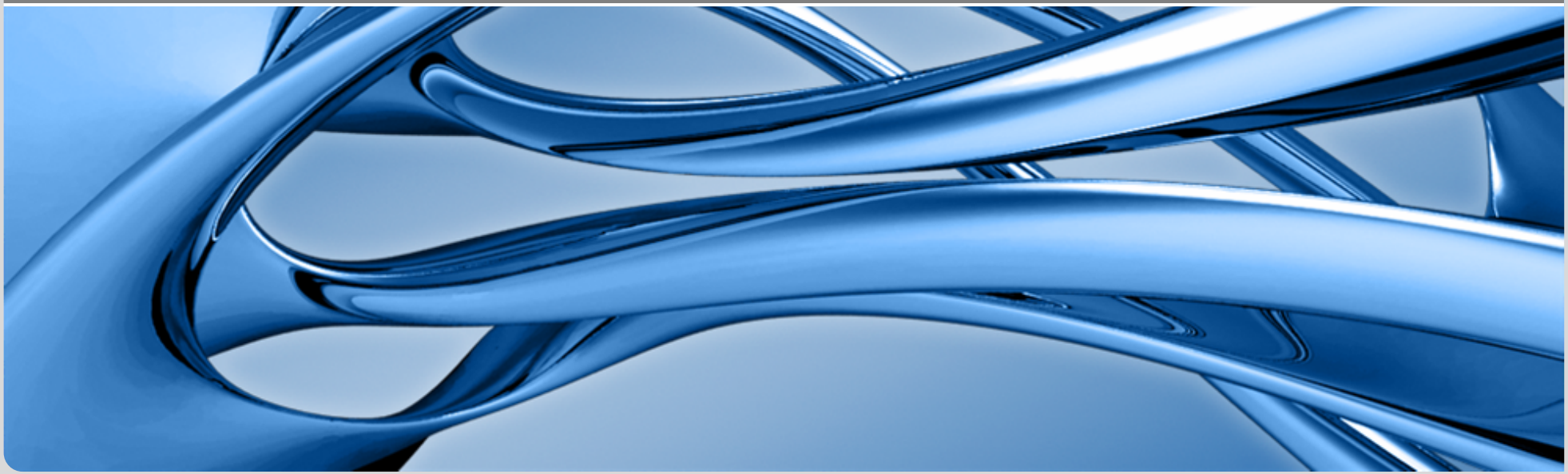
Karlsruher Institut für Technologie

1. Tutorium Rechnerorganisation

Tutorium 7| Grégoire Mercier

KARLSRUHER INSTITUT FÜR TECHNOLOGIE (KIT)



Gliederung

Organisatorisches

C kompilieren

C

Vorstellung

- Grégoire Mercier
- B. Sc. Informatik, 5. Semester
- ubnmv@student.kit.edu
- Homepage der Vorlesung: ti.ira.uka.de
- Jetzt ihr!

Organisatorisches

- Tutorium ist für euch, richtet sich auch nach euch
- Also fragt nach, wenn ihr etwas nicht versteht!

Übungsblätter

- Die Übungsblätter erscheinen immer während einer Woche auf der Vorlesungshomepage
- Einwurfkasten im Keller des Geb. 50.43 (“Infobau”)
- Ist eine sehr gute Gelegenheit, den Stoff nach und nach zu lernen :D

Übungsschein

- freiwillig, aber Klausurbonus (und gute Vorbereitung)
- Bedingungen:
  - alle bis auf 2 Übungsblätter bearbeitet
  - 50% der möglichen Punkte
  - aktive Mitarbeit im Tutorium

Klausur

- Klausur für DT+RO zusammen am Ende des Wintersemesters
- Altklausuren auf der Vorlesungshomepage
- Übungsblätter hilfreich für gutes Abschneiden in der Klausur
- Jeweils 2 Bonuspunkte für den RO- und DT-Schein (nur bei Bestehen)

C-Entwicklungswerkzeuge

- Linux: gcc
- Windows: Microsoft Visual Studio oder MinGW (oder Linux verwenden ;))
- Mac OS X: Xcode etc.
- guter Texteditor

Benutzung

- Befehle für Linux und Windows unterschiedlich
- Navigation in der Konsole: ls (dir) um sich Unterordner anzeigen zu lassen, cd (cd) um sich durch die Ordnerstruktur zu bewegen
- Kompilieren mit gcc: gcc *code*.c -o *programm*
- Ausführen einer ausführbaren Datei: ./*programm* (*programm*)

Hello World

```c
#include<stdio.h>

int main() {
        printf("Hello␣World\n");
        return 0;
}
```

Programmaufbau

- Main-Methode:

```
int main() {
    //do things
    return 0;
}
```

- return optional

Allgemeines

- Einbinden von Headern, hier Aus- und Eingabe

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

- Ausgabe

```
printf("Text");
```

- Ausgabe von Zahlen

```
printf("Zahl: %d",42);
```

Datentypen in C

- Zeichen
  - char (1 Byte), eigentlich kleine Ganzzahl, %c
  - signed und unsigned, Standard hängt von Compiler ab
- ganze Zahlen
  - short (int), oft 2 Byte
  - int, oft 4 Byte, %d
  - long (int), 4 oder 8 Byte, %ld
  - long long (int), 8 Byte
  - signed und unsigned, signed Standard
  - sizeof(short) $\leq$ sizeof(int) $\leq$ sizeof(long) $\leq$ sizeof(long long)
- float (4 Byte), IEEE Gleitkommazahl, %f
- double (8 Byte), IEEE Gleitkommazahl, %f

Kontrollstrukturen

- Bedingung

```
if(condition) {
    //code
} else {
    //code
}
```

Kontrollstrukturen

- Fallunterscheidung

```
switch(expression) {
   case value1:
      //code
      break;
   case value2:
      //code
      break;
   default:
      //code
      break;
}
```

Kontrollstrukturen

- Schleifen

```
for(initialization; end; iteration) {
    //code
}
while(condition) {
    //code
}
do{
    //code
} while(condition)
```

Kontrollstrukturen

- Weitere Kontrollflusssteuerungen
  - break und continue
  - return
  - (goto ..., sollte man nicht nutzen, da schlechter Programmierstil)

Pointer/Zeiger

- Umsetzung der maschinennahen Programmierung in C
- Bezeichnung einer Variable in C, die als Wert eine Adresse hat
- Kennzeichnung durch * nach dem Datentyp (kann auch void für unbekannten Datentyp sein)
- Beispiel: int *a;
- Häufig für Funktionsparameter genutzt
- String ist in C ein char*

Pointer/Zeiger

- Ermittlung der Adresse einer Variable mit &
- Dereferenzierungsoperator (auch mit * bezeichnet) liefert die Daten, die an der Adresse gespeichert sind, auf die der zugehörige Zeiger zeigt
- Sind Daten Teil eines Objekts (z.B. struct) schreibt man a->b anstatt (*a).b
- Hinweis: Zeiger können auf nicht benutzten Speicher zeigen, auf gar keinen Speicher oder auf falsch deklarierte Typen ;)
- "NULL"bedeutet dass Zeiger ins Nichts zeigt

Übung

- Wie wird ein Zeiger auf einem Datentyp deklariert?
- Was bedeutet der Ausdruck &variable in C?
- Was machen die einzelnen Zeilen und was wird am Ende ausgegeben?

```
int a = 42;
int *p;
p = &a;
printf("p = %d, a = %d", p, *p);
```

Übung

Welche dieser Variablen sind gültige Zeiger?

```
int* a, b;
int c, d;
void *e = &c;
int* f = NULL;
int **g = &a;
```

Lösung

- a, e, f, g sind Zeiger
- a zeigt auf zufällige Stelle
- f ist NULL

Operatoren in C

- Arithmetische Operatoren: +, -, *, /, %
- Bit-Operatoren: $\sim$, $\ll$, $\gg$, &, ^, |
- Logische Operatoren: !, &&, ||
- Vergleichsoperatoren: $\leq$, $\leq$, $\geq$, $\geq$, ==, $\neq$
- Kombinierte Operatoren: ++, –, +=, -=, %=, $\gg$=, ...
- Weitere: ? :

Aufgaben

a = 1; b = 2; c = 13; d = 42;

- e = (∼a) - b

Aufgaben

a = 1; b = 2; c = 13; d = 42;

- e = (∼a) - b
- e = -4

Aufgaben

a = 1; b = 2; c = 13; d = 42;

- e = (∼a) - b
- e = -4
- f = (c & d) ≫ b

Aufgaben

a = 1; b = 2; c = 13; d = 42;

- e = ($\sim$a) - b
- e = -4
- f = (c & d) $\gg$ b
- f = 1

Aufgaben

a = 1; b = 2; c = 13; d = 42;

- e = (∼a) - b
- e = -4
- f = (c & d) ≫ b
- f = 1
- h = b++;

Aufgaben

a = 1; b = 2; c = 13; d = 42;

- e = ($\sim$a) - b
- e = -4
- f = (c & d) $\gg$ b
- f = 1
- h = b++;
- h = 2, b = 3

Aufgaben

a = 1; b = 2; c = 13; d = 42;

- e = ($\sim$a) - b
- e = -4
- f = (c & d) $\gg$ b
- f = 1
- h = b++;
- h = 2, b = 3
- i = (a == b)?c:d

Aufgaben

a = 1; b = 2; c = 13; d = 42;

- e = (∼a) - b
- e = -4
- f = (c & d) ≫ b
- f = 1
- h = b++;
- h = 2, b = 3
- i = (a == b)?c:d
- i = d

Wenns reicht

Schreibt ein C-Programm, dass euch folgende Ausdrücke auswertet:

264364 | 153264

1252412 % 153

Schreibt eine Methode void halloWelt(char* c), die den char - Pointer mit “Hallo Welt!“ füllt.
(Vorlage findet ihr unter https://github.com/Ravnson/ROTut/blob/master/main.c)