



My Profile Logout

C/C++ Preprocessor directives | Set 2

isgraph() C library function

How to write your own header file in C?

difftime() C library function

tmpnam() function in C

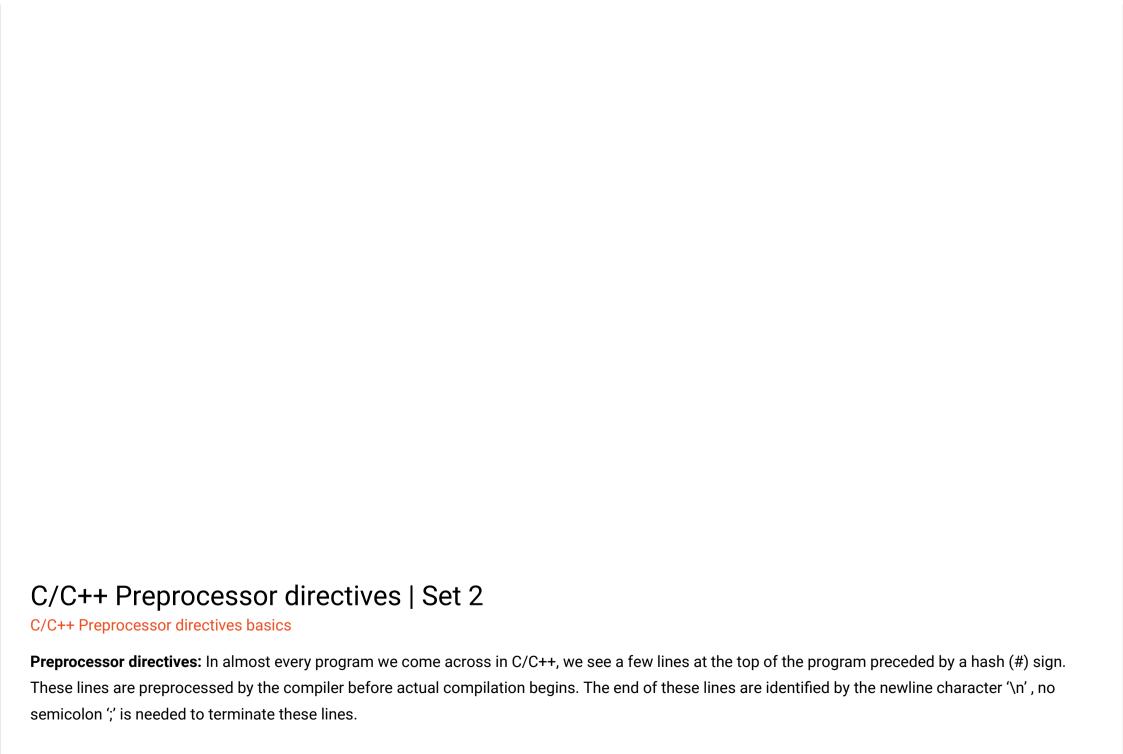
_Generic keyword in C

return statement in C/C++ with Examples

size of char datatype and char array in C

Arrow operator -> in C/C++ with Examples

Logical Not! operator in C with Examples



Preprocessor directives are mostly used in defining macros, evaluating conditional statements, source file inclusion, pragma directive, line control, error		
detection etc.		
In this post, we will discuss about some more types of preprocessor directives given below:		
Conditional Compilation Line control		
3. Error directive		
Let us now have a look about each one of these directives in details:		
• Conditional Compilation: Conditional Compilation directives help to compile a specific portion of the program or let us skip compilation of some specific part of the program based on some conditions. In our previous article, we have discussed about two such directives 'ifdef' and 'endif'. In this post we will discuss #ifndef, #if, #else and #elif.		
1. #ifdef: This directive is the simplest conditional directive. This block is called a conditional group. The controlled text will get included in the preprocessor output iff the macroname is defined. The controlled text inside a conditional will embrace preprocessing directives. They are executed only if the conditional succeeds. You can nest these in multiple layers, but they must be completely nested. In other words, '#endif' always matches the nearest '#ifdef' (or '#ifndef', or '#if'). Also, you can't begin a conditional group in one file and finish it in another.		

Syntax:

```
#ifdef MACRO
     controlled text
#endif /* macroname */
```

2. **#ifndef**: We know that the in #ifdef directive if the macroname is defined, then the block of statements following the #ifdef directive will execute normally but if it is not defined, the compiler will simply skip this block of statements. The #ifndef directive is simply opposite to that of the #ifdef directive. In case of #ifndef, the block of statements between #ifndef and #endif will be executed only if the macro or the identifier with #ifndef is not defined.

Syntax:

```
ifndef macro_name
    statement1;
    statement2;
    statement3;
    .
    .
    statementN;
endif
```

If the macro with name as 'macroname' is not defined using the #define directive then only the block of statements will execute.

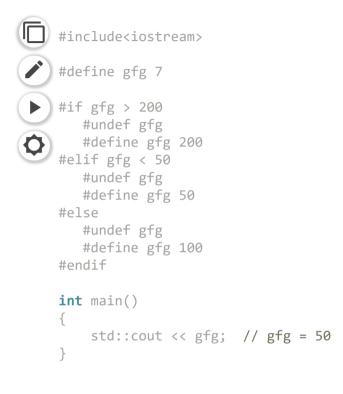
3. **#if, #else and #elif**: These directives works together and control compilation of portions of the program using some conditions. If the condition with the #if directive evaluates to a non zero value, then the group of line immediately after the #if directive will be executed otherwise if the condition with the #elif directive evaluates to a non zero value, then the group of line immediately after the #elif directive will be executed else the lines after #else directive will be executed.

Syntax:

```
#if macro_condition
    statements
#elif macro_condition
    statements
```

```
#else
    statements
#endif
```

Example:



Output:

50

Notice how the entire structure of #if, #elif and #else chained directives ends with #endif.

• Line control (#line): Whenever we compile a program, there are chances of occurrence of some error in the program. Whenever compiler identifies error in the program it provides us with the filename in which error is found along with the list of lines and with the exact line numbers where the error is. This makes easy for us to find and rectify error.

However we can control what information should the compiler provide during errors in compilation using the #line directive.

Syntax:

```
#line number "filename"
```

number – line number that will be assigned to the next code line. The line numbers of successive lines will be increased one by one from this point on.

"filename" - optional parameter that allows to redefine the file name that will be shown.

• Error directive (#error): This directive aborts the compilation process when it is found in the program during compilation and produces an error which is optional and can be specified as a parameter.

Syntax:

```
#error optional error
```

Here, **optional_error** is any error specified by the user which will be shown when this derective is found in the program.

Example:



Output:

```
error: #error GeeksforGeeks not found!
```

References:

- ppd_better_practice_cs.auckland.ac.nz
- http://www.cplusplus.com/doc/tutorial/preprocessor/

This article is contributed by **Amartya Ranjan Saikia** and **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

How a Preprocessor works in C?

- C | Macro & Preprocessor | Question 9
- C | Macro & Preprocessor | Question 8
- C | Macro & Preprocessor | Question 7
- C | Macro & Preprocessor | Question 6
- C | Macro & Preprocessor | Question 5
- C | Macro & Preprocessor | Question 14
- C | Macro & Preprocessor | Question 4

C Macro & Preprocessor Question 1	
C Macro & Preprocessor Question 2	
C Macro & Preprocessor Question 10	
C Macro & Preprocessor Question 11	
C Macro & Preprocessor Question 15	
C Macro & Preprocessor Question 14	
C Macro & Preprocessor Question 13	
Improved By: RishabhPrabhu	
Article Tags: C C++ C Basics C-Macro & Preprocessor cpp-macros Macro & Preprocessor	
Practice Tags: C CPP	
5	
	2.5
To-do Done	Based on 6 vote(s)
Foodback / Suggest Improvement (Notes) (Improve Article)	
L Loodhook/ Suggest Improvement 1 Notes 1 Improve Article 1	

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

A computer science portal for geeks

5th Floor, A-118, Sector-136, Noida, Uttar Pradesh - 201305 feedback@geeksforgeeks.org

COMPANY

About Us Careers Privacy Policy Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos

@geeksforgeeks, Some rights reserved