Einführung in die Programmierung für Physiker

SoSe 2024 – Marc Wagner

Michael Eichberg: eichberg@itp.uni-frankfurt.de

Exercise sheet 3

To be handed in on 10.05.2024.

Exercise 1 [*Knowing your machine*] (2 pts.)

There are two important libraries, which are not very often used, but which are important to know:

- `limits.h`
- `float.h`

Online there are many good reference websites. Most of them are reference for `C` and `C++` at the same time. A nice, not extremely technical one is `http://www.cplusplus.com/reference/clibrary/`. Have a look to the content of the above mentioned header files. Write a short computer program which prints the minimum and maximum numbers which can be stored inside variables of the following types on your machine.

- `short int`
- `int`
- `long int`
- `short unsigned int`
- `unsigned int`
- `long unsigned int`
- `float`
- `double`
- `long double`

Based on the output of your program, can you deduce how many bits are used for `int` and `unsigned int` variables on your machine?
Remark: Some limits are trivial and are thus not listed in the above library.

Exercise 2 [*4 ways to swap variables*] (3 pts.)

To *swap* means to *exchange* and, for our purposes, it is just about

- having two variables initialized with own values,
- assigning to each of the two variables the value originally assigned to the other.

You can work with either `int` or `float` numbers and start with two variables, which we call `a` and `b`. Write a program which swaps `a` and `b` by

(i) making use of a third auxiliary variable `c`,

(ii) implementing a chain of addition/subtraction operations,

(iii) implementing a chain of multiplication/division operations.

Time to Test! Depending on the values of the variables to swap, are there special cases, in which the swapping algorithm fails? Can you find the reason for the failure?

Exercise 3 [*The if-clause*] (1 + 3 = 4 pts.)

One of the most important building blocks in every programming language is the `if`-clause that is needed to test conditions and act according to the logical value of a given condition.

(i) Write a program with `if`-clauses for each of the following cases. Use an appropriate `printf` command in every branch to give information to the user.

  (a) Check, if a non negative integer numer is even.

  (b) Print the largest of two numbers.

  (c) Given 4 numbers, print the smallest.

(ii) In `C` there is the so-called *conditional operator* (also known as *ternary operator*), whose syntax reads

```
condition ? value_if_true : value_if_false
```

and it evaluates to `value_if_true` if `condition` is `true`, to `value_if_false` otherwise. It is important to remark that the conditional operator is an *expression*, whereas `if-else` is a *statement*. Being the result of an expression, the ternary operator can then be used wherever a value is needed, e.g. on the right-hand side in assignments. Practice a bit with this operator using it *exclusively* to accomplish the following tasks (do not use any `if`-clause here).

  (a) Assign the sign of an integer number to a different variable.

  (b) Assign the smallest of 2 numbers to a different variable.

  (c) Assign the largest of 3 numbers to a different variable.

As you might have notice in the last task, code can quickly become hard to read as soon as the conditions become more complicated and, maybe, more than one ternary operator is needed. This teaches you that an `if`-clause should be preferred in every non-trivial case.

Exercise 4 [*Series convergence*] (4 + 3 + 2 + 2 = 11 pts.)

Consider the sum

$$H_z^{(n)} = \sum_{k=1}^{n} \frac{1}{k^z} \qquad z \in \mathbb{R}\,.$$

We want to approximate the series $\zeta(z) \equiv \lim_{n\to\infty} H_z^{(n)}$ for a given z and relative precision ϵ. If the series does not converge, we want to stop the process and print a message.

(i) Write a function which takes z and ϵ as arguments and approximates $\zeta(z)$. Define a constant for the maximum number of terms $n_{\max} = 10^4$. Compute the sum using a `for` loop and return $\zeta(z) \approx H_z^{(n+1)}$, if the precision ϵ is achieved. If n exceeds $n_{\max}$, print an error message and terminate the process via `exit(1)`. As convergence criterium, you can use

$$|H_z^{(n+1)} - H_z^{(n)}| < \epsilon |H_z^{(n)}|\,.$$

Hint: At the top of your code file, below the `includes`, insert the lines

```
#ifndef NMAX
#define NMAX 10000
#endif
```

and remove the line where you previously defined $n_{\max}$ as a constant. With GCC, you can now give $n_{\max}$ different values at compile time via `gcc ... -DNMAX=... main.c`.

(ii) Write a main program to test the above function for values $z = 0.1$, 1, 1.1 and 10, as well as $\epsilon = 10^{-6}$. Print each result with 15 digits precision. What happens if you choose $n_{\max} = 10^8$? Where is the error?

Hint: The series $\lim_{n\to\infty} H_z^{(n)}$ converges for $z > 1$ and diverges for $z \leq 1$.

(iii) Add a function to your code which performs task (i) using a `while` loop and test it.

(iv) Refactor your code, such that the convercence criterium, which returns `true` or `false`, and the summand $\frac{1}{k^z}$ are defined in seperate functions. Reverse the order in which you define these functions. Can you still compile your code? What is a possible reason, that this is not allowed?