

## Lab 1: Numbers and Programming

**Goal:** To gain familiarity with the concepts of

- ▶ basic C++ program structure;
- ▶ input and output,
- ▶ *Flow-of-control*: `if` statements, and `for loops`
- ▶ Computer representation of number – particularly `ints`.

You need to upload something that addresses Q4, even if not complete, by 17:00, Friday 30 January to <https://universityofgalway.instructure.com/courses/44564/assignments/139657>

Lab 2 will develop this further, and will carry more marks.

## Question 1

---

- Q1. (`if/else if/else`-statements) Write a C++ program that prompts the user to enter two integers,  $x$ ,  $y$ , and then reports which quadrant the point  $(x, y)$  is found, or if  $(x, y)$  is on an axis (i.e., one or both are zero).

*Tip:* see [`https://en.wikipedia.org/wiki/Quadrant\_\(plane\_geometry\)`](https://en.wikipedia.org/wiki/Quadrant_(plane_geometry)) for a definition of quadrants I, II, III and IV.

## Question II

Q2. (`cin` and `while`). Write a short C++ program that works as follows.

- ▶ The user is prompted for a number between 1 and 10 (inclusive), storing the input in an `integer` variable, `n`.
- ▶ A `while` loop is used so that, if `n` is not in that range, the user is prompted for it again and again, until they enter one in the correct range.
- ▶ The final value of `n` is displayed, along with a suitable message, that includes the number of attempts taken.

## Question III

- Q3. The following code snippet finds the largest int that is correctly representable by your computer. It also computes the time taken. (Full code at [Lab1-Q3.cpp](#)).

```
18    clock_t start;
19    float diff, diff_seconds;
20    start=clock();
21    int i=1;
22    int j=i+1;
23    while ( i<j )
24    {
25        i++;
26        j=i+1;
27    }
28    diff = (float)(clock()-start);
29    diff_seconds = diff/CLOCKS_PER_SEC;
30    std::cout << "Overflow at i=" << i << std::endl;
31    std::cout << "Computation took " << diff_seconds
32        << " seconds." << std::endl;
```

Read the code carefully, and make sure you understand it. Test it, making sure you compile **without** any optimisations. Do the results agree with the theory covered in class?

## Question IV

Q4. There are other types of integers available in C++, for example,

- ▶ `short int`, which is stored in 2 bytes;
- ▶ `unsigned int`, which is stored in 4 bytes;
- ▶ and `unsigned short int`, which is stored in 2 bytes.

Pick any of these, and modify the code in *Lab1-Q3.cpp* to find the largest possible value it can store. Add a comment to explain if this agrees with theory.

## Extras

---

Note: You can check the number of bytes that a datatype uses, with the `sizeof()` function. Given a variable, or the name of a type, it returns the number of bytes used to store it. You can use this to verify that `short int`, `int` and `long int` use 2, 4 and 8 bytes, respectively.

---

**Optional Extra:** As mentioned, the data type called `long int` uses 8 bytes. Suppose you wanted to use this program to test the largest `long int` your C++ programs can represent. Estimate how long your program would take to run. **Warning: don't actually try this by running the code!!.**

---

In Lab 2, we'll write some code that estimates the smallest and largest `floats` and `doubles` that one can store, and the machine epsilon for these types.

If you'd like to get started early, think about how that can be done.