**Лабораторна робота №2**

**Тема:** Використання методів розширень та узагальнень у C#.

**Мета роботи:** навчитися використовувати методи розширення та узагальнення у

мові програмування C#.

**Хід роботи**

1. Реалізувати методи розширення:

● для класу String:

- інвертування рядка;

- підрахунок кількості входжень заданого у параметрі символа у рядок.

● для одновимірних масивів:

- метод, що визначає скільки разів зустрічається задане значення у

масиві (метод має працювати для одновимірних масивів усіх типів, для

реалізації даного методу розширення використайте узагальнення та їх

обмеження за допомогою “where”);

- метод, що повертає новий масив такого ж типу і формує його з

унікальних елементів (видаляє повтори);

● Написати код для демонстрації роботи реалізованих методів

розширення.

2. Реалізувати узагальнені класи для:

● Реалізувати узагальнений клас для зберігання “розширеного словника”

(для ключа передбачається два значення).

ExtendedDictionary<T, U, V>, 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 пови-

нен мати можливість використання у циклах foreach: foreach(var elem in array) { …

} ● Написати код для демонстрації роботи з реалізованими узагальненими класа-

ми.

**Лістинг Program.cs:**

namespace Lab2

{

static class Program

{

private static void Main()

{

var s= "good morning.";

Console.WriteLine("String: " + s);

Console.WriteLine("Inverted string: " + s.InvertString());

Console.WriteLine("Amount of 'o' letters: " + s.SymbolCount('o'));

Console.WriteLine();

var array = new[] {1, 3, 3, 5, 3, 6, 7, 3, 9};

array.PrintArray();

Console.WriteLine("Amount of '3' in array: " + array.ArrayEntriesCount(3));

Console.WriteLine("Array with removed copies: ");

array.RemoveCopies().PrintArray();

Console.WriteLine();

string[] array2 = { "good", "morning", "good", "excelent", "morning" };

array2.PrintArray();

Console.WriteLine("Amount of 'morning' in array: " + array2.ArrayEntriesCount("morning"));

Console.WriteLine("Array with removed copies: ");

array2.RemoveCopies().PrintArray();

Console.WriteLine();

ExtendedDictionary<string, int, int> d = new ExtendedDictionary<string, int, int>

{

{ "a", 12, 23 },

{ "b", 4, 10 },

};

d.Add("c", 4, 5);

Console.WriteLine("COUNT: " + d.Count);

Console.WriteLine($"{d[1].Key} {d[1].FirstValue} {d[1].SecondValue}");

Console.WriteLine("Check if contain value 12 and 22: " + d.ContainsValue(12, 23));

Console.WriteLine("Check if contain key 'c': " + d.ContainsKey("c"));

Console.WriteLine("Check if contain key 'q': " + d.ContainsKey("q"));

Console.WriteLine($"{d["a"].Key} {d["a"].FirstValue} {d["a"].SecondValue}");

d["a"] = new ExtendedDictionaryElement<string, int, int> { Key = "a", FirstValue = 16, SecondValue = 7 };

Console.WriteLine($"{d["a"].Key} {d["a"].FirstValue} {d["a"].SecondValue}");

foreach (var item in d)

Console.WriteLine($"Key: {item.Key} FirstValue: {item.FirstValue} SecondValue: {item.SecondValue}");

d.Remove("c");

Console.WriteLine("Removed element with key 'c'.");

foreach (var item in d)

Console.WriteLine($"Key: {item.Key} FirstValue: {item.FirstValue} SecondValue: {item.SecondValue}");

Console.ReadKey();

}

private static string InvertString(this string s)

{

char[] charArray = s.ToCharArray();

Array.Reverse(charArray);

return new string(charArray);

}

private static int SymbolCount(this string s, char c)

{

int count = 0;

foreach (char symbol in s)

if (symbol == c)

count++;

return count;

}

private static int ArrayEntriesCount<T>(this T[] array, T o)

{

int count = array.Count(entry => entry.Equals(o));

return count;

}

private static T[] RemoveCopies<T>(this T[] array)

{

return array.Distinct().ToArray();

}

private static void PrintArray<T>(this T[] array)

{

foreach(var item in array)

Console.Write(item + " ");

Console.WriteLine();

}

}

}

**Лістинг ExtendedDictionary.cs:**

using System.Collections;

namespace Lab2

{

internal class ExtendedDictionary<T, U, V> : IEnumerable<ExtendedDictionaryElement<T, U, V>>

{

private List<ExtendedDictionaryElement<T, U, V>> \_list = new List<ExtendedDictionaryElement<T, U, V>>();

public ExtendedDictionary()

{

\_list = new List<ExtendedDictionaryElement<T, U, V>>();

}

public void Add(T key, U firstValue, V secondValue)

{

if (ContainsKey(key))

{

throw new ArgumentException("An element with the same key already exists in the dictionary");

}

\_list.Add(new ExtendedDictionaryElement<T, U, V>

{

Key = key,

FirstValue = firstValue,

SecondValue = secondValue

});

}

public void Remove(T key)

{

\_list.RemoveAll(element => element.Key.Equals(key));

}

public bool ContainsKey(T key) => \_list.Exists(element => element.Key.Equals(key));

public bool ContainsValue(U firstValue, V secondValue) =>

\_list.Exists(element => element.FirstValue.Equals(firstValue) && element.SecondValue.Equals(secondValue));

public int Count => \_list.Count;

public ExtendedDictionaryElement<T, U, V> this[T key]

{

get => \_list.Find(element => element.Key.Equals(key));

set => \_list[\_list.FindIndex(element => element.Key.Equals(key))] = value;

}

public ExtendedDictionaryElement<T, U, V> this[int index]

{

get => \_list[index];

set => \_list[index] = value;

}

public IEnumerator<ExtendedDictionaryElement<T, U, V>> GetEnumerator()

{

return \_list.GetEnumerator();

}

IEnumerator IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

}

public class ExtendedDictionaryElement<T, U, V>

{

public T Key { get; set; }

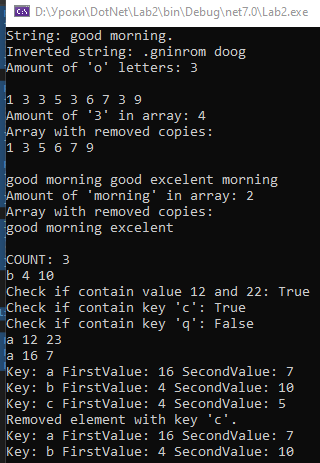
public U FirstValue { get; set; }

public V SecondValue { get; set; }

}

}

**Результат виконання програми:**



**Репозиторій GitLab:** <https://gitlab.com/2022-2026/ipz-22-4/denis-volodymyrovych-petrov/dotnetlab_02_petrov>

**Висновок:** навчився використовувати методи розширення та узагальнення у

мові програмування C#.