Dirk Praetorius,
Thomas Führer,
Michael Feischl

Wintersemester 2013/14
28.10.2013

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 5

Aufgabe 5.1. Schreiben Sie eine 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 $x \in \mathbb{R}$ und für $n < 0$ gilt $x^n = (1/x)^{-n}$. Weiters gilt $0^n = 0$ für $n > 0$. Die Potenz 0^n ist für $n < 0$ nicht definiert. Die Funktion soll in diesem Fall den Wert `0.0/0.0` zurückgeben. Speichern Sie den Source-Code unter `powN.c` in das Verzeichnis `serie05`.

Aufgabe 5.2. Schreiben Sie eine Funktion `sum(n)` die für gegebenes $n \in \mathbb{N}$ die Summe $\sum_{j=1}^{n}(j/2)$ berechnet und zurückgibt. Realisieren Sie diese Summe dabei direkt und nicht in der Form $\frac{1}{2}\sum_{j=1}^{n} j$. Was ist zu beachten? Schreiben Sie ferner ein Hauptprogramm, das den Wert n von der Tastatur einliest und das Ergebnis `sum(n)` am Bildschirm ausgibt. Speichern Sie den Source-Code unter `sum.c` in das Verzeichnis `serie05`.

Aufgabe 5.3. 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. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor $x \in \mathbb{R}^n$ einliest und den geometrischen Mittelwert ausgibt. Die Länge $n \in \mathbb{N}$ des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `geometricMean` ist für beliebige Länge n zu programmieren. Speichern Sie den Source-Code unter `geometricMean.c` in das Verzeichnis `serie05`.

Aufgabe 5.4. 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` errechnet 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 `serie05`.

Aufgabe 5.5. In vielen mathematischen Bibliotheken werden Matrizen $A \in \mathbb{R}^{m \times n}$ spaltenweise gespeichert, d.h. in Form eines Vektors $a \in \mathbb{R}^{mn}$, wobei $a_{j+km} = A_{jk}$ gilt, wenn die Indizierung (wie in C üblich) bei 0 beginnt. Schreiben Sie eine Funktion `mvmultiplication`, die die Matrix-Vektor-Multiplikation einer spaltenweise gespeicherten Matrix $A \in \mathbb{R}^{m \times n}$ mit einem Vektor $x \in \mathbb{R}^n$ realisiert und den Ergebnisvektor $b = Ax \in \mathbb{R}^m$ zurückgibt. Die Dimensionen $m, n \in \mathbb{N}$ können Konstanten im Hauptprogramm sein, müssen aber als Parameter an die Funktion übergeben werden. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem A und x eingelesen werden und $b = Ax$ ausgegeben wird. Speichern Sie den Source-Code unter `mvmultiplication.c` in das Verzeichnis `serie05`.

Aufgabe 5.6. *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. $x_1 \leq x_2 \leq \cdots \leq x_n$, 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 `serie05`.

Aufgabe 5.7. Schreiben Sie ein Programm, welche mehrere *Lotto 6 aus 45* Tipps einliest und überprüft ob der richtige Tipp dabei war. Das Programm soll k Tipps einlesen und in einer $k \times 6$ Matrix speichern. Die Anzahl der Tipps soll dabei eine feste Konstante im Hauptprogramm sein. Danach soll noch ein Vektor a mit dem "richtigen" Tipp eingelesen werden. Überprüfen Sie danach ob Sie einen Lotto 6er haben! *Optional* können Sie Ihr Programm noch erweitern und überprüfen ob einer Ihrer Tipps $3, 4$ oder 5 richtige Zahlen enthält. Speichern Sie den Source-Code unter `tipps.c` in das Verzeichnis `serie05`.
Hinweis: Teilen Sie ihr Programm in mehrere Funktionen auf. Beispielsweise soll eine Funktion die Tipps einlesen, eine andere Funktion soll überprüfen ob ein richtiger Tipp dabei war. Weiters ist es vermutlich sinnvoll die eingelesenen Tipps aufsteigend zu sortieren. Beachten Sie ausserdem, dass in einem Tipp eine Zahl maximal einmal vorkommen darf. Das Programm soll das natürlich berücksichtigen und überprüfen (gleich beim Einlesen)!

Aufgabe 5.8. Erklären Sie den Begriff *Aufwand* mit eigenen Worten. Was versteht man unter *quadratischen* Aufwand? Welchen Aufwand hat die Berechnung der Summe aus Aufgabe 5.2? Welchen Aufwand hat der *Bubble-Sort*-Algorithmus aus Aufgabe 5.6?