Dirk Praetorius,
Michele Ruggeri

Wintersemester 2017/18
23./24.11.2017

Übungen zur Vorlesung Einführung in das Programmieren für TM

Serie 5

Aufgabe 5.1. 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 `serie05`.

Aufgabe 5.2. Write a function `double* dec2bin(int N, int* n)`, which, given a natural number $0 \leq N < 65535$, computes and returns its representation in the binary numeral system. The program has to determine the coefficients $a_i \in \{0,1\}$, $i = 0, \dots, n-1$, such that $N = \sum_{i=0}^{n-1} a_i 2^i$ $(n \leq 16)$. The binary representation of N should be returned without leading zeros. The function 'returns' also the length of the dynamical vector. For instance, for $N = 77$, the function returns the vector `1 0 0 1 1 0 1`. Moreover, write a main program, which reads N from the keyboard and prints to the screen its binary representation. How did you test the correctness of your code? Save your source code as `dec2bin.c` into the directory `serie05`.

Aufgabe 5.3. A well-known root-finding algorithm is the *Newton method.* Let $f : [a,b] \to \mathbb{R}$. Given an initial guess x_0, define the sequence $(x_n)_{n\in\mathbb{N}}$ via

$$x_n = x_{n-1} - f(x_{n-1})/f'(x_{n-1}) \quad \text{for } n \geq 1.$$

Implement the algorithm in a function `double newton(double (*fct)(double), double (*fctprime)(double), double x0, double tau)`, which, given x_0 and a tolerance $\tau > 0$, performs the Newton iteration until

$$|f'(x_n)| \leq \tau$$

or

$$|f(x_n)| \leq \tau \quad \text{and} \quad |x_n - x_{n-1}| \leq \begin{cases} \tau & \text{for } |x_n| \leq \tau, \\ \tau |x_n| & \text{else.} \end{cases}$$

In the first case, print a warning to inform that the result is presumably wrong. The function returns the value x_n of the approximate root. Use `assert` to check whether $\tau > 0$ holds. The function gets function pointers of `double f(double x)` and its derivative `double fprime(double x)`. Write a main program, which reads x_0 and τ from the keyboard and prints x_n to the screen. How can you test your code? What are good examples? What is the connection between the Newton method and Exercise 4.8 from the last exercise sheet? Save your source code as `newton.c` into the directory `serie05`.

Aufgabe 5.4. Given a differentiable function $f : [a,b] \to \mathbb{R}$ and $x \in [a,b]$, the derivative $f'(x)$ can be approximated by the different quotient

$$\Phi(h) := \frac{f(x+h) - f(x)}{h} \quad \text{for } h > 0.$$

Write a function `double* diff(double (*fct)(double), double x, double h0, double tau, int* n)`, which computes the sequence $\Phi(h_n)$, where $h_n := 2^{-n} h_0$, until

$$|\Phi(h_n) - \Phi(h_{n+1})| \leq \begin{cases} \tau & \text{if } |\Phi(h_n)| \leq \tau, \text{or} \\ \tau\, |\Phi(h_n)| & \text{else.} \end{cases}$$

The function gets the function `double f(double x)` via function pointer and returns the vector of the complete sequence $(\Phi(h_0), \dots, \Phi(h_n))$, as well as the length of the vector. Then, write a main program containing suitable examples to test your implementation. How did you test the correctness of your code? Save your source code as `diff.c` into the directory `serie05`.

Aufgabe 5.5. The Aitken Δ^2-method is an acceleration method for sequences. For an injective sequence $(x_n)_{n\in\mathbb{N}}$ with $\lim_{n\to\infty} x_n = x$, define the sequence

$$y_n := x_n - \frac{(x_{n+1}-x_n)^2}{x_{n+2}-2x_{n+1}+x_n}.$$

Under appropriate assumptions on the sequence $(x_n)_{n\in\mathbb{N}}$, it holds that

$$\lim_{n\to\infty} \frac{y_n - x}{x_n - x} = 0,$$

i.e., the sequence $(y_n)_{n\in\mathbb{N}}$ converges towards x faster than $(x_n)_{n\in\mathbb{N}}$. Write a function `double* aitken(double* x, int n)`, which, given a vector $x \in \mathbb{R}^n$ with length $n \geq 3$, computes the vector $y \in \mathbb{R}^{n-2}$. Use `assert` to ensure that $n \geq 3$. Test your implementation with suitable examples. Write a main program, which reads the vector x and the length n from the keyboard and prints the vector y to the screen. What happens for the sequence $x_n = q^n$ with $0 < q < 1$? Save your source code as `aitken.c` into the directory `serie05`.

Aufgabe 5.6. Combine the Aitken method of Exercise 5.5 with the difference quotient $\Phi(h)$ from Exercise 5.4. With $h_n := 2^{-n}h_0$, we consider the sequence $x_n := \Phi(h_n)$ and obtain the sequence (y_n). Write a function `double diffaitken(double (*fct)(double),double x, double h0, double tau)` that receives the function f, the evaluation point x, the step size $h_0 > 0$ as well as the tolerance $\tau > 0$ and returns $y_{n+1} \approx f'(x)$ as soon as there holds that

$$|y_n - y_{n+1}| \leq \begin{cases} \tau, & \text{falls } |y_{n+1}| \leq \tau, \\ \tau\,|y_{n+1}|, & \text{anderenfalls.} \end{cases}$$

The function shall work for arbitrary real-valued functions `double f(double x)`. In each step, display h_{n+1}, $|y_{n+1} - y_n|$ and y_{n+1} on the screen. Save the source code speichere in `serie05`. As example, consider the calculation of $e = \exp(1) = \exp'(1)$ and $\varepsilon = 10^{-12}$. Compare the number of iterations with and without (i.e., $y_n = x_n$) Aiken method. How and with what functions did you test your code? Save your source code as `diffaitken.c` into the directory `serie05`.

Aufgabe 5.7. Write a recursive function `void mergesort(double* x, int n)` which sorts a vector $x \in \mathbb{R}^n$ in ascending order using the *mergesort* algorithm. Use the following strategy:

- If $n \leq 2$, then the vector $x \in \mathbb{R}^n$ is explicitly sorted.
- If $n > 2$, then the vector x is split into two subvectors y and z of half length. Then the function `mergesort` is recursively called for y and z. Finally, y and z are merged into a sorted vector. Use explicitly the fact, that y and z are already sorted at that moment.

Write a main program, which reads the vector x and its length n from the keyboard, sorts it with `mergesort` and prints to the screen the sorted vector. Test your program accurately! What is the computational cost of your function? Save your source code as `mergesort.c` into the directory `serie05`.

Aufgabe 5.8. Implement the mergesort algorithm from Exercise 5.7 without allocating additional vectors in the recursion step. Instead, use pointer arithmetic: If `x` is the base-pointer of the array x (i.e. the pointer to x_0), then `x+k` is the base-pointer of x_k. Hence for the recursion step, it is sufficient, to simply have the base-pointer to x_0, the starting index k and the ending index ℓ of a part of x as input parameters. For the sorted final array you can *uniquely* allocate dynamic memory at the beginning. No additional memory is needed. How did you test the correctness of your code? Save your source code as `mergesort2.c` into the directory `serie05`.