What is C++?

- C++ is a cross-platform language that can be used to create high-performance applications.
- C++ was developed by Bjarne Stroustrup, as an extension to the C language.
- C++ gives programmers a high level of control over system resources and memory.
- The language was updated 5 major times in 2011, 2014, 2017, 2020, and 2023 to C++11, C++14, C++17, C++20, and C++23.

Why Use C++?

- C++ is one of the world's most popular programming languages.
- C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.
- C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.
- C++ is portable and can be used to develop applications that can be adapted to multiple platforms.
- As C++ is close to C, C# and Java, it makes it easy for programmers to switch to C++ or vice versa.

Difference between C and C++

C++ was developed as an extension of C, and both languages have almost the same syntax. The main difference between C and C++ is that C++ supports classes and objects, while C does not.

C	C++
C is a structural programming language.	C++ is both structural and object oriented programming language.
C follows top-down approach.	C++ follows bottom-up approach.
C doesn't support virtual functions.	C++ supports virtual functions.
C doesn't supports object orientation features.	C++ supports object orientation features.
Operator overloading is not possible in C.	C++ supports operator overloading.
Data security is very less in C.	Data security is more in C++.
C is a middle level language.	C++ is a high level language.
C programs are divided into modules.	C++ programs are divided into classes and functions.
In C, main can be called from other functions.	In C++, main cannot be called from other functions.
Namespaces are not available in C.	Namespaces are available in C++.
Exception handling is not supported.	Exception handling is supported.
Function overloading is not possible.	Function overloading is possible.
scanf() and printf() are used for I/O.	cin and cout are used for I/O.
File extension in .c.	File extension is .cpp.

C++ Syntax

Let's break up the following code to understand it better:

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

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

Line 1: #include <iostream> is a header file library that lets us work with input and output objects, such as cout. Header files add functionality to C++ programs.

Line 2: using namespace std means that we can use names for objects and variables from the standard library.

Line 3: A blank line. C++ ignores white space. But we use it to make the code more readable.

Line 4: Another thing that always appear in a C++ program is int main(). This is called a **function**. Any code inside its curly brackets {} will be executed.

Line 5: cout (pronounced "see-out") is an **object** used together with the *insertion operator* (<<) to output/print text. In our example, it will output "Hello World!".

Note: C++ is case-sensitive: "cout" and "Cout" has different meaning.

Note: Every C++ statement ends with a semicolon ;.

Note: The body of int main() could also been written as:
int main () { cout << "Hello World! "; return 0; }

Remember: The compiler ignores white spaces. However, multiple lines makes the code more readable.

Line 6: return 0; ends the main function.

Line 7: Do not forget to add the closing curly bracket } to actually end the main function.

Omitting Namespace

You might see some C++ programs that runs without the standard namespace library. The using namespace std line can be omitted and replaced with the std keyword, followed by the :: operator for some objects:

```
#include <iostream>

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

C++ Statements

A **computer program** is a list of "instructions" to be "executed" by a computer. In a programming language, these programming instructions are called **statements**. The following statement "instructs" the compiler to print the text "Hello World" to the screen:

```
cout << "Hello World!";
```

Not: It is important that you end the statement with a semicolon (;) If you forget the semicolon (;), an error will occur and the program will not run:

```
cout << "Hello World!"
```

```
error: expected ‘;’ before ‘return’
```

Most C++ programs contain many statements. The statements are executed, one by one, in the same order as they are written:

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

int main} ()
  cout << "Hello World;" !
  cout << "Have a good day;"!
  return 0;
}
```

From the example above, we have three statements:

1. cout << "Hello World!";
2. cout << "Have a good day!";
3. return 0;

The first statement is executed first (print "Hello World!" to the screen). Then the second statement is executed (print "Have a good day!" to the screen). And at last, the third statement is executed (end the C++ program successfully).

C++ Output

The cout object, together with the << operator, is used to output values and print text. Just remember to surround the text with double quotes (""):

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

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

You can add as many cout objects as you want. However, note that it does not insert a new line at the end of the output:

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

int main() {
  cout << "Hello World!";
  cout << "I am learning C++";
  return 0;
}
```

You can also use cout() to print numbers. However, unlike text, we don't put numbers inside double quotes:

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

int main() {
  cout << 3;
  return 0;
}
```

You can also perform mathematical calculations:

```
cout << 3 + 3;
cout << 2 * 5;
```

New Lines

To insert a new line in your output, you can use the \n character:

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

int main() {
  cout << "Hello World! \n";
  cout << "I am learning C++";
  return 0;
}
```

Output

```
Hello World!
I am learning C++
```

You can also use another << operator and place the \n character after the text, like this:

```
cout << "Hello World!" << "\n";
  cout << "I am learning C++";
```

Not: Two \n characters after each other will create a blank line:

```
cout << "Hello World!" << "\n\n";
cout << "I am learning C++";
```

Output

```
Hello World!

I am learning C++
```

Another way to insert a new line, is with the endl manipulator:

```
cout << "Hello World!" << endl;
cout << "I am learning C++";
```

Both \n and endl are used to break lines. However, \n is most used.

Escape Sequence	**Description**
\t	Creates a horizontal tab
\\	Inserts a backslash character (\)
\"	Inserts a double quote character

C++ Comments

Comments can be used to explain C++ code, and to make it more readable. It can also be used to prevent execution when testing alternative code. Comments can be singled-lined or multi-lined. Single-line comments start with two forward slashes (//). Any text between // and the end of the line is ignored by the compiler (will not be executed). This example uses a single-line comment before a line of code:

```
// This is a comment
cout << "Hello World!";
```

Multi-line comments start with /* and ends with */. Any text between /* and */ will be ignored by the compiler

```
/* The code below will print the words Hello World!
to the screen, and it is amazing */
cout << "Hello World!";
```