Dirk Praetorius,
Gregor Gantner

Wintersemester 2018/19
08./09.11.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 3

Aufgabe 3.1. Eine (möglicherweise nicht die beste) Art die Zahl π anzunähern liefert die als *Leibniz-Reihe* bekannte Formel

$$\pi = \sum_{k=0}^{\infty} \frac{4(-1)^k}{2k+1}.$$

Die n-te Partialsumme

$$P(n) = \frac{4(-1)^n}{2n+1} + P(n-1)$$

können wir also als rekursive Funktion auffassen, für die $\lim_{n\to\infty} P(n) = \pi$ gilt. Implementieren Sie eine Funktion `double P(int n)` die obige Funktionalität realisiert. Schreiben Sie auch ein Hauptprogramm, das n über die Tastatur einliest und das obige Partialsumme berechnet und ausgibt. Speichern Sie den Source-Code unter `piRekursiv.c` in das Verzeichnis `serie03`.

Aufgabe 3.2. Schreiben Sie eine rekursive 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 $x \in \mathbb{R} \setminus \{0\}$ und für $n < 0$ gilt $x^n = (1/x)^{-n}$. Weiters gilt $0^n = 0$ für $n > 0$. Die Potenz 0^n ist für $n \leq 0$ nicht definiert. Die Funktion soll in diesem Fall den Wert `0.0/0.0` zurückgeben. Für diese Aufgabe dürfen Sie die Funktion `pow` aus der Mathematikbibliothek nicht verwenden. Speichern Sie den Source-Code unter `powN.c` in das Verzeichnis `serie03`.

Aufgabe 3.3. Schreiben Sie zwei Funktionen:

- die Funktion `double skalarProdukt(double u[3], double v[3])`, die zu gegebenen Vektoren $\mathbf{u} = (a, b, c)^T$ und $\mathbf{v} = (x, y, z)^T$ das Skalarprodukt $w = \mathbf{u} \cdot \mathbf{v} = ax + by + cz$ berechnet und zurückgibt;
- die Funktion`void vektorProdukt(double u[3], double v[3], double w[3])`, die zu gegebenen Vektoren $\mathbf{u} = (a, b, c)^T$ und $\mathbf{v} = (x, y, z)^T$ das Vektorprodukt $\mathbf{w} = \mathbf{u} \times \mathbf{v}$ mit

$$\begin{aligned} w_1 &= bz - cy, \\ w_2 &= cx - az, \\ w_3 &= ay - bx \end{aligned}$$

  berechnet.

Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die sechs Parameter a, b, c und x, y, z über die Tastatur eingelesen und die zwei Ergebnisse ausgegeben werden. Speichern Sie den Source-Code unter `produkte.c` in das Verzeichnis `serie03`.

Aufgabe 3.4. Schreiben Sie ein Programm, das einen statischen Vektor x der Länge 1000 anlegt. Für die Koeffizienten $x[i]$ soll gelten, dass $x[i] = i$ für alle $i \in \{0, 1, \ldots, 999\}$. Anschließend soll der Vektor am Bildschirm ausgegeben werden. Sie dürfen keine Schleifen verwenden. Speichern Sie den Source-Code unter `array.c` in das Verzeichnis `serie03`.
Hinweis: Schreiben Sie Funktionen `createVector` und `printVector`, die im Hauptprogramm aufgerufen werden.

Aufgabe 3.5. Die Fibonacci-Folge ist definiert durch $x_0 := 0$, $x_1 := 1$ und $x_{n+1} := x_n + x_{n-1}$. Schreiben Sie eine rekursive Funktion `fibonacciVec`, die zu gegebenem Index n das Folgenglied x_n berechnet. Die Funktion `fibonacciVec` soll im Gegensatz zur Aufgabe 2.7 alle Zwischenergebnisse $x_0, \ldots, x_{n-1}$ im Vektor x speichern. Beachten Sie, dass diese nur einmal berechnet werden. Die Zahl n bezeichnet eine Konstante in Hauptprogramm. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem der Vektor x geeignet deklariert und das Ergebnis ausgegeben wird. Was sind Vor- und Nachteile gegenüber der Funktion `fibonacci` aus Aufgabe 2.7. Speichern Sie den Source-Code unter `fibonacciVec.c` in das Verzeichnis `serie03`.

Aufgabe 3.6. Gegeben sei eine stetige Funktion $f : [a, b] \to \mathbb{R}$ auf einem Intervall $[a, b]$ mit

$$f(a) \cdot f(b) \leq 0.$$

Dann hat f eine Nullstelle x_0, die im Folgenden mittels Bisektion approximiert werden soll: Man definiert $c := (a + b)/2$ als Intervallmittelpunkt. Laut Voraussetzung gilt

$$f(a) \cdot f(c) \leq 0 \quad \text{oder} \quad f(c) \cdot f(b) \leq 0.$$

Im Fall $f(a) \cdot f(c) \leq 0$ ruft man den Bisektionsalgorithmus für das Intervall $[a, c]$ auf, anderenfalls für das Intervall $[c, b]$. Als Abbruchbedingung verwende man $|b - a| \leq \varepsilon$. Wegen $x_0 \in [a, b]$ ist dann beispielsweise c (oder auch a und b) eine Approximation der Nullstelle bis auf einen Fehler ε. Der Funktion `int bisektion( double ab[2], double eps)` werden also die Parameter $a, b \in \mathbb{R}$ und $\varepsilon > 0$ übergeben. Im Fall $f(a) \cdot f(b) > 0$ soll abgebrochen werden. Man realisiere die Übergabe von a und b mittels eines Vektors $[a, b]$ (Call by Reference), sodass der Return-Value angibt, ob beim Bisektionsverfahren ein Fehler (Rückgabewert `-1`) aufgetreten ist oder nicht (Rückgabewert `0`). Bei jedem Funktionsaufruf sollen a, b, $|b - a|$ und $f(c)$ ausgegeben werden. Als Testfunktion verwende man $f(x) = x^2 + \exp(x) - 2$ auf $[0, \infty)$, die man als eigene Funktion realisiere. Schreiben Sie ferner ein Hauptprogramm, das $a, b, \varepsilon > 0$ einliest und eine entsprechende Approximation der Nullstelle ausgibt. Speichern Sie den Source-Code unter `bisektion.c` in das Verzeichnis `serie03`.

Aufgabe 3.7. Gegeben sei ein sortierter Vektor x der Länge n. Schreiben Sie eine rekursive Funktion `findBisection(x,y)`, die einen Index i zurückgibt, für den $x_i = y$ gilt. Falls y nicht in x vorkommt soll 0 zurückgegeben werden. Um einen schnellen Code zu erhalten, durchsuchen Sie nicht einfach den Vektor x on vorne nach hinten (oder umgekehrt). Sondern nutzen Sie die Idee des Bisektionsalgorithmus aus Aufgabe 3.6 geeignet. Wieviele Funktionsaufrufe brauchen Sie maximal falls x einen Vektor der Länge 32 ist? Speichern Sie den Source-Code unter `findBisection.m` in das Verzeichnis `serie03`.

Aufgabe 3.8. Schreiben Sie die Funktion `int geraden(double u[3], double v[3], double s[2])`, die zwei Geraden auf ihre Lage in der Fläche untersucht: Mit vorgegebenen $\mathbf{u} = (a, b, c)^T \in \mathbb{R}^3$ und $\mathbf{v} = (d, e, f)^T \in \mathbb{R}^3$ werden durch die Gleichungen

$$\begin{aligned} ax + by &= c, \\ dx + ey &= f \end{aligned}$$

zwei Geraden in der Ebene festgelegt. Die Funktion `geraden` bestimmt, ob die gegebenen Geraden *parallel* (Rückgabe `1`), *ident* (Rückgabe `0`) oder *schneidend* (Rückgabe `-1`) sind. In letzterem Fall sollen auch die Koordinaten des Schnittpunktes berechnet und zurückgegeben werden (in `s[2]`). Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Parameter über die Tastatur eingelesen werden und `geraden` aufgerufen wird. Speichern Sie den Source-Code unter `geraden.c` in das Verzeichnis `serie03`.