, auf die n_1, n_2, …, n_k zeigen:

```
                          // first last n1        n2    ... nk          nk+1
   // k Aufrufe pushFront(dk)  nk     n1  ->{d1,0}  ->{d2,n1} ->{dk,nk-1}
```

Induktionsschritt: Der (k+1)-te Aufruf von pushFront(d_{k+1}): Einsetzen der Nachbedingung von pushFront

```
   // k+1 Aufr. pushFront(dk+1) nk+1       ->{d1,0}  ->{d2,n1} ->{dk,nk-1}   ->{dk+1,nk}
```

3. b) ----------

Die Zeiger auf die jeweils erzeugten Knoten werden mit n_1, n_2, …, n_k bezeichnet.

Induktionsanfang: Damit man die Struktur sieht, werden die Ergebnisse der ersten zwei Aufrufe dargestellt.

```
                          // first last n1        n2
                             0     0

   // 1. Aufruf pushBack(d1) n1     n1   ->{d1,0}

   // 2. Aufruf pushBack(d2)        n2   ->{d1,n2}  ->{d2,0}
```

Induktionsvoraussetzung: Bei den ersten k Aufrufen wurden k Knoten erzeugt, auf die n_1, n_2, …, n_k zeigen:

```
                          // first last n1        n2    ... nk          nk+1
   // k Aufrufe pushBack(dk)  n1     nk  ->{d1,n2}  ->{d2,n3}  ->{dk,0}
```

Induktionsschritt: Der (k+1)-te Aufruf von pushFront(d_{k+1}): Einsetzen der Nachbedingung von pushBack

```
   // k+1 Aufr. pushBack(dk+1) n1     nk+1 ->{d1,n2}  ->{d2,n3} ->{dk,nk+1}   ->{dk+1,0}
```

3. c) ----------

Die Zeiger auf die jeweils erzeugten Knoten werden mit n_1, n_2, …, n_k bezeichnet. Aus Platzgründen werden *pushFrontDll* mit *pushFDll*, *firstDll* mit *first* und *lastDll* mit *last* abgekürzt:

Induktionsanfang: Damit man die Struktur sieht, werden die Ergebnisse der ersten zwei Aufrufe dargestellt.

```
                          // first last n1          n2
                             0     0

// 1. Aufruf pushFDll(d1)  n1     n1   ->{d1,0,0}

// 2. Aufruf pushFDll(d2)  n2          ->{ d1,0,n2} ->{d2,n1,0}
```

Induktionsvoraussetzung: Bei den ersten k Aufrufen wurden k Knoten erzeugt, auf die n_1, n_2, …, n_k zeigen:

```
                          // first last  n1          ... nk                nk+1
// k Aufrufe pushFDll(dk)   nk     n1   ->{d1,0,n2}    ->{dk,nk-1,0}
```

Induktionsschritt: Der (k+1)-te Aufruf von *pushFrontDll*(d_{k+1}): Einsetzen der Nachbedingung von *pushFrontDll*:

```
// k. Aufruf pushFDll(dk+1)  nk+1   n1   ->{d1,0,n2}  ->{dk,nk-1,nk+1}  ->{dk+1,nk,0}
```

3. d) ----------

Die Zeiger auf die jeweils erzeugten Knoten werden mit n_1, n_2, …, n_k bezeichnet. Aus Platzgründen werden *pushBackDll* mit *pushBDll*, *firstDll* mit *first* und *lastDll* mit *last* abgekürzt:

Induktionsanfang: Damit man die Struktur sieht, werden die Ergebnisse der ersten zwei Aufrufe dargestellt.

```
                          // first last n1             n2
                             0     0

// 1. Aufruf pushBDll(d1)  n1     n1   ->{d1,0,0}

// 2. Aufruf pushBDll(d2)          n2   ->{d1,n2,0}    ->{d2,0,n1}
```

Induktionsvoraussetzung: Bei den ersten k Aufrufen wurden k Knoten erzeugt, auf die n_1, n_2, …, n_k zeigen:

```
                          // first last n1          ... nk            nk+1
// k Aufrufe pushBDll(dk)   n1     nk   ->{d1,n2,0}    ->{dk,0,nk-1}
```

Induktionsschritt: Der (k+1)-te Aufruf von *pushBackDll*(d_{k+1}): Einsetzen der Nachbedingung von *pushBackDll*:

```
// k. Aufruf pushBDll(dk+1)  n1   nk+1  ->{d1,n2,0}     ->{dk,nk+1,nk-1}->{dk+1,0,nk}
```

11.3.21 Aufgabe 3.12.12

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\TreeU.cpp

#include <vcl.h>
#pragma hdrstop
#include "TreeU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
     : TForm(Owner)
```

```
{
}
//---------------------------------------------------------------------


// --------------- Aufgabe 3.12.12 -------------------------

// --------------- Aufgabe 3.12.12, 1 a) -------------------

// Die Definitionen sind in einer extra Datei, damit sie für die
// Aufgaben von Abschnitt 5.3.6 verwendet werden können.

typedef AnsiString dataType;
typedef AnsiString keyType;
#include "TreeDefs.h"

// --------------- Aufgabe 3.12.12, 1 b) -------------------

Treenode* root=0;

void __fastcall TForm1::InsertClick(TObject *Sender)
{
insertBinTreenode(root, Edit1->Text, Edit2->Text);
}


// --------------- Aufgabe 3.12.12, 1 c) -------------------

Treenode* searchBinTree(const keyType& x)
{
Treenode* result=0;
if (root!=0)
  {
    Treenode* i=root;
    while(i!=0 && i->key!=x)
      {
        if (x<i->key) i=i->left;
        else i=i->right;
      }
    if (i!=0 && i->key==x)
    result=i;
  }
return result;
}

void __fastcall TForm1::SearchClick(TObject *Sender)
{
Treenode* p=searchBinTree(Edit1->Text);
if (p!=0)
  Form1->Memo1->Lines->Add(Edit1->Text+" gefunden");
else
  Form1->Memo1->Lines->Add(Edit1->Text+" nicht gefunden");

}


// --------------- Aufgabe 3.12.12, 1 d) -------------------

bool ValueToKey(keyType Key, dataType& Value)
{
Treenode* p=searchBinTree(Key);
if (p==0)
  return false;
else
  {
    Value=p->data;
    return true;
  }
}
```

```
void __fastcall TForm1::KeyToValueClick(TObject *Sender)
{
dataType Value;
if (ValueToKey(Edit1->Text,Value))
  {
    Form1->Memo1->Lines->Add("key="+Edit1->Text+"  data: "+Value);
  }
else
  Form1->Memo1->Lines->Add("key="+Edit1->Text+"  nicht gefunden ");
}


// --------------- Aufgabe 3.12.12, 1 e) -------------------

#include "..\..\CPPUtils\TestUtils.h"

bool testBinTree()
{ // Einige Testfälle
bool result=true;
root=0;
dataType Value;
initTestUtils();

// 1. Suche im leeren Baum sollte funktionieren, und
//    insbesondere nicht zu einem Programmabsturz führen
if (!assertEqual_bool(ValueToKey("1", Value),false,
  "leerer Baum")) result=false;

// 2. Ein Element einfügen und finden:
insertBinTreenode(root,"1", "2");
if (!assertEqual_bool(ValueToKey("1", Value),true,
  "Baum mit '1'")) result=false;
if (!assertEqual(Value,"2",
  "Baum mit '1'")) result=false;

// 2. Zwei Elemente einfügen und finden:
insertBinTreenode(root,"2", "3");
if (!assertEqual(ValueToKey("1", Value),true,
  "Baum mit '1'")) result=false;
if (!assertEqual(Value,"2",
  "Baum mit '1'")) result=false;
if (!assertEqual(ValueToKey("2", Value),true,
  "Baum mit '1' und '2'")) result=false;
if (!assertEqual(Value,"3",
  "Baum mit '1'")) result=false;

return result;
}


void __fastcall TForm1::TestBintreeClick(TObject *Sender)
{
initTestUtils();
if (testBinTree())
  Memo1->Lines->Add("All tests passed, n="+IntToStr(TestUtilsPassCount));
else
  Memo1->Lines->Add(IntToStr(TestUtilsFailCount)+" tests failed");
}


// --------------- Aufgabe 5.3.6 ---------------------------

// --------------- Aufgabe 5.3.6, 1 a) ---------------------

void insertBinTreenode_rec(Treenode*& b,
                           const keyType& k, const dataType& d)
{
if (b==0) b=newTreenode(k,d,0,0);
```

```
else if (k<b->key) insertBinTreenode_rec(b->left,k,d);
else insertBinTreenode_rec(b->right,k,d);
}

void __fastcall TForm1::Insert_recClick(TObject *Sender)
{
insertBinTreenode_rec(root, Edit1->Text, Edit2->Text);
}


// --------------- Aufgabe 5.3.6, 1 b) ---------------------

void processTreenode(Treenode* n)
{
Form1->Memo1->Lines->Add("key: "+n->key+ "data: "+n->data);
}

void traverseTree(Treenode* n)
{
if (n!=0)
  {
    traverseTree(n->left);
    processTreenode(n);
    traverseTree(n->right);
  }
}

void __fastcall TForm1::TraverseClick(TObject *Sender)
{
traverseTree(root);
}


// --------------- Aufgabe 5.3.6, 1 c) ---------------------

Treenode* searchBinTree_rec(Treenode* t, const keyType& x)
{
if (t==0) return 0;
else if (x==t->key) return t;
else if (x<t->key) return searchBinTree_rec(t->left,x);
else return searchBinTree_rec(t->right,x);
}

// --------------- Aufgabe 5.3.6, 1 d) ---------------------

bool ValueToKey_rec(keyType Key, dataType& Value)
{
Treenode* p=searchBinTree_rec(root,Key);
if (p==0)
  return false;
else
  {
    Value=p->data;
    return true;
  }
}

void __fastcall TForm1::KeyToValue_recClick(TObject *Sender)
{
dataType Value;
if (ValueToKey(Edit1->Text,Value))
  {
    Form1->Memo1->Lines->Add("key="+Edit1->Text+"  data: "+Value);
  }
else
  Form1->Memo1->Lines->Add("key="+Edit1->Text+"  nicht gefunden ");
}
```

```
// --------------- Aufgabe 5.3.6, 1 e) ---------------------

bool testBinTree_recursive()
{ // Einige Testfälle
bool result=true;
root=0;
dataType Value;
initTestUtils();

// 1. Suche im leeren Baum sollte funktionieren,
//    insbesondere nicht zu einem Programmabsturz führen
if (!assertEqual(ValueToKey_rec("1", Value),false,
  "leerer Baum")) result=false;

// 2. Ein Element einfügen und finden:
insertBinTreenode_rec(root,"1", "2");
if (!assertEqual(ValueToKey_rec("1", Value),true,
  "Baum mit '1'")) result=false;
if (!assertEqual(Value=="2",true,
  "Baum mit '1'")) result=false;

// 2. Zwei Elemente einfügen und finden:
insertBinTreenode_rec(root,"2", "3");
if (!assertEqual(ValueToKey_rec("1", Value),true,
  "Baum mit '1'")) result=false;
if (!assertEqual(Value=="2",true,
  "Baum mit '1'")) result=false;
if (!assertEqual(ValueToKey_rec("2", Value),true,
  "Baum mit '1' und '2'")) result=false;
if (!assertEqual(Value=="3",true,
  "Baum mit '1'")) result=false;
return result;
}

void __fastcall TForm1::testBintree_recClick(TObject *Sender)
{
initTestUtils();
if (testBinTree_recursive())
  Memo1->Lines->Add("All tests passed, n="+IntToStr(TestUtilsPassCount));
else
  Memo1->Lines->Add(IntToStr(TestUtilsFailCount)+" tests failed");
}
```

11.3.22 Aufgabe 3.12.12

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\TreeDefs.h

#ifndef TREEDEFS_H__
#define TREEDEFS_H__
// --------------- Aufgabe 3.12.12, 1 a) -------------------

// keyType und dataType müssen vor '#include TreeDefs.h' definiert
// werden, z.B.

struct Treenode {
  keyType key;         // die Schlüsselwerte
  dataType data;       // die Nutzdaten
  Treenode* left;
  Treenode* right;
};

Treenode* newTreenode(const keyType& key, const dataType& data,
                              Treenode* left, Treenode* right)
```

```
{ // gibt einen Zeiger auf einen neuen Knoten zurück
Treenode* tmp=new Treenode;
tmp->key = key;
tmp->data = data;
tmp->left = left;
tmp->right = right;
return tmp;
}


// --------------- Aufgabe 3.12.12, 1 b) -------------------

void insertBinTreenode(Treenode* root, const keyType& key,const dataType& data)
{
if (root==0) root=newTreenode(key,data,0,0);
else
  {
    Treenode* i=root;
    Treenode* p;
    while(i!=0)
      {
        p=i;
        if (key<i->key) i=i->left;
        else i=i->right;
      }
    if (key<p->key) p->left=newTreenode(key,data,0,0);
    else p->right=newTreenode(key,data,0,0);
  }
}

#endif
```

11.3.23 Aufgabe 3.12.13

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\12_13.cpp

// --------------- Aufgaben 3.12.13 ------------------------
// --------------- Aufgabe 1 -------------------------------
void ShowTempPath()
{
char lpBuffer[MAX_PATH];
DWORD result=GetTempPath(MAX_PATH,lpBuffer);
if (result == 0)
  Form1->Memo1->Lines->Add("Fehler");
else if (result > MAX_PATH)
  Form1->Memo1->Lines->Add("Puffer zu klein");
Form1->Memo1->Lines->Add(lpBuffer);
}


// --------------- Aufgabe 2 -------------------------------

void ShowDriveStrings()
{
const DWORD BufSize = 4*('Z'-'A'+1)+1;
char lpBuffer[BufSize];
const DWORD result=GetLogicalDriveStrings(BufSize,lpBuffer);
if (result == 0)
  Form1->Memo1->Lines->Add("Fehler");
else if (result > BufSize)
  Form1->Memo1->Lines->Add("Puffer zu klein");
// lpBuffer="c:\<null>d:\<null><null>
// position: 0        4        8
```

```
int i=0;
while ((*(lpBuffer+i)!='\0') && (i<BufSize))
  {
    Form1->Memo1->Lines->Add(lpBuffer+i);
    i=i+4;
  }

// oder z.B.
// for (LPTSTR p=lpBuffer; *p!='\0'; p=p+4)
//    Form1->Memo1->Lines->Add(p);
}
```

11.3.24 Aufgabe 3.12.14

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\12_14.cpp

// Diese Funktionen werden in PointerU.cpp aufgerufen
// --------------- Aufgaben 3.12.14 -------------------------
// --------------- Aufgabe 1 -------------------------------

#include <cstdio>
void Aufgabe_3_12_14_1()
{
char c='A';
// a)
// int n1=strlen(&c);
// Der Speicher ab der Adresse &c wird bis zum nächsten
// Nullterminator durchsucht. Das kann zu einer Zugriffsverletzung
// führen.

const char* s="yodel doodle doo";
// b)
int n2=strlen(strstr(s,"doo")); // n2==10

char* t;
// c)
// int n3=strlen(strstr("doo",s));
// Da s kein Teilstring von "doo" ist, gibt strstr 0 zurück.
// Der Zugriff von strlen auf die Adresse 0 führt aber zu einer
// Zugriffsverletzung.

// d)
strcpy(t,s);
// An der Adresse t ist kein Speicher reserviert.
}


// --------------- Aufgabe 2 -------------------------------

// a)
char* cloneString(const char* s)
{
char* p=new char[strlen(s)+1];
strcpy(p,s);
return p;
}


void Aufgabe_3_12_14_2()
{
char* t="abc";
char* s1=t;
char* s2=cloneString(t);
```

```
// s1 und t zeigen beide auf denselben Speicherbereich.
// s2 und t zeigen auf verschiedene Speicherbereiche.


// b)
// Nach dem Aufruf dieser Funktion darf man den Aufruf von delete nicht
// vergessen.
}
```

11.3.25 Aufgabe 3.12

```
// File: C:\Loesungen_CB2006\Kap_3\3.12\PointerU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "PointerU.h"
//---------------------------------------------------------------------
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------

#include "..\..\CppUtils\TestUtils.h"

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
#ifdef _DEBUG
bool test_All(); // Prototyp
initTestUtils();
testUtilsSummary(Memo1, test_All());
#endif
}


// --------------- Aufgaben 3.12.4 -------------------------

#include "12_4.cpp"

void __fastcall TForm1::Button_3_12_4_1Click(TObject *Sender)
{
Aufgabe_3_12_4_1();
}

void __fastcall TForm1::Button_3_12_4_2Click(TObject *Sender)
{
Aufgabe_3_12_4_2();
}

void __fastcall TForm1::Button_3_12_4_3Click(TObject *Sender)
{
Aufgabe_3_12_4_3();
}

void __fastcall TForm1::Button_3_12_4_4Click(TObject *Sender)
{
Aufgabe_3_12_4_4();
}

void __fastcall TForm1::Button_3_12_4_5Click(TObject *Sender)
{
Aufgabe_3_12_4_5();
}
```

```
// --------------- Aufgaben 3.12.6 -------------------------

#include "12_6.cpp"

void __fastcall TForm1::DynEratosthClick(TObject *Sender)
{
int n=StrToInt(Edit1->Text);
DynSiebEratosthenes(n);
}

void __fastcall TForm1::ReAllocClick(TObject *Sender)
{
test_ReAllocate_1();
test_ReAllocate_2();
}


// --------------- Aufgaben 3.12.7 -------------------------

#include "12_7.cpp"

void __fastcall TForm1::PointerArithmClick(TObject *Sender)
{
Aufgabe_3_12_7();
}


// --------------- Aufgaben 3.12.8 -------------------------

#include "12_8.cpp"

void __fastcall TForm1::ArrayParamClick(TObject *Sender)
{
Aufgabe_3_12_8_1();
}

void __fastcall TForm1::HornerClick(TObject *Sender)
{
test_HornerSinus();
}

void __fastcall TForm1::SizeofClick(TObject *Sender)
{
sizeof_Array();
}


// --------------- Aufgaben 3.12.10 ------------------------

#include "12_10.cpp"

void __fastcall TForm1::Button_3_12_10_1Click(TObject *Sender)
{
initTestUtils();
if (test_Leerzeichen())
  Form1->Memo1->Lines->Add("All tests passed");
else
  Form1->Memo1->Lines->Add(IntToStr(TestUtilsFailCount)+
    " Tests failed from "+ IntToStr(TestUtilsFailCount+TestUtilsPassCount) );
}

void __fastcall TForm1::Button_3_12_10_2Click(TObject *Sender)
{
Aufgabe_3_12_10_2();
}

void __fastcall TForm1::Button_3_12_10_3Click(TObject *Sender)
{
```

```
Aufgabe_3_12_10_3();
}

void __fastcall TForm1::Button_3_12_10_4Click(TObject *Sender)
{
Aufgabe_3_12_10_4();
}

void __fastcall TForm1::Button_3_12_10_5Click(TObject *Sender)
{
Aufgabe_3_12_10_5();
}

void __fastcall TForm1::Button_3_12_10_6Click(TObject *Sender)
{
Aufgabe_3_12_10_6();
}

void __fastcall TForm1::Button_3_12_10_7Click(TObject *Sender)
{
Aufgabe_3_12_10_7();
}


// --------------- Aufgaben 3.12.11 ------------------------
// Siehe Projekt LinkedList


// --------------- Aufgaben 3.12.12 ------------------------
// Siehe Projekt Tree



// --------------- Aufgaben 3.12.13 ------------------------

#include "12_13.cpp"

void __fastcall TForm1::Button_3_12_13_1Click(TObject *Sender)
{
ShowTempPath();
}

void __fastcall TForm1::Button_3_12_13_2Click(TObject *Sender)
{
ShowDriveStrings();
}


// --------------- Aufgaben 3.12.14 ------------------------

#include "12_14.cpp"

void __fastcall TForm1::Button_3_12_14_1Click(TObject *Sender)
{
Aufgabe_3_12_14_1();
}

void __fastcall TForm1::Button_3_12_14_2Click(TObject *Sender)
{
Aufgabe_3_12_14_2();
}


// --------------- Aufgaben 3.12.15 ------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
int* pi;
*pi=17;
```

```
}

void __fastcall TForm1::Button2Click(TObject *Sender)
{
int* pi=new int;
delete pi;
*pi=17;
}

void __fastcall TForm1::Button3Click(TObject *Sender)
{
int* pi=new int[17];
delete pi;
}


bool test_All()
{
bool result=true;
if (!test_Leerzeichen()) result=false;
if (!test_EditDistance()) result=false;
return result;
}
//---------------------------------------------------------------------
```

11.3.26 Aufgabe 3.13

```
// File: C:\Loesungen_CB2006\Kap_3\3.13\StringU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "StringU.h"
//---------------------------------------------------------------------
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------

#include "..\..\CppUtils\TestUtils.h"

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
#ifdef _DEBUG
bool test_All(); // Prototyp
initTestUtils();
testUtilsSummary(Memo1, test_All());
#endif
}



// --------------- Aufgaben 3.13 ---------------------------

// --------------- Aufgabe 1 -------------------------------

// Die folgenden Funktionen sind nur als Übung gedacht.
// Für praktische Anwendungen wird auf die zahlreichen Funktionen
// verwiesen, die man in des 'SysUtils' Unit findet.

// a)

struct Date_t {
  int day, month, year;
```

```
};
// Date instead of Date_t is ambigous (Sysutils::Date)

Date_t StringToDate(AnsiString date)
{
int p1=date.Pos('.');
AnsiString dstr  = date.SubString(1,p1-1);
date.Delete(1,p1);
int p2 = date.Pos('.');
AnsiString mstr  = date.SubString(1,p2-1);
AnsiString ystr  = date.SubString(p2+1,date.Length()-p2+1);
Date_t d;
d.day=StrToIntDef(dstr,-1);
d.month=StrToIntDef(mstr,-1);
d.year=StrToIntDef(ystr,-1);
if ((d.day==-1) || (d.month==-1) || (d.year==-1))
  {
    d.day=-1;
    d.month=-1;
    d.year=-1;
  }
return d;
}

bool assertEqual(Date_t f, Date_t s, AnsiString msg,
                                          TCustomMemo* Memo=Form1->Memo1)
{
return assertTrue((f.day==s.day) && (f.month==s.month) && (f.year==s.year),
                  msg, Memo);
}

bool test_one_date(AnsiString s, int day, int month, int year)
{
Date_t d1=StringToDate(s);
Date_t d2={day,month,year};
return assertEqual(d1,d2,"test_one_date: "+s);
}

bool test_StringToDate()
{
bool result = true;
Date_t t1;
// teste alle üblichen Kombinationen
// Jahr zweistellig, Tag und Monat einstellig und zweistellig
if (!test_one_date("1.2.07",1,2,7)) result=false;
if (!test_one_date("12.3.07",12,3,7)) result=false;
if (!test_one_date("1.12.07",1,12,7)) result=false;
if (!test_one_date("17.18.07",17,18,7)) result=false;

// Jahr vierstellig, Tag und Monat einstellig und zweistellig
if (!test_one_date("1.2.1920",1,2,1920)) result=false;
if (!test_one_date("12.3.1920",12,3,1920)) result=false;
if (!test_one_date("1.12.1920",1,12,1920)) result=false;
if (!test_one_date("17.18.1920",17,18,1920)) result=false;

// Teste unzulässige Formate. Das Programm sollte nicht abstürzen
//                            und -1 zurückgeben:
if (!test_one_date("",   -1,-1,-1)) result=false;
if (!test_one_date(".",  -1,-1,-1)) result=false;
if (!test_one_date("1",  -1,-1,-1)) result=false;
if (!test_one_date("1.", -1,-1,-1)) result=false;
if (!test_one_date("1.2",-1,-1,-1)) result=false;
if (!test_one_date("..", -1,-1,-1)) result=false;
if (!test_one_date("...",-1,-1,-1)) result=false;
if (!test_one_date("abc",-1,-1,-1)) result=false;
if (!test_one_date("1. Dezember ",-1,-1,-1)) result=false;
return result;
}
```

```
void __fastcall TForm1::Button_3_13_1Click(TObject *Sender)
{
Date_t d=StringToDate(Edit1->Text);
Form1->Memo1->Lines->Add(Edit1->Text+": Tag="+IntToStr(d.day)
          +", Monat="+IntToStr(d.month)+", Jahr="+IntToStr(d.year));
}


// --------------- Aufgabe 2 -------------------------------

// Die Lösung zu dieser Aufgabe 2 ist das Projekt tel


// --------------- Aufgabe 3 -------------------------------

void __fastcall TForm1::Button_3_13_3Click(TObject *Sender)
{
Memo1->Font->Name="Courier";
double K=100;
// Currency K=StrToCurr(Form1->Edit1->Text); // bzw. K=100
for (int p=5; p<=15; p++)
  {
    // Currency z = K*p/100;
    double z = K*p/100;
    AnsiString s=Format("K=%8.2f p=%2d%% z=%7.2f",OPENARRAY(TVarRec,(K,p,z)));
    Memo1->Lines->Add(s);
  };
}


// --------------- Aufgabe 4 -------------------------------

AnsiString uintToBinaryAnsiString(unsigned int n)
{
AnsiString s;
for (int i=1; i<=8*sizeof(n); i++)
  {
    if (n%2==1) s = "1"+s;
    else s = "0"+s;
    n = n/2;
    //  Mit den folgenden Anweisungen funktioniert das auch für negative int-
Argumente:
    // if (n&1==1) s = "1"+s;
    // else s = "0"+s;
    // n = n>>1;
  }
return s;
}

bool test_uintToBinaryAnsiString()
{
bool result = true;

if (!assertEqual(uintToBinaryAnsiString(0),
       "00000000000000000000000000000000","toBinAStr 0")) result=false;
if (!assertEqual(uintToBinaryAnsiString(1),
       "00000000000000000000000000000001", "toBinAStr 1")) result=false;
if (!assertEqual(uintToBinaryAnsiString(2),
       "00000000000000000000000000000010", "toBinAStr 2")) result=false;
if (!assertEqual(uintToBinaryAnsiString(3),
       "00000000000000000000000000000011", "toBinAStr 3")) result=false;
if (!assertEqual(uintToBinaryAnsiString(-1),
       "11111111111111111111111111111111", "toBinAStr -1")) result=false;
return result;
}

void __fastcall TForm1::Button_3_13_4Click(TObject *Sender)
{
for (int i=0; i<=8; i++)
```

```
  Memo1->Lines->Add(uintToBinaryAnsiString(i));
}


// --------------- Aufgabe 5 -------------------------------

int Max(int a, int b)
{
if (a>b) return a;
else return b;
}

AnsiString BigIntAdd(AnsiString a, AnsiString b)
{
AnsiString s;
int m=Max(a.Length(),b.Length());
int ue=0;
for (int i=1; i<=m; i++)
  {
    int ai,bi;
    if (i<=a.Length()) ai=a[a.Length()-i+1]-'0';
    else ai=0;
    if (i<=b.Length()) bi=b[b.Length()-i+1]-'0';
    else bi=0;
    int r=(ai+bi+ue)%10;
    ue=(ai+bi+ue)/10;
    AnsiString ziffer[10]={"0","1","2","3","4","5","6","7","8","9"};
    s=ziffer[r]+s;
  }
if (ue>0) s="1"+s;
return s;
}

bool test_BigIntAdd()
{
bool result=true;
if (!assertEqual(BigIntAdd("0", "0"),"0", "BigIntAdd: 0+0==0"))
  result=false;
if (!assertEqual(BigIntAdd("1", "1"),"2", "BigIntAdd: 1+1==2"))
  result=false;
if (!assertEqual(BigIntAdd("1", "2"),"3", "BigIntAdd: 1+2==3"))
  result=false;
if (!assertEqual(BigIntAdd("1", "9"),"10","BigIntAdd: 1+9==10"))
  result=false;
// Check random numbers:
for (int i=0; i<100; i++)
  {
    int op1=rand();
    int op2=rand();
    AnsiString op1s=IntToStr(op1);
    AnsiString op2s=IntToStr(op2);
    AnsiString s=IntToStr(op1+op2);
    if (!assertEqual(BigIntAdd(op1s,op2s),s,
      "BigIntAdd: "+op1s+"+"+op2s+"="+s)) result=false;
  }
return result;
}

void __fastcall TForm1::Button_3_13_5Click(TObject *Sender)
{
AnsiString b="1";
for (int i=1; i<=100; i++)
  {
    b=BigIntAdd(b,b);
    Memo1->Lines->Add(AnsiString("2^")+IntToStr(i)+"="+b);
  }
}
```

```
// --------------- Aufgabe 6 -------------------------------

AnsiString BigIntShiftLeft(AnsiString a)
{
return a+"0";
}

AnsiString BigIntShiftRight(AnsiString a)
{
return a.Delete(a.Length(),1);
}

bool BigIntNull(AnsiString a)
{ // setzt voraus, dass 0 nicht durch "00" o. ä. dargestellt wird
return (a=="0")||(a="");
}

int LastDigit(AnsiString a)
{ // ergibt die letzte Ziffer eines Strings als Zahl
return a[a.Length()]-'0';
}

AnsiString ziffer[10]={"0","1","2","3","4","5","6","7","8","9"};

AnsiString BigIntMultByDigit(AnsiString a, int n)
{ // Multipliziert die String-Zahl a mit der int-Zahl n
AnsiString s;
int ue=0;
for (int i=1; i<=a.Length(); i++)
  {
    int ai=a[a.Length()-i+1]-'0';
    int r=(ai*n + ue)%10;
    ue=(ai*n + ue)/10;
    s=ziffer[r]+s;
  }
if (ue>0) s=ziffer[ue]+s;
return s;
}

AnsiString BigIntMult(AnsiString u, AnsiString v)
{
AnsiString z=""; // Null
AnsiString x=u,y=v;
// assert(z+x*y==u*v);
while (!BigIntNull(x)) // (x != 0)
  {
    // if (x&1) z = z + y;
    z=BigIntAdd(z,BigIntMultByDigit(y,LastDigit(x)));
    y=BigIntShiftLeft(y);  // y = y*2;
    x=BigIntShiftRight(x); // x = x/2;
  }
// assert(z+x*y==u*v && x==0)
if (z=="") z="0";
return z;
}

AnsiString fakultaet(int n)
{
AnsiString result="1";
for (int i=2; i<=n; i++)
  result=BigIntMult(result,IntToStr(i));
return result;
}

double double_fakultaet(int n)
{
double result=1;
for (int i=2; i<=n; i++)
  result=result*i;
```

```
return result;
}

bool test_BigIntMult()
{
bool result=true;
if (!assertEqual(BigIntMult("0", "0"),"0", "BigIntMult: 0*0==0"))
  result=false;
if (!assertEqual(BigIntMult("1", "1"),"1", "BigIntMult: 1*1==2"))
  result=false;
if (!assertEqual(BigIntMult("1", "2"),"2", "BigIntMult: 1*2==2"))
  result=false;
if (!assertEqual(BigIntMult("11", "11"),"121","BigIntMult: 11*11==121"))
  result=false;
// Check random numbers:
for (int i=0; i<100; i++)
  {
    int op1=rand();
    int op2=rand();
    AnsiString op1s=IntToStr(op1);
    AnsiString op2s=IntToStr(op2);
    AnsiString s=IntToStr(op1*op2);
    if (!assertEqual(BigIntMult(op1s,op2s),s,
      "BigIntMult: "+op1s+"+"+op2s+"="+s)) result=false;
  }
return result;
}

void __fastcall TForm1::Button_3_13_6Click(TObject *Sender)
{
for (int i=1; i<=45; i++)
  {
    Memo1->Lines->Add(IntToStr(i)+"!="+fakultaet(i));
    Memo1->Lines->Add(IntToStr(i)+"!="+double_fakultaet(i));
  }
}

// --------------- Aufgabe 7 -------------------------------

bool test_All()
{
bool result=true;
if (!test_StringToDate()) result=false;
if (!test_uintToBinaryAnsiString()) result=false;
if (!test_BigIntAdd()) result=false;
if (!test_BigIntMult()) result=false;
return result;
}
```

11.3.27 Aufgabe 3.13.2

```
// File: C:\Loesungen_CB2006\Kap_3\3.13\TelU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "TelU.h"
//---------------------------------------------------------------------
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
```

```
{
}



// --------------- Aufgabe 4.1.2:  -------------------------


AnsiString TestDir="c:\\test";
AnsiString Filename=TestDir+"\\tel.txt";

void __fastcall TForm1::FormCreate(TObject *Sender)
{
// Diese Eigenschaften kann man auch im Objektinspektor setzen:
Form1->Caption="Telefonverzeichnis "+Filename;
Label1->Caption = "Suchergebnisse";
Search->Caption ="Suchen";
Save->Caption ="Speichern";


// Erzeuge das Verzeichnis, falls es nicht existiert
// damit die Datei auch beim ersten Start des Programms geladen
// werden kann.
if (CreateDirectory(TestDir.c_str(),0))
  ShowMessage("Verzeichnis '"+TestDir+"' angelegt");

EditMemo->Lines->Clear();
if (!FileExists(Filename)) // Erzeuge eine leere Telefonliste
  EditMemo->Lines->SaveToFile(Filename);

EditMemo->Lines->LoadFromFile(Filename);

SearchResultsMemo->Lines->Clear();
Edit1->Clear();
}

// b)
void __fastcall TForm1::SearchClick(TObject *Sender)
{
for (int i=0; i<=EditMemo->Lines->Count-1; i++)
  if (EditMemo->Lines->Strings[i].UpperCase().Pos(Edit1->Text.UpperCase()) > 0)
     SearchResultsMemo->Lines->Add(EditMemo->Lines->Strings[i]);
}

// c)
void __fastcall TForm1::SaveClick(TObject *Sender)
{
EditMemo->Lines->SaveToFile(Filename);
}

// d)
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
{
if (Key==13) SearchClick(Sender);
}
```

11.3.28 Aufgabe 3.14

```
// File: C:\Loesungen_CB2006\Kap_3\3.14\TypedU.cpp

#include <vcl.h>
#pragma hdrstop
```

```
#include "TypedU.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

// --------------- Aufgaben 3.14 ---------------------------

#include <typeinfo>
using namespace std;

void __fastcall TForm1::Button_3_14Click(TObject *Sender)
{
Memo1->Lines->Add(typeid(PBOOL).name());   // int*
Memo1->Lines->Add(typeid(LPLONG).name());  // long*
Memo1->Lines->Add(typeid(LPDWORD).name()); // unsigned long*
Memo1->Lines->Add(typeid(LPVOID).name());  // void*
Memo1->Lines->Add(typeid(LPCVOID).name()); // const void*
}
```

11.3.29 Aufgabe 3.15

```
// File: C:\Loesungen_CB2006\Kap_3\3.15\EnumU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "EnumU.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------

void showBorderstyle()
{
if (Form1->BorderStyle == bsDialog)
  Form1->Dialog->Checked=true;
else if (Form1->BorderStyle == bsSingle)
  Form1->Single->Checked=true;
else if (Form1->BorderStyle == bsSizeable)
  Form1->Sizeable->Checked=true;
else if (Form1->BorderStyle == bsNone)
  Form1->None->Checked=true;
else if (Form1->BorderStyle == bsToolWindow)
  Form1->ToolWindow->Checked=true;
else if (Form1->BorderStyle == bsSizeToolWin)
  Form1->SizeToolWin->Checked=true;
}
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
showBorderstyle();
}


// --------------- Aufgabe 1 ------------------------------

void __fastcall TForm1::Button_3_15_1Click(TObject *Sender)
{
```

```
// a) Ohne den Umlaut in "März" möglich
enum Monate {Januar=1, Februar, /* März, */ April, Mai, Juni,
     Juli,  August, September, Oktober, November, Dezember};
     // nicht möglich: mehrfache Deklarationen
// enum Sommer {Juni, Juli,  August};

// b)
// Nicht möglich, da die Enumeratoren Ganzzahlkonstanten sein müssen.
// enum Monate {Jan = "Januar", Februar = "Februar"};

// c)
// möglich
enum {max=100};
int a[max];
}


// --------------- Aufgabe 2 a) ----------------------------

void __fastcall TForm1::DialogClick(TObject *Sender)
{
Form1->BorderStyle = bsDialog;
showBorderstyle();
}

void __fastcall TForm1::SingleClick(TObject *Sender)
{
Form1->BorderStyle = bsSingle;
showBorderstyle();
}

void __fastcall TForm1::SizeableClick(TObject *Sender)
{
Form1->BorderStyle = bsSizeable;
showBorderstyle();
}

void __fastcall TForm1::NoneClick(TObject *Sender)
{
Form1->BorderStyle = bsNone;
showBorderstyle();
}

void __fastcall TForm1::ToolWindowClick(TObject *Sender)
{
Form1->BorderStyle = bsToolWindow;
showBorderstyle();
}

void __fastcall TForm1::SizeToolWinClick(TObject *Sender)
{
Form1->BorderStyle = bsSizeToolWin;
showBorderstyle();
}


// --------------- Aufgabe 2 b) ----------------------------

/* In der Online-Hilfe findet man:

      typedef TFormBorderStyle TBorderStyle;

   und

     enum   TFormBorderStyle   {   bsNone,   bsSingle,   bsSizeable,   bsDialog,
bsToolWindow, bsSizeToolWin };

   Damit ergibt sich die Lösung z. B. folgendermaßen:
*/
```

```
TFormBorderStyle bs=bsNone;

void __fastcall TForm1::Button_3_15_2Click(TObject *Sender)
{
if (bs<bsSizeToolWin) bs=bs+1;
else bs=bsNone;
Form1->BorderStyle = bs;
showBorderstyle();
}
```

11.3.30 Aufgabe 3.17

```
// File: C:\Loesungen_CB2006\Kap_3\3.17\MemU.cpp

#include <vcl.h>
#pragma hdrstop
#include "MemU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

// --------------- Aufgabe 3.17.2, 1 -----------------------

void verschachtelt()
{
int i,j,k=7;
AnsiString s,t;
  {
  char j;
    {
    float i=0,j=i;
      {
      AnsiString i,j,k;
      i = "ckt"; j = "ra"; k = "vert"; // i, j, k: AnsiString
      s = s+k+j+i+" ";                 // i, j, k, s: AnsiString
      }
    i = 1; j = 2;                      // i, j: int
    t = FloatToStr(i)+" + "+FloatToStr(j)+" = "+FloatToStr(k);//i,j,k: int
      {
      AnsiString t,j,k,i=s;
        {
        char t,j,k;
        t = 's'; j = 'i'; k = 't';       // t, j, k: char
        s = AnsiString(' ')+j+t+k+ ' ';  // s: AnsiString, t, j, k: char
        }
        {
        char t,j,k;
        t = 'a'; j = 'D'; k = 's';       // t, j, k: char
        s = AnsiString(j) + t + k+s;     // s: AnsiString, t, j, k: char
        }
      t = "nz sch"; j = "ja ga"; k = "ön "; // t, j, k: AnsiString
      s = s+j+t+k+i;                        // s, t, j, k, i: AnsiString
      }
    }
  }
Form1->Memo1->Lines->Add(t);  // t: AnsiString
```

```
Form1->Memo1->Lines->Add(s);  // s: AnsiString
}

void __fastcall TForm1::Button_3_17_2Click(TObject *Sender)
{
verschachtelt();
/* Erzeugt die Ausgabe:
     1 + 2 = 7
     Das ist ja ganz schön vertrackt
*/
}


// --------------- Aufgabe 3.17.4, 1 -----------------------

double f(double x, int n)
{
static int count =0;
count=count+n;
int r=1;
for (int i=0; i<count; i++) r=r*x;
return r;
}

void __fastcall TForm1::Button_3_17_4_1Click(TObject *Sender)
{
int x=f(3,1); // for (int i=0; i<1; ...)
int y=f(4,2); // for (int i=0; i<3; ...)
}


// --------------- Aufgabe 3.17.4, 2 -----------------------

int Fibonacci(int n)
{
const int max=50;
static int f[max];
static bool firstTime=true;
if (firstTime)
{ // Wird nur beim ersten Aufruf der Funktion ausgeführt.
  f[0]=0;
  f[1]=1;

  for (int i=2; i<max; i++)
    f[i]=f[i-1]+f[i-2];
  firstTime=false;
}
if (0<=n && n<max) return f[n];
else return -1;
}

void __fastcall TForm1::Button_3_17_4_2Click(TObject *Sender)
{
for (int i=0; i<50; i++)
  Memo1->Lines->Add(IntToStr(Fibonacci(i)));
}
```

11.3.31 Aufgabe 3.18

```
// File: C:\Loesungen_CB2006\Kap_3\3.18\RefParamU.cpp

#include <vcl.h>
#pragma hdrstop
```

```
#include "RefParamU.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}


// --------------- Aufgaben 3.18 ---------------------------

// --------------- Aufgabe a) ------------------------------

int Fibonacci(const int& n)
{
int f=0,x=0,y=1;
for (int i=0; i<n; i++)
  {
    x=y;
    y=f;
    f=x+y;
  }
return f;
}

void zeigeFibonacci(const int& n)
{
Form1->Memo1->Lines->Add("Fibonacci-Zahlen:");
for (int i=0; i<n; i++)
  Form1->Memo1->Lines->Add(IntToStr(i)+":  "+IntToStr(Fibonacci(i)) );
Form1->Memo1->Lines->Add("Falsche Ergebnisse wegen Überlauf für n>=47");
}


// --------------- Aufgabe b) ------------------------------

// #include <cstdlib> // bei manchen Versionen des C++Builders notwendig für rand
double RegentropfenPi(const int& n)
{
int Treffer=0;
for (int i = 0; i<n; i++)
  {
    double x = (0.0+rand())/RAND_MAX;
    double y = (0.0+rand())/RAND_MAX;
    if (x*x+y*y < 1) Treffer++;
  }
return 4.0*Treffer/n;
}

// --------------- Aufgabe c) ------------------------------

struct Date_t {
  int day, month, year;
};

Date_t StringToDate(const AnsiString& date)
{
int p1=date.Pos('.');
AnsiString dstr  = date.SubString(1,p1-1);
date.Delete(1,p1);
int p2 = date.Pos('.');
AnsiString mstr  = date.SubString(1,p2-1);
AnsiString ystr  = date.SubString(p2+1,date.Length()-p2+1);
Date_t d;
d.day=StrToIntDef(dstr,-1);
```

```
d.month=StrToIntDef(mstr,-1);
d.year=StrToIntDef(ystr,-1);
if ((d.day==-1) || (d.month==-1) || (d.year==-1))
  {
    d.day=-1;
    d.month=-1;
    d.year=-1;
  }
return d;
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
zeigeFibonacci(10);
Memo1->Lines->Add(FloatToStr(RegentropfenPi(10000)));
}
```

11.3.32 Aufgabe 3.19

```
// File: C:\Loesungen_CB2006\Kap_3\3.19\StatemU.cpp

#include <vcl.h>
#pragma hdrstop

#include "StatemU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
Edit1->Text="50";
}

#include <stdexcept>
using namespace std;

// --------------- Aufgaben 3.18.2 -------------------------

// --------------- Aufgabe 1 -------------------------------
int Fibonacci(int n)
{
if (n>47) throw logic_error("Fibonacci: invalid argument n>47");
int f=0,x=0,y=1;
for (int i=0; i<n; i++)
  {
    x=y;
    y=f;
    f=x+y;
  }
return f;
}

void __fastcall TForm1::Button_3_18_2_1aClick(TObject *Sender)
{
int n=StrToInt(Edit1->Text);
int y=Fibonacci(n);
Form1->Memo1->Lines->Add("Fibonacci("+Edit1->Text+")="+IntToStr(y));
}

void __fastcall TForm1::Button_3_18_2_1bClick(TObject *Sender)
```

```
{
try {
      int n=StrToInt(Edit1->Text);
      int y=Fibonacci(n);
      Form1->Memo1->Lines->Add("Fibonacci("+Edit1->Text+")="+IntToStr(y));
    }
catch(exception& e)
    {
      Form1->Memo1->Lines->Add(e.what());
    }
}


// --------------- Aufgabe 2 -------------------------------

int Div10By(int n)
{
return 10/n;
}

int AccessViolation()
{
int* p=0;
*p=17; // Zugriffsverletzung
return *p;
}

void __fastcall TForm1::Button_3_18_2_2aClick(TObject *Sender)
{
int n=Div10By(0);
Form1->Memo1->Lines->Add("Div by 0: result="+IntToStr(n));
}

void __fastcall TForm1::Button_3_18_2_2bClick(TObject *Sender)
{
try {
      int n=Div10By(0);
      Form1->Memo1->Lines->Add("Div by 0: result="+IntToStr(n));
    }
catch(Exception& e)
    {
      Form1->Memo1->Lines->Add(e.Message);
    }
}

void __fastcall TForm1::Button_3_18_2_2cClick(TObject *Sender)
{
int n=AccessViolation();
Form1->Memo1->Lines->Add("Div by 0: result="+IntToStr(n));
}

void __fastcall TForm1::Button_3_18_2_2dClick(TObject *Sender)
{
try {
      int n=AccessViolation();
      Form1->Memo1->Lines->Add("Div by 0: result="+IntToStr(n));
    }
catch(Exception& e)
    {
      Form1->Memo1->Lines->Add(e.Message);
    }
}


// --------------- Aufgaben 3.18.3 -------------------------

// --------------- Aufgabe 1 -------------------------------

int LA_Summe,LB_Summe,MA_Summe,MB_Summe,MC_Summe,Summe;
```

```
void LagMatSwitch(char Lagergruppe, char Materialgruppe)
{
switch(Lagergruppe)
  {
    case 'A':
      switch(Materialgruppe)
        {
          case 'A': LA_Summe = LA_Summe + Summe;
                    MA_Summe = MA_Summe + Summe;
                    break;
          case 'B': LA_Summe = LA_Summe + Summe;
                    MB_Summe = MB_Summe + Summe;
                    break;
          case 'C': LA_Summe = LA_Summe + Summe;
                    MC_Summe = MC_Summe + Summe;
                    break;
          default:  ShowMessage("Unzulässige Materialgruppe in Lager A");
        }
    case 'B':
      switch(Materialgruppe)
        {
          case 'A': LB_Summe = LB_Summe + Summe;
                    MA_Summe = MA_Summe + Summe;
                    break;
          case 'B': LB_Summe = LB_Summe + Summe;
                    MB_Summe = MB_Summe + Summe;
                    break;
          default:  ShowMessage("Unzulässige Materialgruppe in Lager B");
        }
    default: ShowMessage("Unzulässige Lagergruppe");
  }
}

void __fastcall TForm1::Button_3_18_3_1Click(TObject *Sender)
{
char Lagergruppe, Materialgruppe;
LagMatSwitch(Lagergruppe, Materialgruppe);
}


// --------------- Aufgabe 2 -------------------------------

void __fastcall TForm1::Button_3_18_3_2Click(TObject *Sender)
{
// Da die Bedingungen nicht durch eine Prüfung auf Gleichheit mit einer
// Konstanten gebildet werden, ist eine Lösung mit switch nicht möglich.
}


// --------------- Aufgabe 3 -------------------------------

void __fastcall TForm1::Button_3_18_3_3Click(TObject *Sender)
{
// Da die Bedingungen nicht durch eine Prüfung auf Gleichheit mit einer
// Konstanten gebildet werden, ist eine Lösung mit switch nicht möglich.
}
```

11.3.33 Aufgabe 3.20

```
// File: C:\Loesungen_CB2006\Kap_3\3.20\ExprU.cpp

#include <vcl.h>
```

```
#pragma hdrstop
#include "ExprU.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------


AnsiString TestDir="c:\\test";

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
// Erzeuge das Verzeichnis, falls es nicht existiert
if (CreateDirectory(TestDir.c_str(),0))
  ShowMessage("directory '"+TestDir+"' created");
}


// --------------- Aufgabe 1. ------------------------------

#include <typeinfo>

int i=0;

void Aufgabe1()
{
using namespace std;
int i=2,j=i,k=-6, m=8, n=10000, x=0;
unsigned u=40000u,v=u;

// Aufgaben
int a=k/i/j+n/j; // == ((k/i)/j)) + (n/j) == -3/2 + 5000 == 4999
Form1->Memo1->Lines->Add(a);
Form1->Memo1->Lines->Add(typeid(k/i/j+n/j).name()); // int

int b= k/i/j++ + m/j++;
// Das Ergebnis ist undefiniert, da es davon abhängt, ob j++ zuerst im
// ersten oder im zweiten Teilausdruck erhöht wird
Form1->Memo1->Lines->Add(b);
Form1->Memo1->Lines->Add(typeid(k/i/j++ + m/j++).name()); // int

int c=i*4%-k*3; // == ((i*4)%(-k))*3 == (8%6)*3=6
// Alle Operatoren sind multiplikative Operatoren und werden
// linksassoziativ ausgewertet:
Form1->Memo1->Lines->Add(c);
Form1->Memo1->Lines->Add(typeid(i*4%-k*3).name()); // int

int d=k/-m-2; // == (k/-m)-2 == 0-2 == -2
Form1->Memo1->Lines->Add(d);
Form1->Memo1->Lines->Add(typeid(k/-m-2).name()); // int

double e= m*n/u/v; // == ((m*n)/u)/v == 2/40000 == 0
// Obwohl die linke Seite den Datentyp double hat, wird der Ausdruck als
// Ganzzahlwert ausgewertet.
Form1->Memo1->Lines->Add(e);
Form1->Memo1->Lines->Add(typeid(m*n/u/v).name()); // unsigned int

int f= ++::i%++i; // == (++::i)%(++i) == 1%3 == 1
Form1->Memo1->Lines->Add(f);
Form1->Memo1->Lines->Add(typeid(++::i%++i).name()); // int

int g = sizeof 17-2; // == (sizeof 17)-2 == 4-2 == 2
Form1->Memo1->Lines->Add(g);
Form1->Memo1->Lines->Add(typeid(sizeof 17-2).name()); // unsigned int
}

void __fastcall TForm1::Button_3_20_19_1Click(TObject *Sender)
```

```
{
Aufgabe1();
}


// --------------- Aufgabe 2. ------------------------------

void __fastcall TForm1::Button_3_20_19_2Click(TObject *Sender)
{
for (int i=0; i<8*sizeof(int); i++)
  Memo1->Lines->Add(IntToStr(i)+": "+IntToStr(1<<i));
    // mit "+" werden AnsiStrings "zusammengeklebt"
// Diese Funktion gibt die Zweierpotenzen aus
}


// --------------- Aufgabe 3. ------------------------------


#include <cstdio>

void __fastcall TForm1::Button_3_20_19_3Click(TObject *Sender)
{
double x = 3.14;
int a=1,b=2;

// a)
int s = a?b:x; // s=2
Form1->Memo1->Lines->Add(typeid(a?b:x).name()); // double

// b)
{
int s = (x > 0) ? 1 : (x < 0) ? -1 : 0;
  // == (x > 0) ? 1 : ((x < 0) ? -1 : 0);
  // Diese Anweisung ist gleichwertig zu
if (x>0) s=1;
else if (x<0) s=-1;
else s=0;
}

// c)
for (int n=0;n<4;n++)
  {
   char s[100];
   std::sprintf(s,"%d file%c found",n,(n==1)?' ':'s');
   Form1->Memo1->Lines->Add(s);
  }
}


// --------------- Aufgabe 4. ------------------------------

void __fastcall TForm1::Button_3_20_19_4Click(TObject *Sender)
{
int    i,j,k;
i=3;j=1;k=2;    k=i+++j;    // k=(i++)+j=4, i=4
Form1->Memo1->Lines->Add("i="+IntToStr(i)+" j="+IntToStr(j)+" k="+IntToStr(k));
i=3;j=1;k=2;    k= ++i+j++; // k=(++i)+(j++)=5, i=4, j=2
Form1->Memo1->Lines->Add("i="+IntToStr(i)+" j="+IntToStr(j)+" k="+IntToStr(k));
i=3;j=1;k=2;    k= -i+++j;  // k=(-(i++))+j=-2, i=4,
Form1->Memo1->Lines->Add("i="+IntToStr(i)+" j="+IntToStr(j)+" k="+IntToStr(k));
i=3;j=1;k=2;    k= +i+-j;   // k=(+i)+(-j)=2
Form1->Memo1->Lines->Add("i="+IntToStr(i)+" j="+IntToStr(j)+" k="+IntToStr(k));
i=3;j=1;k=2;    k= i--+--j; // k=(i--)+(--j)=3, i=2, j=0
Form1->Memo1->Lines->Add("i="+IntToStr(i)+" j="+IntToStr(j)+" k="+IntToStr(k));
}


// --------------- Aufgabe 5. ------------------------------
```

```
// a)
void SetROH(const char* FileName)
{
SetFileAttributes(FileName,FILE_ATTRIBUTE_READONLY|FILE_ATTRIBUTE_HIDDEN);
}

// b)
bool isSystemFile(const char* FileName)
{
DWORD attrib=GetFileAttributes(FileName);
if (attrib!= 0xFFFFFFFF)
  return attrib&FILE_ATTRIBUTE_SYSTEM;
else return false; // z.B. falls die Datei nicht existiert
}

// c)
bool SetFileAttributeToHidden(const char* FileName)
{
DWORD attrib=GetFileAttributes(FileName);
if (attrib!= 0xFFFFFFFF)
  return SetFileAttributes(FileName,attrib|FILE_ATTRIBUTE_HIDDEN);
else return false;
}

// d)
bool RemoveFileAttributeHidden(const char* FileName)
{
DWORD attrib=GetFileAttributes(FileName);
if (attrib!= 0xFFFFFFFF)
  return SetFileAttributes(FileName,attrib&(~FILE_ATTRIBUTE_HIDDEN));
else return false;
}

void __fastcall TForm1::Button_3_20_19_5Click(TObject *Sender)
{
// Vor dem Aufruf dieser Funktion müssen die folgenden Dateien zuerst
// angelegt werden:
AnsiString FileName1=TestDir+"\\test1.dat";
AnsiString FileName2=TestDir+"\\test2.dat";
AnsiString FileName3=TestDir+"\\test3.dat";
AnsiString FileName4=TestDir+"\\test4.dat";

SetROH(FileName1.c_str());

if (isSystemFile(FileName2.c_str()))
  Form1->Memo1->Lines->Add(FileName2+" ist Systemdatei");
else
  Form1->Memo1->Lines->Add(FileName2+" ist keine Systemdatei");

if (SetFileAttributeToHidden(FileName3.c_str()))
  Form1->Memo1->Lines->Add(AnsiString(FileName3)+
                                    " Attribut Hidden gesetzt");
else
  Form1->Memo1->Lines->Add(AnsiString(FileName3)+
                                    " Attribut Hidden nicht gesetzt");

if (RemoveFileAttributeHidden(FileName4.c_str()))
  Form1->Memo1->Lines->Add(AnsiString(FileName4)+
                                    " Attribut Hidden gelöscht");
else
  Form1->Memo1->Lines->Add(AnsiString(FileName4)+
                                    " Attribut Hidden nicht gelöscht");
}
```

11.3.34 Aufgabe 3.21

```
// File: C:\Loesungen_CB2006\Kap_3\3.21\NamespU.cpp

#include <vcl.h>
#pragma hdrstop

#include "NamespU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------


// --------------- Aufgaben 3.21 ---------------------------

// --------------- Aufgabe 1. ------------------------------

namespace N {

// a) b) c)
  int f(const int& n) // Fibonacci
  {
  int fib=0,x=0,y=1;
  for (int i=0; i<n; i++)
    {
      x=y;
      y=fib;
      fib=x+y;
    }
  return fib;
  }

// d)
  namespace N1 {

     int g(int x);

  }
} // end of namespace N

// d)
int N::N1::g(int x)
{
  return x+1;
}

void useNamespace_a_b()
{
// a)
int a=N::f(10);
// b)
using N::f;
int b=f(11);
}

void useNamespace_c()
{
// c)
using namespace N;
int c=N::f(10);
```

```
}

void useNamespace_d1()
{
// a)
int a=N::N1::g(10);
// b)
using N::N1::g;
int b=g(11);
}

void useNamespace_d2()
{
// c)
using namespace N;
int c=N::f(10);
}

void useNamespace_e()
{
// e)
namespace X=N::N1; // alias
// a)
int a=X::g(10);
// b)
using X::g;
int b=g(11);
}


// --------------- Aufgabe 2. ------------------------------

void f(){};
void g(){};

namespace A {
  void f(){};
  void g(){};
}

namespace B {
  using ::f;  // a)
  using A::g; // b)
}

void h()
{
using namespace B;
f();          // c) ::f()
B::g();       // d) A::g()
using A::f;
f();          // e) A::f()
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
useNamespace_a_b();
useNamespace_c();
useNamespace_d1();
useNamespace_d2();
useNamespace_e();
h();
}
```

11.3.35 Aufgabe 3.22

```
// File: C:\Loesungen_CB2006\Kap_3\3.22\PreprU.cpp

#include <vcl.h>
#pragma hdrstop
#include "PreprU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

// --------------- Aufgabe 3.22 a) - e) --------------------

#define NOTRACE
// Die Anweisungen für die TRACE Makros werden erzeugt, wenn NOTRACE
// definiert ist, bevor Trace.cpp eingebunden wird:

// Die Lösung der Aufgaben a) bis e) ist in dieser Datei:
#include "..\..\CppUtils\Trace.h"


const char* LogfileName="c:\\trace.log";
OPEN_TRACELOG(LogfileName);


// --------------- Aufgabe 3.22 e) -------------------------

int Quersumme(int n)
{
TRACE1(n);
if (n<0) n=-n; // berücksichtige negative n
// % ist für negative Operanden compilerabhängig
int s=0;
while (n>0)
  {
    s = s+n%10;
    n = n/10;
    TRACE2(s,n);
  }
return s;
}

int Fibonacci(int n)
{
TRACE1(n);
int f=0,x=0,y=1;
for (int i=0; i<n; i++)
  {
    x=y;
    y=f;
    f=x+y;
    TRACE3(x,y,f);
  }
return f;
}

void test()
{
Quersumme(123);
Fibonacci(10);
/* Der Inhalt der Datei c:\trace.log
```

```
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 54, function: Quersumme, n='123'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 62, function: Quersumme, s='3' n='12'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 62, function: Quersumme, s='5' n='1'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 62, function: Quersumme, s='6' n='0'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 72, function: Fibonacci, n='10'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 79, function: Fibonacci, x='1' y='0'
f='1'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 79, function: Fibonacci, x='0' y='1'
f='1'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 79, function: Fibonacci, x='1' y='1'
f='2'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 79, function: Fibonacci, x='1' y='2'
f='3'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 79, function: Fibonacci, x='2' y='3'
f='5'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 79, function: Fibonacci, x='3' y='5'
f='8'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 79, function: Fibonacci, x='5' y='8'
f='13'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 79, function: Fibonacci, x='8' y='13'
f='21'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 79, function: Fibonacci, x='13' y='21'
f='34'
C:\loesungen\Kap_3\3.15\PreprU.cpp, line: 79, function: Fibonacci, x='21' y='34'
f='55'
*/
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
#ifndef NOTRACE
Memo1->Lines->Add("Logfile: "+AnsiString(LogfileName));
#endif
test();
}


// --------------- Aufgabe 3.22 f) -------------------------

enum {TraceLevel_no=0, TraceLevel_summary=50, TraceLevel_detailled=100};
int TraceLevel=TraceLevel_detailled;

#include <fstream>
using namespace std;
const char* LogfileName1="c:\\trace1.log";

ofstream traceout(LogfileName1);

int Quersumme1(int n)
{
if (TraceLevel>=TraceLevel_summary)
  traceout<<"Quersumme1: n="<<n<<std::endl;
if (n<0) n=-n;
int s=0;
while (n>0)
  {
    s = s+n%10;
    n = n/10;
    if (TraceLevel>=TraceLevel_detailled)
      traceout<<"Quersumme1: s="<<s<<" n="<<n<<std::endl;
  }
return s;
}

int Fibonacci1(int n)
{
if (TraceLevel>=TraceLevel_summary)
  traceout<<"Fibonacci: n="<<n<<std::endl;
int f=0,x=0,y=1;
```

```
for (int i=0; i<n; i++)
  {
    x=y;
    y=f;
    f=x+y;
    if (TraceLevel>=TraceLevel_detailled)
      traceout<<"Fibonacci: x="<<x<<" y="<<y<<" f="<<f<<std::endl;
  }
return f;
}

void test1()
{
Quersumme1(123);
Fibonacci1(10);
}

void __fastcall TForm1::Button2Click(TObject *Sender)
{
Memo1->Lines->Add("Logfile1: "+AnsiString(LogfileName1));
test1();
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
rbTraceLevel_detailled->Checked=true;
}

void __fastcall TForm1::rbTraceLevel_noClick(TObject *Sender)
{
TraceLevel=TraceLevel_no;
}

void __fastcall TForm1::rbTraceLevel_summaryClick(TObject *Sender)
{
TraceLevel=TraceLevel_summary;
}

void __fastcall TForm1::rbTraceLevel_detailledClick(TObject *Sender)
{
TraceLevel=TraceLevel_detailled;
}
```

11.3.36 Aufgabe 3.23 - Header-Datei

```
// File: C:\Loesungen_CB2006\Kap_3\3.23\MyLib.h

#ifndef MyLibH
#define MyLibH
// --------------- Aufgabe 1 -------------------------------


int Quersumme(int n);
double RegentropfenPi(int n);
// Klassendefinitionen kommen in die Header-Datei

struct Date_t {
  int day, month, year;
};

#include <vcl.h>  // für AnsiString notwendig

Date_t StringToDate(AnsiString date);
```

```
// In verschiedenen Dateien gemeinsam genutzte Variable
// werden in der C++-Datei definiert und in der Header-Datei
// mit extern deklariert.
// Eine Initialisierung kann nur in der C++-Datei stattfinden,
// da eine Deklaration mit extern und einer Initialisierung
// eine Definition ist.
extern int GemeinsameVariable;
extern const int Max;
extern int Array[];

#endif // #ifndef MyLibH
```

11.3.37 Aufgabe 3.23 - C++-Datei

```
// File: C:\Loesungen_CB2006\Kap_3\3.23\MyLib.cpp

#pragma hdrstop

#include "MyLib.h"

//---------------------------------------------------------------------

#pragma package(smart_init)


// --------------- Aufgaben 3.4.6 --------------------------


// --------------- Aufgabe 1 -------------------------------

int Quersumme(int n)
{
if (n<0) n=-n; // berücksichtige negative n
// % ist für negative Operanden compilerabhängig
int s=0;
while (n>0)
  {
    s = s+n%10;
    n = n/10;
  }
return s;
}


#include <cstdlib> // bei manchen Versionen des C++Builders notwendig für rand
double RegentropfenPi(int n)
{
using std::rand;
int Treffer=0;
for (int i = 0; i<n; i++)
  {
    double x = (0.0+rand())/RAND_MAX;
    double y = (0.0+rand())/RAND_MAX;
    if (x*x+y*y < 1) Treffer++;
  }
return 4.0*Treffer/n;
}

/*
// Klassendefinitionen kommen in die Header-Datei
struct Date_t {
  int day, month, year;
```

```
};
*/

Date_t StringToDate(AnsiString date)
{
int p1=date.Pos('.');
AnsiString dstr  = date.SubString(1,p1-1);
date.Delete(1,p1);
int p2 = date.Pos('.');
AnsiString mstr  = date.SubString(1,p2-1);
AnsiString ystr  = date.SubString(p2+1,date.Length()-p2+1);
Date_t d;
d.day=StrToIntDef(dstr,-1);
d.month=StrToIntDef(mstr,-1);
d.year=StrToIntDef(ystr,-1);
if ((d.day==-1) || (d.month==-1) || (d.year==-1))
  {
    d.day=-1;
    d.month=-1;
    d.year=-1;
  }
return d;
}

// In verschiedenen Dateien gemeinsam genutzte Variable
// werden in der C++-Datei definiert und in der Header-Datei
// mit extern deklariert.
int GemeinsameVariable=0;
const int Max=100;
int Array[Max];
```

11.3.38 Aufgabe 3.23

```
// File: C:\Loesungen_CB2006\Kap_3\3.23\Unit1.cpp

#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

/*
1. Mit Datei|Neu|Unit eine C++-Datei und eine Header-Datei anlegen
   und diese unter dem Namen MyLib speichern. Dadurch werden die beiden
   Dateien MyLib.cpp und MyLib.h angelegt.

*/


#include "MyLib.h"

// Mehrfach #includes der Header-Datei sollen möglich sein:
#include "MyLib.h"

void zeigeQuerSummen(int a, int b)
{
```

```
Form1->Memo1->Lines->Add("Quersummen");
for (int i=a; i<=b; i++)
  Form1->Memo1->Lines->Add(IntToStr(i)+": "+
                           IntToStr(Quersumme(i)));
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
zeigeQuerSummen(1,StrToInt(Edit1->Text));
Memo1->Lines->Add(FloatToStr(RegentropfenPi(10000)));
Date_t d1=StringToDate("17.12.2006");
}
```

11.3.39 Aufgabe 3.23

```
// File: C:\Loesungen_CB2006\Kap_3\3.23\Unit2.cpp

#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
    : TForm(Owner)
{
}
// --------------- Aufgabe 1 b) ----------------------------

#include "MyLib.h"
double x=RegentropfenPi(100);
```

11.3.40 Aufgabe 3.24 - Dll

```
// File: C:\Loesungen_CB2006\Kap_3\3.24\MyDllU.cpp

#include <vcl.h>
#include <windows.h>
#pragma hdrstop
//---------------------------------------------------------------------
//   Wichtiger Hinweis zur DLL-Speicherverwaltung, falls die DLL die
//   statische Version der Laufzeitbibliothek (RTL) verwendet:
//
//   Wenn die DLL Funktionen exportiert, die String-Objekte (oder
//   Strukturen/Klassen, die verschachtelte Strings enthalten) als
//   Parameter oder Funktionsergebnisse übergibt, muss die Bibliothek
//   MEMMGR.LIB im DLL-Projekt und anderen Projekten, die die DLL
//   verwenden, vorhanden sein. Sie benötigen MEMMGR.LIB auch dann,
//   wenn andere Projekte, die die DLL verwenden, new- oder delete-
//   Operationen auf Klassen anwenden, die nicht von TObject abgeleitet
//   sind und die aus der DLL exportiert werden. Durch das Hinzufügen von
//   MEMMGR.LIB wird die DLL und deren aufrufende EXEs angewiesen,
//   BORLNDMM.DLL als Speicherverwaltung zu benutzen. In diesem Fall
//   sollte die Datei BORLNDMM.DLL zusammen mit der DLL weitergegeben werden.
```

```
//
//   Um die Verwendung von BORLNDMM.DLL, zu vermeiden, sollten String-
//   Informationen als "char *" oder ShortString-Parameter weitergegeben werden.
//
//   Falls die DLL die dynamische Version der RTL verwendet, müssen Sie
//   MEMMGR.LIB nicht explizit angeben.
//---------------------------------------------------------------------

#pragma argsused
int WINAPI DllEntryPoint(HINSTANCE hinst, unsigned long reason, void* lpReserved)
{
    return 1;
}
//---------------------------------------------------------------------

// --------------- Aufgaben 3.4.6 --------------------------


// --------------- Aufgabe 1 -------------------------------

__declspec(dllexport) int commonVariable=0;

__declspec(dllexport) int Quersumme(int n)
{
commonVariable++;
if (n<0) n=-n; // berücksichtige negative n
// % ist für negative Operanden compilerabhängig
int s=0;
while (n>0)
  {
    s = s+n%10;
    n = n/10;
  }
return s;
}


// #include <cstdlib> // bei manchen Versionen des C++Builders notwendig für rand
__declspec(dllexport) double RegentropfenPi(int n)
{
commonVariable++;
int Treffer=0;
for (int i = 0; i<n; i++)
  {
    double x = (0.0+rand())/RAND_MAX;
    double y = (0.0+rand())/RAND_MAX;
    if (x*x+y*y < 1) Treffer++;
  }
return 4.0*Treffer/n;
}


// --------------- Aufgaben 3.13 ---------------------------

// --------------- Aufgabe 1 -------------------------------

// Die folgenden Funktionen sind nur als Übung gedacht.
// Für praktische Anwendungen wird auf die zahlreichen Funktionen
// des SysUtils Unit verwiesen.

// a)

struct __declspec(dllexport) Date_t {
  int day, month, year;
};



__declspec(dllexport) Date_t StringToDate(AnsiString date)
```

```
{
int p1=date.Pos('.');
AnsiString dstr  = date.SubString(1,p1-1);
date.Delete(1,p1);
int p2 = date.Pos('.');
AnsiString mstr  = date.SubString(1,p2-1);
AnsiString ystr  = date.SubString(p2+1,date.Length()-p2+1);
Date_t d;
d.day=StrToIntDef(dstr,-1);
d.month=StrToIntDef(mstr,-1);
d.year=StrToIntDef(ystr,-1);
if ((d.day==-1) || (d.month==-1) || (d.year==-1))
  {
    d.day=-1;
    d.month=-1;
    d.year=-1;
  }
return d;
}
```

11.3.41 Aufgabe 3.24 - Dll

```
// File: C:\Loesungen_CB2006\Kap_3\3.24\UseDllImplicitU.cpp

#include <vcl.h>
#pragma hdrstop

#include "UseDllImplicitU.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------



_declspec(dllimport) int Quersumme(int n);

void zeigeQuerSummen(int a, int b)
{
Form1->Memo1->Lines->Add("Quersummen");
for (int i=a; i<=b; i++)
  Form1->Memo1->Lines->Add(IntToStr(i)+": "+
                           IntToStr(Quersumme(i)));
}

// #include <cstdlib> // bei manchen Versionen des C++Builders notwendig für rand
_declspec(dllimport) double RegentropfenPi(int n);

struct _declspec(dllimport) Date_t {
  int day, month, year;
};

_declspec(dllimport) Date_t StringToDate(AnsiString date);
__declspec(dllimport) int commonVariable;
__declspec(dllimport) const int commonConstant;


void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{
zeigeQuerSummen(1,StrToInt(Edit1->Text));
Memo1->Lines->Add(FloatToStr(RegentropfenPi(10000)));
Date_t d=StringToDate("");
Memo1->Lines->Add("commonVariable="+IntToStr(commonVariable));


}
//---------------------------------------------------------------------
```

11.4 Lösungen Kapitel 4

11.4.1 Aufgaben 4.1

```
// File: C:\Loesungen_CB2006\Kap_4\4.1\StringsU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "StringsU.h"
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------

#include "..\..\CppUtils\TestUtils.h"

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
#ifdef _DEBUG
initTestUtils();
bool test_StringAll(); // Prototyp
testUtilsSummary(Memo1, test_StringAll());
#endif
}

#include <string>
using std::string; // or: using namespace std;


// --------------- Aufgabe 4.1, 1. -------------------------

// siehe

#include "..\..\CppUtils\StringUt.h"

void __fastcall TForm1::stringToInt_Click(TObject *Sender)
{
string s=Edit1->Text.c_str();
bool success;
int x=stringToInt(s,success);
Memo1->Lines->Add(IntToStr(x)+" success="+BoolToStr(success,true));
}

void __fastcall TForm1::stringToDouble_Click(TObject *Sender)
{
string s=Edit1->Text.c_str();
bool success;
double x=stringToDouble(s,success);
Memo1->Lines->Add(FloatToStr(x)+" success="+BoolToStr(success,true));
```

```
}

void __fastcall TForm1::tostring_intClick(TObject *Sender)
{
int x=StrToInt(Edit1->Text);
string s=tostring(x);
Memo1->Lines->Add(s.c_str());
}

void __fastcall TForm1::tostring_doubleClick(TObject *Sender)
{
double x=StrToFloat(Edit1->Text);
string s=tostring(x);
Memo1->Lines->Add(s.c_str());
}


// --------------- Aufgabe 4.1, 2. -------------------------

// a)
struct Date_t {
  int day, month, year;
};

Date_t stringToDate(const string& date)
{
Date_t result;
bool success=true;
string::size_type p1 = date.find(".");
if (p1!=string::npos)
  {
    string dstr = date.substr(0,p1);
    string::size_type p2  = date.find(".",p1+1);
    if (p2!=string::npos)
      {
        string mstr = date.substr(p1+1,p2-p1-1);
        string ystr = date.substr(p2+1,date.length()-p1);

        bool succ;
        result.day   = stringToInt(dstr,succ);
        if (!succ) success=false;
        result.month = stringToInt(mstr,succ);
        if (!succ) success=false;
        result.year  = stringToInt(ystr,succ);
        if (!succ) success=false;
      }
    else success=false;
  }
else success=false;
if (!success)
  {
     Date_t failed={-1,-1,-1};
     result=failed;
  }
return result;
}

// b)
bool assertEqual(Date_t f, Date_t s, AnsiString msg,
                                          TCustomMemo* Memo=Form1->Memo1)
{
return assertTrue((f.day==s.day) && (f.month==s.month) && (f.year==s.year),
                  msg, Memo);
}

bool test_one_date(string s, int day, int month, int year)
{
Date_t d1=stringToDate(s);
Date_t d2={day,month,year};
```

```
return assertEqual(d1,d2,"test_one_date: "+AnsiString(s.c_str()));
}

bool test_StringToDate()
{
bool result = true;
// teste alle üblichen Kombinationen
// Jahr zweistellig, Tag und Monat einstellig und zweistellig
if (!test_one_date("1.2.07",1,2,7)) result=false;
if (!test_one_date("12.3.07",12,3,7)) result=false;
if (!test_one_date("1.12.07",1,12,7)) result=false;
if (!test_one_date("17.18.07",17,18,7)) result=false;

// Jahr vierstellig, Tag und Monat einstellig und zweistellig
if (!test_one_date("1.2.1920",1,2,1920)) result=false;
if (!test_one_date("12.3.1920",12,3,1920)) result=false;
if (!test_one_date("1.12.1920",1,12,1920)) result=false;
if (!test_one_date("17.18.1920",17,18,1920)) result=false;

// Teste unzulässige Formate. Das Programm sollte nicht abstürzen
//                            und -1 zurückgeben:
if (!test_one_date("",   -1,-1,-1)) result=false;
if (!test_one_date(".",  -1,-1,-1)) result=false;
if (!test_one_date("1",  -1,-1,-1)) result=false;
if (!test_one_date("1.", -1,-1,-1)) result=false;
if (!test_one_date("1.2",-1,-1,-1)) result=false;
if (!test_one_date("..", -1,-1,-1)) result=false;
if (!test_one_date("...",-1,-1,-1)) result=false;
if (!test_one_date("abc",-1,-1,-1)) result=false;
if (!test_one_date("1. Dezember ",-1,-1,-1)) result=false;
return result;
}

void __fastcall TForm1::Button_4_1_2Click(TObject *Sender)
{
Date_t d=stringToDate(Edit1->Text.c_str());
Memo1->Lines->Add(Edit1->Text+": Tag="+IntToStr(d.day)+
           " Monat="+IntToStr(d.month)+
           " Jahr="+IntToStr(d.year));
}


// --------------- Aufgabe 4.1, 3. -------------------------

// a)
void tokenize_0(const string& s, string separators=";.,- ")
{ // ";.,- " ist ein Default Argument für den Parameter separators
  // Damit kann tokenize ohne Argument für den zweiten Parameter aufgerufen
  // werden. separators kann auch als lokale Variable definiert werden.
Form1->Memo1->Lines->Add("string Test: "+AnsiString(s.c_str()));
string::size_type pos_tok=s.find_first_not_of(separators);
while (pos_tok!=string::npos)
  { // token gefunden ab Position pos_tok
    string::size_type  pos_sep=s.find_first_of(separators,pos_tok);
    if (pos_sep==string::npos)
      pos_sep=s.length();
    string token=s.substr(pos_tok,pos_sep-pos_tok);

    Form1->Memo1->Lines->Add("'"+AnsiString(token.c_str())+"'");
    pos_tok=s.find_first_not_of(separators,pos_sep+1);
  }
}

void __fastcall TForm1::Button_4_1_3_testClick(TObject *Sender)
{
tokenize_0("");      // leerer String , keine Ausgabe
tokenize_0(" ");     // nur Separator, keine Ausgabe

tokenize_0("123");   // nur token, Ausgabe "123"
```

```
tokenize_0(";123");  // Separator am Anfang, ein token, Ausgabe "123"
tokenize_0("123;");  // Separator am Ende, ein token, Ausgabe "123"
tokenize_0(";123;"); // Separator am Anfang und Ende, ein token, Ausgabe "123"

tokenize_0("456 ab.xy");   //Ausgabe "456", "ab", "xy"
tokenize_0(" 456 ab.xy");  // Separator am Anfang, Ausgabe "789","ab","xy"
tokenize_0("456 ab.xy;");  // Separator am Ende, Ausgabe "789","ab","xy"
tokenize_0("  456  ab.xy;");  //  Separator  am  Anfang  und  am  Ende,  Ausgabe
"789","ab","xy"
}


void __fastcall TForm1::Button_4_1_3Click(TObject *Sender)
{
tokenize_0(Edit1->Text.c_str());  //  separator  at  the  beginning  and  the  end,
output "789","ab","xy"
}


// --------------- Aufgabe 4.1.4 ---------------------------

// siehe StringUt.cpp

void __fastcall TForm1::Button_4_1_4Click(TObject *Sender)
{
AnsiString a1=Edit1->Text;
AnsiString a2=Edit2->Text;
string s1=a1.c_str();
string s2=a2.c_str();

double n=StringSimilarityNGram(s1,s2);
Memo1->Lines->Add("StringSimilarityNGram("+a1+","+a2+")="+
                    FloatToStr(n));
}


// --------------- Aufgabe 4.1.5 ---------------------------

void __fastcall TForm1::Button415Click(TObject *Sender)
{
AnsiString a1=Edit1->Text;
AnsiString a2=Edit2->Text;
string s1=a1.c_str();
string s2=a2.c_str();
string s;
int n=lcsLength(a1.c_str(),a2.c_str(),s);
Memo1->Lines->Add("lcs("+a1+","+a2+")='"+s.c_str()+"' n="
                      +IntToStr(n));
}


#ifdef _DEBUG
bool test_StringAll()
{
bool result=true;
if (!test_StringUtAll()) result=false;
if (!test_StringToDate()) result=false;
return result;
}
#endif
```

11.4.2 Aufgaben 4.2.3

```
// File: C:\Loesungen_CB2006\Kap_4\4.2\2_3.cpp

// #include "D:\Loesungen\Language.h"
// --------------- Aufgaben 4.2.3 --------------------------
// --------------- Aufgabe 1 -------------------------------

#include <vector>
using std::vector; // or: using namespace std;

void SiebEratosthenes_Index(int n)
{
// n muss keine Konstante sein:
// Ersetze
//   const int n=1000;
//   bool prim[n+1];
// durch
vector<bool> prim(n+1);
// p[i]==true, falls i eine Primzahl ist, sonst false
for (int i=0;i<n;i++) prim[i] = true;
for (int i=2; i<=n; i++)
  if (prim[i])
    for (int j=i;  j<=n/i; j++)
      prim[j*i] = false;
for (int i=2; i<=n; i++)
  if (prim[i])
    Form1->Memo1->Lines->Add(IntToStr(i));
};

void SiebEratosthenes_at(int n)
{
vector<bool> prim(n+1);
for (int i=0;i<n;i++) prim.at(i) = true;
for (int i=2; i<=n; i++)
  if (prim.at(i))
    for (int j=i;  j<=n/i; j++)
      prim.at(j*i) = false;
for (int i=2; i<=n; i++)
  if (prim.at(i))
    Form1->Memo1->Lines->Add(IntToStr(i));
};


void SiebEratosthenes_it(int n)
{ // Die anderen beiden Lösungen sind besser
vector<bool> prim(n+1);
typedef vector<bool>::iterator Iterator;
for (Iterator i=prim.begin();i!=prim.end();i++) *i = true;
for (Iterator i=prim.begin()+2;i!=prim.end();i++)
  if (*i)
    for (int j=i-prim.begin();  j<=n/(i-prim.begin()); j++)
      *(prim.begin()+j*(i-prim.begin())) = false;
for (Iterator i=prim.begin()+2;i!=prim.end();i++)
  if (*i)
    Form1->Memo1->Lines->Add(IntToStr(i-prim.begin()));
};




// --------------- Aufgabe 2 -------------------------------

#include <algorithm>
using namespace std;

void Aufg4332()
{
// a)
  int a[10]={9,8,7,6,5,4,3,2,1,0};
```

```
  sort(a,a+10);
  for (int i=0;i<10; i++)
    Form1->Memo1->Lines->Add(a[i]);
// b)
  int b[10];
  copy(a,a+10,b);
  for (int i=0;i<10; i++)
    Form1->Memo1->Lines->Add(b[i]);

// c)
  if (equal (a,a+10,b))
    Form1->Memo1->Lines->Add("gleich");
  else
    Form1->Memo1->Lines->Add("ungleich");
  b[1]=17;
  if (equal (a,a+10,b))
    Form1->Memo1->Lines->Add("gleich");
  else
    Form1->Memo1->Lines->Add("ungleich");
}

// --------------- Aufgabe 3 -------------------------------

// Die Lösung dieser Aufgabe ist das Projekt VecCont

// --------------- Aufgabe 4 -------------------------------

// Die Lösung dieser Aufgabe ist enthalten in:
#include "..\..\CppUtils\StringUt.h"


// --------------- Aufgabe 5 -------------------------------

double SumV(vector<double> v)
{
double s=0;
for (int i=0; i<v.size(); i++)
  s=s+v[i];
return s;
}

double SumA(double* a, int n)
{
double s=0;
for (int i=0; i<n; i++)
  s=s+a[i];
return s;
}

// Bei der Übergabe eines Arrays an eine Funktion muss man die
// Anzahl der Elemente als zusätzlichen Parameter übergeben.
// Das ist bei den Containern der Standardbibliothek nicht
// notwendig, da sie ihre Anzahl "wissen".
```

11.4.3 Aufgaben 4.2.5

```
// File: C:\Loesungen_CB2006\Kap_4\4.2\2_5.cpp

// #include "D:\Loesungen\Language.h"
// --------------- Aufgaben 4.2.5 --------------------------
// --------------- Aufgabe 1 -------------------------------
```

```
void ShowCapacity(int n)
{
vector<int> v;
vector<int>::size_type c, c0=v.capacity();
for (int i=0; i<n; i++)
  {
    v.push_back(i);
    c=v.capacity();
    if (c!=c0)
    Form1->Memo1->Lines->Add("v cap: "+IntToStr(v.capacity())+" "+IntToStr(i));
    c0=c;
  }
v.clear();
Form1->Memo1->Lines->Add("nach erase: "+IntToStr(v.capacity()));
}


// --------------- Aufgabe 2 -------------------------------

// Die Lösung dieser Aufgabe ist das Projekt GaussEl:


// --------------- Aufgabe 3 -------------------------------

void PascalsTriangle(int n)
{
vector<vector<int> > p(n);
// reserviere Speicher für die Dreiecksmatrix:
for (int i=0; i<n; i++)
  p[i].resize(i+1); // p[i] hat i+1 Elemente p[i,0] ... p[i,i]

// Berechne die Werte des Pascal-Dreiecks:
p[0][0]=1;
for (int i=1; i<n; i++)
  {
  p[i][0]=1;
  p[i][i]=1;
  for (int j=1; j<=i-1; j++)
    p[i][j]=p[i-1][j-1]+p[i-1][j];
  };

// Werte anzeigen:
for (int i=0; i<n; i++)
  {
  AnsiString s;
  for (int j=0; j<=i; j++)
    s=s+IntToStr(p[i][j])+" ";
  Form1->Memo1->Lines->Add(s);
  };
}
```

11.4.4 Aufgaben 4.2

```
// File: C:\Loesungen_CB2006\Kap_4\4.2\SeqContU.cpp

#include <vcl.h>
#pragma hdrstop
#include "SeqContU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------
```

```
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
Edit1->Text="100";
}


// --------------- Aufgaben 4.2.3 --------------------------

#include "2_3.cpp"

// --------------- Aufgabe 1 -------------------------------

void __fastcall TForm1::EratosthClick(TObject *Sender)
{
int n=StrToInt(Edit1->Text);
SiebEratosthenes_Index(n);
SiebEratosthenes_at(n);
SiebEratosthenes_it(n);
}


// --------------- Aufgabe 2 -------------------------------

void __fastcall TForm1::STLAlgClick(TObject *Sender)
{
Aufg4332();
}


// --------------- Aufgabe 3 -------------------------------

// Die Lösung dieser Aufgabe ist das Projekt DVVect


// --------------- Aufgabe 4 -------------------------------

void __fastcall TForm1::TokenizeClick(TObject *Sender)
{
vector<AnsiString> v=tokenize(Edit1->Text);
for (vector<AnsiString>::iterator i=v.begin();i!=v.end() ;i++ )
  Memo1->Lines->Add(i->c_str());
}


// --------------- Aufgabe 5 -------------------------------

void __fastcall TForm1::SumClick(TObject *Sender)
{
vector<double> v;
double s1=SumV(v);

double a[10]={1,2,3,4,5};
double s2=SumA(a,5);
}


// --------------- Aufgaben 4.2.5 --------------------------

#include "2_5.cpp"

// --------------- Aufgabe 1 -------------------------------

void __fastcall TForm1::CapacityClick(TObject *Sender)
{
ShowCapacity(10000);
}
```

```
// --------------- Aufgabe 2 -------------------------------

// Die Lösung dieser Aufgabe ist das Projekt GaussEl


// --------------- Aufgabe 3 -------------------------------

void __fastcall TForm1::PascalTriClick(TObject *Sender)
{
PascalsTriangle(20);
}


// --------------- Aufgabe 4.3.6 ---------------------------

// Die Lösung dieser Aufgabe ist im Projekt DVVect enthalten
```

11.4.5 Aufgaben 4.2.3, 3.

```
// File: C:\Loesungen_CB2006\Kap_4\4.2\VecContU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "VecContU.h"
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

// --------------- Aufgabe 4.2.3, 3. ------------------------

#include <algorithm>
using namespace std;

#include <vector>
#include <list>  // für Aufgabe 4.2.7
#include <deque> // für Aufgabe 4.2.7
using namespace std;


typedef vector<AnsiString> Container;
// typedef list<AnsiString> Container;   // für Aufgabe 4.3.7
// typedef deque<AnsiString> Container;  // für Aufgabe 4.3.7


typedef Container::iterator Iterator;
Container c;

// a)
void __fastcall TForm1::EinfuegenClick(TObject *Sender)
{
c.push_back(Edit1->Text);
}

// b)
void __fastcall TForm1::AnzeigenClick(TObject *Sender)
{
for (Iterator i=c.begin(); i!=c.end(); i++)
```

```
  Memo1->Lines->Add(*i);
// Die folgende Variante ist ebenso möglich:
// for (int i=0; i<A.size(); i++)
//  Memo1->Lines->Add(A[i]);
}

// c)
void __fastcall TForm1::LinSuchenClick(TObject *Sender)
{
Iterator i=find(c.begin(),c.end(),Edit1->Text);
while (i!=c.end())
  {
    Memo1->Lines->Add(*i);
    i=find(i+1,c.end(),Edit1->Text);
  }
}

// d)
void __fastcall TForm1::LoeschenClick(TObject *Sender)
{
Iterator i=find(c.begin(),c.end(),Edit1->Text);
if (i!=c.end()) c.erase(i);
else Memo1->Lines->Add("nicht gefunden");
}

// e)
void __fastcall TForm1::SortierenClick(TObject *Sender)
{
sort(c.begin(),c.end()); // Für vector und deque
// c.sort();  // Für list
}

// f)
void __fastcall TForm1::BinSuchenClick(TObject *Sender)
{
if (binary_search(c.begin(), c.end(), Edit1->Text))
  Memo1->Lines->Add(Edit1->Text);
else
  Memo1->Lines->Add("nicht gefunden");
}

// g)
void __fastcall TForm1::SortEinfClick(TObject *Sender)
{
Iterator i=lower_bound(c.begin(), c.end(),Edit1->Text);
c.insert(i,Edit1->Text);
}

// h)
void __fastcall TForm1::SortLoeschenClick(TObject *Sender)
{
Iterator i=lower_bound(c.begin(), c.end(),Edit1->Text);
if (i!=c.end())
  c.erase(i);
}


// --------------- Aufgabe 4.2.7 ---------------------------

void __fastcall TForm1::Button_4_2_7Click(TObject *Sender)
{
// Zur Lösung dieser Aufgabe muss nur der Datentyp des Containers
// geändert und bei list der Aufruf von sort geändert werden.
}
```

11.4.6 Aufgabe 4.2.3, 5.

```
// File: C:\Loesungen_CB2006\Kap_4\4.2\VecDVU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "VecDVU.h"
#pragma resource "*.dfm"

TKBForm1 *KBForm1;
//---------------------------------------------------------------------
__fastcall TKBForm1::TKBForm1(TComponent* Owner)
    : TForm(Owner)
{
}


// --------------- Aufgabe 1 a) ----------------------------

// Damit die Datenstruktur CKontobewegung auch noch in anderen
// Programmen verwendet werden können, in eine eigene Datei:
#include "..\..\CppUtils\KBDecl.h"

// Damit die Funktionen FormToKB und KBToForm auch noch in anderen
// Programmen verwendet werden können, in eine eigene Datei:
#include "..\..\CppUtils\KBForms.h"
// Für die Funktionen KBToForm und FormToKB muß das Formular
// den Namen "KBForm1" haben (im Objektinspektor setzen).


#include<vector>
#include<deque>
#include<list>
using namespace std;

typedef vector<Kontobewegung> Container;
// typedef list<Kontobewegung> Container; // geht nicht
// typedef deque<Kontobewegung> Container; // das geht auch

typedef Container::iterator Iterator;
Container c;
Iterator pos=c.begin();


// --------------- Aufgabe 1 g) ----------------------------

void EnableButtons()
{
KBForm1->first->Enabled =!c.empty();
KBForm1->last->Enabled  =!c.empty();

KBForm1->prev->Enabled=(!c.empty()) && (pos>c.begin());
KBForm1->next->Enabled=(!c.empty()) && (pos<c.end()-1);
KBForm1->Loeschen->Enabled =(!c.empty()) ;//&& (pos<=c.end()-1);
// Speichern ist immer enabled und muss nie geändert werden

// i)
if (c.empty())
  KBForm1->Label1->Caption="Container leer";
else
  KBForm1->Label1->Caption="Satz Nr. "+IntToStr(pos-c.begin()+1)+
                           " von "+IntToStr(c.end()-c.begin());
}

void __fastcall TKBForm1::FormCreate(TObject *Sender)
{
ClearKBForm(KBForm1); // nur zum Testen, damit man nicht so viele Werte
```

```
                        // eingeben muss

EnableButtons();
}

// --------------- Aufgabe 1 b) ----------------------------

void __fastcall TKBForm1::SpeichernClick(TObject *Sender)
{
c.push_back(FormToKB(KBForm1));

/* h) */ // Durch das Einfügen wird pos ungültig. Deshalb
/* h) */ // auf eine Position (z.B. die des angezeigten
/* h) */ // Datensatzes, also des letzten - wegen push_back) setzen.
/* h) */ pos=c.end()-1;

EnableButtons();
}

// --------------- Aufgabe 1 c) ----------------------------

void __fastcall TKBForm1::prevClick(TObject *Sender)
{ // Da dieser Button nur unter der Bedingung 'pos>c.begin()' enabled
  // ist, braucht diese Bedingung hier nicht mit 'if' geprüft werden.
  // Damit der Button immer im richtigen Zustand ist, muss die Funktion
  // EnableButtons nach jeder Veränderung des Containers oder von
  // pos aufgerufen werden.
if (pos>c.begin()) // überflüssig, falls die Buttons immer richtig
  {                // enabled sind.
    pos--;
    KBToForm(*pos,KBForm1);
    EnableButtons();
  }
}

// --------------- Aufgabe 1 d) ----------------------------

void __fastcall TKBForm1::nextClick(TObject *Sender)
{
if (pos<c.end()-1) // Siehe den Kommentar in 'prevClick'
  {
    pos++;
    KBToForm(*pos,KBForm1);
  }
EnableButtons();
}

// --------------- Aufgabe 1 e) ----------------------------

void __fastcall TKBForm1::firstClick(TObject *Sender)
{
if (!c.empty())
  {
    pos=c.begin();
    KBToForm(*pos,KBForm1);
  }
else ClearKBForm(KBForm1); // Container leer
EnableButtons();
}

void __fastcall TKBForm1::lastClick(TObject *Sender)
{
if (!c.empty())
  {
    pos=c.end()-1;
    KBToForm(*pos,KBForm1);
  }
else ClearKBForm(KBForm1); // Container leer
EnableButtons();
```

```
}

// --------------- Aufgabe 1 f) ----------------------------

void __fastcall TKBForm1::LoeschenClick(TObject *Sender)
{
/* h) */ // Durch das Löschen wird pos ungültig. Deshalb 'pos' auf
/* h) */ pos=c.erase(pos); // eine Position setzen, damit der Iterator
                           // auch nach dem Löschen wieder gültig ist.
if (pos!=c.end())
  KBToForm(*pos,KBForm1);
else // pos == c.end(), d.h. das letzte Element wurde gelöscht
  {
    if (!c.empty())
      {
        pos--;
        KBToForm(*pos,KBForm1);
      }
    else // das einzige Element wurde gelöscht
      ClearKBForm(KBForm1);
  }
EnableButtons();
}

// --------------- Aufgabe 1 h) ----------------------------

// In den Lösungen von b) und f) enthalten

// --------------- Aufgabe 1 i) ----------------------------

// In EnableButtons() enthalten.
```

11.4.7 Aufgaben 4.2.5, 2.

```
// File: C:\Loesungen_CB2006\Kap_4\4.2\GaussElU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "GaussElU.h"
#pragma resource "*.dfm"

TForm1 *Form1;
#include "..\..\CppUtils\testutils.h"

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
initTestUtils();
bool test_randomGaussElimination(int k); // Prototyp
testUtilsSummary(Form1->Memo1, test_randomGaussElimination(100));
}


// --------------- Aufgabe 4.2.5, 2. -----------------------

// Gegenüber der Aufgabe 4.2.5, 3 wurden nur die Datentypen der
// Parameter und Argumente geändert.

#include <vector>
using namespace std;

typedef  vector<vector<double> > TMatrix_double;
typedef  vector<double> TVector_double;
```

```
int Zeile_mit_groesstem_Spaltenwert(int i, TMatrix_double a)
{
int Max=i;
for (int j=i+1; j<a[i].size(); j++)
  if (labs(a[j][i])>labs(a[Max][i])) Max =j;
return Max;
}

void vertausche_Zeilen(int z1, int z2, TMatrix_double a, TVector_double b)
{
double h;
for (int i=z1; i<a[i].size(); i++)
  {
    h=a[z1][i];
    a[z1][i]=a[z2][i];
    a[z2][i]=h;
  }
h=b[z1];
b[z1]=b[z2];
b[z2]=h;
}

void  GaussElimination(TMatrix_double  a,  TVector_double  b,  TVector_double&  x
,TVector_double& p, int n)
{// Koeffizienten: a, rechte Seite: b, Lösung: x, Probe: p
for (int i=0; i<n; i++)
  for (int j=i+1; j<n; j++)
    {
      int z1=Zeile_mit_groesstem_Spaltenwert(i,a);
      if (i!=z1) vertausche_Zeilen(i, z1, a, b);
      if (a[i][i]==0)
        { // Abbruch
          ShowMessage("Nicht lösbar");
          return;
        }
      double f=-a[j][i]/a[i][i];
      a[j][i]=0;
      for (int k=i+1;k<n; k++)
        a[j][k]=a[j][k]+f*a[i][k];
      b[j]=b[j]+f*b[i];
    }

// rechte Seite "rückwärts" auflösen:
for (int i=n-1; i>=0; i--)
 {
   x[i]=b[i];
   for (int j=n-1; j>i; j--)
     x[i]=x[i]-a[i][j]*x[j];
   x[i]=x[i]/a[i][i];
 }

// Probe:
for (int i=0; i<n; i++)
  {
    p[i]=-b[i];
    for (int j=0; j<n; j++)
      p[i]=p[i]+a[i][j]*x[j];
  }
}


void Loesung_und_Probe_anzeigen(TVector_double x, TVector_double p, int n)
{

double sum=0;
for (int i=0; i<n; i++)
  {
    AnsiString is=IntToStr(i);
    sum=sum+p[i];
```

```
    Form1->Memo1->Lines->Add("x"+is+"="+FloatToStr(x[i]));
  }
Form1->Memo1->Lines->Add("diff="+FloatToStr(sum));
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
int n=3;
Edit1->Text=IntToStr(n);
StringGrid1->ColCount=n+1;
StringGrid1->RowCount=n;
StringGrid1->Options<<goEditing;
StringGrid1->FixedRows=0;       // keine Zeilenbeschriftung links
StringGrid1->FixedCols=0;       // keine Spaltenüberschriften
StringGrid1->ScrollBars=ssNone; // kein Scrollbars anzeigen
}

void __fastcall TForm1::GaussElimClick(TObject *Sender)
{
int n=StrToInt(Edit1->Text); // Dimension der Koeffizientenmatrix
#if ((__BORLANDC__ >= 0x0580) && (__BORLANDC__ < 0x0590))
// bug in C++Builder 2006:
TMatrix_double a(n,vector<double>(n)); // Koeffizienten
#else
TMatrix_double a(n,n); // Koeffizienten
#endif
TVector_double b(n), // rechte Seite
               x(n), // Lösung
               p(n); // Probe
// Übernehme die Werte aus dem StringGrid:
// Beachte dabei, daß ein StringGrid [Spalte][Zeile] adressiert und
// nicht wie C++ [Zeile][Spalte]

for (int i=0; i<n; i++)
  for (int j=0; j<n; j++)
    {
      if (StringGrid1->Cells[j][i].Trim()==="") StringGrid1->Cells[j][i]="0";
      a[i][j]=StrToFloat(StringGrid1->Cells[j][i]);
    }
for (int j=0; j<n; j++)
  {
    if (StringGrid1->Cells[n][j].Trim()==="") StringGrid1->Cells[n][j]="0";
    b[j]=StrToFloat(StringGrid1->Cells[n][j]);
  }
GaussElimination(a, b, x, p, n);
Loesung_und_Probe_anzeigen(x ,p, n);
}

void __fastcall TForm1::Edit1Change(TObject *Sender)
{ // Lese hier die Größe der Matrix ein (sehr einfach):
if (Edit1->Text.Trim()!="")
  {
    int n=StrToInt(Edit1->Text);
    StringGrid1->ColCount=n+1;
    StringGrid1->RowCount=n;
    // Passe die Größe des StringGrid an:
    StringGrid1->Width=(n+1)*StringGrid1->DefaultColWidth+(n+2)*StringGrid1-
>GridLineWidth;
    StringGrid1->Height=n*StringGrid1->DefaultRowHeight+(n+1)*StringGrid1-
>GridLineWidth;
  }
}

void ArrayTo_3x3_double_Matrix(TMatrix_double& m, double p[3][3])
{
for (int i=0; i<3; i++)
  for (int j=0; j<3; j++)
    {
      m[i][j]=p[i][j];
```

```
      Form1->StringGrid1->Cells[j][i]=FloatToStr(p[i][j]);
    }
}

void ArrayTo_3_double_Vector(TVector_double& v, double p[3])
{
int n=3;
for (int i=0; i<3; i++)
  {
    v[i]=p[i];
    Form1->StringGrid1->Cells[n][i]=FloatToStr(p[i]);
  }
}

int test=0; // Zum Durchlaufen der verschiedenen Testfälle

void TestGaussElim()
{ // Eliminationsverfahren von Gauß
int n=3;
Form1->StringGrid1->ColCount=n+1;
Form1->StringGrid1->RowCount=n;

// Koeffizienten und rechte Seite der verschiedenen Testfälle

double a1[3][3]={{1,2,3},   // coefficients
                 {1,4,6},
                 {2,3,7}};
double b1[3]={1,2,3};       // right hand side

double a2[3][3]={{2,3,-5},  // coefficients
                 {4,8,-3},
                 {-6,1,4}};
double b2[3]={-10,-19,-11}; // right hand side

double a3[3][3]={{1,2,3},   // not solvable
                {1,2,3},
                {2,4,6}};
double b3[3]={1,2,3};       // right hand side


TVector_double b(n);        // coefficients

#if ((__BORLANDC__ >= 0x0580) && (__BORLANDC__ < 0x0590))
// bug in C++Builder 2006 ?
TMatrix_double a(n,vector<double>(n));      // right hand side
#else
TMatrix_double a(n,n);      // right hand side
#endif

test++;
if (test==3) test=0;
if (test==0)
  {
    ArrayTo_3x3_double_Matrix(a,a1);
    ArrayTo_3_double_Vector(b, b1);
  }
else if (test==1)
  {
    ArrayTo_3x3_double_Matrix(a,a2);
    ArrayTo_3_double_Vector(b, b2);
  }
else if (test==2)
  {
    ArrayTo_3x3_double_Matrix(a,a3);
    ArrayTo_3_double_Vector(b, b3);
  }

TVector_double x(3),  // solution
               p(3);  // test
```

```
GaussElimination(a, b, x ,p, n);
Loesung_und_Probe_anzeigen(x, p, n);
}

bool test_randomGaussElimination(int k)
{// teste mit zufälligen Koeffizienten a und b
const int n=3;
bool result = true;
for (int i=0; i<k; i++)
  { //
    double a1[n][n]={{rand(),rand(),rand()},
                     {rand(),rand(),rand()},
                     {rand(),rand(),rand()}};
    double b1[n]={rand(),rand(),rand()};
    TVector_double b(n);        // coefficients
#if ((__BORLANDC__ >= 0x0580) && (__BORLANDC__ < 0x0590))
// Bug in C++Builder 2006: Workaround, because the next line does not work
//                         in C++Builder 2007
    TMatrix_double a(n,vector<double>(n));
#else
// This works in C++Builder 6 and 2007
    TMatrix_double a(n,n);
#endif
    ArrayTo_3x3_double_Matrix(a,a1);
    ArrayTo_3_double_Vector(b, b1);

    TVector_double x(3),  // solution
                   p(3);  // test

    GaussElimination(a, b, x ,p, n);
    double diff=0;
    for (int i=0; i<n; i++)
    diff+=p[i]*p[i];

    if (!assertEqual(diff,0.0,"Gausselim")) result=false;
  }
return result;
}

void __fastcall TForm1::TestClick(TObject *Sender)
{
TestGaussElim();
}
```

11.4.8 Aufgaben 4.3.3, 4.3.4.2, 4.3.8 und 4.3.9

```
// File: C:\Loesungen_CB2006\Kap_4\4.3\FilesU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "FilesU.h"
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------

#include "..\..\CppUtils\TestUtils.h"
#include <string>
using std::string;
const string TestDir="c:\\test";
const string TestTextfile=TestDir+"\\test.txt";
const string TestExefile= TestDir+"\\test.exe";
```

```
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
// Erzeuge das Verzeichnis, falls es nicht existiert
if (CreateDirectory(TestDir.c_str(),0))
  ShowMessage(AnsiString("directory '")+TestDir.c_str()+"' erzeugt");

// Damit die Dateien für die folgenden Aufgaben vorhanden sind:
if (!FileExists(TestTextfile.c_str()))
  {
    BOOL b=CopyFile("\\Loesungen_BCB2006\\Kap_4\\4.3\\FilesU.cpp",
        TestTextfile.c_str(), true);
    if (b) ShowMessage(TestTextfile.c_str()+AnsiString(" erzeugt"));
  }
if (!FileExists(TestExefile.c_str()))
  {
    BOOL b=CopyFile("\\Loesungen_BCB2006\\Kap_4\\4.3\\Debug_Build\\Files.exe",
        TestExefile.c_str(), true);
    if (b) ShowMessage(TestExefile.c_str()+AnsiString(" erzeugt"));
  }
// Die Windows-API-Funktion 'CopyFile' ist in der Online-Hilfe
// zum Win32-SDK beschrieben.

#ifdef _DEBUG
bool test_All(); // Prototyp
initTestUtils();
testUtilsSummary(Memo1,test_All());
#endif
}


// --------------- Aufgaben 4.3.3 --------------------------

// --------------- Aufgabe 1 a) ----------------------------

#include <fstream>
// using namespace std;

int countBytes(const char* fn)
{
char c;
int n=0;
ifstream f(fn,ios::binary);
if (!f) ShowMessage("open error");
while (f.read(&c,sizeof(c)))
  n++;
if (!f.eof()) ShowMessage("read error");
f.close();
return n;
}

int countBytesTxt(const char* fn)
{
char c;
int n=0;
ifstream f(fn);
if (!f) ShowMessage("open error");
while (f.read(&c,sizeof(c)))
  n++;
if (!f.eof()) ShowMessage("read error");
f.close();
return n;
}

int countBytes(const char* fn, bool binary)
{
char c;
int n=0;
ifstream f;
```

```
if (binary) f.open(fn,ios::binary);
else f.open(fn);
// wie
// ifstream f(fn,ios::binary); oder
// ifstream f(fn); // ohne ios::binary ist das Ergebnis sowohl für
                   // Text- als auch für Binärdateien falsch
if (!f) ShowMessage("open error");
while (f.read(&c,sizeof(c)))
  n++;
if (!f.eof()) ShowMessage("read error");
f.close();
return n;
}

void createTestfile(char* fn,int n)
{
ofstream fout((TestDir+fn).c_str());
char a[]={'1','2','3',26,'5','6'};
fout.write(a,n);
fout.close();
}

bool test_countBytes()
{
bool result=true;
createTestfile("0",0);
createTestfile("1",1);
createTestfile("6",6);

int n0=countBytes((TestDir+"0").c_str(),false);
if (!assertEqual(n0,0,"countBytes 0")) result=false;
int n0b=countBytes((TestDir+"0").c_str(),true);
if (!assertEqual(n0b,0,"countBytes 0 b")) result=false;

int n1=countBytes((TestDir+"1").c_str(),false);
if (!assertEqual(n1,1,"countBytes 1")) result=false;
int n1b=countBytes((TestDir+"1").c_str(),true);
if (!assertEqual(n1b,1,"countBytes 1 b")) result=false;

int n2=countBytes((TestDir+"6").c_str(),false);
if (!assertEqual(n2,3,"countBytes 1")) result=false;
int n2b=countBytes((TestDir+"6").c_str(),true);
if (!assertEqual(n2b,6,"countBytes 1 b")) result=false;

return result;
}

void __fastcall TForm1::CountBytesClick(TObject *Sender)
{
int n=countBytes(TestTextfile.c_str());
Memo1->Lines->Add(TestTextfile.c_str()+AnsiString("         (binary         mode):
")+IntToStr(n)+" bytes");
n=countBytesTxt(TestTextfile.c_str());
Memo1->Lines->Add(TestTextfile.c_str()+AnsiString(" (text mode): ")+IntToStr(n)+"
bytes");

n=countBytes(TestExefile.c_str());
Memo1->Lines->Add(TestExefile.c_str()+AnsiString("         (binary         mode):
")+IntToStr(n)+" bytes");
n=countBytesTxt(TestExefile.c_str());
Memo1->Lines->Add(TestExefile.c_str()+AnsiString(" (text mode): ")+IntToStr(n)+"
bytes");


n=countBytes(TestTextfile.c_str(),true);
Memo1->Lines->Add(TestTextfile.c_str()+AnsiString("   (binary):   ")+IntToStr(n)+"
bytes");
n=countBytes(TestTextfile.c_str(),false);
Memo1->Lines->Add(TestTextfile.c_str()+AnsiString(": ")+IntToStr(n)+" bytes");
```

```
n=countBytes(TestExefile.c_str(),true);
Memo1->Lines->Add(TestExefile.c_str()+AnsiString("   (binary):   ")+IntToStr(n)+"
bytes");
n=countBytes(TestExefile.c_str(),false);
Memo1->Lines->Add(TestExefile.c_str()+AnsiString(": ")+IntToStr(n)+" bytes");
// Die Ergebnisse stimmen nur für Dateien, die im Binärmodus geöffnet
// wurden.
}


// --------------- Aufgabe 1 b) ----------------------------

void CopyFile(const char* src, const char* dst)
{
ifstream fin(src,ios::binary);
AnsiString msg="open error ";
if (!fin) ShowMessage(msg+src);
ofstream fout(dst,ios::binary);
if (!fout) ShowMessage(msg+dst);
char c;
while (fin.read(&c,sizeof(c)))
  {
    fout.write(&c,sizeof(c));
    if (!fout) ShowMessage("write error");
  }
if (!fin.eof()) ShowMessage("read error");
fin.close();
fout.close();
}

const string CopyTextDest=TestDir+"\\Ctest.txt";
const string CopyExeDest =TestDir+"\\Ctest.exe";

void test_CopyFile()
{
CopyFile(TestTextfile.c_str(), CopyTextDest.c_str());
CopyFile(TestExefile.c_str(),  CopyExeDest.c_str());
}

void __fastcall TForm1::CopyFilesClick(TObject *Sender)
{
test_CopyFile();
}


// --------------- Aufgabe 1 c) ----------------------------

bool CompareFiles(const char* fn1, const char* fn2)
{
AnsiString msg="open error ";

ifstream f1(fn1,ios::binary);
if (!f1) ShowMessage(msg+fn1);

ifstream f2(fn2,ios::binary);
if (!f2) ShowMessage(msg+fn2);

bool gleich=true; // zwei leere Dateien sind gleich
while (f1 && f2 && gleich)
  {
    char c1,c2;
    f1.read(&c1,sizeof(c1));
    f2.read(&c2,sizeof(c2));

    gleich=(c1==c2);
  }
if (!f1 && !f1.eof()) ShowMessage("read error f1");
if (!f2 && !f2.eof()) ShowMessage("read error f2");
```

```
if (gleich)
  gleich=(f1.eof()==f2.eof()); // Falls die Dateien verschieden lang sind
f1.close();
f2.close();
return gleich;
/*
Die folgende Schleife funktioniert nicht, da wegen der short circuit
evaluation von booleschen Ausdrücken beim Ende von f1 in f2 nicht mehr
gelesen wird:

while (f1.read(&c1,sizeof(c1)) && f2.read(&c2,sizeof(c2)) && gleich)
  gleich=(c1==c2);
*/
}


bool test_copy_and_compare_files()
{
test_CopyFile();
bool result=true;
bool b=CompareFiles(TestTextfile.c_str(), CopyTextDest.c_str());
if (!assertEqual(b,true,"CompareFiles 1")) result=false;
b=CompareFiles(TestExefile.c_str(),  CopyExeDest.c_str());
if (!assertEqual(b,true,"CompareFiles 2"))  result=false;
b=CompareFiles(TestTextfile.c_str(), CopyExeDest.c_str());
if (!assertEqual(b,false,"CompareFiles 3"))  result=false;
b=CompareFiles(TestExefile.c_str(),  CopyTextDest.c_str());
if (!assertEqual(b,false,"CompareFiles 4")) result=false;
return result;
}

bool test_All()
{
bool result=true;
if (!test_countBytes()) result=false;
if (!test_copy_and_compare_files()) result=false;
return result;
}

void __fastcall TForm1::ComparefilesClick(TObject *Sender)
{
if (test_copy_and_compare_files())
  Memo1->Lines->Add("All tests passed");
else
  Memo1->Lines->Add("Some tests failed");
}


// --------------- Aufgabe 1 d) ----------------------------

void WriteNTString(ofstream& f,char* s)
{ // schreibt alle Zeichen des nullterminierten Strings s
  // einschließlich des Nullterminators in den stream f
f.write(s,strlen(s)+1);
}

string ReadNTString(ifstream& f)
{
// Die globale Funktion getline der Standardbibliothek
// von C++ ist ähnlich aufgebaut und sollte dieser Funktion
// vorgezogen werden.
string s;
char c;
while (f.read(&c,1) && c!=0)
  s=s+c;
return s;
}
```

```
void __fastcall TForm1::WritNTSClick(TObject *Sender)
{
const char* fn=(TestDir+"\\nts.txt").c_str();
ofstream out(fn,ios::binary);
if (out)
  {
    WriteNTString(out,"123");
    WriteNTString(out,"der nächste String ist leer");
    WriteNTString(out,"");
    WriteNTString(out,"A");
  }
out.close();
ifstream in(fn,ios::binary);
if (in)
  {
    string s=ReadNTString(in);
    while (in)
      {
        Memo1->Lines->Add(s.c_str());
        s=ReadNTString(in);
      }
  }
in.close();
}


// --------------- Aufgabe 1 e) ----------------------------

void TextFilter(const char* src, const char* dst)
{
AnsiString msg="open error ";
ifstream in(src,ios::binary);
if (!in) ShowMessage(msg+src);
ofstream out(dst,ios::binary);
if (!out) ShowMessage(msg+dst);

char c;
while (in.read(&c,sizeof(c)))
  if ((' '<=c && c<='z') || c=='\n') // bis auf diese Anweisung identisch
                          // mit CopyFile
    if (!out.write(&c,sizeof(c)))
      ShowMessage("write error");
if (!in.eof()) ShowMessage("read error");
in.close();
out.close();
}

void __fastcall TForm1::FileFilterClick(TObject *Sender)
{

TextFilter(TestTextfile.c_str(),(TestDir+"\\filt_t.txt").c_str());
TextFilter(TestExefile.c_str(), (TestDir+"\\filt_e.txt").c_str());
TextFilter((TestDir+"\\test1.doc").c_str(),
           (TestDir+"\\filt_w.txt").c_str());
}



// --------------- Aufgaben 4.3.4 --------------------------

#include "3_4.cpp"

// --------------- Aufgabe 1 -------------------------------

void __fastcall TForm1::TextToHtmlClick(TObject *Sender)
{
OpenDialog1->Filter = "Text (*.txt)|*.txt";
OpenDialog1->InitialDir="\\Loesungen_2006\\Text\\";
if (OpenDialog1->Execute())
```

```
  {
    AnsiString fn=ExtractFileName(OpenDialog1->FileName);
    AnsiString html_fn=ChangeFileExt(fn,".html");
    TextToHTML(OpenDialog1->FileName.c_str(),
(TestDir.c_str()+AnsiString("\\")+html_fn).c_str());
    AnsiString url_dir=TestDir.c_str()+AnsiString("\\");
    if ((url_dir.Length()>=2) && (url_dir[2]==':'))
      url_dir[2]='|';
    WideString url="file:///"+url_dir+html_fn;
    CppWebBrowser1->Navigate(url,0,0,0,0);
    Form1->Caption=html_fn;
  }
}


// --------------- Aufgabe 2 ------------------------------

void __fastcall TForm1::KBToHtmlClick(TObject *Sender)
{
OpenDialog1->InitialDir=TestDir.c_str();
OpenDialog1->Filter = "Daten (*.dat)|*.DAT";
AnsiString default_fn="kb.dat";
AnsiString html_fn="kb.html";
OpenDialog1->FileName=default_fn;
if (OpenDialog1->Execute())
  {
    KBToHTML(OpenDialog1->FileName.c_str(),
         (TestDir.c_str()+AnsiString("\\")+html_fn).c_str());
    AnsiString url_testdir=TestDir.c_str();
    if ((url_testdir.Length()>=2) && (url_testdir[2]==':'))
      url_testdir[2]='|';
    WideString url="file:///"+url_testdir+"\\"+html_fn;
    CppWebBrowser1->Navigate(url,0,0,0,0);
    Form1->Caption="Kontobewegungen";
  }
}
```

11.4.9 Aufgaben 4.3.3

```
// File: C:\Loesungen_CB2006\Kap_4\4.3\SimpleDBU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "SimpleDBU.h"
#pragma resource "*.dfm"
TKBForm1 *KBForm1;

//---------------------------------------------------------------------------
__fastcall TKBForm1::TKBForm1(TComponent* Owner)
    : TForm(Owner)
{
}

// --------------- Aufgabe 4.3.3 ---------------------------

// --------------- Aufgabe 2 a) ----------------------------

#include "..\..\CppUtils\KBDecl.h"
#include "..\..\CppUtils\KBForms.h"

#include <fstream>
#include <string>
std::string s;
using namespace std;
```

```
fstream f;

AnsiString TestDir="c:\\test";
AnsiString Default_FileName="kb.dat";
AnsiString Default_Filter="KB-Dateien (*.dat)|*.dat";

void __fastcall TKBForm1::FormCreate(TObject *Sender)
{
// Erzeuge das Verzeichnis, falls es nicht existiert
if (CreateDirectory(TestDir.c_str(),0))
  ShowMessage("directory '"+TestDir+"' created");

ClearKBForm(KBForm1);

// Menüoptionen:
Neu1->Enabled=true;
ffnen1->Enabled=true;
Schliessen1->Enabled=false;
// Buttons Aufgabe 4.6.3.2:
Speichern->Enabled=false;
next->Enabled=false;
LinSuchen->Enabled=false;
// Buttons Aufgabe 4.6.8:
KBForm1->first->Enabled=false;
KBForm1->last->Enabled =false;
KBForm1->next->Enabled =false;
KBForm1->prev->Enabled =false;
}

void __fastcall TKBForm1::Neu1Click(TObject *Sender)
{
OpenDialog1->Filter = "KB-Dateien (*.dat)|*.dat";
OpenDialog1->FileName="kb.dat";
OpenDialog1->InitialDir=TestDir;

if (OpenDialog1->Execute())
  {
    // Für Aufgabe 4.3.3.2 reicht
    // f.open(OpenDialog1->FileName.c_str(),ios::binary|ios::out);

    // ----- Aufgabe 4.3.5.1 a) ---------------------------
    // Für Aufgabe 4.6.5.1 ist aber dieser Modus notwendig:
    f.open(OpenDialog1->FileName.c_str(),ios::binary|ios::out|ios::in);
    if (!f)
      {
        ShowMessage("Error open "+OpenDialog1->FileName);
        f.clear();
        f.close();
      }
    else
      {
        // Menüoptionen:
        Neu1->Enabled=false;
        ffnen1->Enabled=false;
        Schliessen1->Enabled=true;
        // Buttons Aufgabe 4.6.3.2:
        Speichern->Enabled=true;
        next->Enabled=false;
        LinSuchen->Enabled=false;
        // Buttons Aufgabe 4.6.8:
        KBForm1->first->Enabled=true;
        KBForm1->last->Enabled =true;
        KBForm1->next->Enabled =true;
        KBForm1->prev->Enabled =true;
      }
  }
}

void EnableButtons();
```

```
void __fastcall TKBForm1::SpeichernClick(TObject *Sender)
{
if (f)
  {
    // Für die Aufgabe 4.3.8 muss der Dateizeiger auf die
    // Position am Dateiende gesetzt werden.
    Kontobewegung k=FormToKB(KBForm1);
    f.write((char*)&k,sizeof(k));
    // Die folgenden 3 Anweisungen sind nicht notwendig.
    // Sie sind aber nützlich, falls etwas schiefgeht.
    if (!f) ShowMessage("Fehler bei write ");
    f.flush();
    if (!f) ShowMessage("Fehler bei flush ");
    EnableButtons();
  }
else ShowMessage("Dateifehler");
}

void __fastcall TKBForm1::Schliessen1Click(TObject *Sender)
{
f.close();
// Menüoptionen:
ffnen1->Enabled=true;
Schliessen1->Enabled=false;
Neu1->Enabled=true;
// Buttons Aufgabe 4.3.3.2:
Speichern->Enabled=false;
next->Enabled=false;
LinSuchen->Enabled=false;
// Buttons Aufgabe 4.3.8:
KBForm1->first->Enabled=false;
KBForm1->last->Enabled =false;
KBForm1->next->Enabled =false;
KBForm1->prev->Enabled =false;
}

// --------------- Aufgabe 2 b) ----------------------------

void __fastcall TKBForm1::ffnen1Click(TObject *Sender)
{
OpenDialog1->InitialDir=TestDir;
OpenDialog1->Filter = "KB-Dateien (*.dat)|*.dat";
OpenDialog1->FileName="kb.dat";

if (OpenDialog1->Execute())
  {
    // Für Aufgabe 4.3.3.2 reicht
    // f.open(OpenDialog1->FileName.c_str(),ios::binary|ios::in);

    //----- Aufgabe 4.3.8.1 a) --------------------------
    // Für Aufgabe 4.3.8.1 ist aber dieser Modus notwendig:
    f.open(OpenDialog1->FileName.c_str(),ios::binary|ios::out|ios::in);
    if (!f)
      {
        ShowMessage("open error: "+OpenDialog1->FileName);
        f.clear();
        f.close();
      }
    else
      {
        // Menüoptionen:
        Neu1->Enabled=false;
        ffnen1->Enabled=false;
        Schliessen1->Enabled=true;
        // Buttons Aufgabe 4.3.3.2:
        Speichern->Enabled=true;
        next->Enabled=true;
```

```
        LinSuchen->Enabled=true;
        // Buttons Aufgabe 4.3.8:
        KBForm1->first->Enabled=true;
        KBForm1->last->Enabled =true;
        KBForm1->next->Enabled =true;
        KBForm1->prev->Enabled =true;

        nextClick(Sender); // zeigt den ersten Datensatz an
      }
  }
}

void ReadAndDisplayData()
{
Kontobewegung k;
f.read((char*)&k,sizeof(k));
if (f)
  KBToForm(k,KBForm1);
else
  {
    if (f.eof())
      {
        ShowMessage("Keine weiteren Daten vorhanden");
        f.clear(); // eof ist kein Fehler
        // Dieses clear ist notwendig, damit 'seekg', 'tellg' usw.
        // anschließend funktionieren. Diese geben sonst -1 zurück.
      }
    else
      ShowMessage("Fehler bei read ");
    KBForm1->next->Enabled=false;
    // ClearKBForm(KBForm1);
  }
}

/* ***************************************************************

Die etwas mühsamen Abfragen des Dateizustands in jeder der
Funktionen ReadAndDisplayData, LinSuchen usw. kann man sich
mit Exception-Handling sparen.

*************************************************************** */

void __fastcall TKBForm1::nextClick(TObject *Sender)
{
ReadAndDisplayData();
EnableButtons();
}

// --------------- Aufgabe 2 c) ----------------------------

// Zu dieser Aufgabe gehört auch die Funktion Zufalls_KB in
// "\Loesungen\CppUtils\KBDecl.cpp"

void __fastcall TKBForm1::Zufallsdatei1Click(TObject *Sender)
{
AnsiString s;
srand(1); // jedesmal dieselben Daten erzeugen
if (InputQuery("Testdaten erzeugen","Anzahl der zu erzeugenden Datensätze", s))
  {
    int n = StrToInt(s);
    OpenDialog1->InitialDir=TestDir;
    OpenDialog1->FileName=Default_FileName;
    OpenDialog1->Filter = Default_Filter;
    if (OpenDialog1->Execute())
      {
        ofstream f(OpenDialog1->FileName.c_str(),ios::binary);
        for (int i=0; i<n; i++)
          {
            Kontobewegung k=Zufalls_KB();
```

```
            f.write((char*)&k,sizeof(k));
          }
        f.close();
      }
  }
}

// --------------- Aufgabe 2 d) ----------------------------

void __fastcall TKBForm1::LinSuchenClick(TObject *Sender)
{
if (f)
  {
    Kontobewegung k;
    bool gefunden=false;
    while ((f.read((char*)&k,sizeof(k))) && (!gefunden))
      {
         AnsiString s=k.NameInhaber;
         if (s.Pos(Edit1->Text)>0)
           {
             KBToForm(k,KBForm1);
             gefunden=true;
           }
      }
    if (!gefunden)
      ShowMessage("Kein Datensatz gefunden");
    if (f.eof())
      {
        f.clear(); // Damit nach einem Zurücksetzen des Dateizeigers
                   // wie in 'firstClick' weitergesucht werden kann.
        LinSuchen->Enabled=false;
      }
  }
else ShowMessage("Dateifehler");
}

// --------------- Aufgabe 4.3.4.2 - Listendruck -----------

int Zeilenzahl;
/***/ Kontobewegung Drucksatz, Altsatz;

const int Zeilen_pro_Seite=72;
const int Druckzeilen_pro_Seite=60;
Currency Summe;    // wird vor jedem Start initialisiert (in VorlGS0)
int Seitenzahl;    //  "
int AnzahlFehler;  //  "
ifstream in;       // Datei, die gelesen wird
ofstream out;      // Ausgabedatei
// Die Namensgebung und Aufteilung in die Funktionen wurde so gewählt,
// dass die Funktionen in späteren Aufgaben wiederverwendet werden können.
// Für diese Aufgaben sind auch die mit /***/ gekennzeichneten Zeilen
// notwendig. Für die aktuelle Aufgabe sind sie nicht notwendig.


#include <iomanip>
// Kontobewegung K;

void Blattkopf(char* fn)
{ // druckt Überschriftszeilen am Anfang einer Seite
Seitenzahl++;
out << "Datei: "<<fn<<"     Ein-/Auszahlungen "
    << "Seite "<<Seitenzahl<<endl;
out<<setw(4)<<"Kto."<<" "
   <<left<<setw(20)<<"Kontoinhaber"
   <<right<<setw(10)<<"Datum"
   <<setw(10)<<"Betrag"<<endl<<endl;
Zeilenzahl = 3;
//***/ Drucksatz = K; // damit der erste Datensatz einer neuen Seite
```

```
                    // vollstaendig (einschließlich der KontoNr) ausgedruckt
wird
}

void Blattvorschub()
{ // druckt Schlußzeile und formfeed
out << "Anzahl der fehlerhaften Datensätze: "<<AnzahlFehler
    << '\f'; // formfeed - Blattvorschub
}

AnsiString MyCurrToStr(Currency c)
{ // CurrToStr druckt c=1 als "1" bzw. c=0,1 als "0.1" und nicht als "0,10"
AnsiString s=CurrToStr(c);
if (s.Pos(",")==0) s=s+",0";
if (s.Pos(",")==s.Length()-1) s=s+"0";
return(s);
}

void Drucke_Zeile_0(Kontobewegung k)
{
out<<setw(4)<<k.KontoNr<<" "
   <<left<<setw(20)<<k.NameInhaber
   <<right<<setw(2)<<k.Datum.Tag<<"."
   <<setw(2)<<k.Datum.Monat<<"."<<setw(2)<<k.Datum.Jahr
   <<" "<<k.BewArt<<setw(8)<<setprecision(8)
   <<MyCurrToStr(k.Betrag).c_str()<<endl;
Zeilenzahl++;
}

void VorlGS0(char* infn, char* outfn)
{ //  Anweisungen, die vor der Verarbeitung der Datensätze erfolgen
in.open(infn,ios::binary);
if (!in)
  ShowMessage("Error open input file: "+AnsiString(infn));
out.open(outfn);
if (!out)
  ShowMessage("Error open output file: "+AnsiString(outfn));
Summe = 0;
Seitenzahl = 0;
AnzahlFehler = 0;
Blattkopf(infn);
}

void Bearbeite_DS_Stufe_0(char* infn, Kontobewegung K)
{ // bearbeitet (hier: druckt) einen Datensatz
if (Zeilenzahl >= Druckzeilen_pro_Seite)
  {
    Blattvorschub();
    Blattkopf(infn);
/***/    Drucksatz.KontoNr = K.KontoNr;
    // der erste DS auf einer neuen Seite soll die Kontonr enthalten
  };
if (K.BewArt == '+')      Summe = Summe + K.Betrag;
else if (K.BewArt == '-') Summe = Summe - K.Betrag;
else AnzahlFehler++;
Drucke_Zeile_0(K);
/***/ Altsatz = K; // Altsatz ist der zuletzt verarbeitete Satz der Gruppe
};

Kontobewegung Lese_naechsten_Satz()
{
Kontobewegung k;
in.read((char*)&k,sizeof(k));
/***/ Drucksatz = k;
/***/ Drucksatz.KontoNr = 0; // innerhalb einer Gruppe soll die Kontonr.
                       // nicht ausgedruckt werden
return k;
}
```

```
void NachlGS0()
{ // Anweisungen, die am Ende ausgeführt werden
out<<setw(38)<<"Gesamtsumme: "<<MyCurrToStr(Summe).c_str()<<endl;
Zeilenzahl = Zeilenzahl+2;
Blattvorschub();
in.close();
out.close();
}

void Listendruck(char* infn, char* outfn)
{
Kontobewegung k;
VorlGS0(infn,outfn);
k=Lese_naechsten_Satz();
while (in)
  {
    Bearbeite_DS_Stufe_0(infn,k);
    k=Lese_naechsten_Satz();
  }
NachlGS0();
};

//#include "Unit2.h"
void __fastcall TKBForm1::Listendruck1Click(TObject *Sender)
{
OpenDialog1->InitialDir=TestDir;
OpenDialog1->FileName="kb.sor";
OpenDialog1->Filter = "sortierte KB-Dateien (*.sor)|*.SOR";
if (OpenDialog1->Execute())
  // Listendruck(TestDir+"\\kb.dat",TestDir+"\\kb.out");
   Listendruck(OpenDialog1->FileName.c_str(),(TestDir+"\\kb.out").c_str());
/*
Form2->RichEdit1->Font->Name="Courier";
Form2->RichEdit1->Lines->LoadFromFile(TestDir+"\\kb.out");
Form2->ShowModal();
*/
}
// --------------- Aufgabe 4.3.6 b)  ------------------------

// Wie bei den Lösungen der Aufgabe 4.3.3.2 wäre es auch bei den folgenden
// Lösungen sinnvoll, nach jeder Dateioperation zu prüfen, ob diese
// erfolgreich war: "if (!f) ...". Damit die Logik aber nicht durch
// zusätzliche Anweisungen verdeckt wird, wurde darauf verzichtet.


// Die folgenden Funktionen beschreiben alle relevanten
// Bedingungen und Operation. Mit ihnen kann die Lösung
// leicht auf andere Container und Dateien übertragen werden.

void EnableButtons()
{
if (f)
  {
    int act_pos=f.tellg();
    f.seekg(0,ios::end);
    // f.clear(); überflüssig: tellg und seekg(0,...) erzeugen nie einen
    //                                                          Fehler
    int end_pos=f.tellg();
    f.seekg(act_pos);
    KBForm1->first->Enabled =(end_pos>0);// Enabled, falls Datei nicht leer
    KBForm1->last->Enabled  =(end_pos>0);// Enabled, falls Datei nicht leer
    KBForm1->next->Enabled  =(act_pos<end_pos);
    KBForm1->prev->Enabled  =(act_pos>=2*sizeof(Kontobewegung));
   }
else if (f.eof()) ShowMessage("f.eof() **** "); // dieser Fall dürfte nie
  // eintreten, da nach dem Lesen in ReadAndDisplayData() bei eof clear
  // aufgerufen wird.
else ShowMessage("Dateifehler: !f"); // Was ging schief?
}
```

```
// --------------- Aufgabe 4.3.6 c)  -----------------------

void __fastcall TKBForm1::prevClick(TObject *Sender)
{
if (f)
  {
    int pos=f.tellg();
    if (pos >= 2*sizeof(Kontobewegung))
      {
        f.seekg(pos-2*sizeof(Kontobewegung));
        ReadAndDisplayData();
       }
    else ClearKBForm(KBForm1);
    EnableButtons();
  }
else ShowMessage("Dateifehler: !f"); // Was ging schief?
}


// --------------- Aufgabe 4.3.6 c)  -----------------------

void __fastcall TKBForm1::firstClick(TObject *Sender)
{
if (f)
  {
    f.seekg(0);
    ReadAndDisplayData();
    EnableButtons();
  }
else ShowMessage("Dateifehler: !f"); // Was ging schief?
}

// --------------- Aufgabe 4.3.6 d) ------------------------

void __fastcall TKBForm1::lastClick(TObject *Sender)
{
if (f)
  {
    f.seekg(-1*sizeof(Kontobewegung),ios::end);
    if (f.tellg() >= 0)
      ReadAndDisplayData();
    EnableButtons();
  }
else ShowMessage("Dateifehler: !f"); // Was ging schief?
}

// --------------- Aufgabe 4.3.6 c) ------------------------

void __fastcall TKBForm1::KorrekturClick(TObject *Sender)
{
if (f)
  {
    Kontobewegung k;
    int pos=f.tellg();
    if (pos >= sizeof(k))
      {
        f.seekg(pos-sizeof(k));
        k=FormToKB(KBForm1);
        f.write((char*)&k,sizeof(k));
      }
    else ShowMessage("Keine Korrektur möglich: Datei leer");
  }
else ShowMessage("Dateifehler: !f"); // Was ging schief?
}


// --------------- Aufgabe 4.3.7.1 -------------------------
```

```
#include <vector>
using std::vector; // or using namespace std;

enum TSortierbegriff {sbKontoNr, sbName,sbDatum,sbKontonrUndDatum}
     Sortierbegriff=sbKontoNr;
//     Sortierbegriff=sbKontonrUndDatum;

bool operator<(const Kontobewegung& k1, const Kontobewegung& k2)
{
if (Sortierbegriff==sbKontoNr)
  return k1.KontoNr < k2.KontoNr;
else if (Sortierbegriff==sbName)
  return k1.NameInhaber < k2.NameInhaber;
else if (Sortierbegriff==sbDatum)
  {
    if (k1.Datum.Jahr!=k2.Datum.Jahr)
      return k1.Datum.Jahr<k2.Datum.Jahr;
    else if (k1.Datum.Monat!=k2.Datum.Monat)
      return k1.Datum.Monat<k2.Datum.Monat;
    else return k1.Datum.Tag<k2.Datum.Tag;
  }
else if (Sortierbegriff==sbKontonrUndDatum)
  {
    if (k1.KontoNr!=k2.KontoNr)
      return k1.KontoNr<k2.KontoNr;
    if (k1.Datum.Jahr!=k2.Datum.Jahr)
      return k1.Datum.Jahr<k2.Datum.Jahr;
    else if (k1.Datum.Monat!=k2.Datum.Monat)
      return k1.Datum.Monat<k2.Datum.Monat;
    else return k1.Datum.Tag<k2.Datum.Tag;
  }
else
  {
    ShowMessage("Unzulässiger Sortierbegriff");
    return false;
  }
}

#include <algorithm>
void SortiereKBDatei(char* infn, char* outfn)
{
vector<Kontobewegung> v;
Kontobewegung k;
ifstream in(infn,ios::binary);
while (in.read((char*)&k,sizeof(k)))
  v.push_back(k);
in.close();

sort(v.begin(),v.end());

ofstream out(outfn,ios::binary);
for (size_t i=0; i<v.size(); i++)
  out.write((char*)&v[i],sizeof(v[i]));
out.close();
}

void __fastcall TKBForm1::SortierenClick(TObject *Sender)
{
OpenDialog1->Filter = "KB-Dateien (*.dat)|*.dat";
OpenDialog1->InitialDir=TestDir;
OpenDialog1->FileName="kb.dat";

if (OpenDialog1->Execute())
  {
    AnsiString outfn=OpenDialog1->FileName+".sor";
    SortiereKBDatei(OpenDialog1->FileName.c_str(), outfn.c_str());
  }
}
```

```
const int n=2; // Anzahl der zu mischenden Dateien
ifstream D[n];
ofstream M;
Kontobewegung k[n];

int MinIndex(int n)
{
int m=0;
for (int i=1; i<n; i++)
  if (!D[m]) m=i; // ***
  else if (D[i] || D[m])
    if (k[i]<k[m]) m=i;
return m;
// zu ***:
/*   if D[0].eof() m=1
     if D[1].eof() m=2
     ...
*/
}

// --------------- Aufgabe 4.3.7.2 -------------------------

// zusätzliche Deklarationen für Folgeprüfung:

Kontobewegung LetzterSatz;
int AnzahlFolgefehler;

void Folgepruefung(int i)
{
if (k[i]<LetzterSatz)
  AnzahlFolgefehler++;
};
// Ende zusätzliche Deklarationen für Folgeprüfung


void Mischen2mitFP(const char* fn1, const char* fn2, const char* mfn)
{
// Änderung für Folgeprüfung:
bool ErsterSatz=true;
AnzahlFolgefehler=0;
// Ende Änderung für Folgeprüfung

D[0].open(fn1,ios::binary);
D[1].open(fn2,ios::binary);
M.open(mfn,ios::binary);
D[0].read((char*)&k[0],sizeof(k[0]));
D[1].read((char*)&k[1],sizeof(k[1]));
while (D[0] || D[1])
  {
    int i=MinIndex(2);
    // Änderung für Folgeprüfung:
    if (ErsterSatz) ErsterSatz=false;
    else Folgepruefung(i);
    LetzterSatz=k[i];
    // Ende Änderung für Folgeprüfung
    M.write((char*)&k[i],sizeof(k[i]));
    D[i].read((char*)&k[i],sizeof(k[i]));
//    if (!D[i]) ShowMessage(IntToStr(i));
  }
D[0].close();
D[1].close();
M.close();
if (AnzahlFolgefehler>0)
  ShowMessage(IntToStr(AnzahlFolgefehler)+" Folgefehler");
}

void __fastcall TKBForm1::MischFPClick(TObject *Sender)
{
```

```
OpenDialog1->Filter = "sortierte KB-Dateien (*.sor)|*.sor";
OpenDialog1->FileName="kb.sor.dat";
OpenDialog1->Title="Erste Mischdatei";
OpenDialog1->InitialDir=TestDir;

if (OpenDialog1->Execute())
  {
    AnsiString fn1=OpenDialog1->FileName;
    OpenDialog1->Title="Zweite Mischdatei";
    if (OpenDialog1->Execute())
      {
      Mischen2mitFP(fn1.c_str(),OpenDialog1-
>FileName.c_str(),(TestDir+"\\M.sor").c_str());
      Listendruck((TestDir+"\\M.sor").c_str(),(TestDir+"\\M.tmp").c_str());
/*
        Form2->RichEdit1->Font->Name="Courier";
        Form2->RichEdit1->Lines->LoadFromFile(TestDir+"\\M.tmp");
        Form2->ShowModal();
*/
      }
  }
}

// --------------- Aufgabe 4.3.7.3 -------------------------

Currency GS1Summe;

void VorlGS1(Kontobewegung k) // Anweisungen, die zu Beginn einer Gruppe
{                              // der Stufe 1 ausgeführt werden
GS1Summe = 0;
Drucksatz.KontoNr = 0; // damit am Anfang einer Gruppe
                       // die Kontonummer nicht ausgedruckt wird
out <<setw(4)<<k.KontoNr<<" "<<k.NameInhaber<<endl;
Zeilenzahl++;
}

bool GW_Stufe_1(Kontobewegung k)
{ // Bedingung fnr eine Gruppenwechsel der Stufe 1
return Altsatz.KontoNr != k.KontoNr;
}

void Setze_Gruppenkennzeichen_fuer_Stufe_1(Kontobewegung k)
{ // damit die Schleife mindestens einmal ausgeführt wird
Altsatz.KontoNr = k.KontoNr;
};

void Drucke_Zeile_1()
{
if (Drucksatz.KontoNr == 0) out<<"      ";
else out<<setw(4)<<Drucksatz.KontoNr<<"  ";
if (Drucksatz.Datum.Jahr == 0) out<<"      "; // 6 Leerzeichen
else out<<setw(4)<<Drucksatz.Datum.Jahr<<"  ";
out<<"   "<<setw(2)<<Drucksatz.Datum.Tag<<"."<<setw(2)<<Drucksatz.Datum.Monat
   <<"     "<<Drucksatz.BewArt<<" "<<setw(8)
   <<MyCurrToStr(Drucksatz.Betrag).c_str()<<endl;
Zeilenzahl++;
};

void Bearbeite_DS_Stufe_1(char* infn, Kontobewegung k)
{
if (Zeilenzahl >= Druckzeilen_pro_Seite)
  {
    Blattvorschub();
    Blattkopf(infn);
    Drucksatz.KontoNr = k.KontoNr;
    // der erste DS auf einer neuen Seite soll die Kontonr enthalten
  };
if (k.BewArt == '+')
  GS1Summe = GS1Summe + k.Betrag;
```

```
else if (k.BewArt == '-')
  GS1Summe = GS1Summe - k.Betrag;
else AnzahlFehler++;
Drucke_Zeile_1();
Altsatz = k; // Altsatz ist der zuletzt verarbeitete Satz der Gruppe
}

void NachlGS1() // Anweisungen, die am Ende einer Gruppe der
{               // Stufe 1  ausgeführt werden
Summe = Summe + GS1Summe;
out<<"      "<<"Summe Kontonummer "<<setw(4)<<Altsatz.KontoNr<<":"
   <<"..........."<<setw(8)<<MyCurrToStr(GS1Summe).c_str()<<endl;
out<<endl;
Zeilenzahl = Zeilenzahl+2;
};

void GW1(char* infn, char* outfn)
{
Kontobewegung K;
VorlGS0(infn,outfn);
K=Lese_naechsten_Satz();
while (in)
  {
    VorlGS1(K);
    Setze_Gruppenkennzeichen_fuer_Stufe_1(K);
    // damit die Schleife mindestens einmal ausgeführt wird
    while (!GW_Stufe_1(K)&& (in))
      {
        Bearbeite_DS_Stufe_1(infn,K);
        K=Lese_naechsten_Satz();
      }
    NachlGS1();
  }
NachlGS0();
}


void __fastcall TKBForm1::GrW1Click(TObject *Sender)
{
OpenDialog1->InitialDir=TestDir;
OpenDialog1->FileName="kb.sor";
OpenDialog1->Filter = "sortierte KB-Dateien (*.sor)|*.SOR";
if (OpenDialog1->Execute())
  {
    GW1(OpenDialog1->FileName.c_str(),(TestDir+"\\gw1.txt").c_str());
  }
}
```

11.4.10 Aufgaben 4.3.3

```
// File: C:\Loesungen_CB2006\Kap_4\4.3\3_4.cpp

// --------------- Aufgaben 4.3.4 --------------------------
// --------------- Aufgabe 1 -------------------------------
#include <fstream>
#include <string>
using namespace std;

void PrintTextlines(ostream& fout, char* infn)
{
ifstream fin(infn);
string s;
while (getline(fin,s))
  {
```

```
    for (string::size_type i=0; i<s.length(); i++)
      if      (s[i]=='ä') s.replace(i,1,"&auml;");
      else if (s[i]=='ö') s.replace(i,1,"&ouml;");
      else if (s[i]=='ü') s.replace(i,1,"&uuml;");
      else if (s[i]=='ß') s.replace(i,1,"&szlig;");
      else if (s[i]=='Ä') s.replace(i,1,"&Auml;");
      else if (s[i]=='Ö') s.replace(i,1,"&ouml;");
      else if (s[i]=='Ü') s.replace(i,1,"&Uuml;");
    fout<<"<BR>"<<s<<endl;
  }
if (!fin.eof())
  ShowMessage("Lesefehler: "+AnsiString(infn));
}

void TextToHTML(char* ifn, char* htmlfn)
{
ofstream fout(htmlfn);
fout<<"<HTML>"<<endl
    <<"<HEAD>"<<endl
      <<"<TITLE>"
        <<ifn
      <<"</TITLE>"<<endl
    <<"</HEAD>"<<endl
    <<"<BODY>"<<endl;
    PrintTextlines(fout, ifn);
    fout<<"</BODY>"<<endl;
    fout<<"</HTML>"<<endl;
fout.close(); // überflüssig
}

#include "..\..\CppUtils\KBDecl.h"

void print_KB(ostream& fout, Kontobewegung k)
{
fout<<"<TR>" // Neue Tabellenzeile
    <<"<TD>"<<k.KontoNr
    <<"<TD>"<<k.NameInhaber
    <<"<TD>"<<k.Datum.Tag<<"."<<k.Datum.Monat<<"."<<k.Datum.Jahr
    <<"<TD>"<<k.BewArt
    <<"<TD>"<<k.Betrag
    <<endl;
}

void PrintTable(ifstream& fin, ostream& fout)
{
fout<<"<TABLE BORDER>"<<endl
    <<"<TH>KontoNr"
    <<"<TH>Name"
    <<"<TH>Datum"
    <<"<TH>Bew-Art"
    <<"<TH>Betrag"
    <<endl;

   Kontobewegung k;
   while(fin.read((char*)&k,sizeof(k)))
     print_KB(fout,k);
   if (!fin.eof())
      ShowMessage("Lesefehler");
fout<<"</TABLE>"<<endl;
}

void KBToHTML(char* ifn, char* htmlfn)
{
ifstream fin(ifn,ios::binary);
ofstream fout(htmlfn);
fout<<"<HTML>"<<endl
    <<"<HEAD>"<<endl
      <<"<TITLE>"
        <<ifn
```

```
      <<"</TITLE>"<<endl
  <<"</HEAD>"<<endl
  <<"<BODY>"<<endl;
    PrintTable(fin, fout);
  fout<<"</BODY>"<<endl;
fout<<"</HTML>"<<endl;
fout.close(); // überflüssig
}


// --------------- Aufgabe 2 -------------------------------

// Die Lösung der Aufgabe Listendruck ist in 'SimpleDBU.cpp' enthalten
```

11.4.11 Aufgabe 4.4

```
// File: C:\Loesungen_CB2006\Kap_4\4.4\AssContU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "AssContU.h"
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------

AnsiString TestDir="c:\\test";

#include "..\..\CppUtils\TestUtils.h"

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
#ifdef _DEBUG
initTestUtils();
bool test_All(); // Prototyp
testUtilsSummary(Memo1, test_All());
#endif

KeySuchen->Enabled=false;
Sortieren->Enabled=false;
// Erzeuge das Verzeichnis, falls es nicht existiert
if (CreateDirectory(TestDir.c_str(),0))
  ShowMessage("directory '"+TestDir+"' created");
}


// --------------- Aufgaben 4.4 ----------------------------

#include "..\..\CppUtils\StringUt.h"

// --------------- Aufgabe 1 -------------------------------

#include <map>
using std::map;

typedef int ValueType;
typedef int KeyType;
typedef std::map<KeyType,ValueType> InfoSys;
InfoSys m;
```

```
bool ValueToKey(KeyType key,ValueType& value)
{
InfoSys::iterator pos=m.find(key);
bool found=(pos!=m.end());
if (found)
    value=pos->second;
return found;
}

void __fastcall TForm1::InfoSystClick(TObject *Sender)
{
CHRTimer t;
ValueType data;
t.Start();
bool found=ValueToKey(StrToInt(Edit1->Text), data);
t.End();
if (found)
  Memo1->Lines->Add(data);
else Memo1->Lines->Add("nicht gefunden!");
Memo1->Lines->Add(t.TimeStr());

/*
Aufgrund des logarithmischen Komplexität von m.find ist wegen

   log(1000) ~ log(2^10) = 10
   log(1000000) = log(1000*1000) ~ log(2^20) = 20

eine Verdoppelung der Suchzeiten zu erwarten. Sie sind aber so
klein, dass mit einer Genauigkeit des Timers von 1 Mikrosekunde
keine Unterschiede beobachtet werden konnten:

  Gemessene Suchzeiten bei 1000 Elementen:

    7,71048793979115E-5 Sekunden
    7,54286863675221E-5
    7,79429759131062E-5
    7,87810724283008E-5
    7,79429759131062E-5

  Gemessene Suchzeiten bei 1000000 Elementen:

    9,7219195762584E-5
    7,54286863675221E-5
    8,12953619738849E-5
    7,79429759131062E-5
    7,12382037915486E-5
*/
}

void fillContainer(int n)
{
m.clear();
for (int i=0;i<n; i++)
  m[i]=i;
}

void __fastcall TForm1::CreateDataClick(TObject *Sender)
{
const int n=StrToInt(Edit1->Text);
CHRTimer t;
fillContainer(n);
t.End();
Memo1->Lines->Add(t.TimeStr("fillContainer "+IntToStr(n)+", "));
// Gemessene Zeiten (in Sekunden):
//    100.000 map-Elemente: 0,17 Sekunden
//    1.000.000 map-Elemente: 1,7 Sekunden
}
```

```
// --------------- Aufgabe 2 -------------------------------

#include <set>
//using std::set;

std::set <int> randSet;

int NewRand(unsigned int Max)
{
if (randSet.size()<Max)
  {
    unsigned int Anzahl_vorher=randSet.size(), i;
    while (randSet.size()==Anzahl_vorher) // Endlosschleife, falls
      {                                   // s.size()>=Max
        i=rand()%Max;
        randSet.insert(i);
      }
    return i;
  }
else return -1; // oder Exception auslösen
}

// Diese Lösungsalternative ist mit jedem Container möglich, der eine
// Funktion wie 'find' hat:
int NewRand1(unsigned int Max)
{
if (randSet.size()<Max)
  {
    unsigned int i;
    do // Endlosschleife, falls RndSet.size()>=Max
      i=rand()%Max;
    while (randSet.find(i)!=randSet.end());
    randSet.insert(i);
    return i;
  }
else return -1;
}

void __fastcall TForm1::NewRandClick(TObject *Sender)
{
int n=10;
for (int i=0; i<n+5; i++)
  Memo1->Lines->Add(::NewRand1(n));
}


// --------------- Aufgabe 3 -------------------------------

#include <set>
#include <fstream>
#include <string>


typedef std::set<string> Woerterbuch;

Woerterbuch WoerterbuchLesen(const char* ifn)
{
Woerterbuch dict;
vector<string>  c;
ifstream fin(ifn);
if (fin)
  {
    string z;
    while (getline(fin,z))
      {
        z=getline_BUG_workaround(z); // siehe StringUt.h
        c=tokenize(z);
        dict.insert(c.begin(), c.end());
      }
```

```
    if(!fin.eof())
      ShowMessage(AnsiString("read error: ")+ifn);
  }
else
  ShowMessage(AnsiString("open error: ")+ifn);
fin.close(); // überflüssig, aber kein Fehler
return dict;
}

#include<algorithm>
void TextPruefen(char* ifn, const Woerterbuch& dict)
{
vector<string>  c;
ifstream fin(ifn);
if (fin)
  {
    string z;
    while (getline(fin,z))
      {
        z=getline_BUG_workaround(z); // siehe StringUt.h
        c=tokenize(z);
        for (vector<string>::iterator i=c.begin();i!=c.end(); i++)
        {
          if (dict.find(*i)==dict.end())
          { // nicht im Wörterbuch enthaltenene Wörter ausgeben:
            Form1->Memo1->Lines->Add(AnsiString(z.c_str())+
                                     ": "+AnsiString(i->c_str()));
          }
        }
      }
    if(!fin.eof())
      ShowMessage(AnsiString("read error: ")+ifn);
  }
else
  ShowMessage(AnsiString("open error: ")+ifn);
fin.close(); // überflüssig, aber kein Fehler
}

void RechtschrPruefung()
{
Woerterbuch dict=
  WoerterbuchLesen("\\Loesungen_BCB2006\\Text\\faust.txt");
Form1->Memo1->Lines->Add("Anzahl der Wörterbucheinträge: "
  +IntToStr(dict.size()));
TextPruefen("\\Loesungen_BCB2006\\Text\\Faust_veraendert.txt",dict);
// In Faust1.txt wurden gegenüber faust.txt einige Worte verändert
}

void __fastcall TForm1::RechtSchrPrClick(TObject *Sender)
{
RechtschrPruefung();
}


// --------------- Aufgabe 4 -------------------------------

typedef std::multimap<string,int> MMType;

MMType MakeXRef(vector<string> v) // Ein XRef-Generator in 13 Zeilen
{ // erzeugt eine Crossreference-Liste aus den Strings eines Vektors
MMType mm;
typedef vector<string> Container;
Container c;
int line=1;
for (vector<string>::iterator i=v.begin();i!=v.end(); i++)
  {
    c=tokenize(*i);
    for (Container::iterator token=c.begin();token!=c.end(); token++)
      mm.insert(make_pair(*token,line));
```

```
    line++;
  }
return mm;
}

MMType MakeXRef(char* ifn) // fn: Dateiname
{ // erzeugt eine Crossreference-Liste aus den Wörtern und ihren Zeilennummern
typedef vector<string> Container;
Container c;
MMType mm;
ifstream fin(ifn);
if (fin)
  {
    string z;
    int line=1;
    while (getline(fin,z))
      {
        z=getline_BUG_workaround(z); // siehe StringUt.h
        c=tokenize(z);
        for (Container::iterator token=c.begin();token!=c.end(); token++)
          mm.insert(make_pair(*token,line));
        line++;
      }
    if(!fin.eof())
      ShowMessage(AnsiString("read error: ")+ifn);
  }
else
  ShowMessage(AnsiString("open error: ")+ifn);
return mm;
}


#include <sstream>
void PrintXRef(MMType mm, char* fn=0)
{ // Schreibt eine XRef-Liste in ein Memo oder in die Datei mit dem Namen fn
typedef MMType::iterator Iterator;
ofstream of;
if (fn!=0) of.open(fn);
for (Iterator i=mm.begin(); i!=mm.end(); )
  {
    std::ostringstream out;
    Iterator first=mm.lower_bound(i->first),
             last=mm.upper_bound(i->first);
    out<<i->first<<": ";
    for (Iterator j=first; j!=last; j++)
      {
        out << "  "<<j->second;
        i++; // Mit jedem gefundenen Wert hochzählen
      };
    // i=last++; // Alternative zu i++ in der Schleife
    if (fn!=0) of<<out.str()<<std::endl;
    else Form1->Memo1->Lines->Add(out.str().c_str());
  }
}


void __fastcall TForm1::XRefClick(TObject *Sender)
{
vector<string> v;
v.push_back("Alle meine Entchen");
v.push_back("schwimmen auf dem See,");
v.push_back("schwimmen auf dem See,");

MMType mm=MakeXRef(v);
PrintXRef(mm);
mm.clear();

mm=MakeXRef("..\\AssContU.cpp");
PrintXRef(mm);
```

```
}


// --------------- Aufgabe 5 -------------------------------

#include "..\..\CppUtils\KBDecl.h"
fstream f;
// using namespace std;

typedef std::multimap<int,int> TKBMap;
TKBMap km;

void MMReadKeys(char* fn)
{
Kontobewegung K;
f.open(fn,ios::in|ios::out|ios::binary);
int pos=f.tellg();
while (f.read((char*)&K,sizeof(K)))
  {
    km.insert(std::make_pair(K.KontoNr,pos));
    pos=f.tellg();
  }
f.clear(); // Damit spätere seek-Zugriffe funktionieren
}

void __fastcall TForm1::KeysLesenClick(TObject *Sender)
{
OpenDialog1->InitialDir=TestDir;
OpenDialog1->FileName="kb.dat";
OpenDialog1->Filter = "KB-Dateien (*.dat)|*.dat";
if (OpenDialog1->Execute())
  {
     MMReadKeys(OpenDialog1->FileName.c_str());
     KeySuchen->Enabled=true;
     Sortieren->Enabled=true;
  }
}

bool MMFind(int Key, Kontobewegung& K)
{
TKBMap::iterator i=km.find(Key);
if (i!=km.end())
  {
    f.seekg(i->second);
    f.read((char*)&K,sizeof(K));
    return true;
  }
else return false;
}

void __fastcall TForm1::KeySuchenClick(TObject *Sender)
{
int Key=StrToInt(Edit1->Text);
Kontobewegung K;
if (MMFind(Key, K)) // hier nur eine ganz "primitive" Ausgabe:
  Memo1->Lines->Add(IntToStr(K.KontoNr)+": "+K.NameInhaber);
else
  Memo1->Lines->Add(Edit1->Text+": nicht gefunden.");
}


void MMSort(char* outfn)
{
Kontobewegung K;
ofstream out(outfn,ios::binary);
TKBMap::iterator i;
for (i=km.begin(); i!=km.end(); i++)
  {
```

```
    f.seekg(i->first);
    f.read((char*)&K,sizeof(K));
    out.write((char*)&K,sizeof(K));
  }
out.close();
}

void __fastcall TForm1::SortierenClick(TObject *Sender)
{
OpenDialog1->InitialDir=TestDir;
OpenDialog1->FileName="kb.sor";
OpenDialog1->Filter = "sortierte KB-Dateien (*.sor)|*.sor";
if (OpenDialog1->Execute())
  {
    MMSort(OpenDialog1->FileName.c_str());
  }
}

bool test_ValueToKey(int n)
{
bool result = true;
fillContainer(n);
if (n>0)
  {
    int data;
    bool found=ValueToKey(-1, data);
    if(!assertEqual(found,false,"test_ValueToKey: "+IntToStr(-1)))
      result=false;
    found=ValueToKey(0, data);
    if(!assertEqual(found,true,"test_ValueToKey: "+IntToStr(0)))
      result=false;
    found=ValueToKey(n-1, data);
    if(!assertEqual(found,true,"test_ValueToKey: "+IntToStr(n-1)))
      result=false;
    found=ValueToKey(n, data);
    if(!assertEqual(found,false,"test_ValueToKey: "+IntToStr(n)))
      result=false;
    found=ValueToKey(n+1, data);
    if(!assertEqual(found,false,"test_ValueToKey: "+IntToStr(n+1)))
      result=false;
  }
return result;
}

void create_dictFile(int lines, int wordsPerLine, const char* fn)
{
ofstream fout(fn);
  for (int i=0; i<lines; i++)
    {
    for (int j=0; j<wordsPerLine; j++)
      fout<<"abc ";
    fout<<std::endl;
  }
}

bool test_readDictionary()
{ // teste die Größe des Wörterbuchs

bool result = true;
const char* fn0=(TestDir+"\\0").c_str();
create_dictFile(0, 0, fn0);
Woerterbuch w0=WoerterbuchLesen(fn0);
if(!assertEqual(w0.size(),0,"readdDictionary w0 "))
  result=false;

const char* fn1=(TestDir+"\\1").c_str();
create_dictFile(1, 1, fn1);
Woerterbuch w1=WoerterbuchLesen(fn1);
if(!assertEqual(w1.size(),1,"readdDictionary w1 "))
```

```
  result=false;

const char* fn2=(TestDir+"\\2").c_str();
create_dictFile(3, 3, fn2);
Woerterbuch w2=WoerterbuchLesen(fn2);
if(!assertEqual(w2.size(),1,"readDictionary w2 "))
  result=false;

return result;
}


#ifdef _DEBUG
bool test_All()
{
bool result=true;
if (!test_ValueToKey(1)) result=false;
if (!test_ValueToKey(100)) result=false;
//if (!test_StringToDate()) result=false;
if (!test_readDictionary()) result=false;
return result;
}
#endif
```

11.4.12 Aufgabe 4.5

```
// File: C:\Loesungen_CB2006\Kap_4\4.5\NumericsU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "NumericsU.h"
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}


// --------------- Aufgabe 4.5.1 ---------------------------

// --------------- Aufgabe 1 -------------------------------

#include <complex>
using std::complex; // or using namespace std;

AnsiString ComplexToStr(complex<long double> c)
{
return FloatToStr(real(c))+"+"+FloatToStr(imag(c))+"i";
}

void __fastcall TForm1::QuadGlClick(TObject *Sender)
{
  complex<double> a(2,3), b(4,-5), c(-7,8), x1, x2;
//  complex<double> a(2,0), b(4,0), c(-7,0), x1, x2;
  x1= (-b+sqrt(b*b - 4.0*a*c))/(2.0*a);
  x2= (-b-sqrt(b*b - 4.0*a*c))/(2.0*a);
  Form1->Memo1->Lines->Add(ComplexToStr(x1));
  Form1->Memo1->Lines->Add(ComplexToStr(x2));
// Probe:
  Form1->Memo1->Lines->Add(ComplexToStr(a*x1*x1+b*x1+c));
```

```
  Form1->Memo1->Lines->Add(ComplexToStr(a*x2*x2+b*x2+c));
}

// Mit typedef lassen sich die Datentypen etwas einfacher schreiben:

typedef complex<double> dcomplex;

AnsiString DComplexToStr(dcomplex c)
{
return FloatToStr(real(c))+"+"+FloatToStr(imag(c))+"i";
}


// --------------- Aufgabe 2 -------------------------------

void __fastcall TForm1::EinheitswurzelnClick(TObject *Sender)
{
using std::sin; // oder: using namespace std;
using std::cos; // oder: using namespace std;

const int n=100;
complex<double> w(cos(2*M_PI/n),sin(2*M_PI/n));
for (int i=0; i<n; i++)
  {
    complex<double> z=pow(w,i); // Einheitswurzel
    // z^n muss etwa 1 sein:
    Memo1->Lines->Add(ComplexToStr(pow(z,n)));
  }
}

#include <valarray>
using std::valarray; // oder: using namespace std;

void erzeugeTestdaten(valarray<double>& x, valarray<double>& y, int n, double a_,
double b_)
{
// Aufgabe 4.5.2, 1
// Aufgabe 4.5.2, 2 a): rnd=0;
// Aufgabe 4.5.2, 2 b): rnd>0;
double rnd=5; // mit positivem Wert ergibt sich eine Streuung
for (int i=0; i<n; i++)
  { // Testwerte erzeugen
    x[i] = i;
    y[i] = a_*x[i] + b_ + rnd*((10-1.0)/2-rand()%10);
  }
}

#include "..\..\cpputils\GraphUtils.h"

void Kreuz(TCanvas* Canvas, int x, int y, int w)
{ // zur grafischen Veranschaulichung der kleinsten Quadrate
Canvas->MoveTo(x-w,y);
Canvas->LineTo(x+w,y);
Canvas->MoveTo(x,y+w);
Canvas->LineTo(x,y-w);
}

void zeichnePunkte(TImage* Image1, valarray<double> x, valarray<double> y)
{ // zur grafischen Veranschaulichung der kleinsten Quadrate
  // zeichne die Punkte als Kreuze
int n=x.size();
int x0=x.min();
int x1=x.max();
int y0=y.min();
int y1=y.max();
Image1->Canvas->Pen->Color = clRed; // Kreuze rot zeichnen
for (int i=0; i<n; i++)
  { // #include "\Loesungen\cpputils\GraphUtils.cpp" vorausgesetzt
    // transformiere (x[i],y[i]) in Bildkoordinaten
```

```
    int px=x_Bildschirm(x[i], x0, x1, Image1->Width);
    int py=y_Bildschirm(y[i], y0, y1, Image1->Height);
    Kreuz(Image1->Canvas, px, py, 5);
  }
}

void drawLine(TImage* Image1,valarray<double> x, valarray<double> y, double a,
double b)
{ // zur grafischen Veranschaulichung der kleinsten Quadrate
int n=x.size();
int x0=x.min();
int x1=x.max();
int y0=y.min();
int y1=y.max();
Image1->Canvas->Pen->Color = clBlack; // Gerade schwarz
// Anfangspunkt der Geraden
int px=x_Bildschirm(0,     x0,x1,Image1->Width);
int py=y_Bildschirm(a*0+b, y0,y1,Image1->Height);
Image1->Canvas->MoveTo(px,py);
// Endpunkt der Geraden
px=x_Bildschirm(n-1,     x0,x1,Image1->Width);
py=y_Bildschirm(a*(n-1)+b, y0,y1,Image1->Height);
Image1->Canvas->LineTo(px,py);
}

// Die folgenden Variablen nur deshalb global, da
// die Teilaufgaben in eigenen Funktionen behandelt
// werden:
double sxy, sx, sy, sxx, ym, xm;
const int n=100; // n=2; n=100; n=10000;
std::valarray<double> x(n), y(n);

void kleinste_Quadrate_mit_valarrays()
{ // Aufgabe a)
  sxy = (x*y).sum();
  sx  = x.sum();
  sy  = y.sum();
      sxx = (x*x).sum();
  xm  = sx/n;
  ym  = sy/n;
}

void kleinste_Quadrate_mit_Arrays()
{ // Aufgabe b)
  sxy=0;
  for (int i=0; i<n; i++) sxy=sxy+x[i]*y[i];
  sx=0;
  for (int i=0; i<n; i++) sx=sx+x[i];
  sy=0;
  for (int i=0; i<n; i++) sy=sy+y[i];
  sxx=0;
  for (int i=0; i<n; i++) sxx=sxx+x[i]*x[i];
  xm=sx/n;
  ym=sy/n;
}

void __fastcall TForm1::kleinsteQuadrateClick(TObject *Sender)
{// Methode der kleinsten Quadrate, Aufgabe 4.5.2, 1.
 // erzeugeTestdaten, zeichnePunkte und drawLine nur zur
 //    grafischen Veranschaulichung
const double a_=2, b_=3;
erzeugeTestdaten(x, y, n, a_, b_);
zeichnePunkte(Image1,x,y);

// Berechne a und b aus den Testdaten:
kleinste_Quadrate_mit_valarrays();
// Kleinste_Quadrate_mit_Arrays();
double a=(n*sxy - sx*sy)/(n*sxx - sx*sx);
double b=ym-a*xm;
```

```
Memo1->Lines->Add("Kleinste Quadrate: ");
Memo1->Lines->Add("a_="+FloatToStr(a_)+ " a="+FloatToStr(a));
Memo1->Lines->Add("b_="+FloatToStr(b_)+ " b="+FloatToStr(b));

drawLine(Image1,x,y,a,b);
}
```

11.5 Lösungen Kapitel 5

11.5.1 Aufgaben 5.2

```
// File: C:\Loesungen_CB2006\Kap_5\5.2\FuncPtrU.cpp

#include <vcl.h>
#pragma hdrstop
#include "FuncPtrU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;


// --------------- Aufgaben 5.2 ----------------------------

#include "..\..\CppUtils\TestUtils.h"

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
#ifdef _DEBUG
bool test_FuncPtr(); // Prototyp
initTestUtils();
testUtilsSummary(Memo1, test_FuncPtr());
#endif
}


// --------------- Aufgabe 1 -------------------------------

typedef double(*real_func)(double) ;


#include <cmath>
#include <limits> // für numeric_limits<double>::epsilon
// Anstelle von NearlyEqual aus Testutils.h kann man auch die
// folgende Funktion verwenden:
bool NearlyEqual_1(double x,double y)
{
const double eps=std::numeric_limits<double>::epsilon()*100;
// epsilon ist der Unterschied zwischen 1 und der nächstgrößeren
// double-Zahl.
// const double eps=1E-12 kann ebenso anstelle von epsilon verwendet werden
return (std::fabs(x-y) <= eps/100000000);
}

// a) Für die Ableitung wird ein Näherungswert berechnet.


double Newton_Nullstelle(double x0, real_func f, int& n_it)
{
n_it = 0; // n_it zählt die Iterationen - zur Vermeidung von Endlosschleifen
```

```
double x = x0;
do { x0=x; // Startwert für die nächste Iteration
     n_it++;
     double h = 0.0000001; // für die Berechnung der Ableitung: möglichst
                 // klein, aber doch so groß, daß f(x+h)<>f(x)
     double Abl = (f(x0 + h) - f(x0))/h;
     //  Abl: Näherungswert für die Ableitung bei x0
     x = x0 - f(x0)/Abl;
   }
//while (!NearlyEqual(x,x0,10) && (n_it <= 100));
while (!NearlyEqual_1(x,x0) && (n_it <= 100));
return x;
};

// b) Die Ableitung wird ebenfalls als Funktion übergeben.

double Newton_Nullstelle_mit_Abl(double x0,
                                 real_func f, real_func Abl, int& n_it)
{
n_it = 0; // i zählt die Iterationen - zur Vermeidung von Endlosschleifen
double x = x0;
do { x0 =x; // Startwert für die nächste Iteration
     n_it++;
     x = x0 - f(x0)/Abl(x0);
   }
// while (!NearlyEqual(x,x0,10) && (n_it <= 100));
while (!NearlyEqual_1(x,x0) && (n_it <= 100));
return x;
}


// c) Verwende zum Testen einfache Polynome, da man bei diesen
//    die Ableitung explizit angeben kann.

const int maxDegr=5;
double p[maxDegr];

double testPolyn(double x)
{
return p[4]*x*x*x*x + p[3]*x*x*x + p[2]*x*x + p[1]*x + p[0];
}

double testPolyn_abl(double x)
{ // die Ableitung von testPolyn
return 4*p[4]*x*x*x + 3*p[3]*x*x + 2*p[2]*x + p[1];
}

AnsiString polystr()
{
return
FloatToStr(p[4])+"*x^4+"+FloatToStr(p[3])+"*x^3+"+FloatToStr(p[2])+"*x^2+"+FloatT
oStr(p[1])+"*x+"+FloatToStr(p[0]);
}

bool test_Newton(int n, bool mit_Abl, bool zeige_Ergebnisse)
{
bool result=true;
int max_it=0; // maximale Anzahl der benötigten Iterationen
for (int i=1; i<n; i++)
  {
    int n_it;
    p[4]=rand()%1000;p[3]=rand()%1000;p[2]=rand()%1000;p[1]=rand()%1000;
    // zufällige Polynomkoeffizienten
    p[0]=-1000000; // Damit das Polynom eine Nullstelle hat, ein großer
                   // negativer Wert
    if (p[1]==0 && p[2]==0 && p[3]==0 && p[4]==0)
      { // Ein konstantes Polynom hat keine Nullstelle
        p[1]=10;
      }
```

```
    double x,y;
    try { // bei Polynomen ohne Nullstelle können Exceptions auftreten
        if (mit_Abl)
          x=Newton_Nullstelle(1, testPolyn, n_it);
        else
          x=Newton_Nullstelle_mit_Abl(1, testPolyn,testPolyn_abl, n_it);
        y=testPolyn(x); // Einsetzen der Nullstelle muss 0 ergeben
        if (!assertEqual(y,0.0,polystr()))
          result=false;
       }
    catch (Exception&)
       {
        if (!assertEqual(true,false,"Exception: "+polystr()))
          result=false;
       }
    if (n_it>max_it)
      max_it=n_it;
    if (zeige_Ergebnisse)
    Form1->Memo1->Lines->Add(polystr()+": "+
              Format("x=%g y=%g n_it=%d ",OPENARRAY(TVarRec,(x,y,n_it))));
  };
Form1->Memo1->Lines->Add("max_it="+IntToStr(max_it));
return result;
}

void __fastcall TForm1::NewtonClick(TObject *Sender)
{
test_Newton(100,true,true);
test_Newton(100,false,true);
}

// --------------- Aufgabe 2 -------------------------------

// a)

double Trapezsumme(real_func f, double a, double b, int n)
{
double s = (f(a) + f(b))/2;
double h = (b-a)/n;
for (int i=1; i<n; i++)
  s = s + f(a+i*h);
return s*h;
}

double Simpsonsumme(real_func f, double a, double b, int n)
{
double s = f(a)-f(b);
double h = (b-a)/n;
for (int i=0; i<=(n/2)-1; i++)
  s = s + 4*f(a+(2*i+1)*h) + 2*f(a+2*(i+1)*h);
return s*h/3;
}

// b)
typedef double (*IntegrationsVerfahren)(real_func f, double a, double b, int n);

double iterate(double a, double b, real_func f, IntegrationsVerfahren I)
{
const double eps = 1e-8;
double x_alt = I(f,a,b,1);
double x     = I(f,a,b,2);
int n = 2;
int Max=std::numeric_limits<int>::max();
while (!NearlyEqual(x,x_alt,10) && (n <Max/2))
// while (!NearlyEqual_1(x,x_alt) && (n <Max/2))
  {
    x_alt = x;
    n = 2*n;
    x = I(f,a,b,n);
```

```
  };
return x;
};

// c) Testprogramm für die Integrationsverfahren:


double testPolyn_Stamm(double x)
{
return p[4]*x*x*x*x*x/5+ p[3]*x*x*x*x/4+ p[2]*x*x*x/3 + p[1]*x*x/2 + p[0]*x;
}

bool test_Integration(int n, bool zeige_Ergebnisse)
{
bool result=true;
for (int i=-n; i<=n; i++)
  for (int j=i+1; j<=n;j++)
    {
      double a=i;
      double b=j;
      //
p[4]=rand()%10;p[3]=rand()%10;p[2]=rand()%10;p[1]=rand()%10;p[0]=rand()%10;
      p[4]=rand();p[3]=rand();p[2]=rand();p[1]=rand();p[0]=rand();
      double t=iterate(a,b,testPolyn,Trapezsumme);
      double s=iterate(a,b,testPolyn,Simpsonsumme);
      double x=testPolyn_Stamm(b) - testPolyn_Stamm(a);
      if (!assertEqual(t,x,polystr()))
        result=false;
      if (!assertEqual(s,x,polystr()))
        result=false;
    if (zeige_Ergebnisse)
      Form1->Memo1->Lines->Add(polystr()+Format(" t=%g  s=%g  exact=%g",
        OPENARRAY(TVarRec,(t,s,x))));
    }
return result;
};

double f(double x)
{
return std::sqrt(1-x*x);
}

void __fastcall TForm1::NumIntClick(TObject *Sender)
{
double t=Trapezsumme(f,0,1,10000);
double s=Simpsonsumme(f,0,1,10000);
Memo1->Lines->Add("T="+FloatToStr(4*t));
Memo1->Lines->Add("S="+FloatToStr(4*s));
test_Integration(10,true);
}


// --------------- Aufgabe 3 -------------------------------

void f1(void)
{};

int f2(int,int)
{return 0;};

double f3(double)
{return 0;};

char* f4(char*)
{return "";};

// a1)
typedef void (*F1)();
typedef int (*F2)(int,int);
```

```
typedef double (*F3)(double);
typedef char* (*F4)(char*);

const int MaxF=10;
F1 A1[MaxF];
F2 A2[MaxF];
F3 A3[MaxF];
F4 A4[MaxF];

// a2)
void   (*a1[MaxF])(void);
int    (*a2[MaxF])(int,int);
double (*a3[MaxF])(double);
char*  (*a4[MaxF])(char*);

// b)
void test()
{
a1[0]=f1;
a1[0]();
A1[0]=f1;
A1[0]();

a2[0]=f2;
a2[0](1,1);
A2[0]=f2;
A2[0](1,1);

a3[0]=f3;
a3[0](1);
A3[0]=f3;
A3[0](1);

a4[0]=f4;
a4[0]("");
A4[0]=f4;
A4[0]("");
}

// c)
#include <typeinfo>

void __fastcall TForm1::FktTypenClick(TObject *Sender)
{
test();
Form1->Memo1->Lines->Add(typeid(F1).name());
Form1->Memo1->Lines->Add(typeid(F2).name());
Form1->Memo1->Lines->Add(typeid(F3).name());
Form1->Memo1->Lines->Add(typeid(F4).name());
}


bool test_FuncPtr()
{
bool result=true;
if (!test_Newton(1000,true,false))
  result =false;
if (!test_Newton(1000,false,false))
  result =false;
if (!test_Integration(10,false))
  result =false;
return result;
}
```

11.5.2 Aufgaben 5.3

```
// File: C:\Loesungen_CB2006\Kap_5\5.3\RecursionU.cpp

#include <vcl.h>
#pragma hdrstop
#include "RecursionU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
AnsiString dir="c:\\test";

// Erzeuge das Verzeichnis, falls es nicht existiert
if (CreateDirectory(dir.c_str(),0))
  ShowMessage("directory '"+dir+"' created");
// Voreinstellungen für Aufgabe 4:
fnMask->Text="*.*";
fnMask->Text="*";
DriveComboBox1->Drive='C';
}

#include "..\..\CppUtils\TestUtils.h"

// --------------- Aufgaben 5.3.1 --------------------------

// --------------- Aufgabe 1 -------------------------------

// Rekursive Funktionen

// a)
int rek_Fak(int n)
{ // Fakultät rekursiv berechnen
if (n<=0) return 1;
else return n*rek_Fak(n-1);
}

// b)
int rek_Fib(int n)
{ // Fibonacci rekursiv berechnen
if (n<=0) return 0;
else if (n==1) return 1;
else return rek_Fib(n-1) + rek_Fib(n-2);
}

// c)
int rek_ggT(int m, int n)
{ // ggT rekursiv berechnen
if (n==0) return m;
else return rek_ggT(n, m%n);
}

// a)
int it_Fak(int n)
{
int f=1;
for (int i=2; i<=n; i++)
  f = f*i;
return f;
}

// b)
```

```
int it_Fib(int n)
{ // Fibonacci iterativ berechnen
int fib=0,f0=0,f1=1;
for (int i=0; i<n; i++)
  {
    fib = f0 + f1;
    f1 = f0;
    f0 = fib;
  }
return fib;
}

// c)
int it_ggT(int x, int y)
{
// x=x0, y=y0, ggT(x,y) = ggT(x0,y0)
while (y != 0) //            Ablaufprot. für den Schleifenkörper
  {               //                        x        y        r
   // ggT(x,y)=ggT(x0,y0)                  x0        y0        r0
   int r=x % y;//                                        x0 mod y0    +
   x = y;       //                                y0
   y = r;       //                                 x0 mod y0
   // ggT(x,y) = ggT(y0,x0 mod y0) = ggT(x0,y0)
   // ggT(x,y)=ggT(x0,y0)
  }
// ggT(x,y) = ggT(x0,y0) und y=0
return x;
};

void test_rek()
{
//RichEdit1->Lines->Add("No news are good news");

for (int i=1; i<50; i++)
  if (rek_Fak(i)!=it_Fak(i))
    Form1->Memo1->Lines->Add(IntToStr(i));

for (int i=1; i<50; i++)
  for (int j=1; j<50; j++)
    if (rek_ggT(i,j)!=it_ggT(i,j))
      Form1->Memo1->Lines->Add(IntToStr(i)+" "+IntToStr(j));

for (int i=1; i<30; i++)
  if (rek_Fib(i)!=it_Fib(i))
    Form1->Memo1->Lines->Add(IntToStr(i));
}


bool test_rek_1()
{
bool result=true;
//RichEdit1->Lines->Add("No news are good news");
for (int i=1; i<50; i++)
  for (int j=1; j<50; j++)
    if (!assertEqual(rek_ggT(i,j),it_ggT(i,j),
                     "ggT("+IntToStr(i)+","+IntToStr(j)+")" ))
      Form1->Memo1->Lines->Add(IntToStr(i)+" "+IntToStr(j));

for (int i=1; i<50; i++)
  if (rek_Fak(i)!=it_Fak(i))
    Form1->Memo1->Lines->Add(IntToStr(i));

for (int i=1; i<30; i++)
  if (rek_Fib(i)!=it_Fib(i))
    Form1->Memo1->Lines->Add(IntToStr(i));
}


void __fastcall TForm1::Aufg1Click(TObject *Sender)
```

```
{
test_rek();
}


// --------------- Aufgabe 2 -------------------------------

int ack(int n, int m)
{
if (n==0) return m+1;
else if (m==0) return ack(n-1,1);
else return ack(n-1,ack(n,m-1));
}

#include <cmath>
void __fastcall TForm1::Aufg2Click(TObject *Sender)
{
/*
    ack(0,m) = m+1
    ack(1,m) = m+2
    ack(2,m) = 2*m+3
    ack(3,m) = 2m+3+1
*/
RichEdit1->Lines->Add("No news are good news");

for (int m=0; m<50; m++)
  if (ack(0,m)!=m+1)
    RichEdit1->Lines->Add(m);

for (int m=0; m<50; m++)
  if (ack(1,m)!=m+2)
    RichEdit1->Lines->Add(m);

for (int m=0; m<50; m++)
  if (ack(2,m)!=2*m+3)
    RichEdit1->Lines->Add(m);

// für m>10 dauert das recht lange:
for (int m=0; m<10; m++)
  {
    int a=ack(3,m);
    int e=std::pow(double(2),m+3)-3;
    if (a!=e)
      RichEdit1->Lines->Add(IntToStr(a)+" "+e);
  }
}


//-----------   Koch'sche Kurven -------------------------------------
int round(double x)
{
return int(x+0.5);
}

#include <cmath>
void Koch(TCanvas* C, int n, double leftx, double lefty,
                             double rightx, double righty)
{
const double r = std::sqrt(3.0)/6;//ca. r=0.29;
if (n<=1)
  {
    C->MoveTo(round(leftx), round(lefty));
    C->LineTo(round(rightx), round(righty));
  }
else
  {
    Koch(C, n-1, leftx, lefty,
          (rightx + 2.0*leftx)/3.0,(righty + 2.0*lefty)/3.0);
    Koch(C,n-1,
```

```
            (rightx + 2.0*leftx)/3.0,(righty + 2.0*lefty)/3.0,
            0.5*(rightx+leftx) - r*(lefty-righty),
            0.5*(righty + lefty) + r*(leftx-rightx));
    Koch(C,n-1,
            0.5*(rightx + leftx) - r*(lefty-righty),
            0.5*(righty + lefty) + r*(leftx-rightx),
           (2.0*rightx + leftx)/3.0,(2.0*righty + lefty)/3.0);
    Koch(C,n-1,
           (2.0*rightx + leftx)/3.0,(2.0*righty + lefty)/3.0,
           rightx, righty);
  }
}

void ClearImages(int W, int H)
{
Form1->Image1->Canvas->Brush->Color=clWhite;// FillRect(TRect(0,H,W,H));
Form1->Image2->Canvas->Brush->Color=clWhite;// FillRect(TRect(0,H,W,H));
Form1->Image3->Canvas->Brush->Color=clWhite;// FillRect(TRect(0,H,W,H));
Form1->Image4->Canvas->Brush->Color=clWhite;// FillRect(TRect(0,H,W,H));
Form1->Image1->Canvas->FillRect(TRect(0,0,W,H));
Form1->Image2->Canvas->FillRect(TRect(0,0,W,H));
Form1->Image3->Canvas->FillRect(TRect(0,0,W,H));
Form1->Image4->Canvas->FillRect(TRect(0,0,W,H));
}

int level=1;

void __fastcall TForm1::Koch_Click(TObject *Sender)
{
Form1->Caption="Koch'sche Kurven, n=2, 3, 4, 5";
int W=Image1->ClientWidth-1;
int H=Image1->ClientHeight-3;
ClearImages(W, H+2);
Koch(Image1->Canvas,2,0,H,W,H);
Koch(Image2->Canvas,3,0,H,W,H);
Koch(Image3->Canvas,4,0,H,W,H);
Koch(Image4->Canvas,5,0,H,W,H);
}

// --------------- Aufgabe 5.3.4 ---------------------------

void Dreieck(TCanvas* C, int n, double x, double y, double L)
{
const double r = std::sqrt(3.0);
if (n<=1)
  { // zeichnet ein gleichseitiges Dreieck im Punkt (x,y)
    // mit der Seitenlänge L
    TPoint Eckpunkte[4];
    Eckpunkte[0]=Point(x,y);
    Eckpunkte[1]=Point(x+L,y);
    Eckpunkte[2]=Point(x+L/2,y-L*r/2);
    Eckpunkte[3]=Point(x,y);
    C->Brush->Color = clBlack;
    C->Polygon(Eckpunkte, 3);
  }
else
  {
    Dreieck(C, n-1, x, y, L/2);
    Dreieck(C, n-1, x+L/2, y, L/2);
    Dreieck(C, n-1, x+L/4, y-L*r/4, L/2);
  }
}

void __fastcall TForm1::rekDreiecke_Click(TObject *Sender)
{
Form1->Caption = "Rekursive Dreiecke , n=1, 2, 5, 9";
int W=Image1->ClientWidth-1;
int H=Image1->ClientHeight-3;
ClearImages(W,H+2);
```

```
Dreieck(Form1->Image1->Canvas,1,0,H,H);
Dreieck(Form1->Image2->Canvas,2,0,H,H);
Dreieck(Form1->Image3->Canvas,5,0,H,H);
Dreieck(Form1->Image4->Canvas,9,0,H,H);
}


// --------------- Aufgaben 5.3.6 --------------------------

// --------------- Aufgabe 1 -------------------------------

typedef AnsiString dataType ; // Datentyp der Listendaten
typedef AnsiString keyType ; // Datentyp der Schlüsselwerte

#include "..\..\kap_3\3.12\TreeDefs.h"

// a)
void insertBinTreenode_rec(Treenode*& n, keyType key, dataType data)
{
if (n==0) n = newTreenode(key,data,0,0);
else if (key < n->key) insertBinTreenode_rec(n->left,key,data);
else insertBinTreenode_rec(n->right,key,data);
}

// b)
void processTreenode(Treenode* n)
{
Form1->Memo2->Lines->Add("key:"+n->key+" data:"+n->data);
};

void traverseTree_rec(Treenode* n)
{
if (n!=0)
  {
    traverseTree_rec(n->left);
    processTreenode(n);
    traverseTree_rec(n->right);
  }
}

// c)
Treenode* searchBinTree_rec(Treenode* n, const dataType& x)
{
if (n==0) return 0;
else if (x==n->key) return n;
else if (x<n->key) return searchBinTree_rec(n->left,x);
else return searchBinTree_rec(n->right,x);
}

Treenode* t=0;

void __fastcall TForm1::TreeInsertClick(TObject *Sender)
{
insertBinTreenode_rec(t, Edit1->Text, Edit2->Text);
}

void __fastcall TForm1::ShowTreeClick(TObject *Sender)
{
traverseTree_rec(t);
}

void __fastcall TForm1::SearchTreeClick(TObject *Sender)
{
Treenode* n=searchBinTree_rec(t,Edit1->Text);
if (n!=0)
  Form1->Memo2->Lines->Add(n->key+" gefunden, Daten: "+n->data);
else
  Form1->Memo2->Lines->Add(Edit1->Text+" nicht gefunden");
}
```

```
// --------------- Aufgabe 2 -------------------------------

/* Eigentlich wäre hier der Datentyp int für die Listendaten
   angemessen:

    typedef int dataType ; // Datentyp der Listendaten
    typedef AnsiString keyType ; // Datentyp der Schlüsselwerte

  Damit die Definitionen von Aufgabe 3.12 verwendet werden
  können, wird der Datentyp AnsiString verwendet.
*/

#define VCL_H  // Das ist für assertEqual in
               // StringUt.h notwendig.

struct Listnode {
  dataType data;
  Listnode* next;
};

struct TreenodeWithList {
  keyType key;
  Listnode *first,
           *last;
  TreenodeWithList *left,
           *right;
};

Listnode* newListnode(const dataType& data, Listnode* next)
{ // Wie in Abschnitt 3.12.12
Listnode* n=new Listnode;
n->data = data;
n->next = next;
return n;
}

void insertLastListnode(Listnode*& first, Listnode*& last, const dataType& data)
{ // Wie in Abschnitt 3.12.12
Listnode* n=newListnode(data,0);
if (last==0) first=n;
else last->next=n;
last=n;
}

void insertTreenode(TreenodeWithList*& n, keyType key,dataType data)
{
if (n==0)
  {
    n = new TreenodeWithList;
    n->key   = key;
    n->left  = 0;
    n->right = 0;
    n->first = 0;
    n->last  = 0;
    // Erzeuge einen neuen Listenknoten, auf den n->first und
    // n->last anschließend zeigen:
    insertLastListnode(n->first, n->last,data);
  }
else if (key < n->key) insertTreenode(n->left,key,data);
else if (key > n->key) insertTreenode(n->right,key,data);
else     insertLastListnode(n->first, n->last,data);
}

TreenodeWithList* W=0;
#include "..\..\CppUtils\StringUt.h"
#include <fstream>

void readFile(char* ifn)
```

```
{
ifstream fin(ifn);
int LineNr=0;
string z; // string und nicht AnsiString, da getline einen String
          // als Parameter hat
while (getline(fin,z))
  {
    LineNr++;
    vector<string> v=tokenize(z);
    for (vector<string>::iterator i=v.begin(); i!=v.end(); i++)
      insertTreenode(W, i->c_str(), LineNr);
}
fin.close(); // überflüssig
}

void printNode(TreenodeWithList* n)
{
AnsiString s=n->key+": ";
for (Listnode* i=n->first; i!=0; i=i->next)
  s += i->data+"  ";
Form1->Memo2->Lines->Add(s);
};

void printTree(TreenodeWithList* n)
{ // wie traverse_tree
if (n!=0)
  {
    printTree(n->left);
    printNode(n);
    printTree(n->right);
  }
}

void GenerateConcordanceList(AnsiString infn)
{
readFile(infn.c_str());
printTree(W);
}

void __fastcall TForm1::KonkordanzClick(TObject *Sender)
{
OpenDialog1->Filter = "Textdateien|*.TXT";
OpenDialog1->InitialDir = "..\\..\\..\\text";
if (OpenDialog1->Execute())
  GenerateConcordanceList(OpenDialog1->FileName);
}


// --------------- Aufgabe 5.3.7 ---------------------------

// --------------- Aufgabe 1 -------------------------------

// a)
void show(AnsiString s)
{
static int i=0; // sehr einfach, funktioniert nur beim ersten Aufruf
i++;
Form1->RichEdit1->Lines->Add(IntToStr(i)+" "+s);
}

int searchSubdirs_all(const AnsiString& Path)
{
int n=0;
AnsiString Mask="*";
AnsiString FileName=Path+Mask;
WIN32_FIND_DATA FindFileData;
HANDLE h=FindFirstFile(
             FileName.c_str(), // Zeiger auf den Dateinamen
             &FindFileData);   // Zeiger auf struct
```

```
if (h!=INVALID_HANDLE_VALUE)
  {
    int found=1;
    while (found )
      {
        if ((FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)!=0)
          { // Unterverzeichnis
            if (strcmp(FindFileData.cFileName,".")==0)
              ; // "." ignorieren
            else if (strcmp(FindFileData.cFileName,"..")==0)
              ; // ".." ignorieren
            else // Verzeichnis, rekursiv durchsuchen
              {
                show(Path+FindFileData.cFileName);
                n +=searchSubdirs_all(Path+FindFileData.cFileName+"\\");
              }
          }
        else
          {
            show(Path+FindFileData.cFileName);
            n++;
          }
        found=FindNextFile(h,&FindFileData);
      }
    FindClose(h);
  }
return n;
};

void __fastcall TForm1::Button_aClick(TObject *Sender)
{
RichEdit1->Lines->BeginUpdate();
int n=searchSubdirs_all("\\Loesungen_CB2006\\");
// oder ein anderes Verzeichnis auf ihrem Rechner
Form1->RichEdit1->Lines->Add("Anzahl gefundener Dateien: "+IntToStr(n));
RichEdit1->Lines->EndUpdate();
}

// b)

int searchSubdirs(const AnsiString& Path,const AnsiString& Mask)
{
int n=0;

// Erster Schritt: Suche nach allen Unterverzeichnissen (mit Mask="*")
WIN32_FIND_DATA FindFileData;
AnsiString MaskForAllFiles="*";
AnsiString FileName=Path+MaskForAllFiles;
HANDLE h=FindFirstFile(
             FileName.c_str(), // Zeiger auf den Dateinamen
             &FindFileData);   // Zeiger auf struct
if (h!=INVALID_HANDLE_VALUE)
  {
    int found=1;
    while (found)
      {
        if ((FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)!=0)
          { // Unterverzeichnis
            if (strcmp(FindFileData.cFileName,".")==0)
              ; // "." ignorieren
            else if (strcmp(FindFileData.cFileName,"..")==0)
              ; // ".." ignorieren
            else // Verzeichnis, rekursiv durchsuchen
              {
                // Rekursion:
                n +=searchSubdirs(Path+FindFileData.cFileName+"\\",Mask);
              }
          }
        else
```

```
          {
            // Im ersten Schritt nur Unterverzeichnisse gesucht
          }
        found=FindNextFile(h,&FindFileData);
      }
    FindClose(h);
  }

// Zweiter Schritt: Suche nach allen Dateien mit der als Parameter
//                  übergebenen Maske
FileName=Path+Mask;

h=FindFirstFile(
             FileName.c_str(), // Zeiger auf den Dateinamen
             &FindFileData);   // Zeiger auf struct
if (h!=INVALID_HANDLE_VALUE)
  {
    int found=1;
    while (found)
      {
        if ((FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)!=0)
          { // Unterverzeichnis
            // Im zweiten Schritt Unterverzeichnisse ignorieren
            // und nur Dateien bearbeiten.
          }
        else
          {
            show(Path+FindFileData.cFileName);
            n++;
          }
        found=FindNextFile(h,&FindFileData);
      }
    FindClose(h);
  }
return n;
};

void __fastcall TForm1::Button_bClick(TObject *Sender)
{
AnsiString Path="\\Loesungen_CB2006\\"; // oder ein anderes
                           // Verzeichnis auf ihrem Rechner
AnsiString Mask="*.cpp";
RichEdit1->Lines->BeginUpdate();
int n=searchSubdirs(Path,Mask);
Form1->RichEdit1->Lines->Add("Anzahl gefundener Dateien: "+IntToStr(n));
RichEdit1->Lines->EndUpdate();
}

// c)
void __fastcall TForm1::RecurseDirClick(TObject *Sender)
{
AnsiString Path=AnsiString(DriveComboBox1->Drive)+":\\";
AnsiString Mask=fnMask->Text;
RichEdit1->Lines->BeginUpdate();
int n=searchSubdirs(Path,Mask);
Form1->RichEdit1->Lines->Add("Anzahl gefundener Dateien: "+IntToStr(n));
RichEdit1->Lines->EndUpdate();
}

// --------------- Aufgabe 2 -------------------------------

// Die Lösung ist das Projekt DupFiles
```

11.5.3 Aufgaben 5.3 (calcEx)

```
// File: C:\Loesungen_CB2006\Kap_5\5.3\calcExU.cpp

#include <vcl.h>
#pragma hdrstop
#include "calcExU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------


#include "..\..\CppUtils\Testutils.h"
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
Edit1->Text="";
Memo1->Clear();
#ifdef _DEBUG
bool test_All(); // Prototyp
initTestUtils();
testUtilsSummary(Memo1, test_All());
#endif

InfoLabel->WordWrap=true;
InfoLabel->AutoSize=false;
InfoLabel->Caption=
"In das Eingabefenster kann ein arithmetischer Ausdruck eingegeben "
"werden, der aus Zahlen, den Operatoren +, -, * und /, Klammern () "
"sowie aus sin, cos und pi besteht. Dieser wird nach dem Drücken der "
"Enter-Taste oder dem rechne-Button ausgewertet und im Ergebnisfenster angezeigt.
\r\n"
"Beispiele: 1.23 + 4*(5+6-7) \r\n"
"           (12.34 + 5*6.72)/1.19\r\n"
"           3*sin(4*pi)+5*(cos(2 + 3 - 5))";

}
//---------------------------------------------------------------------

void skipWhitespace(AnsiString s, int& i)
{ // lässt sich auf Tabs usw. erweitern
while ((i <= s.Length()) && (s[i]==' ')) i++;
} // (s[i] != ' ') or "i > s.Length()"

bool isOneOf(char c, AnsiString Str)
{ // true, falls c in Str enthalten ist
return Str.Pos(c)>0;
// isOneOf('a',"abc");  // true
// isOneOf('a',"bcd");  // false
}

double readNumber(AnsiString s, int& i)
{
AnsiString sep=DecimalSeparator;
AnsiString t;
while ((i<=s.Length())&& isOneOf(s[i],"0123456789"+sep))
  t = t + s[i++]; // s[i] zuweisen, anschließend i++
// (s[i] keine Ziffer) or (i > s.Length())
skipWhitespace(s,i);

try {return StrToFloat(t);}
catch(...)
  {
    ShowMessage("Unerwartetes Zeichen: '"+AnsiString(s[i])+"'");
  }
```

```
}

char nextChar(AnsiString s, int& i)
{ // returns the char at the current position i and advances i
char result=0;
if (i <= s.Length())
  result= s[i];
i++;
skipWhitespace(s,i);
return result;
}


// Prototypen für die rekursiven Aufrufe
double multiplicativeExpr(AnsiString s, int& i);
double primaryExpr(AnsiString s, int& i);

void errormessage(int i, AnsiString msg, AnsiString s)
{
AnsiString m=s;
m.Insert("<---",i+1);
if (msg==("") ShowMessage("Syntaxfehler bei Pos. "+IntToStr(i)+"\n"+m);
else ShowMessage(msg+IntToStr(i)+"\n"+m);
}


double additiveExpr(AnsiString s, int& i)
{
double result = multiplicativeExpr(s,i);
while ( (i<=s.Length()) && (isOneOf(s[i],"+-")) )
  {
    char op = nextChar(s,i);
    if (op=='+')
      result = result + multiplicativeExpr(s,i);
    else if (op=='-')
      result = result - multiplicativeExpr(s,i);
  }
return result;
}

double multiplicativeExpr(AnsiString s, int& i)
{
double result = primaryExpr(s,i);
while ( (i <= s.Length()) && (isOneOf(s[i],"*/")) )
  {
    char op = nextChar(s,i);
    if (op=='*')
      result = result * primaryExpr(s,i);
    else if (op=='/')
      result = result / primaryExpr(s,i);
  }
return result;
}

//----- Aufgabe 6.4.3. ------------------------------------------------

#include <cctype>
AnsiString readIdentifier(AnsiString s, int& i)
{ // ähnlich wie die Syntax für einen identifier
using namespace std;
AnsiString result;
if ((i<=s.Length())&& (isalpha(s[i]) || (s[i]=='_') ))
  result = result + UpCase(nextChar(s,i));
while ((i<=s.Length())&& (isalpha(s[i]) || isdigit(s[i]) || (s[i]=='_') ))
  {
    result = result + UpCase(nextChar(s,i));
  }
skipWhitespace(s,i);
return result;
```

```
}

#include <cmath>

// Ende Lösung Aufgabe 3


double primaryExpr(AnsiString s, int& i)
{
using namespace std;
double result;
AnsiString sep=DecimalSeparator;
if (isOneOf(s[i],"1234567890"+sep))
  result=readNumber(s,i);
else if (s[i]=='(')
  {
    nextChar(s,i); // skip '('
    result = additiveExpr(s,i);
    if (!nextChar(s,i)== ')')
      errormessage(i,"')' expected: ",s);
  }
else if (s[i]=='-')
  {
    nextChar(s,i); // skip '-'
    result = -primaryExpr(s,i);
  }
// Lösung von Aufgabe 3.
else if (isalpha(s[i]))
  {
    AnsiString id=readIdentifier(s,i);
    if (id=="PI") result=M_PI;
    else if(id=="PIQ") result=M_PI*M_PI;
    else if (s[i]=='(')
     {
       nextChar(s,i); // skip '('
       if (id=="SIN") result = sin(additiveExpr(s,i));
       else if (id=="COS") result = cos(additiveExpr(s,i));
       else
         {
           result = 0;
           errormessage(i,"invalid function: ",s);
         }
       if (!nextChar(s,i)== ')')
         errormessage(i,"')' expected: ",s);
     }
    else errormessage(i,"'(' expected: ",s);
  }
// Ende Lösung von Aufgabe 3.
else errormessage(i,"",s);
return result;
}

double expression(AnsiString t)
{
AnsiString s=DecimalSeparator;
for (int j=1; j<=t.Length();j++)
  if (t[j]=='.'||t[j]==',')
    t[j]=DecimalSeparator;
int i=1;
skipWhitespace(t,i);
return additiveExpr(t,i);
}


void __fastcall TForm1::rechneClick(TObject *Sender)
{
if (Edit1->Text.Trim()!="")
  {
    double result=expression(Edit1->Text);
```

```
    Memo1->Lines->Add(Edit1->Text+" = "+result);
  }
}
//---------------------------------------------------------------------


void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
{
if (Key==13) rechneClick(Sender);
}
//---------------------------------------------------------------------



bool test_String(AnsiString s, double erg)
{
double r = expression(s);
return assertEqual(r,erg,s);
//  if (erg != r) Form1->Memo1->Lines->Add(s+"="+FloatToStr(r)+
//" nicht: "+FloatToStr(erg));
}

bool teste_parser1()
{
bool result=true;
if (!test_String("1+(2 + 3 )   ",   6)) result=false;
if (!test_String("1+(2 - 3 )   ",   0)) result=false;
if (!test_String("1+(2 * 3 )   ",   7)) result=false;
if (!test_String("1+(6 / 3 )   ",   3)) result=false;

if (!test_String("1-(2 + 3 )   ",   -4)) result=false;
if (!test_String("1-(2 - 3 )   ",   2)) result=false;
if (!test_String("1-(2 * 3 )   ",   -5)) result=false;
if (!test_String("1-(6 / 3 )   ",   -1)) result=false;

if (!test_String("3*(2 + 3 )   ",   15)) result=false;
if (!test_String("3*(2 - 3 )   ",   -3)) result=false;
if (!test_String("3*(2 * 3 )   ",   18)) result=false;
if (!test_String("3*(6 / 3 )   ",   6)) result=false;

if (!test_String("10/(2 + 3 )   ",   2)) result=false;
if (!test_String("10/(2 - 3 )   ",   -10)) result=false;
if (!test_String("12/(2 * 3 )   ",   2)) result=false;
if (!test_String("10/(6 / 3 )   ",   5)) result=false;
return result;
}

bool teste_parser()
{
bool result=true;
if (!test_String(" 1+2/(1 + 1 )   ",   2)) result=false;
if (!test_String("4 - 5 + (10 / 10)",  0)) result=false;
if (!test_String(" 1/sin(1) ",     1.188395105778)) result=false;
if (!test_String("sin(3)*sin(3)+cos(3)*cos(3)   ",1)) result=false;
if   (!test_String("   1.05+(1.88   +   341.01   )/1.07        ",321.507943925234))
result=false;
if (!test_String(" 1+2/(1 + 1 )   ",2)) result=false;
if (!test_String(" -(3 + 4 )   ",-7)) result=false;
if (!test_String(" 10 -(3 + 4 )   ",3)) result=false;
if (!test_String(" 1   ",1)) result=false;
if (!test_String(" -1   ",-1)) result=false;
if (!test_String(" 1,1+  2,2",3.3)) result=false;
if (!test_String("1,1+2,2+3,3",6.6 )) result=false;
if (!test_String("3*4 + 5*6  ",42)) result=false;
if (!test_String("(4 - 5) *10 / 10",-1)) result=false;
if (!test_String("4 -(5 + 10) / 10",2.5)) result=false;
if (!test_String("4 - 5 + -10 / 10",-2)) result=false;
if (!test_String("4 - 5 + (10) / 10)",0)) result=false;
if (!test_String("4 - 5 + (10 / 10)",0)) result=false;
```

```
if (!test_String(" sin(1) ",0.841470984807897)) result=false;
if (!test_String(" 1/sin(1) ",1.18839510577812)) result=false;
if (!test_String(" sin(1+sin(1)) ",0.963590724541833)) result=false;
if (!test_String(" sin(2) ",0.909297426825682)) result=false;
if (!test_String(" sin(1)+sin(2) ",1.75076841163358)) result=false;
if (!test_String(" 1/(sin(1)+sin(2)) ",0.571177771631678)) result=false;
if (!test_String(" 2+1/(sin(1)+sin(2)) ",2.57117777163168)) result=false;
if (!test_String("sin(3)*sin(3)+cos(3)*cos(3)   ",1)) result=false;
if (!test_String(" 1+ pi ",4.14159265358979)) result=false;
if (!test_String(" 1/(1+ pi) ",0.241453007005224)) result=false;
if (!test_String(" piq/(pi*pi) ",1)) result=false;
return result;
}

bool test_random(int n)
{
bool result=true;
for (int i=0; i<n; i++)
  {
    double a=rand()/100.0;
    double b=rand();
    double c=rand();
    double y=a+b+c;
    AnsiString s=FloatToStr(a)+"+"+FloatToStr(b)+"+"+FloatToStr(c);
    if (!test_String(s,y)) result=false;

    y=a+b*c;
    s=FloatToStr(a)+"+"+FloatToStr(b)+"*"+FloatToStr(c);
    if (!test_String(s,y)) result=false;

    y=a+b/c;
    s=FloatToStr(a)+"+"+FloatToStr(b)+"/"+FloatToStr(c);
    if (!test_String(s,y)) result=false;

    y=1/a+1/(b+3.14*c);
    s=" 1/"+FloatToStr(a)+"+1/("+FloatToStr(b)+"+3.14*"+FloatToStr(c)+")";
    if (!test_String(s,y)) result=false;

    y=(1/a+1/(1/b+(1/(3.14*c))));
    s="
(1/"+FloatToStr(a)+"+1/(1/"+FloatToStr(b)+"+(1/(3.14*"+FloatToStr(c)+"))))";
    if (!test_String(s,y)) result=false;
  }

return result;
}

bool test_All()
{
bool result=true;
if (!teste_parser()) result=false;
if (!teste_parser1()) result=false;
if (!test_random(100)) result=false;
return result;
}
```

11.5.4 Aufgaben 5.3 (DupFiles)

```
// File: C:\Loesungen_CB2006\Kap_5\5.3\DupFilesU.cpp

#include <vcl.h>
#pragma hdrstop
#include "DupFilesU.h"
```

```
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
DriveComboBox1->Drive='C';
RichEdit1->ScrollBars=ssVertical;
Form1->Caption="Duplicate Files";
Form1->MultiMap->Checked=true;
OneDrive->Caption="Ein Laufwerk";
AllDrives->Caption="Alle Laufwerke";
}


//----- Aufgabe 5.3.7, 2. -------------------------------------------

typedef AnsiString DataType ; // Datentyp der Listendaten
typedef AnsiString KeyType ; // Datentyp der Schlüsselwerte

// Die Listnode-Knoten und Funktionen sind wie in Abschnitt 3.12.11

struct Listnode {
  DataType data;            // Daten
  Listnode* next;
};

Listnode* newListnode(const DataType& data, Listnode* next)
{ // Gibt einen Zeiger auf einen neuen Listenknoten {d0,nx0} zurück,
  // wobei d0 und nx0 die Argumente für data und next sind.
Listnode* n=new Listnode;
n->data = data;
n->next = next;
return n; // Nachbedingung: Der Rückgabewert zeigt auf einen neuen
}           //                Knoten mit den Elementen {d0,nx0}

void insertLastListnode(Listnode*& first, Listnode*& last, const DataType& data)
// first last Neuer Knoten n
                                        //  f0    l0
{ // Erzeugt einen neuen Listen-Knoten und fügt diesen nach last ein.
  // Last zeigt anschließend auf den letzten und first auf den
  // ersten Knoten der Liste.
Listnode* n=newListnode(data,0); //                      n0 ->{d0,0}
if (last==0) first=n;            //          n0
else last->next=n;               //               l0->{d,n0}
last=n;                          //               n0
// Nachbedingung: Bezeichnet man den Wert von last vor
//   dem Aufruf dieser Funktion mit l0, gilt
//   Fall I,  l0==0: first==n && last==n
//   Fall II, l0!=0: l0->next==n && last==n
}


// ******** Die Treenode-Knoten und Funktionen beginnen hier

struct Treenode {
  KeyType key;
  Listnode *first,
           *last;
  Treenode *left,
           *right;
};

void insertTreenode(Treenode*& n, KeyType key, DataType data)
{
if (n==0)
```

```
  {
    n = new Treenode;
    n->key   = key;
    n->left  = 0;
    n->right = 0;
    n->first = 0;
    n->last  = 0;
    // Erzeuge einen neuen Listenknoten, auf den n->first und
    // n->last anschließend zeigen:
    insertLastListnode(n->first, n->last,data);
  }
else if (key < n->key) insertTreenode(n->left,key,data);
else if (key > n->key) insertTreenode(n->right,key,data);
else     insertLastListnode(n->first, n->last,data);
}

void processTreenode(const Treenode* n)
{
if (n->first->next!=0)
  { // nur Listen mit mehr als einem Knoten ausgeben
    Form1->RichEdit1->Lines->Add(n->key.Trim());
    for (Listnode* i=n->first; i!=0; i=i->next)
      Form1->RichEdit1->Lines->Add("  "+i->data);
  }
};

void traverseTree(const Treenode* n)
{
if (n!=0)
  {
    traverseTree(n->left);
    processTreenode(n);
    traverseTree(n->right);
  }
}


// ******** Die map-Deklarationen und Funktionen beginnen hier

#include <map>
using std::multimap;
typedef multimap<KeyType,DataType> MMType;

void PrintMap(const MMType& mm)
{
for (MMType::const_iterator i=mm.begin(); i!=mm.end(); )
  {
    MMType::const_iterator first=mm.lower_bound(i->first),
                           last=mm.upper_bound(i->first),
                           k=first;
    k++;
    if (k!=last) // nur Listen mit mehr als einem Knoten ausgeben
      {
        Form1->RichEdit1->Lines->Add(i->first.Trim());
        for (MMType::const_iterator j=first; j!=last; j++)
          {
            Form1->RichEdit1->Lines->Add("  "+j->second);
            i++; // Mit jedem gefundenen Wert hochzählen
          };
      }
    else i++;
  }
}

bool use_map()
{
return Form1->MultiMap->Checked;
}
```

```
AnsiString addLeadingBlanks(const AnsiString& s, int n)
{ // Ergänzt den String s um so viele führende Nullen, daß
  // der Funktionswert die Länge n hat. Mit dieser Funktion
  // wird erreicht, daß ein Vergleich der Strings "12" < " 2"
  // dasselbe Ergebnis hat wie der Vergleich der Zahlen 12<2.
  // Ohne führendes Leerzeichen wäre das Ergebnis verschieden.
if (n>s.Length())
  return s.StringOfChar(' ',n-s.Length())+s;
else
  return s;
}

Treenode* W=0;
MMType mm;

void SearchSubdirs(const AnsiString& Pfad)
{
WIN32_FIND_DATA FindFileData;
AnsiString Mask="*.*";
HANDLE h=FindFirstFile(
                       (Pfad+Mask).c_str(), // Zeiger auf den Dateinamen
                        &FindFileData);      // Zeiger auf struct
if (h!=INVALID_HANDLE_VALUE)
  {
    int found=1;
    while (found)
      {
        if ((FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)!=0)
          { // Unterverzeichnis
            if (strcmp(FindFileData.cFileName,".")==0)
              ; // "." ignorieren
            else if (strcmp(FindFileData.cFileName,"..")==0)
              ; // ".." ignorieren
            else // Verzeichnis, rekursiv durchsuchen
              SearchSubdirs(Pfad+FindFileData.cFileName+"\\");
          }
        else
          {
            AnsiString Low=IntToStr(FindFileData.nFileSizeLow);
            AnsiString High=IntToStr(FindFileData.nFileSizeHigh);
            if (High=="0") High=" ";
            KeyType      Key=addLeadingBlanks(High,10)+addLeadingBlanks(Low,10)+"
"+FindFileData.cFileName;
            DataType Data=Pfad+FindFileData.cFileName;
            if (use_map()) mm.insert(MMType::value_type(Key,Data));
            else insertTreenode(W, Key, Data);
          }
        found=FindNextFile(h,&FindFileData);
      }
    FindClose(h);
  }
};


void __fastcall TForm1::OneDriveClick(TObject *Sender)
{ // Durchsucht nur das ausgewählte Laufwerk
RichEdit1->Clear();
TCursor Save_Cursor = Screen->Cursor;
try {
      Screen->Cursor = crHourGlass;    // Cursor als Sanduhr
      AnsiString Laufwerk=AnsiString(DriveComboBox1->Drive)+":\\";
      SearchSubdirs(Laufwerk);
      Form1->RichEdit1->Lines->Add("Data collected");
      try {
            RichEdit1->Lines->BeginUpdate();
            if (use_map()) PrintMap(mm);
            else
              traverseTree(W);
          }
```

```
      __finally
          {
             RichEdit1->Lines->EndUpdate();
          }
    }
__finally
    {
      Screen->Cursor = Save_Cursor;
    }
}

//---------------------------------------------------------------------

void __fastcall TForm1::AllDrivesClick(TObject *Sender)
{ // Durchsucht alle Laufwerke
TCursor Save_Cursor = Screen->Cursor;

Screen->Cursor = crHourGlass;    // Cursor als Sanduhr

RichEdit1->Clear();
for (int i=0; i<DriveComboBox1->Items->Count; i++)
  {
  AnsiString Laufwerk=AnsiString(DriveComboBox1->Items->Strings[i][1])+":\\";
  SearchSubdirs(Laufwerk);
  Form1->RichEdit1->Lines->Add("Data collected");
  }
if (use_map()) PrintMap(mm);
else
  {
    RichEdit1->Lines->BeginUpdate();
    traverseTree(W);
    RichEdit1->Lines->EndUpdate();
  }

Screen->Cursor = Save_Cursor;
}
```

11.5.5 Aufgaben 5.4

```
// File: C:\Loesungen_CB2006\Kap_5\5.4\FunctionsU.cpp

#include <vcl.h>
#pragma hdrstop
#include "FunctionsU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm4 *Form4;
//---------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
    : TForm(Owner)
{
}

// --------------- Aufgaben 5.4 ----------------------------

// --------------- Aufgabe 1 -------------------------------

int a=0;
void Init1()
{
a=1; // globale Variable
}
```

```
void call_Init1()
{
Init1;
// Ohne den Aufrufoperator ist das kein Funktionsaufruf.
// Richtig wäre: Init1();
}

void __fastcall TForm4::Button1Click(TObject *Sender)
{
call_Init1();
}


// --------------- Aufgabe 2 -------------------------------

int f(int& x)
{
x++;
return x;
}

int g(int& x)
{
x=2*x;
return x;
}

void __fastcall TForm4::Button2Click(TObject *Sender)
{
  int x=0;
  int y1=f(x)+g(x);
  x=0;
  int y2=g(x)+f(x);
  Memo1->Lines->Add("y1="+IntToStr(y1)+" y2="+IntToStr(y2));
  // y1=3 y2=1
}
```

11.5.6 Aufgaben 5.5

```
// File: C:\Loesungen_CB2006\Kap_5\5.5\DefArgU.cpp

#include <vcl.h>
#pragma hdrstop
#include "DefArgU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

// --------------- Aufgabe 5.5 ------------------------------

struct Punkt {
  int x, y;
};

Punkt initPunkt(int x=0, int y=0)
{
Punkt p={x,y};
```

```
return p;
}

AnsiString PunktToString(Punkt p)
{
  AnsiString s="("+IntToStr(p.x)+","+IntToStr(p.y)+")";
  return s;
}


void __fastcall TForm1::Button1Click(TObject *Sender)
{
Punkt p1, p2, p3;
p1=initPunkt();
p2=initPunkt(1);
p3=initPunkt(2,3);
Memo1->Lines->Add(PunktToString(p1));
Memo1->Lines->Add(PunktToString(p2));
Memo1->Lines->Add(PunktToString(p3));
}
```

11.5.7 Aufgaben 5.7

```
// File: C:\Loesungen_CB2006\Kap_5\5.7\OverLoadU.cpp

#include <vcl.h>
#pragma hdrstop
#include "OverLoadU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

// --------------- Aufgaben 5.7 ----------------------------

// --------------- Aufgabe 1 -------------------------------

void f(bool b) { }
void f(int i)  { }

void __fastcall TForm1::BoolClick(TObject *Sender)
{
f(true); // f(bool)
f(TRUE); // f(int)
}

// --------------- Aufgabe 2 -------------------------------

struct Punkt {
  int x, y;
};

Punkt initPunkt(int x, int y)
{
Punkt p={x,y};
return p;
}

Punkt initPunkt(int x)
```

```
{
Punkt p={x,0};
return p;
}

Punkt initPunkt()
{
Punkt p={0,0};
return p;
}

// Mit überladenen Funktionen ist das einfacher

AnsiString PunktToString(Punkt p)
{
  AnsiString s="("+IntToStr(p.x)+","+IntToStr(p.y)+")";
  return s;
}


void __fastcall TForm1::Button2Click(TObject *Sender)
{
Punkt p1, p2, p3;
p1=initPunkt();
p2=initPunkt(1);
p3=initPunkt(2,3);
Memo1->Lines->Add(PunktToString(p1));
Memo1->Lines->Add(PunktToString(p2));
Memo1->Lines->Add(PunktToString(p3));
}

// --------------- Aufgabe 3 -------------------------------

#define USE_MATH_H 1
//#define USE_MATH_H 0

#if USE_MATH_H
#include <math.h>
#else
#include <cmath>
using namespace std;
#endif

void __fastcall TForm1::Button1Click(TObject *Sender)
{
double x1=sqrt(1);
// Fehler: Mehrdeutigkeit zwischen 'std::sqrt(double)' und
//        'std::sqrt(long double)'
double x2=sqrt(1.0);
double x3=sqrt(1.0l);
}

// --------------- Aufgabe 4 -------------------------------

struct C { };

// a1) a2)
void fa(C& x) {}
void fa(const C& x) {}

// b1)
void fb1(C& x) {}
void fb1(C x) {}

// b2)
void fb2(const C& x) {}
void fb2(C x) {}

// c1)
```

```
void fc1(long x) {}
void fc1(double x) {}

// c2)
void fc2(const double& x) {}
void fc2(double x) {}

// d1) d2)
void fd1(int x) {}
void fd1(double x) {}

//
void fe(int x) {}
void fe(long x) {}

void f4()
{
C a;
// a1)
fa(a);

const C a_;
// a2)
fa(a_);

// b1)
// fb1(a);   // mehrdeutig
// b2)
// fb2(a_);  // mehrdeutig

long lg;
const long lg_=17;
// c1)
fc1(lg);
// c2)
fc1(lg_);
// d1)
// fd1(lg);  // mehrdeutig
// d2)
// fd1(lg_); // mehrdeutig

int i;
const int i_=17;
// e1
fe(i);
// e2
fe(i_);
// f1
// fc1(i);   // mehrdeutig
// f2
// fc1(i_);  // mehrdeutig
}
```

11.5.8 Aufgaben 5.8

```
// File: C:\Loesungen_CB2006\Kap_5\5.8\GlOpsU.cpp

#include <vcl.h>
#pragma hdrstop
#include "GlOpsU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
#include "..\..\CppUtils\TestUtils.h"

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
#ifdef _DEBUG
bool test_All(); // Prototyp
initTestUtils();
testUtilsSummary(Memo1, test_All());
#endif
}


// --------------- Aufgaben 5.8.2 --------------------------

// --------------- Aufgabe 1 -------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
Memo1->Lines->Text=
"Der Operator ^ kann nur für selbstdefinierte Datentypen überladen "
"werden. Da '-' eine höhere Priorität als '^' hat, wird x^n-1 als "
"x^(n-1) ausgewertet. Üblicherweise ist das aber (x^n)-1";
}


// --------------- Aufgabe 2 -------------------------------

struct Bruch {
  int z,n; // z: Zähler, n: Nenner
};


// a) ==, !=

inline bool operator==(const Bruch& q1, const Bruch& q2)
{
return q1.z*q2.n==q2.z*q1.n;
}

// Die folgenden Anweisungen wurden mit leichten Änderungen aus utility.h
// übernommen. Die Definitionen aus utility.h gehören zum C++-Standard:

inline bool operator!=(const Bruch& x, const Bruch& y)
{
return !(x == y);
}

// b) <, <=, >, >=

inline bool operator<(const Bruch& q1, const Bruch& q2)
{
return q1.z*q2.n < q2.z*q1.n;
}

inline bool operator>(const Bruch& x, const Bruch& y)
{
return y < x;
}

inline bool operator<=(const Bruch& x, const Bruch& y)
{
return !(y < x);
}

inline bool operator>=(const Bruch& x, const Bruch& y)
{
return !(x < y);
}
```

```
// c) +, -, *, /, skalare Multiplikation mit kürzen (-) == *(-1)

int ggT(int a, int b)
{ // siehe Aufgabe 3.7.4, 3.
int x=a;
int y=b;
while (y != 0)
  {
    int r = x%y;
    x = y;
    y = r;
  }
return x; // ggT(a,b)==x;
}

Bruch kuerzeBruch(Bruch q)
{
int t = ggT(q.z,q.n);
if (t==0) return q;
Bruch result = {q.z/t,q.n/t};
return result;
};

inline Bruch operator+(const Bruch& p, const Bruch& q)
{
Bruch result={p.z*q.n + q.z*p.n , p.n*q.n};
return kuerzeBruch(result);
};

inline Bruch operator-(const Bruch& p, const Bruch& q)
{
Bruch result={p.z*q.n - q.z*p.n , p.n*q.n};
return kuerzeBruch(result);
};

inline Bruch operator*(const Bruch& p, const Bruch& q)
{
Bruch result={p.z*q.z,p.n*q.n};
return kuerzeBruch(result);
};

inline Bruch operator/(const Bruch& p, const Bruch& q)
{
Bruch result={p.z*q.n,p.n*q.z};
return kuerzeBruch(result);
};


AnsiString BruchToAnsiString(Bruch q)
{
return IntToStr(q.z)+'/'+IntToStr(q.n);
};


// d) Zum Initialisieren:

Bruch InitBruch(int z,int n=1)
{ // InitBruch(x): x/1
Bruch b={z,n};
return b;
}

Bruch summierePotenzen(Bruch p,int n)
{ // summiere die Potenzen: 1 + q + q^2 ... + q^n
Bruch s=InitBruch(1), q=InitBruch(1);
for (int i=1; i<=n; i++)
  {
    q=q*p;
```

```
    s=s+q;
  }
return s;
}

Bruch Summenformel(Bruch p,int n)
{ // Summenformel: (p^(n+1) - 1)/(p-1)
Bruch q=p, Eins=InitBruch(1);
for (int i=2; i<=n+1; i++) q=q*p;
return (q-Eins)/(p-Eins);
}

#include <cmath> // für pow
bool test_BruchOperatoren(int max_z, int max_n)
{
bool result = true;
for (int z=-max_z; z<max_z; z++)
  for (int n=-max_n; n<max_n; n++)
   if (z!=n && n!=0)
    {
      int N=abs(n);
      Bruch p=InitBruch(z,n); // z/n
      Bruch s1=summierePotenzen(p,N);
      Bruch s2=Summenformel(p,N);
      // s3: setze z/n in die Summenformel ein
      using namespace std;
      Bruch                          s3=kuerzeBruch(InitBruch(pow(double(z),N+1)-
pow(double(n),N+1),pow(double(n),N)*(z-n)));
// Prüfe die Ergebnisse z.B. so
      if (!assertEqual(s1==s2,true,
             "s1="+BruchToAnsiString(s1)+" s2="+BruchToAnsiString(s2) ))
         result=false;
      if (!assertEqual(s1==s3,true,
             "s1="+BruchToAnsiString(s1)+" s2="+BruchToAnsiString(s2) ))
         result=false;
// oder so
      if (s1!=s2 || s1!=s3) // no news are good news
        Form1->Memo1->Lines->Add("p="+BruchToAnsiString(p)+
                                 " s1="+BruchToAnsiString(s1)+
                                 " s2="+BruchToAnsiString(s2)+
                                 " s3="+BruchToAnsiString(s3));
   }
return result;
}

void __fastcall TForm1::testBruchClick(TObject *Sender)
{
test_BruchOperatoren(7,7); // größere Werte: Bereichsüberlauf
}


// --------------- Aufgaben 5.8.4 -------------------------

//----- Aufgabe 1 ------------------------------------------------

#include <iostream>
using namespace std;

ostream& operator<<(ostream& f, const Bruch& b)
{
return f<<b.z<<"|"<<b.n;
}

istream& operator>>(istream& f, Bruch& b)
{
char Bruchstrich;
f>>b.z>>Bruchstrich>>b.n;
return f;
}
```

```
#include <string>
#include <sstream>
string BruchToStr(Bruch b)
{
ostringstream o;
o<<b;
return o.str();
}

Bruch StrToBruch(string s)
{
istringstream i(s);
Bruch b;
i>>b;
return b;
}

void __fastcall TForm1::Bruch_IO_OpClick(TObject *Sender)
{
Bruch b1;//={1,2};
Memo1->Lines->Add(BruchToStr(b1).c_str());
string s="3/4";
Bruch b2=StrToBruch(s);
Memo1->Lines->Add(BruchToStr(b2).c_str());
}

bool test_BruchKonversion(int max_z, int max_n)
{
bool result = true;
for (int z=-max_z; z<max_z; z++)
  for (int n=-max_z; n<max_n; n++)
   if (z!=n && n!=0)
    {
      Bruch b1={z,n};
      string s=BruchToStr(b1);
      Bruch b2=StrToBruch(s);
      if (!assertEqual(b1==b2,true,
             "b1="+BruchToAnsiString(b1)+" b2="+BruchToAnsiString(b2) ))
         result=false;
    }
return result;
}


bool test_All()
{
bool result = true;
if (!test_BruchOperatoren(7,7)) // größere Werte: Bereichsüberlauf
  result=false;
if (!test_BruchKonversion(10, 10))
  result=false;

return result;
}
```

11.6 Lösungen Kapitel 6

11.6.1 Aufgabe 6.1

```
// File: C:\Loesungen_CB2006\Kap_6\6.1\KlassenU.cpp

#include <vcl.h>
#pragma hdrstop
#include "KlassenU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

// --------------- Aufgaben 6.1.5 --------------------------


// --------------- Aufgabe 1 -------------------------------

/* Ein Konstruktor hat die Aufgabe, alle Datenelemente einer Klasse so zu
   initialisieren, dass sie beim Aufruf einer beliebigen Elementfunktion
   in einem definierten Zustand sind.

   In einem Destruktor sollen alle Ressourcen wieder freigegeben werden,
   die in einem Konstruktor reserviert wurden.
*/
class C {
   int n, max;
   int* a;
 public:
  C()
  {
   max=100;
   a=new int[max];
  } // Fehler: n wird nicht initialisiert, aber in show_data verwendet
    //         Falls n initialisiert wird, muss auch a[0] ... a[n]
    //         initialisiert werden.
  C(int i)
  {
   n=1;
   a=new int[100];
   a[0]=i;
  }; // Fehler: max wird nicht initialisiert, aber in add_data verwendet

  C(int i,int j)
  {
   n=2;
   max=100;
   a=new int[100];
   a[1]=i;
   a[2]=j;
  }; // Fehler: max und a[0] werden nicht initialisiert

  void add_data(int d)
  {
  if (n<max-1)
   {
     n++;
     a[n]=d;
   }
  }

  void show_data()
  {
```

```
  for (int i=0; i<n; i++)
   Form1->Memo1->Lines->Add(IntToStr(a[i]));
  }

  // Der Destruktor mit delete[] a fehlt
};

void __fastcall TForm1::Button_1_5_1Click(TObject *Sender)
{
//
}


// --------------- Aufgabe 2 -------------------------------

// a)

class C1 {
  int x,y,z;
 public:
  C1(int x_=0, int y_=0, int z_=0)
  {
  x=x_; y=y_; z=z_;
  }
}; // Da die Elemente nur initialisiert und keine Ressourcen reserviert
   // werden, ist kein Destruktor notwendig

// b)

class C2 {
  int* x;
 public:
  C2(int n)
  {
  x=new int[n];
  }
}; // Destruktor mit delete[] x ist notwendig

// c)

#include <fstream>
using namespace std;
class C3 {
  ifstream f; // eine Klasse der C++-Standardbibliothek
 public:
  C3(const char* FileName)
  {
   f.open(FileName);
  }
}; // Da für die Klasse C3 kein expliziter Konstruktor definiert wird,
   // erzeugt der Compiler einen. Dieser ruft die Destruktoren aller
   // Elemente von C3 auf, also auch den von f. Da ifstream eine Klasse
   // der Standardbibliothek von C++ ist und alle Klassen dieser Bibliothek
   // Destruktoren haben, die sich anständig verhalten und alle ihre
   // Ressourcen wieder freigeben, ist für C3 kein Destruktor notwendig.
   // Ein Destruktor mit close ist aber auch kein Fehler.

void __fastcall TForm1::Button_1_5_2Click(TObject *Sender)
{
//
}


// --------------- Aufgabe 3 -------------------------------

const double pi=3.14159265358979323846;

// a)
class Kreis{
```

```
  double r;
 public:
  Kreis(const double& Radius=1)
  {
  r=Radius;
  }

  double Radius()
  {
  return r;
  }

  void setzeRadius(const double& Radius)
  {
  r=Radius;
  }

  double Flaeche()
  {
  return r*r*pi;
  }

  double Umfang()
  {
  return 2*r*pi;
  }

  AnsiString toStr()
  {
  return "Kreis mit Radius "+FloatToStr(r);
  }
};

// b)
class Quadrat{  // Elementfunktionen außerhalb definieren
  double a;
 public:
  Quadrat(const double& Seitenlaenge=1);

  double Seitenlaenge();
  void setzeSeitenlaenge(const double& Seitenlaenge);
  double Flaeche();
  double Umfang();
  AnsiString toStr();
};

Quadrat::Quadrat(const double& Seitenlaenge)
{
a=Seitenlaenge;
};

double Quadrat::Seitenlaenge()
{
return a;
}

void Quadrat::setzeSeitenlaenge(const double& Seitenlaenge)
{
a=Seitenlaenge;
}

double Quadrat::Flaeche()
{
return a*a;
}

double Quadrat::Umfang()
{
return 4*a;
```

```
}

AnsiString Quadrat::toStr()
{
return "Quadrat mit Seitenlänge "+FloatToStr(a);
}


// c)
class Rechteck{
  double a,b; // Seitenlängen
 public:
  Rechteck(const double& a_, const double& b_)
  {
  a=a_;
  b=b_;
  };

  double Seitenlaenge_a()
  {
  return a;
  }

  double Seitenlaenge_b()
  {
  return b;
  }

  void setzeSeitenlaengen(const double& a_, const double& b_)
  {
  a=a_;
  b=b_;
  }

  double Flaeche()
  {
  return a*b;
  }

  double Umfang()
  {
  return 2*(a+b);
  }

  AnsiString toStr()
  {
  return "Rechteck mit a="+FloatToStr(a)+" b="+FloatToStr(b);
  }
};

// e)
void test_Figuren()
{
Kreis k1(10);
Form1->Memo1->Lines->Add(k1.toStr());
Form1->Memo1->Lines->Add(k1.Umfang());
Form1->Memo1->Lines->Add(k1.Flaeche());
Kreis* pk=new Kreis(10);
Form1->Memo1->Lines->Add(pk->toStr());
Form1->Memo1->Lines->Add(pk->Umfang());
Form1->Memo1->Lines->Add(pk->Flaeche());

Quadrat q(10);
Form1->Memo1->Lines->Add(q.toStr());
Form1->Memo1->Lines->Add(q.Umfang());
Form1->Memo1->Lines->Add(q.Flaeche());
Quadrat* pq=new Quadrat(10);
Form1->Memo1->Lines->Add(pq->toStr());
Form1->Memo1->Lines->Add(pq->Umfang());
```

```
Form1->Memo1->Lines->Add(pq->Flaeche());

Rechteck r(10,20);
Form1->Memo1->Lines->Add(r.toStr());
Form1->Memo1->Lines->Add(r.Umfang());
Form1->Memo1->Lines->Add(r.Flaeche());
Rechteck* pr=new Rechteck(7.5,12.5);
Form1->Memo1->Lines->Add(pr->toStr());
Form1->Memo1->Lines->Add(pr->Umfang());
Form1->Memo1->Lines->Add(pr->Flaeche());
}

void __fastcall TForm1::Button_1_5_3Click(TObject *Sender)
{
test_Figuren();
}


// --------------- Aufgabe 4 -------------------------------

// Nach der Aufgabenstellung liegt diese Lösung nahe:

class Grundstueck {
  char* Anschrift;
  double Kaufpreis;
  double Flaeche;
 public:
  Grundstueck(const char* Anschrift_, const double& Kaufpreis_,
                                       const double& Flaeche_)
  {
  Anschrift = new char[strlen(Anschrift_)+1];
  strcpy(Anschrift,Anschrift_);
  Kaufpreis=Kaufpreis_;
  Flaeche=Flaeche_;
  }

/*
Außerdem sind natürlich noch public Elementfunktionen (die
Schnittstelle) wie setzeAnschrift, Anschrift usw. notwendig, mit denen
man die Datenelemente setzen und lesen kann.

Für den Konstruktor wird oft diese Lösung vorgeschlagen:

  Grundstueck(const char* Anschrift_, const double& Kaufpreis_, const double&
Flaeche_)
  {
  Anschrift = Anschrift_;
  Kaufpreis=Kaufpreis_;
  Flaeche=Flaeche_;
  }

Diese Lösung funktioniert auf den ersten Blick auch. Sie hat aber
die große Schwäche, dass der Zeiger in einem Objekt der Klasse
Grundstueck auf denselben Speicherbereich zeigt wie das Argument.
- Falls einer der beiden verändert wird, ändert sich der andere
  ebenfalls.
- Falls das Argument eine lokale Variable ist, ist es nach
  dem Verlassen der Funktion undefiniert.

Das ist aber ein Fehler. Deswegen sollte man die Anschrift besser
auf eine lokale Kopie des Arguments zeigen lassen:

*/
  ~Grundstueck()
  {
  delete[] Anschrift;
  }

  AnsiString toStr() const // const: konstante Elementfunktion, siehe
```

```
  {          // Abschnitt 8.2.10 "Konstante Klassenelemente und Objekte"
  return "Grundstück in "+AnsiString(Anschrift)+ " KP: "+
         FloatToStr(Kaufpreis)+ " DM  Flaeche: "+FloatToStr(Flaeche);
  }
};

void display(const Grundstueck& g)
{
Form1->Memo1->Lines->Add(g.toStr());
}


/*
In den Klassen Grundstueck usw. muss im Konstruktor der Speicherplatz für
die Anschrift mit new reserviert und das Argument Anschrift_ in diesen
Speicherbereich kopiert werden, da die Anschrift ein Zeiger auf einen
nullterminierten String ist. Wegen new ist außerdem ein Destruktor
notwendig.

Würde man als Datentyp für die Anschrift eine Stringklasse (z.B. string
oder AnsiString) wählen, wäre diese Klasse einfacher, da auch kein
Destruktor notwendig ist:

class Grundstueck {
  string Anschrift;
  double Kaufpreis;
  double Flaeche;
 public:
  Grundstueck(AnsiString Anschrift_, double Kaufpreis_, double Flaeche_)
  {
  Anschrift = Anschrift;
  Kaufpreis=Kaufpreis_;
  Flaeche=Flaeche_;
  }

  string toStr() const // const: konstante Elementfunktion, siehe
  {  // Abschnitt 8.2.10 "Konstante Klassenelemente und Objekte"
  return "Grundstück in "+Anschrift+ " KP: "+FloatToStr(Kaufpreis)+
         " DM  Flaeche: "+FloatToStr(Flaeche);
  }
};

Deshalb sollte man Stringklassen immer dem Datentyp char* vorziehen.

Es liegt nahe, die gemeinsamen Datenelemente 'Anschrift' und 'Kaufpreis'
der verschiedenen Klassen wiederzuverwenden. Diese Möglichkeit wird im
Kapitel über Vererbung behandelt.

Die folgenden Klassen sind ähnlich aufgebaut wie die Klasse CGrundstueck
und zeigen, welche Ersparnis an Arbeits- und Testaufwand die String-
klassen anstelle von nullterminierten Strings hätten.
*/

void test_Immo()
{
Grundstueck g("Tübingen",100000,400);
display(g);
}

void __fastcall TForm1::Button_1_5_4Click(TObject *Sender)
{
test_Immo();
}


// --------------- Aufgabe 5 -------------------------------

namespace N_Aufgabe5 {
```

```
void display(AnsiString s, int i=-1)
{
if (i>=0) s=s+IntToStr(i);
Form1->Memo1->Lines->Add(s);
}

class C{
  int x;
 public:
  C (int x_=0)
  { // Beim Aufruf ohne Argument ein Standard-
    x=x_;                              // konstruktor
    display("  Konstruktor: ", x);
  }
  ~C ()
  {
    display("  Destruktor: ",x);
  }
};

void f1(C c)
{
display("  in f1(): Werteparameter");
};

void f2(const C& c)
{
display("  in f2(): Referenzparameter");
};

C f3(int i)
{
display("  in f3(): return-Wert");
return C(i);
};

void test1()
{
C x(1);
C* z=new C(2);
display("vor x=C(3)");
x=C(3);
display("vor f1(4)");
f1(4);
display("vor f2(x)");
f2(x);
display("vor f3(5)");
x=f3(5);
delete z;
display("Ende von test()");
}

/*
Die Anweisungen der Funktion test1 erzeugen die eingerückten
Meldungen:

C x(1);
  Konstruktor: 1
C* z=new C(2);
  Konstruktor: 2
display("vor x=C(3)");
  vor x=C(3)
x=C(3);
  Konstruktor: 3
  Destruktor: 3
display("vor f1(4)");
  vor f1(4)
f1(4);
  Konstruktor: 4
```

```
  in f1(): Werteparameter
  Destruktor: 4
display("vor f2(x)");
  vor f2(x)
f2(x);
  in f2(): Referenzparameter
display("vor f3(5)");
  vor f3(5)
x=f3(5);
  in f3(): return-Wert
  Konstruktor: 5
  Destruktor: 5
delete z;
  Destruktor: 2
display("Ende von test()");
  Ende von test()
}
  Destruktor: 5

*/

void test2()
{
display("vor p1");
C* p1=new C[2];
delete[] p1;

display("vor p2");
C* p2=new C[2];
delete p2;

display("vor p3");
C* p3=new C;
delete[] p3;

display("vor p4");
C* p4=new C(4);
delete[] p4;

display("Ende von test()");
}
/*
b)    Die Funktion test2 erzeugt unter anderem die folgende Ausgabe:
vor p1
Konstruktor: 0
Konstruktor: 0
Destruktor: 0
Destruktor: 0
vor p2
Konstruktor: 0
Konstruktor: 0
Destruktor: 0
    Die Anweisungen nach display(„vor p3") sind falsch, da ein mit new[] reservierter Speicherbereich mit delete (und nicht mit delete[]) wieder freigegeben wird, bzw. ein mit new reservierter Speicherbereich mit delete[] (und nicht mit delete).
    Gibt man bei dem Ausdruck nach new runde Klammern an, sind die Werte in den Klammern Argumente für den Konstruktor und nicht etwa Arraygrenzen. Deswegen muss nach diesem Aufruf delete und nicht delete[] aufgerufen werden.
*/

} // end of namespace N_Aufgabe5

void __fastcall TForm1::Button_1_5_5Click(TObject *Sender)
{
N_Aufgabe5::test1();
N_Aufgabe5::test2();
}
```

```
// --------------- Aufgabe 6 -------------------------------

class MeinString {
  char* s; // Zeiger auf nullterminierten String
  int n;   // Länge des Strings
 public:
  MeinString(const char* p)  // 1
  { // p muss auf einen nullterminierten String zeigen
    n=strlen(p);
    s=new char[n+1];
    strcpy(s,p);
      };
  MeinString(char c)         // 2
  { // kann auch mit einem int-Argument aufgerufen werden
    n=1;
    s=new char[n+1];
    *s=c;
    *(s+1)='\0';
  };

  const char* c_str()
  { // 'strcpy(s1.c_str(),"abc")' geht ohne "const", aber nicht mit.
    // Da ein solcher Aufruf die Klasseninvariante zerstören kann,
    // ist hier const notwendig.
  return s;
  }

   void replace(int from, const char* x); // für Aufgabe 8.1.7.2
};

void test_6()
{
MeinString s("abc");
Form1->Memo1->Lines->Add(IntToStr(strlen(s.c_str())));
}

void __fastcall TForm1::Button_1_5_6Click(TObject *Sender)
{
test_6();
}


// --------------- Aufgabe 7 -------------------------------

namespace N_Aufgabe_7 { // namespace, um Namenskonflikte
                        //            mit Aufgabe 7 zu vermeiden
class C {
  typedef int T;
  T x;
 public:
  C(T x_)
  {
  x=x_;
  }

  T& data() // T data() wäre zu langsam
  {
  return x;
  }
};

void test()
{
C c(1);
c.data()=17;
/*
   Da der Funktionswert von data ein Referenztyp ist, kann man mit dieser
   public Funktion den Wert des private Elements x verändern und damit die
```

```
   Konsistenzbedingungen verletzen.

   Diesen vermutlich nicht beabsichtigten Effekt kann man verhindern,
   indem man den Rückgabetyp der Funktion data als konstanten
   Referenzparameter definiert:

     const T& data()
     {
     return x;
     }

   Mit einer Elementfunktion, die eine Referenz zurückgibt, können auch
   Datenelemente von Objekten angesprochen werden, die nicht mehr
   existieren.
*/
}

} // end of namespace N_Aufgabe_7


void __fastcall TForm1::Button_1_5_7Click(TObject *Sender)
{
N_Aufgabe_7::test();
}

// --------------- Aufgabe 8 -------------------------------

//     Mit diesem Trick lassen sich alle Vorteile des Zugriffsrechts
//     private zunichte machen.

//     #undef private ist hier unpassend.

//     Wenn eine Klasse aber nicht alle Elementfunktionen zur Verfügung
//     stellt, die ein Anwender benötigt (also nicht vollständig ist),
//     kann dieser Trick aber doch eine gewisse Berechtigung haben.



// --------------- Aufgaben 6.1.6 --------------------------

// --------------- Aufgabe 1 -------------------------------

/*
a) Die Aufgabestellung legt die folgenden Klassen nahe:

      Klasse: Kreis
        Datenelemente: Mittelpunkt, Radius
        Elementfunktionen: Radius (getRadius), setzeRadius
                           Mittelpunkt (getMittelpunkt), setzeMittelpunkt

      Klasse: Quadrat
        Datenelemente: Mittelpunkt, Seitenlaenge
        Elementfunktionen: Seitenlaenge (getSeitenlaenge),
                           setzeSeitenlaenge
                           Mittelpunkt (getMittelpunkt), setzeMittelpunkt

      Klasse: Rechteck
        Datenelemente: Mittelpunkt, Seitenlaenge_a, Seitenlaenge_b
        Elementfunktionen: Seitenlaenge (getSeitenlaenge), setzeSeitenlaenge
                           Mittelpunkt (getMittelpunkt), setzeMittelpunkt

      Klasse Zeichnung
        Datenelemente: Container mit Figuren wie Kreis, Quadrat usw.
                       AnzahlFiguren
        Elementfunktionen: zeichne

   Falls die Figuren in einem Zeichenprogramm auch noch grafisch
   dargestellt werden sollen, sind außerdem noch Eigenschaften
   wie Farbe und eine Strichstärke sowie Funktionen wie zeichnen
```

```
   notwendig. Für die Position der Figuren kann eine Klasse wie
   C2DPunkt hilfreich sein.

   Falls die Figuren einer Zeichnung auch noch angezeigt oder
   ausgedruckt werden sollen, noch Funktionen wie toString.

b) Die Aufgabenstellung legt die folgenden Klassen nahe:

      Klasse: Eingabefeld,
        Datenelement: Text
        Elementfunktionen: getText, setText

      Klasse: Ausgabefeld,
        Datenelement: Text
        Elementfunktionen: getText, setText

      Klasse: Button
        Datenelement: Aufschrift
        Elementfunktionen: OnClick

c) Die Aufgabestellung legt die folgenden Klassen nahe:

      Klasse: Motor
        Elementfunktionen: setzeDrehzahl, leseDrehzahl

      Klasse: Temperaturregler
        Elementfunktionen: setzeTemperatur, leseTemperatur

       Klasse: Wasserstandsregler
        Elementfunktionen: setzeWasserstand, leseWasserstand

       Klasse: Uhr
        Elementfunktionen: ZeitSeitStart

      Klasse Waschmaschine:
        Datenelemente: Motor, Temperaturregler,
                       Wasserstandsregler, Uhr


// --------------- Aufgabe 2 -------------------------------

Ein Anwender benötigt vermutlich auch noch Funktionen,
um den Tag, den Monat und das Jahr einzeln zu setzen:

      class Datum_2 {
       public:
        void setze(int Tag, int Monat, int Jahr)
        {
        int MaxTag=31;
        if ((Monat==4)||(Monat==6)||(Monat==9)||
      (Monat==11)) MaxTag=30;
        else if (Monat==2)
          {
            if (Schaltjahr(Jahr)) MaxTag=29;
            else MaxTag=28;
          }
        if ((1<=Monat) && (Monat<=12) && (1<=Tag) &&
      (Tag<=MaxTag))
          {
            Tag_=Tag;
            Monat_=Monat;
            Jahr_=Jahr;
          }
        else Fehlermeldung("Ungültiges Datum");
        }
        int Tag()   { return Tag_;}
        int Monat() { return Monat_;}
        int Jahr()  { return Jahr_;}
       private:
```

```
        int Tag_, Monat_, Jahr_;
      };

*/


// --------------- Aufgabe 3 -------------------------------

#include "C2DPunkt.cpp"


// --------------- Aufgabe 4 -------------------------------

// Eine set-Funktion ist nur dann angemessen, wenn sie nicht zu einer
// Inkonsistenz der Daten führen kann.
// Außerdem ist der Elementname als Bestandteil des Funktionsnamens oft
// nicht angemessen. Was ist, wenn der Name des Datenelements später
// geändert werden muss? Mit Namen, die eine Funktion inhaltlich
// beschreiben, ist diese Gefahr wesentlich geringer.
// Beispiel: Bei der Klasse MeinString würde so eine Funktion
// get_n erzeugt. Hier wäre der Name "Laenge" wesentlich besser.


// --------------- Aufgaben 6.1.7 --------------------------

// --------------- Aufgabe 1 -------------------------------

#include "C2DPunkt.cpp"
#include "..\..\CppUtils\testutils.h"

bool testC2DPunkt()
{ // teste alle public Elementfunktionen und Konstruktoren
bool result=true;
C2DPunkt p1(2,3);
if (!assertEqual(p1.X(),2.0,"p1.x==2",Form1->Memo1)) result=false;
if (!assertEqual(p1.Y(),3.0,"p1.y==3",Form1->Memo1)) result=false;

C2DPunkt p2(2);
if (!assertEqual(p2.X(),2.0,"p2.x==2",Form1->Memo1)) result=false;
if (!assertEqual(p2.Y(),0.0,"p2.y==0",Form1->Memo1)) result=false;

C2DPunkt p3;
if (!assertEqual(p3.X(),0.0,"p3.x==0",Form1->Memo1)) result=false;
if (!assertEqual(p3.Y(),0.0,"p3.y==0",Form1->Memo1)) result=false;

C2DPunkt p4(p1);
if (!assertEqual(p4.X(),p1.X(),"p3.x==p1.x",Form1->Memo1)) result=false;
if (!assertEqual(p4.Y(),p1.Y(),"p3.y==p1.y",Form1->Memo1)) result=false;

p3.setze(5,6);
if (!assertEqual(p3.X(),5.0,"p1.x==5",Form1->Memo1)) result=false;
if (!assertEqual(p3.Y(),6.0,"p1.y==6",Form1->Memo1)) result=false;

p3.setze(p1);
if (!assertEqual(p3.X(),p1.X(),"p3.x==p1.x",Form1->Memo1)) result=false;
if (!assertEqual(p3.Y(),p1.Y(),"p3.y==p1.y",Form1->Memo1)) result=false;

p3.setzeX(8);
if (!assertEqual(p3.X(),8.0,"p3.x==p1.x",Form1->Memo1)) result=false;
p3.setzeY(9);
if (!assertEqual(p3.Y(),9.0,"p3.y==p1.y",Form1->Memo1)) result=false;

return result;
}

void __fastcall TForm1::Button_1_7_1Click(TObject *Sender)
{
initTestUtils();
if (testC2DPunkt())
```

```
  Memo1->Lines->Add("All tests passed");
else
  Memo1->Lines->Add("Tests failed");
}

// --------------- Aufgabe 2 -------------------------------

/*
a) Bei den Klassen Kreis, Quadrat, Rechteck ist die Konsistenz-
   bedingung immer erfüllt, d.h. true.

b) Viele Funktionen dieser Klassen verändern keine Datenelemente
   und können deshalb auch keine Konsistenzbedingungen zerstören.
   Da die Konsistenzbedingung bei den Klassen Kreis usw. immer
   erfüllt ist, kann sie auch nicht zerstört werden.

c) Wenn Datenelemente f für die Fläche, u für den Umfang usw. in
   die Klasse aufgenommen werden, ist die Konsistenzbedingung nicht
   mehr immer true, sondern im Kreis f=r*r*pi, u=2*pi*r usw.

d) Eine Funktion setzeTag darf den als Argument übergebenen Wert
   nur dann setzen, wenn sie die Konsistenzbedingung nicht verletzt.

e) Wenn die Anschrift den Datentyp char* hat, gilt für sie die
   Konsistenzbedingung, dass Anschrift auf einen reservierten
   Speicherbereich zeigt.
f) Wenn der Datentyp der Anschrift eine Stringklasse ist, wird
   der notwendige Speicher durch die Stringklasse verwaltet.

*/

void __fastcall TForm1::Button_1_7_2Click(TObject *Sender)
{
AnsiString s1a=
"Bei den geometrischen Figuren müssen die Seitenlängen bzw. der "
"Radius >= 0 sein.";

AnsiString s1b=
"Damit die Fläche und der Umfang immer stimmen, müssen sie in jeder "
"Elementfunktion aktualisiert werden. Das ist aber aufwendiger "
"als die Lösung mit den Funktionen.";

Memo1->Lines->Add("1. a)");
Memo1->Lines->Add(s1a);
Memo1->Lines->Add("1. b)");
Memo1->Lines->Add(s1b);
}
```

11.6.2 Aufgabe 6.1.6, 3. (C2DPunkt)

```
// File: C:\Loesungen_CB2006\Kap_6\6.1\C2DPunkt.cpp

#ifndef C2DPUNKT_CPP
#define C2DPUNKT_CPP
// --------------- Aufgabe 3 -------------------------------


// Diese Aufgabenstellung ist natürlich sehr vage.
// Vorstellbar sind aber z.B. die folgenden Anforderungen:
// - Einen C2DPunkt mit einzelnen Koordinaten sowie einem anderen
//   C2DPunkt zu initialisieren.
// - Jede Koordinate einzeln zu setzen und zu lesen
// - Beide Koordinaten auf einen anderen C2DPunkt zu setzen
// - Den Punkt als String darzustellen
```

```
class C2DPunkt{
  double x,y;
 public:
  // Lösung 7.1.6, 3.
  C2DPunkt(const double& x_=0, const double& y_=0){x=x_; y=y_;   }
  C2DPunkt(const C2DPunkt& p) {x=p.x; y=p.y;  }

  // Lösung 7.2.
  // C2DPunkt(const double& x_=0, const double& y_=0):x(x_), y(y_) {  }
  //C2DPunkt(const C2DPunkt& p):x(p.x), y(p.y) {  }

  // Später wird gezeigt, daß dieser zweite Konstruktor ein
  // Copy-Konstruktor ist, der automatisch erzeugt wird und
  // deswegen nicht definiert werden muß.

  // Viele Konstruktoren haben wie hier einen leeren Anweisungsteil
  void setzePosition(const double& x_, const double& y_)
  {
  x=x_;
  y=y_;
  }

  void setzePosition(const C2DPunkt& p)
  {
  x=p.X();
  y=p.Y();
  }
  void setzeX(const double& x_) { x=x_; }
  void setzeY(const double& y_) { y=y_; }
  double X() const { return x; }
  double Y() const { return y; }
  void setze(const double& x_, const double& y_) { x=x_; y=y_; }
  void setze(const C2DPunkt& p) { x=p.x; y=p.y; }

  AnsiString toStr() const
  {
  return "("+FloatToStr(x) + "|" + FloatToStr(y)+")";
  } // { z. B. (2,345|3,45678) }
};

#endif
```

11.6.3 Aufgabe 6.2.2

```
// File: C:\Loesungen_CB2006\Kap_6\6.2\2_2.cpp

// --------------- Aufgaben 6.2.2 --------------------------
// --------------- Aufgabe 1 -------------------------------
class E {
 public:
  E()
  {
  Form1->Memo1->Lines->Add("Standardkonstruktor");
  }

  E(int)
  {
  Form1->Memo1->Lines->Add("int-Konstruktor");
  }

};

class C1 {
```

```
  E e1,e2;
 public:
  C1() { }
  C1(int i):e1(i) { }
  C1(int i,int j):e1(i),e2(j) { }
};

void Konstruktoren()
{
// Die folgenden Definitionen erzeugen die Meldungen:
C1 c0;      // Standardkonstruktor
C1 c1(1);   // Standardkonstruktor
            // int-Konstruktor
C1 c2(1,2); // Standardkonstruktor
            // int-Konstruktor
            // int-Konstruktor
}


// --------------- Aufgabe 2 -------------------------------

// Siehe C2DPunkt.cpp

/*
Ersetze

  Kreis(const double& Radius)
  {
  r=Radius;
  }

  Quadrat(const double& Seitenlaenge){a=Seitenlaenge;};

  Rechteck(const double& a_, const double& b_)
  {
  a=a_;
  b=b_;
  };

durch

  Kreis(const double& Radius):r(Radius) {}
  Quadrat(const double& Seitenlaenge):a(Seitenlaenge){};
  Rechteck(const double& a_, const double& b_):a(a_), b(b_){};

*/

// --------------- Aufgabe 3 -------------------------------

#include "C2DPunkt.cpp"

// Diese Aufgabe zeigt insbesondere, wie Klassen als Bausteine in
// weiteren Klassen verwendet werden.

class C2DKreis{
  double r;
  C2DPunkt pos;
 public:
  C2DKreis(const double& Radius=1, const double& x=0, const double& y=0):
     r(Radius), pos(x,y) { } // Reihenfolge der Deklaration

  C2DKreis(const double& Radius, const C2DPunkt& pos_):
     r(Radius), pos(pos_) { } // Reihenfolge der Deklaration
/*
  Alternativ zu diesen beiden Konstruktoren sind auch die folgenden
  möglich. Beim zweiten dieser Konstruktoren wird ein temporäres Objekt
  als Default-Argument verwendet:

  C2DKreis(const double& Radius, const double& x, const double& y):
```

```
     r(Radius), Position(x,y) { } // Reihenfolge der Deklaration

  C2DKreis(const double& Radius=0, C2DPunkt pos=C2DPunkt(0,0)):
     r(Radius), Position(pos) { } // Reihenfolge der Deklaration
*/

// neu:

  void setzePosition(const C2DPunkt& p)
  {
  pos=p;
  }

  C2DPunkt Position()
  {
  return pos;
  }

  void setzeRadius(const double& r_)
  {
  r=r_;
  }

  double Radius()
  {
  return r;
  }

 AnsiString toStr()
  {
  return "Kreis mit Radius "+FloatToStr(r)+" in "+pos.toStr();
  }
};

void testKreis()
{
C2DKreis k1(1,2,3);
Form1->Memo1->Lines->Add(k1.toStr());
C2DKreis k2(4);
Form1->Memo1->Lines->Add(k2.toStr());
C2DKreis k3;
Form1->Memo1->Lines->Add(k3.toStr());

C2DPunkt p(6,7);
C2DKreis k5(5,p);
Form1->Memo1->Lines->Add(k5.toStr());
C2DKreis k6(8,C2DPunkt(9,10));
Form1->Memo1->Lines->Add(k6.toStr());
double x=k6.Position().X();
}


// --------------- Aufgabe 4 -------------------------------

class Rechteck1 {
  C2DPunkt LinksOben; // Eckpunkt links oben
  double a,b; // Seitenlängen
 public:
  Rechteck1(C2DPunkt Mittelpunkt, double a_, double b_):a(a_), b(b_),
                      LinksOben(Mittelpunkt.X()-a/2, Mittelpunkt.Y()-b/2){ }

  AnsiString toStr()
  {
  return "Links oben: "+LinksOben.toStr();
  }

  AnsiString Kommentar()
  {
   return
```

```
   "Die Elemente eines Objekts werden in der Reihenfolge initialisiert, "
   "in der sie in der Klasse aufgeführt werden. Deshalb wird zuerst "
   "LinksOben initialisiert, und dann erst a und b. Linksoben verwendet "
   "dazu die bisher nicht initialisierten Werte a und b.\n\n"
   "Wenn man die Elemente in einer Initialisiererliste in der Reihenfolge "
   "ihrer Definition in der Klasse aufführt, ist die Gefahr eines solchen "
   "Missverständnisses geringer.\n\n"
   "Die fehlerhafte Initialisierung in der Klasse CRechteck1 vermeidet "
   "man, indem man die Reihenfolge der Datenelemente ändert.";
  }
};

class Rechteck2 { // Reihenfolge der Deklarationen vertauscht
  double a,b; // Seitenlängen
  C2DPunkt LinksOben; // Eckpunkt links oben
 public:
  Rechteck2(C2DPunkt Mittelpunkt, double a_,double b_):
     a(a_),b(b_), LinksOben(Mittelpunkt.X()-a/2, Mittelpunkt.Y()-b/2){ }

  AnsiString toStr()
  {
  return FloatToStr(LinksOben.X())+"|"+
         FloatToStr(LinksOben.Y());
  }
};

void Rechteck1_()
{
C2DPunkt mp(100,100);
Rechteck1 r1(mp, 10, 20);
Form1->Memo1->Lines->Add(r1.toStr());
Form1->Memo1->Lines->Add(r1.Kommentar());
Rechteck2 r2(mp, 10, 20);
Form1->Memo1->Lines->Add(r2.toStr());
}


// --------------- Aufgabe 5 -------------------------------

void MeinString_Konstr()
{
Form1->Memo1->Lines->Add(
"Da dieser Standardkonstruktor die anderen Konstruktoren ergänzen soll, "
"kann im Destruktor nicht mehr entschieden werden, ob der reservierte "
"Speicher mit delete oder mit delete[] freigegeben werden soll." );
}
```

11.6.4 Aufgabe 6.2.2, 3. (C2DPunkt)

```
// File: C:\Loesungen_CB2006\Kap_6\6.2\C2DPunkt.cpp

// --------------- Aufgabe 2 -------------------------------
// Diese Aufgabenstellung ist natürlich sehr vage.
// Vorstellbar sind aber z.B. die folgenden Anforderungen:
// - Einen C2DPunkt mit einzelnen Koordinaten sowie einem anderen
//   C2DPunkt zu initialisieren.
// - Jede Koordinate einzeln zu setzen und zu lesen
// - Beide Koordinaten auf einen anderen C2DPunkt zu setzen
// - Den Punkt als String darzustellen

class C2DPunkt{
  double x,y;
 public:
```

```
  // Lösung 7.1.6, 3.
  // C2DPunkt(const double& x_=0, const double& y_=0){x=x_; y=y_;   }
  // C2DPunkt(const C2DPunkt& p) {x=p.x; y=p.y;  }

  // Lösung 7.2.
  C2DPunkt(const double& x_=0, const double& y_=0):x(x_), y(y_) {  }
  C2DPunkt(const C2DPunkt& p):x(p.x), y(p.y) {  }

  // Später wird gezeigt, daß dieser zweite Konstruktor ein
  // Copy-Konstruktor ist, der automatisch erzeugt wird und
  // deswegen nicht definiert werden muß.

  // Viele Konstruktoren haben wie hier einen leeren Anweisungsteil
  void setzeX(const double& x_) { x=x_; }
  void setzeY(const double& y_) { y=y_; }
  double X() const { return x; }
  double Y() const { return y; }
  void setze(const double& x_, const double& y_) { x=x_; y=y_; }
  void setze(const C2DPunkt& p) { x=p.x; y=p.y; }
  // Später werden wir sehen, dass diese Funktion nicht notwendig ist,
  // da der vom Compiler erzeugte Zuweisungsoperator denselben Effekt hat.

  AnsiString toStr() const
  {
  return "("+FloatToStr(x) + "|" + FloatToStr(y)+")";
  } // { z. B. (2,345|3,45678) }
};
```

11.6.5 Aufgabe 6.2.4

```
// File: C:\Loesungen_CB2006\Kap_6\6.2\2_4.cpp

// --------------- Aufgabe 6.2.4 ---------------------------
// --------------- Aufgabe 1 -------------------------------
/*
Würde man diese Operatoren als Elementfunktionen einer Klasse C
definieren, müsste der linke Operand beim Aufruf das Objekt der
Klasse sein:

    c<<f // Im Gegensatz zur üblichen Konvention f<<c
    c>>f // Im Gegensatz zur üblichen Konvention f>>c

Das widerspricht aber der üblichen Schreibweise mit den Operatoren
<< und >>, bei der das Stream-Objekt immer links vom Operator steht.
Deswegen sollte man die Operatoren << und >> immer durch eine globale
Funktion überladen.
*/


// --------------- Aufgabe 2 -------------------------------

class Bruch {
  int z,n; // z: Zähler, n: Nenner
 public:
  Bruch(int z_, int n_=1):z(z_),n(n_)
  {
  // Nenner == 0 abfangen
  if (n==0) n=1; // oder Exception auslösen
  };

  Bruch operator+=(const Bruch& q) const
  {
  return Bruch(z*q.n + q.z*n , n*q.n);
```

```
  }

// a)
  friend Bruch operator-(const Bruch& p, const Bruch& q);
  friend bool assertEqual(const Bruch& p,const Bruch& q);
  AnsiString toStr() const
  { // Mit __int64 FloatToStr anstelle IntToStr
  return FloatToStr(z)+"/"+FloatToStr(n);
  }
};

// a)
Bruch operator-(const Bruch& p, const Bruch& q)
{
return Bruch(p.z*q.n - q.z*p.n , p.n*q.n);
};

// b)
Bruch operator+(const Bruch& p, const Bruch& q)
{
return Bruch(p)+=q;
// return p+=q; // geht nicht wegen const
};

#include "..\..\CppUtils\testutils.h"

bool assertEqual(const Bruch& p,const Bruch& q)
{
return assertEqual(p.z*q.n,q.z*p.n,"Bruch p="+p.toStr()+" == q="+q.toStr());
}

void TestBruch(int n)
{
Bruch p(1,n);
Bruch s(0,1);
for (int i=0; i<n; i++)
  s=s+p;
bool result=assertEqual(s,Bruch(1,1)); // n*(1/n)==1/1
Bruch q(1,2);
Bruch d=s-q;
if (!assertEqual(d,q)) result=false;   // 1/1-1/2==1/2
if (result)
  Form1->Memo1->Lines->Add("All Tests passed");
else
  Form1->Memo1->Lines->Add("Tests failed");
}

// --------------- Aufgabe 3 -------------------------------

class AssozContainer {
  // Dieser Container ist eine starke Vereinfachung und
  // zeigt nur, wie man den Indexoperator so überladen kann,
  // dass er sich wie in std::map verhält.

  int n; // Anzahl der Elemente im Container
  typedef AnsiString T1;
  typedef AnsiString T2;
 public:
  struct Paar {
    T1 first;
    T2 second;
  };
  Paar a[100]; // nur zur Vereinfachung so einfach

  AssozContainer():n(0) {};

// a)

  T2& operator[](const T1& s)
```

```
  { // lineares Suchen: sehr einfach
  for (int i=0; i<n; i++)
    if (a[i].first==s)
      return a[i].second;
  a[n].first=s;
  return a[n++].second;
  }

  class InvalidIndexException {}; // sehr einfach

  const T2& operator[](const T1& s) const
  { // lineares Suchen: sehr einfach
  for (int i=0; i<n; i++)
    if (a[i].first==s)
      return a[i].second;
  // Falls das gesuchte Element nicht gefunden wird, z.B eine
  // Exception auslösen.
  throw InvalidIndexException();
  }

 void showAll()
  {
  for (int i=0; i<n; i++)
    Form1->Memo1->Lines->Add(a[i].first+": "+a[i].second);
  }

// b)

  class iterator {
    Paar* Position;
   public:
    iterator(Paar* pos):Position(pos) { }

    bool operator!= (const iterator& y)
    {
    return Position != y.Position;
    }

    iterator& operator++(int)
    {
     Position++;
     return *this;
    }

    Paar& operator* ()
    {
     return *Position;
    }

    Paar* operator-> ()
    {
     return Position;
    }
  };

  iterator begin()
  {
  return a;
  }

  iterator end()
  {
  return a+n;
  }
};


void test_AssozCont()
{
```

```
const AssozContainer ca;
AnsiString s1=ca["1"];

AssozContainer a;
Form1->Memo1->Lines->Add("1. ");
a.showAll();

a["Luigi Mafiosi"]="Luigi@palermo.net";
Form1->Memo1->Lines->Add("2. ");
a.showAll();

a["Karl Erbschleicher"]="Karl@aahoohell.kom";
Form1->Memo1->Lines->Add("3. ");
a.showAll();

a["Luigi Mafiosi"]="Luigi@Bankers.net"; // palermo.net war zu langsam
Form1->Memo1->Lines->Add("4. ");
a.showAll();

Form1->Memo1->Lines->Add(a["Karl Erbslaicher"]); // Schreibfehler
Form1->Memo1->Lines->Add("5. ");
a.showAll();
}
```

11.6.6 Aufgabe 6.2.6

```
// File: C:\Loesungen_CB2006\Kap_6\6.2\2_6.cpp

// --------------- Aufgabe 6.2.6 ---------------------------
// --------------- Aufgabe 1 -------------------------------
void SpezEltFktNotwendig()
{
const int n=10;
AnsiString a[n]={
  "a) Nicht notwendig, da alle Klassen keine Zeiger auf dynamisch "
      "reservierten Speicher enthalten.",
  "b) Notwendig, da alle Klassen Zeiger auf dynamisch reservierten"
      " Speicher enthalten.",
  "c) Mit string anstelle char* ist keine dieser Funktionen notwendig.",
  "d) Wenn man alles richtig macht, ist es kein Fehler, diese Funktionen "
      "selbst zu definieren, obwohl sie auch vom Compiler erzeugt werden. "
      "Allerdings haben die selbst definierten Funktionen keine Vorteile "
      "gegenüber den vom Compiler erzeugten. Sie haben aber den Nachteil, "
      "dass sie beim Hinzufügen oder Entfernen von Datenelementen angepasst  "
      "werden müssen, was leicht vergessen werden kann. "
  "e) Am einfachsten ist es, wenn man alle Klassen ohne Zeiger definieren kann."
      "Dann spart man die Notwendigkeit, diese Funktionen zu schreiben, und "
      "außerdem das Risiko von Fehlern, falls man das vergisst. "
      "Diese Beispiele zeigen insbesondere, dass man sich viel Arbeit "
      "sparen kann, wenn man Stringklassen anstelle von nullterminierten "
      "Strings verwendet.",
      "",
};
for (int i=0; i<n; i++)
 Form1->Memo1->Lines->Add(a[i]);
}


// --------------- Aufgabe 2 -------------------------------

void display(AnsiString s, int i=-1)
{
if (i>=0) s=s+IntToStr(i);
```

```
Form1->Memo1->Lines->Add(s);
}

class C{
  int x;
 public:
  C (int x_=0)
  { // Beim Aufruf ohne Argument ein Standard-
    x=x_;                              // konstruktor
    display("  Konstruktor: ", x);
  }

  C (const C& c)
  {
    x=c.x;
    display("  Kopierkonstruktor: ", x);
  }

  C& operator=(const C& c)
  {
    x=c.x;
    display("  operator=: ", x);
    return *this;
  }

  ~C ()
  {
    display("  Destruktor: ",x);
  }

  friend int f3(C c);
  friend int f4(const C& c);
  friend C operator+(const C& c1,const C& c2);
};

C f1(int i)
{
return C(i);
}

C f2(int i)
{
C tmp(i);
return tmp;
}


int f3(C c)
{
return c.x;
}

int f4(const C& c)
{
return c.x;
}

void test1()
{
display("vor C x=C(1)");
C x=C(1); //   Konstruktor: 1
C y=x;    //   Kopierkonstruktor: 1
display("vor x=y");
x=y;      //   operator=: 1
display("vor C z(x)");
C z(x);   //   Kopierkonstruktor: 1
display("vor f1(2)");
f1(2);    //   Konstruktor: 2
          //   Destruktor: 2
```

```
display("vor f2(3)");
f2(3);    //   Konstruktor: 3
          //   Kopierkonstruktor: 3
          //   Destruktor: 3
          //   Destruktor: 3
display("vor f3(4)");
f3(4);    //   Konstruktor: 4
          //   Destruktor: 4
display("vor f3(x)");
f3(x);    //   Kopierkonstruktor: 1
          //   Destruktor: 1
display("vor f4(4)");
f4(4);    //   Konstruktor: 4
          //   Destruktor: 4
display("vor f4(x)");
f4(x);    //   keine Ausgabe
display("Ende von test1");
}         //   Destruktor: 1
          //   Destruktor: 1
          //   Destruktor: 1


class D {
 C c1;
 C c2;
};

void test2()
{
display("vor D d1");
D d1;    //  Konstruktor: 0
         //  Konstruktor: 0
display("vor D d2=d1");
D d2=d1; //  Kopierkonstruktor: 0
         //  Kopierkonstruktor: 0
display("vor d2=d1");
d2=d1;   //  operator=: 0
         //  operator=: 0
display("nach d2=d1");
}        //  Destruktor: 0
         //  Destruktor: 0
         //  Destruktor: 0
         //  Destruktor: 0


// --------------- Aufgabe 3 -------------------------------

void MemCpy()
{
AnsiString s="Das kann zu flachen Kopien führen.";
Form1->Memo1->Lines->Add(s);
}


// --------------- Aufgabe 4 -------------------------------

#include "..\..\CppUtils\TimeUtils.h"
#include <vector>

void test_PrefixPostfixClick()
{
Form1->Memo1->Lines->Add("Aussagekräftige Laufzeiten sind nur in einer "
                         "Release Konfiguration möglich");

using std::vector;
const int n=100000;
typedef vector<int> T;
T v(n);
for (int i=0; i<n; i++) v.push_back(i);
```

```
CHRTimer t1,t2;
int s=0;
t1.Start();
for (T::iterator i=v.begin(); i!=v.end(); i++)
  s+=*i;;
t1.End();
t2.Start();
for (T::iterator i=v.begin(); i!=v.end(); ++i)
  s+=*i;;
t2.End();

CHRTimer t10,t20;
t10.Start();
for (int i=0; i<n; i++)
  s+=i;;
t10.End();
t20.Start();
for (int i=0; i<n; ++i)
  s+=i;;
t20.End();
Form1->Memo1->Lines->Add("Postfix:"+t1.TimeStr());
Form1->Memo1->Lines->Add("Präfix: "+t2.TimeStr());

Form1->Memo1->Lines->Add("int Postfix:"+t10.TimeStr());
Form1->Memo1->Lines->Add("int Präfix: "+t20.TimeStr());
}


// --------------- Aufgabe 5 -------------------------------

#include "..\..\cpputils\FixedP64.h"

// Der Datentyp Currency ist im wesentlichen so definiert wie hier der
// Datentyp Festkomma64.

typedef FixedP64 Festkomma64;

Festkomma64 Est2005(const Festkomma64& x) //
{ // Einkommensteuertarif 2005 nach § 32a (1)
Festkomma64 est;
if (x <= 7664) est = 0;
  // Die Bedingung (0 <= x) && (x <= 7664) würde ESt für negative Werte
  // von x (negative Einkommen sind mit Abschreibungen oder Verlusten
  // möglich) undefiniert lassen.
else if (x <= 12739)
  { //     Die Bedingung (x > 7664) braucht hier nicht mehr geprüft zu
    // werden, da sie sich bereits aus der Negation der letzten Bedingung
    // ergibt.
    Festkomma64 y = (x-7664.0)/10000;
    est = (883.74*y+1500)*y;
  }
else if (x <= 52151)
  {
    Festkomma64 z = (x-12739.0)/10000;
    est = (228.74*z + 2397)*z +989;
  }
else est = 0.42*x - 7914;
return est;
};

#include "..\..\cpputils\testUtils.h"

bool assertEqualEst(FixedP64 x,FixedP64 Soll)
{
return assertEqual(Est2005(x).toStr(),Soll.toStr(),"msg");
}

bool test_Est2005()
```

```
{
// http://home.t-online.de/home/parmentier.ffm/steuer.htm?steuer01.htm
// http://www.steuer.niedersachsen.de/Service/EStEuroTarif3.htm
//
http://www.ey.com/global/download.nsf/Germany/ST_Gestaltungsueberlegungen_zur_Jah
reswende_2004_2005/$file/Gestaltungsueberlegungen.pdf
bool result = true;

if (!assertEqualEst(0,0)) result=false;

if (!assertEqualEst(-1,0)) result=false;
if (!assertEqualEst(0,0)) result=false;

if (!assertEqualEst(7663,0)) result=false;
if (!assertEqualEst(7664,0)) result=false;
if (!assertEqualEst(7665,0)) result=false;
if (!assertEqualEst(7704,6)) result=false;
if (!assertEqualEst(7740,11)) result=false;
if (!assertEqualEst(7812,22)) result=false;

if (!assertEqualEst(9684,339)) result=false;
if (!assertEqualEst(11736,757)) result=false;

if (!assertEqualEst(12708,981)) result=false;
if (!assertEqualEst(12740,989)) result=false;
if (!assertEqualEst(12741,989)) result=false;
if (!assertEqualEst(12744,990)) result=false;

if (!assertEqualEst(52151,13989)) result=false;
if (!assertEqualEst(52152,13989)) result=false;
if (!assertEqualEst(52153,13990)) result=false;
if (!assertEqualEst(52092,13964)) result=false;

if (!assertEqualEst( 20000, 2850)) result=false;
if (!assertEqualEst( 40000, 9223)) result=false;
if (!assertEqualEst( 60000,17286)) result=false;
if (!assertEqualEst( 80000,25686)) result=false;
if (!assertEqualEst(100000,34086)) result=false;
if (!assertEqualEst(120000,42486)) result=false;
if (!assertEqualEst(140000,50886)) result=false;
if (!assertEqualEst(160000,59286)) result=false;
if (!assertEqualEst(180000,67686)) result=false;
if (!assertEqualEst(200000,76086)) result=false;

return result;
}

// --------------- Aufgabe 6 -------------------------------

void KonstrAufrufeVermeiden()
{
AnsiString s="Indem man sie private deklariert. ";
Form1->Memo1->Lines->Add(s);
}

// --------------- Aufgabe 7 -------------------------------

C operator+(const C& c1,const C& c2)
{ // friend in der Klasse C
display("  operator+: ", c1.x+c2.x);
return C(c1.x+c2.x);
}

void test3()
{
C s1,s2,s3,s4;
display("1. vor + ");
s1=C(1)+C(3);
display("2. vor + ");
```

```
C x=C(5),y=C(7);
s2=x+y;
display("3. vor + ");
s3=C(9);
s3=s3+C(11);
display("4. vor + ");
s4=x;
s4=s4+y;
display("Ende von test");
}
```

11.6.7 Aufgabe 6.2.11, 1. (Quadrat.h)

```
// File: C:\Loesungen_CB2006\Kap_6\6.2\Quadrat.h

#ifndef QUADRAT_H
#define QUADRAT_H
#include <vcl.h> // für AnsiString
class Quadrat{  // Elementfunktionen außerhalb definieren
  double a;
 public:
  Quadrat(double Seitenlaenge);
  double Seitenlaenge() const;
  void setze_Seitenlaenge(double Seitenlaenge);
  double Flaeche()   const;
  double Umfang()    const;
  AnsiString toStr() const;
};

#endif // QUADRAT_H
```

11.6.8 Aufgabe 6.2.11, 1. (Quadrat.cpp)

```
// File: C:\Loesungen_CB2006\Kap_6\6.2\Quadrat.cpp

#include "Quadrat.h"
Quadrat::Quadrat(double Seitenlaenge):a(Seitenlaenge){};
double Quadrat::Seitenlaenge() const
{
return a;
}

void Quadrat::setze_Seitenlaenge(double Seitenlaenge)
{
a=Seitenlaenge;
}

double Quadrat::Flaeche()   const
{
return a*a;
}

double Quadrat::Umfang()    const
{
return 4*a;
}

AnsiString Quadrat::toStr() const
```

```
{
return "Quadrat mit Seitenlänge "+FloatToStr(a);
}
```

11.6.9 Aufgabe 6.2.11 1. (Kreis.h)

```
// File: C:\Loesungen_CB2006\Kap_6\6.2\Kreis.h

#ifndef KREIS_H
#define KREIS_H

// Da alle Elementfunktionen von Kreis innerhalb der Klasse definiert
// werden, sind alle diese Definitionen in der Header-Datei enthalten.

const double pi=3.14159265358979323846;

class Kreis{
  double r;
 public:
  Kreis(double Radius):r(Radius) {}

  double Radius() const
  {
  return r;
  }

  void setzeRadius(double Radius)
  {
  r=Radius;
  }

  double Flaeche() const
  {
  return r*r*pi;
  }

  double Umfang() const
  {
  return 2*r*pi;
  }

  AnsiString toStr() const
  {
  return "Kreis mit Radius "+FloatToStr(r);
  }
};

#endif
```

11.6.10 Aufgabe 6.2.11 1. (Kreis.cpp)

```
// File: C:\Loesungen_CB2006\Kap_6\6.2\Kreis.cpp

// Da alle Elementfunktionen von Kreis innerhalb der Klasse definiert
// werden, sind alle diese Definitionen in der Header-Datei enthalten.
```

11.6.11 Aufgabe 6.2.11

```
// File: C:\Loesungen_CB2006\Kap_6\6.2\2_11.cpp

// --------------- Aufgaben 6.2.11 -------------------------
// --------------- Aufgaben 1 und 2 ------------------------
// a)
#include "Quadrat.h"
#include "Kreis.h"

// "Quadrat.cpp" mit "Projekt|Dem Projekt hinzufügen"
// dem Projekt hinzufügen
void Aufg1_2()
{
// b)
Quadrat q(10);
Form1->Memo1->Lines->Add(q.toStr());
Form1->Memo1->Lines->Add(q.Umfang());
Form1->Memo1->Lines->Add(q.Flaeche());
Quadrat* pq=new Quadrat(10);
Form1->Memo1->Lines->Add(pq->toStr());
Form1->Memo1->Lines->Add(pq->Umfang());
Form1->Memo1->Lines->Add(pq->Flaeche());

Kreis k(10);
Form1->Memo1->Lines->Add(k.toStr());
Form1->Memo1->Lines->Add(k.Umfang());
Form1->Memo1->Lines->Add(k.Flaeche());
Kreis* pk=new Kreis(10);
Form1->Memo1->Lines->Add(pk->toStr());
Form1->Memo1->Lines->Add(pk->Umfang());
Form1->Memo1->Lines->Add(pk->Flaeche());
}
/*
c)
Eine Header-Datei wird bei jeder Kompilation komplett übersetzt.
Wenn sie Funktionsdefinitionen enthält, werden auch diese jedesmal
neu übersetzt. Wenn sie dagegen nur Deklarationen enthält, müssen
die Elementfunktionen nur nach einer Änderung neu übersetzt werden.

Das kann bei großen Projekten mit beträchtlich kürzeren
Übersetzungszeiten verbunden sein.

Die außerhalb der Klasse definierten Funktionen sind im Gegensatz
zu den innerhalb der Klasse definierten Funktionen nicht automatisch
inline-Funktion. Um eine damit verbundene eventuell längere Laufzeit
des Programms zu vermeiden, muss man sie explizit als inline-Funktion
kennzeichnen und doch wieder in die Header-Datei aufnehmen, da
inline-Funktionen keine externe Bindung haben.
*/


// --------------- Aufgaben 3 ------------------------------

class Singleton {
  static Singleton* instance_;
 public:
  static Singleton* Instance();
  int data; // besser private und Zugriffsfunktionen
 private: // protected, falls Singleton als Basisklasse verwendet werden soll
  Singleton(){data=17;};
};

Singleton* Singleton::Instance()
{
  if (instance_==0)
    instance_=new Singleton();
```

```
  return instance_;
};

Singleton* Singleton::instance_=0;

Singleton* p1=Singleton::Instance();

void Aufg_Singleton()
{
Singleton* p2=Singleton::Instance();
// p2->data=19;
Form1->Memo1->Lines->Add(IntToStr(p1->data)); // p1->data=19
Form1->Memo1->Lines->Add(IntToStr(p2->data));
// falls p1 ein gewöhnliches globales Objekt wäre, hätte die
// globale Variable den Wert 17
}
```

11.6.12 Aufgabe 6.2

```
// File: C:\Loesungen_CB2006\Kap_6\6.2\KlassenDTU.cpp

#include <vcl.h>
#pragma hdrstop
#include "KlassenDTU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}


// --------------- Aufgaben 6.2 ----------------------------

// --------------- Aufgaben 6.2.2 --------------------------

#include "2_2.cpp"

void __fastcall TForm1::Button_2_2_1Click(TObject *Sender)
{
Konstruktoren();
}

void __fastcall TForm1::Button_2_2_2Click(TObject *Sender)
{
testKreis();
}

void __fastcall TForm1::Button_2_2_3Click(TObject *Sender)
{
Rechteck1_();
}

void __fastcall TForm1::Button_2_2_4Click(TObject *Sender)
{
MeinString_Konstr();
}


// --------------- Aufgaben 6.2.4 --------------------------
```

```
#include "2_4.cpp"

void __fastcall TForm1::Button_2_4_1Click(TObject *Sender)
{
TestBruch(10);
}

void __fastcall TForm1::Button_2_4_2Click(TObject *Sender)
{
test_AssozCont();
}


// --------------- Aufgaben 6.2.6 --------------------------

#include "2_6.cpp"

void __fastcall TForm1::Button_2_6_1Click(TObject *Sender)
{
SpezEltFktNotwendig();
}

void __fastcall TForm1::Button_2_6_2Click(TObject *Sender)
{
display("----------- test1 -----------");
test1();
display("----------- test2 -----------");
test2();
}

void __fastcall TForm1::Button_2_6_3Click(TObject *Sender)
{
MemCpy();
}

void __fastcall TForm1::Button_2_6_4Click(TObject *Sender)
{
test_PrefixPostfixClick();
}

void __fastcall TForm1::Button_2_6_5Click(TObject *Sender)
{
if (test_Est2005())
  Memo1->Lines->Add("All tests passed");
}

void __fastcall TForm1::Button_2_6_6Click(TObject *Sender)
{
KonstrAufrufeVermeiden();
}

void __fastcall TForm1::Button_2_6_7Click(TObject *Sender)
{
test3();
}


// --------------- Aufgaben 6.2.11 -------------------------

#include "2_11.cpp"

void __fastcall TForm1::Button_2_13_1Click(TObject *Sender)
{
Aufg1_2();
}

void __fastcall TForm1::Button_2_13_2Click(TObject *Sender)
{
```

```
Aufg_Singleton();
}
```

11.6.13 Aufgabe 6.3

```
// File: C:\Loesungen_CB2006\Kap_6\6.3\VererbU.cpp

#include <vcl.h>
#pragma hdrstop
#include "VererbU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}


// --------------- Aufgaben 6.3.5 --------------------------

// --------------- Aufgabe 1 -------------------------------

class C {
  int i,j;
 public:
  C(int x,int y): i(x),j(y)
  {
  Form1->Memo1->Lines->Add("Konstruktor C");
  }
  C(): i(0),j(0)
  {
  Form1->Memo1->Lines->Add("Standardkonstruktor C");
  }
  ~C()
  {
  Form1->Memo1->Lines->Add("Destruktor C");
  }
};

class D : public C {
  int k,a,b;
  C c;
 public:
  D(int x=1):c(x,1),a(x),b(0),k(19)
  {
  Form1->Memo1->Lines->Add("Konstruktor-1 D");
  }
  D(int x,int y, int z):C(x,y),a(1),b(2),c(x,y),k(z)
  {
  Form1->Memo1->Lines->Add("Konstruktor-2 D");
  }
  ~D()
  {
  Form1->Memo1->Lines->Add("Destruktor D");
  }
};

class E : public D {
  int m;
  C c;
```

```
  D b;
 public:
  E(int x,int y):b(y),c(2,3),m(x+y)
  {
  Form1->Memo1->Lines->Add("Konstruktor E");
  }
  ~E() {
  Form1->Memo1->Lines->Add("Destruktor E");
  }
};

void __fastcall TForm1::Button_3_5_1Click(TObject *Sender)
{
Form1->Memo1->Lines->Add("C c(1,2);");
C c(1,2);    // Konstruktor C
Form1->Memo1->Lines->Add("D d(1,2,3);");
D d(1,2,3);  // Konstruktor C
             // Konstruktor C
             // Konstruktor-2 D
Form1->Memo1->Lines->Add("E e(1,2);");
E e(1,2);    // Standardkonstruktor C
             // Konstruktor C
             // Konstruktor-1 D
             // Konstruktor C
             // Standardkonstruktor C
             // Konstruktor C
             // Konstruktor-1 D
             // Konstruktor E
}             // Destruktor E
             // Destruktor D
             // Destruktor C
             // Destruktor C
             // Destruktor C
             // Destruktor D
             // Destruktor C
             // Destruktor C
             // Destruktor D
             // Destruktor C
             // Destruktor C
             // Destruktor C


// --------------- Aufgabe 2 -------------------------------

class C1DPunkt {
  double x;
 public:
  C1DPunkt(double x_):x(x_)   {  }
  void setzeX(double x_) {x=x_;}
  double X() const { return x;}

  AnsiString toStr() const
  {
  return FloatToStr(x);
  }

  void anzeigen() const
  {
  Form1->Memo1->Lines->Add(toStr());
  }
};

class C2DPunkt:public C1DPunkt {
  double y;
 public:
  C2DPunkt(double x_,double y_):C1DPunkt(x_),y(y_)   { }
  void setzeY(double y_) { y=y_; }
  double Y() const { return y; }
```

```
  AnsiString toStr() const
  {
  return FloatToStr(X())+"|"+FloatToStr(y);
  }

  void anzeigen() const
  {
  Form1->Memo1->Lines->Add(toStr());
  }
};

class C3DPunkt : public C2DPunkt {
  double z;
 public:
  C3DPunkt(double x_,double y_,double z_):C2DPunkt(x_,y_),z(z_) {   }
  void setzeZ(double z_) {z=z_;}
  double Z() { return z;}

  AnsiString toStr()
  {
  return FloatToStr(X())+"|"+FloatToStr(Y())+"|"+FloatToStr(z);
  }

  void anzeigen()
  {
  Form1->Memo1->Lines->Add(toStr());
  }
};

void test_Punkte()
{
C1DPunkt p1(1);
C2DPunkt p2(1,2);
C3DPunkt p3(1,2,3);
p1.anzeigen();
p2.anzeigen();
p3.anzeigen();
}

void __fastcall TForm1::Button_3_5_2Click(TObject *Sender)
{
test_Punkte();
}


// --------------- Aufgabe 3 -------------------------------

// a)

// b1)

AnsiString b1=
"Wenn diese Klassen Strings der Klassen string bzw. AnsiString statt "
"Zeigern auf nullterminierte Strings enthalten, ist für keine dieser "
"Klassen ein explizit definierter Copy-Konstruktor, Zuweisungsoperator "
"und Destruktor notwendig. \r\n"
"Das erspart die Definition dieser Funktionen und damit viel Arbeit. "
"Deshalb sollte man auf Klassen mit Zeigern möglichst verzichten und "
"Stringklassen anstelle von Zeigern auf nullterminierte Strings verwenden.";

// b2)

AnsiString b2=
"Alle Klassen, die Zeiger enthalten, benötigen einen Destruktor, "
"Copy-Konstruktor und Zuweisungsoperator.";

// b3)

class Immobilie {
```

```
 protected: // Nur zur Vereinfachung - private und get-Funktionen wären
  char* Anschrift;                                          // besser
  double Kaufpreis;
 public:
  Immobilie(char* Anschrift_, double Kaufpreis_):Kaufpreis(Kaufpreis_)
  {
  Anschrift = new char[strlen(Anschrift_)+1];
  strcpy(Anschrift,Anschrift_);
  }

  ~Immobilie()
  {
  delete[] Anschrift;
  }

  AnsiString toStr() const
  {
  return " in "+AnsiString(Anschrift)+ " KP: "+FloatToStr(Kaufpreis) + " DM ";
  }

  Immobilie(const Immobilie& c);              // für b)
  Immobilie& operator=(const Immobilie& rhs); // für b)
};

void display(const Immobilie* i, TMemo* Memo)
{
Memo->Lines->Add(i->toStr());
}

class Gewerbeobjekt : public Immobilie {
  double Nutzflaeche;
  char* Nutzungsart; // z. B. "Büro", "Fabrik"
 public:
  Gewerbeobjekt(char* Anschrift_, double Kaufpreis_, double Nutzflaeche_,
     char*                        Nutzungsart_):Immobilie(Anschrift_,Kaufpreis_),
Nutzflaeche(Nutzflaeche_)
  {
  Nutzungsart=new char[strlen(Nutzungsart_)+1];
  strcpy(Nutzungsart,Nutzungsart_);
  }

  ~Gewerbeobjekt()
  {
  delete[] Nutzungsart;
  }

  AnsiString toStr() const
  {
  return "Gewerbeobjekt"+ Immobilie::toStr()+
    " Nutzfläche: "+FloatToStr(Nutzflaeche)+" qm Nutzungsart: "+Nutzungsart;
  }
  Gewerbeobjekt(const Gewerbeobjekt& c);              // für b)
  Gewerbeobjekt& operator=(const Gewerbeobjekt& rhs); // für b)
};

void display(const Gewerbeobjekt* i, TMemo* Memo)
{
Memo->Lines->Add(i->toStr());
}

void testImmo()
{
//  Gewerbeobjekt(char* Anschrift_, double Kaufpreis_, double Nutzflaeche_, char*
Nutzungsart_)
Gewerbeobjekt gew("München",400000,200,"Lagerhalle");
display(&gew, Form1->Memo1);
}
```

```
Immobilie::Immobilie(const Immobilie& x):Kaufpreis(x.Kaufpreis)
{
Anschrift = new char[strlen(x.Anschrift)+1];
strcpy(Anschrift,x.Anschrift);
}

Immobilie& Immobilie::operator=(const Immobilie& rhs)
{
if (this!=&rhs)
  {
    delete Anschrift;
    Anschrift = new char[strlen(rhs.Anschrift)+1];
    strcpy(Anschrift,rhs.Anschrift);
    Kaufpreis=rhs.Kaufpreis;
  }
return *this;
};

Gewerbeobjekt::Gewerbeobjekt(const                                  Gewerbeobjekt&
x):Immobilie(x),Nutzflaeche(x.Nutzflaeche)
{
Nutzungsart= new char[strlen(x.Nutzungsart)+1];
strcpy(Nutzungsart,x.Nutzungsart);
}

Gewerbeobjekt& Gewerbeobjekt::operator=(const Gewerbeobjekt& rhs)
{
if (this!=&rhs)
  {
    Immobilie::operator=(rhs); // Aufruf von this->C::operator=
    delete Nutzungsart;
    Nutzungsart = new char[strlen(rhs.Nutzungsart)+1];
    strcpy(Nutzungsart,rhs.Nutzungsart);
    Kaufpreis=rhs.Kaufpreis;
  }
return *this;
};

// c)

AnsiString c=
"Diese Hierarchie ist einfacher als die Definition der einzelnen Klassen, "
"bei denen keine Elemente wiederverwendet werden. In dieser Hierarchie "
"ist auch nur für die Klassen ein explizit definierter Copy-Konstruktor, "
"Zuweisungsoperator und Destruktor notwendig, die Zeiger enthalten. Bei "
"Klassen, die Zeiger aus den Basisklassen erben, reicht der Aufruf dieser "
"Funktionen aus der Basisklasse aus. ";


void __fastcall TForm1::Button_3_5_3Click(TObject *Sender)
{
testImmo();
Memo1->Lines->Add("b1)");
Memo1->Lines->Add(b1);
Memo1->Lines->Add("b2)");
Memo1->Lines->Add(b2);
Memo1->Lines->Add("c)");
Memo1->Lines->Add(c);
Memo1->Lines->Add("d)");
Memo1->Lines->Add("");
}


// --------------- Aufgabe 4 -------------------------------

// namespace N_Aufgabe_4 {

// a)
```

```
class CQuadrat1{
  double a;
 public:
  CQuadrat1(double a_):a(a_) {}
};

class CRechteck1:public CQuadrat1{
  double b;
 public:
  CRechteck1(double a_, double b_):CQuadrat1(a_),b(b_) {}
};

// b)

class CRechteck2{
  double a,b;
 public:
  CRechteck2(double a_,double b_):a(a_),b(b_){};
};

class CQuadrat2:public CRechteck2 {
 public:
  CQuadrat2(double a_):CRechteck2(a_,a_) {}
};

// c)

// Bei der Hierarchie b) benötigt jedes Quadrat doppelt soviel
// Speicherplatz wie bei der unter b).

//} // end of namespace N_Aufgabe_4

void __fastcall TForm1::Button_3_5_4Click(TObject *Sender)
{
// N_Aufgabe_4::test();
}


// --------------- Aufgabe 5 -------------------------------


void __fastcall TForm1::Button_3_5_5Click(TObject *Sender)
{
AnsiString s=
"Da Konstruktoren nicht vererbt werden, stehen diese in der abgeleiteten "
"Klasse nicht zur Verfügung. Bei einer Klasse mit vielen nützlichen "
"Konstruktoren wie z.B. einer Stringklasse kann das ein gravierender "
"Nachteil sein. Man kann diese zwar in der abgeleiteten Klasse wieder "
"alle definieren. Es ist aber fraglich, ob sich dieser Aufwand lohnt und "
"man nicht besser beim Aufruf eines vorhandenen Konstruktors eine "
"Funktion wie StrToInt oder StrToFloat verwendet.";
Memo1->Lines->Add("6.");
Memo1->Lines->Add(s);
}


// --------------- Aufgabe 6 -------------------------------

// Indem man in der Basisklasse einen protected Destruktor definiert.

class C6{
 protected:
   ~C6(){}
};

class D6 : public C6 {
};

void test_6()
```

```
{
//C6 c6; // Fehler: Zugriff auf Destruktor nicht möglich
D6 d6;
}

void __fastcall TForm1::Button_3_5_6Click(TObject *Sender)
{
test_6();
}



// --------------- Aufgaben 6.3.9 --------------------------

// --------------- Aufgabe 1 -------------------------------

void __fastcall TForm1::Button_3_9_1Click(TObject *Sender)
{
AnsiString a=
"Ein Automobil hat einen Motor und Räder. Vermutlich wird nie jemand "
"sagen, dass ein Automobil ein Motor oder ein Rad ist.";

AnsiString b=
"Eine Katze bzw. ein Hund ist Tier. Von einer 'hat ein'-Beziehung kann "
"man höchstens dann reden, wenn ein Hund eine Katze gefressen hat oder "
"eine Katze einen Hund hält.";

AnsiString c=
"Ein Landfahrzeug bzw. ein Wasserfahrzeug ist ein Fahrzeug. Ein Automobil "
"ist ein Landfahrzeug, ein Segelboot ist ein Wasserfahrzeug. Von einer "
"'hat ein'-Beziehung kann man höchstens dann reden, wenn ein "
"Wasserfahrzeug ein Landfahrzeug transportiert.";

AnsiString d=
"Hier sind beide Sichtweisen möglich: Ein Abteilungsleiter und eine "
"Sekretärin sind Mitarbeiter. Ein Abteilungsleiter hat eine Sekretärin "
"und Mitarbeiter. Im ersten Fall werden die Mitarbeiter einer Firma "
"modelliert. Im zweiten Fall wird eine Abteilung modelliert. Es soll "
"auch Sekretärinnen geben, die einen Abteilungsleiter haben.";

Memo1->Lines->Add("1.");
Memo1->Lines->Add("a)");
Memo1->Lines->Add(a);
Memo1->Lines->Add("b)");
Memo1->Lines->Add(b);
Memo1->Lines->Add("c)");
Memo1->Lines->Add(c);
Memo1->Lines->Add("d)");
Memo1->Lines->Add(d);
}


// --------------- Aufgabe 2 -------------------------------

namespace N_3_9_2 {

// a)

class CQuadrat1{
  double a;
 public:
  CQuadrat1(double a_):a(a_) {}
  double Flaeche() { return a*a; }
  double Umfang()  { return 4*a; }
};

class CRechteck1:public CQuadrat1{
  double b;
 public:
```

```
  CRechteck1(double a_, double b_):CQuadrat1(a_),b(b_) {}
  // hier sind die geerbten Funktionen Flaeche und
  // Umfang nicht korrekt.
};

// b)

class CRechteck2{
  double a,b;
 public:
  CRechteck2(double a_,double b_):a(a_),b(b_){};
  double Flaeche() { return a*b; }
  double Umfang()  { return 2*(a+b); }
};

class CQuadrat2:public CRechteck2 {
 public:
  CQuadrat2(double a_):CRechteck2(a_,a_) {}
  // hier sind die geerbten Funktionen Flaeche und
  // Umfang korrekt.
};

} // end of namespace N_3_9_2

void __fastcall TForm1::Button_3_9_2Click(TObject *Sender)
{
AnsiString s=
"Bei der Ableitung eines Quadrats von einem Rechteck sind die aus der "
"Basisklasse geerbten Funktionen Flaeche und Umfang auch in der "
"abgeleiteten Klasse korrekt. Bei der umgekehrten Ableitung ist das "
"nicht richtig.";
Memo1->Lines->Add("2.");
Memo1->Lines->Add(s);
}

// --------------- Aufgabe 3 -------------------------------

void __fastcall TForm1::Button_3_9_3Click(TObject *Sender)
{
AnsiString a=
"a) Hier besteht zwischen jeder abgeleiteten Klasse und ihrer "
"   Basisklasse eine ist-ein-Beziehung"
"b) Hier besteht ebenfalls zwischen jeder abgeleiteten Klasse "
"   und ihrer Basisklasse eine ist-ein-Beziehung"
"c) Hier besteht zwischen der abgeleitete Klasse EinFamH und ihrer "
"   Basisklasse keine ist-ein-Beziehung. Außerdem werden die "
"   Datenelemente von Immobilie zweimal geerbet und sind doppelt"
"d) Hier besteht zwischen der abgeleitete Klasse EinFamH und ihrer "
"   Basisklasse EigentW keine ist-ein-Beziehung. "
"e) Hier besteht zwischen der abgeleitete Klasse EinFamH und ihrer "
"   Basisklasse Grundst keine ist-ein-Beziehung. "
"Von allen diesen Hierarchien sind also nur die unter a) und b) gut";
Memo1->Lines->Add("3.");
Memo1->Lines->Add("a)");
Memo1->Lines->Add(a);
AnsiString b=
"Durch eine Klasse wie Wohnimmobilie, von der eine Eigentumswohnung "
"und ein Einfamilienhaus abgeleitet werden. ";
Memo1->Lines->Add("b)");
Memo1->Lines->Add(b);
}


// --------------- Aufgabe 4 -------------------------------

void __fastcall TForm1::Button_3_9_4Click(TObject *Sender)
{
AnsiString s=
"Mit einer Konversion der Basisklasse in eine abgeleitete Klasse könnte "
```

```
"man über ein Objekt der abgeleiteten Klasse ein Element der "
"Basisklasse ansprechen, das überhaupt nicht existiert. ";
Memo1->Lines->Add("5.");
Memo1->Lines->Add(s);
}


// --------------- Aufgaben 6.3.11 -------------------------

// --------------- Aufgabe 1 -------------------------------

#include <cmath>
namespace N_3_11_1 {

// --------------- Aufgabe 1 -------------------------------

class CKreis{
  double r;
  static const double pi;
 public:
  CKreis(double Radius)
  {
  r=Radius;
  }

  double Flaeche()
  {
  return r*r*pi;
  }

  double Umfang()
  {
  return 2*r*pi;
  }

  AnsiString toStr()
  {
  return "Kreis mit Radius "+FloatToStr(r);
  }
};

const double CKreis::pi=3.14;

class C2DPunkt {
  double x,y;
 public:
  C2DPunkt(double x_,double y_):x(x_),y(y_) { }

  double Abstand()
  {
  return std::sqrt(x*x+y*y);
  }

  AnsiString toStr()
  {
  return FloatToStr(x)+"|"+FloatToStr(y);
  }
};

// a)

namespace N_a {

class C2DKreis:public CKreis, public C2DPunkt {
 public:
  C2DKreis(double Radius=1, double x=0, double y=0):
     CKreis(Radius), C2DPunkt(x,y) { }

  C2DKreis(double Radius, C2DPunkt pos):
```

```
     CKreis(Radius), C2DPunkt(pos) { }
/* Eine in dieser Klasse definierte Funktion toStr verdeckt die Funktionen
   toStr der Basisklasse:

  AnsiString toStr()
  {
  return CKreis::toStr()+" Pos: "+C2DPunkt::toStr();
  }
*/
};

AnsiString s1=
"C2DKreis erbt alle public Elementfunktionen der Basisklassen, also "
"Flaeche, Umfang und toStr. Ein Aufruf von toStr über ein Objekt der "
"Klasse C2DKreis ist dann mehrdeutig.";

AnsiString s2=
"Diese Hierarchie ist nicht geeignet, da ein Kreis mit einer Position "
"zwar ein Kreis, aber kein Punkt ist.";

} // end of namespace N_a


// b)

namespace N_b {

class C2DKreis {
  CKreis Kreis;
  C2DPunkt Position;
 public:
  C2DKreis(double Radius=1, double x=0, double y=0):
     Kreis(Radius), Position(x,y) { }

  C2DKreis(double Radius, C2DPunkt pos):
     Kreis(Radius), Position(pos) { }

  // Eine in dieser Klasse definierte Funktion toStr verdeckt keine
  // Funktionen der Basisklasse:
  AnsiString toStr()
  {
  return Kreis.toStr()+" Pos: "+Position.toStr();
  }
};

AnsiString s1=
"Da hier keine Vererbung verwendet wird, erbt C2DKreis keine "
"Elementfunktionen der Klasse CKreis und kann die Funktionen dieser "
"Klasse CKreis auch nicht wiederverwenden.";

AnsiString s2=
"Diese Hierarchie ist nicht geeignet, da ein Kreis mit einer Position ein "
"Kreis ist, aber die Funktionen der Klasse CKreis nicht verwendet.";

} // end of namespace N_b


// c)

namespace N_c {

class C2DKreis:public CKreis {
  C2DPunkt Position;
 public:
  C2DKreis(double Radius=1, double x=0, double y=0):
     CKreis(Radius), Position(x,y) { }

  C2DKreis(double Radius, C2DPunkt pos):
     CKreis(Radius), Position(pos) { }
```

```
};

AnsiString s1=
"C2DKreis erbt alle public Elementfunktionen der Basisklasse, also "
"Flaeche, Umfang und toStr. Alle diese Funktionen liefern auch über "
"ein Objekt der Klasse C2DKreis richtige Ergebnisse. Die "
"Elementfunktionen von C2DPunkt stehen dagegen nicht zur Verfügung.";

AnsiString s2=
"Diese Hierarchie ist geeignet, da ein C2DKreis ein Kreis, aber kein "
"C2DPunkt ist.";

} // end of namespace N_c


// d)

namespace N_d {

class C2DKreis{// Lösung von Aufgabe 8.2.2.2
  double r;
  C2DPunkt Position;
 public:
  C2DKreis(double Radius=1, double x=0, double y=0):
     r(Radius), Position(x,y) { } // Reihenfolge der Deklaration

  C2DKreis(double Radius, C2DPunkt pos):
     r(Radius), Position(pos) { } // Reihenfolge der Deklaration

  AnsiString toStr()
  {
  return "Kreis mit Radius "+FloatToStr(r)+" in Position "+Position.toStr();
  }
};

AnsiString s1=
"Wenn man in der Klasse C2DKreis die Funktionen Flaeche, Abstand usw. "
"für einen Kreis haben will, muss man sie erneut definieren. Diese "
"Redundanz sollte man vermeiden.";

AnsiString s2=
"Abgesehen davon ist diese Alternative geeignet.";

} // end of namespace N_d


void test()
{
// a)
Form1->Memo1->Lines->Add("a)");
N_a::C2DKreis ka(1,2,3);
Form1->Memo1->Lines->Add(FloatToStr(ka.Umfang()));  // das geht
Form1->Memo1->Lines->Add(FloatToStr(ka.Abstand())); // das geht
Form1->Memo1->Lines->Add(FloatToStr(ka.Flaeche())); // das geht
//Form1->Memo1->Lines->Add(ka.toStr());  // Fehler: Mehrdeutig

Form1->Memo1->Lines->Add(N_a::s1);
Form1->Memo1->Lines->Add("");
Form1->Memo1->Lines->Add(N_a::s2);

// b)
Form1->Memo1->Lines->Add("b)");
N_b::C2DKreis kb(1,2,3);
// Form1->Memo1->Lines->Add(FloatToStr(kb.Umfang()));  // geht nicht
// Form1->Memo1->Lines->Add(FloatToStr(kb.Abstand())); // geht nicht
// Form1->Memo1->Lines->Add(FloatToStr(kb.Flaeche())); // geht nicht
Form1->Memo1->Lines->Add(kb.toStr());                  // das geht

Form1->Memo1->Lines->Add(N_b::s1);
```

```
Form1->Memo1->Lines->Add("");
Form1->Memo1->Lines->Add(N_b::s2);

// c)
Form1->Memo1->Lines->Add("c)");
N_c::C2DKreis kc(1,2,3);
Form1->Memo1->Lines->Add(FloatToStr(kc.Umfang()));     // das geht
// Form1->Memo1->Lines->Add(FloatToStr(kc.Abstand())); // geht nicht
Form1->Memo1->Lines->Add(FloatToStr(kc.Flaeche()));    // das geht
Form1->Memo1->Lines->Add(kc.toStr());                  // das geht

Form1->Memo1->Lines->Add(N_c::s1);
Form1->Memo1->Lines->Add("");
Form1->Memo1->Lines->Add(N_c::s2);

// d)
Form1->Memo1->Lines->Add("d)");
N_d::C2DKreis kd(1,2,3);
// Form1->Memo1->Lines->Add(FloatToStr(kd.Umfang()));  // geht nicht
// Form1->Memo1->Lines->Add(FloatToStr(kd.Abstand())); // geht nicht

Form1->Memo1->Lines->Add(N_d::s1);
Form1->Memo1->Lines->Add("");
Form1->Memo1->Lines->Add(N_d::s2);
}


} // end of namespace N_3_11

void __fastcall TForm1::Button_3_11_1Click(TObject *Sender)
{
N_3_11_1::test();
}
```

11.6.14 Aufgabe 6.4

```
// File: C:\Loesungen_CB2006\Kap_6\6.4\VirtFktU.cpp

#include <vcl.h>
#pragma hdrstop
#include "VirtFktU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}


// --------------- Aufgaben 6.4.3 --------------------------

// --------------- Aufgabe 1 -------------------------------

#include <cmath> // für fabs und sqrt

class C1DPunkt{
  double x;
 public:
  C1DPunkt(double x_):x(x_){  }
  void setzeX(double x_) {x=x_;}
```

```
  double X() const { return x; }

  virtual AnsiString toStr() const
  {
  return "("+FloatToStr(x) + ")";
  }

// c)
  void anzeigen() const
  {
  Form1->Memo1->Lines->Add(toStr());
  }

// e)
  double length() const
  {
  return std::fabs(x);
  }
};

class C2DPunkt:public C1DPunkt{
  double y;
 public:
  C2DPunkt(double x_, double y_):C1DPunkt(x_), y(y_) {  }
  void setzeY(double y_) { y=y_; }
  double Y() const { return y; }

  AnsiString toStr() const
  {
  return "("+FloatToStr(X()) + "|" + FloatToStr(Y())+")";
  }

// e)
  double length()
  {
  return std::sqrt(X()*X() + Y()*Y());
  }
};

class C3DPunkt:public C2DPunkt{
  double z;
 public:
  C3DPunkt(double x_, double y_, double z_):C2DPunkt(x_,y_), z(z_) {  }
  void setzeZ(double z_) { z=z_; }
  double Z() const { return z; }

  AnsiString toStr() const
  {
  return "("+FloatToStr(X()) + "|" + FloatToStr(Y())+"|" + FloatToStr(z)+")";
  }

// e)
  double C3DPunkt::length()
  {
  return std::sqrt(X()*X() + Y()*Y() + z*z);
  }
};

// b)
void show(C1DPunkt& p)
{
Form1->Memo1->Lines->Add(p.toStr());
};

// d)
// Unter Projekt|Optionen|Pfade ...
//    einen Pfad auf das Verzeichnis der Boost Bibliothek setzen
#include <boost/shared_ptr.hpp>
```

```
void testVirtuelleFunktionen()
{
// a)
Form1->Memo1->Lines->Add("toStr()");
C1DPunkt* p1=new C1DPunkt(1);
Form1->Memo1->Lines->Add(p1->toStr());
p1=new C2DPunkt(2,3);
Form1->Memo1->Lines->Add(p1->toStr());
p1=new C3DPunkt(4,5,6);
Form1->Memo1->Lines->Add(p1->toStr());

// b)
Form1->Memo1->Lines->Add("show");
C1DPunkt q1(7);
C2DPunkt q2(8,9);
C3DPunkt q3(10,11,12);
show(q1); // C1DPunkt::toStr
show(q2); // C2DPunkt::toStr
show(q3); // C3DPunkt::toStr

// c)
Form1->Memo1->Lines->Add("anzeigen");
q1.anzeigen(); // C1DPunkt::toStr
q2.anzeigen(); // C2DPunkt::toStr
q3.anzeigen(); // C3DPunkt::toStr

// d)
Form1->Memo1->Lines->Add("shared_ptr");
using boost::shared_ptr;
shared_ptr<C1DPunkt> s1(new C1DPunkt(1));
Form1->Memo1->Lines->Add(s1->toStr());
s1.reset(new C2DPunkt(2,3));
Form1->Memo1->Lines->Add(s1->toStr());
s1.reset(new C3DPunkt(4,5,6));
Form1->Memo1->Lines->Add(s1->toStr());
}

void __fastcall TForm1::Button_4_3_1Click(TObject *Sender)
{
testVirtuelleFunktionen();
}

// --------------- Aufgabe 2 -------------------------------

void __fastcall TForm1::Button_4_3_2Click(TObject *Sender)
{
AnsiString s=
"Diese Funktionen können nicht virtuell definiert werden, da alle ihre "
"Parameterlisten verschieden sind.";
Memo1->Lines->Add(s);
}

// --------------- Aufgabe 3 -------------------------------

void __fastcall TForm1::Button_4_3_3Click(TObject *Sender)
{
/*
Wenn die Funktion D::f virtuell wäre, würde mit

  C c;
  D d;

der Aufruf

  g(d,c);

zum Aufruf der Funktion
```

```
  D::f(c)

führen. Diese Funktion würde auf c.e zugreifen, das in C aber gar nicht
existiert.
*/
}

// --------------- Aufgabe 4 -------------------------------

struct C {
  virtual int f(int i=1)  { return i; }
};

struct D:public C {
  virtual int f(int i=2)  { return i; }
};

void testDefaultArgumente()
{
C* pc=new D;
int i=pc->f(); // pc->f(1);
C* pd=new D;
int j=pd->f(); // pd->f(1);
// Das Default-Argument wird immer nach dem statischen Datentyp bestimmt
Form1->Memo1->Lines->Add(i);
Form1->Memo1->Lines->Add(j);
};

void __fastcall TForm1::Button_4_3_4Click(TObject *Sender)
{
testDefaultArgumente();
}


// --------------- Aufgaben 6.4.10 -------------------------

// --------------- Aufgabe 1 -------------------------------

#include <vector>
#include <fstream>
#include <sstream>
#include <string>
using std::endl;

namespace N_Aufg_4_10_1 {

class C2DPunkt{
  double x,y;
 public:
  C2DPunkt(double x_=0, double y_=0):x(x_),y(y_) {  };
//
  virtual ~C2DPunkt() { };

  double X() const {return x;}
  double Y() const {return y;}

  AnsiString toStr() const
  {
  return "("+FloatToStr(x)+"|"+FloatToStr(y)+")";
  }

  // Diese Funktionen sind für die Operationen mit den
  // Figuren notwendig:
  C2DPunkt (ifstream& f)
  { // erzeugt einen C2DPunkt mit den Daten aus dem ifstream
  f>>x>>y;
  }

  friend ostream& operator<<(ostream& f, const C2DPunkt& p);
```

```
  friend istream& operator>>(istream& f, C2DPunkt& p);

  void verschiebe_um(const C2DPunkt& pos)
  {
  x=x+pos.x;
  y=y+pos.y;
  };
};

ostream& operator<<(ostream& f, const C2DPunkt& p)
{
return f<<' '<<p.x<<' '<<p.y<<' ';
}

istream& operator>>(istream& f, C2DPunkt& p)
{
f>>p.x>>p.y;
return f;
}

// a) Verwende eine gemeinsame, abstrakte Basisklasse,
//    von der Kreis, Quadrat, Rechteck usw. abgeleitet
//    werden.

class Figur {
 public:
  virtual ~Figur(){}; // den virtuellen Destruktor nicht vergessen
// a)
  virtual double Flaeche()   const =0;
  virtual double Umfang()    const =0;

// b)
  virtual AnsiString toStr() const =0;
// c)
  virtual void zeichne()     const =0;
  virtual void zeichne(TImage* img)     const =0;
// c)
  virtual void write(ofstream& f) const =0;

// d)
  void loesche(TImage* img) const
  {
  TColor StiftFarbe=img->Canvas->Pen->Color;
  img->Canvas->Pen->Color=img->Canvas->Brush->Color; // Hintergrundfarbe
  zeichne(img); // rufe die rein virtuelle Funktion auf
  img->Canvas->Pen->Color=StiftFarbe;
  }

  virtual void verschiebe_Position_um(const C2DPunkt& deltaPos)=0;
};

// c)
class Zeichnung {
  typedef std::vector<Figur*> Container;
  typedef Container::iterator Iterator;
  // Zeiger auf abstrakte Klassen können verwendet werden
  Container c;
 public:
  Zeichnung(){ };

// c)
  void einfuegen(Figur* p)
  {
  c.push_back(p);
  }

// b)
  void zeichne()
  { // schreibt in Memo1
```

```
  for (Iterator i=c.begin(); i!=c.end(); i++)
    (*i)->zeichne();
  }

  void zeichne(TImage* img)
  { // zeichne auf img
  for (Iterator i=c.begin(); i!=c.end(); i++)
    (*i)->zeichne(img);
  }

// d)
  void loesche(TImage* img)
  {
  for (Iterator i=c.begin(); i!=c.end(); i++)
    (*i)->loesche(img);
  }

  void verschiebe_um(TImage* img, C2DPunkt deltaPos)
  {
  loesche(img);
  for (Iterator i=c.begin(); i!=c.end(); i++)
    (*i)->verschiebe_Position_um(deltaPos);
  zeichne(img);
  }

// e)
  void write(const char* Dateiname)
  {
  ofstream f(Dateiname);
  for (Iterator i=c.begin(); i!=c.end(); i++)
    (*i)->write(f);
  f.close(); // überflüssig, aber nicht falsch
  }

// f)
  void LeseZeichnung(const char* Dateiname);

  void toMemo(TMemo* m)
  {
  for (Iterator i=c.begin(); i!=c.end(); i++)
    m->Lines->Add((*i)->toStr().c_str());
  }

};


class Gerade:public Figur {
  C2DPunkt ap,ep; // Anfangs- und Endpunkt
 public:
  Gerade(C2DPunkt ap_, C2DPunkt ep_):ap(ap_),ep(ep_) {}

  Gerade(ifstream& f):ap(f),ep(f)
  {  } // Erzeugt eine Gerade mit den Daten aus dem ifstream

// a)
  virtual double Flaeche()   const {return 0;}
  virtual double Umfang()    const {return 0;}

// b)
  AnsiString toStr() const
  {
  return "Gerade AP="+ap.toStr()+" EP="+ep.toStr();
  }

// c)
  void zeichne() const
  {
  Form1->Memo1->Lines->Add("zeichne "+toStr());
  };
```

```
  virtual void zeichne(TImage* img) const
  {
  img->Canvas->MoveTo(ap.X(),ap.Y());
  img->Canvas->LineTo(ep.X(),ep.Y());
  };

// d)
  void verschiebe_Position_um(const C2DPunkt& deltaPos)
  {
  ap.verschiebe_um(deltaPos);
  ep.verschiebe_um(deltaPos);
  }

// e)
  void write(ofstream& f) const
  {
  f<<class_id<<' '<<ap<<ep<<endl ;
  }

  static const char* class_id;
};

const char* Gerade::class_id="Gerade";


class Kreis:public Figur {
  C2DPunkt M; // Mittelpunkt
  double r;   // Radius
 public:
  Kreis(C2DPunkt M_, double r_):M(M_),r(r_) {}

// a)
  virtual double Flaeche()   const {return r*r*3.14;}
  virtual double Umfang()    const {return 2*3.14*r;}

// b)
  AnsiString toStr() const
  {
  return "Kreis MP="+M.toStr()+" Radius="+FloatToStr(r);
  }

// c)
  void zeichne() const
  {
  Form1->Memo1->Lines->Add("zeichne Kreis: "+toStr());
  };

  virtual void zeichne(TImage* img) const
  {
  img->Canvas->Ellipse(M.X()-r,M.Y()-r,M.X()+r,M.Y()+r);
//  img->Canvas->Arc(M.x-r,M.y-r,M.x+r,M.y+r,M.x+r,M.y+r,M.x+r,M.y+r);
  };

// e)
  void write(ofstream& f) const
  {
  f<<class_id<<' '<<M<<r<<endl;
  }

  static const char* class_id;

// d)
  void verschiebe_Position_um(const C2DPunkt& deltaPos)
  {
  M.verschiebe_um(deltaPos);
  }

// f)
```

```
  Kreis(ifstream& f)
  { // erzeugt einen Kreis mit den Daten aus dem ifstream
  f>>M>>r;
  }

};

const char* Kreis::class_id="Kreis";


class Rechteck:public Figur {
  C2DPunkt lu,ro; // Eckpunkte links unten und rechts oben
 public:
  Rechteck(C2DPunkt lu_, C2DPunkt ro_):lu(lu_),ro(ro_) {}

// a)
  virtual double Flaeche()   const {return (ro.Y()-lu.Y())*(ro.X()-lu.X()) ;}
  virtual double Umfang()    const {return 2*(ro.Y()-lu.Y()+ ro.X()-lu.X() );}

// b)
  AnsiString toStr() const
  {
  return "Rechteck links unten="+lu.toStr()+" rechts oben="+ro.toStr();
  }

// c)
  void zeichne() const
  {
  Form1->Memo1->Lines->Add("zeichne "+toStr());
  };

  virtual void zeichne(TImage* img) const
  { // zeichnet ein gefülltes Rechteck, das den Hintergrund verdeckt
  img->Canvas->Rectangle(lu.X(),lu.Y(),ro.X(),ro.Y());
  // Mit dem nächsten Aufruf wird nur der Rand gezeichnet
  // TRect Rect(lu.X(),lu.Y(),ro.X(),ro.Y());
  // img->Canvas->FrameRect(Rect);
  };

// e)
  void write(ofstream& f) const
  {
  f<<class_id<<' '<<lu<<ro<<endl;
  }

// f)
  Rechteck(ifstream& f):lu(f),ro(f)
  { // erzeugt ein Rechteck mit den Daten aus dem ifstream
  //   f>>lu>>ro; // wäre genauso möglich
  // Bei dieser Schreibweise würde dann nicht das Risiko bestehen,
  // dass die Reihenfolge der Datenelemente geändert wird und diese
  // in verkehrt zugeordnet werden.
  }

// d)
  void verschiebe_Position_um(const C2DPunkt& deltaPos)
  {
  lu.verschiebe_um(deltaPos);
  ro.verschiebe_um(deltaPos);
  }

  static const char* class_id;
};

const char* Rechteck::class_id="Rechteck";


// f)
void Zeichnung::LeseZeichnung(const char* Dateiname)
```

```
{// Diese Funktion muss bei einer Erweiterung um neue Figuren
 // angepasst werden.
c.clear();
ifstream f(Dateiname);
std::string s;
while (f>>s) // Lese den Namen der Figur
  {
    Figur* pf=0;
    if (s==Gerade::class_id)
      pf=new Gerade(f);
    else if (s==Kreis::class_id)
      pf=new Kreis(f);
    else if (s==Rechteck::class_id)
      pf=new Rechteck(f);
    else
      ShowMessage("unexpected data: "+AnsiString(s.c_str()));
    if (pf!=0)
      c.push_back(pf);
  }
f.close(); // überflüssig, aber nicht falsch
}

// g)
Zeichnung einfache_Zeichnung()
{
Zeichnung z;
z.einfuegen(new Rechteck(C2DPunkt(40,40),C2DPunkt(100,100)));
z.einfuegen(new Gerade(C2DPunkt(40,40),C2DPunkt(70,10)));
z.einfuegen(new Gerade(C2DPunkt(70,10),C2DPunkt(100,40)));
z.einfuegen(new Kreis(C2DPunkt(70,40),5));
return z;

/*
// Erzeuge eine Zeichnung:
Zeichnung z;
for (int i=0; i<10; i++)
  if (i%3==0)
    z.einfuegen(new Gerade(C2DPunkt(i,i+1),C2DPunkt(i+2,i+3)));
  else if (i%3==1)
    z.einfuegen(new Kreis(C2DPunkt(i,i+1),i+2));
  else if (i%3==2)
    z.einfuegen(new Rechteck(C2DPunkt(i,i+1),C2DPunkt(i+2,i+3)));
// Schreibe die Zeichnung in eine Datei:
z.write(Dateiname);
// Zeige die Daten an:
z.toMemo(Form1->Memo1);
return z;
*/
}


// g)
Zeichnung z=einfache_Zeichnung();

void testSchreibeDatei(const char* Dateiname)
{
z.write(Dateiname); // schreibe die Zeichnung in eine Datei
z.toMemo(Form1->Memo1); // zeige die Daten an
}

void testLeseDatei(const char* Dateiname)
{
z.LeseZeichnung(Dateiname); // lese die gespeicherten Daten
z.toMemo(Form1->Memo1); // zeige die Daten an
}

// h)
// Um dieses Programm um neue Figuren zu erweitern, müssen lediglich
// die Klassen für die neuen Figuren definiert werden und die Funktion
```

```
// LeseZeichnung erweitert werden.

} // end of namespace N_Aufg_4_10_1


void __fastcall TForm1::Button_4_10_1zeichnenClick(TObject *Sender)
{
N_Aufg_4_10_1::z.zeichne();
N_Aufg_4_10_1::z.zeichne(Form1->Image1);
}

void __fastcall TForm1::Button_4_10_1loeschenClick(TObject *Sender)
{
N_Aufg_4_10_1::z.loesche(Form1->Image1);
}

void __fastcall TForm1::Button_4_10_1verschiebenClick(TObject *Sender)
{
N_Aufg_4_10_1::C2DPunkt delta(10,5);
N_Aufg_4_10_1::z.verschiebe_um(Form1->Image1,delta);
}

const char* Dateiname="Zeichnung.drw";

void __fastcall TForm1::Button_4_10_1schreibenClick(TObject *Sender)
{
N_Aufg_4_10_1::testSchreibeDatei(Dateiname);
}

void __fastcall TForm1::Button_4_10_1lesenClick(TObject *Sender)
{
N_Aufg_4_10_1::testLeseDatei(Dateiname);
}

// --------------- Aufgabe 2 -------------------------------

void __fastcall TForm1::Button_4_10_2Click(TObject *Sender)
{
AnsiString a=
"a) Die Klassen für die speziellen Konten (Sparkonto, Girokonto, "
"Darlehenskonte usw.) werden von einer Basisklasse 'Konto' abgeleitet.";
AnsiString b=
"b) Die Klassen für die speziellen Steuerelemente (Button, Eingabefeld, "
"Anzeigefeld usw.) werden von einer Basisklasse 'Control' abgeleitet.";
AnsiString c=
"c) Die Klassen für die verschiedenen Motoren werden von einer "
"Basisklasse 'Motor' abgeleitet. \r\n"
"Die Klassen für die verschiedenen Temperaturfühler werden von einer "
"Basisklasse 'Temperaturfuehler' abgeleitet. \r\n"
"Die Klasse 'Waschmaschine' enthält dann für den Motor einen Zeiger "
"auf die Basisklasse 'Motor', für den Temperaturfühler einen Zeiger "
"auf die Basisklasse 'Temperaturfühler' usw. \r\n"
"\r\n"
"Der Teil der Steuerung, der für alle Modelle gleich ist, kann in "
"einer rein virtuellen Funktion definiert werden. \r\n"
"Die verschiedenen Waschmaschinenmodelle können dann durch "
"abgeleitete Klassen dargestellt werden, die im Konstruktor "
"einen Zeiger auf den verwendeten Motor, Temperaturfühler usw. "
"setzen. \r\n";
Memo1->Lines->Add(a);
Memo1->Lines->Add(b);
Memo1->Lines->Add(c);
}

// --------------- Aufgabe 6.4.12 --------------------------

void display(AnsiString s)
{
Form1->Memo1->Lines->Add(s);
```

```
}

class Basisklasse{
 public:
  virtual void anzeigen()=0;
  virtual void drucken()=0;
  virtual void aendern()=0;
};

class Bestellung : public Basisklasse{
 public:
  void anzeigen()
  {
    display("Best anzeigen");
  };
  void drucken()
  {
    display("Best drucken");
  };
  void aendern()
  {
    display("Best aendern");
  };
};

class Lagerbestand : public Basisklasse{
 public:
  void anzeigen()
  {
    display("LB anzeigen");
  };
  void drucken()
  {
    display("LB drucken");
  };
  void aendern()
  {
    display("LB aendern");
  };
};

enum Aktion {anzeigen,drucken,aendern};

void dispatch1(Aktion a, Basisklasse* p)
{
if (a==anzeigen)     p->anzeigen();
else if (a==drucken) p->drucken();
else if (a==aendern) p->aendern();
}

void dispatch(Aktion a, Basisklasse* p)
{
typedef void (Basisklasse::* ZEFB)();
ZEFB Aktionen[]={&Basisklasse::anzeigen, &Basisklasse::drucken,
                 &Basisklasse::aendern};

(p->*Aktionen[a])();
}


void __fastcall TForm1::Button_4_14_1Click(TObject *Sender)
{
Basisklasse* p;

Bestellung best;
p=&best;
dispatch(anzeigen,p);
dispatch(drucken,p);
dispatch(aendern,p);
```

```
Lagerbestand lag;
p=&lag;
dispatch(anzeigen,p);
dispatch(drucken,p);
dispatch(aendern,p);
}
```

11.6.15 Aufgabe 6.5

```
// File: C:\Loesungen_CB2006\Kap_6\6.5\RTTIU.cpp

#include <vcl.h>
#pragma hdrstop
#include "RTTIU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

// --------------- Aufgaben 6.4.5 --------------------------

// --------------- Aufgabe 1 -------------------------------

#include <typeinfo>
using namespace std;

  void Aufgabe_1()
  {
  class C {
    virtual // für Aufgabe b) auskommentieren
    void f(){}; // damit C1 polymorph ist
  } c;

  class D1:public C {
  } d1;

  class D2:public C {
  } d2;

                                    // a)        b)
  c=d1;
  bool b1= typeid(c)==typeid(d1);   // false  // false
  c=d2;
  bool b2= typeid(c)==typeid(d2);   // false  // false

  D1* pd1=&d1;
  D2* pd2=&d2;
  C* pc=pd1;
  bool b3= typeid(*pc)==typeid(d1); // true   // false
  bool b4= typeid(pc)==typeid(pd1); // false  // false
  pc=&c;
  bool b5= typeid(*pc)==typeid(d1); // false  // false

  C& rc=c;
  bool b6= typeid(rc)==typeid(d1);  // false  // false
  bool b7= typeid(c)==typeid(d1);   // false  // false
  rc=d1;
```

```
  bool b8= typeid(rc)==typeid(d1);  // false  // false
  C& rc1=d1;
  bool b9= typeid(rc1)==typeid(d1); // true   // false

  int Breakpoint=1;

{
// c)
  class E1:public D2 {
  } e1;

  class E2:public D2 {
  } e2;

// /*******************  für Aufgabe b) auskommentieren
  D1* pd1=&d1;
  D2* pd2=&d2;
  E1* pe1=&e1;
  E2* pe2=&e2;
  C* pc=&c;
  D2* p1= dynamic_cast<D2*>(pc); // p1=0
  pc=pd1;
  D2* p2= dynamic_cast<D2*>(pc); // p2=0
  pc=pe1;
  D2* p3= dynamic_cast<D2*>(pc); // p2!=0
  pc=pe2;                        // Fehler bei private Ableitung
  D2* p4= dynamic_cast<D2*>(pc); // p2!=0

// d) Welche Ergebnisse würde man in Aufgabe c) erhalten,
//    wenn E2 private und nicht public abgeleitete wäre ?

  C& rc=c;
  C& rd1=d1;
  C& rd2=d2;
  C& re1=e1;
  C& re2=e2;
//  D2 x1= dynamic_cast<D2&>(c); // Exception bad_cast
//  D2 x2= dynamic_cast<D2&>(rd1); // Exception bad_cast
  D2 x3= dynamic_cast<D2&>(rd2); // keine Exception
  D2 x4= dynamic_cast<D2&>(re1); // keine Exception
  D2 x5= dynamic_cast<D2&>(re2); // keine Exception

// ********************/ Ende für Aufgabe b) auskommentieren
}

}

void __fastcall TForm1::Aufg_1Click(TObject *Sender)
{
Aufgabe_1();
}

// --------------- Aufgabe 2 -------------------------------

class C {
 public:
  virtual void f()
  {
  Form1->Memo1->Lines->Add(typeid(*this).name());
  };

  C()  { f(); }
  ~C() { f(); }
};

class D:public C {
 public:
  D()  { f(); }
  ~D() { f(); }
```

```
};

void __fastcall TForm1::Aufg_2Click(TObject *Sender)
{
D d; // Lösung: C D D C
}
```

11.7 Lösungen Kapitel 7

11.7.1 Aufgabe 7.1

```
// File: C:\Loesungen_CB2006\Kap_7\ExceptU.cpp

#include <vcl.h>
#pragma hdrstop
#include "ExceptU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

// --------------- Aufgaben 7.10 ---------------------------

// --------------- Aufgabe 1 -------------------------------
/*
  a)    in keiner der Funktionen f1, f2 usw. eine Exception ausgelöst wird:

        Lösung: f1, f2, f4

  b)    in f1 eine Exception ausgelöst wird

        Lösung: f1, f3, f4

  c)    in f2 eine Exception ausgelöst wird

        Lösung: f1, f2, f3, f4

  d)    in f1 und f3 eine Exception ausgelöst wird

        Lösung: f1, f3, f5

  e)    in f1 und f4 eine Exception ausgelöst wird

        Lösung: f1, f3, f4, f5

*/

void f1(char c)
{
Form1->Memo1->Lines->Add("f1");
if (c=='b') throw 1;
if (c=='d') throw 1;
if (c=='e') throw 1;
};
```

```
void f2(char c)
{
Form1->Memo1->Lines->Add("f2");
if (c=='c') throw 2;
};
void f3(char c)
{
Form1->Memo1->Lines->Add("f3");
if (c=='d') throw 3;
};

void f4(char c)
{
Form1->Memo1->Lines->Add("f4");
if (c=='e') throw 4;
};

void f5(char c)
{
Form1->Memo1->Lines->Add("f5");
};

void f_(char c)
{
Form1->Memo1->Lines->Add(c);
try { try {
           f1(c);
           f2(c);
          }
      catch(...)
          {
           f3(c);
          }
      f4(c);
    }
catch (...)
    {
      f5(c);
    }
}


void __fastcall TForm1::Aufg1Click(TObject *Sender)
{
f_('a');
f_('b');
f_('c');
f_('d');
f_('e');
}


// --------------- Aufgabe 2 -------------------------------

#include <stdexcept> // für logic_error
using namespace std;

void Aufg2(int i)
{
if (i==0) throw i+1;
else if (i==1) throw 'A';
else if (i==2) throw 1.0*i;
else if (i==3) throw "Hallo";
else if (i==4) throw exception();
else throw logic_error("Vorbedingung nicht erfüllt");
}

void test_Aufg2(int i)
{
```

```
try { Aufg2(i);
}
catch(int i)
  {
   Form1->Memo1->Lines->Add("int Exception i="+IntToStr(i));
  }
catch(char i)
  {
   Form1->Memo1->Lines->Add("char Exception i="+IntToStr(i));
  }
catch(double i)
  {
   Form1->Memo1->Lines->Add("double Exception i="+FloatToStr(i));
  }
catch(char* i)
  {
   Form1->Memo1->Lines->Add(AnsiString("char* Exception i=")+i);
  }
catch(logic_error& i)
  {
   Form1->Memo1->Lines->Add(AnsiString("Exception i=")+i.what());
  }
catch(exception& i)
  {
   Form1->Memo1->Lines->Add(AnsiString("Exception i=")+i.what());
  }
};

void __fastcall TForm1::Aufg2Click(TObject *Sender)
{
test_Aufg2(0);
test_Aufg2(1);
test_Aufg2(2);
test_Aufg2(3);
test_Aufg2(4);
test_Aufg2(5);
}


// --------------- Aufgabe 3 -------------------------------

#include <stdexcept>
void f(int i)
{
if (i==0) throw exception();
else if (i==1) throw logic_error("logic error");
else if (i==2) throw range_error("range error");
else if (i==3) throw out_of_range("out of range error");
else throw exception();
}

// a)
void g1(int i)
{
try { f(i); }
catch(logic_error& e)
 { Form1->Memo1->Lines->Add("logisch"); }
catch(out_of_range& e)
 { Form1->Memo1->Lines->Add("range"); }
catch(exception& e)
 { Form1->Memo1->Lines->Add("exception"); }
};

void test_Aufg3a()
{
Form1->Memo1->Lines->Add("3 a)");
g1(0); // exception
g1(1); // logisch
g1(2); // exception
```

```
g1(3); // logisch
}

// b)
void g2(int i)
{
try { f(i); }
catch(logic_error& e)
 { Form1->Memo1->Lines->Add(e.what()); }
catch(out_of_range& e)
 { Form1->Memo1->Lines->Add(e.what()); }
catch(exception& e)
 { Form1->Memo1->Lines->Add(e.what()); }
};

void test_Aufg3b()
{
Form1->Memo1->Lines->Add("3 b)");
g2(0); // exception::what(),    "no named exception thrown"
g2(1); // logic_error::what(),  "logic error"
g2(2); // range_error::what(),  "range error"
g2(3); // out_of_range::what(), "out of range error"
// Dieses Ergebnis erhält man einfacher wie in d)
}

// c)
void g3(int i)
{
try { f(i);  }
catch(exception e)
 { Form1->Memo1->Lines->Add(e.what()); }
};

void test_Aufg3c()
{
Form1->Memo1->Lines->Add("3 c)");
g3(0); // exception::what(), "no named exception thrown"
g3(1); // exception::what(), "no named exception thrown"
g3(2); // exception::what(), "no named exception thrown"
g3(3); // exception::what(), "no named exception thrown"
// Da der Datentyp im Exception-Handler keine Referenz
// ist, wird immer die Funktion 'what' von 'exception'
// aufgerufen.
}

// d)
void g4(int i)
{
try { f(i);  }
catch(exception& e)
 { Form1->Memo1->Lines->Add(e.what()); }
catch(...)
 { Form1->Memo1->Lines->Add("irgendeine Exception"); }
};

void test_Aufg3d()
{
Form1->Memo1->Lines->Add("3 d)");
g4(0); // exception::what(),    "no named exception thrown"
g4(1); // logic_error::what(),  "logic error"
g4(2); // range_error::what(),  "range error"
g4(3); // out_of_range::what(), "out of range error"

}

void __fastcall TForm1::Aufg3Click(TObject *Sender)
{
test_Aufg3a();
test_Aufg3b();
```

```
test_Aufg3c();
test_Aufg3d();
}


// --------------- Aufgabe 4 -------------------------------

class myException{
  string str;
 public:
  myException(const string& msg):str(msg) {  };
  const char* msg() const
  {
  return str.c_str();
  }
};

void f_exc()
{
  //
}
void call_f()
{
  try {
        f_exc();
      }
  catch (Exception& e)
    {
        ShowMessage(e.Message);
    }
  catch (exception& e)
    {
        ShowMessage(e.what());
    }
  catch (myException& e)
    {
        ShowMessage(e.msg());
    }
  catch (...)
    {
        ShowMessage("Was soll das?");
    }

}

void __fastcall TForm1::Aufg4Click(TObject *Sender)
{
//
}

// --------------- Aufgabe 5 -------------------------------

// Die Lösungen sind in StringUt.h
#include "..\CppUtils\StringUt.h"

void test_stringTo(string str, int flag)
{
string s;
double i=-17;
try {
  if (flag==0) i=stringToInt(str);
  else i=stringToDouble(str);
}
catch(exception& e)
 { s=e.what(); }
catch(...)
 { s="stringTo error"; }
Form1->Memo1->Lines->Add(s.c_str()+AnsiString(": ")+FloatToStr(i));
}
```

```
void test_stringTo()
{
Form1->Memo1->Lines->Add("test_stringTo");
// stringToInt
test_stringTo("0",0);
test_stringTo("123",0);
test_stringTo("-123",0);
test_stringTo("123.4",0);
test_stringTo(" 123 ",0);
test_stringTo("1 23",0);
test_stringTo("12A3",0);
test_stringTo("123A",0);
test_stringTo("",0);

// stringToDouble
test_stringTo("0",1);
test_stringTo("0.0",1);
test_stringTo("123",1);
test_stringTo("123.4",1);
test_stringTo("-123.4",1);
test_stringTo(" 123 ",1);
test_stringTo("1 23",1);
test_stringTo("12A3",1);
test_stringTo("123A",1);
test_stringTo("",1);

}

void __fastcall TForm1::Aufg5Click(TObject *Sender)
{
test_stringTo();
}

// --------------- Aufgabe 6 -------------------------------

class my_exception:public exception {
  string str;
 public:
  my_exception(const string& msg):str(msg) {  };
  const char* what() const
  {
  return str.c_str();
  }
};

void f()
{
throw my_exception("ich war's");
}

void Aufgabe5()
{
try { f(); }
catch (exception& e)
{ // fängt alle von exception abgeleiteten Exceptions ab
ShowMessage(e.what());
}
}
/*
void __fastcall TForm1::Aufg5Click(TObject *Sender)
{
Aufgabe5();
}
*/
// --------------- Aufgabe 7 -------------------------------

// Siehe auch Betrand Meyer, Object Oriented Software Construction,
//                           Abschnitt 12.2
```

```
class ENegative {};

#include <cmath>
double Sqrt(double d)
{
try {
  if (d<0) throw ENegative();
  return std::sqrt(d);
}
catch(...)
{
  ShowMessage("negativ");
  return 0;
}
}

void __fastcall TForm1::Aufg7Click(TObject *Sender)
{
double d=Sqrt(1.0)+Sqrt(-1.0);
Form1->Memo1->Lines->Add(FloatToStr(d));
}


// --------------- Aufgabe 8 -------------------------------

struct Datum {
  int Tag;
  int Monat;
  int Jahr;
};

Datum StrToDatum(AnsiString s)
{
// Das Datum kann man wie in Aufgabe 4.1.1 konvertieren oder so:
// Ende Datum
TDateTime d;
d = StrToDate(s);
unsigned short year, month, day;
d.DecodeDate(&year, &month, &day);
Datum dat={day,month,year};
return dat;
}

struct Kontobewegung {
  int KontoNr;
  AnsiString NameInhaber;
  Datum Datum;
  char BewArt;
  Currency Betrag;
};

#include <vector>
vector <Kontobewegung> v;

Kontobewegung getData1(TForm1* Form)
{ // TForm1 soll ein Formular sein, das die Edit-Felder EKontoNr->Text usw.
enthält
// Exceptions nicht abfangen, nur weitergeben
Kontobewegung k;
k.KontoNr=StrToInt(Form->EKontoNr->Text);
k.NameInhaber=Form->EName->Text;
k.Datum=StrToDatum(Form->EDatum->Text.c_str());
Form->EBewart->Text.Trim();
if (Form->EBewart->Text.Length() >= 1)
    k.BewArt =Form1->EBewart->Text[1]; // sehr einfach
k.Betrag = StrToCurr(Form->EBetrag->Text);
return k;
}
```

```
void saveData1(TForm1* Form)
{
try {
  Kontobewegung k;
  k=getData1(Form);
  v.push_back(k);
 }
catch(...)
 {  ShowMessage("Fehler"); }
}



Kontobewegung getData2(TForm1* Form)
{ // Alle Exceptions in einem einzigen try abfangen:
  // schlecht, da undefinierte Werte zurückgegeben werden können
Kontobewegung k;
try {
  k.KontoNr=StrToInt(Form->EKontoNr->Text);
  k.NameInhaber=Form->EName->Text;
  k.Datum=StrToDatum(Form->EDatum->Text.c_str());
  Form->EBewart->Text.Trim();
  if (Form->EBewart->Text.Length() >= 1)
      k.BewArt =Form1->EBewart->Text[1]; // sehr einfach
  k.Betrag = StrToCurr(Form->EBetrag->Text);
 }
catch(...)
{ ShowMessage("Fehler"); }
return k;
}

void saveData2(TForm1* Form)
{
try {
  Kontobewegung k;
  k=getData2(Form1);
  v.push_back(k);
}
catch(...)
{
 ShowMessage("Fehler");
}
}

Kontobewegung getData3(TForm1* Form)
{ // Jede Exceptions in einem eigenen try abfangen
Kontobewegung k;

  Form->EBewart->Text.Trim();
  if (Form->EBewart->Text.Length() >= 1)
      k.BewArt =Form1->EBewart->Text[1]; // sehr einfach
  k.Betrag = StrToCurr(Form->EBetrag->Text);

try { k.KontoNr =StrToInt(Form->EKontoNr->Text); }
catch(...)
 {
  Form1->EKontoNr->SetFocus();
  throw; //
 }
k.NameInhaber=Form->EName->Text; // müsste immer gehen
try { k.Datum=StrToDatum(Form->EDatum->Text.c_str()); }
catch(...)
{
  Form1->EDatum->SetFocus();
  throw;
}
return k;
} // kann mit saveData1 verwendet werden
```

```
void saveData3(TForm1* Form)
{
try {
  Kontobewegung k;
  k=getData3(Form);
  v.push_back(k);
}
catch(...)
{
 ShowMessage("Fehler");
}
}


void __fastcall TForm1::Aufg8Click(TObject *Sender)
{
saveData1(Form1);
}


// --------------- Aufgabe 9 -------------------------------

void h1()
{ // kann zu einem Speicherleck führen
vector<int*> v;
int* p=new int(17);
v.push_back(p);
f();
for (vector<int*>::iterator i=v.begin(); i!=v.end(); i++)
  delete *i;
}

class C {
  vector<int*> v;
 public:
  void push_back(int* p)
  {
  v.push_back(p);
  }
 ~C()
 {
 for (vector<int*>::iterator i=v.begin(); i!=v.end(); i++)
  delete *i;
 }

};

void h2()
{
C v;
v.push_back(new int(17));
f();
}

void __fastcall TForm1::Aufg9Click(TObject *Sender)
{
h1();
h2();
}
```

11.8 Lösungen Kapitel 8

11.8.1 Aufgabe 8.2

```
// File: C:\Loesungen_CB2006\Kap_8\8.2\PropU.cpp

#include <vcl.h>
#pragma hdrstop
#include "PropU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

// --------------- Aufgabe 8.2.1 ---------------------------

class C {
  int fx;
  void setx(int x_){fx=x_*x_;};
  int getx(){return fx;};
 public:
  __property int x = {read=getx, write=setx};
};

void __fastcall TForm1::Button_8_2_1Click(TObject *Sender)
{
C p;
p.x=17;
int y=p.x;
Memo1->Lines->Add(y); // 17*17=289
}
```

11.8.2 Aufgabe 8.3

```
// File: C:\Loesungen_CB2006\Kap_8\8.3\ResizeUnit.h

#ifndef ResizeUnitH
#define ResizeUnitH
//---------------------------------------------------------------------

class TResize{
  int fsw,fsh,
      faw,fah;
  bool initialized,
       PositionAndSize; // true: adjust position and size
                        // false: adjust only position
  int n_Controls; // müßte meist reichen

  TForm* Form;

  struct TWinCoord {
    int Top,
        Left,
        Width,
        Height;
```

```
  };

  TWinCoord* sc;
  int xt(int x)
  { // transformiert x-Koordinaten
  return x*faw/fsw;
  };

  int yt(int y)
  { // transformiert y-Koordinaten
  return y*fah/fsh;
  };

  // Da diese Klasse Zeiger enthält, führt der vom Compiler erzeugte
  // Copy-Konstruktor zu flachen Kopien. Da der aber nie benötigt wird,
  // wird sein Aufruf durch eine private Deklaration unterbunden.
  TResize(TResize&); // Copy-Konstruktor private
  TResize& operator=(TResize&);
  void Init(TForm* Form_);

 public:
  void Resize(TForm* Form_);
  TResize(bool PositionAndSize_=false):initialized(false),
          PositionAndSize(PositionAndSize_) { };
  virtual ~TResize() {delete[] sc;}
};


#endif


// File: C:\Loesungen_CB2006\Kap_8\8.3\ResizeUnit.cpp

// Diese Unit mit Datei|Neu|Unit erzeugen
#include <vcl.h>
#pragma hdrstop

#include "ResizeUnit.h"

#pragma package(smart_init)


// --------------- Aufgabe 8.3 ----------------------------

void TResize::Init(TForm* F)
{
Form=F;
sc=new TWinCoord[F->ComponentCount];
fsw = Form->Width; // Breite des Formulars beim Start
fsh = Form->Height;
for (int i=0; i<Form->ComponentCount;i++ )
  { // Größe der Komponenten beim Start
    sc[i].Top    = ((TControl*)Form->Components[i])->Top;
    sc[i].Left   = ((TControl*)Form->Components[i])->Left;
    // Die Konversion ist auch mit dynamic_cast möglich:
    sc[i].Height = dynamic_cast<TControl*>(Form->Components[i])->Height;
    sc[i].Width  = dynamic_cast<TControl*>(Form->Components[i])->Width;
  }
initialized = true; // Wird bei Create automatisch auf false gesetzt
};

void TResize::Resize(TForm* F)
{
if (!initialized) Init(F);
faw = Form->Width; // Breite des aktuellen Formulars
fah = Form->Height;
for (int i=0; i<Form->ComponentCount; i++)
  if (PositionAndSize) // Position und Größe anpassen
    ((TWinControl*)Form->Components[i])->SetBounds(xt(sc[i].Left),
```

```
        yt(sc[i].Top),xt(sc[i].Width), yt(sc[i].Height));
  else  // nur die Position anpassen
    ((TWinControl*)Form->Components[i])->SetBounds(xt(sc[i].Left),
        yt(sc[i].Top), sc[i].Width, sc[i].Height);
};
```

```
// File: C:\Loesungen_CB2006\Kap_8\8.3\ResizU.cpp

#include <vcl.h>
#pragma hdrstop
#include "ResizU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}


// --------------- Aufgabe 8.3 ----------------------------


#include "ResizeUnit.h"

// Noch "ResizeUnit.cpp" mit Projekt|Dem Projekt hinzufügen

TResize r;          // Passt nur die Größe und nicht die Position an
//TResize r(true); // Passt die Größe und die Position an

void __fastcall TForm1::FormResize(TObject *Sender)
{
r.Resize(Form1);
}
```

11.8.3 Aufgabe 8.4

```
// File: C:\Loesungen_CB2006\Kap_8\8.4\CompU.h

/*
 Das Makro PACKAGE ist in "include\vcl\sysmac.h" folgendermaßen definiert:
 #define PACKAGE   __declspec(package)  // Implemented in a package
 Klassen, die mit ihm definiert wurden, können in einem Package verwendet
 werden. Dieses Makro ist für die Aufgaben 9.8 notwendig.
*/


// --------------- Aufgabe 4: TValueEdit -------------------

class PACKAGE TValueEdit: public TEdit {
  double FValue;
  int FDecimalPlaces; // engl.; decimal places
  AnsiString FormatString;
  void __fastcall SetDecimalPlaces(int x);
  void __fastcall SetValue(double x);
 __published:
   __property double Value = {read=FValue , write=SetValue};
   __property int DecimalPlaces = {read=FDecimalPlaces , write=SetDecimalPlaces};
 public:
```

```
  __fastcall TValueEdit(TForm *Form,int l,int t,int w,int h);
  __fastcall TValueEdit(TComponent* AOwner);
};

// --------------- Aufgabe 5: TColBorderLabel --------------

class PACKAGE TColBorderLabel : public TCustomLabel {
  private:
    TColor FBColor;
    int FBWidth;
    bool FShow;
    TTimer* FTimer;
    int FInterval;
    void __fastcall SetColor(TColor Value);
    void __fastcall SetWidth(int Value);
    void __fastcall SetInterval(int Value);
    void __fastcall Show(bool Value);
    void __fastcall TimerEvent(TObject* Sender);
  protected:
    void __fastcall Paint(); // override;
  public:
   __fastcall TColBorderLabel(TComponent* AOwner,TWinControl* Parent_, int Left_,
int Top_, int Width_, int Height_);
   __fastcall TColBorderLabel(TComponent* AOwner);
  __published:
     // eigene properties

    __property TColor BorderColor = {read=FBColor, write=SetColor};
    __property int BorderWidth={read=FBWidth, write=SetWidth};
    __property bool ShowBorder={read=FShow, write=Show};
    __property int BlinkInterval={read=FInterval, write=SetInterval};

 // TControl veröffentlicht nur wenige Eigenschaften; durch
 // die folgenden Deklarationen werden die unsichtbaren
 // Eigenschaften der Vorgänger sichtbar

    __property Align;         __property Alignment;
    __property AutoSize;      __property Caption;
    __property Color;         __property Cursor;
    __property DragCursor;    __property DragMode;
    __property Enabled;       __property FocusControl;
    __property Font;          __property ParentColor;
    __property ParentFont;    __property ParentShowHint;
    __property ShowAccelChar; __property ShowHint;
    __property Transparent;   __property Visible;
    __property WordWrap;

    __property OnClick;       __property OnDblClick;
    __property OnDragDrop;    __property OnDragOver;
    __property OnEndDrag;     __property OnMouseDown;
    __property OnMouseMove;   __property OnMouseUp;
};



// File: C:\Loesungen_CB2006\Kap_8\8.4\CompU.cpp

// --------------- Aufgabe 4: TValueEdit -------------------
void __fastcall TValueEdit::SetDecimalPlaces(int x)
{
const int m=5; // this can be easily extended
AnsiString fs[m]={"0","0.0","0.00","0.000","0.0000"};
FDecimalPlaces=x;
if ((0<=x) && (x<5)) FormatString=fs[x];
else FormatString="0.00";
SetValue(FValue);
}

void __fastcall TValueEdit::SetValue(double x)
```

```
{
FValue=x;
Text=FormatFloat(FormatString,x);
}

__fastcall TValueEdit::TValueEdit(TComponent* AOwner):TEdit(AOwner)
{
DecimalPlaces=2;
FormatString="0.00";
};

__fastcall TValueEdit::TValueEdit(TForm *Form,
                   int l,int t,int w,int h) :TEdit(Form)
{
Parent = Form;
SetBounds(l,t,w,h);
DecimalPlaces=2;
FormatString="0.00";
};


// --------------- Aufgabe 5: TColBorderLabel --------------
__fastcall                          TColBorderLabel::TColBorderLabel(TComponent*
AOwner):TCustomLabel(AOwner)
{
FTimer = new TTimer(this);
FTimer->OnTimer = TimerEvent;
FTimer->Interval = 300;
FTimer->Enabled = true;
}

__fastcall    TColBorderLabel::TColBorderLabel(TComponent*    AOwner,TWinControl*
Parent_,
            int Left_, int Top_, int Width_, int Height_):TCustomLabel(AOwner)
{
Parent=Parent_;
Canvas->Pen->Width = FBWidth;
Canvas->Pen->Color = FBColor;
FTimer = new TTimer(this);
FTimer->OnTimer = TimerEvent;
FTimer->Interval = 300;
FTimer->Enabled = true;
SetBounds(Left_,Top_,Width_,Height_);
}


void __fastcall TColBorderLabel::Paint ()
{
TCustomLabel::Paint();
if (FShow)
 {
   Canvas->MoveTo(0,0);
   Canvas->LineTo(Width-1,0);
   Canvas->LineTo(Width-1,Height-1);
   Canvas->LineTo(0,Height-1);
   Canvas->LineTo(0,0);
 }
}

void __fastcall TColBorderLabel::SetColor(TColor value)
{
  FBColor = value;
  Canvas->Pen->Color = value;
  Paint();
}

void __fastcall TColBorderLabel::SetWidth(int value)
{
FBWidth = value;
```

```
Canvas->Pen->Width = value;
Paint();
}

void __fastcall TColBorderLabel::SetInterval(int value)
{
FInterval = value;
if (FInterval > 0)
  {
    FTimer->Interval = value;
    FTimer->Enabled = true;
  }
else
  FTimer->Enabled = false;
}

void __fastcall TColBorderLabel::TimerEvent(TObject* Sender)
{
ShowBorder = !ShowBorder;
}

void __fastcall TColBorderLabel::Show(bool value)
{
FShow = value;
Paint();
}



// File: C:\Loesungen_CB2006\Kap_8\8.4\CompEvU.cpp

#include <vcl.h>
#pragma hdrstop

#include "CompEvU.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}



// --------------- Aufgabe 1 -------------------------------

// ClickAction ist im Klassenheader deklariert

void __fastcall TForm1::ClickAction1(TObject *Sender)
{
Button3->Caption="Action 1";
}

void __fastcall TForm1::ClickAction2(TObject *Sender)
{
Button3->Caption="Action 2";
}

void __fastcall TForm1::ClickAction3(TObject *Sender)
{
Button3->Caption="Action 3";
}


void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
```

```
Button3->OnClick=ClickAction1;
}

void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
Button3->OnClick=ClickAction2;
}

void __fastcall TForm1::RadioButton3Click(TObject *Sender)
{
Button3->OnClick=ClickAction3;
}


// --------------- Aufgabe 2 -------------------------------

TEdit* MakeEdit(TForm* F, int l,int t,int w,int h)
{
TEdit* E = new TEdit(F);
E->Parent = F;
E->SetBounds(l, t, w, h);
return E;
};

TLabel* MakeLabel(TForm* F, int l,int t,int w,int h, AnsiString Caption)
{
TLabel* L = new TLabel(F);
L->Parent = F;
L->SetBounds(l, t, w, h);
L->Caption=Caption;
return L;
};

TButton*  MakeButton(TForm*  F,AnsiString  Caption,  int  l,int  t,int  w,int
h,TNotifyEvent E)
{
TButton* B = new TButton(F);
B->Parent = F;
B->SetBounds(l, t, w, h);
B->Caption=Caption;
B->OnClick=E;
return B;
};

class MyForm : public TForm
{
public:
 __fastcall MyForm(TForm* AOwner);

 void __fastcall EndeClick(TObject *Sender);
 void __fastcall LoeschenClick(TObject *Sender);
 void __fastcall SpeichernClick(TObject *Sender);
 TLabel* LVorname;
 TEdit*  EVorname;
 TLabel* LNachname;
 TEdit*  ENachname;
 TLabel* LBussgeld;
 TEdit*  EBussgeld;
 TLabel* LOrdnungswidrigkeit;
 TEdit*  EOrdnungswidrigkeit;

 TButton* BSpeichern;
 TButton* BLoeschen;
 TButton* BEnde;
};

__fastcall MyForm::MyForm(TForm* AOwner):TForm(AOwner,0)
{
Caption="Das städtische Haushaltssanierungsprogramm";
```

```
int LabelLeft=10, EditLeft=150, Top=10, TopInc=40, w=100,h=20;
LVorname=MakeLabel(this,LabelLeft,Top,w,h,"Vorname");

EVorname=MakeEdit (this,EditLeft,Top,w,h);

LNachname=MakeLabel(this,LabelLeft,Top+TopInc,w,h,"Nachname");
ENachname=MakeEdit (this,EditLeft,Top+TopInc,w,h);

LBussgeld=MakeLabel(this,LabelLeft,Top+2*TopInc,w,h,"Bussgeld");
EBussgeld=MakeEdit (this,EditLeft,Top+2*TopInc,w,h);

LOrdnungswidrigkeit=MakeLabel(this,LabelLeft,Top+3*TopInc,w,h,"Ordnungswidrigkeit
");
EOrdnungswidrigkeit=MakeEdit (this,EditLeft,Top+3*TopInc,w,h);

BSpeichern=MakeButton(this,"Daten speichern",10,180,100,h,SpeichernClick);
BLoeschen=MakeButton(this,"Eingabe löschen",120,180,100,h,LoeschenClick);
BEnde=MakeButton(this,"Programm beenden",230,180,100,h,EndeClick);
Show();
SetBounds(0,0,400,250);
};

void __fastcall MyForm::SpeichernClick(TObject *Sender)
{
AnsiString v=EVorname->Text;
AnsiString n=ENachname->Text;
AnsiString b=EBussgeld->Text;
AnsiString o=EOrdnungswidrigkeit->Text;
}

void __fastcall MyForm::LoeschenClick(TObject *Sender)
{
EVorname->Clear();
ENachname->Clear();
EBussgeld->Clear();
EOrdnungswidrigkeit->Clear();
}

void __fastcall MyForm::EndeClick(TObject *Sender)
{
Close();
}

MyForm* MyForm1;

void __fastcall TForm1::CreateFormClick(TObject *Sender)
{
MyForm1=new MyForm(Form1);
}


// --------------- Aufgabe 3 -------------------------------

TMenuItem* NewMenuBarItem(TComponent* Owner, TMainMenu* Menu, AnsiString Caption)
{
TMenuItem *NewMenuItem = new TMenuItem(Owner);
NewMenuItem->Caption = Caption;
Menu->Items->Add(NewMenuItem);
return NewMenuItem;
}

TMenuItem* NewMenuItem(TComponent* Owner, TMenuItem* Menu, AnsiString Caption)
{ // Der Typ des Menu Parameters ist der erste Unterschied zu NewMenuBarItem
TMenuItem *NewMenuItem = new TMenuItem(Owner);
NewMenuItem->Caption = Caption;
Menu->Add(NewMenuItem); // Der zweite Unterschied zu NewMenuBarItem
return NewMenuItem;
}
```

```
// Diese Elementfunktion auch noch in der Klasse deklarieren.
// void __fastcall TForm1::CreatedMenuItem1Click(TObject *Sender);

void __fastcall TForm1::CreatedMenuItem1Click(TObject *Sender)
{
Form1->Memo1->Lines->Add("CreatedMenuItem1Click");
}

void testCreateMenuItems()
{
// Ein Hauptmenü MainMenu1 muss für den nächsten Aufruf existieren:
TMenuItem* mb1=NewMenuBarItem(Form1, Form1->MainMenu1,"File");
TMenuItem* mb2=NewMenuBarItem(Form1, Form1->MainMenu1,"Edit");
// mb1->OnClick=Form1->CreatedMenuItem1Click;
mb2->OnClick=Form1->CreatedMenuItem1Click;
// Normalerweise definiert man für Menüoptionen der Menüleiste keine Ereignisse

TMenuItem* m2=NewMenuItem(Form1, mb1,"Menüeintrag 1");
m2->OnClick=Form1->CreatedMenuItem1Click;
// Verschachtelte Untermenüs erhält man, indem man einen neues
// Menüeintrag in einen anderen Menüeintrag einhängt:
TMenuItem* m3=NewMenuItem(Form1, m2,"verschachtelter Menüeintrag");
m3->OnClick=Form1->CreatedMenuItem1Click;

// TMenuItem* m2=CreateNewMainMenuItem(Form1, Form1->MainMenu1,"Edit");
}

void __fastcall TForm1::AddMenuItemsClick(TObject *Sender)
{
testCreateMenuItems();
}


#include "compu.h"
#include "compu.cpp"

// --------------- Aufgabe 4 -------------------------------

TValueEdit* v1;
TValueEdit* v2;

void __fastcall TForm1::CreateTValEditClick(TObject *Sender)
{
int l=Form1->GroupBox4->Left;
int t=Form1->GroupBox4->Top;
v1=new TValueEdit(Form1,l+10,t+20,100,20);
v2=new TValueEdit(Form1,l+10,t+50,100,20);
}

void __fastcall TForm1::TestTValEditClick(TObject *Sender)
{
static double x=3.14, y=x;
v1->Value=x;
v2->Value=y;
x=x*10;
y=y/10;
}

void __fastcall TForm1::UpDown1Click(TObject *Sender, TUDBtnType Button)
{
v1->DecimalPlaces=UpDown1->Position;
v2->DecimalPlaces=UpDown1->Position;
}


// --------------- Aufgabe 5 -------------------------------

TColBorderLabel* l1;
```

```
TColBorderLabel* l2;

void __fastcall TForm1::CreateColBorderLabelClick(TObject *Sender)
{
l1=new TColBorderLabel(GroupBox5,GroupBox5,10,20,100,20);
l2=new TColBorderLabel(GroupBox5,GroupBox5,10,50,100,20);
l1->Caption="Ooohh, how nice";
l1->BlinkInterval=100;
l1->BorderColor=clRed;
l1->Color=clYellow;

l2->Caption="just like christmas";
l2->BlinkInterval=500;
l2->BorderColor=clBlue;
l2->Color=clWhite;
}


// --------------- Aufgabe 6 -------------------------------

// Alternative 1:
// Universell, aber mehr Schreibaufwand als die andern beiden.

class TMyEdit1:public TEdit {
 public:
    __fastcall TMyEdit1(TComponent* AOwner,TWinControl* Parent_):TEdit(AOwner){
    Parent=Parent_;
    };
};

TMyEdit1* e1=new TMyEdit1(Form1,Form1);


class TMyEdit2:public TEdit {
 public:
    __fastcall TMyEdit2(TComponent* AOwner):TEdit(AOwner)
    {
    // Parent=AOwner; // Cannot convert 'TComponent *' to 'TWinControl *'
    // Da dieser Konstruktor automatisch beim Start des Programms
    // für Komponenten aufgerufen wird, die mit den Hilfsmitteln
    // der visuellen Programmierung auf ein Formular gesetzt wurden,
    // sollte dieser Konstruktor für explizit erzeugte Komponenten
    // nicht aufgerufen werden.
        Parent=(TWinControl*)AOwner; // Das ist möglich, aber nicht
    // empfehlenswert, da das Probleme gibt, falls das Argument kein
    // TWinControl ist
    };
};

TMyEdit2* e2=new TMyEdit2(Form1);


class TMyEdit3:public TEdit {
 public:
    __fastcall TMyEdit3(TWinControl* AOwner):TEdit(AOwner)
    {
    Parent=AOwner; // Das ist möglich, beschränkt aber die Argumente
    // auf ein TWinControl
    };
};

TMyEdit3* e3=new TMyEdit3(Form1);


class TMyEdit4:public TEdit {
 public:
    __fastcall TMyEdit4(TComponent* AOwner,TWinControl* Parent_):TEdit(AOwner)
   {
    Parent=Parent_;
```

```
   };
   __fastcall TMyEdit4(TWinControl* AOwner):TEdit(AOwner)
   {
    Parent=AOwner;
   };
   __fastcall TMyEdit4(TComponent* AOwner):TEdit(AOwner)
   {
   };
    // Diese drei Konstruktoren sind zwar mit dem meisten Schreibaufwand
    // verbunden. Sie ermöglichen aber den Aufruf mit einem oder zwei
    // Argumenten ohne Probleme und ohne Einschränkungen, sowie die
    // Installation der Komponente in die Komponentenpalette
};

TGroupBox* GroupBox1;
TMyEdit4* e4a=new TMyEdit4(Form1);
TMyEdit4* e4b=new TMyEdit4(Form1,GroupBox1);

// Komponentennamen sollten immer mit T beginnen
// Borland verwendet üblicherweise den Anfangsbuchstaben f für
// Datenelemente, die zu properties gehören
```

11.8.4 Aufgabe 8.5

```
// File: C:\Loesungen_CB2006\Kap_8\8.5\BCBBook.h

#ifndef BCBBookH
#define BCBBookH
#include <SysUtils.hpp>
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
//---------------------------------------------------------------------------
class PACKAGE TBCBBook : public TEdit
{
private:
protected:
public:
        __fastcall TBCBBook(TComponent* Owner);
__published:
};


#include "..\9.4\CompU.h"


// Definieren Sie das Makro IncludeSolutions_97, falls die
// Lösungen von Abschnitt 9.7 in das package aufgenommen
// werden sollen.

#define IncludeSolutions_97 1
//#define IncludeSolutions_97 0

#if IncludeSolutions_97
#include "..\9.7\CompU.h"
#endif

//---------------------------------------------------------------------------
#endif


// File: C:\Loesungen_CB2006\Kap_8\8.5\BCBBook.cpp
```

```
/*
Da ich die Komponenten aus den include-Dateien
  #include "..\9.4\CompU.h"
  #include "..\9.4\CompU.cpp"

und

  #include "..\9.7\CompU.h"
  #include "..\9.7\CompU.cpp"

erzeugen wollte und nicht aus den Komponenten, die vom
component wizard erzeugt werden, und da der C++Builder
als Dateinamen den der Komponent wählt, habe ich als
Namen der Komponente den Namen gewählt, den ich als
Dateinamen haben wollte "BCBBook". Diese Komponente
wird hier nicht verwendet.

Siehe dazu die Screenshots in Configuration.doc
*/

//---------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "BCBBook.h"
#pragma package(smart_init)
//---------------------------------------------------------------------
// ValidCtrCheck is used to assure that the components created do not have
// any pure virtual functions.
//

// Das Makro IncludeSolutions_97 wird in bcbbook.h definiert,
// wenn die Lösungen von Abschnitt 9.7 in das Package aufgenommen
// werden sollen.

static inline void ValidCtrCheck(TBCBBook *)
{
        new TBCBBook(NULL);
        new TValueEdit(0,0,0,0,0);
        new TColBorderLabel(0,0,0,0,0,0);
#if IncludeSolutions_97
        new TTabEdit(0,0,0,0,0);
        new TFocusColorEdit(0,0,0,0,0,0);
        new TResizableMemo(0,0,0,0,0,0);
        new TRubberShape(0,0,0,0,0,0);
#endif
}
//---------------------------------------------------------------------
__fastcall TBCBBook::TBCBBook(TComponent* Owner)
        : TEdit(Owner)
{
}


//---------------------------------------------------------------------
#include "..\9.4\CompU.cpp"

#if IncludeSolutions_97
#include "..\9.7\CompU.cpp"
#endif
namespace Bcbbook
{

        void __fastcall PACKAGE Register()
        {
// Auskommentiert, da das nicht benötigt wird (siehe oben)
```

```
/*
                TComponentClass classes[1] = {__classid(TBCBBook)};
                RegisterComponents("BCBBook", classes, 0);
        }
*/
#if IncludeSolutions_97
 const int n=6;
   TComponentClass classes[n] = {
       __classid(TValueEdit),
       __classid(TColBorderLabel),
       __classid(TTabEdit),
       __classid(TFocusColorEdit),
       __classid(TResizableMemo),
       __classid(TRubberShape)
   };
   RegisterComponents("BCBBook", classes, n-1);
#else
 const int n=2;
   TComponentClass classes[n] = {
       __classid(TValueEdit),
       __classid(TColBorderLabel)
   };
   RegisterComponents("BCBBook", classes, n-1);
#endif
        }
}

// C++Builder erzeugt das package im Verzeichnis
// \CBuilder\Projects\Bpl\BCBBookPkg.bpl
// Kopien befinden sich im Verzeichnis copied_from_projects_bpl
```

```
// File: C:\Loesungen_CB2006\Kap_8\8.5\UseCompU.cpp

#include <vcl.h>
#pragma hdrstop

#include "UseCompU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "BCBBook"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm1::UpDown1Click(TObject *Sender, TUDBtnType Button)
{
ValueEdit1->DecimalPlaces=UpDown1->Position;
}
//---------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
static double x=3.14, y=x;
ValueEdit1->Value=x;
ValueEdit2->Value=y;
Edit1->Text=x;
Edit2->Text=y;
x=x*10;
y=y/10;
}
//---------------------------------------------------------------------
```

11.8.5 Aufgabe 8.6

```
// File: C:\Loesungen_CB2006\Kap_8\8.6\ClassRefU.cpp

#include <vcl.h>
#pragma hdrstop
#include "ClassRefU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}


// --------------- Aufgaben 8.6 ----------------------------

void ShowBaseNames(TMetaClass* M)
{
for (TMetaClass* C=M;C!=0; C=C->ClassParent())
  Form1->Memo1->Lines->Add(C->ClassName());
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
ShowBaseNames(__classid(TEdit));
}
```

11.8.6 Aufgabe 8.7

```
// File: C:\Loesungen_CB2006\Kap_8\8.7\123.cpp

// --------------- Aufgabe 1 -------------------------------
void __fastcall TForm1::ButtonCopyClick(TObject *Sender)
{
SendMessage(Memo1->Handle, WM_COPY, 0, 0);
// So ist auch CopyToClipboard in TCustomEdit definiert.
// Dasselbe Ergebnis erzielt man mit
//   Memo1->Perform(WM_COPY,0,0);
}

void __fastcall TForm1::ButtonCutClick(TObject *Sender)
{
SendMessage(Memo1->Handle, WM_CUT, 0, 0);
} // Wie CutToClipboard in TCustomEdit



void __fastcall TForm1::ButtonPasteClick(TObject *Sender)
{
SendMessage(Memo1->Handle, WM_PASTE, 0, 0);
} // Wie PasteFromClipboard in TCustomEdit



// --------------- Aufgabe 2 -------------------------------

void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
{
```

```
if (Key == 13)
  Form1->Perform(WM_NEXTDLGCTL,0,0);
}

void __fastcall TForm1::FormKeyPress(TObject *Sender, char &Key)
{ // falls KeyPreview auf true gesetzt ist
if (Key == 13)
  Form1->Perform(WM_NEXTDLGCTL,0,0);
// dasselbe Ergebnis erzielt man mit
//  SendMessage(Form1->Handle,WM_NEXTDLGCTL,0,0); }
}


// --------------- Aufgabe 3 -------------------------------

/*
 Definiert man in MsgU.h das zweite Makro

   #define DISABLE_WMCLOSE 0
   #define DISABLE_WMCLOSE 1

 kann man das Fenster mit Alt-F4 nicht mehr schliessen.

 In der Klasse TForm1 in MsgU.h nicht vergessen:
   void __fastcall WndProc(TMessage &Message)
*/

#if DISABLE_WMCLOSE
void __fastcall TForm1::WndProc(TMessage &Message)
{
if (WM_CLOSE == Message.Msg);
else TForm::WndProc(Message);
}
#endif

// Das Programm kann mit einem Aufruf von TForm::Close(),
// Run|Program Reset in der IDE und dem task manager beendet werden.



// File: C:\Loesungen_CB2006\Kap_8\8.7\CompU.h

/*
 Das Makro PACKAGE ist in "include\vcl\sysmac.h" folgendermaßen definiert:
 #define PACKAGE   __declspec(package)  // Implemented in a package
 Klassen, die mit ihm definiert wurden, können in einem Package verwendet
 werden. Dieses Makro ist für die Aufgaben 9.8 notwendig.
*/


// --------------- Aufgabe 4: TTabEdit ---------------------
class PACKAGE TTabEdit : public TEdit {
 private:
  bool FEnterNextDlgCtl;
  void __fastcall WMNextDlgCtl(TWMChar& Msg);

  BEGIN_MESSAGE_MAP
    VCL_MESSAGE_HANDLER(WM_CHAR, TWMChar, WMNextDlgCtl);
  END_MESSAGE_MAP(TEdit);
 __published:
  __property bool EnterNextDlgCtl={read=FEnterNextDlgCtl,
                                   write=FEnterNextDlgCtl};
 public:
  __fastcall TTabEdit(TComponent* Owner);
  __fastcall  TTabEdit(TComponent*  AOwner,TWinControl*  Parent_,int  Left_,  int
Top_, int Width_, int Height_);
  __fastcall TTabEdit(TComponent* AOwner, int Left_, int Top_, int Width_, int
Height_);
```

```
};


// --------------- Aufgabe 5: TFocusColorEdit --------------
//#define TFocusColorEdit_Variant 1
//#define TFocusColorEdit_Variant 2

class PACKAGE TFocusColorEdit : public TEdit{
 private:
  TColor prevColor, FFocusColor;

#if TFocusColorEdit_Variant
  void __fastcall WMSetFocus(TWMSetFocus& Message);
  void __fastcall WMKillFocus(TWMKillFocus& Msg);
#endif

#if TFocusColorEdit_Variant==1
  BEGIN_MESSAGE_MAP
    VCL_MESSAGE_HANDLER(WM_SETFOCUS,  TWMSetFocus, WMSetFocus);
    VCL_MESSAGE_HANDLER(WM_KILLFOCUS, TWMKillFocus, WMKillFocus);
  END_MESSAGE_MAP(TEdit);
#endif


#if TFocusColorEdit_Variant==2
    virtual void __fastcall WndProc(Messages::TMessage &Message)
    {
    if (Message.Msg==WM_SETFOCUS)  WMSetFocus(*(((TWMSetFocus*)(&Message))));
    if (Message.Msg==WM_KILLFOCUS) WMKillFocus(*(((TWMKillFocus*)(&Message))));
    TEdit::WndProc(Message);
    };
#endif

   void __fastcall SetFocusColor(TColor C);
  __published:
   __property TColor FocusColor={read=FFocusColor,
                          write=SetFocusColor};
  public:

/*
  Da für die Ereignisse "den Fokus erhalten", bzw. "den Fokus verlieren"
  bereits Funktionszeiger vorhanden sind, ist es am einfachsten, diesen
  Zeigern entsprechende Funktionen zuzuwiesen. Man kann aber ebenso WndProc
  überschreiben oder Dispatch. Diese Lösungsvarianten erhält man mit den
  Makros

   #define TFocusColorEdit_Variant 1
   #define TFocusColorEdit_Variant 2
*/

   void __fastcall MyOnEnter(System::TObject* Sender)
   {
   prevColor = Color;
   FFocusColor = clYellow;
   Color = FFocusColor;
   };

   void __fastcall MyOnExit(System::TObject* Sender)
   {
   Color = prevColor;
   };

   //__fastcall  TFocusColorEdit(TComponent*  AOwner,  int  Left_,  int  Top_,  int
Width_, int Height_);
   __fastcall   TFocusColorEdit::TFocusColorEdit(TComponent*   AOwner,TWinControl*
Parent_,
                 int Left_, int Top_, int Width_, int Height_);
   __fastcall TFocusColorEdit::TFocusColorEdit(TComponent* AOwner);
};
```

```
// --------------- Aufgabe 6: TResizableMemo ---------------

class PACKAGE TResizableMemo :public TMemo {
 private:
  bool LB_Down;
  void __fastcall WMLButtonDown(TWMLButtonDown& Msg);
  void __fastcall WMLButtonUp(TWMLButtonUp& Msg);
  void __fastcall WMMouseMove(TWMLButtonUp& Msg);
  BEGIN_MESSAGE_MAP
    VCL_MESSAGE_HANDLER(WM_LBUTTONDOWN, TWMLButtonDown, WMLButtonDown);
    VCL_MESSAGE_HANDLER(WM_LBUTTONUP,   TWMLButtonUp,   WMLButtonUp);
    VCL_MESSAGE_HANDLER(WM_MOUSEMOVE,   TWMMouseMove,   WMMouseMove);
  END_MESSAGE_MAP(TMemo);
 public:
   __fastcall TResizableMemo(TComponent* AOwner,TWinControl* Parent_, int Left_,
int Top_, int Width_, int Height_);
   __fastcall TResizableMemo(TComponent* AOwner);
};

// --------------- Aufgabe 7: TRubberShape -----------------

class PACKAGE TRubberShape :public TImage {
  enum {rsLine,Low=rsLine,rsRectangle,rsCircle,High=rsCircle} rsShape ;
 private:
  bool LB_Down;
  int x_Start,y_Start,x,y,r;
  void __fastcall WMLButtonDown(TWMLButtonDown& Msg);
  void __fastcall WMLButtonUp(TWMLButtonUp& Msg);
  void __fastcall WMMouseMove(TWMLButtonUp& Msg);
  void __fastcall WMRButtonDblClick(TWMLButtonDown& Msg);
  BEGIN_MESSAGE_MAP
    VCL_MESSAGE_HANDLER(WM_LBUTTONDOWN,  TWMLButtonDown,   WMLButtonDown);
    VCL_MESSAGE_HANDLER(WM_LBUTTONUP,    TWMLButtonUp,     WMLButtonUp);
    VCL_MESSAGE_HANDLER(WM_MOUSEMOVE,    TWMMouseMove,     WMMouseMove);
    VCL_MESSAGE_HANDLER(WM_RBUTTONDBLCLK,TWMRButtonDblClk, WMRButtonDblClick);
  END_MESSAGE_MAP(TImage);
  void __fastcall RectLine(int x1,int y1,int x2,int y2);
  void __fastcall CircLine(int x,int y,int r);
 public:
  __fastcall TRubberShape(TComponent* AOwner, TWinControl* Parent_,
      int Left_, int Top_, int Width_, int Height_);
  __fastcall TRubberShape(TComponent* AOwner);
};



// File: C:\Loesungen_CB2006\Kap_8\8.7\CompU.cpp

// --------------- Aufgabe 4: TTabEdit ---------------------
__fastcall TTabEdit::TTabEdit(TComponent* Owner)
        : TEdit(Owner)
{
}

_fastcall  TTabEdit::TTabEdit(TComponent*  AOwner,TWinControl*  Parent_,int  Left_,
int Top_, int Width_, int Height_):TEdit(AOwner)
{
Parent=Parent_;
SetBounds(Left_,Top_,Width_,Height_);
}

 __fastcall  TTabEdit::TTabEdit(TComponent*  AOwner,  int  Left_,  int  Top_,  int
Width_, int Height_):TEdit(AOwner)
{
// Falls Owner und Parent gleich sind (z.B. beide Form1), ist
// dieser Konstruktor auf den ersten Blick einfacher, da man nur
// eine Argument angeben muss, und nicht zwei wie beim letzten
```

```
// Konstruktor. Aber damit ist es nicht möglich, Owner und Parent
// verschieden zu setzen.
Parent=(TWinControl*)AOwner;
SetBounds(Left_,Top_,Width_,Height_);
};

void __fastcall TTabEdit::WMNextDlgCtl(TWMChar& Msg)
{
if (Msg.CharCode == 13)
  SendMessage(((TForm*)Owner)->Handle,WM_NEXTDLGCTL,0,0);
else
  inherited::Dispatch(&Msg);
};


// --------------- Aufgabe 5: TFocusColorEdit --------------

__fastcall TFocusColorEdit::TFocusColorEdit(TComponent* AOwner):TEdit(AOwner)
{
OnEnter=MyOnEnter;
OnExit=MyOnExit;
}

__fastcall    TFocusColorEdit::TFocusColorEdit(TComponent*    AOwner,TWinControl*
Parent_,int Left_, int Top_, int Width_, int Height_):TEdit(AOwner)
{
Parent=Parent_;
Visible=true;
SetBounds(Left_,Top_,Width_,Height_);
OnEnter=MyOnEnter;
OnExit=MyOnExit;
}


void __fastcall TFocusColorEdit::SetFocusColor(TColor C)
{
FFocusColor = C;
}

#if TFocusColorEdit_Variant
void __fastcall TFocusColorEdit::WMSetFocus(TWMSetFocus& Msg)
{
prevColor = Color;
FFocusColor = clYellow;
Color = FFocusColor;
inherited::Dispatch(&Msg);
}

void __fastcall TFocusColorEdit::WMKillFocus(TWMKillFocus& Msg)
{
Color = prevColor;
inherited::Dispatch(&Msg);
}
#endif


// --------------- Aufgabe 6: TResizableMemo ---------------

__fastcall TResizableMemo::TResizableMemo(TComponent* AOwner):TMemo(AOwner)
{
}

__fastcall     TResizableMemo::TResizableMemo(TComponent*     AOwner,TWinControl*
Parent_,
              int Left_, int Top_, int Width_, int Height_):TMemo(AOwner)
{
Parent=Parent_;
//Parent=(TWinControl*)AOwner;
SetBounds(Left_,Top_,Width_,Height_);
```

```
}

void __fastcall TResizableMemo::WMLButtonDown(TWMMouseMove& Msg)
{
LB_Down = true;
Cursor = crHSplit;
// TMemo::WMLButtonDown(Msg); // Zugriff nicht möglich
TMemo::Dispatch(&Msg); // wie inherited::Dispatch(&Msg);
}

void __fastcall TResizableMemo::WMLButtonUp(TWMMouseMove& Msg)
{
LB_Down = false;
Cursor = crDefault;
inherited::Dispatch(&Msg);
}

void __fastcall TResizableMemo::WMMouseMove(TWMMouseMove& Msg)
{
if (LB_Down) Width = Msg.XPos;
inherited::Dispatch(&Msg);
}


// --------------- Aufgabe 7: TRubberShape -----------------

__fastcall TRubberShape::TRubberShape(TComponent* AOwner,TWinControl* Parent_,
      int Left_, int Top_, int Width_, int Height_):TImage(AOwner)
 {
  Parent=Parent_;
  SetBounds(Left_,Top_,Width_,Height_);
 }

__fastcall TRubberShape::TRubberShape(TComponent* AOwner):TImage(AOwner)
 {
 }

void __fastcall TRubberShape::WMLButtonDown(TWMMouseMove& Msg)
{
LB_Down = true;
x_Start = Msg.Pos.x;
y_Start = Msg.Pos.y;
x = x_Start;
y = y_Start;
r = 0;
Canvas->Pen->Mode = pmNot;
Cursor = crHSplit;
inherited::Dispatch(&Msg);
}

void __fastcall TRubberShape::WMLButtonUp(TWMMouseMove& Msg)
{
LB_Down = false;
Cursor = crDefault;
Canvas->Pen->Mode = pmBlack;
inherited::Dispatch(&Msg);
}

void __fastcall TRubberShape::RectLine(int x1,int y1,int x2,int y2)
{
TPoint p[5]={Point(x1,y1),Point(x2,y1),
                     Point(x2,y2),Point(x1,y2),Point(x1,y1)};
Canvas->Polyline(p,5);
}

void __fastcall TRubberShape::CircLine(int x,int y, int r)
{
Canvas->Arc(x-r,y-r,x+r,y+r,0,0,0,0);
}
```

```
int round (double x)
{
return x+0.5;
}

#include <cmath>
using namespace std;

void __fastcall TRubberShape::WMMouseMove(TWMMouseMove& Msg)
{
if (LB_Down)
  {
    Canvas->MoveTo(x_Start,y_Start);
    switch (rsShape)
    {
      case rsLine: Canvas->LineTo(x,y);
        break;
      case rsRectangle: RectLine(x_Start,y_Start,x,y);
        break;
      case rsCircle: CircLine(x_Start,y_Start,r);
        break;
    };
    x = Msg.Pos.x;
    y = Msg.Pos.y;
    r = round(sqrt((x-x_Start)*(x-x_Start)+(y-y_Start)*(y-y_Start)));
    Canvas->MoveTo(x_Start,y_Start);
    switch (rsShape)
    {
      case rsLine: Canvas->LineTo(x,y);
        break;
      case rsRectangle: RectLine(x_Start,y_Start,x,y);
        break;
      case rsCircle: CircLine(x_Start,y_Start,r);
        break;
    };
  }
inherited::Dispatch(&Msg);
}

void __fastcall TRubberShape::WMRButtonDblClick(TWMLButtonDown& Msg)
{
AnsiString s;
// if (rsShape < High(rsShape)) rsShape++;
if (rsShape < High) rsShape=rsShape+1;
else rsShape = Low;
// else rsShape = low(rsShape);
if (rsShape==rsLine) s = "Line";
else if (rsShape==rsRectangle) s = "Rect";
else if (rsShape==rsCircle)  s = "Circ";
else s = "Undef";
Canvas->TextOut(0,0,s);
}



// File: C:\Loesungen_CB2006\Kap_8\8.7\MsgU.cpp

#include <vcl.h>
#pragma hdrstop

#include "MsgU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
```

```
{
}


// --------------- Aufgaben 1, 2, 3 ------------------------
#include "123.cpp"


void TForm1::TestFocusTransfer()
{
if (RadioButton1->Checked)
  RadioButton1->Caption="Mit Enter in Edit1/Edit2 nicht weiter";
else
  RadioButton1->Caption="Mit Enter in Edit1/Edit2 weiter";

if (RadioButton2->Checked)
  RadioButton2->Caption="Form KeyPreview true";
else
  RadioButton2->Caption="Form KeyPreview false";
Form1->KeyPreview=RadioButton2->Checked;

if (RadioButton3->Checked)
  {
    RadioButton3->Caption="Edit1/Edit2->OnKeyPress=Edit1->OnKeyPress";
    Edit1->OnKeyPress=Edit1KeyPress;
    Edit2->OnKeyPress=Edit1KeyPress;
  }
else
  {
    RadioButton3->Caption="Edit1/Edit1->OnKeyPress=0";
    Edit1->OnKeyPress=0;
    Edit2->OnKeyPress=0;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
RadioButton1->Checked=true;
// Die nächsten beiden Zuweisungen haben denselben Effekt wie OnClick-Funktionen
// von RadioButton2 und RadioButton3 mit denselben Anweisungen wie in
// der OnClick-Funktion von RadioButton1.

RadioButton2->OnClick=RadioButton1Click;
RadioButton3->OnClick=RadioButton1Click;
}

void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
TestFocusTransfer();
}

#include "CompU.h"
#include "CompU.cpp"


// --------------- Aufgabe 4: TTabEdit ---------------------

void __fastcall TForm1::TabEditClick(TObject *Sender)
{
static int Left=GroupBox3->Left+TabEdit->Width+10;
static int Top=GroupBox3->Top+10;
Left=Left+20;
Top=Top+20;
TTabEdit* te=new TTabEdit(Form1,Form1,Left,Top,60,20);
te->Text="TabEdit "+IntToStr(Left);
}


// --------------- Aufgabe 5: TFocusColorEdit --------------
```

```
void __fastcall TForm1::FocusColorEditClick(TObject *Sender)
{
TFocusColorEdit* fce=new TFocusColorEdit(Form1,Form1,
               GroupBox4->Left+FocusColorEdit->Width+20,
               GroupBox4->Top+FocusColorEdit->Top,
               100,
               30);
fce->Parent=Form1;
}


// --------------- Aufgabe 6: TResizableMemo ---------------

void __fastcall TForm1::ResizMemoClick(TObject *Sender)
{
TResizableMemo*   tr=new   TResizableMemo(Form1,Form1,GroupBox6->Left+ResizMemo-
>Width+20,
              GroupBox6->Top+ResizMemo->Top,//+ColBordLabel->Top,
              100, 100);
tr->Text="Der rechte Rand kann mit der Maustaste verzogen "
         "werden, wenn der Mauszeiger im Memo ist";
}


// --------------- Aufgabe 7: TRubberShape -----------------

void __fastcall TForm1::RubberShapeClick(TObject *Sender)
{
TRubberShape*   tr=new   TRubberShape(Form1,Form1,RubberShape->Left+RubberShape-
>Width+20,
              RubberShape->Top,
              150,
              150);
tr->Canvas->TextOut(0,0,"");
}
```

11.9 Lösungen Kapitel 9

11.9.1 Aufgabe 9.1

```
// File: C:\Loesungen_CB2006\Kap_9\9.1\FuncTemplU.cpp

#include <vcl.h>
#pragma hdrstop
#include "FuncTemplU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;


__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}


// --------------- Aufgaben 9.1 ----------------------------
```

```
// --------------- Aufgabe 1 -------------------------------

template <class T>void vertausche(T& a, T& b)
{
T h = a;
a = b;
b = h;
}

template <typename T> bool kleiner(T x,T y)
{
return x<y;
}

template <typename T> bool kleiner(T* x,T* y)
{
return *x<*y;
}

bool kleiner(const char* x, const char* y)
{
return strcmp(x,y)<0;
}

template <typename T> void AuswahlSort(T A[],int n)
{
for (int i=0;i<n-1;i++)
  {
    int iMin = i;
    for (int j=iMin+1;j<n;j++)
      if (kleiner(A[j],A[iMin])) iMin=j;
    vertausche(A[iMin],A[i]);
  }
}

void Aufg1()
{
int a[5]={5,4,6,2,1};
AuswahlSort(a,5);
for (int i=0;i<5; i++)
  Form1->Memo1->Lines->Add(IntToStr(a[i]));

int* p[5]={new int(5),new int(4),new int(6),new int(2),new int(1)};
AuswahlSort<int*>(p,5);
for (int i=0;i<5; i++)
  Form1->Memo1->Lines->Add(IntToStr(*p[i]));

const char* n[5]={"15","14","16","02","01"};
AuswahlSort<const char*>(n,5);
for (int i=0;i<5; i++)
  Form1->Memo1->Lines->Add(n[i]);
}

void __fastcall TForm1::Button_1_1Click(TObject *Sender)
{
Aufg1();
}


// --------------- Aufgabe 2--------------------------------

#include <sstream>
#include <string>
#include "..\..\CppUtils\StringUt.h"
// a)

#ifndef STRINGUT_H__
using std::string;
```

```
template<typename T>
string to_string(T x)
{
std::ostringstream os;
os<<x;
return os.str();
}

template<typename T>
T string_to(string s, bool& success)
{
T result;
std::istringstream is(s);
is>>result;
success=(s.length()==is.tellg());
return result;
}
#endif

struct Bruch {
  int z,n; // z: Zähler, n: Nenner
};

ostream& operator<<(ostream& f, const Bruch& b)
{
return f<<b.z<<"|"<<b.n;
}

istream& operator>>(istream& f, Bruch& b)
{
char Bruchstrich;
f>>b.z>>Bruchstrich>>b.n;
return f;
}

void Aufg_2a()
{
Form1->Memo1->Lines->Add("2 a)");
Form1->Memo1->Lines->Add(("17:      "+to_string(17)).c_str());
Bruch b1={1,2}, b2={3,4};
Form1->Memo1->Lines->Add(("1/2:     "+to_string(b1)).c_str());
Form1->Memo1->Lines->Add(("3/4:     "+to_string(b2)).c_str());
Form1->Memo1->Lines->Add(("1.2345:  "+to_string(1.2345)).c_str());
Form1->Memo1->Lines->Add(("true:    "+to_string(true)).c_str());
Form1->Memo1->Lines->Add(("false:   "+to_string(false)).c_str());
Form1->Memo1->Lines->Add(("hello w: "+to_string("hello w")).c_str());
bool success;
Bruch b5=string_to<Bruch>(string("1/2"));
Bruch b6=string_to<Bruch>("3/4"); // erfordert Parameter char*
Bruch b7=string_to<Bruch>("einhalb"); // löst Exception aus
int i=string_to<int>("1");
double d=string_to<double>("2.3");
int j=string_to<int>("eins");
// Das führt zu einer Zugriffsverletzung:
// char* p=string_to<char*>(string("eins"), success);
// Form1->Memo1->Lines->Add(AnsiString("eins: ")+p);
}


// b)
// Eine explizite Spezialisierung wäre nicht so gut
#ifndef STRINGUT_H__
string to_string(bool x)
{
if (x) return "true";
else return "false";
}
```

```
template<>
bool string_to<bool>(string x, bool& success)
{
success=true;
if (x=="true") return true;
else if (x=="false") return false;
else
  {
    success=false;
    return false;
  }
}
#endif

void Aufg_2b()
{
Form1->Memo1->Lines->Add("2 b)");
bool success;
bool b1=string_to<bool>("true", success);
bool b2=string_to<bool>("false", success);
bool b3=string_to<bool>("tttrue", success);
Form1->Memo1->Lines->Add(("true:    "+to_string(true)).c_str());
Form1->Memo1->Lines->Add(("false:   "+to_string(false)).c_str());
}

// c)
// Eine explizite Spezialisierung wäre nicht so gut
#ifndef STRINGUT_H__
string to_string(AnsiString x)
{
return x.c_str();
}

template<>
AnsiString string_to<AnsiString>(string x, bool& success)
{
success=true;
return x.c_str();
}
#endif

void Aufg_2c()
{
Form1->Memo1->Lines->Add("2 c)");
AnsiString a;
bool success;
string s=to_string(a);
a=string_to<AnsiString>(s);
}


// d)
   // Die Funktions-Templates lexical_cast sind ähnlich
   // aufgebaut wie to_string und string_to. Sie enthalten aber
   // zahlreiche weitere Spezialisierungen und lösen auch
   // eine exception des Typs bad_lexical_cast aus, wenn die Konversion
   // nicht möglich ist.

#include <boost\lexical_cast.hpp>
void Aufg_2d1()
{
using namespace boost;
using boost::lexical_cast;
using boost::bad_lexical_cast;
Form1->Memo1->Lines->Add("2 b)");
Form1->Memo1->Lines->Add(("true:    "+lexical_cast<string>(true) ).c_str());
Form1->Memo1->Lines->Add(("false:   "+lexical_cast<string>(false)).c_str());
}
```

```
// Wem die etwas unhandliche Schreibweise wie in
// nicht gefällt, kann diese auch in Funktionen verpacken wie in

template<typename T> string to_string_(T x)
{
using namespace boost;
using boost::lexical_cast;
using boost::bad_lexical_cast;
return lexical_cast<string>(x);
}

template<typename T> T string_to_(string s)
{
using namespace boost;
using boost::lexical_cast;
using boost::bad_lexical_cast;
return lexical_cast<T>(s);
}

void Aufg_2d2()
{
Form1->Memo1->Lines->Add(("17:      "+to_string_(17)).c_str());
Bruch b1={1,2}, b2={3,4};
Form1->Memo1->Lines->Add(("1/2:     "+to_string_(b1)).c_str());
Form1->Memo1->Lines->Add(("3/4:     "+to_string_(b2)).c_str());
Form1->Memo1->Lines->Add(("1.2345:  "+to_string_(1.2345)).c_str());
Form1->Memo1->Lines->Add(("true:    "+to_string_(true)).c_str());
Form1->Memo1->Lines->Add(("false:   "+to_string_(false)).c_str());
Form1->Memo1->Lines->Add(("hello w: "+to_string_("hello w")).c_str());

Bruch b5=string_to_<Bruch>(string("1/2"));
Bruch b6=string_to_<Bruch>("3/4"); // erfordert Parameter char*
int i=string_to_<int>("1");
double d=string_to_<double>("2.3");
try {
  int j=string_to_<int>("eins"); // löst Exception aus
}
catch (std::exception& e) {
  Form1->Memo1->Lines->Add(e.what());
}
}

void __fastcall TForm1::Button_1_2Click(TObject *Sender)
{
Aufg_2a();
Aufg_2b();
Aufg_2c();
Aufg_2d1();
Aufg_2d2();
}


// --------------- Aufgabe 3--------------------------------

template <class T>
inline T max(const T& a, const T& b)
{
  return  a < b ? b : a;
}
template <class InputIterator, class Function>
Function for_each(InputIterator first,
InputIterator last, Function f)
{
  while (first != last) f(*first++);
  return f;
}
void print(int x)
{
Form1->Memo1->Lines->Add(IntToStr(x));
```

```
}

struct Print{
  Print(){};
  // ...
};


// /*

 // a) Der Compiler bestimmt zuerst den Datentyp der
       // Template-Argumente und verwendet diesen dann,
       // um das Template zu erzeugen.
 int max(const int& a, const int& b)
 {
    return  a < b ? b : a;
 }

 // b)
 double max(const double& a, const double& b)
 {
    return  a < b ? b : a;
 }

 // c)
 // Da die Datentypen der Argumente für a und b verschieden sind, aber
 // die Typ-Parameter denselben Datentyp haben, kann der Compiler nicht
 // entscheiden, welchen er nehmen soll.
 // Zur Lösung könnte man entweder das Template mit zwei Template-Parametern
 // oder beim Aufruf eine explizite Spezialisierung angeben.

 // d)
 typedef void (*F) (int);

 F for_each(int* first, int* last, F f)
 {
   while (first != last) f(*first++);
   return f;
 }

#include <vector>
#include <limits>
using std::vector;
  // e)
 F for_each(vector<int>::iterator first, vector<int>::iterator last, F f)
 {
   while (first != last) f(*first++);
   return f;
 }

   // f)
 Print for_each(vector<int>::iterator first, vector<int>::iterator last, Print f)
 {
////   while (first != last) f(*first++);
  // Da das Argument vom Typ Print nicht aufgerufen werden kann,
  // kann der Aufruf f(*first++) nicht übersetzt werden.
   return f;
 }

 void test_Aufg3()
{
  // test_f(Max(2,3));
  // test_f(Max(4.4,5.5));
  // test_f(Max(6,7.8));
// a)
::max(2,3);
// b)
::max(4.4, 5.5);
// c)
```

```
// max(6,7.8); // das geht nur nach der Änderung
//   template <class T, class U>
//   inline T max(const T& a, const U& b)
//   {
//     return  a < b ? b : a;
//   }


// d)
int a[10];
for_each(a,a+10,print);
// e)
vector<int> v;
for_each(v.begin(),v.end(),print);
// f)
//vector<int> v;
for_each(v.begin(),v.end(),Print());
// Hier ist Print() ein mit dem Standardkonstruktor erzeugtes
// temporäres Objekt der Klasse Print. In Abschnitt 9.3 wird
// gezeigt, wie man mit einem Objekt mit einem überladenen
// Aufrufoperator auch solche Aufrufe durchführen kann.

// Die Aufgaben e) und f) zeigen insbesondere, dass man einem
// Funktions-Template sowohl eine Funktion als auch eine Klasse
// übergeben kann.
}

void __fastcall TForm1::Button_1_3Click(TObject *Sender)
{
test_Aufg3();
}


// --------------- Aufgabe 4--------------------------------

inline bool operator==(const Bruch& q1, const Bruch& q2)
{
return q1.z*q2.n==q2.z*q1.n;
}

inline bool operator<(const Bruch& q1, const Bruch& q2)
{
return q1.z*q2.n < q2.z*q1.n;
}
/*
// Diese Funktionen sind in HP-STL Version in function.h
// so definiert:

template <class T>
inline bool operator!=(const T& x, const T& y)
{
    return !(x == y);
}

template <class T>
inline bool operator>(const T& x, const T& y)
{
    return y < x;
}

template <class T>
inline bool operator<=(const T& x, const T& y)
{
    return !(y < x);
}

template <class T>
inline bool operator>=(const T& x, const T& y)
{
```

```
    return !(x < y);
}

// und in der Dinkumware-Version so:

template<class _Ty> inline
    bool operator!=(const _Ty& _Left, const _Ty& _Right)
    {    // test for inequality, in terms of equality
    return (!(_Left == _Right));
    }

template<class _Ty> inline
    bool operator>(const _Ty& _Left, const _Ty& _Right)
    {    // test if _Left > _Right, in terms of operator<
    return (_Right < _Left);
    }

template<class _Ty> inline
    bool operator<=(const _Ty& _Left, const _Ty& _Right)
    {    // test if _Left <= _Right, in terms of operator<
    return (!(_Right < _Left));
    }

template<class _Ty> inline
    bool operator>=(const _Ty& _Left, const _Ty& _Right)
    {    // test if _Left >= _Right, in terms of operator<
    return (!(_Left < _Right));
    }
*/
// Diese Funktionen stehen nach

#include <utility>

// zur Verfügung:
void test_Aufg4()
{
using namespace std::rel_ops;

Bruch b1={1,2}, b2={3,4};
bool r1=b1<b2;
bool r2=b1<=b2;
bool r3=b1>b2;
bool r4=b1>=b2;
bool r5=b1==b2;
bool r6=b1!=b2;
}


void __fastcall TForm1::Button_1_4Click(TObject *Sender)
{
test_Aufg4();
}
```

11.9.2 Aufgabe 9.2

```
// File: C:\Loesungen_CB2006\Kap_9\9.2\ClassTemplU.cpp

/*
Boost Software License - Version 1.0 - August 17th, 2003
Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
```

```
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.

// Copyright notice for the array example:

/ * The following code declares class array,
 * an STL container (as wrapper) for arrays of constant size.
 *
 * See
 *      http://www.boost.org/libs/array/
 * for documentation.
 *
 * The original author site is at: http://www.josuttis.com/
 *
 * (C) Copyright Nicolai M. Josuttis 2001.
 *
 * Distributed under the Boost Software License, Version 1.0. (See
 * accompanying file LICENSE_1_0.txt or copy at
 * http://www.boost.org/LICENSE_1_0.txt)
 *
 * /

// Copyright notice for the scoped_ptr example:

//  (C) Copyright Greg Colvin and Beman Dawes 1998, 1999.
//  Copyright (c) 2001, 2002 Peter Dimov
//
//  Distributed under the Boost Software License, Version 1.0. (See
//  accompanying file LICENSE_1_0.txt or copy at
//  http://www.boost.org/LICENSE_1_0.txt)
//
//  http://www.boost.org/libs/smart_ptr/scoped_ptr.htm
//


*/


//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "ClassTemplU.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------


__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
// Erzeuge das Verzeichnis, falls es nicht existiert
```

```
AnsiString dir="c:\\test";
if (CreateDirectory(dir.c_str(),0))
  ShowMessage("directory '"+dir+"' created");
}


// --------------- Aufgaben 9.2 ----------------------------


// --------------- Aufgabe 1 -------------------------------

// In der Boost-Bibliothek (http://boost.org, boost\array.hpp)
// ist das fixed size array "array" im Wesentlichen
// folgendermaßen definiert (stark vereinfacht):

#include <stdexcept> // für range_error
#include <sstream>   // für Exception Text

template<class T, unsigned int N>
class Array {
  public:
    T elems[N];    // fixed-size array of elements of type T

    // kein Konstruktor, damit ein Array wie ein C-Array
    // initialisiert werden kann
  public:
    // type definitions
    typedef T              value_type;
    typedef T&             reference;
    typedef const T&       const_reference;
    typedef std::size_t    size_type;

    // operator[]
    reference operator[](unsigned int i)
    {
    if (i>=N) // bzw. (i<0 || i>=N) bei int i;
      {
        std::ostringstream msg;
        msg<<"Array index out of range: "<<i<<" > "<<(N-1);
        throw std::out_of_range(msg.str());
      }
    return elems[i];
    }

    // ohne diese Variante ist der Indexoperator für konstante
    // Arrays wie
    //    const Array<int, n> a={1,2,3};
    // nicht definiert:
    const_reference operator[](unsigned int i) const
    {
    if (i>=N) // bzw. (i<0 || i>=N) bei int i;
      {
        std::ostringstream msg;
        msg<<"Array index out of range: "<<i<<" > "<<(N-1);
        throw std::out_of_range(msg.str());
      }
    return elems[i];
    }

    // size is constant
    static size_type size() { return N; }

};


void test_Array()
{
const int n=100;
Array<int, n> a;
```

```
for (int i=0;i<n;i++) a[i]=i;
int s=0;
for (int i=0;i<n;i++) s=s+a[i];
Form1->Memo1->Lines->Add("s="+IntToStr(s));
int x=a.size();
// Da Array keine Konstruktoren hat, kann es wie ein C-Array
// initialisiert werden:
Array<int, n> ai={1,2,3};

const Array<int, n> c={1,2,3};
for (int i=0;i<n;i++) s=s+c[i];
Form1->Memo1->Lines->Add("s="+IntToStr(s));

try {
      a[n]=0; // löst eine Exception aus
    }
catch (std::exception& e)
    {
      Form1->Memo1->Lines->Add(e.what());
    }
try {
      a[-1]=0; // löst eine Exception aus
    }
catch (std::exception& e)
    {
      Form1->Memo1->Lines->Add(e.what());
    }
}

void __fastcall TForm1::Button_2_1Click(TObject *Sender)
{
test_Array();
}


// --------------- Aufgabe 2 -------------------------------

// In der C++-Standardbibliothek ist das Funktions-Template make_pair
// in <utility> folgendermaßen definiert:

using std::pair;

template <class T1, class T2>
pair<T1, T2> makePair(const T1& x, const T2& y)
{
return pair<T1, T2>(x, y);
}

#include <map>

void test_makePair()
{
using std::map;
using std::string;

map<string,string> ac;
ac.insert(makePair("Daniel","13.10.89"));
if (ac.find("Daniel")!=ac.end())
  {
    Form1->Memo1->Lines->Add("makePair funktioniert");
    string result=ac["Daniel"];
  }
else
  Form1->Memo1->Lines->Add("makePair funktioniert nicht");
}

void __fastcall TForm1::Button_2_2Click(TObject *Sender)
{
test_makePair();
```

```
}


// --------------- Aufgabe 3 -------------------------------

template <typename T>
struct Punkt1 {
  T x,y;
  Punkt1(const T& a,const T& b):x(a),y(b) {}
  Punkt1(const Punkt1& p):x(p.x),y(p.y){ };
};

template <typename T>
struct Punkt2 {
  T x,y;
  Punkt2(const T& a,const T& b):x(a),y(b) {}
  template <typename U>
  Punkt2(const Punkt2<U>& p):x(p.x),y(p.y){ };
};

void Aufg3()
{
Punkt1<int> p1a(1,2);        Punkt2<int> p2a(1,2);
Punkt1<int> p1b=p1a;         Punkt2<int> p2b=p2a;
// Punkt1<double> p1c=p1a; // ohne den zweiten Konstruktor nicht möglich
Punkt2<double> p3b=p2a;
}

void __fastcall TForm1::Button_2_3Click(TObject *Sender)
{
//
}

// --------------- Aufgabe 4 -------------------------------

// In der Boost-Bibliothek (http://boost.org, boost\scoped_ptr.hpp)
// ist das Klassen-Template scoped_ptr im Wesentlichen
// folgendermaßen definiert (stark vereinfacht).
// Dieses Template entspricht der Klasse MyVerySimpleSmartPointer.

template<class T> class scoped_ptr
{
  private:
    T * ptr;

    scoped_ptr(scoped_ptr const &);
    scoped_ptr & operator=(scoped_ptr const &);
  public:
    explicit scoped_ptr(T * p = 0): ptr(p) {} // never throws

    ~scoped_ptr() // never throws
    {
      delete ptr;
    }

    T & operator*() const // never throws
    {
        return *ptr;
    }

    T * operator->() const // never throws
    {
        return ptr;
    }
};

void testMyScopedPtr()
{
scoped_ptr<int> sp1(new int);
```

```
*sp1=17;
int* p2=new int(18);
scoped_ptr<int> sp2(p2);
// sp1=sp2; // nicht möglich
}


void __fastcall TForm1::Button_2_4Click(TObject *Sender)
{
testMyScopedPtr();
}

// --------------- Aufgabe 5 -------------------------------

template<class T>
T Average(const T* data, int numElements)
{
   T sum = 0;
   for (int i=0; i < numElements; ++i)
       sum += data[i];
   return sum / numElements;
};

double Average(const int* data, int numElements)
{
   double sum = 0;
   for (int i=0; i < numElements; ++i)
       sum += data[i];
   return sum / numElements;
};


template<class T> struct float_trait {
   typedef T      T_float;
};

template<> struct float_trait<int> {
   typedef double T_float;
};

template<> struct float_trait<char> {
   typedef double T_float;
};

template<class T>
typename float_trait<T>::T_float average(const T* data,
   int numElements)
{
   typename float_trait<T>::T_float sum = 0;
   for (int i=0; i < numElements; ++i)
       sum += data[i];
   return sum / numElements;
}

/*
Die float_trait-Variante ist nicht so leicht verständlich,
erspart aber doppelten Code (insbesondere, falls noch weitere
Varianten notwendig sind).
*/

void testAverage()
{
const int Max=2;
int d1[Max]={1,2};
double a1=Average(d1,Max);
double d2[Max]={1,2};
double a2=Average(d2,Max);

double a10=average(d1,Max);
```

```
double a21=average(d2,Max);

}

void __fastcall TForm1::Button_2_5Click(TObject *Sender)
{
testAverage();
}


//----- Aufgabe 6 ----------------------------------------------------

template<class charT> struct IntDouble_traits;
template<> struct IntDouble_traits<int>{};
template<> struct IntDouble_traits<double>{};

template <typename T, typename traits=IntDouble_traits<T> >
class NurIntDouble {
  traits t;

  T x;
};

void test_NurIntDouble()
{
NurIntDouble<int> x;
NurIntDouble<double> d;
// NurIntDouble<char> c; // das geht nicht
}

void __fastcall TForm1::Button_2_6Click(TObject *Sender)
{
test_NurIntDouble();
}


// --------------- Aufgabe 7 -------------------------------

#include "..\..\CPPUtils\BinStream.h"

void test_binary_stream()
{
binary_fstream<int> f("c:\\test\\bin",ios::out);
if (f)
 {
    for (int i=0; i<10; i++)
      if (!f.write_bin(i))
        ShowMessage("Fehler write f");
    f.close();
  }
else
  ShowMessage("Fehler open bin");
f.clear();

f.open("c:\\test\\bin");
if (f)
 {
    int x;
    while (f.read_bin(x))
      {
      Form1->Memo1->Lines->Add(IntToStr(x));
      }
    f.close();
  }
else
  ShowMessage("Fehler open bin");

binary_ifstream<int> fin("c:\\test\\bin1");
if (fin)
```

```
 {
    int x;
    while (fin.read_bin(x))
      {
      Form1->Memo1->Lines->Add(IntToStr(x));
      }
    fin.close();
  }
else
  ShowMessage("Fehler open bin1");

binary_ofstream<int> fout("c:\\test\\bin1");
if (fout)
 {
    for (int i=0; i<10; i++)
      if (!fout.write_bin(i))
        ShowMessage("Fehler write fout");
    fout.close();
  }
else
  ShowMessage("Fehler open bin1");
}

void __fastcall TForm1::Button_2_7Click(TObject *Sender)
{
test_binary_stream();
}


//----- Aufgabe 8 ----------------------------------------------------

template<class T> class myAutoPtr
{ // stark vereinfachte Version der Klasse auto_ptr
  T* ptr;
 public:
  myAutoPtr (T* p)  throw() : ptr(p) { }

  myAutoPtr(myAutoPtr<T>& a)  throw() // Copy-Konstruktor
  {
  T* tmp=a.ptr;
  a.ptr=0;
  ptr=tmp;
  }

  myAutoPtr<T>& operator= (myAutoPtr<T>& rhs)  throw() // operator=
  {
  if (ptr != rhs.ptr)
    {
      T* tmp = rhs.ptr;
      rhs.ptr = 0;
      delete ptr;
      ptr = tmp;
    }
  return *this;
  }

  ~myAutoPtr() throw() { delete ptr; }

// members
  T& operator*  ()  const throw() { return *ptr;   }
  T* operator-> ()  const throw() { return ptr;    }
  T* get        ()  const throw() { return ptr;    }
};


void test_myAutoPtr_1()
{
int* p=new int(1);
myAutoPtr<int>ap1(p);
```

```
myAutoPtr<int>ap2(p);
int* pa=new int[10];
myAutoPtr<int>ap3(pa); // zulässig, aber falsch
int i;
myAutoPtr<int>ap4(&i); // zulässig, aber falsch
myAutoPtr<int> ap(new int(1)); // ap.ptr=x mit x>0
myAutoPtr<int> aq(new int);  // aq.ptr=0
aq=ap; // aq.ptr=x, ap.ptr=0
}


class C {
 public:
  C(){
  Form1->Memo1->Lines->Add("Konstruktor ");
  }

  ~C(){
  Form1->Memo1->Lines->Add("Destruktor");
  }

};

void test_myAutoPtr_2()
{
C* pc1=new C;
C* pc2=new C;

myAutoPtr<C> p(pc1);
myAutoPtr<C> q(pc2);
p=q;
myAutoPtr<C> r(p);

C* pc;
//p=pc;
}

void __fastcall TForm1::Button_2_8Click(TObject *Sender)
{
test_myAutoPtr_1();
test_myAutoPtr_2();
}
```

11.9.3 Aufgabe 9.3

```
// File: C:\Loesungen_CB2006\Kap_9\9.3\FuncObjU.cpp

#include <vcl.h>
#pragma hdrstop
#include "FuncObjU.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

#include <algorithm>
```

```
#include <functional>
#include <string>
using std::string;
#include <vector>
using std::vector;


// --------------- Aufgaben 9.3 ----------------------------

// --------------- Aufgabe 1 -------------------------------

// a)
bool AnsiCompStr(const string& s1,const string& s2)
{ // lstrcmp verwendet den Windows-Zeichensatz
return lstrcmp(s1.c_str(),s2.c_str())<0;
// mit <=0 wäre das keine strict weak ordering
}

void Aufg1()
{
// b)
const int n=20; // Anzahl der strings in a
char* a[]={".","-","a","b","c","o","u","s","t","ä",
           "ü","Ä","Ö","Ü","ß","A","O","U","S","T"};
vector<string> v1(a,a+n);
std::sort(v1.begin(), v1.end(),AnsiCompStr);
// c)
vector<string> v2(v1.begin(),v1.end());
std::sort(v2.begin(), v2.end());

for (int i=0; i<v1.size(); i++)
  {
    Form1->Memo1->Lines->Add(AnsiString(v1[i].c_str())+" - "+v2[i].c_str());
  }
}

void __fastcall TForm1::Button_3_1Click(TObject *Sender)
{
Aufg1();
}


// --------------- Aufgabe 2 -------------------------------

// Diese Aufgabe kann auf viele verschiedene Arten gelöst werden:

template<typename T, typename Iterator>
bool is_sorted_1(Iterator first, Iterator last)
{
// Diese Version von is_sorted kann sowohl mit STL-Containern als
// auch mit Arrays verwendet werden. Sie hat aber den kleinen
// Nachteil, dass man den Datentyp immer explizit angeben muss.

// Der Typ-Parameter für den Datentyp T muss der erste Template-
// Parameter sein, damit der Datentyp des Iterators aus dem
// Argument abgeleitet werden kann.

return std::adjacent_find(first,last,std::greater<T>())==last;
}

void test_is_sorted_1()
{
const int Max=5;
int a1[Max]={2,1,2,5,7};
// Aufruf von is_sorted mit Array:
bool b1=is_sorted_1<int>(a1, a1+5); // false
int a2[Max]={1,1,2,5,7};
bool b2=is_sorted_1<int>(a2, a2+5); // true
int a3[Max]={1,1,2,7,5};
```

```
bool b3=is_sorted_1<int>(a3, a3+5); // false

vector<int> v1(a1,a1+Max);
// Aufruf von is_sorted mit vector:
bool b4=is_sorted_1<int>(v1.begin(), v1.end()); // false
};


template<typename Iterator>
bool is_sorted_2(Iterator first, Iterator last)
{ // kannn nur mit Iteratoren eines STL-Containers aufgerufen werden,
  // die ein Element value_type haben.
return std::adjacent_find(first,last,
                std::greater<Iterator::value_type>())==last;
/*
Die folgenden Anweisungen sind gleichwertig:
   typedef typename Iterator::value_type T;
   return adjacent_find(first,last,greater<T>())==last;

Die Ableitung des Typs der Elemente aus dem Typ des Iterators geht
nicht;

   typedef Iterator* T;
*/
}

template<typename Iterator, typename binaryPredicate>
bool is_sorted_2(Iterator first, Iterator last,
                                binaryPredicate comp)
{ // kann mit STL-Containern und Arrays aufgerufen werden
return adjacent_find(first,last,comp)==last;
}

template <typename Container>
bool is_sorted(const Container& c)
{ // kann wegen begin() und end() nur mit STL-Containern aufgerufen werden
return adjacent_find(c.begin(), c.end())==c.end();
}

template <typename Container, typename binaryPredicate>
bool is_sorted(const Container& c, binaryPredicate comp)
{
return adjacent_find(c.begin(), c.end() ,comp)==c.end();
}

template<typename Container>
bool is_sorted_2(const Container& c)
{ // Bei einem Container kann als Datentyp der zu vergleichenden Elemente
  // auch aus der Datentyp der Container-Elemente verwendet werden.
return (std::adjacent_find(c.begin(),c.end(),
            std::greater<Container::value_type>())==c.end());
}

void test_is_sorted_2()
{
const int Max=5;
int a1[Max]={2,1,2,5,7};
// Aufruf von is_sorted mit Array:
bool b1=is_sorted_2(a1, a1+5, std::greater<int>()); // false
int a2[Max]={1,1,2,5,7};
bool b2=is_sorted_2(a2, a2+5,std::greater<int>()); // true
int a3[Max]={1,1,2,7,5};
bool b3=is_sorted_2(a3, a3+5, std::greater<int>()); // false

// Aufruf von is_sorted mit vector:
vector<int> v1(a1,a1+Max);
bool b6=is_sorted_2(v1.begin(), v1.end(),std::greater<int>()); // false
vector<int> v2(a2,a2+Max);
bool b7=is_sorted_2(v2.begin(), v2.end(),std::greater<int>()); // true
```

```
vector<int> v3(a3,a3+Max);
bool b8=is_sorted_2(v3.begin(), v3.end(),std::greater<int>()); // false

bool b10=is_sorted(v1);
bool b11=is_sorted(v2);
bool b12=is_sorted(v3);

bool b20=is_sorted(v1,std::greater<int>());
bool b21=is_sorted(v2,std::greater<int>());
bool b22=is_sorted(v3,std::greater<int>());
}


// d)
namespace Boost_is_sorted  { // ********************************
 // In der Boost-Bibliothek findet man auch noch diese Variante:

 // http://www.boost.org/boost/detail/algorithm.hpp

 // (C) Copyright Jeremy Siek 2001.
 // Distributed under the Boost Software License, Version 1.0. (See accompany-
 // ing file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)


 template <typename ForwardIterator>
  bool is_sorted(ForwardIterator first, ForwardIterator last)
  {
    if (first == last)
      return true;

    ForwardIterator next = first;
    for (++next; next != last; first = next, ++next) {
      if (*next < *first)
        return false;
    }
    return true;
  }

  template <typename ForwardIterator, typename StrictWeakOrdering>
  bool is_sorted(ForwardIterator first, ForwardIterator last,
                 StrictWeakOrdering comp)
  {
    if (first == last)
      return true;

    ForwardIterator next = first;
    for (++next; next != last; first = next, ++next) {
      if (comp(*next, *first))
        return false;
    }

    return true;
  }

} // end of namespace N3 ********************************



inline bool notAnsiCompStr(const string& s1,const string& s2)
{ // lstrcmp verwendet den Windows-Zeichensatz
return !AnsiCompStr(s1,s2);
}

void test_is_sorted_strings()
{
using namespace std;
// b)
const int n=20; // Anzahl der strings in a
char* a[]={".","-","a","b","c","o","u","s","t","ä",
```

```
           "ü","Ä","Ö","Ü","ß","A","O","U","S","T"};
vector<string> v1(a,a+n);

bool r1=is_sorted_2(v1.begin(),v1.end(),notAnsiCompStr);
sort(v1.begin(), v1.end(),AnsiCompStr);
bool r2=is_sorted_2(v1.begin(),v1.end(),notAnsiCompStr);
}

void __fastcall TForm1::Button_3_2Click(TObject *Sender)
{
test_is_sorted_1();
test_is_sorted_2();
test_is_sorted_strings();
}


// --------------- Aufgabe 3 -------------------------------

class C {
 public:
  virtual void f()
  {
  Form1->Memo1->Lines->Add("C");
  }
  virtual void g(int i)
  {
  Form1->Memo1->Lines->Add("C-g"+IntToStr(i));
  }
};

class D : public C{
 public:
  void f()
  {
  Form1->Memo1->Lines->Add("D");
  }
  void g(int i)
  {
  Form1->Memo1->Lines->Add("D-g"+IntToStr(i));
  }
};

class E : public D{
 public:
  void f()
  {
  Form1->Memo1->Lines->Add("E");
  }
  void g(int i)
  {
  Form1->Memo1->Lines->Add("E-g"+IntToStr(i));
  }
};

void Aufg3()
{
using namespace std;
C c;
D d;
E e;
vector<C*> v;
v.push_back(&c);
v.push_back(&d);
v.push_back(&e);
// for_each(v.begin(),v.end(),C::f); // das geht nicht
// a)
for_each(v.begin(),v.end(),mem_fun(&C::f));
// b)
for_each(v.begin(),v.end(),bind2nd(mem_fun(&C::g),17));
```

```
// c)
{
vector<C> w;
w.push_back(c);
w.push_back(d);
w.push_back(e);
for_each(w.begin(),w.end(),mem_fun_ref(&C::f));
}
}

void __fastcall TForm1::Button_3_3Click(TObject *Sender)
{
Aufg3();
}
```

11.9.4 Aufgabe 9.4

```
// File: C:\Loesungen_CB2006\Kap_9\9.4\IteratU.cpp

#include <vcl.h>
#pragma hdrstop
#include "IteratU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}


// --------------- Aufgaben 9.4 ----------------------------

// --------------- Aufgabe 1 -------------------------------

#include <fstream>
#include <algorithm>
using namespace std;

// a)
template<typename T,typename Iterator>
void writeToFile(Iterator first, Iterator last, char* fn)
{
ofstream f(fn);
copy(first, last, ostream_iterator<T>(f,"\n"));
}

// b)
template<typename T> void copyFile(char* fn1,char* fn2)
{
ifstream f1(fn1);
ofstream f2(fn2);
copy(istream_iterator<T>(f1),
     istream_iterator<T>(),ostream_iterator<T>(f2,"\n"));
}

// c)
```

```
#include <iterator>
#include <vector>

template<typename T> void sortFile(char* fn1,char* fn2)
{
ifstream f1(fn1);
vector<T> v;
copy(istream_iterator<T>(f1),
     istream_iterator<T>(),back_inserter(v));
sort(v.begin(),v.end());
ofstream f2(fn2);
copy(v.begin(), v.end(), ostream_iterator<T>(f2,"\n"));
}

// d)

#include <functional>

template<typename T, typename Iterator>
bool is_sorted(Iterator first, Iterator last)
{
return adjacent_find(first,last,greater<T>())==last;
}

template<typename T>
bool FileIsSorted(char* fn)
{
ifstream f(fn);
return is_sorted<T>(istream_iterator<T>(f),istream_iterator<T>());
}

// e)

template<typename T> void print(T x)
{
Form1->Memo1->Lines->Add(x);
}

template<typename T> void showFile(char* fn)
{
print<char*>(fn);
ifstream f(fn);
for_each(istream_iterator<T>(f),istream_iterator<T>(),print<T>);
}

// g)
struct Bruch {
  Bruch():z(0),n(1) {}
  Bruch(int z_,int n_):z(z_),n(n_) {}
  int z,n; // z: Zähler, n: Nenner
};

ostream& operator<<(ostream& f, const Bruch& b)
{
return f<<b.z<<"|"<<b.n;
}

istream& operator>>(istream& f, Bruch& b)
{
char Bruchstrich;
f>>b.z>>Bruchstrich>>b.n;
return f;
}


void Aufg1()
{
// f)
```

```
int as[3]={1,2,3};                          // sortiert
writeToFile<int>(as,as+3,"s.txt");
int au[3]={1,3,2};                          // unsortiert
writeToFile<int>(au,au+3,"u.txt");
int a0[1];
writeToFile<int>(a0,a0,"0.txt");   // 0 Elemente
int a1[1]={1};
writeToFile<int>(a1,a1+1,"1.txt"); // 1 Element

bool b1=FileIsSorted<int>("s.txt");
bool b2=FileIsSorted<int>("u.txt");
bool b3=FileIsSorted<int>("0.txt");
bool b4=FileIsSorted<int>("1.txt");

copyFile<int>("s.txt", "sc.txt");
copyFile<int>("u.txt", "uc.txt");
copyFile<int>("0.txt", "0c.txt");
copyFile<int>("1.txt", "1c.txt");

sortFile<int>("s.txt", "ss.txt");
sortFile<int>("u.txt", "us.txt");
sortFile<int>("0.txt", "0s.txt");
sortFile<int>("1.txt", "1s.txt");

bool b1s=FileIsSorted<int>("ss.txt");
bool b2s=FileIsSorted<int>("us.txt");
bool b3s=FileIsSorted<int>("0s.txt");
bool b4s=FileIsSorted<int>("1s.txt");

showFile<int>("s.txt");
showFile<int>("u.txt");
showFile<int>("0.txt");
showFile<int>("1.txt");

showFile<int>("ss.txt");
showFile<int>("us.txt");
showFile<int>("0s.txt");
showFile<int>("1s.txt");

// g)
vector<Bruch> vb;
vb.push_back(Bruch(1,2));
vb.push_back(Bruch(1,3));
vb.push_back(Bruch(1,4));
writeToFile<Bruch>(vb.begin(),vb.end(),"b.txt");
}

void __fastcall TForm1::Button_4_1Click(TObject *Sender)
{
Aufg1();
}


// --------------- Aufgabe 2 -------------------------------

/* Zur Lösung dieser Aufgabe müssen der Klasse Array
   lediglich diese Definitonen hinzugefügt werden:

    typedef T*             iterator;
    iterator begin() { return elems; }
    iterator end() { return elems+N; }
*/

#include <stdexcept> // für range_error
#include <sstream>   // für Exception Text

template<class T, unsigned int N>
class Array {
  public:
```

```
    T elems[N];    // fixed-size array of elements of type T

    // kein Konstruktor, damit ein Array wie ein C-Array
    // initialisiert werden kann
  public:
    // type definitions
    typedef T              value_type;
    typedef T&             reference;
    typedef const T&       const_reference;
    typedef std::size_t    size_type;

    // operator[]
    reference operator[](unsigned int i)
    {
    if (i>=N) // bzw. (i<0 || i>=N) bei int i;
      {
        std::ostringstream msg;
        msg<<"Array index out of range: "<<i<<" > "<<(N-1);
        throw std::out_of_range(msg.str());
      }
    return elems[i];
    }

    // ohne diese Variante ist der Indexoperator für konstante
    // Arrays wie
    //    const Array<int, n> a={1,2,3};
    // nicht definiert:
    const_reference operator[](unsigned int i) const
    {
    if (i>=N) // bzw. (i<0 || i>=N) bei int i;
      {
        std::ostringstream msg;
        msg<<"Array index out of range: "<<i<<" > "<<(N-1);
        throw std::out_of_range(msg.str());
      }
    return elems[i];
    }

    // size is constant
    static size_type size() { return N; }

    // Ergänzungen:

    typedef T*             iterator;
    iterator begin() { return elems; }
    iterator end() { return elems+N; }

    // Die nächsten drei Definitionen ermöglichen den Aufruf
    // von STL-Algorithmen mit konstanten Array.
    typedef const T*       const_iterator;
    const_iterator begin() const { return elems; }
    const_iterator end() const { return elems+N; }
};

#include <algorithm>
void test_Array()
{
Array<int,5> a={1,9,-1, 6,2};
std::sort(a.begin(),a.end());


}


void __fastcall TForm1::Button_4_2Click(TObject *Sender)
{
test_Array();
}
```

```
// --------------- Aufgabe 3 -------------------------------

template<> void print(Bruch x)
{
Form1->Memo1->Lines->Add(AnsiString(x.z)+"|"+AnsiString(x.n));
}

void Aufg3()
{
ifstream f("s.txt");
istream_iterator<int> beg1(f);
istream_iterator<int> end1;
vector<int> v1(beg1,end1);
for_each(v1.begin(),v1.end(),print<int>);

ifstream fb("b.txt");
istream_iterator<Bruch> beg2(fb);
istream_iterator<Bruch> end2;
vector<Bruch> vb(beg2,end2);
for_each(vb.begin(),vb.end(),print<Bruch>);
}

void __fastcall TForm1::Button_4_3Click(TObject *Sender)
{
Aufg3();
}
```

11.9.5 Aufgabe 9.5

```
// File: C:\Loesungen_CB2006\Kap_9\9.5\AlgoU.cpp

#include <vcl.h>
#pragma hdrstop
#include "AlgoU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

#include <vector>
using std::vector;
#include <fstream>
#include <algorithm>

// --------------- Aufgaben 9.5.5 --------------------------

// --------------- Aufgabe 1 -------------------------------

// a)
template<typename Container1, typename Container2>
bool Equal(Container1 c1, Container2 c2)
{
return (c1.size()==c2.size())
         && std::equal(c1.begin(),c1.end(),c2.begin());
/* Das ist wegen der short-circuit Auswertung im Wesentlichen
```

```
   gleichwertig zu

    if (c1.size()!=c2.size())
      return false;
    else
      return equal(c1.begin(),c1.end(),c2.begin());
*/
}

// b)
template<typename Container>
int Count(Container c, typename Container::value_type x)
{
return std::count(c.begin(),c.end(),x);
};

// c)
#include <iterator>
template<typename T>
int CountFileElements(const char* fn, T y)
{ // Anzahl der Dateielemente mit dem Wert y
std::ifstream fi(fn);
return std::count(std::istream_iterator<T>(fi),
                                std::istream_iterator<T>(),y);
}

#include <set>
void Aufg1()
{
int a[5]={1,2,3,4,5};
vector<int> v(a,a+5);
std::set<char> s(a,a+5);
bool b1=Equal(s,v);
int i1=Count(v,1);
int i2=CountFileElements("c:\\test\\u.txt",1); // u.txt von Aufgabe 10.4
}

void __fastcall TForm1::Button_5_5_1Click(TObject *Sender)
{
Aufg1();
}


// --------------- Aufgabe 2 -------------------------------

// a)
template<typename T> T MinValue(char* fn)
{
std::ifstream fi(fn);
return *min_element(std::istream_iterator<T>(fi),std::istream_iterator<T>());
}

// b)
template<typename C, typename T> T MinValue(const C& a)
{
return *min_element(a.begin(),a.end());
}

// c)
template<typename T> T MinValue(T* a, int n)
{
return *std::min_element(a,a+n);
}

void Aufg2()
{
int i=MinValue<int>("c:\\test\\u.txt"); // u.txt von Aufgabe 10.4
int a[3]={3,2,5};
int ai=MinValue(a,3);
```

```
vector<int> v(a,a+3);
int vi=MinValue<vector<int>,int >(v);
}

void __fastcall TForm1::Button_5_5_2Click(TObject *Sender)
{
Aufg2();
}

// --------------- Aufgabe 3 -------------------------------

/* Zur Orientierung die Algorithmen aus stl.zip

template <class InputIterator, class Function>
Function for_each (InputIterator first,
InputIterator last, Function f)
{
    while (first != last) f(*first++);
    return f;
}

template <class InputIterator, class Predicate>
InputIterator find_if(InputIterator first, InputIterator
last, Predicate pred)
{
while (first != last && !pred(*first)) ++first;
    return first;
}

string s="123 und 123";
string::iterator p=find(s.begin(),s.end(),'2');
while (p!=s.end())
  {
    Form1->Memo1->Lines->Add(*p);
    p=find(p+1,s.end(),'2');
  }
*/


// a)
template<typename Container, typename Predicate, typename Function>
void for_each_if(Container c, Predicate pred, Function f)
{
Container::iterator i=find_if(c.begin(),c.end(),pred);
while (i!=c.end())
  {
    f(*i);
    i=find_if(++i,c.end(),pred);
  }
}

// b)
template<typename T, typename Predicate, typename Function>
void for_each_if(T* a, int n, Predicate pred, Function f)
{
T* i=std::find_if(a,a+n,pred);
while (i!=a+n)
  {
    f(*i);
    i=std::find_if(++i,a+n,pred);
  }
}

// c)
template<typename T, typename Predicate, typename Function>
void for_each_if(char* fn, Predicate pred, Function f)
{
std::ifstream fi(fn);
const std::istream_iterator<T> end=std::istream_iterator<T>();
```

```
std::istream_iterator<T> i=std::find_if(std::istream_iterator<T>(fi),end,pred);
while (i!=end)
  {
    f(*i);
    i=find_if(++i,end,pred);
  }
}

bool all(int i)
{
return true;
}

bool odd(int i)
{
return i%2==1;
}

void print(int i)
{
Form1->Memo1->Lines->Add(i);
}

bool first_odd(std::pair<int,int> i)
{
return i.first%2==1;
}

void print_pair(std::pair<int,int> i)
{
Form1->Memo1->Lines->Add(IntToStr(i.first)+": "+IntToStr(i.second));
}

// Auch diese Lösungsvarianten sind möglich:
template<typename Container, typename Predicate, typename Function>
void for_each_if_1(Container c, Predicate pred, Function f)
{
for (Container::iterator i=c.begin();i!=c.end();i++)
  if (pred(*i)) f(*i);
}

template<typename T, typename Predicate, typename Function>
void for_each_if_1(T* a, int n, Predicate pred, Function f)
{
for (int i=0;i<n;i++)
  if (pred(*(a+i))) f(*(a+i));
}

template<typename T, typename Predicate, typename Function>
void for_each_if_1(char* fn, Predicate pred, Function f)
{
ifstream fi(fn);
T i;
while (!(fi>>i))
  {
    if (pred(i))
      f(i);
  }
}

#include <map>
void Aufg3()
{
int a[5]={1,2,3,4,5};
vector<int> vi(a,a+5);
Form1->Memo1->Lines->Add("1-5:");
for_each_if(vi,all,print);
for_each_if_1(vi,all,print);
```

```
for_each_if(a, 5, all, print);
for_each_if_1(a, 5, all, print);
Form1->Memo1->Lines->Add("u:");
for_each_if<int>("c:\\test\\u.txt",all,print);
for_each_if_1<int>("c:\\test\\u.txt",all,print);

Form1->Memo1->Lines->Add("odd:");
for_each_if(vi,odd,print);
for_each_if_1(vi,odd,print);

for_each_if(a, 5, odd, print);
for_each_if_1(a, 5, odd, print);
// return;
Form1->Memo1->Lines->Add("odd-2:");
for_each_if<int>("c:\\test\\u.txt",odd,print);

for_each_if_1(vi,all,print);

std::map<int,int> m;
m.insert(std::make_pair(1,1));
m.insert(std::make_pair(2,4));
m.insert(std::make_pair(3,9));
for_each_if(m,first_odd,print_pair);
}

void __fastcall TForm1::Button_5_5_3Click(TObject *Sender)
{
Aufg3();
}


// --------------- Aufgabe 4 -------------------------------

namespace my_STL {

/*
In algo.h aus "ftp://butler.hpl.hp.com/stl/stl.zip" sind
nur ältere Versionen von count und count_if enthalten, die
nicht mehr zum aktuellen C++-Standard gehören.

In der STL von SGI (http://www.sgi.com/tech/stl/download.html)
sind diese Algorithmen folgendermaßen implementiert:
*/

using namespace std;

template <class InputIterator, class T>
typename iterator_traits<InputIterator>::difference_type
count(InputIterator first, InputIterator last, const T& value)
{
  typename iterator_traits<InputIterator>::difference_type n = 0;
  for ( ; first != last; ++first)
    if (*first == value)
      ++n;
  return n;
}

template <class InputIterator, class Predicate>
typename iterator_traits<InputIterator>::difference_type
count_if(InputIterator first, InputIterator last, Predicate pred)
{
  typename iterator_traits<InputIterator>::difference_type n = 0;
  for ( ; first != last; ++first)
    if (pred(*first))
      ++n;
  return n;
}

/* **** alte Versionen (nicht mehr im C++-Standard, aber aus STL.zip
```

```
template <class InputIterator, class T, class Size>
void count(InputIterator first, InputIterator last, const T& value,
       Size& n) {
    while (first != last)
    if (*first++ == value) ++n;
}

template <class InputIterator, class Predicate, class Size>
void count_if(InputIterator first, InputIterator last, Predicate pred,
          Size& n) {
    while (first != last)
    if (pred(*first++)) ++n;
}
*/

// In algo.h aus "ftp://butler.hpl.hp.com/stl/stl.zip" sind
// die Algorithmen min_element usw. folgendermaßen implementiert:

template <class ForwardIterator>
ForwardIterator min_element(ForwardIterator first, ForwardIterator last)
{
    if (first == last) return first;
    ForwardIterator result = first;
    while (++first != last)
    if (*first < *result) result = first;
    return result;
}

template <class ForwardIterator, class Compare>
ForwardIterator min_element(ForwardIterator first, ForwardIterator last,
                Compare comp)
{
    if (first == last) return first;
    ForwardIterator result = first;
    while (++first != last)
    if (comp(*first, *result)) result = first;
    return result;
}

template <class ForwardIterator>
ForwardIterator adjacent_find(ForwardIterator first, ForwardIterator last)
{
    if (first == last) return last;
    ForwardIterator next = first;
    while(++next != last) {
    if (*first == *next) return first;
    first = next;
    }
    return last;
}

template <class ForwardIterator, class BinaryPredicate>
ForwardIterator adjacent_find(ForwardIterator first, ForwardIterator last,
                  BinaryPredicate binary_pred)
{
    if (first == last) return last;
    ForwardIterator next = first;
    while(++next != last) {
    if (binary_pred(*first, *next)) return first;
    first = next;
    }
    return last;
}


template<typename Container, typename Function>
void for_each(Container c, Function f)
{
std::for_each(c.begin(),c.end(),f);
```

```
}

template<typename T, typename Function>
void for_each(char* fn, Function f)
{
ifstream fi(fn);
std::for_each(istream_iterator<T>(fi),istream_iterator<T>(),f);
}

template<typename T, typename Function>
void for_each(T* a, int n, Function f)
{
std::for_each(a, a+n, f);
}

} // end of namespace my_STL


void __fastcall TForm1::Button_5_5_5Click(TObject *Sender)
{
int a[5]={1,5,2,4,4};
vector<int> v(a,a+5);
std::set<double> s(a,a+5);
my_STL::for_each(v,print);
my_STL::for_each(s,print);
my_STL::for_each<int>("c:\\test\\u.txt",print);
my_STL::for_each(a,5,print);
int i=my_STL::count(v.begin(),v.end(),1);
int j=my_STL::count_if(v.begin(),v.end(),odd);
int k=*my_STL::min_element(v.begin(),v.end());
bool found=my_STL::adjacent_find(v.begin(),v.end())!=v.end();
}


//------ Aufgabe 9.5.19 ---------------------------------------------

// --------------- Aufgabe 1 -------------------------------

template <typename Container>
typename Container::value_type Median(Container c)
{
nth_element(c.begin(),c.begin()+c.size()/2,c.end());
return *(c.begin()+c.size()/2);
}

void test_Median()
{
vector<double> v;
double md=Median(v);
}

void __fastcall TForm1::Button_5_19_1Click(TObject *Sender)
{
test_Median();
}


// --------------- Aufgabe 2 -------------------------------

template <typename Container1, typename Container2>
void MaxElements(Container1 src, int n, Container2& dest)
{
partial_sort(src.begin(),src.begin()+n,src.end(),
             std::greater<typename Container1::value_type>());
copy(src.begin(),src.begin()+n,back_inserter(dest));
}

template <typename T, typename Container>
void MaxElements(char* fn, int n, Container& dest)
```

```
{
ifstream f(fn);
vector<T> v;
std::copy(std::istream_iterator<T>(f),std::istream_iterator<T>(),std::back_insert
er(v));
MaxElements(v,n,dest);
}

template <typename T, typename Container2>
void MaxElements(T* src, int n, int m, Container2& dest)
{
std::partial_sort(src,src+n,src+m,std::greater<T>());
copy(src,src+n,back_inserter(dest));
}


void test_MaxElements()
{
int a[6]={1,2,3,4,5,6};
vector<int> v(a,a+6);
vector<int> s;
MaxElements(v,3,s);
MaxElements(a,2,6,s);
my_STL::for_each(s,print);
MaxElements<int>("c:\\test\\u.txt",2,s);
my_STL::for_each(a,6,print);
my_STL::for_each(s,print);
}

void __fastcall TForm1::Button_5_19_2Click(TObject *Sender)
{
test_MaxElements();
}


// --------------- Aufgabe 3 -------------------------------

template <typename Container>
void Remove(Container& c, typename Container::value_type x)
{
c.erase(remove(c.begin(), c.end(),x),c.end());
}

void test_Remove()
{
int a[4]={1,2,3,4};
vector<int> v(a,a+4);
Remove (v,2);
my_STL::for_each(v,print);
}

void __fastcall TForm1::Button_5_19_3Click(TObject *Sender)
{
test_Remove();
}


// --------------- Aufgabe 4 -------------------------------

template <typename T>
void merge_files(char* in1fn, char* in2fn, char* outfn)
{
using namespace std;
ifstream in1(in1fn);
ifstream in2(in2fn);
ofstream out(outfn);
merge(istream_iterator<T>(in1),
      istream_iterator<T>(),
      istream_iterator<T>(in2),
```

```
      istream_iterator<T>(),
      ostream_iterator<T>(out,"\n"));
}

void make_file(char* fn, int n)
{
ofstream f(fn);
for (int i=0; i<n; i++)
  f<<i<<std::endl;
}


void test_merge_files()
{
using std::string;
string s1="125", s2="126", s;
bool b1=binary_search(s1.begin(),s1.end(),'2'); // b=true
bool b2=binary_search(s1.begin(),s1.end(),'3'); // b=false
merge(s1.begin(),s1.end(),s2.begin(),s2.end(),back_inserter(s));  // s="112256"

char* in1fn="int1.dat";
char* in2fn="int2.dat";
char* outfn="int.dat";
make_file(in1fn, 100);
make_file(in2fn, 100);

merge_files<int>(in1fn,in2fn, outfn);
}

void __fastcall TForm1::Button_5_19_4Click(TObject *Sender)
{
test_merge_files();
}

// --------------- Aufgabe 5 -------------------------------

void write_call(vector<int> p, ofstream& f)
{
  f<<"f(";
  for (vector<int>::iterator i=p.begin();i!=p.end()-1;i++)
    f<<*i<<",";
  f<<*(p.end()-1)<<");"<<std::endl;
}

void make_calls()
{
ofstream f("calls.cpp");
int args[4]={1,2,3,4};
vector<int> p(args,args+4);
write_call(p,f);
while (next_permutation(p.begin(),p.end()))
  write_call(p,f);
}

void __fastcall TForm1::Button_5_19_5Click(TObject *Sender)
{
using namespace std;
string s1="1544256", s2=s1;
partition(s1.begin(),s1.end(), bind1st(greater<char>
  (),'4' )); // s=1244556
stable_partition(s2.begin(),s2.end(),
 bind1st(greater<char>(),'4' )); // s=1254456
make_calls();
}
```

11.10 Lösungen Kapitel 10

11.10.1 Aufgabe 10.1

```
// File: C:\Loesungen_CB2006\Kap_10\10.1\FormularAnsichtAlsText.txt

Die Struktur sieht man in der Strukturansicht (Ansicht|Struktur)
oder in der Ansicht des Formulars als Text (im Kontextmenü
des Formulars: Ansicht als Text):
object Form1: TForm1
  Left = 0
  Top = 0
  Caption = 'Form1'
  ClientHeight = 286
  ClientWidth = 426
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object ActionMainMenuBar1: TActionMainMenuBar
    Left = 0
    Top = 0
    Width = 426
    Height = 24
    UseSystemFont = False
    ActionManager = ActionManager1
    Caption = 'ActionMainMenuBar1'
    ColorMap.HighlightColor = 15660791
    ColorMap.BtnSelectedColor = clBtnFace
    ColorMap.UnusedColor = 15660791
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -11
    Font.Name = 'Tahoma'
    Font.Style = []
    ParentShowHint = False
    ShowHint = True
    Spacing = 0
  end
  object ControlBar1: TControlBar
    Left = 0
    Top = 24
    Width = 426
    Height = 33
    Align = alTop
    TabOrder = 1
    object ActionToolBar1: TActionToolBar
      Left = 32
      Top = 2
      Width = 81
      Height = 48
      ActionManager = ActionManager1
      Caption = 'ActionToolBar1'
      ColorMap.HighlightColor = 15660791
      ColorMap.BtnSelectedColor = clBtnFace
      ColorMap.UnusedColor = 15660791
```

```
      Spacing = 0
    end
    object ActionToolBar2: TActionToolBar
      Left = 201
      Top = 2
      Width = 200
      Height = 48
      ActionManager = ActionManager1
      Caption = 'ActionToolBar2'
      ColorMap.HighlightColor = 15660791
      ColorMap.BtnSelectedColor = clBtnFace
      ColorMap.UnusedColor = 15660791
      Spacing = 0
    end
  end
  object Memo1: TMemo
    Left = 0
    Top = 57
    Width = 426
    Height = 229
    Align = alClient
    Lines.Strings = (
      'Memo1')
    TabOrder = 2
  end
  object ActionManager1: TActionManager
    FileName = 'MostRecentlyUsed.dat'
    ActionBars = <
      item
        Items = <
          item
            Items = <
              item
                Action = FileOpen1
                ImageIndex = 3
                ShortCut = 16463
              end
              item
                Action = FileSaveAs1
                ImageIndex = 4
              end
              item
                Action = FilePrintSetup1
              end
              item
                Action = FileExit1
                ImageIndex = 5
              end>
            Caption = '&Datei'
          end
          item
            Items = <
              item
                Action = EditCut1
                ImageIndex = 0
                ShortCut = 16472
              end
              item
                Action = EditCopy1
                ImageIndex = 1
                ShortCut = 16451
              end
              item
                Action = EditPaste1
                ImageIndex = 2
                ShortCut = 16470
              end>
            Caption = '&Bearbeiten'
          end>
```

```
      ActionBar = ActionMainMenuBar1
    end
    item
      Items = <
        item
          Action = FileOpen1
          ImageIndex = 3
          ShortCut = 16463
        end
        item
          Action = FileExit1
          ImageIndex = 5
        end
        item
          Action = FilePrintSetup1
        end
        item
          Action = FileSaveAs1
          ImageIndex = 4
        end>
    end
    item
      Items = <
        item
          Action = FileOpen1
          ImageIndex = 3
          ShortCut = 16463
        end
        item
          Action = FileSaveAs1
          ImageIndex = 4
        end>
    end
    item
      Items = <
        item
          Action = FileExit1
          ImageIndex = 5
        end
        item
          Action = FilePrintSetup1
        end
        item
          Action = FileSaveAs1
          ImageIndex = 4
        end
        item
          Action = FileOpen1
          ImageIndex = 3
          ShortCut = 16463
        end>
    end
    item
      Items.CaptionOptions = coNone
      Items = <
        item
          Action = EditPaste1
          ImageIndex = 2
          ShortCut = 16470
        end
        item
          Action = EditCopy1
          ImageIndex = 1
          ShortCut = 16451
        end
        item
          Action = FileExit1
          ImageIndex = 5
        end
```

```
        item
          Action = FileSaveAs1
          ImageIndex = 4
        end
        item
          Action = FileOpen1
          ImageIndex = 3
          ShortCut = 16463
        end
        item
          Action = EditCut1
          ImageIndex = 0
          ShortCut = 16472
        end>
      end
      item
        Items.CaptionOptions = coNone
        Items = <
        item
          Action = FileSaveAs1
          ImageIndex = 4
        end
        item
          Action = FileOpen1
          ImageIndex = 3
          ShortCut = 16463
        end>
        ActionBar = ActionToolBar1
      end
      item
        Items.CaptionOptions = coNone
        Items = <
        item
          Action = EditPaste1
          ImageIndex = 2
          ShortCut = 16470
        end
        item
          Action = EditCopy1
          ImageIndex = 1
          ShortCut = 16451
        end
        item
          Action = CustomizeActionBars1
        end
        item
          Action = EditCut1
          ImageIndex = 0
          ShortCut = 16472
        end>
        ActionBar = ActionToolBar2
      end>
    Images = ImageList1
    Left = 16
    Top = 232
    StyleName = 'XP Style'
    object Action1: TAction
      Caption = 'Action1'
    end
    object EditCut1: TEditCut
      Category = 'Bearbeiten'
      Caption = '&Ausschneiden'
      Enabled = False
      Hint = 'Ausschneiden|Markiertes Objekt in die Zwischenablage verschieben'
      ImageIndex = 0
      ShortCut = 16472
    end
    object EditCopy1: TEditCopy
      Category = 'Bearbeiten'
```

```
      Caption = '&Kopieren'
      Enabled = False
      Hint = 'Kopieren|Markiertes Objekt in die Zwischenablage kopieren'
      ImageIndex = 1
      ShortCut = 16451
    end
    object EditPaste1: TEditPaste
      Category = 'Bearbeiten'
      Caption = '&Einf'#252'gen'
      Hint = 'Einf'#252'gen|Inhalt der Zwischenablage einf'#252'gen'
      ImageIndex = 2
      ShortCut = 16470
    end
    object FileOpen1: TFileOpen
      Category = 'Datei'
      Caption = #214'&ffnen...'
      Hint = #214'ffnen|Vorhandene Datei '#246'ffnen'
      ImageIndex = 3
      ShortCut = 16463
      BeforeExecute = FileOpen1BeforeExecute
      OnAccept = FileOpen1Accept
    end
    object FileSaveAs1: TFileSaveAs
      Category = 'Datei'
      Caption = 'Speichern &unter...'
      Hint = 'Speichern unter|Aktive Datei unter einem neuen Namen speichern'
      ImageIndex = 4
      OnAccept = FileSaveAs1Accept
    end
    object FilePrintSetup1: TFilePrintSetup
      Category = 'Datei'
      Caption = '&Druckereinstellungen...'
      Hint = 'Druckereinstellungen'
    end
    object FileExit1: TFileExit
      Category = 'Datei'
      Caption = '&Beenden-1'
      Hint = 'Beenden|Anwendung beenden'
      ImageIndex = 5
    end
    object CustomizeActionBars1: TCustomizeActionBars
      Category = 'Tools'
      Caption = 'A&npassen...'
      CustomizeDlg.StayOnTop = False
    end
  end
  object ImageList1: TImageList
    Left = 64
    Top = 232
    Bitmap = {
    .....
      }
  end
end
```

11.10.2 Aufgabe 10.2

```
// File: C:\Loesungen_CB2006\Kap_10\10.2\Unit1.cpp

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
```

```
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::LoginClick(TObject *Sender)
{
if (Form2->ShowModal()==mrOk)
  {
    Form1->Caption=Form2->Edit1->Text;
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
if (MessageDlg("Bitte bestätigen Sie den Kaufvertrag", mtInformation,
      TMsgDlgButtons() << mbOK << mbCancel, 0)==mrOk)
  {
    ShowMessage("Wir buchen den Rechnungsbetrag von Ihrem Konto ab");
  }
else
  {
    ShowMessage("Das werden Sie noch bereuen");
  }
 }
//---------------------------------------------------------------------------



// File: C:\Loesungen_CB2006\Kap_10\10.2\Unit2.cpp

#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
    : TForm(Owner)
{
   // Diese Eigenschaften kann man auch im Objektinspektor setzen:
   Abbrechen->Cancel=true;
   Abbrechen->ModalResult=mrCancel;
   OK->Default=true;
   OK->ModalResult=mrOk;
}
//---------------------------------------------------------------------------



// File: C:\Loesungen_CB2006\Kap_10\10.2\Kontobewegung.h

#ifndef KontobewegungH
#define KontobewegungH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
```

```
class TKontobewegung_Frame : public TFrame
{
__published:    // Von der IDE verwaltete Komponenten
    TGroupBox *GroupBox1;
    TLabel *Label1;
    TLabel *Kontoinhaber;
    TLabel *Datum;
    TLabel *Betrag;
    TEdit *Edit1;
    TEdit *Edit2;
    TLabel *Bewegungsart;
    TEdit *Edit3;
    TEdit *Edit4;
    TEdit *Edit5;
private:    // Benutzer-Deklarationen
public:        // Benutzer-Deklarationen
    __fastcall TKontobewegung_Frame(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TKontobewegung_Frame *Kontobewegung_Frame;
//---------------------------------------------------------------------------
#endif
```

11.10.3 Aufgabe 10.4

```
// File: C:\Loesungen_CB2006\Kap_10\10.4\TreeViewU.cpp

#include <vcl.h>
#pragma hdrstop
#include "TreeViewU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
TreeView1->Items->Add(TreeView1->TopItem, "Meine Struktur:");

ListView1->Columns->Add();
ListView1->Columns->Items[0]->Caption="H1";
ListView1->Columns->Add()->Caption="H2";
ListView1->Columns->Add()->Caption="H3";
}

// --------------- Aufgabe 10.4, 1 -------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
TreeView1->Items->BeginUpdate();
for (int i=0 ; i<10;i++)
  {
    TTreeNode* n=TreeView1->Items->
                  Add(TreeView1->TopItem,IntToStr(i));
    for (int j=0 ;j<20;j++ )
      {
        TTreeNode* n1=TreeView1->Items->
                      AddChild(n,IntToStr(i)+"-"+IntToStr(j));
        for (int k=0 ;k<30;k++ )
          {
            TreeView1->Items->AddChild(n1,
                  IntToStr(i)+"-"+IntToStr(j)+"-"+IntToStr(k));
          }
```

```
      }
  }
TreeView1->Items->EndUpdate();
}


// --------------- Aufgabe 10.4, 3  ------------------------

// --------------- Aufgabe 10.4, 3 a) ----------------------

void __fastcall TForm1::NewSiblingNodeClick(TObject *Sender)
{
if (TreeView1->Selected!=0)
   TreeView1->Items->Add(TreeView1->Selected,Edit1->Text);
}

void __fastcall TForm1::NewSubNodeClick(TObject *Sender)
{
if (TreeView1->Selected!=0)
  TreeView1->Items->AddChild(TreeView1->Selected,Edit2->Text);
}

void __fastcall TForm1::ClearNode1Click(TObject *Sender)
{
if (TreeView1->Selected!=0)
  TreeView1->Selected->Delete();
}

void __fastcall TForm1::ClearTree1Click(TObject *Sender)
{
TreeView1->Items->Clear();
}


// --------------- Aufgabe 10.4, 3 b) ----------------------

void __fastcall TForm1::TreeView1Click(TObject *Sender)
{
if (TreeView1->Selected!=0)
  {
    ListView1->Items->Clear();
    ListView1->Items->BeginUpdate();
    TTreeNode* n=TreeView1->Selected->getFirstChild();
    while (n!=0)
      {
        TListItem* i=ListView1->Items->Add(); // erzeugt Item[0]
        i->Caption=n->Text;
        n=n->getNextSibling();
      }
    ListView1->Items->EndUpdate();
  }
// etwas besser:
/* if (TreeView1->Selected!=0)
  {
    ListView1->Items->Clear();
    try {
      ListView1->Items->BeginUpdate();
      TTreeNode* n=TreeView1->Selected->getFirstChild();
      while (n!=0)
        {
          TListItem* i=ListView1->Items->Add(); // erzeugt Item[0]
          i->Caption=n->Text;
          n=n->getNextSibling();
        }
    } __finally  {
      ListView1->Items->EndUpdate();
    }
  }
*/
```

```
}

// --------------- Aufgabe 10.4, 3 c) ----------------------

void __fastcall TForm1::RadioButtonIconClick(TObject *Sender)
{
ListView1->ViewStyle=vsIcon;
}
//---------------------------------------------------------------------
void __fastcall TForm1::RadioButtonListClick(TObject *Sender)
{
ListView1->ViewStyle=vsList;
}
//---------------------------------------------------------------------
void __fastcall TForm1::RadioButtonSmallIconClick(TObject *Sender)
{
ListView1->ViewStyle=vsSmallIcon;
}
//---------------------------------------------------------------------
void __fastcall TForm1::RadioButtonReportClick(TObject *Sender)
{
ListView1->ViewStyle=vsReport;
}


// --------------- Aufgabe 10.4, 3 d) ----------------------

void __fastcall TForm1::ButtonLoadFromFileClick(TObject *Sender)
{
if (OpenDialog1->Execute())
  {
    TreeView1->LoadFromFile(OpenDialog1->FileName);
  }
}

void __fastcall TForm1::ButtonSaveToFileClick(TObject *Sender)
{
if (SaveDialog1->Execute())
  {
    TreeView1->SaveToFile(SaveDialog1->FileName);
  }
}
```

11.10.4 Aufgabe 10.4.2 DirTree

```
// File: C:\Loesungen_CB2006\Kap_10\10.4\DirTreeU.cpp

#include <vcl.h>
#pragma hdrstop
#include "DirTreeU.h"
// #include "DirTreeDebugForm.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
// #pragma link "cdiroutl"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
//Form1->Caption="Show directory tree and size ";
Form1->Caption="Show directory tree ";
```

```
TreeView1->Align=alLeft;
Occupied_->Checked=true;
sumSubDir->Checked=true;

ListView1->Columns->Add();
ListView1->Columns->Items[0]->Caption="File";
ListView1->Columns->Add()->Caption="Size";
ListView1->Columns->Add()->Caption="Date";
ListView1->Columns->Items[0]->Width=300;
ListView1->Columns->Items[1]->Width=100;
ListView1->Columns->Items[2]->Width=200;

EditMask->Text="*";
}
/*
Die Elemente des ListView kann man auch über die Eigenschaft Item von
ListViewItemCollection und ihren Index ansprechen (Item[0], Item[1] usw.). Die
Anzahl der Elemente ist der Wert der Eigenschaft Count.
Beispiel:     Mit den folgenden Anweisungen erhält man dasselbe ListView wie im
letzten Beispiel:
listView1->Items->Add(); // erzeugt Item[0]
listView1->Items->Add(); // erzeugt Item[1]
listView1->Items->Add(); // erzeugt Item[2]
listView1->Items->Item[0]->Caption="a";
listView1->Items->Item[1]->Caption="b";
listView1->Items->Item[2]->Caption="c";
listView1->Items->Item[2]->SubItems->Add("c1");
listView1->Items->Item[2]->SubItems->Add("c1");

*/

//----- Aufgabe 10.3.2 ------------------------------------------------

void showFilesOfDirectory(TListView* lv, const AnsiString& path)
{
lv->Items->Clear();
lv->Items->BeginUpdate();
lv->Columns->Items[0]->Caption=path;
WIN32_FIND_DATA FindFileData;
HANDLE h=FindFirstFile(
            path.c_str(),    // Zeiger auf den Dateinamen
            &FindFileData); // Zeiger auf struct
if (h!=INVALID_HANDLE_VALUE)
  {
    int found=1;
    while (found)
      {
        if ((FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)!=0)
          ; // skip subdirectories
        else
          {
          TListItem* i=lv->Items->Add();
          if (i!=0)
            {
              i->Caption=FindFileData.cFileName;

              LARGE_INTEGER c;
              c.u.LowPart=FindFileData.nFileSizeLow;
              c.u.HighPart=FindFileData.nFileSizeHigh;
              i->SubItems->Add(FloatToStr(c.QuadPart));

              SYSTEMTIME st;
              if (FileTimeToSystemTime(&FindFileData.ftLastWriteTime,&st))
                i->SubItems->Add(DateTimeToStr(SystemTimeToDateTime(st)));
            }
          }
        found=FindNextFile(h,&FindFileData);
      }
  }
```

```
FindClose(h);
lv->Items->EndUpdate();
}

// Die Funktionen ClusterSize und Occupied werden nur benötigt, um
// nicht nur die Summe der Dateigrößen anzuzeigen, sondern auch den
// von den Dateien belegten Speicherplatz
// Falls use_occupied false ist, wird als Verzeichnisgröße der Wert
// angezeigt, der im Windows Explorer unter Eigenschaften eines
// Verzeichnisses als "Größe" bezeichnet wird. Andernfalls wird der
// unter "Größe auf dem Datenträger" angezeigte Wert berechnet.

int ClusterSize(char* root)
{
// "d:\\"
LPCTSTR RootPathName=root;    // address of root path
DWORD SectorsPerCluster;    // address of sectors per cluster
DWORD BytesPerSector;    // address of bytes per sector
DWORD NumberOfFreeClusters;    // address of number of free clusters
DWORD TotalNumberOfClusters;     // address of total number of clusters

bool res1=GetDiskFreeSpace(
    RootPathName,    // address of root path
    &SectorsPerCluster,    // address of sectors per cluster
    &BytesPerSector,    // address of bytes per sector
    &NumberOfFreeClusters,    // address of number of free clusters
    &TotalNumberOfClusters     // address of total number of clusters
   );
return SectorsPerCluster*BytesPerSector;
}

bool use_occupied=true;

long long Occupied(long long x)
{
AnsiString Laufwerk=AnsiString(Form1->DriveComboBox1->Drive)+":\\";

static int cs=ClusterSize(Laufwerk.c_str());
//Form1->Caption="Cluster size: "+IntToStr(cs);
if (use_occupied)
  {
    if (x==0) return 0;
    else if (x%4096==0) return x;
    __int64  y=((x/cs)+1)*cs;
    return y;
  }
else return x;
}

//----- Aufgabe 10.3.4, a) -------------------------------------------

void   SearchSubdirs(AnsiString   path,   TTreeNode*   n,   TTreeView*   tv,   bool
firstCall=false)
{
if (firstCall)
  n=tv->Items->Add(tv->TopItem,path);
else
  n=tv->Items->AddChild(n,path);
WIN32_FIND_DATA FindFileData;
AnsiString pattern=path+"\\*"; // for directories always "*"
HANDLE h=FindFirstFile(pattern.c_str(),&FindFileData);
if (h!=INVALID_HANDLE_VALUE)
  {
    int found=1;
    while (found)
      {
        if ((FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)!=0)
          { // Unterverzeichnis
            if ( (strcmp(FindFileData.cFileName,".")==0) ||
```

```
                 (strcmp(FindFileData.cFileName,"..")==0) )
               ; // ignore "." and ".."
             else // Verzeichnis, rekursiv durchsuchen
               SearchSubdirs(path+"\\"+FindFileData.cFileName,n,tv);
           }
         else; // Dateien ignorieren
         found=FindNextFile(h,&FindFileData);
       }
   }
FindClose(h);
};



//----- Aufgabe 10.3.4, a) -------------------------------------------

// a)
/*
class MyTreeNode:public TTreeNode {
 public:
  AnsiString FileName;
//double FilesSize;
  MyTreeNode(TTreeNodes* AOwner,AnsiString fn_,double fs):
    TTreeNode(AOwner),FileName(fn_)/ *,FilesSize(fs)* / {}
};

void __fastcall TForm1::TreeView1CreateNodeClass(TCustomTreeView *Sender,
      TTreeNodeClass &NodeClass)
{
NodeClass=__classid(MyTreeNode);
}

*/
//----- Aufgabe 6.6.6.4 ----------------------------------------------
void __fastcall TForm1::TreeView1Click(TObject *Sender)
{ // Füllt ListView1 mit den Dateien im Verzeichnis FileName
ListView1->ViewStyle=vsReport;
if (TreeView1->Selected!=0)
  {
    TTreeNode* t1=TreeView1->Selected;
    AnsiString fn=t1->Text;
    if (sumSubDir->Checked)
      fn=*static_cast<AnsiString*>(t1->Data);
    showFilesOfDirectory(ListView1, fn+"\\"+Form1->EditMask->Text);
  }
}

// b)

//----- Aufgabe 10.3.4, a) -------------------------------------------

long long SearchSubdirs(AnsiString path, AnsiString fn, TTreeNode* n,
                     TTreeView* tv, bool firstCall=false)
{
// rekursive Aufrufe erfolgen mit dem Default-Argument von firstCall.
// Beim ersten Aufruf in ButtonClick ist dieses Argument true.
if (firstCall)
  n=tv->Items->Add(tv->TopItem,path);
else
  n=tv->Items->AddChild(n,fn);
n->Data=new AnsiString(path+fn);
WIN32_FIND_DATA FindFileData;
AnsiString pattern=path+fn+"\\*"; // for directories always "*"
HANDLE h=FindFirstFile(pattern.c_str(),&FindFileData);
long long totalFileSize=0;
if (h!=INVALID_HANDLE_VALUE)
  {
    int found=1;
    while (found)
```

```
      {
        if ((FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)!=0)
          { // Unterverzeichnis
            if ( (strcmp(FindFileData.cFileName,".")==0) ||
                 (strcmp(FindFileData.cFileName,"..")==0) )
              ; // ignore "." and ".."
            else // Verzeichnis, rekursiv durchsuchen

totalFileSize+=SearchSubdirs(path+fn+"\\",FindFileData.cFileName,n,tv);
          }
        else // Datei gefunden
          {
            LARGE_INTEGER c1;
            c1.u.LowPart=FindFileData.nFileSizeLow;
            c1.u.HighPart=FindFileData.nFileSizeHigh;
            totalFileSize+=Occupied(c1.QuadPart);
          }
        found=FindNextFile(h,&FindFileData);
      }
  }
FindClose(h);
n->Text=n->Text+"  "+FloatToStrF(totalFileSize,ffNumber,18,0);
return totalFileSize;
};


void __fastcall TForm1::ShowDirClick(TObject *Sender)
{
Form1->TreeView1->Items->BeginUpdate();
TCursor CursorOld=Screen->Cursor;
Screen->Cursor=crHourGlass; // Sanduhr
AnsiString Laufwerk=AnsiString(DriveComboBox1->Drive)+":";
if (sumSubDir->Checked)
  SearchSubdirs(Laufwerk, "", Form1->TreeView1->TopItem,Form1->TreeView1, true);
else
  SearchSubdirs(Laufwerk,Form1->TreeView1->TopItem,Form1->TreeView1,true);

Screen->Cursor=CursorOld;
Form1->TreeView1->Items->EndUpdate();
}

void __fastcall TForm1::SortClick(TObject *Sender)
{
TreeView1->SortType=Listactns::stBoth;
TreeView1->AlphaSort();
}


//---------------------------------------------------------------------

void __fastcall TForm1::Occupied_Click(TObject *Sender)
{
use_occupied=Form1->Occupied_->Checked;
}

void __fastcall TForm1::SaveTreeViewClick(TObject *Sender)
{
SaveDialog1->FileName="c:\\tv.txt";
if (SaveDialog1->Execute());
  TreeView1->SaveToFile(SaveDialog1->FileName);
}

void __fastcall TForm1::ClearTreeClick(TObject *Sender)
{
TreeView1->Items->Clear();
}

//---------------------------------------------------------------------
```

```
//---------------------------------------------------------------------
/*
int ColumnToSort = 0; // class member would be preferable

void __fastcall TForm1::ListView1Compare(TObject *Sender, TListItem *Item1,
      TListItem *Item2, int Data, int &Compare)
{
if (ColumnToSort == 0)
    Compare = CompareText(Item1->Caption,Item2->Caption);
else if (ColumnToSort == 1)
  {
    int ix = ColumnToSort - 1;
    if     (StrToInt(Item1->SubItems->Strings[ix])<     StrToInt(Item2->SubItems-
>Strings[ix]))
      Compare = -1;
    else  if  (StrToInt(Item1->SubItems->Strings[ix])>  StrToInt(Item2->SubItems-
>Strings[ix]))
      Compare = 1;
    else
      Compare = 0;
  }
else
  {
    int ix = ColumnToSort - 1;
    if        (StrToDateTime(Item1->SubItems->Strings[ix]) < StrToDateTime(Item2-
>SubItems->Strings[ix]))
      Compare = -1;
    else if (StrToDateTime(Item1->SubItems->Strings[ix]) > StrToDateTime(Item2-
>SubItems->Strings[ix]))
      Compare = 1;
    else
      Compare = 0;
  }
}
*/
//int __stdcall MyCustomSort(TListItem *Item1,
//      TListItem *Item2, long ColumnToSort)

int compareAsInt(const AnsiString& s1, const AnsiString& s2)
{
if (StrToInt(s1)< StrToInt(s2))
  return -1;
else if (StrToInt(s1)> StrToInt(s2))
  return 1;
else
  return 0;
}

int compareAsDate(const AnsiString& s1, const AnsiString& s2)
{
if (StrToDateTime(s1)< StrToDateTime(s2))
  return -1;
else if (StrToDateTime(s1)> StrToDateTime(s2))
  return 1;
else
  return 0;
}

int __stdcall MyCustomSort(long It1,
      long It2, long ColumnToSort)
{
TListItem* Item1=(TListItem*)It1; // Typecast - nicht schön
TListItem* Item2=(TListItem*)It2; // Typecast - nicht schön
int ix = ColumnToSort - 1;
if (ColumnToSort == 0)
  return CompareText(Item1->Caption,Item2->Caption);
else if (ColumnToSort == 1)
```

```
  return compareAsInt(Item1->SubItems->Strings[ix],Item2->SubItems->Strings[ix]);
    /*
    if     (StrToInt(Item1->SubItems->Strings[ix])<     StrToInt(Item2->SubItems-
>Strings[ix]))
      return -1;
    else  if  (StrToInt(Item1->SubItems->Strings[ix])>  StrToInt(Item2->SubItems-
>Strings[ix]))
      return 1;
    else
      return 0;
    */
else
  return              compareAsDate(Item1->SubItems->Strings[ix],Item2->SubItems-
>Strings[ix]);
    /*
    if       (StrToDateTime(Item1->SubItems->Strings[ix]) < StrToDateTime(Item2-
>SubItems->Strings[ix]))
      return -1;
    else if (StrToDateTime(Item1->SubItems->Strings[ix]) > StrToDateTime(Item2-
>SubItems->Strings[ix]))
      return 1;
    else
      return 0;
    */
}
/*
Call CustomSort to sort the items in the list using the ordering
function defined by the SortProc parameter. The SortProc parameter
specifies an ordering function that compares the list items passed
as lParam1 and lPartcustomam2. The ordering function returns an integer
that indicates whether lParam1 is the same as lParam2 (return value
is 0), lParam1 is greater than lParam2 (return value is greater
than 0), or lParam1 is less than lParam2 (return value is less
than 0). The lParam parameter of CustomSort is an optional value
that is passed as the third parameter to the ordering function.
If the SortProc parameter is nil (Delphi) or NULL (C++),
CustomSort compares the list items by generating an OnCompare
event. This allows the OnCompare event handler to provide
different ordering criteria (such as ascending vs. descending
order), based on the value of the lParam parameter.
If the ordering function is not provided and there is no
OnCompare event handler, CustomSort sorts items alphabetically
by their Caption values.
CustomSort returns true if the list is successfully sorted.
Warning:  CustomSort does not work if the application is running
in virtual mode.

*/

//int MyCustomSort(TListItem *Item1,
//      TListItem *Item2, int Data, int &Compare)
void __fastcall TForm1::ListView1ColumnClick(TObject *Sender,
      TListColumn *Column)
{
  //ColumnToSort = Column->Index;
  // ((TCustomListView *)Sender)->AlphaSort();
  ListView1->Items->BeginUpdate();
  ListView1->CustomSort(MyCustomSort,Column->Index);
  ListView1->Items->EndUpdate();
}
//---------------------------------------------------------------------
```

11.10.5 Aufgabe 10.5

```
// File: C:\Loesungen_CB2006\Kap_10\10.5\RichEdU.cpp

#include <vcl.h>
#pragma hdrstop
#include "RichEdU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"


//----- Aufgabe 10.5 ------------------------------------------------

TForm1 *Form1;
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
RichEdit1->ScrollBars=ssVertical;
}

void CreateRTF(TRichEdit* r)
{
r->Clear();
r->SelAttributes->Name= "Arial";
r->SelAttributes->Size=15;
r->SelAttributes->Style<<fsBold;
r->Lines->Add("Quadratzahlen\r\n");

for (int i=1;i<10;i++)
  {
    r->SelAttributes->Name= "Courier New";
    r->SelAttributes->Size=12;
    r->Lines->Add("i="+IntToStr(i)+" i*i="+IntToStr(i*i));
  }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
CreateRTF(RichEdit1);
}


void __fastcall TForm1::ffnen1Click(TObject *Sender)
{
OpenDialog1->Filter="RTF-Dateien|*.rtf";
if (OpenDialog1->Execute())
  {
    RichEdit1->Lines->LoadFromFile(OpenDialog1->FileName);
  }
}

void __fastcall TForm1::Speichern1Click(TObject *Sender)
{
SaveDialog1->FileName="c:\\test1.rtf";
if (OpenDialog1->Execute())
  {
    RichEdit1->Lines->SaveToFile(OpenDialog1->FileName);
  }
}

void DruckenMitRichEdit()
{
TRichEdit* r=new TRichEdit(Form1);
r->Parent=Form1; // siehe Abschnitt 8.4
r->Visible=false; // RichEdit nicht sichtbar
for (int i=0; i<10; i++)
  r->Lines->Add("Line "+IntToStr(i));
```

```
r->Print("Titel des Druck-Jobs");
delete r;
}

void __fastcall TForm1::btnPrintClick(TObject *Sender)
{
DruckenMitRichEdit();
}
```

11.10.6 Aufgabe 10.7

```
// File: C:\Loesungen_CB2006\Kap_10\10.7\AnalogRU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "AnalogRU.h"
//---------------------------------------------------------------------
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
b->OnChange=aChange;
d->OnChange=aChange;
}

//---------------------------------------------------------------------
void __fastcall TForm1::aChange(TObject *Sender)
{
if (b->Position*b->Position != a->Position*d->Position)
  {
    Label_x->Caption = FloatToStr((0.0L+b->Position - d->Position)/
                    (b->Position*b->Position - a->Position*d->Position));
    Label_y->Caption = FloatToStr((0.0L+b->Position - a->Position)/
                    (b->Position*b->Position - a->Position*d->Position));
    // Probe:
    Label_test1->Caption=FloatToStr(a->Position*StrToFloat(Label_x->Caption)+b-
>Position*StrToFloat(Label_y->Caption)-1);
    Label_test2->Caption=FloatToStr(b->Position*StrToFloat(Label_x->Caption)+d-
>Position*StrToFloat(Label_y->Caption)-1);
  }
else
  {
    Label_x->Caption = "Division durch 0";
    Label_y->Caption = "Division durch 0";
  }
Edit_a->Text = IntToStr(a->Position);
Edit_b->Text = IntToStr(b->Position);
Edit_c->Text = IntToStr(d->Position);
}
//---------------------------------------------------------------------

/*
  Die Anweisungen in der Funktion aChange kann man auch in die Funktion
  bChange und dChange übernehmen. Dann reicht in jeder dieser Funktionen
  eine einzige Zuweisung an Edita->Text, Editb->Text oder Editc->Text aus.

  Sobald Funktionen behandelt sind, liegt es nahe, diese in einer einzigen
  Funktion zusammenzufassen, die dann in aChange, bChange und dChange
  aufgerufen wird.

  Noch einfacher ist es, die Funktion aChange den Ereignissen OnChange
```

```
  der Scrollbars b und c zuzuordnen. Dazu kann man sie im Objektinspektor
  beim entsprechenden Ereignis im Pulldown-Menü auswählen.

  Die beiden Anweisungen in FormCreate haben denselben Effekt. Diese Technik
  wird aber erst in Kapitel 9.4 vorgestellt.
*/



void __fastcall TForm1::GroupBox2Click(TObject *Sender)
{
static int i=0;
i++;
if (i%2==1)
  {
    GroupBox1->Caption="Lösung";
    GroupBox2->Caption="Probe";
    GroupBox3->Caption=  "Gleichungen";
    Form1->Caption="Lineares symmetrisches Gleichungssystem";
  }
else
  {
    GroupBox1->Caption="Solution";
    GroupBox2->Caption="Test";
    GroupBox3->Caption=  "Equations";
    Form1->Caption="Linear Symmetric System of Equations";
  }
}
```

11.10.7 Aufgabe 10.10

```
// File: C:\Loesungen_CB2006\Kap_10\10.10\TimerU.cpp

#include <vcl.h>
#pragma hdrstop
#include "TimerU.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//----- Aufgabe 10.10 ------------------------------------------------

//----- Aufgabe 10.10.1 ----------------------------------------------

#include "..\..\CppUtils\TimeUtils.h"

TDateTime start;
double hr_start;
__fastcall TForm1::TForm1(TComponent* Owner)
   : TForm(Owner)
{
Timer1->Enabled=false;
Timer1->Interval=300;
Timer2->Interval=100;
start=Now();
hr_start=HRTimeInSec();
}


// a)
bool c=true;
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
```

```
if (c)
  Edit1->Color=clYellow;
else
  Edit1->Color=clWindow;
c=!c;
}

void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
Timer1->Enabled=true;
}

void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
Timer1->Enabled=false;
Edit1->Color=clWindow;
}


// b)
int c1=0;
int c2=0;

void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
c1=c1+1000/Timer2->Interval;
}

#include "..\..\CppUtils\TimeUtils.h"

void __fastcall TForm1::Timer3Timer(TObject *Sender)
{
c2=c2+1000/Timer3->Interval;
Memo1->Lines->Add("c1="+IntToStr(c1)+" c2="+IntToStr(c2)+
" t="+TimeToStr(Now()-start)+
" hr_t="+FloatToStr(HRTimeInSec()-hr_start));
}

void __fastcall TForm1::btnStartStopClick(TObject *Sender)
{
Timer2->Enabled=!Timer2->Enabled;
Timer3->Enabled=!Timer3->Enabled;
}
```

11.10.8 Aufgabe 10.11

```
// File: C:\Loesungen_CB2006\Kap_10\10.11\ThreadsU.cpp

#include <vcl.h>
#pragma hdrstop

#include "ThreadsU.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
#include "ThreadsU1.h"

void __fastcall TForm1::btnNoThreadClick(TObject *Sender)
{
//int rek_Fib(int n); // in Threads1.cpp
Memo1->Lines->Add(IntToStr(rek_Fib(45)));
}

void __fastcall TForm1::btnThread1Click(TObject *Sender)
{

MyThread1* t=new MyThread1(true);
t->testcase=1;

t->FreeOnTerminate=true;
// t->Priority=tpLower;
t->Resume(); // nicht t->Execute(); aufrufen
}

void __fastcall TForm1::btnThrNoSyncClick(TObject *Sender)
{
MyThread1* t1=new MyThread1(true);
t1->testcase=2;
t1->FreeOnTerminate=true;
t1->Resume();

MyThread1* t2=new MyThread1(true);
t2->testcase=2;
t2->FreeOnTerminate=true;
t2->Resume();
}
//---------------------------------------------------------------------

void __fastcall TForm1::btnThrSyncClick(TObject *Sender)
{
MyThread1* t1=new MyThread1(true);
t1->testcase=3;
t1->FreeOnTerminate=true;
t1->Resume();

MyThread1* t2=new MyThread1(true);
t2->testcase=3;
t2->FreeOnTerminate=true;
t2->Resume();
}



// File: C:\Loesungen_CB2006\Kap_10\10.11\ThreadsU1.h

#ifndef Threads_CB2007U1H
#define Threads_CB2007U1H
//---------------------------------------------------------------------
#include <Classes.hpp>
//---------------------------------------------------------------------
class MyThread1 : public TThread
{
  typedef struct tagTHREADNAME_INFO
  {
    DWORD dwType;     // must be 0x1000
    LPCSTR szName;    // pointer to name (in user addr space)
    DWORD dwThreadID; // thread ID (-1=caller thread)
    DWORD dwFlags;    // reserved for future use, must be zero
  } THREADNAME_INFO;
private:
  void SetName();
  void __fastcall VCLAccess();
  void __fastcall ShowMyExceptionMessage();
```

```
protected:
    void __fastcall Execute();
public:
    __fastcall MyThread1(bool CreateSuspended);
  int testcase; // very primitive parameter passing to execute
};

int rek_Fib(int n);

//---------------------------------------------------------------------
#endif


// File: C:\Loesungen_CB2006\Kap_10\10.11\ThreadsU1.cpp

#include <vcl.h>
#pragma hdrstop

#include "ThreadsU1.h"
#pragma package(smart_init)
//---------------------------------------------------------------------

//   Important: Methods and properties of objects in VCL can only be
//   used in a method called using Synchronize, for example:
//
//      Synchronize(&UpdateCaption);
//
//   where UpdateCaption could look like:
//
//      void __fastcall MyThread1::UpdateCaption()
//      {
//        Form1->Caption = "Updated in a thread";
//      }
//---------------------------------------------------------------------

__fastcall MyThread1::MyThread1(bool CreateSuspended)
    : TThread(CreateSuspended)
{
}
//---------------------------------------------------------------------
void MyThread1::SetName()
{
    THREADNAME_INFO info;
    info.dwType = 0x1000;
    info.szName = "MyThread1";
    info.dwThreadID = -1;
    info.dwFlags = 0;

    __try
    {
        RaiseException(  0x406D1388,  0,  sizeof(info)/sizeof(DWORD),(DWORD*)&info
);
    }
    __except (EXCEPTION_CONTINUE_EXECUTION)
    {
    }
}
//---------------------------------------------------------------------
int rek_Fib(int n)
{ // Fibonacci-Zahlen rekursiv berechnen
if (n<=0) return 0;
else if (n==1) return 1;
else return rek_Fib(n-1) + rek_Fib(n-2);
}

int result=0;


#include "ThreadsU.h"
```

```
#include <SyncObjs.hpp>

int g=-1;

int f_no_sync()
{
int j=0;
for (int i=0; i<1000;i++)
  {
    g++;
    Sleep(1); // warte 1 Millisekunde
    if (g != 0)
      j++;
    g--;
  }
return j;
}

TCriticalSection* lock1=new TCriticalSection;
TCriticalSection* lock2=new TCriticalSection;

int ThreadFunction2()
{
throw Exception("hiohohoh");
int j=0;
for (int i=0; i<1000;i++)
  {
    lock1->Acquire();
      g++;
    lock1->Release();
    Sleep(1); // setzt die Ausführung für 1 Millisekunde aus
    lock1->Acquire();
      if (g != 0)
        j++;
      g--;
    lock1->Release();
  }
return j;
}

int j=0;

void __fastcall MyThread1::VCLAccess()
{
Form1->Memo1->Lines->Add(IntToStr(j));
}

AnsiString ExceptionMsg;

void __fastcall MyThread1::ShowMyExceptionMessage()
{
Form1->Memo1->Lines->Add(ExceptionMsg);
}

void __fastcall MyThread1::Execute()
{
try {
    SetName();
        //---- Place thread code here ----
    if (testcase==1)
      {
        j=rek_Fib(45);
        Synchronize(&VCLAccess);
      }
    else if (testcase==2)
      {
        j=f_no_sync();
        Synchronize(&VCLAccess);
      }
```

```
    else if (testcase==3)
      {
        lock2->Acquire();
        try {
              j=ThreadFunction2();
            }
        __finally
            {
              lock2->Release();
            }
        Synchronize(&VCLAccess);
      }
    else
      Form1->Memo1->Lines->Add("invalid testcase="+IntToStr(testcase));

}
catch (Exception& e)
{
   ExceptionMsg=e.Message;
   Synchronize(&ShowMyExceptionMessage);
} catch (...) {
   ExceptionMsg="Exception ...";
   Synchronize(&ShowMyExceptionMessage);
}
}
//---------------------------------------------------------------------
```

11.10.9 Aufgabe 10.13

```
// File: C:\Loesungen_CB2006\Kap_10\10.13\GraphicsU.cpp

#include <vcl.h>
#pragma hdrstop
#include "GraphicsU.h"
//---------------------------------------------------------------------
#pragma resource "*.dfm"
TForm1 *Form1;


__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

// --------------- Aufgaben 10.13 --------------------------

// --------------- Aufgabe 1 -------------------------------

void PrintMemo(TMemo* Memo)
{
if (Form1->FontDialog1->Execute())
  Printer()->Canvas->Font->Assign(Form1->FontDialog1->Font);
Printer()->BeginDoc(); // Initialisiert Druckauftrag
int y=0;     // y-Position der Zeile
int ZA=10;   // Zeilenabstand
int LR=10;   // linker Rand
int Seite=0; // aktuelle Seitenzahl
 for (int i=0; i<Memo->Lines->Count;i++)
  {
     AnsiString z=Memo->Lines->Strings[i];
     if ((y+Printer()->Canvas->TextHeight(z) >
            Printer()->PageHeight) || (Seite==0))
       {
```

```
         if (Seite>0) Printer()->NewPage();
         Seite++;
         AnsiString Kopfzeile="Kopfzeile    Seite: "+IntToStr(Seite);
         Printer()->Canvas->TextOut(LR,0,Kopfzeile);
         y=100; // Abstand zum Text
       }
     Printer()->Canvas->TextOut(LR,y,z);
     y=y+ZA+Printer()->Canvas->TextHeight(z);
  }
Printer()->EndDoc(); // Beendet Druckauftrag
}

void FillMemo(TMemo* Memo, int n)
{ // only for testing
Memo->Clear();
for (int i=0; i<n; i++)
  {
    Memo->Lines->Add(IntToStr(i)+". Line");
}
}

// #include <vcl\printers.hpp>

void __fastcall TForm1::DruckeMemoClick(TObject *Sender)
{
FillMemo(Memo1, 57);
PrintMemo(Memo1);
}

// --------------- Aufgabe 2 -------------------------------


// Die Lösung der Aufgaben a) bis c) ist in der Datei GraphUtils.cpp
// enthalten:
#include "..\..\CppUtils\GraphUtils.h";

// --------------- Aufgabe 2 d) ----------------------------

void zeichneFunktion(TCanvas* Canvas, int func,double x0, double y0, double x1,
double y1)
{
if      (func==1) Form1->Caption="y=sin(x*x)";
else if (func==2) Form1->Caption="y= exp(x)";
else if (func==3) Form1->Caption="y= x*x";
else if (func==4) Form1->Caption="y= 1/(1+x*x)";
else if (func==5) Form1->Caption="y= x*sin(x)";

Canvas->Pen->Color = clRed;
for (int px=0; px<Canvas->ClipRect.Width(); px++)
  {
    // transformiere px in Welt-Koordinaten
    double x=x_Welt(px, x0, x1, Canvas->ClipRect.Width());
    using namespace std;
    double y;
    if      (func==1) y= sin(x*x);
    else if (func==2) y= exp(x);
    else if (func==3) y= x*x;
    else if (func==4) y= 1/(1+x*x);
    else if (func==5) y= x*sin(x);

    // transformiere y in Bildkoordinaten
    int py=y_Bildschirm(y, y0, y1, Canvas->ClipRect.Height());
    if (px == 0) Canvas->MoveTo(px,py);
    else         Canvas->LineTo(px,py);
  }
}

int func=0;
void __fastcall TForm1::Function1Click(TObject *Sender)
```

```
{
Clear(Image1->Canvas);
func++;
if (func >5) func=1;
double x0, y0, x1, y1, dx, dy;
if      (func==1) { x0=-4; y0=-1; x1=4; y1=1;   dx=1; dy=0.2; }
else if (func==2) { x0=-1; y0=0;  x1=6; y1=100; dx=1; dy=10; }
else if (func==3) { x0=-2; y0=-2; x1=2; y1=4;   dx=1; dy=1; }
else if (func==4) { x0= 0; y0=0;  x1=4; y1=1;   dx=1; dy=0.2; }
else if (func==5) { x0=-2; y0=-6; x1=8; y1=10;  dx=1; dy=1; }


zeichneGitternetz(Form1->Image1->Canvas, x0,x1,dx, Form1->Image1->ClientWidth,
           y0,y1,dy, Form1->Image1->ClientHeight);

zeichneFunktion(Form1->Image1->Canvas,func,x0, y0, x1, y1);
}


double f1(double x)
{
return std::sin(x*x);
}

double f2(double x)
{
return std::exp(x);
}

double f3(double x)
{
return x*x;
}

double f4(double x)
{
return 1/(1+x*x);
}

double f5(double x)
{
return x*std::sin(x);
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
Clear(Image1->Canvas);
func++;
if (func >5) func=1;
double (*f)(double);
double x0, x1;
if      (func==1) { x0=-4; x1=4; f=f1;}
else if (func==2) { x0=-1; x1=6; f=f2;}
else if (func==3) { x0=-2; x1=2; f=f3;}
else if (func==4) { x0= 0; x1=4; f=f4;}
else if (func==5) { x0=-2; x1=8; f=f5;}

zeichneFunktion(Form1->Image1->Canvas,f,x0, x1);
}

void __fastcall TForm1::ClearImagClick(TObject *Sender)
{
Clear(Image1->Canvas);
}

// --------------- Aufgabe 3 -------------------------------

/*
Wie die Aufgabe mit der Fakultät schon gezeigt hat, wachsen die Fakultäten
```

```
sehr schnell an und liegen bereits ab n=13 außerhalb des mit 32-bit
darstellbaren Ganzzahlbereichs. Deshalb ist die folgende Funktion nur
für relativ kleine n einsetzbar:
*/

// Eine genauere Betrachtung der Formel für bin(n,k) zeigt, dass sich zwei
// Faktoren im Zähler und im Nenner immer wegkürzen. Damit erhält man mit
// auch für größere Werte von n und k korrekte Ergebnisse. Allerdings erhält
// man auch mit dieser Version ab n=30 negative Werte. Verwendet man hier
// Gleitkommazahlen, erhält man auch für große Werte von n und k richtige
// Ergebnisse:


void InitCanvas(TCanvas* Canvas)
{
Canvas->Font->Name="Courier";
Canvas->Font->Name="Diese Schrift gibt es nicht";
if (Form1->FontDialog1->Execute())
Form1->Canvas->Font->Assign(Form1->FontDialog1->Font);
// nicht: Image1->Canvas->Font=FontDialog1->Font;
//           da nur die Zeiger zugewiesen werden
};

double binom(int n, int k)
{
double result = 1;
for (int i=1; i<=k; i++)
  result = result * (n-i+1) / i;
return result;
};

/*
Da die Summe binom(n,0)+binom(n,1)+ ... binom(n,n) immer den Wert 2^n
ergeben muss, kann man mit der Funktion binom_test
prüfen, ob die Ergebnisse dieser Funktion plausibl sind.  Ein Test zeigt,
dass man auch für n=10000 richtige Werte erhält.
*/

double binom_test(int n)
{ // Die Summe mus "2 hoch n" sein
long double result=0;
for (int i=0; i<=n; i++)
  result=result+binom(n,i);
return result;
}

void __fastcall TForm1::BinomKoeffClick(TObject *Sender)
{
// Zum Testen:
// for (int i=5000; i<5020; i++)
//     Memo1->Lines->Add(FloatToStr(binom_test(i))+"   "+FloatToStr(powl(2.0,i))+"
"+FloatToStr(Potenz(2.0,i)));

AnsiString s;
for (int n=200; n<210; n++)
  {
    s = "";
    for (int k=0; k<n; k++)
      {
        s=s+FloatToStr(binom(n,k))+" ";
        Memo1->Lines->Add(s);
      };
  }
}

int round(double x)
{
return x+0.5;
}
```

```
void zeichneBinomialverteilung(TCanvas* canvas, int n)
{
Clear(canvas);
canvas->TextOut(10,10,"Binomialverteilung, n="+IntToStr(n));
canvas->Pen->Color = clBlack;
int w = canvas->ClipRect.Width()-20;
int h = canvas->ClipRect.Height()-20;
double max = binom(n,n/2);
for (int i=0; i<=n; i++)
  {
    int x1=i*w /(n+1.0);
    int y1=h;
    int x2=(i+1)*w /(n+1.0);
    int y2=round(h-(h)*(binom(n,i)/max));
    if (y1==y2)
      { // Rechtecke der Höhe 0 als Linie zeichnen
        canvas->MoveTo(x1,y1);
        canvas->LineTo(x2,y1);
      }
    else
      canvas->Rectangle(x1,y1, x2,y2);
  }
}

void __fastcall TForm1::HistogrammClick(TObject *Sender)
{
zeichneBinomialverteilung(Image1->Canvas, 25);
};

// --------------- Aufgabe 5 -------------------------------

void zeichne_RaeuberBeute(TCanvas* Canvas)
{
Clear(Canvas);
const double zb = 0.05,  fb = 0.001,
             sr = 0.05,  fr = 2*fb;

// Startwerte:
double r_alt = 100;
double b_alt = 50;

// zur Skalierung der y-Werte
double y0 = -10;
double y1 = 300;

int pr, pb;

for (int px=0; px<Canvas->ClipRect.Width(); px++)
  {
    double r = r_alt - sr*r_alt + fr*r_alt*b_alt;
    double b = b_alt + zb*b_alt - fb*r_alt*b_alt;
//    Form1->Memo1->Lines->Add(Format("r=%.4g b=%.4g",OPENARRAY(TVarRec,(r,b))));

    int pr_alt = pr; // undefinierte Werte beim
    int pb_alt = pb; // ersten Durchlauf irrelevant
    // transformiere y in Bild-Koordinaten
    pr=y_Bildschirm(r, y0, y1, Canvas->ClipRect.Height());
    pb=y_Bildschirm(b, y0, y1, Canvas->ClipRect.Height());

    Canvas->MoveTo(px-1,pr_alt);
    Canvas->Pen->Color = clRed;
    if (px==0) Canvas->MoveTo(px,pr);
    else Canvas->LineTo(px,pr);

    Canvas->MoveTo(px-1,pb_alt);
    Canvas->Pen->Color = clBlue;
    if (px==0) Canvas->MoveTo(px,pb);
    else Canvas->LineTo(px,pb);
```

```
    Canvas->MoveTo(pr_alt,pb_alt);
    Canvas->Pen->Color = clGreen;
    if (px==0) Canvas->MoveTo(pr,pb);
    else Canvas->LineTo(pr,pb);
    r_alt = r;
    b_alt = b;
  };
}

void __fastcall TForm1::RaeuberBeuteClick(TObject *Sender)
{
Form1->Caption = "Räuber-Beute-Modell";
zeichne_RaeuberBeute(Form1->Image1->Canvas);
}
```

11.10.10 Aufgabe 10.13.4 (Fraktale)

```
// File: C:\Loesungen_CB2006\Kap_10\10.13\FraktalU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "FraktalU.h"
//---------------------------------------------------------------------
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------

#include <float.h> // for  _control87


//----- Aufgabe 10.13.4 ---------------------------------------------

// a)

double  x0= -2;    // links oben
double  y0= 1.25;

double  x1 = 0.5; // rechts unten
double  y1 = -1.25;

// c)
enum  {Mandelbrot, Julia} Art_des_Fraktals=Mandelbrot;

long double jx= -0.745, jy= 0.1125; // für die Julia-Mengen

bool aborted=false; // b) Damit die Schleife unterbrochen werden kann

TColor IndexColor(int i)
{
i=i%256; // maximal 255 Farben
int r30=i%30;    // i->[0..29]
int t=255-5*r30; // [0..29]->[255..145]
if (i<30)       return RGB(255-8*i,255,255); // weiss..türkis
else if (i<60)  return RGB(0,255-8*r30,255); // türkis..blau
else if (i<90)  return RGB(0,0,t); // blau .. dunkelblau
else if (i<120) return RGB(0,t,0); // grün .. dunkelgrün
else if (i<150) return RGB(t,0,0); // rot .. dunkelrot
else if (i<180) return RGB(t,t,0);
else if (i<210) return RGB(t,0,t);
else if (i<240) return RGB(0,t,t);
else            return RGB(t,t,t); // Graustufen
}
```

```
#include "..\..\CppUtils\GraphUtils.h"

void zeichneFraktal(TImage* Image)
{
const int max_it =  200; // 50,100,200
aborted = false; // für b)
for (int px=0; (px <= Image->Width-1) && (!aborted); px++)
  {
    for (int py=0;(py<=Image->Height-1) && (!aborted); py++)
       {
          double x = x_Welt(px,x0,x1,Image->Width);
          double y = y_Welt(py,y1,y0,Image->Height);

          double x_it_alt = x;
          double y_it_alt = y;
          double dist;
          int i = 0;
          do {
               i++;
/*
  Eigentlich müssen diese Berechnungen durchgeführt werden:
               x_it = x_it_alt*x_it_alt - y_it_alt*y_it_alt + x;
               y_it = 2*x_it_alt*y_it_alt + y;
               dist = x_it*x_it+y_it*y_it;
  Die folgenden Operationen sparen aber einige Multiplikationen:
*/
               double sqr_x = x_it_alt*x_it_alt;
               double sqr_y = y_it_alt*y_it_alt;
               double pr_xy = x_it_alt*y_it_alt;
               double x_it,y_it;
               if (Art_des_Fraktals==Mandelbrot)
                 {
                   x_it = sqr_x - sqr_y + x;
                   y_it = pr_xy + pr_xy + y;
                  }
// c)
               else if (Art_des_Fraktals==Julia)
                 {
                   x_it = sqr_x - sqr_y + jx;
                   y_it = pr_xy + pr_xy + jy;
                  }
               dist = sqr_x + sqr_y;
               x_it_alt = x_it;
               y_it_alt = y_it;
             }
          while ((i <= max_it) && (dist <= 4));
          TColor color=IndexColor(i);
          if (i > max_it) color = clBlack;
          Image->Canvas->Pixels[px][py] = color;
        }
     // Damit ein eventuelles Anklicken des Buttons "Abbruch" erkannt wird
     // und die Pixel gezeichnet werden.
     if (px%10==0) Application->ProcessMessages();
   }
}

void __fastcall TForm1::ZeichneClick(TObject *Sender)
{
Form1->Caption="Fraktal: x0="+FloatToStr(x0)+" y0="+FloatToStr(y0)+
                       " x1="+FloatToStr(x1)+" y1="+FloatToStr(y1);
zeichneFraktal(Image1);
}


void __fastcall TForm1::AbbruchClick(TObject *Sender)
{
// Da das Zeichnen eines Fraktals relativ lange dauern kann,
// kann man es durch ein Anklicken dieses Buttons abbrechen.
```

```
aborted=true;
}

// b)
double x_start, y_start; // Position bei MouseDown

void __fastcall TForm1::Image1MouseDown(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y)
{
x_start = x_Welt(X,x0,x1,Image1->Width);
y_start = y_Welt(Y,y1,y0,Image1->Height);
}

void __fastcall TForm1::Image1MouseUp(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y)
{
x1 = x_Welt(X,x0,x1,Image1->Width);
y1 = y_Welt(Y,y1,y0,Image1->Height);

x0 = x_start;
y0 = y_start;
}


// d)
struct TMandelbrotParam {
  char* Name;
  double x0; double y0;
  double x1; double y1;
};

const int nMBAW=6;
TMandelbrotParam MBPar[nMBAW]=
  {
    {"Apfelmännchen",       -2,     1.25,   0.5,  -1.25},
    {"Seepferdchen",        -0.88,  0.18,  -0.70,  0.0},
    {"Seepferdchen",       -0.73206,0.1570,-0.73196,0.1569},
    {"Apfelmännchen unten", -0.17, -1.02,  -0.15, -1.04},
    {"Apfelmännchen links",-1.80,   0.03,  -1.74, -0.03},
    {"Spirale",            -0.750,  0.105, -0.740, 0.095}
  };

struct TJuliaParam {
  char* Name;
  double x0; double y0;
  double x1; double y1;
  double jx; double jy;
};

const int nJAW=9;
TJuliaParam JPar[nJAW]=
  { // Bezeichnungen teilweise nach Peitgen u. a. (1986)
    { "Peitgen Fig. 15", -1.6, -1.2, 1.6, 1.2, -0.74543, 0.11301},
    { "Peitgen Map 18",   -1, -1.2,   1, 1.2,   0.32, 0.043},
    { "Peitgen Map 20",   -2, -1.5,   2, 1.5,  -0.12375, 0.56508},
    { "Siegel Disk", -2, -1.5,   2, 1.5,  -0.39054, -0.58679},
    { "Fatou Staub", -2, -1.5,   2, 1.5,  0.11031, -0.67037},
    { "Zweig",   -2, -1.5,   2, 1.5,  0, 1},
    { "Zweig mit Perlen",   -2, -1.5,   2, 1.5,  -0.15652, 1.03225},
    { "Julia 6",   -2, -1.5,   2, 1.5,  -0.72, 0.53},
    { "Julia 7",   -2, -1.5,   2, 1.5,  -1.776, 0.4},
  };

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
// Complex operations sometimes perform floating point operations
// that throw exceptions. With this call all exceptions from the
// FPU are suppressed. See "float.h"
```

```
_control87(MCW_EM, MCW_EM);
for (int i=0; i< nMBAW; i++)
  ListBox1->Items->Add(MBPar[i].Name);
for (int i=0; i< nJAW; i++)
  ListBox2->Items->Add(JPar[i].Name);

Label1->AutoSize=false;
Label1->WordWrap=true;
Label1->Caption="In der Zeichnung kann ein rechteckiger Bereich mit "
                "der Maus markiert werden. Die linken obere Ecke erhält "
                "man beim Drücken der Maustaste und die recht untere "
                "beim Loslassen.";
}

void __fastcall TForm1::ListBox1Click(TObject *Sender)
{
Art_des_Fraktals=Mandelbrot;
int i=ListBox1->ItemIndex;
if (i<0) i=0;
x0 = MBPar[i].x0;
y0 = MBPar[i].y0; // links oben
x1 = MBPar[i].x1;
y1 = MBPar[i].y1; // rechts unten
}

void __fastcall TForm1::ListBox2Click(TObject *Sender)
{
Art_des_Fraktals=Julia;
int i=ListBox2->ItemIndex;
if (i<0) i=0;
x0 = JPar[i].x0;
y0 = JPar[i].y0;
x1 = JPar[i].x1;
y1 = JPar[i].y1;
jx = JPar[i].jx;
jy = JPar[i].jy;
}
```

11.10.11 Word Beispiel 10.14

```
// File: C:\Loesungen_CB2006\Kap_10\10.14\WordU.cpp

#include <vcl.h>
#pragma hdrstop

#include "WordU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "Word_XP_srvr"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm1::btnStartWordClick(TObject *Sender)
{
try {
  WordApplication1->Connect();
  WordApplication1->set_Visible(true);
  /// C++Builder 5: WordApplication1->Visible=true;
```

```
  WordApplication1->Documents->Add(TNoParam(),TNoParam(),
  TNoParam(),TNoParam()); // neues Dokument
  /// C++Builder 6: WordApplication1->Documents->Add();
  WordApplication1->Selection->Font->Bold=true;
  WordApplication1->Selection->Font->Size=15;
  WordApplication1->Selection->TypeText(
  StringToOleStr("Fette Überschrift mit 15 Punkt"));
  WordApplication1->Selection->TypeParagraph();
  WordApplication1->Selection->TypeParagraph();
  WordApplication1->Selection->Font->Bold=false;
  WordApplication1->Selection->Font->Size=10;
  WordApplication1->Selection->TypeText(
  StringToOleStr("Nicht fetter Text mit 10 Punkt."));
  WordApplication1->Selection->TypeParagraph();
  WordApplication1->ChangeFileOpenDirectory(
  StringToOleStr("C:\\test"));
  WordApplication1->ActiveDocument->SaveAs(OleVariant(
  StringToOleStr("test.doc")));
  WordApplication1->PrintOut();
}
catch (Exception& e)
{
  ShowMessage("Word error: "+e.Message );
  WordApplication1->Disconnect();
}
}
//---------------------------------------------------------------------
```

11.10.12 Aufgabe 10.15

```
// File: C:\Loesungen_CB2006\Kap_10\10.15\DBU.cpp

#include <vcl.h>
#pragma hdrstop

#include "DBU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;

const enum {BDE_DB, ADO_DB } DB_Type=ADO_DB;

void Init_BDE_Table(TTableType t)
{ // only to illustrate a connection to BDE databases
Form1->Table1->TableType = t; // or ttDBase usw.
if (t==ttDefault || t==ttParadox||t==ttDBase)
  {
    Form1->Table1->DatabaseName = "c:\\test";
    Form1->Table1->TableName = "KB";

    Form1->Table2->DatabaseName = "c:\\test";
    Form1->Table2->TableName = "KS";
  }
else if (t==ttFoxPro||t==ttASCII)
  {
    Form1->Table1->DatabaseName = "c:\\test";
    Form1->Table1->TableName = "KB";

    Form1->Table2->DatabaseName = "c:\\test";
    Form1->Table2->TableName = "KS";
  }
else
```

```
  ShowMessage("Invalid TableType in Init_BDE_Table");
}


__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
if (DB_Type==BDE_DB)
  {
    // Erzeuge das Verzeichnis, falls es nicht existiert
    AnsiString dir="c:\\test";
    if (CreateDirectory(dir.c_str(),0))
      ShowMessage("directory '"+dir+"' created");
   Init_BDE_Table(ttParadox);
  }
}

//---------------------------------------------------------------------

const AnsiString fnKontoNr="KontoNr";
const AnsiString fnDatum="Datum";
const AnsiString fnBewart="Bewart";
const AnsiString fnBetrag="Betrag";

//---------------------------------------------------------------------

void SetConnectionString(TADOConnection* c)
{
/*
c->ConnectionString="Provider=SQLOLEDB.1;Integrated Security=SSPI;"
                    "Persist Security Info=False;Initial Catalog="
                    "SQLExpress_db1;Data Source=RI-2-8-GHZ-NEU\\SQLEXPRESS";
*/
}

void InitADOTable(TADOTable* ADOTable,TADOConnection* ADOConnection)
{
// SetConnectionString(ADOConnection);
ADOTable->Connection=ADOConnection;
ADOTable->Active=false; // TableName kann nur bei einer nicht
                        // aktiven Table gesetzt werden
ADOTable->TableName="Kontobew";
ADOTable->Active=true;
    //DataSource1->DataSet = ADOTable1;     // wird im OI angeboten
    //DBGrid1->DataSource = DataSource1; // wird im OI angeboten
    //DBNavigator1->DataSource=DataSource1;
}


void __fastcall TForm1::btnInitTableClick(TObject *Sender)
{
if (DB_Type==ADO_DB)
  {
    InitADOTable(ADOTable1,ADOConnection1);
  }
else if (DB_Type==BDE_DB)
  {
    Init_BDE_Table(ttParadox);
    //DataSource1->DataSet = Table1;     // wird im OI angeboten
    //DBGrid1->DataSource = DataSource1; // wird im OI angeboten
    Table1->Active=true;

    //DBNavigator1->DataSource=DataSource1;
  }
}

//---------------------------------------------------------------------

void fillKontobew(TDataSet* t, int n)
```

```
{
t->Active = true;
for (int i=0;i<n;i++)
 {
   t->Append();
   t->FieldByName(fnKontoNr)->AsInteger=1000+i;
   t->FieldByName(fnDatum)->AsDateTime=Date()+i;
   AnsiString PlusMinus[2]={"+","-"};
   t->FieldValues[fnBewart]=PlusMinus[i%2];
   t->FieldValues[fnBetrag]=1000-i/100.0;
   t->Post();
 }
}

void __fastcall TForm1::btnFillTableClick(TObject *Sender)
{
if (DB_Type==ADO_DB)
  {
    InitADOTable(ADOTable1,ADOConnection1);
    fillKontobew(ADOTable1, 50);
  }
else if (DB_Type==BDE_DB)
  {
    fillKontobew(Table1, 50);
  }
}
//---------------------------------------------------------------------

void showInMemo(TDataSet* t)
{
t->First();
while (!t->Eof)
  {
    AnsiString s=t->FieldByName("KontoNr")->AsString+" "+
                 t->FieldByName("Datum")->AsString+" "+
                 t->FieldByName("Betrag")->AsString+" ";
    Form1->Memo1->Lines->Add(s);
    t->Next();
  }
}

void __fastcall TForm1::btnShowInMemoClick(TObject *Sender)
{
if (DB_Type==ADO_DB)
  {
    showInMemo(ADOTable1);
  }
else if (DB_Type==BDE_DB)
  {
    showInMemo(ADOTable1);
  }
}
//---------------------------------------------------------------------

void initGrid_1(TDataSet* DataSet)
{
Form1->DataSource1->DataSet = DataSet; // Form1->ADOTable1;
Form1->DBGrid1->DataSource = Form1->DataSource1;
Form1->DBNavigator1->DataSource=Form1->DataSource1;
}

void __fastcall TForm1::btnInitGrid_1Click(TObject *Sender)
{
Memo1->Visible=false;
if (DB_Type==ADO_DB)
  {
    InitADOTable(ADOTable1,ADOConnection1);
    initGrid_1(Form1->ADOTable1);
  }
```

```
else if (DB_Type==BDE_DB)
  {
    Init_BDE_Table(ttParadox);
    Table1->Active=true;
    Form1->DataSource1->DataSet = Form1->Table1;
    Form1->DBGrid1->DataSource = Form1->DataSource1;
    Form1->DBNavigator1->DataSource=Form1->DataSource1;
  }
}
//---------------------------------------------------------------------

void simpleADOQuery(int test, TADOQuery* Query)
{
Query->Close(); // beim ersten Aufruf unnötig, aber kein Fehler
Query->SQL->Clear();
if (test==1)
  Query->SQL->Add("SELECT * FROM Kontobew ");
else if (test==2)
 {
  Query->SQL->Add("SELECT  *  FROM  Kontobew  WHERE  KontoNr  <  1020  and  Betrag  <
990");
  Query->SQL->Add("ORDER BY Kontobew.KontoNr,Kontobew.Datum");
 }
Query->Open();
}

void simpleBDEQuery(TQuery* Query)
{
Query->Close(); // beim ersten Aufruf unnötig, aber kein Fehler
Query->SQL->Clear();
Query->SQL->Add("SELECT * FROM 'c:\\test\\KB.db\' ");
Query->Open();
}

void __fastcall TForm1::btnSQL_1Click(TObject *Sender)
{
ADOQuery1->Connection=ADOConnection1;
simpleADOQuery(2,ADOQuery1);
showInMemo(ADOQuery1);
    initGrid_1(ADOQuery1);

}
//---------------------------------------------------------------------
```

11.10.13 Aufgabe 10.18

```
// File: C:\Loesungen_CB2006\Kap_10\10.18\SetU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "SetU.h"
#pragma resource "*.dfm"
TForm1 *Form1;


__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}


// --------------- Aufgabe 10.18 ---------------------------
```

```
void __fastcall TForm1::BoldClick(TObject *Sender)
{
if (Bold->Checked)
  Label1->Font->Style=Label1->Font->Style<<fsBold;
else
  Label1->Font->Style=Label1->Font->Style>>fsBold;
}

void __fastcall TForm1::ItalicClick(TObject *Sender)
{
if (Italic->Checked)
  Label1->Font->Style=Label1->Font->Style<<fsItalic;
else
  Label1->Font->Style=Label1->Font->Style>>fsItalic;
}

void __fastcall TForm1::UnderlineClick(TObject *Sender)
{
if (Underline->Checked)
  Label1->Font->Style=Label1->Font->Style<<fsUnderline;
else
  Label1->Font->Style=Label1->Font->Style>>fsUnderline;
}

void __fastcall TForm1::StrikeOutClick(TObject *Sender)
{
if (StrikeOut->Checked)
  Label1->Font->Style=Label1->Font->Style<<fsStrikeOut;
else
  Label1->Font->Style=Label1->Font->Style>>fsStrikeOut;
}

void __fastcall TForm1::Edit1KeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
AnsiString s;
// Diese Zustände werden sowohl bei Maus- als auch bei
// Tastaturereignissen angezeigt:
if (Shift.Contains(ssShift)) s=s+"Shift ";
if (Shift.Contains(ssAlt))   s=s+"Alt ";
if (Shift.Contains(ssCtrl))  s=s+"Ctrl ";
// Diese Zustände werden nur bei Tastaturereignissen
// angezeigt:
if (Shift.Contains(ssLeft))  s=s+"Left ";
if (Shift.Contains(ssRight)) s=s+"Right ";
if (Shift.Contains(ssMiddle))s=s+"Middle ";
if (Shift.Contains(ssDouble))s=s+"Double ";
Memo1->Lines->Add(s);
/*
ssShift   Die Taste Umschalt wird gedrückt.
ssAlt     Die Taste Alt wird gedrückt.
ssCtrl    Die Taste Strg wird gedrückt.
ssLeft    Die linke Maustaste wird gedrückt.
ssRight   Die rechte Maustaste wird gedrückt.
ssMiddle  Die mittlere Maustaste wird gedrückt.
ssDouble  Es wurde mit der Maus doppelgeklickt.
*/
}
```

11.10.14 Aufgabe 10.19

```
// File: C:\Loesungen_CB2006\Kap_10\10.19\InitOGL.cpp
```

```
HDC InitOpenGL(HWND Handle,bool UseDoubleBuffer)
{
PIXELFORMATDESCRIPTOR pfd;
memset(&pfd,0,sizeof(pfd)); // setze alle Felder auf 0
pfd.nSize      = sizeof(PIXELFORMATDESCRIPTOR);
pfd.nVersion   = 1; // Versionsnummer, 1 notwendig
pfd.dwFlags    =    // Eigenschaften des Pixelpuffers
    PFD_DRAW_TO_WINDOW | // Zeichne in ein Fenster
    PFD_SUPPORT_OPENGL;  // OpenGL anstelle von GDI
if (UseDoubleBuffer)// Doppelbuffer verwenden
  pfd.dwFlags |= PFD_DOUBLEBUFFER; // flag setzen
pfd.iPixelType = PFD_TYPE_RGBA; // RGBA Pixel
pfd.cColorBits = 24; // Farbtiefe 24 Bit
pfd.cDepthBits = 32; // Tiefenpuffer 32 Bits
pfd.iLayerType = PFD_MAIN_PLANE; // Einzige Möglichkeit

HDC dc = GetDC(Handle);
int iPF = ChoosePixelFormat(dc, &pfd);
if (iPF==0) ShowMessage("Fehler bei ChoosePixelFormat");

SetPixelFormat(dc, iPF, &pfd );
GLContext= wglCreateContext(dc);
if (GLContext) wglMakeCurrent(dc, GLContext);
else ShowMessage("wglMakeCurrent nicht erfolgreich");
return dc;
}



// File: C:\Loesungen_CB2006\Kap_10\10.19\Lights.cpp

#include <vcl.h>
#pragma hdrstop

#include "Lights.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------


#include <gl\gl.h>

void DrawScene();

//---------------------------------------------------------------------

void SetLightPosition(GLdouble x,GLdouble y,GLdouble z)
{
GLfloat position[] = {50-x, 50-y, 50-z, 1}; // w!=0: position, w==0: direction
glLightfv(GL_LIGHT0, GL_POSITION, position);
DrawScene();

Form2->LabAmbLightXPosValue->Caption=FloatToStr(position[0]);
Form2->LabAmbLightYPosValue->Caption=FloatToStr(position[1]);
Form2->LabAmbLightZPosValue->Caption=FloatToStr(position[2]);
}

void __fastcall TForm2::SBPositionXChange(TObject *Sender)
{
SetLightPosition(SBPositionX->Position,SBPositionY->Position,SBPositionZ-
>Position);
}
```

```
void __fastcall TForm2::SBPositionYChange(TObject *Sender)
{
SBPositionXChange(Sender); // same effect as
//      SetLightPosition(SBPositionX->Position,SBPositionY->Position,SBPositionZ-
>Position);
}

void __fastcall TForm2::SBPositionZChange(TObject *Sender)
{
SBPositionXChange(Sender);
}

//---------------------------------------------------------------------

void SetAmbientLight(GLdouble r,GLdouble g,GLdouble b)
{
GLfloat light[] = {r/100,g/100,b/100, 1};
glLightfv(GL_LIGHT0, GL_AMBIENT, light);
DrawScene();

Form2->LabAmbLightRedValue->Caption=FloatToStr(light[0]);
Form2->LabAmbLightGreenValue->Caption=FloatToStr(light[1]);
Form2->LabAmbLightBlueValue->Caption=FloatToStr(light[2]);
}

void __fastcall TForm2::SBAmbLightRedChange(TObject *Sender)
{
SetAmbientLight(SBAmbLightRed->Position,SBAmbLightGreen->Position,SBAmbLightBlue-
>Position);
}

void __fastcall TForm2::SBAmbLightGreenChange(TObject *Sender)
{
SBAmbLightRedChange(Sender);
}

void __fastcall TForm2::SBAmbLightBlueChange(TObject *Sender)
{
SBAmbLightRedChange(Sender);
}

//---------------------------------------------------------------------

void SetDiffuseAndSpecularLight(GLdouble r,GLdouble g,GLdouble b)
{
GLfloat light[] = {r/100,g/100,b/100, 1};
// Woo p. 182: For realistic effects, set specular
// and diffuse light to the same values.
glLightfv(GL_LIGHT0, GL_DIFFUSE,  light);
glLightfv(GL_LIGHT0, GL_SPECULAR, light);
DrawScene();

Form2->LabDiffSpecLightRedValue->Caption=FloatToStr(light[0]);
Form2->LabDiffSpecLightGreenValue->Caption=FloatToStr(light[1]);
Form2->LabDiffSpecLightBlueValue->Caption=FloatToStr(light[2]);
}

void __fastcall TForm2::SBDiffSpecLightRedChange(TObject *Sender)
{
SetDiffuseAndSpecularLight(SBDiffSpecLightRed->Position,SBDiffSpecLightGreen-
>Position,SBDiffSpecLightBlue->Position);
}

void __fastcall TForm2::SBDiffSpecLightGreenChange(TObject *Sender)
{
SBDiffSpecLightRedChange(Sender);
}
```

```
void __fastcall TForm2::SBDiffSpecLightBlueChange(TObject *Sender)
{
SBDiffSpecLightRedChange(Sender);
}

//---------------------------------------------------------------------

void SetAmbAndDiffuseMaterial(GLdouble r,GLdouble g,GLdouble b)
{
GLfloat mat[] = {r/100,g/100,b/100, 1};
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat);
DrawScene();

Form2->LabMatRed->Caption=FloatToStr(mat[0]);
Form2->LabMatGreen->Caption=FloatToStr(mat[1]);
Form2->LabMatBlue->Caption=FloatToStr(mat[2]);
}

void __fastcall TForm2::SBAmbDiffMatRedChange(TObject *Sender)
{
SetAmbAndDiffuseMaterial(SBAmbDiffMatRed->Position,SBAmbDiffMatGreen-
>Position,SBAmbDiffMatBlue->Position);
}

void __fastcall TForm2::SBAmbDiffMatGreenChange(TObject *Sender)
{
SBAmbDiffMatRedChange(Sender);
}

void __fastcall TForm2::SBAmbDiffMatBlueChange(TObject *Sender)
{
SBAmbDiffMatRedChange(Sender);
}

//---------------------------------------------------------------------

void SetSpecularMaterial(GLdouble r,GLdouble g,GLdouble b)
{
GLfloat mat[] = {r/100, g/100, b/100, 1};
glMaterialfv(GL_FRONT, GL_SPECULAR, mat);
DrawScene();

Form2->LabMatSpecRed->Caption=FloatToStr(mat[0]);
Form2->LabMatSpecGreen->Caption=FloatToStr(mat[1]);
Form2->LabMatSpecBlue->Caption=FloatToStr(mat[2]);
}

void __fastcall TForm2::SBSpecMatRedChange(TObject *Sender)
{
SetSpecularMaterial(SBSpecMatRed->Position,SBSpecMatGreen-
>Position,SBSpecMatBlue->Position);
}

void __fastcall TForm2::SBSpecMatGreenChange(TObject *Sender)
{
SBSpecMatRedChange(Sender);
}

void __fastcall TForm2::SBSpecMatBlueChange(TObject *Sender)
{
SBSpecMatRedChange(Sender);
}

void __fastcall TForm2::SBMatShininessChange(TObject *Sender)
{
GLfloat mat_shininess[] = {SBMatShininess->Position};
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
DrawScene();
Form2->LabMatShininess->Caption=FloatToStr(mat_shininess[0]);
```

```
}



// File: C:\Loesungen_CB2006\Kap_10\10.19\OpenGLU.cpp

#include <vcl.h>
#pragma hdrstop
#include "OpenGLU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------

#include <float.h> // for  _control87
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
// OpenGL functions sometimes perform floating point operations that
// throw exceptions. With this call all exceptions from the FPU
// are suppressed. See "float.h"
_control87(MCW_EM, MCW_EM);
}
//---------------------------------------------------------------------


void DrawScene();

HDC dc;
HGLRC GLContext;
#include "InitOGL.cpp";
const bool UseDoubleBuffer = true; // with false images may flicker

#include <gl/gl.h>
#include <gl/glu.h>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
dc=InitOpenGL(Form1->Handle,UseDoubleBuffer);
glEnable(GL_DEPTH_TEST);  // necessary for the depthbuffer
glClearColor(1, 1, 1, 1); // white background
// damit ein Betrachter im Ursprung das O
glMatrixMode(GL_MODELVIEW);
// verschiebt den Ursprung nach hinten,
glTranslated(0,0,-2);

// die folgenden Anweisungen sind nur für die Bitmaps
StatusBar1->Visible=false;
StatusBar1->SimplePanel=false;
StatusBar1->SimpleText="R: Rotate, T: Translate, up, down, left, right, shift-up,
shift-down";
Scene->Visible =true;
Lights->Visible =true;
glLineWidth(3);
}
//---------------------------------------------------------------------
void __fastcall TForm1::FormDestroy(TObject *Sender)
{
ReleaseDC(Handle,dc);
wglMakeCurrent(dc, NULL);
wglDeleteContext(GLContext);
}

void __fastcall TForm1::FormResize(TObject *Sender)
{ // FormResize is called after FormCreate
glViewport(0, 0, ClientWidth, ClientHeight);
glMatrixMode(GL_PROJECTION);
```

```
glLoadIdentity();

GLdouble aspect =(double)ClientWidth/ClientHeight;
// viewing angle 60°, aspect=Width/Height, für z<1 und z>10 ausblenden:
gluPerspective(60, aspect, 0.1, 100);

// make the modelview matrix the actual matrix
glMatrixMode(GL_MODELVIEW);
DrawScene();
}

void __fastcall TForm1::FormPaint(TObject *Sender)
{
FormResize(Sender);
}

//---------------------------------------------------------------------

GLfloat colRed[]       = {1, 0, 0},
        colGreen[]     = {0, 1, 0},
        colBlue[]      = {0, 0, 1},
        colYellow[]    = {1, 1, 0},
        colDarkGray[]  = {0.2, 0.2, 0.2},
        colGray[]      = {0.5, 0.5, 0.5},
        colLightGray[] = {0.8, 0.8, 0.8},
        // colCyan[]       = {0, 1, 1},
        // colMagenta[]    = {1, 0, 1},
        colBlack[]    = {0, 0, 0};
//---------------------------------------------------------------------


void CoordinateSystem(GLdouble x, GLdouble y, GLdouble z)
{
glBegin(GL_LINES);
  glColor3fv(colRed);
    glVertex3d(-x, 0, 0);
    glVertex3d(x, 0, 0);
  glColor3fv(colGreen);
    glVertex3d(0, -y, 0);
    glVertex3d(0, y, 0);
  glColor3fv(colBlue);
    glVertex3d(0, 0, z);
    glVertex3d(0, 0, -z);
glEnd();
double h=0.05;
glBegin(GL_TRIANGLES); // Arrow at the end of the x-Axis
  glColor3fv(colRed);
  glVertex3d(x-h, h, 0);
  glVertex3d(x, 0, 0);
  glVertex3d(x-h, -h, 0);
glEnd();

glBegin(GL_TRIANGLES);
  glColor3fv(colGreen);
  glVertex3d(+h,y-h, 0);
  glVertex3d(0, y, 0);
  glVertex3d(-h,y-h, 0);
glEnd();

glBegin(GL_TRIANGLES);
  glColor3fv(colBlue);
  glVertex3d(0,+h, z-h);
  glVertex3d(0, 0, z);
  glVertex3d(0,-h, z-h);
glEnd();
// glScaled(x,y,z);
}

void Quad(GLdouble n[], GLdouble v0[], GLdouble v1[],
```

```
              GLdouble v2[],GLdouble v3[], GLenum mode)
{ // Viereck mit den Eckpunkten v0, v1, v2 und v3
glBegin(mode);
  glNormal3dv(n);  // Normale
  glVertex3dv(v0);
  glVertex3dv(v1);
  glVertex3dv(v2);
  glVertex3dv(v3);
glEnd();
}

/*
void Quader(GLdouble x0, GLdouble y0, GLdouble z0,
            GLdouble x1, GLdouble y1, GLdouble z1, GLenum mode)
{
  GLdouble tmp;
  // sicherstellen, daß x0<=x1, y0<=y1 und z0<=z1 gilt
  if (x0 > x1) {tmp = x0; x0 = x1; x1 = tmp;}
  if (y0 > y1) {tmp = y0; y0 = y1; y1 = tmp;}
  if (z0 > z1) {tmp = z0; z0 = z1; z1 = tmp;}

  GLdouble v[8][3] =     // Eckpunkte     v[7] ---- v[6]
           {{x0,y0,z1}, //=v[0]           /|        /|
            {x1,y0,z1}, //=v[1]        v[3] ---- v[2]|
            {x1,y1,z1}, //=v[2]         |  |      |  |
            {x0,y1,z1}, //=v[3]         |  |      |  |
            {x0,y0,z0}, //=v[4]         |  |      |  |
            {x1,y0,z0}, //=v[5]         | v[4] ---| v[5]
            {x1,y1,z0}, //=v[6]         |/        |/
            {x0,y1,z0}};//=v[7]        v[0] ---- v[1]

  GLdouble Normale_vorn[]  = { 0, 0, 1},
           Normale_hinten[]= { 0, 0,-1},
           Normale_rechts[]= { 1, 0, 0},
           Normale_links[] = {-1, 0, 0},
           Normale_oben[]  = { 0, 1, 0},
           Normale_unten[] = { 0,-1, 0};

  Viereck(Normale_vorn,  v[0],v[1],v[2],v[3],mode); // vorne
  Viereck(Normale_rechts,v[1],v[5],v[6],v[2],mode); // rechts
  Viereck(Normale_hinten,v[5],v[4],v[7],v[6],mode); // hinten
  Viereck(Normale_links, v[4],v[0],v[3],v[7],mode); // links
  Viereck(Normale_oben,  v[3],v[2],v[6],v[7],mode); // oben
  Viereck(Normale_unten, v[1],v[0],v[4],v[5],mode); // unten
// von außen betrachtet alle im Gegen-Uhrzeigersinn, OpenGL Prog Guide S. 82
}
*/

void ColoredCube(GLdouble x0, GLdouble y0, GLdouble z0,
                 GLdouble x1, GLdouble y1, GLdouble z1, GLenum mode)
{ // wire cube: mode=GL_LINE_LOOP, full sides: mode=GL_QUADS
  // to ensure x0<=x1, y0<=y1 und z0<=z1:
  if (x0 > x1) {GLdouble tmp = x0; x0 = x1; x1 = tmp;}
  if (y0 > y1) {GLdouble tmp = y0; y0 = y1; y1 = tmp;}
  if (z0 > z1) {GLdouble tmp = z0; z0 = z1; z1 = tmp;}

  GLdouble v[8][3] =     // corners       v[7] ---- v[6]
           {{x0,y0,z1}, //=v[0]           /|        /|
            {x1,y0,z1}, //=v[1]        v[3] ---- v[2]|
            {x1,y1,z1}, //=v[2]         |  |      |  |
            {x0,y1,z1}, //=v[3]         |  |      |  |
            {x0,y0,z0}, //=v[4]         |  |      |  |
            {x1,y0,z0}, //=v[5]         | v[4] ---| v[5]
            {x1,y1,z0}, //=v[6]         |/        |/
            {x0,y1,z0}};//=v[7]        v[0] ---- v[1]

  GLdouble FrontNormal[]={0,0,1}, BackNormal[]={ 0, 0,-1},
           RightNormal[]={1,0,0}, LeftNormal[]={-1, 0, 0},
           UpNormal[]   ={0,1,0}, DownNormal[]={ 0,-1, 0};
```

```
// glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colBlue);
glColor3fv(colRed);
Quad(FrontNormal,  v[0],v[1],v[2],v[3],mode); // front
// glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colGreen);
glColor3fv(colGreen);
Quad(RightNormal,v[1],v[5],v[6],v[2],mode); // right
// glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colBlue);
glColor3fv(colBlue);
Quad(BackNormal,v[5],v[4],v[7],v[6],mode); // back
// glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colYellow);
glColor3fv(colYellow);
Quad(LeftNormal, v[4],v[0],v[3],v[7],mode); // left
// glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colLightGray);
glColor3fv(colLightGray);
Quad(UpNormal,  v[3],v[2],v[6],v[7],mode); // up
// glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colDarkGray);
glColor3fv(colDarkGray);
Quad(DownNormal, v[1],v[0],v[4],v[5],mode); // down
}

void City()
{
srand(1); // always the same random numbers
const int n = 10;
const double w=2.0/n;
for (int x=-n/2; x<n/2; x++)
  for (int z=-n/2; z<n/2; z++)
    ColoredCube((x-w)/n, 0,                  (z-w)/n,
                (x+w)/n, (1+rand()%8)/10.0, (z+w)/n, GL_QUADS);
}

/*
void glutWireSphere(GLdouble radius, GLint slices, GLint stacks)
{ // from  glut-3.7\lib\glut\glut_shapes.c
  QUAD_OBJ_INIT();
  gluQuadricDrawStyle(quadObj, GLU_LINE);
  gluQuadricNormals(quadObj, GLU_SMOOTH);
  // If we ever changed/used the texture or orientation state
  // of quadObj, we'd need to change it to the defaults here
  // with gluQuadricTexture and/or gluQuadricOrientation.
  gluSphere(quadObj, radius, slices, stacks);
}

void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks)
{ // from  glut-3.7\lib\glut\glut_shapes.c
  QUAD_OBJ_INIT();
  gluQuadricDrawStyle(quadObj, GLU_FILL);
  gluQuadricNormals(quadObj, GLU_SMOOTH);
  // If we ever changed/used the texture or orientation state
  // of quadObj, we'd need to change it to the defaults here
  // with gluQuadricTexture and/or gluQuadricOrientation.
  gluSphere(quadObj, radius, slices, stacks);
}
*/


void Sphere(GLdouble radius, GLint slices, GLenum style, GLfloat* color)
{ // ähnlich wie in glut-3.7\lib\glut\glut_shapes.c
static GLUquadricObj *quadObj=gluNewQuadric();
gluQuadricDrawStyle(quadObj, style);
gluQuadricNormals(quadObj, GLU_SMOOTH);
glColor3fv(color);
gluSphere(quadObj, radius, slices, slices/*stacks*/);
}

void Cylinder(GLdouble baseRadius, GLdouble topRadius, GLdouble height, GLenum
style, GLfloat* color)
{ // ähnlich wie in glut-3.7\lib\glut\glut_shapes.c
```

```
static GLUquadricObj *quadObj=gluNewQuadric();
gluQuadricDrawStyle (quadObj, style);
gluQuadricNormals(quadObj, GLU_SMOOTH);
GLint slices=30, stacks=10;
glColor3fv(color);
gluCylinder(quadObj, baseRadius, topRadius, height, slices, stacks);
}

void Speichenrad(int n)
{
glPushMatrix();
Sphere(0.2,20,GLU_FILL,colRed); // Kugel im Mittelpunkt
for (int i=0; i<n; i++)
  {
    glRotated(360.0/n,0,0,1); // 360/n Grad weiter drehen
    glPushMatrix();
      glRotated(-90,1,0,0); // in die x-z-Ebene drehen
      Cylinder(0.02, 0.02, 1,GLU_FILL, colGray);
      glTranslated(0,0,1);
      Sphere(0.1,20,GLU_FILL,colBlue);
    glPopMatrix();
  }
glPopMatrix();
}

void IndependentMotions()
{
static int angle=0;
angle=angle+5;
glPushMatrix();
  glTranslated(1,0,0);
  glRotated(angle-10,0,0,1);
  Speichenrad(10);
glPopMatrix();
glPushMatrix();
  glTranslated(-1,0,0);
  glRotated(-angle,0,0,1);
  Speichenrad(10);
glPopMatrix();
}

// für RotateTranslate:

void InitScene()
{ // Initialisiere die Szene so, dass die Modelle
  // beim Aufruf von RotateTranslate ins Bild passen
glLoadIdentity();
glTranslated(-0.8,-0.1,-2.5);
}

void Modell()
{
CoordinateSystem(1,1,1);
ColoredCube(-0.1, -0.1, -0.1, 0.1, 0.1, 0.1, GL_QUADS);
}

void RotateTranslate(bool b)
{ // b==true: first translate, then rotate
  // b==false: first rotate, then translate
InitScene();
Modell();
if (b) glTranslated(1.5,0,0);
glRotated(20,0,0,1);
if (!b) glTranslated(1.5,0,0);
Modell();
}
```

```
void InitLights()
{
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
GLfloat light_position[] = {1, 5, 5, 0};
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
GLfloat mat[] = { 0, 0, 1, 1};
glMaterialfv(GL_FRONT, GL_AMBIENT, mat);
GLfloat light[] = { 0, 0, 0.7, 1.0 };
glLightfv(GL_LIGHT0, GL_AMBIENT, light);
}


//----- Aufgabe 1. a) ------------------------------------------------

void Arrow()
{
glBegin(GL_LINES);
    glVertex3d(-1, 0, 0);
    glVertex3d(1, 0, 0);
glEnd();
double h=0.05;
glBegin(GL_TRIANGLES);
  glVertex3d(1-h, h, 0);
  glVertex3d(1, 0, 0);
  glVertex3d(1-h, -h, 0);
glEnd();
}

void CoordinateSystem()
{
glPushMatrix();
glColor3fv(colRed);
Arrow();
glPushMatrix();
  glRotated(90,0,0,1);
  glColor3fv(colGreen);
  Arrow();
glPopMatrix();
glPushMatrix();
  glRotated(-90,0,1,0);
  glColor3fv(colBlue);
  Arrow();
glPopMatrix();
glPopMatrix();
}


//----- Aufgabe 1. b) ------------------------------------------------

void H2O(double radH=0.2, double radO=0.1, double dist=0.8)
{ // simple model of a water molecule
glPushMatrix();
Sphere(radH,20,GLU_FILL,colRed); // H
for (int i=0; i<2; i++)          // 2O
  {
    glRotated(120,0,0,1);
    glPushMatrix();
      glRotated(-90,1,0,0); // in die x-z-Ebene drehen
      Cylinder(0.02, 0.02, dist,GLU_FILL, colYellow);
      glTranslated(0,0,dist);
      Sphere(radO,20,GLU_FILL,colBlue);
    glPopMatrix();
  }
glPopMatrix();
}


//----- Aufgabe 1. c) ------------------------------------------------
```

```
void RotatingCubeAndCylinder()
{
static int angle=0;
angle=angle+5;
glPushMatrix();
  glTranslated(1,0,0);
  glRotated(angle-10,0,1,1);
  double w=0.2;
  ColoredCube(-w, -w, -w, w, w, w, GL_QUADS);
glPopMatrix();
glPushMatrix();
  glTranslated(-1,0,0);
  glRotated(-angle,1,0,1);
  Cylinder(0.2, 0, 1 ,GLU_FILL, colYellow);
glPopMatrix();
}


//----- Aufgabe 1. d) ------------------------------------------------

void City2(const int n)
{
srand(1); // Jedesmal dieselbe Folge von Zufallszahlen
const double w=2.0/n;
for (int x=-n/2; x<n/2; x++)
  for (int z=-n/2; z<n/2; z++)
   {
   glPushMatrix();
     glTranslated((x-w)/n,0,(z-w)/n);
     ColoredCube(0, 0, 0,
                 2*w/n, (1+rand()%8)/10.0, 2*w/n, GL_QUADS);
   glPopMatrix();
   }
}


//----- Aufgabe 1. e) ------------------------------------------------

void Robot()
{
static int i=0, angle;
i++;
if (i==36) i=0;
if (i<18) angle=5*i;
else angle=180-5*i;

glPushMatrix();
  glRotatef (-90, 1, 0, 0);
  Cylinder(0.2, 0.2, 1, GLU_FILL, colGreen); // body
glPopMatrix();
glPushMatrix();
  glTranslatef (0, 1.3, 0);
  Sphere(0.3,20,GLU_FILL,colBlue);  // head
glPopMatrix();
glPushMatrix();
  glRotatef (-90, 1, 0, 0);
  glTranslatef (0, 0.2, 0.8);
  glRotatef (90+angle, 0, 1, 0);
  Cylinder(0.1, 0.1, 0.6, GLU_FILL, colRed); // arm
glPopMatrix();
glPushMatrix();
  glRotatef (-90, 1, 0, 0);
  glTranslatef (0, -0.2, 0.8);
  glRotatef (90+angle, 0, 1, 0);
  Cylinder(0.1, 0.1, 0.6, GLU_FILL, colRed); // arm
glPopMatrix();
}
```

```
//----- Aufgabe 1. f) ------------------------------------------------

void  Solarsystem(bool  LocalTransf,  GLenum  style,  bool  TexturedEarth,  bool
TwoMoons)
{
void TexturedSphere(GLdouble radius); // in textures.cpp

if (LocalTransf) glPushMatrix(); // siehe Abschnitt 11.1.5
else glLoadIdentity();
static int time = 0, day = 0;
const int incr = 25; // in hours
time = time+incr;
day = day+time/24;
time = time%24;
day = day %365;
// look from above:
glTranslated(0, 0, -10.0);
glRotated(10, 1, 0, 0);

Sphere(1,50,GLU_FILL,colYellow); // yellow sun

// The earth rotates in 365 days round the sun:
glRotated(360.0*day/365, 0, 1, 0); // rotate around the y-axis
glTranslated(4, 0, 0); // distance earth - moon
// The earth rotates in 24 hours around her axis:
glRotatef(360.0*time/24.0, 0, 1, 0);

if (LocalTransf)
  {
    glPushMatrix();
    glRotatef(90,1,0,0); // north pole up
    if (TexturedEarth) TexturedSphere(0.5);
    // With style=GLU_LINE and slices=5 one sees the
    // rotatation of the earth around her axis:
    else Sphere(0.5,5,style,colBlue);  // blue earth
    glPopMatrix();
  }
else if (TexturedEarth) TexturedSphere(0.5);
else Sphere(0.5,5,style,colBlue);  // blue earth

// The moon rotates 12.5 times each year around the earth. He doesn't
// rotate around an axis, but shows always the same side to the earth:

if (TwoMoons) glPushMatrix();
glRotated(360.0*12.5*day/365, 0, 1, 0);
glTranslated(2, 0, 0); // distance earth - moon
Sphere(0.2,50,GLU_FILL,colRed);// red moon

if (TwoMoons)
  {
    glPopMatrix();
    glRotated(360.0*9*day/365, 0, 1, 0);
    glTranslated(1.5, 0, 0); // distance earth - moon
    Sphere(0.2,50,GLU_FILL,colGreen);// green moon
  }
if (LocalTransf) glPopMatrix();
}

// Text:

void Solarsystem(bool LocalTransf, GLenum style)
{
if (LocalTransf) glPushMatrix(); // siehe Abschnitt 11.1.5
else glLoadIdentity();
static int time = 0, day = 0;
const int incr = 25; // in hours
time = time+incr;
day = day+time/24;
```

```
time = time%24;
day = day %365;
// look from above:
glTranslated(0, 0, -10.0);
glRotated(10, 1, 0, 0);

Sphere(1,50,GLU_FILL,colYellow); // yellow sun

// The earth rotates in 365 days round the sun:
glRotated(360.0*day/365, 0, 1, 0); // rotate around the y-axis
glTranslated(4, 0, 0); // distance earth - sun
// The earth rotates in 24 hours around her axis:
glRotatef(360.0*time/24.0, 0, 1, 0);

Sphere(0.5,5,style,colBlue);  // blue earth

// The moon rotates 12.5 times each year around the earth. He doesn't
// rotate around an axis, but shows always the same side to the earth:

glRotated((360*12.5*day)/365, 0, 1, 0);
glTranslated(2, 0, 0); // distance earth - moon
Sphere(0.2,50,GLU_FILL,colRed);// red moon

if (LocalTransf) glPopMatrix();
}



//----- Aufgabe 2. ------------------------------------------------

enum {trRotate, trTranslate} Transformation=trTranslate;

void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
if (Key=='R') Transformation=trRotate;
else if (Key=='T') Transformation=trTranslate;

bool ShiftPressed=Shift.Contains(ssShift);

if (Transformation==trTranslate)
  { //VK_UP, VK_DOWN: Pfeiltasten hoch, tief usw.
    if      (Key==VK_UP   && !ShiftPressed)  glTranslated(0,0, 0.1);
    else if (Key==VK_DOWN && !ShiftPressed)  glTranslated(0,0,-0.1);
    else if (Key==VK_UP)    glTranslated(0, 0.1, 0);
    else if (Key==VK_DOWN)  glTranslated(0,-0.1, 0);
    else if (Key==VK_LEFT)  glTranslated(-0.1, 0, 0);
    else if (Key==VK_RIGHT) glTranslated( 0.1, 0, 0);
  }
else if (Transformation==trRotate)
  {
    if(Key==VK_UP && !ShiftPressed)    glRotated(-5, 1, 0, 0);
    else if(Key == VK_DOWN && !ShiftPressed)
      glRotated(5, 1, 0, 0);
    else if(Key == VK_UP)    glRotated(-5, 0, 0, 1);
    else if(Key == VK_DOWN)  glRotated(5, 0, 0, 1);
    else if(Key == VK_LEFT)  glRotated(-5, 0, 1, 0);
    else if(Key == VK_RIGHT) glRotated(5, 0, 1, 0);
  }
DrawScene();
}


//----- Aufgabe 3. ------------------------------------------------

#include "Lights.h"
void __fastcall TForm1::LightsOnClick(TObject *Sender)
{
InitLights();
```

```
Form2->Show();
Form1->LightsOn->Checked=true;
Form1->LightsOff->Checked=false;
}

void __fastcall TForm1::LightsOffClick(TObject *Sender)
{
Form1->LightsOn->Checked=false;
Form1->LightsOff->Checked=true;
glDisable(GL_LIGHTING);
glDisable(GL_LIGHT0);
}


enum {modCity, modSolarsystemOneMoon, modSolarsystemTwoMoons,
      modIndependentMotions, modRobot, modTree , modOnePaperplane,
      modSomeAnimatedPaperplanes, modSimpleTexturedFigures,
      modRotatingCubeAndCylinder, modH2O, modSolarsystemTexturedEarth,
      modSimpleTexturedRoom} Model=modCity;

bool ShowCoordinateSystem = true;

void __fastcall TForm1::CoordinateSystemClick(TObject *Sender)
{
ShowCoordinateSystem =! ShowCoordinateSystem;
CoordinateSystem->Checked =ShowCoordinateSystem;
}

void __fastcall TForm1::rotatingcubeandcylinder1Click(TObject *Sender)
{
Model=modRotatingCubeAndCylinder;
}

void __fastcall TForm1::City1Click(TObject *Sender)
{
Model=modCity;
}

void __fastcall TForm1::H201Click(TObject *Sender)
{
Model=modH2O;
}

void __fastcall TForm1::RobotClick(TObject *Sender)
{
Model=modRobot;
}

void __fastcall TForm1::Solarsystem1Click(TObject *Sender)
{
Model=modSolarsystemOneMoon;
}

void __fastcall TForm1::SolarsystemTwoMoonsClick(TObject *Sender)
{
Model=modSolarsystemTwoMoons;
}

void __fastcall TForm1::independantmotions1Click(TObject *Sender)
{
Model=modIndependentMotions;
}

void __fastcall TForm1::recursiveTree1Click(TObject *Sender)
{
Model=modTree;
}

void __fastcall TForm1::onepaperplane1Click(TObject *Sender)
```

```
{
Model=modOnePaperplane;
}

void __fastcall TForm1::someanimatedpaperplanes1Click(TObject *Sender)
{
Model=modSomeAnimatedPaperplanes;
}

void __fastcall TForm1::texturedearth1Click(TObject *Sender)
{
Model=modSolarsystemTexturedEarth;
}

//----- Aufgabe 4. ------------------------------------------------

#include "Textures.cpp"
void __fastcall TForm1::LoadBitmapClick(TObject *Sender)
{
if (OpenPictureDialog1->Execute())
  {
  LoadWindowsBitmap(OpenPictureDialog1->FileName.c_str());
  InitTexture();
  DrawScene();
  }
}

//----- Aufgabe 4. a) ---------------------------------------------

void __fastcall TForm1::SimpleTextureClick(TObject *Sender)
{
MakeTexture();
InitTexture();
DrawScene();
}

void __fastcall TForm1::SimpleFigures1Click(TObject *Sender)
{
Model=modSimpleTexturedFigures;
DrawScene();
}

//----- Aufgabe 4. a) ---------------------------------------------

bool inSimpleTexturedRoom=false;

void __fastcall TForm1::simpleroom1Click(TObject *Sender)
{
inSimpleTexturedRoom=!inSimpleTexturedRoom;
}

#include "RecTree.cpp"
#include "paperplane.cpp"

void  DrawScene()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

double w=5;
if (inSimpleTexturedRoom)
  TexturedCube(-w, -w, -2*w, w, w, 2*w, GL_QUADS);

// Model =-1;
// RotateTranslate(false);
// Solarsystem(false, GLU_LINE);

if (ShowCoordinateSystem)
  CoordinateSystem();
```

```
if (Model==modH2O) H2O();
else if (Model==modRotatingCubeAndCylinder) RotatingCubeAndCylinder();
else if (Model==modCity) City2(10);
else if (Model==modRobot) Robot();
//  void  Solarsystem(bool  PushPop,  GLenum  style,  bool  TexturedEarth,  bool
TwoMoons)
else if (Model==modSolarsystemTwoMoons) Solarsystem(true,GLU_FILL,false,true);
else if (Model==modSolarsystemOneMoon)  Solarsystem(false,GLU_LINE,false,false);
else if (Model==modIndependentMotions) IndependentMotions();
else if (Model==modOnePaperplane) Paperplane(true);
else if (Model==modSomeAnimatedPaperplanes) AnimatedPaperplanes(3,false);

else                    if                    (Model==modSolarsystemTexturedEarth)
Solarsystem(true,GLU_LINE,true,false);
else  if  (Model==modSimpleTexturedRoom)  TexturedCube(-2,  -2,  -10,  2,  2,  10,
GL_QUADS);
else if (Model==modSimpleTexturedFigures) TexturedFigures();
else if (Model==modTree)
  {
    bool LightsOn=glIsEnabled(GL_LIGHTING);
    if (!LightsOn)
      {
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
      }
    RecursiveTree(6);
    if (!LightsOn)
      {
        glDisable(GL_LIGHTING);
        glDisable(GL_LIGHT0);
      }
  }

if (UseDoubleBuffer) SwapBuffers(dc);
else glFlush();
}
```

```
// File: C:\Loesungen_CB2006\Kap_10\10.19\paperplane.cpp

#include <cmath>
using namespace std; // oder besser std::
void  Normal3P(const  double  v1[3],  const  double  v2[3],  const  double  v3[3])//,
double n[3])
{ // calculates normal to the points v1, v2, v3
double v[3]={v1[0]-v2[0],v1[1]-v2[1],v1[2]-v2[2]},//v1-v2
       w[3]={v2[0]-v3[0],v2[1]-v3[1],v2[2]-v3[2]},//v2-v3
       c[3]={v[1]*w[2]-v[2]*w[1],v[2]*w[0]-v[0]*w[2],
             v[0]*w[1] - v[1]*w[0]},
       d=sqrt(c[0]*c[0]+c[1]*c[1]+c[2]*c[2]);
if (d==0) d=1; // nicht schön, aber auch nicht schlimm
double n[3]={c[0]/d,c[1]/d,c[2]/d};
glNormal3dv(n);
}

void Paperplane(bool colored)
{
//glShadeModel(GL_FLAT); // Damit erhält das gesamte Primitiv dieselbe Farbe.
// So sieht man, wie sich der Papierflieger aus Dreiecken zusammensetzt.
// Bei der Voreinstellung glShadeModel(GL_SMOOTH) werden die Farben
// zwischen den Ecken interpoliert.
glBegin(GL_TRIANGLE_STRIP);
 if (colored) glMaterialfv(GL_FRONT, GL_AMBIENT, colRed);
 glColor3fv(colRed);
   GLdouble p0[]={-0.7, 0, -0.1}; // linker Flügel
   GLdouble p1[]={-0.1, 0, 0};    // linker Flügel
   GLdouble p2[]={-0.1, 0.7, 0};  // linker Flügel
```

```
   Normal3P(p0,p1,p2);
   glVertex3dv(p0);
   glVertex3dv(p1);
   glVertex3dv(p2);
 glColor3fv(colGreen);
 if (colored) glMaterialfv(GL_FRONT, GL_AMBIENT, colGreen);
   GLdouble p3[]={0, 0, -0.3}; // linke Seite
   Normal3P(p1,p3,p2);
   glVertex3dv(p3);
 glColor3fv(colBlue);
 if (colored) glMaterialfv(GL_FRONT, GL_AMBIENT, colBlue);
   GLdouble p4[]={0, 0.8, -0.3}; // linke Seite
   Normal3P(p3,p4,p2);
   glVertex3dv(p4);
 glColor3fv(colYellow);
 if (colored) glMaterialfv(GL_FRONT, GL_AMBIENT, colYellow);
   GLdouble p5[]={0.1, 0, 0}; // rechte Seite
   Normal3P(p3,p5,p4);
   glVertex3dv(p5);
 glColor3fv(colGreen);
 if (colored) glMaterialfv(GL_FRONT, GL_AMBIENT, colGreen);
   GLdouble p6[]={0.1, 0.7, 0}; // rechte Seite
   Normal3P(p3,p5,p6);
   glVertex3dv(p6);
 glColor3fv(colRed);
 if (colored) glMaterialfv(GL_FRONT, GL_AMBIENT, colRed);
   GLdouble p7[]={0.7, 0, -0.1}; // Spitze rechter Flügel
   Normal3P(p5,p7,p6);
   glVertex3dv(p7);
glEnd();
//glShadeModel(GL_SMOOTH); // Voreinstellung wieder herstellen.
}


//----- Aufgabe 1. g) ------------------------------------------------

double Speed(int i)
{
double d=(i+4)/200.0;
if (i&1) d=-d;
return d;
}

void AnimatedPaperplanes(int n, bool colored)
{
static int calls=0;
calls++;
for (int i = 0; i < n; i++)
  {
    double t = (calls+10)*Speed(i);
    double x = 2*sin(2*t);
    double y = sin(t/3.4) - 0.5;
    double z = cos(t)-1.5;
//    double angle = 180*((atan(2.0) + M_PI_2) * sin(t) - M_PI_2)/M_PI;
// double xx=atan(2.0) + M_PI_2;     // 2.678
     double angle = 180*(2.678*sin(t)-1.57)/3.14;
//    double angle = 180*sin(t)/3.14;
    if (Speed(i) < 0) angle += 180;
    glPushMatrix();
      glTranslated(x, y, z);
      glRotated(290, 1, 0, 0);
      glRotated(angle, 0, 0, 1);
      Paperplane(colored);
    glPopMatrix();
  }
}
```

```
// File: C:\Loesungen_CB2006\Kap_10\10.19\RecTree.cpp

// rekursiver Baum:
void Stem()
{
GLfloat tree_mat[] = { 0.5, 0.3, 0.1, 1.0 };
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, tree_mat);

//GLUquadricObj *cylquad = gluNewQuadric();
glPushMatrix();
  glRotatef(-90, 1, 0, 0);
//  gluCylinder(cylquad, 0.1, 0.08, 1, 10, 2 );
  Cylinder(0.1, 0.08, 1, GLU_FILL, colGray);
glPopMatrix();
}

void Leaf()
{
GLfloat leaf_mat[] = { 0.0, 0.7, 0.0, 1.0 };
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, leaf_mat);

glBegin(GL_TRIANGLES);
  glNormal3f(-0.1, 0, 0.25);  // nicht normalisiert
  glVertex3f(0, 0, 0);
  glVertex3f(0.3, 0.3, 0.1);
  glVertex3f(0.3, -0.3, 0);
glEnd();
}

void StemAndLeaves0()
{
glPushMatrix();
  Stem();
  for(int i = 0; i < 3; i++)
  {
    glTranslatef(0, 0.333, 0);
    glRotatef(90, 0, 1, 0);
    glPushMatrix();
      glRotatef(0, 0, 1, 0);
      glRotatef(50, 1, 0, 0);
      Leaf();
    glPopMatrix();
    glPushMatrix();
      glRotatef(180, 0, 1, 0);
      glRotatef(60, 1, 0, 0);
      Leaf();
    glPopMatrix();
  }
//glPopAttrib();
glPopMatrix();
}

void StemWithLeaves()
{
glPushMatrix();
  Stem();
  for(int i = 0; i < 6; i++)
  {
    glTranslatef(0, 1.0/6, 0);
    glRotatef(120, 0, 1, 0);
    glPushMatrix();
      glRotatef(50, 1, 0, 0);
      Leaf();
    glPopMatrix();
  }
glPopMatrix();
}

void RecursiveTree(int level)
```

```
{
  if (level <= 0)
    StemWithLeaves();
  else
  {
    Stem();
    glPushMatrix();
      glTranslatef(0, 1, 0);
      glScalef(0.7, 0.7, 0.7);
      for (int i=0; i<3; i++)
      {
        double x=(i+1)/4.0;
        glPushMatrix();
          glRotatef(i*120 + x*40, 0, 1, 0);
          glRotatef(30 + x*20, 0, 0, 1);
          RecursiveTree(level - 1);
        glPopMatrix();
      }
    glPopMatrix();
  }
}


// File: C:\Loesungen_CB2006\Kap_10\10.19\Textures.cpp

void MakeTexture()
{
const int Width=64,Height=64; // muss Zweierpotenz sein
GLubyte Image[Width][Height][3]; // RGB
for (int i = 0; i < Width; i++)
  for (int j = 0; j < Height; j++)
    {
      int c = (((i&0x8)==0)^((j&0x8)==0)); // 0 oder 1
      Image[i][j][0] = c*255; // schwarz oder weiss
      Image[i][j][1] = c*255;
      Image[i][j][2] = c*255;
    }
glTexImage2D(GL_TEXTURE_2D, 0, 3, Width, Height,
             0, GL_RGB, GL_UNSIGNED_BYTE, Image);
}


void LoadWindowsBitmap(char* fn)
{
Graphics::TBitmap* b=new Graphics::TBitmap();
b->LoadFromFile(fn);
GLubyte *bmp =new GLubyte[b->Width*b->Height*3];
for (int i = 0; i < b->Height; i++)
  for (int j = 0; j < b->Width; j++)
    {
      int Pixel=b->Canvas->Pixels[j][i];
      bmp[3*(i*b->Width+j)+0] = GetRValue(Pixel);
      bmp[3*(i*b->Width+j)+1] = GetGValue(Pixel);
      bmp[3*(i*b->Width+j)+2] = GetBValue(Pixel);
    }
// Anstelle von glTexImage2D wird hier gluBuild2DMipmaps aufgerufen.
// Bei dieser Funktion müssen die Arraygrenzen keine Zweierpotenzen sein.
gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB, b->Width, b->Height,
                  GL_RGB, GL_UNSIGNED_BYTE, bmp);
delete b;
delete[] bmp;
}

void InitTexture()
{
// MakeTexture(); // oder LoadWindowsBitmap("...");
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
//    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
//    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
```

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
}


void TexturedQuad(GLdouble v0[], GLdouble v1[],
                  GLdouble v2[], GLdouble v3[])
{ // Viereck mit den Eckpunkten v0, v1, v2 und v3
glEnable(GL_TEXTURE_2D);
glBegin(GL_QUADS);
  glTexCoord2d(0,0); glVertex3dv(v0);
  glTexCoord2d(0,1); glVertex3dv(v1);
  glTexCoord2d(1,1); glVertex3dv(v2);
  glTexCoord2d(1,0); glVertex3dv(v3);
glEnd();
glDisable(GL_TEXTURE_2D);
}

void TexturedCube(GLdouble x0, GLdouble y0, GLdouble z0,
                GLdouble x1, GLdouble y1, GLdouble z1, GLenum mode)
{
  // to ensure x0<=x1, y0<=y1 und z0<=z1:
  if (x0 > x1) {GLdouble tmp = x0; x0 = x1; x1 = tmp;}
  if (y0 > y1) {GLdouble tmp = y0; y0 = y1; y1 = tmp;}
  if (z0 > z1) {GLdouble tmp = z0; z0 = z1; z1 = tmp;}

  GLdouble v[8][3] =     // Eckpunkte     v[7] ---- v[6]
            {{x0,y0,z1}, //=v[0]           /|        /|
             {x1,y0,z1}, //=v[1]        v[3] ---- v[2]|
             {x1,y1,z1}, //=v[2]         |  |      |  |
             {x0,y1,z1}, //=v[3]         |  |      |  |
             {x0,y0,z0}, //=v[4]         |  |      |  |
             {x1,y0,z0}, //=v[5]         | v[4] ---| v[5]
             {x1,y1,z0}, //=v[6]         |/        |/
             {x0,y1,z0}};//=v[7]        v[0] ---- v[1]

  TexturedQuad(v[0],v[1],v[2],v[3]); // front
  TexturedQuad(v[1],v[5],v[6],v[2]); // right
  TexturedQuad(v[5],v[4],v[7],v[6]); // back
  TexturedQuad(v[4],v[0],v[3],v[7]); // left
  TexturedQuad(v[3],v[2],v[6],v[7]); // up
  TexturedQuad(v[1],v[0],v[4],v[5]); // down
}

void TexturedSphere(GLdouble radius)
{ // as in glut-3.7\lib\glut\glut_shapes.c
static GLUquadricObj *quadObj=gluNewQuadric();
glEnable(GL_TEXTURE_2D);
  gluQuadricDrawStyle(quadObj, GLU_FILL);
  gluQuadricNormals(quadObj, GLU_SMOOTH);
  gluQuadricTexture(quadObj, GL_TRUE);
  GLint slices=30, stacks=30;
  gluSphere(quadObj, radius, slices, stacks);
glDisable(GL_TEXTURE_2D);
}


//----- Aufgabe 4. a) ------------------------------------------------

void TexturedFigures()
{
glPushMatrix();
  GLdouble v0[]={-1.2, 0, -1}, v1[]={-1.2, 1, -1},
           v2[]={-0.2, 1, -1}, v3[]={-0.2, 0, -1};
  TexturedQuad(v0,v1,v2,v3);
  glTranslated(1, 0.5, -3.0);
  TexturedSphere(1);
glPopMatrix();
```

```
}
```

11.10.15 Aufg abe 10.20

```
// File: C:\Loesungen_CB2006\Kap_10\10.20\W32FileU.cpp

#include <vcl.h>
#pragma hdrstop

#include "W32FileU.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm1::FormCreate(TObject *Sender)
{
// Erzeuge das Verzeichnis, falls es nicht existiert
AnsiString dir="c:\\test";
if (CreateDirectory(dir.c_str(),0))
  ShowMessage("directory '"+dir+"' created");
}

//---------------------------------------------------------------------
void ShowLastError(AnsiString where)
{
MessageBox(NULL,SysErrorMessage(GetLastError()).c_str(),
           where.c_str(),MB_OK|MB_ICONERROR);
/* Ohne VCL:
LPTSTR lpMsgBuf; // char*
FormatMessage( // siehe Online-Hilfe zu Win32
  FORMAT_MESSAGE_ALLOCATE_BUFFER |
FORMAT_MESSAGE_FROM_SYSTEM,
  NULL,
  GetLastError(),
  MAKELANGID(LANG_NEUTRAL,SUBLANG_DEFAULT),//Default language
 (LPTSTR)&lpMsgBuf, //Adresse des nullterminierten Textes
  0,
  NULL);
MessageBox(NULL,lpMsgBuf,where.c_str(),
MB_OK|MB_ICONERROR);
LocalFree( lpMsgBuf ); // Free the buffer.
*/
}


//----- Aufgabe 11.2.1 ------------------------------------------------

HANDLE h1;

void __fastcall TForm1::Open1Click(TObject *Sender)
{
char* fn="c:\\test\\test.dat";
h1=CreateFile(
              fn, // Dateiname
              GENERIC_READ|GENERIC_WRITE,
              FILE_SHARE_READ|FILE_SHARE_WRITE,
              0, // 0: keine SECURITY_ATTRIBUTES
              OPEN_ALWAYS,         // immer neu anlegen
```

```
              FILE_ATTRIBUTE_NORMAL, // Dateiattribut
              0); // kein TemplateFile
if (h1==INVALID_HANDLE_VALUE)
  ShowLastError("CreateFile 1");
}


typedef int DataRec;
const int RecSize=sizeof(DataRec);
int RecNr;

void __fastcall TForm1::Write1Click(TObject *Sender)
{
DWORD lpNumberOfBytesWritten;
for (int i=0; i<10; i++)
  {
  DataRec x=i;
  BOOL b=WriteFile(
      h1, // Handle der Datei
      &x,    // Adresse der Daten
      RecSize,    // Anzahl der zu schreibenden Bytes
      &lpNumberOfBytesWritten,      // Adresse für die Anzahl der geschriebenen
Bytes
      0);     // für overlapped I/O , sonst 0
  if (!b)
    ShowLastError("CreateFile 1");
  }
}

void __fastcall TForm1::Read1Click(TObject *Sender)
{
DWORD sfp=SetFilePointer(
    h1,    // Handle der Datei
    0,    // Anzahl der Bytes, um die der File-Pointer
// relativ zum Startpunkt bewegt werden soll
    0,    // Für Bewegungen > 232-2 Bytes
    FILE_BEGIN);     // Startpunkt
if (sfp== 0xFFFFFFFF)
  ShowMessage("Fehler bei SetFilePointer");

DWORD NumberOfBytesRead;
DataRec x;
bool success = true, eof=false;
while (success && !eof)
 {
  success=ReadFile(h1,&x,sizeof(x),&NumberOfBytesRead,0);
  if (!success) ShowLastError("ReadFile");
  eof=NumberOfBytesRead<sizeof(x);
  if (!eof) // Datensatz kann bearbeitet werden
    Form1->Memo1->Lines->Add(IntToStr(x));
 }
}

void __fastcall TForm1::Close1Click(TObject *Sender)
{
CloseHandle(h1);
}


//----- Aufgabe 11.2.2 ------------------------------------------------

HANDLE OpenFile(char* fn)
{
HANDLE h=CreateFile(fn,GENERIC_READ,
              0,0,OPEN_ALWAYS,FILE_ATTRIBUTE_NORMAL,0);
if (h==INVALID_HANDLE_VALUE)
  ShowLastError("CreateFile");
return h;
}
```

```
void __fastcall TForm1::FehlermeldungClick(TObject *Sender)
{
OpenFile(":::"); // Unzulässiger Dateiname

HANDLE h=OpenFile("c:\\test.dat");
DWORD x,NumberOfBytesRead;
bool    success=WriteFile(h,&x,sizeof(x),&NumberOfBytesRead,0);    //    Schreiben
unzulässig
  if (!success) ShowLastError("ReadFile");

success=WriteFile(0,&x,sizeof(x),&NumberOfBytesRead,0); // Unzulässiges Handle
  if (!success) ShowLastError("ReadFile");

OpenFile("c:\\test.dat"); // Ein zweites Mal öffnen
}


//----- Aufgabe 11.2.3 ----------------------------------------------

HANDLE lfh;

void __fastcall TForm1::OpenSharedClick(TObject *Sender)
{
char* fn="c:\\test\\test.dat";
lfh= CreateFile(
              fn, // Dateiname
              GENERIC_READ|GENERIC_WRITE,
              FILE_SHARE_READ|FILE_SHARE_WRITE,
              0, // 0: keine SECURITY_ATTRIBUTES
              OPEN_ALWAYS,         // immer neu anlegen
              FILE_ATTRIBUTE_NORMAL, // Dateiattribut
              0); // kein TemplateFile
if (lfh==INVALID_HANDLE_VALUE)
  ShowLastError("CreateFile 1");
}

void __fastcall TForm1::CloseClick(TObject *Sender)
{
CloseHandle(lfh);
}

void __fastcall TForm1::LockClick(TObject *Sender)
{
if (LockFile(lfh,RecNr*RecSize,0, RecSize,0))
  {
    // ShowMessage("locked");
  }
else
  {
    ShowMessage("Already locked");
  }
}

void __fastcall TForm1::UnlockClick(TObject *Sender)
{
if (UnlockFile(lfh,RecNr*RecSize,0, RecSize,0))
  {
    // ShowMessage("Unlocked");
  }
else
  {
    ShowMessage("Unlock not possible");
  }
}

void __fastcall TForm1::UpDown1Click(TObject *Sender, TUDBtnType Button)
{
```

```
RecNr=UpDown1->Position;
}



void __fastcall TForm1::ReadClick(TObject *Sender)
{
DWORD sfp=SetFilePointer(
    lfh,    // Handle der Datei
    RecNr*RecSize,    // Anzahl der Bytes, um die der File-Pointer
// relativ zum Startpunkt bewegt werden soll
    0,    // Für Bewegungen > 2^32-2 Bytes
    FILE_BEGIN);     // Startpunkt
if (sfp== 0xFFFFFFFF)
  ShowMessage("Fehler bei SetFilePointer");

DWORD NumberOfBytesRead;
DataRec x;
bool success=ReadFile(lfh,&x,sizeof(x),&NumberOfBytesRead,0);
if (!success) ShowLastError("ReadFile");
bool eof=NumberOfBytesRead<sizeof(x);
if (eof)
  ShowMessage("eof bei ReadFile");
Edit2->Text=IntToStr(x);
}
//---------------------------------------------------------------------

void __fastcall TForm1::WriteClick(TObject *Sender)
{
DWORD sfp=SetFilePointer(
    lfh,    // Handle der Datei
    RecNr*RecSize,    // Anzahl der Bytes, um die der File-Pointer
// relativ zum Startpunkt bewegt werden soll
    0,    // Für Bewegungen > 2^32-2 Bytes
    FILE_BEGIN);     // Startpunkt
if (sfp== 0xFFFFFFFF)
  ShowMessage("Fehler bei SetFilePointer");
  DataRec x=StrToInt(Edit2->Text);
  DWORD NumberOfBytesWritten;
  BOOL b=WriteFile(
      lfh, // Handle der Datei
      &x,    // Adresse der Daten
      RecSize,    // Anzahl der zu schreibenden Bytes
      &NumberOfBytesWritten,    // Adresse für die Anzahl der geschriebenen Bytes
      0);     // für overlapped I/O , sonst 0
  if (!b)
    ShowLastError("WriteFile");
}
```

11.10.16 Aufgabe 10.21

```
// File: C:\Loesungen_CB2006\Kap_10\10.21\SerCommU.cpp

#include <vcl\vcl.h>
#pragma hdrstop
#include "SerCommU.h"
//---------------------------------------------------------------------
// #pragma link "ComPort"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
```

```
void ShowLastError(AnsiString where)
{ // In der VCL einfach mit SysErrorMessage:
MessageBox(NULL,SysErrorMessage(GetLastError()).c_str(),where.c_str(),MB_OK|MB_IC
ONERROR);
/* Ohne SysErrorMessage:
LPTSTR lpMsgBuf; // char*
FormatMessage( // siehe Online-Hilfe zu Win32
    FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
    NULL,
    GetLastError(),
    MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
    (LPTSTR)&lpMsgBuf, // Adresse des nullterminierten Meldungstextes
    0,
    NULL);
MessageBox( NULL, lpMsgBuf,where.c_str(),MB_OK|MB_ICONERROR);
LocalFree( lpMsgBuf ); // Free the buffer.
*/
}


void ShowTextAndValue(TMemo* t,AnsiString s,int Value)
{
t->Lines->Add(s+IntToStr(Value));
}

void showDCB(TMemo* M,HANDLE hCom)
{
DCB dcb; // Device Control Block
BOOL fSuccess = GetCommState(hCom, &dcb); // DCB lesen
if (!fSuccess) ShowLastError("GetCommState");
//    DWORD DCBlength;           // sizeof(DCB)
//    DWORD BaudRate;            // current baud rate
ShowTextAndValue(M,"baud rate: ",dcb.BaudRate);
ShowTextAndValue(M,"parity: ",dcb.fParity);
ShowTextAndValue(M,"CTS output flow control: ",dcb.fOutxCtsFlow);
ShowTextAndValue(M,"DSR output flow control: ",dcb.fOutxDsrFlow);
ShowTextAndValue(M,"DTR flow control type: ",dcb.fDtrControl);
ShowTextAndValue(M,"DSR sensitivity: ",dcb.fDsrSensitivity);
ShowTextAndValue(M,"RTS flow control: ",dcb.fRtsControl);
}

HANDLE OpenComm(char* Port)
{ // Öffnet den Port und gibt sein Handle zurück
HANDLE hCom = CreateFile(Port,// z.B. "COM1",
    GENERIC_READ | GENERIC_WRITE, // Zum Senden und Empfangen
    0,  // Für comm devices exclusive-access notwendig
    0,  // Keine security attributes
    OPEN_EXISTING, // Für comm devices OPEN_EXISTING notwendig
    0,  // In den folgenden Beispielen kein overlapped I/O
    0); // Für comm devices muß hTemplate NULL sein
if (hCom == INVALID_HANDLE_VALUE)
  ShowLastError("CreateFile: "+AnsiString(Port));
return hCom;
}

void SetDCB(HANDLE hCom)
{
DCB dcb; // Device Control Block
BOOL fSuccess = GetCommState(hCom, &dcb); // DCB lesen
if (!fSuccess) ShowLastError("GetCommState");
// Setze die Baudrate=9600, 8 Datenbits, keine Parity, 1 Stopbit:
dcb.BaudRate = 9600; // Baudrate=9600
dcb.ByteSize = 8;    // 8 Datenbits
dcb.Parity   = NOPARITY;   // keine Parity
dcb.StopBits = ONESTOPBIT; // 1 Stopbit

bool NoFlowControl=false,
     HardwareFlowControl=false,
     SoftwareFlowControl=true;
```

```
// Oft spricht man auch von Hardware- bzw. Software-Handshake
// Die folgenden Ausführungen nach dem MSDN-Artikel
if (NoFlowControl)
{
  // kein Hardware Flowcontrol:
  dcb.fOutxCtsFlow=false;
  dcb.fOutxDsrFlow=false;
  // kein Software Flowcontrol:
  dcb.fInX=false;  // für Empfänger
  dcb.fOutX=false; // für Sender

  dcb.fDsrSensitivity=false;
}
else
  {
    if (HardwareFlowControl)
    {
      dcb.fOutxCtsFlow=true;
      dcb.fOutxDsrFlow=true;
    }
    // Hier kein else: Software- und HardwareFlowControl sind
    // gleichzeitig möglich, auch wenn das nicht üblich ist
    if (SoftwareFlowControl)
    {
       dcb.fInX=true;  // für Empfänger
       dcb.fOutX=true; // für Sender
       // Die folgenden Elemente steuern SoftwareFlowControl,
       // Sie müssen aber nicht gesetzt werden. Oft reichen die
       // Voreinstellungen.
       // dcb.fTXContinueOnXoff=false;
       // dcb.XonLim = ...;
       // dcb.XoffLim = ...;
       // dcb.XoffChar = ...;
       // dcb.XonChar = ...;
    }
  }

// SetCommState konfiguriert die serielle Schnittstelle
fSuccess = SetCommState(hCom, &dcb);
if (!fSuccess) ShowLastError("SetCommState");
showDCB(Form1->Memo1,hCom);
}

void SetReadTimeouts(HANDLE hCom)
{ // Der Aufruf von SetCommTimeouts ist notwendig,
  // da ohne diesen die Ergebnisse von Readfile undefiniert sind
COMMTIMEOUTS t;
// Alle Wert in Millisekunden
// Werte für ReadFile:
t.ReadIntervalTimeout=100;  // Timeout zwischen zwei Zeichen
t.ReadTotalTimeoutMultiplier=10; // pro Zeichen
t.ReadTotalTimeoutConstant=1;   //
// Wenn mit ReadFile n Zeichen gelesen werden sollen, tritt ein Timeout
// nach ReadTotalTimeoutMultiplier*n+ReadTotalTimeoutConstant ms ein.

// Werte für WriteFile: wenn beide 0 sind,
// kein Timeout beim Schreiben
t.WriteTotalTimeoutMultiplier=0;
t.WriteTotalTimeoutConstant=0;
if (!SetCommTimeouts(hCom,&t))
  ShowLastError("SetCommTimeouts");
}


void showCommProp(TMemo* M, HANDLE hCom)
{
/*typedef struct _COMMPROP { // cmmp
    WORD  wPacketLength;       // packet size, in bytes
    WORD  wPacketVersion;      // packet version
```

```
    DWORD dwServiceMask;       // services implemented
    DWORD dwReserved1;         // reserved
    DWORD dwMaxTxQueue;        // max Tx bufsize, in bytes
    DWORD dwMaxRxQueue;        // max Rx bufsize, in bytes
    DWORD dwMaxBaud;           // max baud rate, in bps
    DWORD dwProvSubType;       // specific provider type

    DWORD dwProvCapabilities;  // capabilities supported
    DWORD dwSettableParams;    // changable parameters
    DWORD dwSettableBaud;      // allowable baud rates
    WORD  wSettableData;       // allowable byte sizes
    WORD  wSettableStopParity; // stop bits/parity allowed
    DWORD dwCurrentTxQueue;    // Tx buffer size, in bytes
    DWORD dwCurrentRxQueue;    // Rx buffer size, in bytes
    DWORD dwProvSpec1;         // provider-specific data
    DWORD dwProvSpec2;         // provider-specific data

    WCHAR wcProvChar[1];       // provider-specific data
} COMMPROP;
*/
_COMMPROP CP;
GetCommProperties(hCom,&CP);
ShowTextAndValue(M,"max Tx bufsize, in bytes: ",CP.dwMaxTxQueue);
ShowTextAndValue(M,"max Rx bufsize, in bytes: ",CP.dwMaxRxQueue);
ShowTextAndValue(M,"Tx buffer size, in bytes: ",CP.dwCurrentTxQueue);
ShowTextAndValue(M,"Rx buffer size, in bytes: ",CP.dwCurrentRxQueue);
}

HANDLE hComSend,hComReceive;

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
BEmpfangClick(Form1);

hComSend=OpenComm("COM2");
hComReceive=OpenComm("COM1");

SetDCB(hComSend);
SetDCB(hComReceive);
SetReadTimeouts(hComReceive);
showCommProp(Form1->Memo1,hComSend);
showCommProp(Form1->Memo1,hComReceive);
}

void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
CloseHandle(hComSend);
CloseHandle(hComReceive);
}


int SendData(char Data[],int n)
{
DWORD NumberOfBytesWritten;    // Anzahl der gesendeten Bytes
bool b=WriteFile(hComSend,
         Data,
         n,  // Anzahl der zu sendenden Bytes
         &NumberOfBytesWritten, // Adresse, an die der Wert geschrieben wird
         0); // kein overlapped I/O
if (!b)
  ShowLastError("SendData");
return NumberOfBytesWritten;
}


void __fastcall TForm1::BSendenClick(TObject *Sender)
{
// int n=SendData("abcdefghijklmno",10);
```

```
int n=SendData(Edit1->Text.c_str(),Edit1->Text.Length());
Memo1->Lines->Add("Written: "+IntToStr(n));
}

DWORD ReceiveData(char* Data,int n)
{
DWORD NumberOfBytesRead;     // Anzahl der gelesenen Bytes
bool b=ReadFile(hComReceive, // handle des Com-Ports
         Data, // Adresse des Datenpuffers
         n,    // Anzahl der zu lesenden Bytes
         &NumberOfBytesRead, // Adresse, an die der Wert geschrieben wird
         0);     // kein overlapped I/O
if (!b)
  ShowLastError("ReceiveData");
return NumberOfBytesRead;
}

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
const int BufSize = 3;
char Buffer[BufSize];    // address of buffer that receives data
DWORD  NumberOfBytesRead=ReceiveData(Buffer,BufSize);
if (NumberOfBytesRead > 0)
 {
  AnsiString s;
  for (DWORD i=0;i<NumberOfBytesRead;i++)
    s=s+Buffer[i];
  Memo1->Lines->Add("Gelesen n="+IntToStr(NumberOfBytesRead)+" ch="+s);
 }
else
  {
  Memo1->Lines->Add("Nichts empfangen");
  }
}

void __fastcall TForm1::BEmpfangClick(TObject *Sender)
{
if (Timer1->Enabled)
  {
  Timer1->Enabled =false;
  BEmpfang->Caption="Read-Timer disabled";
  }
else
  {
  Timer1->Enabled=true;
  BEmpfang->Caption="Read-Timer aktiv";
  };
}
```

Buch-CD

Verzeichnis	Inhalt
Loesungen_CB2006	Die Lösungen nahezu aller Übungsaufgaben jeweils im Rahmen eines vollständigen C++Builder-Projekts. Die Unterverzeichnisse entsprechen den Buchkapiteln.
Buchtext	Der Buchtext als pdf-Datei
Boost	Die Boost-Bibliothek Version 1.34.1, siehe http://www.boost.org/ Hier finden Sie auch neuere Versionen und weitere Informationen. Bitte beachten Sie die Lizenzbedingungen (weitgehend freies Nutzungsrecht) http://www.boost.org/more/license_info.html

Index

C++-Schulungen
Workshops – Beratung – Coaching

Richard Kaiser führt seit über 20 Jahren Seminare über Programmiersprachen durch. Diese Seminare werden vor allem als Firmenseminare (inhouse) durchgeführt. Die Inhalte können an die Wünsche der Teilnehmer angepasst werden.

Im Vordergrund stehen dabei Zusammenhänge und Sprachkonzepte. Der Lehrstoff wird durch Übungen ergänzt, in denen die Teilnehmer praxisnahe Programme entwickeln.

►C++ mit dem C++Builder

Drei aufeinander abgestimmte Seminare: Behandelt wird der gesamte Sprachumfang des C++-Standards und die wichtigsten Komponenten des C++Builders.

1. C/C++ Grundlagen mit dem C++Builder
2. Objektorientierte Programmierung
3. Templates und die STL

►Microsoft Visual C++ 2005 und .NET

Fünf aufeinander abgestimmte Seminare: Behandelt wird der gesamte Sprachumfang des C++-Standards sowie die C++/CLI-Erweiterungen und die .NET Bibliothek.

1. Einführung in Visual Studio 2005
2. C/C++ Grundlagen
3. Objektorientierte Programmierung
4. C++/CLI-Erweiterungen und .NET-Klassen
5. Templates und die C++-Standardbibliothek

►Nähere Informationen: http://www.rkaiser.de/