Prof. Dr. Wolfgang Effelsberg
Dirk Stegemann Mannheim, den 30.05.2005

Vorlesung **Programmierkurs 2**, Sommersemester 2005

Übungsblatt 7

Abgabe spätestens am Montag, 06. Juni 2005, um 13.45 Uhr.

Aufgabe 1: Call-by-Value und Call-by-Reference

(5 Punkte, Abgabedatei `minmax.c`)

(i) Schreibe eine C-Funktion `minmaxVal`, der ein Zeiger auf ein `int`-Feld und die Länge des Felds übergeben werden und die den Wert des kleinsten und den Wert des größten Feldeintrags über den Rückgabewert der Funktion an das aufrufende Programm zurückliefert.

(ii) Implementiere weiterhin eine C-Funktion `minmaxRef` mit der gleichen Funktionalität wie `minmaxVal`, allerdings soll `minmaxRef` `void` als Rückgabetyp besitzen.

(iii) Gib eine `main` Funktion an, die die beiden Funktionen testet.

Aufgabe 2: Large Integers

(6 Punkte, Abgabedatei `addlarge.c`)

Schreibe eine C-Funktion

`unsigned int add(unsigned int *a, unsigned int *b, unsigned int *c)`,

mit der man zwei große vorzeichenlose Ganzzahlen addieren kann. Die Parameter `a` und `b` zeigen auf den Anfang der Speicherbereiche, in denen die Summanden a bzw. b abgelegt sind, und der Parameter `c` zeigt auf den Speicherbereich, in den das Ergebnis $c = a + b$ geschrieben werden soll. Die Funktion soll davon ausgehen, dass jeder dieser Speicherbereiche die Größe von 4 `unsigned int`s besitzt und die Zahlen mit dem most significant Bit beginnend in den Bereichen abgelegt werden. Falls bei der Addition von a und b ein carry auftritt, d.h. die Summe von a und b passt nicht in 4 `unsigned int`s, soll die Funktion die hinteren Bits des Ergebnisses im Ergebnis-Speicherbereich ablegen und den Wert 1 zurückgeben. Andernfalls ist der Rückgabewert 0.

Schreibe auch eine `main` Funktion, die deine Funktion `add` testet, indem sie zwei zufällige Summanden a und b erzeugt, mit Hilfe der Funktion `add` addiert und die Summanden zusammen mit dem Ergebnis jeweils in Hexadezimaldarstellung ausgibt.

Beispiel:

```
a = 0x 06887bea 57dfe177 12b373c7 7db3af4b
b = 0x 1c550cd5 22578a3f 0e1ba7e0 44110127
c = 0x022dd88bf 7a376bb6 20cf1ba7 c1c4b072
```

Hinweis: Um eine Zahl x in n-stelliger Hexadezimaldarstellung auszugeben, kannst du die Anweisung `printf("%.*x", n, x)` verwenden. Diese Anweisung erzeugt z.B. für $x = 3$ und $n = 8$ die Ausgabe `00000003`.

Aufgabe 3: Spielkarten aufnehmen

(9 Punkte, Abgabedatei `karten.c`)

Datenstruktur: Wir verwenden ein Kartenblatt mit 32 Karten, in dem jede Kombination aus Farbe (Karo, Herz, Pik, Kreuz) und Wert (7, 8, 9, 10, Bube, Dame, König, As) genau einmal vorkommt. Es bleibt dir überlassen, welche Darstellung du für eine solche Spielkarte wählst.

Karten aufnehmen: Das Programm soll nacheinander 10 zufällige Karten ziehen, wobei nicht zweimal die gleiche Karte vorkommen darf. Eine gezogene Karte wird sofort in eine verkettete Liste einsortiert. Dabei soll zunächst nach Farbe und dann nach Wert sortiert werden, wobei die obige Reihenfolge der Farben und Werte gilt (also z.B. Karo < Herz und 10 < Bube).

Liste ausgeben: Wenn alle zehn Karten gezogen wurden, soll der Inhalt der sortierten Liste am Bildschirm ausgegeben werden. Dabei sind in jedem Fall die oben angegebenen Bezeichnungen für die Spielkarten zu verwenden (also z.B. „Herz Dame“ oder „Pik As“).