Dirk Praetorius,
Michele Ruggeri

Wintersemester 2015/16
04.-06.11.2015

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 4

Aufgabe 4.1. 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 `serie04`. Vergleichen Sie Ihre Implementierung mit Ihrem Code aus Aufgabe 3.5. Diskutieren Sie Vor- und Nachteile der beiden Implementierungen!

Aufgabe 4.2. 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 4.3. 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 `serie04`.

Aufgabe 4.4. 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 4.5. 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. Die Länge $n \in \mathbb{N}$ soll eine Konstante im Hauptprogramm sein, die Funktion `skalarprodukt` ist aber für beliebige Längen zu programmieren. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem $x, y \in \mathbb{R}^n$ eingelesen werden und `skalarprodukt` aufgerufen wird. Speichern Sie den Source-Code unter `skalarprodukt.c` in das Verzeichnis `serie04`.

Aufgabe 4.6. 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, in dem `geometricMean` aufgerufen wird. Speichern Sie den Source-Code unter `geometricMean.c` in das Verzeichnis `serie04`.

Aufgabe 4.7. Schreiben Sie eine Funktion `maxcompare`, die für zwei gegebene Vektoren $a, b \in \mathbb{R}^n$ zählt wie oft das Maximum von a und b mit der Bezeichnung $M := \max\{a_i, b_i \mid i = 1, \ldots, n\}$ im Vektor a und b an der gleichen Stelle vorkommt. Zum Beispiel soll die Funktion für die Vektoren $a = (1.1, 4, 2e-4, 4, 3, 4, -1.5)$ und $b = (2.2, 4, 4, 2e-5, 4, -1, 2.7, 4)$ den Wert 2 zurückgeben, da das Maximum $M = 4$ in beiden Vektoren an zwei Stellen nämlich $a_2 = b_2 = a_5 = b_5 = M$, gleichermaßen vorkommt. Wenn etwa M nur entweder in a oder b vorkommt, dann soll die Funktion klarerweise 0 zurückgeben. Die Länge $n \in \mathbb{N}$ soll eine Konstante im Hauptprogramm sein, die Funktion `maxcompare` ist aber für beliebige Längen zu programmieren. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem $a, b \in \mathbb{R}^n$ eingelesen werden und `maxcompare` aufgerufen wird. Speichern Sie den Source-Code unter `maxcompare.c` in das Verzeichnis `serie04`.

Aufgabe 4.8. Schreiben Sie ein `main`-Programm, das die ersten k 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 $k = 5$ 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`

Realisieren Sie das Pascal'sche Dreieck möglichst rechenökonomisch, insbesondere ohne die Darstellung der Einträge über den Binomialkoeffizienten. Speichern Sie immer nur *eine* Zeile in *einem* Vektor der Länge k und überlegen Sie sich, wie Sie den Vektor in jedem Schritt geeignet überschreiben. Die Länge $k \in \mathbb{N}$ soll eine Konstante im Hauptprogramm sein. Speichern Sie den Source-Code unter `pascal.c` in das Verzeichnis `serie04`.