

Міністерство освіти і науки України
Державний університет „Житомирська політехніка”

Кафедра ІПЗ
Група: ІПЗ-21-2

Технології розробки додатків .NET Core
Лабораторна робота №2
«Використання методів розширень та узагальнень в C#

Виконав:

Троцюк Ю.М.

Прийняв:

Чижмотря О.В.

					22.121.26.000 – Лр2					
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи			Літ.	Арк.	Аркушів
Розроб.		Троцюк Ю.М.								
Перевір.		Чижмoтpя O.B							1	8
Керівник								ФІКТ, гр. ІПЗ-21-2		
Н. контр.										
Затверд.										

Мета роботи: навчитися використовувати методи розширень та узагальнень в мові програмування C#.

Виконання роботи:

2.1. Реалізувати методи розширення для класу string. Інвертування рядка та підрахунок кількості входжень заданого у параметрі символа в рядок.

Лістинг класу StringExtensions:

```
//Методи розширення для класу string
namespace DotNetLab2
{
    public static class StringExtensions
    {
        //Інвертування рядка
        public static string? InverToString(this string? str)
        {
            string? strNew = null;

            if (str != null)
                for (int i = str.Length - 1; i >= 0; i--)
                    strNew += str[i];
            return strNew;
        }

        //Підрахунок кількості входжень заданого у параметрі символа у рядок
        public static int CharCount(this string? str, char ch)
        {
            int count = 0;
            if (str != null)
                for (int i = 0; i < str.Length; i++)
                    if (ch == str[i])
                        count++;
            return count;
        }
    }
}
```

Лістинг класу Program:

```
using System;
using DotNetLab2;

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

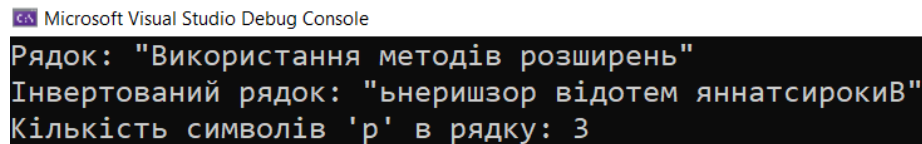
string? text = "Використання методів розширень";
Console.WriteLine($"Рядок: \"{text}\"");
```

					22.121.26.000 – Лр2	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		2

```
//Інвертування рядка
string? inv_text;
inv_text = text.InverToString();
Console.WriteLine($"Інвертований рядок: \"{inv_text}\"");

//Підрахунок кількості входжень заданого у параметрі символа у рядок
char a = 'p';
int count = text.CharCount(a);
Console.WriteLine($"Кількість символів '{a}' в рядку: {count}\n");
```

Результат виконання:



```
Microsoft Visual Studio Debug Console
Рядок: "Використання методів розширень"
Інвертований рядок: "ьнеришзор відотем яннатсирокиВ"
Кількість символів 'р' в рядку: 3
```

Рис.1 – Демонстрація методів розширення для string

2.2. Реалізувати методи розширення для одновимірних масивів. Метод, що визначає скільки разів зустрічається задане значення у масиві (метод має працювати для одновимірних масивів усіх типів). Метод, що повертає новий масив такого ж типу і формує його з унікальних елементів (видаляє повтори);

Лістинг класу *OneDimensionalArrayExnestsions*:

```
//Методи розширення для одновимірних масивів

namespace DotNetLab2
{
    public static class OneDimensionalArrayExnestsions
    {
        //Визначення скільки разів повторюється задане значення в масиві
        public static int ValueCount<T>(this T[] array, T value)
        {
            int count = 0;
            if (array != null)
                for (int i = 0; i < array.Length; i++)
                    if (value.ToString()==array[i].ToString())
                        count++;
            return count;
        }
        //Видалення повторів масиву
        public static T[] UnoqueArray<T>(this T[] array)
        {
            return array.Distinct().ToArray();
        }
    }
}
```

Лістинг класу *Program*:

					22.121.26.000 – Лр2	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

int[] array = {3,3,4,6,3,4,9};
Console.Write("Масив: ");
for (int i = 0; i < array.Length; i++)
    Console.Write($"{array[i]} ");

//Визначення скільки разів повторюється задане значення в масиві
int value = 4;
int calc = array.ValueCount(value);
Console.WriteLine($"{value} повторюється {calc} разів");

//Видалення повторів масиву
int[] new_n = array.UnoqueueArray();
Console.Write("Новий масив: ");
for(int i = 0; i < new_n.Length; i++)
    Console.Write($"{new_n[i]} ");

```

Результат виконання:

```

Масив: 3 3 4 6 3 4 9
Значення '4' повторюється 2 разів
Новий масив: 3 4 6 9

```

Рис.2 – Демонстрація методів розширення для одновимірного масиву

Також ці методи розширення працюють і для інших типів даних:

```

string[] array = {"bool", "int", "string", "int", "int", "char"};
Console.Write("Масив: ");
for (int i = 0; i < array.Length; i++)
    Console.Write($"{array[i]} ");

//Визначення скільки разів повторюється задане значення в масиві
string value = "int";
int calc = array.ValueCount(value);
Console.WriteLine($"{value} повторюється {calc} разів");

//Видалення повторів масиву
string[] new_n = array.UnoqueueArray();
Console.Write("Новий масив: ");
for(int i = 0; i < new_n.Length; i++)
    Console.Write($"{new_n[i]} ");

```

Результат виконання:

```

Масив: bool int string int int char
Значення 'int' повторюється 3 разів
Новий масив: bool int string char

```

Рис.3 – Демонстрація методів розширення для одновимірного масиву з іншим типом даних

3. Реалізувати узагальнений клас для зберігання розширеного словника. ExtendedDictionary, де T - тип даних ключа, U - тип даних першого значення, V - тип даних другого значення. Передбачити операції: додавання елемента у словник; видалення елемента з словника за заданим ключем; перевірка наявності елемента із заданим ключем; перевірка наявності елемента із заданим значенням (значення1 та значення2); повернення елемента за заданим ключем (реалізувати операцію індексування); властивість, що повертає кількість елементів; Представлення елемента словника реалізувати у вигляді окремого класу ExtendedDictionaryElement, передбачивши властивості для доступу до ключа, першого та другого значення. Словник повинен мати можливість використання у циклах foreach: foreach(var elem in array) { ... }

Лістинг класу ExtendedDictionaryElement:

```
namespace DotNetLab2
{
    public class ExtendedDictionaryElement<T, U, V>
    {
        public T Key { get; set; }
        public U First_value { get; set; }
        public V Second_value { get; set; }
        public static int CountElements { get; set; }

        public ExtendedDictionaryElement(T key, U first_value, V second_value)
        {
            Key = key;
            First_value = first_value;
            Second_value = second_value;

            CountElements += 1;
        }

        public static ExtendedDictionaryElement<T, U, V>[]
        AddElement(ExtendedDictionaryElement<string, string, string>[] array, T key, U
        first_value, V second_value)
        {
            ExtendedDictionaryElement<T, U, V>[] new_array = new
            ExtendedDictionaryElement<T, U, V>[array.Length + 1];
            Array.Copy(array, new_array, array.Length);
            new_array[array.Length] = new ExtendedDictionaryElement<T, U, V>(key,
            first_value, second_value);
            return new_array;
        }

        public static ExtendedDictionaryElement<T, U, V>[]
        RemoveElement(ExtendedDictionaryElement<string, string, string>[] array, T key)
        {
            int index = -1;
            for (int i = 0; i < array.Length; i++)
```

					22.121.26.000 – Лр2	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

```

        if (array[i].Key.ToString() == key.ToString())
            index = i;
        ExtendedDictionaryElement<T, U, V>[] array_new = new
ExtendedDictionaryElement<T, U, V>[array.Length - 1];
        if (index >= 0)
        {
            Array.Copy(array, 0, array_new, 0, index);
            Array.Copy(array, index + 1, array_new, index, array.Length - index - 1);
            CountElements--;
        }
        return array_new;
    }

    public static int CheckByKey(ExtendedDictionaryElement<string, string, string>[]
array, T key)
    {
        int index = -1;
        for (int i = 0; i < array.Length; i++)
            if (array[i].Key.ToString() == key.ToString())
                index = i;
        return index;
    }

    public static int CheckByValues(ExtendedDictionaryElement<string, string,
string>[] array, U first_value, V second_value)
    {
        int index = -1;
        for (int i = 0; i < array.Length; i++)
            if (array[i].First_value.ToString() == first_value.ToString() &&
array[i].Second_value.ToString() == second_value.ToString())
                index = i;

        return index;
    }

    public static void PrintDictionary(ExtendedDictionaryElement<string, string,
string>[] array)
    {
        Console.WriteLine("\n\nРозширений словник:");
        foreach (var elem in array)
        {
            Console.WriteLine($"Ключ: {elem.Key}\t Значення: {elem.First_value}\t
Значення: {elem.Second_value}\t");
        }
        Console.WriteLine($"Кількість записів: {CountElements}");
    }

    public static void PrintElement(ExtendedDictionaryElement<string, string, string>
elem)
    {
        Console.WriteLine("Елемент:");
        Console.WriteLine($"Ключ: {elem.Key}\t Значення: {elem.First_value}\t
Значення: {elem.Second_value}\t");
    }
}

```

Лістинг класу Program:

```

//Створення словника з ініціалізація першого елемента, та виведення в консоль
ExtendedDictionaryElement<string, string, string>[] dictionary = new
ExtendedDictionaryElement<string, string, string>[1];

```

					22.121.26.000 – Лр2	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

```

dictionary[0] = new ExtendedDictionaryElement<string, string, string>("cry", "кричати",
"плакати");
ExtendedDictionaryElement<string, string, string>.PrintDictionary(dictionary);

//Додавання елементів в словник, та виведення в консоль
dictionary = ExtendedDictionaryElement<string, string, string>.AddElement(dictionary,
"connect", "з'єднувати", "зв'язувати");
dictionary = ExtendedDictionaryElement<string, string, string>.AddElement(dictionary,
"cat", "кошеня", "кіт");
ExtendedDictionaryElement<string, string, string>.PrintDictionary(dictionary);

//Видалення елемента за заданим ключем
dictionary = ExtendedDictionaryElement<string, string, string>.RemoveElement(dictionary,
"connect");
ExtendedDictionaryElement<string, string, string>.PrintDictionary(dictionary);

//Перевірка наявності елемента із заданим ключем (якщо знайдено, то повертається індекс.
Інакше -1).
//В разі успіху виводимо цей елемент в консоль
int indx = ExtendedDictionaryElement<string, string, string>.CheckByKey(dictionary,
"cry");
Console.WriteLine($"\\n\\nІндекс знайденого елемента за ключем: {indx}");
if(indx>=0)
    ExtendedDictionaryElement<string, string, string>.PrintElement(dictionary[indx]);

//Перевірка наявності елемента із заданими значеннями 1 та 2 (якщо знайдено, то
повертається індекс. Інакше -1).
//В разі успіху виводимо цей елемент в консоль
indx = ExtendedDictionaryElement<string, string, string>.CheckByValues(dictionary,
"кошеня", "кіт");
Console.WriteLine($"\\n\\nІндекс знайденого елемента за значеннями: {indx}");
if (indx >= 0)
    ExtendedDictionaryElement<string, string, string>.PrintElement(dictionary[indx]);

```

Результат виконання:

```

Розширений словник:
Ключ: cry          Значення: кричати      Значення: плакати
Ключ: connect      Значення: з'єднувати   Значення: зв'язувати
Ключ: cat          Значення: кошеня      Значення: кіт
Кількість записів: 3

Розширений словник:
Ключ: cry          Значення: кричати      Значення: плакати
Ключ: cat          Значення: кошеня      Значення: кіт
Кількість записів: 2

Індекс знайденого елемента за ключем: 0
Елемент:
Ключ: cry          Значення: кричати      Значення: плакати

Індекс знайденого елемента за значеннями: 1
Елемент:
Ключ: cat          Значення: кошеня      Значення: кіт

```

Рис.4 – Реалізація узагальненого класу “Елемент розширеного словника”

Висновок: На даній лабораторній роботі було засвоєно такі теми, як псевдоніми для класів, статичний імпорт класів, методи розширення, локальні функції, деконструктори, узагальнення, обмеження узагальнень, nullable-типи, операція null-об'єднання, операція умовного null, об'єкти в switch. Та закріплено під час виконання даної лабораторної роботи.

					22.121.26.000 – Лр2	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		