Dirk Praetorius,
Michele Ruggeri

Wintersemester 2016/17
27.10.2016

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 $x_{n+1} := x_n + x_{n-1}$ 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 4.2. 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 4.3. 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 4.4. Schreiben Sie eine Funktion `maxabs`, die von einem gegebenem Vektor $x \in \mathbb{R}^n$ das erste Element x_j mit maximalem Betrag berechnet und zurückgibt, d.h. $|x_j| = \max\{|x_i| \,:\, i = 1, \dots, n\}$ und für alle x_i mit $|x_i| = |x_j|$ gilt $i \geq j$. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor x einliest und das Ergebnis von `maxabs` ausgibt. Der Vektor x soll dabei mittels statischem Array realisiert werden. Die Länge des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `maxabs` ist aber für beliebige Länge zu implementieren. Speichern Sie den Source-Code unter `maxabs.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 `maxcompare`, die für zwei gegebene Vektoren $a, b \in \mathbb{R}^n$ zählt, wie oft das Maximum von a und b mit der Bezeichnung $M := \max\{a_i, b_i \,|\, i = 1, \dots, n\}$ im Vektor a und b an der gleichen Stelle vorkommt. Zum Beispiel soll die Funktion für die Vektoren $a = (1.1, 4, 2e-4, 4, 4, 3, 4, -1.5)$ 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 4.7. Schreiben Sie eine Funktion `eratosthenes`, die das *Sieb des Eratosthenes* realisiert. Dies ist ein Algorithmus, mit dem alle Primzahlen bis zu einer bestimmten Zahl `nmax` berechnet werden können (benannt nach dem griechischen Mathematiker Eratosthenes). Der Algorithmus sieht folgendermaßen aus:

- Man legt eine Liste (Vektor) $\texttt{prim} = (2, \ldots, \texttt{nmax}) \in \mathbb{N}^{\texttt{nmax}-1}$ an.
- Man streicht aus der Liste alle Vielfachen der ersten Zahl (also der Zahl 2).
- Wähle, solange es noch höhere Zahlen gibt, die nächsthöhere nicht durchgestrichene Zahl und streiche alle ihre Vielfachen.

Am Ende sollen alle Primzahlen von $2, \ldots,$`nmax` ausgegeben werden. Geben Sie außerdem die Anzahl der gefundenen Primzahlen mit aus. Realisieren Sie das Streichen, indem Sie die entsprechenden Einträge auf 0 setzen. Die Zahl `nmax` soll eine Konstante im Hauptprogramm sein. Speichern Sie den Source-Code unter `eratosthenes.c` in das Verzeichnis `serie04`. Wie groß ist der Aufwand zur Berechnung in Abhängigkeit von `nmax`?

Aufgabe 4.8. Gegeben seien die Summen

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

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`.