Predefined Macros & Other Pre-processor Directives

Let's start **preprocessor directives** as we have already covered #define and #include in the previous tutorial. In this, we will move forward to a few others.

#undef:

As can be guessed by the name, it is used to undefine a macro to eliminate its definition.

Example:

```
#define E 1111


#undef E
```

E is not defined for any value after using #undef, so it holds no value in the above example.

#ifdef:

It is used to check whether a macro is defined or not. If it is defined, then it executes the code.

Example:

```
#ifdef M
//execute code if true i.e., the macro is defined
```

#ifndef:

It works exactly the opposite of ifdef. Meaning that it executes the code if the macro is undefined.

Example:

```
#ifdef M
//execute code if true i.e., the macro is defined
```

#if:

It checks whether the given condition is true or not. If true, then it executes the code.

Example:

```
#if condition
// execute code if true i.e. condition satisfied
```

#else:

If the condition of ‘if’ is false, then the else is executed.

Example:

```
#if condition
// execute code if true else pass it to #else

#else
// execute code if “if condition” is false
```

#elif:

It is used to insert more conditions between if and else. If the “if statement” is true, then elif won’t be checked.

Example:

```
#if condition
// execute code if true else pass it to #elif

#elif expression
// execute code if true else pass it to #else

#else
//else code
```

#pragma:

Pragma is used to issue some **special commands** to compiler.

Let's discuss some **pre-defined macros** now. A pre-defined macro is a macro that has already been defined or understood by C preprocessor and does not need a definition.

__DATE__:

It prints the current date on to the screen. The date format it follows is MMMDDYYY.

Syntax:

```
#include <stdio.h>
int main()
{
                printf( __DATE__);
                return 0;
}
```

Output**:**

```
Sep 15 2020
```

Note: The output will be the current date.

__TIME__:

It prints the current time on to the screen. The date format it follows is HH:MM:SS.

Syntax:

```
#include <stdio.h>
int main()
{
                printf( __TIME__);
                return 0;
}
```

Output:

```
17:15:20
```

Note: The output will be the current time.

__FILE__:

It prints the current file name on to the screen. The name will be printed as a string literal.

Syntax:

```
#include <stdio.h>
int main()
{
                printf( __FILE__);
                return 0;
}
```

Output:

```
main.c
```

__LINE__:

It prints the current line number on to the screen. The number will be printed as a decimal constant.

Syntax:

```
#include <stdio.h>
int main()
{
                printf( “%d”__LINE__);
                return 0;
}
```

Output:

```
4
```

__STDC__:

It is used to check whether our program is being compiler using ANSI standard or not. It will return 1 if true.

Syntax:

```
#include <stdio.h>
int main()

    printf("%d\n", __STDC__);
    return 0;
}
```

Output:

```
1
```