**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**

(шифр, прізвище, ім'я, по батькові)

*ІС-13,Харчук А.В.*

**Перевірив**

(прізвище, ім'я, по батькові)

*Бардін В.*

Київ 2023

Варіант 1

Завдання



Код програми

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Runtime.InteropServices.JavaScript.JSType;

namespace CollectionRealisation

{

public class MyStack<T>: IEnumerable<T>, ICollection

{

//use this capacity with ctor()

private const int DefaultCapacity = 4;

//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 MyStack() : this(DefaultCapacity)

{

}

//set \_capacity, \_size and create arr with user's capacity

public MyStack(int capacity)

{

if (capacity < 0)

{

throw new ArgumentOutOfRangeException(nameof(capacity));

}

\_items = new T[capacity];

\_size = 0;

}

public int Count => \_size;

public bool IsReadOnly => false;

public bool IsSynchronized => false;

public object SyncRoot => this;

public void Clear()

{

Array.Clear(\_items, 0, \_size);

\_size = 0;

}

//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;

}

IEnumerator IEnumerable.GetEnumerator()

{

return new MyStackEnumerator(this);

}

public IEnumerator<T> GetEnumerator()

{

return new MyStackEnumerator(this);

}

//get top element of the stack

public T Peek()

{

if (\_size>0)

{

return \_items[\_size - 1];

}

throw new InvalidOperationException("EmptyStack");

}

//get top element of the stack and remove it from the stack

public T Pop()

{

if (\_size > 0)

{

\_size--;

var result = \_items[\_size];

\_items[\_size] = default(T)!;

return result;

}

throw new InvalidOperationException("EmptyStack");

}

//Push element to the top of the stack

public void Push(T item)

{

if (\_size+1>\_items.Length)

{

Resize();

}

\_items[\_size] = item;

\_size++;

}

private void Resize()

{

var newCapacity = \_items.Length \*2; // double the capacity

var newArray = new T[newCapacity]; // create arr with new capacity

Array.Copy(\_items, newArray, \_size);

\_items = newArray;

}

public override string ToString()

{

return string.Format("Count = {0}", \_size);

}

//copy data form this list to user's array starting from index

public void CopyTo(Array array, int index)

{

ArgumentNullException.ThrowIfNull(array);

if (array is not null && array.Rank is not 1)

{

throw new InvalidDataException("Arr rank is not 1");

}

try

{

Array.Copy(\_items, 0, array!, index, \_size);

Array.Reverse(array!, index, \_size);

}

catch (ArrayTypeMismatchException)

{

throw new ArgumentException(nameof(array));

}

}

public void CopyTo(T[] array, int index)

{

ArgumentNullException.ThrowIfNull(array);

if (array is not null && array.Rank is not 1)

{

throw new InvalidDataException("Arr rank is not 1");

}

int srcIndex = 0;

int dstIndex = index + \_size;

while (srcIndex < \_size)

{

array![--dstIndex] = \_items[srcIndex++];

}

}

public class MyStackEnumerator : IEnumerator<T>

{

private MyStack<T> \_stack;

private int \_index;

private T \_current;

internal MyStackEnumerator(MyStack<T> stack)

{

\_stack = stack;

\_index = stack.Count - 1;

\_current = default(T)!;

}

public T Current => \_current;

object IEnumerator.Current => \_current!;

public bool MoveNext()

{

if (\_index >= 0)

{

\_current = \_stack.\_items[\_index];

\_index--;

return true;

}

return false;

}

public void Reset()

{

\_index = \_stack.Count - 1;

\_current = default!;

}

public void Dispose()

{

}

}

}

}

Код демонстрації виконання програми

using CollectionRealisation;

using CollectionRealisation.ConsoleDemo;

#region StackDemo

var myIntStack = new MyStack<int>();

for (int i = 1; i <= 5; i++)

{

myIntStack.Push(i);

}

ShowIEnumerable<int>.Show(myIntStack);

//create double list

var myDoubleStack = new MyStack<double>();

var rendom = new Random();

for (int i = 1; i <= 5; i++)

{

myDoubleStack.Push(i + rendom.NextDouble());

}

ShowIEnumerable<double>.Show(myDoubleStack);

//create char list

var myCharStack = new MyStack<char>();

for (int i = 0; i < 5; i++)

{

myCharStack.Push((char)(97 + i));

}

ShowIEnumerable<char>.Show(myCharStack);

//Peek

var topElement = myCharStack.Peek();

Console.WriteLine(string.Format("the Peeked element from the stack is {0}", topElement));

ShowIEnumerable<char>.Show(myCharStack);

//Pop

topElement = myCharStack.Pop();

Console.WriteLine(string.Format("the Poped element from the stack is {0}", topElement));

ShowIEnumerable<char>.Show(myCharStack);

Console.WriteLine("#as you see the size of the stack decreases");

Console.WriteLine();

//Push

myCharStack.Push('O');

Console.WriteLine("myCharStack.Push('O');");

ShowIEnumerable<char>.Show(myCharStack);

var charToSerch = 'a';

//Contains

//check if myCharList Contains charToSerch

if (myCharStack.Contains(charToSerch))

{

Console.WriteLine(string.Format("myCharStack contains {0}", charToSerch));

}

else

{

Console.WriteLine(string.Format("myCharStack do not {0}", charToSerch));

}

Console.WriteLine();

//CopyTo

var destArr = new char[10];

for (int i = 0; i < destArr.Length; i++)

{

destArr[i] = (char)(110 + i);

}

Console.WriteLine("destArr");

ShowIEnumerable<char>.Show(destArr);

myCharStack.CopyTo(destArr, 0);

Console.WriteLine("myCharStack.CopyTo(destArr, 0);");

Console.WriteLine();

Console.WriteLine("destArr after CopyTo");

ShowIEnumerable<char>.Show(destArr);

//Clear

Console.WriteLine("remove all elems from stack");

myCharStack.Clear();

ShowIEnumerable<char>.Show(myCharStack);

#endregion

Приклад виконання програми

Count = 5

5

4

3

2

1

Count = 5

5,450724766705878

4,766526057558748

3,3124340314084826

2,508260584470312

1,0434000210018435

Count = 5

e

d

c

b

a

the Peeked element from the stack is e

Count = 5

e

d

c

b

a

the Poped element from the stack is e

Count = 4

d

c

b

a

#as you see the size of the stack decreases

myCharStack.Push('O');

Count = 5

O

d

c

b

a

myCharStack contains a

destArr

System.Char[]

n

o

p

q

r

s

t

u

v

w

myCharStack.CopyTo(destArr, 0);

destArr after CopyTo

System.Char[]

O

d

c

b

a

s

t

u

v

w

remove all elems from stack

Count = 0