Dirk Praetorius, Sommersemester 2014
Thomas Führer 31.03.2014

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 5

Aufgabe 5.1. Write a function `minmaxmean` which computes and returns the minimum, maximum, and the mean value $\frac{1}{n}\sum_{j=1}^{n}$ of a given vector $x \in \mathbb{R}^n$. Additionally, write a main program that reads in a vector $x \in \mathbb{R}^n$ and prints out the minimum, maximum, and mean value of it. The length n of the vector should be constant in the main program, but the function `minmaxmean` should be programmed for arbitrary lengths n.

Aufgabe 5.2. Write a function `lcm` that computes the *least common multiple* of two given natural numbers $a, b \in \mathbb{N}$. For the solution, you can either compute the prim factors of both numbers or use the relation $ab = \gcd(a,b) \cdot \mathrm{lcm}(a,b)$, where $\gcd(a,b)$ denotes the *greatest common divisor*.

Aufgabe 5.3. Write a function `exponential` which approximates the value $\exp(x)$ by the partial sum

$$S_N(x) := \sum_{j=0}^{N} \frac{x^j}{j!},$$

where $N \in \mathbb{N}$ satisfies the condition

$$\Big|\frac{x^{N+1}}{(N+1)!}\Big| \le \Big|\frac{x^N}{N!}\Big| \le \varepsilon$$

for a given tolerance $\varepsilon > 0$. The computation of the summands $x^j/j!$ should be realized efficiently. Compare the absolute errors $|S_N(x) - \exp(x)|$ for different values of ε and evaluation points $x \in \mathbb{R}$.

Aufgabe 5.4. The quotient sequence $(a_{n+1}/a_n)_{n\in\mathbb{N}}$ corresponding to the Fibonnaci-sequence $(a_n)_{n\in\mathbb{N}}$,

$$a_0 := 1, \quad a_1 := 1, \quad a_n := a_{n-1} + a_{n-2} \quad \text{für } n \ge 2,$$

converges towards the *golden ratio* $(1+\sqrt{5})/2$. In particular, the difference sequence

$$b_n := \frac{a_{n+1}}{a_n} - \frac{a_n}{a_{n-1}}$$

converges towards 0. Write a function `cauchy` that returns, for given $k \in \mathbb{N}$, the smallest $n \in \mathbb{N}$ such that $|b_n| \le 1/k$. Moreover, write a main program that reads in $k \in \mathbb{N}$ and prints out the index $n \in \mathbb{N}$.

Aufgabe 5.5. The *Bubble-Sort* algorithm is an inefficient, but short sorting algorithm which works as follows: You run through the entries of a given vector $x \in \mathbb{R}^n$ several times. In every run, each entry x_j of is compared to its successor x_{j+1} and if $x_j > x_{j+1}$, the two entries x_j,x_{j+1} are swapped. After the first complete run through the vector, one knows that (at least) the last element is sorted correctly, i.e. the last element x_n is the maximum of the vector. Thus, in the next run one only has to go up-to the last-but-one entry of the vector. How many loops do you need for this algorithm? Write a function `bubblesort` which sorts a given vector $x \in \mathbb{R}^n$ with this algorithm. Additionally, write a main program that reads in $x \in \mathbb{R}^n$ and sorts it. The length n should be constant. However, your function `bubblesort` should be programmed for aribtrary lengths n.

Aufgabe 5.6. Let the two series

$$a_N := \sum_{n=0}^{N} \frac{1}{(n+1)^2} \quad \text{und} \quad b_M := a_M^2 = \sum_{m=0}^{M}\sum_{k=0}^{m} \frac{1}{(k+1)^2(m-k+1)^2}$$

be given. Write a program that measures the time used for the computation of a_N resp. b_M for different values of N resp. M. Print out the results tabularly. Do the results meet your expectations? *Hint:* Think of the computational complexity (Aufwand) for the computation of a_N resp. b_M.

Aufgabe 5.7. The function squareVector should square all entries of a given vector $x \in \mathbb{R}^n$, i.e., the input $(-1, 2, 0)$ should be turned into $(1, 4, 0)$. The input vector should be passed as a pointer.

```
#include <stdio.h>

int squareVec(double vec, int n) {
  int j=0;
  for(j=1, j<dim; --j) {
    *vec[j] = &vec[j] * &vec[j];
  }
  return vec;
}

main() {
  double vec[3] = {-1.0,2.0,0.0};
  int j=0;

  squareVec(vec,3);
  for(j=0; j<3; ++j) {
    printf("vec[%d] = %f ",j,vec[j]);
  }
  printf("\n");
}
```

Change *only* the function squareVec, such that the main programm prints out the correct result. How many errors do you find? What is the computational complexity (Aufwand) of squareVec?

Aufgabe 5.8. Which types of comments do you know? What is the output of the following code and why?

```
#include <stdio.h>

/*int f(double x) {
    return (int) x;
  }
*/

main() {
  int x = 4;
  int y = 2*x*/* f(0.1)+3
              */1/4;
  // y = 1;
  printf("y = %d\n",y); // Print out result
}
```