Prof. Dr. Wolfgang Effelsberg
Dirk Stegemann Mannheim, den 25.04.2005

Vorlesung **Programmierkurs 2**, Sommersemester 2005

Übungsblatt 2

Abgabe spätestens am Montag, 02. Mai 2005, um 13.45 Uhr.

Aufgabe 1: Compiler und Debugger

(8 Punkte, Abgabe nur schriftlich)

(i) (3 Punkte) Was bewirken die gcc-Optionen `-ansi`, `-Wall` und `-g`?

(ii) (5 Punkte) Finde die syntaktischen Fehler in dem Programm `wasbinich.c`, das von der Übungswebseite und in dotlrn heruntergeladen werden kann. Wie lautet die Ausgabe des Programms? Welche Werte nimmt die Variable `a` jeweils an der mit (1) gekennzeichneten Stelle an?

Hinweis: Für die Lösung der Aufgabe kannst du z.B. den Debugger `gdb` benutzen, der im PiPool zur Verfügung steht.

Aufgabe 2: Operatoren in C

(7 Punkte, Abgabe nur schriftlich)

Schreibe ein kleines C-Programm, das die Operatoren `a++`, `&` und `=` verwendet. Das Programm soll so gestaltet sein, dass sich seine Ausgabe ändert, wenn man `a++` durch `++a`, `&` durch `&&` oder `=` durch `==` ersetzt. Das geänderte Programm soll immer noch ohne Fehlermeldungen compilieren, Warnungen sind diesmal aber ausdrücklich erlaubt. Erläutere kurz, wie sich die Programme verhalten und warum es zu den unterschiedlichen Ausgaben kommt.

Aufgabe 3: Characters in C

(5 Punkte, Abgabedatei `Buchstaben.c`)

Schreibe ein C-Programm, das jeweils in einer Zeile alle Elemente aus $\{a, \ldots, f\} \times \{a, \ldots, f\}$ ausgibt, d.h. alle Buchstabenpaare $aa, ab, ac, \ldots, fe, ff$. Gehe vereinfachend davon aus, dass die Zeichen a bis f zusammenhängend im Zeichensatz der Maschine angeordnet sind.

Du kannst für das Programm die aus Java bekannten `for`-Schleifen benutzen. Beachte jedoch, dass in ANSI C Variablendeklarationen nur am Anfang eines Blocks zulässig sind. Anstatt z.B.

```
public static void main(String args[]){
  for (int i = 1 ; i <= 10 ; i++){
     i = 2*i;
  }
}
```

in Java muss man in ANSI C schreiben

```
int main(){
  int i;
  for (i=1 ; i <= 10 ; i++){
     i = 2*i;
  }
}
```

Ein- und Ausgabe in C

Die Befehle, die in C die Ein- und Ausgabe steuern, sind in der Headerdatei `stdio.h` deklariert. Diese muss (so wie im „Hello World“ Programm vom letzten Übungsblatt) zu Beginn des Quellcodes mit der Anweisung

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

in das Programm eingebunden werden.

Ausgabe: Wir benutzen für die Übungsaufgaben zunächst nur die Ausgabefunktion `printf`. Eine einfache Anwendung dieser Funktion ist ebenfalls im „Hello World“ Programm zu sehen: Man übergibt der Funktion einfach den String, den man ausgeben möchte, also z.B.

```
printf("Ich bin ein Ausgabestring.\n");
```

Ähnlich wie bei der Java-Methode `System.out.print` muss man bei `printf` Zeilenumbrüche ggf. selbst durchführen, indem man das Steuerzeichen `\n` an den String anhängt.

Um den Wert der Variablen `int zahl = 12` auszugeben, würde man in Java z.B. schreiben

```
System.out.print("Es gibt " + zahl + " Monate.\n");
```

Die Ausgabe von Variableninhalten in C folgt einem anderen Konzept, man muss zunächst in den String einen Platzhalter für die Variable einfügen und hinter den String dann die Variable selbst:

```
printf("Es gibt %i Monate.\n", zahl);
```

Hierbei ist `%i` der Platzhalter für einen `int`-Wert. Analog ist z.B. `%f` der Platzhalter für einen `float`-Wert. Um mehrere Variablen auszugeben, geht man genauso vor:

```
int zahl, quadrat;
zahl = 5;
quadrat = zahl * zahl;
printf("Das Quadrat von %i ist %i.\n", zahl, quadrat);
```

Dabei wird der j-te Platzhalter durch die j-te Variable ersetzt.

Eingabe: Die Eingabe funktioniert ähnlich wie die Ausgabe. Wir beschränken uns zunächst auf den Befehl `scanf`. Auch hier gibt es einen String mit Platzhaltern, den sog. *Formatkontrollstring* mit z.B. `%i` für `int`-Werte und `%f` für `float`-Werte. Beispiel:

```
int zahl;
printf("Gib eine Zahl ein: ");
scanf("%i", &zahl);
```

Beachte, dass im `scanf`-Funktionsaufruf die Variable `zahl` mit dem Adressoperator `&` verwendet wird. Warum das so ist, werden wir im Vorlesungskapitel *Zeiger und komplexe Datenstrukturen* behandeln. Für den Augenblick genügt es zu wissen, dass es so sein muss.

Weiterführende Informationen zur Ein- und Ausgabe in C gibt es in jedem guten C-Buch und natürlich später in der Vorlesung im Kapitel *Dateien, Ein- und Ausgabe*.