## Lab work #1

**Topics**

* Basic Input/Output
* Formatted output
* Types and auto
* Editing, compiling and executing programs

**Exercises**

1. Write a program that generates the following table:

1990 135

1991 7290

1992 11300

1993 16200

Use a single cout statement for all output.

1. Write a program that generates the following output:  
   10

20

19  
Use an integer constant for the 10, an arithmetic assignment operator to generate the 20, and a decrement operator to generate the 19.

1. Write a temperature-conversion program that gives the user the option of converting Fahrenheit to Celsius or Celsius to Fahrenheit. Then carry out the conversion. Use floating-point numbers. Interaction with the program might look like this:

Type 1 to convert Fahrenheit to Celsius,

2 to convert Celsius to Fahrenheit: 1

Enter temperature in Fahrenheit: 70

In Celsius that’s 21.111111

1. If you have two fractions, a/b and c/d, their sum can be obtained from the formula

a c a\*d + b\*c

--- + --- = -----------

b d b\*d

Write a program that asks the user to enter two fractions, and then displays their sum in fractional form. You don’t need to reduce it to the lowest terms. You can take advantage of the fact that the extraction operator (>>) can be chained to read in more than one quantity at once: cin >> a >> dummychar >> b;

1. By default, output is right-justified in its field. You can left-justify text output using the manipulator ios::left. Example:

std::cout << "Left :\n" << std::right << "Right " << std::endl;

Use this manipulator, along with the method setw(), to help generate the following output:

Last name First name Town Country

----------------------------------------------------------

Silva Joao Aveiro Portugal

Almeida Maria Madrid Spain

Santos Miguel Moscow Russia

1. Consider the C module point (still under development), which allows working with the representation of points in 2D space. The file point.h defines a data structure (Point2D) to represent a point as a pair of real values(x, y) and the prototype of the function print(Point2D p). The implementation of this function is in the file point.cpp and that writes the point in the format (#.#, #.#). On the other hand, the main program is written in the file ex1.cpp.
   1. Download the source code [provided here](https://drive.google.com/file/d/1agK41z3QD88iHXKD-Y5MGjKeNTNmeGeK/view?usp=sharing) and unzip the file;
   2. Analyze carefully each file and relate its contents;
   3. Try to compile each .cpp file individually;
   4. Try to link all object files for generating the executable file and run the program;
   5. Delete all object files and the executable program. Analyze carefully the contents of the makefile. Try to use make for compiling and generating the executable file again. Try to make just one specific target;
   6. Delete again all object files and the executable program. Analyze carefully the contents of the file CMakeLists.txt. Try to use cmake and make for compiling and generating the executable file again.

1. Complete the development of the module from the previous exercise, by implementing the following functionalities:
   1. A function that asks the user for a point to be introduced through the keyboard;
   2. A function that computes the distance between two points.

To test the module, write a program that asks the user for points until the point (0, 0) is entered. The program should count how many points were entered and calculate the sum of their distances to the origin, i.e., to the point (0, 0). The program should also identify which point is furthest from the origin.

The interaction of the program will be as follows (try to respect the formatting of the example below):

Enter a point: 1 0

Enter a point: 0 -4

Enter a point: -2 0

Enter a point: 0 3

Enter a point: 0 0

Number of points entered: 4

Sum of distances to the origin: 10.0

Furthest point from the origin: (0.0, -4.0)

1. Assume that you want to generate a table of multiples of any given number. Write a program that allows the user to enter the number and then generates the table, formatting it into 10 columns and 20 lines. Interaction with the program should look like this (only the first three lines are shown):

Enter a number: 7

7 14 21 28 35 42 49 56 63 70

77 84 91 98 105 112 119 126 133 140

147 154 161 168 175 182 189 196 203 210

1. Construct a program that displays a pyramid of Xs on the screen. The pyramid should look like the following (the number of lines should be an input by the user):

X

XXX

XXXXX

XXXXXXX

XXXXXXXXX