**Лабораторная работа № 70-71**

**«Язык интегрированных запросов»**

**Цель работы:** получение навыков работы фильтрации информации с помощью LINQ запросов.

**Задания для лабораторной работы:**

void task\_0() {

var students = new [] {

new {Name = "Aleksandr", Age = 20},

new {Name = "Vasily", Age = 23},

new {Name = "Petr", Age = 21} };

var selection = from student in students where student.Age > 20 select student.Name;

foreach (var student in selection.ToList()) { Console.WriteLine(student); }

}



**Оптимизируйте его как LINQ запрос.**

На каждую LINQ команду вам даются данные и условие по которому нужно отсортировать ваши данные, используя LINQ запросы. Необходимо вывести полученные результаты, первичные результаты и текст запроса.

**Задание 1:** Использование **Where**

**Данные**: Список сотрудников в виде массива объектов:

**Условие**: Найдите всех сотрудников, работающих в департаменте "IT".

var result = from emp in employees where emp.Department == "IT" select emp;



**Задание** 2: Использование **Select**

**Данные**: Список продуктов в виде массива объектов:

**Условие**: Создайте новую коллекцию, которая будет содержать только названия продуктов.

var result = from product in products select product.Name;



**Задание** 3: Использование **First**

**Данные**: Список книг в виде массива объектов:

**Условие**: Найдите первую книгу, автором которой является "Лев Толстой".

var result = books.Where(book => book.Author == "Лев Толстой").First();

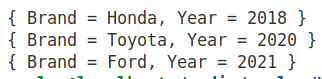


**Задание** 4: Использование **OrderBy**

**Данные**: Список автомобилей в виде массива объектов:

**Условие**: Сортируйте список автомобилей по году выпуска в порядке возрастания.

var result = from car in cars orderby car.Year select car;



**Задание** 5: Использование **GroupBy**

**Данные**: Список студентов и их оценки в виде массива объектов:

**Условие**: Группируйте студентов по их оценкам.

var result = students.GroupBy(student => student.Grade);



**Задание** 6: Использование **Count**

**Данные**: Список фильмов в виде массива объектов:

**Условие**: Подсчитайте количество фильмов в жанре "Фантастика".

var result = movies.Count(movi => movi.Genre == "Фантастика");



**Задание** 7: Использование **Any**

**Данные**: Список клиентов в виде массива объектов:

**Условие**: Убедитесь, есть ли среди клиентов активные клиенты.

var result = clients.Any(client => client.IsActive);



**Задание** 8: Использование **Last**

**Данные**: Список любимых фильмов в виде массива объектов:

**Условие**: Найдите последний фильм в списке по году выпуска.

var result = movies.OrderBy(mov => mov.Year).Last();

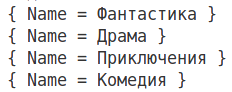


**Задание** 9: Использование **Distinct**

**Данные**: СписокGenres в виде массива объектов:

**Условие**: Получите список уникальных жанров.

var result = genres.Distinct();

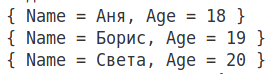


**Задание** 10: Использование **Take**

**Данные**: Список студентов в виде массива объектов:

**Условие**: Возьмите первых 3 студента из списка.

var result = students.Take(3);

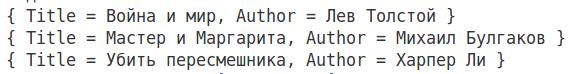


**Задание** 11: Использование **Skip**

**Данные**: Список книг в виде массива объектов:

**Условие**: Пропустите первые 2 книги и выведите оставшиеся.

var result = books.Skip(2);

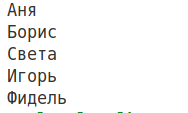


**Задание** 12: Использование **SelectMany**

**Данные**: Список курсов и студентов в виде массивов объектов:

**Условие**: Получите список всех студентов, зарегистрированных на курсы.

var result = courses.SelectMany(course => course.Students);

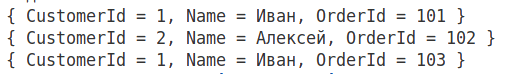


**Задание** 13: Использование **Join**

**Данные**: Списки заказов и клиентов в виде массивов объектов:

**Условие**: Соедините заказы с клиентами и получите список заказов с именами клиентов.

var result = orders.Join(customers, order => order.CustomerId, customer => customer.CustomerId, (order, customer) => new {CustomerId = order.CustomerId, Name = customer.Name, OrderId = order.OrderId});

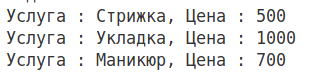


**Задание** 14: Использование **ToList**

**Данные**: Список услуг в виде массива объектов:

**Условие**: Создайте список услуг с ценами, указав только названия услуг и их цены.

var result = services.Select(ser => new {ser.ServiceName, ser.Price}).ToList();

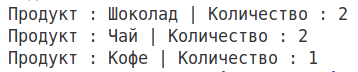


**Задание** 15: Использование **GroupBy** и **Select**

**Данные**: Список заказов в виде массива объектов:

**Условие**: Группируйте заказы по продуктам и вычислите общее количество проданных единиц каждого продукта.

var result = orders.GroupBy(order => order.Product);

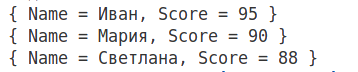


**Задание** 16: Использование **OrderByDescending** и **Take**

**Данные**: Список студентов и их оценки в виде массива объектов:

**Условие**: Найдите топ-3 студента с наивысшими оценками.

var result = students.OrderByDescending(student => student.Score).Take(3).ToList();



**Задание** 17: Использование **Any** с **условиями**

**Данные**: Список продуктов в виде массива объектов:

**Условие**: Проверьте, есть ли доступные продукты по цене выше 90.

var result = products.Any(product => product.Price > 90);

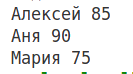


**Задание** 18: Использование **Zip**

**Данные**: Списки студентов и их оценок в виде массивов объектов:

**Условие**: Создайте новый список объектов с именем студента и его оценкой, используя метод Zip.

var result = students.Zip(scores, (first\_name, score) => first\_name + " " + score);



**Задание** 19: Использование **TakeWhile**

**Данные**: Список чисел в виде массива:

**Условие**: Возьмите числа из списка, пока они меньше 6.

var result = numbers.TakeWhile(num => num < 6);



**Задание** 20: Использование **Aggregate**

**Данные**: Список доходов в виде массива:

**Условие**: Вычислите общую сумму доходов, используя метод Aggregate.

var result = profits.Aggregate((current, next) => current + next);



**Задание** 21: Использование **Select** с **GroupBy**

**Данные**: Список студентов и их оценки в виде массива объектов:

**Условие**: Группируйте оценки по студентам и получите итоговую среднюю оценку для каждого студента.

var result = studentGrades.GroupBy(student => student.Name

).Select(student => new {

Name = student.Key, Average = student.Average(grade => grade.Grade)

}

);



**Контрольные вопросы:**

1. Что такое LINQ? Объясните, что такое LINQ и какие задачи он решает.

Позволяет в декларативной форме описать работу с коллекцией элементов

1. Каковы основные преимущества использования LINQ?

сохраняет намерение

1. Перечислите преимущества использования LINQ по сравнению с традиционными запросами к данным.

краткие и читаемые, особенно при фильтрации нескольких условий

1. В чем разница между LINQ to Objects и LINQ to SQL?

**LINQ to Objects**: применяется для работы с массивами и коллекциями

**LINQ to Entities**:используется при обращении к базам данных через технологию **Entity Framework**

1. Что делает команда First и как она отличается от FirstOrDefault?

First(): Если последовательность пуста или ни один элемент не удовлетворяет условию, First() выдаст исключение InvalidOperationException.

FirstOrDefault(): Если последовательность пуста или ни один элемент не удовлетворяет условию, FirstOrDefault() вернет значение по умолчанию для типа

1. Для чего используется метод Join в LINQ?

Для объеденения коллекций по полям коллекций

1. Как надлежащим образом использовать ToList(), ToArray() и AsEnumerable()? В каких ситуациях следует использовать каждую из этих функций?

ToList(): Когда необходим, поддержка динамических изменений и наличие

ToArray(): нужен доступ по индексу, а результат известен заранее и не изменится.

AsEnumerable(): отложить выполнение запросов до момента фактического обращения к данным.

1. Какие факторы могут повлиять на производительность LINQ-запросов?

Количество элементов в коллекции над которой работает LINQ  
Количество операторов LINQ  
Количество операторов сортировки / группировки

1. Как объединить несколько методов LINQ в одном запросе?  
   Перечислить методы LINQ по порядку

Вывод: Я получил навыки работы фильтрации информации с помощью LINQ запросов.