Dirk Praetorius
Michele Ruggeri

Sommersemester 2015
24.03.2015

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 3

Aufgabe 3.1. 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 3.2. Schreiben Sie eine `void`-Funktion `teiler`, die für eine gegebene Zahl $x \in \mathbb{N} := \{1, 2, 3, \dots\}$ ausgibt, ob diese durch 2, durch 3 oder durch 6 teilbar ist. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Integer x einliest und `teiler` aufruft. Speichern Sie den Source-Code unter `teiler.c` in das Verzeichnis `serie03`.

Aufgabe 3.3. 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 3.4. Schreiben Sie eine `void`-Funktion `geraden`, die zwei Geraden auf ihre Lage in der Fläche untersucht: Mit vorgegebenen Zahlen a, b, c und d, e, f werden durch die Gleichungen

$$ax + by = c,$$
$$dx + ey = f$$

zwei Geraden in der Ebene festgelegt. Die Funktion `geraden` gebe aus, ob die in Form der Parameter $a, b, c, d, e, f \in \mathbb{R}$ gegebenen Geraden *parallel*, *ident* oder *schneidend* sind. In letzterem Fall sollen auch die Koordinaten des Schnittpunktes berechnet und ausgegeben werden. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Parameter a, b, c und d, e, f über die Tastatur eingelesen und `geraden` aufgerufen werden. Speichern Sie den Source-Code unter `geraden.c` in das Verzeichnis `serie03`.

Aufgabe 3.5. Implementieren Sie folgendes Computerspiel. Der Computer merke sich eine zufällige Zahl zwischen 0 und 15. Sie haben maximal drei Versuche um die richtige Zahl zu erraten. Geben Sie beim ersten oder zweiten Versuch eine falsche Zahl an, soll der Computer Ihnen mitteilen, ob die angegebene Zahl größer oder kleiner als die gesuchte Zahl ist. Wenn man auch beim dritten Versuch daneben liegt, soll die richtige Zahl angezeigt werden. Zufallszahlen zwischen 0 und 15 können Sie folgendermaßen erzeugen: Zunächst binden Sie die Headerdateien `stdlib.h` und `time.h` in Ihr Programm ein. Danach können Sie in einer beliebigen Funktion mit

```
srand( (unsigned) time(NULL) );
zufallszahl = rand() % 16;
```

eine Zufallszahl zwischen 0 und 15 generieren. Die Variable `zufallszahl` ist dabei vom Typ `int`. Speichern Sie den Source-Code unter `spiel.c` in das Verzeichnis `serie03`.

Aufgabe 3.6. Schreiben Sie eine rekursive Funktion `binomial`, die den Binomialkoeffizienten $\binom{n}{k}$ berechnet. Verwenden Sie dazu das Additionstheorem

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1} \quad \text{für } 1 \leq k < n$$

mit $\binom{n}{0} = 1 = \binom{n}{n}$ für $n \in \mathbb{N}_0$. 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 3.7. 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 3.8. Wiederholen Sie die Begriffe *Lifetime & Scope*. Was gibt folgendes Programm aus?

```
1  #include <stdio.h>
2
3  int max(int,int);
4
5  main() {
6    int x = 1;
7    int y = 2;
8    int z = 3;
9
10   printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
11
12   {
13     int x = 100;
14     y = 2;
15     z = max(x,y);
16     printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
17
18     {
19       int z = y;
20       y = 200;
21
22       printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
23     }
24     printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
25   }
26   printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
27 }
28
29 int max(int x, int y) {
30   if(x>=y) {
31     return x;
32   }
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.