C++ Lab3 (2 classes): Loops

- *The source code needed for this lab is available on Moodle.*
- *More (English) programming vocabulary is given at the end of this lab.*

Objectives

1. Become familiar with the syntax of the different loops.
2. Understand when to use each one of the loops.
3. Discover classic uses of loops.

General instructions

1. Read the questions completely before coding.
2. Prepare a test set before coding.
3. Write or modify a program based on examples from the course and past labs.
4. Test using the test set. Your tests must be commented at the end of your files.

WORK TO HAND IN: Exercise 2 after the first session. Exercise 5 after the second session. ALWAYS include the code and test cases.

Exercise 1: Divisors

Write a program that inputs a strictly positive integer ***n*** (no need to check it) and that displays the list of its divisors. Hint: we use a **for** loop to iterate through each number ***i*** between **2** and ***n-1*** and we test if ***i*** is a divisor of ***n***. Reminder: ***i*** is a divisor of ***n*** if **n%i** is equal to 0). Test your program.

Bu adding a variable, modify your program so that it also displays at the end the number of divisors of ***n*** (do not count 1 and n). Test.

Exercise 2: A little drawing

a) Write a program to input a number ***nb*** and display ***nb*** stars.
Sample execution:
   - Display: How many stars?
   - Input: 7
   - Display: *******

b) Add the necessary code to check that ***nb*** is strictly positive and ask the user for a new number if it is not.
Sample execution:
   - Display: How many stars?
   - Input: -2
   - Display: Error, your number must be strictly positive. Try again:
   - Input: 4
   - Display: ****

c) Modify your program to input a number ***nbL*** and display a rectangle of ***nbL*** lines of ***nb*** stars. Check that ***nbL*** is strictly positive and ask again in case of an error.
Sample execution:
   - Display: How many stars?
   - Input: 7
   - Display: How many lines?
   - Input: 4

- Display:
```
*******
*******
*******
*******
```

d) Modify your program to display a rectangle with a character given by the user: input a character ***myChar***, and display ***nbL*** lines of ***nb*** characters ***myChar***.

e) **Optional** (for those who are fast): Write a program that inputs a number ***n*** and displays a triangle with base ***n***. Check that ***n*** is odd (display an error message if it is not). Add your tests as comments at the end of the program.
Example with n = 5
```
    *
  * * *
* * * * *
```
Hint: declare 3 variables: one for the number of lines (depends on the base: n/2 + 1), one for the number of spaces at the beginning of a line, and another one for the number of characters to display on the current line.

➔ Upload your code for question e or d, including test cases.

Exercise 3: Correct the errors

A student wrote the following program to compute the average of his grades (sum of grades / number of grades). He's having trouble because:

- in some cases, the program falls into an infinite loop,
- the average is not always correct.

To help him correct his program,

a) copy the error.cpp file from Moodle,
b) test the program,
c) find when the infinite error happens and correct the error(s),
d) find when the results are incorrect and the correct the error(s),
e) add comments for the student to clearly identify the different parts of the code.

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

int main()
{
    float grade, sumGrades, nbGrades, average;
    char rep;

    do
    {
        sumGrades = 0;
        nbGrades = 0;
        cout << "Input a grade: " << endl;
        cin >> grade;
        while (grade < 0 || grade > 20)
        {
            cout << "Error: grade should range between 0 and 20." << endl;
            cout << "Input again: " << endl;
        }
        sumGrades = sumGrades + grade;
        nbGrades = nbGrades + 1;
        cout << "More grades? (answer with y or n)" << endl;
        cin >> rep;
    }
    while (rep == 'y');

    average = sumGrades / nbGrades;
    cout << "The average is " << average << "." << endl;

    return 0;
}
```

Exercise 4: Finding divisors within an interval

a) Write a program that asks the user to input a strictly positive number ***max***, then displays all integers divisible by 7 that are smaller than ***max***.

b) Modify your program to also ask for a strictly positive number ***min*** (check that ***min*** < ***max***) and then display only the integers divisible by 7 that range between ***min*** and ***max,*** bounds included.

c) Make the display clearer by displaying 10 values per line.

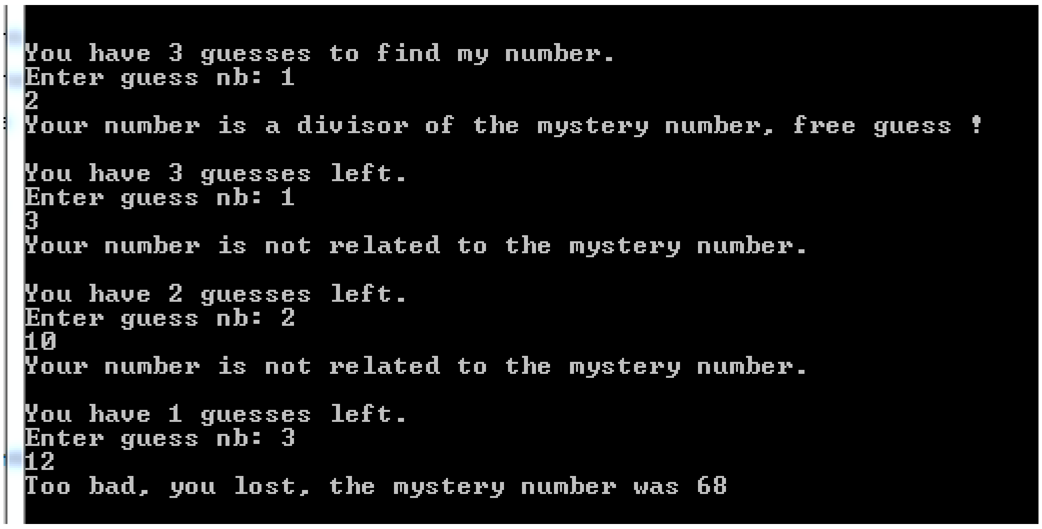
Exercise 5: Guess my number *** This program has to be uploaded to Moodle.

Write a program that asks the user to guess a randomly generated number ranging between 1 and 100. At each guess, the user gets one the following hints:

- The given number is a multiple of the number to be guessed.
- The given number is a divisor of the number to be guessed.
- The given number is neither a multiple nor a divisor of the number to be guessed.

Modify your program to allow 10 guesses maximum. Guesses falling into cases a) and b) above do not count! The program displays the number of each guess and the number of remaining guesses. At the end of the program, either congratulate the user, or give the number that they failed to guess.

Here is the output of a possible run (information about random number generation on next page)



```
You have 3 guesses to find my number.
Enter guess nb: 1
2
Your number is a divisor of the mystery number, free guess !

You have 3 guesses left.
Enter guess nb: 1
3
Your number is not related to the mystery number.

You have 2 guesses left.
Enter guess nb: 2
10
Your number is not related to the mystery number.

You have 1 guesses left.
Enter guess nb: 3
12
Too bad, you lost, the mystery number was 68
```

To generate a random number between 1 and 100 in C++ use the following statements:

```
    #include <cstdlib>    // for srand and rand
    #include <ctime>      // for time
    srand(time(NULL)); // initialize generator using the current time
    alea = rand()%100 + 1; // random number between 1 and 100
```

For more information, if you are interested, check here:
http://www.cplusplus.com/reference/cstdlib/rand/

➔ Upload your code (with tests, added as comments after the program) to Moodle.
Keep your program, it will be used again in a subsequent lab dealing with subprograms.

Exercise 6 (optional): Counting the number of digits in a number

Given a positive integer ***x***, write a program that counts the number of digits that ***x*** is made up of.
Hint: divide ***x*** by 10 as long as you don't get 0 …
Test your program with this set of test cases:

	Input data	Purpose of the test	Expected result
– test 1:	12345	*general case*	5
– test 2:	5	*one digit*	1
– test 3:	100	*multiple of 10*	3
– test 4:	0	*0*	1

Exercise 7 (optional): Prime numbers

Write a program that determines if an integer ***p*** larger than 2, input by the user, is prime.

a) To do so, divide ***p*** by all integers that are smaller than ***p***. If no division has a null remainder, than ***p*** is prime.
b) Limit the number of divisions that are necessary by checking if ***p*** is even. If it is, it is not prime. If it is not, it is not necessary to divide it by multiples of 2. Divisions can be stopped when the divisor is larger than half of p.

Exercise 8 (optional): GCD

The greatest common divisor (gcd) of two or more integers (at least one of which is not zero), is the largest positive integer that divides the numbers without a remainder. For example, the GCD of 8 and 12 is 4.

Write a program that finds the GCD of two integers using Euclid's algorithm:
- Do an integer division of the largest number by the smallest.
- Replace the largest number by the smallest and the smallest by the remainder of the integer division.
- Continue until the remainder is null.
- The GCD is the last non-null remainder.