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

**ЛАБОРАТОРНА РОБОТА № 4**

**з курсу**

**«Архітектура програмної платформи .NET»**

**Студент: Бондарєва Є.Ю.**

**Група: № 42 ІПЗ**

**Факультет: Математики, інформатики**

**та фізики**

**Перевірив: Січкаренко В.О.**

Тема: Використання масивів та колекцій даних з загальної системи типів (CTS).

Мета: Ознайомитись з масивами та колекціями з загальної системи типів (CTS). Застосування CTS в процесі розробки міжплатформного програмного забезпечення на платформі .NET.

Загальні завдання:

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. Обрати тему з переліку запропонованих або визначити власну;

5. Створити новий консольний проект на мові програмування C# в локальному репозиторії;

6. Визначити необхідні колекції, які будуть використовуватись в обраній темі;

7. Під час реалізації програми використовувати роботу з елементами (додавання, видалення, оновлення);

8. Продемонструвати роботу програми;

9. Результат роботи програми зберегти у вигляді скріншоту (png або jpeg) в репозиторії з проектом;

10. Індексувати проект (git add);

11. Зафіксувати зміни (git commit);

12. Надіслати зміни у віддалений репозиторій (git push);

13. Створити та виконати запит на зміни (pull request);

14. Надіслати посилання на поточну лабораторну роботу у віддаленому репозиторії в GitHub.

**Хід роботи:**

Тема ообраної мною роботи: Текстовий редактор.

Для цієї лабораторної роботи було створено окрему гілку в Git – laba4 і новий консольний проект на мові програмування C#.

A screenshot of a computer screen

Description automatically generated

Рисунок 1 - Створення гілки і новий консольний проект на мові програмування C#

A screenshot of a computer

Description automatically generated

Рисунок 2 – Створена гілка і проект в провіднику в папці репозиторія

A screenshot of a computer

Description automatically generated

Рисунок 3 – Вікно GitHub з новою гілкою

Далі я відкриваю створений проект в середовищі Visual Studio для написання коду програми для визначення додаткових колекції, які будуть використані у темі проєкту: текстовий редактор та під час реалізації програми використовуватимемо роботу з елементами (додавання, видалення, оновлення). Рішення буде наведено в лістингу коду 1.

Лістинг коду 1:

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

// Створення об'єкта текстового редактора

TextEditor editor = new TextEditor();

bool exit = false;

while (!exit)

{

Console.OutputEncoding = System.Text.Encoding.Unicode;

Console.WriteLine("\nОберіть дію:");

Console.WriteLine("1. Створити новий документ");

Console.WriteLine("2. Додати текст до документа");

Console.WriteLine("3. Очистити документ");

Console.WriteLine("4. Показати вміст документа");

Console.WriteLine("5. Вибрати документ");

Console.WriteLine("6. Видалити документ");

Console.WriteLine("7. Вийти");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

Console.Write("Введіть назву документа: ");

string title = Console.ReadLine();

editor.CreateDocument(title);

break;

case "2":

if (editor.CurrentDocument != null)

{

Console.Write("Введіть текст для додавання: ");

string text = Console.ReadLine();

editor.AddTextToDocument(text);

}

else

{

Console.WriteLine("Документ ще не створено.");

}

break;

case "3":

editor.ClearDocument();

break;

case "4":

editor.DisplayDocument();

break;

case "5":

editor.SelectDocument();

break;

case "6":

editor.DeleteDocument();

break;

case "7":

exit = true;

Console.WriteLine("Вихід з програми...");

break;

default:

Console.WriteLine("Невірний вибір. Спробуйте ще раз.");

break;

}

}

}

}

class Document

{

public string Title { get; set; }

public string Content { get; private set; }

public Document(string title)

{

Title = title;

Content = string.Empty;

}

public void AddText(string text)

{

Content += text + "\n";

}

public void ClearText()

{

Content = string.Empty;

}

public void Display()

{

Console.WriteLine($"--- {Title} ---");

Console.WriteLine(Content == string.Empty ? "Документ порожній" : Content);

}

}

class TextEditor

{

private List<Document> documents; // Колекція документів

public Document CurrentDocument { get; private set; }

public TextEditor()

{

documents = new List<Document>(); // Ініціалізація колекції документів

}

public void CreateDocument(string title)

{

CurrentDocument = new Document(title);

documents.Add(CurrentDocument); // Додаємо новий документ у колекцію

Console.WriteLine($"Документ '{title}' створено.");

}

public void AddTextToDocument(string text)

{

if (CurrentDocument != null)

{

CurrentDocument.AddText(text);

Console.WriteLine("Текст додано до документа.");

}

else

{

Console.WriteLine("Документ не створено.");

}

}

public void ClearDocument()

{

if (CurrentDocument != null)

{

CurrentDocument.ClearText();

Console.WriteLine("Документ очищено.");

}

else

{

Console.WriteLine("Документ не створено.");

}

}

public void DisplayDocument()

{

if (CurrentDocument != null)

{

CurrentDocument.Display();

}

else

{

Console.WriteLine("Документ не створено.");

}

}

public void SelectDocument()

{

Console.WriteLine("Оберіть документ зі списку:");

for (int i = 0; i < documents.Count; i++)

{

Console.WriteLine($"{i + 1}. {documents[i].Title}");

}

Console.Write("Введіть номер документа: ");

if (int.TryParse(Console.ReadLine(), out int index) && index > 0 && index <= documents.Count)

{

CurrentDocument = documents[index - 1];

Console.WriteLine($"Документ '{CurrentDocument.Title}' обрано.");

}

else

{

Console.WriteLine("Невірний номер документа.");

}

}

public void DeleteDocument()

{

if (CurrentDocument != null)

{

documents.Remove(CurrentDocument);

Console.WriteLine($"Документ '{CurrentDocument.Title}' видалено.");

CurrentDocument = null; // Скидаємо поточний документ

}

else

{

Console.WriteLine("Документ не створено.");

}

}

}

}

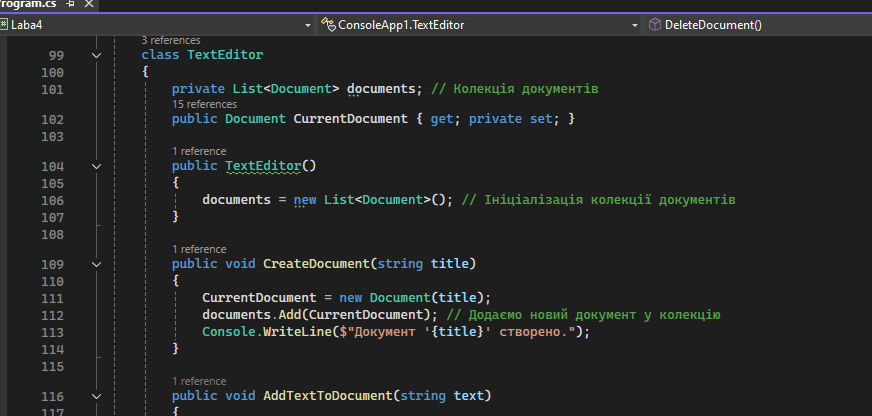


Рисунок 4 – Відображення створеної колекції у проекті в Visual Studio

A computer screen shot of a computer program

Description automatically generated

Рисунок 5 – Компіляція коду для виявлення помилок

Помилок виявлено не було результат виконання програми рис.9-10. Для фіксації проекту на гіті відкриємо його в Visual Studio Code і за допомогою терміналу зафіксуємо на гілці.

A screenshot of a computer program

Description automatically generated

Рисунок 6 – вікно в Visual Studio

Фіксуємо зміни в Git (рис.7).

A screenshot of a computer program

Description automatically generated

Рисунок 7 – Фіксація змін

Перегляд створеного проекту для перевірки в консольному застосуванні як повідомлялось на початку лабораторної роботи.

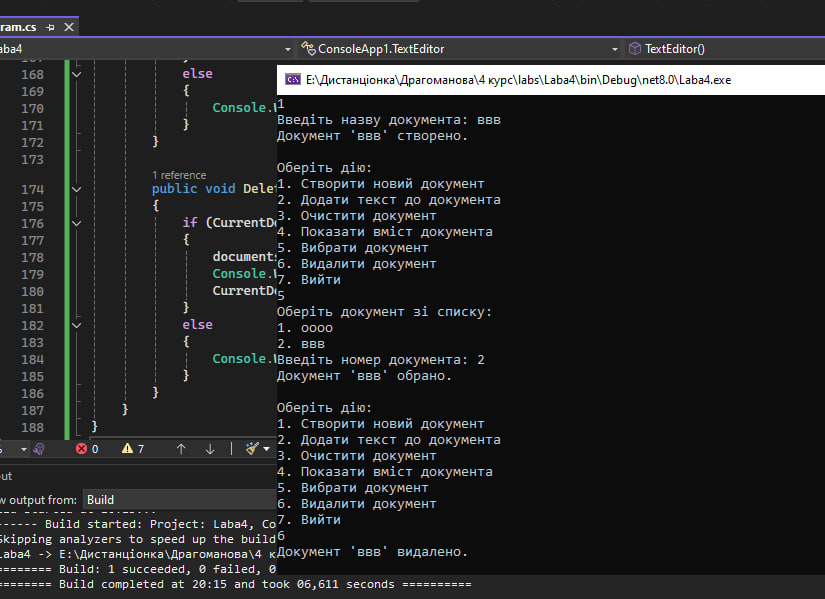


Рисунок 8 – Демонстрація роботи пректу

Поєднуємо код з запиту на зміни з головною гілкою(рис.9).

A screenshot of a computer program

Description automatically generated

Рисунок 9 – Демонстрація роботи проекту

A screenshot of a computer

Description automatically generated

Рисунок 10 – Файли в репозитарії

Надіслати викладачу посилання на створений репозиторій в Git:

<https://github.com/Lizokbondarieva/labs/tree/laba4/Laba4>

Висновки

В результаті виконання лабораторної роботи мною було створено нову гілку на репозиторій створено новий проект по темі «Текстовий редактор». В програмі визначено і створено додаткову колекцію, яку було використано в проєкті, під час реалізації програми використовували роботу з елементами (додавання, видалення, оновлення). В ході виконання лабораторної роботи помилок та труднощів не виникло та було досягнуто поставленого результату.

Контрольні запитання

**1. Що таке масив даних?**

Масив даних – це впорядкована колекція елементів одного типу, що зберігаються в суміжній області пам'яті. Кожен елемент масиву має свій унікальний індекс, який використовується для доступу до нього. Масиви забезпечують ефективний спосіб зберігання та обробки великих обсягів даних одного типу.

**2. Що таке індекс? Що таке індексатор?**

* **Індекс:** Це ціле число, яке вказує на позицію елемента в масиві. Індексація починається з нуля (тобто перший елемент має індекс 0).
* **Індексатор:** Це член класу, який дозволяє отримувати або встановлювати значення елементів об'єкта за допомогою синтаксису, схожого на індексацію масиву. Індексатори забезпечують більш гнучкий доступ до елементів колекцій.

**3. Що таке ініціалізація колекції?**

Ініціалізація колекції – це процес присвоєння початкових значень елементам колекції під час її створення. Ініціалізація може бути як явною (вказання конкретних значень), так і неявною (використання значень за замовчуванням).

**4. Опишіть процес створення масиву даних стандартним способом**

C#

// Створення масиву цілих чисел розміром 5

int[] numbers = new int[5];

// Ініціалізація масиву при створенні

string[] names = new string[] { "Іван", "Марія", "Петро" };

Використовуйте цей код [обачно](file:///C:\faq#coding).

**5. Опишіть процес створення масиву з використанням класу System.Array системи CTS**

Клас System.Array є базовим класом для всіх масивів у C#. Він надає ряд методів для роботи з масивами, але для створення масивів зазвичай використовують синтаксис, описаний у попередньому пункті.

**6. Які інтерфейси реалізує клас System.Array?**

Клас System.Array реалізує кілька інтерфейсів, включаючи ICollection, IEnumerable, і IList. Ці інтерфейси надають додаткові можливості для роботи з масивами, такі як перерахування елементів, пошук елементів та сортування.

**7. Опишіть процес ітерації по елементам масиву, які існують варіанти проходження по елементам колекції?**

* **Цикл for:** Найпоширеніший спосіб перебору елементів масиву за індексом.
* **Цикл foreach:** Більш зручний спосіб перебору елементів без явного використання індексів.
* **Методи LINQ:** Забезпечують потужні можливості для фільтрації, проектування та агрегування даних в масивах.

**8. Призначення простору імен System.Collections. Які простори імен включені в простір імен System.Collections?**

Простір імен System.Collections містить інтерфейси та класи, які представляють різноманітні колекції, такі як списки, хеш-таблиці, стеки та черги. Він є основою для створення власних колекцій.

**9. Призначення IComparer, Comparer, IComparable**

* **IComparer:** Інтерфейс, який використовується для визначення користувацького порівняння об'єктів.
* **Comparer:** Статичний клас, який містить методи для створення об'єктів IComparer.
* **IComparable:** Інтерфейс, який дозволяє об'єктам порівнювати себе з іншими об'єктами того ж типу.

**10. Класифікація колекцій з простору імен System.Collections**

Колекції з простору імен System.Collections можна класифікувати за різними критеріями:

* **За типом доступу:**
  + **Списки:** ArrayList, List<T>
  + **Словники:** Hashtable, Dictionary<TKey, TValue>
  + **Черги:** Queue
  + **Стеки:** Stack
* **За порядком елементів:**
  + **Впорядковані:** SortedList
  + **Неупорядковані:** ArrayList, Hashtable

```

**Використані джерела інформації:**

1. https://moodle.fmif.udu.edu.ua/mod/assign/view.php?id=31274