When we consider a C++ program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what a class, object, method, and instance variable mean

Object – Objects have states and behaviors. Example: A dog has states (color, name, breed) as well as behaviors (wagging, barking, eating). An object is an instance of a class.
Class – A class can be defined as a template/blueprint that describes the behaviors or states that object of its type supports.
Methods – A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated, and all the actions are executed.
Instance Variables – Each object has its own unique set of instance variables. An object's state is created by the values assigned to these instance variables.

C++ Program Structure

The basic structure of a C++ program consists of the following parts:

Header file inclusion section: This is the section where we include all required header files whose functions we are going to use in the program.
Namespace section: This is the section where we use the namespace.
The main() section: In this section, we write our main code. The main() function is an entry point of any C++ programming code from where the program's execution starts.

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

// main() is where program execution begins.
int main() {
   cout << "Hello World"; // prints Hello World
   return 0;
}
```

- The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, the header **<iostream>** is needed.
-
- The line **using namespace std;** tells the compiler to use the std namespace. Namespaces are a relatively recent addition to C++.
-
- The next line '**// main() is where program execution begins.**' is a single-line comment available in C++. Single-line comments begin with // and stop at the end of the line.
-
- The line **int main()** is the main function where program execution begins.
-
- The next line **cout << "Hello World";** causes the message "Hello World" to be displayed on the screen.
-
- The next line **return 0;** terminates main() function and causes it to return the value 0 to the calling process.

Semicolons and Blocks in C++

In C++, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.

```
x = y;
y = y + 1;
add(x, y);
```

```
{
   cout << "Hello World"; // prints Hello World
   return 0;
}
```

C++ does not recognize the end of the line as a terminator. For this reason, it does not matter where you put a statement in a line.

C++ Identifiers

A C++ identifier is a name used to identify a variable, function, class, module, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).

C++ does not allow punctuation characters such as @, $, and % within identifiers. C++ is a case-sensitive programming language. Thus, **Manpower** and **manpower** are two different identifiers in C++.

C++ Keywords

The following list shows the reserved words in C++. These reserved words may not be used as constant or variable or any other identifier names.

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union

const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	

Whitespace in C++

A line containing only whitespace, possibly with a comment, is known as a blank line, and C++ compiler totally ignores it.

Whitespace is the term used in C++ to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as int, ends and the next element begins.

Statement 1

```
int age;
```

In the above statement there must be at least one whitespace character (usually a space) between int and age for the compiler to be able to distinguish them.

Statement 2

```
fruit = apples + oranges;   // Get the total fruit
```

In the above statement 2, no whitespace characters are necessary between fruit and =, or between = and apples, although you are free to include some if you wish for readability purpose.

C++ Comments

Program comments are explanatory statements that you can include in the C++ code. These comments help anyone reading the source code. All programming languages allow for some form of comments.

Types of C++ Comments

C++ supports two types of comments: **single-line comments** and **multi-line comments**. All characters available inside any comment are ignored by the C++ compiler.

The types of C++ comments are explained in detail in the next sections:

1. C++ Single-line Comments

A single-line comment starts with `//`, extending to the end of the line. These comments can last only till the end of the line, and the next line leads to a new comment.

Syntax

The following syntax shows how to use a single-line comment in C++:

```
//comments
```

2. C++ Multi-line Comments

Multi-line comments start with **/*** and end with ***/**. Any text in between these symbols is treated as a comment only.

Syntax

The following syntax shows how to use a multi-line comment in C++:

```
/* jsdnsak */
```

Single-line or Multi-line Comments - When to Use?

Single-line comments are generally used for short lines of comments in general. This is seen in cases where we have to mention a small hint for the algorithm in the code.

Multi-line comments are generally used for longer lines of comments, where the visibility of the whole comment line is necessary. The longer the length of the comment, the more number of statements are needed by the multi-line comments.

Purpose of Comments

Comments are used for various purposes in C++. Some of the main areas of application of comments are given as follows:

1. To represent a short and concise step in the program for users to understand better.
2. To explain a step in a detailed way that is not expressed explicitly in the code.
3. To leave different hints for users to grab in the code itself.
4. To leave comments for fun or recreation.
5. To temporarily disable part of the code for debugging purposes.
6. To add metadata to the code for future purposes.
7. To create documentation for the code, for example, in Github pages.