



153



15



A lambda expression is a convenient way of defining an anonymous (unnamed) function that can be passed around as a variable or as a parameter to a method call. Many LINQ methods take a function (called a **delegate**) as a parameter. Here is an example of what a lambda expression looks like:

```
Func<int, int> multiplyByFive = num => num * 5;  
// Returns 35  
int result = multiplyByFive(7);
```

The expression `num => num * 5` is a lambda expression. The `=>` operator is called the "lambda operator". In this example, `num` is an input parameter to the anonymous function, and the return value of this function is `num * 5`. So when `multiplyByFive` is called with a parameter of `7`, the result is `7 * 5`, or `35`.

Parameter(s)

Notice that the `num` parameter doesn't explicitly specify a data type. The compiler always infers the data type of lambda expression parameters from context. In this case, the context is that the lambda expression is stored in a variable of type `Func<int, int>`. This means that it takes an `int` parameter and returns an `int` result.

You can also create lambda expressions with more than one parameter, as shown here:

```
Func<int, int, int> multiplyTwoNumbers = (a, b) =>  
a * b;  
// Returns 35
```