Content 42

#define and #include Preprocessor Directives

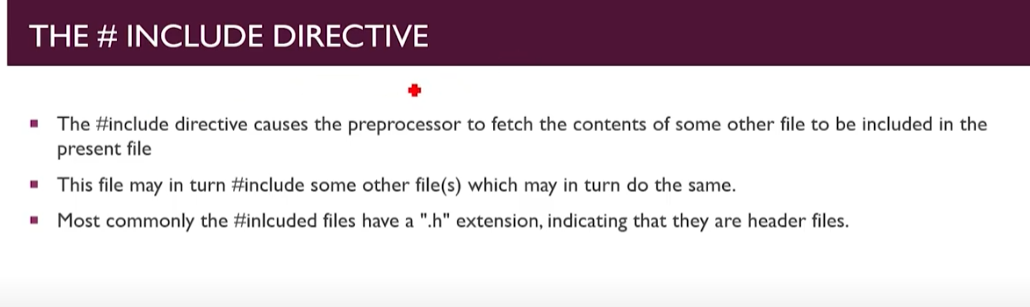
#### #include:-

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

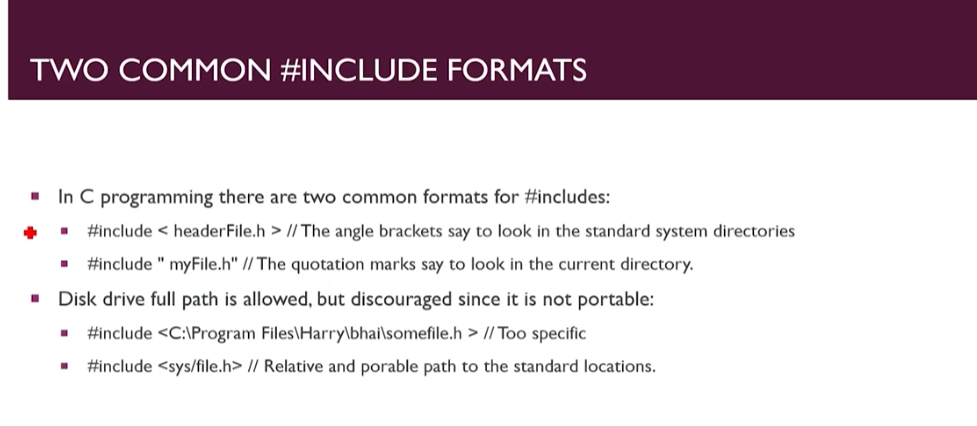


#### Syntax:-

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.

**Program main.c**

#include<stdio.h>

int main(int argc, char const \*argv[])

{

    printf("Hello");

    return 0;

}

**“With the help of #include directive we can merge or include two more files, we also move then here and there”.**

Here I am giving an example in which ‘I had used main.c(program no1) program as a directive to another one (program no2) and also called an function from program 1 to program 2,now lets see.

**Program main.c which I used as a directive;**

#include<stdio.h>

int func(int sum)

{

    int a=2,b=3;

    sum= a+b;

    printf("The sum is %d",sum);

    return sum;

}

int main2(int argc, char const \*argv[])

{

    printf("Hello \n");

    // int a;

    // func(a);

    return 0;

}

But here I made main as main2 otherwise it will throw an error because

here becomes main int both the program so program get confuse which main body he had to execute.

**Program2 Which is mine actual program:**

#include<stdio.h>

#include"main.c" //Created main.c as a directive

int main(int argc, char const \*argv[])

{

    int var=0;

    printf("This is me %d",var);

     printf("\nI had used main.c as directive\n");

    int a;

    func(a);

    return 0;

}

**Output:**

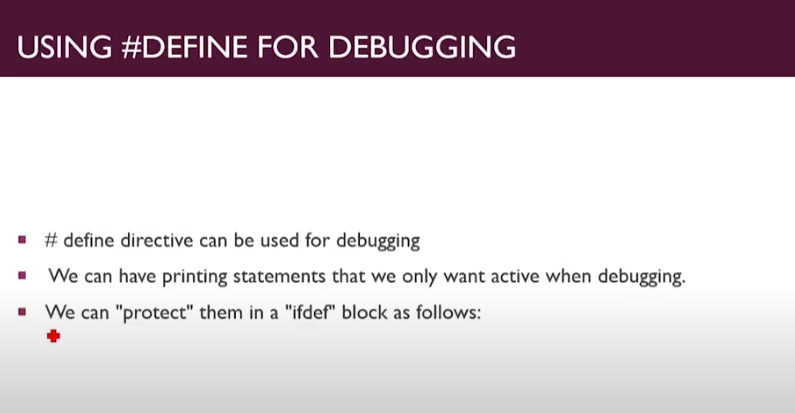
This is me 0

I had used main.c as directive

The sum is 5

#### #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.



**Code1:**

#include <stdio.h>

#define PI 3.14

int main(int argc, char const \*argv[])

{

    float var=PI;

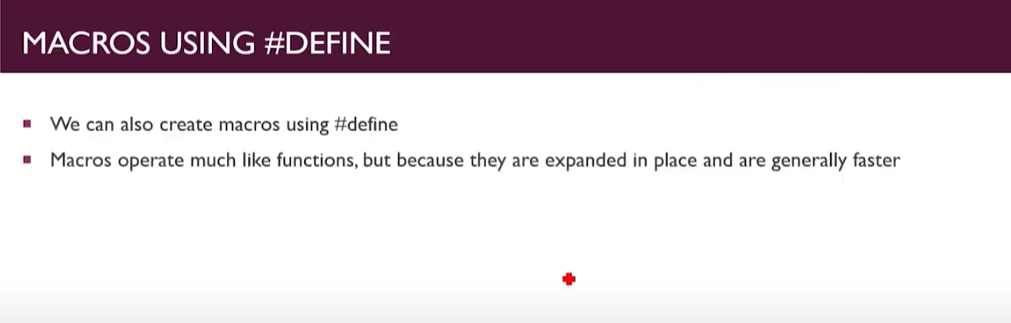
    printf("This is PI= %.3f", var);

    return 0;

}

**Output:**

This is PI= 3.140



**Code for Macros:**

//for macros Directive

#include <stdio.h>

#define PI 3.14

#define square(r) r \*r //here it is macros and r\*r can be replaced by r+r, r-r

int main(int argc, char const \*argv[])

{

    float var = PI;

    int r = 4;

    printf("The area of circle= %f", PI \* square(r));

    return 0;

}

**Output:**

The area of circle= 50.240