Dirk Praetorius,
Michele Ruggeri

Sommersemester 2017
28.03.2017

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 4

Aufgabe 4.1. Schreiben Sie die Funktion `int = binomial(int n, int k, int type)` die den Binomialkoeffizienten $\binom{n}{k}$ berechnet und zurückgibt. Dies läßt sich auf verschiedene Weisen realisieren:

- direkt in der Form $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ unter Verwendung einer Funktion für die Faktorielle (`type=1`),
- in gekürzter Form $\binom{n}{k} = \frac{n\cdot(n-1)\cdots(n-k+1)}{k\cdot(k-1)\cdots 1}$ mittels geeigneter Schleifen (`type=2`),
- mittels einer rekursiven Funktion, die das Additionstheorem $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ benutzt (`type=3`).

Realisieren Sie alle drei Varianten und zusätzlich ein Hauptprogramm, das die Zahlen n und k über die Tastatur einliest und den Binomialkoeffizien berechnet und ausgibt. Speichern Sie den Source-Code unter `binomial.c` in das Verzeichnis `serie04`.

Aufgabe 4.2. 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 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.
- In jedem Schritt wird die oberste Scheibe eines Stapels aud einen der anderen Stapel gesetzt. D.h. es kann jeweils nur die erste Scheibe bewegt 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.
Sei n die Anzahl der Scheiben ($n = 64$ in der Legende). Ein rekursiver Algorithmus, der dieses Problem löst, lässt sich wie folgt formulieren: Um die obersten $m \leq n$ 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.

Die Wahl $m = n$, $i = 1$ und $j = 3$ löst das Problem. Schreiben Sie eine rekursive Funktion `void hanoi(int m, int i, int j)` die diesen Algorithmus implementiert. Jeder einzelne Schritt soll dabei am Display protokolliert werden. Zum Beispiel:

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

Schreiben Sie auch ein Hauptprogramm, das n über die Tastatur einliest und die Liste der Schritte ausgibt. Zum Testen verwende man $n \ll 64$, z.B. $n = 3, 4, 5$. Speichern Sie den Source-Code unter `hanoi.c` in das Verzeichnis `serie04`.
Fakultativ (1 bonus-Punkt): Es lässt sich zeigen, dass eine optimale Lösung des Problems $2^n - 1$ Schritte benötigt. Ist die obige Rekursion optimal?

Aufgabe 4.3. Schreiben Sie eine Funktion `double powN(double x, int n)`, welche x^n für einen ganzzahligen Exponenten $n \in \mathbb{Z}$ berechnet. Es gilt $x^0 = 1$ für alle $x \in \mathbb{R}\setminus\{0\}$ und für $n < 0$ gilt $x^n = (1/x)^{-n}$. Weiters gilt $0^n = 0$ für $n > 0$. Die Potenz 0^n ist für $n \leq 0$ nicht definiert. Die Funktion soll in diesem Fall den Wert `0.0/0.0` zurückgeben. Für diese Aufgabe dürfen Sie die Funktion `pow` aus der Mathematikbibliothek nicht verwenden. Speichern Sie den Source-Code unter `powN.c` in das Verzeichnis `serie04`.

Aufgabe 4.4. 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 `serie04`.

Aufgabe 4.5. Schreiben Sie eine `void`-Funktion `vielfache(k,nmax)`, die alle ganzzahligen Vielfachen der Zahl $k \in \mathbb{N}$, die $\leq n_{\max} \in \mathbb{N}$ sind, am Bildschirm ausgibt. Die Ausgabe erfolge zeilenweise in der Form

```
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
```

beispielsweise für den Fall $k = 5$ und $n_{\max} = 19$. Ferner schreibe man ein Hauptprogramm, das die Daten k und n von der Tastatur einliest und `vielfache(k,nmax)` aufruft. Speichern Sie den Source-Code unter `vielfache.c` in das Verzeichnis `serie04`.

Aufgabe 4.6. Schreiben Sie ein main-Programm, das $n \in \mathbb{N}$ über die Tastatur einliest und die ersten n Stufen des Pascal'schen Dreiecks ausgibt: Jede Zeile dieses Schemas beginnt und endet mit 1. Die restlichen Zahlen werden als Summe nebeneinanderstehender Zahlen der vorhergegangenen Zeile gebildet. Für $n = 5$ sieht das Pascal'sche Dreieck dann folgendermaßen aus:

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

Siehe auch:

`http://de.wikipedia.org/wiki/Pascalsches_Dreieck`

Speichern Sie den Source-Code unter `pascal.c` in das Verzeichnis `serie04`.

Aufgabe 4.7. Schreiben Sie eine Funktion `geometricMean`, die von einem gegebenem Vektor $x \in \mathbb{R}^n_{\geq 0}$ den geometrischen Mittelwert

$$\overline{x}_{\text{geom}} = \sqrt[n]{\prod_{j=1}^{n} x_j}$$

berechnet und zurückgibt. Die Länge $n \in \mathbb{N}$ soll eine Konstante im Hauptprogramm sein, die Funktion `geometricMean` ist aber für beliebige Längen zu programmieren. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das x über die Tastatur einliest und den geometrischen Mittelwert berechnet und ausgibt. Speichern Sie den Source-Code unter `geometricMean.c` in das Verzeichnis `serie04`.

Aufgabe 4.8. Die Frobeniusnorm einer Matrix $A \in \mathbb{R}^{m\times n}$ ist durch

$$\|A\|_F := \Big(\sum_{j=1}^{m}\sum_{k=1}^{n} A_{jk}^2\Big)^{1/2}$$

definiert. Schreiben Sie eine Funktion `frobeniusnorm`, die für gegebene Matrix A und gegebene Dimensionen $m, n \in \mathbb{N}$ die Frobeniusnorm berechnet und zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Zeilen- und Spaltendimensionen $m, n \in \mathbb{N}$ und A eingelesen werden und $\|A\|_F$ ausgegeben wird. Die Matrix A soll spaltenweise gespeichert werden (EPROG Folien, Seite 75). Die Dimensionen $m, n \in \mathbb{N}$ sollen zwei Konstanten im Hauptprogramm sein, die Funktion `frobeniusnorm` ist aber für beliebige Dimensionen zu programmieren. Speichern Sie den Source-Code unter `frobeniusnorm.c` in das Verzeichnis `serie04`.