Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни
«Сучасні технології розробки WEB-застосувань на платформі Microsoft.NET»

«Узагальнені типи (Generic) з підтримкою подій. Колекції»

Виконав *ІС-13,Харчук А.В.* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Бардін В.* _____
(прізвище, ім'я, по батькові)

Київ 2023

Варіант 5

Завдання

5	Динамічний массив	Див. List<T>	Збереження даних за допомогою динамічно зв’язаного списку

Код програми

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;

namespace CollectionRealisation
{
    public class MyList<T> : IList<T> , ICollection<T>
    {
        //List

        //use this capacity with ctor()
        private const int DefaultCapacity = 4;

        //_items.Length
        private int _capacity;

        //the arr, where data is stored
        private T[] _items;

        //num of elements is allredy set to _items
        private int _size;

        //reference to ctor with int param and pass default capacity
        public MyList():this(DefaultCapacity)
        {

        }

        //set _capacity, _size and create arr with user's capacity
        public MyList(int capacity)
        {
            _capacity = capacity;
            _items = new T[_capacity];
            _size = 0;
        }

        //override the [] to referance to the item by index
        public T this[int index]
```

```
{
    get {
        //check if index is valid
        if (!IndexIsBetweenZeroAndSize(index))
        {
            throw new IndexOutOfRangeException();
        }
        return _items[index]; }
    set { _items[index] = value;}
}

public int Count => _size;

public bool IsReadOnly => false;

public void Add(T item)
{
    //check if overflow the arr capacity

    if (_size >= _capacity)
    {
        this.Resize();
    }
    _items[_size] = item;
    _size = _size + 1;
}
//set num of items to 0, create new T[] size of capacity
public void Clear()
{
    _size = 0;
    _items = new T[_capacity];
}

//go through items and search for match
public bool Contains(T item)
{
    for (int i = 0; i < _size; i++)
    {
        if (_items[i]is not null && _items[i]!.Equals(item))
        {
            return true;
        }
    }
    return false;
}

//copy data form this list to user's array starting from arrayIndex
public void CopyTo(T[] array, int arrayIndex)
{
    if (array is not null && array.Rank is not 1)
    {
        throw  new InvalidDataException("Arr rank is not 1");
    }
    Array.Copy(_items, 0, array!, arrayIndex, _size);
}


public int IndexOf(T item)
{
    return Array.IndexOf(_items, item);
}

//Insert item to the list and push items with bigger indexes to the tail end
public void Insert(int index, T item)
{
```

```
    if (index<0||index>_size+1)
    {
        throw new ArgumentOutOfRangeException();
    }
    else if (index == _size+1)
    {
        this.Add(item);
    }
    else
    {
        if (_size==_capacity)
        {
            Resize();
        }
        Array.Copy(_items, index, _items, index+1, _size-index);

        _items[index] = item;

        _size++;

    }
}

public bool Remove(T item)
{
    int index = this.IndexOf(item);
    if (index>=0)
    {
        this.RemoveAt(index);
        return true;
    }
    return false;
}

public void RemoveAt(int index)
{
    if (!IndexIsBetweenZeroAndSize(index))
    {
        throw new ArgumentOutOfRangeException();
    }

    if (index < _size)
    {
        Array.Copy(_items, index + 1, _items, index, _size - index);
    }
    _items[_size] = default!;
    _size--;
}

IEnumerator IEnumerable.GetEnumerator()
{
    throw new NotImplementedException();
}
public IEnumerator<T> GetEnumerator()
{
    return new MyListEnumerator<T>(this);
}
private bool IndexIsBetweenZeroAndSize(int index) => index >= 0 && index < _size;

private void Resize()
{
    _capacity *= 2; //  double the capacity
    var newArray = new T[_capacity]; // create arr with new capacity
    Array.Copy(_items, newArray, _size);
    _items = newArray;
```

```
        }

        public override string ToString()
        {
            return string.Format("Count = {0}", _size);
        }

    }
}
```

Код Енумератора

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

namespace CollectionRealisation
{
    internal class MyListEnumerator<T> : IEnumerator<T>
    {
        private IList<T> _list;
        private int _index;

        private T _current;


        internal MyListEnumerator(IList<T> list)
        {
            _list = list;
            _index = 0;



            if (_list.Any() is true)
            {
                _current = _list[_index];
            }
            else
            {
                _current = default!;
            }
        }
        public T Current => _current;

        object IEnumerator.Current => _current!;

        public bool MoveNext()
        {
            if (_index<_list.Count)
            {
                _current = _list[_index];
                _index ++;
                return true;
            }
            return false;
        }
```

```
        public void Reset()
        {
            _index = 0;
            _current = _list[0];
        }

        public void Dispose()
        {

        }
    }
}
```

Код демонстрації виконання програми

```
// See https://aka.ms/new-console-template for more information
using CollectionRealisation;
using CollectionRealisation.ConsoleDemo;

Console.WriteLine("Hello, World!");

//create int list
var myIntList = new MyList<int>() {1,2,3,4,5 };

ShowArrays<int>.Show(myIntList);

//create double list
var myDoubleList = new MyList<double>() { 1.3, 2.3, 3.3, 4.3, 5.3 };

ShowArrays<double>.Show(myDoubleList);

//create char list
var myCharList = new MyList<char>();

for (int i = 0; i < 5; i++)
{
    myCharList.Add((char)(97 + i));
}

ShowArrays<char>.Show(myCharList);


//set char to serch
var charToSerch = 'a';

Console.WriteLine();

//Contains
//check if myCharList Contains charToSerch
if (myCharList.Contains(charToSerch))
{
    Console.WriteLine(string.Format("myCharList contains {0}", charToSerch));
}
else
{
    Console.WriteLine(string.Format("myCharList do not {0}", charToSerch));
}
```

```
Console.WriteLine();

//Insert
Console.WriteLine("insert 'y' to list");

myCharList.Insert(3, 'y');

ShowArrays<char>.Show(myCharList);

//IndexOf
Console.WriteLine("index of 'y' is {0}", myCharList.IndexOf('y'));

ShowArrays<char>.Show(myCharList);


//CopyTo
var destArr = new char[10];

for (int i = 0; i < destArr.Length; i++)
{
    destArr[i] = (char)(110 + i);
}
Console.WriteLine("destArr");
ShowArrays<char>.Show(destArr);

myCharList.CopyTo(destArr, 0);

Console.WriteLine("myCharList.CopyTo(destArr, 0);");
Console.WriteLine();
Console.WriteLine("destArr after CopyTo");

ShowArrays<char>.Show(destArr);


//Remove
Console.WriteLine("remove 'a' from list");

myCharList.Remove(charToSerch);

ShowArrays<char>.Show(myCharList);


//RemoveAt
Console.WriteLine("remove 1-st elem from list");

myCharList.RemoveAt(1);

ShowArrays<char>.Show(myCharList);


//Clear
Console.WriteLine("remove all elems from list");

myCharList.Clear();

ShowArrays<char>.Show(myCharList);
```

Приклад виконання програми

Hello, World!

Count = 5
1
2
3
4
5

Count = 5
1,3
2,3
3,3
4,3
5,3

Count = 5
a
b
c
d
e

myCharList contains a

insert 'y' to list

Count = 6
a
b
c
y
d
e

index of 'y' is 3

Count = 6
a
b
c
y
d
e

destArr

System.Char[]
n
o
p
q
r
s
t

u
v
w

myCharList.CopyTo(destArr, 0);

destArr after CopyTo

System.Char[]
a
b
c
y
d
e
t
u
v
w

remove 'a' from list

Count = 5
b
c
y
d
e

remove 1-st elem from list

Count = 4

b

y

d

e

remove all elems from list

Count = 0