Dirk Praetorius,
Michele Ruggeri

Wintersemester 2017/18
9./10.11.2017

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 3

Aufgabe 3.1. Write a `void`-function `divisor` which checks if a given number $x \in \mathbb{N} := \{1, 2, 3, \dots\}$ is divisible by 2, 3, or 6. Additionally, write a main program that reads in the number x, then calls the function `divisor`, and prints out the result. Save your source code as `teiler.c` into the directory `serie03`.

Aufgabe 3.2. Write a `void`-function `curve_sketching` which does a curve sketching for a quadratic function $p(x) = a + bx + cx^2$ with coefficients $a, b, c \in \mathbb{R}$. If existing, compute the extremum (and which kind of extremum it is). Additionally, if existing, compute the roots of the function. Otherwise, print on the screen, that the function does not have an extremum resp. roots. Moreover, write a main program which reads in the parameters a, b, c and which calls the function. Save your source code as `curve_sketching.c` into the directory `serie03`.

Aufgabe 3.3. Write a `void`-function `money` that calculates given an amount of money $n \in \mathbb{N}$ the minimal number of bank notes (500 €, 100 €, 50 €, 20 €, 10 €, 5 €) resp. coins (2 €, 1 €) such that the sum equals the value of n. This number shall be displayed on the screen. For example, for $n = 351$, one should get the following output

```
3 x 100 EUR
1 x 50 EUR
1 x 1 EUR
```

Write a main program which reads the value $n \in \mathbb{N}$ and which calls the function `money`. Save your source code as `money.c` into the directory `serie03`.

Aufgabe 3.4. Write a program that allocates a static vector x of length 1000. The coefficients shall satisfy $x[i] = i$ for all $i \in \{0, 1, \dots, 999\}$. Next, the vector shall be displayed on the screen. You must not use `for`-loops. Save your source code as `array.c` into the directory `serie03`.
Hint: Write functions `createVector` and `printVector` that are called in the main program.

Aufgabe 3.5. Write a recursive function `double powN(double x, int n)` which computes x^n for all exponents $n \in \mathbb{Z}$ and $x \in \mathbb{R}$. It holds $x^0 = 1$ for all $x \in \mathbb{R} \setminus \{0\}$. For $n < 0$ use $x^n = (1/x)^{-n}$. Moreover, $0^n = 0$ for $n > 0$. The term 0^n for $n \leq 0$ is not defined. In that case, the function should return the value `0.0/0.0`. You must not use the function `pow` from the math library. Save your source code as `powN.c` into the directory `serie03`.

Aufgabe 3.6. One way (not the best way) to approximate the number π is the so called *Leibniz formula*

$$\pi = \sum_{k=0}^{\infty} \frac{4(-1)^k}{2k+1}.$$

The n-th partial sum

$$P(n) = \frac{4(-1)^n}{2n+1} + P(n-1)$$

can be interpretated as a recursive function and it holds $\lim_{n\to\infty} P(n) = \pi$. Write a function `double P(int n)` that computes $P(n)$. Moreover, write a main program that reads $n \in \mathbb{N}$ from the keyboard and prints out the resulting n-th partial sum $P(n)$. Save your source code as `pirecursive.c` into the directory `serie03`.

Aufgabe 3.7. According to an old legend, there was a temple in Hanoi, which contained a large room with three time-worn posts in it surrounded by 64 golden disks of different diameters. When the temple was erected, the disks were arranged in a neat stack in ascending order of size on the first rod, the smallest at the top, thus making a conical shape. Since that time, the temple priests have been moving the disks with the objective of moving the entire stack to the third rod (preserving the ascending order). The second rode is auxiliary. All disk movements must be in accordance with the following immutable rules:

- Only one disk can be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack, i.e., a disk can only be moved if it is the uppermost disk on a stack.
- No disk may be placed on top of a smaller disk.

According to the legend, when the last move will be completed, the world will end.
The task can be accomplished with the use of a recursive algorithm. Let n denote the total number of disks ($n = 64$ in the original legend). In order to move the upper $m \leq n$ disks located on the i-th rode to the j-th rod ($i, j \in \{1, 2, 3\}$), proceed as follows:

1. Move the upper $m-1$ disks from the i-th rod to the k-th rod, with $k \notin \{i, j\}$;
2. The largest of the m disks is now on the top of the i-th rod and can be moved to the j-th rod;
3. Finally, the $m-1$ disks from in Step 1 can be moved from the k-th rod to the j-th rod.

The choice of $m = n$, $i = 1$ and $j = 3$ in the above algorithm solves the priest task. Please write a recursive function `void hanoi(int m, int i, int j)` which implements the algorithm. Any single movement of any disk must be printed out on the screen, e.g.,

```
Move a disk from Rode 2 to Rode 3.
```

Furthermore, write a main program that reads n from the keyboard and prints out the list of all movements. To test the algorithm, use $n \ll 64$, e.g., $n = 3, 4, 5$. Save your source code as `hanoi.c` into the directory `serie03`.

Aufgabe 3.8. Recall the meanings of the terms *Lifetime & Scope*. What is the output of the following code lines and explain why?

```
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     x = 100;
14     y = 4;
15     int z = max(x,y);
16     printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
17
18     {
19       x = y;
20       int 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 }
```

Draw a timeline and visualize the lifetime and the scope of the variables `x,y,z`. Moreover, mark all blocks and functions.