## Features of C# 3.0

C# 3.0 is the current version, and was released on [19 November](http://en.wikipedia.org/wiki/November_19) [2007](http://en.wikipedia.org/wiki/2007) as part of .NET Framework 3.5. It includes new features inspired by [functional programming](http://en.wikipedia.org/wiki/Functional_programming) languages such as [Haskell](http://en.wikipedia.org/wiki/Haskell_(programming_language)) and [ML](http://en.wikipedia.org/wiki/ML_(programming_language)), and is driven largely by the introduction of the [Language Integrated Query](http://en.wikipedia.org/wiki/Language_Integrated_Query) (LINQ) pattern to the Common Language Runtime.[[13]](http://en.wikipedia.org/wiki/C_Sharp_(programming_language)#cite_note-12#cite_note-12) It is not currently standardized by any [standards organisation](http://en.wikipedia.org/wiki/Standards_organisation).

### [[edit](http://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&action=edit&section=18)] LINQ (Language-Integrated Query)

[LINQ](http://en.wikipedia.org/wiki/LINQ) is an extensible, general-purpose query language for many kinds of data sources (including plain object collections, XML documents, databases, and so on) which is tightly integrated with other C# language facilities. The syntax heavily borrows from SQL. An example:

int[] array = { 1, 5, 2, 10, 7 };

*// Select squares of all odd numbers in the array sorted in descending order*

IEnumerable<int> query = from x in array

where x % 2 == 1

orderby x descending

select x \* x;

### [[edit](http://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&action=edit&section=19)] Object initializers

Customer c = new Customer(); c.Name = "James";

can be written

Customer c = new Customer { Name="James" };

### [[edit](http://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&action=edit&section=20)] Collection initializers

MyList list = new MyList();

list.Add(1);

list.Add(2);

can be written as

MyList list = new MyList { 1, 2 };

assuming that MyList implements System.Collections.IEnumerable and has a public Add method[[14]](http://en.wikipedia.org/wiki/C_Sharp_(programming_language)#cite_note-13#cite_note-13)

### [[edit](http://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&action=edit&section=21)] Anonymous types

var x = new { FirstName="James", LastName="Frank" }; [[15]](http://en.wikipedia.org/wiki/C_Sharp_(programming_language)#cite_note-14#cite_note-14)

### [[edit](http://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&action=edit&section=22)] Local variable type inference

Local variable [type inference](http://en.wikipedia.org/wiki/Type_inference):

var x = new Dictionary<string, List<float>>();

is interchangeable with

Dictionary<string, List<float>> x = new Dictionary<string, List<float>>();

This feature is not just a convenient [syntactic sugar](http://en.wikipedia.org/wiki/Syntactic_sugar) for shorter local variable declarations, but it is also required for the declaration of variables of anonymous types.

### [[edit](http://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&action=edit&section=23)] Lambda expressions

[Lambda](http://en.wikipedia.org/wiki/Closure_(computer_science)) expressions provide a concise way to write first-class anonymous function values. Compare the following C# 2.0 snippet:

listOfFoo.Where(delegate(Foo x) { return x.Size > 10; })

with this C# 3.0 equivalent:

listOfFoo.Where(x => x.Size > 10);

In the above examples, lambda expressions are merely short-hand syntax for anonymous delegates with type inference for parameters and return type. However, depending on the context they are used in, a C# compiler can also transform lambdas into [ASTs](http://en.wikipedia.org/wiki/Abstract_syntax_tree) that can then be processed at run-time. In the example above, if listOfFoo is not a plain in-memory collection, but a wrapper around a database table, it could use this technique to translate the body of the lambda into the equivalent SQL expression for optimized execution. Either way, the lambda expression itself looks exactly the same in the code, so the way it is used at run-time is transparent to the client.

### [[edit](http://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&action=edit&section=24)] Automatic properties

The compiler will generate a private instance variable and the appropriate getter and setter given code such as: public string Name { get; private set; }

### [[edit](http://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&action=edit&section=25)] Extension methods

Extension methods are a form of syntactic sugar providing the illusion of adding new methods to the existing class outside its definition. In practice, an extension method is a static method that is callable as if it was an instance method; the receiver of the call is bound to the first parameter of the method, decorated with keyword this:

public static class StringExtensions

{

public static string Left(this string s, int n)

{

return s.Substring(0, n);

}

}

string s = "foo";

s.Left(3); *// same as StringExtensions.Left(s, 3);*

### [[edit](http://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&action=edit&section=26)] Partial methods

Partial methods allow code generators to generate method declarations as extension points that are only included in the source code compilation if someone actually implements it in another portion of a partial class.[[16]](http://en.wikipedia.org/wiki/C_Sharp_(programming_language)#cite_note-15#cite_note-15)