

# Überblick

Zunächst werden die Ziele der Veranstaltung und die Organisation erläutert. Dann wird verdeutlicht, weshalb als Sprache C++ verwendet wird.

## Inhalt

1. Ziele .....	3
2. Organisation der Veranstaltung .....	9
2.1. Vorlesung.....	9
2.2. Praktikum .....	10
2.3. Hörsaalübungen .....	12
2.4. Klausur .....	13
2.5. Persönliche Selbstorganisation .....	14
2.5.1. Skript der Veranstaltung .....	15
3. Die Sprache .....	16
3.1. Warum C++ .....	16
3.2. Die Programmiersprache C .....	17

3.3. C++ .....	18
3.4. Zeittafel .....	19
4. Literatur.....	20

# 1. Ziele

In der Veranstaltung wird "**Programmieren**" vermittelt.

Programmieren:

- bedeutet **Strukturierung** komplexer Probleme.
  - divide and conquer (teile und herrsche)
  - Übung für viele andere Aufgaben
- ist eine äußerst **creative** Tätigkeit.
  - Entwurf und Realisierung
  - nicht nur kodieren vorgegebener Abläufe
- setzt handwerkliche **Fertigkeiten** voraus.
  - Beherrschung der Ausdrucksmittel einer Programmiersprache

## Das Bild des Programmierers

- war früher:
  - unverstandener **Freak**, Hacker und Guru zugleich:
    - tricky programming
    - Hauptsache, der Compiler versteht mich
- ist heute:
  - Ingenieur, **Informatiker**
    - sieht Programmieren als eine Tätigkeit im systematischen Entwurfsprozess
    - ist Teamwork, nicht Einzelkämpfer
    - hat das Ziel, wieder verwendbaren Code zu entwickeln, d.h leichte Verständlichkeit für andere Programmierer ist ihm wichtig

Der Erfinder der Programmiersprache C++, B. Stroustrup hat Programmieren wie folgt umrissen:

- Programm-Design und Programmieren sind menschliche Aktivitäten.
- Es gibt kein "allgemeines Rezept", das Intelligenz, Erfahrung und "guten Geschmack" ersetzen kann.
- Das Experimentieren ist für alle nicht-trivialen Software-Projekte essentiell.
- Entwurf und Programmieren sind iterative Aktivitäten.
- Die Systeme, die wir entwickeln, tendieren stets zu einer Komplexitäts-Obergrenze, die durch uns selbst und unsere Werkzeuge gesetzt ist.

Zum Programmieren sind Werkzeuge erforderlich:

- Programmiersprachen
  - Ausdrucksmittel zum Spezifizieren der Aktionen, die der Rechner ausführen soll.
  - Beispiele:
    - C++, Java
    - C, Pascal
    - Prolog
    - Occam
    - FP
- Entwicklungsumgebungen
  - Unterstützen bei der Programmentwicklung
    - Kodieren in Programmiersprache
    - Übersetzen und Binden
    - Ausführen des Programms und Test
    - Dokumentation
    - Versionsführung

- Beispiele:
  - Forte for Java
  - NetBeans, Eclipse
  - Microsoft Developer Studio

**Welche Programmiersprachen und Entwicklungswerkzeuge kennen Sie?**

**Woher kennen Sie diese Werkzeuge?**

Sie werden lernen, „Programm-Texte“ wie den Folgenden zu lesen und zu schreiben.

```
#include <iostream>
#include <set>
#include <ctime>    // für time()
const int ZAHLEN = 45;
const int NUMTIP = 6;

unsigned int zufallsZahl() {          // erzeuge Zufallszahl zwischen 1 und 45
    return 1 + random() % ZAHLEN;
}
int main() {
    typedef set<unsigned int> IntSet;
    IntSet tip;

    srandom(time(NULL));              // initialisiere den Zufallszahlengenerator
    while (tip.size() < NUMTIP) { //Zufallszahl in die Menge einfügen
        tip.insert(zufallsZahl());
    }

    //Menge ausgeben
    for (IntSet::iterator i = tip.begin(); i != tip.end(); ++i) {
        cout << *i << endl;
    }
    return 0;
}
```

## Wer kann schon was mit dem Programmtext anfangen?

## 2. Organisation der Veranstaltung

Die Veranstaltung ist unterteilt in

- Vorlesungen (4 SWS)
- in Vorlesung integrierte Hörsaalübungen und
- Praktika (2 SWS)

### 2.1. Vorlesung

Die Vorlesung dient der zusammenhängenden Darstellung und Vermittlung von Grund- und Spezialwissen, sowie methodischer Kenntnisse rund um die Thematik „Programmieren in C++“.

- Die Vorlesung ist **kein** Monolog!
- Sie sollten stets **Nachfragen**, wenn etwas unklar geblieben ist oder Sie einen Sachverhalt anders sehen.
- Die dargestellten Programme sollen von Ihnen "erfahren" werden. Dazu werden Sie die Programme in der Vorlesung auf Ihre **Notebooks** herunterladen, verändern und laufen lassen.
- Wer kein Notebook besitzt, kann sich eins für die Dauer der Veranstaltung beim Fachbereich ausleihen.

## 2.2.Praktikum

Im Praktikum werden Übungsaufgaben von Ihnen selbstständig bearbeitet.

### Ablauf:

1. Die Übungsaufgaben können über meine Homepage herunter geladen werden.
2. Die Aufgaben werden in Hörsaalübungen vorbesprochen.
3. Die Lösung wird von Ihnen zu Hause (oder in nicht belegten Labors) erarbeitet:
  - Verfeinerung des Entwurfs
  - Ausarbeitung des Programms inklusiv Eintippen
  - setzt Editor voraus, aber nicht unbedingt Compiler
4. Im Praktikum wird der Entwurf in den Rechner eingegeben bzw. von Ihrem Datenträger oder Ihrem persönlichen Laufwerk (userv) auf die Festplatte kopiert und:
  - a) der Entwurf wird noch mal verfeinert;
  - b) das Programm wird übersetzt, getestet und
  - c) zur Abnahme vorbereitet.

d)Die Lösung wird von mir (oder einem Tutor) testiert, wenn

- die Lösung erklärt werden kann,
- das Programm ablauffähig ist und
- die Lösung nachvollziehbar ist.

Folgende **Regeln** gelten für die Praktika:

- Die Teilnahme am Praktikumsterminen ist Pflicht  
**Ein Testat gibt es i.d.R. nur zum jeweiligen Termin!**  
Bewertung: erfolgreich / nicht erfolgreich, also keine Noten
- Erfolgreiche Teilnahme am Praktikum ist Zulassungsvoraussetzung für Klausur:  
**alle 6 Praktikumsübungen müssen testiert sein**
- Aushang von Gruppeneinteilung, Terminen und Logins
- jeder bekommt ein persönliches Login
- Gruppenarbeit: jeweils 2 Studierende an einem PC erarbeiten gemeinsam ein Programm
  - aber **jede(r) muss eine Kopie besitzen**
  - keine Arbeitsteilung: jede(r) muss alles beherrschen
  - gemeinsame Vorbereitung ist empfehlenswert

- Modell "Fahrschule":
  - Autofahren lernt man nicht als Beifahrer
  - "Fahrer" bedient Tastatur und Maus
  - "Beifahrer" beobachtet, denkt mit, berät
- in jeder Übung werden die Rollen getauscht!
- Abschreiben und Kopieren ist verboten

Programmieren lernt man nur durch **eigenes Üben**; wer mit Programmieren echte Schwierigkeiten hat, sollte sich ernsthaft fragen, ob Informatik das richtige Fach für ihn ist!!!

### **2.3. Hörsaalübungen**

In Hörsaalübungen werden Aufgabenstellungen in kleinen Gruppen gemeinsam bearbeitet. Die Aufgabenstellung richtet sich am aktuellen Inhalt der Veranstaltung.

Die gefundene Lösung wird am Ende der Übung von einer Gruppe an der Tafel erklärt bzw. werden je nach Aufgabenstellung Lösungsansätze diskutiert.

## 2.4.Klausur

Am Ende des 1. Semesters findet eine Prüfung statt.

Zulassungsvoraussetzung ist die erfolgreiche Teilnahme an den Praktika, nachgewiesen durch die Testate.

Es ist eine **praktische** Prüfung am **PC**:

- Dauer 3 Zeitstunden
- jede(r) muss ein kleines Programm schreiben und zum Laufen bringen
- Aufgabenstellung definiert verschiedene Ausbaustufen des Programms zwecks Abbildung auf die Notenskala

Bewertung:

- nicht einfach "läuft / läuft nicht"
- saubere Programmstruktur ist wichtig!

## 2.5. Persönliche Selbstorganisation

- Selbstverantwortliches Lernen
  - Fragen stellen
  - Üben, üben, üben ...
  - Wissenslücken identifizieren und systematisch füllen
- Gruppenarbeit
  - muss allen Beteiligten nützen
  - nicht für Trittbrettfahrer
- Besuch der Vorlesungen
  - wird nicht kontrolliert
  - ist aber dringend zu empfehlen
- Sprachkenntnisse
  - Englisch unbedingt pflegen!
  - auch Originalliteratur lesen und Fachbegriffe lernen

### **2.5.1. Skript der Veranstaltung**

Ein Skript der Veranstaltung ist über meine Homepage abrufbar. Die Ausarbeitung basiert teilweise auf den Arbeiten des Kollegen Kreling.

Auf meiner Webseite sind auch die Praktikumsaufgaben veröffentlicht und aktuelle Informationen publiziert.

## 3. Die Sprache

### 3.1. Warum C++

Die Wahl der Programmiersprache für das Grundstudium wurde im Fachbereich damals kontrovers diskutiert:

- war bis 1995 Modula-2
- Eiffel ist elegant, aber ohne praktische Bedeutung
- Java war 1995 noch kein Thema

Als oberstes Ziel soll Objektorientierte Programmierung vermittelt werden.

C++ hat eine hohe Praxisrelevanz: C und C++ haben sich in der Praxis durchgesetzt.

C++ beinhaltet viele Konzepte, so dass einer Sprache ein großer Bereich aus der „Theorie und Konzepte von Programmiersprachen“ abdeckbar ist.

Aber:

- Im Hochschulbereich und in der Praxis lässt sich ein Trend zu Java beobachten.
- Java wird ab dem 3.Semester als zweite Programmiersprache behandelt.
- Ein mittelfristiger Wechsel von C++ zu Java als erste Programmiersprache wird im Fachbereich diskutiert. Dann wird aber C++ weiterhin behandelt.

## 3.2. Die Programmiersprache C

C als Systemprogrammiersprache ist

- mit Unix „groß geworden“,
- ist nicht herstellerspezifisch,
- wurde anfangs (Mitte 70er) kostenlos im Hochschulbereich verteilt:
  - kein Support, aber *hacking the kernel* für jeden
  - ganze Generation von Absolventen mit Unix und C aufgewachsen
  - diese sorgte später in der Industrie für Verbreitung

Es existieren Implementierungen

- zunächst im technisch-wissenschaftlichen Bereich (DEC PDP-11)
- dann im gesamten Workstation Sektor
- mittlerweile auf praktisch allen Plattformen von Mikro bis Mainframe

### 3.3.C++

C++ ist eine Obermenge von C:

- alle C++ Compiler können genauso gut C

Ein sanfter Übergang von C ist möglich:

- C  $\Rightarrow$  C++, prozedural  $\Rightarrow$  objektorientiert
- sichert Ausbildungsinvestments von Firmen

Oft wird aber C++ nur als „besseres C“ verwendet:

- man kann offiziell C++ programmieren
- Objektorientierung ist "in"
- und in Wirklichkeit ist ganz konventionelles C
- Umdenken ist nämlich gerade für alte Hasen manchmal schwierig

### 3.4. Zeittafel

1967	BCPL
1970	B
1972	erste C Implementierung implementiert bei AT&T, federführend: Dennis Ritchi
1978	Kernighan, Ritchie: <i>The C Programming Language</i> (First Edition)
1980	C mit Klassen implementiert bei AT&T
1983	Prägung des Namens: C++
1988	ANSI C Standard
1990	Ellis, Stroustrup: <i>The Annotated C++ Reference Manual</i>
1998	ISO / ANSI C++ Standard (ISO/IEC 14882:1998)
2003	ISO/IEC 14882:2003, Nachbesserung der 98er Norm
2010	C++10, kommende Version des Standardisierungskommitees mit besserer Unterstützung für Systemprogrammierung und Integration von boost.

## 4. Literatur

- 1) Breymann, C++, Einführung und professionelle Programmierung  
Hanser 2001  
(einfach geschrieben, sehr gut geeignet zur Nachbereitung der Vorlesung)
- 2) B. Eckel, Thinking in C++  
Prentice Hall 2000  
(auch als Online Version im Internet erhältlich)
- 3) Bjarne Stroustrup, Die C++ Programmiersprache  
Addison-Wesley  
(Original vom Erfinder der Programmiersprache; sehr tiefgehend; teilweise schwer zu lesen)