

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
Національний технічний університет України
Факультет інформатики та обчислювальної техніки
«Київський політехнічний інститут імені Ігоря Сікорського»
Кафедра інформаційних систем та технологій

Звіт
з лабораторної роботи № 3
«Колекції C#. Списки. Словники. JSON. LINQ»
з дисципліни
«Програмування - 2. Структури даних та алгоритми»

Варіант № 23

Перевірів:

доц. Корнага Ярослав Ігорович

Виконала: Павлова Софія

Студентка гр. ІС-12 , ФІОТ

1 курс,

залікова книжка № ІС-1224

Лабораторна робота № 3

Тема: Колекції C#. Списки. Словники. JSON. LINQ.

Мета: Вивчити принципи ООП, списки та опанувати LINQ.

Обладнання: Персональні комп'ютери.

ЗАВДАННЯ

Завдання 1.

Написати програму згідно отриманого завдання використовуючи колекції C#.

Варіант 23:

23. Створити список з елементів папки на вашому комп'ютері (користувач може обрати папку) і її підпапок

Код:

Program.cs

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

namespace proga_2_lab_3_1
{
    class Program
    {
        static void Main(string[] args)
        {
            int index = 0;
            bool choosed = false;

            Folder[] _folder =
            {
                new Folder() { Name = "C:\\", Num = 0 },
                new Folder() { Name = "Download", Num = 1 },
                new Folder() { Name = "Users", Num = 2 },
                new Folder() { Name = "ProgramData", Num = 3 },
                new Folder() { Name = "Documents", Num = 4 },
                new Folder() { Name = "Windows", Num = 5 },
                new Folder() { Name = "TelegramDesktop", Num = 6 },
                new Folder() { Name = "Admin", Num = 7 },
                new Folder() { Name = "Common", Num = 8 },
                new Folder() { Name = "Apps", Num = 9 },
                new Folder() { Name = "MyDocuments", Num = 10 },
                new Folder() { Name = "Microsoft", Num = 11 },
                new Folder() { Name = "Intel", Num = 12 },
                new Folder() { Name = "...", Num = 13 },
                new Folder() { Name = "", Num = 14 },
            }
        }
    }
}
```

```

};
_folder[0].PreviousFolder = _folder[13];
_folder[1].PreviousFolder = _folder[0];
_folder[2].PreviousFolder = _folder[0];
_folder[3].PreviousFolder = _folder[0];
_folder[4].PreviousFolder = _folder[0];
_folder[5].PreviousFolder = _folder[0];
_folder[6].PreviousFolder = _folder[1];
_folder[7].PreviousFolder = _folder[2];
_folder[8].PreviousFolder = _folder[2];
_folder[9].PreviousFolder = _folder[5];
_folder[10].PreviousFolder = _folder[4];
_folder[11].PreviousFolder = _folder[3];
_folder[12].PreviousFolder = _folder[3];
_folder[13].PreviousFolder = _folder[14];
_folder[14].PreviousFolder = _folder[14];

while (true)
{
    if (choosed)
    {
        Console.Clear();
        if (index > 12 || index < 0)
Console.WriteLine("No such folder!");
        else
        {
            Console.WriteLine("You entered \" " +
_folder[index].Name + "\" folder:");
            if (_folder[index].PreviousFolder.Num <= 5
&& _folder[index].PreviousFolder.Num >= 1) Console.WriteLine("No
included folders!");
            foreach (Folder fol in _folder)
                if (fol.PreviousFolder ==
_folder[index]) Console.WriteLine(fol);
        }
        Console.WriteLine("\n-----Catalog---
-----");
        foreach (Folder fol in _folder)
            if (fol.PreviousFolder.Num == 0 ||
fol.PreviousFolder.Num == 13) Console.WriteLine(fol);
        Console.WriteLine("-----
-----");
        Console.WriteLine("Choose the folder to see it's
included folders: ");
        index = Convert.ToInt32(Console.ReadLine());
    }
}

```

```

        choosed = true;
    }
}
}

```

Base.cs

```

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

namespace proga_2_lab_3_1
{
    public class Base<T> where T : Base<T>
    {
        public static Dictionary<Guid, T> Items = new
Dictionary<Guid, T>();

        public Guid Id { get; private set; }

        public Base()
        {
            Id = Guid.NewGuid();
            Items.Add(Id, (T)this);
        }
    }
}

```

Folder.cs

```

using System;

namespace proga_2_lab_3_1
{
    public class Folder : Base<Folder>
    {
        public string Name { get; set; }
        public int Num { get; set; }

        // Попередня папка
    }
}

```

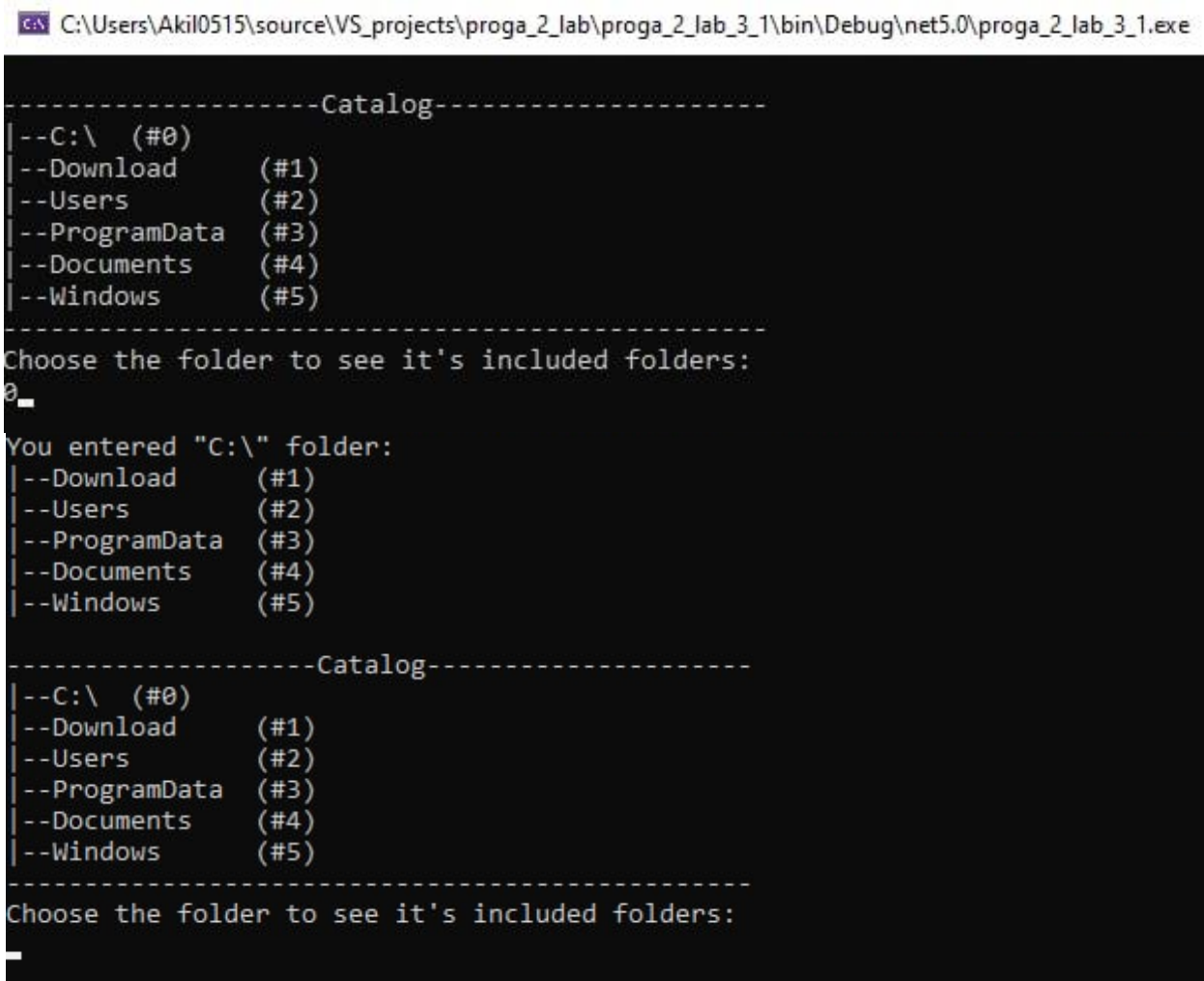
```

private Guid _prevFoldId;
public Folder PreviousFolder
{
    get { return Folder.Items[_prevFoldId]; }
    set { _prevFoldId = value.Id; }
}

public override string ToString()
{
    return "|--" + Name + "\t(#" + Num + ")";
}
}
}

```

Результат:



```

C:\Users\Akil0515\source\VS_projects\proga_2_lab\proga_2_lab_3_1\bin\Debug\net5.0\proga_2_lab_3_1.exe
-----Catalog-----
|--C:\  (#0)
|--Download  (#1)
|--Users    (#2)
|--ProgramData  (#3)
|--Documents  (#4)
|--Windows   (#5)
-----
Choose the folder to see it's included folders:
0_
You entered "C:\" folder:
|--Download  (#1)
|--Users    (#2)
|--ProgramData  (#3)
|--Documents  (#4)
|--Windows   (#5)
-----Catalog-----
|--C:\  (#0)
|--Download  (#1)
|--Users    (#2)
|--ProgramData  (#3)
|--Documents  (#4)
|--Windows   (#5)
-----
Choose the folder to see it's included folders:
_

```

Рис. 1. Виведений результат виконання завдання 1

Завдання 2

Написати програму згідно виданого завдання використовуючи словники Dictionary в C#. Якщо результатом виконання програми є словник, **зберегти** цей результат у JSON файл

Варіант 23:

23. Створити словник з двох списків без втрати дублікатів. Вхідний словник: ['Class-V', 'Class-VI', 'Class-VII', 'Class-VIII'], [1, 2, 2, 3] Результат програми: Словник {'Class-VII': {2}, 'Class-VI': {2}, 'Class-VIII': {3}, 'Class-V': {1}}

Код:

Program.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using Newtonsoft.Json;

namespace proga_2_lab_3_2
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            Dictionary<int, DictionaryItem> items1 =
BuildDictionary(i);
            i++;
            Dictionary<int, DictionaryItem> items2 =
BuildDictionary(i);
            i++;
            Dictionary<int, DictionaryItem> itemsFinal =
BuildDictionary(i);

            // LINQ Query
            IEnumerable<DictionaryItem> subset1 =
MakeQuery(items1);
            IEnumerable<DictionaryItem> subset2 =
MakeQuery(items2);

            for (int j = 0; j < items1.Count; j++)
            {
```

```

        itemsFinal.Add(j,
CombineDictionaries(subset1.ElementAt(j), subset2.ElementAt(j)));
    }

    foreach (DictionaryItem theItem in itemsFinal.Values)
        Console.WriteLine(theItem);

    // Saving to JSON
    var toJson = JsonConvert.SerializeObject(itemsFinal);

File.WriteAllText("C:\\Users\\Akil0515\\source\\VS_projects\\proga_
2_lab\\proga_2_lab_3_2\\FinalList.json", toJson);
    Console.WriteLine("\nJSON:\nFinalList: \n" + toJson);
}

    // Function of creating of LINQ Query
    private static IOrderedEnumerable<DictionaryItem>
MakeQuery(Dictionary<int, DictionaryItem> items)
    {
        return from theItem in items.Values
                orderby theItem.Num
                select theItem;
    }

    private static Dictionary<int, DictionaryItem>
BuildDictionary(int i)
    {
        if (i == 0)
            return new Dictionary<int, DictionaryItem>
            {
                {1,
= 4 }},
                {2,
= 2 }},
                {3,
Num = 1 }},
                {4,
Num = 3 }}
            };
        else if (i == 1)
            return new Dictionary<int, DictionaryItem>
            {

```



```

        { 1,
            new DictionaryItem() { Name="1", Num = 4
        }},
        { 2,
            new DictionaryItem() { Name="2", Num = 2
        }},
        { 3,
            new DictionaryItem() { Name="2", Num = 1
        }},
        { 4,
            new DictionaryItem() { Name="3", Num = 3 }}
    };
    else return new Dictionary<int, DictionaryItem> { };
}

private static DictionaryItem
CombineDictionaries(DictionaryItem lb1, DictionaryItem lb2)
{
    return new DictionaryItem() { Name = lb1 + " " + lb2};
}
}

```

DictionaryItem.cs

```

namespace proga_2_lab_3_2
{
    public class DictionaryItem
    {
        public string Name { get; set; }
        public int Num { get; set; }

        public override string ToString()
        {
            return Name;
        }
    }
}

```

FinalList.json

```
{"0":{"Name":"Class-VII 2","Num":0},  
"1":{"Name":"Class-VI 2","Num":0},  
"2":{"Name":"Class-VIII 3","Num":0},  
"3":{"Name":"Class-V 1","Num":0}}
```

Результат:

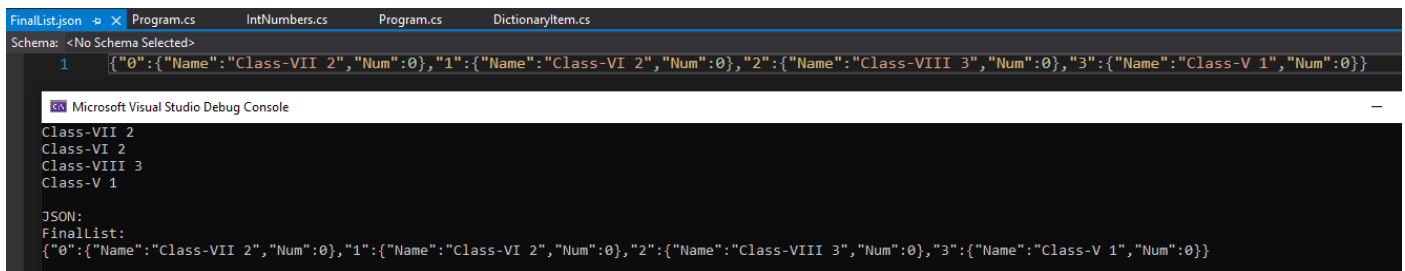


Рис. 2. Виведений результат виконання завдання 2

Завдання 3.

Написати програму згідно виданого завдання використовуючи лише LINQ методи. В кінці завдання в дужках наведена підказка, які методи LINQ могли б вам допомогти у вирішенні задачі

Варіант 23:

23. Дана послідовність додатніх цілих чисел. Обробляючи тільки непарні числа, отримати послідовність їх строкових уявлень і впорядкувати її в лексикографічному порядку по зростанню. (3)

Код:

Program.cs

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
  
namespace proga_2_lab_3_3  
{  
    class Program  
    {
```

```

static void Main(string[] args)
{
    List<IntNumbers> list1 = BuildList();
    // LINQ Query
    IOrderedEnumerable<IntNumbers> subset1 =
MakeQuery(list1);

    Console.WriteLine("List of int numbers:");
    foreach (IntNumbers num in list1)
        Console.Write(num.GetHashCode() + "\t");
    Console.WriteLine("\n\nDue to string values of numbers
sorted list:");
    foreach (IntNumbers num in subset1)
        Console.Write(num.GetHashCode() + "\t");
    Console.WriteLine("\n\n");
}

// Function of creating of LINQ Query
private static IOrderedEnumerable<IntNumbers>
MakeQuery(List<IntNumbers> items)
{
    return from theItem in items
           where theItem.Num % 2 != 0
           orderby theItem.Num.ToString()
           select theItem;
}

private static List<IntNumbers> BuildList()
{
    return new List<IntNumbers>
    {
        { new IntNumbers() { Num = 0 }},
        { new IntNumbers() { Num = 15 }},
        { new IntNumbers() { Num = 2 }},
        { new IntNumbers() { Num = 3 }},
        { new IntNumbers() { Num = 4 }},
        { new IntNumbers() { Num = 23 }},
        { new IntNumbers() { Num = 6 }},
        { new IntNumbers() { Num = 137 }},
        { new IntNumbers() { Num = 8 }},
        { new IntNumbers() { Num = 1 }},
        { new IntNumbers() { Num = 995 }}
    };
}
}
}

```

IntNumbers.cs

```
namespace proga_2_lab_3_3
{
    public class IntNumbers
    {
        public int Num { get; set; }

        public override int GetHashCode()
        {
            return Num;
        }
    }
}
```

Результат:

Microsoft Visual Studio Debug Console

```
List of int numbers:
0      15      2      3      4      23      6      137      8      1      995
Due to string values of numbers sorted list:
1      137      15      23      3      995
```

Рис. 3. Виведений результат виконання завдання 3

Висновки:

У ході виконання лабораторної роботи я дізналась, про методи LINQ у мові програмування C#, та колекції, а саме списки та словники. Зрозуміла причини та принципи їх реалізації.

Я навчилась на практиці реалізовувати списки та словники, використовувати методи LINQ для сортування елементів колекцій і розв'язувати за допомогою них поставлені задачі.