1. Übungsblatt:
Programmierpraktikum II (SS 2002)

Abgabe: 6. Mai 2002

Aufgabe 1: Programmerstellung

(5 Punkte)

Das Standard-Programm zur Ausgabe von *"Hello World"* sieht in C wie folgt aus:

```
#include <stdio.h>
int main() {
  printf("Hello World!\n");
  return 0;
}
```

Anhand dieses Programmes soll nun die Umwandlung von C-Quellcode in ein ausführbares Programm demonstriert werden. Dazu werden die vier Schritte *Präprozessing*, *Kompilieren*, *Assemblieren* und *Linken* durchlaufen. Beantworte hierzu die folgenden Fragen:

a) Was leisten die vier Schritte genau?

b) Gib für jeden der vier Schritte die Befehlszeile an, mit der man die Codeerzeugung mit dem `gcc`-Compiler nach Ausführung dieses Schrittes beenden kann. Wie sieht das Output des *Kompilier*-Schrittes für das "Hello World"-Programm aus (vollständiges Listing)?

c) Wie groß (in Bytes) ist das Zwischenergebnis nach dem Assemblieren, wie groß das ausführbare Programm? Begründe den Unterschied!

Tipp: Zur Beantwortung der Fragen können z.B. die `man`-Pages des Befehls `gcc` und natürlich die Literatur zur Vorlesung herangezogen werden.

Aufgabe 2: Syntaktische/Semantische Fehler

(4 Punkte)

Seien a, b und c reelle Zahlen und *feld* ein Feld aus sechs reellen Zahlen *feld*[0] bis *feld*[5]. Die folgenden Codefragmente in Pseudocode enthalten jeweils einen Fehler. Finde den Fehler, beschreibe ihn und gib an, ob es sich um einen syntaktischen oder einen semantischen Fehler handelt.

a)
```
a = 5
b = a
c = (7 + a / ((b - a) * a)
```

b)
```
a = 5
b = a
c = 7 + a / (a - b)
```

c)
```
if not (a = 1)
then
  c = log(a)
else
  c = log(1-a)
```

d)
```
if not (a = 1)
then
  c = log(a)
then
  c = log(c)
else
  c = log(2a)
```

e)
```
a = 3.1415927 / 2
c = cos(sin(cos(a))
```

f)
```
a = 1
for i=1 to 6
  a = a*2
  feld[i] = a
```

g)
```
a = 5
for i=1 to 3
  b = feld[i]
  a * b = feld[i] + 2
```

h)
```
a = -2
b = 1
c = 128
for i=1 to 7
  b = b * a
c = 300 mod (b+c)
```

Aufgabe 3: Primzahlpotenzen

(5 Punkte)

Schreibe ein C-Programm, das die Primzahlpotenzen $7^0, 7^1, \ldots, 7^{10}$ berechnet und am Bildschirm in der Form

```
7^0 = 1
7^1 = 7
...
```

ausgibt.
Tipp: Du kannst die aus Java bekannte `for`-Anweisung benutzen. Beachte aber, dass in C alle verwendeten Variablen zu Beginn der Funktion (hier z.B. `main()`) deklariert werden müssen!

Aufgabe 4: Euro-Umrechner

(6 Punkte)

Schreibe ein C-Programm, das DM in Euro umrechnet oder umgekehrt. Dazu liest es zunächst den Betrag und dann die Währung (0 für DM oder 1 für Euro) von der Standardeingabe. Dann rechnet es den korrespondierenden Wert (entweder Euro oder DM) aus und gibt ihn an die Standardausgabe zurück. Falls als Währung weder 0 noch 1 angegeben wurden, soll das Programm einen Fehler melden und sich beenden.
Tipp: Das Programm benötigt eine `if`-Anweisung, die genauso funktioniert wie bei Java.

Ein-/Ausgabe in C

Die Befehle, die in C die Ein-/Ausgabe steuern, sind in der Headerdatei `stdio.h` deklariert. Diese muss (so wie im Beispielprogramm "Hello World" gezeigt) zu Beginn des Quellcodes mit einer `#include`-Anweisung eingebunden werden.

Ausgabe: Wir benutzen im Rahmen dieses Kurses zunächst nur die Ausgabefunktion `printf`. Eine besonders 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("Dies ist ein Output-String.\n");
```

Die einzige Besonderheit hier ist die Zeichenfolge `\n` - sie steht lediglich für einen Zeilenumbruch.

Schwieriger wird das Ganze, wenn man den Wert von Variablen ausgeben will. Dazu muss man zunächst in den String einen Platzhalter für die auszugebende Variable einfügen und dann hinter dem String die Variable selbst. Ein Beispiel:

```
int zahl = 12;
printf("Es gibt %i Monate.\n", zahl);
```

Hier ist also `%i` der Platzhalter für einen Integer-Wert. Analog ist z.B. `%f` der Platzhalter für einen Float-Wert. Sollen mehrere Variablen ausgegeben werden, so geht man genauso vor:

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

Hier wird also offensichtlich der erste Platzhalter durch die erste Variable und der zweite Platzhalter durch die zweite Variable ersetzt.

Eingabe: Die Eingabe funktioniert ähnlich wie die Ausgabe. Wir beschränken uns hier zunächst auf den Befehl `scanf`. Auch hier wird wieder ein String verwendet, der sogenannte *Formatkontrollstring*. Dieser enthält ähnliche Platzhalter für die Eingabe wie beim `printf`-Befehl, also z.B. `%i` für Integer- oder `%f` für Float-Werte. Bsp.:

```
int zahl;
printf("Gib eine ganze Zahl ein!\n");
scanf("%i", &zahl);
```

Beachte hier vor allem, dass als Argument nicht `zahl`, sondern `&zahl` verwendet wird. Für eine Erklärung dieses `&`-Zeichens müssen wir uns allerdings noch ein paar Wochen gedulden - im Augenblick genügt es zu wissen, dass es da hingehört.

Weiterführendes: Wer sich mit solch spärlichen Erklärungen nicht abspeisen lassen will, der kann natürlich gerne ein Buch zur Hand nehmen - die Befehle `printf` und `scanf` sind in beiden Büchern zur Vorlesung ausführlich erklärt. Andernfalls müsst ihr noch ein paar Wochen warten, bis in der Vorlesung das Kapitel "Ein- und Ausgabe" behandelt wird.