Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Sommersemester 2009
25. März 2009

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 3

Die Aufgaben mit Stern (*) sind bis zur Übung in der kommenden Woche vorzubereiten und werden dort abgeprüft. Kopieren Sie bitte die Source-Codes auf Ihren Account auf der `lva.student.tuwien.ac.at` *in ein Unterverzeichnis* `serie03`*. Überprüfen Sie vor der Übung, ob Ihre Source-Codes mit dem* `gcc` *compiliert werden können. In den folgenden Übungsaufgaben sollen* ***Zählschleifen*** *sowie* ***Pointer*** *und* ***Vektoren*** *geübt werden.*

Aufgabe 21*. Gegeben sei ein Polynom $p(x) = \sum_{j=0}^{n} a_j x^j$ in Form seines Koeffizientenvektors $a = (a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$. Schreiben Sie eine Funktion `evalpolynomial`, die für gegebenen Koeffizientenvektor a und Auswertungspunkt x den Funktionswert $p(x)$ berechnet. Die Funktion `pow` zur Berechnung von x^j soll *nicht* verwendet werden. Schreiben Sie eine Funktion, die möglichst nur *eine* Schleife verwendet. Der Grad $n \in \mathbb{N}$ des Polynoms soll eine Konstante im Hauptprogramm sein, die Funktion `evalpolynomial` soll aber beliebigen Grad zulassen. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Koeffizienten a_j sowie der Auswertungspunkt x eingelesen werden und $p(x)$ ausgegeben wird. Speichern Sie den Source-Code unter `evalpolynomial.c` in das Verzeichnis `serie03`.

Aufgabe 22*. *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? Man schreibe eine Funktion `bubblesort`, die ein gegebenes Array $x \in \mathbb{R}^n$ mittels Bubble-Sort aufsteigend sortiert, d.h. $x_1 \leq x_2 \leq \cdots \leq x_n$, und zurückgibt. Ferner schreibe man 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. Was ist der Vorteil von Bubble-Sort gegenüber Selection-Sort aus der Vorlesung? Speichern Sie den Source-Code unter `bubblesort.c` in das Verzeichnis `serie03`.

Aufgabe 23*. Was sind die Bestandteil einer Gleitpunktzahl des Zahlensystems $\mathbb{F}(b, p, e_{\min}, e_{\max})$? Wie groß ist die kleinste positive normalisierte Gleitpunktzahl in $\mathbb{F}(16, 6, -64, 63)$? Was ist ein implizites erstes Bit? Kann man bei dezimalen Gleitpunktzahlen (mit Basis 10) ein implizites erstes Bit verwenden?

Aufgabe 24*. Man schreibe 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. $|x_j| = \max\{|x_i| : i = 1, \ldots, n\}$ und für $|x_i| = |x_j|$ gilt $i \geq j$. Ferner schreibe man ein aufrufendes Hauptprogramm, das die Dimension $n \in \mathbb{N}$ und den Vektor x einliest und das Ergebnis von `maxabs` ausgibt. Die Länge des Vektors soll insbesondere nicht als Konstante im Hauptprogramm definiert werden. Speichern Sie den Source-Code unter `maxabs.c` in das Verzeichnis `serie03`.

Aufgabe 25. Die Gleitpunkt-Arithmetik ist nicht assoziativ. Das numerische Ergebnis der Summe $\sum_{j=0}^{n} x^j/j!$ hängt daher von der Summationsreihenfolge ab. Man schreibe Funktionen `forward` und `backward`, die diese Summe auf zwei Weisen berechnen: `forward` entspricht Vorwärtssummation $j = 0, \ldots, n$, `backward` entspricht Rückwärtssummation $j = n, \ldots, 0$. Welche Variante wird besser sein – zumindest für positives x und großes n? Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Schranke $n \in \mathbb{N}$ eingelesen und die Ergebnisse von `forward` bzw. `backward` ausgegeben werden.

Aufgabe 26. Man schreibe eine Funktion `prim`, die überprüft, ob eine natürliche Zahl $n \in \mathbb{N}$ eine Primzahl ist (Rückgabewert 1) oder nicht (Rückgabewert 0). Ferner schreibe man ein aufrufendes Hauptprogramm, das den Wert n einliest und ausgibt, ob es sich um eine Primzahl handelt.

Aufgabe 27. Schreiben Sie eine Funktion, die von einem gegebenem Vektor $x \in \mathbb{R}^n$ das Minimum $\min_{j=1}^{n} x_j$, das Maximum $\max_{j=1}^{n} x_j$ und den Mittelwert $\frac{1}{n}\sum_{j=1}^{n} x_j$ berechnet und zurückgibt. Schreiben Sie zwei Versionen der Funktion mit folgenden Deklarationen:

```
double* minmaxmean1(double* x,int n);
void minmaxmean2(double* x,int n,double* min,double* max,double* mean)
```

Ferner schreibe man ein aufrufendes Hauptprogramm, das n und x einliest und die von `minmaxmean1` und `minmaxmean2` berechneten Kenngrößen ausgibt.

Aufgabe 28. 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? Man schreibe eine Funktion `sum`, die dieses Vorgehen realisiert. Ferner schreibe man ein Hauptprogramm, das n einliest und $\sum_{j=1}^{n}(-1)^j/j$ ausgibt.

Aufgabe 29. 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 $x < y$. Man schreibe eine Prozedur `pythagoras`, die zu gegebener Schranke $n \in \mathbb{N}$ alle pythagoräischen Zahlentripel mit $x < y < z \leq n$ bestimmt und ausgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, in dem die Schranke n eingelesen und `pythagoras` aufgerufen wird.

Aufgabe 30. 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 n, x und p eingelesen werden und $\|x\|_p$ ausgegeben wird. Testen Sie Ihr Programm mit verschiedenen Werten für p bei festem Vektor x. Was beobachten Sie für $p \to \infty$?