#include

In the C Programming, the #include directive causes the pre-processor to fetch the contents of another file into the source code. It will add the content at the point where the #include directive is found. The #include directive is typically used to include the C header files for the C functions that are held outside of the current source file.

A header file is a file which contains the extension ".h". This extension has C macro definitions and C function declarations to be shared between several source files. Header files are of two types:

- files that the programmer writes
- files that come with the compiler.

We request to use a header file in our program by including it with the C preprocessing directive #include, like as we have seen the stdio.h header file, which comes along with the compiler

Syntax:-

The syntax for the #include directive is:

```
#include <header file>
```

The **#include <headerfile>** tells the compiler to look for the directory where system header files are held.

OR

```
#include "header file."
```

The **#include "header file"** tells the compiler to look in the current directory from where the program is running.

Note: If a header file is included within the symbol <>, the pre-processor will search a predetermined directory path to locate the header file. If the header file is enclosed in quotation mark " ", the pre-processor will look for the header file in the same directory as the source file.

Example:-

Following is the example of how to use #include directives in C program.

In this example, we are using the #include directive to include the *stdio.h* header file which is required to use the printf() function from standard C library function, which will print the given argument whether it is string or integer on the screen.

```
#include <stdio.h>
int main()
{
 printf("Code with harry”);
 return 0;
}
```

#define

In the **C** Programming, The #define pre-processor directive is used to define pre-processor variable, constant or macro. Macro operate much like functions. The #define can use any basic data type. This pre-processor directive can be used to replace a word with a number globally. It acts as if an editor did a global search-and-replace edit of the file.

We can use the #define directive for the debugging purpose. We can have print statements that will be only active while debugging.

Syntax:-

The syntax for using #define in the C language is:

```
#define constant_name value
```

OR

```
#define constant_name (expression)
```

- **Constant_name:** The name of the constant.
- **Value:** The value of the constant.
- **Expression:** It is an expression whose value is assigned to the constant. The *expression* must be enclosed in parentheses if it contains any operators.

The following is an example of a #define directive to define a numeric constant:

```
#define Max_length 10
#define Area_OF_Circle(r) (3.1415*(r)*(r))
```

In this example, the constant named *Max_length* would contain the value of 10.

```
#include <stdio.h>
#define PI 3.14
main() {
  printf("%f",PI);
}
```

Output: 3.140000

Beware of two common errors while using the #defines. The most common error is adding extra characters that don't belong. For Example

```
#define MAXARRAYSIZE  = 100 //incorrect
#define MAXARRAYSIZE 100;//correct
```