Dirk Praetorius,
Michele Ruggeri

Wintersemester 2017/18
16./17.11.2017

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 4

Aufgabe 4.1. Write a function `lcm`, which computes for two given natural numbers $a, b \in \mathbb{N}$ the least common multiple and returns it. Additionally, write a main program, which reads in the numbers $a, b \in \mathbb{N}$ and prints the least common multiple on the screen. How did you test the correctness of your code? Save your source code as `lcm.c` into the directory `serie04`.

Aufgabe 4.2. Write a `void`-function `printUnique`, which prints for a given vector $x \in \mathbb{R}^n$ all unique elements on the screen. Additionally, write a main program, which reads in the vector x and which calls the function. The length of the vector should be a constant in the main program. The function `printUnique` should be programmed for any given n. Check via `assert`, that $n \in \mathbb{N}$. How did you test the correctness of your code? Save your source code as `printUnique.c` into the directory `serie04`.

Aufgabe 4.3. Write a `void`-function `cross`, which prints for given $n \in \{1, 2, \ldots, 9\}$ a "X" consisting of $2n - 1$ lines on the screen. For $n = 6$, one should get for example the following output:

```
1          1
 2        2
  3      3
   4    4
    5  5
     6
    5  5
   4    4
  3      3
 2        2
1          1
```

Check via `assert`, that $n \in \{1, 2, \ldots, 9\}$. Write a main program, which reads in the parameter n and which calls the function `cross`. How did you test the correctness of your code? Save your source code as `cross.c` into the directory `serie04`.

Aufgabe 4.4. A natural number $a \in \mathbb{N}$ is called Armstrong-number if and only if the sum of its digits to the power of the number of digits coincides with a itself, i.e., $a := 10^{n-1} \cdot a_n + 10^{n-2} \cdot a_{n-1} + \ldots + 10^0 \cdot a_1$ with $n \in \mathbb{N}$, $a_i \in \{0, 1, \ldots, 9\}$, $0 < i \leq n$, $a_n \neq 0$, is an Armstrong-number, if $a_1^n + a_1^n + \ldots + a_n^n = a$. For example, the first Armstrong-numbers are $1, 2, \ldots, 8, 9, 153, 370, \ldots$
Write a `void`-function `armstrong`, that obtains a lower bound $b \in \mathbb{N}$ and an upper bound $c \in \mathbb{N}$ with $b \leq c$ and that returns all Armstrong numbers within these bounds. Check via `assert` if $b \leq c$ Write a main program which reads in the bounds and calls the function `armstrong`. How did you test the correctness of your code? Save your source code as `armstrong.c` into the directory `serie04`.

Aufgabe 4.5. Write a function `energy` that returns the energy $e = \sum_{j=1}^{n} x_j^2$ of a given vector $x \in \mathbb{R}^n$. Further, write a main program which reads in the vector x and calls the function The dimension $n \in \mathbb{N}$ is a constant in the main program, the function `energy` should be implemented for arbitrary dimensions. How did you test the correctness of your code? Save your source code as `energy.c` into the directory `serie04`.

Aufgabe 4.6. Let x be a sequence of 10 numbers (static array of type `int`) and y a combination of 3 numbers (array of type `int`) which are both read from the keyboard. Write a function `check` that obtains the two arrays as input and checks if the combination y is contained in the sequence x (return value 1) or not (return value 0). Further, write a main program which reads in the arrays x and y and calls the function. How did you test the correctness of your code? Save your source code as `check.c` into the directory `serie04`.

Aufgabe 4.7. The cosine function can be represented as a series via

$$\cos(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}.$$

The corresponding n-th partial sum is given by

$$C_n(x) = \sum_{k=0}^{n} (-1)^k \frac{x^{2k}}{(2k)!}.$$

Write a *nonrecursive* function `cos_new`, which, given $x \in \mathbb{R}$ and $\tau > 0$, returns the first value of $C_n(x)$ such that

$$|C_n(x) - C_{n-1}(x)|/|C_n(x)| \leq \tau \quad \text{or} \quad |C_n(x)| \leq \tau.$$

Then, write a main program, which reads $x \in \mathbb{R}$ and $\tau > 0$ from the keyboard, calls the function and displays the computed value $C_n(x)$, as well as the value $\cos(x)$, the absolute error $|C_n(x) - \cos(x)|$ and the relative error $|C_n(x) - \cos(x)|/|\cos(x)|$ (provided $\cos(x) \neq 0$). Write the function in such a way that only one loop is needed and that x^{2k} and $(2k)!$ are computed in a cheap way. This means, you must not use specific functions to compute the power or the factorial. How did you test the correctness of your code? Save your source code as `cos.c` into the directory `serie04`.

Aufgabe 4.8. For $x > 0$, the recursively defined sequence

$$x_1 := \frac{1}{2}(1 + x), \quad x_{n+1} := \frac{1}{2}\left(x_n + \frac{x}{x_n}\right) \quad \text{for } n \geq 1$$

converges towards $\sqrt{x}$. Write a *nonrecursive* function `sqrt_new` which computes for given $x > 0$ and $\tau > 0$ the *first* element x_n of the sequence that satisfies

$$\frac{|x_n - x_{n+1}|}{|x_n|} \leq \tau \quad \text{or} \quad |x_n| \leq \tau.$$

Check via `assert`, that $x > 0$. Moreover, write a main program which reads in x and τ, computes the approximation x_n of $\sqrt{x}$ and compares it to the exact value, i.e. prints out the absolute error $|x_n - \sqrt{x}|$. How did you test the correctness of your code? Save your source code as `sqrt_new.c` into the directory `serie04`.