**Лабораторная работа № 70-71**

**«Язык интегрированных запросов»**

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

**Ход работы**

1. Оптимизировать код.

var result = students.Where(x => x.Age > 20).Select(x => x.Name);

1. Найдите всех сотрудников, работающих в департаменте "IT" (рис. 1).

var result = employees.Where(x => x.Department == "IT");



Рис. 1 – результат выполнения программы

1. Создайте новую коллекцию, которая будет содержать только названия продуктов (рис. 2).

var result = products.Select(x => x.Name);



Рис. 2 – результат выполнения программы

1. Найдите первую книгу, автором которой является "Лев Толстой" (рис. 3).

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



Рис. 3 – результат выполнения программы

1. Сортируйте список автомобилей по году выпуска в порядке возрастания (рис. 4).

var result = cars.OrderBy(car => car.Year);

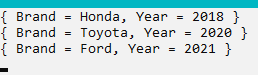


Рис. 4 – результат выполнения программы

1. Группируйте студентов по их оценкам (рис. 5).

var result = students.GroupBy(x => x.Grade).ToList();

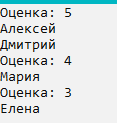


Рис. 5 – результат выполнения программы

1. Подсчитайте количество фильмов в жанре "Фантастика" (рис. 6).

int number = movies.Count(mov => mov.Genre == "Фантастика");



Рис. 6 – результат выполнения программы

1. Убедитесь, есть ли среди клиентов активные клиенты (рис. 7).

bool IsActive = clients.Any(x => x.IsActive);



Рис. 7 – результат выполнения программы

1. Найдите последний фильм в списке по году выпуска (рис. 8).

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



Рис. 8 – результат выполнения программы

1. Получите список уникальных жанров (рис. 9).

var result = genres.Distinct();

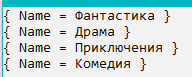


Рис. 9 – результат выполнения программы

1. Возьмите первых 3 студента из списка (рис. 10).

var result = students.Take(3);

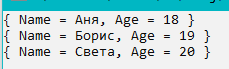


Рис. 10 – результат выполнения программы

1. Пропустите первые 2 книги и выведите оставшиеся (рис. 11).

var result = books.Skip(2);

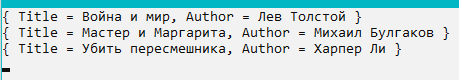


Рис. 11 – результат выполнения программы

1. Получите список всех студентов, зарегистрированных на курсы (рис. 12).

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

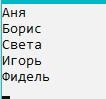


Рис. 12 – результат выполнения программы

1. Соедините заказы с клиентами и получите список заказов с именами клиентов (рис. 13).

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

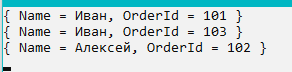


Рис. 13 – результат выполнения программы

1. Создайте список услуг с ценами, указав только названия услуг и их цены (рис. 14).

var result = services.ToList();

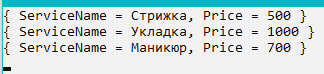


Рис. 14 – результат выполнения программы

1. Группируйте заказы по продуктам и вычислите общее количество проданных единиц каждого продукта (рис. 15).

var result = from x in orders

group new

{

x.Product,

x.Quantity

} by x.Product into xgroup

select new

{

Name = xgroup.Key,

Quantity = xgroup.Sum(x => x.Quantity)

};

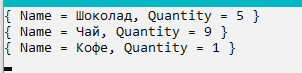


Рис. 15 – результат выполнения программы

1. Найдите топ-3 студента с наивысшими оценками (рис. 16).

var result = (from x in students

orderby x.Score descending

select x).Take(3);



Рис. 16 – результат выполнения программы

1. Проверьте, есть ли доступные продукты по цене выше 90 (рис. 17).

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



Рис. 17 – результат выполнения программы

1. Создайте новый список объектов с именем студента и его оценкой, используя метод Zip (рис. 18).

var result = students.Zip(scores, (x, y) => new { Name = x, Score = y });

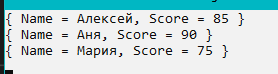


Рис. 18 – результат выполнения программы

1. Возьмите числа из списка, пока они меньше 6 (рис. 19).

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



Рис. 19 – результат выполнения программы

1. Вычислите общую сумму доходов, используя метод Aggregate (рис. 20).

var result = incomes.Aggregate((x, y) => x + y);



Рис. 20 – результат выполнения программы

1. Группируйте оценки по студентам и получите итоговую среднюю оценку для каждого студента (рис. 21).

var result = from x in studentGrades

group new

{

x.Name,

x.Grade

} by x.Name into xgrouped

select new

{

Name = xgrouped.Key,

MeanGrade = xgrouped.Average(x => x.Grade)

};

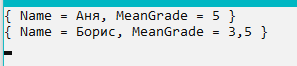


Рис. 21 – результат выполнения программы

**Контрольные вопросы:**

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

LINQ (Язык интегрированных запросов) — это мощный инструмент в C#, который позволяет разработчикам выполнять запросы к различным источникам данных (таким как массивы, коллекции, базы данных и XML) с использованием единого и простого синтаксиса.

1. Каковы основные преимущества использования LINQ?

Удобство и производительность

1. Перечислите преимущества использования LINQ по сравнению с традиционными запросами к данным.
2. В чем разница между LINQ to Objects и LINQ to SQL?

To object - Позволяет выполнять запросы к коллекциям в памяти, таким как массивы и списки;

To SQL - Даёт возможность взаимодействовать с базами данных и выполнять SQL-запросы через объектно-ориентированный подход.

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

Команда они выдает первый элемент списка по условию, если же first ничего не находит он выдает исключение в отличии от FirstOrDefault, который выдает стандартное значение без исключения.

1. Для чего используется метод Join в LINQ?

Для объединения двух разнотипных наборов данных в один.

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

Когда необходим список – ToList;

Когда необходим массив – ToArray;

Когда необходим IEnumerable – AsEnumerable.

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

* тип источника;
* размер источника;
* структура запроса.

1. Как объединить несколько методов LINQ в одном запросе?

После одно запроса можно сразу применить другой:

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

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