FH MÜNSTER
University of Applied Sciences

Programmieren in C++

Teil 1 – Einführung

Prof. Dr. Kathrin Ungru
Fachbereich Elektrotechnik und Informatik

kathrin.ungru@fh-muenster.de

Grundlagen
Inhalt

- Von C nach C++
- Der C++ Standard
- Wo und wie häufig wird C++ genutzt?
- Wann ist C++ Sinnvoll?
- Programmieren mit C++: Erste Schritte (etwas Wiederholung)

Programmiersprachen
Einordnung

Machinensprachen
bestehen aus 1en und 0en und werden definiert durch Hardwarearchitektur

Assembly
elementare Operationen werden durch Wörter ersetzt und mit Hilfe des Assembler in Maschinensprache übersetzt

High-Level Sprachen
wie C, C++, Java, C#, Swift and Visual Basic

Programmiersprachen
Einordnung

High-Level Sprachen
wie C, C++, Java, C#, Swift und Visual Basic, …

- **Angelehnt an gesprochene Sprache zur besseren Lesbarkeit durch Programmierer**
- **Enthalten außerdem häufig genutzte mathematische Operationen und Symbole**
- **Kompilieren: Übersetzung der High-Level Sprache in Maschinensprache**

Spezielle High-Level Sprachen sind Interpreter-Sprachen:

- Direkte Übersetzung „Interpretation“ ohne Kompilieren
- Auch Skriptsprachen, wie JavaScript und Python sind Interpreter-Sprachen

Progammiersprachen
Objektorientierte Programmierung

Was ist objektorientierte Programmierung?

- Ziel von OOP ist zunächst die Programmierung von Software näher an die tägliche Erfahrung des Menschen heranzubringen.
- Objekte aus der realen Welt werden in einem objektorientierten Programm abgebildet.
- Die Abbildungen im Programm sind wiederum Objekte.

Beispiel: Ein Quadrat, welches von einem Programm auf dem Bildschirm gezeichnet wird, wird auch als Quadrat programmiert.

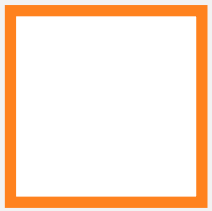
Progammiersprachen
Objektorientierte Programmierung

Prozedurale Programmierung:

Ein Programm besteht aus der Reihe nach ausgeführte *Anweisungen*. (Ähnlich einem Kochrezept)

Ein globaler Anfangszustand des Programmes wird dabei in einen globalen Endzustand überführt.

*Beispiel: Ein Programm besteht aus einer **Anweisungsabfolge** zum zeichnen eines Quadrates.*



Zeichne Quadrat einer Farbe und Größe:

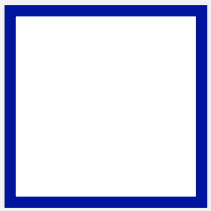
1. *Berechne nacheinander die 4 Ecken des Quadrats einer bestimmten Größe*
2. *zeichne Linien einer bestimmten Farbe zwischen den Ecken*

Objektorientierten Programmierung:

Die Grundlage der Programmierung bilden *Objekte*, die bestimmte Eigenschaften und Fähigkeiten haben.

Jedes Objekt hat einen eigenen (lokalen) Anfangs- und Endzustand.

*Beispiel: In einem Programm gibt es ein **Objekt** namens Quadrat mit folgendem Aufbau:*



Quadrat:

Eigenschaften:
- Farbe
- Größe

Fähigkeiten:
- zeichne

Progammiersprachen
Objektorientierte Programmierung

Was sind die Vorteile von OOP?

(+) Bessere **Lesbarkeit**, da Programm näher an der Realität.

(+) **Kapselung** von Daten und somit eine höhere Sicherheit von Software.

(+) **Wiederverwendbarkeit** von Software und Modularisierung von Programmen

→ OOP führt dazu, dass Objekte einfacher ausgetauscht oder um bestimmte Eigenschaften erweitert werden können.

→ Eigenschaften von Objekten können an ähnliche Objekte weitergegeben (vererbt) werden.

👍 Vor allem bei großen Projekten spart dies Zeit und Geld (und den Entwickelnden Nerven).

Und was sind die Nachteile von OOP?

(-) Es muss erst einiges an Aufwand in Software-Architektur und Planung gesteckt werden.

→ Bei kleinen Projekten nicht unbedingt sinnvoll. (Kosten / Nutzen abwägen!)

→ Einzelne Algorithmen (z.B. Sortierungs-, Such- oder Optimierungsalgorithmen oder auch numerische Berechnungen), die auf Effizienz getrimmt werden, sollten besser prozedural und nicht objektorientiert programmiert werden!

Progammiersprachen
Objektorientierte Programmierung

Was sind Objektorientierte Programmiersprachen?

- Die grundsätzlichen Ideen der OOP können auch mit nicht-objektorientieren Programmiersprachen umgesetzt werden.
  Beispiel: „objektorientierte Programmierung“ in C mit `struct` *als Ersatz für Klassen*.

- Objektorientierte Programmiersprachen unterstützen jedoch das Konzept indem Sie Strukturen und Mechanismen wie **Klassen** und **Polymorphie** anbieten, mit denen das Konzept der objektorientierten Programmierung leichter umgesetzt werden kann.

Programmiersprachen

PYPL Ranking

- The PYPL **PopularitY of Programming Language** Index basierend auf der Häufigkeit der Suche nach Programmier-Tutorials auf Google.
- https://pypl.github.io

Worldwide, Mar 2023 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.91 %	-0.6 %
2		Java	16.58 %	-1.6 %
3		JavaScript	9.67 %	+0.6 %
4		C/C++	6.93 %	-0.5 %
5		C#	6.88 %	-0.5 %
6		PHP	5.19 %	-0.6 %
7		R	4.23 %	-0.2 %
8	↑	TypeScript	2.81 %	+0.6 %
9	↑	Swift	2.28 %	+0.2 %
10	↓↓	Objective-C	2.26 %	+0.0 %
11	↑↑↑	Rust	2.03 %	+1.0 %
12	↑	Go	1.93 %	+0.7 %
13	↓	Kotlin	1.82 %	+0.2 %
14	↓↓↓	Matlab	1.66 %	-0.3 %
15	↑	Ruby	1.1 %	+0.3 %
16	↑↑	Ada	1.01 %	+0.4 %
17	↓↓	VBA	1.01 %	+0.1 %
18	↑↑	Dart	0.81 %	+0.3 %
19	↓↓	Scala	0.62 %	+0.0 %
20	↑	Lua	0.58 %	+0.2 %

Programmiersprachen
und Energieeffizienz

FH MÜNSTER
University of Applied Sciences

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2017, October). Energy efficiency across programming languages: how do energy, time, and memory relate?. In *Proceedings of the 10th ACM SIGPLAN international conference on software language engineering* (pp. 256-267).

Objektorientierte Programmiersprachen

C++ vs. Java

C++	Java
Keine Garbage Collection	Garbage Collection
Plattform spezifischer Compiler und C++ Umgebung: Programm kompilieren auf jeder Plattform (Für die ein Compiler zur Verfügung steht)	JRE (Java Runtime Engine): Programm ausführen auf jeder Plattform (die Java unterstützt)
Es existieren verschiedene Implementierungen des Standards: gcc, clang, etc.	Viele Implementierungen mit einem Zoo an Lizensen
i.d.R. schneller, da für jede Architektur speziell kompiliert	i.d.R langsamer, da JRE zwischen Programm und Maschine liegt

Die Programmiersprache C++

Warum C++?

- C++ wurde 1979 von **Bjarne Stroustrup** in den Bell Laboratorien als Erweiterung der Programmiersprache C entwickelt.
- Ursprünglich hieß die Programmiersprache “C with Classes“ und wurde Anfang der 80er in **C++** umbenannt.
- **C++** ermöglicht sowohl die **effiziente** und **maschinennahe Programmierung** als auch eine Programmierung auf hohem Abstraktionsniveau.
- Neben erweiterten Funktionalitäten im Vergleich zu C bietet C++ eine Unterstützung für das Konzept der **Objektorientierten Programmierung**.
- Ein weiterer wichtiger Baustein ist die generische Programmierung mit Hilfe von **Templates**.
- Der Standard definiert auch eine umfangreiche **Standardbibliothek**, zu der verschiedene Implementierungen existieren.

Die Programmiersprache C++

Warum C++?

- Betriebssysteme und Treiber werden u.a. in C/C++ geschrieben.
- Zugriff auf Gerätetreiber erfolgt daher häufig über C/C++.
- Mikrocontroller können mit C/C++ programmiert werden.
- Viele Bibliotheken sind in C/C++ geschrieben und sehr effizient, wenn Sie aus einem C/C++ Programm aufgerufen werden:

  Beispiele:

  Bibliothekssammlung u.a. effiziente Algorithmen, Mathematik, GPGPU

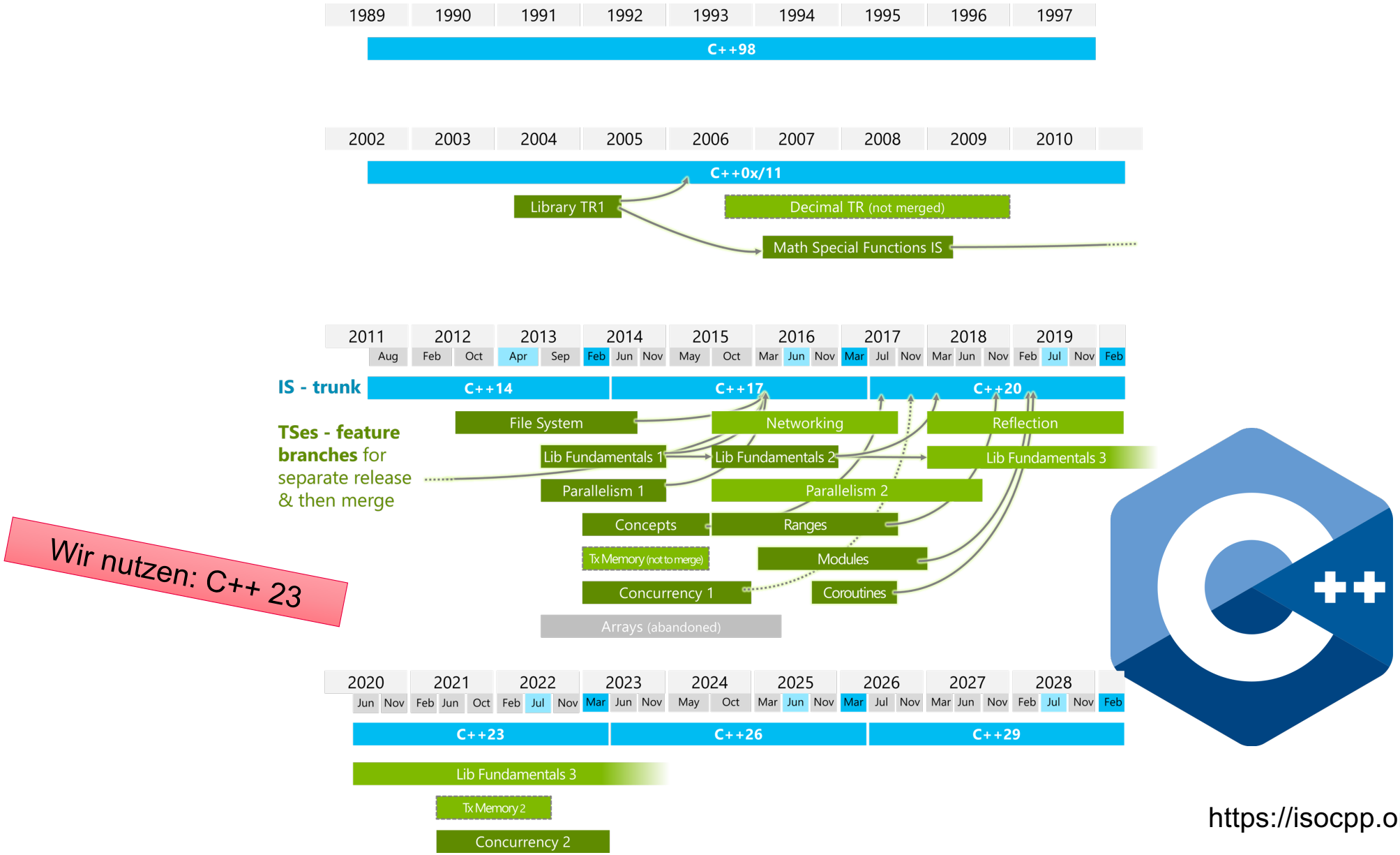
  GPGPU Programmierung

  Cryptograpie

  GPGPU Programmierung

  GUI Programmierung

- Andersherum stellen viele High-Level Programmiersprachen wie Python oder Matlab Schnittstellen bereit, um Laufzeitrelevante Codeteile in C++ auszulagern.



Wir nutzen: C++ 23

https://isocpp.org/

Die Programmiersprache C++
Referenzen

ISO-Standard:

- ISO/IEC 14882:2017

Web:

- https://isocpp.org/ (Keine Referenz, aber Infos rund um den Standard)
- https://en.cppreference.com (!)
- https://www.cplusplus.com

Ein erstes C++ Programm

C++ vs. C

FH MÜNSTER
University of Applied Sciences

C

```c
// Headerdatei der Standard Bibliothek
#include <stdio.h>


/* main()-Funktion als
Programm Einstiegspunkt */
int main() {
    /* Hello World ausgeben  */
  printf("Hello, World!");
  /* Beenden des Programms mit Rückgabewert 0 */
  return 0;
}
```

Ein erstes C++ Programm

C vs. C++

FH MÜNSTER
University of Applied Sciences

C

```c
// Headerdatei der Standardbibliothek
#include <stdio.h>


/* main()-Funktion als
Programm Einstiegspunkt */
int main() {
  /* Hello World ausgeben  */
  printf("Hello, World!");
  /* Beenden des Programms mit Rückgabewert 0 */
  return 0;
}
```

C++

```cpp
// Headerdatei der Standardbibliothek
#include <iostream>


/* main()-Funktion als
Programm Einstiegspunkt */
int main() {
  // Hello World ausgeben
  std::cout << "Hello, World!";
  // Beenden des Programms mit Rückgabewert 0
  return 0;
}
```

Ein erstes C++ Programm

“Hello World"

FH MÜNSTER
University of Applied Sciences

Zeile 2: Eine Headerdatei der C++ Standard Bibliothek, hier **`<iostream>`**, wird mit **`#include`** eingefügt.

Zeile 5: Die **`main()`**-Funktion mit Rückgabewert **`int`** ist der Einstiegspunkt für jedes C++ Programm.

Zeile 8: Mit **`std::cout << "Hello, World!";`** wird der Text „Hello World“ ausgegeben. **`std::cout`** (=character ouput) ist ein Objekt aus der Headerdatei **`<iostream>`** und empfängt den Output-Stream ähnlich wie **`stdout`** in C.

C++

```
1  // Headerdatei der Standardbibliothek
2  #include <iostream>
3
4  /* main()-Funktion als
5  Programm Einstiegspunkt */
6  int main() {
7    // Hello World ausgeben
8    std::cout << "Hello, World!";
9    // Beenden des Programms mit Rückgabewert 0
10   return 0;
11 }
```

Ein erstes C++ Programm

"Hello, World"

FH MÜNSTER
University of Applied Sciences

Zeile 3: Besagt, dass der Namensraum `std` genutzt wird. Im Code kann `std::` weggelassen werden (später hierzu mehr).

C++

```
1  // Headerdatei der Standardbibliothek
2  #include <iostream>
3  using namespace std;
4  /* main()-Funktion als
5  Programm Einstiegspunkt */
6  int main() {
7    // Hello World ausgeben
8    cout << "Hello, World!";
9    // Beenden des Programms mit Rückgabewert 0
10   return 0;
11 }
```

Ein erstes C++ Programm

"Hello, World!"

FH MÜNSTER
University of Applied Sciences

//	Kommentar bis Zeilenende
/* ... */	Kommentar über mehrere Zeilen
{ }	Block
()	Innerhalb von Klammern können Informationen z.B. an Funktionen übergeben werden.
;	Ende einer Anweisung
#	Steht am Zeilenanfang einer Compilerdirektive, die eine Anweisung an den Präprozessor darstellt

C++

```
1  // Headerdatei der Standardbibliothek
2  #include <iostream>
3  using namespace std;
4  /* main()-Funktion als
5  Programm Einstiegspunkt */
6  int main() {
7    // Hello World ausgeben
8    cout << "Hello, World!";
9    // Beenden des Programms mit Rückgabewert 0
10   return 0;
11 }
```

Erstellen eines C++ Programms

Dateiendungen

Quelldateien

`.C, .cc, .cpp, .cxx, .c++`

Headerdateien

`.h, .hh, .hpp, .hxx, .h++`

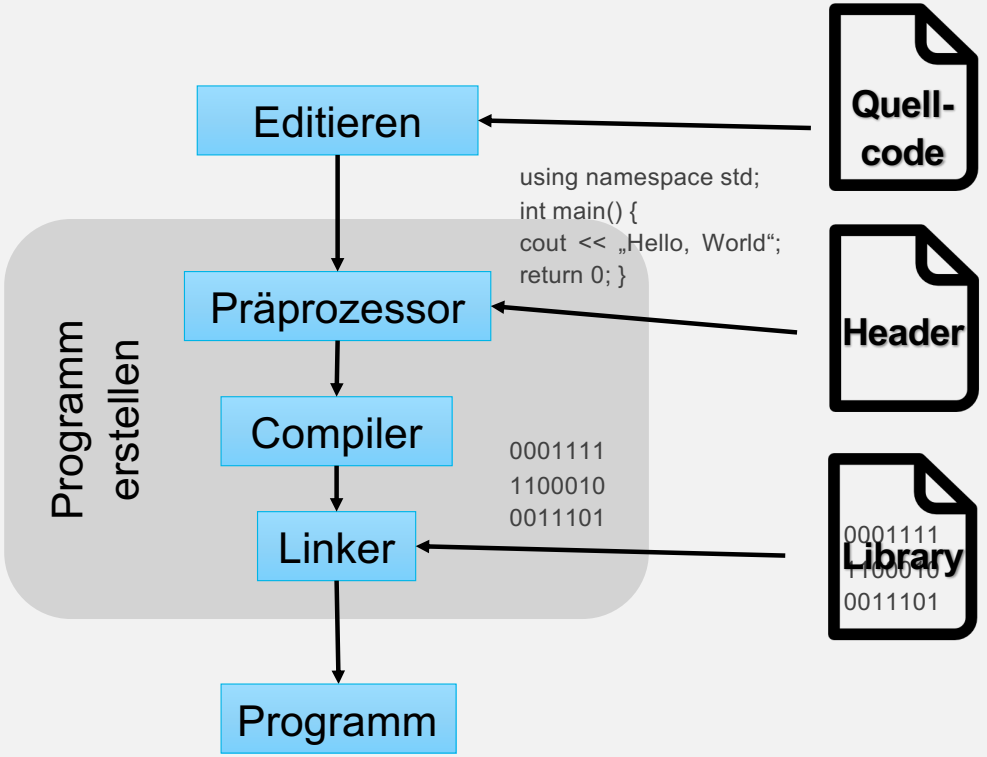
Diese Veranstaltung: `.cpp` für Quell- und `.hpp` für Headerdateien

Hinweis: Standardheader wie `iostream`, tragen keine Dateiendung.

Erstellen eines C++ Programms

Ablauf

```
1  // Headerdatei der Standard Bibliothek
2  #include <iostream>
3  using namespace std;
4  /* main()-Funktion als
5  Programm Einstiegspunkt */
6  int main() {
7    // Hello World ausgeben
8    cout << "Hello, World";
9    // Beenden des Programms mit Rückgabewert 0
10   return 0;
11 }
```

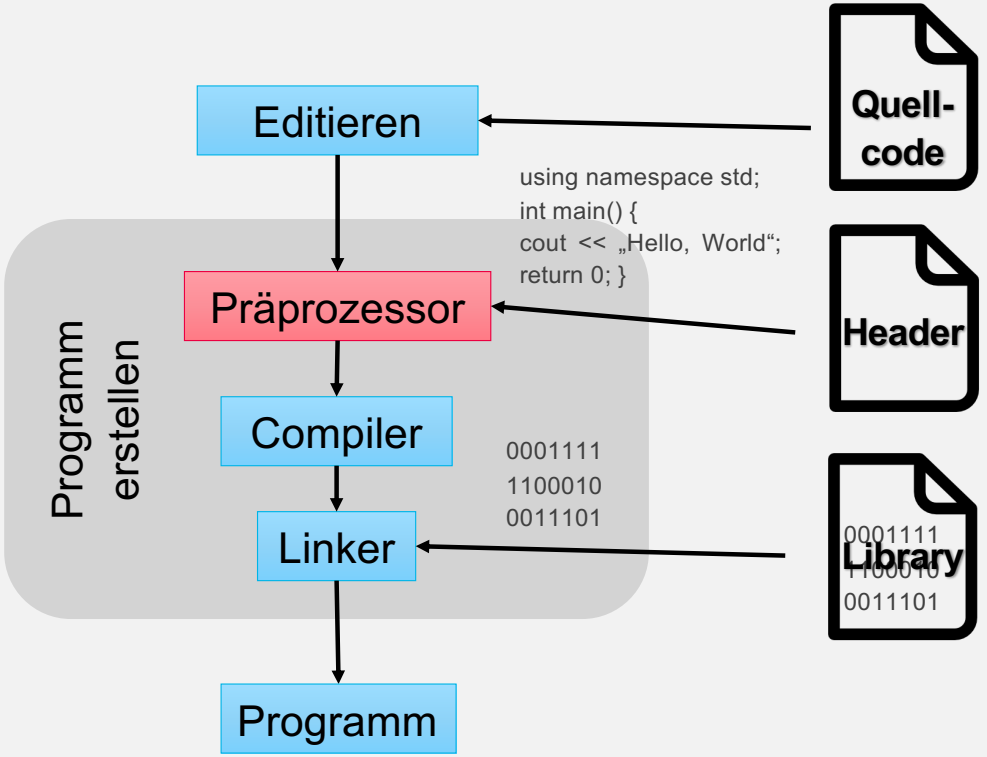


Erstellen eines C++ Programms

Präprozessor

Der **Präprozessor** editiert automatisch den Quellcode mit Hilfe sogenannter Compilerdirektiven. Dies sind Anweisungen, die im Präprozessor vor der Kompilierung ausgeführt werden.

Beispiel:
Die Anweisung `#include <iostream>` *fügt Quellcode aus der Standard-headerdatei* `iostream` *ein. In der Headerdatei befindet sich die Definition von* `cout <<`.

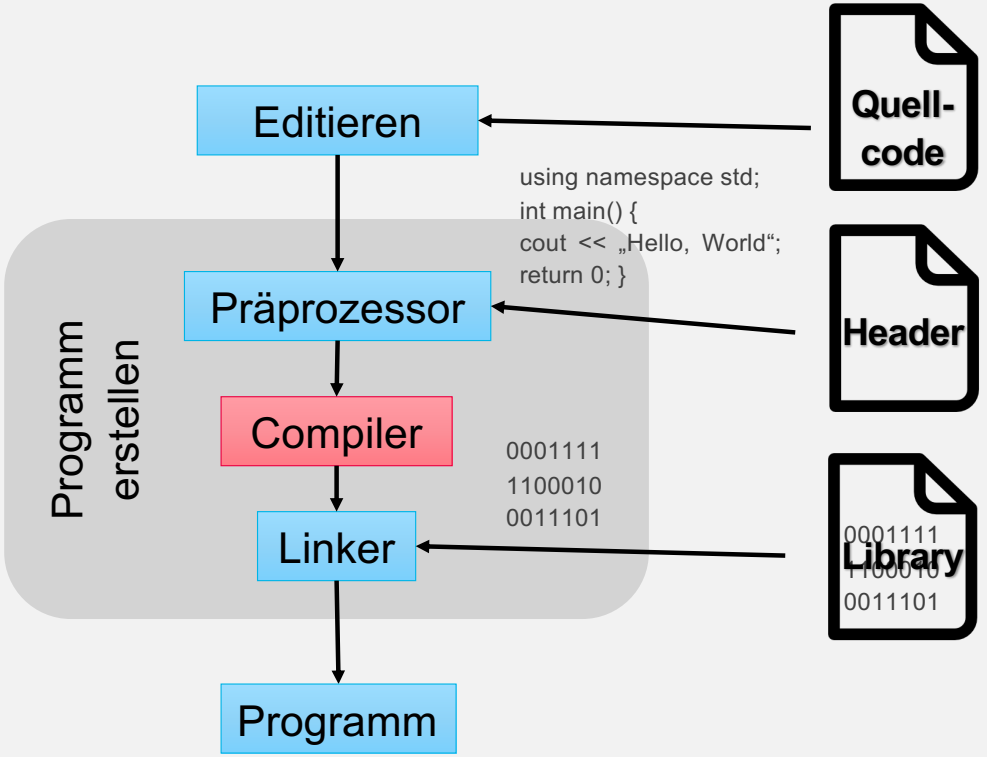


Erstellen eines C++ Programms

Compiler

Der **Compiler** erzeugt aus Quellcode Maschinencode der von der CPU verarbeitet werden kann. Der Maschinencode wird für jede Quelldatei in einer Objektdatei mit binärem Inhalt gespeichert. Sollte der Quellcode Fehler enthalten gibt der Compiler Warnungen oder Fehlermeldungen aus.

Beispiel:
Die Quelldatei `main.cpp` *wird in die Objektdatei* `main.o` *übersetzt.*

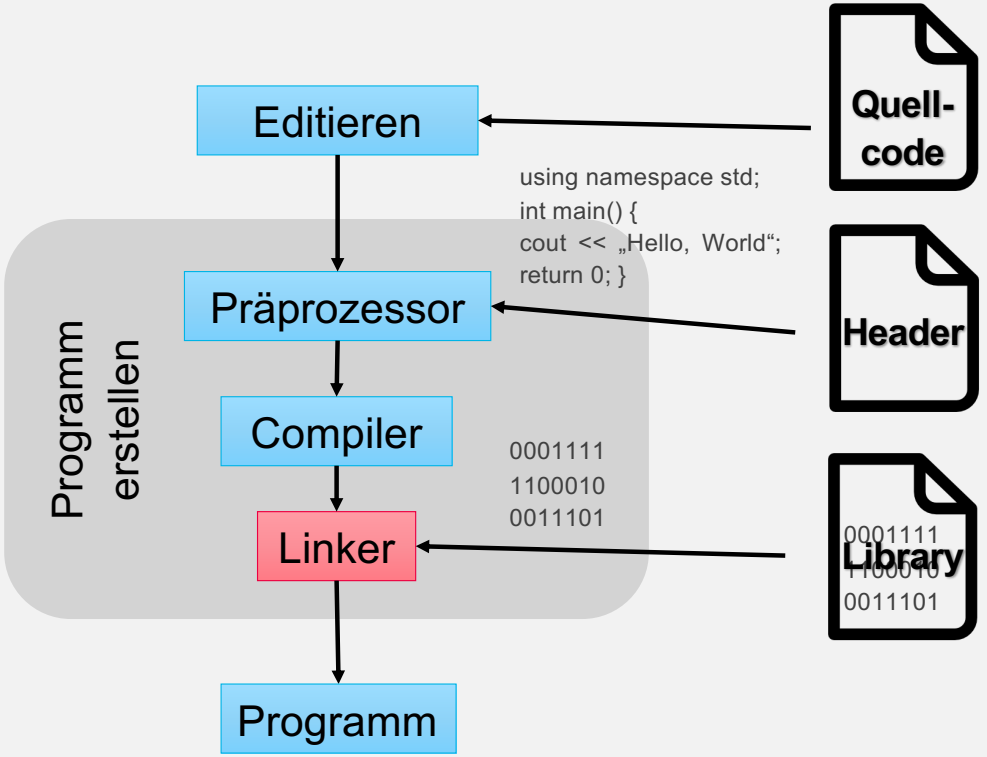


Erstellen eines C++ Programms

Linker

Der **Linker** verbindet die binären Objektdateien aus dem Compiler mit binären Modulen aus Bibliotheken (Libraries).

Beispiel:
Das Modul aus der Standardbibliothek, welches die Deklaration von `cout <<` *beinhaltet, wird mit der Objektdatei main.o verknüpft. Erst jetzt ist das Programm lauffähig.*



Erstellen eines C++ Programms

Beispiel – GNU Compiler Collection (GCC)

FH MÜNSTER
University of Applied Sciences

Aufrufe in der Kommandozeile:

Kompilieren	`g++ -c main.cpp # erzeugt die Objektdatei main.o`
Linken	`g++ main.o # erzeugt das ausführbare Programm a.out`
Kompilieren & Linken	`g++ main.cpp # erzeugt a.out in nur einem Schritt`

Kompilieren & Linken in C++ 23:
`g++ -std=c++23 main.cpp`

Auf Windows Systemen wird GCC via MinGW oder Cygwin installiert.

Erstellen eines C++ Programms

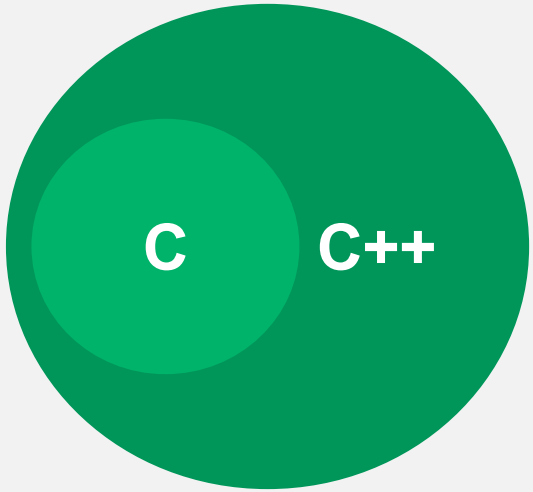
Einige Compiler

Compiler-Paket	Programmiersprachen	Betriebssysteme	Plattformen	Mehr Details
GCC	C, C++, Objective-C, Fortran, Ada, Go, D	Linux, macOS, Windows, …	x86, x64, PowerPC, ARM, u.v.m.	gcc.gnu.org
Clang	C, C++, Objective-C, Objective-C++	Linux, macOS, Windows, …	LLVM (Low Level Virtual Machine)	clang.llvm.org
Visual C++	C, C++	Windows	x86, x64, ARM	
Intel C++	C, C++	Linux, macOS, Windows, …	x86, x64, IA-64	Für Intel und AMD Prozessoren, sehr optimiert

Stand: September 2020

Sprachmerkmale
C++ vs. C

- C++ ist eine Erweiterung von C und ist somit weitgehend mit C kompatibel.



➢ Dies bedeutet allerdings nicht, dass jedes C Programm ohne Änderung direkt in C++ kompiliert werden kann!

Sprachmerkmale
C++ vs. C

- **Über C hinaus bietet C++**
  - Weitere Datentypen und Typumwandlungsmöglichkeiten
  - Klassen mit Mehrfachvererbung und virtuellen Funktionen
  - Exception-Handling (Ausnahmebehandlung)
  - Templates (Schablonen)
  - Namepaces (Namensräume)
  - Inline-Funktionen
  - Überladung von Operatoren und Funktionen
  - Referenzen
  - Operatoren zur Verwaltung des dynamischen Speichers
  - Die C++-Standardbibliothek

FH MÜNSTER
University of Applied Sciences

Programmieren in C++

Prof. Dr. Kathrin Ungru
Fachbereich Elektrotechnik und Informatik

kathrin.ungru@fh-muenster.de