

# 1. C/C++ Grundlagen mit dem C++Builder 2009

Dieser Kurs richtet sich an erfahrene Programmierer, die lernen wollen, wie man mit dem C++Builder C++-Programme für Windows erstellt. Er verbindet eine Einführung in die Grundlagen der Programmiersprachen C und C++ mit einer Einführung in die Anwendungsentwicklung mit dem C++Builder. Dabei werden vor allem diejenigen Sprachelemente von C++ behandelt, die auch zum Sprachumfang von C gehören. Außerdem werden die wichtigsten Komponenten des C++Builders vorgestellt, die man üblicherweise für Windows-Programme benötigt.

Dieser Kurs ist der erste von drei aufeinander abgestimmten Kursen, in denen der gesamte Sprachumfang des aktuellen ANSI/ISO-Standards von C++ und die wichtigsten Komponenten des C++Builders behandelt werden. Dabei stehen Zusammenhänge und Sprachkonzepte im Vordergrund vor Detailinformationen, die man auch in der Online-Hilfe findet.

**Voraussetzungen:** Gute Windows-Kenntnisse und Erfahrung in der Programmierung (z.B. mit Delphi, Borland Pascal, Java, Visual Basic, C, Assembler usw.)

**Zielgruppe:** Software-Entwickler, die auf C, C++ und den C++Builder umsteigen wollen.

**Aufteilung:** Vortrag mit vielen Übungen, in denen praxisnahe Programme entwickelt werden

**Dauer:** 5 Tage

## 1 Die Entwicklungsumgebung

- Projekte, Projektdateien und Projektoptionen
- Programmierhilfen
- Packages und eigenständig ausführbare Programme
- Der integrierte Debugger

## 2 Die wichtigsten VCL-Komponenten für Benutzeroberflächen

- TLabel* und *TEdit*
- Memos, ListBoxen, ComboBoxen und die Klasse *TStrings*
- Buttons und Ereignisse
- CheckBoxen, RadioButtons, GroupBox, Panel, RadioGroup und ScrollBar
- Hauptmenüs und Popup-Menüs
- Die Komponenten der Seite Dialoge
- Der Aufruf von eigenen Formularen und modale Fenster
- Die Komponenten der Seite „Zusätzlich“
- Mehrseitige Dialoge
- ImageList und ListView
- Formatierte Texte

## 3 Elementare Datentypen und Anweisungen

- Ganzzahldatentypen und Standardkonversionen
- Der Datentyp *bool*, die *char*-Datentypen und *\_\_int64*
- Gleitkommatypen und mathematische Funktionen
- Arrays, Zeiger und Zeigerarithmetik
- Strukturen und Klassen
- Zeiger und Strings
- Dynamisch erzeugte Variablen: *new* und *delete*
- Zeiger als Parameter und die Zeigertypen der Windows-API

Nullterminierte Strings und Zeiger auf *char*  
*typedef* und *typeid*-Ausdrücke  
Die Stringklasse *AnsiString*  
Referenztypen, Werte- und Referenzparameter  
Präprozessoranweisungen  
Exception Handling: *try* und *throw*  
Namensbereiche

## 4 Einige Klassen der Standardbibliothek

Die Stringklassen *string* und *wstring*  
Sequenzielle Container der Standardbibliothek  
Dateibearbeitung mit den Stream-Klassen  
Assoziative Container  
C++0x-Erweiterungen der Standardbibliothek

## 5 Funktionen

Werteparameter und Referenzparameter  
Default-Argumente  
Zeiger auf Funktionen  
Der Aufruf von Funktionen aus Delphi im C++Builder  
*inline*-Funktionen  
Überladene Funktionen  
Überladene Operatoren mit globalen Operatorfunktionen  
Referenzen als Funktionswerte  
Die Ein- und Ausgabe von selbst definierten Datentypen

## 6 Separate Kompilation und Bibliotheken

Projekte, Bindung und Header-Dateien  
Der Aufruf von in C geschriebenen Funktionen  
Dynamic Link Libraries (DLLs)  
Implizit und explizit geladene DLLs

## 7 Datenbank-Komponenten der VCL

Verbindung mit ADO-Datenbanken – der Connection-String  
Tabellen und die Komponente *TDataSet*  
Tabellendaten lesen und schreiben  
Die Anzeige von Tabellen mit einem *DBGrid*  
SQL-Abfragen

## 2. Objektorientierte Programmierung in C++ mit dem C++Builder, Exception-Handling und die VCL

Dieser Kurs richtet sich an Programmierer mit guten Kenntnissen der Programmiersprache C, die die objektorientierte C++-Programmierung mit dem C++Builder lernen wollen. Dabei werden zusammen mit den Sprachelementen von C++ auch die objektorientierte Analyse und das objektorientierte Design behandelt. Die oft nicht einfachen Alternativen beim Entwurf von Klassenhierarchien werden ausführlich diskutiert. Als Beispiel für eine Klassenhierarchie wird die Klassenbibliothek des C++Builders (VCL) vorgestellt. Beim Exception-Handling werden nicht nur die Sprachelemente, sondern auch die Auswirkungen auf das Programmdesign gezeigt.

Dieser Kurs ist der zweite von drei aufeinander abgestimmten Kursen, in denen der gesamte Sprachumfang des aktuellen ANSI/ISO-Standards von C++ und die wichtigsten Komponenten des C++Builders behandelt werden. Dabei stehen Zusammenhänge und Sprachkonzepte im Vordergrund vor Detailinformationen, die man auch in der Online-Hilfe findet.

**Voraussetzungen:** Gute Kenntnisse und Programmiererfahrungen in C im Umfang des Kurses „C/C++ Grundlagen mit dem C++Builder“.

**Zielgruppe:** Erfahrene C- und C++-Programmierer

**Aufteilung:** Vortrag mit vielen Übungen, in denen praxisnahe Programme entwickelt werden

**Dauer:** 5 Tage

### 1 Objektorientierte Programmierung

#### 1.1 Klassen

- Datenelemente und Elementfunktionen
- Datenkapselung: Die Zugriffsrechte `private` und `public`
- Der Aufruf von Elementfunktionen und der *this*-Zeiger
- Konstruktoren und Destruktoren
- OO Analyse und Design: Der Entwurf von Klassen
- Ein wenig Programmierlogik: Klasseninvarianten und Korrektheit
- UML-Diagramme mit Together im C++Builder

#### 1.2 Klassen als Datentypen

- Der Standardkonstruktor
- Objekte als Klasselemente und Elementinitialisierer
- friend*-Funktionen und -Klassen
- Überladene Operatoren als Elementfunktionen
- Der Copy-Konstruktor
- Der Zuweisungsoperator `=` für Klassen
- Benutzerdefinierte Konversionen
- Explizite Konstruktoren
- Statische Klasselemente
- Konstante Klasselemente und Objekte
- Klassen und Header-Dateien

### 1.3 Vererbung

- Die Elemente von abgeleiteten Klassen
- Zugriffsrechte auf die Elemente von Basisklassen
- Konstruktoren, Destruktoren und implizit erzeugte Funktionen
- Vererbung bei Formularen im C++Builder
- OO Design: *public* Vererbung und „ist ein“-Beziehungen
- OO Design: Komposition und „hat ein“-Beziehungen
- Konversionen zwischen *public* abgeleiteten Klassen
- protected* und *private* abgeleitete Klassen
- Mehrfachvererbung und virtuelle Basisklassen

### 1.4 Virtuelle Funktionen, späte Bindung und Polymorphie

- Der statische und der dynamische Datentyp
- Virtuelle Funktionen
- Die interne Realisierung von virtuellen Funktionen: *vptr* und *vtbl*
- OO-Design: Der Einsatzbereich von virtuellen Funktionen
- Virtuelle Konstruktoren und Destruktoren
- Virtuelle Funktionen in Konstruktoren und Destruktoren
- Virtuelle Funktionen und Erweiterbarkeit
- Rein virtuelle Funktionen und abstrakte Klassen
- OO-Design: Virtuelle Funktionen und abstrakte Basisklassen
- Zeiger auf Klassenelemente

### 1.5 Laufzeit-Typinformationen

- Typinformationen mit dem Operator *typeid*
- Typkonversionen mit *dynamic\_cast* und *static\_cast*

## 2 Die Bibliothek der visuellen Komponenten (VCL)

- Besonderheiten der VCL
- Visuelle Programmierung und Properties (Eigenschaften)
- Die Klassenhierarchie der VCL
- Selbst definierte Komponenten und ihre Ereignisse
- Botschaften (Messages)
- Die Erweiterung der Komponentenpalette
  
- Die Steuerung von MS-Office: Word-Dokumente erzeugen
- Internet-Komponenten
- Kaufmännische Rechnungen: Die Klassen *Currency* und *BCD*
- Klassen und Funktionen zu Uhrzeit und Kalenderdatum
- Die Klasse *Set*

## 3 Exception-Handling (Ausnahmebehandlung)

- Die *try*-Anweisung
- Exception-Handler und Exceptions der Standardbibliothek
- Vordefinierte Exceptions der VCL
- Der Programmablauf bei Exceptions
- Das vordefinierte Exception-Handling der VCL
- throw*-Ausdrücke und selbst definierte Exceptions
- Fehler, Exceptions und die Korrektheit von Programmen
- Die Freigabe von Ressourcen bei Exceptions
- Die Klasse *auto\_ptr*
- Exception-Spezifikationen
- Die Funktion *terminate*

## 3. Templates und die STL mit dem C++Builder

Templates sind Vorlagen, aus denen der Compiler Klassen und Funktionen erzeugt. Die Standard Template Library (STL) ist eine Bibliothek, die zum ANSI/ISO-Standard von C++ gehört und die auf der Basis von Templates implementiert ist. Sie enthält viele Algorithmen und Datenstrukturen, die eine einfache Lösung häufig auftretender Programmieraufgaben ermöglichen. Aufgrund ihrer speziellen Architektur ist sie vielseitiger, leistungsfähiger, sicherer und einfacher zu benutzen als viele andere Bibliotheken. Da sowohl Templates als auch die STL in der jetzt gültigen Form erst relativ spät in C++ aufgenommen wurden, sind sie noch nicht sehr bekannt.

In diesem Kurs werden Templates und die STL vorgestellt. Dabei werden die Vorteile der STL-Klassen gegenüber den entsprechenden konventionellen Sprachelementen gezeigt (z.B. Stringklassen – nullterminierte Strings, Containerklassen – Arrays, Streamklassen – *stdio*-Funktionen). Außerdem wird gezeigt, wie die STL, ihre Container und ihre Algorithmen aufgebaut sind und wie man die STL um eigene Algorithmen erweitern kann.

Dieser Kurs ist der dritte von drei aufeinander abgestimmten Kursen, in denen der gesamte Sprachumfang des aktuellen ANSI/ISO-Standards von C++ und die wichtigsten Komponenten des C++Builders behandelt werden. Dabei stehen Zusammenhänge und Sprachkonzepte im Vordergrund vor Detailinformationen, die man auch in der Online-Hilfe findet.

**Voraussetzungen:** Gute Kenntnisse in C++ und der objektorientierten Programmierung mit C++ (im Umfang des Grundlagen-Kurses und des OO-Kurses).

**Zielgruppe:** Erfahrene C++-Programmierer

**Aufteilung:** Vortrag mit vielen Übungen, in denen praxisnahe Programme entwickelt werden

**Dauer:** 5 Tage

### 1 Die Stringklasse *string* der C++-Standardbibliothek

Die Stringklasse *string*  
Stringstreams

### 2 Container der Standardbibliothek

#### 2.1 Sequenzielle Container

Die Container-Klassen *vector*, *list* und *deque*  
Algorithmen der Standardbibliothek  
Die Container-Adapter *stack*, *queue* und *priority\_queue*  
Container mit Zeigern

#### 2.2 Assoziative Container

Die Container *set* und *multiset*  
Die Container *map* und *multimap*  
Iteratoren der assoziativen Container

#### 2.3 Dateien

Stream-Variablen, ihre Verbindung mit Dateien und ihr Zustand  
Fehler und der Zustand von Stream-Variablen  
Lesen und Schreiben von Binärdaten mit *read* und *write*

Lesen und Schreiben von Daten mit den Operatoren << und >>  
Manipulatoren und Funktionen zur Formatierung von Texten  
Dateibearbeitung im Direktzugriff

### **2.4 Die numerischen Klassen der Standardbibliothek**

Komplexe Zahlen, Valarrays und Slices

## **3 Templates und die STL**

### **3.1 Generische Funktionen: Funktions-Templates**

Funktions-Templates mit Typ-Parametern  
Spezialisierungen von Funktions-Templates  
Funktions-Templates mit Nicht-Typ-Parametern  
Explizit instanziierte Funktions-Templates  
Explizit spezialisierte und überladene Templates  
Rekursive Funktions-Templates

### **3.2 Generische Klassen: Klassen-Templates**

Klassen-Templates mit Typ-Parametern  
Spezialisierungen von Klassen-Templates  
Templates mit Nicht-Typ-Parametern  
Explizit instanziierte Klassen-Templates  
Partielle und vollständige Spezialisierungen  
Elemente und friend-Funktionen von Klassen-Templates  
Ableitungen von Templates  
Exportierte Templates

### **3.3 Funktionsobjekte in der STL**

Der Aufrufoperator ()  
Prädikate und arithmetische Funktionsobjekte  
Binder, Funktionsadapter und Negatoren

### **3.4 Iteratoren und die STL-Algorithmen**

Die verschiedenen Arten von Iteratoren  
Umkehriteratoren  
Einfügefunktionen und Einfügeiteratoren  
Stream-Iteratoren  
Container-Konstruktoren mit Iteratoren  
STL-Algorithmen für alle Elemente eines Containers

### **3.5 Die Algorithmen der STL**

Lineares Suchen und Zählen  
Suche nach Teilfolgen  
Minimum und Maximum  
Elemente vertauschen  
Kopieren von Bereichen  
Elemente transformieren, ersetzen und entfernen  
Partitionen und Sortieren  
Binäres Suchen in sortierten Bereichen  
Mischen von sortierten Bereichen  
Mengenoperationen auf sortierten Bereichen