Basic Structure & Syntax

Programming in C++ involves following a basic structure throughout. To understand that basic structure, the first program we learned writing in C++ will be referred to.

```
#include <iostream>


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

Copy

Here's what it can be broken down to.

Pre-processor commands/ Header files

It is common for C++ programs to include many built-in elements from the standard C++ library, including classes, keywords, constants, operators, etc. It is necessary to include an appropriate header file in a program in order to use such pre-defined elements.
In the above program, #include <iostream> was the line put to include the header file iostream. The iostream library helps us to get input data and show output data. The iostream library also has many more uses and error facilities; it is not only limited to input and output.
Header file are both system defined and user defined. To know more about header files, go to the documentary here, https://en.cppreference.com/w/cpp/header.

Definition Section

Here, all the variables, or other user-defined data types are declared. These variables are used throughout the program and all the functions.

Function Declaration

- After the definition of all the other entities, here we declare all the functions a program needs. These are generally user-defined.
- Every program contains one main parent function which tells the compiler where to start the execution of the program.

- All the statements that are to be executed are written in the main function.
- Only the instructions enclosed in curly braces {} are considered for execution by the compiler.
- After all instructions in the main function have been executed, control leaves the main function and the program ends.

A C++ program is made up of different tokens combined. These tokens include:

- Keywords
- Identifiers
- Constants
- String Literal
- Symbols & Operators

1. Keywords

Keywords are reserved words that can not be used elsewhere in the program for naming a variable or a function. They have a specific function or task and they are solely used for that. Their functionalities are pre-defined.
One such example of a keyword could be return which is used to build return statements for functions. Other examples are auto, if, default, etc. There is a list of reserved keywords which cannot be reused by the programmer or overloaded. One can find the list here, https://en.cppreference.com/w/cpp/keyword.

2. Identifiers

Identifiers are names given to variables or functions to differentiate them from one another. Their definitions are solely based on our choice but there are a few rules that we have to follow while naming identifiers. One such rule says that the name can not contain special symbols such as @, -, *, <, etc.
C++ is a case-sensitive language so an identifier containing a capital letter and another one containing a small letter in the same place will be different. For example, the three words: Code, code, and cOde can be used as three different identifiers.

3. Constants

Constants are very similar to a variable and they can also be of any data type. The only difference between a constant and a variable is that a constant's value never changes. We will see constants in more detail in the upcoming tutorials.

4. String Literal

String literals or string constants are a sequence of characters enclosed in double quotation marks. Escape sequences are also string literals.

5. Symbols and Operators

Symbols are special characters reserved to perform certain actions. Using them lets the compiler know what specific tasks should be performed on the given data. Several examples of symbols are arithmetical operators such as +, *, or bitwise operators such as ^, &.