Dirk Praetorius
Michele Ruggeri

Summer semester 2019
March 15, 2019

# Scientific programming in mathematics

## Exercise sheet 2

### Functions & Recursion

**Exercise 2.1.** Write a function `radian`, which, given the magnitude of an angle $\theta \in \mathbb{R}^+$ measured in degrees, computes and returns its measure in radians. The computed value $\psi$ must be in reduced form, i.e., $\psi \in [0, 2\pi)$. Save your source code as `radian.c` into the directory `series02`.

**Exercise 2.2.** Given the expression of a function $f : \mathbb{R} \to \mathbb{R}$, a *curve sketching* is the study of the function aimed at producing a rough plot of the plane curve $y = f(x)$. Write a `void` function `curveSketching` that, given $a, b, c \in \mathbb{R}$, performs the curve sketching of a quadratic function $f(x) = a + bx + cx^2$. Specifically, analyzing with conditional statements the input coefficients $a, b, c \in \mathbb{R}$, the function determines and prints to the screen the type of the curve (note that the curve can be either a parabola or a straight line). Then, the function determines and prints to the screen the intersections of the curve with the coordinate axes. Finally, the function computes and prints to the screen the extrema of the function (if existing), specifying also the corresponding type (maximum or minimum). Moreover, write a main program, which reads the parameters $a, b, c$ from the keyboard and calls the function. Save your source code as `curveSketching.c` into the directory `series02`.

**Exercise 2.3.** Write a `void` function `points` which, given three points $(x, y)$, $(u, v)$, and $(a, b)$ in $\mathbb{R}^2$, checks whether they lie on the same line and prints to the screen the result. Moreover, write a main program which reads the six coordinates of the points from the keyboard and calls the function. Save your source code as `points.c` into the directory `series02`.

**Exercise 2.4.** Write a `void` function `quadrant`, which, given the two coordinates of a point $(x, y) \in \mathbb{R}^2$, prints to the screen the position of the point. Specifically: The function tells whether the point lies on one of the two coordinate axes. If this is not the case, then the function prints to the screen the number of the quadrant in which the point $(x, y)$ is located. Moreover, write a main program which reads the coordinates of the point from the keyboard and calls the function. Save your source code as `quadrant.c` into the directory `series02`.

**Exercise 2.5.** Write a recursive function `binomial` which computes and returns the binomial coefficient $\binom{n}{k}$ of two given integers $0 \le k \le n$. Use the addition formula

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1} \quad \text{for } 1 \le k < n$$

with $\binom{n}{0} = 1 = \binom{n}{n}$. Write a main program which reads $k$ and $n$ from the keyboard and prints to the screen the result $\binom{n}{k}$. Save your source code as `binomial.c` into the directory `series02`.

**Exercise 2.6.** Write a recursive function `division`, which computes and returns the result of the integer division $m/n$ (division without remainder) of two given integers $m \ge 0$ und $n > 0$. Only the arithmetic operations `+` and `-` can be used by the function. Moreover, write a main program which reads $m$ and $n$ from the keyboard and print the result $m/n$ to the screen. *Hint:* For any $y \neq 0$, it holds that $x/y = 1 + (x - y)/y$. Save your source code as `division.c` into the directory `series02`.

**Exercise 2.7.** The Fibonacci sequence is recursively defined by $x_0 := 0$, $x_1 := 1$, and $x_{n+1} := x_n + x_{n-1}$. Write a recursive function `fibonacci`, which computes and returns $x_n$ for given $n \in \mathbb{N}_0$. Then, write a main program, which reads $n$ from the keyboard and prints to the screen the value of $x_n$. Save your source code as `fibonacci.c` into the directory `series02`.

**Exercise 2.8.** 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 `series02`.