

Grundlagen der Informatik und Programmierung 2

Einführung

Was ist in C++ anders als in C?

Prof. Dr. Tom Vierjahn

Visual Computing (<https://vc.w-hs.de>)
Fachbereich Wirtschaft und Informationstechnik
Campus Bocholt

Sommersemester 2020



Veröffentlicht unter der Creative-Commons-Lizenz
Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

Dateiversion 9. April 2020, 23:28 — Seite 1

Dateinamen

Dateiendungen:

- ▶ Quelltext: .cpp
- ▶ Header: .hpp

Beispiel-Projektmappe (CMakeLists.txt):

```
add_executable(main
    main.cpp
)
```

Ausgabe

main.cpp

```
#include <iostream>

int main(int argc, char** argv) {
    std::cout << "G'day!" << std::endl;

    return EXIT_SUCCESS;
}
```

main.cpp

```
#include <iostream>

int main(int argc, char** argv) {
    int i = 0;
    std::cout << "Enter integer: ";
    std::cin >> i;
    std::cout << "integer: " << i << std::endl;

    float f = 0.0;
    std::cout << "Enter float: ";
    std::cin >> f;
    std::cout << "float: " << f << std::endl;

    return EXIT_SUCCESS;
}
```

main.cpp

```
#include <iostream>
#include <string>

int main(int argc, char** argv) {
    std::string greeting("G'day!");

    std::cout << greeting << std::endl;

    return EXIT_SUCCESS;
}
```

„klassische“ Initialisierung

```
int a = 5;
int b[3] = {1, 2, 3};
std::string c = "foo";
std::string d("bar");
```

Uniform Initialization mittels Braced-Init-Lists

```
int a{5};
int b[3]{1, 2, 3};
std::string c{"foo"};
std::string d{"bar"};
```

- ▶ C++11 bietet eine vereinheitlichte Schreibweise, um Variablen zu initialisieren.
- ▶ flexibler
- ▶ ggf. effizienter

Schlüsselwörter struct und enum

Datenstruktur (C)

```
struct DataStructure {  
    int i;  
    float f;  
};  
struct DataStructure data;
```

Datenstruktur (C++)

```
struct DataStructure {  
    int i;  
    float f;  
};  
DataStructure data;
```

Enumeration (C)

```
enum Enumeration {  
    kOne,  
    kTwo,  
};  
enum Enumeration mode;
```

Enumeration (C++)

```
enum Enumeration {  
    kOne,  
    kTwo,  
};  
Enumeration mode;
```

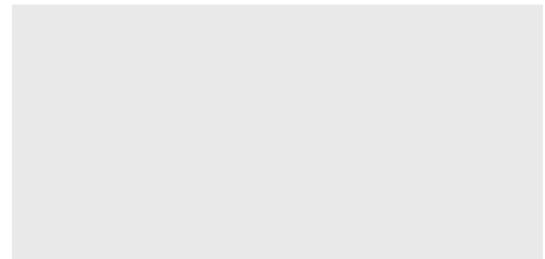
Datentyp bool

Code

```
bool a = true;
bool b = false;
bool c = !b;
bool d = 3 > 2;
bool e = 7;

std::cout << "a : " << a << std::endl;
std::cout << "b : " << b << std::endl;
std::cout << "c : " << c << std::endl;
std::cout << "d : " << d << std::endl;
std::cout << "e : " << e << std::endl;
```

Ausgabe



- ▶ C++ hat eigenen Datentyp bool für die Werte true und false.

Makro-Konstanten (C)

```
#define NUM_ITEMS 5
int array[NUM_ITEMS];
```

„echte“ Konstanten (C++)

```
const unsigned int num_items = 5;
int array[num_items];
```

Funktionsdefinition

```
void print_fraction(int numerator, int denominator = 1) {  
    std::cout << "(" << numerator << "/" << denominator << ")" << std::endl;  
}
```

Aufruf

```
print_fraction(3, 5);  
print_fraction(17, 1);  
print_fraction(42);
```

Ausgabe

- ▶ Funktionsparametern können Default-Werte zugewiesen werden.
- ▶ Diese werden verwendet für beim Aufruf nicht angegebene Parameterwerte.
- ▶ Default-Parameter stehen immer am Ende der Parameterliste.
- ▶ Parameter können beim Aufruf nur vom Ende an weggelassen werden.
- ▶ Ist ein Funktionsprototyp vorhanden, stehen Default-Werte nur hier.

- ▶ Ein- und Ausgabe (mehr folgt)
- ▶ Zeichenketten
- ▶ Uniform Initialization
- ▶ struct, enum
- ▶ bool
- ▶ Konstanten
- ▶ Default-Werte für Funktionsparameter

Prof. Dr. Tom Vierjahn

- ▶ E-Mail: tom.vierjahn@w-hs.de

Visual Computing

- ▶ Web: <https://vc.w-hs.de>
- ▶ YouTube: Visual Computing WH
- ▶ Twitter: @VisComputingWH

Westfälische Hochschule
Fachbereich Wirtschaft und Informationstechnik
Campus Bocholt



Veröffentlicht unter der Creative-Commons-Lizenz
Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)