

PROGRAMMIEREN II

DHBW Stuttgart Campus Horb INF2017

ALBERNES ZAUBERSTABGEFUCHTEL
UND KINDISCHE HEXEREIEN WIRD ES
HIER NICHT GEBEN!



DER DOZENT STELLT SICH VOR

- Torsten Hopf (26)
- B.Sc. an der DHBW (INF2013)
- Senior Consultant und Architekt bei MHP Management- und IT-Beratung GmbH
- C++ Erfahrung im Großbankenbereich



DIE STUDENTEN STELLEN SICH VOR

- Name
- Firma
- Programmiererfahrungen
- Erwartungen an die VL
- Einen Fakt den die Kommilitonen nicht über dich wissen



ALLGEMEINE REGELN

- Mitarbeit ist Pflicht
- Wir versuchen hier Spaß zu haben
- Respektvoller Umgang
- Wer Fragen hat, einfach raus damit
- Computer nur während der Coding-Sessions

ORGANISATORISCHES

- Prüfungsleistung
 - Abgabe einer Projektarbeit (2-3 Personen)
 - Bonus: Mitarbeit in der VL

ABLAUF DER STUNDEN

- Vorlesung und Programmieren im Wechsel
- Zum Schluss der Stunde wird eine kleine Aufgabe für die nächste Woche gestellt
- Diese wird bei Start der Vorlesung vorgestellt und Fragen beantwortet

ZIELE DER VORLESUNG

- Einstieg in die Programmiersprache C++
- Verstehen welche Anwendungsgebiete C++ besitzt
- Unterschiede zwischen Java und C++
- Fortgeschrittene Kenntnisse in der Programmierung vermitteln (Sprachunabhängig)

NICHT BESTANDTEIL DER ZIELE

- Wirklich jede Perversität von C++ kennenlernen
- C (ohne ++) bis in die Tiefe analysieren

GESCHICHTE



C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, it blows away your whole leg.

— *Bjarne Stroustrup* —

AZ QUOTES

GESCHICHTE

- Erscheinungsjahr 1985
- Entstand ursprünglich als „C with classes“
- Im Vergleich zu C ein Gewinn an Komfort
- Standardisierung in 98
- Ab 2011 (C++11) völlige Überarbeitung der Sprache

EINSATZGEBIETE

- Hardwarenahe Anwendungsfälle
- Betriebssysteme
- Netzwerktechnik und Protokolle
- Compiler


HELLO WORLD IN C(++)

```
/**
 * Like a java import, just more mighty. More coming soon...
 */
#include <iostream>

// basic main method
int main()
{
    // C way of saying hello
    // %s means that you can put in there every string
    printf("Hello Mr %s ", "CppDev");
    return 0;
}
```

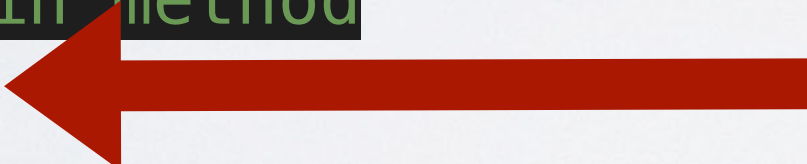
HELLO WORLD IN C(++)

```
/**  
 * Like a java import, just more mighty. More comming soon...  
 */  
#include <iostream>  
  
// basic main method  
int main()  
{  
    // C way of saying hello  
    // %s means that you can put in there every string  
    printf("Hello Mr %s ", "CppDev");  
    return 0;  
}
```



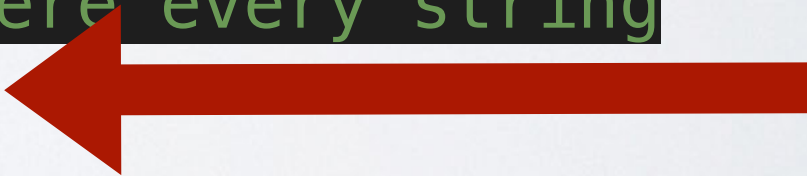
HELLO WORLD IN C(++)

```
/**  
 * Like a java import, just more mighty. More comming soon...  
 */  
#include <iostream>  
  
// basic main method  
int main()  
{  
    // C way of saying hello  
    // %s means that you can put in there every string  
    printf("Hello Mr %s ", "CppDev");  
    return 0;  
}
```




HELLO WORLD IN C(++)

```
/**  
 * Like a java import, just more mighty. More comming soon...  
 */  
#include <iostream>  
  
// basic main method  
int main()  
{  
    // C way of saying hello  
    // %s means that you can put in there every string  
    printf("Hello Mr %s ", "CppDev");  
    return 0;  
}
```



HELLO WORLD IN C(++)

```
/**  
 * Like a java import, just more mighty. More coming soon...  
 */  
#include <iostream>  
  
// basic main method  
int main()  
{  
    // C way of saying hello  
    // %s means that you can put in there every string  
    printf("Hello Mr %s ", "CppDev");  
    return 0;  
}
```



WAS HABEN WIR JETZT SCHON GELEHRT

- `#include` um auf andere Libraries zugreifen zu können
- Die `main`-Methode gibt eine Ganzzahl zurück (0=OK, alles andere Fehler)
- Im Gegensatz zu Java können wir auch ohne Klassen programmieren
- Statische Methoden können ohne `Keyword` verwendet werden
- Gültiger C Code ist immer auch gültigen C++ Code

C++ STANDARDTYPEN

Datentyp	Größe (Bit)	Beispiel
bool	Compilerabhängig	true
char	(Mindestens) 8	's'
int	(Mindestens) 16	1337
short	(Mindestens) 16	456
long	(Mindestens) 32	133742
long long	(Mindestens) 64	133713371337
float	32	1.234
double	64	1.2342353443534534

BEDINGUNGEN

```
if (guest.compare(name) == 0)
```

BEDINGUNGEN

- Wie in Java mit Booleschen-Ausdrücken

SCHLEIFEN I

```
// Good new way to create an array of five strings
std::array<std::string, 5> guests = {"Paul", "Ralf", "Olaf", "Toni", "Karl"};

// classical for loop
for (int i = 0; i < guests.size(); ++i)
{
    // std::cout is a stream which directly leads to the standard out stream of the terminal
    // '<<' is the overloaded operator to write something in the stream
    std::cout << "Hello Mr. " << guests[i] << std::endl
               << "\t Your seat is number " << i << std::endl;
}
```


SCHLEIFEN II

```
for (std::string &guest : guests)
{
    if (guest.compare(name) == 0)
    {
        std::cout << "The guest " << name << " has already arrived" << std::endl;
        return;
    }
}
```



SCHLEIFEN II

```
for (std::string &guest : guests)
{
    if (guest.compare(name) == 0)
    {
        std::cout << "The guest " << name << " has already arrived" << std::endl;
        return;
    }
}
```

SCHLEIFEN III

```
while (int selection = ShowMenuSelect())  
{  
      
}
```

•

SCHLEIFEN IV

```
do{  
    // ... cool stuff  
}while(true == 1);
```



ZUSAMMENGEFASST

- Klassische Schleifen in C++ funktionieren wie in Java, oder eher andersherum :-)
- For-Each-Schleifen funktionieren wie in Java nur müssen wir auf Pass-By-Value und Pass-By-Reference achten

MOMENT! PASS-BY-REFERENCE

- & Operator zeigt an das eine Referenz übergeben werden soll
- Standardmäßig: Kopie (Unterschied zu Java!!!)

KLEINE AUFGABE

- Schreibe ein Programm, dass von 10 rückwärts zählt und die Buchstaben von A bis Z ausgibt
- TIPP: %c für Character %i für Integer in printf nutzen

SWITCH STATEMENTS

```
switch (selection)
{
    case 1:
        PrintGuests(guests);
        break;
    case 2:
        AddGuest(guests);
        break;
    case 3:
        CheckIfGuestHasArrived(guests);
        break;
    default:
        std::cout << "No valid Option was selected." << std::endl;
}
```

SWITCH ZUSAMMENGEFASST

- Wie in Java
- Obacht mit break
- Immer ein default anbieten!


FUNKTIONEN DEKLARIEREN

Was verbirgt sich hier alles?

```
/*  
* Displays all heroes in the vector and makes it possible  
* to select one.  
* The return value is the position of the element in the  
vector.  
*/  
int ShowHeroSelect(std::vector<std::string> &heroes);
```

FUNKTIONEN DEKLARIEREN

```
/*  
 * Displays all heroes in the vector and makes it possible  
 * to select one.  
 * The return value is the position of the element in the  
 * vector.  
 */  
int ShowHeroSelect(std::vector<std::string> &heroes);
```



Return Type

FUNKTIONEN DEKLARIEREN

```
/*  
* Displays all heroes in the vector and makes it possible  
* to select one.  
* The return value is the position of the element in the  
vector.  
*/  
int ShowHeroSelect(std::vector<std::string> &heroes);
```



Methodenname

FUNKTIONEN DEKLARIEREN

```
/*  
 * Displays all heroes in the vector and makes it possible  
 * to select one.  
 * The return value is the position of the element in the  
 * vector.  
 */  
int ShowHeroSelect(std::vector<std::string> &heroes);
```



Parametertyp


FUNKTIONEN DEKLARIEREN

```
/*  
 * Displays all heroes in the vector and makes it possible  
 * to select one.  
 * The return value is the position of the element in the  
 * vector.  
 */  
int ShowHeroSelect(std::vector<std::string> &heroes);
```


Parametername

FUNKTIONEN DEKLARIEREN

```
/*  
* Displays all heroes in the vector and makes it possible  
* to select one.  
* The return value is the position of the element in the  
vector.  
*/  
int ShowHeroSelect(std::vector<std::string> &heroes);
```



Referenz

FUNKTIONEN DEKLARIEREN

```
/*  
* Displays all heroes in the vector and makes it possible  
* to select one.  
* The return value is the position of the element in the  
vector.  
*/  
int ShowHeroSelect(std::vector<std::string> &heroes);
```



Template- / Referenzparameter

FUNKTIONEN DEKLARIEREN

```
/*  
* Displays all heroes in the vector and makes it possible  
* to select one.  
* The return value is the position of the element in the  
vector.  
*/  
int ShowHeroSelect(std::vector<std::string> &heroes);
```



Namespace

STANDARD LIBRARY

- Wann immer ihr etwas aus der Standardlibrary verwenden wollt `std::` zur Abgrenzung davor schreiben (später mal mehr dazu)

(SYSTEM) OUT IN C++ STYLE

```
std::cout << "Select what you want to do:"  
    << std::endl  
    << "1:\t Show available guests" << std::endl  
    << "2:\t Add guest to the list" << std::endl  
    << "3:\t Ask if a guest has already arrived" << std::endl  
    << "0: \t Exit the program" << std::endl;
```

-

READING FROM THE STREAM

```
// defining the variable to put the user input into  
int selection;  
// '>>' operator reads from a stream, in this case the  
// standard input stream  
// type conversation is done by cpp, isn't that great?  
std::cin >> selection;
```

ERWEITERTE STANDARTYPEN

- `std::string`
- `std::vector`

STRINGS

```
std::string name;  
std::string name2="Harry Potter"
```

VECTOR

```
std::vector<std::string> guests;  
guests.push_back("Hermine Granger");
```

USE IT TOGETHER

```
std::string newGuest;  
std::cin >> newGuest; // not in vc++
```

```
guests.push_back(newGuest);
```

-