Hello world

Sebastian Dyroff

21. September 2009

Inhaltsverzeichnis

Organisatorisches

Hello World

Typen und Operatoren

Programmfluss

Weitere Konstrukte

Nützliche Tipps

Inhalte dieser Veranstaltung

- LE 1: Konzept von C, Syntax, Hello World
- LE 2: Eingabe und Ausgabe
- LE 3: Arrays, Pointer, Strings und Argumente
- LE 4: Unions, Structs, Speichermanagement
- LE 5: Compiler, Präprozessor
- LE 6: Debugging I
- LE 7: Wie schreibt man guten Code
- LE 8: Debugging II
- LE 9: Libraries, wo geht es weiter

Motivation

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

Achtung

keine Scheine oder Leistungsbescheinigungen

Warum machen wir das

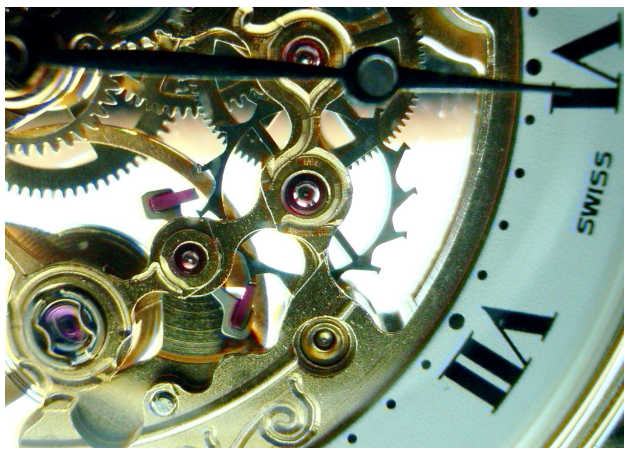
- wir haben Spaß am Unterichten
- Erfahrungen sammeln
- . . . wir bekommen ein wenig Geld dafür

Übungen

finden im TEL 106 und TEL 206 statt
Übungsaufgaben: *http://wiki.freitagsrunde.org/Ckurs*

Tagesablauf

9:00 - 10:00	erster Vortrag
10:30 - 12:00	erste Übung
12:00 - 13:00	Mittagspause
13:00 - 14:00	zweiter Vortrag
14:30 - 16:00	zweite Übung



Feedback

- es wird Feedbackzettel geben
- damit wir uns verbessern
- jeden Tag Feedbackzettel ausfüllen
- jeder Einzelne möchte wissen, wie er war

Fragen?

Jetzt geht es aber dann los

Hello World

HelloWorld.c

```c
#include <stdio.h>

int main(void)
{
        printf("Hello world!\n");
        return 0;
}
```

Hello World

HelloWorld.c

```c
#include <stdio.h>

int main(void)
{
        printf("Hello world!\n");
        return 0;
}
```

die main Funktion gibt einen int zurück

Hello World

HelloWorld.c

```c
#include <stdio.h>

int main(void)
{
        printf("Hello world!\n");
        return 0;
}
```

die main Funktion gibt einen int zurück

- auf der Konsole sieht das Ganze so aus

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

Kompilieren

- wir verwenden die **G**NU **C**ompiler **C**ollection
- als Option ist für uns erstmal -o wichtig
- dieser gibt der Ausgabedatei einen Namen

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

Kompilieren

- wir verwenden die **G**NU **C**ompiler **C**ollection
- als Option ist für uns erstmal -o wichtig
- dieser gibt der Ausgabedatei einen Namen

```
$ 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
$
```

Ausführen

- ausführen können wir das Ganze mit

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

Typen

Ganze Zahlen

	signed	**unsigned**
char	{-128 . . . 127}	{0 . . . 255}
short	{-32768 . . . 32767}	{0 . . . 65535}
int	{-2147483648 . . . 2147483647}	{0 . . . 4294967295}
long[1]	-	-
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 }

[1] Auf 32-Bit Solaris Sparcs genauso wie int

Operatoren

Bitweiße Operatoren

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

Operatoren

Weitere Operatoren

Operator	Beschreibung
$<, >, <=, >=, ==, !=$	Vergleichsoperatoren
$!, \&\&, \|\|$	Logische Operatoren
$+, -, *, /, \%$	Rechenoperationen
$++, --$	Postfix/Prefix Inkrement
$+=, -=, *=, /=, \%=$	Rechenoperation mit Zuweißung

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

Fallunterscheidungen

Eine beispielhafte Fallunterscheidung

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

Fallunterscheidungen

- 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  }
```

- dabei steht die 0 für false
- alle anderen Werte bedeuten true

Fallunterscheidungen

```
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

Funktionen

```
1 Rueckgabetyp Funktionsname(Parametertyp Parameter)
2 {
3
4 }
```

- müssen immer deklariert werden bevor sie benutzt werden

Was tut dieser Code?

```
1  ...
2  {
3                    _ _ _
4                  _ _ _ _
5               _ _ _ _ _ _ _ _
6             _ _ _ _ _ _ _ _ _ _
7           _ _ _ _ _ _ _ _ _ _ _ _
8         _ _ _ _ _ _ _ _ _ _ _ _ _ _
9       _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
10    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
11  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
12    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
13  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
14    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
15  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
16      _ _ _ _ _ _ _ _ _ _ _ _ _ _
17    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
18                 _ _ _ _
19 }
```

Kommentare

- nicht immer hat man so „schönen“ Code der sich selbst erklärt
- dafür gibt es Kommentare
- mehrzeilige Kommentare sollten immer funktionieren

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

Schleifen

- aus Java bekannte Schleifen gibt es in C auch
- for Schleife um einen bekannten Bereich durchzulaufen
- while Schleifen für spezielle Abbruchbedingungen

Schleifen

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

- i ausserhalb des Schleifenkopfes deklarieren
- muss bei manchen Kompilern sein

Schleifen

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 ...
```

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

Enums

```c
#include <stdio.h>
int main(void)
{
        enum day_t {Mon, Tue, Wed, Thu, Fri, Sat, Sun};
        int midnight=0;
        enum day_t current_day = Mon;

        while(!midnight){
                midnight = is_now_midnight();
        }
        current_day++;
        switch(current_day){
                case Tue: printf("It is tuesday!\n");

        return 0;
}
```

Enums

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

Define

```
1  include <stdio.h>
2  #define my_constant 0
3
4  int main(void)
5  {
6          if(my_constant)
7          {
8                  printf("Result FALSE\n");
9          }
10         else
11         {
12                 printf("Result TRUE\n");
13         }
14 }
```

Define

- werden vor dem Kompilieren ersetzt
- deshalb kein =
- und auch kein Semikolon am Ende

Fehlermeldungen

```c
#include <stdio.h>
int main(void) /* a hello world program */
{
        printf("Hello world!\n")
        return 0;
}
```

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

Fehlermeldungen

```c
int main(void) /* a hello world program */
{
        printf("Hallo Welt\n");
        return 0;
}
```

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

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
```

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;
...
```

Ausblick

Die Vorlesung heute Nachmittag wird Eingabe und Ausgabe behandeln

Ein Beispiel:

```c
#include <stdio.h>

int main(void)
{
        int i = 15;
        printf("i hat den Wert %d!\n", i);
}
```

Credits

- Urheber des Fotos ‘Telefunken_hochhaus.jpg’ ist Nico78 Nico Thom at de.wikipedia
- 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