4. Übungsblatt: Programmierpraktikum II (SS 2003)

Abgabe: 2. Juni 2003

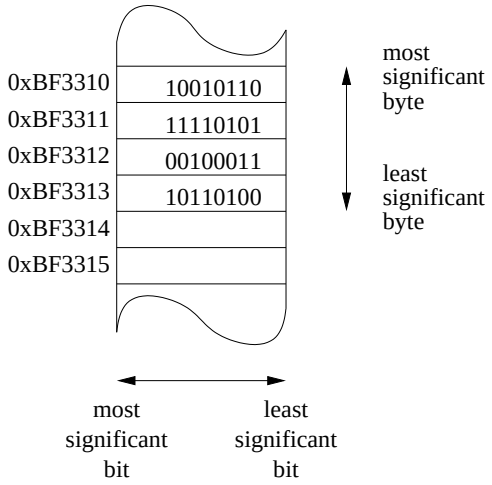
Bemerkungen:

- Es sind alle Abgabehinweise auf der Übungswebseite zu beachten!
- Als Programmbibliotheken sind `stdio.h`, `stdlib.h`, `time.h` und (für die Cracks) `errno.h` zugelassen.

Aufgabe 1: Typkonversion

(7 Punkte - keine Programmieraufgabe!)

Der Inhalt des Speichers sehe wie folgt aus:



Welchen Dezimalwert hat

a) ein `int`-Wert (Länge 4 Byte),

b) ein `unsigned int`-Wert (Länge 4 Byte),

c) ein `short int`-Wert (Länge 2 Byte),

d) ein `char`-Wert (Länge 1 Byte),

e) ein `float`-Wert (Vorzeichen 1 Bit, Mantisse 10 Bit, Exponent 5 Bit),

f) ein `double`-Wert (Vorzeichen 1 Bit, Mantisse 24 Bit, Exponent 7 Bit),

der ab Adresse `BF3310` gespeichert ist, wenn für alle `signed`-Werte Zweierkomplementdarstellung verwendet wurde?

g) Welchen Hexadezimal-Wert hat der `int`-Wert (Länge 4 Byte), der ab Adresse `BF3310` gespeichert ist?

Alle Antworten sind zu begründen (Rechenweg o.ä.).

Aufgabe 2: Freitag der Dreizehnte

(8 Punkte, Abgabedatei: thirteen.c)

Der Gregorianischer Kalender wurde am Freitag, den 15.Oktober 1582 eingeführt. Von da an wurden Schaltjahre wie folgt gezählt:

- Ein Jahr, dessen Jahreszahl durch 4 teilbar ist, ist ein Schaltjahr.
- Ein Jahr, dessen Jahreszahl durch 100 teilbar ist, ist doch kein Schaltjahr.
- Ein Jahr, dessen Jahreszahl durch 400 teilbar ist, ist doch wieder ein Schaltjahr.

Alles klar? Nun die Aufgabe: Schreibe ein Programm, das ausgibt, wie oft welcher Wochentag (Montag bis Sonntag) zwischen dem 15.10.1582 und dem 2.6.2003 auf den 13. Tag eines Monats gefallen ist. Ziel ist es, herauszufinden, ob Freitag der Dreizehnte häufiger vorkommt als z.B. Montag der Dreizehnte.

Tipp: Es mag hilfreich sein, für dieses Programm Felder zu verwenden, die ja im Prinzip schon aus Java bekannt sind. In C wird ein Feld mit dem Befehl

```
datentyp feldname[konstante];
```

angelegt und kann bei Bedarf sofort initialisiert werden, also z.B.

```
int quadrate[5] = {0,1,4,9,16};
```

Natürlich kann man auch nur ein leeres Feld anlegen und es später vollschreiben. Beachte dabei aber, dass der Inhalt eines Feldes genau wie bei allen anderen Variablen nicht definiert ist, solange man es nicht initialisiert hat. Bsp.:

```
const int laenge = 5;
int quadrate[laenge];  // Feldinhalt undefiniert;
int i;
for(i=0; i<laenge; i++)
  quadrate[i] = i*i;
```

Wie man sieht, läuft der Feldindex genau wie bei Java von 0 bis `laenge-1`.

Aufgabe 3: Streichholzspiel

(7 Punkte, Abgabedatei: match.c)

Schreibe ein C-Programm, das das berühmte Streichholzspiel für zwei Spieler simuliert:

- Das Programm zieht eine zufällige Zahl z von Streichhölzern ($10 \leq z \leq 100$). Zum Ziehen von Zufallszahlen siehe unten.
- Das Programm fragt wechselweise jeden Spieler, wie viele Streichhölzer er vom Stapel wegnehmen will (es müssen 1, 2 oder 3 Streichhölzer genommen werden), und gibt die neue Zahl von Streichhölzern aus.
- Wer das letzte Streichholz wegnimmt, hat verloren. Das Programm erklärt den anderen Spieler zum Sieger.

Die Gestaltung der Ein-/Ausgabe ist dir überlassen, solange das Programm den obigen Anforderungen genügt.

Unbepunktete Zusatzaufgabe (Abgabedatei: winmatch.c): Ändere das Programm so ab, dass ein Spieler gegen den Computer spielt und dass der Computer versucht, das Spiel zu gewinnen!

Erzeugung von Zufallszahlen

Der Standard-Zufallsgenerator von C wird in der Datei `stdlib.h` beschrieben. Auch wenn er für "ernsthafte" Anwendungen (z.B. Simulationen oder Verschlüsselung) oft zu schlecht ist, reicht er für unsere Zwecke doch aus.

Initialisierung: Der Zufallsgenerator muss initialisiert werden, sonst liefert er bei jedem Programmstart die gleiche Folge von Zufallszahlen. Eine beliebte Variante besteht darin, zur Initialisierung die aktuelle Uhrzeit zu verwenden. Dazu muss man noch die Datei `time.h` mit einbinden und den Befehl `srand(time(0))` aufrufen.

Zahlenerzeugung: Der Befehl `rand()` liefert eine Pseudo-Zufallszahl im Bereich zwischen 0 und einem Wert `RAND_MAX` zurück, der ebenfalls in `stdlib.h` definiert ist. Wer kleinere Zufallszahlen benötigt, kann mittels `x = rand()%n` Zahlen zwischen 0 und $n-1$ erzeugen.