Dirk Praetorius,
Thomas Führer,
Michael Feischl

Wintersemester 2013/14
15.10.2013

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 3

Aufgabe 3.1. Schreiben Sie eine Funktion `bogenmass`, die einen im Gradmaß gegebenen Winkel $\theta \in \mathbb{R}^+$ ins Bogenmaß umrechnet. Dabei soll der Rückgabewert ψ in reduzierter Form als $\psi \in [0, 2\pi)$ zurückgegeben werden. Speichern Sie den Source-Code unter `bogenmass.c` in das Verzeichnis `serie03`.

Aufgabe 3.2. Schreiben Sie eine Funktion `rundung`, die für eine gegebene Zahl $x \in \mathbb{R}$ die Zahl $n \in \mathbb{N}$ zurückliefert, die x am nächsten liegt. Falls x genau in der Mitte zwischen zwei ganzen Zahlen liegt, werde die größere zurückgeliefert. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das die Zahl x einliest und gerundet ausgibt. Speichern Sie den Source-Code unter `rundung.c` in das Verzeichnis `serie03`.

Aufgabe 3.3. Schreiben Sie eine rekursive Funktion `binomial`, die den Binomialkoeffizienten $\binom{n}{k}$ berechnet. Verwenden Sie dazu das Additionstheorem $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$. Schreiben Sie ein aufrufendes Hauptprogramm, in dem $k, n \in \mathbb{N}_0$ mit $k \leq n$ eingelesen und $\binom{n}{k}$, berechnet und ausgeben werden. Speichern Sie den Source-Code unter `binomial.c` in das Verzeichnis `serie03`.

Aufgabe 3.4. 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. Schreiben Sie weiters eine nicht rekursive Funktion `fibonacci`, die dasselbe leistet, wobei die Berechnung aber über geeignete Schleifen realisiert wird. Welche der beiden Funktionen ist effizienter und warum? Speichern Sie den Source-Code unter `fibonacci.c` in das Verzeichnis `serie03`.

Aufgabe 3.5. 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 $(-1)^n$ benötigen sie **keine** Potenzfunktion. Überlegen Sie sich hierzu wie $(-1)^n$ von n abhängt. Speichern Sie den Source-Code unter `piRekursiv.c` in das Verzeichnis `serie03`.

Aufgabe 3.6. 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 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 3.7. Schreiben Sie ein `main()`-Programm, das 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

$$w_1 = bz - cy$$
$$w_2 = cx - az$$
$$w_3 = ay - bx$$

berechnet und ausgibt. Die Vektoren $\mathbf{u}, \mathbf{v}, \mathbf{w}$ sollen im Programm als `double`-Array der Länge 3 gespeichert werden. Speichern Sie den Source-Code unter `vektorprodukt.c` in das Verzeichnis `serie03`.

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;
  }
  else {
    return y;
  }
}
```

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