Dirk Praetorius,
Thomas Führer,
Michael Feischl

Wintersemester 2013/14
22.10.2013

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 4

Aufgabe 4.1. Schreiben Sie eine nicht-rekursive Funktion `binomial`, die den Binomialkoeffizienten $\binom{n}{k}$ berechnet. Dazu realisiere man die gekürzte Form $\binom{n}{k} = \frac{n\cdot(n-1)\cdots(n-k+1)}{1\cdot 2\cdots k} = \frac{n}{1}\cdot\frac{n-1}{2}\cdots\frac{n-k+1}{k}$ 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 `serie04`.

Aufgabe 4.2. 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 4.3. 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 4.4. Schreiben Sie eine Funktion `energy`, die für einen gegebenen Vektor $x \in \mathbb{R}^n$ die Energie $e = \sum_{j=1}^n x_j^2$ zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor x einliest und die Energie ausgibt. Die Dimension $n \in \mathbb{N}$ soll eine Konstante im Hauptprogramm sein, die Funktion `energy` ist für beliebige Dimension zu programmieren. Speichern Sie den Source-Code unter `energy.c` in das Verzeichnis `serie04`.

Aufgabe 4.5. Schreiben Sie eine Funktion `mean`, die von einem gegebenem Vektor $x \in \mathbb{Z}^n$ den Mittelwert $\frac{1}{n}\sum_{j=1}^n x_j$ berechnet und zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor $x \in \mathbb{Z}^n$ einliest und den Mittelwert ausgibt. Die Länge $n \in \mathbb{N}$ des Vektors soll eine Konstante im Hauptprogramm sein die Funktion `mean` ist für beliebige Länge n zu programmieren. Speichern Sie den Source-Code unter `mean.c` in das Verzeichnis `serie04`.

Aufgabe 4.6. 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 $x \in \mathbb{R}^n$, dessen Länge n 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. Die Dimension $n \in \mathbb{N}$ soll eine Konstante im Hauptprogramm sein, die Funktion `pnorm` soll aber beliebige Dimension zulassen. Testen Sie Ihr Programm mit verschiedenen Werten für p bei festem Vektor x. Was beobachten Sie für $p \to \infty$? Speichern Sie den Source-Code unter `pnorm.c` in das Verzeichnis `serie04`.

Aufgabe 4.7. Schreiben Sie eine Funktion `maxcount`, die von einem Vektor $x \in \mathbb{R}^n$ das Maximum berechnet und die Anzahl zurückliefert, wie oft dieses im Vektor vorkommt. Speichern Sie den Source-Code unter `maxcount.c` in das Verzeichnis `serie04`.

Aufgabe 4.8. Folgendes Programm soll den maximalen Eintrag einer gegebenen Matrix bestimmen. Als Ergebnis wird `5.0000` ausgegeben. Wo liegt der Fehler?

```
#include <stdio.h>

main() {
  double A[2][3] = { {1,2,3},{6,-4,5} };
  double max = A[0][0];

  int j=0, k=0;

  for(j=0; j<2; j=j+1) {
    for(k=1; k<3; k=k+1) {
      if(A[j][k] > max) {
        max = A[j][k];
      }
    }
  }
  printf("Maximum = %f\n",max);
}
```

Beheben Sie den Fehler und erweitern Sie das Programm so, dass auch das Minimum aller Matrixeinträge bestimmt wird. Speichern Sie den Source-Code unter `maxminmatrix.c` in das Verzeichnis `serie04`.