Zaawansowane programowanie obiektowe - wykład 6

Opracowanie: dr Piotr Jastrzębski

LINQ - język zapytań

LINQ

Language INtegrated Query (LINQ) – część technologii Microsoft .NET, której podstawy teoretyczne zostały opracowane przez Erika Meijera. Technologia LINQ umożliwia zadawanie pytań na obiektach. Składnia języka LINQ jest prosta i przypomina SQL.

- LINQ to Objects
- LINQ to XML
- LINQ to SQL

- Sekwencja - każdy obiekt implementujący interfejs `IEnumerable<T>`.
- Element - każdy składnik sekwencji.
- Sekwencja lokalna - reprezentuje lokalną kolekcję obiektów przechowywanych w pamięci.
- Operator zapytania - metoda przekształcająca sekwencję.
- Sekwencja wejściowa/wyjściowa

Standardowe operatory zapytań

Standardowe operatory zapytań - link.

```
string[] imiona = { "Anna", "Ada", "Tomasz", "Zofia" };
IEnumerable<string> filtr = Enumerable.Where
                (imiona, n => n.Length >= 4);
foreach (var s in filtr)
{
    Console.WriteLine(s);
}
Console.ReadKey();
```

Alternatywne opcje:

- wyrażenie lambda

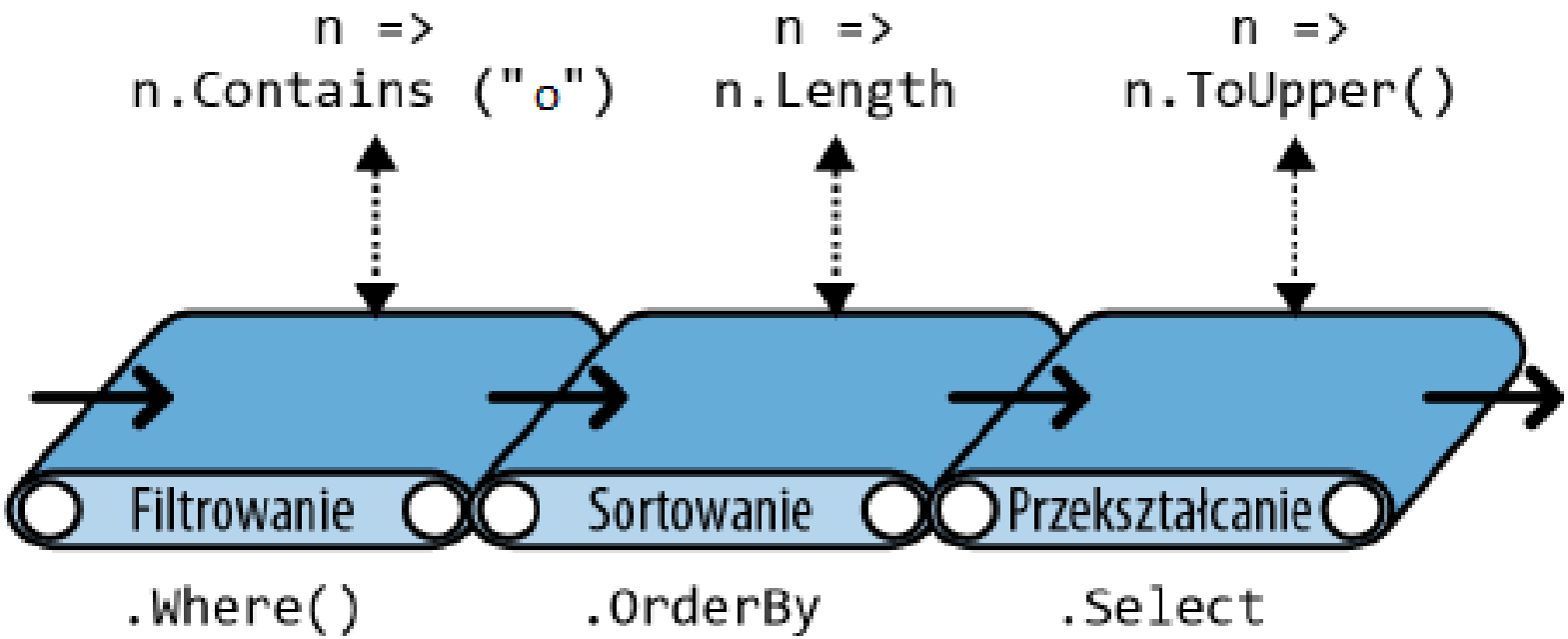
```
IEnumerable<string> filtr = imiona.Where
                            (n => n.Length >= 4);
```

- wyrażenia zapytań/list comprehension

```
IEnumerable<string> filtr = from n in imiona
                            where n.Length >= 4
                            select n;
```

```
IEnumerable<string> filteredNames = names.Where
                                (n => n.Contains ("g"));
```

```
IEnumerable<string> filtr = imiona
    .Where(n => n.Contains("o"))
    .OrderBy(n => n.Length)
    .Select(n => n.ToUpper());
```


n =>
n.Contains ("o")
n =>
n.Length
n =>
n.ToUpper()
Filtrowanie
Sortowanie
Przekształcanie
.Where()
.OrderBy
.Select

Rysunek 1:

Porządkowanie naturalne

```
int[] numbers = { 10, 9, 8, 7, 6 };
IEnumerable<int> firstThree = numbers.Take(3);
IEnumerable<int> lastTwo = numbers.Skip(3);
IEnumerable<int> reversed = numbers.Reverse();
```

Inne przykłady

```
int[] numbers = { 10, 9, 8, 7, 6 };
int firstNumber = numbers.First();
int lastNumber = numbers.Last();
int secondNumber = numbers.ElementAt(1);
int secondLowest = numbers.OrderBy(n => n).Skip(1).First();
int count = numbers.Count();
int min = numbers.Min();
bool hasTheNumberNine = numbers.Contains(9);
bool hasMoreThanZeroElements = numbers.Any();
bool hasAnOddElement = numbers.Any(n => n % 2 != 0);
```

```
int[] seq1 = { 1, 2, 3 };
int[] seq2 = { 3, 4, 5 };
IEnumerable<int> concat = seq1.Concat(seq2);
IEnumerable<int> union = seq1.Union(seq2);
```

```
string[] imiona = { "Anna", "Ada", "Tomasz", "Zofia" };
IEnumerable<string> filtr =
  from n in imiona
  where n.Contains("o")
  orderby n.Length
  select n.ToUpper();
```

Składnia mieszana

```
string[] imiona = { "Anna", "Ada", "Tomasz", "Zofia" };
int ile = (from n in imiona where n.Contains("o")
                    select n).Count();
```

Wyrażenia opóźnione

```
var numbers = new List<int>() { 1 };
IEnumerable<int> query = numbers.Select (n => n * 5);
numbers.Add (2);
foreach (int n in query)
{
    Console.WriteLine(n);
}
```

Ponowne obliczanie

```
var numbers = new List<int>() { 2, 4 };
IEnumerable<int> query = numbers.Select(n => n * 5);
foreach (int n in query)
{
    Console.WriteLine(n);
}
numbers.Clear();
Console.WriteLine("po clear");
foreach (int n in query)
{
    Console.WriteLine(n);
}
Console.ReadKey();
```

```
var numbers = new List<int>() { 1, 2 };
List<int> timesTen = numbers
.Select(n => n * 10)
.ToList();
numbers.Clear();
Console.WriteLine(timesTen.Count);
Console.ReadKey();
```

Przechwytywanie zmiennych

```
int[] numbers = { 1, 2 };
int factor = 3;
IEnumerable<int> query = numbers.Select(n => n * factor);
factor = 5;
foreach (int n in query)
{
    Console.WriteLine(n);
}
Console.ReadKey();
```

```
IEnumerable<char> query = "Nie tego się spodziewaliśmy";
string vowels = "aeęiouy";
for (int i = 0; i < vowels.Length; i++)
    query = query.Where(c => c != vowels[i]);
foreach (char c in query) Console.Write(c);
Console.ReadKey();
```

Naprawa I:

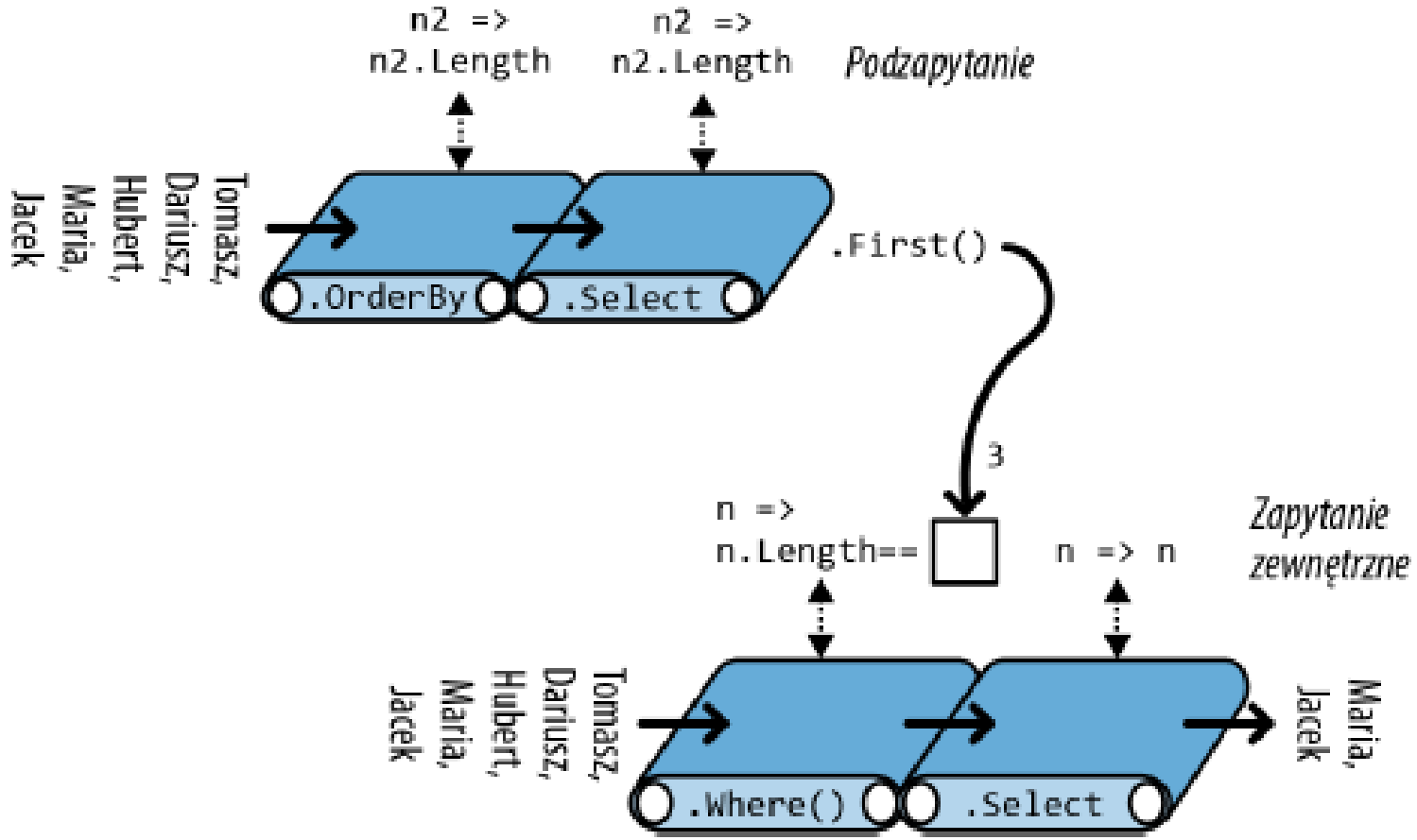
```
for (int i = 0; i < vowels.Length; i++)
{
    char vowel = vowels[i];
    query = query.Where (c => c != vowel);
}
```

Naprawa II:

```
foreach (char vowel in vowels)
    query = query.Where (c => c != vowel);
```

Podzapytania

```
string[] names = { "Tomasz", "Dariusz", "Hubert", "Maria",
                   "Jacek" };
IEnumerable<string> outerQuery = names
    .Where (n => n.Length == names.OrderBy
                             (n2 => n2.Length)
    .Select (n2 => n2.Length).First());
```


n2 =>
n2.Length
n2 =>
n2.Length
Podzapytanie
Tomasz,
Dariusz,
Hubert,
Maria,
Jacek
.OrderBy
.Select
.First()
3
n =>
n.Length==
n => n
Zapytanie
zewnętrzne
Tomasz,
Dariusz,
Hubert,
Maria,
Jacek
.Where()
.Select
Maria,
Jacek

Rysunek 2:

into

```
string[] names = { "Tomasz", "Dariusz", "Hubert", "Maria",
                   "Jacek" };
IEnumerable<string> query =
    from n in names
    select n.Replace("a", "").Replace("e", "")
            .Replace("i", "").Replace("o", "")
            .Replace("u", "").Replace("y", "")
    into noVowel
    where noVowel.Length > 2
    orderby noVowel
    select noVowel;
```

Opakowanie zapytań

```
IEnumerable<string> query =
  from n1 in
  (
    from n2 in names
    select n2.Replace ("a", "").Replace ("e", "")
             .Replace ("i", "").Replace ("o", "")
             .Replace ("u", "").Replace ("y", "")
  )
  where n1.Length > 2 orderby n1 select n1;
```

Inicjalizatory obiektów

```
class TempProjectionItem
{
  public string Original; // oryginalne imię
  public string Vowelless; // imię pozbawione samogłosek
}
```

```
string[] names = { "Tomasz", "Dariusz", "Hubert", "Maria",
                   "Jacek" };
IEnumerable<TempProjectionItem> temp =
  from n in names
  select new TempProjectionItem
  {
    Original = n,
    Vowelless = n.Replace ("a", "").Replace ("e", "")
        .Replace ("i", "").Replace ("o", "")
        .Replace ("u", "").Replace ("y", "")
  };
```

```
IEnumerable<string> query = from item in temp
        where item.Vowelless.Length > 2
        select item.Original;
```

Typy anonimowe

```
string[] names = { "Tomasz", "Dariusz", "Hubert", "Maria",
                   "Jacek" };
IEnumerable<string> query =
  from n in names
  let vowelless = n.Replace("a", "").Replace("e", "")
          .Replace("i", "").Replace("o", "")
          .Replace("u", "").Replace("y", "")
  where vowelless.Length > 2
  orderby vowelless
  select n;
```

Czy programowanie obiektowe się kończy?

Koniec obiektówki?

Do poczytania - link.

```
Func<int, int> f = x => x + 1;
Func<int, int> g = x => x + 2;
Func<Func<int, int>, Func<int, int>, int, int>
            fog = (f1, g1, x) => f1.Invoke(g1.Invoke(x));
Console.WriteLine(fog(f,g,4));
Console.ReadKey();
```

Bibliografia

- Wikipedia.
- Joseph Albahari, Ben Albahari, C# 7.0 w pigułce, wyd. Helion, 2018.