Dirk Praetorius,
Gregor Gantner

Wintersemester 2018/19
22./23.11.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 4

Aufgabe 4.1. Schreiben Sie eine nicht-rekursive 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 4.2. Gegeben sei eine 10-stellige Zahlenfolge x (statisches Array vom Typ `int`) und eine 3-stellige Zahlenkombination y (Array vom Typ `int`), die jeweils von der Tastatur eingelesen werden. Man schreibe eine Funktion `check`, die die beiden Arrays bekommt und überprüft, ob die Zahlenkombination y in der Zahlenfolge x vorkommt (Rückgabewert 1), oder nicht (Rückgabewert 0). Zusätzlich schreibe man ein aufrufendes Hauptprogramm, in dem die Felder x und y eingelesen werden. Wie haben Sie Ihren Code auf Korrektheit getestet? Speichern Sie den Source-Code unter `check.c` in das Verzeichnis `serie04`.

Aufgabe 4.3. Schreiben Sie eine Funktion `geometricMean`, die von einem gegebenem Vektor $x \in \mathbb{R}^n_{\geq 0}$ den geometrischen Mittelwert

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

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 4.4. 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 `serie04`.

Aufgabe 4.5. 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. 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`.

Aufgabe 4.6. Gegeben seien die Summen

$$a_N := \sum_{n=0}^{N} \frac{1}{(n+1)^2} \quad \text{und} \quad b_N := \sum_{n=0}^{N} \sum_{k=0}^{n} \frac{1}{(k+1)^2(n-k+1)^2}.$$

Schreiben Sie zwei Funktionen, welche für gegebene $N \in \mathbb{N}$ die Zeit messen, um $(a_N)^2$ bzw. b_N zu berechnen. Wie groß ist der Aufwand bei der Berechnung von $(a_N)^2$ bzw. b_N? Z.B.: Falls die Funktionen für $N = 10^3$ eine Laufzeit von 3 Sekunden haben, welche Laufzeit erwarten Sie aufgrund des Aufwands für $N = 10^4$? Schreiben Sie ferner ein Hauptprogramm, welches für verschiedene Werte von N die Ergebnisse in Form einer Tabelle am Bildschirm ausgibt. Entsprechen die Resultate Ihren Erwartungen? Wie haben Sie Ihr Programm getestet? Speichern Sie den Source-Code unter `zeitmessung.c` in das Verzeichnis `serie04`.

Aufgabe 4.7. Die Fibonacci-Folge ist definiert durch $x_0 := 0$, $x_1 := 1$ und $x_{n+1} := x_n + x_{n-1}$ für $n \geq 1$. 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`. Wie groß ist der Aufwand zur Berechnug von x_k? Vergleichen Sie Ihre Implementierung mit Ihrem Code aus Aufgabe 3.5. Diskutieren Sie Vor- und Nachteile der beiden Implementierungen!

Aufgabe 4.8. *Bubblesort* ist ein Sortier-Algorithmus: Man vergleicht aufsteigend jedes Element eines Arrays x_j mit seinem Nachfolger x_{j+1} und – falls notwendig – vertauscht die beiden. Nach dem ersten Durchlauf muss zumindest das letzte Element bereits am richtigen Platz sein. Der nächste Durchlauf muss also nur noch bis zur vorletzten Stelle gehen, usw. Wie viele geschachtelte Schleifen braucht dieses Vorgehen? Schreiben Sie eine Funktion `bubblesort`, die ein gegebenes Array $x \in \mathbb{R}^n$ mittels Bubble-Sort aufsteigend sortiert, d.h. $x_1 \leq x_2 \leq \cdots \leq x_n$. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor x einliest und in sortierter Reihenfolge ausgibt. Die Länge des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `bubblesort` ist für beliebige Länge n zu programmieren. Bestimmen Sie den Aufwand Ihrer Funktion. Speichern Sie den Source-Code unter `bubblesort.c` in das Verzeichnis `serie04`.