Tut 1 (Tutor)

Monday, 22. October 2018 19:12

0. ORGANISATORISCHES

- Umfrage: ☐ ET ☐ Automotive ☐ Andere ?

> Fu Zheng (Frank)
> f.zheng@campus.tu-berlin.de
> Sprechstunde: Mi 15-16h, MAR 6.029

- Sprechstunde in Form einer betreuten Rechnerzeit
  → Fragen zur VL, Tut, HA
  → Was tun wenn der Prof. mich meldet ?

0.1 InfTech Modul

- Jeder mit ISIS vertraut ? Einschreibung zw. notwendig
  → Lehrmaterialien, HA-Blätter
  → Ort zur HA-Abgabe & Online-Tests (dazu später)

- Formalitäten im Folgenden sind auch auf dem Merkblatt (GLP)
- LV ist in 3 Blöcken gegliedert

> - C: HA 1...5
> - C++: HA 6...9
> - RA: HA 10...13
>
> Abschluss mit Online-Test
>
> ① Programmieren in C/C++
> → Essentials zum Programmieren in C
> → Grundlagen der Objektorient. Programmierung in C++
>
> ② Rechneraufbau
> → Wie rechnet ein Computer
> → Wie ist er aufgebaut ?

- Jeder Block wird mit einem Blocktest abgeschlossen werden (ISIS)
  → 2/3 müssen zur Klausurzulassung bestanden sein
  → Außerdem max. 1 HA-Blatt durchfallen pro Block

0.1.1. Hausaufgaben

- Pro Woche muss ein HA-Blatt bearbeitet werden
  → Abgabe in 4er-Gruppen, Online auf ISIS
  ↳ Jetzt Gedanken machen, wer mit wem. Liste später
  → Zum WE auf ISIS bereit
  → Ganz oder gar nicht: 50% oder mehr = bestanden

0.1.2 Klausur

- Das gonne für die Zulassung zur Klausur
  → 2 v. 3 Online Tests mit mehr als 50% bestehen
  → max. 1 HA pro Block nicht bestehen
- Am Ende des Semesters
- Klausur ist [illegible]

- Alle Klarheiten beseitigt?

1. Let's C the world

- Einstieg in die Welt der Programmierung
- Wer schonmal C/C++ programmiert?
  → Something else?

1.1 Hello World

- Tradition, mit einem Hello World Programm zu beginnen
- Wir schreiben mit einem beliebigen Texteditor
  → An sich würde Notepad auch gehen
  → Besser: Notepad++ (Win), Atom (Mac)
  Hervorhebung wichtiger Schlüsselwörter, intelligente Formatierung
- Wie sieht ein Programm aus?
  → Sinnvoll: HIER fängt das Programm an.
  Egal ob Hello World oder Need For Speed Most Wanted

```
int main (void) {
}
```

- int vor main steht für Rückgabedatentyp: Zahl
  → Wenn Programm endet, dient eine Zahl dazu dem Betriebssystem mitzuteilen dass das Program fertig ist

```
return 0;
```
← Hinter jedem Ausdruck!

- 0 i.d.R. für alles ok.
  → ≠0: Infos über Fehlermeldung

- Jetzt "Hello World" ausgeben

```
printf("Hello World\n");
```

- Woher weiß man was Printf macht?
  → Niedergeschrieben in einer Bibliothek namens Stdio
  ↳ Standard Input/Output → Ein/Ausgabe

```
#include <stdio.h>
```
← Hier sind Fkt wie printf umgesetzt

1.2 Kompilieren

- Effektiv genauso funktional wie ne Einkaufsliste
- Müssen aus dem Code ein PROGRAMM basteln, dass der Computer es versteht
- Compiler: Helfer, um Code in unlesbare Maschinensprache zu übersetzen (binär)
- Wir: "gcc" aus dem Terminal holen

```
gcc helloworld.c -std=c11 -Wall -o helloworld
```
(-std=c11: Parameter; -Wall: und zeige alle Warnungen an!)

Hey —, mache aus — unter Vers. des C11-Standards das Programm helloworld.exe...

ℹ Ordnerwechsel: cd

1.3 Compilerfehler

- Was passiert wenn ...
  - Semikolon vergessen?
  - { weg?
  - int vergessen?
  - ; hinter #include ?

- Error: Programm kompiliert nicht. Fehler suchen!
- Warning: Kompiliert, könnte aber unerwartetes machen

1.4 Printf im Detail

- Printf: Print Formatted
- Zeilenumbruch etc mit sog. Escape-Sequenzen
- ### ... Hallo? Wer laut sich? ^M

1.5 Kommentare

- Woran erkennt man einen schlechten Programmierer?

→ Kein Kommentar

Code kommentieren

```
// Eine Zeile
/* Mehr-
   zeilig */
```

- Verbessert Lesbarkeit. Was habe ich vor 2 Wochen geschrieben?
- Sonst keine Funktion, wird nicht mitkompiliert

2. VARIABLEN & DATENTYPEN

2.1 Variablen

① Variable ...
- Behälter für Werte
- Typ, Name und Wert
- Nur Wert kann im Betr. ge. werden

2. VARIABLEN & DATENTYPEN

Menge	Bsp.	C1011 Datentyp
IN	0, 5, 42	unsigned {char, short, int, long}
Z	-2, -73, 42	{char, short, int, long}
Q/R	3.1415, -2.176	float, double

- Deklaration & Definition:

```
typ name;
int teilnehmer;
```

- + Initialisierung

```
float note = 1.3;
```

- Mathematik: Platzhalter für ne Zahl: x
- Informatik: Leicht anders. "Behälter" für Daten

→ Mache Platz im Speicher für eine Variable namens x, x soll 7 sein

- 3 Eigenschaften: Name, Typ, Wert

→ Typ: Was will ich speichern? Zahl? Bild? Kommazahl?

↳ Bestimmt, wie viel Speicher Variable einnimmt

→ Name: Programmierer gibt der Variable einen sinnvollen Namen

↳ Spielregeln zeigen ! (VL)

→ Wert: -7,3 oder 3000

- Wie sagt man dem Programm: Here comes my Variable ?

→ Durch Deklaration (Bekanntmachung)

→ Definition: Reserviere gleichzeitig passenden Speicher (Regelfall)

→ Initialisierung: Weise gleich einen Wert zu

2.2 Datentypen

- Welche Datentypen gibt es ?

→ Zahlenmengen !

- Warum so viele DT ?

→ Sind unterschiedlich groß.

char: 1 Byte, int: 4 Bytes (große Zahlen !)

→ Bsp: Größe mit Sizeof ermitteln → Demo !

→ Alter eines Wildschweins: 0-255 ausreichend → char

→ Einwohnerzahl USA ? → unsigned int

→ Ähnlich HA !

2.3.1 printf mit Variablen

- Printf kann auch Variablen auf der Konsole ausgeben mith. Escape-Sequences

Variablen ausgeben

```
int f = 150000;
printf("Sie haben %d Follower", f);
```

3. AUSDRÜCKE & OPERATOREN

- Bisher: Es gibt Variablen

→ Jetzt: Mit Variablen arbeiten, z.B. verrechnen

- Aufg. 3a): Literale sind Konstanten

→ Mathematisches Verständnis bedienen

↳ Interaktiv !

Literale

... sind Konstanten, die mit Variablen verrechnet werden

```
int a = (5-x) / 3;
```

Operatoren

Primär	( )
unär	-, + (Vorzeichen)
Multipl.	*, /, %
Add.	+, -
Zuweisung	=

→ Wenn nicht explizit angegeben, wird z.B 3 als int aufgefasst

- Welche Operatoren gibt es ?

→ Präzedenz wie in der Mathematik

i a = a + 4 ; gleich wie a += 4 ;

3.1 Casten

- Typ-Umwandlung wird auch als Cast bezeichnet

Typecast

Bsp.:
```
int i = 1 ;
i = i + 4 ;      // i = 5

double d = 2.5 ;
i = d + i ;      // i = ?
```

- Was passiert hier genau?

Cast immer in größeres Format
→ int i → double i

↓

Addiere nun d + i : 7.5

↓

zurück ins Zielformat. Was passiert mit 0.5 ?

→ Wird brutal herausgeschnitten (immer abgerundet)!

- Wertvolle Erkenntnis für Aufg. 3b)

→ 5,4 letzteres weil 0.5 → 0, Integer-Division!

Implizit: Compiler interpretiert Cast

Explizit: Konversion erzwingen

```
double Bruch = 5 / 4 ;
double abrund = (int)Bruch ;
```

- Aufg. 3d) Wer sieht das Problem?

i Bei einem Cast float → int werden Nachkommastellen abgeschnitten
→ i.d.R. unerwünscht