Dirk Praetorius
Marcus Page

Sommersemester 2011
21. März 2011

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 3

Aufgabe 3.1*. Erweitern Sie Ihre Funktion `evalpol2` aus Aufgabe 2.5 dahingehend, dass nun Polynome beliebigen Grades ausgewertet werden können. Hierzu soll der Funktion zusätzlich zum Auswertungspunkt $x \in \mathbb{R}$, der Grad $n \in \mathbb{N}$ des Polynoms übergeben werden. Die entsprechenden Koeffizienten übergeben Sie als Vektor dessen Länge eine Konstante im Hauptprogramm sein soll. Zur Berechnung der Potenzen von x können Sie die `pow`-Routine der mathematischen Bibliothek verwenden. Speichern Sie den Source-Code unter `evalpolNew.c` in das Verzeichnis `serie03`.

Aufgabe 3.2*. Schreiben Sie eine Funktion `diffpol`, welche die Koeffizienten der Ableitung eines Polynoms berechnet. Welche Eingabeparameter benötigen Sie hierfür? Der Koeffizientenvektor soll schließlich ausgegeben werden. Speichern Sie den Source-Code unter `diffpol.c` in das Verzeichnis `serie03`.

Aufgabe 3.3*. Für $n \in \mathbb{N}$ ist die Fakultätsfunktion rekursiv definiert durch

$$n! := \begin{cases} 1 & \text{falls } n = 0, \\ (n-1)! \cdot n & \text{falls } n > 0. \end{cases}$$

Schreiben Sie eine (nicht rekursive) Funktion `factorial`, welche den größten Wert der Fakultätsfunktion unter einer gegebenen Schranke $m \in \mathbb{N}$ berechnet. Die Schranke m soll in einem aufrufenden Hauptprogramm von der Tastatur eingelesen und entsprechend übergeben werden. Das Programm soll dann den Wert und den entsprechenden Index n ausgeben. Speichern Sie den Source-Code unter `factorial.c` in das Verzeichnis `serie03`.

Aufgabe 3.4*. Schreiben Sie eine Funktion `scanfpositive`, die vom Benutzer die Eingabe einer positiven Zahl $\tau > 0$ verlangt und diese dann zurückgibt. Die Eingabe soll solange wiederholt werden, bis die eingegebene Zahl $\tau \in \mathbb{R}$ strikt positiv ist, d.h. bei Eingabe einer Zahl $\tau \leq 0$ wird der Benutzer zu erneuter Eingabe aufgefordert. Speichern Sie den Source-Code unter `scanfpositive.c` in das Verzeichnis `serie03`.

Aufgabe 3.5. In dieser Aufgabe sollen Sie erneut Polynome und deren Ableitungen auswerten. Gegeben sei der feste Grad $n \in \mathbb{N}$ eines Polynoms $f \in \mathcal{C}^\infty(\mathbb{R})$. Im Hauptprogramm sollen nun die Koeffizienten $x_0, \dots, x_n \in \mathbb{N}$ von der Tastatur eingelesen werden. Zusätzlich wird über Tastatureingabe entschieden werden, ob das Polynom selbst, oder dessen Ableitung an einer Stelle $x \in \mathbb{R}$ (auch über die Tastatur einzulesen) ausgewertet werden soll. Überlegen Sie sich, wie Sie diese Unterscheidung sinnvoll implementieren können. Zum Schluss soll der entsprechende Wert ausgegeben werden. Orientieren Sie sich hierbei an den Aufgaben 3.1 bzw. 3.2. Speichern Sie den Source-Code unter `polyfun.c` in das Verzeichnis `serie03`.

Aufgabe 3.6. Schreiben Sie eine `void`-Funktion `vielfache(k,nmax)`, die alle ganzzahligen Vielfachen der Zahl $k \in \mathbb{N}$, die $\leq n_{\max} \in \mathbb{N}$ sind, am Bildschirm ausgibt. Die Ausgabe erfolge zeilenweise in der Form

```
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
```

beispielsweise für den Fall $k = 5$ und $n_{\max} = 19$. Ferner schreibe man ein Hauptprogramm, das die Daten k und $n_{\max}$ von der Tastatur einliest und `vielfache(k,nmax)` aufruft. Speichern Sie den Source-Code unter `vielfache.c` in das Verzeichnis `serie03`.

Aufgabe 3.7. 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 $x < y$. Schreiben Sie eine `void`-Funktion `pythagoras`, die zu gegebener Schranke $n \in \mathbb{N}$ alle pythagoräischen Zahlentripel mit $x < y < z \leq n$ bestimmt und ausgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Schranke n eingelesen und `pythagoras` aufgerufen wird. Speichern Sie den Source-Code unter `pythagoras.c` in das Verzeichnis `serie03`.

Aufgabe 3.8. Es sei eine Toleranz $\tau > 0$ von der Tastatur einzulesen. Schreiben Sie nun eine Funktion, welche die Summe

$$\sum_{k=0}^{N} \frac{x^k}{k!}$$

für ein festes $x \in \mathbb{R}$ berechnet. Die obere Grenze $N \in \mathbb{N}$ soll hierbei die kleinste Ganze Zahl sein, für die

$$\left| \frac{x^N}{N!} \right| \leq \tau$$

gilt. Vermeiden Sie die Verwendung von Funktionen, die x^k oder $k!$ berechnen. Nutzen Sie stattdessen die Gleichheit

$$\frac{x^k}{k!} = \frac{x^{k-1}}{(k-1)!} \cdot \frac{x}{k}.$$

Warum ist dieser Hinweis wichtig?

Aufgabe 3.9. Schreiben Sie eine Funktion `wurzelschranke`, die zu einer gegebenen Zahl $x \geq 0$ die natürliche Zahl $k \in \mathbb{N}_0$ mit $k \leq \sqrt{x} < k + 1$ zurückgibt. Dabei dürfen weder die Wurzel-Funktion `sqrt`, noch Rundungsoperationen (z.B. `floor` oder `ceil` etc.) verwendet werden. Schreiben Sie ferner ein aufrufendes Hauptprogramm das $x \in \mathbb{R}$ einliest und $k \in \mathbb{N}$ ausgibt. Speichern Sie den Source-Code unter `wurzelschranke.c` in das Verzeichnis `serie03`.

Aufgabe 3.10. Schreiben Sie eine Funktion `findme` die folgendes realisiert: Zu Beginn wird (z.B. durch eine andere Person) im Hauptprogramm eine beliebige Zahl $n \in \mathbb{N}$ von der Tastatur eingelesen und an `findme` übergeben. Ziel ist es nun diese Zahl zu erraten. Hierzu soll die Funktion den Benutzer so lange zu neuen Eingaben auffordern, bis die entsprechende Zahl getroffen ist. Damit das nicht allzu lange dauert soll die Funktion nach jeder Eingabe einen Tip in der Form

`Die gesuchte Zahl ist größer als die letzte Eingabe`,

bzw.

`Die gesuchte Zahl ist kleiner als die letzte Eingabe`

abgeben. Lassen Sie nun Ihren Tutor raten! Speichern Sie den Source-Code unter `findme.c` in das Verzeichnis `serie03`.