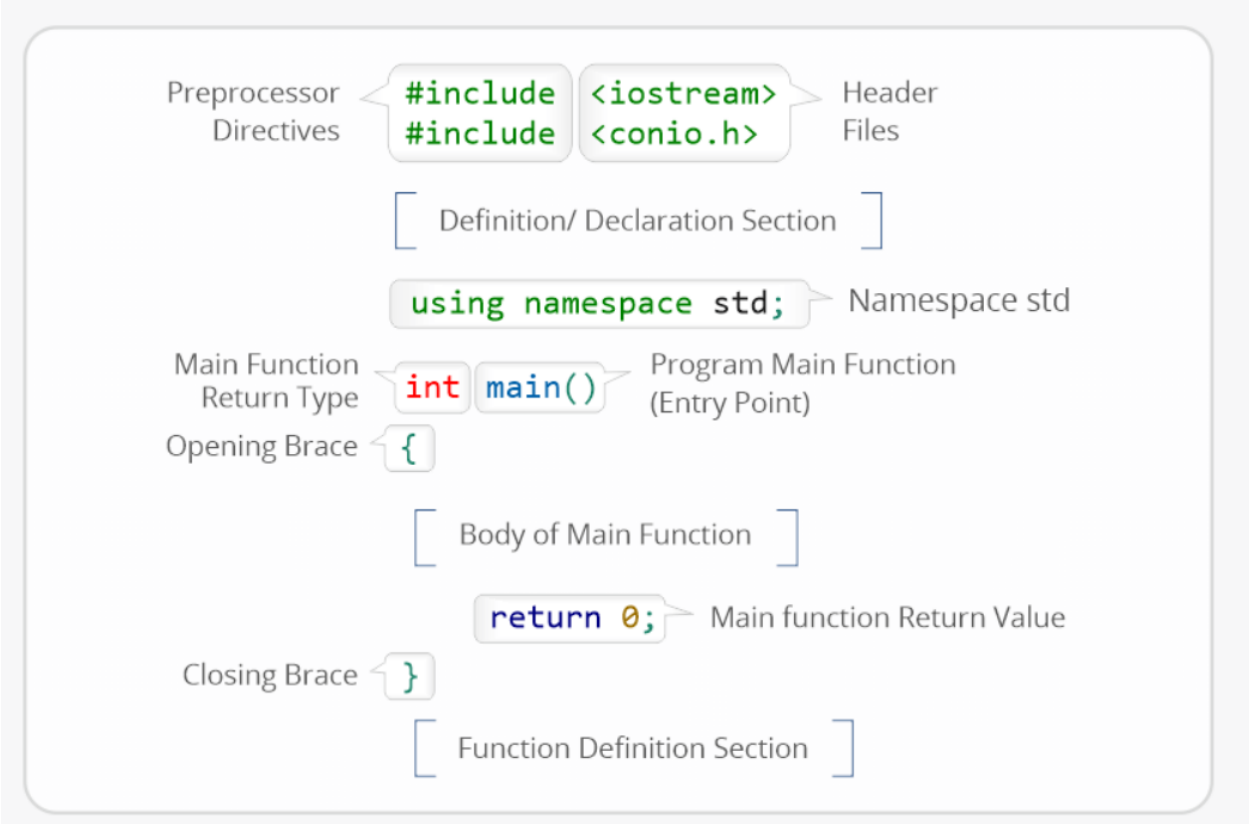


```
#include <iostream>
#include <conio.h>

using namespace std;

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

Preprocessor Directive

The Preprocessor Directive begins with the character **#**. It is used to include the necessary header file in a C++ program before compilation.

Header File

The Header File contains the function declaration and macro definition for C++ in-built library functions which we use in our C++ program during programming. When we include header file

in C++ program using **#include <filename.h>** command, all code inside the header file is included in the C++ program and then the program is sent to the compiler for compilation.

Namespace std

When we use **using namespace std** into the C++ program, then it does not require writing **std::** in front of standard commands throughout the code. Namespace std contains all the classes, objects and functions of the standard C++ library.

Definition/Declaration Section

This section is used to define **macro**, **structure**, **class** and **global** variables to be used in the programs, that means you can use these variables throughout the program.

Program Main Function (Entry Point)

In C++, the **main** function is treated as the entry point of the program, it has a return type (and in some cases accepts inputs via parameters). The main function is called by the operating system when the user runs the program.

Main Function Return Type

In the latest standard of C++, **int** is used as the return type of the main function. It means that the C++ program on its successful completion will return a value which is of type **int**. The default value of the return type is 0 if the program execution is normal.

Opening Brace

This is called the Opening Brace **{**. Whatever we will write inside the main function, we will write it after the Opening Brace.

Body of Main Function

In this section, we will write our C++ program, which will be executed by the **main** function after compilation.

Main Function Return Value

The return value for main is used to indicate how the program exited. If the program execution is normal, a 0 return value is used. Abnormal termination(errors, invalid inputs, segmentation faults, etc.) is usually terminated by a non-zero return.

Closing Brace

This is called the Closing Brace **}**. We use the Closing Brace at the end of the program.

Function Definition Section

When we want to define our function that fulfills a particular requirement, we can define them in this section.

std Namespace Using :: Operator

The first way we access identifiers in the std namespace is by directly qualifying the identifier with the prefix std::. Here,

- std is the C++ standard library namespace
- :: is the scope resolution operator

For example,

Namespaces can be accessed in multiple ways:

- using namespace std;
- using std :: cout;

Example One

```
#include <iostream>

int main() {

  std::string first_name;

  std::cout << "Enter your first name: ";

  std::cin >> first_name;

  std::cout << "Hello " << first_name << "!" << std::endl;

  std::cout << "Welcome!";

  return 0;

}
```

Output

Enter your first name: Marty

Hello Marty!

Welcome!

In the above example, we are using identifiers from the std namespace directly using the scope resolution operator **::**

Notice that we have prefixed std:: before string, cin, cout, and endl by writing:

- *std::string*
- *std::cin*
- *std::cout*
- *std::endl*

If we remove the std:: prefix from the codes above, we will get an error.

std Namespace With using Declaration

We can bring selected identifiers to the current scope with the help of the **using declaration**. To do this, we utilize the using keyword.By doing this, we won't need to prefix the specified identifiers with std::. For example,

```
#include <iostream>// using declaration for cout, endl and string

using std::cout;

using std::endl;

using std::string;

int main() {

  string first_name ;

  cout << "Enter your first name: ";

  std::cin >> first_name;

  cout << "Hello " << first_name << "!" << endl;

  cout << "Welcome!";

  return 0;
```

}

Output

Enter your first name: Scarlet

Hello Scarlet!

Welcome!

In the above example, we have used the using declaration for the identifiers we want to use from the std namespace:

using std::cout;

using std::endl;

using std::string;

Here, we are telling the compiler that we want to bring only the identifiers cout, endl, and string from the standard namespace to the current scope.

This allows us to prevent explicitly adding prefix std:: whenever we need to access any of those identifiers.

Notice that we have used std::cin instead of cin in our program:

std::cin >> first_name;

This is because we have not included std::cin in our using declaration.

std Namespace With using Directive

We can use the **using directive** to bring **all the identifiers** of the namespace std as if they were declared globally. To do this, we utilize the using keyword.

By doing this, we can:

- avoid using the std:: prefix
- avoid utilizing the using declaration repeatedly

For example,

#include <iostream>

```
// using directive

using namespace std;

int main() {

  string first_name ;

  cout << "Enter your first name: ";

  cin >> first_name;

  cout << "Hello " << first_name << "!" << endl;

  cout << "Welcome!";

  return 0;

}
```

Output

Enter your first name: Jack

Hello Jack!

Welcome!

In the above example, we have used the using directive to bring all the identifiers of the std namespace to our program, including the string, cout, cin, and endl identifiers.

Notes:

- The **using declaration** only brings the specified identifiers of the namespace into the current scope.
- The **using directive** brings all the identifiers of a namespace into the current scope.