Dirk Praetorius,
Marcus Page,
Thomas Führer

Sommersemester 2013
09.04.2013

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 4

Aufgabe 4.1. 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 4.2. 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 $|x_i| = |x_j|$ gilt $i \geq j$. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor x einliest und das Ergebnis von `maxabs` ausgibt. Die Länge des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `maxabs` ist für beliebige Länge zu implementieren. Speichern Sie den Source-Code unter `maxabs.c` in das Verzeichnis `serie04`.

Aufgabe 4.3. Schreiben Sie eine Funktion `mean`, die von einem gegebenem Vektor $x \in \mathbb{Z}^n$ den Mittelwert $\frac{1}{n}\sum_{j=1}^{n} x_j$ berechnet und zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor $x \in \mathbb{Z}^n$ einliest und den Mittelwert ausgibt. Die Länge $n \in \mathbb{N}$ des Vektors soll eine Konstante im Hauptprogramm sein die Funktion `mean` ist für beliebige Länge n zu programmieren. Speichern Sie den Source-Code unter `mean.c` in das Verzeichnis `serie04`.

Aufgabe 4.4. 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 4.5. Schreiben Sie eine Funktion `prim`, die überprüft, ob eine natürliche Zahl $n \in \mathbb{N}$ eine Primzahl ist (Rückgabewert 1) oder nicht (Rückgabewert 0). Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Wert n einliest und ausgibt, ob es sich um eine Primzahl handelt. Speichern Sie den Source-Code unter `prim.c` in das Verzeichnis `serie04`.

Aufgabe 4.6. 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: $x^{(<)}$ enthält alle Elemente $< x_1$, $x^{(\geq)}$ enthält alle Elemente $\geq x_1$. Diese Teillisten werden rekursiv sortiert. Anschließend wird das Ergebnis zusammengesetzt. Es besteht (in der Reihenfolge) aus der unteren Liste $x^{(<)}$, dem Pivotelement und der oberen Liste $x^{(\geq)}$. — Eine direkte Implementation dieses Algorithmus hat allerdings den Nachteil, dass zusätzlicher Speicher benötigt wird. Um dies zu vermeiden, geht man wie folgt 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 bereits in der Form $(x^{(<)}, x_1, x^{(\geq)})$ vor. Es müssen nun nur noch die Teillisten sortiert werden. Speichern Sie den Source-Code unter `quicksort.c` in das Verzeichnis `serie04`.

Aufgabe 4.7. Schreiben Sie eine Funktion `power`, die für gegebene reelle Zahlen $x > 1$ und $C > 0$ die kleinste Zahl $n \in \mathbb{N}$ berechnet mit $x^n > C$. Dabei soll die Funktion `log` nicht verwendet werden. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem x und C eingelesen werden und n ausgegeben wird.

Aufgabe 4.8. Schreiben Sie eine Funktion `exponential`, die den Funktionswert $\exp(x)$ approximativ berechnet: Dazu berechnen Sie die Partialsumme

$$S_N(x) := \sum_{j=0}^{N} \frac{x^j}{j!},$$

wobei die Summationsgrenze $N \in \mathbb{N}$ durch das Kriterium

$$\Big|\frac{x^{N+1}}{(N+1)!}\Big| \leq \Big|\frac{x^N}{N!}\Big| \leq \varepsilon$$

für eine gegebene Toleranz $\varepsilon > 0$ bestimmt werde. Intern realisiere man die Berechnung der Summationsglieder $x^j/j!$ möglichst rechenökonomisch. Vergleichen Sie den Fehler $|S_N(x) - \exp(x)|$ für verschiedene Wahlen von $\varepsilon > 0$ und Auswertungspunkten $x \in \mathbb{R}$. Speichern Sie den Source-Code unter `exponential.c` in das Verzeichnis `serie04`.