Dirk Praetorius,
Michele Ruggeri

Wintersemester 2015/16
04.-06.11.2015

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 4

Aufgabe 4.1. The Fibonacci sequence is defined by $x_0 := 0$, $x_1 := 1$ and $x_{n+1} := x_n + x_{n-1}$. 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 `serie04`. Compare your implementation to your code from Exercise 3.5. Discuss the advantages and disadvantages of both implementations!

Aufgabe 4.2. Write a *nonrecursive* function `binomial` which computes the binomial coefficient $\binom{n}{k}$. Use an appropriate loop and the identity $\binom{n}{k} = \frac{n\cdot(n-1)\cdots(n-k+1)}{1\cdot 2\cdots k} = \frac{n}{1}\cdot\frac{n-1}{2}\cdots\frac{n-k+1}{k}$. Additionally, write a main program that reads in the values $k, n \in \mathbb{N}_0$ with $k \le n$ and prints out $\binom{n}{k}$. Save your source code as `binomial.c` into the directory `serie04`.

Aufgabe 4.3. A triple $(x, y, z) \in \mathbb{N}^3$ of natural number is called a *Pythagorean triple* if it holds $x^2+y^2 = z^2$. The most common example would be $(3, 4, 5)$. Obviously we have $z > \max\{x, y\}$ as well as $x \ne y$ and without the loss of generality we can assume $x < y$. Write a void function `pythagoras`, that, for a given $n \in \mathbb{N}$ calculates and prints all Pythagorean tiples $x < y < z \le n$. Moreover, write a main-programme, that reads in n and calls `pythagoras`. Save your source code as `pythagoras.c` into the directory `serie04`..

Aufgabe 4.4. Write a `void`-function `multiple(k,nmax)`, which computes and displays all the integer multiples of $k \in \mathbb{N}$ which are $\le n_{\max} \in \mathbb{N}$. For instance, for $k = 5$ and $n_{\max} = 19$, the function yields the output

```
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15.
```

Then, write a main program, which reads the values k and n from the keyboard and calls `multiple(k,nmax)`. Save your source code as `multiple.c` into the directory `serie04`.

Aufgabe 4.5. Write a function `scalarproduct`, which, given two vectors $x, y \in \mathbb{R}^n$, computes the scalar product $x \cdot y := \sum_{j=1}^n x_j y_j$. The length $n \in \mathbb{N}$ should be a constant in the main-programme, but the function `scalarproduct` should be programmed for arbitrary lengths n. Furthermore, write a main program which reads in $x, y \in \mathbb{R}^n$ and calls `scalarproduct`. Save your source code as `scalarproduct.c` into the directory `serie04`.

Aufgabe 4.6. 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 $x \in \mathbb{R}^n_{\ge 0}$ The length $n \in \mathbb{N}$ should be a constant in the main-programme, but the function `geometricMean` should be programmed for arbitrary lengths n. Furthermore, write a main program which reads in $x \in \mathbb{R}^n$ and calls `geometricMean`. Save your source code as `geometricMean.c` into the directory `serie04`.

Aufgabe 4.7. Write a function `maxcompare`, that counts for given $a, b \in \mathbb{R}^n$ how often the maximum of the vectors a and b denoted by $M := \max\{a_i, b_i \,|\, i = 1, \dots, n\}$ is represented in a and b at the same position. For example for the vectors $a = (1.1, 4, 2e-4, 4, 4, 3, 4, -1.5)$ and $b = (2.2, 4, 4, 2e-5, 4, -1, 2.7, 4)$ we have $M = 4$. The function should thus return 2, since $a_2 = b_2 = a_5 = b_5 = M = 4$. If, for example, M is represented only in a or b, then, clearly, the function should return 0. The length $n \in \mathbb{N}$ should be a constant in the main-programme, but the function `maxcompare` should be programmed for arbitrary lengths n. Furthermore, write a main program which reads in $a, b \in \mathbb{R}^n$ and calls `maxcompare`. Save your source code as `maxcompare.c` into the directory `serie04`.

Aufgabe 4.8. Write a `main`-programme that prints the first k lines of Pascal's triangle: Every line starts and ends with 1. The remaining entries are the sum of two neighbouring entries from the line above. For $k = 5$ we have for example

$$\begin{array}{ccccccccc} & & & & 1 & & & & \\ & & & 1 & & 1 & & & \\ & & 1 & & 2 & & 1 & & \\ & 1 & & 3 & & 3 & & 1 & \\ 1 & & 4 & & 6 & & 4 & & 1 \end{array}$$

See also:

`http://en.wikipedia.org/wiki/Pascal's_triangle`

The function should be implemented efficiently. You must not use the representation of the entries with the binomial coefficients. Furthermore, always store only *one* line in *one* vector with the length k and overwrite the vector in each step in an appropriate way. The length $k \in \mathbb{N}$ should be a constant in the main-programme. Save your source code as `pascal.c` into the directory `serie04`.