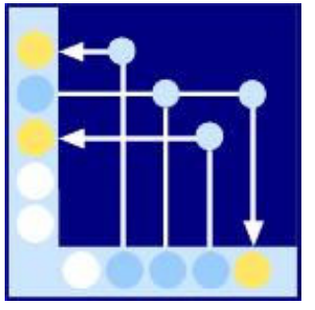
Hochschule Aalen
*Fakultät Elektronik und Informatik*
*Studienbereich Informatik*



# Programmieren in C++

Vorlesung im Wintersemester 2024/2025

*Prof. Dr. habil. Christian Heinlein*

christian.heinleins.net

# 1 Einleitung

## 1.1 Vorlesungsüberblick

### Ziele

- Vermittlung wesentlicher Konzepte von „modernem“ C++
- Schwerpunkt auf den „Besonderheiten“ der Sprache gegenüber anderen Programmiersprachen

## Form der Wissensvermittlung

- Vorlesung
  - Folien auf der Vorlesungs-Webseite
  - In der Regel kapitelweise, normalerweise vor Beginn des jeweiligen Kapitels
  - Bei Bedarf nachträgliche Ergänzungen oder Korrekturen
- Übung
  - Regelmäßige Aufgaben als Anregung zum eigenen Programmieren
  - Teilweise Besprechung in der Vorlesung
  - Musterlösung auf der Vorlesungs-Webseite
- Selbststudium

## Geplante Themen

- Grundlegende Datentypen, Operatoren und Anweisungen
- Klassen, einfache und mehrfache Vererbung, dynamisches Binden
- Konstruktoren, Destruktoren, Kopieren und Verschieben von Objekten
- Überladen von Funktionen und Operatoren
- Typ- und Funktionsschablonen (templates) inklusive Typkategorien (concepts)
- Funktionsobjekte, Lambda-Ausdrücke
- Container und Iteratoren
- Module

## Zugangsvoraussetzungen

- Zwingende Voraussetzung für die Zulassung zur Prüfung sind bestandene Prüfungen in Strukturierter und Objektorientierter Programmierung (Modul IN-57004 oder DS-43004).

# 1.2 Kurze Geschichte von C++

- 1979 Bjarne Stroustrup entwickelt „C with classes“
- 1984 Umbenennung in C++
- 1998 Erster Standard ISO/IEC 14882 (C++98)
- 2003 Überarbeitung/Korrektur des Standards von 1998 (C++03)
- 2011 Neuer Standard mit sehr vielen Erweiterungen
  (C++11, seit 2002 als C++0x bezeichnet, weil Fertigstellung vor 2010 erwartet wurde)
  Ab hier spricht man üblicherweise von „modernem“ C++.
- 2014 Kleine Erweiterung gegenüber C++11
  (C++14, zuvor als C++1y bezeichnet)
- 2017 Mittelgroße Erweiterung gegenüber C++14
  (C++17, zuvor als C++1z bezeichnet)
- 2020 Mittelgroße Erweiterung gegenüber C++17
  (C++20, zuvor als C++2a bezeichnet)
  Grundlage für diese Vorlesung
- 2023 Der nächste (noch nicht offiziell verabschiedete) Standard
  (C++23, vorläufig als C++2b bezeichnet)

# 1.3 Hallo, Welt!

```
// Definitionsdateien der C++-Standardbibliothek haben kein Suffix.
#include <iostream>

// Namensbereich std einbinden,
// in dem sich alle Namen der C++-Standardbibliothek befinden.
using namespace std;

// main hat Resultattyp int.
// Der Resultatwert wird als Exitstatus des Programms
// an das Betriebssystem zurückgeliefert.
// Parameter argc (Anzahl der Kommandozeilenargumente)
// und argv (Reihe mit den Kommandozeilenargumenten)
// sind optional.
int main (int argc, char* argv []) {
  // cout ist der Standardausgabestrom.
  // endl ist ein Zeilentrenner.
  cout << "Hallo, Welt!" << endl;
}
// Wenn main keine return-Anweisung ausführt, ist der Exitstatus 0.
// (Für andere Funktionen mit Resultattyp ungleich void wäre das
// Verhalten undefiniert.)
```

# 1.4 Bekannte C++-Übersetzer

## 1.4.1 GCC (GNU Compiler Collection)

- Je nach Version, werden unterschiedliche C++-Standards unterstützt.
- Seit Version 12 wird C++20 größtenteils unterstützt.
  (Vor allem die Unterstützung für Module ist noch unvollständig.)
- Seit Version 15 werden auch bereits große Teile von C++23 unterstützt.
- Siehe auch https://gcc.gnu.org/projects/cxx-status.html

## 1.4.2 Clang (Frontend für LLVM)

- Auch hier werden ja nach Version unterschiedliche C++-Standards unterstützt.
- Seit Version 19 wird C++20 größtenteils unterstützt.
  (Auch hier ist vor allem die Unterstützung für Module noch unvollständig.)
- Seit Version 19 werden auch bereits große Teile von C++23 unterstützt.
- Siehe auch https://clang.llvm.org/cxx_status.html

## 1.4.3 Wichtige Aufrufoptionen

- `-std=c++{98,03,11,14,17,20,2b}`
  Auswahl der Sprachversion

- `-c`
  Quelldatei(en) nur übersetzen, aber nicht zu einem ausführbaren Programm zusammenbinden

- `-o` *`program`*
  Ausführbares Programm mit dem Namen *`program`* statt `a.out` erzeugen

- `-O{0,1,2,3}`
  Zielcode mehr oder weniger stark optimieren

- `-g`
  Informationen für Debugger (z. B. GDB) in das übersetzte Programm integrieren

- `-D` *`name`* `[`*`definition`*`]`
  Quasi `#define` *`name definition`* bzw. `#define` *`name`* `1` (wenn *`definition`* nicht angegeben ist) von der Kommandozeile aus

- `-I` *`directory`*
  `#include`-Dateien zusätzlich im Verzeichnis *`directory`* suchen

- `-L` *`directory`*
  Bibliotheken zusätzlich im Verzeichnis *`directory`* suchen

- `-l`*`library`*
  Bibliothek mit dem Namen *`library`* zum Programm hinzufügen

# 1.5 Literaturhinweise


- B. Stroustrup: *Die C++-Programmiersprache* (Aktuell zum C++11-Standard). Carl Hanser Verlag, München, 2015.
  (Korrektes Deutsch wäre wohl eher: *Die Programmiersprache C++*)
- S. Meyers: *Effective Modern C++*. O'Reilly, Sebastopol, CA, 2014.
- ISO-C++-Standards 1998, 2003, 2011, 2014, 2017, 2020
  (bzw. die entsprechenden Entwurfsdokumente)
- http:// www.cppreference.com
- http://www.cplusplus.com/reference
- https://de.wikipedia.org
- https://en.wikipedia.org