Dirk Praetorius,
Marcus Page,
Thomas Führer

Sommersemester 2013
12.03.2013

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 2

Aufgabe 2.1. Schreiben Sie eine `void`-Funktion `vektorprodukt`, die zu gegebenen Vektoren $\mathbf{u} = (a, b, c)^T$ und $\mathbf{v} = (x, y, z)^T$ das Vektorprodukt $\mathbf{w} = \mathbf{u} \times \mathbf{v}$ mit

$$w_1 = bz - cy$$
$$w_2 = cx - az$$
$$w_3 = ay - bx$$

berechnet und ausgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Parameter a, b, c, x, y, z eingelesen und die Funktion aufgerufen werden. Speichern Sie den Source-Code unter `vektorprodukt.c` in das Verzeichnis `serie02`.

Aufgabe 2.2. Schreiben Sie eine Funktion `evenorodd(n)`, die den Wert 1 zurückgibt, wenn $n \in \mathbb{N}$ gerade ist, und 0, wenn n ungerade ist. Schreiben Sie ein Hauptprogramm, das den Wert n von der Tastatur einliest und am Bildschirm ausgibt, ob n gerade oder ungerade ist. Speichern Sie den Source-Code unter `evenorodd.m` in das Verzeichnis `serie02`.

Aufgabe 2.3. Schreiben Sie eine Funktion `dabs`, die für $x \in \mathbb{R}$ den Absolutbetrag $|x|$ zurückliefert. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem x über die Tastatur eingelesen und $|x|$ ausgegeben werden. Speichern Sie den Source-Code unter `dabs.c` in das Verzeichnis `serie02`.

Aufgabe 2.4. Schreiben Sie eine Funktion `folgenglied`, die für gegebenes $n \in \mathbb{N}$ das Folgenglied $a_n := (-1)^n/n$ zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem n eingelesen und a_n ausgegeben werden. Speichern Sie den Source-Code unter `folgenglied.c` in das Verzeichnis `serie02`.

Aufgabe 2.5. Schreiben Sie eine `void`-Funktion `quadrant`, die für einen Punkt $(x, y) \in \mathbb{R}^2$ ausgibt, ob (x, y) auf einer der Achsen des Koordinatensystems liegt. Falls nicht, soll ausgegeben werden, in welchem Quadranten (x, y) liegt. Schreiben Sie ferner ein Hauptprogramm, in dem $x, y \in \mathbb{R}$ eingelesen werden. Speichern Sie den Source-Code unter `quadrant.c` in das Verzeichnis `serie02`.

Aufgabe 2.6. Erklären Sie den Begriff *Rekursion*. Welche Komponenten muss eine rekursive Funktion mindestens haben?

Aufgabe 2.7. Schreiben Sie eine rekursive Funktion `binomial`, die den Binomialkoeffizienten $\binom{n}{k}$ berechnet. Verwenden Sie dazu das Additionstheorem $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$. Schreiben Sie ein aufrufendes Hauptprogramm, in dem $k, n \in \mathbb{N}_0$ mit $k \leq n$ eingelesen und $\binom{n}{k}$, berechnet und ausgeben werden. Speichern Sie den Source-Code unter `binomial.c` in das Verzeichnis `serie02`.

Aufgabe 2.8. Nach einer alten Legende gibt es in Hanoi einen Tempel, in dem sich folgende rituelle Anordnung befindet: Es handelt sich um 3 Pfosten und um $n = 64$ goldene Scheiben, die in der Mitte ein Loch haben, so dass sie auf die Pfosten gestapelt werden können. Die 64 Scheiben haben paarweise verschiedenen Durchmesser. Als der Tempel erbaut wurde, wurden diese Scheiben der Größe nach aufsteigend auf den ersten Pfosten gestapelt. Die Aufgabe der Priester in diesem Tempel besteht darin, diese Scheiben systematisch unter Zuhilfenahme des zweiten Pfostens so umzustapeln, dass sich diese am Schluss in derselben Anordnung wie zu Beginn auf dem dritten Pfosten befinden. Dabei sind folgende Regeln zu beachten:

- Die Scheiben dürfen nur einzeln verschoben werden.
- Eine Scheibe darf nie auf einer anderen Scheibe kleineren Durchmessers zu liegen kommen.

Das Ende der Welt, so sagt die Legende, ist durch denjenigen Zeitpunkt vorherbestimmt, an dem die Priester diese Aufgabe erfüllt haben werden.
Ein rekursiver Algorithmus, der dieses Problem löst, lässt sich wie folgt formulieren: Um die obersten m auf Pfosten i befindlichen Scheiben auf Pfosten j zu verschieben, verschiebt man

1. die obersten $m-1$ Scheiben von Pfosten i auf Pfosten $k \notin \{i, j\}$,
2. die grösste der besagten m Scheiben von Pfosten i auf Pfosten j,
3. und schliesslich die $m-1$ in Schritt 1 auf Pfosten k verschobenen Scheiben auf Pfosten j.

Man beginnt mit $m = n$, $i = 1$ und $j = 3$. Man entwickle eine rekursive Funktion, die diesen Algorithmus implementiert. Jeder einzelne Schritt soll dabei am Display protokolliert werden. Zum Beispiel:

```
Scheibe 1 wandert vom 2. zum 3. Pfosten
```

Zum Testen verwende man $n \ll 64$, z.B. $n = 3, 4, 5$. Speichern Sie den Source-Code unter `hanoi.c` in das Verzeichnis `serie02`.