Dirk Praetorius
Giovanni Di Fratta

Summer semester 2021
March 26, 2021

Scientific programming in mathematics

Exercise sheet 3

For Loops, Computational Complexity, Comments

Exercise 3.1. 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 `series03`.

Exercise 3.2. 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 `series03`.

Exercise 3.3. One way (not the best way) to approximate the number π is based on the so-called *Leibniz formula*

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

In particular, for any $n \in \mathbb{N}_0$, the n-th partial sum

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

can be understood as an approximation of π (it holds that $\lim_{n\to\infty} S_n = \pi$). Write a function `double partialSum(int n)` that computes S_n for given $n \in \mathbb{N}_0$. Implement two different versions of the function: one is recursive, one computes the partial sum with a suitable loop. Moreover, write a main program that reads $n \in \mathbb{N}_0$ from the keyboard and prints the resulting approximation S_n of π to the screen. Save your source code as `piApproximation.c` into the directory `series03`.

Exercise 3.4. 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 $n \le 0$ 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 `series03`.

Exercise 3.5. The Fibonacci sequence is defined by $x_0 := 0$, $x_1 := 1$ and $x_{n+1} := x_n + x_{n-1}$ for $n \ge 1$. Write a *nonrecursive* function `fibonacci(k)`, which, given an index k, computes and returns x_k. Then, write a main program which reads k from the keyboard and displays x_k. Save your source code as `fibonacci.c` into the directory `series03`.

Exercise 3.6. Write a non-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 `series03`.

Exercise 3.7. Let x be a finite sequence of numbers (dynamic array of type `int`) and $n \in \mathbb{Z}$ some given bound. Write a function `y=cut(x,n)` that removes all entries $x(j)$ of x with $x(j) \geq n$, i.e., y is a shortened (reallocated) x. Further, write a main program which reads in the vector x and the bound n. How did you check your code for correctness? Save your source code as `cut.c` into the directory `series03`.

Exercise 3.8. Write a function `geometricMean` that computes and returns the geometric mean value

$$\overline{x}_{\mathrm{geom}} = \sqrt[n]{\prod_{j=1}^{n} x_j}$$

of a given vector $x \in \mathbb{R}^n_{\geq 0}$. The length $n \in \mathbb{N}$ should be a constant in the main program, but the function `geometricMean` should be implemented for arbitrary lengths n. Furthermore, write a main program that provides the input vector, call the function, and prints to the screen its geometric mean. How did you test the correctness of your code? Save your source code as `geometricmean.c` into the directory `series03`.