

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*