arguments debugger delete eval final function goto implements in instanceof let native package private protected public static synchronized transient typeof var volatile with yield export extends import super

[abstract](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/abstract)

[boolean](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/bool) type – only holds true or false

[break](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/break) allows for a controlled exit from a loop (for,do-while,while) or switch block  
The break statement terminates the closest enclosing loop or [switch](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/switch) statement in which it appears. Control is passed to the statement that follows the terminated statement, if any.

[byte](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/byte) 0 to 255 Unsigned 8-bit integer

[case](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/switch) Each case label specifies a pattern to compare to the match expression (the caseSwitch variable in the previous examples). If they match, control is transferred to the switch section that contains the first matching case label. If no case label pattern matches the match expression, control is transferred to the section with the default case label, if there is one. If there is no default case, no statements in any switch section are executed, and control is transferred outside the switch statement.  
switch (caseSwitch)

{

case 1:

Console.WriteLine("Case 1");

break;

case 2:

Console.WriteLine("Case 2");

break;

default:

Console.WriteLine("Default case");

break;

}

[catch](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-catch) The try-catch statement consists of a try block followed by one or more catch clauses, which specify handlers for different exceptions.  
catch (FileNotFoundException e)  
{   
 // FileNotFoundExceptions are handled here.   
}   
catch (IOException e)   
{   
 // Extract some information from this exception, and then   
 // throw it to the parent method.   
 if (e.Source != null)   
 Console.WriteLine("IOException source: {0}", e.Source);   
 throw;   
}

[char](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/char) U+0000 to U+FFFF Unicode 16-bit character

[class](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/class) Classes are declared using the keyword class

[const](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/const) You use the const keyword to declare a constant field or a constant local. Constant fields and locals aren't variables and may not be modified. Constants can be numbers, Boolean values, strings, or a null reference. Don’t create a constant to represent information that you expect to change at any time.   
const int x = 0;  
public const double gravitationalConstant = 6.673e-11;  
private const string productName = "Visual C#";

[continue](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/continue) The continue statement passes control to the next iteration of the enclosing [while](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/while), [do](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/do), [for](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/for), or [foreach](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/foreach-in) statement in which it appears.  
for (int i = 1; i <= 10; i++)

{

if (i < 9)

{

continue;

}

Console.WriteLine(i);

}

[default](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/default) The default case specifies the switch section to execute if the match expression does not match any other case label. If a default case is not present and the match expression does not match any other case label, program flow falls through the switch statement.  
The default case can appear in any order in the switch statement. Regardless of its order in the source code, it is always evaluated last, after all case labels have been evaluated.

[do](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/do) The do statement executes a statement or a block of statements repeatedly until a specified expression evaluates to false. The body of the loop must be enclosed in braces, {}, unless it consists of a single statement. In that case, the braces are optional.

int x = 0;

do

{

Console.WriteLine(x);

x++;

} while (x < 5);

[double](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/double) ±5.0 × 10−324 to ±1.7 × 10308 15-16 digits

[else](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/if-else) An if statement identifies which statement to run based on the value of a Boolean expression. Below the Boolean variable result is set to true and then checked in the if statement. The output is ”The condition is true”.

bool condition = true;

if (condition)

{

Console.WriteLine("The variable is set to true.");

}

else

{

Console.WriteLine("The variable is set to false.");

}

[enum](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/enum) The enum keyword is used to declare an enumeration, a distinct type that consists of a set of named constants called the enumerator list.  
enum Days {Sat, Sun, Mon, Tue, Wed, Thu, Fri};

[false](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/false) represents the boolean value false

[finally](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-finally) A common usage of catch and finally together is to obtain and use resources in a try block, deal with exceptional circumstances in a catch block, and release the resources in the finally block.  
For more information and examples on re-throwing exceptions, see [try-catch](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-catch) and [Throwing Exceptions](https://docs.microsoft.com/en-us/dotnet/standard/exceptions/index). For more information about the finally block, see [try-finally](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-finally).

string path = @"c:\users\public\test.txt";

System.IO.StreamReader file = new System.IO.StreamReader(path);

char[] buffer = new char[10];

try

{

file.ReadBlock(buffer, index, buffer.Length);

}

catch (System.IO.IOException e)

{

Console.WriteLine("Error reading: {0}. Msg = {1}", path, e.Message);

}

finally

{

if (file != null)

{

file.Close();

}

}

[float](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/float) -3.4 × 1038to +3.4 × 1038 7 digits

[for](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/for) By using a for loop, you can run a statement or a block of statements repeatedly until a specified expression evaluates to false. This kind of loop is useful for iterating over arrays and for other applications in which you know in advance how many times you want the loop to iterate.

for (int i = 1; i <= 5; i++)

{

Console.WriteLine(i);

}

[for](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/foreach-in) of The foreach statement repeats a group of embedded statements for each element in an array or an object collection that implement an [IEnumerable<T>](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.ienumerable-1) interface. The foreach statement is used to iterate through the collection to get the information that you want, but cannot be used to add or remove items from the source collection to avoid unpredictable side effects. If you need to add or remove items from the source collection, use a [for](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/for) loop.  
The embedded statements continue to execute for each element in the array or collection. After the iteration has been completed for all the elements in the collection, control is transferred to the next statement following the foreach block.  
At any point within the foreach block, you can break out of the loop by using the [break](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/break) keyword, or step to the next iteration in the loop by using the [continue](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/continue) keyword.

int[] fibarray = new int[] { 0, 1, 1, 2, 3, 5, 8, 13 };

foreach (int element of fibarray)

{

System.Console.WriteLine(element);

}

[if](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/if-else) An if statement identifies which statement to run based on the value of a Boolean expression. In the following example, the Boolean variable result is set to true and then checked in the if statement. The output is The condition is true.

bool condition = true;

if (condition)

{

Console.WriteLine("The variable is set to true.");

}

else

{

Console.WriteLine("The variable is set to false.");

}

[int](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/int) -2,147,483,648 to 2,147,483,647 Signed 32-bit integer

[interface](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/interface) An interface contains only the signatures of [methods](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/methods), [properties](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/properties), [events](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/events/index) or [indexers](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/indexers/index). A class or struct that implements the interface must implement the members of the interface that are specified in the interface definition. In the following example, class ImplementationClass must implement a method named SampleMethod that has no parameters and returns void.

[is](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/is) Checks if an object is compatible with a given type, or (starting with C# 7) tests an expression against a pattern.  
if (obj is Person) {

// Do something if obj is a Person.

}

[long](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/long) -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Signed 64-bit integer

[new](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/new) Used to create objects and invoke constructors.

[null](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/null) The null keyword is a literal that represents a null reference, one that does not refer to any object. null is the default value of reference-type variables. Ordinary value types cannot be null. However, C# 2.0 introduced nullable value types.

[object](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/object) The object type is an alias for [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object) in the .NET Framework. All types, predefined and user-defined, reference types and value types, inherit directly or indirectly from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object). You can assign values of any type to variables of type object.

[private](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/private)

[protected](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected)

[public](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/public)

[return](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/return) The return statement terminates execution of the method in which it appears and returns control to the calling method. It can also return an optional value. If the method is a void type, the return statement can be omitted.  
If the return statement is inside a try block, the finally block, if one exists, will be executed before control returns to the calling method.

[short](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/short) -32,768 to 32,767 Signed 16-bit integer

[string](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/string) The string type represents a sequence of zero or more Unicode characters. string is an alias for [String](https://docs.microsoft.com/en-us/dotnet/api/system.string) in the .NET Framework. Although string is a reference type, the equality operators (== and !=) are defined to compare the values of string objects, not references. This makes testing for string equality more intuitive.

[switch](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/switch) switch is a selection statement that chooses a single switch section to execute from a list of candidates based on a pattern match with the match expression.   
the match expression must be an expression that returns a value of the following types:

* a char.
* a string.
* a bool.
* an integral value, such as an int or a long.
* an enum value.

**Case labels**  
Each case label specifies a valueto compare to the match expression (the caseSwitch variable in the previous examples). If matched, go to the matching **case** label. If no case label pattern matches the match expression, go to the **default** case label, if there is one. If there is no default case, no statements in any switch section are executed, and control is transferred outside the switch statement.

Random rnd = new Random();

int caseSwitch = rnd.Next(1,4);

switch (caseSwitch)

{

case 1:

Console.WriteLine("Case 1");

break;

case 2:

case 3:

Console.WriteLine($"Case {caseSwitch}");

break;

default:

Console.WriteLine($"An unexpected value ({caseSwitch})");

break;

}

[this](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/this) The this keyword refers to the current instance of the class and is also used as a modifier of the first parameter of an extension method.

[throw](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/throw) Signals the occurrence of an exception during program execution.

[true](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/true) represents the boolean value true

[try](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-catch) The try-catch statement consists of a try block followed by one or more catch clauses, which specify handlers for different exceptions.

[typeof](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/typeof) Used to obtain the System.Type object for a type. A typeof expression takes the following form:

System.Type type = typeof(int);

[void](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/void) When used as the return type for a method, void specifies that the method doesn't return a value. void isn't allowed in the parameter list of a method.

[while](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/while) The while statement executes a statement or a block of statements until a specified expression evaluates to false.  
int n = 1;

while (n < 6)

{

Console.WriteLine("Current value of n is {0}", n);

n++;

}