1. Что такое LINQ?

Язык интегрированных запросов (Language Integrated Query  — LINQ ) дает возможность писать запросы к коллекциям объектов. Язык LINQ позволяет запрашивать любую коллекцию, реализующую интерфейс IEnumerable.

Оператор запроса  — это метод, который трансформирует последовательность. Типичный оператор запроса принимает входную последовательность и выдает трансформированную выходную последовательность.

Запрос — это выражение, которое трансформирует последовательности с помощью одного или более операторов запросов. Простейший запрос состоит из одной входной последовательности и одного оператора.

1. В чем разница между отложенными операциями  и не отложенными операциями LINQ to Object?

При отложенном выполнении LINQ-выражение не выполняется, пока не будет произведена итерация или перебор по выборке, например, в цикле foreach. Обычно подобные операции возвращают объект IEnumerable<T> или IOrderedEnumerable<T>. Полный список отложенных операций LINQ:

* **AsEnumerable**
* **Cast**
* **Concat**
* **DefaultIfEmpty**
* **Distinct**
* **Except**
* **GroupBy**
* **GroupJoin**
* **Intersect**
* **Join**
* **OfType**
* **OrderBy**
* **OrderByDescending**
* **Range**
* **Repeat**
* **Reverse**
* **Select**
* **SelectMany**
* **Skip**
* **SkipWhile**
* **Take**
* **TakeWhile**
* **ThenBy**
* **ThenByDescending**
* **Union**
* **Where**

Важно понимать, что переменная запроса сама по себе не выполняет никаких действий и не возвращает никаких данных. Она только хранит набор команд, которые необходимы для получения результатов. То есть выполнение запроса после его создания откладывается. Само получение результатов производится при переборе в цикле foreach.

С помощью ряда методов мы можем применить немедленное выполнение запроса. Это методы, которые возвращают одно атомарное значение или один элемент или данные типов Array, List и Dictionary. Полный список подобных операций в LINQ:

* **Aggregate**
* **All**
* **Any**
* **Average**
* **Contains**
* **Count**
* **ElementAt**
* **ElementAtOrDefault**
* **Empty**
* **First**
* **FirstOrDefault**
* **Last**
* **LastOrDefault**
* **LongCount**
* **Max**
* **Min**
* **SequenceEqual**
* **Single**
* **SingleOrDefault**
* **Sum**
* **ToArray**
* **ToDictionary**
* **ToList**
* **ToLookup**

1. Что такое лямбда-выражения?

Лямбда-выражение в языке программирования C# - это синтаксическое средство для определения анонимных методов. Лямбда-выражения предоставляют более краткий и удобный синтаксис для создания методов, особенно в контексте функционального программирования. Они часто используются в LINQ (Language Integrated Query) для создания анонимных функций или делегатов, которые передаются в методы LINQ для выполнения запросов к данным.

1. \*\*Входные параметры:\*\* Лямбда-выражение может принимать входные параметры (аргументы), которые указываются слева от оператора `=>`. Например, `(x, y) => x + y` принимает два параметра `x` и `y`.

2. \*\*Оператор `=>`:\*\* Этот оператор разделяет входные параметры и тело лямбда-выражения. Он читается как "стремится к" или "переходит в". Например, `(x, y) => x + y` означает "лямбда с параметрами `x` и `y`, возвращающая `x + y`".

3. \*\*Тело выражения:\*\* Тело лямбда-выражения содержит код, который выполняется при вызове лямбда-функции. Например, в `(x, y) => x + y` тело выражения - `x + y`.

1. Какие есть группы операции в LINQ to Object? Перечислите
2. Агрегация (Count, Min, Max)
3. Преобразование (Cast, ofType, ToArray, ToList, ToDictionary)
4. Конкатенация (Concat)
5. Элемент (Last, First, Single , ElemetAt+ Default)
6. Множество (Except, Distinct, Union)
7. Генерация (Empty, Range, Repeat)
8. Соединение (Join, GroupJoin)
9. Упорядочивание (OrderBy, ThenBy, Reverse,….)
10. Проекция (Select, SelectMany)
11. Разбиение (Skip, Take , +While)
12. Ограничение (Where)
13. Квантификатор (Any, All, Contains)
14. Эквивалентность(SequeceEqual)
15. Как используется операция Where в LINQ to Object?

В LINQ to Objects операция `Where` представляет собой оператор запроса, который используется для фильтрации элементов коллекции на основе определенного условия.

- Операция `Where` принимает в качестве аргумента предикат — лямбда-выражение или делегат, определяющий условие фильтрации.

- Каждый элемент коллекции проверяется на соответствие предикату, и только те элементы, для которых предикат возвращает `true`, остаются в результате.

- Операция `Where` является отложенной, что означает, что фактическое выполнение фильтрации происходит только при обращении к результатам запроса.

- Результат операции `Where` является объектом, реализующим интерфейс `IEnumerable<T>`, где `T` - тип элементов исходной коллекции.

- В LINQ операция `Where` представляет собой метод расширения, что позволяет использовать ее в синтаксисе запросов или в виде цепочки методов расширений.

1. Как используется операция Select ?

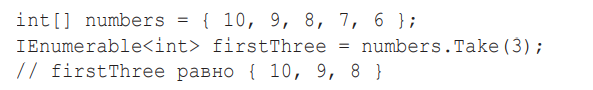
Операция `Select` в LINQ to Objects используется для проекции (отображения) элементов коллекции на новую форму. Эта операция позволяет создавать новые объекты или извлекать значения из существующих объектов в исходной коллекции.

- Операция `Select` принимает в качестве аргумента трансформирующую функцию (лямбда-выражение или делегат), которая определяет, как каждый элемент исходной коллекции должен быть преобразован.

- Вы можете использовать операцию `Select` для создания новых объектов на основе свойств или значений существующих объектов в коллекции.

1. Как используются операции Take, Skip?

Оператор Take выдает первые x элементов, отбрасывая остальные:



Оператор Skip пропускает первые x элементов и выдает остальные:



1. Как используется операция Concat ?

Операция `Concat` в LINQ to Objects используется для объединения двух коллекций в одну коллекцию. Эта операция создает новую коллекцию, содержащую все элементы из первой коллекции, за которой следуют все элементы из второй коллекции.

- Операция `Concat` принимает две коллекции и возвращает новую коллекцию, содержащую все элементы обеих исходных коллекций (без удаления дубликатов).

1. Как используется операция OrderBy?

Операция `OrderBy` в LINQ to Objects используется для сортировки элементов коллекции в порядке возрастания по определенному критерию. Эта операция создает новую упорядоченную последовательность на основе заданного ключа сортировки.

Вы можете использовать `OrderByDescending` для сортировки в порядке убывания. Также, вы можете использовать `ThenBy` и `ThenByDescending` для добавления дополнительных критериев сортировки в случае, если первые ключи сортировки равны.

1. Как используется операция Join?

- Операция `Join` объединяет две коллекции на основе значения указанного ключа. Это позволяет объединить элементы, у которых ключи равны.

- Ключи для объединения могут быть любого типа данных, и они должны быть совместимыми.

- Результатом операции `Join` является новая коллекция, состоящая из элементов, у которых значения ключей совпадают.

1. Как используются операции Distinct, Union, Except и Intersect?

Операция `Distinct` в LINQ to Objects используется для удаления дубликатов из коллекции, оставляя только уникальные элементы.

Операция `Union` в LINQ to Objects используется для объединения двух коллекций, при этом исключаются повторяющиеся элементы.

Операция `Except` в LINQ to Objects используется для получения разности двух коллекций, то есть всех элементов, присутствующих в первой коллекции, но отсутствующих во второй, возвращает новую коллекцию.

Операция `Intersect` в LINQ to Objects используется для получения пересечения двух коллекций, то есть всех элементов, присутствующих в обеих коллекциях, возвращает новую коллекцию.

1. [Как используются операции First, Last,](https://professorweb.ru/my/LINQ/base/level3/3_5.php) [Any, All и Contains](https://professorweb.ru/my/LINQ/base/level3/3_7.php)?

Операция `First` в LINQ to Objects используется для получения первого элемента из коллекции, который соответствует заданному условию.

- Если в коллекции нет элементов, соответствующих условию, `First` выбрасывает исключение `InvalidOperationException`. Вместо этого можно использовать `FirstOrDefault`, чтобы получить значение по умолчанию (например, `0` для чисел или `null` для ссылочных типов).

Операция `Last` в LINQ to Objects используется для получения последнего элемента из коллекции, который соответствует заданному условию.

- Если в коллекции нет элементов, соответствующих условию, `Last` выбрасывает исключение `InvalidOperationException`. Вместо этого можно использовать `LastOrDefault`, чтобы получить значение по умолчанию.

Операция `Any` в LINQ to Objects используется для проверки, содержит ли коллекция хотя бы один элемент, удовлетворяющий заданному условию.

- Операция `Any` возвращает `true`, если хотя бы один элемент в коллекции удовлетворяет заданному условию, и `false`, если нет.

Операция `All` в LINQ to Objects используется для проверки, удовлетворяют ли все элементы коллекции заданному условию.

- Операция `All` возвращает `true`, если все элементы в коллекции удовлетворяют заданному условию, и `false`, если хотя бы один элемент не удовлетворяет условию.

Операция `Contains` в LINQ to Objects используется для проверки, содержит ли коллекция определенный элемент.

- Операция `Contains` возвращает `true`, если элемент присутствует в коллекции, и `false`, если отсутствует.

1. [Как используются операции Count, Sum](https://professorweb.ru/my/LINQ/base/level3/3_8.php), [Min и Max](https://professorweb.ru/my/LINQ/base/level3/3_9.php), [Average?](https://professorweb.ru/my/LINQ/base/level3/3_10.php)

Операция `Count` в LINQ to Objects используется для подсчета количества элементов в коллекции, которые удовлетворяют заданному условию.

Операция `Sum` в LINQ to Objects используется для вычисления суммы значений числовой коллекции.

Операция `Min` в LINQ to Objects используется для нахождения минимального значения в числовой коллекции.

Операция `Max` в LINQ to Objects используется для нахождения максимального значения в числовой коллекции.

Операция `Average` в LINQ to Objects используется для вычисления среднего значения чисел в коллекции. Вот некоторые ключевые аспекты операции `Average`:

1. Что выведет на экран данный код?

|  |
| --- |
| class Test  {  public static void Main()  {  List<int> list = new List<int>();  list.AddRange(new int[] { 3, 1, 4, 8, 10, 4 });    List<int> some = list.FindAll(i => (i>=9));  foreach (int x in some)  Console.Write(x);  }  } |

1. Создается пустой список `list` типа `List<int>`.

2. С использованием метода `AddRange` в список добавляются элементы из массива `{ 3, 1, 4, 8, 10, 4 }`.

3. Создается новый список `some`, который получается из `list` с использованием метода `FindAll`. Этот метод принимает предикат (лямбда-выражение), который определяет условие выбора элементов. В данном случае предикат `(i >= 9)` означает выбор элементов, которые больше или равны 9.

4. Происходит вывод элементов списка `some` в консоль в цикле `foreach`.

Элементы `list`: `[3, 1, 4, 8, 10, 4]`

Элементы, удовлетворяющие условию `(i >= 9)`: `[10]`

Таким образом, код выведет в консоль только один элемент, который больше или равен 9, и результат будет: 10