

Programming Exercise 7

Due Date and Time: June 2, 2009, 23:59

You should work out the exercises independently. Group cooperation is not allowed. Submit the sensibly documented program files. (Files for Bsp. 5: `bsp_5.cpp`, `bsp_5_fkt.cpp`, `bsp_5_fkt.hpp`) Copy these to your directory in the global submission folder.

21. In Problem 19 (Exercise 6), we only initialized the lengths of vectors (number of elements) statically.

Modify your code (alternately the example solution) corresponding to §6.4 of the lecture notes so that the vectors **dynamisch** are dynamically allocated after the query of the vector length n and then carry out the computation from problem 19. Don't forget the explicit deallocation from dynamic memory. (1 Pt.)

Input data (n): (4), (20), (31553)

22. In mathematics, the Fibonacci numbers form a sequence defined recursively by the following formulas:

$$F_n = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ F_{n-1} + F_{n-2} & \text{if } n > 1. \end{cases}$$

Write a **recursive** function `fibonacci(n)` that computes the n -th element of the sequence. Show that the ratio of a term with respect to the previous one tends to $\frac{1 + \sqrt{5}}{2}$. (1 Pt.)

Input data (n): (2), (20), (60)

23. Modify your main program from the previous problem so that:

- the Variable n is given as a **Kommandozeilenparameter** (Command line parameter), see §7.6 of the lecture notes.
- Use `cin` (as previously) to input the value, in the event that no command line parameter has been given. (3 Pts.)
- Write an additional function for computing the Fibonacci numbers, which is realized within a loop.
- Compare the runtimes of your two Fibonacci functions (from problems 22 and 23) by means of **Zeitmessung** (time measurement) in §11.2 of the lecture notes.

Input data (n): (4), (20), (40), (60)

24. Implement a function `riemann` that (numerically) integrates a given function $f(x)$ by means of the Riemann Sum (1 Pt.)

$$\int_a^b f(x)dx = \sum_{j=1}^n f(a + jh) \cdot h$$

where the given interval from a to b is partitioned into n given, uniformly-sized subintervals. The function to be integrated is also given INPUT-size of the list of parameters. See the example function `Bisect3` in §7.8¹ of the lecture (also in `Bisect3.cpp`²).

Test data: `sqrt()` in $[1, 4]$, `cos()` in $[0, \pi/2]$, `log()` in $[1, e]$

25. Implement a function `derivative` that accepts an input function $f(x)$ and that numerically differentiates it on a specified point x , using a difference step length h . Use the formula (1 Pt.)

$$f'(x) = \frac{f(x + h) - f(x - h)}{2h}$$

Test data: `sqrt()` in $x = 4$, `cos()` in $x = \pi/6$, `log()` in $x = 1$,

26. In this exercise, you must unconditionally **adhere to the given file designations**, which will be `bsp_26_fkt.hpp` for the header file and `bsp_26_fkt.cpp` for the source file of your written functions. Copy your functions `derivative` and `riemann` from the previous two exercises in the header and source files and test their correct usage in a short main program. (1 Pt.)

_____ **Ende für LAK** _____

27. Write a improved functions `riemann` and `derivative` where the given integral (respectively, the derivative value), which returns an accuracy of ϵ . Save these functions in the files `bsp_26_fkt.hpp`, `bsp_26_fkt.cpp` as well. Test their correct usage with the value $\epsilon = 10^{-4}$ with the test data from exercises 24 and 25 in your main program. (2 Pts.)

Pay special attention to the differentiation, in that the step length h is not too small. (Machine epsilon in §3.9³ alternately, `Reihe.cpp`⁴).

¹<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/Script/html/node81.html>

²<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/Examples/Bisect3.cpp>

³<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/Script/html/node45.html>

⁴<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/Examples/Reihe.cpp>

28. Newton's Method is very efficient at determining the zeros (root-finding) of a function $f(x)$. Newton-Iteration⁵

$$x_{n+1} := x_n - f(x_n)/f'(x_n)$$

for a suitable initial guess $x_0 \in [a, b]$ (It is not required that your program validate the mathematical condition that the function $x_n - f(x_n)/f'(x_n)$ is a contraction map). (4 Pts.)

Use your function from exercises 25 and 26 and extend the files `bsp_26_fkt.hpp`, `bsp_26_fkt.cpp` with your function `newton_diff_approx()`. Test your root-finding solver (and verify the solution) in the main program with the following functions:

$$\begin{aligned} \text{Test data: } f(x) &:= \sin(x) - x/2 \\ g(x) &:= -(x - 1.234567)(x + 0.987654) \\ h(x) &:= 3.0 - e^x \\ z(x) &:= \cos(x) - e^{x \ln |x|} \end{aligned}$$

general instructions:

The commands `break`, `continue`, `goto` are **not permitted**.

Submission of solutions:

The submission of the solutions (as `*.cpp`-Files, `*.hpp`, ..., **no exe**-Files) may be carried out exclusively by copying these files in your directory (initially created as) `Lastname_Firstname` into the global submission folder `y: \\pers.ad.uni-graz.at\fs\ou\621\stud_haase>Lastname_Firstname`.

Instructions on how to do this can be found at: LV-Homepage⁶

The file names **must** follow the format `bsp_nummer`, with the corresponding file extension. For example, with Beispiel 1 (example 1) the file `bsp_1.cpp` would be submitted. Files with names which deviate from this formula will not be considered as submitted files.

⁵http://en.wikipedia.org/wiki/Newton's_method

⁶<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS08/index.html>