Dirk Praetorius,
Marcus Page

Sommersemester 2012
12.03.2012

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 2

In der aktuellen Übung, sollen die Konzepte *Verzweigung* (Aufgaben 1, 2, 3, 5), *Zählschleifen* (Aufgaben 3, 4, 8, 10), *Rekursion* (Aufgaben 4, 7, 9) sowie der Umgang mit der mathematischen Bibliothek (Aufgabe 6) geübt und vertieft werden. Sie sollten hierzu auch die Folien aus der Vorlesung nochmals durchgehen.

Aufgabe 2.1*. Schreiben Sie eine `void`-Funktion `teiler`, die für eine gegebene Zahl $x \in \mathbb{N} := \{1, 2, 3, \dots\}$ ausgibt, ob diese durch 2, durch 3 oder durch 6 teilbar ist. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Integer x einliest und `teiler` aufruft. Speichern Sie den Source-Code unter `teiler.c` in das Verzeichnis `serie02`.

Aufgabe 2.2*. Schreiben Sie eine Funktion `folgenglied`, die für gegebenes $n \in \mathbb{N}$ das Folgenglied $a_n := (-1)^n/n$ zurückgibt. Hierbei soll `pow` aus der mathematischen Bibliothek *nicht* verwendet werden. 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 2.3*. Ein Tripel $(x, y, z) \in \mathbb{N}^3$ natürlicher Zahlen heißt *pythagoräisches Zahlentripel*, falls $x^2+y^2 = z^2$ gilt. Das wohl bekannteste Beispiel ist $(3, 4, 5)$. Offensichtlich gelten $z > \max\{x, y\}$ sowie $x \neq y$ und ohne Beschränkung der Allgemeinheit ferner $x < y$. Schreiben Sie eine `void`-Funktion `pythagoras`, die zu gegebener Schranke $n \in \mathbb{N}$ alle pythagoräischen Zahlentripel mit $x < y < z \leq n$ bestimmt und ausgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Schranke n eingelesen und `pythagoras` aufgerufen wird. Speichern Sie den Source-Code unter `pythagoras.c` in das Verzeichnis `serie02`.

Aufgabe 2.4*. Die Fibonacci-Folge ist definiert durch $x_0 := 0$, $x_1 := 1$ und $x_{n+1} = x_n + x_{n-1}$. Schreiben Sie eine rekursive Funktion `fibonacciRek`, die zu gegebenem Index n das Folgenglied x_n zurückgibt. Schreiben Sie weiters eine nicht rekursive Funktion `fibonacci`, die dasselbe leistet, wobei die Berechnung aber über geeignete Schleifen realisiert wird. Welche der beiden Funktionen ist effizienter und warum? Speichern Sie den Source-Code unter `fibonacci.m` in das Verzeichnis `serie02`.

Aufgabe 2.5. Schreiben Sie eine Funktion `rundung`, die für eine gegeben Zahl $x \in \mathbb{R}$ die Zahl $n \in \mathbb{N}$ zurückliefert, die x am nächsten liegt. Falls x genau in der Mitte zwischen zwei ganzen Zahlen liegt, werde die größere zurückgeliefert. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das die Zahl x einliest und gerundet ausgibt. Speichern Sie den Source-Code unter `rundung.c` in das Verzeichnis `serie02`.

Aufgabe 2.6. Schreiben Sie eine Funktion `pol2cart`, die gegebene Polarkoordinaten (r, φ) in kartesische Koordinaten umrechnet. Verwenden Sie hierbei die mathematische Bibliothek. Speichern Sie den Source-Code unter `pol2cart.c` in das Verzeichnis `serie02`.

Aufgabe 2.7. 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`.

Aufgabe 2.8. Für $p \in [1, \infty)$ ist die ℓ_p-Norm auf $\mathbb{R}^n$ definiert durch

$$\|x\|_p := \Big(\sum_{j=1}^{n} |x_j|^p \Big)^{1/p}.$$

Schreiben Sie eine Funktion `pnorm`, die einen Vektor $x \in \mathbb{R}^5$ (d.h. 5 reelle Zahlen) sowie $p \in [1, \infty)$ übernimmt und $\|x\|_p$ zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm in dem x und p eingelesen werden und $\|x\|_p$ ausgegeben wird. Testen Sie Ihr Programm mit verschiedenen Werten. Speichern Sie den Source-Code unter `pnorm.c` in das Verzeichnis `serie02`.

Aufgabe 2.9. 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 ferner eine Funktion `binomial2`, die den Binomialkoeffizienten mittels $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ berechnet. Hierzu können Sie die rekursive Funktion `faktorielle` aus der Vorlesung, die zu gegebenem $n \in \mathbb{N}$ die Faktorielle $n!$ berechnet, verwenden. Schreiben Sie ein aufrufendes Hauptprogramm, in dem $k, n \in \mathbb{N}_0$ mit $k \leq n$ eingelesen und $\binom{n}{k}$, berechnet auf beide Weisen, ausgeben werden. Speichern Sie den Source-Code unter `binomial.c` in das Verzeichnis `serie02`.

Aufgabe 2.10. Schreiben Sie eine nicht-rekursive Funktion `binomial`, die den Binomialkoeffizienten $\binom{n}{k}$ berechnet. Dazu realisiere man die gekürzte Form $\binom{n}{k} = \frac{n \cdot (n-1) \cdots (n-k+1)}{k \cdot (k-1) \cdots 1}$ mittels geeigneter Schleifen. Schreiben Sie ein aufrufendes Hauptprogramm, in dem $k, n \in \mathbb{N}_0$ mit $k \leq n$ eingelesen werden und $\binom{n}{k}$ ausgeben wird. Speichern Sie den Source-Code unter `binomial.c` in das Verzeichnis `serie02`. Welche der drei Implementierungen (vgl. Aufgabe 2.9) führt auf die effizienteste Variante und warum? Testen Sie ihre Programme mit $n = 14$.