Lab 1 supplementary notes

C++ is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language that supports procedural, object-oriented, and generic programming.

C++ is regarded as a **middle-level** language, as it comprises a combination of both high-level and low-level language features.

C++ was developed by Bjarne Stroustrup starting in 1979 at Bell Labs in Murray Hill, New Jersey, as an enhancement to the C language and originally named C with Classes but later it was renamed C++ in 1983.

C++ is a superset of C, and that virtually any legal C program is a legal C++ program.

Note: A programming language is said to use static typing when type checking is performed during compile-time as opposed to run-time.

Learning C++

The most important thing while learning C++ is to focus on concepts.The purpose of learning a programming language is to become a better programmer; that is, to become more effective at designing and implementing new systems and at maintaining old ones.

Use of C++

C++ is used by hundreds of thousands of programmers in essentially every application domain.
C++ is being highly used to write device drivers and other software that rely on direct manipulation of hardware under real-time constraints.

C++ is widely used for teaching and research because it is clean enough for successful teaching of basic concepts.

Text Editor/IDE:

The files you create with your editor are called source files and for C++ they typically are named with the extension .cpp, .cp, or .c.

C++ Compiler:

This is an actual C++ compiler, which will be used to compile your source code into final executable program.

Most C++ compilers don't care what extension you give to your source code, but if you don't specify otherwise, many will use .cpp by default.

C++ Program Structure:

```
// my first program in C++

#include <iostream>
using namespace std;

int main ()
{
  cout << "Hello World!";
  return 0;
}
```

```
Hello World!
```

#include <iostream>

Lines beginning with a hash sign (#) are directives for the preprocessor. They are not regular code lines with expressions but indications for the compiler's preprocessor. In this case the directive #include <iostream> tells the preprocessor to include the iostream standard file. This specific file (iostream) includes the declarations of the basic standard input-output library in C++, and it is included because its functionality is going to be used later in the program.

using namespace std;

All the elements of the standard C++ library are declared within what is called a namespace, the namespace with the name *std*. So in order to access its functionality we declare with this expression that we will be using these entities. This line is very frequent in C++ programs that use the standard library, and in fact it will be included in most of the source codes included in these tutorials.

int main ()

This line corresponds to the beginning of the definition of the main function. The main function is the point by where all C++ programs start their execution, independently of its location within the source code. It does not matter whether there are other functions with other names defined before or after it – the instructions contained within this function's definition will always be the first ones to be executed in any C++ program. For that same reason, it is essential that all C++ programs have a main function.

The word main is followed in the code by a pair of parentheses (()). That is because it is a function declaration: In C++, what differentiates a function declaration from other types of expressions are these parentheses that follow its name. Optionally, these parentheses may enclose a list of parameters within them.

Right after these parentheses we can find the body of the main function enclosed in braces ({}). What is contained within these braces is what the function does when it is executed.

cout << "Hello World!";

This line is a C++ statement. A statement is a simple or compound expression that can actually produce some effect. In fact, this statement performs the only action that generates a visible effect in our first program. cout represents the standard output stream in C++, and the meaning of the entire statement is to insert a sequence of characters (in this case the Hello World sequence of characters) into the standard output stream (which usually is the screen).

cout is declared in the iostream standard file within the std namespace, so that's why we needed to include that specific file and to declare that we were going to use this specific namespace earlier in our code.

Notice that the statement ends with a semicolon character (;). This character is used to mark the end of the statement and in fact it must be included at the end of all expression statements in all C++ programs (one of the most common syntax errors is indeed to forget to include some semicolon after a statement).

return 0;

The return statement causes the main function to finish. return may be followed by a return code (in our example is followed by the return code 0). A return code of 0 for the main function is generally interpreted as the program worked as expected without any errors during its execution. This is the most usual way to end a C++ console program.

Comments in C++

Program comments are explanatory statements that you can include in the C++ code. These comments help anyone reading the source code. All programming languages allow for some form of comments. C++ supports single-line and multi-line comments. All characters available inside any comment are ignored by C++ compiler.C++ comments start with /* and end with */. For example:

```
/* This is a comment */

/* C++ comments can also
 * span multiple lines
 */
```

A comment can also start with //, extending to the end of the line. For example:

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

main()
{
   cout << "Hello World"; // prints Hello World

   return 0;
}
```

When the above code is compiled, it will ignore **// prints Hello World** and final executable will produce the following result:

```
Hello World
```

Within a /* and */ comment, // characters have no special meaning. Within a // comment, /* and */ have no special meaning. Thus, you can "nest" one kind of comment within the other kind. For example:

```
/* Comment out printing of Hello World:
```

```
cout << "Hello World"; // prints Hello World

*/
```

How to run a C++ program using Dev C++?

- ✓ Open dev c++

- ✓ Create an new c++ source file
- ✓ Write the c++ code, save it.
- ✓ Verify that the code is correct by visual inspection
- ✓ Compile your code by clicking on the compile toolbar (shortcut F9)
- ✓ If there is a syntax error, debug it. Save it
- ✓ Compile again
- ✓ If errors, debug again.
- ✓ If no errors then execute/run the program. (shortcut F10)