**C# Keywords and Identifiers**

In this tutorial, we will learn about keywords (reserved words) and identifiers in C# programming language.

**C# Keywords**

Keywords are predefined sets of reserved words that have special meaning in a program. The meaning of keywords cannot be changed, neither can they be directly used as identifiers in a program.

For example,

long mobileNum;

Here, long is a keyword and mobileNum is a variable (identifier). long has a special meaning in C# i.e. it is used to declare variables of type long and this function cannot be changed.

Also, keywords like long, int, char, etc. cannot be used as identifiers. So, we cannot have something like:

long long;

C# has a total of 79 keywords. All these keywords are in lowercase. Here is a complete list of all C# keywords.

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | as | base | bool |
| break | byte | case | catch |
| char | checked | class | const |
| continue | decimal | default | delegate |
| do | double | else | enum |
| event | explicit | extern | false |
| finally | fixed | float | for |
| foreach | goto | if | implicit |
| in | in (generic modifier) | int | interface |
| internal | is | lock | long |
| namespace | new | null | object |
| operator | out | out (generic modifier) | override |
| params | private | protected | public |
| readonly | ref | return | sbyte |
| sealed | short | sizeof | stackalloc |
| static | string | struct | switch |
| this | throw | true | try |
| typeof | uint | ulong | unchecked |
| unsafe | ushort | using | using static |
| void | volatile | while |  |

Although keywords are reserved words, they can be used as identifiers if @ is added as prefix. For example,

int @void;

The above statement will create a variable @void of type int.

**Contextual Keywords**

|  |  |  |
| --- | --- | --- |
| add | alias | ascending |
| async | await | descending |
| dynamic | from | get |
| global | group | into |
| join | let | orderby |
| partial (type) | partial (method) | remove |
| select | set | value |
| var | when (filter condition) | where (generic type constraint) |
| yield |  |  |

Besides regular keywords, C# has 25 contextual keywords. Contextual keywords have specific meaning in a limited program context and can be used as identifiers outside that context. They are not reserved words in C#.

If you are interested to know the function of every keywords, I suggest you visit [C# keywords](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/index) (official C# docs).

**C# Identifiers**

Identifiers are the name given to entities such as variables, methods, classes, etc. They are tokens in a program which uniquely identify an element. For example,

int value;

Here, value is the name of variable. Hence it is an identifier. Reserved keywords cannot be used as identifiers unless @ is added as prefix. For example,

int break;

This statement will generate an error in compile time. To learn more about variables, visit [C# Variables](https://www.programiz.com/csharp-programming/variables-primitive-data-types).

**Rules for Naming an Identifier**

* An identifier cannot be a C# keyword.
* An identifier must begin with a letter, an underscore or @ symbol. The remaining part of identifier can contain letters, digits and underscore symbol.
* Whitespaces are not allowed. Neither it can have symbols other than letter, digits and underscore.
* Identifiers are case-sensitive. So, getName, GetName and getname represents 3 different identifiers.

Here are some of the valid and invalid identifiers:

| Identifiers | Remarks |
| --- | --- |
| number | Valid |
| calculateMarks | Valid |
| hello$ | Invalid (Contains $) |
| name1 | Valid |
| @if | Valid (Keyword with prefix @) |
| if | Invalid (C# Keyword) |
| My name | Invalid (Contains whitespace) |
| \_hello\_hi | Valid |

**Example: Find list of keywords and identifiers in a program**

Just to clear the concept, let's find the list of keywords and identifiers in the program we wrote in [C# Hello World](https://www.programiz.com/csharp-programming/hello-world).

using System;

namespace HelloWorld

{

class Hello

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

}

}

}

| Keywords | Identifiers |
| --- | --- |
| using | System |
| namespace | HelloWorld (namespace) |
| class | Hello (class) |
| static | Main (method) |
| void | args |
| string | Console |
|  | WriteLine |

The "Hello World!" inside WriteLine method is a string literal.