Lambda expressions are how anonymous functions are created.  
  
Lambda expressions are anonymous functions that contain expressions or sequence of operators. All lambda expressions use the lambda operator =>, that can be read as “goes to” or “becomes”.   
  
The left side of the lambda operator specifies the input parameters and the right side holds an expression or a code block that works with the entry parameters. Usually lambda expressions are used as predicates or instead of delegates (a type that references a method).  
  
**Expression Lambdas**

Parameter => expressionParameter-list => expressionCount => count + 2;Sum => sum + 2;n => n % 2 == 0

The lambda operator => divides a lambda expression into two parts. The left side is the input parameter and the right side is the lambda body.

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. public static class demo
5. {
6. public static void Main()
7. {
8. List<int> list = new List<int>() { 1, 2, 3, 4, 5, 6 };
9. List<int> evenNumbers = list.FindAll(x => (x % 2) == 0);
11. foreach (var num in evenNumbers)
12. {
13. Console.Write("{0} ", num);
14. }
15. Console.WriteLine();
16. Console.Read();
18. }
19. }

The preceding example loops through the entire collection of numbers and each element (named x) is checked to determine if the number is a multiple of 2 (using the Boolean expression (x % 2) == 0).

using System;

1. using System.Collections.Generic;
2. using System.Linq;
3. class Dog
4. {
5. public string Name { get; set; }
6. public int Age { get; set; }
7. }
8. class demo{
9. static void Main()
10. {
11. List<Dog> dogs = new List<Dog>() {
12. new Dog { Name = "Rex", Age = 4 },
13. new Dog { Name = "Sean", Age = 0 },
14. new Dog { Name = "Stacy", Age = 3 }
15. };
16. var names = dogs.Select(x => x.Name);
17. foreach (var name in names)
18. {
19. Console.WriteLine(name);
21. }
22. Console.Read();
23. }
24. }

**Output**

*Rex  
Sean  
Stacy*

**Using Lambda Expressions with Anonymous Types**

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. class Dog
5. {
6. public string Name { get; set; }
7. public int Age { get; set; }
8. }
9. class demo{
10. static void Main()
11. {
12. List<Dog> dogs = new List<Dog>() {
13. new Dog { Name = "Rex", Age = 4 },
14. new Dog { Name = "Sean", Age = 0 },
15. new Dog { Name = "Stacy", Age = 3 }
16. };
17. var newDogsList = dogs.Select(x => new { Age = x.Age, FirstLetter = x.Name[0] });
18. foreach (var item in newDogsList)
19. {
20. Console.WriteLine(item);
21. }
22. Console.Read();
23. }
24. }

**Sorting**

The following is an examle of sorting with a lambda expression:

1. var sortedDogs = dogs.OrderByDescending(x => x.Age);
2. foreach (var dog in sortedDogs)
3. {
4. Console.WriteLine(string.Format("Dog {0} is {1} years old.", dog.Name, dog.Age));
5. }

/Get the average of the odd Fibonacci numbers in the series...

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace lambdaexample {

class Program {

static void Main(string[] args) {

int[] fibNum = { 1, 1, 2, 3, 5, 8, 13, 21, 34 };

double averageValue = fibNum.Where(num ⇒ num % 2 == 1).Average();

Console.WriteLine(averageValue);

Console.ReadLine();

}

}

}

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Linq.Expressions;

namespace lambdaexample {

class Program {

static void Main(string[] args) {

int[] source = new[] { 3, 8, 4, 6, 1, 7, 9, 2, 4, 8 };

foreach (int i in source.Where(x ⇒

{

if (x <= 3)

return true;

else if (x >= 7)

return true;

return false;

}

))

Console.WriteLine(i);

Console.ReadLine();

}

}

}

* A lambda expression can return a value and may have parameters.
* Parameters can be defined in a myriad of ways with a lambda expression.
* If there is single statement in a lambda expression, there is no need of curly brackets whereas if there are multiple statements, curly brackets as well as return value are essential to write.
* With lambda expressions, it is possible to access variables present outside of the lambda expression block by a feature known as closure. Use of closure should be done cautiously to avoid any problem.
* It is impossible to execute any unsafe code inside any lambda expression.
* Lambda expressions are not meant to be used on the operator’s left side.