Dirk Praetorius,
Michele Ruggeri

Wintersemester 2015/16
18.-20.11.2015

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 6

Aufgabe 6.1. Schreiben Sie eine Funktion `kgV(a,b)`, die das kleinste gemeinsame Vielfache zweier natürlicher Zahlen $a, b \in \mathbb{N}$ berechnet. Zur Lösung können Sie entweder die Primfaktoren beider Zahlen berechnen oder den Zusammenhang $a \cdot b = ggT(a,b) \cdot kgV(a,b)$ berücksichtigen. Speichern Sie den Source-Code unter `kgv.c` in das Verzeichnis `serie06`.

Aufgabe 6.2. Alternativ zum Bisektionsverfahren aus der Vorlesung kann eine Nullstelle von $f : [a,b] \to \mathbb{R}$ auch mit dem *Sekantenverfahren* berechnet werden. Dabei sind x_0 und x_1 gegebene Startwerte und man definiert induktiv x_{n+1} als Nullstelle der Geraden durch $(x_{n-1}, f(x_{n-1}))$ und $(x_n, f(x_n))$, d.h.

$$x_{n+1} := x_n - f(x_n)\,\frac{x_{n-1} - x_n}{f(x_{n-1}) - f(x_n)}$$

Schreiben Sie eine Funktion `sekante(x0,x1,tau)` die die Folge der Iterierten berechnet, bis entweder

$$|f(x_n) - f(x_{n-1})| \le \tau$$

oder

$$|f(x_n)| \le \tau \quad \text{und} \quad |x_n - x_{n-1}| \le \begin{cases} \tau & \text{für } |x_n| \le \tau, \\ \tau|x_n| & \text{sonst} \end{cases}$$

gilt. Es werde dann x_n als Approximation einer Nullstelle z_0 von f zurückgegeben. Im ersten Fall gebe man zusätzlich eine Warnung aus, dass das numerische Ergebnis vermutlich falsch ist.
Verwenden Sie ein geeignetes Beispiel zum Test. Schreiben Sie ein aufrufendes Hauptprogramm, in dem x_0 und x_1 eingelesen werden und x_n ausgegeben wird. Speichern Sie den Source-Code unter `sekante.c` in das Verzeichnis `serie06`.

Aufgabe 6.3. Man erweitere *MinSort* aus der Vorlesung um einen Parameter `type`, sodass die Funktion einen Vektor $x \in \mathbb{R}^n$ wahlweise aufsteigend (`type = 1`) oder absteigend (`type = -1`) sortiert. Schreiben Sie ein aufrufendes Hauptprogramm, in dem $x \in \mathbb{R}^n$ und die Sortierrichtung `type` eingegeben werden und der sortierte Vektor ausgegeben wird. Die Länge n des Vektors soll eine Konstante im Hauptprogramm sein, Ihre Implementierung von *MinSort* soll aber für beliebige Länge n funktionieren. Speichern Sie den Source-Code unter `minsort.c` in das Verzeichnis `serie06`.

Aufgabe 6.4. *Bubble-Sort* ist ein ineffizienter, aber kurzer 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. $x_1 \le x_2 \le \cdots \le x_n$, und zurückgibt. 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. Speichern Sie den Source-Code unter `bubblesort.c` in das Verzeichnis `serie06`.

Aufgabe 6.5. Implementieren Sie den *Quicksort*-Algorithmus, um einen Vektor $x \in \mathbb{R}^n$ zu sortieren: *Quicksort* wählt willkürlich ein Pivotelement aus der zu sortierenden Liste x, z.B. x_1. Dann zerlegt man die Liste in zwei Teillisten $x^{(<)}$ und $x^{(\ge)}$ und und das Pivotelement x_1: $x^{(<)}$ enthält dabei alle Elemente $< x_1$, $x^{(\ge)}$ enthält nur Elemente $\ge x_1$. $x^{(<)}$ und $x^{(\ge)}$ werden rekursiv sortiert. Anschließend

wird das Ergebnis zusammengesetzt. Eine direkte Implementierung dieses Algorithmus hat allerdings den Nachteil, dass zusätzlicher Speicher benötigt wird. Um dies zu vermeiden, gehen Sie nun folgendermaßen vor: Beginnend mit $j = 2$ sucht man ein Element $x_j \geq x_1$, d.h. x_j gehört zu $x^{(\geq)}$. Ferner sucht man beginnend bei $k = n$ ein Element $x_k < x_1$, d.h. x_k gehört zu $x^{(<)}$. In diesem Fall vertauscht man x_j und x_k. Wenn sich die Zähler j und k treffen, liegt die Liste x in der Form $(x_1, x^{(<)}, x^{(\geq)})$ vor. Mit einer weiteren Vertauschung erreicht man sofort die Gestalt $(x^{(<)}, x_1, x^{(\geq)})$. Es müssen nur noch $x^{(<)}$ und $x^{(\geq)}$ rekursiv sortiert werden. Schreiben darüberhinaus ein `main`-Programm, in dem Sie x einlesen und Quicksort aufrufen. Die Länge n soll eine Konstante im Hauptprogramm sein, Ihre Implementierung soll aber für beliebige Längen funktionieren. Speichern Sie den Source-Code unter `quicksort.c` in das Verzeichnis `serie06`.

Aufgabe 6.6. Gegeben seien die Summen

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

Schreiben Sei ein Programm, welches für verschiedene Werte von N bzw. M die Zeit misst um a_N bzw. b_M zu berechnen. Geben Sie anschließend die Ergebnisse in Form einer Tabelle am Bildschirm aus. Entsprechen die Resultate Ihren Erwartungen? Speichern Sie den Source-Code unter `zeitmessung.c` in das Verzeichnis `serie06`. *Hinweis:* Überlegen Sie sich wie groß der Aufwand bei der Berechnung von a_N bzw. b_M ist.

Aufgabe 6.7. Sie legen ihr Kapital bei ihrer Hausbank zu einem fixen Jahreszinssatz an. Schreiben Sie eine Funktion `endkapital` welches aus den Werten Laufzeit $n \in \mathbb{N}$, Jahreszinssatz p (in Prozent %), und Startkapital $x \in \mathbb{R}_{\geq 0}$ das Endkapital nach n Jahren zurückgibt. Dabei soll die Funktion ihren Kontostand in folgender Form

```
Jahr    Kapital
====    =======
0      1000.00
1      1010.00
2      1020.10
3      1030.30
..      .......
10      1104.62
```

ausgeben. (Bei diesem Beispiel wurde $p = 1$, $n = 10$, und $x = 1000.00$ gewählt.) Schreiben Sie ferner eine Funktion `laufzeit`, welche berechnet wie lange Sie mindestens Ihr Startkapital x bei einem Zinssatz p anlegen müssen um ein Endkapital von mindestens $x_{\max}$ zu haben. Die Funktion soll x, p, und $x_{\max}$ als Eingabe erhalten. Weiters schreiben Sie ein Hauptprogramm welches die beiden Funktionen testet. Wie lange müssen Sie warten um Euromillionär zu werden, wenn Sie $x = 1000$ Euro bei einem Fixzinssatz von $p = 4$ anlegen? Speichern Sie den Source-Code unter `kapital.c` in das Verzeichnis `serie06`.

Aufgabe 6.8. Welchen Aufwand hat Bubblesort aus Aufgabe 6.4 im besten und schlimmsten Fall? Begründen Sie Ihre Antwort!