Topic2(Program Structure)

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;
}
```

Example Explained

> **Line 1 :** `#include <iostream>` is a **header file library** that lets us work with input and output objects, such as `cout` (used in line 5). 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 :** 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++ Output (Print Text)

> The `cout` object, together with the `<<` operator, is used to output values/print text:

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

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

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

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

New Lines

> To insert a new line, you can use the `\n` character:

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

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

> **Tip :** Two `\n` characters after each other will create a blank line:

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

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

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

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

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

Escape Sequence

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

> 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 is a Comment
cout << "Hello World!";
```

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

Multi-line Comments

> Multi-line comments start with `/*` and end with `*/`

> Any text between `/*` and `*/` will be ignored by the complier:

```
/*
This is a multi-line comment.
It can span several lines
*/
cout << "Hello World!";
```