C++ Programming

Exercise Sheet 1
Software Engineering Group EIM-I
Philipp Schubert
May 03, 2019

Solutions to this sheet are due on 10.05.2019 til 14:00. Please hand in a digital version of your answers via e-mail. The e-mail’s subject has to contain cppp19. Do zip-compress your solutions. Note: If you copy text elements / code elements from other sources, clearly mark those elements and state the source. Copying solutions from other students is prohibited. All of your files that belong to your solution have to be contained in a single .zip file that is named according to the following naming scheme: <name>_<surname>_solution_<X>.zip. Replace <name> and <surname> with your actual name and replace <X> with the number of the exercise sheet. You can look up your results using this link: [Google Sheets Link]

This exercise sheet will help to make yourself familiar with the different kinds of control flow that exist in C++. Furthermore, you will start learning how to express mathematical and real-world problems in the C++ programming language. This is the first step to develop a computational thinking. You can achieve 16 points in total.

Exercise 1.

a) Write a program that reads an integer from the command line and checks if it is greater than or equal to 0 and smaller than or equal to 100! The program should print the result of the check to the command line. (2 P.)

b) Write another program that reads an integer from the command line and checks if the integer
1. is greater than 0
2. holds check 1. and is in addition dividable by 4
3. holds check 2. and is in addition dividable by 3
Print which of the checks hold for the integer. (Hint: use nested if statements) (2 P.)

Exercise 2.

Obviously you wish to analyze sequenced DNA now. DNA is (usually) made up from four different kinds of bases: guanin ‘G’/‘g’, adenin ‘A’/‘a’, cytosin ‘C’/‘c’ and thymin ‘T’/‘t’. The following std::string with the name dna stores the DNA sequence you would like to analyze. Until now, we only used strings as literals and never defined a variable of type string. We will learn about the non-built-in string data type std::string in the next lecture. For this exercise it is sufficient to include the header file string and to
know how to iterate a string which is shown in the code below. Use the code snippet in order to solve the next few tasks. You can download the code snippet here: [https://www.hni.uni-paderborn.de/fileadmin/Fachgruppen/Softwaretechnik/Lehre/CPP_Programming/SS2019/code_01.zip](https://www.hni.uni-paderborn.de/fileadmin/Fachgruppen/Softwaretechnik/Lehre/CPP_Programming/SS2019/code_01.zip)

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
    const string dna = "AGTcccaaaGTGACCAATGAatataAATCG";
    // this range - for loop iterates the string 'dna'
    for (char base : dna) {
        // you can use the variable 'base' inside this loop
    }
    return 0;
}
```

a) Iterate `dna` and use a switch statement in order to count the occurrences of each of the four different DNA bases! Use one counter variable for each base. Print the number of occurrences for each base on the command line! (1 P.)

b) Extend your program such that it is able to read a string from the command line! You can use the `string` data type in combination with `cin` just like you did for the built-in data types. (1 P.)

c) As a final extension to your DNA-processing-program, add some functionality that tells the user how many DNA bases are encoded as lower case letters and how many are encoded as upper case letters! (1 P.)

d) In which cases should you prefer a `switch` statement over an `if` statement and why? (1 P.)

**Exercise 3.**

Consider the following program that performs a simple numeric integration of \( \int_0^1 \frac{4}{1+x^2} \, dx \) and prints the result on the command line. The source code of this program can be found here: [https://www.hni.uni-paderborn.de/fileadmin/Fachgruppen/Softwaretechnik/Lehre/CPP_Programming/SS2019/code_01.zip](https://www.hni.uni-paderborn.de/fileadmin/Fachgruppen/Softwaretechnik/Lehre/CPP_Programming/SS2019/code_01.zip). Compile and run the program.

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    const long double from = 0.0;
    const long double to = 1.0;
    long double integral_val = 0.0;
    long double x = from;
    const size_t N = 1000000;
    const long double step_width = abs(from - to) / static_cast<long double>(N);
    for (size_t n = 0; n < N; ++n)
    {
        integral_val += 4 / (1 + x * x);
        x += step_width;
    }
    integral_val /= N;
    cout << integral_val << endl;
    return 0;
}
```
a) Modify and use the above numerical integrator program such that it calculates \( \int_{0}^{1} 3x^2 \, dx \). (1 P.)

b) Modify and use the above numerical integrator program such that it calculates \( \int_{0}^{1} \sqrt{2x} \, dx \). (1 P.)

c) Write a small program that computes \( \sum_{k=1}^{100} k \). (1 P.)

d) Write another small program that computes \( \sum_{i=1}^{10} (\sum_{j=1}^{10} i) \). (1 P.)

Exercise 4.
Write a program that prints the pattern to the command line which are shown in

a) figure 1 (1 P.)

b) figure 2 (1 P.)

c) figure 3 (2 P.)

Use nested loops and if statements! (All figures are 10 × 10 characters.)

Figure 1: Pattern A.

Figure 2: Pattern B.

Figure 3: Pattern C.