

C# - PREPROCESSOR DIRECTIVES

https://www.tutorialspoint.com/csharp/csharp_preprocessor_directives.htm

Copyright © tutorialspoint.com

Advertisements

The preprocessor directives give instruction to the compiler to preprocess the information before actual compilation starts.

All preprocessor directives begin with #, and only white-space characters may appear before a preprocessor directive on a line. Preprocessor directives are not statements, so they do not end with a semicolon ;.

C# compiler does not have a separate preprocessor; however, the directives are processed as if there was one. In C# the preprocessor directives are used to help in conditional compilation. Unlike C and C++ directives, they are not used to create macros. A preprocessor directive must be the only instruction on a line.

Preprocessor Directives in C#

The following table lists the preprocessor directives available in C# –

Sr.No.	Preprocessor Directive & Description
1	#define It defines a sequence of characters, called symbol.
2	#undef It allows you to undefine a symbol.
3	#if It allows testing a symbol or symbols to see if they evaluate to true.
4	#else It allows to create a compound conditional directive, along with #if.
5	#elif It allows creating a compound conditional directive.

6	#endif Specifies the end of a conditional directive.
7	#line It lets you modify the compiler's line number and <i>optionally</i> the file name output for errors and warnings.
8	#error It allows generating an error from a specific location in your code.
9	#warning It allows generating a level one warning from a specific location in your code.
10	#region It lets you specify a block of code that you can expand or collapse when using the outlining feature of the Visual Studio Code Editor.
11	#endregion It marks the end of a #region block.

The #define Preprocessor

The #define preprocessor directive creates symbolic constants.

#define lets you define a symbol such that, by using the symbol as the expression passed to the #if directive, the expression evaluates to true. Its syntax is as follows –

```
#define symbol
```

The following program illustrates this –

[Live Demo](#)

```
#define PI
using System;

namespace PreprocessorDApp1 {
    class Program {
        static void Main(string[] args) {
            #if (PI)
                Console.WriteLine("PI is defined");
            #else
                Console.WriteLine("PI is not defined");
            #endif
            Console.ReadKey();
        }
    }
}
```

When the above code is compiled and executed, it produces the following result –

```
PI is defined
```

Conditional Directives

You can use the `#if` directive to create a conditional directive. Conditional directives are useful for testing a symbol or symbols to check if they evaluate to true. If they do evaluate to true, the compiler evaluates all the code between the `#if` and the next directive.

Syntax for conditional directive is –

```
#if symbol [operator symbol]...
```

Where, *symbol* is the name of the symbol you want to test. You can also use true and false or prepend the symbol with the negation operator.

The *operator symbol* is the operator used for evaluating the symbol. Operators could be either of the following –

- `==` *equality*
- `!=` *inequality*
- `&&` *and*
- `||` *or*

You can also group symbols and operators with parentheses. Conditional directives are used for compiling code for a debug build or when compiling for a specific configuration. A conditional directive beginning with a `#if` directive must explicitly be terminated with a `#endif` directive.

The following program demonstrates use of conditional directives –

[Live Demo](#)

```
#define DEBUG
#define VC_V10
using System;

public class TestClass {
```

```
public static void Main() {  
    #if (DEBUG && !VC_V10)  
        Console.WriteLine("DEBUG is defined");  
    #elif (!DEBUG && VC_V10)  
        Console.WriteLine("VC_V10 is defined");  
    #elif (DEBUG && VC_V10)  
        Console.WriteLine("DEBUG and VC_V10 are defined");  
    #else  
        Console.WriteLine("DEBUG and VC_V10 are not defined");  
    #endif  
    Console.ReadKey();  
}
```

When the above code is compiled and executed, it produces the following result –

```
DEBUG and VC_V10 are defined
```