**Learning C++**

C++ is a programming language created by Bjarne Stroupstup and his team at bell laboratories in 1979. 40 years later, it is one of the most widely used languages in the world: we can find c++ applications everywhere, from the bottom of the oceans to the surface of mars. As the name implies, C++ was derived from the C language; Bjarne’s goal was to add object-oriented programming into C, a language well respected for its portability and low level functionality.

Why? – it is fast and flexible, well supported and it forces you to think in creative ways.

Aims – lesson 1: learn the basic concepts, and you’ll write your very first C++ program.

C++ programs are stored in files which usually have the file extension .cpp, which simply stands for c plus plus. The code inside our c++ file is a classic first step all new programmers take- they are greet the world through the terminal.

Graphical user interface, application

Description automatically generated

C++ runs from line to line from top to bottom.

Structure:

Include libraries - #include <iostream>

The main () function - int main()

beginning of function - {

what the program does

end of function - }

example:

* Std::cout is the ‘character output stream’. It is pronounced “see-out”
* << is an operator that comes right after it.
* “hello world!/n” is what’s being outputted here. You need to double quotes around text. The \n is a special character that indicates a new line
* ; is a punctuation that tells the computer that you are at the end of a statement. It is similar to a period in a sentence.
* Graphical user interface, application

  Description automatically generated
* Text

  Description automatically generated

Lesson 2:

C+ is a compiled language. That means that to get a c++ program to run, you must first translate it from a human-readable form to something a machine can understand. That translation is done by a program called a compiler. When you program in c++, you mainly go through 4 phases during development.

1. Code – writing the program
2. Save – saving the program
3. Compile – compiling via the terminal
4. Execute – executing via the terminal

And repeat debug the errors if needed

Compile: a computer can only understand machine code. A compiler can translate the c++ programs that we write into machine code. You call on the compiler by using other terminal, which is the black panel to the right of the code editor that contains dollar sign. To compile a file, you need to type g++ followed by the file name in the terminal and press enter.

G++ hello.cp

The compiler will then translate the c++ program hello.cpp and create a machine code file called a.out.

Execute: to execute the new machine code file, all you need to do is type ./ and the machine code file name in the terminal and press enter. In this case our compiled file name is a.out. putting it all together, we end up with the following : a/a.out.

The executable file will then be loaded to computer memory and the computers cpu executes the program one instruction at a time.

Compile: sometimes when we compile, we want to give the output executable file a specific name. to do so, the compile command is slightly different. We still need to write g++ and the file name in the terminal. After that, there should be -o and then the name that you want to give to the executable file: g++ hello.cpp – o hello

The compiler will then translate the c++ program hello.cpp and create a machine code file called hello.

Execture: to execute the new machine code file, all you need to do is type ./ and the machine code file name in the terminal. ./hello

The executable file will then be loaded to computer memory and the computers cpu will execute the program one instruction at a time.

Programming is often highly collaborative. In addition, our own code can quickly become difficult to understand when we return to it- sometimes only a few hours later. For these reasons, its often useful to leave notes in our code for ourselves or other developers. As we write a c++ program, we can write comments in the code that the compiler will ignore as our programs run. These comments exist for human readers.

Comments can explain what the code is doing, leave instructions for developers using the code, or add any other useful annotations.

There are two types of code comments in c++:

* A single line comment will comment out a single line and is denoted with two forward slashes // preceding it:
* ‘ // prints “hi!” to the terminal

Std::cout << “hi”;

* You caN ALSO USE A SINGLE LINE COMMENT AFTER A LINE OF CODE:

‘ std::cout << “hi!”; // prints “hi!”

* A multi-line comment will comment out multiple lines and is denoted with /\* to begin the comment, and \*/ to end the comment:

‘/\*this is all commented

Std::cout << “hi!”;

None of this is going to run! \*/

Introduction to variables

The hello world program simply writes to the screen. It does not read anything, calculate anything or allow for user input. Real programs tend to produce results based on some input that the user of the program gives, rather than just outputting the same thing every time.

To read something from the keyboard, we first need somewhere in the computers memory to store date. That’s where variables come in.

A variable is simply a name that represents a particular piece of your computers memory that has been set aside for you to store, retrieve and use date

* Int: integer numbers
* Double: floating point numbers
* Char: individual characters
* String: a sequence of characters
* Bool: true/false values

Every variable has a type, which represents the kind of information you can store inside of it. It tells your compiler how much memory to set aside for the variable and defines what you can do with the variable.

Step 1: declare a variable!

You must declare a variable before you can use it.

Suppose we are building a game and we want to keep track of a players score that goes from 0 – 10 . we need a variable

Before we can use a variable, we must declare, or create it. To declare a variable, we need to provide two things:

* A type for the variable
* A name for the variable

So to declare an integer variable called score, we need to write ‘ int score; ‘

* The int is the type of variable
* The score is the name of the variable
* The ; is how we end a statement

In c++, variable names consist only of upper/lower case letters, digits, and/or underscores.

Note: c++ is known as a strongly typed language. If you try not to give an integer value a decimal number, you are going to get unexpected results, warnings, or errors.

Step 2: initialize a variable

After we declare a variable, we can give it a value.

Suppose that we have a declared an int variable called score, to set it to 0, we can simply write: score = o;

* The score is the name of the variable
* The = indicates assignment
* The 0 is the value you want to store inside the variable

Note: in c++ , a single equal sign does not really mean equal. It means assign. In the code above, we are assigning the score variable a value of 0.