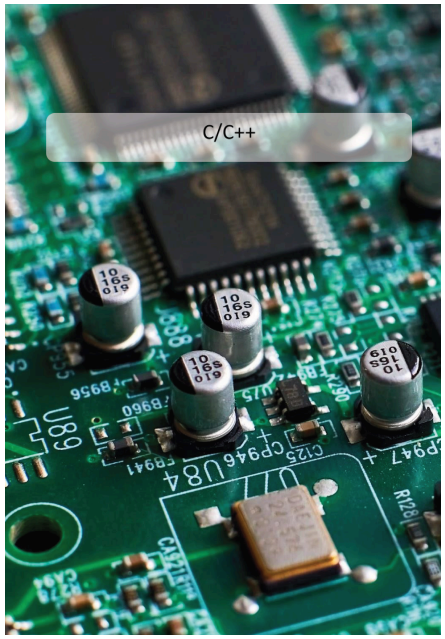
Agenda C/C++



Agenda zu C: Wiederholung / Vertiefung / Auffrischung wichtiger Themen

Zeiger

- Definition
- Zeiger (Pointer) initialisieren (Referenzierung), Nullzeiger
- Dereferenzierung
- Zeigerarithmetik
- Anwendung: Parameterübergabemechanismus *Call-by-Value* und *Call-by-Address*

Funktionen

- Funktionsdeklaration und Aufruf
- Rückgabewert und Parameter
- Lokale und globale Variablen
- Parameterübergabemechanismus *Call-by-Value* und *Call-by-Address*

- Funktionen und der *Stack* (*Stapel*)

Dynamische Speicherverwaltung

- Speicher reservieren und freigeben: `malloc` und `free`
- Dynamische Speicherverwaltung und der *Heap* (*Halde*)

Felder (Arrays)

- Deklaration und Initialisierung
- Auf Feldelemente zugreifen
- Felder in Schleifen durchlaufen

Strukturen

- Definition und Anwendung
- Schlüsselwort `struct`
- Zugriff auf Elemente
- Initialisierung von Strukturen
- Felder von Strukturen
- Schlüsselwort `typedef`

Agenda zu C++

Klassen und Objekte

- Dateiorganisation (Header-Dateien, Implementierungs-Dateien)
- Begriffe *Klasse* und *Objekt*
- Nomenklator
- Punkt-Operator für den Zugriff
- Zugriffsklassen (`private`, `public`)
- *setter-*/*getter*-Methoden
- Der `this`-Operator

Initialisierung von Objekten: Konstruktoren

- Was sind Konstruktoren?
- Überladen von Konstruktoren
- Element-Initialisierungsliste (*Member Initializer List*)

- Konvertierungskonstruktoren

Statische Datenelemente (`static`)

- Statische Datenelemente
- Statische Elementfunktionen

Referenzen

- Was ist eine Referenz?
- Der Adreßoperator & bei Referenzen
- Referenzen können nicht erneut zugewiesen werden
- Null-Zeiger und Null-Referenzen
- Zeiger und Referenzen im Vergleich

Technik der Übergabe von Parametern

- Parameter mit Referenzen übergeben
- Parameterübergabemechanismen im Vergleich: *Call-by-Value*, *Call-by-Address* und *Call-by-Reference*

Überladen von Operatoren

- Stelligkeit und Priorität
- Assoziativität
- Realisierung: Innerhalb und außerhalb der Klasse

Ein- und Ausgabe mit *Streams*

- Ein Überblick über Streams
- Eingabe mit `std::cin`
- Ausgabe mit `std::cout`

Namensräume / Namespaces

- Hintergrund für das Konzept der "Namensräume"
- Schlüsselwörter `using` und `namespace`

Zeichenketten: Klasse `std::string`

- Einbinden der Klasse `std::string` aus der C++–Standardbibliothek
- Die Klasse `std::string` exemplarisch betrachtet

Exceptions / Behandlung von Ausnahmen

- Was sind Exceptions?
- Komponenten in der Behandlung von Ausnahmen
- Schlüsselwörter `try` und `catch`

Dynamische Speicherverwaltung

- Speicherbereiche eines C++–Programms
- Lebensdauer von Variablen in einem C++–Programm
- Stack und Heap (Stapel und Halde)
- Operatoren `new` und `delete`

“Rule of Three”: Kopierkonstruktor, Wertzuweisungsoperator und Destruktor

- Objekte mit dynamischen Daten
- Ein Destruktor zum automatisierten Freigeben von allokierten Speicher
- Objekte kopieren und zuweisen
- *Flat Copy* versus *Deep Copy*
- Überladen von Kopierkonstruktor und Wertzuweisungsoperator
- Die “Rule of Three”

Vererbung

- Entwicklung einer Hierarchie von Klassen
- Konstruktoren von abgeleiteten Klassen und Basisklassen
- Verwendung von `public`, `protected` und `private`
- Überschreiben von Methoden
- Schlüsselwörter `virtual`, `override` und `final`
- Polymorphismus
- Schnittstellekonzept (Interface)
- Kontrakt mit mehreren Schnittstellen
- Abstrakten Basisklasse
- Unterschied Schnittstelle versus abstrakte Basisklasse

Templates / Schablonen

- Was sind Templates?
- Ein universeller Taschenrechner

STL (Standard Template Library)

- Container
- Iteratoren
- Algorithmen
- Funktoren

Ausblick auf Modern C++

- Verschiebe-Semantik (*Move*-Semantik)
- Schlüsselwort `auto`
- Lambdas
- Intelligente Zeiger (Smart Pointer)
