Dirk Praetorius,
Michele Ruggeri

Wintersemester 2015/16
28.10.-30.10.2015

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 3

Aufgabe 3.1. Schreiben Sie eine `void`-Funktion `vektorprodukt`, 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 und ausgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Vektoren $\mathbf{u}, \mathbf{v}$ eingelesen und die Funktion aufgerufen werden. Speichern Sie den Source-Code unter `vektorprodukt.c` in das Verzeichnis `serie03`.

Aufgabe 3.2. Schreiben Sie eine `void`-Funktion `dreieck`, die für gegebene Seitenlängen $a, b, c \in \mathbb{R}$ mit $a, b, c \geq 0$ feststellt, ob es sich bei dem zugehörigen Dreieck um ein allgemeines, gleichschenkeliges, gleichseitiges, rechtwinkeliges, eindimensional „entartetes" oder um ein „unmögliches" Dreieck handelt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem a, b und c eingelesen werden und die Funktion aufgerufen wird. Speichern Sie den Source-Code unter `dreieck.c` in das Verzeichnis `serie03`.

Aufgabe 3.3. Schreiben Sie eine `void`-Funktion `sort3`, der drei Zahlen $x, y, z \in \mathbb{R}$ übergeben werden und die diese Zahlen fallend sortiert ausgibt, d.h. zuerst das Maximum $\max\{x, y, z\}$ und zuletzt das Minimum $\min\{x, y, z\}$. Schreiben Sie ferner ein aufrufendes Hauptprogramm in dem die Zahlen x, y, z eingelesen und die Funktion aufgerufen werden. Speichern Sie den Source-Code unter `sort3.c` in das Verzeichnis `serie03`.

Aufgabe 3.4. Schreiben Sie eine `void`-Funktion `roman`, die eine natürliche Zahl $x \in \mathbb{N}$ mit $x \leq 99$ im römische Zahlenformat ausgibt. Zur Erinnerung:

$$C \mathrel{\widehat{=}} 100,\ L \mathrel{\widehat{=}} 50,\ X \mathrel{\widehat{=}} 10,\ V \mathrel{\widehat{=}} 5,\ I \mathrel{\widehat{=}} 1.$$

abei soll die Subtraktionsregel bei der Darstellung angewandt werden, d.h. schreiben Sie IV statt IIII für 4. Überlegen Sie sich zunächst eine Lösung für $x \leq 9$, wobei die römischen Zahlen für $1, \ldots, 9$ gerade durch

$$I,\ II,\ III,\ IV,\ V,\ VI,\ VII,\ VIII,\ IX$$

gegeben sind. Schreiben Sie sich dann auf analoge Weise die 10er Ziffer auf. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Zahl x eingelesen und die Funktion `roman` aufgerufen werden. Speichern Sie den Source-Code unter `roman.c` in das Verzeichnis `serie03`.

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 `fibonacciRek`, die zu gegebenem Index n das Folgenglied x_n zurückgibt. Speichern Sie den Source-Code unter `fibonacci.c` in das Verzeichnis `serie03`.

Aufgabe 3.6. 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 obige Partialsumme für verschiedene Werte berechnet.
Hinweis: Für die Potenzen $(-1)^n$ können Sie vorgehen wie in Aufgabe 2.3. Speichern Sie den Source-Code unter `piRekursiv.c` in das Verzeichnis `serie03`.

Aufgabe 3.7. 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 rekursive Funktion `sqrt_`, die für gegebene $x > 0$ und $\tau > 0$ als Ergebnis das erste Folgenglied $y = x_n$ zurückgibt, für das gilt

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

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. Speichern Sie den Source-Code unter `sqrt.c` in das Verzeichnis `serie03`.
Hinweis: Zur Berechnung von $\sqrt{x}$ können Sie die Funktion `sqrt` aus der Mathematikbibliothek verwenden. Um den Absolutbetrag einer reellen Zahl zu bestimmen, darf die Funktion `fabs` aus der Mathematikbibliothek verwendet werden.

Aufgabe 3.8. Wiederholen Sie die Begriffe *Lifetime & Scope*. Was gibt folgendes Programm aus?

```
#include <stdio.h>

int max(int,int);

main() {
  int x = 1;
  int y = 2;
  int z = 3;

  printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);

  {
    int x = 100;
    y = 2;
    z = max(x,y);
    printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);

    {
      int z = y;
      y = 200;

      printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
    }
    printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
  }
  printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
}

int max(int x, int y) {
  if(x>=y) {
    return x;
  }
```

```
33    else {
34      return y;
35    }
36  }
```

Zeichnen Sie einen Zeitstrahl, wo sie die Lifetime und den Scope der Variablen `x,y,z` auftragen. Kennzeichnen Sie die einzelnen Blöcke bzw. Funktionen.