***Кафедра комп’ютерної та програмної інженерії***

**ЛАБОРАТОРНА РОБОТА №4**

**з курсу**

**«Універсальна кроссплатформна розробка на платформі .Net Core»**

**Студент: Перловська А. О.**

**Група: ІПЗ 42**

**Факультет: Математики, інформатики**

**та фізики**

**Перевірив: Січкаренко В. О.**

**Лабораторна робота №4**

**Тема:** використання LINQ та PLINQ.

**Мета:** ознайомитись з синтаксисом LINQ та PLINQ. Застосування LINQ та PLINQ в процесі розробки міжплатформного програмного забезпечення на платформі .NET.

1. Ознайомитися з теоретичною частиною;
2. Перейти в локальний репозиторій з лабораторними роботами;
3. Створити окрему гілку (git checkout -b) в системі контролю версій Git для поточної лабораторної роботи;
4. Створити новий консольний проект на мові програмування C# в локальному репозиторії;
5. Визначити та створити необхідні колекції, які будуть використовуватись в обраній темі;
6. Використати мінімум 3 методи розширення вибірки даних (на власний розсуд);
7. Використати мінімум 3 методи розширення зміни порядку даних (на власний розсуд);
8. Використати мінімум 2 методи розширення вибірки даних (на власний розсуд);
9. Використати мінімум 1 метод розширення управління запитами (на власний розсуд);
10. Продемонструвати роботу програми;
11. Результат роботи програми зберегти у вигляді скріншоту (png або jpeg) в репозиторії з проектом;
12. Індексувати проект (git add);
13. Зафіксувати зміни (git commit);
14. Надіслати зміни у віддалений репозиторій (git push);
15. Створити та виконати запит на зміни (pull request);
16. Надіслати посилання на поточну лабораторну роботу у віддаленому репозиторії в GitHub.

**Хід роботи**

Для початку я відкрила командну строку, щоб почати роботу з Git (рис. 1).

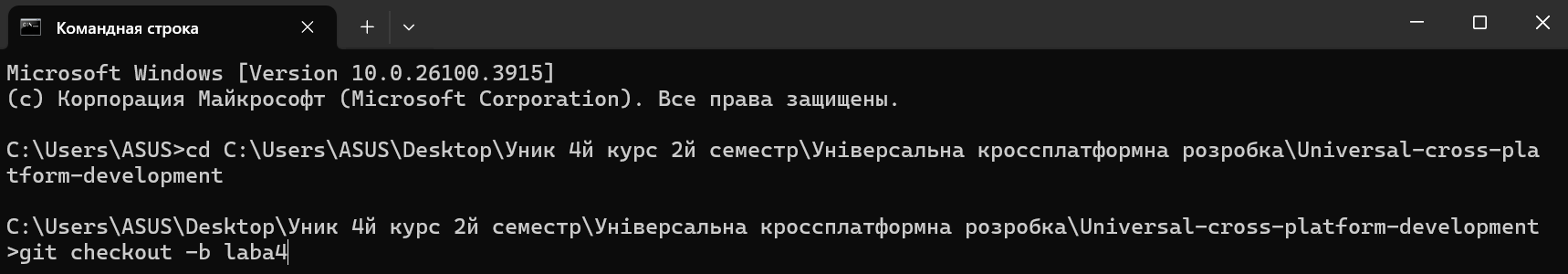
****

Рисунок 1 – Перехід до локального репозиторію та створення нової гілки

Далі я обрала тему №2 «Система бібліографічного обліку» та запустила Visual Studio та обрала відповідний тип проекту, додала назви та налаштувала його.

Потім я написала код (рис. 2-4) для консольного застосунку на мові програмування C#. У програмі моделюється список книг, які зберігаються у бібліотеці.

Я створила клас Book, який має властивості: Title (назва книги), Author (автор), Genre (жанр) та Year (рік видання).

У методі Main() я створила список книг List<Book> з шістьма об'єктами. Далі використано різні методи розширення LINQ, щоб показати можливості роботи з колекціями:

* Методи вибірки даних
* **Where()** – для вибірки книг певного жанру. Наприклад, я вивів усі книги жанру «Роман»: **var romans = books.Where(b => b.Genre == "Роман")**;
* **Select()** – для отримання лише назв книг (без решти даних): **var titles = books.Select(b => b.Title)**;
* **Take()** – для вибірки перших **n** елементів із колекції. Наприклад, я вивів **перші 3 книги за алфавітом** за назвою: **var firstThree = books.OrderBy(b => b.Title).Take(3**).
* Метод управління запитами
* **Any()** – для перевірки, чи є хоча б одна книга, видана до 1950 року: **bool oldBooksExist = books.Any(b => b.Year < 1950)**.
* Методи зміни порядку даних
* **OrderBy()** – сортування книг за алфавітом: **var firstThree = books.OrderBy(b => b.Title).Take(3)**;
* **OrderByDescending()** – сортування книг за роком видання у спадному порядку: **var sortedByYear = books.OrderByDescending(b => b.Year)**;
* **Reverse()** – перевертання порядку елементів у списку: **var reversed = books.AsEnumerable().Reverse()**;
* **ThenBy()** – додаткове сортування (спочатку за жанром, потім за роком): **var multiSorted = books.OrderBy(b => b.Genre).ThenBy(b => b.Year)**.

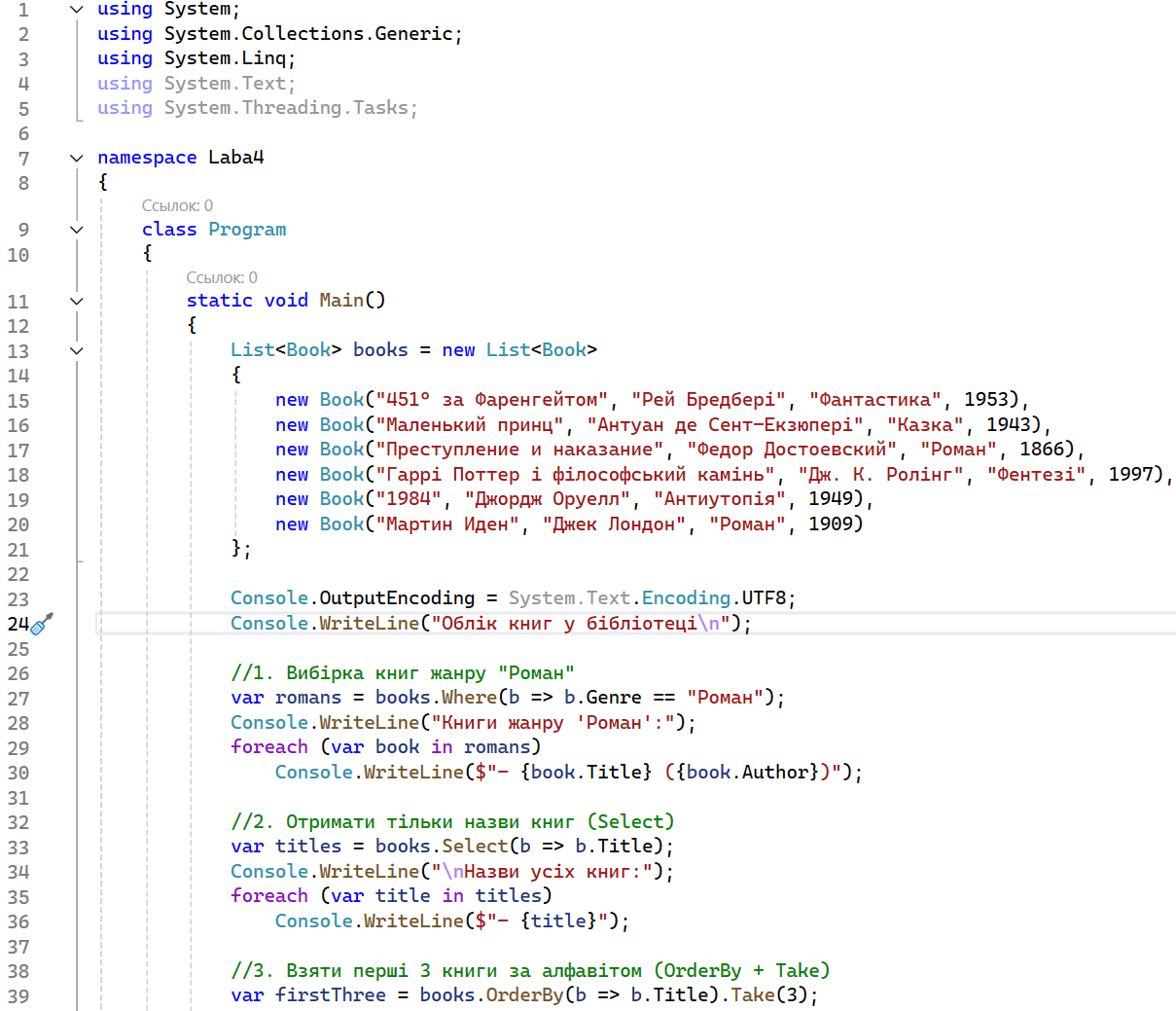


Рисунок 2 – Початок роботи з кодом

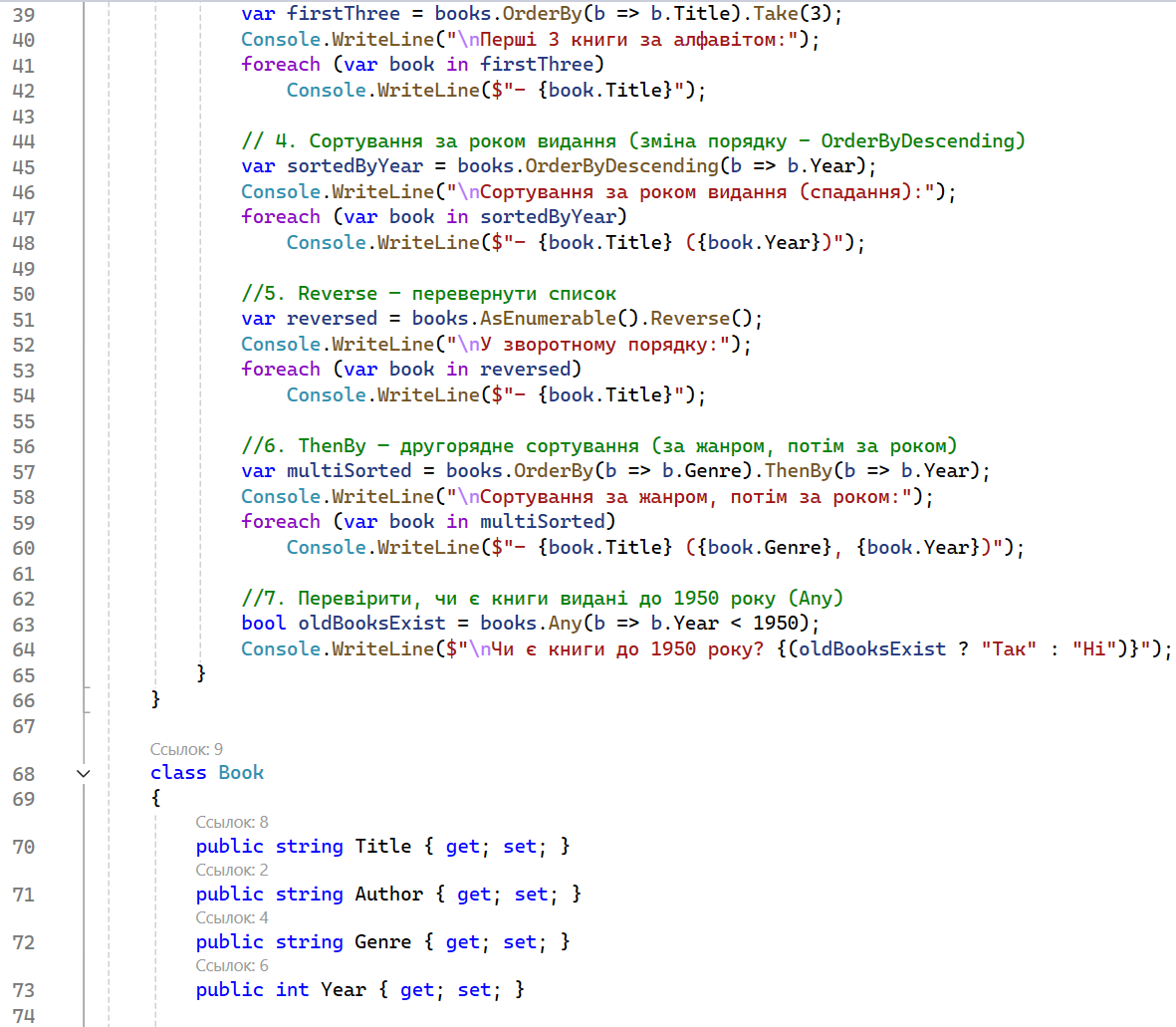


Рисунок 3 – Продовження роботи з кодом

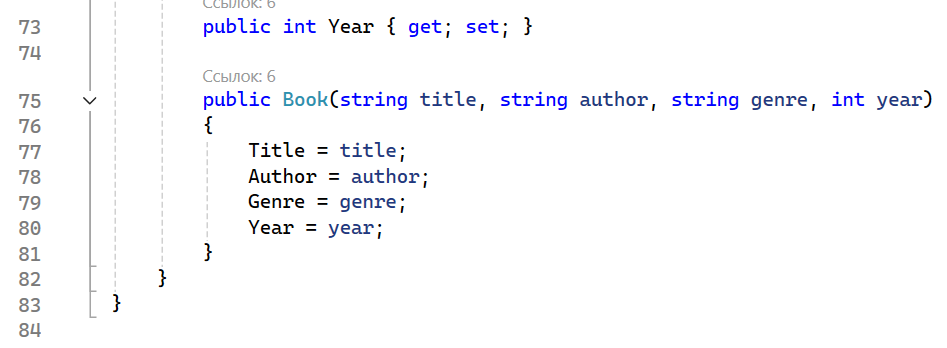


Рисунок 4 – Завершення роботи з кодом

На рисунку 5 показано приклад, як працює цей додаток:

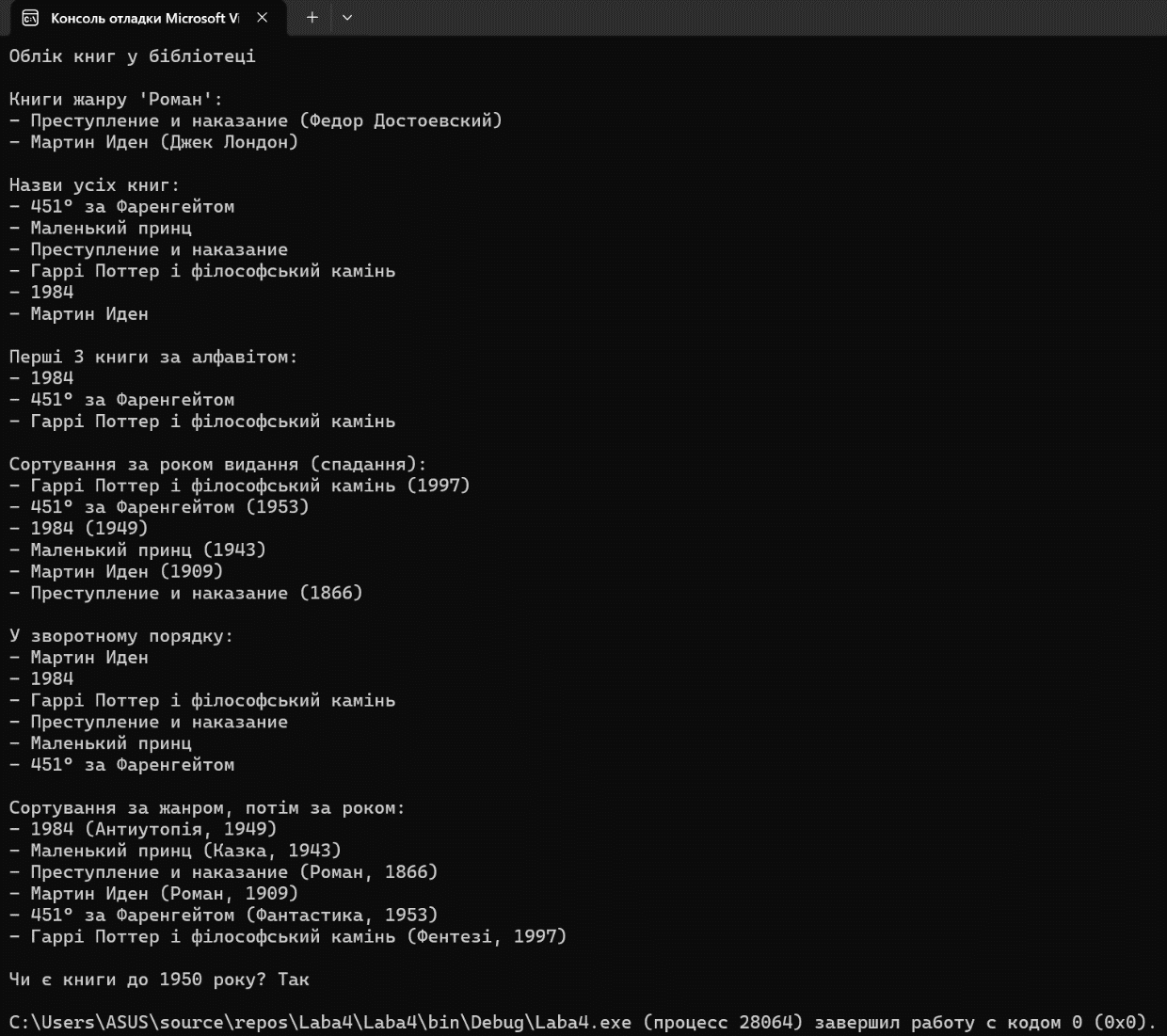


Рисунок 5 – Демонстрація роботи

Після того, як написання програми було завершене я додала в папку скріншот з демонстрацією роботи програми за допомогою команди git status,індексувала проект через git add; зафіксувала зміни, використовуючи git commit та надіслала зміни у віддалений репозиторій за допомогою git push (рис. 6).

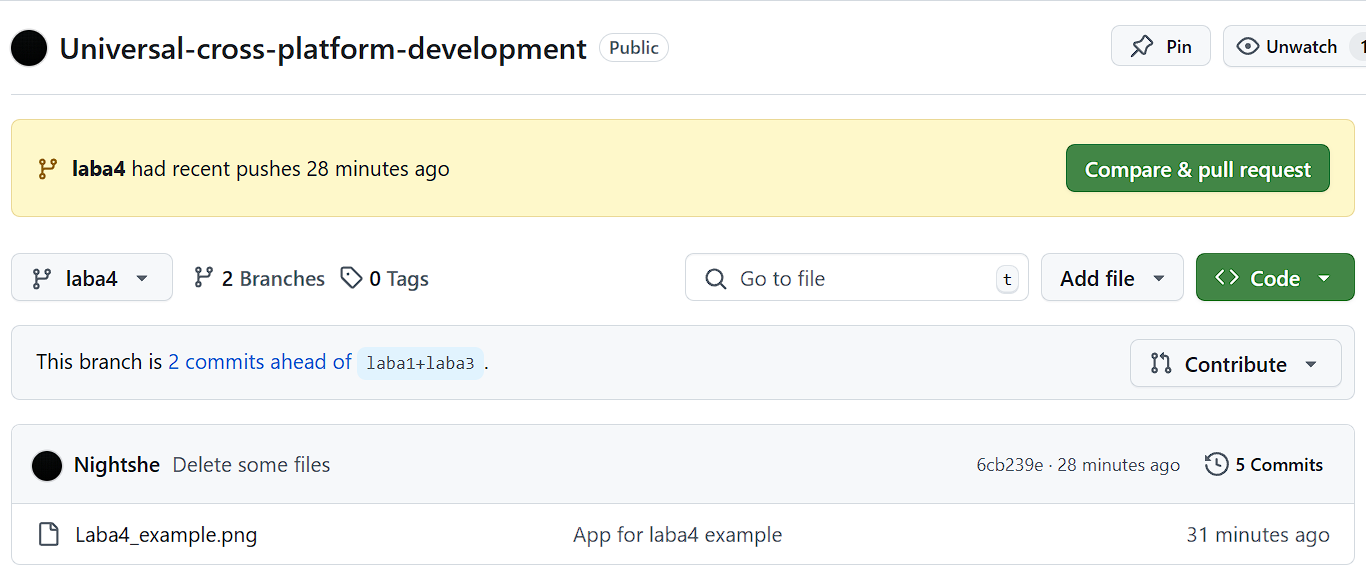


Рисунок 6 – Результат додавання в репозиторій

**Висновок:** у цій лабораторній роботі я навчилась використовувати LINQ-запити до списків об'єктів у C#. Я застосувала різні методи розширення для вибірки, сортування та перевірки умов. Завдяки LINQ код став зрозумілим і легко читається, особливо цей підхід зручний для фільтрації та обробки даних у програмах.

**Контрольні запитання**

1. **Основні операції опрацювання даних:**

* фільтрація (Where);
* сортування (OrderBy, OrderByDescending);
* вибірка полів (Select);
* групування (GroupBy);
* агрегація (Count, Sum, Average);
* об’єднання (Join);
* перетворення (SelectMany, Cast);
* управління елементами (Take, Skip, First, Single).

1. **Що таке мова запитів? Які існують мови запитів?**

Мова запитів – це формалізований спосіб доступу до даних шляхом написання спеціальних виразів (запитів), які дозволяють фільтрувати, сортувати, групувати тощо.

Приклади мов запитів:

* SQL;
* LINQ (Language Integrated Query);
* XQuery (для XML);
* SPARQL (для RDF);
* GraphQL (для API).

1. **Для яких джерел даних існують розширення LINQ/PLINQ?**

* LINQ to Objects (для масивів і колекцій);
* LINQ to XML;
* LINQ to SQL;
* LINQ to DataSet;
* LINQ to Entities (Entity Framework);
* LINQ to JSON (через бібліотеки);
* PLINQ – для паралельної обробки даних у LINQ to Objects.

1. **Різниця між LINQ to SQL , LINQ to DataSet, LINQ to Entities.**

|  |  |  |
| --- | --- | --- |
| **Назва** | **Джерело даних** | **Особливості** |
| LINQ to SQL | SQL Server | Працює напряму з базою, створює об'єкти з таблиць |
| LINQ to DataSet | ADO.NET DataSet | Працює з даними в пам’яті, зручне для дисконектного доступу |
| LINQ to Entities | Entity Framework | Потужніша ORM, підтримує різні СУБД, гнучка модель |

1. **Різниця між використанням методів розширення та синтаксису запитів LINQ/PLINQ.**

Методи розширення – це методи, що викликаються через крапку (**books.Where(...).Select(...)**).

Синтаксис запитів – схожий на SQL (**from b in books where b.Genre == "Роман" select b**).

Обидва підходи еквівалентні за результатом, різниця – у стилі написання.

1. **Призначення ключового слова let під час використання синтаксису запитів LINQ/PLINQ?**

Ключове слово **let** дозволяє зберігати результат обчислення у змінну всередині запиту. Це покращує читабельність і дозволяє уникнути повторних обчислень.

Приклад:

**from b in books**

**let shortTitle = b.Title.Substring(0, 5)**

**select shortTitle;**

1. **Призначення методів розширення вибірки. Приклади таких методів.**

Ці методи дозволяють фільтрувати або обирати конкретні елементи.

Приклади:

* Where(predicate);
* First(), FirstOrDefault();
* Single(), SingleOrDefault();
* Select(selector);
* Take(n), Skip(n).

1. **Призначення методів розширення зміни порядку. Приклади таких методів.**

Ці методи змінюють порядок елементів.

Приклади:

* OrderBy(keySelector);
* OrderByDescending(keySelector);
* ThenBy(keySelector);
* ThenByDescending(keySelector);
* Reverse().

1. **Призначення методів агрегації. Приклади таких методів.**

Методи, які обчислюють агреговані значення по колекції.

Приклади:

* Count();
* Sum(selector);
* Average(selector);
* Min(selector);
* Max(selector).

1. **Призначення методів управління запитами. Приклади таких методів.**

Методи, що керують кількістю або черговістю елементів.

Приклади:

* Take(n) — взяти перші n елементів;
* Skip(n) — пропустити перші n елементів;
* TakeWhile(condition);
* SkipWhile(condition);
* Distinct() — унікальні значення.