C++

Objektorientierte Programmierung mit C++

2021, Tobias Joschko

Inhalt

- **Vorstellung**
- Einleitung, Geschichte
- Grundlagen
- Klassen
- Vererbung
- Zusätzliche Themen
- Die Standardbibliothek

Vorstellung

- Ablauf
  - Termine → 22 Stunden → Verteilt auf 8 Termine
  - Kombination Vorlesung + Labor
  - Übungsaufgaben
- Inhalt
  - UML
- Klausur (voraussichtlich 08.12.21)
- Übungsprojekte
  - Klasse Zahl
    - → Vererben an Klasse Bruch oder Komplex
  - ToDo Liste mit Klassen auf der command line
  - Spiel Schere-Stein-Papier oder TicTacToe

Inhalt

- Vorstellung
- **Einleitung, Geschichte**
- Grundlagen
- Klassen
- Vererbung
- Zusätzliche Themen
- Die Standardbibliothek

Einleitung, Geschichte

- Prozedural vs. Objektorientiert
- Einkapselung, Polymorphismus, Vererbung
- Statische und dynamische Bindung
- Geschichte von C++
- Literatur

Einleitung, Geschichte
Prozedural vs. Objektorientiert

- In traditionellen prozeduralen Programmiersprachen (z.B. C) besteht das Programm aus einer Einheit von Daten und Funktionen (lokal & global).
- Thematische Zusammenfassung von Daten und Funktionen zu Gruppen → Objekte
- Objekte sind für eigene Daten zuständig.
- Interaktion von unterschiedlichen Objekten bilden ein Programm.
- Dies ermöglicht eine verbesserte Strukturierung von (insbesondere komplexen) Programmen.

Einleitung, Geschichte
Einkapselung, Polymorphismus, Vererbung

- Einkapselung:
  - Objekt kennt eigene Daten und Operationen (Methoden).
  - Man kann Daten verbergen und nur über Funktionen modifiziert lassen.
  - Programmierer muss die Aufgabe in eine Interaktion von Objekten umwandeln.
- Polymorphismus:
  - Methoden und Operanden, welche dieselbe Aufgabe an unterschiedlichen Objekten ausführen dürfen den gleichen Namen haben. Die steigert die Lesbarkeit von Programmen.
  - → (Bruch) a + (Bruch) b --- (Vektor) a + (Vektor) b
- Vererbung:
  - Hierarchische Verknüpfung von Objekten.
  - Objekte der Unterklasse haben Zugriff auf Methoden der Oberklasse (z.B. Student kann Methoden von Person nutzen).

Einleitung, Geschichte
Statische und dynamische Bindung

- Im Gegensatz zu traditionellen Programmiersprachen (z.B. C) erlaubt C++ die dynamische Bindung von Identifiern an ihre Typen während der Laufzeit und nicht schon vorher durch den Compiler (--> Polymorphismus).
- Diese Zuordnung während der Laufzeit erlaubt dem Programmierer mehr Freiheiten und ermöglicht polymorphe Methoden

```
string A = "aaa"
string B = "bbb"
string C = a + b; // (C = "aaabbb")
```

Einleitung, Geschichte
Geschichte von C++

- Bjarne Stroustrup startete mit der Erweiterung für C in den Bell Labs von AT&T in 1979. Zunächst wurde es “C mit Klassen“ genannt.
- Video von Bjarne Stroustrup: "Why I create C++"
- Ca. 1998: erster Iso Standard mit C++98
- 2011: C++11 als große Neuerung mit der Standardbibliothek
- C++14
- C++17
- C++20

Einleitung, Geschichte
Literatur

- Bücher:
  - Stroustrup → Siehe homepage
  - Allgemein C++ Bücher aus der Bibliothek
- Online Quellen:
  - http://www.cplusplus.com/doc/tutorial/
  - https://en.cppreference.com/w/
  - http://www.stroustrup.com
  - https://de.wikibooks.org/wiki/C%2B%2B-Programmierung
  - https://www.cprogramming.com/
  - Codecademy Cheatsheet
  - Project Euler
- Sonstiges:

Inhalt

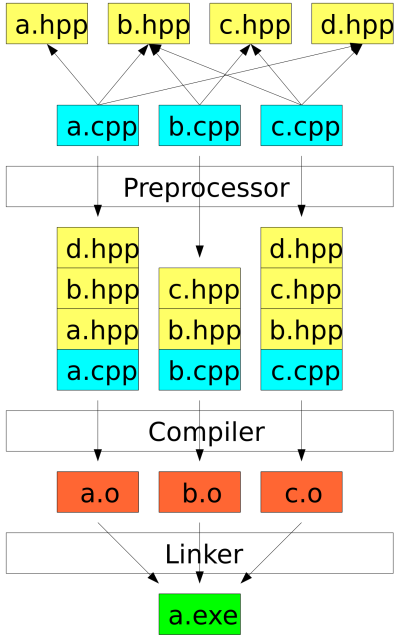
- Vorstellung
- Einleitung, Geschichte
- **Grundlagen**
- Klassen
- Vererbung
- Zusätzliche Themen
- Die Standardbibliothek

Grundlagen

- Starten eines C++ Programms
- Grundstruktur eines Programms
- IDE
- Ein- / Ausgabe
- Statische Polymorphie
- Referenzen in C++
- Konstanten
- Datentypen
- Aufgaben

Grundlagen
Starten eines C++ Programms

- Zwei Arten Programme auf Computern auszuführen:
  - Interpreter Programmiersprachen werden zeilenweise ausgeführt (z.B. Perl, Python, Java).
  - Compiler die Code in Maschinencode Übersetzen und die als eigenständige Programme ausgeführt werden können (z.B. C, Pascal, C++).
- Compiler-Sprachen brauche keine zusätzliche Laufzeitumgebung und sind in der Regel performanter.
- Erstellen eines C++ Programms:
  - ***Sourcecode*** → ***Objektdatei*** → **Maschinencode**
  - g++ beispiel.cpp → a.out (Unix) oder a.exe (Windows)
  - g++ beispiel.cpp -o HelloWorld → HelloWorld oder Helloworld.exe



Grundlagen
Allgemeine Programmstruktur

- Ein C++ Programm kann aus unterschiedlichen Funktionen bestehen die auf verschiedene Dateien verteilt sein können, es gibt aber immer nur eine main() Funktionen von derer die anderen (in-)direkt aufgerufen werden.
- Aufbau eines C++ Moduls:

[Präprozessor Direktiven]
[Typ, Klassen - Deklarationen]
[Definition und Deklaration von Variablen (globale Variablen)]
[Deklaration von Funktionen (=Prototypen)]
Definition von Funktionen (nur eine Funktion wird main genannt)

Grundlagen
IDE

- Visual Studio Code
- (Eclipse)
- GitHub → Source Code Verwaltung
- Andere IDEs ?

Grundlagen
Namespace

- C++ bindet sehr viele Variablen-, Klassen- und Funktionsdefinitionen in include-Files. Dadurch besteht die Gefahr, dass gleiche Namen verwendet werden → Namespace
- Variablen, Klassen, etc. können in C++ in einem Namespace definiert werden.
- Zugriff mit Doppelpunkt ::

```
int x;       // voller Name ::x
Namespace A {
  int x;     // voller Name A::x
}
```

Grundlagen
Namespace

- Funktionen und Klassen der Standardbibliothek befinden sich im *Defaultnamespace* std

```
std::string name;
std::cout << "Hallo";
```

```
using namespace std;
string name;
cout << "Hallo";
```

- Nachteil von ‚using namespace std‘: Man holt sich alle Namen in seinen aktuellen Scope. Vielleicht auch welche von denen man nicht weiß → Namenskonflikt

- Andere Möglichkeit:

```
#include <iostream >
using std::cout; // std::cout hereinholen
```

Grundlagen
Ein-/Ausgabe

- C++ unterstützt gesamtes I/O-System von C plus eigene Objektorientierte Routinen, die für eigene Klassen erweitert Werden können.
- In C++ arbeitet man mit Streams:

Stream	Bedeutung	Standardgerät	I/O-Operator
cin	standard input	Tastatur	> >
cout	standard output	Bildschirm	<<
cerr	err	Bildschirm	> >

- Namespace std: std::cin und std::cout
- Mit Unix Befehlen umlenken:
  - a.out < eingabe.txt
  - a.out > ausgabe.txt

Grundlagen
Statische Polymorphie (Funktionsüberladung)

- In C++ können mehrere unterschiedliche Funktionen den gleichen Namen habe → Bedingung: verschiedene Parameter (Anzahl, Typ)

```
void ausgabe (int x){
  cout << "Integer = " << x << endl;
}
void ausgabe (double x) {
  cout << "Double = " << x << endl;
}

int main () {
  ausgabe(2);
  ausgabe(3.141);
}
```

```
Integer = 2
Double = 3.141
```

Grundlagen
Referenzen in C++

- Eine Referenz ist ein Synonym (anderer Name, Alias) für eine bereits existierend Variable.

```
int x;      // Definition von x belegt Speicher
int &z = x; // x und z greifen auf die gleicher Speicherstelle
z = 8;      // x hat auch den Wert 8
```

- Unterschied zum Pointer in C:

```
void quadrat (int *x) {
  (*x) = (*x) * (*x);
}
int k = 2;
quadrat (&k);
```

```
void quadrat (int &x) {
  x = x * x;
}
int k = 2;
quadrat (k);
```

Grundlagen
Konstanten

- In C++ können Objekte definiert werden, denen ein fester Wert zugewiesen wird, der sich im gesamten Programm nicht ändern darf.
- Objektzugriff wird vom Compiler geprüft und eine Zuweisung ergibt eine Fehlermeldung

```
const int MAX = 10; // in C, #define MAX 10
const double PI = 3.1415927; // #define PI 3.14
Pi += 2; //Fehlermeldung beim Kompilieren
```

Grundlagen
Datentypen

C	C++	Inhalt
stdio.h		printf …
	iostream	cout, cin …
math.h	cmath	sin, cos, sqrt ..
stdlib.h	cstblib	malloc, exit, ..
string.h	cstring	strlen, strcpy, …
	string	Klasse string
	vector	Klasse vector

- u.v.m

Grundlagen
Datentypen

- std::string

```
string str1("Hallo"), str2 = "wie gehts", str3;
str3 = str1;
str3 += str2;
```

- std::vector

```
vector<string> vec1;
vec1.push_back(str1);
vec1.push_back(str2);
vector<string> vec2 (vec1);
cout << " Size von vec2 = " << vec2.size() << endl;
```

Grundlagen
Datentypen

- std::map

```
map<string, int> m { {"GPU", 30}, {"SSD", 128} };
m["CPU"] = 20;
```

- Etc → Siehe Standard Bibliothek

Grundlagen
Übungsaufgaben

- I/O Nutzen: Zahlen einlesen bis 0
  - Addieren → Summe
  - Als Vector zurück geben
- Funktion Überladen zum quadrieren unterschiedlicher Datentypen
- Telefonbuch mit std::map