# **№ 10 LINQ to Object**

## Задание

<https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/>

1. Задайте массив типа string, содержащий 12 месяцев (June, July, May, December, January ….). Используя LINQ to Object напишите запрос выбирающий последовательность месяцев с длиной строки равной n, запрос возвращающий только летние и зимние месяцы, запрос вывода месяцев в алфавитном порядке, запрос считающий месяцы содержащие букву «u» и длиной имени не менее 4-х..
2. Создайте коллекцию List<T> и параметризируйте ее типом (классом) из лабораторной №2 (при необходимости реализуйте нужные интерфейсы). Заполните ее минимум 10 элементами.

Если в задании указано свойство, которым ваш класс не обладает, то его нужно расширить, чтобы класс соответствовал условию. Один из запросов реализуйте используя язык LINQ и используя методы расширения LINQ.

1. На основе LINQ сформируйте следующие запросы по вариантам. При необходимости добавьте в класс T (тип параметра) свойства.
2. Придумайте и напишите свой собственный запрос, в котором было бы не менее 5 операторов из разных категорий: условия, проекций, упоря дочивания, группировки, агрегирования, кванторов и разбиения.
3. Придумайте запрос с оператором Join

|  |  |
| --- | --- |
| Вариант 1 | *количество векторов, содержащих 0;*  *список векторов с наименьшим модулем.*  *массив векторов (один) длины (количество элементов) 3,5,7.*  *максимальный вектор*  *первый вектор с отрицательным значением*  *упорядоченный список векторов по размеру* |
| Вариант 2 | *список рейсов для заданного пункта назначения;*  *количество рейсов для заданного дня недели*  *Рейс который вылетает в понедельник раньше всех*  *Рейс который вылетает в среду или пятницу позже всех*  *Список рейсов, упорядоченных по времени вылета* |
| Вариант 3 | *список студентов заданной специальности по алфавиту;*  *список заданной учебной группы и факультета*  *самого молодого студента*  *количество студентов заданной группы упорядоченных по фамилии*  *первого студента с заданным именем* |
| Вариант 4 | *список покупателей в алфавитном порядке;*  *список покупателей, у которых номер кредитной карточки находится в заданном интервале*  *максимального покупателя (критерии определит самостоятельно)*  *первых пят покупателей с максимальной суммой на карте* |
| Вариант 5 | *список абитуриентов, имеющих неудовлетворительные оценки;*  *список абитуриентов, у которых сумма баллов выше заданной;*  *количество абитуриентов с 10-ками по определенному предмету*  *массив абитуриентов упорядоченных по алфавиту*  *4 последних абитуриента с самой низкой успеваемостью* |
| Вариант 6 | *список книг заданного автора и года;*  *список книг, выпущенных после заданного года*  *самую тонкую книгу*  *5 первых самых толстых книг по низкой цене*  *Список книг отсортированных по цене* |
| Вариант 7 | *список квартир, имеющих заданное число комнат;*  *пять первых квартир на заданной улице заданного дома*  *количество квартир на определенной улице*  *список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке;* |
| Вариант 8 | *сведения об абонентах, у которых время внутригородских разговоров превышает заданное;*  *сведения об абонентах, которые пользовались междугородной связью;*  *количество абонентов с заданным значением дебета*  *максимального абонента (по вашему критерию)*  *упорядоченный список абонентов по фамилии* |
| Вариант 9 | *Создать массив объектов. Вывести: список автомобилей заданной марки;*  *список автомобилей заданной модели, которые эксплуатируются больше n лет;*  *количество автомобильной заданного цвета и диапазона цены*  *самый старый автомобиль*  *первых пять самых новых автомобилей*  *упорядоченный массив по цене* |
| Вариант 10 | *список товаров для заданного наименования;*  *список товаров для заданного наименования, цена которых не превосходит заданную;*  *количество наименований цена которых больше 100*  *максимальный товар (ваш критерий максимальности)*  *упорядоченный набор товаров по производителю, а потом по количеству.* |
| Вариант 11 | *Вывести:*  *список поездов, следующих до заданного пункта назначения;*  *список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;*  *максимальный поезд по количеству мест*  *последние пять поездов по времени отправления*  *упорядоченный список поездов по пункту назначения в алфавитном порядке* |
| Вариант 12 | *список автобусов для заданного номера маршрута;*  *список автобусов, которые эксплуатируются больше заданного срока;*  *минимальный по пробегу автобус*  *последние два автобуса максимальные по пробегу*  *упорядоченный список автобусов по номеру* |
| Вариант 13 | *список рейсов для заданного пункта назначения;*  *список рейсов для заданного дня недели;*  *максимальны по дню недели рейс*  *все рейсы в определенный день недели и с самым поздним временем вылета*  *упорядоченные по дню и времени рейсы*  *количество рейсов для заданного типа самолета* |
| Вариант 14 | *стек с наименьшим/наибольшим верхним элементом;*  *список стеков, содержащих отрицательные элементы.*  *Минимальный стек*  *Массив стеков длины 1 и 3*  *Первый стек с нулевым элементом*  *Упорядоченный массив стеков по сумме элементов* |
| Вариант 15 | *множества с наименьшей/наибольшей суммой элементов;*  *список множеств, содержащих отрицательные элементы.*  *Количество множеств, содержащих заданное значение*  *Максимальное множество*  *Первое множество с заданным элементом*  *Упорядоченный массив множеств по первому элементу* |
| Вариант 16 | *список дат для заданного года;*  *список дат, которые имеют заданный месяц*  *количество дат в определённом диапазоне*  *максимальную дату*  *Первую дату для заданного дня*  *Упорядоченный список дат (хронологически)* |
| Вариант 17 | *количество строк длины n и т*  *список строк, которые содержат заданное слово.*  *Максимальную строку*  *Первую строку, содержащую точку или ?*  *Последнюю строку с самым коротким словом*  *Упорядоченный массив по первому слову* |
| Вариант 18 | *матрицу с наибольшим/наименьшим количеством единиц;*  *список матриц с равным количеством заданного символа в каждой строке*  *Максимальную матрицу*  *Количество матриц заданного размера*  *Упорядоченный список матриц по количеству единиц* |
| Вариант 19 | *группы окружностей, центры которых лежат на одной прямой;*  *наибольший и наименьший по площади (периметру) объект;*  *Количество окружностей заданного радиуса*  *Первую окружность, лежащую в первой четверти*  *Упорядоченный список окружностей по площади* |
| Вариант 20 | *количество четырехугольников разного типа (квадрат, прямоугольник, ромб, произвольный)*  *определить для каждой группы наибольший и наименьший по площади (периметру) объект*  *Массив квадратов со стронной не более x*  *Упорядоченный по периметру массив прямоугольников* |
| Вариант 21 | *количество векторов, содержащих 0;*  *список векторов с наименьшим модулем.*  *массив векторов (один) длины (количество элементов) 3,5,7.*  *максимальный вектор*  *первый вектор с отрицательным значением*  *упорядоченный список векторов по размеру* |
| Вариант 22 | *вектора с заданным числом единиц/нулей;*  *определить и вывести равные вектора в коллекции*  *максимальный вектор*  *первый вектор с n единицами*  *упорядоченный вектор по числу единиц* |
| Вариант 23 | *время с заданным значением часов;*  *списки времен по группам: ночь, утро, день, вечер*  *минимальное время*  *Первое время в котором часы и минуты совпадают*  *Упорядоченный список времен* |
| Вариант 24 | *массивы только с четными/нечетными элементами;*  *массив с наибольшей суммой элементов*  *минимальный массив*  *количество массивов, содержащих заданное значение*  *количество равных массивов*  *упорядоченный массив массивов по первому элементу* |
| Вариант 25 | *подсчитать количество треугольников разного типа (равносторонний, равнобедренный, прямоугольный, произвольный).*  *определить для каждой группы наибольший и наименьший по периметру объект*  *минимальный по площади треугольник*  *треугольники со длинами сторон из диапазона <n и >m*  *упорядоченный по сумме длин сторон массив треугольников* |
| Вариант 26 | *множества только с четными/нечетными элементами;*  *множества, содержащие отрицательные элементы.*  *Количество пустых множеств*  *Список множеств длины которых принадлежат заданному диапазону*  *Минимальное множество* |

## Вопросы

1. Что такое LINQ?
2. В чем разница между отложенными операциями  и не отложенными операциями LINQ to Object?
3. Что такое лямбда-выражения?
4. Какие есть группы операции в LINQ to Object? Перечислите
5. Как используется операция Where в LINQ to Object?
6. Как используется операция Select ?
7. Как используются операции Take, Skip?
8. Как используется операция Concat ?
9. Как используется операция OrderBy?
10. Как используется операция Join?
11. Как используются операции Distinct, Union, Except и Intersect?
12. [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)?
13. [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)
14. 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);

}

}

## Краткие теоретические сведения

### LINQ - язык интегрированных запросов

[*LINQ to Objects*](http://professorweb.ru/my/LINQ/base/level1/linq_index.php)

LINQ to Objects - название, данное API-интерфейсу IEnumerable<T> для *стандартных операций запросов (Standard Query Operators)*. Именно LINQ to Objects позволяет выполнять запросы к массивам и находящимся в памяти коллекциям данных. Стандартные операции запросов - это статические методы класса System.Linq.Enumerable, которые используются для создания запросов LINQ to Objects.

[*LINQ to XML*](http://professorweb.ru/my/LINQ/linq_xml/level5/linq_to_xml_index.php)

LINQ to XML — название, назначенное API-интерфейсу LINQ, который ориентирован на работу с XML. В Microsoft не только добавили необходимые библиотеки XML для работы с LINQ, но также восполнили недостатки стандартной модели XML DOM, существенно облегчив работу с XML. Прошли времена, когда нужно было создавать XmlDocument только для того, чтобы поработать с небольшим фрагментом XML-кода. Чтобы воспользоваться преимуществами LINQ to XML, в проект понадобится добавить ссылку на сборку System.Xml.Linq.dll и директиву using System.Xml.Linq.

[*LINQ to DataSet и SQL*](http://professorweb.ru/my/LINQ/linq_sql/level8/linq_to_dataset_and_sql_index.php)

LINQ to DataSet — название, данное API-интерфейсу LINQ, который предназначен для работы с DataSet. У многих разработчиков есть масса кода, полагающегося на DataSet. Те, кто не хотят отставать от новых веяний, но и не готовы переписывать свой код, благодаря этому интерфейсу могут воспользоваться всей мощью LINQ.

LINQ to SQL — наименование, присвоенное API-интерфейсу IQueryable<T>, который позволяет запросам LINQ работать с базой данных Microsoft SQL Server. Чтобы воспользоваться преимуществами LINQ to SQL в проект понадобится добавить ссылку на сборку System.Data.Linq.dll, а также директиву using System.Data.Linq.

[*LINQ to Entities*](http://professorweb.ru/my/LINQ/linq_entities/level14/linq_to_entities_index.php)

LINQ to Entities — альтернативный API-интерфейс LINQ, используемый для обращения к базе данных. Он отделяет сущностную объектную модель от физической базы данных, вводя логическое отображение между ними двумя. С таким отделением возрастает мощь и гибкость, но также растет и сложность. Если нужна более высокая гибкость, чем обеспечивается LINQ to SQL, имеет смысл рассмотреть эту альтернативу.

В частности, когда необходимо ослабить связь между сущностной объектной моделью и базой данных, если сущностные объекты конструируются из нескольких таблиц или требуется большая гибкость в моделировании сущностных объектов, то в этом случае LINQ to Entities может стать оптимальным выбором.

[*Parallel LINQ*](http://professorweb.ru/my/LINQ/PLINQ/level16/plinq_index.php)

Формально отдельного продукта LINQ, который нужно было бы получать отдельно, не существует. LINQ полностью интегрирован в .NET Framework, начиная с версии 3.5 и Visual Studio 2008. В NET 4.0 и Visual Studio 2010 добавлена поддержка средств Parallel LINQ.

### Запросы LINQ

Одним из привлекательных для разработчиков средств LINQ является SQL-подобный синтаксис, доступный в LINQ-запросах. Синтаксис предоставлен через расширение языка C#, которое называется выражения запросов. Выражения запросов позволяют запросам LINQ принимать форму, подобную SQL, всего лишь с рядом небольших отличий.

Для выполнения запроса LINQ выражения запросов не обязательны. Альтернативой является использование стандартной точечной нотации C# с вызовом методов на объектах и классах. Во многих случаях применение стандартной точечной нотации оказывается более предпочтительным, поскольку она более наглядно демонстрирует, что в действительности происходит и когда.

При записи запроса в стандартной точечной нотации не происходит никакой трансформации при компиляции. Именно поэтому во многих примерах не используется синтаксис выражений запросов, а предпочтение отдается стандартному синтаксису точечной нотации. Получить представление о различиях между этими двумя синтаксисами лучше всего на примере:

string[] names = {

"Adams", "Arthur", "Buchanan", "Bush", "Carter", "Cleveland",

"Clinton", "Coolidge", "Eisenhower", "Fillmore", "Ford", "Garfield",

"Grant", "Harding", "Harrison", "Hayes", "Hoover", "Jackson",

"Jefferson", "Johnson", "Kennedy", "Lincoln", "Madison", "McKinley",

"Monroe", "Nixon", "Obama", "Pierce", "Polk", "Reagan", "Roosevelt",

"Taft", "Taylor", "Truman", "Tyler", "Van Buren", "Washington", "Wilson"};

// Использование точечной нотации

IEnumerable<string> sequence = names

.Where(n => n.Length < 6)

.Select(n => n);

// Использование синтаксиса выражения запроса

IEnumerable<string> sequence = from n in names

where n.Length < 6

select n;

foreach (string name in sequence)

{

Console.WriteLine("{0}", name);

}

Синтаксис выражений запросов поддерживается только для наиболее распространенных операций запросов: Where, Select, SelectMany, Join, GroupJoin, GroupBy, OrderBy, ThenBy, OrderByDescending и ThenByDescending.

Выражения запросов должны подчиняться перечисленным ниже правилам:

1. Выражение должно начинаться с конструкции from.
2. Остальная часть выражения может содержать ноль или более конструкций from, let или where. **Конструкция from** — это генератор, который объявляет одну или более переменных диапазона, перечисляющих последовательность или соединение нескольких последовательностей. **Конструкция let** представляет переменную диапазона и присваивает ей значение. **Конструкция where** фильтрует элементы из входной последовательности или соединения несколько входных последовательностей в выходную последовательность.
3. Остальная часть выражения запроса может затем включать конструкцию orderby, содержащую одно или более полей сортировки с необязательным направлением упорядочивания. Направлением может быть *ascending (по возрастанию)* или*descending (по убыванию)*.
4. Затем в оставшейся части выражения может идти конструкция select или group.
5. Наконец в оставшейся части выражения может следовать необязательная конструкция продолжения. Такой конструкцией может быть либо into, ноль или более конструкций join, или же другая повторяющаяся последовательность перечисленных элементов, начиная с конструкций из правила 2. **Конструкция into** направляет результаты запроса в воображаемую выходную последовательность, которая служит конструкцией from для последующих выражения запросов, начиная с конструкций из правила 2.