

## Лабораторна робота №2

Тема: Використання методів розширень та узагальнень у C#.

Завдання:

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.

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<T, U, V>, передбачивши властивості для доступу до ключа, першого та другого значення.

Словник повинен мати можливість використання у циклах foreach:

```
foreach(var elem in array) { ... }
```

- Написати код для демонстрації роботи з реалізованими узагальненими класами.

Лістинг коду:

```
using System;
using System.Linq;

namespace DotNetLab2._2
{
```

```

public static class StringExtension
{
    public static string ReverseString(this string str)
    {
        char[] chars = str.ToCharArray();
        Array.Reverse(chars);
        return new string(chars);
    }
    public static int Count(this string input , char c)
    {
        return input.Split(c).Length - 1;
    }
}

class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Reverse");

        Console.WriteLine("----");

        string str = "Reverse";
        string reverse = str.ReverseString();
        Console.WriteLine(reverse);

        Console.WriteLine("----");

        string input = "Reverse";
        int freq = input.Count('l');
        Console.WriteLine(freq);

        Console.WriteLine("----");

        Console.WriteLine("Array: 2, 5, 4, 4, 3, 1, 6");
        int[] array = new int[] { 2, 5, 4, 4, 3, 1, 6 };
        var k = 4;
        var col = array.Where(x => x == k).Count();
        Console.WriteLine($"Element 4 repeated {col} times");
        int[] distinct = array.Distinct().ToArray();

        Console.WriteLine(String.Join(",", distinct));

    }
}

```

Результат роботи:



```
Reverse
-----
esreveR
-----
0
-----
Array: 2, 5, 4, 4, 3, 1, 6
Element 4 repeated 2 times
2,5,4,3,1,6

C:\Users\bzagl\source\repos\DotNetLab2\DotNetLab2.2\bin\Debug\net5.0\DotNetLab2.2.exe (process 11356) exited with code 0
.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Рис 1. Результат виконання

Лістинг коду DictionaryClass:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Collections;

namespace Dictionary
{
    class DictionaryClass<Tkey,Tvalue> : IEnumerable
    {
        private List<Item<Tkey, Tvalue>> Items = new List<Item<Tkey, Tvalue>>();

        private List<Tkey> Keys = new List<Tkey>();
        public int Count => Items.Count;

        public DictionaryClass()
        {
        }

        public void Add(Item<Tkey, Tvalue> item)
        {
            if(!Keys.Contains(item.Key))
            {
                Keys.Add(item.Key);
                Items.Add(item);
            }
        }

        public Tvalue Search (Tkey key)
        {
            if (Keys.Contains(key))
            {
                return Items.Single(i => i.Key.Equals(key)).Value;
            }
            return default(Tvalue);
        }
        public void Remove(Tkey key)
        {
            if (Keys.Contains(key))
            {

```

```

        Keys.Remove(key);
        Items.Remove(Items.Single(i => i.Key.Equals(key)));
    }

}

IEnumerator IEnumerable.GetEnumerator()
{
    return Items.GetEnumerator();
}
}
}

```

### Лістинг коду Item:

```

using System.Collections;

namespace Dictionary
{
    class Item<TKey, TValue>
    {
        public TKey Key { get; set; }
        public TValue Value { get; set; }

        public Item(TKey key , TValue value)
        {
            Key = key;
            Value = value;
        }

        public override int GetHashCode()
        {
            return Key.GetHashCode();
        }
        public override string ToString()
        {
            return Value.ToString();
        }
    }
}

```

### Лістинг коду Program

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dictionary
{
    class Program
    {
        static void Main(string[] args)
        {
            var DictionaryClass = new DictionaryClass<int, string>();

```

```

DictionaryClass.Add(new Item<int, string>(1, "One"));
DictionaryClass.Add(new Item<int, string>(2, "Two"));
DictionaryClass.Add(new Item<int, string>(3, "Three"));
DictionaryClass.Add(new Item<int, string>(4, "Four"));
DictionaryClass.Add(new Item<int, string>(5, "Five"));

foreach (var item in DictionaryClass)
{
    Console.WriteLine(item);
}

Console.WriteLine(DictionaryClass.Search(7) ?? "Not found");
Console.WriteLine(DictionaryClass.Search(1) ?? "Not found");

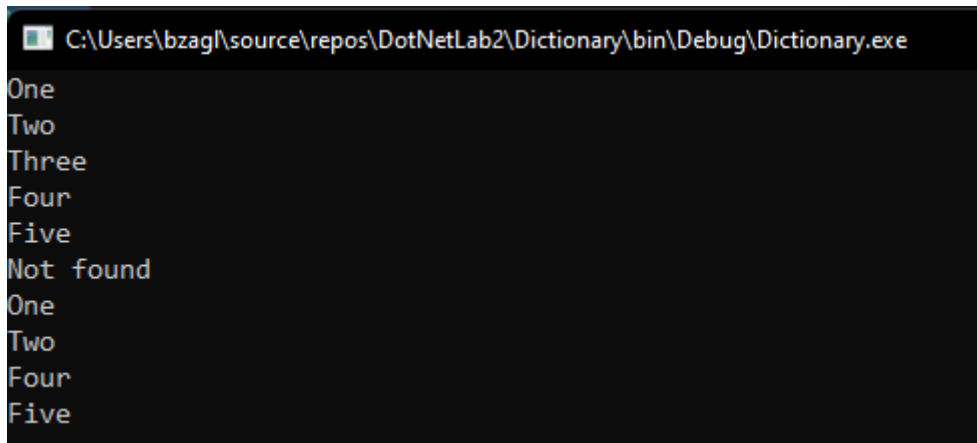
DictionaryClass.Remove(3);
DictionaryClass.Remove(1);

foreach (var item in DictionaryClass)
{
    Console.WriteLine(item);
}

Console.ReadLine();
}
}
}

```

Результат виконання:



```

C:\Users\bzagl\source\repos\DotNetLab2\Dictionary\bin\Debug\Dictionary.exe
One
Two
Three
Four
Five
Not found
One
Two
Four
Five

```

Рис 2. Результат виконання

Висновок: в ході виконання лабораторної роботи навчився використовувати методи розширень та узагальнення у C#.

Посилання на GitLab: <https://gitlab.com/2021-2025/ipz-21-2/zagladko-bohdan/dotnetlab2>