Anatomy of the Lambda Expression

C# 3.0(.NET 3.5) introduced the lambda expression along with LINQ. The lambda expression is a shorter way of representing [anonymous method](https://www.tutorialsteacher.com/csharp/csharp-anonymous-method) using some special syntax.

For example, following anonymous method checks if student is teenager or not:

Example: Anonymous Method in C#

delegate(Student s) { return s.Age > 12 && s.Age < 20; };

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-HabxnO)

Example: Anonymous method in VB.Net

Dim isStudentTeenAger = Function(s As Student) As Boolean

Return s.Age > 12 And s.Age < 20

End Function

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-77xAMs)

The above anonymous method can be represented using a Lambda Expression in C# and VB.Net as below:

Example: Lambda Expression in C#

s => s.Age > 12 && s.Age < 20

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-rqSYnO)

Example: Lambda Expression in VB.Net

Function(s) s.Age > 12 And s.Age < 20

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-R13yq1)

Let's see how the lambda expression evolved from the following anonymous method.

Example: Anonymous method in C#

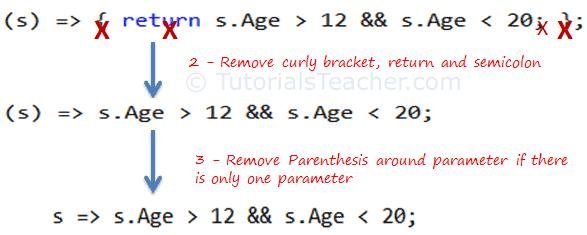
delegate(Student s) { return s.Age > 12 && s.Age < 20; };

The Lambda expression evolves from anonymous method by first removing the delegate keyword and parameter type and adding a lambda operator =>.

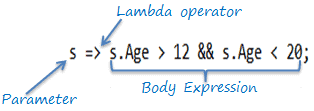
Lambda Expression from Anonymous Method

The above lambda expression is absolutely valid, but we don't need the curly braces, return and semicolon if we have only one statement that returns a value. So we can eliminate it.

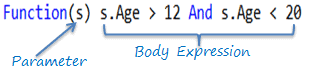
Also, we can remove parenthesis (), if we have only one parameter.

[](https://www.tutorialsteacher.com/Content/images/linq/lambda-expression-2.png)Lambda Expression from Anonymous Method

Thus, we got the lambda expression: s => s.Age > 12 && s.Age < 20 where **s** is a parameter, **=>** is the lambda operator and **s.Age > 12 && s.Age < 20** is the body expression:

[](https://www.tutorialsteacher.com/Content/images/linq/lambda-expression-structure.png)Lambda Expression Structure in C#

Same way we got lambda expression in VB.Net can be written as below:

[](https://www.tutorialsteacher.com/Content/images/linq/lambda-expression-vb.png)Lambda Expression Structure in VB.Net

The lambda expression can be invoked same way as delegate using ().

 Note:

VB.Net doesn't support lambda operator =>

Lambda Expression with Multiple Parameters

You can wrap the parameters in parenthesis if you need to pass more than one parameter, as below:

Example: Specify Multiple Parameters in Lambda Expression C#

**(s, youngAge)** => s.Age >= youngage;

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-PWGyET)

You can also give type of each parameters if parameters are confusing:

Example: Specify Parameter Type

**(Student s,int youngAge)** => s.Age >= youngage;

Example: Specify Multiple Parameters in Lambda Expression VB.Net

Function(s, youngAge) s.Age >= youngAge

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-52ROJw)

Lambda Expression without Parameter

It is not necessary to have atleast one parameter in a lambda expression. The lambda expression can be specify without any parameter also.

Example: Lambda Expression without Parameter

**()** => Console.WriteLine("Parameter less lambda expression")

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-1JQVyo)

Multiple Statements in Lambda Expression Body

You can wrap expressions in curly braces if you want to have more than one statement in the body:

Example: Multi Statements Lambda expression C#

(s, youngAge) =>

**{**

Console.WriteLine("Lambda expression with multiple statements in the body");

Return s.Age >= youngAge;

**}**

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-oGZEjZ)

Example: Multi Statements Lambda Expression VB.Net

Function(s , youngAge)

Console.WriteLine("Lambda expression with multiple statements in the body")

Return s.Age >= youngAge

End Function

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-HA4Mqu)

Declare Local Variable in Lambda Expression Body

You can declare a variable in the expression body to use it anywhere in the expression body, as below:

Example: Local Variable in Lambda expression C#

s =>

{

**int youngAge = 18;**

Console.WriteLine("Lambda expression with multiple statements in the body");

return s.Age >= youngAge;

}

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-S1K7xU)

Example: Local Variable in Lambda Expression VB.Net

Function(s)

**Dim youngAge As Integer = 18**

Console.WriteLine("Lambda expression with multiple statements in the body")

Return s.Age >= youngAge

End Function

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-p76aBt)

Lambda expression can also be assigned to built-in delegates such as [Func](https://www.tutorialsteacher.com/csharp/csharp-func-delegate), [Action](https://www.tutorialsteacher.com/csharp/csharp-action-delegate) and [Predicate](https://www.tutorialsteacher.com/csharp/csharp-predicate).

Assign Lambda Expression to Delegate

The lambda expression can be assigned to Func<in T, out TResult> type delegate. The last parameter type in a Func delegate is the return type and rest are input parameters. Visit [Func delegate](https://www.tutorialsteacher.com/csharp/csharp-func-delegate) section of C# tutorials to know more about it.

Consider the following lambda expression to find out whether a student is a teenager or not.

Example: Lambda Expression Assigned to Func Delegate C#

Func<Student, bool> isStudentTeenAger = s => s.age > 12 && s.age < 20;

Student std = new Student() { age = 21 };

bool isTeen = isStudentTeenAger(std);// returns false

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-MfwAY6)

Example: Lamda Expression Assigned to Func Delegate VB.Net

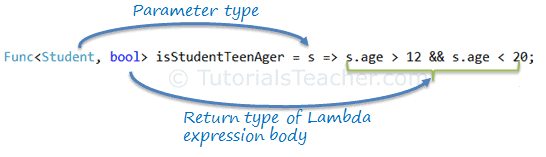
Dim isStudentTeenAger As Func(Of Student, Boolean) = Function(s) s.Age > 12 And s.Age < 20

Dim stud As New Student With {.Age = 21}

Dim isTeen As Boolean = isStudentTeenAger(stud) // returns false

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-dh5cUY)

In the above example, the Func delegate expects the first input parameter to be of Student type and the return type to be boolean. The lambda expression s => s.age > 12 && s.age < 20 satisfies the Func<Student, bool> delegate requirement, as shown below:

[](https://www.tutorialsteacher.com/Content/images/linq/func-with-lambda-expression.png)Func delegate with Lambda Expression

The Func<> delegate shown above, would turn out to be a function as shown below.

bool isStudentTeenAger(Student s)

{

return s.Age > 12 && s.Age < 20;

}

Action Delegate

Unlike the Func delegate, an Action delegate can only have input parameters. Use the [Action delegate](https://www.tutorialsteacher.com/csharp/csharp-action-delegate) type when you don't need to return any value from lambda expression.

Example: Lamda Expression Assigned to Action Delegate C#

Action<Student> PrintStudentDetail = s => Console.WriteLine("Name: {0}, Age: {1} ", s.StudentName, s.Age);

Student std = new Student(){ StudentName = "Bill", Age=21};

PrintStudentDetail(std);//output: Name: Bill, Age: 21

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-TilB9f)

Example: Lamda Expression Assigned to Action Delegate VB.Net

Dim printStudentDetail As Action(Of Student) = Sub(s) Console.WriteLine("Name: {0}, Age: {1} ", s.StudentName, s.Age)

Dim stud As New Student With {.StudentName = "Bill", .Age = 21}

printStudentDetail(stud)//output: Name: Bill, Age: 21

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-nCWoaG)

Lambda Expression in LINQ Query

Usually lambda expression is used with LINQ query. Enumerable static class includes Where extension method for IEnumerable<T> that accepts Func<TSource,bool>. So, the Where() extension method for IEnumerable<Student> collection is required to pass Func<Student,bool>, as shown below:

Func delegate parameter in Where extension method

So now, you can pass the lambda expression assigned to the Func delegate to the Where() extension method in the method syntax as shown below:

Example: Func Delegate in LINQ Method Syntax

IList<Student> studentList = new List<Student>(){...};

Func<Student, bool> isStudentTeenAger = s => s.age > 12 && s.age < 20;

var teenStudents = studentList.Where(isStudentTeenAger).ToList<Student>();

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-tCgBt7)

Example: Func Delegate in LINQ Query Syntax

IList<Student> studentList = new List<Student>(){...};

Func<Student, bool> isStudentTeenAger = s => s.age > 12 && s.age < 20;

var teenStudents = from s in studentList

where isStudentTeenAger(s)

select s;

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-VmWuAS)

You can follow the same method in VB.Net to pass Func delegate.

 Points to Remember :

1. Lambda Expression is a shorter way of representing anonymous method.
2. Lambda Expression syntax: *parameters => body expression*
3. Lambda Expression can have zero parameter.
4. Lambda Expression can have multiple parameters in parenthesis ().
5. Lambda Expression can have multiple statements in body expression in curly brackets {}.
6. Lambda Expression can be assigned to Func, Action or Predicate delegate.
7. Lambda Expression can be invoked in a similar way to delegate.