

**Richard Kaiser**

*www.rkaiser.de*

# Lösungen

## zu den Übungsaufgaben in



3/22



# Vorwort

Der Ausgangspunkt für dieses Buch war der Wunsch nach einem C++-Lehrbuch, in dem von Anfang an Programme für eine grafische Windows-Benutzeroberfläche entwickelt werden, und nicht wie meist üblich Konsolen-Anwendungen. Programme, in denen Ein- und Ausgaben über eine Konsole erfolgen, sind für viele Anfänger weltfremd wie steinzeitliche DOS-Programme und schrecken davon ab, sich überhaupt mit C++ beschäftigen zu wollen.

Windows Forms Anwendungen sind ein idealer Rahmen für C++-Programme mit einer attraktiven Benutzeroberfläche: Der Zugriff auf Windows Steuerelemente (Buttons, TextBoxen usw.) ist einfach. Der Unterschied zu einem Standard C++ Programm ist meist nur, dass Ein- und Ausgaben über ein Windows-Steuerelement (meist eine TextBox) erfolgen

```
textBox1->AppendText("Hello World");
```

während in Standard-C++ die Konsole mit *cout* verwendet wird:

```
cout << "Hello world" << endl;
```

Aber nicht nur Studierende können von C++ mit einer grafischen Benutzeroberfläche profitieren. Mit Windows Forms Projekten können bestehende C oder C++-Programme ohne großen Aufwand mit einer grafischen Benutzeroberfläche ausgestattet werden. Und wer C oder C++ kann und keine neue Sprache für eine GUI lernen will, kann seine bestehenden Programme mit einfachen Mitteln schöner und leichter bedienbar machen.

C++ hat sich in den letzten Jahren rasant entwickelt: Die Neuerungen von C++11, C++14, C++17 und C++20 haben viele Verbesserungen und neue Möglichkeiten gebracht. Vieles, was 2010 noch gut und empfehlenswert war, kann heute besser und sicherer gemacht werden.

Das merkt man als Buchautor und Trainer, der diese ganze Entwicklung begleitet hat, besonders deutlich: Vieles, was man früher einmal geschrieben hat, sollte man heute anders machen. Zwar würde es immer noch kompiliert werden. Aber es ist kein modernes C++ mehr, das dem aktuellen Stand der Technik entspricht und alle Vorteile nutzt.

Dieses Buch stellt C++ für Windows Forms Anwendungen auf dem Stand von Visual Studio 2022 im März 2022 vor. Das ist der Umfang von C++17.

Dieses Buch entstand einerseits aus meinen Vorlesungen an der Dualen Hochschule Lörrach und andererseits aus zahlreichen Seminaren für Firmen, die sich an Software-Entwickler richten, die C++ professionell einsetzen. Dementsprechend richtet es sich einerseits an **Studierende ohne Vorkenntnisse**. Die Lernziele für diese Gruppe sind:

- Modernes C++ auf dem Stand von C++17 zu lernen. C++ ist nach wie vor eine der am häufigsten eingesetzten Programmiersprachen.
- Programmieren zu lernen, d.h. Programme zu schreiben, die konkrete, vorgegebene Aufgaben lösen. Das ist nur mit viel Übung möglich. Programmieren lernt man nur, indem man es tut. Deshalb enthält dieses Buch auch viele Übungsaufgaben. Es ist unerlässlich, zahlreiche Aufgaben selbständig zu lösen (das müssen nicht die aus diesem Buch sein).

Der Schwierigkeitsgrad der Aufgaben reicht von einfachen Wiederholungen des Textes bis zu kleinen Projektchen, die ein gewisses Maß an selbständiger Arbeit erfordern. Lösungen stehen auf [www.rkaiser.de](http://www.rkaiser.de) als Visual Studio Projekte zum Download bereit.

- Eine moderne Entwicklungsumgebung kennen- und einsetzen lernen. Visual Studio ist das in der Industrie wohl am häufigsten eingesetzte Werkzeug zur Software-Entwicklung.
- Man hört immer wieder, dass C++ zu schwierig ist, um damit das Programmieren zu lernen. Dieses Buch soll ein in vielen Jahren erprobtes Gegenargument zu dieser Meinung sein. Damit will ich die Komplexität von C++ nicht abstreiten. Aber wer C++ kann, findet sich leicht mit C#, Java usw. zurecht. Der umgekehrte Weg ist meist schwieriger.

Dieses Buch richtet sich aber ebenso an **professionelle Software-Entwickler** mit einer jahrelangen C++-Praxis. Viele C++-Neuerungen machen elementare Sprachkonzepte einfacher und sicherer. Dazu kommen anspruchsvollere Neuerungen, die bessere und effizientere Lösungen ermöglichen, die so vor einigen Jahren noch nicht möglich waren. Sowohl die einfachen als auch die anspruchsvolleren Möglichkeiten haben sich in der industriellen Praxis oft noch nicht herumgesprochen. Oft wird mit C++ noch wie vor 30 Jahren mit C programmiert. Das ist fehleranfällig, unnötig aufwendig und verschenkt Vorteile und Möglichkeiten.

- Für professionelle Entwickler soll dieses Buch einen nahezu vollständigen Überblick über C++17 geben und die Sprachelemente so vorstellen, dass sie in der praktischen Arbeit eingesetzt werden können. Die Erfahrungen aus meinen Firmenseminaren geben mir die Hoffnung, dass die allermeisten Themen abgedeckt sind, die im industriellen Einsatz notwendig sind.
- Da fast ausschließlich Standard-C++ behandelt wird, ist dieses Buch nicht auf Visual Studio beschränkt. Praktisch alle Ausführungen gelten für jeden standardkonformen C++-Compiler (gcc, Intel, Embarcadero usw.).
- Das moderne C++ bietet auch Programmierern von eingebetteten Anwendungen viele Vorteile gegenüber dem in diesem Anwendungsbereich verbreiteten C. Mit dem modernen C++ gibt es noch weniger Gründe als vorher, solche Anwendungen in C zu schreiben.
- Professionelle Entwickler sollen sich nicht an den Übungsaufgaben stören. Sie haben schon genügend Aufgaben gemacht, sonst wären sie keine Profis geworden. Und außerdem bietet das Tagesgeschäft genügend Übungsaufgaben.

Ich habe nicht nur versucht, die Sprachelemente anschaulich zu beschreiben, sondern auch Kriterien dafür anzugeben, wann und wie man sie sinnvoll einsetzen kann. Viele Empfehlungen aus style guides sind in den Text eingearbeitet, ohne dass explizit darauf hingewiesen wird.

Dieses Buch erscheint in zwei weitgehend identischen Ausgaben:

- **In der vorliegenden Ausgabe** werden Programme mit einer grafischen Benutzeroberfläche geschrieben. Alle Ein- und Ausgaben erfolgen über eine Windows-Benutzeroberfläche.
- **In der anderen Ausgabe**, „C++ mit Visual Studio 2019“ werden reine Standard-C++-Programme geschrieben, d.h. ohne graphische Benutzeroberfläche. Alle Ein- und Ausgaben erfolgen über die Konsole.

Der Unterschied zwischen den beiden Ausgaben ist meist nur, dass in „C++ mit Visual Studio 2022 und Windows Forms Anwendungen“ Ein- und Ausgaben über ein Windows-Steuerelement (meist eine TextBox) erfolgen, während in „C++ mit Visual Studio 2019“ die Konsole verwendet wird.

Dazu kommen noch etwa 100 Seiten mit unterschiedlichen Inhalten, da diese in Visual Studio nur für den jeweiligen Anwendungstyp verfügbar sind. Die folgenden Themen sind nur in „C++ mit Visual Studio 2019“ enthalten: 15.10 parallele Algorithmen der Standardbibliothek, 2.13 C++ Core Guideline Checks, 11.2 polymorphe Memory Ressourcen (pmr.), Sprachelemente von C++20: 2.14 Module, 14.6 concepts, 4.2.4 `std::span`, 8.2.5 der dreifache Vergleichsoperator `<=>` usw.

Dagegen sind diese Themen nur in der Ausgabe für Windows Forms Anwendungen enthalten: „Steuerelemente für die Benutzeroberfläche“ und „C++/CLI, .NET-Bibliotheken und C++ Interoperabilität“ (Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.** und **Fehler! Verweisquelle konnte nicht gefunden werden.**).

Die vorliegende Ausgabe unterscheidet sich von der Ausgabe für Visual Studio 2019 nur durch einige kleinere Ergänzungen, Korrekturen und das Kapitel über Multithreading. Die Ausgabe für Visual Studio 2019 ist aber eine umfassende Überarbeitung meiner Bücher über C++ mit Visual Studio 2017. Obwohl die 2019-er Ausgabe nur 2 Jahre später als die 2017-er Ausgabe erschienen ist, ist über die Hälfte komplett neu:

- Große Teile, die heute nicht mehr aktuell sind, wurden entfernt. Vieles wurde auf den aktuellen Stand des modernen C++ gebracht. Wichtige Erweiterungen kamen dazu.

- Eigentlich wollte ich den Umfang wieder auf ca. 800 Seiten begrenzen, um potenzielle Leser nicht schon allein durch das Gewicht und die Fülle des Stoffs abzuschrecken. Das ist mir aber trotz intensivster Bemühungen nicht gelungen.

Anregungen, Korrekturhinweise und Verbesserungsvorschläge sind willkommen. Bitte senden Sie diese an die Mail-Adresse auf meiner Internetseite [www.rkaiser.de](http://www.rkaiser.de).

Bei meinen Seminarteilnehmern und Studenten bedanke ich mich für die zahlreichen Anregungen. Dem Team vom Springer-Verlag danke ich für die Unterstützung und Geduld.

Tübingen, im März 2022

Richard Kaiser



# Inhalt

<b>1</b>	<b>Loesungen Kapitel 2. Windows Steuerelemente .....</b>	<b>11</b>
1.1	Aufgabe 2.1.....	12
1.2	Aufgabe 2.2.....	16
1.3	Aufgabe 2.3.....	20
1.4	Aufgabe 2.4.....	25
1.5	Aufgabe 2.5.....	28
1.6	Aufgabe 2.6.....	32
1.7	Aufgabe 2.7.....	36
1.8	Aufgabe 2.9.....	41
<b>2</b>	<b>Lösungen Kapitel 3. Elementare Datentypen.....</b>	<b>47</b>
2.1	Aufgaben 3.1 Syntaxregeln.....	48
2.2	Aufgaben 3.2 Variablen und Bezeichner .....	49
2.3	Aufgabe 3.3.2 Ganzzahl-literale.....	50
2.4	Aufgabe 3.3.6 Operatoren und die üblichen Konversionen .....	52
2.5	Aufgabe 3.3.8 Der Datentyp bool.....	53
2.6	Aufgabe 3.4 Kontrollstrukturen und Funktionen.....	55
2.7	Aufgabe 3.4.1 Die if- und die Verbundanweisung .....	56
2.8	Aufgabe 3.4.4 Ganzzahl-Funktionen.....	58
2.9	Aufgabe 3.4.7 Zufallszahlen .....	63
2.10	Aufgabe 3.4.8 Default-Argumente .....	64
2.11	Aufgabe 3.4.10 Programmierstil .....	65
2.12	Aufgabe 3.4.11 rekursive Funktionen.....	66
2.13	Aufgabe 3.4.12 switch .....	69
2.14	Aufgabe 3.5 Gleitkommatypen .....	70
2.15	Aufgabe 3.6.6 Systematisches Testen.....	73
2.16	Aufgabe 3.6.7 Unittests .....	75
2.17	Aufgabe 3.6.8 Ablaufprotokolle.....	77
2.18	Aufgabe 3.6.9 Symbolische Ablaufprotokolle .....	80
2.19	Aufgabe 3.9 Exception-Handling .....	82
2.20	Aufgabe 3.10 Namensbereiche.....	83
2.21	Aufgabe 3.11 Präprozessoranweisungen .....	84
<b>3</b>	<b>Lösungen Kapitel 4. String .....</b>	<b>85</b>
3.1	Aufgabe 4.3 Elementfunktionen.....	86
3.2	Aufgabe 4.5.1 Stringkonversionen .....	97
3.3	Aufgabe 4.8 Reguläre Ausdrücke.....	100
3.4	Form1.h für Kapitel 4 .....	105
<b>4</b>	<b>Lösungen Kapitel 5. Arrays und Container .....</b>	<b>113</b>
4.1	Aufgabe 5.2.3 Eindimensionale Arrays .....	114

4.2	Aufgabe 5.2.4 Die Initialisierung von Arrays bei ihrer Definition.....	119
4.3	Aufgabe 5.3 Arrays als Container .....	120
4.4	Aufgabe 5.4 Mehrdimensionale Arrays .....	121
4.5	Aufgabe 5.5 Dynamische Programmierung .....	122
4.6	Form1.h für Kapitel 5 .....	123
4.7	Aufgabe 5.3 Arrays als Container .....	128
4.8	Form1.h für Kapitel 5.4 .....	133
4.9	Aufgabe 6.3 Einfache Selbstdefinierte Datentypen .....	139
4.10	Aufgabe 6.1 Mit struct definierte Klassen .....	140
<b>5 Lösungen Kapitel 7. Zeiger.....</b>		<b>141</b>
5.1	Aufgabe 7.2 Adressoperator und Zeiger .....	142
5.2	Aufgabe 7.4 Ablaufprotokolle für Zeigervariable .....	143
5.3	Aufgabe 7.5 Dynamisch erzeugte eindimensionale Arrays.....	144
5.4	Aufgabe 7.6 Stringlitterale, nullterminierte Strings, .....	146
5.5	Aufgabe 7.7 Arrays, Zeiger und Zeigerarithmetik.....	149
5.6	Aufgabe 7.8 Verkettete Listen.....	150
5.7	Aufgabe 7.9 Binärbäume .....	161
<b>6 Lösungen Kapitel 8. Überladene Funktionen und Operatoren ....</b>		<b>165</b>
6.1	Aufgabe 8.2 Überladene Funktionen.....	166
6.2	Aufgabe 8.3.1 Globale Operatorfunktionen .....	168
6.3	Aufgabe 8.3.2 Ein-/Ausgabeoperatoren.....	171
6.4	Aufgabe 8.4 Referenztypen, Wert- und Ref-Parameter .....	172
<b>7 Lösungen Kapitel 9. Objektorientierte Programmierung .....</b>		<b>175</b>
7.1	Aufgabe 9.1.5 Klassen.....	176
7.2	Aufgabe 9.1.11 Typ-Inferenz .....	184
7.3	Aufgabe 9.2.2 Elementinitialisierer.....	186
7.4	Aufgabe 9.2.4 Überladene Operatoren .....	190
7.5	Aufgabe 9.2.6 Kopierkonstruktor und Zuweisungsoperator .....	192
7.6	Aufgabe 9.2.8 Konvertierende und explizite Konstruktoren.....	195
7.7	Aufgabe 9.2.10 Statische Klassenelemente.....	196
7.8	Aufgabe 9.2.12 Konstante Objekte und Elementfunktionen .....	197
7.9	Aufgabe 9.2.13 std::function.....	198
7.10	Aufgabe 9.2.15 Klassen und Header-Dateien .....	200
7.11	Aufgabe 9.3.4 Konstruktoren, Destruktoren und implizite Konversionen	201
7.12	Aufgabe 9.3.8 Konversionen zwischen Klassen.....	205
7.13	Aufgabe 9.3.9 Mehrfachvererbung.....	206
7.14	Aufgabe 9.4.3 Virtuelle Funktionen .....	209
<b>8 Lösungen Kapitel 10. Namensbereiche .....</b>		<b>213</b>
8.1	Aufgabe 10.5 Namensbereiche.....	214
<b>9 Lösungen Kapitel 11. Exception-Handling.....</b>		<b>217</b>
<b>10 Lösungen Kapitel 12. Containerklassen der Standardbibliothek.</b>		<b>227</b>
10.1	Aufgabe 12.1.5 Containerklassen.....	228
10.2	Aufgabe 12.1.6 Speicherverwaltung bei Vektoren .....	230
10.3	Aufgabe 12.1.7 Mehrdimensionale Vektoren.....	231
10.4	Aufgabe 12.2.3 Assoziative Container .....	232



## **11 Lösungen Kapitel 13. Dateibearbeitung mit den Streamklassen.237**

11.1 Aufgabe 13.3 Binärdateien .....	238
11.2 Aufgabe 13.4 Textdateien.....	242

## **12 Lösungen Kapitel 14. Funktoren, Funktionsobjekte und Lambda-Ausdrücke 245**

12.1 Aufgabe 14.2 Prädikate und Vergleichsfunktionen .....	246
12.2 Aufgabe 14.4 Lambda-Ausdrücke.....	249

## **13 Lösungen Kapitel 15. Templates .....251**

13.1 Aufgabe 15.1 Funktions-Templates.....	252
13.2 Aufgabe 15.2 Klassen-Templates.....	256
13.3 Aufgabe 15.4.3 type traits und static#assert.....	261
13.4 Aufgabe 15.5 Variadische Templates.....	262

## **14 Lösungen Kapitel 16. STL-Algorithmen .....263**

14.1 Aufgabe 16.1 Iteratoren.....	264
14.2 Aufgabe 16.8 STL-Algorithmen .....	268
14.3 Aufgabe 16.9.2 Valarrays .....	270
14.4 Aufgabe 16.9.4 Komplexe Zahlen.....	271

## **15 Lösungen Kapitel 16. Multithreading .....273**

15.1 Aufgabe 16.1.3 Funktionen als Threads starten .....	274
15.2 Aufgabe 16.1.4 Exceptions in Threads .....	276
15.3 Aufgabe 16.2.2 Kritische Bereiche mit Mutex und lockguard sperren..	277

## **16 Lösungen Kapitel 18. Smart Pointer .....281**

16.1 Aufgabe 18.3.....	282
------------------------	-----



# 1 Loesungen Kapitel 2. Windows Steuerelemente

```
// ##### Copyright notice #####
// Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
// "Richard Kaiser: C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen",
// ISBN Hardcover: 978-3-662-65256-5, ISBN eBook: 978-3-662-65257-2
//
// Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie den
// folgenden Hinweis enthalten:
// Copyright Richard Kaiser 2022, https://www.rkaiser.de/
// C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen"
// In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder
// Folgeschäden übernommen, die sich aus der Nutzung ergeben.
//
// It is permitted to use this source text in programs of your own, provided that it
// contains the two lines
// Copyright Richard Kaiser 2022, https://www.rkaiser.de/
// C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen"
// In no event shall liability be accepted for any direct, indirect, incidental or
// consequential damages arising from the use of these source texts.
// ##### end of Copyright notice #####
```

## 1.1 Aufgabe 2.1

```
// c:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesung_Kap_2_1\Form1.h

#pragma once
namespace CppCLRWinFormsProject {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Zusammenfassung für Form1
    ///
    /// Warnung: Wenn Sie den Namen dieser Klasse ändern, müssen Sie auch
    /// die Ressourcendateiname-Eigenschaft für das Tool zur Kompilierung verwalteter
    /// Ressourcen ändern,
    /// das allen RESX-Dateien zugewiesen ist, von denen diese Klasse abhängt.
    /// Anderenfalls können die Designer nicht korrekt mit den lokalisierten Ressourcen
    /// arbeiten, die diesem Formular zugewiesen sind.
    /// </summary>

    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO: Konstruktorcode hier hinzufügen.
            //
            textBoxVorname->Text = "Jonathan";
            textBoxNachname->Text = "Hacker";
            textBoxOW->Text = "zu schnell geparkt";
        }

    protected:
        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::Label^ labelVorname;
    private: System::Windows::Forms::Label^ labelNachname;
    private: System::Windows::Forms::Label^ labelBussgeld;
    private: System::Windows::Forms::Label^ labelLOW;
    private: System::Windows::Forms::Label^ labelMasseinheit;
    protected:

    protected:
    private: System::Windows::Forms::TextBox^ textBoxVorname;
    private: System::Windows::Forms::TextBox^ textBoxNachname;
    private: System::Windows::Forms::TextBox^ textBoxBussgeld;
    private: System::Windows::Forms::TextBox^ textBoxOW;

    private: System::Windows::Forms::Button^ buttonSpeichern;
    private: System::Windows::Forms::Button^ buttonLoeschen;
    private: System::Windows::Forms::Button^ buttonEnde;

```

```

private:
    /// <summary>
    /// Erforderliche Designervariable.
    /// </summary>
    System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Erforderliche Methode für die Designerunterstützung.
    /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
    /// </summary>
    void InitializeComponent(void)
    {
        this->labelVorname = (gcnew System::Windows::Forms::Label());
        this->labelNachname = (gcnew System::Windows::Forms::Label());
        this->labelBussgeld = (gcnew System::Windows::Forms::Label());
        this->labelOW = (gcnew System::Windows::Forms::Label());
        this->labelMasseinheit = (gcnew System::Windows::Forms::Label());
        this->textBoxVorname = (gcnew System::Windows::Forms::TextBox());
        this->textBoxNachname = (gcnew System::Windows::Forms::TextBox());
        this->textBoxBussgeld = (gcnew System::Windows::Forms::TextBox());
        this->textBoxOW = (gcnew System::Windows::Forms::TextBox());
        this->buttonSpeichern = (gcnew System::Windows::Forms::Button());
        this->buttonLoeschen = (gcnew System::Windows::Forms::Button());
        this->buttonEnde = (gcnew System::Windows::Forms::Button());
        this->SuspendLayout();
        //
        // labelVorname
        //
        this->labelVorname->AutoSize = true;
        this->labelVorname->Location = System::Drawing::Point(20, 26);
        this->labelVorname->Name = L"labelVorname";
        this->labelVorname->Size = System::Drawing::Size(49, 13);
        this->labelVorname->TabIndex = 0;
        this->labelVorname->Text = L"Vorname";
        //
        // labelNachname
        //
        this->labelNachname->AutoSize = true;
        this->labelNachname->Location = System::Drawing::Point(20, 57);
        this->labelNachname->Name = L"labelNachname";
        this->labelNachname->Size = System::Drawing::Size(59, 13);
        this->labelNachname->TabIndex = 1;
        this->labelNachname->Text = L"Nachname";
        //
        // labelBussgeld
        //
        this->labelBussgeld->AutoSize = true;
        this->labelBussgeld->Location = System::Drawing::Point(20, 88);
        this->labelBussgeld->Name = L"labelBussgeld";
        this->labelBussgeld->Size = System::Drawing::Size(50, 13);
        this->labelBussgeld->TabIndex = 2;
        this->labelBussgeld->Text = L"Bussgeld";
        //
        // labelOW
        //
        this->labelOW->AutoSize = true;
        this->labelOW->Location = System::Drawing::Point(20, 119);
        this->labelOW->Name = L"labelOW";
        this->labelOW->Size = System::Drawing::Size(97, 13);
        this->labelOW->TabIndex = 3;
        this->labelOW->Text = L"Ordnungswidrigkeit";
        //
        // labelMasseinheit
        //
        this->labelMasseinheit->AutoSize = true;
        this->labelMasseinheit->Location = System::Drawing::Point(233, 92);
        this->labelMasseinheit->Name = L"labelMasseinheit";
    }

```

```

this->labelMasseinheit->Size = System::Drawing::Size(40, 13);
this->labelMasseinheit->TabIndex = 4;
this->labelMasseinheit->Text = L"€";
//
// textBoxVorname
//
this->textBoxVorname->Location = System::Drawing::Point(132, 23);
this->textBoxVorname->Name = L"textBoxVorname";
this->textBoxVorname->Size = System::Drawing::Size(100, 20);
this->textBoxVorname->TabIndex = 5;
//
// textBoxNachname
//
this->textBoxNachname->Location = System::Drawing::Point(132, 54);
this->textBoxNachname->Name = L"textBoxNachname";
this->textBoxNachname->Size = System::Drawing::Size(100, 20);
this->textBoxNachname->TabIndex = 6;
//
// textBoxBussgeld
//
this->textBoxBussgeld->Location = System::Drawing::Point(132, 85);
this->textBoxBussgeld->Name = L"textBoxBussgeld";
this->textBoxBussgeld->Size = System::Drawing::Size(100, 20);
this->textBoxBussgeld->TabIndex = 7;
//
// textBoxOW
//
this->textBoxOW->Location = System::Drawing::Point(132, 116);
this->textBoxOW->Name = L"textBoxOW";
this->textBoxOW->Size = System::Drawing::Size(100, 20);
this->textBoxOW->TabIndex = 8;
//
// buttonSpeichern
//
this->buttonSpeichern->Location = System::Drawing::Point(23, 160);
this->buttonSpeichern->Name = L"buttonSpeichern";
this->buttonSpeichern->Size = System::Drawing::Size(94, 23);
this->buttonSpeichern->TabIndex = 9;
this->buttonSpeichern->Text = L"Daten speichern";
this->buttonSpeichern->UseVisualStyleBackColor = true;
this->buttonSpeichern->Click += gcnew System::EventHandler(this,
&Form1::buttonSpeichern_Click);
//
// buttonLoeschen
//
this->buttonLoeschen->Location = System::Drawing::Point(127, 160);
this->buttonLoeschen->Name = L"buttonLoeschen";
this->buttonLoeschen->Size = System::Drawing::Size(98, 23);
this->buttonLoeschen->TabIndex = 10;
this->buttonLoeschen->Text = L"Eingabe löschen";
this->buttonLoeschen->UseVisualStyleBackColor = true;
this->buttonLoeschen->Click += gcnew System::EventHandler(this,
&Form1::buttonLoeschen_Click);
//
// buttonEnde
//
this->buttonEnde->Location = System::Drawing::Point(235, 160);
this->buttonEnde->Name = L"buttonEnde";
this->buttonEnde->Size = System::Drawing::Size(111, 23);
this->buttonEnde->TabIndex = 11;
this->buttonEnde->Text = L"Programm beenden";
this->buttonEnde->UseVisualStyleBackColor = true;
this->buttonEnde->Click += gcnew System::EventHandler(this, &Form1::button3_Click);
//
// Form1
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(408, 194);

```

```
        this->Controls->Add(this->buttonEnde);
        this->Controls->Add(this->buttonLoeschen);
        this->Controls->Add(this->buttonSpeichern);
        this->Controls->Add(this->textBoxOW);
        this->Controls->Add(this->textBoxBussgeld);
        this->Controls->Add(this->textBoxNachname);
        this->Controls->Add(this->textBoxVorname);
        this->Controls->Add(this->labelMasseinheit);
        this->Controls->Add(this->labelOW);
        this->Controls->Add(this->labelBussgeld);
        this->Controls->Add(this->labelNachname);
        this->Controls->Add(this->labelVorname);
        this->Name = L"Form1";
        this->Text = L"Das städtische Haushaltssanierungsprogramm";
        this->ResumeLayout(false);
        this->PerformLayout();
    }
#pragma endregion
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
{
    Close();
}

private: System::Void buttonLoeschen_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBoxVorname->Clear();
    textBoxNachname->Clear();
    textBoxBussgeld->Clear();
    textBoxOW->Clear();
}

private: System::Void buttonSpeichern_Click(System::Object^ sender, System::EventArgs^ e)
{
}

};
}
```

## 1.2 Aufgabe 2.2

```
// c:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesung_Kap_2_2\Form1.h

#pragma once
namespace CppCLRWinFormsProject {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Zusammenfassung für Form1
    ///
    /// Warnung: Wenn Sie den Namen dieser Klasse ändern, müssen Sie auch
    /// die Ressourcendateiname-Eigenschaft für das Tool zur Kompilierung verwalteter
    /// Ressourcen ändern,
    /// das allen RESX-Dateien zugewiesen ist, von denen diese Klasse abhängt.
    /// Anderenfalls können die Designer nicht korrekt mit den lokalisierten Ressourcen
    /// arbeiten, die diesem Formular zugewiesen sind.
    /// </summary>

    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO: Konstruktorcode hier hinzufügen.
            //
        }

    protected:
        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }
    private: System::Windows::Forms::Label^ label1;
    protected:
    private: System::Windows::Forms::Button^ buttonLinksAusrichten;
    private: System::Windows::Forms::Button^ buttonRechtsAusrichten;
    private: System::Windows::Forms::Button^ buttonUnsichtbar;
    private: System::Windows::Forms::Button^ buttonSichtbar;
    private: System::Windows::Forms::Button^ buttonLinks;
    private: System::Windows::Forms::Button^ buttonRechts;

    private:
        /// <summary>
        /// Erforderliche Designervariable.
        /// </summary>
        System::ComponentModel::Container ^components;
    }
#pragma region Windows Form Designer generated code
    /// <summary>
    /// Erforderliche Methode für die Designerunterstützung.
    /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
    /// </summary>
    void InitializeComponent(void)

```



```

{
    this->label1 = (gcnew System::Windows::Forms::Label());
    this->buttonLinksAusrichten = (gcnew System::Windows::Forms::Button());
    this->buttonRechtsAusrichten = (gcnew System::Windows::Forms::Button());
    this->buttonUnsichtbar = (gcnew System::Windows::Forms::Button());
    this->buttonSichtbar = (gcnew System::Windows::Forms::Button());
    this->buttonLinks = (gcnew System::Windows::Forms::Button());
    this->buttonRechts = (gcnew System::Windows::Forms::Button());
    this->SuspendLayout();
    //
    // label1
    //
    this->label1->BackColor = System::Drawing::Color::Yellow;
    this->label1->Location = System::Drawing::Point(24, 32);
    this->label1->Name = L"label1";
    this->label1->Size = System::Drawing::Size(245, 18);
    this->label1->TabIndex = 0;
    this->label1->Text = L"label1";
    //
    // buttonLinksAusrichten
    //
    this->buttonLinksAusrichten->Location = System::Drawing::Point(24, 69);
    this->buttonLinksAusrichten->Name = L"buttonLinksAusrichten";
    this->buttonLinksAusrichten->Size = System::Drawing::Size(93, 23);
    this->buttonLinksAusrichten->TabIndex = 1;
    this->buttonLinksAusrichten->Text = L"links ausrichten";
    this->buttonLinksAusrichten->UseVisualStyleBackColor = true;
    this->buttonLinksAusrichten->Click += gcnew System::EventHandler(this,
&Form1::buttonLinksAusrichten_Click);
    //
    // buttonRechtsAusrichten
    //
    this->buttonRechtsAusrichten->Location = System::Drawing::Point(168, 69);
    this->buttonRechtsAusrichten->Name = L"buttonRechtsAusrichten";
    this->buttonRechtsAusrichten->Size = System::Drawing::Size(101, 23);
    this->buttonRechtsAusrichten->TabIndex = 2;
    this->buttonRechtsAusrichten->Text = L"rechts ausrichten";
    this->buttonRechtsAusrichten->UseVisualStyleBackColor = true;
    this->buttonRechtsAusrichten->Click += gcnew System::EventHandler(this,
&Form1::buttonRechtsAusrichten_Click);
    //
    // buttonUnsichtbar
    //
    this->buttonUnsichtbar->Location = System::Drawing::Point(24, 115);
    this->buttonUnsichtbar->Name = L"buttonUnsichtbar";
    this->buttonUnsichtbar->Size = System::Drawing::Size(75, 23);
    this->buttonUnsichtbar->TabIndex = 3;
    this->buttonUnsichtbar->Text = L"unsichtbar";
    this->buttonUnsichtbar->UseVisualStyleBackColor = true;
    this->buttonUnsichtbar->Click += gcnew System::EventHandler(this,
&Form1::buttonUnsichtbar_Click);
    //
    // buttonSichtbar
    //
    this->buttonSichtbar->Location = System::Drawing::Point(194, 114);
    this->buttonSichtbar->Name = L"buttonSichtbar";
    this->buttonSichtbar->Size = System::Drawing::Size(75, 23);
    this->buttonSichtbar->TabIndex = 4;
    this->buttonSichtbar->Text = L"sichtbar";
    this->buttonSichtbar->UseVisualStyleBackColor = true;
    this->buttonSichtbar->Click += gcnew System::EventHandler(this,
&Form1::buttonSichtbar_Click);
    //
    // buttonLinks
    //
    this->buttonLinks->Location = System::Drawing::Point(24, 159);
    this->buttonLinks->Name = L"buttonLinks";
    this->buttonLinks->Size = System::Drawing::Size(75, 23);
    this->buttonLinks->TabIndex = 5;
}

```

```

this->buttonLinks->Text = L"links";
this->buttonLinks->UseVisualStyleBackColor = true;
this->buttonLinks->Click += gcnew System::EventHandler(this, &Form1::buttonLinks_Click);
//
// buttonRechts
//
this->buttonRechts->Location = System::Drawing::Point(194, 159);
this->buttonRechts->Name = L"buttonRechts";
this->buttonRechts->Size = System::Drawing::Size(75, 23);
this->buttonRechts->TabIndex = 6;
this->buttonRechts->Text = L"rechts";
this->buttonRechts->UseVisualStyleBackColor = true;
this->buttonRechts->Click += gcnew System::EventHandler(this, &Form1::buttonRechts_Click);
//
// Form1
//
this->AutoSizeDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(292, 193);
this->Controls->Add(this->buttonRechts);
this->Controls->Add(this->buttonLinks);
this->Controls->Add(this->buttonSichtbar);
this->Controls->Add(this->buttonUnsichtbar);
this->Controls->Add(this->buttonRechtsAusrichten);
this->Controls->Add(this->buttonLinksAusrichten);
this->Controls->Add(this->label1);
this->Name = L"Form1";
this->Text = L"Form1";
this->ResumeLayout(false);
}
#pragma endregion

private: System::Void buttonLinksAusrichten_Click(System::Object^ sender,
System::EventArgs^ e)
{
    label1->TextAlign = ContentAlignment::MiddleLeft;
}

private: System::Void buttonRechtsAusrichten_Click(System::Object^ sender,
System::EventArgs^ e)
{
    label1->TextAlign = ContentAlignment::MiddleRight;
}

private: System::Void buttonUnsichtbar_Click(System::Object^ sender,
System::EventArgs^ e)
{
    label1->Visible = false;
}

private: System::Void buttonSichtbar_Click(System::Object^ sender,
System::EventArgs^ e)
{
    label1->Visible = true;
}

private: System::Void buttonLinks_Click(System::Object^ sender,
System::EventArgs^ e)
{
    label1->Left = 0;
}

private: System::Void buttonRechts_Click(System::Object^ sender,
System::EventArgs^ e)
{
    label1->Left = this->ClientRectangle.Width - label1->Width;
}

```

```
private: System::Void buttonZusatzaufgabe_Click(System::Object^ sender,
System::EventArgs^ e)
{
    // Zusatzaufgabe:
    label1->Left = this->ClientRectangle.Width - label1->Width;
    // oder auch
    // label1->Left = this->ClientSize.Width - label1->Width;
}
};
}
```

### 1.3 Aufgabe 2.3

```
// c:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesung_Kap_2_3\Form1.h
```

```
#pragma once
```

```
namespace CppCLRWinFormsProject {
```

```
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
```

```
/// <summary>
/// Zusammenfassung für Form1
///
/// Warnung: Wenn Sie den Namen dieser Klasse ändern, müssen Sie auch
/// die Ressourcendateiname-Eigenschaft für das Tool zur Kompilierung verwalteter
Ressourcen ändern,
/// das allen RESX-Dateien zugewiesen ist, von denen diese Klasse abhängt.
/// Anderenfalls können die Designer nicht korrekt mit den lokalisierten Ressourcen
/// arbeiten, die diesem Formular zugewiesen sind.
/// </summary>
```

```
public ref class Form1 : public System::Windows::Forms::Form
```

```
{
public:
    Form1(void)
    {
        InitializeComponent();
        //
        //TODO: Konstruktorcode hier hinzufügen.
        //
    }
}
```

```
protected:
    /// <summary>
    /// Verwendete Ressourcen bereinigen.
    /// </summary>
    ~Form1()
    {
        if (components)
        {
            delete components;
        }
    }
}
```

```
private: System::Windows::Forms::TextBox^ textBox1;
protected:
private: System::Windows::Forms::TextBox^ textBox2;
private: System::Windows::Forms::TextBox^ textBox3;
private: System::Windows::Forms::Button^ AddInt;
private: System::Windows::Forms::Button^ buttonClear;
private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::Button^ button1;
private: System::Windows::Forms::Button^ button2;
private: System::Windows::Forms::Button^ button3;
private: System::Windows::Forms::Button^ button4;
private: System::Windows::Forms::Button^ button5;
private: System::Windows::Forms::TextBox^ textBox4;
```

```
private:
    /// <summary>
    /// Erforderliche Designervariable.
    /// </summary>
```

```
System::ComponentModel::Container ^components;
```

```
#pragma region Windows Form Designer generated code
/// <summary>
/// Erforderliche Methode für die Designerunterstützung.
/// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
/// </summary>
void InitializeComponent(void)
{
    this->textBox1 = (gcnew System::Windows::Forms::TextBox());
    this->textBox2 = (gcnew System::Windows::Forms::TextBox());
    this->textBox3 = (gcnew System::Windows::Forms::TextBox());
    this->AddInt = (gcnew System::Windows::Forms::Button());
    this->buttonClear = (gcnew System::Windows::Forms::Button());
    this->label1 = (gcnew System::Windows::Forms::Label());
    this->label2 = (gcnew System::Windows::Forms::Label());
    this->button1 = (gcnew System::Windows::Forms::Button());
    this->button2 = (gcnew System::Windows::Forms::Button());
    this->button3 = (gcnew System::Windows::Forms::Button());
    this->button4 = (gcnew System::Windows::Forms::Button());
    this->button5 = (gcnew System::Windows::Forms::Button());
    this->textBox4 = (gcnew System::Windows::Forms::TextBox());
    this->SuspendLayout();
    //
    // textBox1
    //
    this->textBox1->Location = System::Drawing::Point(12, 28);
    this->textBox1->Name = L"textBox1";
    this->textBox1->Size = System::Drawing::Size(100, 20);
    this->textBox1->TabIndex = 0;
    //
    // textBox2
    //
    this->textBox2->Location = System::Drawing::Point(155, 28);
    this->textBox2->Name = L"textBox2";
    this->textBox2->Size = System::Drawing::Size(100, 20);
    this->textBox2->TabIndex = 1;
    //
    // textBox3
    //
    this->textBox3->Location = System::Drawing::Point(321, 28);
    this->textBox3->Name = L"textBox3";
    this->textBox3->Size = System::Drawing::Size(100, 20);
    this->textBox3->TabIndex = 2;
    //
    // AddInt
    //
    this->AddInt->Location = System::Drawing::Point(27, 73);
    this->AddInt->Name = L"AddInt";
    this->AddInt->Size = System::Drawing::Size(75, 23);
    this->AddInt->TabIndex = 3;
    this->AddInt->Text = L"add int";
    this->AddInt->UseVisualStyleBackColor = true;
    this->AddInt->Click += gcnew System::EventHandler(this, &Form1::AddInt_Click);
    //
    // buttonClear
    //
    this->buttonClear->Location = System::Drawing::Point(346, 73);
    this->buttonClear->Name = L"buttonClear";
    this->buttonClear->Size = System::Drawing::Size(75, 23);
    this->buttonClear->TabIndex = 4;
    this->buttonClear->Text = L"clear";
    this->buttonClear->UseVisualStyleBackColor = true;
    this->buttonClear->Click += gcnew System::EventHandler(this, &Form1::buttonClear_Click);
    //
    // label1
    //
    this->label1->AutoSize = true;
    this->label1->Location = System::Drawing::Point(127, 31);
```

```

this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(13, 13);
this->label1->TabIndex = 5;
this->label1->Text = L"+";
//
// label2
//
this->label2->AutoSize = true;
this->label2->Location = System::Drawing::Point(281, 31);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(13, 13);
this->label2->TabIndex = 6;
this->label2->Text = L="";
//
// button1
//
this->button1->Location = System::Drawing::Point(165, 73);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(75, 23);
this->button1->TabIndex = 7;
this->button1->Text = L"add double";
this->button1->UseVisualStyleBackColor = true;
this->button1->Click += gcnew System::EventHandler(this, &Form1::button1_Click);
//
// button2
//
this->button2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
this->button2->Location = System::Drawing::Point(43, 119);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(26, 34);
this->button2->TabIndex = 8;
this->button2->Text = L"+";
this->button2->UseVisualStyleBackColor = true;
this->button2->Click += gcnew System::EventHandler(this, &Form1::button2_Click);
//
// button3
//
this->button3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
this->button3->Location = System::Drawing::Point(150, 119);
this->button3->Name = L"button3";
this->button3->Size = System::Drawing::Size(26, 34);
this->button3->TabIndex = 9;
this->button3->Text = L"-";
this->button3->UseVisualStyleBackColor = true;
this->button3->Click += gcnew System::EventHandler(this, &Form1::button3_Click);
//
// button4
//
this->button4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
this->button4->Location = System::Drawing::Point(257, 119);
this->button4->Name = L"button4";
this->button4->Size = System::Drawing::Size(26, 34);
this->button4->TabIndex = 10;
this->button4->Text = L"*";
this->button4->UseVisualStyleBackColor = true;
this->button4->Click += gcnew System::EventHandler(this, &Form1::button4_Click);
//
// button5
//
this->button5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
this->button5->Location = System::Drawing::Point(364, 119);

```

```

    this->button5->Name = L"button5";
    this->button5->Size = System::Drawing::Size(26, 34);
    this->button5->TabIndex = 11;
    this->button5->Text = L"/";
    this->button5->UseVisualStyleBackColor = true;
    this->button5->Click += gcnew System::EventHandler(this, &Form1::button5_Click);
    //
    // textBox4
    //
    this->textBox4->Location = System::Drawing::Point(46, 159);
    this->textBox4->Multiline = true;
    this->textBox4->Name = L"textBox4";
    this->textBox4->Size = System::Drawing::Size(344, 73);
    this->textBox4->TabIndex = 12;
    //
    // Form1
    //
    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
    this->ClientSize = System::Drawing::Size(433, 234);
    this->Controls->Add(this->textBox4);
    this->Controls->Add(this->button5);
    this->Controls->Add(this->button4);
    this->Controls->Add(this->button3);
    this->Controls->Add(this->button2);
    this->Controls->Add(this->button1);
    this->Controls->Add(this->label2);
    this->Controls->Add(this->label1);
    this->Controls->Add(this->buttonClear);
    this->Controls->Add(this->AddInt);
    this->Controls->Add(this->textBox3);
    this->Controls->Add(this->textBox2);
    this->Controls->Add(this->textBox1);
    this->Name = L"Form1";
    this->Text = L"Einfacher Rechner";
    this->ResumeLayout(false);
    this->PerformLayout();
}
#pragma endregion

private: System::Void AddInt_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox3->Text = (Convert::ToInt32(textBox1->Text) +
        Convert::ToInt32(textBox2->Text)).ToString();
}

private: System::Void buttonClear_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox1->Clear();
    textBox2->Clear();
    textBox3->Clear();
}

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox3->Text = (Convert::ToDouble(textBox1->Text) +
        Convert::ToDouble(textBox2->Text)).ToString();
}

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox3->Text = (Convert::ToDouble(textBox1->Text) +
        Convert::ToDouble(textBox2->Text)).ToString();
    label1->Text = "+";
    textBox4->AppendText(String::Format("{0}+{1}={2}\r\n", textBox1->Text, textBox2->Text,
    textBox3->Text));
}

```

```
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox3->Text = (Convert::ToDouble(textBox1->Text) -
        Convert::ToDouble(textBox2->Text)).ToString();
    label1->Text = "-";
    textBox4->AppendText(String::Format("{0}-{1}={2}\r\n", textBox1->Text, textBox2->Text,
textBox3->Text));
}

private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox3->Text = (Convert::ToDouble(textBox1->Text) *
        Convert::ToDouble(textBox2->Text)).ToString();
    label1->Text = "*";
    textBox4->AppendText(String::Format("{0}*{1}={2}\r\n", textBox1->Text, textBox2->Text,
textBox3->Text));
}

private: System::Void button5_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox3->Text = (Convert::ToDouble(textBox1->Text) /
        Convert::ToDouble(textBox2->Text)).ToString();
    label1->Text = "/";
    textBox4->AppendText(String::Format("{0}/{1}={2}\r\n", textBox1->Text, textBox2->Text,
textBox3->Text));
}

};
}
```



## 1.4 Aufgabe 2.4

```
// c:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesung_Kap_2_4\Form1.h

#pragma once

namespace CppCLRWinFormsProject {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Zusammenfassung für Form1
    ///
    /// Warnung: Wenn Sie den Namen dieser Klasse ändern, müssen Sie auch
    /// die Ressourcendateiname-Eigenschaft für das Tool zur Kompilierung verwalteter
    Ressourcen ändern,
    /// das allen RESX-Dateien zugewiesen ist, von denen diese Klasse abhängt.
    /// Anderenfalls können die Designer nicht korrekt mit den lokalisierten Ressourcen
    /// arbeiten, die diesem Formular zugewiesen sind.
    /// </summary>

    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO: Konstruktorcode hier hinzufügen.
            //
        }

    protected:
        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }

    protected:
    private: System::Windows::Forms::ListBox^ listBox1;
    private: System::Windows::Forms::Button^ button1;
    private: System::Windows::Forms::Button^ button2;

    private: System::Windows::Forms::ComboBox^ comboBox1;
    private: System::Windows::Forms::TextBox^ textBox1;

    private: System::Windows::Forms::Label^ label1;
    private: System::Windows::Forms::Label^ label2;

    private:
        /// <summary>
        /// Erforderliche Designervariable.
        /// </summary>
        System::ComponentModel::Container ^components;
    
```

```

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Erforderliche Methode für die Designerunterstützung.
    /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
    /// </summary>
    void InitializeComponent(void)
    {
        this->listBox1 = (gcnew System::Windows::Forms::ListBox());
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->button2 = (gcnew System::Windows::Forms::Button());
        this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->label2 = (gcnew System::Windows::Forms::Label());
        this->SuspendLayout();
        //
        // listBox1
        //
        this->listBox1->FormattingEnabled = true;
        this->listBox1->Location = System::Drawing::Point(12, 12);
        this->listBox1->Name = L"listBox1";
        this->listBox1->Size = System::Drawing::Size(125, 121);
        this->listBox1->TabIndex = 1;
        this->listBox1->SelectedIndexChanged += gcnew System::EventHandler(this,
&Form1::listBox1_SelectedIndexChanged);
        //
        // button1
        //
        this->button1->Location = System::Drawing::Point(12, 168);
        this->button1->Name = L"button1";
        this->button1->Size = System::Drawing::Size(75, 23);
        this->button1->TabIndex = 2;
        this->button1->Text = L"button1";
        this->button1->UseVisualStyleBackColor = true;
        this->button1->Click += gcnew System::EventHandler(this, &Form1::button1_Click);
        //
        // button2
        //
        this->button2->Location = System::Drawing::Point(12, 139);
        this->button2->Name = L"button2";
        this->button2->Size = System::Drawing::Size(75, 23);
        this->button2->TabIndex = 3;
        this->button2->Text = L"button2";
        this->button2->UseVisualStyleBackColor = true;
        this->button2->Click += gcnew System::EventHandler(this, &Form1::button2_Click);
        //
        // comboBox1
        //
        this->comboBox1->FormattingEnabled = true;
        this->comboBox1->Location = System::Drawing::Point(98, 141);
        this->comboBox1->Name = L"comboBox1";
        this->comboBox1->Size = System::Drawing::Size(121, 21);
        this->comboBox1->TabIndex = 4;
        //
        // textBox1
        //
        this->textBox1->Location = System::Drawing::Point(98, 171);
        this->textBox1->Name = L"textBox1";
        this->textBox1->Size = System::Drawing::Size(100, 20);
        this->textBox1->TabIndex = 5;
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->Location = System::Drawing::Point(163, 21);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(35, 13);
        this->label1->TabIndex = 6;
    }

```

```

    this->label1->Text = L"label1";
    //
    // label2
    //
    this->label2->AutoSize = true;
    this->label2->Location = System::Drawing::Point(236, 149);
    this->label2->Name = L"label2";
    this->label2->Size = System::Drawing::Size(35, 13);
    this->label2->TabIndex = 7;
    this->label2->Text = L"label2";
    //
    // Form1
    //
    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
    this->ClientSize = System::Drawing::Size(307, 201);
    this->Controls->Add(this->label2);
    this->Controls->Add(this->label1);
    this->Controls->Add(this->textBox1);
    this->Controls->Add(this->comboBox1);
    this->Controls->Add(this->button2);
    this->Controls->Add(this->button1);
    this->Controls->Add(this->listBox1);
    this->Name = L"Form1";
    this->Text = L"Form1";
    this->ResumeLayout(false);
    this->PerformLayout();
}
#pragma endregion

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
    listBox1->Items->Add(textBox1->Text);
    comboBox1->Items->Add(textBox1->Text);
}

private: System::Void listBox1_SelectedIndexChanged(System::Object^ sender, System::EventArgs^
e)
{
    label1->Text = listBox1->Text;
    // oder
    // label1->Text=listBox1->SelectedItem->ToString();
}

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
{
    label2->Text = comboBox1->Text;
}

};
}

```

## 1.5 Aufgabe 2.5

```
// c:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesung_Kap_2_5\Form1.h

#pragma once

namespace CppCLRWinFormsProject {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Zusammenfassung für Form1
    ///
    /// Warnung: Wenn Sie den Namen dieser Klasse ändern, müssen Sie auch
    /// die Ressourcendateiname-Eigenschaft für das Tool zur Kompilierung verwalteter
    /// Ressourcen ändern,
    /// das allen RESX-Dateien zugewiesen ist, von denen diese Klasse abhängt.
    /// Anderenfalls können die Designer nicht korrekt mit den lokalisierten Ressourcen
    /// arbeiten, die diesem Formular zugewiesen sind.
    /// </summary>

    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void) // der sogenannte Konstruktor, der
        { // beim Erzeugen des Formulars aufgerufen wird
            InitializeComponent();
            //
            //TODO: Konstruktorcode hier hinzufügen.
            //
            Text = "Events";
        }

    protected:
        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::TextBox^ textBox1;
    private: System::Windows::Forms::Button^ Test_button;
    private: System::Windows::Forms::Button^ Clear_button;
    private: System::Windows::Forms::Button^ Jump_button;
    private: System::Windows::Forms::Label^ label1;

    private:
        /// <summary>
        /// Erforderliche Designervariable.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Erforderliche Methode für die Designerunterstützung.
        /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
        /// </summary>

```

```

void InitializeComponent(void)
{
    this->textBox1 = (gcnew System::Windows::Forms::TextBox());
    this->Test_button = (gcnew System::Windows::Forms::Button());
    this->Clear_button = (gcnew System::Windows::Forms::Button());
    this->Jump_button = (gcnew System::Windows::Forms::Button());
    this->label1 = (gcnew System::Windows::Forms::Label());
    this->SuspendLayout();
    //
    // textBox1
    //
    this->textBox1->Location = System::Drawing::Point(2, 2);
    this->textBox1->Multiline = true;
    this->textBox1->Name = L"textBox1";
    this->textBox1->Size = System::Drawing::Size(207, 194);
    this->textBox1->TabIndex = 9;
    //
    // Test_button
    //
    this->Test_button->Location = System::Drawing::Point(229, 2);
    this->Test_button->Name = L"Test_button";
    this->Test_button->Size = System::Drawing::Size(75, 23);
    this->Test_button->TabIndex = 18;
    this->Test_button->Text = L"Test";
    this->Test_button->UseVisualStyleBackColor = true;
    this->Test_button->Click += gcnew System::EventHandler(this, &Form1::Test_button_Click);
    this->Test_button->Enter += gcnew System::EventHandler(this, &Form1::Test_button_Enter);
    this->Test_button->KeyDown += gcnew System::Windows::Forms::KeyEventHandler(this,
&Form1::Test_button_KeyDown);
    this->Test_button->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this,
&Form1::Test_button_KeyPress);
    this->Test_button->KeyUp += gcnew System::Windows::Forms::KeyEventHandler(this,
&Form1::Test_button_KeyUp);
    this->Test_button->Leave += gcnew System::EventHandler(this, &Form1::Test_button_Leave);
    this->Test_button->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this,
&Form1::Test_button_MouseDown);
    this->Test_button->MouseMove += gcnew System::Windows::Forms::MouseEventHandler(this,
&Form1::Test_button_MouseMove);
    this->Test_button->MouseUp += gcnew System::Windows::Forms::MouseEventHandler(this,
&Form1::Test_button_MouseUp);
    //
    // Clear_button
    //
    this->Clear_button->Location = System::Drawing::Point(229, 42);
    this->Clear_button->Name = L"Clear_button";
    this->Clear_button->Size = System::Drawing::Size(75, 23);
    this->Clear_button->TabIndex = 19;
    this->Clear_button->Text = L"Clear";
    this->Clear_button->UseVisualStyleBackColor = true;
    this->Clear_button->Click += gcnew System::EventHandler(this,
&Form1::Clear_button_Click);
    //
    // Jump_button
    //
    this->Jump_button->Location = System::Drawing::Point(228, 196);
    this->Jump_button->Name = L"Jump_button";
    this->Jump_button->Size = System::Drawing::Size(75, 23);
    this->Jump_button->TabIndex = 20;
    this->Jump_button->Text = L"Jump";
    this->Jump_button->UseVisualStyleBackColor = true;
    this->Jump_button->Click += gcnew System::EventHandler(this, &Form1::Jump_button_Click);
    this->Jump_button->MouseMove += gcnew System::Windows::Forms::MouseEventHandler(this,
&Form1::Jump_button_MouseMove);
    //
    // label1
    //
    this->label1->AutoSize = true;
    this->label1->Location = System::Drawing::Point(237, 83);
    this->label1->Name = L"label1";
}

```

```

this->label1->Size = System::Drawing::Size(35, 13);
this->label1->TabIndex = 21;
this->label1->Text = L"label1";
//
// Form1
//
this->AutoScaledDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(315, 223);
this->Controls->Add(this->label1);
this->Controls->Add(this->Jump_button);
this->Controls->Add(this->Clear_button);
this->Controls->Add(this->Test_button);
this->Controls->Add(this->textBox1);
this->Name = L"Form1";
this->Text = L"Form1";
this->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion

private: System::Void Test_button_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox1->AppendText("Click\r\n");
}

private: System::Void Test_button_Enter(System::Object^ sender, System::EventArgs^ e)
{
    textBox1->AppendText("Enter\r\n");
}

private: System::Void Test_button_Leave(System::Object^ sender, System::EventArgs^ e)
{
    textBox1->AppendText("Leave\r\n");
}

private: System::Void Test_button_MouseDown(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e)
{
    textBox1->AppendText("MouseDown\r\n");
}

private: System::Void Test_button_MouseUp(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e)
{
    textBox1->AppendText("MouseUp\r\n");
}

private: System::Void Test_button_MouseMove(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e)
{
    label1->Text = (e->X).ToString() + "|" + (e->Y).ToString();
}

private: System::Void Test_button_KeyPress(System::Object^ sender,
System::Windows::Forms::KeyPressEventArgs^ e)
{
    textBox1->AppendText("KeyPress: " + (e->KeyChar).ToString() + "\r\n");
}

private: System::Void Test_button_KeyUp(System::Object^ sender,
System::Windows::Forms::KeyEventArgs^ e)
{
    textBox1->AppendText("KeyUp: " + (e->KeyValue).ToString() + "\r\n");
}

```

```
private: System::Void Test_button_KeyDown(System::Object^ sender,
System::Windows::Forms::KeyEventArgs^ e)
{
    textBox1->AppendText("KeyDown: " + (e->KeyValue).ToString() + "\r\n");
}

private: System::Void Clear_button_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox1->Clear();
}

private: System::Void Jump_button_Click(System::Object^ sender, System::EventArgs^ e)
{
    Jump_button->Text = "you got me";
}

private: System::Void Jump_button_MouseMove(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e)
{
    Jump_button->Left = Width - Jump_button->Left - Jump_button->Width;
}

};
}
```

## 1.6 Aufgabe 2.6

```
// c:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesung_Kap_2_6\Form1.h

#pragma once

namespace CppCLRWinFormsProject {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Zusammenfassung für Form1
    ///
    /// Warnung: Wenn Sie den Namen dieser Klasse ändern, müssen Sie auch
    /// die Ressourcendateiname-Eigenschaft für das Tool zur Kompilierung verwalteter
    /// Ressourcen ändern,
    /// das allen RESX-Dateien zugewiesen ist, von denen diese Klasse abhängt.
    /// Anderenfalls können die Designer nicht korrekt mit den lokalisierten Ressourcen
    /// arbeiten, die diesem Formular zugewiesen sind.
    /// </summary>

    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO: Konstruktorcode hier hinzufügen.
            //
        }

    protected:
        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::CheckBox^ checkBox1;
    private: System::Windows::Forms::CheckBox^ checkBox2;
    private: System::Windows::Forms::CheckBox^ checkBox3;
    private: System::Windows::Forms::RadioButton^ radioButton1;
    private: System::Windows::Forms::RadioButton^ radioButton2;
    private: System::Windows::Forms::Button^ Test;
    private: System::Windows::Forms::RadioButton^ radioButton3;
    private: System::Windows::Forms::Button^ button1;

    protected:

    private:
        /// <summary>
        /// Erforderliche Designervariable.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
```



```

/// Erforderliche Methode für die Designerunterstützung.
/// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
/// </summary>
void InitializeComponent(void)
{
    this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
    this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());
    this->checkBox3 = (gcnew System::Windows::Forms::CheckBox());
    this->radioButton1 = (gcnew System::Windows::Forms::RadioButton());
    this->radioButton2 = (gcnew System::Windows::Forms::RadioButton());
    this->Test = (gcnew System::Windows::Forms::Button());
    this->radioButton3 = (gcnew System::Windows::Forms::RadioButton());
    this->button1 = (gcnew System::Windows::Forms::Button());
    this->SuspendLayout();
    //
    // checkBox1
    //
    this->checkBox1->AutoSize = true;
    this->checkBox1->Location = System::Drawing::Point(12, 12);
    this->checkBox1->Name = L"checkBox1";
    this->checkBox1->Size = System::Drawing::Size(96, 17);
    this->checkBox1->TabIndex = 0;
    this->checkBox1->Text = L"enable/disable";
    this->checkBox1->UseVisualStyleBackColor = true;
    //
    // checkBox2
    //
    this->checkBox2->AutoSize = true;
    this->checkBox2->Location = System::Drawing::Point(12, 46);
    this->checkBox2->Name = L"checkBox2";
    this->checkBox2->Size = System::Drawing::Size(70, 17);
    this->checkBox2->TabIndex = 1;
    this->checkBox2->Text = L"Aufschrift";
    this->checkBox2->UseVisualStyleBackColor = true;
    //
    // checkBox3
    //
    this->checkBox3->AutoSize = true;
    this->checkBox3->Location = System::Drawing::Point(12, 80);
    this->checkBox3->Name = L"checkBox3";
    this->checkBox3->Size = System::Drawing::Size(76, 17);
    this->checkBox3->TabIndex = 2;
    this->checkBox3->Text = L"show/hide\r\n";
    this->checkBox3->UseVisualStyleBackColor = true;
    //
    // radioButton1
    //
    this->radioButton1->AutoSize = true;
    this->radioButton1->Location = System::Drawing::Point(171, 19);
    this->radioButton1->Name = L"radioButton1";
    this->radioButton1->Size = System::Drawing::Size(37, 17);
    this->radioButton1->TabIndex = 3;
    this->radioButton1->TabStop = true;
    this->radioButton1->Text = L"rot";
    this->radioButton1->UseVisualStyleBackColor = true;
    this->radioButton1->CheckedChanged += gcnew System::EventHandler(this,
&Form1::radioButton1_CheckedChanged);
    //
    // radioButton2
    //
    this->radioButton2->AutoSize = true;
    this->radioButton2->Location = System::Drawing::Point(172, 50);
    this->radioButton2->Name = L"radioButton2";
    this->radioButton2->Size = System::Drawing::Size(52, 17);
    this->radioButton2->TabIndex = 4;
    this->radioButton2->TabStop = true;
    this->radioButton2->Text = L"gruen";
    this->radioButton2->UseVisualStyleBackColor = true;
    //

```

```

    // Test
    //
    this->Test->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
    static_cast<System::Byte>(0)));
    this->Test->Location = System::Drawing::Point(142, 119);
    this->Test->Name = L"Test";
    this->Test->Size = System::Drawing::Size(75, 23);
    this->Test->TabIndex = 7;
    this->Test->Text = L"Test";
    this->Test->UseVisualStyleBackColor = true;
    this->Test->Click += gcnew System::EventHandler(this, &Form1::button2_Click);
    //
    // radioButton3
    //
    this->radioButton3->AutoSize = true;
    this->radioButton3->Location = System::Drawing::Point(172, 80);
    this->radioButton3->Name = L"radioButton3";
    this->radioButton3->Size = System::Drawing::Size(45, 17);
    this->radioButton3->TabIndex = 8;
    this->radioButton3->TabStop = true;
    this->radioButton3->Text = L"blau";
    this->radioButton3->UseVisualStyleBackColor = true;
    //
    // button1
    //
    this->button1->Location = System::Drawing::Point(33, 119);
    this->button1->Name = L"button1";
    this->button1->Size = System::Drawing::Size(75, 23);
    this->button1->TabIndex = 9;
    this->button1->Text = L"button1";
    this->button1->UseVisualStyleBackColor = true;
    //
    // Form1
    //
    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
    this->ClientSize = System::Drawing::Size(259, 147);
    this->Controls->Add(this->button1);
    this->Controls->Add(this->radioButton3);
    this->Controls->Add(this->Test);
    this->Controls->Add(this->radioButton2);
    this->Controls->Add(this->radioButton1);
    this->Controls->Add(this->checkBox3);
    this->Controls->Add(this->checkBox2);
    this->Controls->Add(this->checkBox1);
    this->Name = L"Form1";
    this->Text = L"Form1";
    this->Click += gcnew System::EventHandler(this, &Form1::Form1_Click);
    this->ResumeLayout(false);
    this->PerformLayout();

}
#pragma endregion

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
{
    if (checkBox1->Checked) // a1)
    {
        checkBox2->Enabled = true;
        radioButton1->Enabled = true;
        button1->Enabled = true;
    }
    else
    {
        checkBox2->Enabled = false;
        radioButton1->Enabled = false;
        button1->Enabled = false;
    }
}

```

```

// alternativ // a1)
checkBox2->Enabled = checkBox1->Checked;
radioButton1->Enabled = checkBox1->Checked;
button1->Enabled = checkBox1->Checked;

if (checkBox2->Checked) // a2)
    button1->Text = "Hello world";
else
    button1->Text = "Hello moon";

button1->Visible = checkBox3->Checked; // a3)

if (radioButton1->Checked) // a4)
    checkBox1->BackColor = Color::Red;
if (radioButton2->Checked)
    checkBox1->BackColor = Color::Green;
if (radioButton3->Checked)
    checkBox1->BackColor = Color::Blue;
// alternativ:
if (radioButton1->Checked) // a4)
    checkBox1->BackColor = Color::Red;
else if (radioButton2->Checked)
    checkBox1->BackColor = Color::Green;
else
    checkBox1->BackColor = Color::Blue;
}

private: System::Void radioButton1_CheckedChanged(System::Object^ sender, System::EventArgs^
e)
{
    button1->BackColor = Color::Red;
}

private: System::Void Form1_Click(System::Object^ sender, System::EventArgs^ e)
{
    checkBox2->Text = "caption";
    radioButton1->Text = "red";
    radioButton2->Text = "green";
    radioButton3->Text = "blue";
}

};
}

```

## 1.7 Aufgabe 2.7

```
// c:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesung_Kap_2_7\Form1.h

#pragma once

namespace CppCLRWinFormsProject {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Summary for Form1
    ///
    /// WARNING: If you change the name of this class, you will need to change the
    /// 'Resource File Name' property for the managed resource compiler tool
    /// associated with all .resx files this class depends on. Otherwise,
    /// the designers will not be able to interact properly with localized
    /// resources associated with this form.
    /// </summary>

    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::TabControl^ tabControl1;
    protected:
    private: System::Windows::Forms::TabPage^ tabPage1;
    private: System::Windows::Forms::TabPage^ tabPage2;
    private: System::Windows::Forms::GroupBox^ groupBox2;
    private: System::Windows::Forms::GroupBox^ groupBox1;
    private: System::Windows::Forms::RadioButton^ radioButton2;
    private: System::Windows::Forms::RadioButton^ radioButton1;
    private: System::Windows::Forms::TabPage^ tabPage3;
    private: System::Windows::Forms::Panel^ panel1;
    private: System::Windows::Forms::Button^ button3;
    private: System::Windows::Forms::Button^ button2;
    private: System::Windows::Forms::Button^ button1;
    private: System::Windows::Forms::RadioButton^ radioButton6;
    private: System::Windows::Forms::RadioButton^ radioButton5;
    private: System::Windows::Forms::RadioButton^ radioButton4;
    private: System::Windows::Forms::RadioButton^ radioButton3;
    private: System::Windows::Forms::Button^ button4;

    private:

```

```

    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        this->tabControl1 = (gcnew System::Windows::Forms::TabControl());
        this->tabPage1 = (gcnew System::Windows::Forms::TabPage());
        this->tabPage2 = (gcnew System::Windows::Forms::TabPage());
        this->tabPage3 = (gcnew System::Windows::Forms::TabPage());
        this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
        this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
        this->radioButton1 = (gcnew System::Windows::Forms::RadioButton());
        this->radioButton2 = (gcnew System::Windows::Forms::RadioButton());
        this->radioButton3 = (gcnew System::Windows::Forms::RadioButton());
        this->radioButton4 = (gcnew System::Windows::Forms::RadioButton());
        this->radioButton5 = (gcnew System::Windows::Forms::RadioButton());
        this->radioButton6 = (gcnew System::Windows::Forms::RadioButton());
        this->panel1 = (gcnew System::Windows::Forms::Panel());
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->button2 = (gcnew System::Windows::Forms::Button());
        this->button3 = (gcnew System::Windows::Forms::Button());
        this->button4 = (gcnew System::Windows::Forms::Button());
        this->tabControl1->SuspendLayout();
        this->tabPage1->SuspendLayout();
        this->tabPage3->SuspendLayout();
        this->groupBox1->SuspendLayout();
        this->groupBox2->SuspendLayout();
        this->panel1->SuspendLayout();
        this->SuspendLayout();
        //
        // tabControl1
        //
        this->tabControl1->Controls->Add(this->tabPage1);
        this->tabControl1->Controls->Add(this->tabPage2);
        this->tabControl1->Controls->Add(this->tabPage3);
        this->tabControl1->Dock = System::Windows::Forms::DockStyle::Fill;
        this->tabControl1->Location = System::Drawing::Point(0, 0);
        this->tabControl1->Name = L"tabControl1";
        this->tabControl1->SelectedIndex = 0;
        this->tabControl1->Size = System::Drawing::Size(281, 179);
        this->tabControl1->TabIndex = 0;
        //
        // tabPage1
        //
        this->tabPage1->Controls->Add(this->panel1);
        this->tabPage1->Controls->Add(this->groupBox2);
        this->tabPage1->Controls->Add(this->groupBox1);
        this->tabPage1->Location = System::Drawing::Point(4, 22);
        this->tabPage1->Name = L"tabPage1";
        this->tabPage1->Padding = System::Windows::Forms::Padding(3);
        this->tabPage1->Size = System::Drawing::Size(273, 153);
        this->tabPage1->TabIndex = 0;
        this->tabPage1->Text = L"a";
        this->tabPage1->UseVisualStyleBackColor = true;
        //
        // tabPage2
        //
        this->tabPage2->Location = System::Drawing::Point(4, 22);
        this->tabPage2->Name = L"tabPage2";
        this->tabPage2->Padding = System::Windows::Forms::Padding(3);
        this->tabPage2->Size = System::Drawing::Size(273, 153);
        this->tabPage2->TabIndex = 1;
    }

```

```

this->tabPage2->Text = L"b");
this->tabPage2->UseVisualStyleBackColor = true;
//
// tabPage3
//
this->tabPage3->Controls->Add(this->button4);
this->tabPage3->Location = System::Drawing::Point(4, 22);
this->tabPage3->Name = L"tabPage3";
this->tabPage3->Padding = System::Windows::Forms::Padding(3);
this->tabPage3->Size = System::Drawing::Size(273, 153);
this->tabPage3->TabIndex = 2;
this->tabPage3->Text = L"c");
this->tabPage3->UseVisualStyleBackColor = true;
//
// groupBox1
//
this->groupBox1->Controls->Add(this->radioButton3);
this->groupBox1->Controls->Add(this->radioButton2);
this->groupBox1->Controls->Add(this->radioButton1);
this->groupBox1->Location = System::Drawing::Point(8, 6);
this->groupBox1->Name = L"groupBox1";
this->groupBox1->Size = System::Drawing::Size(119, 92);
this->groupBox1->TabIndex = 0;
this->groupBox1->TabStop = false;
this->groupBox1->Text = L"groupBox1";
//
// groupBox2
//
this->groupBox2->Controls->Add(this->radioButton6);
this->groupBox2->Controls->Add(this->radioButton5);
this->groupBox2->Controls->Add(this->radioButton4);
this->groupBox2->Location = System::Drawing::Point(133, 6);
this->groupBox2->Name = L"groupBox2";
this->groupBox2->Size = System::Drawing::Size(131, 92);
this->groupBox2->TabIndex = 1;
this->groupBox2->TabStop = false;
this->groupBox2->Text = L"groupBox2";
//
// radioButton1
//
this->radioButton1->AutoSize = true;
this->radioButton1->Location = System::Drawing::Point(6, 19);
this->radioButton1->Name = L"radioButton1";
this->radioButton1->Size = System::Drawing::Size(85, 17);
this->radioButton1->TabIndex = 0;
this->radioButton1->TabStop = true;
this->radioButton1->Text = L"radioButton1";
this->radioButton1->UseVisualStyleBackColor = true;
//
// radioButton2
//
this->radioButton2->AutoSize = true;
this->radioButton2->Location = System::Drawing::Point(6, 42);
this->radioButton2->Name = L"radioButton2";
this->radioButton2->Size = System::Drawing::Size(85, 17);
this->radioButton2->TabIndex = 1;
this->radioButton2->TabStop = true;
this->radioButton2->Text = L"radioButton2";
this->radioButton2->UseVisualStyleBackColor = true;
//
// radioButton3
//
this->radioButton3->AutoSize = true;
this->radioButton3->Location = System::Drawing::Point(6, 65);
this->radioButton3->Name = L"radioButton3";
this->radioButton3->Size = System::Drawing::Size(85, 17);
this->radioButton3->TabIndex = 2;
this->radioButton3->TabStop = true;
this->radioButton3->Text = L"radioButton3";

```

```
this->radioButton3->UseVisualStyleBackColor = true;
//
// radioButton4
//
this->radioButton4->AutoSize = true;
this->radioButton4->Location = System::Drawing::Point(6, 19);
this->radioButton4->Name = L"radioButton4";
this->radioButton4->Size = System::Drawing::Size(85, 17);
this->radioButton4->TabIndex = 0;
this->radioButton4->TabStop = true;
this->radioButton4->Text = L"radioButton4";
this->radioButton4->UseVisualStyleBackColor = true;
//
// radioButton5
//
this->radioButton5->AutoSize = true;
this->radioButton5->Location = System::Drawing::Point(6, 42);
this->radioButton5->Name = L"radioButton5";
this->radioButton5->Size = System::Drawing::Size(85, 17);
this->radioButton5->TabIndex = 1;
this->radioButton5->TabStop = true;
this->radioButton5->Text = L"radioButton5";
this->radioButton5->UseVisualStyleBackColor = true;
//
// radioButton6
//
this->radioButton6->AutoSize = true;
this->radioButton6->Location = System::Drawing::Point(6, 65);
this->radioButton6->Name = L"radioButton6";
this->radioButton6->Size = System::Drawing::Size(85, 17);
this->radioButton6->TabIndex = 2;
this->radioButton6->TabStop = true;
this->radioButton6->Text = L"radioButton6";
this->radioButton6->UseVisualStyleBackColor = true;
//
// panel1
//
this->panel1->Controls->Add(this->button3);
this->panel1->Controls->Add(this->button2);
this->panel1->Controls->Add(this->button1);
this->panel1->Location = System::Drawing::Point(8, 108);
this->panel1->Name = L"panel1";
this->panel1->Size = System::Drawing::Size(256, 36);
this->panel1->TabIndex = 2;
//
// button1
//
this->button1->Location = System::Drawing::Point(3, 10);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(75, 23);
this->button1->TabIndex = 0;
this->button1->Text = L"OK";
this->button1->UseVisualStyleBackColor = true;
//
// button2
//
this->button2->Location = System::Drawing::Point(84, 10);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(75, 23);
this->button2->TabIndex = 1;
this->button2->Text = L"Hilfe";
this->button2->UseVisualStyleBackColor = true;
//
// button3
//
this->button3->Location = System::Drawing::Point(165, 10);
this->button3->Name = L"button3";
this->button3->Size = System::Drawing::Size(75, 23);
this->button3->TabIndex = 2;
```

```
this->button3->Text = L"Abbruch";
this->button3->UseVisualStyleBackColor = true;
//
// button4
//
this->button4->Location = System::Drawing::Point(89, 71);
this->button4->Name = L"button4";
this->button4->Size = System::Drawing::Size(75, 23);
this->button4->TabIndex = 0;
this->button4->Text = L"button4";
this->button4->UseVisualStyleBackColor = true;
//
// Form1
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(281, 179);
this->Controls->Add(this->tabControl1);
this->Name = L"Form1";
this->Text = L"Form1";
this->tabControl1->ResumeLayout(false);
this->tabPage1->ResumeLayout(false);
this->tabPage3->ResumeLayout(false);
this->groupBox1->ResumeLayout(false);
this->groupBox1->PerformLayout();
this->groupBox2->ResumeLayout(false);
this->groupBox2->PerformLayout();
this->panel1->ResumeLayout(false);
this->ResumeLayout(false);
}
#pragma endregion
};
}
```



## 1.8 Aufgabe 2.9

```
// c:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesung_Kap_2_9\Form1.h

#pragma once

namespace CppCLRWinFormsProject {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Zusammenfassung für Form1
    ///
    /// Warnung: Wenn Sie den Namen dieser Klasse ändern, müssen Sie auch
    /// die Ressourcendateiname-Eigenschaft für das Tool zur Kompilierung verwalteter
    /// Ressourcen ändern,
    /// das allen RESX-Dateien zugewiesen ist, von denen diese Klasse abhängt.
    /// Anderenfalls können die Designer nicht korrekt mit den lokalisierten Ressourcen
    /// arbeiten, die diesem Formular zugewiesen sind.
    /// </summary>

    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO: Konstruktorcode hier hinzufügen.
            //
        }

    protected:
        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::MenuStrip^ menuStrip1;
    protected:
    private: System::Windows::Forms::ContextMenuStrip^ contextMenuStrip1;
    private: System::Windows::Forms::ToolStripMenuItem^ dateiToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^ öffnenToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^ speichernToolStripMenuItem;
    private: System::Windows::Forms::ToolStripSeparator^ toolStripMenuItem1;
    private: System::Windows::Forms::ToolStripMenuItem^ schließenToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^ bearbeitenToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^ allesMarkierenToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^ kopierenToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^ ausschneidenToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^ farbenToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^ schriftToolStripMenuItem;
    private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;
    private: System::Windows::Forms::SaveFileDialog^ saveFileDialog1;
    private: System::Windows::Forms::PrintDialog^ printDialog1;
    private: System::Windows::Forms::ToolStripMenuItem^ toolStripMenuItem2;
    private: System::Windows::Forms::ColorDialog^ colorDialog1;
    private: System::Windows::Forms::FontDialog^ fontDialog1;

```

```

private: System::Windows::Forms::TextBox^ textBox1;
private: System::ComponentModel::IContainer^ components;

private:
    /// <summary>
    /// Erforderliche Designervariable.
    /// </summary>

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Erforderliche Methode für die Designerunterstützung.
    /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
    /// </summary>
    void InitializeComponent(void)
    {
        this->components = (gcnew System::ComponentModel::Container());
        this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
        this->dateiToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->öffnenToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->speichernToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->toolStripMenuItem2 = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->toolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripSeparator());
        this->schließenToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->bearbeitenToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->allesMarkierenToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->kopierenToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->ausschneidenToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->contextMenuStrip1 = (gcnew System::Windows::Forms::ContextMenuStrip(this-
>components));
        this->farbenToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->schriftToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());
        this->saveFileDialog1 = (gcnew System::Windows::Forms::SaveFileDialog());
        this->printDialog1 = (gcnew System::Windows::Forms::PrintDialog());
        this->colorDialog1 = (gcnew System::Windows::Forms::ColorDialog());
        this->fontDialog1 = (gcnew System::Windows::Forms::FontDialog());
        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
        this->menuStrip1->SuspendLayout();
        this->contextMenuStrip1->SuspendLayout();
        this->SuspendLayout();
        //
        // menuStrip1
        //
        this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^
>(2) {
            this->dateiToolStripMenuItem,
            this->bearbeitenToolStripMenuItem
        });
        this->menuStrip1->Location = System::Drawing::Point(0, 0);
        this->menuStrip1->Name = L"menuStrip1";
        this->menuStrip1->Size = System::Drawing::Size(292, 24);
        this->menuStrip1->TabIndex = 0;
        this->menuStrip1->Text = L"menuStrip1";
        //
        // dateiToolStripMenuItem
        //
        this->dateiToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(5) {
            this->öffnenToolStripMenuItem,
            this->speichernToolStripMenuItem, this->toolStripMenuItem2, this->toolStripMenuItem1,
this->schließenToolStripMenuItem
        });
        this->dateiToolStripMenuItem->Name = L"dateiToolStripMenuItem";
        this->dateiToolStripMenuItem->Size = System::Drawing::Size(44, 20);
        this->dateiToolStripMenuItem->Text = L"Datei";
        //
        // öffnenToolStripMenuItem
        //
        this->öffnenToolStripMenuItem->Name = L"öffnenToolStripMenuItem";

```

```

    this->öffnenToolStripMenuItem->Size = System::Drawing::Size(132, 22);
    this->öffnenToolStripMenuItem->Text = L"Öffnen";
    this->öffnenToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::öffnenToolStripMenuItem_Click);
    //
    // speichernToolStripMenuItem
    //
    this->speichernToolStripMenuItem->Name = L"speichernToolStripMenuItem";
    this->speichernToolStripMenuItem->Size = System::Drawing::Size(132, 22);
    this->speichernToolStripMenuItem->Text = L"Speichern";
    this->speichernToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::speichernToolStripMenuItem_Click);
    //
    // toolStripMenuItem2
    //
    this->toolStripMenuItem2->Name = L"toolStripMenuItem2";
    this->toolStripMenuItem2->Size = System::Drawing::Size(132, 22);
    this->toolStripMenuItem2->Text = L"Drucken";
    this->toolStripMenuItem2->Click += gcnew System::EventHandler(this,
&Form1::toolStripMenuItem2_Click);
    //
    // toolStripMenuItem1
    //
    this->toolStripMenuItem1->Name = L"toolStripMenuItem1";
    this->toolStripMenuItem1->Size = System::Drawing::Size(129, 6);
    //
    // schließenToolStripMenuItem
    //
    this->schließenToolStripMenuItem->Name = L"schließenToolStripMenuItem";
    this->schließenToolStripMenuItem->Size = System::Drawing::Size(132, 22);
    this->schließenToolStripMenuItem->Text = L"Schließen";
    this->schließenToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::schließenToolStripMenuItem_Click);
    //
    // bearbeitenToolStripMenuItem
    //
    this->bearbeitenToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(3) {
        this->allesMarkierenToolStripMenuItem,
        this->kopierenToolStripMenuItem, this->ausschneidenToolStripMenuItem
    });
    this->bearbeitenToolStripMenuItem->Name = L"bearbeitenToolStripMenuItem";
    this->bearbeitenToolStripMenuItem->Size = System::Drawing::Size(71, 20);
    this->bearbeitenToolStripMenuItem->Text = L"Bearbeiten";
    //
    // allesMarkierenToolStripMenuItem
    //
    this->allesMarkierenToolStripMenuItem->Name = L"allesMarkierenToolStripMenuItem";
    this->allesMarkierenToolStripMenuItem->Size = System::Drawing::Size(157, 22);
    this->allesMarkierenToolStripMenuItem->Text = L"Alles markieren";
    this->allesMarkierenToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::allesMarkierenToolStripMenuItem_Click);
    //
    // kopierenToolStripMenuItem
    //
    this->kopierenToolStripMenuItem->Name = L"kopierenToolStripMenuItem";
    this->kopierenToolStripMenuItem->Size = System::Drawing::Size(157, 22);
    this->kopierenToolStripMenuItem->Text = L"Kopieren";
    this->kopierenToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::kopierenToolStripMenuItem_Click);
    //
    // ausschneidenToolStripMenuItem
    //
    this->ausschneidenToolStripMenuItem->Name = L"ausschneidenToolStripMenuItem";
    this->ausschneidenToolStripMenuItem->Size = System::Drawing::Size(157, 22);
    this->ausschneidenToolStripMenuItem->Text = L"Ausschneiden";
    this->ausschneidenToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::ausschneidenToolStripMenuItem_Click);
    //

```

```

    // contextMenuStrip1
    //
    this->contextMenuStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(2) {
        this->farbenToolStripMenuItem,
        this->schriftToolStripMenuItem
    });
    this->contextMenuStrip1->Name = L"contextMenuStrip1";
    this->contextMenuStrip1->Size = System::Drawing::Size(120, 48);
    //
    // farbenToolStripMenuItem
    //
    this->farbenToolStripMenuItem->Name = L"farbenToolStripMenuItem";
    this->farbenToolStripMenuItem->Size = System::Drawing::Size(119, 22);
    this->farbenToolStripMenuItem->Text = L"Farben";
    this->farbenToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::farbenToolStripMenuItem_Click);
    //
    // schriftToolStripMenuItem
    //
    this->schriftToolStripMenuItem->Name = L"schriftToolStripMenuItem";
    this->schriftToolStripMenuItem->Size = System::Drawing::Size(119, 22);
    this->schriftToolStripMenuItem->Text = L"Schrift";
    this->schriftToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::schriftToolStripMenuItem_Click);
    //
    // openFileDialog1
    //
    this->openFileDialog1->FileName = L"openFileDialog1";
    //
    // printDialog1
    //
    this->printDialog1->UseEXDialog = true;
    //
    // textBox1
    //
    this->textBox1->ContextMenuStrip = this->contextMenuStrip1;
    this->textBox1->Dock = System::Windows::Forms::DockStyle::Fill;
    this->textBox1->Location = System::Drawing::Point(0, 24);
    this->textBox1->Multiline = true;
    this->textBox1->Name = L"textBox1";
    this->textBox1->Size = System::Drawing::Size(292, 242);
    this->textBox1->TabIndex = 1;
    //
    // Form1
    //
    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
    this->ClientSize = System::Drawing::Size(292, 266);
    this->Controls->Add(this->textBox1);
    this->Controls->Add(this->menuStrip1);
    this->MainMenuStrip = this->menuStrip1;
    this->Name = L"Form1";
    this->Text = L"Form1";
    this->menuStrip1->ResumeLayout(false);
    this->menuStrip1->PerformLayout();
    this->contextMenuStrip1->ResumeLayout(false);
    this->ResumeLayout(false);
    this->PerformLayout();
}

#pragma endregion

private: System::Void öffnenToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^
e)
{
    openFileDialog1->Filter = "txt-Dateien|*.txt";
    if (openFileDialog1->ShowDialog() ==
        System::Windows::Forms::DialogResult::OK)

```

```

    {
        textBox1->Text = IO::File::ReadAllText(openFileDialog1->FileName);
    }
}

private: System::Void speichernToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e)
{
    if (saveFileDialog1->ShowDialog() ==
        System::Windows::Forms::DialogResult::OK)
    {
        IO::File::WriteAllText(saveFileDialog1->FileName, textBox1->Text);
    }
}

private: System::Void schließenToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e)
{
    this->Close();
}

private: System::Void allesMarkierenToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e)
{
    textBox1->SelectAll();
}

private: System::Void kopierenToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e)
{
    textBox1->Copy();
}

private: System::Void ausschneidenToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e)
{
    textBox1->Cut();
}

private: System::Void toolStripMenuItem2_Click(System::Object^ sender, System::EventArgs^ e)
{
    printDialog1->ShowDialog();
}

private: System::Void farbenToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^
e)
{
    if (colorDialog1->ShowDialog() ==
        System::Windows::Forms::DialogResult::OK)
    {
        textBox1->BackColor = colorDialog1->Color;
    }
}

private: System::Void schriftToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^
e)
{
    if (fontDialog1->ShowDialog() ==
        System::Windows::Forms::DialogResult::OK)
    {
        textBox1->Font = fontDialog1->Font;
    }
}
};
}

```



## 2 Lösungen Kapitel 3. Elementare Datentypen

```
##### Copyright notice #####  
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook  
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",  
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
```

```
#include <cstdlib>  
#include <random>  
#include <ctime>  
#include <string>
```

```
namespace N_Loesungen_Elementare_Datentypen  
{  
    using namespace System;  
    using namespace System::Windows::Forms;
```

## 2.1 Aufgaben 3.1 Syntaxregeln

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h  
  
/*  
"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. "  
*/
```



## 2.2 Aufgaben 3.2 Variablen und Bezeichner

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h

// ----- Aufgabe 1. -----

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
*/
// Der nächste Name enthalten länderspezifische Buchstaben ("ü"),
// die bei vielen ältern Compilern nicht zulässig waren.
// In modernen C++-Compilern sind sie zulässig.
Einwohner_von_Tübingen; // zulässig in Visual C++, aber nicht in gcc

// ----- Aufgabe 2. -----

int i = 0; // beide Definitionen außerhalb der
int j;    // Formulkasse Form1

void Lsg_Variablen(TextBox^ textBox1)
{
    using namespace System;
    int k = 1;
    textBox1->AppendText(String::Format("i={0}\r\n", i)); // 0
    textBox1->AppendText(String::Format("j={0}\r\n", j)); // 0: globale Variablen werden
                                                         // immer initialisiert
    textBox1->AppendText(String::Format("k={0}\r\n", k)); // 1
    int i;
    textBox1->AppendText(String::Format("i={0}\r\n", i)); // ein unbestimmter Wert
}
}
```

## 2.3 Aufgabe 3.3.2 Ganzzahliterale

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h

// ----- Aufgabe 1. -----

void Lsg_GanzzahlLiterale_1(TextBox^ tb)
{
    // 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
    using namespace System;
    tb->AppendText("Aufgabe Binärsystem:" + System::Environment::NewLine);
    int i1 = 37;
    // Mit der Format-Angabe {0:X} wird ein Wert hexadezimal angezeigt:
    tb->AppendText(String::Format("1.c) dez: {0} hex: {0:X} \r\n", i1));
    int j1 = -37;
    tb->AppendText(String::Format("1.c) dez: {0} hex: {0:X} \r\n", j1));
    //    d) 37 + (-25)

    //          37: 00100101bin
    //          -25: 11100111bin
    //          -----
    //          100001100bin, mit 8 bits: 00001100bin = 12dez

    //    e) 25 + (-37)

    //          25: 00011001bin
    //          -37: 11011011bin
    //          -----
    //          11110100bin --> Zweierkomplement: 00001100bin
}

// ----- Aufgabe 2. -----

void Lsg_GanzzahlLiterale_2(TextBox^ tb)
{
    // 2. a)

    // Zweierkomplement von ab=1010 1011: 0101 0101 bin = 55 hex = 85 dez,
    // also -85

    tb->AppendText("Aufgabe 3.3.2:\r\n");
    signed char i2 = 0xab;
    tb->AppendText(String::Format("2.a) dez: {0} hex: {0:X} \r\n", i2));

    //    b)

    // ab = 10*16 + 11 = 171

    unsigned char j2 = 0xab;
    tb->AppendText(String::Format("2.b) dez: {0} hex: {0:X} \r\n", j2));
}

// ----- Aufgabe 3. -----
```

```
void Lsg_GanzzahlLiterale_3(TextBox^ tb)
{
    using namespace System;
    tb->AppendText(String::Format("3.3.2, 3.: 030={0}\r\n", 030));
    // Vorwahl Berlin // 24,
    // da wegen der führenden 0 ein Oktalliteral
    tb->AppendText(String::Format("017+15={0}\r\n", 017 + 15)); // 30
    tb->AppendText(String::Format("0x12+10={0}\r\n", 0x12 + 10)); // 30
}
```

## 2.4 Aufgabe 3.3.6 Operatoren und die üblichen Konversionen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h

void Lsg_Ganzzahl_Operatoren(TextBox^ tb)
{
    // ----- Aufgabe 1. -----
    { // a)
        unsigned char b = 255;
        unsigned char c = 0;
        short i = 1234567898;
        b = b + 1;          // 0
        c = c - 1;         // 255
        int d = i / 100;
        int e = i % 100;
        int f = i + i;

        tb->AppendText(String::Format("b={0} c={1} d={2} e={3} f={4} \r\n",
            b, c, d, e, f));
    }
    { // b) nicht zulässige Zeilen wurden auskommentiert:
        unsigned char b = { 255 };
        unsigned char c = { 0 };
        // short i = { 1234567898 };
        // b = { b + 1 };
        // c = { c - 1 };
        int d = { i / 100 };
        int e = { i % 100 };
        int f = { i + i };

        tb->AppendText(String::Format("b={0} c={1} d={2} e={3} f={4} \r\n",
            b, c, d, e, f));
    }
}

// ----- Aufgabe 2. -----
/*
    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.
*/

// ----- Aufgabe 3. -----
// a)
// int a=x^x;

// a erhält immer den Wert 0, da die bitweise Verknüpfung derselben
// Operanden mit xor immer den Wert 0 hat.

// b)
// int b=x&1;

// b ist genau dann 1, wenn x ungerade ist.

// ----- Aufgabe 4. -----

// Im Bitmuster von x und y darf nicht an derselben Stelle eine 1 stehen.
```

## 2.5 Aufgabe 3.3.8 Der Datentyp bool

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
```

```
void Lsg_char()
{
    // Die Variablen in dieser Aufgabe sind mit willkürlichen Werten
    // initialisiert, was natürlich keinen Sinn macht. Allerdings macht
    // es auch wenig Sinn, diese Anweisungen isoliert auszuführen. Es
    // wäre angemessener, diese Anweisungen als Funktionen zu formulieren,
    // und die Variablen als Parameter zu übergeben. Dann werden sie beim
    // Aufruf initialisiert.

    // ----- Aufgabe 1. -----
    // a)
    char c = 'x'; // Initialisierung nur, damit das kompiliert wird
    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 2. -----
    int Jahr = 2021;
    bool Schaltjahr = ((Jahr % 4 == 0) && (Jahr % 100 != 0)) || (Jahr % 400 == 0);

    // ----- Aufgabe 3. -----

    int j1 = 2021, j2 = 2022, m1 = 4, m2 = 5, t1 = 10, t2 = 11;
    bool vorher = (j1 < j2) || ((j1 == j2) && ((m1 < m2) || ((m1 == m2) && (t1 < t2))));

    // ----- Aufgabe 4. -----

    /*
    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    1&1 == 1 (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 auslösen. Z.B. erhält man
    mit 1&2 den Wert false, während 1&&2 true ist.
    */

    // ----- Aufgabe 5. -----
    /*
    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) ...  
*/  
}
```

## 2.6 Aufgabe 3.4 Kontrollstrukturen und Funktionen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h

void initInConstructor(TextBox^ textBox2, TextBox^ textBox3)
{ //
  // Damit es keine Konversionsfehler gibt, wenn man vergisst, einen
  // ganzzahligen Wert in das Eingabefenster einzugeben.
  // Diese Funktion im Form1-Konstruktor aufrufen.

  textBox2->Text = "10";
  textBox3->Text = "50";
}
```

## 2.7 Aufgabe 3.4.1 Die if- und die Verbundanweisung

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
```

```
void Lsg_if(TextBox^ tb)
{
    { // ----- Aufgabe 1 -----

        int x = 0, Punkte = 0, i = 0; // Die Werte 0 wurden nur angegeben, damit
        // das Programm kompiliert wird. Sie sind für diese Aufgabe ohne Bedeutung.

        // a)
        if (x = 17) tb->AppendText("Volltreffer\r\n");
        // "x=17" ist eine Zuweisung und keine Prüfungen auf Gleichheit
        // Syntaktisch zulässig, aber semantisch wird der Ausdruck x=17
        // in den int-Wert 17 konvertiert, und den in den bool-Wert true.

        // b)
        if (i >= 1 && i <= 10)
            tb->AppendText("Volltreffer\r\n");
        // Syntaktisch und semantisch korrekt. Verknüpfte Bedingungen müssen
        // nicht geklammert werden.

        // c)
        // if b && (i=j*x) tb->AppendText("Volltreffer\r\n");
        // Syntaxfehler: Die Bedingung in einer if-Anweisung muss in einer Klammer
        // stehen. "=" ist ein Zuweisung und keine Prüfung auf Gleichheit.

        // d)
        if (Punkte >= 0) tb->Text = "Extremes Pech";
        else if (Punkte >= 20) tb->Text = "Ziemliches Pech";
        else if (Punkte >= 40) tb->Text = "Ein wenig Glück gehabt";

        // Syntaktisch korrekt, aber der zweite und dritte else-Zweig werden nie
        // erreicht. Vermutlich „<“ mit „>“ verwechselt.
    }
}
```

```
{ // ----- Aufgabe 2 -----
```

```
/* Hier 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 gehört), 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:

```

if (Bewegungsart=='+') Kontostand = Kontostand + Betrag;
else if (Bewegungsart=='-') Kontostand = Kontostand - Betrag;
else Fehlermeldung();
*/
}
{ // ----- Aufgabe 3 -----

int LA_Summe = 0, LB_Summe = 0, MA_Summe = 0, MB_Summe = 0, MC_Summe = 0, Summe = 0;
char Lagergruppe = 'A', Materialgruppe = 'B';
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 MessageBox::Show("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 MessageBox::Show("Unzulässige Materialgruppe in Lager B");
}
else MessageBox::Show("Unzulässige Lagergruppe");
}
{ // ----- Aufgabe 4 -----

int t1 = 10, m1 = 2, j1 = 2017; // Datum 1
// Initialisierung mit beliebigen Werten, damit das kompiliert wird
int t2 = 2, m2 = 10, j2 = 2017; // Datum 2
bool vorher;
if (j1 != j2) vorher = j1 < j2;
else /* j1==j2 */ if (m1 != m2)
    vorher = m1 < m2;
else // (j1==j2) and (m1==m2)
    vorher = 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;
}
}

```

## 2.8 Aufgabe 3.4.4 Ganzzahl-Funktionen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
```

```
// ----- Aufgabe 1 -----
```

```
// a)
int plus1(int n)
{
    return n + 1;
}
```

```
// b) plus1 kann dann in Form1.h wie in Abschnitt 2.11.1 aufgerufen werden.
```

```
// ----- Aufgabe 2 -----
```

```
/*
```

Falls Sie die Lösung dieser Aufgabe nicht sofort finden, können Sie versuchen, diese zunächst für einfache Spezialfälle zu finden, und diese Spezialfälle dann zu verallgemeinern.

Erster Spezialfall: n einstellig

```
int Quersumme(int n)
{
    int s = 0;
    s = s + n;
    return s;
}
```

Zweiter Spezialfall: n zweistellig

```
int Quersumme(int n)
{
    int s = 0;
    s = s + n/10 + n%10;
    return s;
}
```

Berechnet man die Quersumme von  $n=123$  als Summe  $3 + 2 + 1$  (und nicht als Summe  $1 + 2 + 3$ ), erhält man die erste Ziffer 3 mit  $n\%10$ . Nachdem man diese zu einer Summe  $s$  addiert hat, kann man diese "abschneiden", indem man  $n$  durch  $n/10$  (12) ersetzt. Diese Schritte wiederholt man in einer Schleife, bis  $n==0$  ist.

```
*/
```

```
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;
}
```

```
// b), c) Diese Funktion kann dann in Form1.h wie in Abschnitt 2.11.1 aufgerufen werden.
```

```
// ----- Aufgabe 3 -----
```

```
/*
```

Falls Sie die Lösung dieser Aufgabe nicht sofort finden, können Sie versuchen, diese zunächst für einfache Spezialfälle zu finden, und diese Spezialfälle dann zu verallgemeinern.

Erster Spezialfall:  $n=2$

Mit den Anfangswerten

```
x = 0, y = 1
```

erhält man die nächste Fibonacci-Zahl durch

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

Die nächste Fibonacci-Zahl  $f$  erhält man dann mit den Werten

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

Diese Schritte wiederholt man in einer Schleife.

```
*/
// a)
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) Diese Funktion kann dann in Form1.h wie in Abschnitt 2.11.1 aufgerufen werden.

// ----- Aufgabe 4 -----

void for_Schleife_kleiner_gleich(TextBox^ tb, int n)
{
    for (unsigned int u = 0; u <= n - 1; u++) // warning: Konflikt zwischen
        tb->AppendText(u.ToString() +
            System::Environment::NewLine); // signed und unsigned

    // 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++) // warning: Konflikt zwischen signed
        tb->AppendText(u.ToString() + System::Environment::NewLine); // unsigned
}

// b) Diese Funktion kann dann in Form1.h wie in Abschnitt 2.11.1 aufgerufen werden.

// ----- Aufgabe 5 -----
// a)
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_Variante_2(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;
    }
}

// b)
void zeigePrimzahlen(TextBox^ tb, int n)
{ // Diese Funktion kann dann in Form1.h wie in Abschnitt 2.11.1 aufgerufen werden.
    tb->AppendText("Primzahlen:" + System::Environment::NewLine);
    for (int i = 1; i < n; i++)
        if (istPrim(i))
            tb->AppendText(i.ToString() + " ");
    tb->AppendText(System::Environment::NewLine);
}

// c)
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(TextBox^ tb, int n)
{ // Diese Funktion kann dann in Form1.h wie in Abschnitt 2.11.1 aufgerufen werden.
    using namespace System;
    tb->AppendText("Goldbach'sche Vermutung:" + System::Environment::NewLine);
    for (int i = 6; i < n; i = i + 2)
        tb->AppendText(String::Format("Goldbach({0})={1}\r\n", i, Goldbach(i)));
}

// ----- Aufgabe 6 -----

int zeigePythagTripel(TextBox^ tb, int n, bool show = false)
{ // Diese Funktion kann dann in Form1.h wie in Abschnitt 2.11.1 aufgerufen werden.
    using namespace System;
    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)
                {
                    if (show)
                        tb->AppendText(String::Format(
                            "a={0} b={1} c={2}\r\n", a, b, c));
                    nt++;
                }
    return nt;
}

// ----- Aufgabe 7 -----

int f3nplus1(TextBox^ tb, int n, bool show = false)
{

```

```

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)
        tb->AppendText(" n=" + n.ToString() + System::Environment::NewLine);
}
return m;
}

void zeige3nplus1(TextBox^ tb, int n)
{
    for (int i = 0; i < n; i++)
    {
        int k = f3nplus1(tb, i);
        tb->AppendText("i=" + i.ToString() + " k=" + k.ToString() +
            System::Environment::NewLine);
    }
} // Diese Funktion kann dann in Form1.h wie in Abschnitt 2.11.1 aufgerufen werden.

// ----- Aufgabe 8 -----
std::string Osterdatum(int J)
{
    int M = -1, N = -1;
    std::string 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 ";

    int A = J % 19;
    int B = J % 4;
    int C = J % 7;
    int D = (19 * A + M) % 30;
    int E = (2 * B + 4 * C + 6 * D + N) % 7;
    if ((22 + D + E >= 1) && (22 + D + E <= 31)) // in [1..31])
    {
        int OTag = 22 + D + E;
        int PTag = OTag + 49;
        s = s + std::to_string(J) + " Ostern: " + std::to_string(OTag) + ". März Pfingsten: ";
        if (PTag > 31 + 30 + 31) // Tage im März + April + Mai
            s = s + std::to_string(PTag - 92) + ". Juni";
        else if (PTag > 30 + 31) // Tage im April + Mai
            s = s + std::to_string(PTag - 61) + ". Mai";
        else s = s + " unmöglich"; // Wegen OTag >= 22 ist PTag >= 71
    }
    else
    {
        int OTag = D + E - 9;
        int PTag = OTag + 49;
        if (OTag == 26) OTag = 19;
        else if ((OTag == 25) && (D == 28) && (E == 6) && (A > 10)) OTag = 18;
        s = s + std::to_string(J) + " Ostern: " + std::to_string(OTag) + ". April Pfingsten: ";
        if (PTag > 30 + 31) // Tage im April + Mai
            s = s + std::to_string(PTag - 61) + ". Juni";
        else if (PTag > 30) // Tage im April
            s = s + std::to_string(PTag - 30) + ". Mai";
        else s = s + " unmöglich"; // wegen OTag >= -9 ist PTag >= 40
    }
    return s;
}
}

```

```

void zeigeOsterdatum(TextBox^ tb, int von, int bis)
{
    for (int J = von; J <= bis; J++)
        tb->AppendText(rkl::to_String(Osterdatum(J)) +
            System::Environment::NewLine);
} // Diese Funktion kann dann in Form1.h wie in Abschnitt 2.11.1 aufgerufen werden.

// ----- Aufgabe 9 -----

/*
1. plus1:

    Es wird immer ein Wert zurückgegeben.

2. 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_1(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.

4. 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.

5. bool prim(int n)
   int Goldbach(int n)

    Da in jedem Zweig der if-Anweisung ein Wert zurückgegeben wird,
    ist der Funktionswert immer definiert.

6. int zeige_PythagZahlentripel(int n): wie 1.

7. int f3nplus1(int n, bool show): gibt entweder -1 oder m zurück,
   falls das keine Endlosschleife ist.

8. Osterdatum(int J): wie 1.
*/

```

## 2.9 Aufgabe 3.4.7 Zufallszahlen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h

// ----- Aufgabe 1. -----

void zeigeZufallszahlen(TextBox^ tb)
{
    // Nach
    // using std::rand;
    // kann man statt std::rand einfach nur rand verwenden

    tb->AppendText("rand ohne srand" + System::Environment::NewLine);
    for (int i = 0; i < 5; i++)
    {
        tb->AppendText(std::rand().ToString() + System::Environment::NewLine);
    }

    tb->AppendText("rand mit srand(std::time(0))" +
        System::Environment::NewLine);

    srand(std::time(0));
    for (int i = 0; i < 5; i++)
    {
        tb->AppendText(std::rand().ToString() + System::Environment::NewLine);
    }

    tb->AppendText("minstd_rand ohne init" + System::Environment::NewLine);
    std::minstd_rand rnd;
    for (int i = 0; i < 5; i++)
    {
        int r = rnd();
        tb->AppendText(r.ToString() + System::Environment::NewLine);
    }

    tb->AppendText("5 aus 49 mit init std::time(0)" +
        System::Environment::NewLine);
    rnd.seed(std::time(0)); // oder: std::minstd_rand rnd(std::time(0));
    for (int i = 0; i < 5; i++)
    {
        int r = 1 + rnd() % 49;
        tb->AppendText(r.ToString() + System::Environment::NewLine);
    }
} // Diese Funktion kann dann in Form1.h wie in Abschnitt 2.11.1 aufgerufen werden.
```

## 2.10 Aufgabe 3.4.8 Default-Argumente

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
// ----- Aufgabe 1. -----
// Nur nach der ersten Variante, da nach
// #include "MeineFunktion.h"
// die Funktion inc nur in dieser Variante
// ein Default-Argument hat.
```



## 2.11 Aufgabe 3.4.10 Programmierstil

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
// ----- Aufgabe 1. -----

// Diese Funktion führt offensichtlich viele verschiedene Aufgaben aus.
// Deshalb gibt es keinen passenden Namen für sie.

// ----- Aufgabe 2. -----
/* Ein Beispiel dafür, wie verwirrend falsche Namen sein können.
int x = Plus(1, Product(Minus(3, 4), Plus(5, 6)));
    = Plus(1, Product(    12,        -1))
    = Plus(1,11)
    = 10
*/
```

## 2.12 Aufgabe 3.4.11 rekursive Funktionen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
// ----- Aufgabe 1 -----

// 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
  {
    // ggT(x,y)=ggT(x0,y0)
    // x0 y0 r
    // x0 mod y0 + r0
    int r = x % y; //
    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 zeige_rek(TextBox^ tb)
{
  tb->AppendText("Rekursion: No news are good news" +
    System::Environment::NewLine);
}
```

```

for (int i = 1; i < 50; i++)
    if (rek_Fak(i) != it_Fak(i))
        tb->AppendText(i.ToString() + System::Environment::NewLine);

for (int i = 1; i < 50; i++)
    for (int j = 1; j < 50; j++)
        if (rek_ggT(i, j) != it_ggT(i, j))
            tb->AppendText(i.ToString() + " " + j.ToString() +
                System::Environment::NewLine);

for (int i = 1; i < 30; i++)
    if (rek_Fib(i) != it_Fib(i))
        tb->AppendText(i.ToString() + System::Environment::NewLine);
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_rekursive_Funktionen()
{
    for (int i = 1; i < 50; i++)
        rkl::Assert::AreEqual(rek_Fak(i), it_Fak(i), "rek_Fak/it_Fak, i=" +
            std::to_string(i));

    for (int i = 1; i < 30; i++)
        rkl::Assert::AreEqual(rek_Fib(i), it_Fib(i), "rek_Fib/it_Fib, i=" +
            std::to_string(i));

    for (int i = 1; i < 50; i++)
        for (int j = 1; j < 50; j++)
            rkl::Assert::AreEqual(rek_ggT(i, j), it_ggT(i, j),
                "ggT(" + std::to_string(i) + ", " + std::to_string(j) + ")");
}
#endif

// ----- Aufgabe 2 -----
int Ackermann(int n, int m)
{ // Ackermann-Funktion
    if (n == 0) return m + 1;
    else if (m == 0) return Ackermann(n - 1, 1);
    else return Ackermann(n - 1, Ackermann(n, m - 1));
}

void zeige_Ackermann(TextBox^ tb)
{
    /*
    Ackermann(0,m) = m+1
    Ackermann(1,m) = m+2
    Ackermann(2,m) = 2*m+3
    Ackermann(3,m) = 2^(m+3)+1
    */
    tb->AppendText("Ackermann: No news are good news" +
        System::Environment::NewLine);

    for (int m = 0; m < 50; m++)
        if (Ackermann(0, m) != m + 1)
            tb->AppendText("Unterschied für ack(0," + m.ToString() + ")\r\n");

    for (int m = 0; m < 50; m++)
        if (Ackermann(1, m) != m + 2)
            tb->AppendText("Unterschied für ack(1," + m.ToString() + ")\r\n");

    for (int m = 0; m < 50; m++)
        if (Ackermann(2, m) != 2 * m + 3)
            tb->AppendText("Unterschied für ack(2," + m.ToString() + ")\r\n");

    // für m>10 dauert das recht lange:
    for (int m = 0; m < 8; m++)
    {
        int a = Ackermann(3, m);
        int e = std::pow(double(2), m + 3) - 3;
    }
}

```

```

    if (a != e)
        tb->AppendText("Unterschied für ack(3," + m.ToString() + ")\r\n");
    }
}

#ifdef SIMPLEUNITTESTS_H
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_Ackermann(int max)
{
    /*
    Ackermann(0,m) = m+1
    Ackermann(1,m) = m+2
    Ackermann(2,m) = 2*m+3
    Ackermann(3,m) = 2^(m+3)+1
    */
    for (int m = 0; m < max; m++)
        rkl::Assert::AreEqual(m + 1, Ackermann(0, m), "ack(0," +
            std::to_string(m) + ")");

    for (int m = 0; m < max; m++)
        rkl::Assert::AreEqual(m + 2, Ackermann(1, m), "ack(1," +
            std::to_string(m) + ")");

    for (int m = 0; m < max; m++)
        rkl::Assert::AreEqual(2 * m + 3, Ackermann(2, m), "ack(2," +
            std::to_string(m) + ")");

    // für m>10 dauert das recht lange:
    for (int m = 0; m < 8; m++)
    {
        int a = Ackermann(3, m);
        int e = std::pow(double(2), m + 3) - 3;
        rkl::Assert::AreEqual(e, a, "Ackermann(3," + std::to_string(m) + ")");
    }
}
#endif

#ifdef SIMPLEUNITTESTS_H
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_Rekursion()
{
    rkl::Assert::Init("Unittests_Rekursion");
    Unittests_rekursive_Funktionen();
    Unittests_Ackermann(30);
    rkl::Assert::Summary();
}
#endif

```

## 2.13 Aufgabe 3.4.12 switch

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
// ----- 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: MessageBox::Show("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: MessageBox::Show("Unzulässige Materialgruppe in Lager B");
            }
        default: MessageBox::Show("Unzulässige Lagergruppe");
    }
}

// ----- Aufgabe 2 -----

// Da die Bedingungen nicht durch eine Prüfung auf Gleichheit mit einer
// Konstanten gebildet werden, ist eine Lösung mit switch nicht möglich.
```

## 2.14 Aufgabe 3.5 Gleitkommadatentypen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
#define _USE_MATH_DEFINES
// ----- Aufgabe 1. -----

void GleitkommaSyntax(TextBox^ tb)
{
    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, wird von links nach rechts ausgewertet
    int i5 = j / (z / y);        // 3
    int i6 = j / (y - j / k);    // 4

    using namespace System;
    tb->AppendText(String::Format("d1={0} d2={1} \r\n", d1, d2));
    tb->AppendText(String::Format("i1={0} i2={1} \r\n", i1, i2));
    tb->AppendText(String::Format("i4={0} i5={1} i6={2} \r\n", i4, i5, i6));
}

// ----- Aufgabe 2. -----

double RegentropfenPi(int n)
{
    int Treffer = 0;
    for (int i = 0; i < n; i++)
    {
        double x = (0.0 + rand()) / RAND_MAX; // für rand: #include <cstdlib>
        double y = (0.0 + rand()) / RAND_MAX;
        if (x * x + y * y < 1) Treffer++;
    }
    return 4.0 * Treffer / n;
}

void zeigeRegentropfenPi(TextBox^ tb, int n)
{
    tb->AppendText(RegentropfenPi(n).ToString() + System::Environment::NewLine);
}

// ----- Aufgabe 3. -----

int Fakultae(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 zeigeFakultaet(TextBox^ tb)
{
    using namespace System;
    for (int i = 1; i < 30; i++)
        tb->AppendText(String::Format("Fakultaet({0})={1} \r\n", i, Fakultae(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);
        tb->AppendText(String::Format("Fakultät({0})={1} Sek={2} Jahre={3}\r\n",
            i, Fakultaet(i), Sekunden, Jahre));
    }
}

// ----- Aufgabe 4. -----

void HypothekenRestschuld(TextBox^ tb, double Betrag, double Zinssatz, double Tilgungsrate)
{
    using namespace System;
    // Die Ergebnisse stimmen mit denen von Tilgungsrechnern im Internet überein (z.B.
    // http://www.interhyp.de/interhyp/servlet/interhyp?cmd=showPartnerTilgungsrechnerContent&LAF
    // PARTNER=sueddeutsche).
    double Restschuld = Betrag;
    double Annuitaet = Restschuld * (Zinssatz + Tilgungsrate) / 100;
    int Jahre = 0;

    String^ s = String::Format("Hypothek: {0} Annuitaet: {1}", Restschuld, Annuitaet);
    tb->AppendText(s);
    while (Restschuld > 0)
    {
        double Zinsen = Restschuld * Zinssatz / 100;
        Restschuld = Restschuld - (Annuitaet - Zinsen);
        Jahre++;
        s = String::Format("Ende {0}. Jahr: Restschuld={1:f2} Zinsen={2:f2} Tilgung={3:f2}",
            Jahre, Restschuld, Zinsen, Annuitaet - Zinsen);
        tb->AppendText(s + System::Environment::NewLine);
    }
};

// ----- Aufgabe 5. -----

// #include <cmath> für fabs, cmath ist etwas C++-spezifischer als <math.h>

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 zeigeTrapezsumme(TextBox^ tb, int n)
{
    double t = TrapezSumme(0, 1, n);
    tb->AppendText("Trapezsumme=" + (4 * t).ToString() +
        System::Environment::NewLine);
}

// ----- 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 zeigeMises(TextBox^ tb)
{
    for (int i = 1; i < 50; i++)
        // if (i%5 == 0)
        tb->AppendText("n=" + i.ToString() + " p=" + Mises(i).ToString() +
            System::Environment::NewLine);
}

```

```
}  
  
// ----- Aufgabe 7. -----  
  
int RoundToInt(double x)  
{  
    if (x >= 0)  
        return x + 0.5;  
    else  
        return x - 0.5;  
}  
  
void zeigeRoundToInt(TextBox^ tb)  
{  
    for (double x = -2; x < 2; x = x + 0.1)  
        tb->AppendText(x.ToString() + ": " + RoundToInt(x).ToString() +  
            System::Environment::NewLine);  
}  
  
// ----- Aufgabe 8. -----  
  
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.  
}
```



## 2.15 Aufgabe 3.6.6 Systematisches Testen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
```

```
// ----- 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:  $1 \leq n \leq 9$ ; 2 mal:  $10 \leq n \leq 99$

Das wird mit den folgenden Testfällen erreicht:

```
{n=-1;Quersumme=1 | n=0;Quersumme=0 | n=12;Quersumme=3 }
```

b) Schleife 0 mal:  $n \leq 0$ ; 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:  $4 \leq n \leq 8$ , if nie:  $n = 5, 7$ ; if ja:  $n = 4, 6, 8$

2 mal:  $9 \leq n \leq 15$ , if nie:  $n = 11, 13$ ; if ja:  $9, 10, 12, 14, 15$

Das wird mit den folgenden Testfällen erreicht:

```
{n=1;istPrim=false | n=2;istPrim=true | n=3;istPrim=true | n=4;istPrim=false |  
n=5;istPrim=true | n=9;istPrim=false | n=11;istPrim=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:  $n \leq 1$   
 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:  $n \leq 1$   $n = 1$   
 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)  $2+2^{\sqrt{\text{MAX}}}=2+2^{65536}$  Tests
- d)  $1+2^{((\text{MAX}-4)/2)}$  Tests

\*/

// ----- Aufgabe 2. -----

/\*

- Funktionen, deren Ergebnis sehr aufwendig zu berechnen ist, wie z.B. Wettervorhersagen.
- Funktionen mit Zufallszahlen (wie z.B. RegentropfenPi) 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.

- Das Ergebnis der Funktion

```
HypothekenRestschuld(double Betrag, double Zinssatz, double Tilgungsrate)
```

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
```

- 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 zeitaufwendig.

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.

\*/

## 2.16 Aufgabe 3.6.7 Unittests

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h

#ifdef SIMPLEUNITTESTS_H
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

// ----- Aufgabe 1 -----

void Unittests_QuerSumme()
{ // ein einfacher Test für die Funktion QuerSumme
  // a) {n=-1;Quersumme=1 | n=0;Quersumme=0 | n=12;Quersumme=3 }
  rkl::Assert::AreEqual(QuerSumme(-1), 1, "Quersumme(-1)");
  rkl::Assert::AreEqual(QuerSumme(0), 0, "Quersumme( 0)");
  rkl::Assert::AreEqual(QuerSumme(12), 3, "Quersumme(12)");
}

void Unittests_Fibonacci()
{ // b) {n=-1,Fibonacci=0 | n=0,Fibonacci=0 | n=1,Fibonacci=1 |
  //      n=2,Fibonacci=1 }
  rkl::Assert::AreEqual(Fibonacci(-1), 0, "Fibonacci(-1)");
  rkl::Assert::AreEqual(Fibonacci(0), 0, "Fibonacci(0)");
  rkl::Assert::AreEqual(Fibonacci(1), 1, "Fibonacci(1)");
  rkl::Assert::AreEqual(Fibonacci(2), 1, "Fibonacci(2)");
}

void Unittests_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 }
  rkl::Assert::AreEqual(istPrim(1), false, "istPrim(1)");
  rkl::Assert::AreEqual(istPrim(2), true, "istPrim(2)");
  rkl::Assert::AreEqual(istPrim(3), true, "istPrim(3)");
  rkl::Assert::AreEqual(istPrim(4), false, "istPrim(4)");
  rkl::Assert::AreEqual(istPrim(5), true, "istPrim(5)");
  rkl::Assert::AreEqual(istPrim(9), false, "istPrim(9)");
  rkl::Assert::AreEqual(istPrim(11), true, "istPrim(11)");
}

void Unittests_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 }
  rkl::Assert::AreEqual(Goldbach(2), -1, "Goldbach(2)");
  rkl::Assert::AreEqual(Goldbach(3), -1, "Goldbach(3)");
  rkl::Assert::AreEqual(Goldbach(4), 1, "Goldbach(4)");
  rkl::Assert::AreEqual(Goldbach(5), -1, "Goldbach(5)");
  rkl::Assert::AreEqual(Goldbach(6), 1, "Goldbach(6)");
  rkl::Assert::AreEqual(Goldbach(100), 6, "Goldbach(100)");
  rkl::Assert::AreEqual(Goldbach(1000), 28, "Goldbach(1000)");
}

void Unittests_PythagTripel(TextBox^ tb)
{ // e) {n=0;Pythag=0 | n=1;Pythag=0 | n=5;Pythag=1 | n=50;Pythag=26}
  rkl::Assert::AreEqual(zeigePythagTripel(tb, 0, false), 0, "zeige_PythagTripel(0)");
  rkl::Assert::AreEqual(zeigePythagTripel(tb, 1, false), 0, "zeige_PythagTripel(1)");
  rkl::Assert::AreEqual(zeigePythagTripel(tb, 5, false), 1, "zeige_PythagTripel(5)");
  rkl::Assert::AreEqual(zeigePythagTripel(tb, 50, false), 26, "zeige_PythagTripel(26)");
}

void Unittests_f3nplus1(TextBox^ tb)
{ // f) {n=1;f3nplus1=0 | n=2;f3nplus1=1 | n=3;f3nplus1=7 |
  //      n=4;f3nplus1=2 }
  rkl::Assert::AreEqual(f3nplus1(tb, 1, false), 0, "f3nplus1(1)");
  rkl::Assert::AreEqual(f3nplus1(tb, 2, false), 1, "f3nplus1(2)");
  rkl::Assert::AreEqual(f3nplus1(tb, 3, false), 7, "f3nplus1(3)");
  rkl::Assert::AreEqual(f3nplus1(tb, 4, false), 2, "f3nplus1(4)");
}

```

```

void Unittests_Fakultaet()
{ // 3. {n=0,Fakultaet=1 | n=1,Fakultaet=1 | n=2,Fakultaet=2 |
  //      n=3,Fakultaet=6 }
  bool result = true;
  rkl::Assert::AreEqual(Fakultaet(0), 1, "Fakultaet(0)");
  rkl::Assert::AreEqual(Fakultaet(1), 1, "Fakultaet(1)");
  rkl::Assert::AreEqual(Fakultaet(2), 2, "Fakultaet(2)");
  rkl::Assert::AreEqual(Fakultaet(3), 6, "Fakultaet(3)");

  // 3. {n=0,doubleFakulaet=1 | n=1,doubleFakulaet=1 |
  //      n=2,doubleFakulaet=2 | n=3,doubleFakulaet=6 }
  rkl::Assert::AreEqual(doubleFakulaet(0), 1, "doubleFakulaet(0)");
  rkl::Assert::AreEqual(doubleFakulaet(1), 1, "doubleFakulaet(1)");
  rkl::Assert::AreEqual(doubleFakulaet(2), 2, "doubleFakulaet(2)");
  rkl::Assert::AreEqual(doubleFakulaet(3), 6, "doubleFakulaet(3)");
}

// 4. Mit den bisher vorgestellten Sprachelementen sind keine
//      automatischen Tests möglich

// 5. Mit den bisher vorgestellten Sprachelementen sind keine
//      automatischen Tests möglich

void Unittests_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) }
  rkl::Assert::AreEqual(Mises(1), 0, "Mises(1)");
  rkl::Assert::AreEqual(Mises(2), 1 - 364.0 / 365, "Mises(2)");
  rkl::Assert::AreEqual(Mises(3), 1 - (364.0 * 363) / (365.0 * 365), "Mises(3)");
  rkl::Assert::AreEqual(Mises(4), 1 - (364.0 * 363 * 362) / (365.0 * 365 * 365),
    "Mises(4)");
}

void Unittests_RoundToInt()
{ // 7. {n=3.64,RoundToInt=4 | n=3.14,RoundToInt=3 |
  //      n=-3.14,RoundToInt=-3 | n=-3.64,RoundToInt=-4 }
  rkl::Assert::AreEqual(RoundToInt(3.64), 4, "RoundToInt(3.64)");
  rkl::Assert::AreEqual(RoundToInt(3.14), 3, "RoundToInt(3.14)");
  rkl::Assert::AreEqual(RoundToInt(-3.14), -3, "RoundToInt(-3.14)");
  rkl::Assert::AreEqual(RoundToInt(-3.64), -4, "RoundToInt(-3.64)");
}

void Unittests_3_6(TextBox^ tb)
{
  rkl::Assert::Init("Unittests_3_6"); // Tests initialisieren
  Unittests_QuerSumme();
  Unittests_Fibonacci();
  Unittests_prim();
  Unittests_Goldbach();
  Unittests_PythagTripel(tb);
  Unittests_f3nplus1(tb);

  Unittests_Fakultaet();
  Unittests_RoundToInt();
  Unittests_Mises();
  rkl::Assert::Summary(); // Testzusammenfassung ausgeben
}
#endif // #ifndef SIMPLEUNITTESTS_H__

```

## 2.17 Aufgabe 3.6.8 Ablaufprotokolle

// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen\_ab\_Kap\_3\Loesung\_Kap\_3\_Elementar.h

### 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;
}
```

	<i>n</i>	<i>f</i>	<i>i</i>	<i>i&lt;=n</i>	<i>result</i>
	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

### 1. b) Ablaufprotokoll für *Fibonacci(4)*

	<i>x</i>	<i>y</i>	<i>f</i>	<i>i</i>	<i>i&lt;n</i>
int f=0,x=0,y=1;	0	1	0 // f <sub>0</sub>		
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 <sub>1</sub>		
i++ ;				1	
i<n					true
x=y;	0				
y=f;		1			
f=x+y			0+1 // f <sub>2</sub>		
i++ ;				2	
i<n					true
x=y;	1				
y=f;		1			
f=x+y			1+1=2 // f <sub>3</sub>		
i++ ;				3	
i<n					true
x=y;	1				
y=f;		2			
f=x+y;			1+2=3 // f <sub>4</sub>		
i++ ;				4	
i<n					false

1. c) 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;
}

```

	<i>n</i>	<i>s</i>	<i>n&gt;0</i>	<i>result</i>
	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

1. d) Ablaufprotokoll für *istPrim(17)*

```

bool istPrim(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.

	<i>n</i>	<i>i</i>	<i>i*i&lt;n</i>	<i>n%i==0</i>	<i>result</i>
	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 *istPrim(18)*

	<i>n</i>	<i>i</i>	<i>i*i&lt;n</i>	<i>n%i==0</i>	<i>result</i>
	18				
for (int i=2; i*i<=n; i++)					
i=2;		2			
i*i<=n;			true		
if (n%i==0) return false;					false

1. e) 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
}
```

	<i>n</i>	<i>q</i>	<i>i</i>	<i>i&lt;n-1</i>	<i>result</i>
	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-1;				false	
return 1-q;					1-q

## 2. 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 3.6.9.

## 2.18 Aufgabe 3.6.9 Symbolische Ablaufprotokolle

// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen\_ab\_Kap\_3\Loesung\_Kap\_3\_Elementar.h

1.a)  $n$  ungerade, d.h.  $n/2 = \frac{n-1}{2}$  :

$x$	$n$	$p$
$x_0$	$n_0$	$p_0$
$p = p * x$		$p_0 * x_0$
$n = n/2$	$\frac{n_0 - 1}{2}$	
$x = x * x$	$x_0 * x_0$	

$$p * x^n = (p_0 * x_0) * (x_0 * x_0)^{(n_0-1)/2} = p_0 * x_0 * x_0^{n_0-1} = p_0 * x_0^{n_0} = u^v$$

b)  $n$  gerade, d.h.  $n/2 = \frac{n}{2}$  :

$x$	$n$	$p$
$x_0$	$n_0$	$p_0$
$n = n/2$	$\frac{n_0}{2}$	
$x = x * x$	$x_0 * x_0$	

$$p * x^n = p_0 * (x_0 * x_0)^{(n_0/2)} = p_0 * x_0^{n_0} = u^v$$

c)  $s = 1 + 2 + \dots + i$

$s$	$i$
$1+2+\dots+i_0$	$i_0$
$i = i + 1$	$i_0+1$
$s = s + i$	$1+2+\dots+i_0+(i_0+1)$

Wegen  $i=i_0+1$  gilt anschließend

$$s = 1 + 2 + \dots + i_0 + (i_0 + 1) = 1 + 2 + \dots + (i-1) + i$$

d)  $s = 1^2 + 2^2 + \dots + i^2$

$s$	$i$
$1^2+2^2+\dots+i_0^2$	$i_0$
$i = i + 1$	$i_0 + 1$
$s = s + i * i$	$1^2+2^2+\dots+i_0^2+(i_0+1)^2$

Wegen  $i=i_0+1$  gilt anschließend

$$s = 1^2 + 2^2 + \dots + i_0^2 + (i_0 + 1)^2 = 1^2 + 2^2 + \dots + (i-1)^2 + i^2$$



$$e) s = 1^2 + 2^2 + \dots + i^2$$

	<i>s</i>	<i>i</i>
	$1^2 + 2^2 + \dots + i_0^2$	$i_0$
$s = s + i * i$	$1^2 + 2^2 + \dots + i_0^2 + i_0^2$	
$i = i + 1$		$i_0 + 1$

Wegen  $i=i_0+1$  gilt anschließend

$$s = 1^2 + 2^2 + \dots + i_0^2 + i_0^2 = 1^2 + 2^2 + \dots + 2 * (i-1)^2$$

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.

## 2.19 Aufgabe 3.9 Exception-Handling

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
int throwException(int n)
{
    if (n < 0)
        throw std::invalid_argument("throwException: negatives Argument nicht zulässig");
    return n;
}

int verschachtelterAufruf(int n)
{
    return throwException(n);
}

void Aufruf_throwException(TextBox ^ textBox1, int Aufgabe)
{
    int result = 0;
    if (Aufgabe == 1) // a)
    {
        try
        {
            result = throwException(-1);
        }
        catch (std::exception& e)
        {
            System::String^ msg = gcnew System::String(e.what());
            textBox1->AppendText(msg + "\r\n");
        }
    }
    else if (Aufgabe == 2) // b)
    {
        result = throwException(-1);
    }
    else // c)
    {
        try
        {
            result = verschachtelterAufruf(-1);
        }
        catch (std::exception& e)
        {
            System::String^ msg = gcnew System::String(e.what());
            textBox1->AppendText(msg + "\r\n");
        }
    }
}
```

## 2.20 Aufgabe 3.10 Namensbereiche

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h  
// Die Lösung dieser Aufgabe sind die Namensbereiche  
// in dieser Quelltextdatei.
```

## 2.21 Aufgabe 3.11 Präprozessoranweisungen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_3_Elementar.h
#include "a.h" // int i = 0;
#include "b.h" // #include "a.h"
                // int j = i + 1;

// a): Durch diese #include-Anweisungen wird die Definition von i
//      zirkulär (rekursiv) in die aktuelle Datei aufgenommen

// b): Das kann man mit einem #pragma once oder mit einem Include-Guard
//      unterbinden:
//      Entweder in die Dateien

//      #pragma once

//      aufnehmen, oder
//      a.h:
//      #ifndef _A_INCLUDE_GUARD
//      #define _A_INCLUDE_GUARD

//      int a = 0;

//      #endif /* _A_INCLUDE_GUARD */

//      b.h:
//      #ifndef _B_INCLUDE_GUARD
//      #define _B_INCLUDE_GUARD

//      #include "a.h"
//      int b = a + 1;

//      #endif /* _B_INCLUDE_GUARD */

#ifdef SIMPLEUNITTESTS_H__

void Unittests_Kap_3(TextBox ^ tb)
{
    rkl::Assert::Init("Unittests Kapitel 3"); // Tests initialisieren
    Unittests_3_6(tb);
    Unittests_Rekursion();
    rkl::Assert::Summary(); // Testzusammenfassung ausgeben
}
#endif // #ifdef SIMPLEUNITTESTS_H__

void zeige_Ergebnisse(TextBox ^ tb)
{
    Lsg_Variablen(tb);
    Lsg_Ganzzahl_Operatoren(tb);
    Lsg_char();
    Lsg_if(tb);
    // ...
    //
}

// Man kann auch die ButtonClick-Funktionen aufrufen. Dann muss
// die aufrufende Funktion in der Formularklasse definiert werden:

void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
    // button_plus1_Click(nullptr, e);
}

} // end of namespace N_Aufgaben_Elementare_Datentypen
```

### 3 Lösungen Kapitel 4. String

```
##### Copyright notice #####  
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook  
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",  
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_4_Strings.h
```

```
#include <string>  
#include <regex>
```

```
namespace N_Loesungen_Strings {
```

### 3.1 Aufgabe 4.3 Elementfunktionen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_4_Strings.h

using namespace System;
using namespace System::Windows::Forms;

// ----- Aufgabe 1 -----

// a)

int AnzahlZeichen(const std::string& s, char c)
{
    int n = 0;
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] == c)
            n++;
    }
    return n;
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_AnzahlZeichen()
{
    rkl::Assert::WriteLine("Unittests_AnzahlZeichen");
    rkl::Assert::AreEqual(0, AnzahlZeichen("", 'x'), "1. AnzahlZeichen 'x'");
    rkl::Assert::AreEqual(0, AnzahlZeichen(" ", 'e'), "2. AnzahlZeichen 'e'");
    rkl::Assert::AreEqual(1, AnzahlZeichen(" ", ' '), "3. AnzahlZeichen ' '");
    rkl::Assert::AreEqual(4, AnzahlZeichen("Alle meine Entchen", 'e'), "4. AnzahlZeichen
'e'");
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// b)

bool Contains(const std::string& s, const std::string& c)
{
    bool result = (s.find(c) != std::string::npos);
    return result;
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_Contains()
{
    rkl::Assert::WriteLine("Unittests_Contains");
    rkl::Assert::AreEqual(true, Contains("", ""), "Contains 0");
    rkl::Assert::AreEqual(true, Contains("x", "x"), "Contains 1");
    rkl::Assert::AreEqual(true, Contains("Maxl", "x"), "Contains 2");
    rkl::Assert::AreEqual(true, Contains("Maxxl", "xx"), "Contains 3");
    rkl::Assert::AreEqual(true, Contains("Maxxyz", "z"), "Contains 4");
    rkl::Assert::AreEqual(false, Contains("xyz", "a"), "Contains 5");
}
#endif // #ifdef SIMPLEUNITTESTS_H__
```

```

// c)

bool Split3(std::string s, std::string& s1, std::string& s2, std::string& s3)
{
    bool success = false;
    int p1 = s.find('.');
    if (p1 >= 1)
    {
        s1 = s.substr(0, p1);
        int p2 = s.find('.', p1 + 1);
        if (p2 >= p1 + 1)
        {
            s2 = s.substr(p1 + 1, p2 - p1 - 1);
            s3 = s.substr(p2 + 1);
            success = true; // 2 Punkte gefunden
        }
    }
    return success;
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void assertEquals(std::string s1_exp, std::string s2_exp, std::string s3_exp,
    bool res_exp,
    std::string s1_act, std::string s2_act, std::string s3_act,
    bool res_act, std::string msg)
{
    rk1::Assert::AreEqual(s1_exp, s1_act, "s1 " + msg);
    rk1::Assert::AreEqual(s2_exp, s2_act, "s2 " + msg);
    rk1::Assert::AreEqual(s3_exp, s3_act, "s3 " + msg);
    rk1::Assert::AreEqual(res_exp, res_act, "result " + msg);
}

void test_Split3(std::string s, std::string s1_exp, std::string s2_exp, std::string
s3_exp, bool res_exp)
{
    std::string t, m, j;
    bool result = Split3(s, t, m, j);
    assertEquals(t, m, j, result, s1_exp, s2_exp, s3_exp, res_exp, "test_Split3: " + s);
}

void Unittests_Split3()
{
    rk1::Assert::WriteLine("Unittests_Split3");
    bool result = true;
    // teste alle üblichen Kombinationen
    // Jahr zweistellig, Tag und Monat einstellig und zweistellig
    test_Split3("1.2.07", "1", "2", "07", true);
    test_Split3("12.3.07", "12", "3", "07", true);
    test_Split3("1.12.07", "1", "12", "07", true);
    test_Split3("17.18.07", "17", "18", "07", true);

    // Jahr vierstellig, Tag und Monat einstellig und zweistellig
    test_Split3("1.2.1920", "1", "2", "1920", true);
    test_Split3("12.3.1920", "12", "3", "1920", true);
    test_Split3("1.12.1920", "1", "12", "1920", true);
    test_Split3("17.18.1920", "17", "18", "1920", true);

    // Teste unzulässige Formate. Das Programm sollte nicht abstürzen:
    test_Split3("", "", "", "", false);
    test_Split3(".", "", "", "", false);
    test_Split3("1", "", "", "", false);
    test_Split3("1.", "1", "", "", false);
    test_Split3("1.2", "1", "", "", false);
    test_Split3("..", "", "", "", false);
    test_Split3("...", "", "", "", false);
    test_Split3("abc", "", "", "", false);
    test_Split3("1. Dezember ", "1", "", "", false);
}
#endif // #ifdef SIMPLEUNITTESTS_H__

```





```

// d)

std::string TrimLeft(const std::string& str)
{
    size_t first = str.find_first_not_of(' ');
    if (std::string::npos == first)
    { // alle Zeichen sind ' '
        return ""; // str;
    }
    else // 0<=first<str.length()
    { // bis zur Position first alles weglassen
        return str.substr(first);
    }
}

std::string TrimRight(const std::string& str)
{
    size_t first = str.find_last_not_of(' ');
    if (std::string::npos == first)
    { // alle Zeichen sind ' '
        return "";
    }
    else
    {
        return str.substr(0, first + 1);
    }
}

#ifdef SIMPLEUNITTESTS_H
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_Trim()
{
    std::string s = TrimLeft(" abc "); // s = "abc "
    rkl::Assert::WriteLine("Unittests_Trim");
    rkl::Assert::AreEqual("", TrimLeft(""), "TrimLeft 0");
    rkl::Assert::AreEqual("x", TrimLeft("x"), "TrimLeft 1");
    rkl::Assert::AreEqual("x", TrimLeft(" x"), "TrimLeft 2");
    rkl::Assert::AreEqual("x ", TrimLeft(" x "), "TrimLeft 3");
    rkl::Assert::AreEqual("x y", TrimLeft(" x y"), "TrimLeft 4");
    rkl::Assert::AreEqual("x y ", TrimLeft(" x y "), "TrimLeft 5");
    rkl::Assert::AreEqual("", TrimLeft(" "), "TrimLeft 6");
    rkl::Assert::AreEqual("", TrimLeft(" "), "TrimLeft 7");

    rkl::Assert::AreEqual("", TrimRight(""), "TrimRight 0");
    rkl::Assert::AreEqual("x", TrimRight("x"), "TrimRight 1");
    rkl::Assert::AreEqual(" x", TrimRight(" x"), "TrimRight 2");
    rkl::Assert::AreEqual(" x", TrimRight(" x "), "TrimRight 3");
    rkl::Assert::AreEqual(" x y", TrimRight(" x y"), "TrimRight 4");
    rkl::Assert::AreEqual(" x y", TrimRight(" x y "), "TrimRight 5");
    rkl::Assert::AreEqual("", TrimRight(" "), "TrimRight 6");
    rkl::Assert::AreEqual("", TrimRight(" "), "TrimRight 7");
}
#endif // #ifdef SIMPLEUNITTESTS_H

```

```

// e)

int ReplaceAll(std::string& str, const std::string& from, const std::string& to)
{
    int n = 0;
    if (from == "")
        return 0;
    size_t pos = 0;
    while (pos != std::string::npos)
    {
        str.replace(pos, from.length(), to);
        n++;
        pos += to.length();
        pos = str.find(from, pos);
    }
    return n;
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_ReplaceAll()
{
    rkl::Assert::WriteLine("Unittests_ReplaceAll");
    std::string s = "";
    rkl::Assert::AreEqual(0, ReplaceAll(s, "", "x"), "ReplaceAll 0");
    rkl::Assert::AreEqual("", s, "ReplaceAll 0s");
    s = "a";
    rkl::Assert::AreEqual(1, ReplaceAll(s, "a", "x"), "ReplaceAll 1");
    rkl::Assert::AreEqual("x", s, "ReplaceAll 1s");
    s = "aa";
    rkl::Assert::AreEqual(2, ReplaceAll(s, "a", "x"), "ReplaceAll 2");
    rkl::Assert::AreEqual("xx", s, "ReplaceAll 2s");
    s = "baa";
    rkl::Assert::AreEqual(2, ReplaceAll(s, "a", "x"), "ReplaceAll 3");
    rkl::Assert::AreEqual("bxx", s, "ReplaceAll 3s");
    s = "baab";
    rkl::Assert::AreEqual(2, ReplaceAll(s, "a", "x"), "ReplaceAll 4");
    rkl::Assert::AreEqual("bxxb", s, "ReplaceAll 4s");

    s = "baabaa";
    rkl::Assert::AreEqual(2, ReplaceAll(s, "ba", "x"), "ReplaceAll 10");
    rkl::Assert::AreEqual("xaxa", s, "ReplaceAll 10s");

    s = "baabaa";
    rkl::Assert::AreEqual(2, ReplaceAll(s, "aa", "xyz"), "ReplaceAll 11");
    rkl::Assert::AreEqual("bxyzbxyz", s, "ReplaceAll 11s");
}
#endif // #ifdef SIMPLEUNITTESTS_H__

```

```
// ----- Aufgabe 2. -----  
// a)  
void tokenize_0(TextBox^ tb, const std::string& s, std::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.  
  using std::string;  
  tb->AppendText("tokenize string: " + rkl::to_String(s) + "\r\n");  
  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);  
    tb->AppendText("token: " + rkl::to_String(token) + "\r\n");  
    pos_tok = s.find_first_not_of(separators, pos_sep + 1);  
  }  
}  
  
void zeigeTokenize_0(TextBox^ tb)  
{  
  tokenize_0(tb, ""); // leerer String , keine Ausgabe  
  tokenize_0(tb, " "); // nur Separator, keine Ausgabe  
  tokenize_0(tb, "123"); // nur ein token, Ausgabe "123"  
  tokenize_0(tb, ";123"); // Separator am Anfang, ein token, Ausgabe "123"  
  tokenize_0(tb, "123;"); // Separator am Ende, ein token, Ausgabe "123"  
  tokenize_0(tb, ";123;"); // Separator am Anfang und Ende, ein token, Ausgabe "123"  
  tokenize_0(tb, "456 ab.xy"); // Ausgabe "456", "ab", "xy"  
  tokenize_0(tb, " 456 ab.xy"); // Separator am Anfang, Ausgabe "456","ab","xy"  
  tokenize_0(tb, "456 ab.xy"); // Separator am Ende, Ausgabe "456","ab","xy"  
  tokenize_0(tb, " 456 ab.xy;"); // Separator am Anfang und am Ende, Ausgabe  
    // "456", "ab", "xy"  
}  
  
void zeigeTokenize_0(TextBox^ tbin, TextBox^ tbout)  
{  
  std::string in = rkl::to_string(tbin->Text);  
  tokenize_0(tbout, in);  
}
```

```
// ----- Aufgabe 3. -----

std::string generatePassword(int n)
{ // sehr einfach
  srand(std::time(0)); // #include <ctime>
  std::string result;
  // for (int i = 0; i < n; i++) // kann zu result.length()<n führen
  while (result.length() < n)
  {
    char c = rand() % 127;
    if (c > ' ' && c < 'z')
      result += c;
  }
  return result;
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_generatePassword()
{
  rkl::Assert::WriteLine("Unittests_generatePassword");
  // Ohne (int) ist der Aufruf mehrdeutig, da i den Datentyp int
  // und length den Datentyp size_t hat.
  for (int i = 0; i < 10; i++)
    rkl::Assert::AreEqual(i, (int)generatePassword(i).length(),
      "generatePassword(" + std::to_string(i) + ")");
  // Da die erzeugten Passwörter zufällig sind,
  // nur die Länge prüfen.
}
#endif // #ifdef SIMPLEUNITTESTS_H__
```

```
// ----- Aufgabe 4. -----

std::string uintToBinaryString(unsigned int n)
{
    std::string 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;
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_uintToBinaryString()
{
    rkl::Assert::WriteLine("Unittests_uintToBinaryString");
    std::string s = "00000000000000000000000000000000";
    rkl::Assert::AreEqual(uintToBinaryString(0), s, "toBinStr 0");
    s = "0000000000000000000000000000000001";
    rkl::Assert::AreEqual(uintToBinaryString(1), s, "toBinStr 1");
    s = "0000000000000000000000000000000010";
    rkl::Assert::AreEqual(uintToBinaryString(2), s, "toBinStr 2");
    s = "0000000000000000000000000000000011";
    rkl::Assert::AreEqual(uintToBinaryString(3), s, "toBinStr 3");
    s = "11111111111111111111111111111111";
    rkl::Assert::AreEqual(uintToBinaryString(-1), s, "toBinStr -1");
}
#endif // #ifdef SIMPLEUNITTESTS_H__

void show_uintToBinaryString(TextBox^ tb, String^ s)
{
    for (int i = 0; i <= 8; i++)
        tb->AppendText(rkl::to_String(uintToBinaryString(i)) + "\r\n");

    int x;
    if (int::TryParse(s, x))
    {
        std::string result = uintToBinaryString(x);
        tb->AppendText(rkl::to_String(result) + "\r\n");
    }
    else
    {
        MessageBox::Show("int-Wert eingeben");
    }
}
}
```

```
// ----- Aufgabe 5. -----

int Max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}

std::string BigIntAdd(std::string a, std::string b)
{
    std::string s;
    int m;
    m = Max(a.length(), b.length());
    int ue = 0;
    for (int i = 0; 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;
        std::string
            ziffer[] = { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9" };
        s = ziffer[r] + s;
    }
    if (ue > 0) s = "1" + s;
    return s;
}

void showBigIntAdd(TextBox^ tb, String^ a, String^ b)
{ // Diese Funktion kann bei einem ButtonClick aufgerufen werden:
  // showBigIntAdd(textBox1, textBox2->Text, textBox3->Text);
  int x, y;
  if (int::TryParse(a, x) && int::TryParse(b, y))
  {
      std::string sa = rkl::to_string(a);
      std::string sb = rkl::to_string(b);
      tb->AppendText(a + "+" + b + "=" + rkl::to_string(BigIntAdd(sa, sb)) + "\r\n");
  }
  else
      MessageBox::Show("int-Werte eingeben");

  return;
  // oder
  std::string s = "1";
  for (int i = 1; i <= 100; i++)
  {
      s = BigIntAdd(s, s);
      tb->AppendText("2^" + i.ToString() + "=" + rkl::to_string(s) + "\r\n");
  }
}

#ifdef SIMPLEUNITTESTS_H
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_BigIntAdd(int n)
{
    rkl::Assert::WriteLine("Unittests_BigIntAdd");
    std::string s = "0";
    rkl::Assert::AreEqual(BigIntAdd("0", "0"), s, "BigIntAdd: 0+0==0");
    s = "2";
    rkl::Assert::AreEqual(BigIntAdd("1", "1"), s, "BigIntAdd: 1+1==2");
    s = "3";
    rkl::Assert::AreEqual(BigIntAdd("1", "2"), s, "BigIntAdd: 1+2==3");
}
#endif

```

```
s = "10";
rk1::Assert::AreEqual(BigIntAdd("1", "9"), s, "BigIntAdd: 1+9==10");
// Check random numbers:
for (int i = 0; i < n; i++)
{
    int op1 = rand();
    int op2 = rand();
    std::string op1s = std::to_string(op1);
    std::string op2s = std::to_string(op2);
    std::string s = std::to_string(op1 + op2);
    rk1::Assert::AreEqual(BigIntAdd(op1s, op2s), s,
        "BigIntAdd: " + op1s + "+" + op2s + "=" + s);
}
}
#endif // #ifndef SIMPLEUNITTESTS_H__
```





## 3.2 Aufgabe 4.5.1 Stringkonversionen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_4_Strings.h

// ----- Aufgabe 1 -----

bool stringToDate(std::string date, int& Tag, int& Monat, int& Jahr)
{
    bool success = false;
    Tag = -1;
    Monat = -1;
    Jahr = -1;
    std::string dstr, mstr, ystr;
    if (Split3(date, dstr, mstr, ystr))
    {
        success = true; // 2 Punkte gefunden
        try { // oder mit try_parse aus dem Buchmanuskript
            Tag = std::stoi(dstr);
            Monat = std::stoi(mstr);
            Jahr = std::stoi(ystr);
        }
        catch (...)
        {
            success = false;
        }
    }
    return success;
}

void showStringToDate(TextBox^ tb, std::string s)
{
    int Tag = 0, Monat = 0, Jahr = 0;
    if (stringToDate(s, Tag, Monat, Jahr))
        tb->AppendText(rkl::to_String(s) + ": Tag=" + Tag.ToString() + " Monat=" +
Monat.ToString() +
            " Jahr=" + Jahr.ToString() + "\r\n");
    else
        tb->AppendText(rkl::to_String(s) + "kann nicht konvertiert werden");
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void AssertEqual_Date(std::string s, int Tag_exp, int Mon_exp, int Jahr_exp, bool res_exp)
{ // prüft alle Ergebnisse von stringToDate
    int t, m, j;
    bool res_act = stringToDate(s, t, m, j);
    if (res_act)
    {
        rkl::Assert::AreEqual(Tag_exp, t, "Tag " + s);
        rkl::Assert::AreEqual(Mon_exp, m, "Monat " + s);
        rkl::Assert::AreEqual(Jahr_exp, j, "Jahr " + s);
    }
    rkl::Assert::AreEqual(res_exp, res_act, "result " + s);
}

void Unittests_StringToDate()
{
    rkl::Assert::WriteLine("Unittests_StringToDate");
    bool result = true;
    // teste alle üblichen Kombinationen
    // Jahr zweistellig, Tag und Monat einstellig und zweistellig
    AssertEqual_Date("1.2.07", 1, 2, 7, true);
    AssertEqual_Date("12.3.07", 12, 3, 7, true);
    AssertEqual_Date("1.12.07", 1, 12, 7, true);
    AssertEqual_Date("17.18.07", 17, 18, 7, true);
}
#endif
```

```
// Jahr vierstellig, Tag und Monat einstellig und zweistellig
AssertEqual_Date("1.2.1920", 1, 2, 1920, true);
AssertEqual_Date("12.3.1920", 12, 3, 1920, true);
AssertEqual_Date("1.12.1920", 1, 12, 1920, true);
AssertEqual_Date("17.18.1920", 17, 18, 1920, true);

// Teste unzulässige Formate. Das Programm sollte nicht abstürzen
// und -1 zurückgeben:
AssertEqual_Date("", -1, -1, -1, false);
AssertEqual_Date(".", -1, -1, -1, false);
AssertEqual_Date("1", -1, -1, -1, false);
AssertEqual_Date("1.", -1, -1, -1, false);
AssertEqual_Date("1.2", -1, -1, -1, false);
AssertEqual_Date("..", -1, -1, -1, false);
AssertEqual_Date("...", -1, -1, -1, false);
AssertEqual_Date("abc", -1, -1, -1, false);
AssertEqual_Date("1. Dezember ", -1, -1, -1, false);
}
#endif // #ifndef SIMPLEUNITTESTS_H__

// ----- Aufgabe 2 -----

// Den Datentyp des Parameters 'result' in 'double' ändern
// 'stoi' durch 'stod' ersetzen:
bool try_parse(std::string s, double& result)
{
    try {
        size_t idx;
        result = std::stod(s, &idx);
        return (idx == s.length()); // Alles wurde
    } // konvertiert
    catch (...) {
        return false;
    }
}
```



### 3.3 Aufgabe 4.8 Reguläre Ausdrücke

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_4_Strings.h
// Die regulären Ausdrücke sind teilweise etwas vereinfacht.

// a)
bool Contains2(std::string Line, std::string s1, std::string s2)
{ // Line enthält s1 und dann s2, dazwischen beliebige Zeichen
  const std::string p = s1 + ".*" + s2;
  try
  {
    const std::regex pattern(p);
    return std::regex_search(Line, pattern);
  }
  catch (std::exception& e)
  {
    MessageBox::Show("Exception '" + rkl::to_String(p) + "' \r\n");
    return false;
  }
}

// b)
bool ContainsSpacesBetween2(std::string Line, std::string s1, std::string s2)
{ // Line enthält s1 und dann s2, dazwischen Leerzeichen
  const std::string p = s1 + "\\s*" + s2;
  const std::string p_raw = s1 + R"(\s*)" + s2;
  try
  {
    const std::regex pattern(p_raw);
    return std::regex_search(Line, pattern);
  }
  catch (std::exception& e)
  {
    MessageBox::Show("Exception '" + rkl::to_String(p) + "' \r\n");
    return false;
  }
}

// c)
bool ContainsSpacesBetween2Words(std::string Line, std::string s1, std::string s2)
{ // Line enthält s1 und s2 als eigenständige Worte, dazwischen Leerzeichen
  const std::string p = "\\b(" + s1 + ")\\s*\\b(" + s2 + ")";
  const std::string p_raw = R"(\b())" + s1 + R"()\s*\b()" + s2 + R"()";
  try
  {
    const std::regex pattern(p_raw);
    return std::regex_search(Line, pattern);
  }
  catch (std::exception& e)
  {
    MessageBox::Show("Exception '" + rkl::to_String(p) + "' \r\n");
    return false;
  }
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_Contains2()
{
  using std::string;
  string id = "RegExpr Contains2 ";
  rkl::Assert::WriteLine("Unittests_" + id);

  string s1 = "int x=1";
  string s2 = "int x=1";
  string s3 = "int x =1";
  string s4 = "uint x=1";
  string s5 = "uint x=1";
  string s6 = "uint x =1";
}
#endif
```

```
string s7 = "int ax=1";
string s8 = "int ax=1";
string s9 = "int ax =1";

// a)
rk1::Assert::AreEqual(true, Contains2(s1, "int", "x"), id + "C2-1");
rk1::Assert::AreEqual(true, Contains2(s2, "int", "x"), id + "C2-2");
rk1::Assert::AreEqual(true, Contains2(s3, "int", "x"), id + "C2-3");
rk1::Assert::AreEqual(true, Contains2(s4, "int", "x"), id + "C2-4");
rk1::Assert::AreEqual(true, Contains2(s5, "int", "x"), id + "C2-5");
rk1::Assert::AreEqual(true, Contains2(s6, "int", "x"), id + "C2-6");
rk1::Assert::AreEqual(true, Contains2(s7, "int", "x"), id + "C2-7");
rk1::Assert::AreEqual(true, Contains2(s8, "int", "x"), id + "C2-8");
rk1::Assert::AreEqual(true, Contains2(s9, "int", "x"), id + "C2-9");

// b)
id = "Regexr ContainsSpacesBetween2 ";
rk1::Assert::WriteLine("Unittests_" + id);
rk1::Assert::AreEqual(true,
    ContainsSpacesBetween2(s1, "int", "x"), id + "CS2-1");
rk1::Assert::AreEqual(true,
    ContainsSpacesBetween2(s2, "int", "x"), id + "CS2-2");
rk1::Assert::AreEqual(true,
    ContainsSpacesBetween2(s3, "int", "x"), id + "CS2-3");
rk1::Assert::AreEqual(true,
    ContainsSpacesBetween2(s4, "int", "x"), id + "CS2-4");
rk1::Assert::AreEqual(true,
    ContainsSpacesBetween2(s5, "int", "x"), id + "CS2-5");
rk1::Assert::AreEqual(true,
    ContainsSpacesBetween2(s6, "int", "x"), id + "CS2-6");
rk1::Assert::AreEqual(false,
    ContainsSpacesBetween2(s7, "int", "x"), id + "CS2-7");
rk1::Assert::AreEqual(false,
    ContainsSpacesBetween2(s8, "int", "x"), id + "CS2-8");
rk1::Assert::AreEqual(false,
    ContainsSpacesBetween2(s9, "int", "x"), id + "CS2-9");

// c)
id = "Regexr ContainsSpacesBetween2Words ";
rk1::Assert::WriteLine("Unittests_" + id);
rk1::Assert::AreEqual(true,
    ContainsSpacesBetween2Words(s1, "int", "x"), id + "CS2w-1");
rk1::Assert::AreEqual(true,
    ContainsSpacesBetween2Words(s2, "int", "x"), id + "CS2w-2");
rk1::Assert::AreEqual(true,
    ContainsSpacesBetween2Words(s3, "int", "x"), id + "CS2w-3");
rk1::Assert::AreEqual(false,
    ContainsSpacesBetween2Words(s4, "int", "x"), id + "CS2w-4");
rk1::Assert::AreEqual(false,
    ContainsSpacesBetween2Words(s5, "int", "x"), id + "CS2w-5");
rk1::Assert::AreEqual(false,
    ContainsSpacesBetween2Words(s6, "int", "x"), id + "CS2w-6");
rk1::Assert::AreEqual(false,
    ContainsSpacesBetween2Words(s7, "int", "x"), id + "CS2w-7");
rk1::Assert::AreEqual(false,
    ContainsSpacesBetween2Words(s8, "int", "x"), id + "CS2w-8");
rk1::Assert::AreEqual(false,
    ContainsSpacesBetween2Words(s9, "int", "x"), id + "CS2w-9");
}
#endif // #ifdef SIMPLEUNITTESTS_H__
```

```

// d)
bool FunctionCall(std::string Line, std::string FunctionName)
{ // FunctionName, beliebig viele Leerzeichen, (
  const std::string p = "\\b(" + FunctionName + ")\s*\b(";
  const std::string p_raw = R"(\b()" + FunctionName + R"()\s*\b()";
  try
  {
    const std::regex pattern(p_raw);
    return std::regex_search(Line, pattern);
  }
  catch (std::exception& e)
  {
    MessageBox::Show("Exception '" + rkl::to_String(p) + "' \r\n");
    return false;
  }
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_FunctionCall()
{
  // d)
  std::string id = "RegExpr FunctionCall ";
  rkl::Assert::WriteLine("Unittests_" + id);
  rkl::Assert::AreEqual(false, FunctionCall("f12=12", "f12"), id + "1");
  rkl::Assert::AreEqual(true, FunctionCall("y = f12(x)", "f12"), id + "2");
  rkl::Assert::AreEqual(true, FunctionCall("y = f12( x)", "f12"), id + "3");
  rkl::Assert::AreEqual(true, FunctionCall("y = f12 (x)", "f12"), id + "4");
  rkl::Assert::AreEqual(true, FunctionCall("y = f12 ( x", "f12"), id + "5");
  rkl::Assert::AreEqual(true, FunctionCall(" int f12( ", "f12"), id + "6");
  rkl::Assert::AreEqual(false, FunctionCall(" f123( ", "f12"), id + "7");
  rkl::Assert::AreEqual(false, FunctionCall(" af123( ", "f12"), id + "8");
  rkl::Assert::AreEqual(false, FunctionCall(" af12( ", "f12"), id + "9");
  rkl::Assert::AreEqual(false, FunctionCall("f1 2(", "f12"), id + "10");
  rkl::Assert::AreEqual(false, FunctionCall("f12", "f12"), id + "11");
}
#endif // #ifdef SIMPLEUNITTESTS_H__

```

```
// e)
bool AssignmentOf(std::string Line, std::string VariableName)
{ // Ein =, beliebig viele Leerzeichen, VariableName
  std::string p = "{1}\\s*(" + VariableName + ")";
  std::string p_raw = R"({1}\\s*(" + VariableName + R"())";
  try
  {
    const std::regex pattern(p_raw);
    return std::regex_search(Line, pattern);
  }
  catch (std::exception& e)
  {
    MessageBox::Show("Exception '" + rkl::to_String(p) + "' \r\n");
    return false;
  }
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_AssignmentOf()
{
  //e)
  std::string id = "RegExpr AssignmentOf ";
  rkl::Assert::WriteLine("Unittests_" + id);
  rkl::Assert::AreEqual(true, AssignmentOf("y = xyz", "xyz"), id + "1");
  rkl::Assert::AreEqual(true, AssignmentOf("y= xyz", "xyz"), id + "2");
  rkl::Assert::AreEqual(true, AssignmentOf("y =xyz", "xyz"), id + "3");
  rkl::Assert::AreEqual(true, AssignmentOf("int y=xyz", "xyz"), id + "4");
  rkl::Assert::AreEqual(true, AssignmentOf("xyz=xyz", "xyz"), id + "5");
  rkl::Assert::AreEqual(true, AssignmentOf(" y = xyz(", "xyz"), id + "6");
  rkl::Assert::AreEqual(false, AssignmentOf(" int xyz", "xyz"), id + "6");
  rkl::Assert::AreEqual(false, AssignmentOf(" xyz(", "xyz"), id + "6");
}
#endif // #ifdef SIMPLEUNITTESTS_H__
```

```

// f)
bool ClassDefinition(std::string Line)
{ // ohne Templates und weitere Sonderfälle
  const std::string p = "(class |struct )\\s*\\w*";
  const std::string p_raw = R"((class |struct )\s*\w*)";
  try
  {
    const std::regex pattern(p_raw);
    return std::regex_search(Line, pattern);
  }
  catch (std::exception& e)
  {
    MessageBox::Show("Exception '" + rkl::to_String(p) + "' \r\n");
    return false;
  }
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_ClassDefinition()
{
  //f)
  std::string id = "RegExpr ClassDefinition ";
  rkl::Assert::WriteLine("Unittests_" + id);
  rkl::Assert::AreEqual(true, ClassDefinition("class X {}"), id + "1");
  rkl::Assert::AreEqual(true, ClassDefinition("// class X {}"), id + "2");
  rkl::Assert::AreEqual(true, ClassDefinition(" class X:public A {}"), id + "3");
  rkl::Assert::AreEqual(true, ClassDefinition(" struct X {}"), id + "4");
  rkl::Assert::AreEqual(false, ClassDefinition("class1 X {}"), id + "5");
  rkl::Assert::AreEqual(false, ClassDefinition("321"), id + "6");
}
#endif // #ifdef SIMPLEUNITTESTS_H__

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_RegExpr()
{
  Unittests_Contains2();
  Unittests_FunctionCall();
  Unittests_AssignmentOf();
  Unittests_ClassDefinition();
}
#endif // #ifdef SIMPLEUNITTESTS_H__

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_Kap_4_strings()
{
  rkl::Assert::Init("Unittests_Kap_4_Strings"); // rkl::Assert::LogTestCases::No );
  Unittests_AnzahlZeichen();
  Unittests_Contains();
  Unittests_Split3();
  Unittests_Trim();
  Unittests_ReplaceAll();
  Unittests_generatePassword();
  Unittests_uintToBinaryString();
  Unittests_BigIntAdd(100);
  Unittests_StringToDate();
  Unittests_RegExpr();
  rkl::Assert::Summary();
}
#endif // #ifdef SIMPLEUNITTESTS_H__
} // end of namespace N_Loesungen_Strings

```



### 3.4 Form1.h für Kapitel 4

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Form1-Kap-4.h
#pragma once
```

```
#include "d:\CPPUtils\CppCLR_Utils.h"
#include "D:\CPPUtils\SimpleUnitTests.h"
#include "d:\cpp-2015.bo0\Loesungen\Loesung_Kap_4_Strings.h"
```

```
namespace CppCLR_WinformsProjekt {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Zusammenfassung für Form1
    /// </summary>
    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO: Konstruktorcode hier hinzufügen.
            //
        }

    protected:
        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }
    private: System::Windows::Forms::GroupBox^ groupBox1;
    private: System::Windows::Forms::Button^ button_generatePassword;
    private: System::Windows::Forms::Button^ button_tokenize_0;
    private: System::Windows::Forms::Button^ button_ReplaceAll;

    private: System::Windows::Forms::Button^ button_Trim;
```

```

private: System::Windows::Forms::Button^ button_Split3;
private: System::Windows::Forms::Button^ button_Contains;
private: System::Windows::Forms::Button^ button_AnzahlZeichen;
private: System::Windows::Forms::Button^ button_BigIntAdd;
private: System::Windows::Forms::Button^ button_uintToBinaryString;
private: System::Windows::Forms::GroupBox^ groupBox2;

private: System::Windows::Forms::Button^ button_stringToDate;
private: System::Windows::Forms::TextBox^ textBox1;
private: System::Windows::Forms::GroupBox^ groupBox3;
private: System::Windows::Forms::TextBox^ textBox_out;

private: System::Windows::Forms::GroupBox^ groupBox4;
private: System::Windows::Forms::TextBox^ textBox2;
private: System::Windows::Forms::TextBox^ textBox3;

protected:

private:
    /// <summary>
    /// Erforderliche Designervariable.
    /// </summary>
    System::ComponentModel::Container ^components;

    /// <summary>
    /// Erforderliche Methode für die Designerunterstützung.
    /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
    /// </summary>
    void InitializeComponent(void)
    {
        this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
        this->button_BigIntAdd = (gcnew System::Windows::Forms::Button());
        this->button_uintToBinaryString = (gcnew System::Windows::Forms::Button());
        this->button_generatePassword = (gcnew System::Windows::Forms::Button());
        this->button_tokenize_0 = (gcnew System::Windows::Forms::Button());
        this->button_ReplaceAll = (gcnew System::Windows::Forms::Button());
        this->button_Trim = (gcnew System::Windows::Forms::Button());
        this->button_Split3 = (gcnew System::Windows::Forms::Button());
        this->button_Contains = (gcnew System::Windows::Forms::Button());
        this->button_AnzahlZeichen = (gcnew System::Windows::Forms::Button());
        this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
        this->button_stringToDate = (gcnew System::Windows::Forms::Button());
        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
        this->groupBox3 = (gcnew System::Windows::Forms::GroupBox());
        this->textBox3 = (gcnew System::Windows::Forms::TextBox());
        this->textBox2 = (gcnew System::Windows::Forms::TextBox());
        this->textBox_out = (gcnew System::Windows::Forms::TextBox());
        this->groupBox4 = (gcnew System::Windows::Forms::GroupBox());
        this->groupBox1->SuspendLayout();
        this->groupBox2->SuspendLayout();
        this->groupBox3->SuspendLayout();
        this->groupBox4->SuspendLayout();
        this->SuspendLayout();
        //
        // groupBox1
        //
        this->groupBox1->Controls->Add(this->button_BigIntAdd);
        this->groupBox1->Controls->Add(this->button_uintToBinaryString);
        this->groupBox1->Controls->Add(this->button_generatePassword);
        this->groupBox1->Controls->Add(this->button_tokenize_0);
        this->groupBox1->Controls->Add(this->button_ReplaceAll);
        this->groupBox1->Controls->Add(this->button_Trim);
        this->groupBox1->Controls->Add(this->button_Split3);
        this->groupBox1->Controls->Add(this->button_Contains);
        this->groupBox1->Controls->Add(this->button_AnzahlZeichen);
        this->groupBox1->Location = System::Drawing::Point(171, 12);
        this->groupBox1->Name = L"groupBox1";
        this->groupBox1->Size = System::Drawing::Size(234, 254);
        this->groupBox1->TabIndex = 0;
        this->groupBox1->TabStop = false;
    }

```

```

this->groupBox1->Text = L"Aufgaben 4.3";
//
// button_BigIntAdd
//
this->button_BigIntAdd->Location = System::Drawing::Point(19, 220);
this->button_BigIntAdd->Name = L"button_BigIntAdd";
this->button_BigIntAdd->Size = System::Drawing::Size(75, 23);
this->button_BigIntAdd->TabIndex = 8;
this->button_BigIntAdd->Text = L"5. BigIntAdd";
this->button_BigIntAdd->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
this->button_BigIntAdd->UseVisualStyleBackColor = true;
this->button_BigIntAdd->Click +=
    gcnw System::EventHandler(this, &Form1::button_BigIntAdd_Click);
//
// button_uintToBinaryString
//
this->button_uintToBinaryString->Location = System::Drawing::Point(19, 191);
this->button_uintToBinaryString->Name = L"button_uintToBinaryString";
this->button_uintToBinaryString->Size = System::Drawing::Size(123, 23);
this->button_uintToBinaryString->TabIndex = 7;
this->button_uintToBinaryString->Text = L"4. uintToBinaryString";
this->button_uintToBinaryString->UseVisualStyleBackColor = true;
this->button_uintToBinaryString->Click +=
    gcnw System::EventHandler(this, &Form1::button_uintToBinaryString_Click);
//
// button_generatePassword
//
this->button_generatePassword->Location = System::Drawing::Point(19, 162);
this->button_generatePassword->Name = L"button_generatePassword";
this->button_generatePassword->Size = System::Drawing::Size(123, 23);
this->button_generatePassword->TabIndex = 6;
this->button_generatePassword->Text = L"3. generatePassword";
this->button_generatePassword->TextAlign
System::Drawing::ContentAlignment::MiddleLeft;
this->button_generatePassword->UseVisualStyleBackColor = true;
this->button_generatePassword->Click +=
    gcnw System::EventHandler(this, &Form1::button_generatePassword_Click);
//
// button_tokenize_0
//
this->button_tokenize_0->Location = System::Drawing::Point(19, 133);
this->button_tokenize_0->Name = L"button_tokenize_0";
this->button_tokenize_0->Size = System::Drawing::Size(123, 23);
this->button_tokenize_0->TabIndex = 5;
this->button_tokenize_0->Text = L"2. tokenize_0";
this->button_tokenize_0->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
this->button_tokenize_0->UseVisualStyleBackColor = true;
this->button_tokenize_0->Click +=
    gcnw System::EventHandler(this, &Form1::button_tokenize_0_Click);
//
// button_ReplaceAll
//
this->button_ReplaceAll->Location = System::Drawing::Point(125, 86);
this->button_ReplaceAll->Name = L"button_ReplaceAll";
this->button_ReplaceAll->Size = System::Drawing::Size(93, 23);
this->button_ReplaceAll->TabIndex = 4;
this->button_ReplaceAll->Text = L"1 e) ReplaceAll";
this->button_ReplaceAll->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
this->button_ReplaceAll->UseVisualStyleBackColor = true;
this->button_ReplaceAll->Click +=
    gcnw System::EventHandler(this, &Form1::button_ReplaceAll_Click);
//
// button_Trim
//
this->button_Trim->Location = System::Drawing::Point(19, 86);
this->button_Trim->Name = L"button_Trim";
this->button_Trim->Size = System::Drawing::Size(79, 23);
this->button_Trim->TabIndex = 3;
this->button_Trim->Text = L"1 d) Trim";
this->button_Trim->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
this->button_Trim->UseVisualStyleBackColor = true;

```

```

this->button_Trim->Click +=
    gnew System::EventHandler(this, &Form1::button_Trim_Click);
//
// button_Split3
//
this->button_Split3->Location = System::Drawing::Point(125, 57);
this->button_Split3->Name = L"button_Split3";
this->button_Split3->Size = System::Drawing::Size(63, 23);
this->button_Split3->TabIndex = 2;
this->button_Split3->Text = L"1 c) Split3";
this->button_Split3->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
this->button_Split3->UseVisualStyleBackColor = true;
this->button_Split3->Click +=
    gnew System::EventHandler(this, &Form1::button_Split3_Click);
//
// button_Contains
//
this->button_Contains->Location = System::Drawing::Point(19, 57);
this->button_Contains->Name = L"button_Contains";
this->button_Contains->Size = System::Drawing::Size(79, 23);
this->button_Contains->TabIndex = 1;
this->button_Contains->Text = L"1 b) Contains";
this->button_Contains->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
this->button_Contains->UseVisualStyleBackColor = true;
this->button_Contains->Click +=
    gnew System::EventHandler(this, &Form1::button_Contains_Click);
//
// button_AnzahlZeichen
//
this->button_AnzahlZeichen->Location = System::Drawing::Point(19, 28);
this->button_AnzahlZeichen->Name = L"button_AnzahlZeichen";
this->button_AnzahlZeichen->Size = System::Drawing::Size(114, 23);
this->button_AnzahlZeichen->TabIndex = 0;
this->button_AnzahlZeichen->Text = L"1 a) Anzahl Zeichen";
this->button_AnzahlZeichen->TextAlign
System::Drawing::ContentAlignment::MiddleLeft;
this->button_AnzahlZeichen->UseVisualStyleBackColor = true;
this->button_AnzahlZeichen->Click +=
    gnew System::EventHandler(this, &Form1::button_AnzahlZeichen_Click);
//
// groupBox2
//
this->groupBox2->Controls->Add(this->button_stringToDate);
this->groupBox2->Location = System::Drawing::Point(177, 279);
this->groupBox2->Name = L"groupBox2";
this->groupBox2->Size = System::Drawing::Size(228, 56);
this->groupBox2->TabIndex = 1;
this->groupBox2->TabStop = false;
this->groupBox2->Text = L"Aufgaben 4.5";
//
// button_stringToDate
//
this->button_stringToDate->Location = System::Drawing::Point(14, 21);
this->button_stringToDate->Name = L"button_stringToDate";
this->button_stringToDate->Size = System::Drawing::Size(113, 23);
this->button_stringToDate->TabIndex = 0;
this->button_stringToDate->Text = L"1. stringToDate";
this->button_stringToDate->UseVisualStyleBackColor = true;
this->button_stringToDate->Click +=
    gnew System::EventHandler(this, &Form1::button_stringToDate_Click);
//
// textBox1
//
this->textBox1->Location = System::Drawing::Point(26, 19);
this->textBox1->Name = L"textBox1";
this->textBox1->Size = System::Drawing::Size(100, 20);
this->textBox1->TabIndex = 2;
//
// groupBox3
//
this->groupBox3->Controls->Add(this->textBox3);

```

```

this->groupBox3->Controls->Add(this->textBox2);
this->groupBox3->Controls->Add(this->textBox1);
this->groupBox3->Location = System::Drawing::Point(20, 21);
this->groupBox3->Name = L"groupBox3";
this->groupBox3->Size = System::Drawing::Size(145, 100);
this->groupBox3->TabIndex = 3;
this->groupBox3->TabStop = false;
this->groupBox3->Text = L"Eingabe:";
//
// textBox3
//
this->textBox3->Location = System::Drawing::Point(26, 74);
this->textBox3->Name = L"textBox3";
this->textBox3->Size = System::Drawing::Size(100, 20);
this->textBox3->TabIndex = 4;
//
// textBox2
//
this->textBox2->Location = System::Drawing::Point(26, 45);
this->textBox2->Name = L"textBox2";
this->textBox2->Size = System::Drawing::Size(100, 20);
this->textBox2->TabIndex = 3;
//
// textBox_out
//
this->textBox_out->Location = System::Drawing::Point(6, 19);
this->textBox_out->Multiline = true;
this->textBox_out->Name = L"textBox_out";
this->textBox_out->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;
this->textBox_out->Size = System::Drawing::Size(141, 165);
this->textBox_out->TabIndex = 4;
//
// groupBox4
//
this->groupBox4->Controls->Add(this->textBox_out);
this->groupBox4->Location = System::Drawing::Point(20, 145);
this->groupBox4->Name = L"groupBox4";
this->groupBox4->Size = System::Drawing::Size(153, 190);
this->groupBox4->TabIndex = 5;
this->groupBox4->TabStop = false;
this->groupBox4->Text = L"Ausgabe";
//
// Form1
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(417, 339);
this->Controls->Add(this->groupBox4);
this->Controls->Add(this->groupBox3);
this->Controls->Add(this->groupBox2);
this->Controls->Add(this->groupBox1);
this->Name = L"Form1";
this->Text = L"Form1";
this->groupBox1->ResumeLayout(false);
this->groupBox2->ResumeLayout(false);
this->groupBox3->ResumeLayout(false);
this->groupBox3->PerformLayout();
this->groupBox4->ResumeLayout(false);
this->groupBox4->PerformLayout();
this->ResumeLayout(false);
}
private: System::Void
button_AnzahlZeichen_Click(System::Object^ sender, System::EventArgs^ e)
{
    int n = N_Loesungen_Strings::
        AnzahlZeichen(rkl::to_string(textBox1->Text), ' ');
    textBox_out->AppendText("Anzahl Leerzeichen=" +
        n.ToString() + System::Environment::NewLine);
}

```

```

private: System::Void
    button_Contains_Click(System::Object^ sender, System::EventArgs^ e)
    {
        bool result =
            N_Loesungen_Strings::Contains(rk1::to_string(textBox1->Text),
                rk1::to_string(textBox2->Text));
        if (result)
            textBox_out->AppendText("'" + textBox1->Text + "' enthält '" +
                textBox2->Text + System::Environment::NewLine);
        else
            textBox_out->AppendText("'" + textBox1->Text + "' enthält nicht '" +
                textBox2->Text + System::Environment::NewLine);
    }

private: System::Void
    button_Split3_Click(System::Object^ sender, System::EventArgs^ e)
    {
        std::string d, m, j;
        bool result = N_Loesungen_Strings::Split3(rk1::to_string(textBox1->Text),
            d, m, j); // b = true, d="11", m="12", j="2017"
        if (result)
            textBox_out->AppendText("1:'" + rk1::to_string(d) +
                "' 2:'" + rk1::to_string(m) + "' " +
                "' 3:'" + rk1::to_string(j) + "'\r\n");
        else
            textBox_out->AppendText("keine 3 Teilstrings gefunden \r\n");
    }

private: System::Void
    button_Trim_Click(System::Object^ sender, System::EventArgs^ e)
    {
        std::string left_trimmed =
            N_Loesungen_Strings::TrimLeft(rk1::to_string(textBox1->Text));
        std::string right_trimmed =
            N_Loesungen_Strings::TrimRight(rk1::to_string(textBox1->Text));
        textBox_out->AppendText("1:'" + rk1::to_string(left_trimmed) + "'\r\n");
        textBox_out->AppendText("2:'" + rk1::to_string(right_trimmed) + "'\r\n");
    }

private: System::Void
    button_ReplaceAll_Click(System::Object^ sender, System::EventArgs^ e)
    {
        std::string str = rk1::to_string(textBox1->Text);
        std::string from = rk1::to_string(textBox2->Text);
        std::string to = rk1::to_string(textBox3->Text);
        int n = N_Loesungen_Strings::ReplaceAll(str, from, to);
        textBox_out->AppendText("'" + rk1::to_string(str) + "' n=" +
            n.ToString() + System::Environment::NewLine);
    }

private: System::Void
    button_tokenize_0_Click(System::Object^ sender, System::EventArgs^ e)
    {
        N_Loesungen_Strings::tokenize_0(textBox_out,
            rk1::to_string(textBox1->Text), ".");
    }

private: System::Void
    button_generatePassword_Click(System::Object^ sender, System::EventArgs^ e)
    {
        int n = 5;
        if (int::TryParse(textBox1->Text, n))
            ;
        else
            n = 5;
        std::string p = N_Loesungen_Strings::generatePassword(n);
        textBox_out->AppendText(rk1::to_string(p) + System::Environment::NewLine);
    }

private: System::Void
    button_uintToBinaryString_Click(System::Object^ sender, System::EventArgs^ e)

```

```
{
    for (unsigned int u = 1; u < 10; u++)
    {
        std::string s = N_Loesungen_Strings::uintToBinaryString(u);
        textBox_out->AppendText(rk1::to_String(s) + System::Environment::NewLine);
    }
}

private: System::Void
    button_BigIntAdd_Click(System::Object^ sender, System::EventArgs^ e)
{
    std::string op1 = rk1::to_string(textBox1->Text);
    std::string op2 = rk1::to_string(textBox2->Text);
    std::string result = rk1::to_string(textBox3->Text);
    std::string s = N_Loesungen_Strings::BigIntAdd(op1, op2);
    textBox_out->AppendText(rk1::to_String(s) + System::Environment::NewLine);
}

private: System::Void
    button_stringToDate_Click(System::Object^ sender, System::EventArgs^ e)
{
    std::string d = rk1::to_string(textBox1->Text);
    N_Loesungen_Strings::showStringToDate(textBox1, d);
}

};
}
```





## 4 Lösungen Kapitel 5. Arrays und Container

```
##### Copyright notice #####  
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook  
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",  
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_5_ArraysContainer.h
```

```
#pragma once  
#include <cmath>  
#include <string>  
#include <array>  
#include <vector>  
#include <thread>  
  
void f() {};  
void test_thread()  
{  
    std::thread t(f);  
}
```

## 4.1 Aufgabe 5.2.3 Eindimensionale Arrays

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_5_ArraysContainer.h
namespace N_Loesungen_Arrays
{
    using namespace System;
    using namespace System::Windows::Forms;

    // ----- Aufgabe 1. -----

    void ArrayAufgabe_1()
    {
        {
            int a[10];
            for (int i = 1; i <= 10; i++) a[i] = 0;    // a)
            int b[2], c[2];
            b[0] = 0; b[1] = 1;
            // c = b;                                // b)
            int x = b[b[0]];                          // c)
            c[0] = 0; c[1] = 1;
            if (b == c) x++;                          // d)
            // a = b;                                // e)

            /*
            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 Werte der Arrays
            e) wie b)
            */
        }
        {
            std::array<int,10> a;
            for (int i = 1; i <= 10; i++) a[i] = 0;    // a)
            std::array<int, 10> b, c;
            b[0] = 0; b[1] = 1;
            c = b;                                    // b)
            int x = b[b[0]];                          // c)
            c[0] = 0; c[1] = 1;
            if (b == c) x++;                          // d)
            // a = b;                                // e)

            /*
            a) Fehler: Zugriff auf a[10]
            b) Zuweisung von std::array ist möglich
            c) x=0
            d) b==c vergleicht die Inhalte und nicht die Adressen der Arrays
            e) // Fehler: array<int,10> und array<int,2> sind verschiedene Typen
            */
        }
    }
}

// ----- Aufgabe 2 -----

void SiebEratosthenes(TextBox^ tb)
{
    const int n = 1000; // Eine Konstante verwenden, damit die
    // Anzahl der Elemente des Arrays leicht geändert werden kann.
    // Mit einem vector wie in c) kann n auch eine Variable sein,
    // z.B. ein Parameter von SiebEratosthenes
    //// Für die Lösung von a), b) oder c) einfach nur die
    //// jeweilige Variante aktivieren durch Entfernen von //
    // std::array<bool, n + 1> prim; // a)
    //// oder
    // bool prim[n + 1]; // b)
    //// oder
    std::vector<bool> prim(n + 1, true); // c)
    // p[i]==true, falls i eine Primzahl ist, sonst false
}
```

```

for (int i = 0; i <= n; i++)
    prim[i] = true; // d) prim.at(i) = true;
for (int i = 2; i <= n; i++)
    if (prim[i]) // d) if (prim.at(i))
        for (int j = i; j <= n / i; j++)
            prim[j*i] = false; // d) prim.at(j * i) = false;
for (int i = 2; i <= n; i++)
    if (prim[i]) // d) if (prim.at(i))
    {
        tb->AppendText((i).ToString() + " ");
    }
};

// ----- Aufgabe 3 -----

std::vector<std::string> tokenize(const std::string& s, std::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<std::string> result;
  std::string::size_type pos_tok = s.find_first_not_of(separators);
  while (pos_tok != std::string::npos)
  { // token gefunden ab Position pos_tok
    std::string::size_type pos_sep = s.find_first_of(separators, pos_tok);
    if (pos_sep == std::string::npos)
        pos_sep = s.length();
    std::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;
}

// c)

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void assertEqual(std::vector<std::string> exp, std::vector<std::string> act,
  std::string msg)
{
    rkl::Assert::AreEqual(exp == act, true, msg);
}

std::vector<std::string> make_string_vector(std::string s1 = "",
  std::string s2 = "", std::string s3 = "")
{
    std::vector<std::string> result;
    if (s1 != "")
        result.push_back(s1);
    if (s2 != "")
        result.push_back(s2);
    if (s3 != "")
        result.push_back(s3);
    return result;
}

void Unittests_tokenize()
{
    const int nTests = 10;
    std::array<std::string, nTests> s; // wie std::string s[nTests];
    std::vector<std::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 4.3
// 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++)
    assertEquals(tokenize(s[i]), st[i], "test_tokenize i=" + std::to_string(i));
}
#endif // #ifndef SIMPLEUNITTESTS_H__

// ----- Aufgabe 4 -----

double MisesSimulation(TextBox^ tb, int AnzahlPersonen)
{
    std::array<int, 365> g; // wie 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]++;
        bool gefunden = false;
        for (int j = 0; j < 365; j++)
            if (g[j] > 1) gefunden = true;
        if (gefunden) AnzahlTreffer++;
    };
    tb->AppendText("n=" + (AnzahlPersonen).ToString() + " w="
        + (1.0*AnzahlTreffer / AnzahlVersuche).ToString() + System::Environment::NewLine);
    return 1.0*AnzahlTreffer / AnzahlVersuche;
}

void TestMisesSimulation(TextBox^ tb)
{
    MisesSimulation(tb, 23);
    MisesSimulation(tb, 23);
    MisesSimulation(tb, 23);
    MisesSimulation(tb, 23);
    MisesSimulation(tb, 23);

    MisesSimulation(tb, 10);
    MisesSimulation(tb, 20);
    MisesSimulation(tb, 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 5 -----

// a)
double Horner(double x)
{ // p: Array mit den Koeffizienten des Polynoms
  const int n = 2; // Der Grad des Polynoms
  std::array<double,n+1> p = { 17,0,3 }; // Koeffizienten von 17+3*x^2
  // oder
  // std::vector<double> p = { 17,0,3 }; // Koeffizienten von 17+3*x^2
  // oder
  // 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;
}

// Ein vector kann im Gegensatz zu einem Array mit einer beliebigen
// Anzahl von Koeffizienten als Parameter übergeben werden

double Horner(std::vector<double> p, double x)
{ // p: Koeffizienten des Polynoms
  double s = 0;
  for (int i = 0; i < p.size() + 1; i++)
    s = s * x + p[p.size() - i];
  return s;
}

/* Symbolische Ablaufprotokolle
double Horner(double x) mit n=0
{ // p: Array mit den Koeffizienten des Polynoms
  const int n = 0; // Der Grad des Polynoms
  double p[0 + 1]; // Koeffizienten: p[0]
  double s = 0;
  for (int i = 0; i < 0 + 1; i++)
    s = s*x + p[0 - i];
  Ablaufprotokoll:
      i      s
  s = s*x + p[n - i];    0    0*x + p[0-0]
  return s; // p[0]
}

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

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

```

```

        = p[2]*x^2 + p[2-1]*x + p[0]
return s; // p[2]*x^2 + p[1]*x + p[0]
}
*/

/*
b)

```

Induktionsanfang: Nach der ersten Zuweisung  $s=s*x+\dots$   
in der Schleife gilt ( $i==1$ ) und ( $s==p[n]$ ).

Induktionsvoraussetzung:  
Nach der  $k$ -ten Zuweisung  $s=s*x+\dots$   
in der Schleife (vor  $i++$ ) gilt ( $i==k$ ) und

$$s==p[n]*x^k + p[n-1]*x^{(k-1)} + \dots + p[n-k+1]*x+p[n-k]$$

Beim nächsten Schleifendurchlauf erhält  $i$  den Wert ( $i==k+1$ )  
und  $s$  den Wert:

$$s==p[n]*x^{(k+1)} + p[n-1]*x^k + \dots + p[n-k+1]*x^2+p[n-k]*x+p[n-(k+1)]$$

Das ist gerade die Induktionsbehauptung für  $k+1$ .

Beim letzten Durchlauf hat  $i$  den Wert  $n$ :

$$s==p[n]*x^n + p[n-1]*x^{(n-1)} + \dots + p[n-n+1]*x+p[n-n]$$

```

*/

// c)
double Horner_sin(double x)
{ // p: Array mit den Koeffizienten der Taylorreihe für Sinus
  double k3 = 3 * 2 * 1;
  double k5 = 5 * 4 * k3;
  double k7 = 7 * 6 * k5;
  const int n = 7; // Der Grad des Polynoms
  std::array<double,n + 1> p = { 0, 1, 0, -1 / k3, 0, +1 / k5, 0, -1 / k7 };
  // oder mit einem C-Array
  // 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;

  // oder mit einem vector-Parameter
  {
    std::vector<double> SinusPolynom =
      { 0, 1, 0, -1 / k3, 0, +1 / k5, 0, -1 / k7 };
    s = Horner(SinusPolynom, x);
    return s;
  }
}

void test_HornerSinus(TextBox^ tb)
{
  for (int i = 1; i <= 31; i++)
  {
    double x = i / 10.0;
    double s1 = std::sin(x); // #include <cmath>
    double s2 = Horner_sin(x);
    tb->AppendText("x=" + x.ToString() + " sin=" + s1.ToString() +
      " Horn-sin=" + (s2).ToString() + " diff=" + (std::abs(s1 - s2)).ToString() +
      System::Environment::NewLine);

    // Offensichtlich liefert die Taylor-Reihe für kleine Werte
    // eine gute Näherung
  }
}

```

## 4.2 Aufgabe 5.2.4 Die Initialisierung von Arrays bei ihrer Definition

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_5_ArraysContainer.h

// ----- Aufgabe 1. -----

int ai0[3]; // global, alle Elemente 0

void InitArrays()
{
    int ail[3]; // lokal, alle Elemente haben undefinierte Werte
    int ai2[3] = { 1, 2, 3 }; // {1,2,3}
    int ai3[3] = { 1, 2 }; // {1,2,0}
    int ai4[3] = {}; // alle Elemente 0
}

// ----- Aufgabe 2. -----

int Fibonacci_aus_Array(int n)
{
    const int max = 47;
    std::array<int, max> f = { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34 }; // 10 Werte
    if ((0 <= n) && (n < 10)) return f[n];
    else if (n < max)
    {
        for (int i = 10; i <= n; i++)
            f[i] = f[i - 1] + f[i - 2];
        return f[n];
    }
    else return -1; // oder besser eine Exception auslösen
    // oder mit einem C-Array
    {
        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 < max)
        {
            for (int i = 10; i <= n; i++)
                f[i] = f[i - 1] + f[i - 2];
            return f[n];
        }
        else return -1; // oder besser eine Exception auslösen
    }
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_Fibonacci_aus_Array()
{
    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++)
        rkl::Assert::AreEqual(Fibonacci_aus_Array(i), f[i], "FibonacciArray(" +
std::to_string(i) + ")");
}
#endif // #ifdef SIMPLEUNITTESTS_H__
```

### 4.3 Aufgabe 5.3 Arrays als Container

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_5_ArraysContainer.h  
// Die Lösung der Aufgabe 5.3 ist das Projekt Loesungen_Kap_5_3_mit_GUI
```



## 4.4 Aufgabe 5.4 Mehrdimensionale Arrays

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_5_ArraysContainer.h
// ----- Aufgabe 1. -----

void FindMinMaxInMultidimArray(TextBox^ tb)
{
    // a)
    const int m = 10, n = 20;
    std::array<std::array<int, 10>, 20> a2;
    int a2_c[m][n]; // mit einem C-Array
    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;
            }
    tb->AppendText((min_i).ToString() + ":" + (min_j).ToString() + "\r\n");
    // b)
    const int p = 10;
    std::array<std::array<int, m>, n>, p> a3;
    int a3_c[m][n][p]; // C-Array
    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;
                }
    tb->AppendText((min_i).ToString() + ":" + (min_j).ToString() + ":" +
(min_k).ToString() + "\r\n");
}
```

## 4.5 Aufgabe 5.5 Dynamische Programmierung

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_5_ArraysContainer.h

// using intTyp = int;
// const int max_n_binom = 30; // Mit int ab 33 negative Werte - overflow

using intTyp = long long;
const int max_n_binom = 70; // Mit long long ab 65 negative Werte - overflow

long long binom(int n, int k)
{
    std::array< std::array<intTyp, max_n_binom >, max_n_binom > a = {};
    // oder als mehrdimensionales C-Array:
    // intTyp a[max_n_binom][max_n_binom] = { 0 }; // initialisiert alle Arrayelemente
mit 0
    for (int i = 0; i < max_n_binom; 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_n_binom)
        return a[n][k];
    else
        return 0;
}

void zeigePascalDreieck(TextBox^ tb)
{
    tb->AppendText("Pascal Dreieck\r\n");
    for (int i = 1; i < max_n_binom; i++)
        {
            String^ s;
            for (int j = 0; j < i; j++)
                s += (binom(i, j)).ToString() + " ";
            tb->AppendText(s + System::Environment::NewLine);
        }
}

#ifdef SIMPLEUNITTESTS_H__
void alle_Unittests()
{
    rk1::Assert::Init("Unittests_Kap_5_Arrays");
    // rk1::Assert::LogTestCases::No );
    bool ask = false;
    Unittests_tokenize();
    Unittests_Fibonacci_aus_Array();
    rk1::Assert::Summary();
}
#endif // #ifdef SIMPLEUNITTESTS_H__

} // end of namespace N_Loesungen_Arrays

#pragma once
```

## 4.6 Form1.h für Kapitel 5

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Form1-Kap-5.h
```

```
#include "d:\CPPUtils\CppCLR_Utils.h"
```

```
#include "D:\CPPUtils\SimpleUnitTests.h"
```

```
#include "D:\cpp-2015.boon\Loesungen\Loesung_Kap_5_ArraysContainer.h"
```

```
namespace CppCLR_WinformsProjekt {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Zusammenfassung für Form1
    /// </summary>
    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
            this->Text = "Aufgaben Kap. 5";
            //
            //TODO: Konstruktorcode hier hinzufügen.
            //
        }

    protected:
        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::TextBox^ textBox1;
    protected:

    private: System::Windows::Forms::GroupBox^ groupBox1;
    private: System::Windows::Forms::Button^ button_Horner_Sinus;
    private: System::Windows::Forms::Button^ button2;
```

```

private: System::Windows::Forms::Button^ button_Eratosthenes;
private: System::Windows::Forms::GroupBox^ groupBox2;
private: System::Windows::Forms::Button^ button_Fibonacci;

private: System::Windows::Forms::GroupBox^ groupBox3;
private: System::Windows::Forms::Button^ button_Pascal_Dreieck;
private: System::Windows::Forms::Button^ button_Unittests;

private:
    /// <summary>
    /// Erforderliche Designervariable.
    /// </summary>
    System::ComponentModel::Container ^components;

    /// <summary>
    /// Erforderliche Methode für die Designerunterstützung.
    /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
    /// </summary>
    void InitializeComponent(void)
    {
        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
        this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
        this->button_Horner_Sinus = (gcnew System::Windows::Forms::Button());
        this->button2 = (gcnew System::Windows::Forms::Button());
        this->button_Eratosthenes = (gcnew System::Windows::Forms::Button());
        this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
        this->button_Fibonacci = (gcnew System::Windows::Forms::Button());
        this->groupBox3 = (gcnew System::Windows::Forms::GroupBox());
        this->button_Pascal_Dreieck = (gcnew System::Windows::Forms::Button());
        this->button_Unittests = (gcnew System::Windows::Forms::Button());
        this->groupBox1->SuspendLayout();
        this->groupBox2->SuspendLayout();
        this->groupBox3->SuspendLayout();
        this->SuspendLayout();
        //
        // textBox1
        //
        this->textBox1->Anchor =
static_cast<System::Windows::Forms::AnchorStyles>(((System::Windows::Forms::AnchorStyles:
:Top | System::Windows::Forms::AnchorStyles::Bottom)
    | System::Windows::Forms::AnchorStyles::Left)
    | System::Windows::Forms::AnchorStyles::Right));
        this->textBox1->Location = System::Drawing::Point(12, 12);
        this->textBox1->Multiline = true;
        this->textBox1->Name = L"textBox1";
        this->textBox1->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;
        this->textBox1->Size = System::Drawing::Size(121, 237);
        this->textBox1->TabIndex = 0;
        //
        // groupBox1
        //
        this->groupBox1->Anchor =
static_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::T
op | System::Windows::Forms::AnchorStyles::Right));
        this->groupBox1->Controls->Add(this->button_Horner_Sinus);
        this->groupBox1->Controls->Add(this->button2);
        this->groupBox1->Controls->Add(this->button_Eratosthenes);
        this->groupBox1->Location = System::Drawing::Point(140, 12);
        this->groupBox1->Name = L"groupBox1";
        this->groupBox1->Size = System::Drawing::Size(132, 106);
        this->groupBox1->TabIndex = 2;
        this->groupBox1->TabStop = false;
        this->groupBox1->Text = L"5.2";
        //
        // button_Horner_Sinus
        //
        this->button_Horner_Sinus->Location = System::Drawing::Point(18, 77);
        this->button_Horner_Sinus->Name = L"button_Horner_Sinus";
        this->button_Horner_Sinus->Size = System::Drawing::Size(108, 23);
        this->button_Horner_Sinus->TabIndex = 2;
        this->button_Horner_Sinus->Text = L"4. c) Horner Sinus";

```

```

        this->button_Horner_Sinus->TextAlign                                     =
System::Drawing::ContentAlignment::MiddleLeft;
        this->button_Horner_Sinus->UseVisualStyleBackColor = true;
        this->button_Horner_Sinus->Click      +=      gcnew      System::EventHandler(this,
&Form1::button_Horner_Sinus_Click);
        //
        // button2
        //
        this->button2->Location = System::Drawing::Point(18, 48);
        this->button2->Name = L"button2";
        this->button2->Size = System::Drawing::Size(108, 23);
        this->button2->TabIndex = 1;
        this->button2->Text = L"3. Mises Simulation";
        this->button2->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
        this->button2->UseVisualStyleBackColor = true;
        this->button2->Click += gcnew System::EventHandler(this, &Form1::button2_Click);
        //
        // button_Eratosthenes
        //
        this->button_Eratosthenes->Location = System::Drawing::Point(18, 19);
        this->button_Eratosthenes->Name = L"button_Eratosthenes";
        this->button_Eratosthenes->Size = System::Drawing::Size(108, 23);
        this->button_Eratosthenes->TabIndex = 0;
        this->button_Eratosthenes->Text = L"2. Eratosthenes";
        this->button_Eratosthenes->TextAlign                                     =
System::Drawing::ContentAlignment::MiddleLeft;
        this->button_Eratosthenes->UseVisualStyleBackColor = true;
        this->button_Eratosthenes->Click      +=      gcnew      System::EventHandler(this,
&Form1::button_Eratosthenes_Click);
        //
        // groupBox2
        //
        this->groupBox2->Anchor                                               =
static_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::T
op | System::Windows::Forms::AnchorStyles::Right));
        this->groupBox2->Controls->Add(this->button_Fibonacci);
        this->groupBox2->Location = System::Drawing::Point(140, 124);
        this->groupBox2->Name = L"groupBox2";
        this->groupBox2->Size = System::Drawing::Size(133, 50);
        this->groupBox2->TabIndex = 3;
        this->groupBox2->TabStop = false;
        this->groupBox2->Text = L"5.3";
        //
        // button_Fibonacci
        //
        this->button_Fibonacci->Location = System::Drawing::Point(19, 19);
        this->button_Fibonacci->Name = L"button_Fibonacci";
        this->button_Fibonacci->Size = System::Drawing::Size(108, 23);
        this->button_Fibonacci->TabIndex = 0;
        this->button_Fibonacci->Text = L"2. Fibonacci";
        this->button_Fibonacci->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
        this->button_Fibonacci->UseVisualStyleBackColor = true;
        this->button_Fibonacci->Click      +=      gcnew      System::EventHandler(this,
&Form1::button_Fibonacci_Click);
        //
        // groupBox3
        //
        this->groupBox3->Anchor                                               =
static_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::T
op | System::Windows::Forms::AnchorStyles::Right));
        this->groupBox3->Controls->Add(this->button_Pascal_Dreieck);
        this->groupBox3->Location = System::Drawing::Point(140, 180);
        this->groupBox3->Name = L"groupBox3";
        this->groupBox3->Size = System::Drawing::Size(131, 49);
        this->groupBox3->TabIndex = 4;
        this->groupBox3->TabStop = false;
        this->groupBox3->Text = L"5.6";
        //
        // button_Pascal_Dreieck
        //
        this->button_Pascal_Dreieck->Location = System::Drawing::Point(17, 19);

```

```

        this->button_Pascal_Dreieck->Name = L"button_Pascal_Dreieck";
        this->button_Pascal_Dreieck->Size = System::Drawing::Size(108, 23);
        this->button_Pascal_Dreieck->TabIndex = 0;
        this->button_Pascal_Dreieck->Text = L"Pascal-Dreieck";
        this->button_Pascal_Dreieck->TextAlign
System::Drawing::ContentAlignment::MiddleLeft;
        this->button_Pascal_Dreieck->UseVisualStyleBackColor = true;
        this->button_Pascal_Dreieck->Click += gcnw System::EventHandler(this,
&Form1::button_Pascal_Dreieck_Click);
        //
        // button_Unittests
        //
        this->button_Unittests->Location = System::Drawing::Point(140, 235);
        this->button_Unittests->Name = L"button_Unittests";
        this->button_Unittests->Size = System::Drawing::Size(127, 23);
        this->button_Unittests->TabIndex = 5;
        this->button_Unittests->Text = L"Unittests";
        this->button_Unittests->UseVisualStyleBackColor = true;
        this->button_Unittests->Click += gcnw System::EventHandler(this,
&Form1::button_Unittests_Click);
        //
        // Form1
        //
        this->AutoSizeDimensions = System::Drawing::SizeF(6, 13);
        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
        this->ClientSize = System::Drawing::Size(284, 261);
        this->Controls->Add(this->button_Unittests);
        this->Controls->Add(this->groupBox3);
        this->Controls->Add(this->groupBox2);
        this->Controls->Add(this->groupBox1);
        this->Controls->Add(this->textBox1);
        this->Name = L"Form1";
        this->Text = L"Form1";
        this->groupBox1->ResumeLayout(false);
        this->groupBox2->ResumeLayout(false);
        this->groupBox3->ResumeLayout(false);
        this->ResumeLayout(false);
        this->PerformLayout();

    }

private: System::Void
    button_Eratosthenes_Click(System::Object^ sender, System::EventArgs^ e)
    {
        N_Loesungen_Arrays::SiebEratosthenes(textBox1);
    }

private: System::Void
    button2_Click(System::Object^ sender, System::EventArgs^ e)
    {
        N_Loesungen_Arrays::TestMisesSimulation(textBox1);
    }

private: System::Void
    button_Horner_Sinus_Click(System::Object^ sender, System::EventArgs^ e)
    {
        N_Loesungen_Arrays::test_HornerSinus(textBox1);
    }

private: System::Void
    button_Fibonacci_Click(System::Object^ sender, System::EventArgs^ e)
    {
        for (int i = 1; i < 20; i++)
            textBox1->AppendText("Fibo(" + i.ToString() + ")=" +
                N_Loesungen_Arrays::Fibonacci_aus_Array(i).ToString() +
                System::Environment::NewLine);
    }

private: System::Void
    button_Pascal_Dreieck_Click(System::Object^ sender, System::EventArgs^ e)
    {
        N_Loesungen_Arrays::zeigePascalDreieck(textBox1);
    }

```

```
    }  
  
private: System::Void  
    button_Unittests_Click(System::Object^ sender, System::EventArgs^ e)  
    {  
        N_Loesungen_Arrays::Unittests_Kap_5_Arrays();  
    }  
};  
}
```

## 4.7 Aufgabe 5.3 Arrays als Container

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-
Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_5_Arrays_als_Container.h
```

```
#pragma once
```

```
#ifndef _DEBUG
#include "D:\CPPUtils\SimpleUnitTests.h"
#endif
```

```
#include <string>
#include <cstdlib>
#include <array>
```

```
namespace N_Arrays_als_Container
{
    using namespace System;
    using namespace System::Windows::Forms;
```

```
// ----- Aufgabe 1. -----
```

```
const int maxElements = 5; // Mit einem kleinen Wert von maxElements
                          // kann man die Prüfungen auf die Obergrenze
                          // einfach testen.
```

```
int nElements = 0; // Anzahl der Arrayelemente
                  // belegt im Array: A[0] .. A[nElements-1]
using T = int; // wie typedef int T;
std::array<T, maxElements> A;
// oder als C-Array:
// 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.
}

// b)
void showArray(TextBox^ tb)
{
    for (int i = 0; i < nElements; i++)
        tb->AppendText(A[i].ToString() + " ");
    tb->AppendText(System::Environment::NewLine);
}

// 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  $n < \text{maxElements}$  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;
}

```

```

// e)

bool eraseElement(int pos)
{
    bool result = true;
    if ((0 <= pos) && (pos < nElements))
    {
        A[pos] = A[nElements - 1];
        nElements--;
    }
    else result = false;
    return result;
}

```

```

// f)

void Auswahlort()
{
    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;
    }
}

```

```

}

// g)
int findBinary(T s)
{
    int M = -1;
    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;
}

// 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
            p++; // Alle Elemente ab Position p um eine Position nach hinten verschieben
        // (Short-Circuit-Evaluation)
        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;
}

// 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;
}

// k)

#ifdef SIMPLEUNITTESTS_H__
void Unittests_1()
{
    // teste pushBack, findLinear und eraseElement
    nElements = 0;
}

```

```

    rk1::Assert::AreEqual(findLinear(1, 0), 0, "findLinear(1,0) - a");
    pushBack(1);
    rk1::Assert::AreEqual(findLinear(1, 0), 0, "findLinear(1,0) - b");
    pushBack(2);
    rk1::Assert::AreEqual(findLinear(2, 0), 1, "findLinear(2,0)");
    pushBack(3);
    rk1::Assert::AreEqual(findLinear(3, 0), 2, "findLinear(3,0) - a");
    // Ist das gefundene Element auch wirklich das erste?
    pushBack(3);
    rk1::Assert::AreEqual(findLinear(3, 0), 2, "findLinear(3,0) - b");
    // teste eraseElement
    eraseElement(0);
    rk1::Assert::AreEqual(findLinear(0, 0), nElements, "nach eraseElement(0,0)");
}

void Unittests_SetAndSort2Elements(T e1, T e2)
{
    nElements = 0;
    pushBack(e1);
    pushBack(e2);
    Auswahlort();
}

void Unittests_SetAndSort3Elements(T e1, T e2, T e3)
{
    nElements = 0;
    pushBack(e1);
    pushBack(e2);
    pushBack(e3);
    Auswahlort();
}

void Unittests_Auswahlort_0123()
{ // teste Auswahlort mit 0, 1, 2 und 3 Elementen
    nElements = 0;
    Auswahlort(); // Auswahlort sollte bei einem leeren Array nicht abstürzen
    pushBack(1);
    Auswahlort(); // Array mit einem Element
    rk1::Assert::AreEqual(findLinear(1, 0), 0, "findLinear(1,0)");

    Unittests_SetAndSort2Elements(1, 2); // bereits sortiert
    rk1::Assert::AreEqual(findLinear(1, 0), 0, "sort 2, 1a");
    rk1::Assert::AreEqual(findLinear(2, 0), 1, "sort 2, 1b");
    Unittests_SetAndSort2Elements(2, 1); // umgekehrte Reihenfolge
    rk1::Assert::AreEqual(findLinear(1, 0), 0, "sort 2, 2a");
    rk1::Assert::AreEqual(findLinear(2, 0), 1, "sort 2, 2b");

    Unittests_SetAndSort3Elements(1, 2, 3); // testen alle Permutationen
    rk1::Assert::AreEqual(findLinear(1, 0), 0, "sort 3, 1a");
    rk1::Assert::AreEqual(findLinear(2, 0), 1, "sort 3, 1b");
    rk1::Assert::AreEqual(findLinear(3, 0), 2, "sort 3, 1c");
    Unittests_SetAndSort3Elements(1, 3, 2);
    rk1::Assert::AreEqual(findLinear(1, 0), 0, "sort 3, 1a");
    rk1::Assert::AreEqual(findLinear(2, 0), 1, "sort 3, 1b");
    rk1::Assert::AreEqual(findLinear(3, 0), 2, "sort 3, 1c");
    Unittests_SetAndSort3Elements(2, 1, 3);
    rk1::Assert::AreEqual(findLinear(1, 0), 0, "sort 3, 1a");
    rk1::Assert::AreEqual(findLinear(2, 0), 1, "sort 3, 1b");
    rk1::Assert::AreEqual(findLinear(3, 0), 2, "sort 3, 1c");
    Unittests_SetAndSort3Elements(2, 3, 1);
    rk1::Assert::AreEqual(findLinear(1, 0), 0, "sort 3, 1a");
    rk1::Assert::AreEqual(findLinear(2, 0), 1, "sort 3, 1b");
    rk1::Assert::AreEqual(findLinear(3, 0), 2, "sort 3, 1c");
    Unittests_SetAndSort3Elements(3, 1, 2);
    rk1::Assert::AreEqual(findLinear(1, 0), 0, "sort 3, 1a");
    rk1::Assert::AreEqual(findLinear(2, 0), 1, "sort 3, 1b");
    rk1::Assert::AreEqual(findLinear(3, 0), 2, "sort 3, 1c");
    Unittests_SetAndSort3Elements(3, 2, 1);
    rk1::Assert::AreEqual(findLinear(1, 0), 0, "sort 3, 1a");
    rk1::Assert::AreEqual(findLinear(2, 0), 1, "sort 3, 1b");
    rk1::Assert::AreEqual(findLinear(3, 0), 2, "sort 3, 1c");
}

```

```

}

bool isSorted()
{
    bool result = true;
    for (int i = 0; i < nElements - 1; i++)
        if (A[i] > A[i + 1]) result = false;
    return result;
}

void Unittests_Auswahlort_n(int n)
{ // teste Auswahlort mit n Zufallszahlen
    nElements = 0;
    for (int i = 0; i < n; i++)
        pushBack(std::rand());
    Auswahlort();
    rkl::Assert::AreEqual(isSorted(), true, "sort n");
}

void Unittests_2()
{ // teste findBinary, insertSorted und eraseSorted
    bool result = true;
    nElements = 0;
    rkl::Assert::AreEqual(findBinary(1), 0, "findBinary(1,0)");
    pushBack(1);
    rkl::Assert::AreEqual(findLinear(1, 0), 0, "findLinear(1,0)");
    pushBack(2);
    rkl::Assert::AreEqual(findLinear(2, 0), 1, "findLinear(2,0)");
    pushBack(3);
    rkl::Assert::AreEqual(findLinear(3, 0), 2, "findLinear(2,0)");
    // teste eraseElement
    eraseElement(0);
    rkl::Assert::AreEqual(findLinear(0, 0), nElements, "nach eraseElement(2,0)");
}

void Unittests_ArrayAlsContainer()
{
    Unittests_1();
    Unittests_Auswahlort_0123();
    for (int i = 0; i < 100; i++)
        Unittests_Auswahlort_n(i);
    Unittests_2();
}

void alle_Unittests(bool ask)
{
    // Eventuell vorher fragen, ob die Daten gelöscht werden sollen
    if (!ask || MessageBox::Show("Die Daten im Array werden gelöscht - weiter?",
        "Unittests_Array 5.3",
        MessageBoxButtons::YesNo) == System::Windows::Forms::DialogResult::Yes)
    {
        rkl::Assert::Init("Unittests_ArrayAlsContainer"); // Tests initialisieren
        Unittests_ArrayAlsContainer();
        rkl::Assert::Summary(); // Testzusammenfassung ausgeben
    }
}

#endif // #ifdef SIMPLEUNITTESTS_H__
}
#pragma once

```

## 4.8 Form1.h für Kapitel 5.4

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Form1-Kap-5-4.h
#include "d:\CPPUtils\CppCLR_Utils.h"
#include "D:\CPPUtils\SimpleUnitTests.h"
#include "D:\cpp-2015.boon\Loesungen\Loesung_Kap_5_Arrays_als_Container.h"
```

```
namespace CppCLR_WinformsProjekt {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Zusammenfassung für Form1
    /// </summary>
    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO: Konstruktorcode hier hinzufügen.
            //
        }

    protected:
        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::TextBox^  Ausgabe_textBox;
    protected:
    private: System::Windows::Forms::TextBox^  Eingabe_textBox;
    private: System::Windows::Forms::Label^  labell;
    private: System::Windows::Forms::Button^  button_Einfügen;
    private: System::Windows::Forms::Button^  button_Anzeigen;
    private: System::Windows::Forms::Button^  button_Linear_suchen;
    private: System::Windows::Forms::Button^  button_Löschen;
```

```

private: System::Windows::Forms::Button^  button_sortieren;
private: System::Windows::Forms::Button^  button_Binär_suchen;
private: System::Windows::Forms::Button^  button_Sortiert_einfügen;
private: System::Windows::Forms::Button^  button_sortiert_löschen;

private:
    /// <summary>
    /// Erforderliche Designervariable.
    /// </summary>
    System::ComponentModel::Container ^components;

    /// <summary>
    /// Erforderliche Methode für die Designerunterstützung.
    /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
    /// </summary>
    void InitializeComponent(void)
    {
        this->Ausgabe_textBox = (gcnew System::Windows::Forms::TextBox());
        this->Eingabe_textBox = (gcnew System::Windows::Forms::TextBox());
        this->labell = (gcnew System::Windows::Forms::Label());
        this->button_Einfügen = (gcnew System::Windows::Forms::Button());
        this->button_Anzeigen = (gcnew System::Windows::Forms::Button());
        this->button_Linear_suchen = (gcnew System::Windows::Forms::Button());
        this->button_Löschen = (gcnew System::Windows::Forms::Button());
        this->button_sortieren = (gcnew System::Windows::Forms::Button());
        this->button_Binär_suchen = (gcnew System::Windows::Forms::Button());
        this->button_Sortiert_einfügen = (gcnew System::Windows::Forms::Button());
        this->button_sortiert_löschen = (gcnew System::Windows::Forms::Button());
        this->SuspendLayout();
        //
        // Ausgabe_textBox
        //
        this->Ausgabe_textBox->Location = System::Drawing::Point(17, 19);
        this->Ausgabe_textBox->Multiline = true;
        this->Ausgabe_textBox->Name = L"Ausgabe_textBox";
        this->Ausgabe_textBox->Size = System::Drawing::Size(167, 269);
        this->Ausgabe_textBox->TabIndex = 0;
        //
        // Eingabe_textBox
        //
        this->Eingabe_textBox->Location = System::Drawing::Point(256, 16);
        this->Eingabe_textBox->Name = L"Eingabe_textBox";
        this->Eingabe_textBox->Size = System::Drawing::Size(86, 20);
        this->Eingabe_textBox->TabIndex = 1;
        //
        // labell
        //
        this->labell->AutoSize = true;
        this->labell->Location = System::Drawing::Point(190, 16);
        this->labell->Name = L"labell";
        this->labell->Size = System::Drawing::Size(49, 13);
        this->labell->TabIndex = 2;
        this->labell->Text = L"Eingabe:";
        //
        // button_Einfügen
        //
        this->button_Einfügen->Location = System::Drawing::Point(219, 62);
        this->button_Einfügen->Name = L"button_Einfügen";
        this->button_Einfügen->Size = System::Drawing::Size(108, 23);
        this->button_Einfügen->TabIndex = 3;
        this->button_Einfügen->Text = L"a) Einfügen";
        this->button_Einfügen->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
        this->button_Einfügen->UseVisualStyleBackColor = true;
        this->button_Einfügen->Click += gcnew System::EventHandler(this,
&Form1::button_Einfügen_Click);
        //
        // button_Anzeigen
        //
        this->button_Anzeigen->Location = System::Drawing::Point(219, 91);
        this->button_Anzeigen->Name = L"button_Anzeigen";
        this->button_Anzeigen->Size = System::Drawing::Size(108, 23);
    }

```

```

    this->button_Anzeigen->TabIndex = 4;
    this->button_Anzeigen->Text = L"b) Anzeigen";
    this->button_Anzeigen->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
    this->button_Anzeigen->UseVisualStyleBackColor = true;
    this->button_Anzeigen->Click += gcnew System::EventHandler(this,
&Form1::button_Anzeigen_Click);
    //
    // button_Linear_suchen
    //
    this->button_Linear_suchen->Location = System::Drawing::Point(219, 120);
    this->button_Linear_suchen->Name = L"button_Linear_suchen";
    this->button_Linear_suchen->Size = System::Drawing::Size(108, 23);
    this->button_Linear_suchen->TabIndex = 5;
    this->button_Linear_suchen->Text = L"d) Linear suchen";
    this->button_Linear_suchen->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
    this->button_Linear_suchen->UseVisualStyleBackColor = true;
    this->button_Linear_suchen->Click += gcnew System::EventHandler(this,
&Form1::button_Linear_suchen_Click);
    //
    // button_Löschen
    //
    this->button_Löschen->Location = System::Drawing::Point(219, 149);
    this->button_Löschen->Name = L"button_Löschen";
    this->button_Löschen->Size = System::Drawing::Size(108, 23);
    this->button_Löschen->TabIndex = 6;
    this->button_Löschen->Text = L"e) Löschen";
    this->button_Löschen->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
    this->button_Löschen->UseVisualStyleBackColor = true;
    this->button_Löschen->Click += gcnew System::EventHandler(this,
&Form1::button_Löschen_Click);
    //
    // button_sortieren
    //
    this->button_sortieren->Location = System::Drawing::Point(219, 178);
    this->button_sortieren->Name = L"button_sortieren";
    this->button_sortieren->Size = System::Drawing::Size(108, 23);
    this->button_sortieren->TabIndex = 7;
    this->button_sortieren->Text = L"f) Sortieren";
    this->button_sortieren->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
    this->button_sortieren->UseVisualStyleBackColor = true;
    this->button_sortieren->Click += gcnew System::EventHandler(this,
&Form1::button_sortieren_Click);
    //
    // button_Binär_suchen
    //
    this->button_Binär_suchen->Location = System::Drawing::Point(219, 207);
    this->button_Binär_suchen->Name = L"button_Binär_suchen";
    this->button_Binär_suchen->Size = System::Drawing::Size(108, 23);
    this->button_Binär_suchen->TabIndex = 8;
    this->button_Binär_suchen->Text = L"g) Binär suchen";
    this->button_Binär_suchen->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
    this->button_Binär_suchen->UseVisualStyleBackColor = true;
    this->button_Binär_suchen->Click += gcnew System::EventHandler(this,
&Form1::button_Binär_suchen_Click);
    //
    // button_Sortiert_einfügen
    //
    this->button_Sortiert_einfügen->Location = System::Drawing::Point(219, 236);
    this->button_Sortiert_einfügen->Name = L"button_Sortiert_einfügen";
    this->button_Sortiert_einfügen->Size = System::Drawing::Size(108, 23);
    this->button_Sortiert_einfügen->TabIndex = 9;
    this->button_Sortiert_einfügen->Text = L"h) Sortiert einfügen";
    this->button_Sortiert_einfügen->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
    this->button_Sortiert_einfügen->UseVisualStyleBackColor = true;
    this->button_Sortiert_einfügen->Click += gcnew System::EventHandler(this,
&Form1::button_Sortiert_einfügen_Click);
    //
    // button_sortiert_löschen

```

```

//
this->button_sortiert_löschen->Location = System::Drawing::Point(219, 265);
this->button_sortiert_löschen->Name = L"button_sortiert_löschen";
this->button_sortiert_löschen->Size = System::Drawing::Size(108, 23);
this->button_sortiert_löschen->TabIndex = 10;
this->button_sortiert_löschen->Text = L"j) Sortiert löschen";
this->button_sortiert_löschen->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
this->button_sortiert_löschen->UseVisualStyleBackColor = true;
this->button_sortiert_löschen->Click += gcnew System::EventHandler(this,
&Form1::button_sortiert_löschen_Click);
//
// Form1
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(354, 297);
this->Controls->Add(this->button_sortiert_löschen);
this->Controls->Add(this->button_Sortiert_einfügen);
this->Controls->Add(this->button_Binär_suchen);
this->Controls->Add(this->button_sortieren);
this->Controls->Add(this->button_Löschen);
this->Controls->Add(this->button_Linear_suchen);
this->Controls->Add(this->button_Anzeigen);
this->Controls->Add(this->button_Einfügen);
this->Controls->Add(this->label1);
this->Controls->Add(this->Eingabe_textBox);
this->Controls->Add(this->Ausgabe_textBox);
this->Name = L"Form1";
this->Text = L"Aufgabe 5.4: Arrays als Container";
this->ResumeLayout(false);
this->PerformLayout();

}

private: System::Void
button_Einfügen_Click(System::Object^ sender, System::EventArgs^ e)
{ // a)
int x;
if (int::TryParse(Eingabe_textBox->Text, x))
{
if (!N_Arrays_als_Container::pushBack(x))
MessageBox::Show("Alle Plätze belegt");
}
else MessageBox::Show("kein int-Wert: '" + Eingabe_textBox->Text + "'");
}

private: System::Void
button_Anzeigen_Click(System::Object^ sender, System::EventArgs^ e)
{ // b)
N_Arrays_als_Container::showArray(Ausgabe_textBox);
}

private: System::Void
button_Linear_suchen_Click(System::Object^ sender, System::EventArgs^ e)
{
int x;
if (int::TryParse(Eingabe_textBox->Text, x))
{
int i = 0;
int Anzahl_gefundene_Werte = 0;
while (i < N_Arrays_als_Container::nElements)
{
i = N_Arrays_als_Container::findLinear(x, i);
if (i < N_Arrays_als_Container::nElements)
{
Ausgabe_textBox->AppendText((
N_Arrays_als_Container::A[i]).ToString() + " ");
i++;
Anzahl_gefundene_Werte++;
}
}
}
}

```



```

    }
    if (Anzahl_gefundene_Werte == 0)
        Ausgabe_textBox->AppendText(x.ToString() + " nicht gefunden" +
            System::Environment::NewLine);
    else
        Ausgabe_textBox->AppendText(System::Environment::NewLine);
    }
else MessageBox::Show("kein int-Wert: '" + Eingabe_textBox->Text + "'");
}

private: System::Void
    button_Löschen_Click(System::Object^ sender, System::EventArgs^ e)
{ // d)
    int x;
    if (int::TryParse(Eingabe_textBox->Text, x))
    {
        int p = N_Arrays_als_Container::findLinear(x, 0);
        if (p != N_Arrays_als_Container::nElements)
        {
            if (N_Arrays_als_Container::eraseElement(p))
                MessageBox::Show(x.ToString() + " gelöscht an Pos." + (p).ToString());
            else
                MessageBox::Show(x.ToString() + " nicht gelöscht");
        }
        else
            MessageBox::Show(x.ToString() + " existiert nicht");
    }
    else MessageBox::Show("kein int-Wert: '" + Eingabe_textBox->Text + "'");
}

private: System::Void
    button_sortieren_Click(System::Object^ sender, System::EventArgs^ e)
{ // f)
    N_Arrays_als_Container::Auswahlort();
}

private: System::Void
    button_Binär_suchen_Click(System::Object^ sender, System::EventArgs^ e)
{ // g)
    int x;
    if (int::TryParse(Eingabe_textBox->Text, x))
    {
        int pos = N_Arrays_als_Container::findBinary(x);
        if (pos < N_Arrays_als_Container::nElements)
            Ausgabe_textBox->AppendText(N_Arrays_als_Container::A[pos].ToString() +
                " existiert an Position " + pos.ToString() + "\r\n");
        else
            Ausgabe_textBox->AppendText(x.ToString() + " nicht gefunden" +
                System::Environment::NewLine);
    }
    else MessageBox::Show("kein int-Wert: '" + Eingabe_textBox->Text + "'");
}

private: System::Void
    button_Sortiert_einfügen_Click(System::Object^ sender, System::EventArgs^ e)
{
    // bool result = false;
    int x;
    if (int::TryParse(Eingabe_textBox->Text, x))
    {
        if (N_Arrays_als_Container::insertSorted(x))
            ; // result = true;
        else MessageBox::Show("Alle Plätze belegt");
    }
    else MessageBox::Show("kein int-Wert: '" + Eingabe_textBox->Text + "'");
    // return result;
}

private: System::Void
    button_sortiert_löschen_Click(System::Object^ sender, System::EventArgs^ e)
{

```

```
int x;
if (int::TryParse(Eingabe_textBox->Text, x))
{
    if (!N_Arrays_als_Container::eraseSorted(x))
        MessageBox::Show(x.ToString() + " existiert nicht");
}
else MessageBox::Show("kein int-Wert: '" + Eingabe_textBox->Text + "'");
};
}
```

## 4.9 Aufgabe 6.3 Einfache Selbstdefinierte Datentypen

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder
Folgeschäden übernommen, die sich aus der Nutzung ergeben.
```

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
In no event shall liability be accepted for any direct, indirect, incidental or consequential
damages arising from the use of these source texts.
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_6_SelbstdefDatentypen.h
```

```
#include <string>
namespace N_Selbstdefinierte_Datentypen
{
    using namespace System;
    using namespace System::Windows::Forms;
```

## 4.10 Aufgabe 6.1 Mit struct definierte Klassen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_6_SelbstdefDatentypen.h
// ----- Aufgabe 1 -----

struct Girokonto {
    struct Adresse_ {
        std::string Anrede;
        struct Name_ {
            std::string Vorname;
            std::string Nachname;
        } Name;
        struct Anschrift_ {
            std::string PLZ; // wegen Ausland als String
            std::string Ort;
            std::string Strasse;
            std::string Hausnummer; // wegen 7b
        } Anschrift;
        std::string Ausland;
        struct Telefon_ {
            std::string Vorwahl; // wegen führender Nullen
            std::string Telnr;
        } Telefon;
    } Adresse;
    int Kontonr;
    double Kontostand,
        Kreditlimit;
};

void useGirokonto()
{
    Girokonto g;
    g.Adresse.Anrede = "Ihre Majestät";
    g.Adresse.Name.Vorname = "Emelie";
    g.Adresse.Name.Nachname = "König";
    g.Kontonr = 1000;
    g.Adresse.Telefon.Vorwahl = "007";
    g.Adresse.Telefon.Vorwahl = "1234";
}

} // end of namespace N_Selbstdefinierte_Datentypen
```

## 5 Lösungen Kapitel 7. Zeiger

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_7_Zeiger.h
```

```
// using namespace std; // Okt. 2019, nur für test-track
```

```
void ptr_swap(int* x, int* y)
```

```
{
    int h = *x;
    *x = *y;
    *y = h;
}
```

```
#include <string>
```

```
#include "d:\CPPUtils\CppCLR_Utils.h"
```

```
namespace N_Loesungen_Zeiger
```

```
{
    using namespace System::Windows::Forms;
```

## 5.1 Aufgabe 7.2 Adressoperator und Zeiger

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_7_Zeiger.h
// ----- Aufgabe 1 -----

void Aufgabe_7_2_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
    // i)
    *pi = i**pc; // *pi==5*5
    // j)
    pi = 0; // pi kann nicht dereferenziert werden
}

// ----- Aufgabe 2 -----

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

void Aufgabe_7_2_2(TextBox^ tb)
{
    // a)
    int j = 10;
    int* p = &j;
    *p = 17;
    tb->AppendText(j.ToString() + "\r\n"); // 17 wegen p==&j
    tb->AppendText((*p).ToString() + "\r\n"); // 17

    // b)
    *new(int) = 17;
    // Erzeugt eine Variable dynamisch, ohne ihre Adresse zu speichern.
    // Deswegen kann dieser Speicher nicht mehr freigegeben werden
    // (Speicherleck).

    // c)
    tb->AppendText((*f_retp()).ToString() + "\r\n");

    // 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.
}

```

## 5.2 Aufgabe 7.4 Ablaufprotokolle für Zeigervariable

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_7_Zeiger.h
// ----- Aufgabe 1 -----

void ptr_swap(int* x, int* y)
{
    //          x      *x   y   *y   h
    //          x0     *x0  y0   *y0
    int h = *x; //
    *x = *y;    //      *y0
    *y = h;     //          *x0
} // *x=*y0 and *y=*x0

void zeige_ptr_swap(TextBox^ tb)
{
    int x = 17, y = 18;
    tb->AppendText("x=" + x.ToString() + " y=" + y.ToString() + "\r\n"); // x=17 y=18
    ptr_swap(&x, &y);
    tb->AppendText("x=" + x.ToString() + " y=" + y.ToString() + "\r\n"); // x=18 y=17
}
```

### 5.3 Aufgabe 7.5 Dynamisch erzeugte eindimensionale Arrays

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_7_Zeiger.h
// ----- Aufgabe 1. -----
void DynSiebEratosthenes(TextBox^ tb, int n)
{
    // Die einzigen beiden Änderungen: n muss keine Konstante
    // mehr sein, und das Array wird dynamisch erzeugt:
    //                               const int n=1000;
    bool* prim = new bool[n + 1]; // statt 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])
            tb->AppendText(i.ToString() + "\r\n");
    delete[] prim;
};

// ----- Aufgabe 2. -----

// a)
using T =int;    // T irgendein Datentyp, wie typedef int T;

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;    // nicht vergessen
}

void test_ReAllocate_2()
{
    // a)
    int* p1;
    ReAllocate(p1, 0, 100);
    // Der nicht initialisierte Zeiger p1 wird in
    // ReAllocate mit delete[] freigegeben - Fehler.

    // b)
    // Der mit new int initialisierte Zeiger p2 wird in
    // ReAllocate mit delete[] freigegeben - Fehler.

    // c)
    int x;
    int* p3 = &x;
    ReAllocate(p3, 0, 100);
}
```



```
// Die mit int x angelegte Variable wird in  
// ReAllocate mit delete[] freigegeben.  
// p3 zeigt nicht auf einen Speicherbereich, der  
// mit new angelegt wurde. Der Aufruf von delete  
// in ReAllocate ist ein Fehler.  
  
// d)  
// wie in test_ReAllocate_1 - das funktioniert.  
}
```

## 5.4 Aufgabe 7.6 Stringlitterale, nullterminierte Strings, ...

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_7_Zeiger.h
// ----- 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)
void Test_Leerzeichen(TextBox^ tb)
{ // ein einfacher Test für die Funktion Leerzeichen
  // { LZ("");0 | LZ("1");0 | LZ("1 ");1 | LZ(" 1");1 | LZ(" 1 ");2 }
  // Am einfachsten: Die Werte im Debugger anschauen
  int n1 = Leerzeichen(""); // 0
  int n2 = Leerzeichen("1"); // 0
  int n3 = Leerzeichen("1 "); // 1
  int n4 = Leerzeichen(" 1"); // 1
  int n5 = Leerzeichen(" 1 "); // 2
  // oder die Werte in einer TextBox ausgeben:
  tb->AppendText("Leerzeichen('') = " + n1.ToString() + "\r\n");
  tb->AppendText("Leerzeichen('1') = " + n2.ToString() + "\r\n");
  tb->AppendText("Leerzeichen('1 ') = " + n3.ToString() + "\r\n");
  tb->AppendText("Leerzeichen(' 1') = " + n4.ToString() + "\r\n");
  tb->AppendText("Leerzeichen(' 1 ') = " + n5.ToString() + "\r\n");
}
#ifdef SIMPLEUNITTESTS_H
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_Leerzeichen()
{ // ein einfacher Test für die Funktion Leerzeichen
  // { LZ("");0 | LZ("1");0 | LZ("1 ");1 | LZ(" 1");1 | LZ(" 1 ");2 }

  rkl::Assert::AreEqual(0, Leerzeichen(""), "Leerzeichen('')");
  rkl::Assert::AreEqual(0, Leerzeichen("1"), "Leerzeichen('1')");
  rkl::Assert::AreEqual(1, Leerzeichen("1 "), "Leerzeichen('1 ')");
  rkl::Assert::AreEqual(1, Leerzeichen(" 1"), "Leerzeichen(' 1')");
  rkl::Assert::AreEqual(2, Leerzeichen(" 1 "), "Leerzeichen(' 1 ')");
}
#endif // #ifdef SIMPLEUNITTESTS_H

// ----- Aufgabe 2. -----

void charPtr_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 charPtr_3(TextBox^ tb)
{
    char const* s = "blablabla";
    // a)
    if (s == " "); // Vergleicht die Adressen der beiden Stringlitterale
                  // "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)
    std::string s1 = s;
    std::string s2 = s + 1;
    std::string s3 = s + 20;

    tb->AppendText(rk1::to_String(s) + "\r\n"); // "blablabla"
    tb->AppendText(rk1::to_String(s + 1) + "\r\n"); // "lablabla"
    tb->AppendText(rk1::to_String(s + 20) + "\r\n"); // 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 -----

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 Checksum_Aufruf()
{
    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 5 -----

```

```
void charPtr_5() // konstante Zeiger
{
    const int i = 17;
    int* p1 = nullptr;    char* q1 = nullptr;
    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; // cannot 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 // In VS2019 ein Fehler
    // h)
    q2 = q1;
    // i)
    // *q3='x'; // cannot modify const object
    // j)
    // *q4='x'; // Syntaktisch ok, aber kein Speicher reserviert für *p4.
}
```

## 5.5 Aufgabe 7.7 Arrays, Zeiger und Zeigerarithmetik

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_7_Zeiger.h
// ----- Aufgabe 1. -----

void Aufgabe_Zeigerarithmetik(TextBox^ tb)
{
    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;
    tb->AppendText((*p).ToString() + "\r\n"); // = a[0] = 1
    tb->AppendText((*p + 1).ToString() + "\r\n"); // = a[0]+1 = 2
    tb->AppendText((*p) + 1).ToString() + "\r\n"); // = a[0]+1 = 2
    tb->AppendText(*(p + 1).ToString() + "\r\n"); // = a[1] = 3
    tb->AppendText(*(p + 3 * *p).ToString() + "\r\n"); // = a[3] = 7
    tb->AppendText(*p * *p).ToString() + "\r\n"); // = a[0]*a[0] = 1
}
```

## 5.6 Aufgabe 7.8 Verkettete Listen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_7_Zeiger.h

namespace N_LinkedList {
    // ----- Aufgabe 1 -----

    using T = int; // Falls Werte anderer Datentypen als int in der
                  // Liste gespeichert werden sollen, muss der
                  // Datentyp nur hier geändert werden.
    // oder statt using
    // typedef int T;

    struct Listnode {
        T data;           // Daten
        Listnode* next;
    };

    Listnode* first = nullptr;
    Listnode* last = nullptr;
    // 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 VerketteteListe{ ... } 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 == nullptr)
          last = first;                    //          n0
                                           // Nachbedingung:
                                           // Fall I, f0==l0==0: first==last->{d0,0}
                                           // Fall II, l0!=0:   first->{d0,f0}
    }

    // Size ist für einige der folgenden Tests notwendig. Diese Funktion
    // prüft insbesondere auch, ob mit t=t->next die Bedingung t==0
    // erreicht wird.
    int Size(Listnode* t)
    { // gibt die Anzahl der auf t folgenden Knoten zurück
      int n = 0;
      while (t != nullptr)
      {
          n++;
          t = t->next;
      }
      return n;
    }

    void clearList(); // Prototyp, siehe Aufgabe h)
                    // Für die Aufgaben bis h) reicht es, in dieser Funktion
                    // first und last den Wert 0 zuzuweisen.

#ifdef SIMPLEUNITTESTS_H
    // Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

```

```

void Unittests_pushFront(int n)
{ // Füge Knoten ein und prüfe first, last und Size
  rk1::Assert::WriteLine("test_pushFront n=" + std::to_string(n));
  clearList();
  pushFront(5);
  rk1::Assert::AreEqual(first->data, 5, "pushFront 5 first");
  rk1::Assert::AreEqual(last->data, 5, "pushFront 5 last");
  rk1::Assert::AreEqual(Size(first), 1, "Size()==1");

  pushFront(10);
  rk1::Assert::AreEqual(first->data, 10, "pushFront 10 first");
  rk1::Assert::AreEqual(last->data, 5, "pushFront 5 last");
  rk1::Assert::AreEqual(Size(first), 2, "Size()==2");

  // Viele Tests
  clearList();
  for (int i = 0; i < n; i++)
  {
    pushFront(i);
    rk1::Assert::AreEqual(first->data, i, "pushFront 0 first");
    rk1::Assert::AreEqual(last->data, 0, "pushFront n last");
    rk1::Assert::AreEqual(Size(first), i + 1, "Size()==n");
  }
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// ----- Aufgabe 1 b) -----
void showList(TextBox^ tb, Listnode* start)
{ // Gibt alle Daten der Liste ab der Position start aus
  tb->AppendText("Liste:\r\n");
  for (Listnode* tmp = start; tmp != 0; tmp = tmp->next)
    tb->AppendText(tmp->data.ToString() + " ");
  tb->AppendText("\r\n");
}

void test_showList()
{ // Ein automatischer Test von showList müsste den Text in der
  // TextBox prüfen. Da das relativ aufwendig ist, scheint ein
  // manueller Test angemessener zu sein.
}
// ----- 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* findLinear(Listnode* start, const T& x)
{
  Listnode* found = nullptr;
  Listnode* tmp = start;
  while (tmp != nullptr && found == nullptr)
  {
    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 FindLinearClick(TextBox^ tb, int x)
{
    Listnode* p = findLinear(first, x);
    if (p == nullptr)
        tb->AppendText(x.ToString() + ": nicht gefunden\r\n");
    while (p != nullptr)
    {
        tb->AppendText(p->data.ToString() + ": gefunden");
        p = findLinear(p->next, x);
    }
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_findLinear()
{ // Einige Testfälle
    rk1::Assert::WriteLine("test_findLinear");
    clearList();
    // 1. Suche in der leeren Liste sollte funktionieren,
    // insbesondere nicht zu einem Programmabsturz führen
    rk1::Assert::AreEqual(int(findLinear(first, 1)), 0, "find in empty list");

    // 2. Ein Element einfügen und finden:
    pushFront(1);
    rk1::Assert::AreEqual(1, findLinear(first, 1)->data, "Liste mit '1'");
    //pushFront(2);
    // suche einen Wert, der nicht in der Liste enthalten ist
    rk1::Assert::AreEqual(0, int(findLinear(first, 2)), "Liste mit '1'");
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// ----- Aufgabe 1 e) -----

void insertLastListnode(const T& data) // first last Neuer Knoten n
// f0 10
{ // 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, nullptr); // n0 ->{d0,0}
    if (last == nullptr) first = n; // n0
    else last->next = n; // 10->{d,n0}
    last = n; // n0
// Nachbedingung: Bezeichnet man den Wert
von last vor // dem Aufruf dieser Funktion mit 10, gilt
// Fall I, 10==0: first==n && last==n
// Fall II, 10!=0: 10->next==n && last==n
}

void pushBack(const T& data)
{
    insertLastListnode(data);
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

```



```

void Unittests_pushBack(int n)
{
    rk1::Assert::WriteLine("test_pushBack" + std::to_string(n));
    clearList();
    insertLastListnode(5);
    rk1::Assert::AreEqual(first->data, 5, "pushBack 5 first");
    rk1::Assert::AreEqual(last->data, 5, "pushBack 5 last");
    rk1::Assert::AreEqual(Size(first), 1, "Size()==1");

    insertLastListnode(10);
    rk1::Assert::AreEqual(first->data, 5, "pushBack 10 first");
    rk1::Assert::AreEqual(last->data, 10, "pushBack 5 last");
    rk1::Assert::AreEqual(Size(first), 2, "Size()==2");

    // Viele Tests
    clearList();
    for (int i = 0; i < n; i++)
    {
        insertLastListnode(i);
        rk1::Assert::AreEqual(first->data, 0, "pushBack 0 first");
        rk1::Assert::AreEqual(last->data, i, "pushBack n last");
        rk1::Assert::AreEqual(Size(first), i + 1, "Size()==n");
    }
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// ----- Aufgabe 1 f) -----

Listnode* findBefore(const T& data)
{
    // Wird von insertSorted verwendet: Gibt einen Zeiger auf den ersten Knoten
    // zurück, für den data<=tmp->data gilt
    Listnode* result = first;
    Listnode* tmp = first;
    while (tmp != nullptr && 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 == nullptr)
    { // 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);
    }
  }
}

bool isSorted(Listnode* n)
{
  bool result = true;
  Listnode* tmp = n;
  T old;
  if (tmp != nullptr)
    old = tmp->data;
  while (tmp != nullptr && result)
  {
    if (tmp->data < old)
      result = false;
    old = tmp->data;
    tmp = tmp->next;
  }
  return result;
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_isSorted()
{
  rk1::Assert::WriteLine("test_isSorted");
  clearList();
  rk1::Assert::AreEqual(isSorted(first), true, "insertSorted: leere Liste");
  insertSorted(5);
  rk1::Assert::AreEqual(isSorted(first), true, "insertSorted: Liste mit einem
Element");
  insertSorted(3);
  rk1::Assert::AreEqual(isSorted(first), true, "insertSorted: Liste mit einem
Element");
  insertSorted(7);
  rk1::Assert::AreEqual(isSorted(first), true, "insertSorted: three element list not
sorted");
}

void Unittests_insertSorted_random(int max)
{
  rk1::Assert::WriteLine("test_insertSorted_random" + std::to_string(max));
  clearList();
  for (int i = 0; i < max; i++)
  {
    insertSorted(rand());
    rk1::Assert::AreEqual(isSorted(first), true, "insertSorted_random");
  }
  rk1::Assert::AreEqual(isSorted(first), true, std::to_string(max) + "element list
sorted");
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// ----- Aufgabe 1 g) -----

void eraseListnode(Listnode*& pn)
{ // entfernt *p aus der Liste
  if (pn != nullptr)//falls pn==0, nichts machen oder Fehlermeldung
  {
    Listnode* tmp = pn;
    pn = pn->next;
    delete tmp;
  }
}

```

```

}

void clearList()
{ // löscht alle Knoten der Liste
  while (first != nullptr)
    eraseListNode(first);
  last = nullptr;
}

void test_clearList()
{
  // siehe dazu die Ausführungen in Abschnitt 7.4.3
}

// ----- Aufgabe 1 h) -----
/*
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.
*/

#ifdef SIMPLEUNITTESTS_H__
  // Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

  void Unittests_LinkedList(int n)
  {
    rk1::Assert::Init("Unittests_LinkedList");
    Unittests_pushFront(n);
    // showList(tb, first);
    Unittests_findLinear();
    Unittests_pushBack(n);
    // showList(tb, first);
    Unittests_isSorted();
    Unittests_insertSorted_random(n);
    // showList(tb, first);
    rk1::Assert::Summary();
  }
#endif // #ifdef SIMPLEUNITTESTS_H__

// ----- 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 == nullptr)
  { // erster Aufruf
    firstDll = newDllListNode(data, 0, 0); //n0 n0->{d0,0,0}
    lastDll = firstDll; // n0
  }
  else
  { // ab dem zweiten Aufruf
    firstDll = newDllListNode(data,
firstDll, nullptr); //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 clearDllList(); // Prototyp, siehe Aufgabe

int Size(DllListNode* t)
{
  int n = 0;
  while (t != nullptr)
  {
    n++;
    t = t->next;
  }
  return n;
}

int SizeRev(DllListNode* t)
{
  int n = 0;
  while (t != nullptr)
  {
    n++;
    t = t->prev;
  }
  return n;
}

#ifdef SIMPLEUNITTESTS_H_
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_pushFrontDll(int n)
{
  clearDllList();
  pushFrontDll(5);
  rk1::Assert::AreEqual(firstDll->data, 5, "pushFrontDll 5 first");
  rk1::Assert::AreEqual(lastDll->data, 5, "pushFrontDll 5 last");
  rk1::Assert::AreEqual(Size(firstDll), 1, "Dll Size()==1");
  rk1::Assert::AreEqual(SizeRev(lastDll), 1, "Dll SizeRev()==1");
}

```

```

pushFrontDll(10);
rk1::Assert::AreEqual(firstDll->data, 10, "pushFront Dll 10 first");
rk1::Assert::AreEqual(lastDll->data, 5, "pushFront Dll 5 last");
rk1::Assert::AreEqual(Size(firstDll), 2, "Dll Size()==2");
rk1::Assert::AreEqual(SizeRev(lastDll), 2, "Dll SizeRev()==2");

// Viele Tests
clearDllList();
for (int i = 0; i < n; i++)
{
    pushFrontDll(i);
    rk1::Assert::AreEqual(firstDll->data, i, "pushFront 0 first");
    rk1::Assert::AreEqual(lastDll->data, 0, "pushFront n last");
    rk1::Assert::AreEqual(Size(firstDll), i + 1, "Size()==n");
    rk1::Assert::AreEqual(SizeRev(lastDll), i + 1, "SizeRev()==n");
}
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// ----- Aufgabe 2 c) -----
void showDllForw(TextBox^ tb, DllListnode* start)
{ // Gibt alle Daten der Liste ab der Position start aus
  tb->AppendText("Dll-Liste vorwärts:\r\n");
  DllListnode* tmp = start;
  while (tmp != nullptr)
  {
      tb->AppendText(tmp->data.ToString() + " ");
      tmp = tmp->next;
  }
  tb->AppendText("\r\n");
}

// ----- Aufgabe 2 d) -----

void showDllRev(TextBox^ tb, DllListnode* start)
{ // Gibt alle Daten der Liste ab der Position first aus
  tb->AppendText("Liste rückwärts:\r\n");
  DllListnode* tmp = start;
  while (tmp != nullptr)
  {
      tb->AppendText(tmp->data.ToString() + " ");
      tmp = tmp->prev;
  }
  tb->AppendText("\r\n");
}

// ----- 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
lD=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 == nullptr)

```

```

    {
        firstDll = newDllListNode(data, nullptr, nullptr);
        lastDll = firstDll;
    }
    else
    {
        DllListNode* n = newDllListNode(data, nullptr, 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,*}
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_pushBackDll(int n)
{
    clearDllList();
    pushBackDll(5);
    rk1::Assert::AreEqual(firstDll->data, 5, "pushBack 5 first");
    rk1::Assert::AreEqual(lastDll->data, 5, "pushBack 5 last");
    rk1::Assert::AreEqual(Size(firstDll), 1, "Size()==1");
    rk1::Assert::AreEqual(SizeRev(lastDll), 1, "SizeRev()==1");

    pushBackDll(10);
    rk1::Assert::AreEqual(firstDll->data, 5, "pushBack 10 first");
    rk1::Assert::AreEqual(lastDll->data, 10, "pushBack 10 last");
    rk1::Assert::AreEqual(Size(firstDll), 2, "Size()==2");
    rk1::Assert::AreEqual(SizeRev(lastDll), 2, "SizeRev()==2");

    // Viele Tests
    clearDllList();
    for (int i = 0; i < n; i++)
    {
        pushBackDll(i);
        rk1::Assert::AreEqual(firstDll->data, 0, "pushBack 0 first");
        rk1::Assert::AreEqual(lastDll->data, i, "pushBack n last");
        rk1::Assert::AreEqual(Size(firstDll), i + 1, "Size()==n");
        rk1::Assert::AreEqual(SizeRev(lastDll), i + 1, "SizeRev()==n");
    }
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// ----- 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
lD=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* findDllLinear(DllListNode* start, const T& x)
{
    DllListNode* found = nullptr;
    DllListNode* tmp = start;

```

```

while (tmp != nullptr && found == nullptr)
{
    if (tmp->data == x) found = tmp;
    tmp = tmp->next;
}
return found;
}

void eraseDllListNode(DllListNode*& pn)
{ // entfernt *pn aus der Liste
  if (pn != nullptr) // für pn==nullptr nichts machen
  {
    DllListNode* tmp = pn;
    if (pn->next != nullptr && pn->prev != nullptr)
    { // pn zeigt weder auf den ersten noch den letzten Knoten
      // firstDll und lastDll bleiben unverändert
      pn->prev->next = pn->next;
      pn->next->prev = pn->prev;
    }
    else if (pn->next == nullptr && pn->prev != nullptr)
    { // pn zeigt auf den letzten Knoten, hat aber davor noch Knoten
      // lastDll muss angepasst werden.
      pn->prev->next = nullptr;
      lastDll = pn->prev;
    }
    else if (pn->next != nullptr && pn->prev == nullptr)
    { // pn zeigt auf den ersten Knoten, hat aber danach noch Knoten
      // firstDll muss angepasst werden.
      firstDll = pn->next;
      firstDll->prev = nullptr;
    }
    else if (pn->next == nullptr && pn->prev == nullptr)
    { // pn zeigt auf den einzigen Knoten der Liste.
      // firstDll muss angepasst werden.
      firstDll = nullptr;
      lastDll = nullptr;
    }
    delete tmp;
  }
}

void EraseDllListNodeClick(TextBox^ tb, int x)
{
  DllListNode* p = findDllLinear(firstDll, x);
  if (p != nullptr)
  {
    eraseDllListNode(p);
    tb->AppendText(x.ToString() + ": gelöscht \r\n");
  }
  else
    tb->AppendText(x.ToString() + ": nicht gefunden \r\n");
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_eraseDllListNode(int max)
{
  rkl::Assert::WriteLine("test_eraseDllListNode" + std::to_string(max));
  clearDllList();
  for (int i = 0; i < max; i++)
    pushFrontDll(i);

  for (int i = 0; i < max / 2; i++)
  {
    int x = rand() % max;          // lösche zufällige Werte
                                // die nächsten 3 Anweisungen stellen sicher, dass auch
das erste,
                                // letzte und ein zufällig
    if (i == 0) x = 0;           // lösche das erste Element
    else if (i == 1) x = max - 1; // lösche das letzte Element
  }
}

```

```

else if (i == 2) x = max; // lösche ein nicht vorhandenes Element
                        // rk1::Assert::WriteLine("lösche "+x.ToString());
DllListNode* p = findDllLinear(firstDll, x);
if (p != nullptr)
    eraseDllListNode(p);
// showDllForw(tb, firstDll);
// Application::DoEvents();
}
for (int i = 0; i < max; i++)
{ // lösche alle restlichen Elemente
  // rk1::Assert::WriteLine("lösche "+i.ToString());
  DllListNode* p = findDllLinear(firstDll, i);
  if (p != nullptr)
    eraseDllListNode(p);
  // showDllForw(tb, firstDll);
  // Application::DoEvents();
}
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// ----- Aufgabe 2 i) -----

void clearDllList()
{ // löscht alle Knoten der Liste
  while (firstDll != nullptr) eraseDllListNode(firstDll);
  firstDll = nullptr;
  lastDll = nullptr;
}

// ----- Aufgabe 2 j) -----

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_DllList(int n)
{
  rk1::Assert::Init("Unittests_DllList");
  Unittests_pushFrontDll(n);
  Unittests_pushBackDll(n);
  Unittests_eraseDllListNode(n);
  rk1::Assert::Summary();
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// ----- Aufgabe 3 -----

//$$$ InsertWordDoc Ind.rtf

} // end of namespace N_LinkedList

```



## 5.7 Aufgabe 7.9 Binärbäume

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_7_Zeiger.h

// ----- Aufgabe 1 -----

// ----- Aufgabe 1 a) -----

using dataType = int; // wie typedef int dataType;
using keyType = int; // wie typedef int keyType;

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 1 b) -----

void insertBinTreenode(Treenode*& root, const keyType& key, const dataType& data)
{
    if (root == nullptr)
        root = newTreenode(key, data, nullptr, nullptr);
    else
    {
        Treenode* i = root;
        Treenode* p = nullptr;
        while (i != nullptr)
        {
            p = i;
            if (key < i->key) i = i->left;
            else i = i->right;
        }
        if (key < p->key)
            p->left = newTreenode(key, data, nullptr, nullptr);
        else
            p->right = newTreenode(key, data, nullptr, nullptr);
    }
}

Treenode* root = nullptr;

void InsertClick(keyType x, dataType y)
{
    insertBinTreenode(root, x, y);
}

// ----- Aufgabe 1 c) -----

Treenode* searchBinTree(const keyType& x)
{
    Treenode* result = nullptr;
    if (root != nullptr)
    {
        Treenode* i = root;
        while (i != nullptr && i->key != x)

```

```

    {
        if (x < i->key) i = i->left;
        else i = i->right;
    }
    if (i != nullptr && i->key == x)
        result = i;
}
return result;
}

bool ValueToKey(const keyType& Key, dataType& Value)
{
    Treenode* p = searchBinTree(Key);
    if (p == nullptr)
        return false;
    else
    {
        Value = p->data;
        return true;
    }
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_BinTree()
{ // Einige einfache Testfälle
    root = nullptr;
    dataType Value;
    rkl::Assert::Init("Unittests_BinTree");

    // 1. Suche im leeren Baum sollte funktionieren, und
    // insbesondere nicht zu einem Programmabsturz führen
    rkl::Assert::AreEqual(ValueToKey(1, Value), false, "leerer Baum");

    // 2. Ein Element einfügen und finden:
    insertBinTreenode(root, 1, 2);
    rkl::Assert::AreEqual(ValueToKey(1, Value), true, "Baum mit '1'");
    rkl::Assert::AreEqual(Value, 2, "Baum mit '1'");

    // 2. Zwei Elemente einfügen und finden:
    insertBinTreenode(root, 2, 3);
    rkl::Assert::AreEqual(ValueToKey(1, Value), true, "Baum mit '1'");
    rkl::Assert::AreEqual(Value, 2, "Baum mit '1'");
    rkl::Assert::AreEqual(ValueToKey(2, Value), true, "Baum mit '1' und '2'");
    rkl::Assert::AreEqual(Value, 3, "Baum mit '1' und '2'");
    rkl::Assert::Summary();
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// d)
void insertBinTreenode_rec(Treenode*& node,
    const keyType& key, const dataType& data)
{
    if (node == nullptr)
        node = newTreenode(key, data, nullptr, nullptr);
    else if (key < node->key)
        insertBinTreenode_rec(node->left, key, data);
    else
        insertBinTreenode_rec(node->right, key, data);
}

// e)
void processTreenode(TextBox^ tb, Treenode* n)
{
    tb->AppendText("key:" + n->key + " data:" + n->data + "\r\n");
};

void traverseTree_rec(TextBox^ tb, Treenode* n)

```

```

{
    if (n != nullptr)
    {
        traverseTree_rec(tb, n->left);
        processTreenode(tb, n);
        traverseTree_rec(tb, n->right);
    }
}

void TraverseClick(TextBox^ tb)
{
    traverseTree_rec(tb, root);
}

// f)
Treenode* searchBinTree_rec(Treenode* tn, const dataType& x)
{
    if (tn == nullptr)
        return nullptr;
    else if (x == tn->key)
        return tn;
    else if (x < tn->key)
        return searchBinTree_rec(tn->left, x);
    else
        return searchBinTree_rec(tn->right, x);
}

bool ValueToKey_rec(keyType Key, dataType& Value)
{
    Treenode* p = searchBinTree_rec(root, Key);
    if (p == nullptr)
        return false;
    else
    {
        Value = p->data;
        return true;
    }
}

#ifdef SIMPLEUNITTESTS_H_
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_BinTree_recursive()
{ // Einige Testfälle
    rk1::Assert::Init("Unittests_BinTree_recursive");

    root = nullptr;
    dataType Value;

    // 1. Suche im leeren Baum sollte funktionieren,
    // insbesondere nicht zu einem Programmabsturz führen
    rk1::Assert::AreEqual((int)searchBinTree_rec(root, 1), 0, "searchBinTree: leerer
Baum");
    rk1::Assert::AreEqual(ValueToKey_rec(1, Value), false, "ValueToKey_rec: leerer Baum");

    // 2. Suche in einem Baum mit einem Element:
    insertBinTreenode_rec(root, 1, 10);
    // rk1::Assert::AreNotEqual((int)searchBinTree_rec(root, 1), 1, "searchBinTree: 1
Element, gefunden");
    rk1::Assert::AreEqual((int)searchBinTree_rec(root, 2), 0, "searchBinTree: 1 Element,
Knoten mit key 2 nicht gefunden");

    rk1::Assert::AreEqual(ValueToKey_rec(1, Value), true, "Baum mit key '1', Knoten mit key
1 gefunden");
    rk1::Assert::AreEqual(Value == 10, true, "Baum mit key '1', Knoten mit Wert 10
gefunden");

    // 3. Suche in einem Baum mit 3 Elementen:
    insertBinTreenode_rec(root, 2, 20);
    rk1::Assert::AreEqual((int)searchBinTree_rec(root, 3), 0, "searchBinTree: 2 Elemente,
Knoten mit key 3 nicht gefunden");
}

```

```

    rkl::Assert::AreEqual(ValueToKey_rec(2, Value), true, "Baum mit 2 Elementen, Knoten mit
key 2 gefunden");
    rkl::Assert::AreEqual(Value == 20, true, "Baum mit 2 Elementen, Knoten mit Wert 20
gefunden");

    const int MaxTests = 10;
    for (int i = 3; i < MaxTests; i++)
    {
        int x = i;
        if (i < 50) x = -i;
        insertBinTreenode_rec(root, x, 10 * x);
    }

    for (int i = 3; i < MaxTests; i++)
    {
        int x = i;
        if (i < 50) x = -i;

        rkl::Assert::AreEqual(ValueToKey_rec(x, Value), true, "Baum mit vielen Elementen,
Knoten gefunden");
        rkl::Assert::AreEqual(Value == 10 * x, true, "Baum mit vielen Elementen, Knoten mit
Wert gefunden");
    }
    rkl::Assert::Summary();
}

#endif // #ifndef SIMPLEUNITTESTS_H__

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

void Unittests_Zeiger()
{
    int n = 100;
    rkl::Assert::Init("Unittests Zeiger");
    Unittests_Leerzeichen();
    N_LinkedList::Unittests_LinkedList(n);
    N_LinkedList::Unittests_DllList(n);
    Unittests_BinTree();
    Unittests_BinTree_recursive();
    rkl::Assert::Summary();
}
#endif // #ifndef SIMPLEUNITTESTS_H__

} // end of namespace N_Loesungen_Kap_Zeiger

```

## 6 Lösungen Kapitel 8. Überladene Funktionen und Operatoren

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_8_Ueberlad_Fkt.h
```

```
#include <cmath> // für pow bei Bruchrechnung
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
```

```
namespace N_Loesungen_Ueberladene_Funktionen_und_Operatoren
{
    using namespace System;
    using namespace System::Windows::Forms;
```

## 6.1 Aufgabe 8.2 Überladene Funktionen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_8_Ueberlad_Fkt.h

// ----- Aufgabe 1 -----

// a)
int fkt(double d) { return 0; }; // ursprüngliche Funktion
int fkt(int i) { return 1; }; // später hinzugefügt
int x = fkt(42);

// Vor der Ergänzung wird die double-Variante aufgerufen,
// danach die int-Variante, d.h. x erhält vorher den Wert 0
// und danach den Wert 1.

// b)
// Für Parameter eines Klassentyps ohne impliziten Konversionsoperator
// findet keine Konversion statt.

// ----- Aufgabe 2 -----

// a)
void f(std::string s, std::string format) {};
void f(std::string s) {};
void f() {};
// Die Funktionen unter a) können durch eine Funktion
// mit Default-Argumenten ersetzt werden:

void f1(std::string s = "", std::string format = "") {}

// b)
void g(int n) {};
void g(double d) {};
void g(double d, std::string s) {};

// Diese Funktionen können nicht durch eine Funktion mit
// Default-Argumenten ersetzt werden, da die Parameter-Typen
// verschieden sind.

// c)

namespace N_InitBruch {

    struct Bruch {
        int z, n; // z: Zähler, n: Nenner
    };

    Bruch initBruch(int z, int n)
    {
        return { z,n };
    }

    Bruch initBruch(int z)
    {
        return { z,1 };
    }

    Bruch initBruch()
    {
        return { 1,1 };
    }

    // Variante mit Default-Argumenten:

    Bruch initBruch_(int z = 1, int n = 1) // mit einer Klasse Bruch: Konstruktor
    { // InitBruch(x): x/1
        return { z,n };
    } //
} // end of namespace N_InitBruch

// d)
```

```
// Default-Argumente sind meist die bessere Alternative, da
// nur eine einzige Funktion geschrieben und gewartet werden muss.

// ----- Aufgabe 3 -----

float sqrt(float x) { return 0; };
double sqrt(double x) { return 0; };
long double sqrt(long double x) { return 0; };

void test_sqrt()
{
    // double x1 = sqrt(1); // mehrdeutig
    double x2 = sqrt(1.0);
    double x3 = sqrt(1.0l); // ein long double Argument
}
```

## 6.2 Aufgabe 8.3.1 Globale Operatorfunktionen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_8_Ueberlad_Fkt.h

// ----- Aufgabe 1 -----

// 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)
{
    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);
};

// d) Zum Initialisieren:

Bruch initBruch(int z = 1, int n = 1) // mit einer Klasse Bruch: Konstruktor
{ // InitBruch(x): x/1
    return { z, n };
}

std::string to_string(Bruch q)
{
    return std::to_string(q.z) + "/" + std::to_string(q.n);
};

void zeigeBruchrechnung(TextBox^ tb)
{
    Bruch b1 = initBruch(1, 2);
    Bruch b2 = initBruch(3, 4);

    std::string r_plus = to_string(b1) + "+" + to_string(b2) + "=" + to_string(b1 + b2);
    std::string r_minus = to_string(b1) + "-" + to_string(b2) + "=" + to_string(b1 - b2);
    std::string r_mult = to_string(b1) + "*" + to_string(b2) + "=" + to_string(b1*b2);
    std::string r_div = to_string(b1) + "/" + to_string(b2) + "=" + to_string(b1 / b2);

    tb->AppendText(rkl::to_String(r_plus + "\r\n"));
    tb->AppendText(rkl::to_String(r_minus + "\r\n"));
    tb->AppendText(rkl::to_String(r_mult + "\r\n"));
    tb->AppendText(rkl::to_String(r_div + "\r\n"));
}

// d)
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);
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
bool Unittests_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 std::pow;
          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
          rkl::Assert::AreEqual(s1 == s2, true,
            "s1=" + to_string(s1) + " s2=" + to_string(s2));
          //result=false;
          rkl::Assert::AreEqual(s1 == s3, true,
            "s1=" + to_string(s1) + " s2=" + to_string(s2));
          //result=false;
        }
  return result;
}

void BruchOperatoren_Aufruf()
{
  Unittests_BruchOperatoren(7, 7); // größere Werte: Bereichsüberlauf
}
#endif // #ifdef SIMPLEUNITTESTS_H__

```

### 6.3 Aufgabe 8.3.2 Ein-/Ausgabeoperatoren

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_8_Ueberlad_Fkt.h
//----- Aufgabe 1 -----

std::ostream& operator<<(std::ostream& f, const Bruch& b)
{
    return f << b.z << "|" << b.n;
}

std::istream& operator >> (std::istream& f, Bruch& b)
{
    char Bruchstrich; // Bruchstrich einlesen und ignorieren.
    // Eventuell auch noch prüfen, ob tatsächlich ein Bruchstrich
    f >> b.z >> Bruchstrich >> b.n;
    return f;
}

std::string BruchToStr(Bruch b)
{
    std::ostringstream o;
    o << b;
    return o.str();
}

Bruch StrToBruch(std::string s)
{
    std::istringstream i(s);
    Bruch b;
    i >> b;
    return b;
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
bool Unittests_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 };
                std::string s = BruchToStr(b1);
                Bruch b2 = StrToBruch(s);
                rkl::Assert::AreEqual(b1 == b2, true,
                    "b1=" + to_string(b1) + " b2=" + to_string(b2));
            }
    return result;
}
#endif // #ifdef SIMPLEUNITTESTS_H__
```

## 6.4 Aufgabe 8.4 Referenztypen, Wert- und Ref-Parameter

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_8_Ueberlad_Fkt.h
```

```
// ----- Aufgabe 1 -----

// a)
void vertauschel(int& x, int& y)
{
    // if (&x == &y)
    //   x           y
    //   x0          x0

    {
        x = x + y; // x0+x0          x0+x0
        y = x - y; // (x0+x0)-(x0+x0)=0 (x0+x0)-(x0+x0)=0
        x = x - y; // 0-0=0
    }
} // x==0 && y==0

// Unabhängig vom ursprünglichen Wert des Arguments
// hat dieses nach dem Aufruf immer den Wert 0

// b)
// Falls die Argumente gleich sind, nichts machen:

void vertauschela(int& x, int& y)
{
    //   x           y
    //   x0          y0
    if (&x != &y)
    {
        x = x + y; // x0+y0          x0+y0
        y = x - y; // (x0+y0)-(x0+y0)=0 (x0+y0)-(x0+y0)=0
        x = x - y; // 0-0=0
    }
} // x==y0 && y==x0

// c)
void vertausche(int& a, int& b)
{
    // hier werden die Argumente nur kopiert. Deswegen
    // funktioniert sie auch mit zwei gleichen Argumenten.
    int h = a;
    a = b;
    b = h;
}

void Button_3_18_3Click(TextBox^ tb)
{
    int a = 17, b = 18;
    tb->AppendText("vor vertauschel(a,b): a=" + a.ToString() + " b=" +
        b.ToString() + "\r\n");
    vertauschel(a, b);
    tb->AppendText("nach vertauschel(a,b): a=" + a.ToString() + " b=" +
        b.ToString() + "\r\n");

    tb->AppendText("vor vertauschel(a,a): a=" + a.ToString() + "\r\n");
    vertauschel(a, a);
    tb->AppendText("nach vertauschel(a,a): a=" + a.ToString() + "\r\n");

    a = 17;
    b = 18;
    tb->AppendText("vor vertausche(a,b): a=" + a.ToString() + " b=" +
        b.ToString() + "\r\n");
    vertausche(a, b);
    tb->AppendText("nach vertausche(a,b): a=" + a.ToString() + " b=" +
        b.ToString() + "\r\n");
    tb->AppendText("vor vertausche(a,a): a=" + a.ToString() + "\r\n");
    vertausche(a, a);
}
```

```
    tb->AppendText("nach vertausche(a,a): a=" + a.ToString() + "\r\n");
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Alle_Unittests()
{
    rkl::Assert::Init("Unittests_GlobOperatorfunktionen");
    Unittests_BruchOperatoren(7, 7); // größere Werte: Bereichsüberlauf
    Unittests_BruchKonversion(10, 10);
    rkl::Assert::Summary();
}
#endif // #ifdef SIMPLEUNITTESTS_H__

} // end of namespace N_Ueberladene_Funktionen_und_Operatoren
```



## 7 Lösungen Kapitel 9. Objektorientierte Programmierung

```
##### Copyright notice #####  
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook  
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",  
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_00_1.h
```

## 7.1 Aufgabe 9.1.5 Klassen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_OO_1.h
#include <fstream>

namespace N_Loesungen_Klassen
{
    using namespace System::Windows::Forms;

    // ----- 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(TextBox^ tb)
        {
            for (int i = 0; i < n; i++)
                tb->AppendText(a[i].ToString() + "\r\n");
        }

        // Der Destruktor mit delete[] a fehlt
    };

    // ----- 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)
class C3 { // #include <fstream>
    std::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.

// ----- 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;
    }
// d)
    double Flaeche()
    {
        return r * r * pi;
    }

    double Umfang()
    {
        return 2 * r * pi;
    }

    std::string toStr()
    {
        return "Kreis mit Radius " + std::to_string(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();  
    std::string toStr();  
};  
  
Quadrat::Quadrat(const double& Seitenlaenge)  
{  
    a = Seitenlaenge;  
};  
  
double Quadrat::Seitenlaenge()  
{  
    return a;  
}  
  
void Quadrat::setzeSeitenlaenge(const double& Seitenlaenge)  
{  
    a = Seitenlaenge;  
}  
  
// d)  
double Quadrat::Flaeche()  
{  
    return a * a;  
}  
  
double Quadrat::Umfang()  
{  
    return 4 * a;  
}  
  
std::string Quadrat::toStr()  
{  
    return "Quadrat mit Seitenlänge " + std::to_string(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_;
}

// d)
double Flaeche()
{
    return a * b;
}

double Umfang()
{
    return 2 * (a + b);
}

std::string toStr()
{
    return "Rechteck mit a=" + std::to_string(a) + " b=" + std::to_string(b);
}
};

// e)
#include "C2DPunkt.h"

// e)
void test_Figuren(TextBox^ tb)
{
    // noch unvollständig, am besten mit Unittests testen
    Kreis k1(10);
    tb->AppendText(rkl::to_String(k1.toStr()) + "\r\n");
    tb->AppendText("Umfang=" + k1.Umfang().ToString("f2"));
    tb->AppendText(" Flaeche=" + k1.Flache().ToString("f2") + "\r\n");
    Kreis* pk = new Kreis(10);
    tb->AppendText(rkl::to_String(pk->toStr()) + "\r\n");
    tb->AppendText("Umfang=" + pk->Umfang().ToString("f2"));
    tb->AppendText(" Flaeche=" + pk->Flache().ToString("f2") + "\r\n");

    Quadrat q(10);
    tb->AppendText(rkl::to_String(q.toStr()) + "\r\n");
    tb->AppendText("Umfang=" + q.Umfang().ToString());
    tb->AppendText(" Flaeche=" + q.Flache().ToString() + "\r\n");
    Quadrat* pq = new Quadrat(10);
    tb->AppendText(rkl::to_String(pq->toStr()) + "\r\n");
    tb->AppendText("Umfang=" + pq->Umfang().ToString());
    tb->AppendText(" Flaeche=" + pq->Flache().ToString() + "\r\n");

    Rechteck r(10, 20);
    tb->AppendText(rkl::to_String(r.toStr()) + "\r\n");
    tb->AppendText("Umfang=" + r.Umfang().ToString());
    tb->AppendText(" Flaeche=" + r.Flache().ToString() + "\r\n");
    Rechteck* pr = new Rechteck(7.5, 12.5);
    tb->AppendText(rkl::to_String(pr->toStr()) + "\r\n");
    tb->AppendText("Umfang=" + pr->Umfang().ToString());
    tb->AppendText(" Flaeche=" + pr->Flache().ToString() + "\r\n");
}

// ----- Aufgabe 4 -----

namespace N_Aufgabe_4 {
    void display(std::string s, int i = -1)
    {
        if (i >= 0) s = s + std::to_string(i);
        System::Diagnostics::Trace::WriteLine(rkl::to_String(s));
        // gibt s im Ausgabe-Fenster (Ansicht|Ausgabe) aus

        // rkl::StaticTextBox::WriteLine(rkl::to_String(s));
        rkl::StaticTextBox::WriteLine(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(): Wertparameter");
};

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(): Wertparameter
  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_Aufgabe4
```

```
// ----- Aufgabe 5 -----
```

```

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 test_MeinString(TextBox^ tb)
{
  MeinString s("abc");
  tb->AppendText((strlen(s.c_str())).ToString());
}

// ----- Aufgabe 6 -----

namespace N_Aufgabe_6 { // namespace, um Namenskonflikte
                        // mit Aufgabe 5 zu vermeiden
  class C {
    using T = int; // oder 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 Rückgabetyt 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.
     */
  }

// ----- Aufgabe 7 -----

// 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.

```

```
} // end of namespace N_Aufgabe_8
```

## 7.2 Aufgabe 9.1.11 Typ-Inferenz

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_OO_1.h

namespace N_Typ_Inferenz {
    using namespace System::Windows::Forms;
    // ----- Aufgabe 1 -----

    int fii(int x) { return 0; }

    void Typ_Inferenz_Aufgaben()
    {
        // 1.

        int i = 17;
        int* pi = &i;
        const int ci = 19;
        const int* cpi = &i;

        // auto    a;        // Fehler: Intialisierer fehlt
        auto    b = &i; // Datentyp von b: int*
        // b = 19;        // Fehler: Zuweisung von int an int*
        auto    c(i);    // Datentyp von c: int
        c = 20;        // das geht
        auto    d{ i }; // Datentyp von d: int
        d = 21;        // vor Visual Studio 2015 ging das nicht
        auto e = fii(0); // Datentyp von e: int
        auto f = fii;    // Datentyp von f: int(*) (int)
                        // f = 23;        // Fehler
        f(24);        // das geht
        auto g = new int(25); // Datentyp von g: int*
                        // g = 26;
        auto h = ci;    // Datentyp von g: int
        h = 27;        // das geht, h ist nicht const
        const auto j = i; // Datentyp von j: const int
                        // constexpr auto k = j; // Fehler: j ist keine Compilezeit-
Konstante

        // ----- Aufgabe 2 -----
        {
            using Meintyp = char; // char soll eventuell zu wchar_t geändert werden
            // typedef char Meintyp; // char soll eventuell zu wchar_t geändert werden
            Meintyp v1 = 'x';
            Meintyp v2;
        }
        { // Mit auto erspart man das typedef
            char v1 = 'x';
            auto v2 = v1;
        }
    } // end of namespace N_Typ_Inferenz

#ifdef SIMPLEUNITTESTS_H__
    // Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert

    void Unittests_Klassen()
    {
        rk1::Assert::Init("Unittests_Klassen");
        Unittests_C2DPunkt();
        rk1::Assert::Summary();
    }
#endif // #ifdef SIMPLEUNITTESTS_H__

} // end of namespace N_Loesungen_Klassen

#include <string>
#include <functional>
#include <vector>
```



```
namespace N_Loesungen_Klassen_als_Datentypen
{
    using System::Windows::Forms::TextBox;
```

### 7.3 Aufgabe 9.2.2 Elementinitialisierer

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_00_2.h
namespace N_Elementinit {
```

```
// ----- Aufgabe 1 -----
```

```
class E {
public:
    E()
    {
        System::Diagnostics::Trace::WriteLine("Standardkonstruktor");
    }

    E(int)
    {
        System::Diagnostics::Trace::WriteLine("int-Konstruktor");
    }
};
```

```
class C1 {
    E e1, e2;
public:
    C1() { }
    C1(int i) :e1(i) { }
    C1(int i, int j) :e1(i), e2(j) { }
};
```

```
void Konstrukturen()
{
    // Die folgenden Definitionen erzeugen die Meldungen:
    C1 c0; // Standardkonstruktor
    C1 c1(1); // Standardkonstruktor
    // int-Konstruktor
    C1 c2(1, 2); // Standardkonstruktor
    // int-Konstruktor
    // int-Konstruktor

    /* Wenn der Datentyp von E keine Klasse wäre, hätte
    - e1 und e2 in c0 einen undefinierten Wert
    - e2 in c1 einen undefinierten Wert und e1 den Wert 1
    - e1 und e2 in c2 die Werte 1 und 2
    */
}
```

```
// ----- Aufgabe 2 -----
```

```

// Siehe C2DPunkt.h

/*
Ersetze die Konstruktoren

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.h"
using N_Loesungen_Klassen::C2DPunkt; // aus Loesung_Kap_9_00_1.h
// 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;
    }

    std::string toStr()
    {
        return "Kreis mit Radius " + std::to_string(r) + " in " + pos.toStr();
    }
};

void testKreis(TextBox^ tb)
{
    C2DKreis k1(1, 2, 3);
    tb->AppendText(rk1::to_String(k1.toStr()));
    C2DKreis k2(4);
    tb->AppendText(rk1::to_String(k2.toStr()));
    C2DKreis k3;
    tb->AppendText(rk1::to_String(k3.toStr()));

    C2DPunkt p(6, 7);
    C2DKreis k5(5, p);
    tb->AppendText(rk1::to_String(k5.toStr()));
    C2DKreis k6(8, C2DPunkt(9, 10));
    tb->AppendText(rk1::to_String(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) { }

    std::string toStr()
    {
        return "Links oben: " + LinksOben.toStr();
    }

    // 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.

    // Wenn man die Elemente in einer Initialisiererliste in der Reihenfolge
    // ihrer Definition in der Klasse aufführt, ist die Gefahr eines solchen
    // Missverständnisses geringer.

    // Die fehlerhafte Initialisierung in der Klasse Rechteck1 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) { }

    std::string toStr()
    {
        return LinksOben.toStr();
    }
}

```

```
};

void Rechteck1_(TextBox^ tb)
{
    C2DPunkt mp(100, 100);
    Rechteck1 r1(mp, 10, 20);
    tb->AppendText(rk1::to_String(r1.toStr()));
    Rechteck2 r2(mp, 10, 20);
    tb->AppendText(rk1::to_String(r2.toStr()));
}
// ----- Aufgabe 5 -----

// 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.

} // end of namespace N_9_2_2 {
```

## 7.4 Aufgabe 9.2.4 Überladene Operatoren

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_00_2.h
namespace N_ueberladene_Operatoren {

    // ----- 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);
        std::string toStr() const
        {
            return std::to_string(z) + "/" + std::to_string(n);
        }

        bool equal(const Bruch& q)
        {
            return z * q.n == q.z * 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
    };

#ifdef SIMPLEUNITTESTS_H
    // Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
    void assertEquals(Bruch p, const Bruch& q)
    {
        {
            rkl::Assert::AreEqual(p.equal(q), true, "Bruch p=" + p.toStr() + " == q=" + q.toStr());
        }
    }
#endif
}

```

```
void Unittests_Bruch(int n)
{
    rkl::Assert::Init("Unittests_Bruch");

    Bruch p(1, n); // 1/n
    Bruch s(0, 1); // 0
    for (int i = 0; i < n; i++)
        s = s + p;
    assertEquals(s, Bruch(1, 1)); // n*(1/n)==1/1
    Bruch q(1, 2);
    Bruch d = s - q;
    assertEquals(d, q); // 1/1-1/2==1/2
    rkl::Assert::Summary();
}
#endif

} // end of namespace N_ueberladene_Operatoren
```

## 7.5 Aufgabe 9.2.6 Kopierkonstruktor und Zuweisungsoperator

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_00_2.h

namespace N_KopKonst_ZuwOp {

    // ----- Aufgabe 1 -----

    // 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.
    // Das gilt auch, wenn man
    //     smart pointer anstelle von Zeigern verwendet, oder
    //     Container-Klassen anstelle von selbstgestrickten Klassen mit
    //     dynamischem Speicher.

    // ----- Aufgabe 2 -----

    // a)
    namespace N_a
    {
        class MeinString {
            char* s; // Zeiger auf nullterminierten String
            int n; // Länge des Strings
        public:
            MeinString(const char* p) // 1
            {
                n = strlen(p);
                s = new char[n + 1];
                strcpy(s, p);
            };
            ~MeinString() { delete[] s; }
        };

        void test()
        {
            MeinString s1("abc"), s2("31234");
            s1 = s2; // Da MeinString keinen explizit definierten Zuweisungs-
                // operator hat, erzeugt der Compiler einen, der eine flache
                // Kopie durchführt. Deshalb werden bei dieser Zuweisung
                // nur die Zeiger kopiert. s1 und s2 zeigen dann auf dieselbe
                // Adresse.
        } // Beim Verlassen dieser Funktion werden die Destruktoren von s1 und
            // s2 aufgerufen. Da s2 dieselbe Adress hat wie s1, wird delete[] mit
            // dieser Adresse ein zweites Mal aufgerufen, was zu einem
            // Programmabbruch führt.
    } // end of namespace N_a

    // b), c)
    namespace N_b_c
```



```

{
class MeinString {
    char* s; // Zeiger auf nullterminierten String
    int n; // Länge des Strings
public:
    MeinString(const char* p) // 1
    {
        n = strlen(p);
        s = new char[n + 1];
        strcpy(s, p);
    };

    // b)
    // Der Fehler aus a) wird durch die Definition eines Zuweisungsoperators
    // behoben. Wenn Sie die Funktion test jetzt aufrufen, stürzt das Programm
    // nicht mehr ab.

    MeinString& operator=(const MeinString& x)
    {
        if (this != &x) // a
        {
            delete[] s; // b
            n = x.n; // 1
            s = new char[n + 1]; // 2
            strcpy(s, x.s); // 3
        }
        return *this; // c
    };

    // c)
    // Wenn man in test durch eine Anweisung wie
    // MeinString s3(s1);
    // noch den Kopierkonstruktor aufruft, erhält man denselben Fehler
    // wie in b). Dieser Fehler wird durch die Definition eines
    // Kopierkonstruktors behoben:

    MeinString(const MeinString& x) // Kopierkonstruktor
    {
        n = x.n; // 1
        s = new char[n + 1]; // 2
        strcpy(s, x.s); // 3
    }; //dieser Konstruktor stellt die Klasseninvariante her

    ~MeinString() { delete[] s; }
};

}; // end of namespace N_b_c

namespace N_Meinstring_mit_smart_pointer
{
    // c)
    // Einfacher als die Definition eines Zuweisungsoperators und Konstruktors
    // ist die Verwendung eines Smartpointers anstelle eines Zeigers.

    class MeinString {
        // std::shared_ptr<char> s = std::make_shared<char>();
        // char* s; // Zeiger auf nullterminierten String
        std::shared_ptr<char[]> s{ new char[1]{'\0'} };
        int n; // Länge des Strings
    public:
        MeinString(const char* p) // 1
        {
            n = strlen(p);
            s.reset(new char[n + 1]);
            strcpy(s.get(), p);
        };
        // ~MeinString() { delete[] s; } nicht notwendig
    };

    //

```

```
void test()
{
    MeinString s1("abc");
    MeinString s2(s1);
    s1 = s2;
} // kein Programmabsturz

} // end of namespace N_Meinstring_mit_smart_pointer

} // end of namespace N_KopKonst_ZuwOp

// ----- Aufgabe 3 -----

namespace N_Zuweisungsoperator_außerhalb_der_Klasse
{
    // Lösung: Wenn ein Zuweisungsoperator außerhalb
    // der Klasse definiert werden könnte, könnte eine
    // Zuweisung vor dieser Definition ein anderes
    // Ergebnis haben als eine Zuweisung nach dieser
    // Definition. Das wäre aber ziemlich verwirrend.

    class MeineKlasse {
        // Hier soll kein Zuweisungsoperator definiert sein.
        // Deshalb erzeugt ihn der Compiler selbst.
        // ...
    };

    MeineKlasse a, b;

    void Zuweisung1()
    {
        a = b; // Hier wird der vom Compiler erzeugte
              // Zuweisungsoperator verwendet, da der Compiler
    } // nicht weiß, dass später noch ein anderer definiert wird.

    /* Falls das möglich wäre:
    // Ein selbstdefinierter Zuweisungsoperator:
    MeineKlasse & operator=(MeineKlasse& lhs, const MeineKlasse& rhs)
    {
        // ...
    }
    */
    void Zuweisung2()
    {
        a = b; // Hier wird der selbstdefinierte
              // Zuweisungsoperator verwendet, der
    } // eventuell andere Anweisungen ausführt.
} // end of namespace N_Zuweisungsoperator_außerhalb_der_Klasse
```

## 7.6 Aufgabe 9.2.8 Konvertierende und explizite Konstruktoren

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_00_2.h

namespace N_explicit {
    class Kreis {
        double r;
    public:
        explicit Kreis(double Radius = 1) :r(Radius) {}
    };

    class Quadrat {
        double a;
    public:
        explicit Quadrat(double Seitenlaenge = 1) :a(Seitenlaenge) {};
    };
    void show(Kreis k) {}
    void show(Quadrat k) {}

    // Fügt man dem Quelltext die Klasse Quadrat ohne "explicit" hinzu,
    // ist der Aufruf show(1) von a) mehrdeutig.

    void Loesung()
    {
        Kreis k;
        k = Kreis(1);
        //
        show(k);
        // show(1); // mehrdeutig
        // a): beide werden kompiliert
        // b): Der Aufruf show(1) ist mehrdeutig. Nach
        //      dieser Erweiterung werden die bisherigen Anweisungen
        //      nicht mehr kompiliert.
        // c): nur show(k) wird kompiliert
        // d): Wie in c) geht show(1) nicht. Allerdings ist die
        //      Fehlermeldung präziser.

        Quadrat q;
        show(q);
    }
} // end of namespace N_explicit
```

## 7.7 Aufgabe 9.2.10 Statische Klasselemente

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_OO_2.h
```

```
// ----- Aufgabe 1 -----
```

```
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_ == nullptr)
        instance_ = new Singleton();
    return instance_;
};

Singleton* Singleton::instance_ = nullptr;

Singleton* p1 = Singleton::Instance();

void Aufg_Singleton(TextBox^ tb)
{
    Singleton* p2 = Singleton::Instance();
    p2->data = 19;
    tb->AppendText(p1->data.ToString() + "\r\n"); // p1->data=19
    tb->AppendText(p2->data.ToString() + "\r\n");
    // falls p1 ein gewöhnliches globales Objekt wäre, hätte die
    // globale Variable den Wert 17
}
```

## 7.8 Aufgabe 9.2.12 Konstante Objekte und Elementfunktionen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_OO_2.h
// ----- Aufgabe 1 -----
// a) b)
namespace N_constElementFkt {

    class Kreis {
    public:
        double r;
        Kreis(const double& Radius = 1)
        {
            r = Radius;
        }
        double Radius() const // <<-
        {
            return r;
        }
        std::string toStr() const // <<-
        {
            return "Kreis mit Radius " + std::to_string(Radius());
        }
    };

    void show1(Kreis k)
    {
        std::string s = k.toStr();
    }
    void show2(const Kreis& k)
    {
        std::string s = k.toStr(); // dieser Aufruf ist ohne
    } // const Elementfunktionen nicht möglich

    // c) const bei allen Elementfunktionen angeben, die direkt
    //     oder indirekt aufgerufen werden.

} // end of N_constElementFkt
```

## 7.9 Aufgabe 9.2.13 std::function

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_OO_2.h
// ----- Aufgabe 1 -----

namespace N_std_function {
    // using std::function;

    // globale Funktionen für Aufgabe a)
    void g1() { };
    int g2(int x, std::string y) { return x; }
    std::string g3(int x) { return "s:" + std::to_string(x); };

    class C {
    public:
        // statische Elementfunktionen für b)
        static void s1(void) {};
        static int s2(int x, std::string y) { return 10 * x; };
        static std::string s3(int x) { return "s: " + std::to_string(x); };

        // gewöhnliche Elementfunktionen für c)
        void m1(void) {};
        int m2(int x, std::string y) { return 100 * x; };
        std::string m3(int x) { return "s: " + std::to_string(x); };
    };

    void test_function_assign()
    {
        { // a) globale Funktion
            std::function<void(void)> glo = g1;
            std::function<void()> glo1 = g1;
            auto glo2 = g1;
            // Diese Funktionen können so aufgerufen werden:
            glo();
            glo1();
            glo2();

            std::function<int(int x, std::string y)> g2o = g2;
            int r1 = g2o(1, ""); // r1=

            std::function<std::string(int x)> g3o = g3;
            std::string s1 = g3o(1); // s1="";
        }
        { // b) statische Elementfunktion
            std::function<void(void)> slo = C::s1;
            std::function<int(int x, std::string y)> s2o = C::s2;
            std::function<std::string(int x)> s3o = C::s3;
        }
        { // c) nicht statische Elementfunktion
            C c;
            std::function<void(void)> m1o = std::bind(&C::m1, c);
            std::function<int(int x, std::string y)> m2o = std::bind(&C::m2, c, 1, "");
            std::function<std::string(int x)> m3o = std::bind(&C::m3, c, 1);
        }
        { // d) Funktionszeiger
            void(*fp1)();
            fp1 = g1;
            std::function<void(void)> flo = fp1;

            int(*fp2)(int x, std::string y);
            fp2 = g2;
            std::function<int(int x, std::string y)> f2o = fp2;

            std::string(*fp3)(int x);
            fp3 = g3;
            std::function<std::string(int x)> f3o = fp3;
        }
    }
}
```

```
// e)
void call1(std::function<void(void)> f1,
           std::function<int(int x, std::string y)> f2,
           std::function<std::string(int x)> f3)
{
    f1();
    int i2 = f2(1, "");
    std::string s3 = f3(1);
}

} // end of N_std_function
```

## 7.10 Aufgabe 9.2.15 Klassen und Header-Dateien

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_OO_2.h
namespace N_9_2_15 {

#include "Kreis.cpp"
#include "Quadrat.cpp"
#include "Rechteck.cpp"
    /* Im Projektmappen-Explorer den Quelldateien hinzufügen:
        // #include "Kreis.cpp"
        // #include "Quadrat.cpp"
        // #include "Rechteck.cpp"
    */
}
// Kreis.h

// Kreis.cpp

// Quadrat.h

// Quadrat.cpp

// Rechteck.h

// Rechteck.cpp

#ifdef SIMPLEUNITTESTS_H__
void Unittests()
{
    N_ueberladene_Operatoren::Unittests_Bruch(10);
}
#endif // #ifdef SIMPLEUNITTESTS_H__
} // end of namespace N_Loesungen_Klassen_als_Datentypen
```



## 7.11 Aufgabe 9.3.4 Konstruktoren, Destruktoren und implizite Konversionen

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_00_3.h
#include <string>
namespace N_Loesungen_Vererbung
{
    using namespace System::Windows::Forms;
    using namespace System;

    // ----- Aufgabe 1 -----

    class C {
    public:
        int i, j;
        static void display(String^ s)
        { // Da dem Standardkonstruktor und dem Destruktor keine TextBox
          // übergeben werden kann, erfolgt die Ausgabe entweder mit
          System::Diagnostics::Trace::WriteLine(s); // gibt s im Ausgabefenster
          // (Ansicht|Ausgabe) aus
          // oder mit
          // rk1::StaticTextBox::WriteLine(rk1::to_string(s)); // siehe 3.6.2
          // in eine TextBox, die vorher mit
          // rk1::StaticTextBox::Init(textBox1); // siehe 3.6.2
          // gesetzt wird.
        }
    };

    class D : public C {
    public:
        int k, a, b;
        C c;
        D(int x = 1) :c(x, 1), a(x), b(0), k(19)
        {
            display("Konstruktor-1 D");
        }
        D(int x, int y, int z) :C(x, y), a(1), b(2), c(x, y), k(z)
        {

```

```

        display("Konstruktor-2 D");
    }
    ~D()
    {
        display("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)
    {
        display("Konstruktor E");
    }
    ~E()
    {
        display("Destruktor E");
    }
};

void Button_9_3_4_1Click(TextBox^ tb)
{
    C::display("C c(1,2);\r\n");
    C c(1, 2);    // Konstruktor C
    C::display("D d(1,2,3);");
    D d(1, 2, 3); // Konstruktor C
                    // Konstruktor C
                    // Konstruktor-2 D
    C::display("E e(1,2);");

    E e(1, 2);    // Standardkonstruktor C
    // Konstruktor C
    // Konstruktor-1 D
    // Konstruktor C
    // Standardkonstruktor C
    // Konstruktor 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 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; }

    std::string toStr() const
    {
        return std::to_string(x);
    }

    void anzeigen(TextBox^ tb) const
    {

```

```

        tb->AppendText(rkl::to_String(toStr()) + "\r\n");
    }
};

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; }

    std::string toStr() const
    {
        return std::to_string(X()) + "|" + std::to_string(y);
    }

    void anzeigen(TextBox^ tb) const
    {
        tb->AppendText(rkl::to_String(toStr()) + "\r\n");
    }
};

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; }

    std::string toStr()const
    {
        return std::to_string(X()) + "|" + std::to_string(Y()) + "|" + std::to_string(z);
    }

    void anzeigen(TextBox^ tb) const
    {
        tb->AppendText(rkl::to_String(toStr()) + "\r\n");
    }
};

void Zeige_9_3_4_2(TextBox ^ tb)
{
    C1DPunkt p1(1);
    C2DPunkt p2(1, 2);
    C3DPunkt p3(1, 2, 3);
    p1.anzeigen(tb);
    p2.anzeigen(tb);
    p3.anzeigen(tb);
}

// ----- Aufgabe 3 -----

// a)
namespace N_a
{
    class Quadrat {
        double a;
    public:
        Quadrat(double a_) :a(a_) {}
    };

    class Rechteck :public Quadrat {
        double b;
    public:
        Rechteck(double a_, double b_) :Quadrat(a_), b(b_) {}
    };
}

// b)

```

```
namespace N_b
{
    class Rechteck {
        double a, b;
    public:
        Rechteck(double a_, double b_) :a(a_), b(b_) {};
    };

    class Quadrat :public Rechteck {
    public:
        Quadrat(double a_) :Rechteck(a_, a_) {}
    };
}

// c)
// Bei der Hierarchie in a) sind die geerbten Funktionen Umfang und
// Fläche in der abgeleiteten Klasse falsch. Bei b) sind sie richtig.

// d)
// Bei der Hierarchie b) benötigt jedes Quadrat doppelt soviel
// Speicherplatz wie bei der unter a).
```

## 7.12 Aufgabe 9.3.8 Konversionen zwischen Klassen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_00_3.h

// ----- Aufgabe 1 -----
/*
a) Ein Automobil hat einen Motor und Räder. Vermutlich wird nie jemand
sagen, dass ein Automobil ein Motor oder ein Rad ist.

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.

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.

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.
*/

// ----- Aufgabe 2 -----

/*
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.
*/
```

## 7.13 Aufgabe 9.3.9 Mehrfachvererbung

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_OO_3.h

namespace N_Mehrfachvererbung { // namespace to avoid name conflicts
    // ----- Aufgabe 1 -----
    class Kreis {
        double r;
        static const double pi;
    public:
        Kreis(double Radius)
        {
            r = Radius;
        }
        double Flaeche()
        {
            return r * r * pi;
        }
        double Umfang()
        {
            return 2 * r * pi;
        }
        std::string toStr() const
        {
            return "Kreis mit Radius " + std::to_string(r);
        }
    };
    const double Kreis::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);
        }
        std::string toStr() const
        {
            return std::to_string(x) + "|" + std::to_string(y);
        }
    };
};

// a)
namespace N_a {
    class C2DKreis :public Kreis, public C2DPunkt {
    public:
        C2DKreis(double Radius = 1, double x = 0, double y = 0) :
            Kreis(Radius), C2DPunkt(x, y) { }
        C2DKreis(double Radius, C2DPunkt pos) :
            Kreis(Radius), C2DPunkt(pos) { }
        // Eine in dieser Klasse definierte Funktion toStr verdeckt die Funktionen
        // toStr der Basisklasse:
    };

    void test()
    {
        C2DKreis k;
        // std::string s1 = k.toStr(); // Fehler: mehrdeutig
        // 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.
        //
        // 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 {
    Kreis k;
    C2DPunkt Position;
public:
    C2DKreis(double Radius = 1, double x = 0, double y = 0) :
        k(Radius), Position(x, y) { }
    C2DKreis(double Radius, C2DPunkt pos) :
        k(Radius), Position(pos) { }
    // Eine in dieser Klasse definierte Funktion toString verdeckt keine
    // Funktionen der Basisklasse:
    std::string toString() const
    {
        return k.toString() + " Pos: " + Position.toString();
    }
};
// Da hier keine Vererbung verwendet wird, erbt C2DKreis keine
// Elementfunktionen der Klasse Kreis und kann die Funktionen dieser
// Klasse Kreis auch nicht wiederverwenden.
//
// Diese Hierarchie ist nicht geeignet, da ein Kreis mit einer Position ein
// Kreis ist, aber die Funktionen der Klasse Kreis nicht verwendet.
} // end of namespace N_b

// c)
namespace N_c {
class C2DKreis :public 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) { }
};
// C2DKreis erbt alle public Elementfunktionen der Basisklasse, also
// Fläche, Umfang und toString. Alle diese Funktionen liefern auch über
// ein Objekt der Klasse C2DKreis richtige Ergebnisse. Die
// Elementfunktionen von C2DPunkt stehen dagegen nicht zur Verfügung.
//
// 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 6.2.2, 3.
    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
    std::string toString()
    {
        return "Kreis mit Radius " + std::to_string(r) + " in Position " +
Position.toString();
    }
};
// Wenn man in der Klasse C2DKreis die Funktionen Fläche, Abstand usw.
// für einen Kreis haben will, muss man sie erneut definieren. Diese
// Redundanz sollte man vermeiden.";

// Abgesehen davon ist diese Alternative geeignet.;
} // end of namespace N_d

void Button_9_3_9_1Click(TextBox^ tb)
{
    // a)
    tb->AppendText("a\r\n");
    N_a::C2DKreis ka(1, 2, 3);
    tb->AppendText(ka.Umfang().ToString() + "\r\n"); // das geht

```

```

tb->AppendText(ka.Abstand().ToString() + "\r\n"); // das geht
tb->AppendText(ka.Flaeche().ToString() + "\r\n"); // das geht
// tb->AppendText(ka.toStr()); // Mehrdeutig
// error: Mehrdeutiger Zugriff von "toStr".
// könnte "toStr" in Basis "N_6_3:N_3_11_1:Kreis" sein
// oder könnte "toStr" in Basis "N_6_3:N_3_11_1:C2DPunkt" sein
// b)
tb->AppendText("b)\r\n");
N_b::C2DKreis kb(1, 2, 3);
// tb->AppendText(kb.Umfang().ToString()+"\r\n"); // geht nicht
// tb->AppendText(kb.Abstand().ToString()+"\r\n"); // geht nicht
// tb->AppendText(kb.Flaeche().ToString()+"\r\n"); // geht nicht
tb->AppendText(rkl::to_String(kb.toStr()) + "\r\n"); // das geht
// c)
tb->AppendText("c)\r\n");
N_c::C2DKreis kc(1, 2, 3);
tb->AppendText(kc.Umfang().ToString() + "\r\n"); // das geht
// tb->AppendText(kc.Abstand().ToString()+"\r\n"); // geht nicht
tb->AppendText(kc.Flaeche().ToString() + "\r\n"); // das geht
tb->AppendText(rkl::to_String(kc.toStr()) + "\r\n"); // das geht
// d)
tb->AppendText("d)\r\n");
N_d::C2DKreis kd(1, 2, 3);
//tb->AppendText(kd.Umfang().ToString()+"\r\n"); // geht nicht
//tb->AppendText(kd.Abstand().ToString()+"\r\n"); // geht nicht
}

} // end of namespace N_Mehrfachvererbung

void Unittests()
{
}
} // end of namespace N_Loesungen_Vererbung

```



## 7.14 Aufgabe 9.4.3 Virtuelle Funktionen

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
```

```
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_9_OO_4.h
```

```
#include <string>
#include <cmath> // für fabs und sqrt
namespace N_Loesungen_Polymorphie
{
    using namespace System;
    using namespace System::Windows::Forms;

    void display(String^ s, int i = -1)
    { // Da dem Standardkonstruktor und dem Destruktor keine
      // TextBox übergeben werden kann, erfolgt die Ausgabe
      // entweder mit
      if (i >= 0) // Ohne int-Argument wird nur s ausgegeben
          s = s + i.ToString();
      System::Diagnostics::Trace::WriteLine(s); // gibt s im
      //          Ausgabefenster (Ansicht|Ausgabe) aus
      // oder mit
      rkl::StaticTextBox::WriteLine(rkl::to_string(s));
      // (siehe 3.6.2) in eine TextBox, die vorher mit
      // rkl::StaticTextBox::Init(textBox1); // siehe 3.6.2
      // gesetzt wird.
    }

    // ----- Aufgabe 1 -----
    namespace N_Aufgabe_1
    {
        class C
        {
            virtual void f0();
            virtual void f1() const;
            virtual void f2(int n = 0);
            virtual int f3(int);
            void f4();
            virtual void f5(const int n);
            virtual void f6(const int& n);
            virtual C& f7();
        };
        class D : public C
        {
            // virtual override void f0() ; // Fehler: override muss hinten stehen
            // virtual void f1() override; // Fehler: C::f0 wird wegen const
            // nicht überschrieben
        };
    };
};
```

```

// void f2() override; // C::f hat einen int-Parameter
int f3(int = 0) override;
// void f4() override; // Fehler: C::f4 ist nicht virtuell
void f5(int n) override; // int und const int werden als gleich betrachtet
// void f6(int& n) override; // int und int& werden nicht als gleich betrachtet
D& f7() override;
};
}

// ----- Aufgabe 2 -----

class C1DPunkt {
    double x;
public:
    C1DPunkt(double x_) :x(x_) { }
    void setzeX(double x_) { x = x_; }
    double X() const { return x; }

    virtual std::string toStr() const
    {
        return "(" + std::to_string(x) + ")";
    }

    // c)
    void anzeigen(TextBox^ tb) const
    {
        tb->AppendText(rkl::to_String(toStr()) + "\r\n");
    }
    // d)
    virtual 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; }

    std::string toStr() const override
    {
        return "(" + std::to_string(X()) + "|" + std::to_string(Y()) + ")";
    }

    // d)
    double length() const override
    {
        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; }

    std::string toStr() const override
    {
        return "(" + std::to_string(X()) + "|" + std::to_string(Y()) + "|" + std::to_string(z)
+ ")";
    }

    // d)
    double length() const override
    {
        return std::sqrt(X() * X() + Y() * Y() + z * z);
    }
};

```

```

    }
};

// b)
void show(TextBox^ tb, C1DPunkt& p)
{
    tb->AppendText(rkl::to_String(p.toStr()) + "\r\n");
};

void testVirtuelleFunktionen(TextBox^ tb)
{
    // a)
    tb->AppendText("toStr() \r\n");
    C1DPunkt* p1 = new C1DPunkt(1);
    tb->AppendText(rkl::to_String(p1->toStr()) + "\r\n");
    p1 = new C2DPunkt(2, 3);
    tb->AppendText(rkl::to_String(p1->toStr()) + "\r\n");
    p1 = new C3DPunkt(4, 5, 6);
    tb->AppendText(rkl::to_String(p1->toStr()) + "\r\n");

    // b)
    tb->AppendText("show \r\n");
    C1DPunkt q1(7);
    C2DPunkt q2(8, 9);
    C3DPunkt q3(10, 11, 12);
    show(tb, q1); // C1DPunkt::toStr
    show(tb, q2); // C2DPunkt::toStr
    show(tb, q3); // C3DPunkt::toStr

    // c)
    tb->AppendText("anzeigen\r\n");
    q1.anzeigen(tb); // C1DPunkt::toStr
    q2.anzeigen(tb); // C2DPunkt::toStr
    q3.anzeigen(tb); // C3DPunkt::toStr

    // d)
    tb->AppendText("length() \r\n");
    p1 = new C1DPunkt(1);
    double l1 = p1->length();
    tb->AppendText(rkl::to_String(std::to_string(l1)) + "\r\n");
    p1 = new C2DPunkt(3, 4);
    double l2 = p1->length(); // 5
    tb->AppendText(rkl::to_String(std::to_string(l2)) + "\r\n");
    p1 = new C3DPunkt(1, 2, 2);
    double l3 = p1->length(); // 3
    tb->AppendText(rkl::to_String(std::to_string(l3)) + "\r\n");
}

// ----- Aufgabe 3 -----

// "Diese Funktionen können nicht virtuell definiert werden, da alle ihre "
// "Parameterlisten verschieden sind.

// ----- Aufgabe 4 -----

namespace N_Aufgabe_3
{
    namespace N_vorher
    {
        class C {
        public:
            virtual std::string f(int x)
            {
                return "C::f";
            }
        };
        class D :public C
        {
        public:

```

```

    virtual std::string f(int x) // Da die Parameterlisten gleich sind,
    {                               // überschreibt D::f C::f
        return "D::f";
    }
};

void test(TextBox^ tb)
{
    C* pc = new D();
    tb->AppendText(rkl::to_String(pc->f(1)));
}
} // end of namespace N_vorher

namespace N_nachher
{
    class C {
    public:
        virtual std::string f(const int& x)
        {
            return "C::f";
        }
    };
    class D :public C
    {
    public:
        virtual std::string f(int x) // Jetzt sind die Parameterlisten nicht
        {                               // mehr gleich, weshalb D::f C::f nicht
            return "D::f";             // mehr überschreibt.
        }
    };

    void test(TextBox^ tb)
    {
        C* pc = new D();
        tb->AppendText(rkl::to_String(pc->f(1)));
    }
} // end of namespace N_nachher

void test(TextBox^ tb)
{
    N_vorher::test(tb); // D::f
    N_nachher::test(tb); // C::f
}

} // end of namespace N_Aufgabe_3

void test(TextBox^ tb)
{
    testVirtuelleFunktionen(tb);
}

} // end of namespace N_Loesungen_Polymorphie

```

## 8 Lösungen Kapitel 10. Namensbereiche

```
##### Copyright notice #####  
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook  
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",  
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_10_Namensber.h
```

## 8.1 Aufgabe 10.5 Namensbereiche

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_10_Namensber.h
// ----- Aufgabe 1. -----

namespace N_namespaces
{
    using namespace System;
    using namespace System::Windows::Forms;

    // a) b) c)
    namespace N {
        int f(int n)
        {
            return n + 1;
        }
    }

    // d)
    namespace N1 {

        class C
        {
            int g(int n);
            int h(int n);
            int i(int n);
        };

    } // end of namespace N1
}

// a)
void a()
{
    int r = N::f(1);
}

// b)
void b()
{
    using N::f;
    int r = f(2);
}

// c)
void c()
{
    using namespace N;
    int r = f(3);
}

// oder a), b), c)
void useNamespace_a_b_c()
{
    // a)
    int a = N::f(10);
    { // b)
        using N::f;
        int r = f(2);
    }
    { // c)
        using namespace N;
        int c = f(10);
    }
}

// e)
int N::N1::C::g(int n)
{
    return n + 1;
}
```

```
}

// f)
namespace A = N::N1;
int A::C::i(int x)
{
    return x + 1;
}

// ----- Aufgabe 2. -----

/*
Wenn man in einem Namensbereich eine #include-Anweisung mit einer
Header-Datei der Standard-Bibliothek aufnimmt, führt das meist zu
einer Vielzahl von Fehlermeldungen:

    namespace N_include_2 {
        #include<unordered_map>
    }

Wenn man diese Header-Datei zuvor (z.B. in einer vorherigen #include-
Datei) außerhalb eines Namensbereichs aufnimmt

    #pragma once
    #include<unordered_map>
    namespace N_include_1 {
    }

wird sie bei einem zweiten #include kein zweites Mal eingebunden.

*/

} // end of namespace N_namespaces

#include "include-1.h" // beide global, nicht in einem Namensbereich
#include "include-2.h"
```





## 9 Lösungen Kapitel 11. Exception-Handling

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_11_Exceptions.h
```

```
#include <vector>
#include <stdexcept> // für logic_error
#include <cmath>
```

```
namespace N_Loesungen_Exception_Handling
```

```
{
    using namespace System;
    using namespace System::Windows::Forms;
```

```
// ----- 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(TextBox^ tb, char c)
{
    tb->AppendText("f1 \r\n");
    if (c == 'b') throw 1;
    if (c == 'd') throw 1;
    if (c == 'e') throw 1;
};

void f2(TextBox^ tb, char c)
{
    tb->AppendText("f2 \r\n");
    if (c == 'c') throw 2;
};

void f3(TextBox^ tb, char c)
{
    tb->AppendText("f3 \r\n");
    if (c == 'd') throw 3;
};

void f4(TextBox^ tb, char c)
{
    tb->AppendText("f4 \r\n");
    if (c == 'e') throw 4;
};

void f5(TextBox^ tb, char c)
{
    tb->AppendText("f5 \r\n");
};

void f_(TextBox^ tb, Char c)
{
    tb->AppendText(c.ToString() + "\r\n");
    try {
        try {
            f1(tb, c);
            f2(tb, c);
        }
        catch (...)
        {
            f3(tb, c);
        }
        f4(tb, c);
    }
    catch (...)
    {
        f5(tb, c);
    }
}

void Aufgabe_1(TextBox^ tb)
{
    f_(tb, 'a');
    f_(tb, 'b');
    f_(tb, 'c');
    f_(tb, 'd');
    f_(tb, 'e');
}

// ----- Aufgabe 2 -----

void f1() {};
void f2() {};
void f3() {};
void f4() {};

void Aufgabe_2_a()
{
    try {
```

```
        f1();
        f2();
        f3();
        f4();
    }
    catch (...)
    {
    }
}

void Aufgabe_2_b()
{
    try {
        f1();
        f2();
    }
    catch (...)
    {
    }
    try {
        f3();
        f4();
    }
    catch (...)
    {
    }
}

void Aufgabe_2_c()
{
    try { f1(); }
    catch (...)
    {
    }
    try { f2(); }
    catch (...)
    {
    }
    try { f3(); }
    catch (...)
    {
    }
    try { f4(); }
    catch (...)
    {
    }
}

void Aufgabe_2()
{
    Aufgabe_2_a();
    Aufgabe_2_b();
    Aufgabe_2_c();
}

// ----- Aufgabe 3 -----

void Aufg3(int i)
{
    if (i == 0) throw i + 1;           // a)
    else if (i == 1) throw 1.0*i;     // b)
    else if (i == 2) throw std::exception(); // c)
    else throw std::logic_error("Vorbedingung nicht erfüllt"); // d)
}

void test_Aufg3(TextBox^ tb, int i)
{
    try {
        Aufg3(i);
    }
    catch (int i)
    {
    }
}
```

```

    tb->AppendText("int Exception i=" + i.ToString() + "\r\n");
}
catch (double d)
{
    tb->AppendText("double Exception d=" + d.ToString() + "\r\n");
}
catch (std::logic_error& e)
{
    tb->AppendText("Exception logic_error what=" + rkl::to_String(e.what()) + "\r\n");
}
catch (std::exception& e)
{
    tb->AppendText("Exception what=" + rkl::to_String(e.what()) + "\r\n");
}
};

void Aufgabe_3(TextBox^ tb)
{
    test_Aufg3(tb, 0);
    test_Aufg3(tb, 1);
    test_Aufg3(tb, 2);
    test_Aufg3(tb, 3);
}

// ----- Aufgabe 4 -----

void f(int i)
{ // #include <stdexcept>
  if (i == 0) throw std::exception();
  else if (i == 1) throw std::logic_error("logic error");
  else if (i == 2) throw std::range_error("range error");
  else if (i == 3) throw std::out_of_range("out of range error");
  else throw std::exception();
}

// a)
void g1(TextBox^ tb, int i)
{
    try { f(i); }
    catch (std::logic_error& e)
    {
        tb->AppendText("logisch \r\n");
    }
    catch (std::out_of_range& e)
    {
        tb->AppendText("range \r\n");
    }
    catch (std::exception& e)
    {
        tb->AppendText("exception \r\n");
    }
};

void test_Aufg4a(TextBox^ tb)
{
    tb->AppendText("3 a) \r\n");
    g1(tb, 0); // exception
    g1(tb, 1); // logisch
    g1(tb, 2); // exception
    g1(tb, 3); // logisch
}

// b)
void g2(TextBox^ tb, int i)
{
    try { f(i); }
    catch (std::logic_error& e)
    {
        tb->AppendText(rkl::to_String(e.what()) + "\r\n");
    }
}

```

```
    catch (std::out_of_range& e)
    {
        tb->AppendText(rkl::to_String(e.what()) + "\r\n");
    }
    catch (std::exception& e)
    {
        tb->AppendText(rkl::to_String(e.what()) + "\r\n");
    }
};

void test_Aufg4b(TextBox^ tb)
{
    tb->AppendText("4 b) \r\n");
    g2(tb, 0); // exception::what(), "no named exception thrown"
    g2(tb, 1); // logic_error::what(), "logic error"
    g2(tb, 2); // range_error::what(), "range error"
    g2(tb, 3); // out_of_range::what(), "out of range error"
                // Dieses Ergebnis erhält man einfacher wie in d)
}

// c)
void g3(TextBox^ tb, int i)
{
    try { f(i); }
    catch (std::exception e)
    {
        tb->AppendText(rkl::to_String(e.what()) + "\r\n");
    }
};

void test_Aufg4c(TextBox^ tb)
{
    tb->AppendText("4 c) \r\n");
    g3(tb, 0); // exception::what(), "no named exception thrown"
    g3(tb, 1); // exception::what(), "no named exception thrown"
    g3(tb, 2); // exception::what(), "no named exception thrown"
    g3(tb, 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(TextBox^ tb, int i)
{
    try { f(i); }
    catch (std::exception& e)
    {
        tb->AppendText(rkl::to_String(e.what()) + "\r\n");
    }
    catch (...)
    {
        tb->AppendText("irgendeine Exception");
    }
};

void test_Aufg4d(TextBox^ tb)
{
    tb->AppendText("3 d) \r\n");
    g4(tb, 0); // exception::what(), "no named exception thrown"
    g4(tb, 1); // logic_error::what(), "logic error"
    g4(tb, 2); // range_error::what(), "range error"
    g4(tb, 3); // out_of_range::what(), "out of range error"
}

void Aufgabe_4(TextBox^ tb)
{
    test_Aufg4a(tb);
    test_Aufg4b(tb);
    test_Aufg4c(tb);
    test_Aufg4d(tb);
}
```

```
// ----- Aufgabe 5 -----

class MeineException_1 : public std::exception
{
public:
    MeineException_1(std::string msg) noexcept :message1(msg) {}
    MeineException_1(const char* msg) noexcept :message2(msg) {}
private:
    std::string message1;
    const char* message2;
}; // Bei dieser Version gibt what den leeren String der Basisklasse aus
void test_MeineException_1(TextBox^ textBox1)
{ // analog für MeineException_2 und MeineException_3
    try {
        throw MeineException_1("MeineException_1");
    }
    catch (MeineException_1& e) // eigene abgeleitete Klasse
    {
        textBox1->AppendText(rkl::to_String(e.what())); // leerer String
    }
    catch (std::exception& e) // C++ Standardbibliothek
    {
        textBox1->AppendText(rkl::to_String(e.what()));
    }
    catch (Exception^ e) // C++/CLR .NET
    {
        textBox1->AppendText(e->Message);
    }
    catch (...) // alle weiteren Exceptions
    {
        textBox1->AppendText("Was war das?");
    }
}

class MeineException_2 : public std::exception
{
public:
    MeineException_2(std::string msg) noexcept :message1(msg) {}
    MeineException_2(const char* msg) noexcept :message2(msg) {}
    const char* what() const noexcept override {
        return message1.c_str(); // Variante 1
        return message2; // Variante 2
    };
private:
    std::string message1; // Variante 1
    const char* message2; // Variante 2
};
// Die Daten aus der Basisklasse exception werden nicht genutzt

class MeineException_3 : public std::runtime_error
{
public:
    // MeineException_3(std::string msg) noexcept :message1(msg) {}
    // MeineException_3(const char* msg) noexcept :message2(msg) {}
    // Fehler: std::runtime_error hat keinen Standardkonstruktor
private:
    std::string message1;
    const char* message2;
};

// ----- Aufgabe 6 -----

// Siehe auch Betrand Meyer, Object Oriented Software Construction,
// Abschnitt 12.2

class ENegative {};
```

```

double Sqrt(double d)
{
    try {
        if (d < 0) throw ENegative();
        return std::sqrt(d); // #include <cmath>
    }
    catch (...)
    {
        MessageBox::Show("negativ");
        return 0;
    }
}

/*
Diese Funktion hat 3 Schwächen:
a) statt catch(...) sollte man catch(ENegative) verwenden

b) sie kann einfacher formuliert werden:
*/

double Sqrt1(double d)
{
    if (d < 0)
    {
        MessageBox::Show("negativ");
        return 0;
    }
    else
        return std::sqrt(d);
}

/*
c) Die Klasse ENegative sollte von std::exception abgeleitet werden und
die üblichen Anforderungen erfüllen:
*/

class negative_exception :public std::exception
{ // #include <exception>
    std::string str;
public:
    negative_exception(const std::string& msg) :str(msg) { };
    const char* what() const throw() override { return str.c_str(); }
};

/*
d) Auch mit den Varianten a), b) und c) ist diese Funktion nicht sehr gut, da
der Anwender mit einer Meldung konfrontiert wird, mit der er vermutlich
nicht viel anfangen kann. Es wäre besser, in dieser Funktion nur eine
Exception auszulösen und diese in der aufrufenden Funktion zu behandeln:
*/

double Sqrt2(double d)
{
    if (d < 0)
        throw negative_exception("Bitte geben Sie einen positven Wert ein");
    return std::sqrt(d);
}

void useSqrt2(double d)
{
    try {
        // ...
        double y = Sqrt2(d);
        // ...
    }
    catch (const negative_exception& e)
    {
        MessageBox::Show(rkl::to_String(e.what()));
    }
}

```

```

// ----- Aufgabe 7 -----

void h1()
{ // kann zu einem Speicherleck führen
  std::vector<int*> v = { new int(17) };
  // f();
  for (auto i = v.begin(); i != v.end(); i++)
    delete *i;
}

// Ersetzt man die Zeiger in h1 durch Objekte mit RAII, wird bei
// einer Exception in f() der Destruktor dieser Objekte
// aufgerufen, der den Speicher freigibt.

// Solche Objekt werden am besten mit smart pointern definiert.
// Alternativ:

class C {
  std::vector<int*> v;
public:
  void push_back(int* p)
  {
    v.push_back(p);
  }
  ~C()
  {
    for (auto i = v.begin(); i != v.end(); i++)
      delete *i;
  }
};

void h2()
{
  C v;
  v.push_back(new int(17));
  // f();
}

void Aufgabe_7()
{
  h1();
  h2();
}

// ----- Aufgabe 8 -----

namespace N_Aufgabe8 {
  // https://www.securecoding.cert.org/confluence/display/cplusplus/ERR60-
  // CPP.+Exception+objects+must+be+nothrow+copy+constructible

  // a)
  class C {
  int* a;
  int* b;
  void init() noexcept(false) // Vor Visual Studio 2015 ohne noexcept
  {
    // ...
  }

public:
  C() : a(nullptr), b(nullptr)
  {
    a = new int();
    b = new int();
    init();
  }

  ~C()

```



```
        {
            delete a;
            delete b;
        }
};

// b) Überarbeiten Sie die Klasse C so, dass auch dann, wenn init
//     eine Exception auslöst, kein Speicherleck auftritt.

class C_Lsg { // Im Kapitel Smart-Pointer wird noch eine elegantere
    int* a; // Lösung vorgestellt
    int* b;
protected:
    void init() noexcept(false); // Vor Visual Studio 2015 ohne noexcept
public:
    C_Lsg()
    {
        try {
            a = new int();
            b = new int();
            init();
        }
        catch (...)
        {
            delete a;
            delete b;
            a = nullptr;
            b = nullptr;
            throw;
        }
    }
};

} // end of namespace N_Aufgabe8

} // end of namespace N_Loesungen_Exception_Handling
```



## 10 Lösungen Kapitel 12. Containerklassen der Standardbibliothek

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_12_Container.h
```

```
#include <fstream>
#include <string>
#include <algorithm>
#include <map>
#include <set>
#include <fstream>
#include <sstream>
#include <vector>
#include <random>
```

```
#include "FileNames.h"
```

```
namespace N_Loesungen_Container {
    using namespace System::Windows::Forms;
```

## 10.1 Aufgabe 12.1.5 Containerklassen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_12_Container.h

// ----- Aufgabe 1 -----

void Alg(TextBox^ tb)
{
    // a)
    std::array<int, 10> a = { 9,8,7,6,5,4,3,2,1,0 }; // oder int a[10] =
    std::sort(a.begin(), a.end());
    { // oder mit einem C-Array
        int a[10] = { 9,8,7,6,5,4,3,2,1,0 };
        std::sort(a, a + 10);
    }
    for (int i = 0; i < 10; i++)
        tb->AppendText(a[i].ToString() + "\r\n");

    // b)
    std::array<int, 10> b; // oder int b[10];
    std::copy(a.begin(), a.end(), b.begin());
    for (int i = 0; i < 10; i++)
        tb->AppendText(b[i].ToString() + "\r\n");

    // c)
    if (std::equal(a.begin(), a.end(), b.begin()))
        tb->AppendText("gleich \r\n");
    else
        tb->AppendText("ungleich \r\n");
    b[1] = 17;
    if (std::equal(a.begin(), a.end(), b.begin()))
        tb->AppendText("gleich \r\n");
    else
        tb->AppendText("ungleich \r\n");
}

// ----- Aufgabe 2 -----

void GeprüfteIteratoren(TextBox^ tb, int test)
{
    std::vector<int> v;
    std::vector<int>::iterator it;

    std::vector<int> v1;
    std::vector<int> v3(10);

    // a)
    if (test == 1)
        *it = 17;

    // b)
    else if (test == 2)
    {
        it = v.begin();
        *it;
    }

    // c)
    else if (test == 3)
    {
        it = v.begin();
        v.push_back(17);
        *it;
    }

    // d)
    else if (test == 4)
        *v.end() = 17;
}
```

```
// e)
else if (test == 5)
    copy(v1.begin(), v.end(), v3.begin());

// f)
else if (test == 6)
    copy(v.end(), v.begin(), v3.begin());

else
    MessageBox::Show("invalid testcase");
}
```

## 10.2 Aufgabe 12.1.6 Speicherverwaltung bei Vektoren

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_12_Container.h
// ----- Aufgabe 1 -----

void ShowCapacity(TextBox^ tb, int n)
{
    std::vector<int> v;
    std::vector<int>::size_type cap_neu, cap_alt = v.capacity();
    for (int i = 0; i < n; i++)
    {
        v.push_back(i);
        cap_neu = v.capacity();
        if (cap_neu != cap_alt)
            tb->AppendText("v cap: " + v.capacity().ToString() + " " + i.ToString() + "\r\n");
        cap_alt = cap_neu;
    }
    v.clear();
    tb->AppendText("nach erase: " + v.capacity().ToString() + "\r\n");
}
```

## 10.3 Aufgabe 12.1.7 Mehrdimensionale Vektoren

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_12_Container.h
// ----- Aufgabe 1 -----

void PascalDreieck(TextBox^ tb, int n)
{
    std::vector<std::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++)
    {
        std::string s = "";
        for (int j = 0; j <= i; j++)
            s = s + std::to_string(p[i][j]) + " ";
        tb->AppendText(rkl::to_String(s) + "\r\n");
    };
}
```

## 10.4 Aufgabe 12.2.3 Assoziative Container

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_12_Container.h

// ----- Aufgabe 1 -----

using ValueType = int; // wie typedef int ValueType;
using KeyType = int; // wie typedef int KeyType;
using InfoSys = std::map<KeyType, ValueType>;
// wie 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 InfoSystClick(TextBox^ tb, KeyType x)
{
    ValueType data;
    bool found = ValueToKey(x, data);
    if (found)
        tb->AppendText("gefunden: " + data + "\r\n");
    else
        tb->AppendText("nicht gefunden!\r\n");
    /*
    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;
}

#ifdef SIMPLEUNITTESTS_H
void Unittests_ValueToKey(int n)
{
    fillContainer(n);
    if (n > 0)
    {

```



```

    int data;
    bool found = ValueToKey(-1, data);
    rkl::Assert::AreEqual(found, false, "test_ValueToKey: " + std::to_string(-1));
    found = ValueToKey(0, data);
    rkl::Assert::AreEqual(found, true, "test_ValueToKey: " + std::to_string(0));
    found = ValueToKey(n - 1, data);
    rkl::Assert::AreEqual(found, true, "test_ValueToKey: " + std::to_string(n - 1));
    found = ValueToKey(n, data);
    rkl::Assert::AreEqual(found, false, "test_ValueToKey: " + std::to_string(n));
    found = ValueToKey(n + 1, data);
    rkl::Assert::AreEqual(found, false, "test_ValueToKey: " + std::to_string(n + 1));
}
}
#endif // #ifdef SIMPLEUNITTESTS_H__

// ----- Aufgabe 2 -----
std::set<int> randSet;
std::mt19937 gen1;
std::uniform_int<int> rnd(1, 100); // #include <random>

int NewRand(unsigned int Max)
{
    if (randSet.size() < Max)
    {
        unsigned int Anzahl_vorher = randSet.size(), r;
        while (randSet.size() == Anzahl_vorher) // Endlosschleife, falls
            // s.size()>=Max
            r = rnd(gen1); // oder r = rand() % Max;
        randSet.insert(r);
    }
    return r;
}
else return -1; // oder Exception auslösen
}

void NewRandClick(TextBox ^ tb)
{
    int n = 10;
    for (int i = 0; i < n + 5; i++)
        tb->AppendText(NewRand(n) + "\r\n");
}

// ----- Aufgabe 3 -----

using Woerterbuch = std::set<std::string>;
// wie typedef std::set<std::string> Woerterbuch;

Woerterbuch WoerterbuchLesen(const std::string & ifn)
{
    Woerterbuch dict;
    std::vector<std::string> c;
    std::ifstream fin(ifn);
    if (fin)
    {
        std::string z;
        while (getline(fin, z))
        {
            c = N_Loesungen_Arrays::tokenize(z);
            dict.insert(c.begin(), c.end());
        }
        if (!fin.eof())
            MessageBox::Show(rkl::to_String("read error: " + ifn));
    }
    else
        MessageBox::Show(rkl::to_String("open error: " + ifn));
    fin.close(); // überflüssig, aber kein Fehler
    return dict;
}

```

```

void TextPruefen(TextBox ^ tb, const std::string & ifn, const Woerterbuch & dict)
{
    std::vector<std::string> c;
    std::ifstream fin(ifn);
    if (fin)
    {
        std::string z;
        while (getline(fin, z))
        {
            c = N_Loesungen_Arrays::tokenize(z);
            for (auto i = c.begin(); i != c.end(); i++)
            {
                if (dict.find(*i) == dict.end())
                { // nicht im Wörterbuch enthaltene Wörter ausgeben:
                    tb->AppendText(rkl::to_String(z + ": " + *i + "\r\n"));
                }
            }
        }
        if (!fin.eof())
            MessageBox::Show(rkl::to_String("read error: " + ifn));
    }
    else
        MessageBox::Show(rkl::to_String("open error: " + ifn));
    fin.close(); // überflüssig, aber kein Fehler
}

void RechtschrPruefung(TextBox^ tb)
{
    Woerterbuch dict =
        WoerterbuchLesen(N_Dateinamen::TestTextfile); // faust.txt
    tb->AppendText("Anzahl der Wörterbucheinträge: "
        + dict.size().ToString() + "\r\n");
    TextPruefen(tb, N_Dateinamen::Faust_veraendert, dict);
    // In Faust_veraendert wurden gegenüber faust.txt einige Worte verändert
}

// ----- Aufgabe 4 -----

using MMType = std::multimap<std::string, int>;
// wie typedef std::multimap<std::string, int> MMType;

MMType MakeXRef(std::vector<std::string> v) // Ein XRef-Generator in 13 Zeilen
{ // erzeugt eine Crossreference-Liste aus den Strings eines Vektors
    MMType mm;
    int line = 1;
    for (auto i = v.begin(); i != v.end(); i++)
    {
        std::vector<std::string> c = N_Loesungen_Arrays::tokenize(*i);
        for (auto token = c.begin(); token != c.end(); token++)
            mm.insert(make_pair(*token, line));
        line++;
    }
    return mm;
}

MMType MakeXRef(std::string ifn) // fn: Dateiname
{ // erzeugt eine Crossreference-Liste aus den Wörtern der Zeilen einer Datei
    MMType mm;
    std::ifstream fin(ifn);
    if (fin)
    {
        std::string z;
        int line = 1;
        while (getline(fin, z))
        {
            std::vector<std::string> c = N_Loesungen_Arrays::tokenize(z);
            for (auto token = c.begin(); token != c.end(); token++)
                mm.insert(make_pair(*token, line));
            line++;
        }
    }
}

```

```

    }
    if (!fin.eof())
        MessageBox::Show(rkl::to_String("read error: " + ifn));
    }
    else
        MessageBox::Show(rkl::to_String("open error: " + ifn));
return mm;
}

void PrintXRef(TextBox^ tb, MType mm, char* fn = 0)
{ // Schreibt eine XRef-Liste in die TextBox oder in
  // die Datei mit dem Namen fn
  std::ofstream of;
  if (fn != 0) of.open(fn);
  for (auto i = mm.begin(); i != mm.end(); )
  {
    std::ostringstream out;
    auto first = mm.lower_bound(i->first),
        last = mm.upper_bound(i->first);
    out << i->first << ": ";
    for (auto 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 tb->AppendText(rkl::to_String(out.str()));
  }
}

void XRefClick(TextBox^ tb)
{
  std::vector<std::string> v = {
    "Alle meine Entchen",
    "schwimmen auf dem See,"
    "schwimmen auf dem See," };

  MType mm = MakeXRef(v);
  PrintXRef(tb, mm);
  mm.clear();

  mm = MakeXRef(N_Dateinamen::TestCppfile);
  PrintXRef(tb, mm);
}

#ifdef SIMPLEUNITTESTS_H__
void Unittests_AssCont()
{
  rkl::Assert::Init("Unittests_AssCont");
  Unittests_ValueToKey(1);
  Unittests_ValueToKey(100);
  rkl::Assert::Summary();
}
#endif // #ifdef SIMPLEUNITTESTS_H__

} // end of namespace N_Loesungen_Container

```



## 11 Lösungen Kapitel 13. Dateibearbeitung mit den Streamklassen

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_13_FileStreams.h
```

```
#include <fstream>
#include <string>
```

```
#include "FileNames.h"
```

```
namespace N_Loesungen_Streamklassen
{
    using namespace System;
    using namespace System::Windows::Forms;
```

## 11.1 Aufgabe 13.3 Binärdateien

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_13_FileStreams.h

// ----- Aufgabe 1 a) -----

int countBytes(std::string fn)
{
    char c;
    int n = 0;
    std::ifstream f(fn, std::ios::binary);
    if (!f)
        MessageBox::Show(rkl::to_String("open error countBytes " + fn));
    while (f.read(&c, sizeof(c)))
        n++;
    if (!f.eof())
        MessageBox::Show(rkl::to_String("read error countBytes " + fn));
    f.close();
    return n;
}

int countBytesTxt(std::string fn)
{
    char c;
    int n = 0;
    std::ifstream f(fn); // bis auf ios::binary identisch mit countBytes
    if (!f)
        MessageBox::Show(rkl::to_String("open error countBytesTxt " + fn));
    while (f.read(&c, sizeof(c)))
        n++;
    if (!f.eof())
        MessageBox::Show(rkl::to_String("read error countBytesTxt " + fn));
    f.close();
    return n;
}

int countBytes(std::string fn, bool binary)
{
    char c;
    int n = 0;
    std::ifstream f;
    if (binary) f.open(fn, std::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)
        MessageBox::Show(rkl::to_String("open error countBytes fn=" + fn));
    while (f.read(&c, sizeof(c)))
        n++;
    if (!f.eof())
        MessageBox::Show(rkl::to_String(("read error countBytes fn=") + fn));
    f.close();
    return n;
}

void createTestfile(std::string fn, int n)
{
    std::ofstream fout(fn);
    char a[] = { '1', '2', '3', 26, '5', '6' };
    fout.write(a, n);
    fout.close();
}

#ifdef SIMPLEUNITTESTS_H
void Unittests_countBytes()
{

```

```

const std::string TestDir = N_Dateinamen::TestDir;
createTestfile(TestDir + "0.dat", 0);
createTestfile(TestDir + "1.dat", 1);
createTestfile(TestDir + "6.dat", 6);

int n0 = countBytes(TestDir + "0.dat", false);

rk1::Assert::AreEqual(n0, 0, "countBytes 0");
int n0b = countBytes(TestDir + "0.dat", true);
rk1::Assert::AreEqual(n0b, 0, "countBytes 0 b");

int n1 = countBytes(TestDir + "1.dat", false);
rk1::Assert::AreEqual(n1, 1, "countBytes 1");
int n1b = countBytes(TestDir + "1.dat", true);
rk1::Assert::AreEqual(n1b, 1, "countBytes 1 b");

int n2 = countBytes(TestDir + "6.dat", false);
rk1::Assert::AreEqual(n2, 3, "countBytes 1");
int n2b = countBytes(TestDir + "6.dat", true);
rk1::Assert::AreEqual(n2b, 6, "countBytes 1 b");
}
#endif // #ifdef SIMPLEUNITTESTS_H__

void CountBytesClick(TextBox^ tb)
{
    const std::string TestTextfile = N_Dateinamen::TestTextfile;
    const std::string TestExefile = N_Dateinamen::TestExefile;
    // entweder
    int n = countBytes(TestTextfile);
    tb->AppendText(rk1::to_String(TestTextfile) + " (binary mode): " + n.ToString() + "
bytes\r\n");
    n = countBytesTxt(TestTextfile);
    tb->AppendText(rk1::to_String(TestTextfile) + " (text mode): " + n.ToString() + "
bytes\r\n");

    n = countBytes(TestExefile);
    tb->AppendText(rk1::to_String(TestExefile) + " (binary mode): " + n.ToString() + "
bytes\r\n");
    n = countBytesTxt(TestExefile);
    tb->AppendText(rk1::to_String(TestExefile) + " (text mode): " + n.ToString() + "
bytes\r\n");

    // oder
    n = countBytes(TestTextfile, true);
    tb->AppendText(rk1::to_String(TestTextfile) + " (binary): " + n.ToString() + "
bytes\r\n");
    n = countBytes(TestTextfile, false);
    tb->AppendText(rk1::to_String(N_Dateinamen::TestTextfile) + ": " + n.ToString() + "
bytes\r\n");

    n = countBytes(TestExefile, true);
    tb->AppendText(rk1::to_String(TestExefile) + " (binary): " + n.ToString() + "
bytes\r\n");
    n = countBytes(TestExefile, false);
    tb->AppendText(rk1::to_String(TestExefile) + ": " + n.ToString() + " bytes\r\n");
    // Die Ergebnisse stimmen nur für Dateien, die im Binärmodus geöffnet
    // wurden.
}

// ----- Aufgabe 1 b) -----

void CopyFile(const std::string& src, const std::string& dst)
{
    std::ifstream fin(src, std::ios::binary);
    if (!fin)
        MessageBox::Show(rk1::to_String("open error CopyFile src " + src));
    std::ofstream fout(dst, std::ios::binary);
    if (!fout)
        MessageBox::Show(rk1::to_String("open error CopyFile dst " + dst));
}

```

```

char c;
while (fin.read(&c, sizeof(c)))
{
    fout.write(&c, sizeof(c));
    if (!fout)
        MessageBox::Show("write error CopyFile ");
}
if (!fin.eof())
    MessageBox::Show(rk1::to_String("read error CopyFile src ") + src));
fin.close();
fout.close();
}

const std::string CopyTextDest = N_Dateinamen::TestDir + "Ctest.txt";
const std::string CopyExeDest = N_Dateinamen::TestDir + "Ctest.exe";

void test_CopyFile()
{
    CopyFile(N_Dateinamen::TestTextfile, CopyTextDest);
    CopyFile(N_Dateinamen::TestExefile, CopyExeDest);
}

// ----- Aufgabe 1 c) -----

bool CompareFiles(const std::string fn1, const std::string fn2)
{
    std::string msg = "open error ";

    std::ifstream f1(fn1, std::ios::binary);
    if (!f1)
        MessageBox::Show(rk1::to_String("open error CompareFiles fn1" + fn1));

    std::ifstream f2(fn2, std::ios::binary);
    if (!f2)
        MessageBox::Show(rk1::to_String("open error CompareFiles fn2" + 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())
        MessageBox::Show(rk1::to_String("read error CompareFiles fn1=" + fn1));
    if (!f2 && !f2.eof())
        MessageBox::Show(rk1::to_String("read error CompareFiles fn2" + fn2));

    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);
    */
}

#ifdef SIMPLEUNITTESTS_H__
// Dieses Makro wird durch ein #include "SimpleUnitTests.h" definiert
void Unittests_copy_and_compare_files()
{
    test_CopyFile();
}

```



```
bool b = CompareFiles(N_Dateinamen::TestTextfile, CopyTextDest);
rk1::Assert::AreEqual(b, true, "CompareFiles 1");
b = CompareFiles(N_Dateinamen::TestExefile, CopyExeDest);
rk1::Assert::AreEqual(b, true, "CompareFiles 2");
b = CompareFiles(N_Dateinamen::TestTextfile, CopyExeDest);
rk1::Assert::AreEqual(b, false, "CompareFiles 3");
b = CompareFiles(N_Dateinamen::TestExefile, CopyTextDest);
rk1::Assert::AreEqual(b, false, "CompareFiles 4");
}

void Unittests_Files()
{
    rk1::Assert::Init("Unittest_Files");
    Unittests_countBytes();
    Unittests_copy_and_compare_files();
    rk1::Assert::Summary();
}
#endif // #ifndef SIMPLEUNITTESTS_H__
```

## 11.2 Aufgabe 13.4 Textdateien

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_13_FileStreams.h

// ----- Aufgabe 1 -----

void CsvTabelleAnlegen(std::string out_fn)
{
    std::ofstream fout(out_fn);
    if (fout)
    {
        fout << "n ; n*n ; n*n*n " << std::endl;
        for (int i = 5; i <= 10; i++)
            fout << i << ";" << i * i << ";" << i * i * i << ";" << std::endl;
        fout.close(); // überflüssig
    }
    else
        MessageBox::Show(rkl::to_String("open error " + out_fn));
}

// ----- Aufgabe 2 -----

void PrintTextlines(std::ostream& fout, std::string in_fn)
{
    std::ifstream fin(in_fn);
    if (fin)
    {
        std::string s;
        while (getline(fin, s))
        {
            for (std::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;");
            }
            if (fout << "<BR>" << s << std::endl)
                MessageBox::Show(rkl::to_String("write error "));
        }
    }
    else
        MessageBox::Show(rkl::to_String("open error " + in_fn));
}

void TextToHTML(std::string ifn, std::string htmlfn)
{
    std::ofstream fout(htmlfn);
    fout << "<HTML>" << std::endl
        << "<HEAD>" << std::endl
        << "<TITLE>"
        << ifn
        << "</TITLE>" << std::endl
        << "</HEAD>" << std::endl
        << "<BODY>" << std::endl;
    PrintTextlines(fout, ifn);
    fout << "</BODY>" << std::endl;
    fout << "</HTML>" << std::endl;
    fout.close(); // überflüssig
}

} // end of namespace N_Loesungen_Streamklassen
// Lösungen zu "Richard Kaiser: C++ mit Microsoft Visual C++ 2008", ISBN: 978-3-540-23869-0
// Copyright Richard Kaiser, 2009. http://www.rkaiser.de. All rights reserved.
```

// Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie einen Hinweis auf die Quelle (die ersten beiden Zeilen) enthalten. In keinem Fall wird eine Haftung für direkte, indirekte, zufällige oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

// It is permitted to use this source text in programs of your own, provided that it contains a reference to the source (the first two lines above). All liabilities for use of the code are disclaimed.

```
#pragma once

#include <string>
#include <filesystem>
#include <vector>
namespace N_Dateinamen // Die im Folgenden verwendeten Dateinamen
{
    // Die verwendeten Dateien vorher noch in diese Verzeichnisse kopieren
    const std::string TestDir = "c:\\VS2022-CppBuch-Kaiser-WinForms-Loesungen\\Testdateien\\";
    const std::string TestHTMLfile = TestDir + "Faust.html";
    const std::string TestExefile = TestDir + "MeinCpp2022Projekt.exe";
    const std::string TestCppfile = TestDir + "Form1.h";
    const std::string TestTextfile = TestDir + "Faust.txt";
    const std::string Faust_veraendert = TestDir + "Faust_veraendert.txt";

    /* werden die überhaupt benötigt?
    const std::string TestKBfile = TestDir + "kb.dat";
    const std::string TestKBHTMLfile = TestDir + "kb.html";
    const std::string TestCppfile = TestDir + "AssCont.h";
    */

    namespace N_prüfe_ob_Dateien_vorhanden_sind
    {
        bool FileExists(std::string fn)
        {
            std::filesystem::path p(fn);
            bool result = std::filesystem::exists(p);
        /*
            if (!result)
            {
                // alt, .NET: if (!System::IO::File::Exists(fn))
                // System::Windows::Forms::MessageBox("Datei fehlt");
            }
        */
            return result;
        }
    }

    void checkRequiredFiles(System::Windows::Forms::TextBox^ tb)
    { // prüfe, ob die für einige Beispiele und Aufgaben notwendigen
      // Dateien vorhanden sind
      std::vector<std::string> req = { TestHTMLfile, TestExefile,
        /* TestKBfile, TestKBHTMLfile, */
        TestCppfile, TestTextfile, Faust_veraendert };
      tb->AppendText(rkl::to_String(" check required files")+"\r\n");
      for (auto& i :req)
      {
          std::string message;
          if (FileExists(i))
          {
              message = "'" + i + "' existiert";
          }
          else
          {
              message = " --> '" + i + "' fehlt!";
          }
          tb->AppendText(rkl::to_String(message) + "\r\n");
      }
      /*
      bool error = false;
      if (FileExists(TestExefile))
          error = true;
      */
    }

```

```
    if (FileExists(TestTextfile))
        error = true;
    // if (FileExists(TestKBfile))
    //     error = true;
    // if (FileExists(Faust_veraendert))
    //     error = true;
    if (FileExists(TestCppfile))
        error = true;
    */
}

} // end of namespace N_prüfe_ob_Dateien_vorhanden_sind
} // end of namespace N_Dateinamen
```

## 12 Lösungen Kapitel 14. Funktoren, Funktionsobjekte und Lambda-Ausdrücke

```
##### Copyright notice #####  
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook  
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",  
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\loesung_kap_14_funktoren.h
```

```
#include <string>  
#include <iostream>  
#include <ctime>  
#include <vector>  
#include <algorithm>  
#include <functional>
```

```
namespace N_Loesungen_Funktoren {  
    using namespace System;  
    using namespace System::Windows::Forms;
```

## 12.1 Aufgabe 14.2 Prädikate und Vergleichsfunktionen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\loesung_kap_14_funktoren.h

// ----- Aufgabe 1 -----

// a)
bool absteigend(int a, int b) // Eine Funktion mit zwei Parametern des
{                               // Datentyps der Elemente des Containers
    return a > b;
}

struct Absteigend {
    bool operator()(int a, int b)
    {
        return a > b;
    }
};

void zeige_Container(TextBox^ tb, std::vector<int> v)
{
    for (auto& i : v) // ab Visual Studio 2015, sonst gewöhnliche for-Schleife
        tb->AppendText(i.ToString() + " ");
    tb->AppendText("\r\n");
}

void Sortiere_1_a()
{
    std::vector<int> v = { 3,1,5,1,7 };

    // a)
    // mit einer Funktion:
    std::sort(v.begin(), v.end(), absteigend); // v = { 7, 5, 3, 1, 1 }
    // mit einem Funktor:
    std::sort(v.begin(), v.end(), Absteigend());
    // mit einem vordefinierten Funktor:
    std::sort(v.begin(), v.end(), std::greater<int>());

    // greater_equal geht nicht:
    // std::sort(v.begin(), v.end(), std::greater_equal<int>()); // Debug-Fehler
}

// b)
struct Absteigend_s {
    bool operator()(std::string a, std::string b)
    {
        return a > b;
    }
};

bool absteigend_s(std::string a, std::string b)
{ // eine überladene Funktion für int-Parameter geht nicht
    return a > b;
}

void Sortiere_1_b(TextBox^ tb)
{
    tb->AppendText("Sortiere_1_b \r\n");
    std::vector<int> v = { 3,1,5,1,7 };

    std::vector<std::string> vs = { "de","abc","def" };
    std::sort(vs.begin(), vs.end()); // aufsteigend
    std::sort(vs.begin(), vs.end(), std::greater<std::string>());
    std::sort(vs.begin(), vs.end(), absteigend_s);
    std::sort(vs.begin(), vs.end(), Absteigend_s());

    for (auto& i : v)
    {
        tb->AppendText(Convert::ToString(i) + " ");
    }
}
```

```

}

// c)
struct FileInfo {
    std::string Name; // Name der Datei
    long long Length; // Dateigröße
};

// e)
bool nach_Groesse_aufsteigend(const FileInfo & a, const FileInfo & b)
{
    return a.Length < b.Length;
}

// f)
struct nach_Groesse_Absteigend
{ // alternativ Funktor
    bool operator()(FileInfo a, FileInfo b)
    {
        return a.Length > b.Length;
    }
};

// g)
bool nach_Name_aufsteigend(const FileInfo & a, const FileInfo & b)
{
    return a.Name < b.Name;
}

// h)
bool nach_Name_absteigend(const FileInfo & a, const FileInfo & b)
{
    return a.Name < b.Name;
}

void sortiere_Dateien()
{
    std::vector<FileInfo> vf = { { "config.sys",100 }, { "Windows.h",1000 }, { "small.h",10
} };
    // e)
    std::sort(vf.begin(), vf.end(), nach_Groesse_aufsteigend);
    // f)
    std::sort(vf.begin(), vf.end(), nach_Groesse_Absteigend());
    // g)
    std::sort(vf.begin(), vf.end(), nach_Name_aufsteigend);
    // h)
    std::sort(vf.begin(), vf.end(), nach_Name_absteigend);
    int br = 1;
}

// ----- Aufgabe 2 -----

bool Contains(const std::string & src, const std::string & pattern)
{
    bool result = src.find(pattern) != std::string::npos;
    return result;
}

bool sortiereNachFehlerart(const std::string & s1, const std::string & s2)
{
    if (Contains(s1, "Fehler") != Contains(s2, "Fehler"))
        return Contains(s1, "Fehler");
    if (Contains(s1, "Warnung") != Contains(s2, "Warnung"))
        return Contains(s1, "Warnung");
    if (Contains(s1, "Info") != Contains(s2, "Info"))
        return Contains(s1, "Fehler");
    return false;
}

void sortiere_Meldungen()
{

```

```
std::vector<std::string> Meldungen = {
    "2.12.2021, 2.15: Fehler 0815: Festplatte voll",
    "2.2.2022, 2.18: Warnung: LAN-Kabel abgezogen",
    "2.2.2022, 2.19: Info: Keine Internetverbindung",
    "2.2.2022, 3.18: Info: LAN-Verbindung neu aufgebaut",
    "3.4.2022, 20.25: Fehler 0816: Kein Strom",
    "12.6.2022, 3.32: Fehler 0715: CPU zu heiß",
    "12.6.2022, 4.08: Fehler 0716: CPU zu kalt"
};
std::sort(Meldungen.begin(), Meldungen.end(), sortiereNachFehlerart);
}
```



## 12.2 Aufgabe 14.4 Lambda-Ausdrücke

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\loesung_kap_14_funktoren.h
// ----- Aufgabe 1 -----

void f(std::function<int(int)>) { };

void A()
{
    // a)
    std::function<void(void)> f1 = []() {};
    std::function<int(double x, std::string y)> f2 =
        [](double x, std::string y) {return x; };
    std::function<double(int x)> f3 = [](int x) { return 0.0; };
    std::function<std::string(int x)> f4 = [](int x) { return ""; };
    // Aufrufe:
    f1();
    int r2 = f2(3.1, "");
    double r3 = f3(1);
    std::string r4 = f4(1);

    // b)

    // f([](int x) { return x + 1}); // ; fehlt vor }
    f([](int x) { return x + 1; });
    // f([](int x) { return x + 1; }); // ; nach }
    // f([]() { return 3.14; }); // Parameter fehlt
    // f([] x{ return x + 1 }); // x
    // f([]()->int { return 3.14; });
    // f([]->int { return 3.14; });
    // f([](int x) return x + 1; // Klammern um return fehlen
}

// ----- Aufgabe 2 -----

void sortiere_mit_Lambdas(std::vector<FileInfo> vf)
{
    // e)
    std::sort(vf.begin(), vf.end(), [](const FileInfo& a, const FileInfo& b)
        { return a.Length < b.Length; }); // nach_Groesse_aufsteigend

    // f)
    std::sort(vf.begin(), vf.end(), [](const FileInfo& a, const FileInfo& b)
        { return a.Length > b.Length; }); // nach_Groesse_absteigend

    // g)
    std::sort(vf.begin(), vf.end(), [](const FileInfo& a, const FileInfo& b)
        { return a.Name < b.Name; }); // nach_Name_aufsteigend

    // h)
    std::sort(vf.begin(), vf.end(), [](const FileInfo& a, const FileInfo& b)
        { return a.Name < b.Name; }); // nach_Name_absteigend
}

void test(TextBox ^ tb)
{
    Sortiere_1_a();
    Sortiere_1_b(tb);
    sortiere_Dateien();
    sortiere_Meldungen();
    std::vector<FileInfo> vf = { {"",100}, {"",10}, {"",1000} };
    sortiere_mit_Lambdas(vf);
    A();
}

} // end of namespace N_Loesungen_Funktoren
```



## 13 Lösungen Kapitel 15. Templates

```
##### Copyright notice #####  
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook  
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",  
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\loesung_kap_15_templates.h
```

```
#include <string>  
#include <sstream>  
#include <fstream>  
#include <stdexcept> // für range_error  
#include <vector>  
#include <algorithm>  
#include <set>  
#include <map>  
#include <iterator>  
#include <utility>  
#include <tuple>  
#include <functional>  
#include "FileNames.h"
```

## 13.1 Aufgabe 15.1 Funktions-Templates

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\loesung_kap_15_templates.h

namespace N_Loesungen_Funktions_Templates {
    using namespace System::Windows::Forms;

    // ----- 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 std::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 test_Aufgabe_1(TextBox^ tb)
    {
        tb->AppendText("Aufgabe 1 \r\n");

        int a[5] = { 5,4,6,2,1 };
        AuswahlSort(a, 5);
        for (int i = 0; i < 5; i++)
            tb->AppendText(String::Format("int a[i]={0} \r\n", 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++)
            tb->AppendText(String::Format("int* a[i]={0} \r\n", *p[i]));

        const char* n[5] = { "15","14","16","02","01" };
        AuswahlSort<const char*>(n, 5);
        for (int i = 0; i < 5; i++)
            tb->AppendText(String::Format("char* a[i]={0} \r\n", rkl::to_String(n[i])));
    }

    // ----- Aufgabe 2 -----

    template <class T>
    T max1(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)
{
    // ...
}

struct Print {
    int operator()(int x) {};
    // ...
};

// a)
// Der Compiler bestimmt zuerst den Datentyp der Template-Argumente
// und verwendet diesen dann, um das Template zu erzeugen.
// Um Namenskonflikte zu vermeiden, werden die von max erzeugten
// Funktionen max2, max3 usw. genannt.
int max2(const int& a, const int& b)
{
    return a < b ? b : a;
}

// b)
double max3(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 std::function<void(int)> F;
using F = std::function<void(int)>;
// oder
typedef void(*F1) (int);
// oder
using F2 = std::function<void(int)>;
// oder
using F3 = void(*) (int);

F for_each(int* first, int* last, F f)
{
    while (first != last) f(*first++);
    return f;
}

// e)
F for_each2(std::vector<int>::iterator first, std::vector<int>::iterator last, F f)
{
    while (first != last) f(*first++);
    return f;
}

// f)
Print for_each3(std::vector<int>::iterator first, std::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;
}

// ----- Aufgabe 3 -----
struct Bruch {
    int z, n; // z: Zähler, n: Nenner
};

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;
}

namespace N_HP_STL {
    // 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 Visual Studio so:

    namespace rel_ops
    { // nested namespace to hide relational operators from std
        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>

// im Namensbereich rel_ops zur Verfügung.
} // end of namespace N_HP_STL{

void test_Aufgabe_3()
{
    /*
#define _SILENCE_CXX20_REL_OPS_DEPRECATION_WARNING 1
    // The namespace std::rel_ops and its contents are deprecated in C++20.
    // Their use is superseded by C++20's <=> operator and automatic rewrites
    // of relational expressions.

    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;
#undef _SILENCE_CXX20_REL_OPS_DEPRECATION_WARNING
*/
}

// ----- Aufgabe 4 -----

namespace N_variadic_template {

} // end of namespace N_variadic_template

} // namespace N_Loesungen_FuncTempl
```

## 13.2 Aufgabe 15.2 Klassen-Templates

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\loesung_kap_15_templates.h
namespace N_Klassen_Templates {

    /*
    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
    //
    */

    using namespace System::Windows::Forms;

    const std::string TestDir = N_Dateinamen::TestDir;

    // ----- Aufgabe 1 -----

    // In der Boost-Bibliothek (http://boost.org, boost/array.hpp)
    // ist das fixed size array "Array" im Wesentlichen
```



```

// folgendermaßen definiert (stark vereinfacht):

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
    using value_type = T;           // oder typedef T value_type;
    using reference = T&;          // oder typedef T& reference;
    using const_reference = const T&; // oder typedef const T& const_reference;
    using size_type = std::size_t; // oder typedef std::size_t size_type;

    // operator[]
    reference operator[](size_type i)
    {
        if (i >= N) // bzw. (i<0 || i>=N) bei int i;
        {
            std::string msg = "Array index out of range: " +
                std::to_string(i) + " > " + std::to_string(N - 1);
            throw std::out_of_range(msg);
        }
        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::string msg = "Array index out of range: " +
                std::to_string(i) + " > " + std::to_string(N - 1);
            throw std::out_of_range(msg);
        }
        return elems[i];
    }

    // size is constant
    static size_type size() { return N; }
};

void test_Aufgabe_1(TextBox^ tb)
{
    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];
    tb->AppendText(String::Format("s={0} \r\n", 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];
    tb->AppendText(String::Format("s={0} \r\n", s));
    try {
        a[n] = 0; // löst eine Exception aus
    }
    catch (std::exception& e)
    {
        tb->AppendText(String::Format("a[i]= " + rkl::to_String(e.what()) + "\r\n"));
    }
}

```

```

}
try {
    a[-1] = 0; // löst eine Exception aus
}
catch (std::exception& e)
{
    tb->AppendText(String::Format("a[i]= " + rk1::to_String(e.what()) + "\r\n"));
}
}

```

```
// ----- Aufgabe 2 -----
```

```

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 test_Aufgabe_3()
{
    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;
}

```

```
// ----- Aufgabe 3 -----
```

```

// 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&);
    // Der private Zuweisungsoperator verhindert Zuweisungen
    // In C++11 würde man das mit "=delete" machen.
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 test_Aufgabe_scoped_ptr()
{
    scoped_ptr<int> sp1(new int);
    *sp1 = 17;
    int* p2 = new int(18);
    scoped_ptr<int> sp2(p2);
    // sp1=sp2; // nicht möglich
}

// ----- Aufgabe 4 -----

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 {
    using T_float = T; // wie typedef T T_float;
};

template<> struct float_trait<int> {
    using T_float = double; // wie typedef double T_float;
};

template<> struct float_trait<char> {
    using T_float = double; // wie 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 test_Aufgabe_Average(TextBox^ tb)
{
    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 a5 = average(d1, Max);
    double a6 = average(d2, Max);

    tb->AppendText(String::Format("int d1[Max]={1,2}: Average={0}\r\n", a1));
}

```

```
tb->AppendText(String::Format("double d2 [Max]={1,2}: Average={0}\r\n", a2));  
tb->AppendText(String::Format("int d1 [Max]={1,2}: average={0}\r\n", a5));  
tb->AppendText(String::Format("double d2 [Max]={1,2}: average={0}\r\n", a6));  
}
```

### 13.3 Aufgabe 15.4.3 type traits und static\_assert

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\loesung_kap_15_templates.h
// Indem man in die Datei eine Compilezeit-Konstante aufnimmt
class MyLib {
public: const static int version = 1;
};

// und diese dann in der Datei prüft, die diese verwendet:

void test_static_assert()
{
    static_assert(MyLib::version > 0, "MyLib::version > 0 required");
}

} // end of namespace N_ClassTempl {
```

## 13.4 Aufgabe 15.5 Variadische Templates

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\loesung_kap_15_templates.h
namespace N_variadische_Templates {
    using namespace System::Windows::Forms;

    template<typename T>
    auto Summel(T v)
    {
        return v;
    }

    template<typename T, typename... Args>
    auto Summel(T first, Args... args)
    {
        return first + Summel(args...);
    }

    auto Summe2()
    {
        return 0;
    }

    /* oder
    template<typename T> // damit geht dann auch s2b
    auto Summe2()
    {
        return T{};
    }
    */
    template<typename T, typename... Args>
    auto Summe2(T first, Args... args)
    {
        return first + Summe2(args...);
    }

    void test_Aufgaben()
    {
        // a)
        auto s1a = Summel(1, 2, 3.0); // Lösung: s1a=6.0
        // b)
        // auto s1b = Summel("1", "2", "3"); // Fehler: + nicht für char* definiert
        // c)
        std::string s1 = "1", s2 = "2", s3 = "3";
        auto s1c = Summel(s1, s2, s3); // s1c="abc"

        // d)
        // auto s1d = Summel(); // Fehler: kann nicht ohne Argument aufgerufen werden

        // e)
        auto s2a = Summe2(1, 2, 3.0); // Lösung: s2a=6.0

        // f)
        // auto s2b = Summe2(s1, s2, s3); // Fehler: Keine Abbruchbedingung für string
        // g)
        auto s2c = Summe2(); // int s2c=0;
    }
} // end of namespace N_Typ_Inferenz
```

## 14 Lösungen Kapitel 16. STL-Algorithmen

```
##### Copyright notice #####  
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook  
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",  
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/  
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_16_STL_Algorithmen.h
```

```
#include <string>  
#include <fstream>  
#include <stdexcept> // für range_error  
#include <vector>  
#include <algorithm>  
#include <set>  
#include <map>  
#include <iterator>  
#include <utility>  
#include <tuple>  
#include <functional>  
#include <valarray>  
#include <complex>  
#include <cmath>  
#include "Filenames.h"
```

## 14.1 Aufgabe 16.1 Iteratoren

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_16_STL_Algorithmen.h
namespace N_Iteratoren
{
    using namespace System::Windows::Forms;

    // ----- Aufgabe 1 -----

    // a)
    template<typename T, typename Iterator>
    void writeToFile(Iterator first, Iterator last, std::string fn)
    {
        std::ofstream f(fn);
        std::copy(first, last, std::ostream_iterator<T>(f, "\n"));
    }

    // b)
    template<typename T>
    void copyFile(std::string fn1, std::string fn2)
    { // nur als Übungsaufgabe, Filesystem bietet bessere Möglichkeiten
        std::ifstream f1(fn1);
        std::ofstream f2(fn2);
        std::copy(std::istream_iterator<T>(f1),
            std::istream_iterator<T>(), std::ostream_iterator<T>(f2, "\n"));
    }

    // c)
    template<typename T>
    void sortFile(std::string fn1, std::string fn2)
    {
        std::ifstream f1(fn1);
        std::vector<T> v;
        std::copy(std::istream_iterator<T>(f1),
            std::istream_iterator<T>(), std::back_inserter(v));
        std::sort(v.begin(), v.end());
        std::ofstream f2(fn2);
        std::copy(v.begin(), v.end(), std::ostream_iterator<T>(f2, "\n"));
    }

    // d)
    template<typename T>
    void showFile(TextBox^ tb, std::string fn)
    {
        tb->AppendText("Dateiname:" + rk1::to_String(fn) + "\r\n");
        std::ifstream f(fn);
        T x;
        while (f >> x)
        {
            tb->AppendText(rk1::to_String(std::to_string(x) + "\r\n"));
        }
    }

    // e)
    struct Bruch {
        Bruch() :z(0), n(1) {}
        Bruch(int z_, int n_) :z(z_), n(n_) {}
        int z, n; // z: Zähler, n: Nenner
    };

    std::ostream& operator<<(std::ostream& f, const Bruch& b)
    {
        return f << b.z << "|" << b.n;
    }

    std::istream& operator >>(std::istream& f, Bruch& b)

```



```

{
    char Bruchstrich;
    f >> b.z >> Bruchstrich >> b.n;
    return f;
}

void Test_a_bis_d(TextBox^ tb)
{
    const std::string TestDir = N_Dateinamen::TestDir;
    // a), e)
    int as[3] = { 1,2,3 }; // sortiert
    writeToFile<int>(as, as + 3, TestDir + "s.txt");
    int au[3] = { 1,3,2 }; // unsortiert
    writeToFile<int>(au, au + 3, TestDir + "u.txt");
    int a0[1];
    writeToFile<int>(a0, a0, TestDir + "0.txt"); // 0 Elemente
    int a1[1] = { 1 };
    writeToFile<int>(a1, a1 + 1, TestDir + "1.txt"); // 1 Element

    // b), e)
    copyFile<int>(TestDir + "s.txt", TestDir + "sc.txt");
    copyFile<int>(TestDir + "u.txt", TestDir + "uc.txt");
    copyFile<int>(TestDir + "0.txt", TestDir + "0c.txt");
    copyFile<int>(TestDir + "1.txt", TestDir + "1c.txt");

    // c), e)
    sortFile<int>(TestDir + "s.txt", TestDir + "s_sor.txt");
    sortFile<int>(TestDir + "u.txt", TestDir + "u_sor.txt");
    sortFile<int>(TestDir + "0.txt", TestDir + "0_sor.txt");
    sortFile<int>(TestDir + "1.txt", TestDir + "1_sor.txt");

    // d)
    showFile<int>(tb, TestDir + "u.txt");
    showFile<int>(tb, TestDir + "0.txt");
    showFile<int>(tb, TestDir + "1.txt");

    showFile<int>(tb, TestDir + "s_sor.txt");
    showFile<int>(tb, TestDir + "u_sor.txt");
    showFile<int>(tb, TestDir + "0_sor.txt");
    showFile<int>(tb, TestDir + "1_sor.txt");

    // f)
    std::vector<Bruch> vb = { Bruch(1, 2), Bruch(1, 3), Bruch(1, 4) };
    writeToFile<N_Iteratoren::Bruch>(vb.begin(), vb.end(), TestDir + "b.txt");
}

// ----- Aufgabe 2 -----

/* Zur Lösung dieser Aufgabe müssen der Klasse Array
lediglich diese Definitionen 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
using value_type = T;
using reference = T&;
using const_reference = const T&;
using size_type = std::size_t;

// 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:

using iterator = T*; // 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.
using const_iterator = const T* ;// typedef const T* const_iterator;
const_iterator begin() const { return elems; }
const_iterator end() const { return elems + N; }
};

void test_Array()
{
    Array<int, 5> a = { 1,9,-1, 6,2 };
    std::sort(a.begin(), a.end());
    {
        const int n = 100;
        Array<int, n> a;
        for (int i = 0; i < n; i++) a[i] = n - i;
        std::sort(a.begin(), a.end());
    }
}

// ----- Aufgabe 3 -----

```

```
void Vektor_mit_Konstruktor_füllen()
{
    std::ifstream f("s.txt");
    std::istream_iterator<int> beg1(f);
    std::istream_iterator<int> end1;
    std::vector<int> v1(beg1, end1);

    std::ifstream fb("b.txt");
    std::istream_iterator<Bruch> beg2(fb);
    std::istream_iterator<Bruch> end2;
    std::vector<Bruch> vb(beg2, end2);
}
} // end of namespace N_Iteratoren
```

## 14.2 Aufgabe 16.8 STL-Algorithmen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_16_STL_Algorithmen.h
namespace N_STL_Algorithmen
{
    using namespace System::Windows::Forms;

    const std::string TestDir = N_Dateinamen::TestDir;

    namespace Meine_STL
    {
        //----- Aufgabe 1 -----
        // Die Algorithmen der STL sind mit dem hier auskommentierten
        // und durch int ersetzten difference_type etwas universeller.

        template <class InputIterator, class T>
        // typename iterator_traits<InputIterator>::difference_type
        int count(InputIterator first, InputIterator last, const T& value)
        {
            // typename iterator_traits<InputIterator>::difference_type
            int n = 0;
            for (; first != last; ++first)
                if (*first == value)
                    ++n;
            return n;
        }

        template <class InputIterator, class Predicate>
        // typename iterator_traits<InputIterator>::difference_type
        int count_if(InputIterator first, InputIterator last, Predicate pred)
        {
            // typename iterator_traits<InputIterator>::difference_type
            int 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;
}

} // end of namespace Meine_STL

void Aufgabe_1()
{
    std::vector<int> v = { 1,5,2,4,4 };
    std::set<double> s = { 1,5,2,4,4 };
    int i = Meine_STL::count(v.begin(), v.end(), 1);
    int j = Meine_STL::count_if(v.begin(), v.end(), [](int i) {return i % 2 == 1; });
    int k = *Meine_STL::min_element(v.begin(), v.end());
}

//----- Aufgabe 2 -----

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()
{
    std::vector<double> v1 = { 1, 2, 4, 4, 4, 5, 15 };
    double md1 = Median(v1); // 4

    std::vector<int> v2{ 5, 6, 4, 3, 2, 6, 7, 9, 3 };
    double md2 = Median(v2); // 5
}
```

### 14.3 Aufgabe 16.9.2 Valarrays

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_16_STL_Algorithmen.h

void erzeugeTestdaten(std::valarray<double>& x, std::valarray<double>& y,
    int n, double a_, double b_)
{
    // Aufgabe 2 a): rnd=0;
    // Aufgabe 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);
    }
}

// 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 1. 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 1. 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;
}
```

## 14.4 Aufgabe 16.9.4 Komplexe Zahlen

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_16_STL_Algorithmen.h

String^ ComplexToStr(std::complex<double> c)
{
    return real(c).ToString() + "+" + imag(c).ToString() + "i";
}

void QuadGlClick(TextBox^ tb)
{
    std::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);
    tb->AppendText(ComplexToStr(x1) + "\r\n");
    tb->AppendText(ComplexToStr(x2) + "\r\n");
    // Probe:
    tb->AppendText(ComplexToStr(a * x1 * x1 + b * x1 + c) + "\r\n");
    tb->AppendText(ComplexToStr(a * x2 * x2 + b * x2 + c) + "\r\n");
}

// ----- Aufgabe 2 -----

void EinheitswurzelnClick(TextBox^ tb)
{
    const int n = 100;

    const double pi = 3.14159265358979323846;

    std::complex<double> w(std::cos(2 * pi / n), std::sin(2 * pi / n));
    for (int i = 0; i < n; i++)
    {
        std::complex<double> z = pow(w, i); // Einheitswurzel
        // z^n muss etwa 1 sein:
        tb->AppendText(ComplexToStr(pow(z, n)) + "\r\n");
    }
}

} // end of namespace N_STL_Algorithmen
```





## 15 Lösungen Kapitel 16. Multithreading

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_18_MultiThreading.h
```

```
#include <thread>
#include <atomic>
#include <shared_mutex>
#include <chrono>
#include <string>
```

```
namespace N_Loesungen_Multithreading
{
    using namespace System;
    using namespace System::Windows::Forms;

    using std::thread;
    using std::string;
```

## 15.1 Aufgabe 16.1.3 Funktionen als Threads starten

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_18_MultiThreading.h

/*
- f1: eine globale Funktion mit einem Wertparameter
- f2: eine globale Funktion mit drei Wertparametern
- f3: eine globale Funktion mit einem Referenzparameter
- f4: eine globale Funktion mit einem konstanten Referenzparameter
- f5: eine gewöhnliche Elementfunktionen mit einem Wertparameter
- f6: eine statische Elementfunktion mit einem Wertparameter
*/

std::string f1(int x) { return std::to_string(x); }
int f2(int x, std::string y, int z) { return x + std::stoi(y) + z; }
void f3(int& i) { i = i + 1; }
int f4(const std::string& s) { return s.length(); }

class C {
public:
    int f5(int x) { return x + 1; }
    static int f6(int x) { return x + 1; }
};

void Aufgabe_1()
{
    string s = "17";
    C c;

    { // a)
        int i = 0;
        thread t1(f1, 1);
        thread t2(f2, 1, s, 2);
        thread t3(f3, std::ref(i));
        thread t4(f4, std::cref(s));
        thread t5(&C::f5, c, i);
        thread t6(C::f6, i);
        t1.join(); t2.join(); t3.join();
        t4.join(); t5.join(); t6.join();
        // Mit thread kann man keinen Funktionswert zurückgeben
    }

    { // b) Mit Initialisiererlisten
        int i = 0;
        thread t1{ f1, 1 };
        thread t2{ f2, 1, s, 2 };
        thread t3{ f3, std::ref(i) };
        thread t4{ f4, std::cref(s) };
        thread t5{ &C::f5, &c, i };
        thread t6{ C::f6, i };
        t1.join(); t2.join(); t3.join();
        t4.join(); t5.join(); t6.join();
    }

    { // c) Mit Lambda-Ausdrücken
        string r1;
        int r2, r4, r5, r6;
        int i = 0;
        thread t1([&r1] {r1 = f1(1); });
        thread t2([&r2, s] {r2 = f2(1, s, 2); });
        thread t3([&i] {f3(i); });
        thread t4([&r4, s] { r4 = f4(s); });
        thread t5([&r5, i] {C c; r5 = c.f5(i); });
        thread t6([&r6, i] {r6 = C::f6(i); });
        t1.join(); t2.join(); t3.join();
        t4.join(); t5.join(); t6.join();
    }
}
}
```



## 15.2 Aufgabe 16.1.4 Exceptions in Threads

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_18_MultiThreading.h

int throw_0(int n)
{
    if (n == 0)
        throw(std::logic_error("throw_0, n==0"));
    // if (n == 1) throw(std::exception("n==1")); //nur in MS C++, nicht Standard
    return n;
}

void parallel(std::function<void()> f1, std::function<void()> f2)
{
    thread t{ f1 }; // starte f1 in einem neuen thread
    f2();          // starte f2 in diesem thread
    t.join();      // warte auf f1
}

void Exception_Aufgabe_1(int p)
{ // Die Funktion throw_0 soll wie im Text mit dem Argument 0 eine
  // Exception auslösen. Beschreiben Sie den Programmablauf für jeden der
  // Aufrufe von parallel.
  if (p == 1) parallel([] {throw_0(1); }, [] {throw_0(1); });
  // f1 und f2 werden parallel abgearbeitet. Nachdem sie fertig sind,
  // wird t.join aufgerufen und die Funktion verlassen.
  if (p == 2) parallel([] {throw_0(1); }, [] {throw_0(0); });
  // f1 wird in einem Thread gestartet. Dann wird f2 gestartet und löst
  // eine Exception aus. Da diese nicht abgefangen wird, wird parallel
  // ohne join verlassen, was zu einem Programmabbruch führt.
  if (p == 3) parallel([] {throw_0(0); }, [] {throw_0(1); });
  // f1 wird in einem Thread gestartet und löst eine Exception aus. Das
  // führt zu einem Programmabbruch
  if (p == 4) parallel([] {throw_0(0); }, [] {throw_0(0); });
  // Je nachdem, ob zuerst die Exception im Thread oder in
  // f2 ausgeführt wird, erhält man eine andere Ursache für den
  // Programmabbruch.
}
```

## 15.3 Aufgabe 16.2.2 Kritische Bereiche mit Mutex und lockguard sperren

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_18_MultiThreading.h

// ----- Aufgabe 1. -----

class SperreFrüherOderSpäter {
    std::shared_mutex mtx;
    int kritischerBereich;
    int zeitaufwendigeOperation(int n) { return n; }

    void sperreFrüher()
    {
        std::scoped_lock<std::shared_mutex> lg(mtx);
        kritischerBereich = zeitaufwendigeOperation(17);
    }

    void sperreSpäter()
    {
        int n = zeitaufwendigeOperation(17);
        std::scoped_lock<std::shared_mutex> lg(mtx);
        kritischerBereich = n;
    }

public:
    void starteThreads()
    { /* Starte sperreFrüher oder sperreSpäter oft als Threads */
    }

    /*
    Lösung:
    Beim Aufruf über So bleibt der Zugriff auf kritischerBereich
    während der gesamten Ausführung der zeitaufwendigen Operation
    gesperrt. Bei OderSo dagegen nur während der kurzen Zuweisung.
    */
};

// ----- Aufgabe 2. -----

class ZweiKritischeBereiche {
    std::shared_mutex mtx;
    int kritischerBereich1;
    int kritischerBereich2;
    void zeitaufwendigeOperation1(int n)
    { /* arbeite mit kritischerBereich1 */
    }
    void zeitaufwendigeOperation2(int n)
    { /* arbeite mit kritischerBereich2 */
    }

    void Funktion1()
    {
        std::scoped_lock<std::shared_mutex> lg(mtx);
        zeitaufwendigeOperation1(17);
    }

    void Funktion2()
    {
        std::scoped_lock<std::shared_mutex> lg(mtx);
        zeitaufwendigeOperation2(17);
    }

public:
    void starteThreads()
    {
        // starte Funktion1 und Funktion2 in verschiedenen Threads
    }
    /*
    Lösung:
    Vermutlich ist es effizienter, die beiden kritischen Bereiche

```

```

    durch zwei verschiedene Mutexe zu sperren.
    */
};

// ----- Aufgabe 3. -----

class Logger { // sehr einfach, nur die Grundidee
    std::ofstream fout;
public:
    Logger(string fn)
    {
        fout.open(fn);
    }
    void log(string msg)
    {
        fout << msg << std::endl;
    }
};

class Multithread_Logger { // sehr einfach, nur die Grundidee
// Lösung: Nur einen mtx in die Klasse aufnehmen, und diesen
// vor dem Schreiben in log sperren
    std::ofstream fout;
    std::shared_mutex mtx;
public:
    Multithread_Logger(string fn)
    {
        fout.open(fn);
    }
    void log(string msg)
    {
        std::scoped_lock<std::shared_mutex> lg(mtx);
        fout << msg << std::endl;
    }
};

void test_Logger()
{
    const std::string TestDir = N_Dateinamen::TestDir; // oder z.B. "c:\\test\\"
    Logger Log(TestDir+"log.txt");
    Log.log("Meldung 1");
    Log.log("Meldung 2");
}

// ----- Aufgabe 4. -----
template<typename T>
class Multithread_vector // sehr einfach, nur die Grundidee
{
    std::vector<T> v;
    std::shared_mutex mtx;
public:
    void push_back(const T& x)
    {
        std::scoped_lock<std::shared_mutex> lg(mtx);
        v.push_back(x);
    }

    std::vector<T> to_vector()
    { // damit die üblichen Operationen für einen
      // Vektor verfügbar sind
        std::scoped_lock<std::shared_mutex> lg(mtx);
        return v;
    }
};

void alleAufgaben()
{
    Aufgabe_1();
    Exception_Aufgabe_1(0);
}

```

```
}  
} // end of namespace N_Loesungen_Multithreading
```





## 16 Lösungen Kapitel 18. Smart Pointer

```
##### Copyright notice #####
Lösungen zu Aufgaben aus dem Lehrbuch / Solutions to Exercises from the textbook
"Richard Kaiser: C++ mit Visual Studio 2022 und Windows Forms Anwendungen",
ISBN Hardcover: 978-3-662-65256-5, ISBN eBooks: 978-3-662-65257-2
```

Diese Quelltexte dürfen in eigenen Programmen verwendet werden, wenn sie diese beiden Zeilen enthalten:

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In keinem Fall wird eine Haftung für direkte, indirekte, zufällige Schäden oder Folgeschäden übernommen, die sich aus der Nutzung ergeben.

It is permitted to use this source text in programs of your own, provided that it contains the two lines

```
Copyright Richard Kaiser 2022, https://www.rkaiser.de/
C++ mit Microsoft Studio 2022 und Windows Forms Anwendungen
```

In no event shall liability be accepted for any direct, indirect, incidental or consequential damages arising from the use of these source texts.

```
##### end of Copyright notice #####
```

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_18_SmartPointer.h
```

```
#include <memory>
#include <algorithm>
#include <vector>
```

**16.1 Aufgabe 18.3**

```
// C:\VS2022-CppBuch-Kaiser-WinForms-Loesungen\Loesungen_ab_Kap_3\Loesung_Kap_18_SmartPointer.h
namespace N_Loesungen_Smart_Pointer
{
    using namespace System::Windows::Forms;

    using std::unique_ptr;
    using std::make_unique;
    using std::shared_ptr;
    using std::make_shared;
    // ----- Aufgabe 1. -----

    void Aufgabe_1()
    {
        std::vector<unique_ptr<int>> vu;
        std::vector<shared_ptr<int>> vs;
        // a)
        { // Ein vector<unique_ptr> kann nur mit Move-Semantik gefüllt werden,
          // nicht mit Kopier-Semantik
            unique_ptr<int> up = unique_ptr<int>(new int(2));

            // Kopiersemantik geht mit unique_ptr nicht:
            // vu.push_back(up);
            // vu.emplace_back(up);

            // Move-Semantik geht mit unique_ptr:
            vu.push_back(unique_ptr<int>(new int(2)));
            vu.emplace_back(unique_ptr<int>(new int(2)));

            // Move-Semantik geht mit make_unique:
            vu.push_back(make_unique<int>(2));
            vu.emplace_back(make_unique<int>(2));
        }
        // b)
        {
            // Ein vector<shared_ptr> kann mit Move- und KopierSemantik
            // gefüllt werden:
            shared_ptr<int> sp = shared_ptr<int>(new int(2));

            // Kopiersemantik geht mit shared_ptr:
            vs.push_back(sp);
            vs.emplace_back(sp);

            // Move-Semantik geht auch:
            vs.push_back(make_shared<int>(2)); // Das geht
            vs.emplace_back(make_shared<int>(3)); // Das geht
        }
        // c)
        {
            for (int i = 0; i < 10; i++)
                vs.push_back(make_shared<int>(10 - i));
            std::sort(vs.begin(), vs.end(), [](shared_ptr<int> x, shared_ptr<int> y)
                {return std::greater<int>()(*x, *y); });

            for (int i = 0; i < 10; i++)
                vu.push_back(make_unique<int>(10 - i));
            // std::sort(vu.begin(), vu.end(), [](unique_ptr<int> x,
            // unique_ptr<int> y) {return std::greater<int>()(*x, *y); });
        }
    }

    // ----- Aufgabe 2. -----

    class C_alt {
        int* p;
    };
}
```

```

public:
    C_alt(int n) :p(new int(n)) {}
    C_alt(const C_alt& x)
    {
        // ...
    }
    C_alt& operator=(const C_alt& x)
    {
        // ...
    }
    ~C_alt() { delete p; }
};

class C_neu_SP {
    std::shared_ptr<int> p;
public:
    C_neu_SP(int n) :p(make_shared<int>(n)) {}
    // Alle diese folgenden Funktionen sind überflüssig,
    // da der Datentyp von p eine Klasse ist:
    // C_neu(const C_neu& x) // der Kopierkonstruktor ist überflüssig
    // C_neu& operator=(const C_neu& x) // ebenfalls ist überflüssig
    // ~C_alt() { delete p; } // der Destruktor ist überflüssig
};

class C_neu_UP {
    std::unique_ptr<int> p;
public:
    C_neu_UP(int n) :p(make_unique<int>(n)) {}
    // Alle diese folgenden Funktionen sind überflüssig,
    // da der Datentyp von p eine Klasse ist:
    // C_neu(const C_neu& x) // der Kopierkonstruktor ist überflüssig
    // C_neu& operator=(const C_neu& x) // ebenfalls ist überflüssig
    // ~C_alt() { delete p; } // der Destruktor ist überflüssig
};

void Aufgabe_2()
{
    C_neu_SP s1(17);
    C_neu_SP s2 = s1;
    s2 = s1;

    // Kopieren geht mit C_neu_UP nicht
    C_neu_UP c1(17);
    // C_neu_UP c2 = c1;
    // c2 = c1;
}

// ----- Aufgabe 3. -----

// a)
class C_neu {
    // std::unique_ptr<int> a;
    // std::unique_ptr<int> b;
    std::shared_ptr<int> a;
    std::shared_ptr<int> b;
// b) Mit einem unique_ptr können Variablen des Typs C_neu
// nicht kopiert werden.
    void init() noexcept(false) {}; // Vor VS 2015 ohne noexcept
public:
    // C_neu() : a(std::make_unique<int>(int())), b(new int())
    C_neu() : a(std::make_shared<int>(int())), b(new int())
    {
        // Da a und b jetzt Klassen sind, wird für sie bei
        // einer Exception der Destruktor aufgerufen.
        init();
    }
};

void Aufgabe_3()
{

```

```

    C_neu c;
}

// c) Da C Zeiger enthält, fehlen der Destruktor,
//     Kopierkonstruktor und der Zuweisungsoperator.

// d) C_neu enthält keine Zeiger. Der Destruktor/Kopier-
//     konstruktor/Zuweisungsoperator von C_neu ruft die
//     Destruktoren/Kopierkonstruktoren/Zuweisungsoperatoren
//     aller Datenelemente auf. Deswegen sind diese
//     Funktionen in C_neu nicht notwendig.

// ----- Aufgabe 4. -----
class C1 {
public:
    C1() = default;
    C1(const C1&) {};
};

void Aufgabe_4a()
{ // http://stackoverflow.com/questions/701456/what-are-potential-dangers-when-using-
boostshared_ptr
    C1* p1 = new C1();
    shared_ptr<C1> s1(p1);
    shared_ptr<C1> s2(p1);
    /* Lösung:
    Da zwei shared_ptr auf dieselbe dynamische Variable zeigen
    wird für diese zwei mal delete aufgerufen.

    Mit make_shared kann dieser Fehler nicht auftreten.
    */
}

void Aufgabe_4b()
{ // http://stackoverflow.com/questions/701456/what-are-potential-dangers-when-using-
boostshared_ptr
    shared_ptr<C1> s1(new C1());
    shared_ptr<C1> s2(s1.get());
    /*
    Da zwei shared_ptr auf dieselbe dynamische Variable zeigen
    wird für diese zwei mal delete aufgerufen.
    */
}

void Aufgabe_4c(TextBox^ tb)
{ // http://stackoverflow.com/questions/701456/what-are-potential-dangers-when-using-
boostshared_ptr
    int* p = new int(17);
    {
        shared_ptr<int> s1(p);
    }
    tb->AppendText((*p).ToString() + "\r\n");
    /*
    Die dynamische Variable *p wird im Block freigegeben. Der anschließende Zugriff
    ist nicht definiert.
    */
}

// ----- Aufgabe 5. -----

struct Node1
{
    shared_ptr<Node1> next;
};

void Z_1(TextBox^ tb)
{
    Node1* n1 = new Node1;
    Node1* n2 = new Node1;
    shared_ptr<Node1> first(n1);
    first->next = shared_ptr<Node1>(n2);
}

```

```

    tb->AppendText("first.uc=" + first.use_count().ToString() + " first.next.uc=" +
first.get()->next.use_count().ToString() + "\r\n");
    nl->next = first; // Zyklus
    tb->AppendText("first.uc=" + first.use_count().ToString() + " first.next.uc=" +
first.get()->next.use_count().ToString() + "\r\n");
    // first.uc=2 first.next.uc=2
} // memory leak

// ----- Aufgabe 6. -----

void SP_WParameter(std::shared_ptr<int> sp, std::shared_ptr<int>& out)
{
    sp.get();
    int uc4 = sp.use_count(); // uc4=3
    int uc5 = out.use_count(); // uc5=3
    std::shared_ptr<int> sp1 = sp;
    int uc6 = out.use_count(); // uc6=4
}

void test_SP_Parameter()
{
    std::shared_ptr<int> s1 = std::make_shared<int>(3);
    int uc1 = s1.use_count(); // uc1=1
    std::shared_ptr<int> s2 = s1;
    int uc2 = s1.use_count(); // uc2=2
    SP_WParameter(s1, s1);
    int uc3 = s1.use_count(); // uc3=2
}

namespace N_Meinstring_smart_pointer
{
    class MeinString {
        char* s; // Zeiger auf nullterminierten String
        int n; // Länge des Strings
    public:
        MeinString(const char* p) // 1
        {
            n = strlen(p);
            s = new char[n + 1];
            strcpy(s, p);
        };

        MeinString(const MeinString& x) // Kopierkonstruktor
        {
            n = x.n; // 1
            s = new char[n + 1]; // 2
            strcpy(s, x.s); // 3
        }; //dieser Konstruktor stellt die Klasseninvariante her

        MeinString& operator=(const MeinString& x)
        {
            if (this != &x) // a
            {
                delete[] s; // b
                n = x.n; // 1
                s = new char[n + 1]; // 2
                strcpy(s, x.s); // 3
            }
            return *this; // c
        };
        ~MeinString() { delete[] s; }
    };

    class MeinString_smart_pointer {
        // std::shared_ptr<char> s = std::make_shared<char>();
        // char* s; // Zeiger auf nullterminierten String
        std::shared_ptr<char[]> s{ new char[1]{'\0'} };
        int n; // Länge des Strings
    public:

```

```
MeinString_smart_pointer(const char* p) // 1
{
    n = strlen(p);
    s.reset(new char[n + 1]);
    strcpy(s.get(), p);
};
// ~MeinString() { delete[] s; } nicht notwendig
};
void testOhneZuweisungsOp()
{
    {
        using namespace N_Meinstring_smart_pointer;
        MeinString_smart_pointer s1("abc"), s2(s1), s3("31234");
        s1 = s3;
    }
    MeinString s1("abc"), s2("31234");
    s1 = s2;
    // MeinString s3(s1);
}

void test_Aufgaben()
{
    Aufgabe_1();
    Aufgabe_2();
    Aufgabe_3();
    // Aufgabe 4 führt zu einem Programmabbruch
    // Aufgabe_4a();
    // Aufgabe_4b();
    // Aufgabe_4c();
    // Z_1();
    test_SP_Parameter();
}

} // end of namespace N_Loesungen_Smart_Pointer
```