Лямбда-выражения в C#

Докладчик: **Кофнов Олег Владимирович**,
слушатель курса ЦПСМИ математико-механического факультета СПбГУ

Пример λ-выражения

```
string[] words = { "cherry", "apple","blueberry" };
int shortestWord = words.Min(w => w.Length);

/* При исполнении данного кода переменной
   shortestWord будет присвоено значение
   длины самого короткого слова из массива
   words */
```

Лямбда-исчисление

Ля́мбда-исчисле́ние (*λ-исчисление, лямбда-исчисление*) — формальная система, разработанная американским математиком **Алонзо Чёрчем**.

В основу λ-исчисления положены две фундаментальные операции: аппликация и абстракция. Аппликация означает применение или вызов функции по отношению к заданному значению.

Абстракция или λ-абстракция в свою очередь строит функции по заданным выражениям.

Пример: выражение $\lambda x.2 \bullet x + 1$ обозначает функцию, ставящую в соответствие каждому x значение $2 \bullet x + 1$,

$(\lambda x.2 \bullet x + 1)\ 3$ -- означает, что в терм $2 \bullet x + 1$ вместо переменной x подставляется **3** и вычисляется выражение **2 • 3 + 1 = 7**

Лямбда-выражение (C#, .Net)

Лямбда-выражение — это анонимная функция, которая содержит выражения и операторы и может использоваться для создания делегатов или типов дерева выражений.

Анонимная функция (анонимный метод) – функция (метод), не имеющая идентификатора (имени) и объявленная прямо в месте использования. Анонимные методы были представлены в C# 2.0, а в версиях C# 3.0 и более поздних лямбда-выражения заменяют эти методы и являются предпочтительным способом написания встроенного кода.

Пример анонимного метода:

```
// Обработчик клика мышки
button1.Click += delegate(System.Object o, System.EventArgs e)
                { System.Windows.Forms.MessageBox.Show("Click!"); };
```

Лямбда-оператор =>

Во всех лямбда-выражениях используется лямбда-оператор =>, который читается как "переходит в". Левая часть лямбда-оператора определяет параметры ввода (если таковые имеются), а правая часть содержит выражение или блок оператора.

Оператор => имеет тот же приоритет, что и оператор присваивания (=) и является правоассоциативным, то есть сначала выполняются выражения справа от оператора, а затем он сам.

Лямбда-выражение x => x * x

```
delegate int del(int i);            //Объявили тип del - делегат
del myDelegate = x => x * x;//Объявили переменную-делегат
                        // и присвоили ей значение –                    //
    лямбда-выражение
int j = myDelegate(5);          //Результат: j = 25
/*
```

Внимание! Для myDelegate и любой другой переменной типа del аргумент может быть только одно число типа int и возвращаемое значение также будет типа int. Это требование относится ко всем переменным-выражениям типа del.

```
*/
```

λ-операторы в запросах LINQ

LINQ – язык интегрированных запросов, технология, позволяющая использовать синтаксис языка последовательных запросов (SQL) в языках платформы .Net Framework.

Лямбда-операторы используются в запросах LINQ на основе методов в качестве аргументов стандартных методов операторов запроса, таких как **Enumerable.Where** и **Queryable.Where**.

Пример λ-выражения в LINQ-запросе

```
class SimpleLambda
{
    static void Main()
    {
        int[] scores = { 90, 71, 82, 93, 75, 82 };
/* Подсчет количества элементов массива scores, значение
     которых больше восьмидесяти */
        int highScoreCount = scores.Where(n => n > 80).Count();

        Console.WriteLine("{0} scores are greater than 80",
     highScoreCount);
        // Результат: 4 scores are greater than 80
    }
}
```

Допустимый синтаксис

```
(input parameters) => expression    //1
(x, y) => x == y                    //2
(int x, string s) => s.Length > x   //3
() => SomeMethod()                  //4
(input parameters) => {statement;}  //5
```

```
delegate void TestDelegate(string s);
TestDelegate myDel = n => { string s = n + " " + "World";    //6
        Console.WriteLine(s); };
myDel("Hello");
```

Вывод типа в λ-выражениях

При написании лямбда-выражений обычно не требуется указывать тип параметров ввода, поскольку компилятор может вывести этот тип на основе тела лямбда-выражения. Для большинства стандартных операторов запроса первый ввод имеет тип элементов в исходной последовательности.

```
Customer[] customers = new Customer[100];
…
/* код заполнения массива объектами */
…
customers.Where(c => c.City == "London");
/* переменная ввода расценивается как объект Customer */
```

Используются следующие основные правила для лямбда-выражений:

- Лямбда-выражение должно содержать то же число параметров, что и тип делегата.
- Каждый параметр ввода в лямбда-выражении должен быть неявно преобразуемым в соответствующий параметр делегата.
- Возвращаемое значение лямбда-выражения (если таковое имеется) должно быть неявно преобразуемым в возвращаемый тип делегата.

Область действия переменной в лямбда-выражениях

Лямбда-выражения могут ссылаться на внешние переменные, попадающие в область действия включающего их метода или типа, в котором определено это выражение.

Переменные, захваченные таким способом, сохраняются для использования в лямбда-выражениях даже в том случае, если эти переменные иначе попадают за границы области действия и уничтожаются сборщиком мусора.

Внешняя переменная должна быть определенным образом назначена, прежде чем она сможет использоваться в лямбда-выражениях.

Следующие правила применимы к области действия переменной в лямбда-выражениях:

Захваченная переменная не будет уничтожена сборщиком мусора до тех пор, пока делегат, который на нее ссылается, не выйдет за границы области.

Переменная, введенная в лямбда-выражение, невидима во внешнем методе.

Лямбда-выражение не может непосредственно захватывать параметры ref или out из включающего их метода.

Оператор Return в лямбда-выражении не вызывает возвращение значения методом.

Лямбда-выражение не может содержать оператор goto, оператор break или оператор continue, целевой объект которого находится вне тела либо в теле содержащейся анонимной функции.

Источники:

- http://www.wikipedia.org
- http://msdn.microsoft.com/ru-ru/library
- http://www.rsdn.ru/article/dotnet/cslambda.xml
- http://codedev.ru/2009/12/21/csharp-introduction-to-lambda/
- http://www.cyberguru.ru/dotnet/csharp-net/linq-operators-page4.html