Hello world

Paul-David Brodmann

13. September 2011

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

Inhaltsverzeichnis

Freitagsrunde

- Studenteninitiative
- Gesamte Fak. IV: ET, TI, Info
- Organisieren Kurse, Kickerturniere, Gremienarbeit, Beamerverleih, Keysignings, Einführungswochen, Klausurensammlungen, ...
- `www.freitagsrunde.org`
- Freitags, 14 Uhr im FR 5518

Organisatorisches

Es gibt eine ISIS-Seite für diese Veranstaltung.

- `www.isis.tu-berlin.de/course/view.php?id=5108`
- Vortragsfolien
- Übungsaufgaben
- Aktuelle Informationen

Organisatorisches

Es gibt eine ISIS-Seite für diese Veranstaltung.

- `www.isis.tu-berlin.de/course/view.php?id=5108`
- Vortragsfolien
- Übungsaufgaben
- Aktuelle Informationen

Achtung

Heute aber nicht.... denn ISIS ist **aus**
Mirror unter: `wiki.freitagsrunde.org/Ckurs`

Motivation

Was könnt ihr (hoffentlich) aus dieser Woche mitnehmen?

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

Achtung

keine Scheine oder Leistungsbescheinigungen

Motivation

Warum machen wir das?

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

Inhalte dieser Veranstaltung

Wie läuft der Kurs ab?

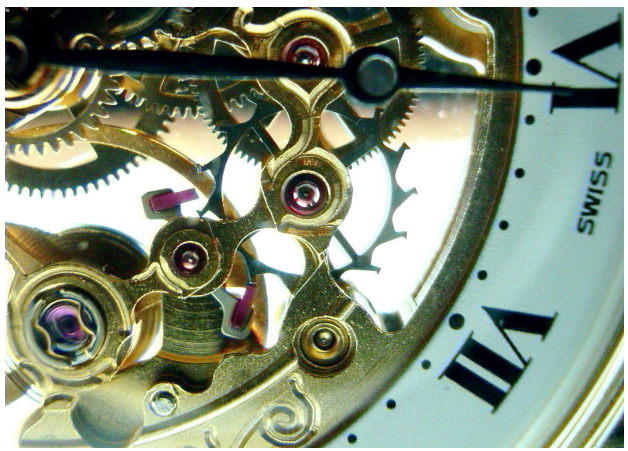
- VL 1: Konzept von C, Syntax, Hello World
- Tut 1: *printf*, *scanf*, *fopen*
- Tut 2: *enum*, *struct*, *union*
- VL 2: Malloc
- VL 3: Pointer
- Tut 3: Präprozessor, Markos
- Tut 4: Debugging
- Tut 5: Libraries, SVN, IDE

Ablaufplan

Zeit	**Dienstag**	**Mittwoch**	**Donnerstag**	**Freitag**
10:15 - 11:15	Vorlesung 1 (H 2032)	Tutorium 2 (TEL 1/2)	Vorlesung 3 (H 2032)	Tutorium 4 (TEL 1/2)
11:30 - 13:00	Übung	Übung	Übung	Übung
13:00 - 14:00	Mittagspause			
14:15 - 15:15	Tutorium 1 (TEL 1/2)	Vorlesung 2 (H 2032)	Tutorium 3 (TEL 1/2)	Tutorium 5 (TEL 1/2)
15:30 - 17:00	Übung	Übung	Übung	Übung

Tagesablauf

10:15 - 11:15	Vorlesung/Tutorium
11:30 - 13:00	Übung
13:00 - 14:00	Mittagspause
14:15 - 15:00	Vorlesung/Tutorium
15:30 - 17:00	Übung



Feedback

Wie könnt ihr uns sagen was euch fehlt und was ihr gut findet?

- es wird anonyme Feedbackzettel geben
- persönliches Feedback für die Vortragenden/Tutoren
- Übungsfeedback über das ISIS-System am Ende des C-Kurs

Fragen?

Hello World

HelloWorld.c

```c
#include <stdio.h>

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

Hello World

HelloWorld.c

```c
#include <stdio.h>

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

die *main* Funktion gibt einen *int* zurück

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

Kompilieren

- wir verwenden die **G**NU **C**ompiler **C**ollection
- Aufruf über die Kommandozeile

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

Kompilieren

- wir verwenden die **G**NU **C**ompiler **C**ollection
- Aufruf über die Kommandozeile

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

- nach dem Kompilieren sieht das Ganze so aus:

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

Kompilieren

Die wichtigsten gcc-Parameter

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

Ausführen

Ausgeführt wird das so erstellte Programm mit

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

Ausführen

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!
$
```

Typen

Ganze Zahlen

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

Fließkommazahlen

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

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	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 =

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4
|	1010_2 | $1100_2 = 1110_2$	37 | 11 =

Operatoren

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4
|	1010_2 | $1100_2 = 1110_2$	37 | 11 = 47

Operatoren

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4
|	1010_2 | $1100_2 = 1110_2$	37 | 11 = 47
^	1010_2 ^ $1100_2 = 0110_2$	5 ^ 3 =

Operatoren

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4
|	1010_2 | $1100_2 = 1110_2$	37 | 11 = 47
^	1010_2 ^ $1100_2 = 0110_2$	5 ^ 3 = 6

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4
|	1010_2 | $1100_2 = 1110_2$	37 | 11 = 47
^	1010_2 ^ $1100_2 = 0110_2$	5 ^ 3 = 6
<<	1011_2 << 2 = 101100_2	8 << 2 =

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4
|	1010_2 | $1100_2 = 1110_2$	37 | 11 = 47
^	1010_2 ^ $1100_2 = 0110_2$	5 ^ 3 = 6
<<	$1011_2 << 2 = 101100_2$	8 << 2 = 32

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4
|	1010_2 | $1100_2 = 1110_2$	37 | 11 = 47
^	1010_2 ^ $1100_2 = 0110_2$	5 ^ 3 = 6
<<	$1011_2 << 2 = 101100_2$	8 << 2 = 32
>>	$1011_2 >> 2 = 0010_2$	256 >> 2 =

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4
|	1010_2 | $1100_2 = 1110_2$	37 | 11 = 47
^	1010_2 ^ $1100_2 = 0110_2$	5 ^ 3 = 6
<<	1011_2 << 2 = 101100_2	8 << 2 = 32
>>	1011_2 >> 2 = 0010_2	256 >> 2 = 64

Operatoren

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4
|	1010_2 | $1100_2 = 1110_2$	37 | 11 = 47
^	1010_2 ^ $1100_2 = 0110_2$	5 ^ 3 = 6
<<	$1011_2 << 2 = 101100_2$	8 << 2 = 32
>>	$1011_2 >> 2 = 0010_2$	256 >> 2 = 64
~	~$1010_2 = 0101_2$	~0 =

Beispiele zu den bitweisen Operatoren

Operator	Beispiele	Übung
&	1010_2 & $1100_2 = 1000_2$	36 & 4 = 4
|	1010_2 | $1100_2 = 1110_2$	37 | 11 = 47
^	1010_2 ^ $1100_2 = 0110_2$	5 ^ 3 = 6
<<	$1011_2 << 2 = 101100_2$	8 << 2 = 32
>>	$1011_2 >> 2 = 0010_2$	256 >> 2 = 64
~	~$1010_2 = 0101_2$	~0 = -1

Weitere Operatoren

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

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

Funktionen

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

- müssen immer deklariert werden bevor sie benutzt werden

Funktionen

```
1  #include <stdio.h>
2  int main()
3  {
4          hello();
5          return 0;
6  }
7
8  void hello()
9  {
10         printf("Hello, World\n");
11 }
```

Funktionen

```
1  #include <stdio.h>
2  int main()
3  {
4          hello();
5          return 0;
6  }
7
8  void hello()
9  {
10         printf("Hello, World\n");
11 }
```

- 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:8: warning: conflicting types for 'hello'
hello.c:4: note: previous implicit declaration of 'hello' was here
$
```

Funktionen

Lösungen:

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

Funktionen

Lösungen:

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

```
1  #include <stdio.h>
2  void hello();
3
4  int main()
5  {
6    hello();
7    return 0;
8  }
9
10 void hello()
11 {
12 printf("Hello, World\n");
13 }
```

Funktionen

Lösungen:

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

```c
#include <stdio.h>
void hello();

int main()
{
  hello();
  return 0;
}

void hello()
{
printf("Hello, World\n");
}
```

Funktionsdeklaration:

```
Rueckgabetyp Funktionsname(Parametertyp1, Parametertyp2, ...);
```

Programmfluss - if

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  }
```

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

Programmfluss - for

```
1 ...
2 int i=0;
3 for( ; i<=10; i++)
4 {
5         /* do sth */
6 }
7 ...
```

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 ...
```

Weitere Konstrukte - Kommentare

- 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

```c
#include <stdio.h>
int main()
{
    enum day_t {Mon, Tue, Wed, Thu, Fri, Sat, Sun};
    enum day_t current_day = get_Day();
    switch(current_day % 7 ){
        case Sat:
        case Sun: printf("It is Weekend! \n");
                  break;
        default:  printf("It is not Weekend!\n");
                  break;
    }
    return 0;
}
```

Weitere Konstrukte - Enums

Welche Vorteile bringen 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

Nützliche Tipps - Fehlermeldungen

```c
#include <stdio.h>
int main() /* 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"
$
```

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

```c
int main() /* 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’
$
```

Nützliche Tipps - Fehlermeldungen

```c
int main() /* 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’
$
```

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 '__STDC_VERSION__'
	zum auslesen der C-Version
C99	komplexe Zahlen, _Bool, *restricted*

Nützliche Tipps - Ausblick

Das Tutorium heute Nachmittag wird Eingabe und Ausgabe behandeln

Ein Beispiel:

```c
#include <stdio.h>

int main()
{
        int i = 15;
        printf("i hat den Wert %d!\n", i);
        return 0;
}
```

Credits

- 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 mhofstrand von flickr.com

Raumaufteilung

Nun teilen wir uns in zwei gleich große Gruppen auf.
Und gehen in die Räume:

TEL 103

TEL 203