**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

**IКНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 8

**З дисципліни:** *“Алгоритми та структури даних”*

**На тему:** *“Лінійні структури даних”*

**Лектор:**

доц. каф. ПЗ

Коротєєва Т.О.

**Виконала:**

ст. групи ПЗ-26

Матолінець Л. А.

**Прийняв:**

асистент каф. ПЗ

Симець І.І.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_\_ .

Львів – 2022

**Тема роботи:** Лінійні структури даних**.**

**Мета роботи**:  Познайомитися з лінійними структурами даних (стек, черга, дек, список) та отримати навички програмування алгоритмів, що їх обробляють.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

**Варіант 7**

**Варіант 7**: черга дійсних.

**ХІД ВИКОНАННЯ**

//myArray.cs

using System;

namespace SortingApp

{

public class myQueue

{

private double[] queue;

private int head = 0;

private int tail = 0;

private int size = 0;

public myQueue() {

queue = new double[0];

}

public myQueue(int size = 5)

{

this.size = size;

queue = new double[size];

}

public void enqueue(double element)

{

if (tail > size)

throw new Exception("Queue Overflow");

queue[tail] = element;

tail++;

}

public void setSize(int size)

{

this.size = size;

}

public double dequeue()

{

if (head == 0)

throw new Exception("Queue is empty");

double element = queue[head];

head++;

return element;

}

public bool isEmpty() {

bool empty = false;

\_ = this.size > 0 ? empty = false : empty = true;

return empty;

}

public double peek() {

if (head == 0)

throw new Exception("Queue is empty");

else

{

return queue[head];

}

}

public int getTail()

{

return this.tail;

}

public bool contains(double elem)

{

bool containsValue = false;

if (head != 0)

{

foreach (var item in queue)

{

if (item == elem)

{

containsValue = true;

break;

}

}

}

return containsValue;

}

public void clear()

{

if (head != 0)

{

for (int i = head; i < tail; i++) queue[i] = 0;

}

size = 0;

head = 0;

tail = 0;

}

public double getMin()

{

double min = queue[0];

foreach(var item in queue) { if (item < min) { min = item; } }

return min;

}

public int getSize() { return tail; }

public double getMax()

{

double max = queue[0];

foreach (var item in queue) { if (item > max) { max = item; } }

return max;

}

public double thirdElemFromBegin()

{

double val = 0;

if (size >= 3) {

for (int i = head; i < 3; i++)

{

val = queue[i];

}

}

return val;

}

public double getSecondFromEnd()

{

double val = 0;

if (size >= 2)

{

int i = tail;

--i;

--i;

val = queue[i];

}

return val;

}

public double elemBeforeMin()

{

double val = 0;

int index = 0;

double min = queue[0];

for (int i = 0; i < size; i++) {

if (queue[i] < min) {

min = queue[i];

index = i;

}

}

--index;

val = queue[index];

return val;

}

public int getCurrentPosition(double x)

{

int index = head;

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

{

if (x == queue[i]) index = i;

}

return index;

}

public double elemAfterMax()

{

double val = 0;

int index = 0;

double max = queue[0];

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

{

if (queue[i] > max)

{

max = queue[i];

index = i;

}

}

val = queue[++index];

return val;

}

public String getQueue()

{

String str = "";

foreach (var item in queue)

{

str += item + "; ";

}

return str;

}

public double[] getQueueArr() { return queue; }

public myQueue merge(myQueue queue1)

{

myQueue myNewQueue = new myQueue(size + queue1.getSize());

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

{

myNewQueue.enqueue(queue[i]);

}

for(int i = 0; i < queue1.getSize(); i++)

{

myNewQueue.enqueue(queue1.getQueueArr()[i]);

}

return myNewQueue;

}

}

}

}

//MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Numerics;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

namespace SortingApp

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

///

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

myQueue queue1;

myQueue queue2;

private void addArray\_Click(object sender, RoutedEventArgs e)

{

int size1 = System.Convert.ToInt32(sizeField.Text);

queue1 = new myQueue(size1);

}

private void addArray\_Click1(object sender, RoutedEventArgs e)

{

int size2 = System.Convert.ToInt32(sizeField.Text);

queue2 = new myQueue(size2);

}

private void addElement\_Click(object sender, RoutedEventArgs e)

{

queue1.enqueue(System.Convert.ToDouble(elemField.Text));

}

private void addElement1\_Click(object sender, RoutedEventArgs e)

{

queue2.enqueue(System.Convert.ToDouble(elemField.Text));

}

private void amount\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void showPosition\_Clicked(object sender, TextChangedEventArgs e)

{

}

private void show\_Click(object sender, RoutedEventArgs e)

{

amount.Text = System.Convert.ToString(queue1.getSize());

min.Text = System.Convert.ToString(queue1.getMin());

max.Text = System.Convert.ToString(queue1.getMax());

firstTask.Text = System.Convert.ToString(queue1.thirdElemFromBegin()) + ", " + System.Convert.ToString(queue1.getSecondFromEnd());

secondTask.Text = System.Convert.ToString(queue1.elemBeforeMin()) + ", " + System.Convert.ToString(queue1.elemAfterMax());

// curPos.Text = System.Convert.ToString(queue1.getTail());

double x = System.Convert.ToDouble(curPos\_Copy.Text);

int pos = queue1.getCurrentPosition(x);

curPos.Text = System.Convert.ToString(pos);

firstQueue.Text = queue1.getQueue();

secondQueue.Text = queue2.getQueue();

}

private void merge\_Click(object sender, RoutedEventArgs e)

{

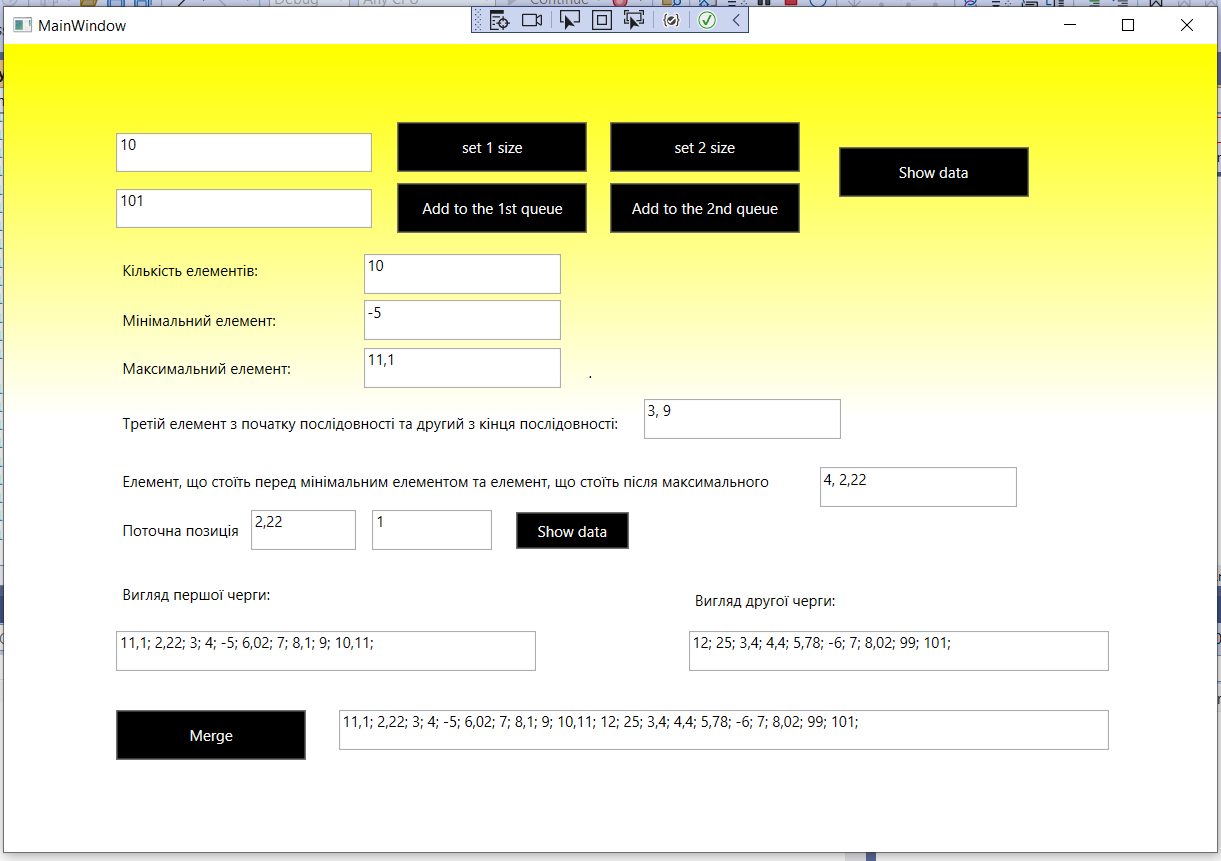
myQueue queue3 = queue1.merge(queue2);

merged.Text = queue3.getQueue();

}

}

}



*Рис. 1 Робота програми з двома структурами даних, розміри яких – 10 та 10*

**Висновок**

На цій лабораторній роботі я працювала з лінійною структурою даних – чергою. Також реалізувала індивідуальне завдання, в якому створила чергу дійсних та реалізувала всі основні функції, передбачені завданням, попри те створила фунції enqueuer(), dequeue(), peek() тощо, які характерні для цієї структури даних.