Dirk Praetorius,
Thomas Führer,
Michele Ruggeri

Wintersemester 2014/15
28.10.2014

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 5

Aufgabe 5.1. 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 `serie05`.

Aufgabe 5.2. 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 $x \in \mathbb{R}$ und für $n < 0$ gilt $x^n = (1/x)^{-n}$. Weiters gilt $0^n = 0$ für $n > 0$. Die Potenz 0^n ist für $n < 0$ 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 `serie05`.

Aufgabe 5.3. Die Fibonacci-Folge ist definiert durch $x_0 := 0$, $x_1 := 1$ und $x_{n+1} := x_n + x_{n-1}$. Schreiben Sie eine *nicht-rekursive* Funktion `fibonacci(k)`, die zu gegebenem Index k das Folgenglied x_k berechnet und zurückgibt. Schreiben Sie ferner ein Hauptprogramm, das k von der Tastatur einliest und x_k am Bildschirm ausgibt. Speichern Sie den Source-Code unter `fibonacci.c` in das Verzeichnis `serie05`.

Aufgabe 5.4. Um die Summe $\sum_{j=1}^{n}(-1)^j/j$ zu berechnen, ist es numerisch günstig, zunächst die negativen und die positiven Beiträge getrennt zu summieren und erst abschließend beide Teilsummen zu addieren. Warum? Schreiben Sie eine Funktion `sum`, die dieses Vorgehen realisiert. Schreiben Sie ferner ein Hauptprogramm, das n einliest und $\sum_{j=1}^{n}(-1)^j/j$ ausgibt. Speichern Sie den Source-Code unter `sum.c` in das Verzeichnis `serie05`.

Aufgabe 5.5. 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. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor $x \in \mathbb{R}^n$ einliest und den geometrischen Mittelwert ausgibt. Die Länge $n \in \mathbb{N}$ des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `geometricMean` ist für beliebige Länge n zu programmieren. Speichern Sie den Source-Code unter `geometricMean.c` in das Verzeichnis `serie05`.

Aufgabe 5.6. Schreiben Sie eine Funktion `minmaxmean`, die von einem gegebenem Vektor $x \in \mathbb{N}^n$ das Minimum, das Maximum und den Mittelwert $\frac{1}{n}\sum_{j=1}^{n} x_j$ berechnet und geeignet zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor $x \in \mathbb{N}^n$ einliest und Minimum, Maximum und Mittelwert ausgibt. Die Länge $n \in \mathbb{N}$ des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `minmaxmean` ist für beliebige Länge n zu programmieren. Speichern Sie den Source-Code unter `minmaxmean.c` in das Verzeichnis `serie05`.

Aufgabe 5.7. Schreiben Sie eine Funktion `skalarprodukt`, die zu gegebenen Vektoren $x, y \in \mathbb{R}^n$ das Skalarprodukt $x \cdot y := \sum_{j=1}^{n} x_j y_j$ berechnet. Ferner schreibe man ein aufrufendes Hauptprogramm, das die Vektoren x und y einliest und $x \cdot y$ ausgibt. Die Länge n der Vektoren soll eine Konstante im Hauptprogramm sein, die Funktion `skalarprodukt` ist für beliebige Länge n zu programmieren. Speichern Sie den Source-Code unter `skalarprodukt.c` in das Verzeichnis `serie05`.

Aufgabe 5.8. 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 `serie05`.