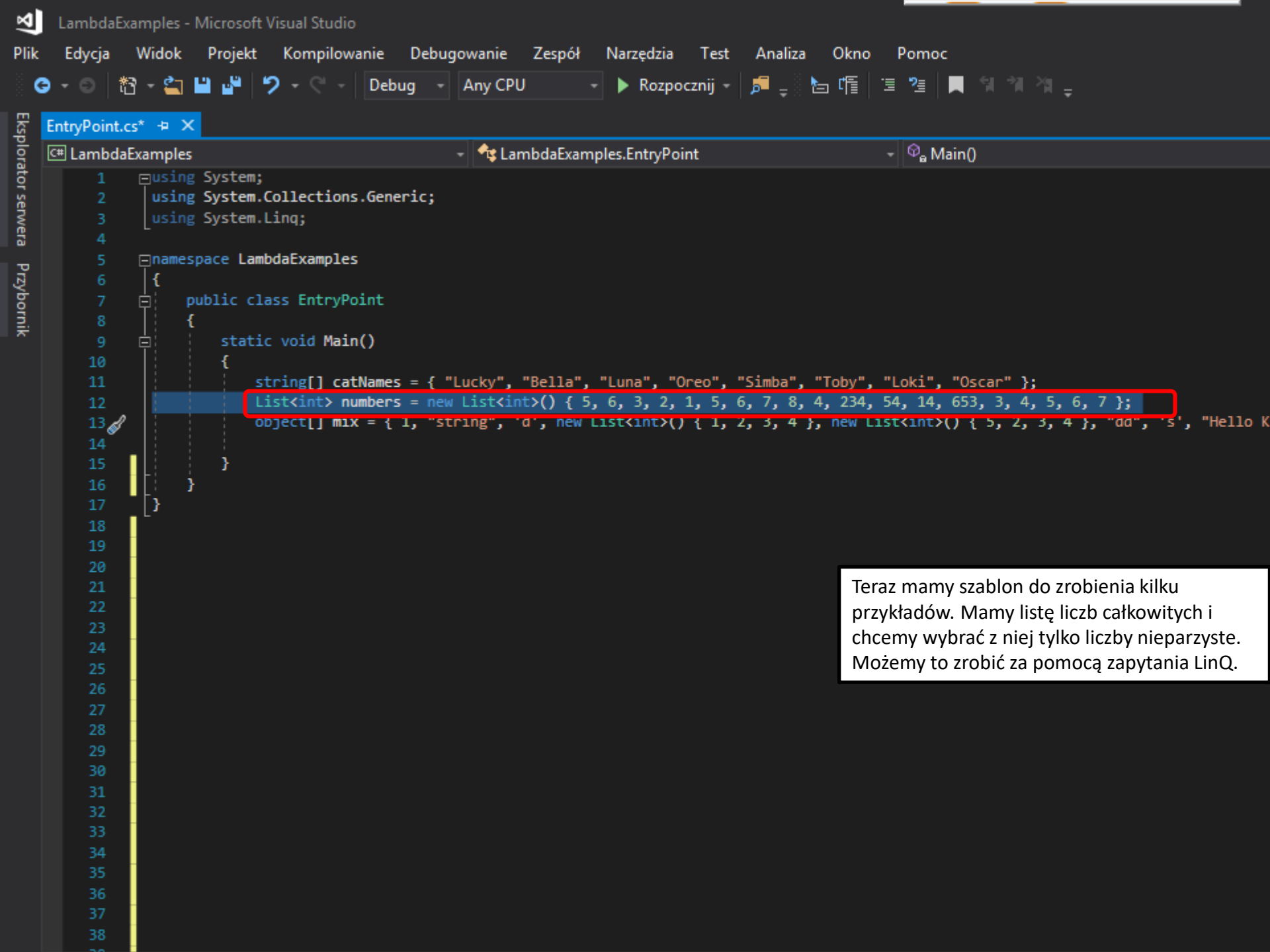
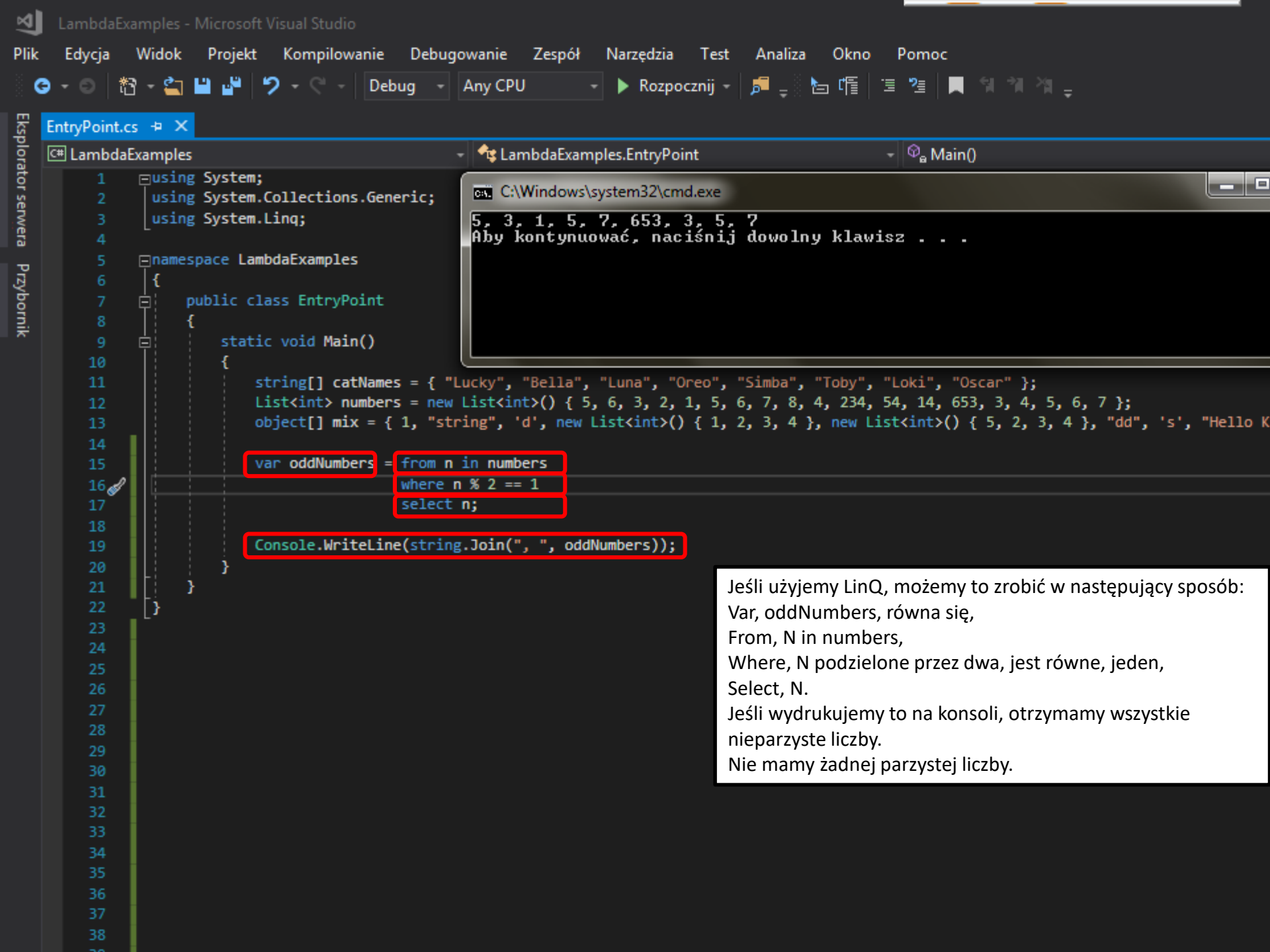
# Lambda expressions

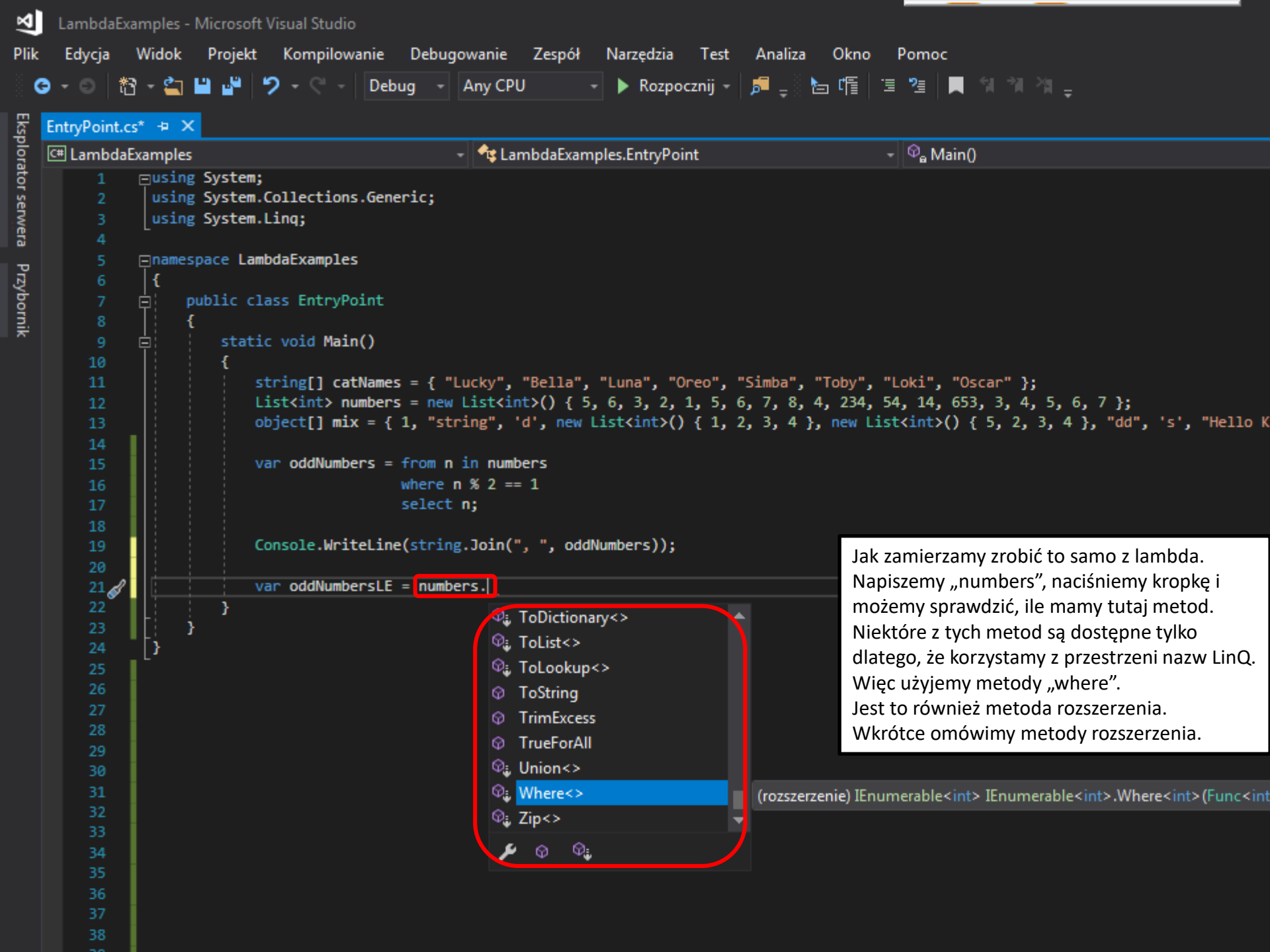
- Denoted by the lambda operator =>
- Input => (work on the input);
- N => ((N%2) == 1);

Operator lambda, który jest strzałką „znak równości i większy niż”.
Służy do oddzielenia wartości wejściowych po lewej stronie strzałki i treści kodu po prawej stronie.
Zasadniczo mamy wejście po lewej stronie i trochę kodu, który będzie działał na tym wejściu.
To po prostu metoda bez dekoracji.
Pozwala ci stworzyć twoją metodę w tym samym miejscu, w którym będziesz używać, jeśli trzeba jej użyć tylko raz.
Oszczędza to wysiłku pisania oddzielnej metody.
I jest tu prosty przykład dotyczący wyrażenia lambda.
Mamy jedno wejście, które jest tylko zmienną N. Powiedzmy, że to liczba.
Sprawdzamy, czy N podzielony przez dwa ma resztę 1, a następnie zwrócimy pewną wartość do zmiennej wyrażenia lambda.
Sprawdzamy, czy dana liczba dzieli się przez dwa z resztą, czy bez reszty.
Jeśli otrzymamy 0, jest to liczba parzysta.
Jeśli otrzymamy 1, jest to liczba nieparzysta.
Zasadniczo sprawdzamy, czy jest to liczba nieparzysta.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs*
LambdaExamples
LambdaExamples.EntryPoint
Main()
Eksplorator serwera
Przybornik
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4
5 namespace LambdaExamples
6 {
7 public class EntryPoint
8 {
9 static void Main()
10 {
11 string[] catNames = { "Lucky", "Bella", "Luna", "Oreo", "Simba", "Toby", "Loki", "Oscar" };
12 List<int> numbers = new List<int>() { 5, 6, 3, 2, 1, 5, 6, 7, 8, 4, 234, 54, 14, 653, 3, 4, 5, 6, 7 };
13 object[] mix = { 1, "string", 'd', new List<int>() { 1, 2, 3, 4 }, new List<int>() { 5, 2, 3, 4 }, "dd", 's', "Hello K
14
15 }
16 }
17 }
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
Teraz mamy szablon do zrobienia kilku przykładów. Mamy listę liczb całkowitych i chcemy wybrać z niej tylko liczby nieparzyste. Możemy to zrobić za pomocą zapytania LinQ.




Jeśli użyjemy LinQ, możemy to zrobić w następujący sposób:
Var, oddNumbers, równa się,
From, N in numbers,
Where, N podzielone przez dwa, jest równe, jeden,
Select, N.
Jeśli wydrukujemy to na konsoli, otrzymamy wszystkie nieparzyste liczby.
Nie mamy żadnej parzystej liczby.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs*
LambdaExamples
LambdaExamples.EntryPoint
Main()
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
string[] catNames = { "Lucky", "Bella", "Luna", "Oreo", "Simba", "Toby", "Loki", "Oscar" };
List<int> numbers = new List<int>() { 5, 6, 3, 2, 1, 5, 6, 7, 8, 4, 234, 54, 14, 653, 3, 4, 5, 6, 7 };
object[] mix = { 1, "string", 'd', new List<int>() { 1, 2, 3, 4 }, new List<int>() { 5, 2, 3, 4 }, "dd", 's', "Hello K
var oddNumbers = from n in numbers
where n % 2 == 1
select n;
Console.WriteLine(string.Join(", ", oddNumbers));
var oddNumbersLE = numbers.
ToDictionary<>
ToList<>
ToLookup<>
ToString
TrimExcess
TrueForAll
Union<>
Where<>
Zip<>
(rozszerzenie) IEnumerable<int> IEnumerable<int>.Where<int>(Func<int
Jak zamierzamy zrobić to samo z lambda.
Napiszemy „numbers”, naciśniemy kropkę i
możemy sprawdzić, ile mamy tutaj metod.
Niektóre z tych metod są dostępne tylko
dlatego, że korzystamy z przestrzeni nazw LinQ.
Więc użyjemy metody „where”.
Jest to również metoda rozszerzenia.
Wkrótce omówimy metody rozszerzenia.

LambdaExamples - Microsoft Visual Studio

Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc

Debug | Any CPU | Rozpocznij

Eksplorator serwera | Przybornik

EntryPoint.cs

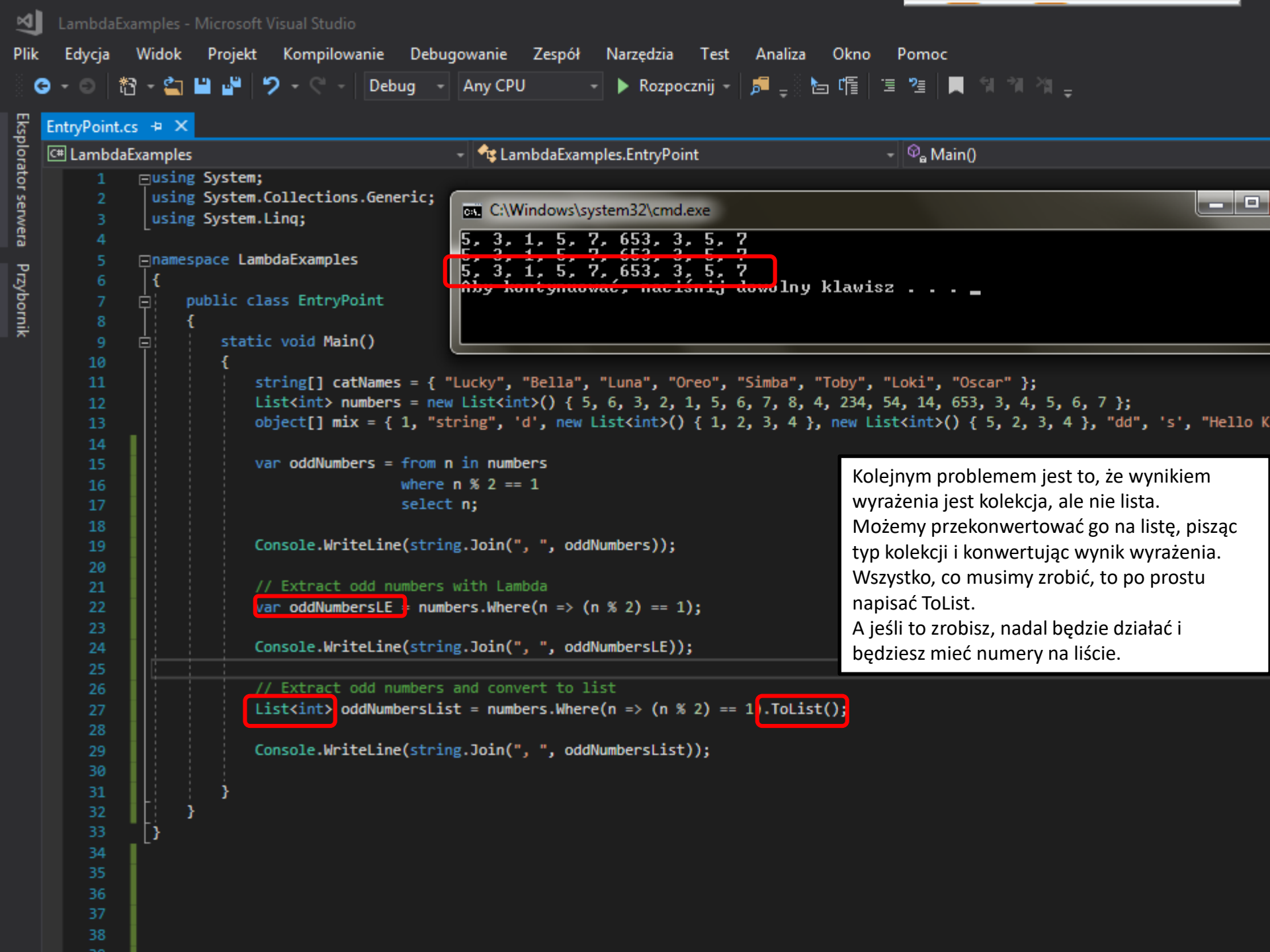
LambdaExamples | LambdaExamples.EntryPoint | Main()

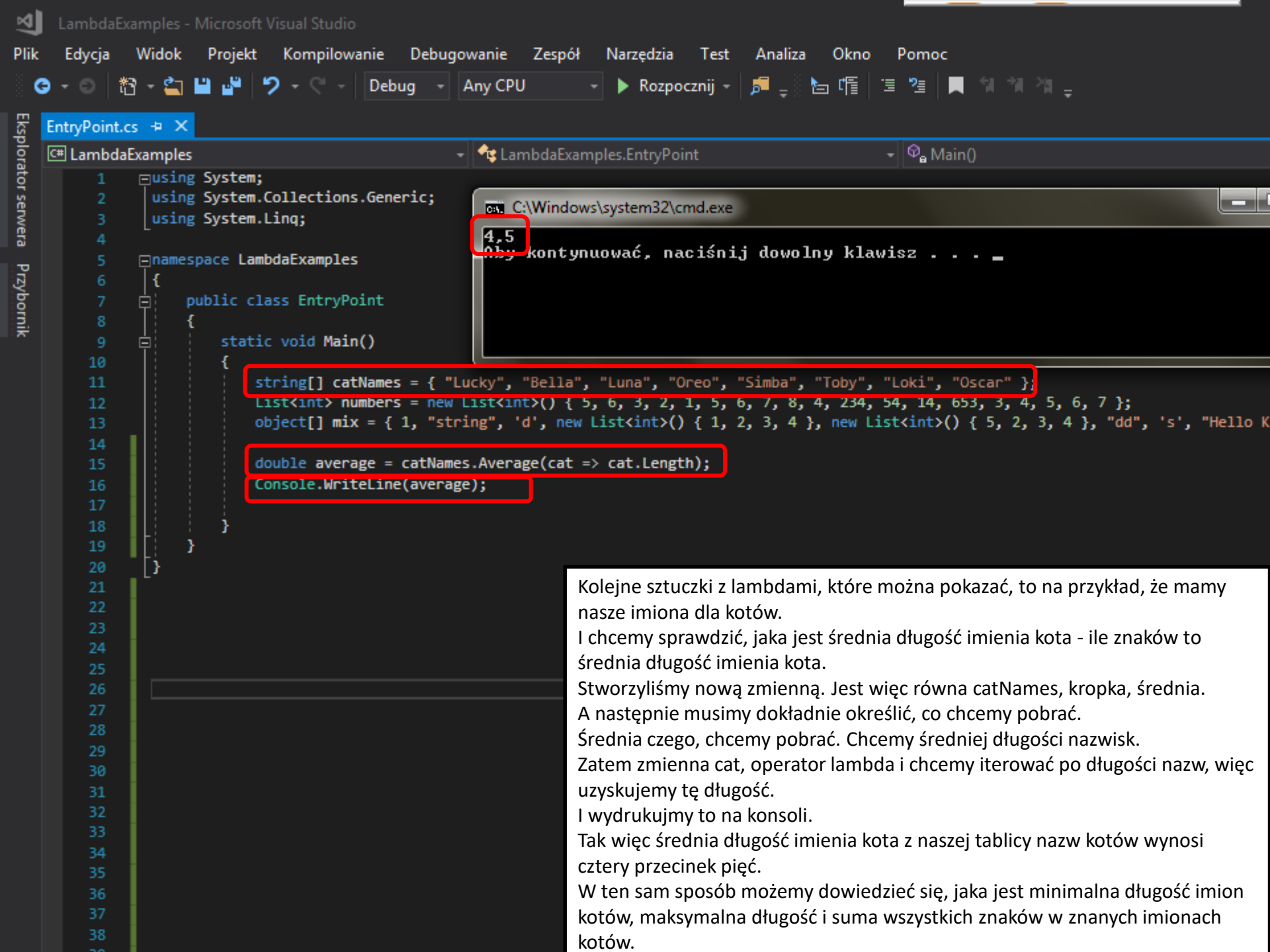
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4
5  namespace LambdaExamples
6  {
7      public class EntryPoint
8      {
9          static void Main()
10         {
11             string[] catNames = { "Lucky", "Bella", "Luna", "Oreo", "Simba", "Toby", "Loki", "Oscar" };
12             List<int> numbers = new List<int>() { 5, 6, 3, 2, 1, 5, 6, 7, 8, 4, 234, 54, 14, 653, 3, 4, 5, 6, 7 };
13             object[] mix = { 1, "string", 'd', new List<int>() { 1, 2, 3, 4 }, new List<int>() { 5, 2, 3, 4 }, "dd", 's', "Hello K
14
15             var oddNumbers = from n in numbers
16                              where n % 2 == 1
17                              select n;
18
19             Console.WriteLine(string.Join(", ", oddNumbers));
20
21             // Extract odd numbers with Lambda
22             var oddNumbersLE = numbers.Where(n => (n % 2) == 1);
23
24             Console.WriteLine(st .Join(" oddNumb E));
25
26         }
27     }
28 }
29
30
31
32
33
34
35
36
37
38
```

C:\Windows\system32\cmd.exe

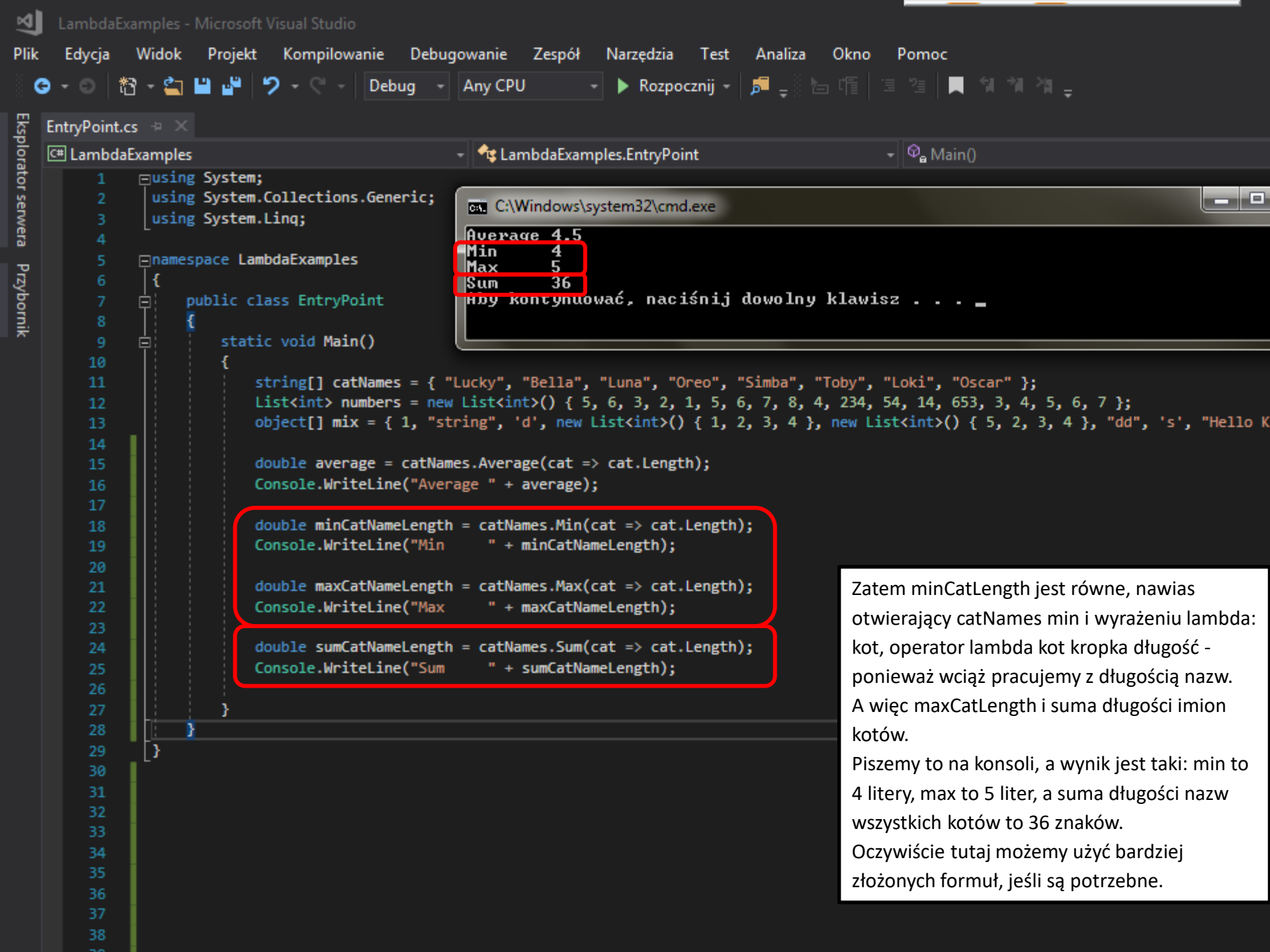
```
5, 3, 1, 5, 7, 653, 3, 5, 7
5, 3, 1, 5, 7, 653, 3, 5, 7
Aby kontynuować, naciśnij dowolny klawisz . . .
```

Po prostu stwórzmy nowe wyrażenie lambda.
Powiemy: GDZIE, N, operator lambda, N podzielony przez dwa ma resztę 1. W porządku. Jaki jest rezultat naszej pracy?
Jak to działa? N jest zmienną iteracyjną; podobnie jak w zapytaniu LinQ; podobnie jak w przypadku pętli ForEach.
I będzie iterować w tablicy liczb.
Więc cokolwiek mamy jako kolekcję - zadziała na niej wyrażenie lambda. Po lewej stronie operatora lambda mamy iterator dla tej kolekcji.
Działa dokładnie tak samo, jak w zapytaniu LinQ w przykładzie. Pobiera każdą liczbę za pomocą zmiennej i sprawdza, czy pasuje do naszych warunków.
A jeśli to prawda, dodaje ją do naszej nowej kolekcji.
Więc jeszcze bardziej redukujemy kod.
Dzięki pętli ForEach robimy to w siedmiu-ośmiu wierszach.
Z LinQ robimy to w trzech wierszach.
Z lambda robimy to w jednym wierszu kodu.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs
LambdaExamples
LambdaExamples.EntryPoint
Main()
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
string[] catNames = { "Lucky", "Bella", "Luna", "Oreo", "Simba", "Toby", "Loki", "Oscar" };
List<int> numbers = new List<int>() { 5, 6, 3, 2, 1, 5, 6, 7, 8, 4, 234, 54, 14, 653, 3, 4, 5, 6, 7 };
object[] mix = { 1, "string", 'd', new List<int>() { 1, 2, 3, 4 }, new List<int>() { 5, 2, 3, 4 }, "dd", 's', "Hello K
var oddNumbers = from n in numbers
where n % 2 == 1
select n;
Console.WriteLine(string.Join(", ", oddNumbers));
// Extract odd numbers with Lambda
var oddNumbersLE = numbers.Where(n => (n % 2) == 1);
Console.WriteLine(string.Join(", ", oddNumbersLE));
// Extract odd numbers and convert to list
List<int> oddNumbersList = numbers.Where(n => (n % 2) == 1).ToList();
Console.WriteLine(string.Join(", ", oddNumbersList));
}
}
}
C:\Windows\system32\cmd.exe
5, 3, 1, 5, 7, 653, 3, 5, 7
5, 3, 1, 5, 7, 653, 3, 5, 7
Kolejnym problemem jest to, że wynikiem wyrażenia jest kolekcja, ale nie lista.
Możemy przekonwertować go na listę, pisząc typ kolekcji i konwertując wynik wyrażenia.
Wszystko, co musimy zrobić, to po prostu napisać ToList.
A jeśli to zrobisz, nadal będzie działać i będziesz mieć numery na liście.



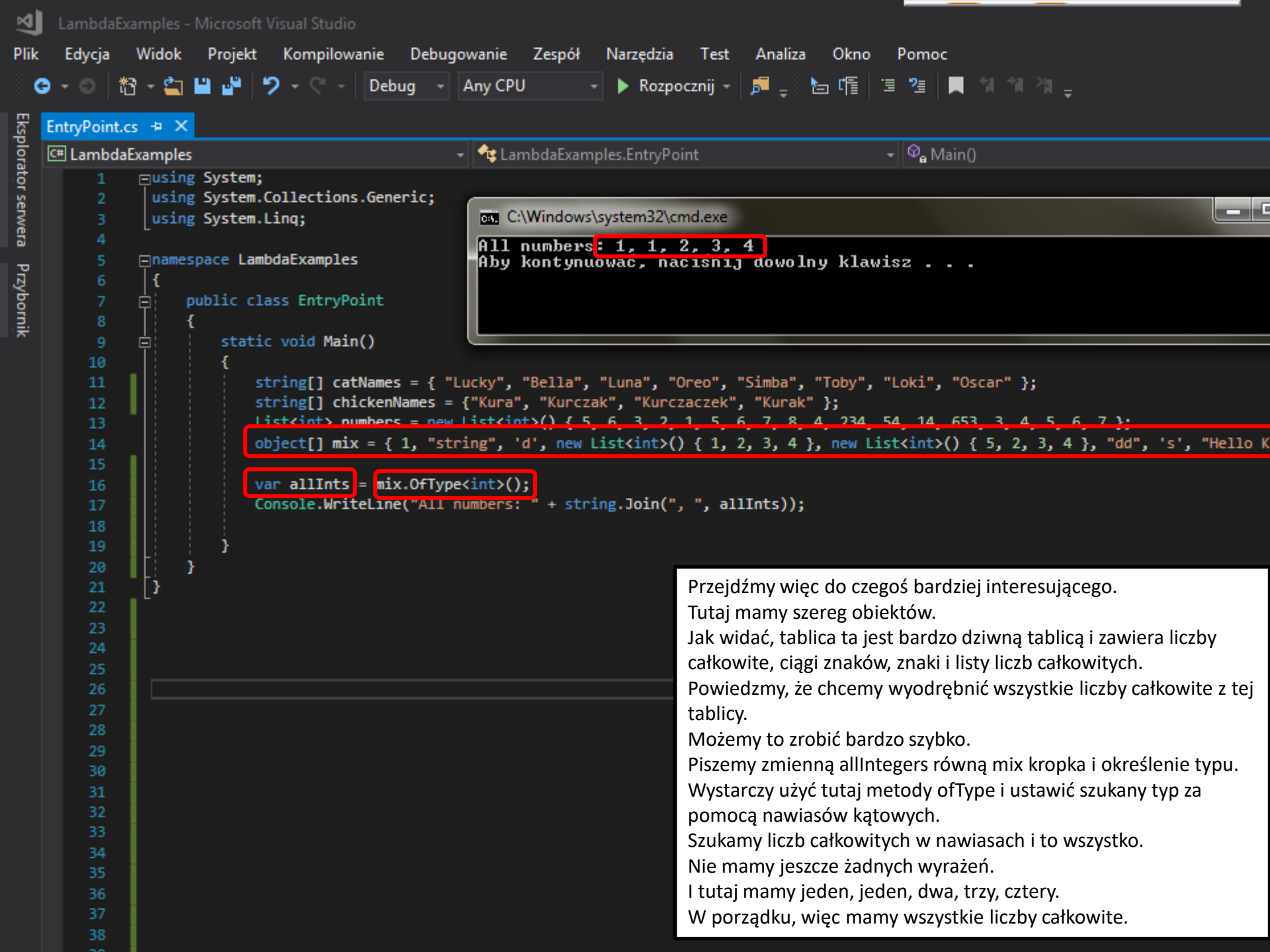

Kolejne sztuczki z lambdami, które można pokazać, to na przykład, że mamy nasze imiona dla kotów.
I chcemy sprawdzić, jaka jest średnia długość imienia kota - ile znaków to średnia długość imienia kota.
Stworzyliśmy nową zmienną. Jest więc równa catNames, kropka, średnia.
A następnie musimy dokładnie określić, co chcemy pobrać.
Średnia czego, chcemy pobrać. Chcemy średniej długości nazwisk.
Zatem zmienna cat, operator lambda i chcemy iterować po długości nazw, więc uzyskujemy tę długość.
I wydrukujmy to na konsoli.
Tak więc średnia długość imienia kota z naszej tablicy nazw kotów wynosi cztery przecinek pięć.
W ten sam sposób możemy dowiedzieć się, jaka jest minimalna długość imion kotów, maksymalna długość i suma wszystkich znaków w znanych imionach kotów.



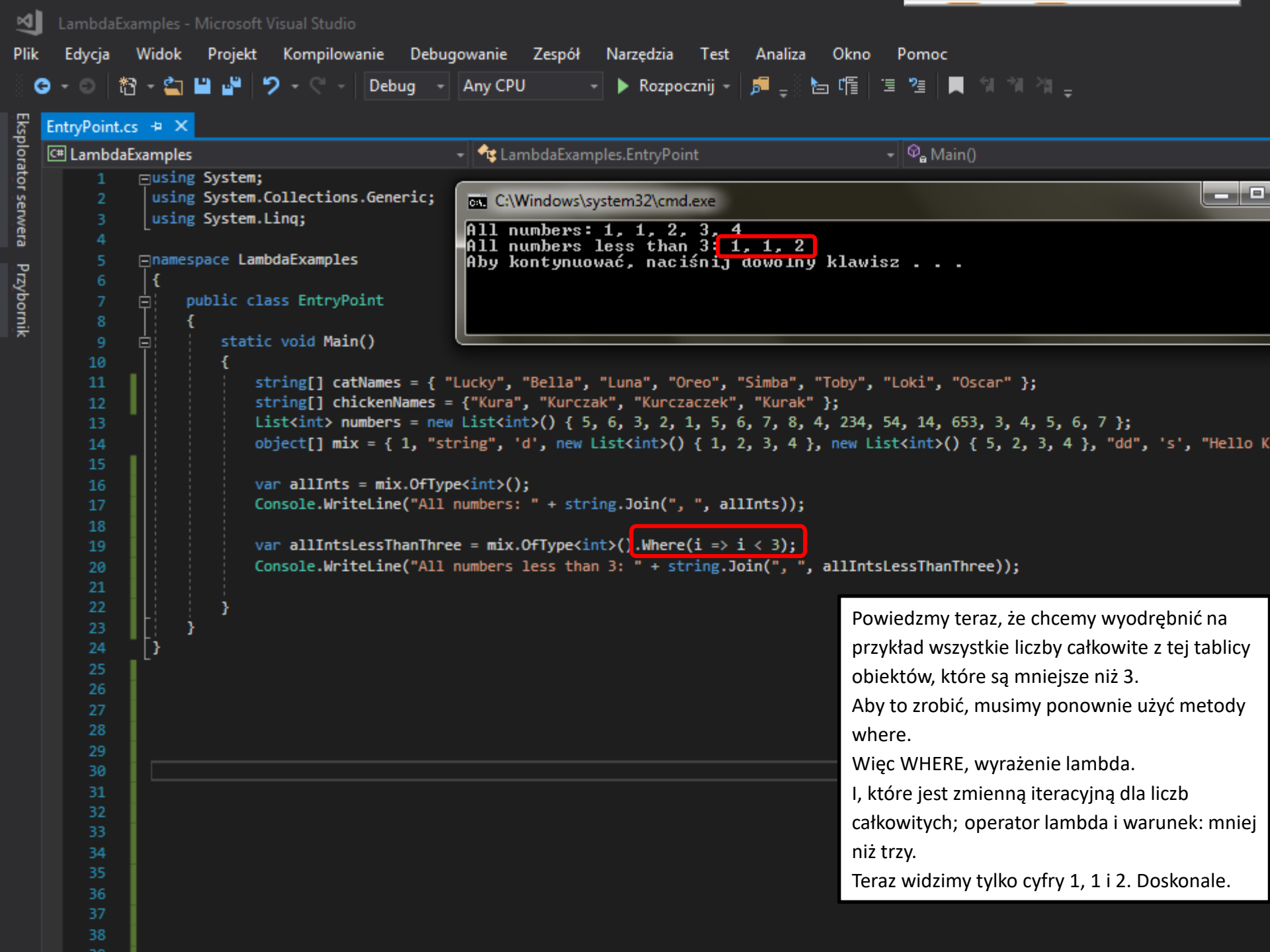

Zatem minCatLength jest równe, nawias otwierający catNames min i wyrażeniu lambda: kot, operator lambda kot kropka długość - ponieważ wciąż pracujemy z długością nazw. A więc maxCatLength i suma długości imion kotów.

Piszemy to na konsoli, a wynik jest taki: min to 4 litery, max to 5 liter, a suma długości nazw wszystkich kotów to 36 znaków.

Oczywiście tutaj możemy użyć bardziej złożonych formuł, jeśli są potrzebne.




Przejdźmy więc do czegoś bardziej interesującego.
Tutaj mamy szereg obiektów.
Jak widać, tablica ta jest bardzo dziwną tablicą i zawiera liczby całkowite, ciągi znaków, znaki i listy liczb całkowitych.
Powiedzmy, że chcemy wyodrębnić wszystkie liczby całkowite z tej tablicy.
Możemy to zrobić bardzo szybko.
Piszemy zmienną allIntegers równą mix kropka i określenie typu.
Wystarczy użyć tutaj metody ofType i ustawić szukany typ za pomocą nawiasów kątowych.
Szukamy liczb całkowitych w nawiasach i to wszystko.
Nie mamy jeszcze żadnych wyrażeń.
I tutaj mamy jeden, jeden, dwa, trzy, cztery.
W porządku, więc mamy wszystkie liczby całkowite.




Powiedzmy teraz, że chcemy wyodrębnić na przykład wszystkie liczby całkowite z tej tablicy obiektów, które są mniejsze niż 3.
Aby to zrobić, musimy ponownie użyć metody where.
Więc WHERE, wyrażenie lambda.
I, które jest zmienną iteracyjną dla liczb całkowitych; operator lambda i warunek: mniej niż trzy.
Teraz widzimy tylko cyfry 1, 1 i 2. Doskonale.

LambdaExamples - Microsoft Visual Studio

Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc

Debug | Any CPU | Rozpocznij

Eksplorator serwera | Przybornik

EntryPoint.cs

LambdaExamples | LambdaExamples.EntryPoint | Main()

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4
5  namespace LambdaExamples
6  {
7      public class EntryPoint
8      {
9          static void Main()
10         {
11             string[] catNames = { "Lucky", "Bella", "Luna", "Oreo", "Simba", "Toby", "Loki", "Oscar" };
12             string[] chickenNames = {"Kura", "Kurczak", "Kurczaczek", "Kurak" };
13             List<int> numbers = new List<int>() { 5, 6, 3, 2, 1, 5, 6, 7, 8, 4, 234, 54, 14, 653, 3, 4, 5, 6, 7 };
14             object[] mix = { 1, "string", 'd', new List<int>() { 1, 2, 3, 4 }, new List<int>() { 5, 2, 3, 4 }, "dd", 's', "Hello K
15
16             var allInts = mix.OfType<int>();
17             Console.WriteLine("All numbers: " + string.Join(", ", allInts));
18
19             var allIntsLessThanThree = mix.OfType<int>().Where(i => i < 3);
20             Console.WriteLine("All numbers less than 3: " + string.Join(", ", allIntsLessThanThree));
21
22             var containsIntLists = mix.OfType<List<int>>().ToList();
23             for (int i = 0; i < containsIntLists.Count; i++)
24             {
25                 Console.WriteLine($"Int list[{i}]: " + string.Join(", ", containsIntLists[i]));
26             }
27
28         }
29     }
30 }
31
32
33
34
35
36
37
38
```

C:\Windows\system32\cmd.exe

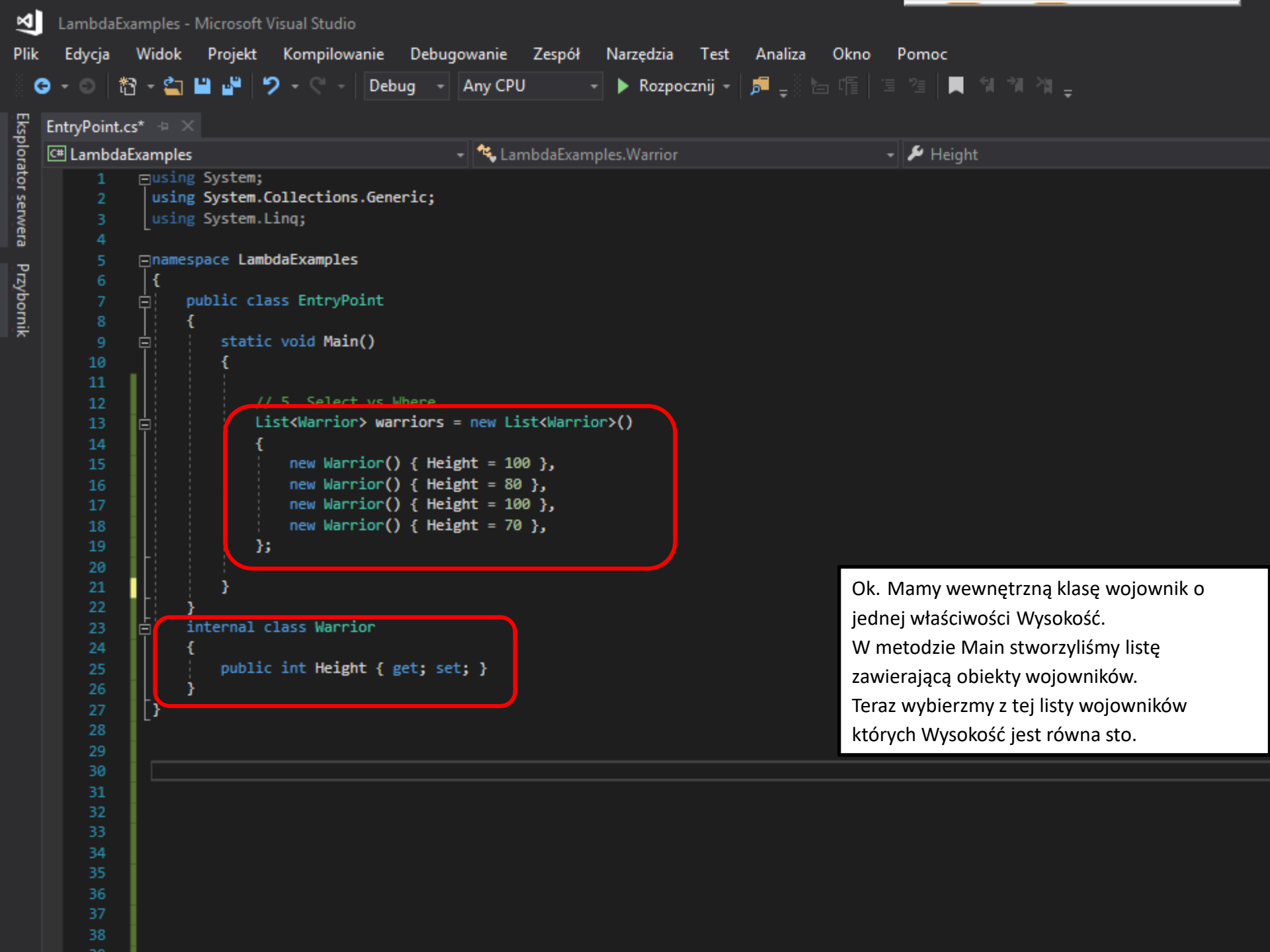
```
All numbers: 1, 1, 2, 3, 4
All numbers less than 3: 1, 1, 2
Int list[0]: 1, 2, 3, 4
Int list[1]: 5, 2, 3, 4
Aby kontynuować, naciśnij dowolny klawisz . . .
```

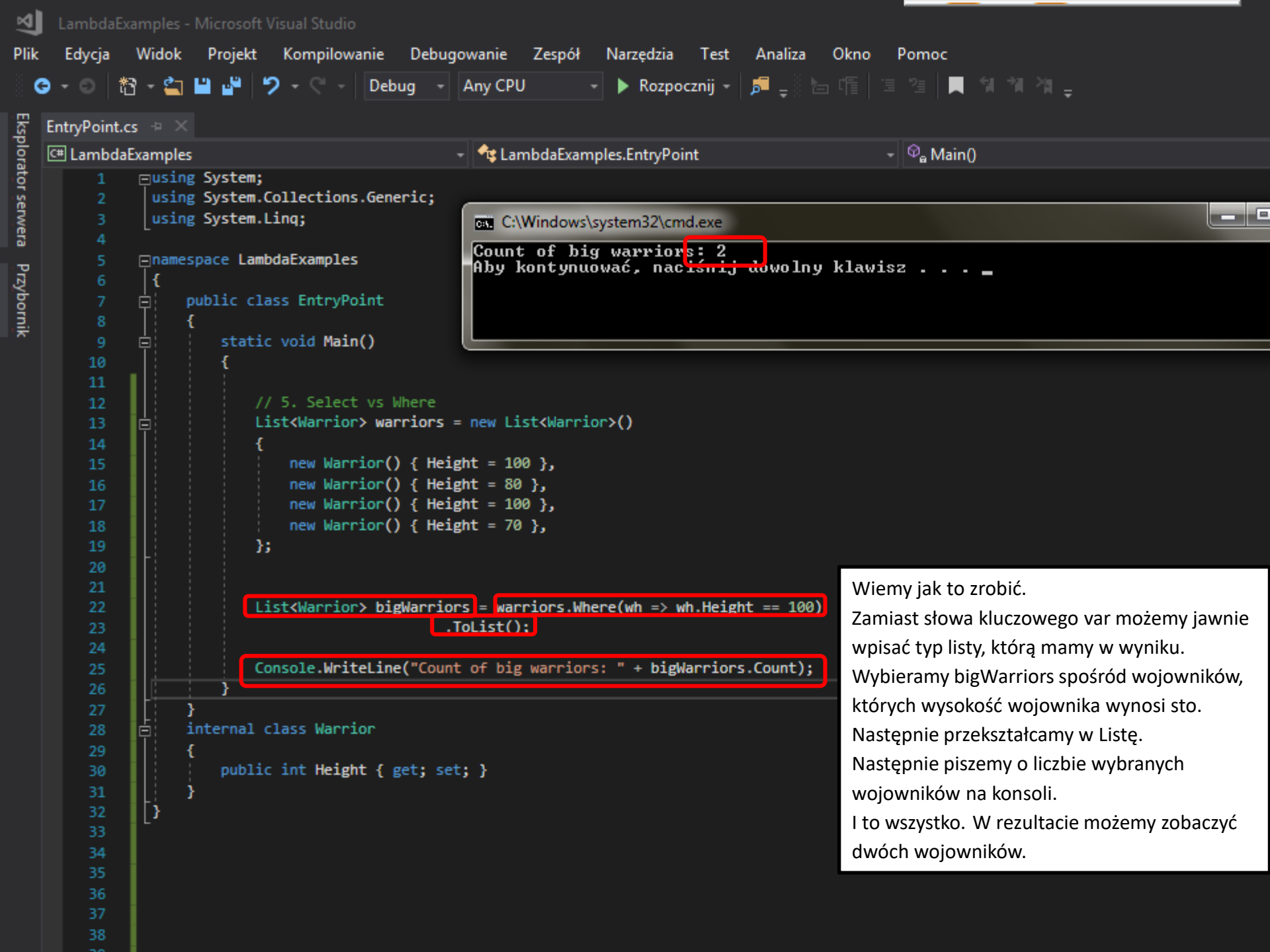
Ostatnim przykładem, jaki zrobimy, jest wyodrębnienie list liczb całkowitych.
Więc zrobimy to w następujący sposób: Var containsIntLists równe mix.
Więc szukamy typów, które są listami liczb całkowitych typów, więc piszemy w nawiasach kątowych: lista liczb całkowitych typów. Więc teraz mamy trochę więcej nawiasów kątowych.
Potem napiszemy toList, ponieważ chcemy mieć możliwość korzystania z tych list. Teraz wyprowadzimy zawartość zmiennej containsIntLists.
Piszemy: pętla For i iterujemy od 0 do rozmiaru naszej listy.
Do każdej listy zapisujemy numer listy i jej zawartość za pomocą JOIN z klasy string.
Rezultatem są dwie listy: Lista zero z elementami: 1, 2, 3, 4 i lista pierwsza z elementami 5, 2, 3, 4.

# Select method

Difference between *where* and *select* methods

Porozmawiamy o różnicy między metodami select a where.
Pokażmy więc różnicę na prostym przykładzie.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs*
LambdaExamples
LambdaExamples.Warrior
Height
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
// 5. Select vs Where
List<Warrior> warriors = new List<Warrior>()
{
new Warrior() { Height = 100 },
new Warrior() { Height = 80 },
new Warrior() { Height = 100 },
new Warrior() { Height = 70 },
};
}
}
internal class Warrior
{
public int Height { get; set; }
}
}
Ok. Mamy wewnętrzną klasę wojownik o jednej właściwości Wysokość.
W metodzie Main stworzyliśmy listę zawierającą obiekty wojowników.
Teraz wybierzmy z tej listy wojowników których Wysokość jest równa sto.




Wiemy jak to zrobić.
Zamiast słowa kluczowego var możemy jawnie wpisać typ listy, którą mamy w wyniku.
Wybieramy bigWarriors spośród wojowników, których wysokość wojownika wynosi sto.
Następnie przekształcamy w Listę.
Następnie piszemy o liczbie wybranych wojowników na konsoli.
I to wszystko. W rezultacie możemy zobaczyć dwóch wojowników.

LambdaExamples - Microsoft Visual Studio

Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc

Debug | Any CPU | Rozpocznij

EntryPoint.cs

LambdaExamples | LambdaExamples.Warrior | Height

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace LambdaExamples
{
    public class EntryPoint
    {
        static void Main()
        {

            // 5. Select vs Where
            List<Warrior> warriors = new List<Warrior>()
            {
                new Warrior() { Height = 100 },
                new Warrior() { Height = 80 },
                new Warrior() { Height = 100 },
                new Warrior() { Height = 70 },
            };


            List<Warrior> bigWarriors = warriors.Where(wh => wh.Height == 100)
                                        .ToList();

            Console.WriteLine("Count of big warriors: " + bigWarriors.Count);

            List<int> bigWarriorsHeights = warriors.Where(wh => wh.Height == 100)
                                                   .Select(wh => wh.Height)
                                                   .ToList();


            Console.WriteLine("Big warriors Heights items: " + string.Join(", ", bigWarriorsHeights));
        }
    }
    internal class Warrior
    {
        public int Height { get; set; }
    }
}
```

C:\Windows\system32\cmd.exe

```
Count of big warriors: 2
Big warriors Heights items: 100, 100
Aby kontynuować, naciśnij dowolny klawisz . . .
```

Jeśli chcesz otrzymać kolekcję, która zawiera tylko wysokości, a wysokość wynosi 100, musimy napisać:

Po nazwie kolekcji wpisujemy metodę WHERE - w której wybieramy wszystkie elementy, w których właściwość Height jest równa sto.

Następnie piszemy metodę SELECT - jako parametr wpisujemy wyrażenie lambda, które wybiera tylko właściwość Height.

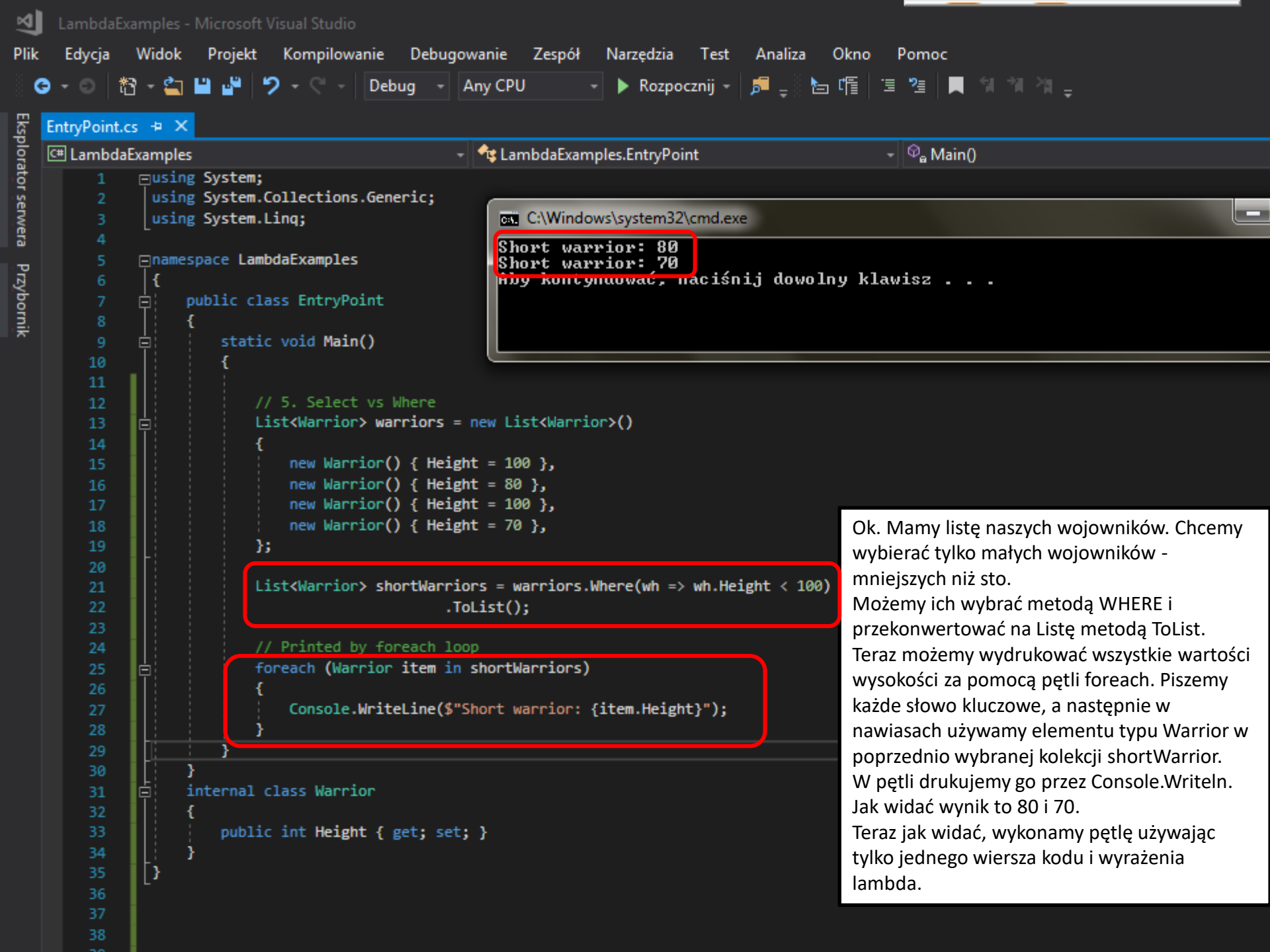
A teraz, przechodząc na Listę, mamy Listę wszystkich liczb o wartości sto.

Teraz możemy użyć metody Join do wypisania elementów.

Istnieją dwa. Oczywiście o wartości 100.

# ForEach method

Teraz zostanie pokazane, jak możemy używać metody ForEach w kolekcjach zamiast tradycyjnej pętli foreach.




Ok. Mamy listę naszych wojowników. Chcemy wybierać tylko małych wojowników - mniejszych niż sto.
Możemy ich wybrać metodą WHERE i przekonwertować na Listę metodą ToList.
Teraz możemy wydrukować wszystkie wartości wysokości za pomocą pętli foreach. Piszemy każde słowo kluczowe, a następnie w nawiasach używamy elementu typu Warrior w poprzednio wybranej kolekcji shortWarrior.
W pętli drukujemy go przez Console.Writeln.
Jak widać wynik to 80 i 70.
Teraz jak widać, wykonamy pętlę używając tylko jednego wiersza kodu i wyrażenia lambda.

LambdaExamples - Microsoft Visual Studio

Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc

Debug | Any CPU | Rozpocznij

EntryPoint.cs

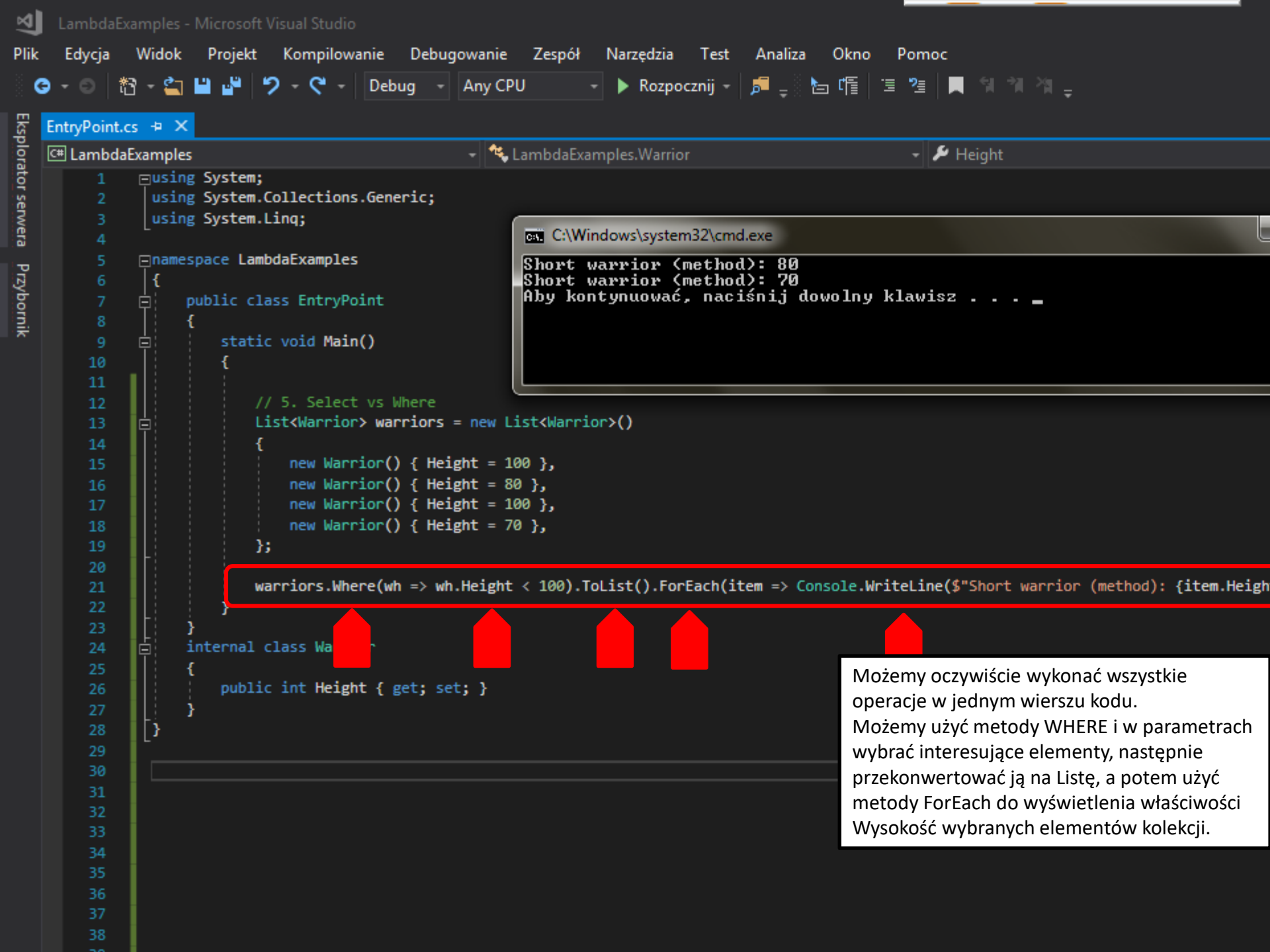
LambdaExamples | LambdaExamples.Warrior | Height

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace LambdaExamples
{
    public class EntryPoint
    {
        static void Main()
        {

            // 5. Select vs Where
            List<Warrior> warriors = new List<Warrior>()
            {
                new Warrior() { Height = 100 },
                new Warrior() { Height = 80 },
                new Warrior() { Height = 100 },
                new Warrior() { Height = 70 },
            };

            List<Warrior> shortWarriors = warriors.Where(wh => wh.Height < 100)
                                                  .ToList();

            // Printed by foreach loop
            foreach (Warrior item in shortWarriors)
            {
                Console.WriteLine($"Short warrior: {item.Height}");
            }

            // Printed by ForEach method
            shortWarriors.ForEach(item => Console.WriteLine($"Short warrior (method): {item.Height}"));
        }
    }
    internal class Warrior
    {
        public int Height { get; set; }
    }
}
```

C:\Windows\system32\cmd.exe

```
Short warrior: 80
Short warrior: 70
Short warrior (method): 80
Short warrior (method): 70
Aby kontynuować, naciśnij dowolny klawisz . . .
```

W tym przykładzie widzimy metodę tworzenia pętli ForEach przy użyciu metody listy ForEach. Używając więc listy z wybranymi elementami możemy napisać metodę ForEach, w której parametrem jest wyrażenie lambda. Najpierw piszemy element jako zmienną iteracyjną, a następnie operator lambda i kod do wykonania dla każdego elementu na liście. Wynik jest taki sam jak przy użyciu pętli foreach.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs
LambdaExamples
LambdaExamples.Warrior
Height
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
// 5. Select vs Where
List<Warrior> warriors = new List<Warrior>()
{
new Warrior() { Height = 100 },
new Warrior() { Height = 80 },
new Warrior() { Height = 100 },
new Warrior() { Height = 70 },
};
warriors.Where(wh => wh.Height < 100).ToList().ForEach(item => Console.WriteLine($"Short warrior (method): {item.Heigh
}
}
internal class Wa
{
public int Height { get; set; }
}
}
C:\Windows\system32\cmd.exe
Short warrior (method): 80
Short warrior (method): 70
Aby kontynuować, naciśnij dowolny klawisz . . .
Możemy oczywiście wykonać wszystkie operacje w jednym wierszu kodu.
Możemy użyć metody WHERE i w parametrach wybrać interesujące elementy, następnie przekonwertować ją na Listę, a potem użyć metody ForEach do wyświetlenia właściwości Wysokość wybranych elementów kolekcji.