C++ is a compiled language  
humans can read C++ but computers can’t read it so we need something that can convert the C++ code to the code that can computer understand, and that thing called compiler and the operation of transforming the code is called compilation.

C++ is also known with linking:

Linking is getting a bunch o compiled code to one executable file, and that the executable file is what the user get and use.

C++ introduced 2 concepts:

1. Object Oriented Programming (OOP).
   1. Is a way to structure our code in things called classes.
2. Generic Programming.
   1. With Generic Programming we able to use or create structure that work well with different types of data.
   2. It looks like and array or array list we call it in C++ “Vectors” and also you can call it container.

C and C++ are two separated Programming Languages.

C++ came after C and almost everything in C is in C++.

You can think in C++ like a superset of C.

How to start writing C++ on windows.

You can follow this video to full setup of c++ using MinGW and configure it on VS Code <https://www.youtube.com/watch?v=DIw02CaEusY>

C++ Files extensions:

1. .c
2. .cc
3. .cpp
4. .c++
5. .cp
6. .cxx

Note the .C - case matters in GCC, .c is a C file whereas .C is a C++ file (if you let the compiler decide what it is compiling that is).

How to compile c++ file:

1. Open terminal.
2. Write g++ path/file.ext  
   it will generate new file commonly called “a.out/a.exe”

How to run the app

1. Open terminal
2. Write the file name with the ext for the extract file

How to write function:

1. The type of the output of the function
2. Function name
3. Function body.
4. Function may be has inputs and must return something that thing is called output
5. The input is called parameters.
6. The output is called with return keyword.
7. In the function body we write statements.

You can think with the function as an machine that does specific job.

Example:

#include <iostream>

int main(){

    std::cout << "Hello World\n";

    return 0;

}

In the example before std::cout we use it to write in the console.

Std::cout be available only when include it in C++ program when including iostream

Cout is defined in iostream

Notes:

1. Each statements must ends with semicolon “;”
2. Std is an acronym to standard
3. Iostream is acronym to input output stream

Std:: is called a namespace prefix:

namespace is a set of symbols that are used to organize objects of various kinds, so that these objects may be referred to by name. A namespace ensures that all the identifiers within it have unique names so that they can be easily identified.

Namespaces were introduced into C++ to resolve identifier name conflicts.

This ensured that two objects can have the same name and yet be treated differently if they belonged to different namespaces.

You can also write it like:

#include <iostream>

int main(){

    using namespace std;

    cout << "Hello World\n";

    return 0;

}

But it’s not the best practice.

The technical term of that is using directive.

When we import a namespace we are essentially pulling all type definitions into the current scope.

The std namespace is huge. It has hundreds of predefined identifiers, so it is possible that a developer may overlook the fact there is another definition of their intended object in the std library. Unaware of this they may proceed to specify their own implementation and expect it to be used in later parts of the program. Thus there would exist two definitions for the same type in the current namespace. This is not allowed in C++, and even if the program compiles there is no way of knowing which definition is being used where.

The solution to the problem is to explicitly specify to which namespace our identifier belongs to using the scope operator (::).

Source: <https://www.geeksforgeeks.org/using-namespace-std-considered-bad-practice/>

There is another way to use also namespace and it called using decleration

You can write it like:

#include <iostream>

int main(){

    using std::cout;

    cout << "Hello World\n";

    return 0;

}

The benefit here we use it in cout only.

Cout:

* Cout = console out
* Cout is an example for an object
* Cout is describing the console output in other words it’s a tool to reach the console
* Cout is an instance from ostream “output stream”.