- Preprocessor Directives in C++

```
#include <bits/stdc++.h>

using namespace std;


int main()

{

   cout<<"Hello World!";


   return 0;

}
```

Output:

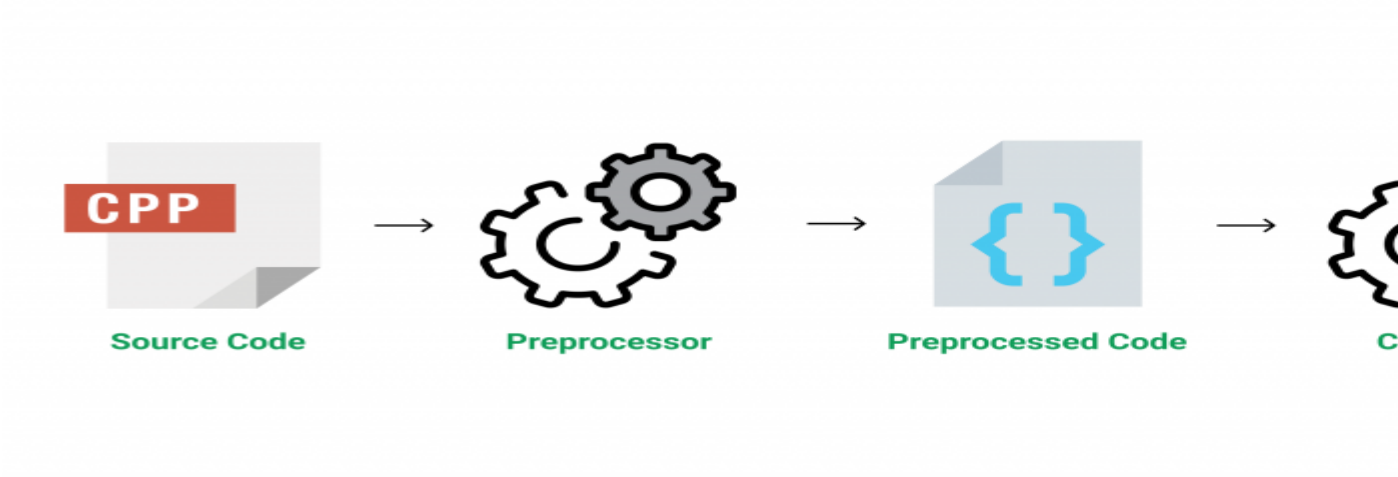
```
Hello World!
```

The program begins with the highlighted statement (in grey): **#include**, which technically is known as a **preprocessor directive**. There are other such directives as well such as **#define**, **#ifdef**, **#pragma** etc.

So, what is a preprocessor directive?

A preprocessor directive is a statement which gets processed by the C++ preprocessor before compilation.

In layman terms, such statements are evaluated prior to the procedure of generation of executable code.



For basic programming in C++, we only need to understand the **#include** and

the **#define** directives.

1. **#include directive**: This type of preprocessor directive tells the compiler to include a file in the source code program and are also known as File Inclusion preprocessor directives. There are two types of files which can be included by the user in the program:

    - **Header File or Standard files**: These files contains definition of pre-defined functions like printf(), scanf() etc. These files must be included for working with these functions. Different function are declared in different header files. For example standard I/O funuctions are in 'iostream' file whereas functions which perform string operations are in 'string' file.

      ```
      #include <file_name>
      ```

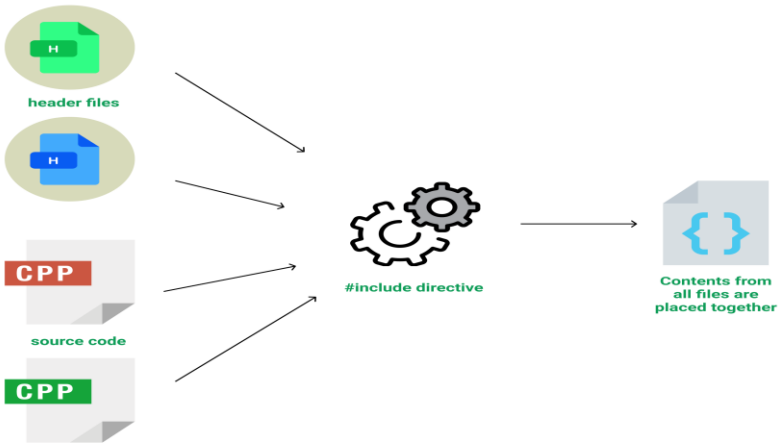
      where *file_name* is the name of file to be included. The '<' and '>' brackets tells the compiler to look for the file in standard directory.

    - **User-defined Files**: When a program becomes very large, it is good practice to divide it into smaller files and include whenever needed. These types of files are user defined files. These files can be included as:

      ```
      #include "filename"
      ```

2. The #include directive instructs the preprocessor to fetch the contents of the file encapsulated within the quotes/arrow and placing it in the source file before compilation (simple copy-paste in layman terms). Some of the important includes are listed below:

    - ***#include < iostream >***: Defines input/output stream objects (cin, cout etc.)
    - ***#include < cstdio >***: C-standard input/output functions (printf, scanf, fscanf, fprintf, fgets etc.)
    - ***#include < cstdlib >***: General purpose utilities functions (random number generation, dynamic memory allocation, integer arithmetic, conversions) (atoi, rand, srand, calloc, malloc)
    - ***#include < bits/stdc++.h >***: Master directive to include all standard header files. (Note that it is not a standard header file and might not be available in compilers other than GCC)
    - ***#include "user-defined-file"***: When we use quotes instead of arrows, we instruct to include user-defined files (.h, .cpp). (useful in projects where one needs to keep custom-functions together)



3. **#define directive**: This directive is used to declare **MACRO**s in the program. A MACRO is a piece of code which is designated a name. When the file is processed by the preprocessor, all appearances of the MACRO name gets replaced by its definition.

   Example:

```
#include <bits/stdc++.h>

#define PI 3.14159265

using namespace std;

int main()

{

    int r=5;

    cout<<"Perimeter of Circle: "<<(2 * PI * r)<<endl;

    cout<<"Area of Circle: "<<(PI * r * r)<<endl;

    return 0;

}
```

Output:

```
Perimeter of Circle: 31.4159
Area of Circle: 78.5398
```

Each appearance gets replaced by the value of *PI*, resulting in (2 * 3.14159265 * r) and (3.14159265 * r * r). Some of the better use-cases of the #define directive can be found in competitive-programming. Such declarations at the beginning of the program greatly increases coding speed (during competitions & placement tests).

```
#define f(l,r) for (int i = l; i < r; i++)
#define ll long long
```

```
#define ull unsigned long long
#define abs(x) (x < 0 ? (-x) : x)
```