Dirk Praetorius,
Gregor Gantner

Sommersemester 2018
17.04.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 5

Aufgabe 5.1. 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. Schreibens Sie weiters ein aufrufendes Hauptprogramm. Wie haben Sie Ihren Code auf Korrektheit getestet? Speichern Sie den Source-Code unter `scanfpositive.c` in das Verzeichnis `serie05`.

Aufgabe 5.2. Schreiben Sie eine *nicht-rekursive* 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. Stellen weiters Sie mittels `assert` sicher, dass $x > 1$ und $C > 0$ gilt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem x und C eingelesen werden und n ausgegeben wird. Speichern Sie den Source-Code unter `power.c` in das Verzeichnis `serie05`.

Aufgabe 5.3. Schreiben Sie eine Funktion `kgV`, die zu zwei gegebenen natürlichen Zahlen $a, b \in \mathbb{N}$ das kleinste gemeinsame Vielfache berechnet und zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das die Zahlen $a, b \in \mathbb{N}$ einliest und das dazugehörige kleinste gemeinsame Vielfache ausgibt. Wie haben Sie Ihren Code auf Korrektheit getestet? Speichern Sie den Source-Code unter `kgV.c` in das Verzeichnis `serie05`.

Aufgabe 5.4. Die Quotientenfolge $(a_{n+1}/a_n)_{n\in\mathbb{N}}$ zur Fibonacci-Folge $(a_n)_{n\in\mathbb{N}}$,

$$a_0 := 1, \quad a_1 := 1, \quad a_n := a_{n-1} + a_{n-2} \quad \text{für } n \geq 2,$$

konvergiert gegen den goldenen Schnitt $(1 + \sqrt{5})/2$. Insbesondere konvergiert die Differenz

$$b_n := \frac{a_{n+1}}{a_n} - \frac{a_n}{a_{n-1}}$$

gegen Null. Schreiben Sie eine *nicht-rekursive* Funktion `cauchy`, die zu gegebenem $k \in \mathbb{N}$ die kleinste Zahl $n \in \mathbb{N}$ mit $|b_n| \leq 1/k$ zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das die Zahl $k \in \mathbb{N}$ einliest und den zugehörigen Index $n \in \mathbb{N}$ ausgibt. Wie haben Sie Ihren Code auf Korrektheit getestet? Speichern Sie den Source-Code unter `goldenerSchnitt.c` in das Verzeichnis `serie05`.

Aufgabe 5.5. Für $x > 0$ konvergiert die Folge

$$x_1 := \frac{1}{2}(1 + x), \quad x_{n+1} := \frac{1}{2}\Big(x_n + \frac{x}{x_n}\Big) \quad \text{für } n \geq 1$$

gegen $\sqrt{x}$. Schreiben Sie eine *nicht-rekursive* Funktion `sqrt_new`, die für gegebene $x > 0$ und $\tau > 0$ als Ergebnis das erste Folgenglied $y = x_n$ zurückgibt, für das gilt

$$\frac{|x_n - x_{n+1}|}{|x_n|} \leq \tau \quad \text{oder} \quad |x_n| \leq \tau.$$

Überprüfen Sie mittels `assert`, dass $x > 0$. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem x eingelesen und neben der Approximation x_n von $\sqrt{x}$ auch der exakte Wert sowie der absolute Fehler $|x_n - \sqrt{x}|$ ausgegeben werden. Wie haben Sie Ihren Code auf Korrektheit getestet? Speichern Sie den Source-Code unter `sqrt_new.c` in das Verzeichnis `serie05`.

Aufgabe 5.6. Die Cosinus-Funktion hat die Darstellung $\cos(x) = \sum_{k=0}^{\infty}(-1)^k \frac{x^{2k}}{(2k)!}$. Wir betrachten die Partialsummen

$$C_n(x) = \sum_{k=0}^{n}(-1)^k \frac{x^{2k}}{(2k)!}.$$

Schreiben Sie eine *nicht-rekursive* Funktion `cos_new`, die für gegebene $x \in \mathbb{R}$ und $\tau > 0$ den Wert $C_n(x)$ zurückliefert, sobald

$$|C_n(x) - C_{n-1}(x)|/|C_n(x)| \le \tau \quad \text{oder} \quad |C_n(x)| \le \tau$$

gilt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem $x \in \mathbb{R}$ und $\tau > 0$ eingelesen werden. Neben dem berechneten Wert $C_n(x)$ sollen auch der korrekte Wert $\cos(x)$ und der absolute Fehler $|C_n(x) - \cos(x)|$ ausgegeben werden sowie der relative Fehler $|C_n(x) - \cos(x)|/|\cos(x)|$ im Fall $\cos(x) \neq 0$. Schreiben Sie die Funktion möglichst so, dass diese mit einer Schleife auskommt und dass x^{2k} und $(2k)!$ möglichst kostensparend realisiert werden. Man vermeide also insbesondere (vor- oder selbst implementierte) Funktionen zur Berechnung der Potenz oder der Faktoriellen. Wie haben Sie Ihren Code auf Korrektheit getestet? Speichern Sie den Source-Code unter `cos.c` in das Verzeichnis `serie05`.

Aufgabe 5.7. Gegeben seien die Summen

$$a_N := \sum_{n=0}^{N} \frac{1}{(n+1)^2} \quad \text{und} \quad b_N := \sum_{n=0}^{N}\sum_{k=0}^{n} \frac{1}{(k+1)^2(n-k+1)^2}.$$

Schreiben Sie zwei Funktionen, welche für gegebene $N \in \mathbb{N}$ die Zeit messen, um $(a_N)^2$ bzw. b_N zu berechnen. Wie groß ist der Aufwand bei der Berechnung von $(a_N)^2$ bzw. b_N? Z.B.: Falls die Funktionen für $N = 10^3$ eine Laufzeit von 3 Sekunden haben, welche Laufzeit erwarten Sie aufgrund des Aufwands für $N = 10^4$? Schreiben Sie ferner ein Hauptprogramm, welches für verschiedene Werte von N die Ergebnisse in Form einer Tabelle am Bildschirm ausgibt. Entsprechen die Resultate Ihren Erwartungen? Wie haben Sie Ihr Programm getestet? Speichern Sie den Source-Code unter `zeitmessung.c` in das Verzeichnis `serie05`.

Aufgabe 5.8. Welche Arten von Kommentaren gibt es? Was gibt folgender Code aus und warum?

```
#include <stdio.h>

/*int f(double x) {
    return (int) x;
  }
*/

main() {
  int x = 4;
  int y = 2*x*/* f(0.1)+3
            */1/4;
  // y = 1;
  printf("y = %d\n",y); // Ausgabe
}
```