

# Hello world

Paul-David Brodmann

15. September 2010



This work is licensed under the *Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License*.

Organisatorisches

Hello World

Kompilieren und Ausführen

Typen und Operatoren

Funktionen

Programmfluss

Weitere Konstrukte

Nützliche Tipps

Credits



- ▶ Studentische Initiative voller Studenten
- ▶ Gesamte Fak. IV: ET, TI, Info
- ▶ Organisiert Kurse, Kickerturniere, Gremienarbeit, Beamerverleihe, Keysignings, Einführungswochen, Klausurensammlungen, ...
- ▶ [www.freitagsrunde.org](http://www.freitagsrunde.org)
- ▶ Freitags, 13 Uhr im FR 5046

# Inhalte dieser Veranstaltung

VL 1: Konzept von C, Syntax, Hello World

VL 2: Eingabe und Ausgabe

VL 3: Arrays, Pointer, Strings und Argumente

VL 4: Unions, Structs, Speichermanagement

VL 5: Compiler, Präprozessor

VL 6: Debugging I, Vorstellung der Wochenendaufgabe

## ***Wochenende***

VL 7: Von der Aufgabe zum Code

VL 8: Debugging II

VL 9: Libraries, wo geht es weiter

Was könnt ihr (hoffentlich) aus dieser Woche mitnehmen?

- ▶ C-Kenntnisse
- ▶ Starthilfe in TechGl 3
- ▶ Programmiererfahrungen durch Übungen
- ▶ hoffentlich ein wenig Spaß

## Achtung

keine Scheine oder Leistungsbescheinigungen

Warum machen wir das?

- ▶ wir haben Spaß am Unterrichten
- ▶ Erfahrungen im Vortragen sammeln
- ▶ ...wir bekommen ein wenig Geld dafür

# Tagesablauf

10:00 - 11:00	erster Vortrag
11:30 - 13:00	erste Übung
13:00 - 14:00	Mittagspause
14:00 - 15:00	zweiter Vortrag
15:30 - 17:00	zweite Übung





Mittwoch      FR 5535/5538/5539  
FR 6043/6514/6519  
Donnerstag   FR 5535/5538/5539  
FR 6043/6514/6519

Freitag      TEL 106/206  
Montag      TEL 106/206  
Dienstag    TEL 106/206



- ▶ es wird anonyme Feedbackzettel geben
- ▶ persönliches Feedback für den jeweiligen Vortragenden
- ▶ jeden Tag Feedbackzettel ausfüllen

# Fragen?

# Hello World

## HelloWorld.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello world!\n");
6     return 0;
7 }
```

die main Funktion gibt einen int zurück

- ▶ Rückgabewert = 0  $\Rightarrow$  alles OK
- ▶ Rückgabewert > 0  $\Rightarrow$  Fehler bei der Ausführung

- ▶ wir verwenden die **GNU Compiler Collection**
- ▶ Aufruf über die Komandozeile

```
$ gcc HelloWorld.c -o HelloWorld  
$
```

- ▶ nach dem Kompilieren sieht das Ganze so aus:

```
$ ls -l  
-rwx--x--x 1 dyroff all 5784 Sep 15 16:39 HelloWorld  
-rw----- 1 dyroff all 82 Sep 15 16:39 HelloWorld.c  
$
```

Die wichtigsten gcc-Parameter

Parameter	Beschreibung
-o <i>filename</i>	Ausgabedatei bekommt den Namen <i>filename</i>
-Wall	Warnings werden angezeigt
-Wextra	noch mehr Warnings werden angezeigt
-ggdb	Debuginformationen werden erzeugt
-O <i>n</i>	Optimierungsgrad $n = 0-3$

Ausgeführt wird das so erstellte Programm mit

```
$ ./HelloWorld  
Hello world!  
$
```

Kompilieren und Ausführen kann man auch in einem Schritt

```
$ gcc HelloWorld.c -o HelloWorld && ./HelloWorld  
Hello world!  
$
```

## Ganze Zahlen

	signed	unsigned
char	{-128, ..., 127}	{0, ..., 255}
short	{-32768, ..., 32767}	{0, ..., 65535}
int	{-2147483648, ..., 2147483647}	{0, ..., 4294967295}
long <sup>1</sup>	-	-
long long	{ -9.22E+18, ..., 9.22E+18 }	{ 0, ..., 1.84E+19 }

## Fließkommazahlen

float	{ 1.40E-45, ..., 3.40E+38 }
double	{ 4.94E-324, ..., 1.80E+308 }

---

<sup>1</sup>Auf 32-Bit Solaris Sparcs genauso wie int

## Bitweise Operatoren

Operator	Wertigkeit	Bezeichnung
&	binär	bitweise Und-Verknüpfung
	binär	bitweise Oder-Verknüpfung
^	binär	XOR-Verknüpfung
<<	binär	Bit-Verschiebung nach links (shift left)
>>	binär	Bit-Verschiebung nach rechts (shift right)
~	unär	bitweises Komplement



## Beispiele zu den bitweisen Operatoren

Operator	Beispiele	
&	$1010 \ \& \ 1100 = 1000$	$36 \ \& \ 4 = 4$
	$1010 \   \ 1100 = 1110$	$37 \   \ 11 = 47$
^	$1010 \ ^ \ 1100 = 0110$	$5 \ ^ \ 3 = 6$
<<	$1011 \ << \ 2 = 101100$	$8 \ << \ 2 = 32$
>>	$1011 \ >> \ 2 = 0010$	$256 \ >> \ 2 = 64$
~	$\sim 1010 = 0101$	$\sim 0 = -1$

## Weitere Operatoren

Operator	Beschreibung
<, >, <=, >=, ==, !=	Vergleichsoperatoren
!, &&,	Logische Operatoren
+, -, *, /, %	Rechenoperationen
++, --	Postfix-/Prefix- Inkrement, Dekrement
+ =, - =, * =, / =, % =	Rechenoperation mit Zuweisung
& =,   =, ^ =, << =, >> =	

- ▶ im Allgemeinen gelten die Rechenregeln, z.B. Punkt vor Strich
- ▶ im Zweifel einfach Klammern

```
1 Rueckgabetyt Funktionsname(Parametertyp Parameter, ...)  
2 {  
3  
4 }
```

- müssen immer deklariert werden bevor sie benutzt werden

# Funktionen

```
1  int main()  
2  {  
3      hello();  
4      return 0;  
5  }  
6  
7  void hello()  
8  {  
9      printf("Hello , World\n");  
10 }
```

- ▶ führt zu einem Fehler da die Funktion 'hello' noch nicht bekannt ist
- ▶ folgender Fehler wird auf der Konsole ausgegeben

```
$ gcc -o hello hello.c  
hello.c:12: warning: conflicting types for 'hello'  
hello.c:6: note: previous implicit declaration of 'hello' was here  
$
```

## Lösungen:

- ▶ Reihenfolge der Funktionen main() und hello() tauschen
- ▶ Funktionsdeklaration vor der Benutzung der Funktion

```
1 void hello();  
2  
3 int main()  
4 {  
5     hello();  
6     return 0;  
7 }  
8  
9 void hello()  
10 {  
11     printf("Hello , World\n");  
12 }
```

## Funktionsdeklaration:

```
1 Rueckgabety p Funktionsname (Parametertyp1 , Parametertyp2 , ... );
```

## Eine beispielhafte Fallunterscheidung

```
1  if (1 == 0)
2  {
3      stop_world();
4  }
5  else
6  {
7      accel_world();
8  }
```

# Programmfluss - if

- ▶ in C gibt es keinen Typen für Wahrheitswerte
- ▶ Vergleiche werden mit int's gemacht

```
1  int i=1;  
2  if (i)  
3  {  
4      /* do sth */  
5  }
```

## Achtung

- ▶ nur die 0 steht für *false*
- ▶ alle anderen Werte bedeuten *true*

# Programmfluss - ?:

- ▶ in C gibt es noch eine weitere Form des if-Statements
- ▶ und zwar den tertiären ?:-Operator

```
condition ? value if true : value if false
```

```
1 int gehalt = alter > 40 ? 400 : 200;
```



# Programmfluss - switch

```
1  int i=0;
2  switch (i){
3      case 0: do_sth ();
4              break ;
5      default: do_sth_else ();
6              break ;
7  }
```

- das break darf nicht vergessen werden, sonst wird der nächste Fall auch mit durchlaufen

# Programmfluss - Schleifen

- ▶ aus Java bekannte Schleifen gibt es in C auch
- ▶ for-Schleifen um einen bekannten Bereich zu durchlaufen
- ▶ while-Schleifen für spezielle Abbruchbedingungen



```
1  int i=0;  
2  for( i=0 ; i<=10; i++)  
3  {  
4      /* do sth */  
5  }
```

## Achtung

- ▶ i außerhalb des Schleifenkopfes deklarieren

# Programmfluss - while

Zwei Schleifen zum Vergleich:

```
1  ...  
2  int test=0;  
3  while(test){  
4      printf("It works!\n");  
5  }  
6  ...
```

```
1  ...  
2  int test=1234572;  
3  while(test){  
4      printf("It works!\n");  
5  }  
6  ...
```

- ▶ nicht immer hat man so „schönen“ Code der sich selbst erklärt
- ▶ dafür gibt es Kommentare
- ▶ mehrzeilige Kommentare mit `/* ... */` funktionieren immer

```
1  int main() /* a hello world program */  
2  {  
3      printf("Hello world!\n");  
4      return 0;  
5  }
```

## Weitere Konstrukte - Arrays

```
1  int i=0;
2  int an_array[32]; /* an array of length 32 */
3  for( i=0 ; i < 32 ; i++)
4  {
5      an_array[i] = i+1;
6  }
```

- ▶ die Länge eines Array muss man sich selbst merken
- ▶ am besten eine Variable dafür deklarieren

# Weitere Konstrukte - Enums

```
1 #include <stdio.h>
2 int main()
3 {
4     enum day_t {Mon, Tue, Wed, Thu, Fri, Sat, Sun};
5     enum day_t current_day = get_Day();
6     switch (current_day % 7 ) {
7         case Sat:
8         case Sun: printf("It is Weekend! \n");
9                     break;
10        default:  printf("It is not Weekend!\n");
11                    break;
12    }
13    return 0;
14 }
```

- ▶ Enums machen den Code lesbarer
- ▶ bestehen aus int's
- ▶ der Compiler wandelt diese einfach um
- ▶ d.h. im Maschinencode wird nirgends unser Bezeichner 'Mon' vorkommen



# Nützliche Tipps - Fehlermeldungen

```
1 #include <stdio.h>
2 int main() /* a hello world program */
3 {
4     printf("Hello world!\n")
5     return 0;
6 }
```

```
$ gcc fehler.c -o fehler
fehler.c: In function 'main':
fehler.c:5: error: syntax error before "return"
$
```

## Auflösung:

In Zeile 4 fehlt das Semikolon

# Nützliche Tipps - Fehlermeldungen

```
1  int main() /* a hello world program */  
2  {  
3      printf("Hallo Welt\n");  
4      return 0;  
5  }
```

```
$ gcc fehler.c -o fehler  
fehler.c: In function 'main':  
fehler.c:3: warning: incompatible implicit declaration  
      of built-in function 'printf'  
$
```

## Auflösung:

Hier wurde das '#include <stdio.h>' vergessen

# Nützliche Tipps - man

- ▶ das UNIX 'man' Kommando enthält unter anderem folgende Sektionen
  - ▶ Sektion 2: System Calls
  - ▶ Sektion 3: C Library Functions
- ▶ mit -s kann man 'man' Sektionen Übergeben

```
$ man -s3 printf
```

# Nützliche Tipps - man

## NAME

printf, fprintf, sprintf, snprintf,  
vprintf, vfprintf, vsprintf,  
vsnprintf - formatted output conversion

## SYNOPSIS

```
#include <stdio.h>
```

```
int printf(const char *format, ...);
```

...

## DESCRIPTION

The functions in the printf() family produce output according to a format as described below. The functions printf() and vprintf() write output to stdout, the standard output stream;

...

# Nützliche Tipps - man

- ▶ mit dem 'man' Kommando können auch die Dokumentationen zu ganzen c-Header-Dateien abgerufen werden

```
$ man stdio.h  
$
```

- ▶ dazu müssen jedoch unter Ubuntu die Pakete 'manpages-posix' und 'manpages-posix-dev' installiert werden.

```
$ sudo apt-get install manpages-posix manpages-posix-dev  
$
```

# Nützliche Tipps - C-Versionen

Version	Änderungen
K&R-C	Ursprungsversion, wenig Standard Bibliotheken , wird kaum noch verwendet
C90	Funktionsprototypen, Präprozessor, Bibliotheken normiert
C95	Standard Makros, insbesondere ' <code>__STDC_VERSION__</code> ' zum auslesen der C-Version
C99	komplexe Zahlen, <code>_Bool</code> , <i>restricted</i>

Die Vorlesung heute Nachmittag wird Eingabe und Ausgabe behandeln

Ein Beispiel:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i = 15;
6     printf("i hat den Wert %d!\n", i);
7 }
```

- ▶ Teile des Vortrags stammen von Sebastian Dyroff
- ▶ Urheber des Fotos 'watch.jpg' ist jurvetson von flickr.com
- ▶ Urheber des Fotos 'zahnrad.jpg' ist obenson von flickr.com
- ▶ Urheber des Fotos 'Schleife.jpg' ist mhofstrandvon flickr.com



Nun teilen wir uns in zwei etwa gleich große Gruppen auf.

Jetzt gehen wir gemeinsam in die Räume

**FR 5535/5538/5539**

**FR 6043/6514/6519**