Операции Distinct, Union, Except и Intersect

Операции множеств используются для выполнения математических операций с множествами на последовательностях.

Distinct

Операция Distinct удаляет дублированные элементы из входной последовательности. У операции Distinct есть один прототип, описанный ниже:

C#

```
public static IEnumerable<T> Distinct<T>(
      this IEnumerable<T> source);
```

Эта операция возвращает объект, перечисляющий элементы входной последовательности source и выдающий последовательность, в которой каждый элемент не эквивалентен предыдущим выданным. Эквивалентность элементов определяется методами GetHashCode и Equals.

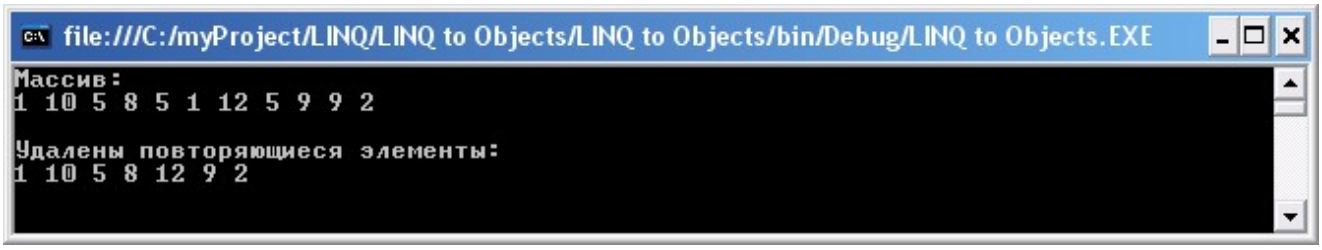
Давайте рассмотрим пример использования операции Distinct:

C#

```
int[] arr = { 1, 10, 5, 8, 5, 1, 12, 5, 9, 9, 2 };
            Console.WriteLine("Массив: ");
            foreach (int i in arr)
                Console.Write(i + " ");

            IEnumerable<int> nums = arr.Distinct();

            Console.WriteLine("\n\nУдалены повторяющиеся элементы: ");
            foreach (int i in nums)
                Console.Write(i + " ");
```



Union

Операция Union возвращает объединение множеств из двух исходных последовательностей. У этой операции имеется один прототип, описанный ниже:

C#

```
public static IEnumerable<T> Union<T>(
      this IEnumerable<T> first,
      IEnumerable<T> second);
```

Эта операция возвращает объект, который сначала перечисляет элементы последовательности по имени first, выдавая последовательность, в которой каждый элемент не эквивалентен предыдущим выданным, затем перечисляет вторую входную последовательность second, опять-таки, выдавая последовательность без повторений. Эквивалентность элементов определяется методами GetHashCode и Equals.

Чтобы продемонстрировать разницу между операцией Union и описанной ранее операцией Concat, в примере, представленном ниже, создаются последовательности first и second из массива cars, что приведет к дублированию пятого элемента в обеих последовательностях. Затем отображается количество элементов в массиве cars, а также в последовательностях first и second, наряду с количеством элементов в конкатенированной и объединенной последовательностях:

C#

Пройди тесты

C# тест (легкий) (https://professorweb.ru/test/c-sharp-test.html)

.NET тест (средний) (https://professorweb.ru/test/asp-test.html)

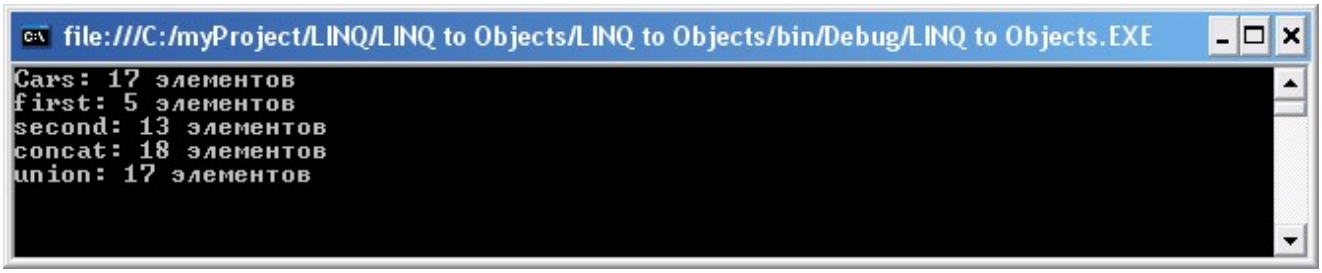
```
string[] cars = { "Alfa Romeo", "Aston Martin", "Audi", "Nissan", "Chevrolet",  "Chrysler",
"Dodge", "BMW",
                            "Ferrari",  "Bentley", "Ford", "Lexus", "Mercedes", "Toyota",
"Volvo", "Subaru", "Жигули :)"};

            IEnumerable<string> first = cars.Take(5);
            IEnumerable<string> second = cars.Skip(4);

            // Поскольку пропущены 4 элемента, пятый элемент
            // должен присутствовать в обеих последовательностях
            IEnumerable<string> concat = first.Concat<string>(second);
            IEnumerable<string> union = first.Union<string>(second);

            Console.WriteLine(@"Cars: {0} элементов
first: {1} элементов
second: {2} элементов
concat: {3} элементов
union: {4} элементов",cars.Count(),first.Count(),second.Count(),concat.Count(),union.Count
());
```

В конечном итоге последовательность concat должна иметь на один элемент больше, чем массив cars. Последовательность union должна содержать то же количество элементов, что и массив cars. Это доказывают результаты выполнения кода:



Intersect

Операция Intersect возвращает пересечение множеств из двух исходных последовательностей. Операция Intersect имеет один прототип, описанный ниже:

C#

```
public static IEnumerable<T> Intersect<T> (
    this IEnumerable<T> first,
    IEnumerable<T> second);
```

Пройди тесты

C# тест (легкий) (https://professorweb.ru/test/c-sharp-test.html)

.NET тест (средний) (https://professorweb.ru/test/asp-test.html)

Эта операция возвращает объект, который сначала перечисляет элементы последовательности по имени first, выбирая оттуда каждый элемент, который не эквивалентен предыдущему выбранному элементу. Затем он перечисляет вторую входную последовательность, помечая любой элемент, имеющийся в обеих последовательностях, для включения в выходную последовательность. Затем осуществляется проход по помеченным элементам, с помещением их в выходную последовательность в том порядке, в котором они были собраны. Эквивалентность элементов определяется с помощью методов GetHashCode и Equals.

Чтобы продемонстрировать применение операции Intersect, ниже используются операции Take и Skip для генерации двух последовательностей и получения некоторого их перекрытия, как в примере с Union, где был намеренно дублирован пятый элемент. После вызова операции Intersect на этих двух сгенерированных последовательностях в возвращаемой последовательности intersect должен оказаться только дублированный пятый элемент:

C#

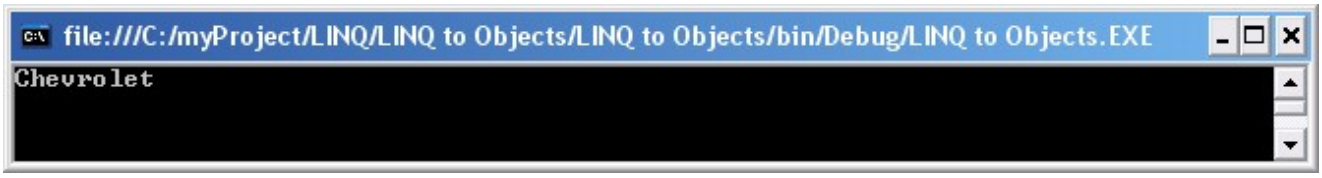
```
string[] cars = { "Alfa Romeo", "Aston Martin", "Audi", "Nissan", "Chevrolet",  "Chrysler",
"Dodge", "BMW",
                            "Ferrari",  "Bentley", "Ford", "Lexus", "Mercedes", "Toyota",
"Volvo", "Subaru", "Жигули :)"};

// 5 элемент присутствует в обоих коллекциях
IEnumerable<string> first = cars.Take(5);
IEnumerable<string> second = cars.Skip(4);

IEnumerable<string> auto = first.Intersect(second);

foreach (string s in auto)
        Console.WriteLine(s);
```

В конечном итоге должна быть получена последовательность auto, которая состоит только из одного элемента, содержащего дублированный пятый элемент массива cars — Chevrolet:



Итак, LINQ работает! Насколько часто вам приходилось ранее выполнять операции над множествами элементов из двух коллекций? Было ли это трудно? Благодаря LINQ, все сложности в прошлом.

Except

Операция Except возвращает последовательность, содержащую все элементы первой последовательности, которых нет во второй последовательности. Эта операция имеет один прототип, описанный ниже:

C#

```
public static IEnumerable<T> Except<T>(
       this IEnumerable<T> first,
       IEnumerable<T> second);
```

Эта операция возвращает объект, который при перечислении перебирает элементы входной последовательности по имени second, собирая все элементы, которые не эквивалентны ранее собранным. Затем происходит перечисление входной последовательности first, с выдачей каждого ее элемента, которого нет в коллекции из второй последовательности. Эквивалентность одного элемента другому определяется с использованием их методов GetHashCode и Equals.

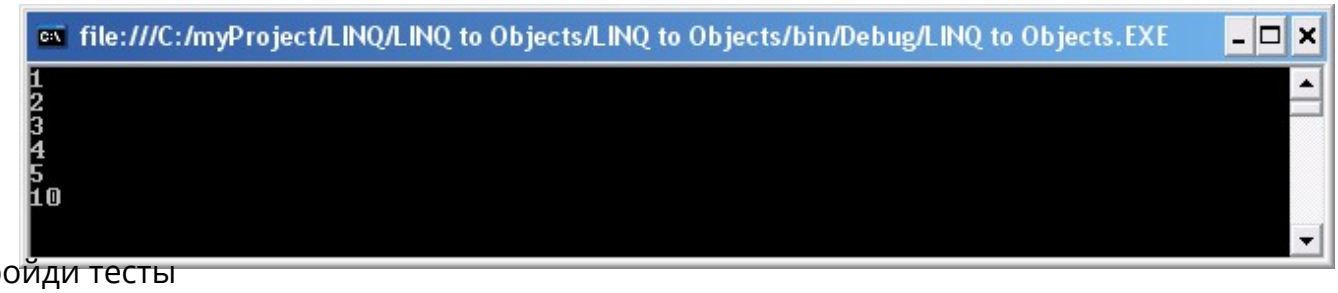
В следующем примере используются два массива arr1 и arr2. С помощью операции Except будут удалены взаимоисключающие элементы коллекций:

C#

```
int[] arr1 = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
int[] arr2 = { 6, 7, 8, 9 };

IEnumerable<int> nums = arr1.Except<int>(arr2);

foreach (int i in nums)
        Console.WriteLine(i);
```



Пройди тесты

C# тест (легкий) (https://professorweb.ru/test/c-sharp-test.html)

.NET тест (средний) (https://professorweb.ru/test/asp-test.html)

Назад (2_8.php)	8	9	10	Вперед (2_10.php)

Пройди тесты

C# тест (легкий) (https://professorweb.ru/test/c-sharp-test.html)

.NET тест (средний) (https://professorweb.ru/test/asp-test.html)