Structure of a C++ Program

- The basic structure of a C++ Program:
  - Basic Components
  - Preprocessor Directives
  - The main function
  - Namespaces
  - Comments
  - Basic I/O
- Components of a C++ Program:
  - Keywords
  - Identifiers
  - Operators
  - Punctuation
  - Syntax

KEYWORDS:

- Have a special meaning in C++.
- Are reserved by the C++ language.

IDENTIFIERS:

- Programmer-defined names.
- Not a part of the C++ language.
- Used to name variables, functions, etc.

OPERATORS:

- Arithmetic operators, assignment, <<, >>
- Are reserved by the C++ language.

PUNCTUATION: Special characters that seperate, terminate items.

SYNTAX:

- How the programming elements are put together to form a program.
- Programming languages have rules.

COMMENTS:

- Ignored by the compiler.
- Used to explain your code.
- Two styles:

```
// Single Line
/* Multi
line */
```

FUNCTIONS:

- main() is a required function in C++.
- Break up your code into units of functionality.
- Can optionally receive and return information.

PREPROCESSOR DIRECTIVES:

- The C++ preprocessor is a program that processes your source code before the compiler sees it.
- It first strips all the comments from the source file and replaces each comment with a single space. Then it looks for preprocessor directives and executes them.
- Pre processor directives are lines that begin with "#".
- When the preprocessor sees this directive, it replaces the pound include line with the file that it's referring to that it recursively processes the file as well.

THE main() function :

- Every C++ program must have exactly 1 main() function.
- Starting point of program execution.
- Return 0 indicates successful program execution.
- 2 versions that are both valid.

```
int main()
{
    // Code
    return 0;
}
```

program.exe

```
int main(int argc, char* argv[])
{
    // Code
    return 0;
}
```

program.exe argument1 argument2

Namespaces :

As C++ programs become more and more complex, our programs become a combination of our own code, C++ standard library code and libraries from third party developers and their code. This leads to naming conflicts. We use namespaces to solve these naming conflicts.

Eg) `Using namespace std;`

OR

```
using std::cout ;
using std::cin ;
using std::endl ;
```

We are not getting any other names from the standard namespace.

BASIC I/O :

cout, cin, cerr and clog are objects representing streams.

cout
- Standard output stream
- Console

cin
- Standard input stream
- Keyboard

<<
- Insertion operator
- Output streams

>>
- Extraction operator
- Input streams

cout and <<

- Insert data into cout stream

```
cout << data;
```

- Can be chained

```
cout << "Data 1 is " << data1;
```

- Does not automatically add line breaks

```
cout << "Data 1 is " << data1 << endl;
cout << "Data 1 is " << data1 << "\n";
```

cin and >>

- Extract data from the cin stream based on data's type

```
cin >> data;
```

- Can be chained

```
cin >> data1 >> data2;
```

- Can fail if entered value cannot be interpreted. Data could have undetermined value.