Министерство науки и высшего образования Российской Федерации

Федеральное государственное образовательное учреждение

высшего образования

**ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра систем автоматизации управления

**Дисциплина: Алгоритмы и структуры данных**

**Отчёт по лабораторной работе №5**

**Тема: «Конструкция условия switch/case. Работа со односвязными и двусвязными списками в C#. Стек, очередь, дек»**

**Вариант 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил студент  группы ИТб 2302-02-20 |  | Ердяков Роман Александрович |  |
|  |  |
| Проверила |  | Кашина Елена Вячеславовна |  |

СОДЕРЖАНИЕ

[1 Задание 3](#_Toc200821511)

[2 Тестирование (задача 1) 4](#_Toc200821512)

[3 Программа на C# (задача 1): 5](#_Toc200821513)

[4 Тестирование (задача 2) 8](#_Toc200821514)

[5 Программа на C# (задача 2): 9](#_Toc200821515)

[6 Тестирование (задача 3) 11](#_Toc200821516)

[7 Программа на C# (задача 3): 12](#_Toc200821517)

[8 Тестирование (задача 4) 16](#_Toc200821518)

[9 Программа на C# (задача 4): 17](#_Toc200821519)

[10 Тестирование (задача 5) 19](#_Toc200821520)

[11 Программа на C# (задача 5): 20](#_Toc200821521)

[Тестирование (задача 6) 24](#_Toc200821522)

[12 Программа на C# (задача 6): 26](#_Toc200821523)

[13 Тестирование (задача 7) 28](#_Toc200821524)

[14 Программа на C# (задача 7): 30](#_Toc200821525)

[15 Тестирование (задача 8) 32](#_Toc200821526)

[16 Программа на C# (задача 8): 33](#_Toc200821527)

[17 Вывод 35](#_Toc200821528)

# Задание

Задача 1.

Создать калькулятор со строкой ввода и кнопками 0,1,…9, +, -, \*, /, =.

Задача 2.

Найти наименьший четный среди положительных элементов массива А(10).

Задача 3.

Реализовать консольное приложение на C# для работы с линейным списком (в зависимости от варианта — List<T> или LinkedList<T>). Список формируется методом вставки элемента в начало. Далее реализуются функции для:

1. проверки, пуст ли список;
2. нахождения среднего арифметического;
3. обмена первого и последнего элементов;
4. проверки упорядоченности элементов;
5. нахождения суммы последних двух элементов;
6. операции над двумя списками: объединение, пересечение, разность, симметрическая разность;
7. слияние отсортированных списков в один отсортированный.

Задача 4.

Реализовать стек и очередь методами классов Stack и Queue (см. лекции или информацию по ссылкам ниже).

Задача 5.

Реализовать дек.

Задача 6.

В поле ListBox вывести в виде списка элементы массива строк. Реализовать ввод массива в richTextBox.

Задача 7.

Считать список из файла1. Изменить его (любым образом). Вывести в файл2. А из файла2 вывести данные в ListBox.

Задача 8.

Считать из ListBox1 информацию в файл. В файле, между первой и второй строкой, вставить строку «Привет!». Считать из файла в список List. Отсортировать строки списка. Используя метод BinarySearch() определить, есть ли строка «Привет!» в списке. Если есть, добавить в этой строке «, Коля», чтобы получилось «Привет, Коля!» и вывести сообщение в окно сообщений «Ура, добавили!». Получившийся список вывести в ListView добавив каждой строке изображения.

# Тестирование (задача 1)

Примеры выполнения программы представлены на рисунке 1, 2.

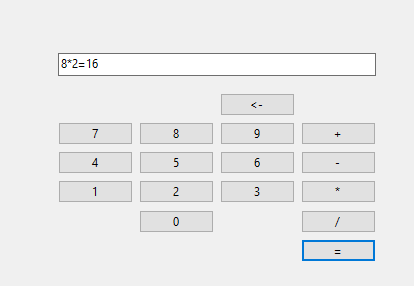


Рисунок 1 – Экранная форма программы

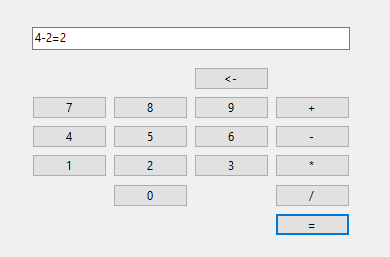


Рисунок 2 – Экранная форма программы

# Программа на C# (задача 1):

using System;

using System.Windows.Forms;

namespace lab5\_1

{

public partial class Form1 : Form

{

private string currentInput = "";

private string expression = "";

private string lastOperator = "";

private double result = 0;

public Form1()

{

InitializeComponent();

AssignButtonHandlers();

}

private void AssignButtonHandlers()

{

// Цифры

button1.Click += (s, e) => AppendDigit("0");

button2.Click += (s, e) => AppendDigit("1");

button3.Click += (s, e) => AppendDigit("2");

button4.Click += (s, e) => AppendDigit("3");

button5.Click += (s, e) => AppendDigit("4");

button6.Click += (s, e) => AppendDigit("5");

button7.Click += (s, e) => AppendDigit("6");

button8.Click += (s, e) => AppendDigit("7");

button9.Click += (s, e) => AppendDigit("8");

button10.Click += (s, e) => AppendDigit("9");

// Операции

button12.Click += (s, e) => SetOperator("+");

button13.Click += (s, e) => SetOperator("-");

button14.Click += (s, e) => SetOperator("\*");

button15.Click += (s, e) => SetOperator("/");

button11.Click += (s, e) => Backspace();

button16.Click += (s, e) => CalculateResult();

}

private void AppendDigit(string digit)

{

currentInput += digit;

expression += digit;

textBox1.Text = expression;

}

private void SetOperator(string op)

{

if (double.TryParse(currentInput, out double parsed))

{

result = parsed;

lastOperator = op;

currentInput = "";

expression += op;

textBox1.Text = expression;

}

}

private void CalculateResult()

{

if (double.TryParse(currentInput, out double secondOperand))

{

double finalResult = result;

switch (lastOperator)

{

case "+":

finalResult += secondOperand;

break;

case "-":

finalResult -= secondOperand;

break;

case "\*":

finalResult \*= secondOperand;

break;

case "/":

if (secondOperand != 0)

finalResult /= secondOperand;

else

{

MessageBox.Show("Деление на ноль!", "Ошибка");

return;

}

break;

default:

return;

}

expression += "=" + finalResult;

textBox1.Text = expression;

// Сброс для следующего выражения

currentInput = finalResult.ToString();

expression = "";

lastOperator = "";

result = finalResult;

}

}

private void Backspace()

{

if (currentInput.Length > 0)

{

currentInput = currentInput.Substring(0, currentInput.Length - 1);

expression = expression.Substring(0, expression.Length - 1);

textBox1.Text = expression;

}

}

}

}

# Тестирование (задача 2)

Программа представляет собой заказ напитков. Пользователь выбирает напиток, размер стакана, количество порций и добавки (сахар, лимон, молоко), а затем при нажатии кнопки "Заказать" сформированный заказ отображается в списке. Примеры выполнения программы представлены на рисунках 3, 4.

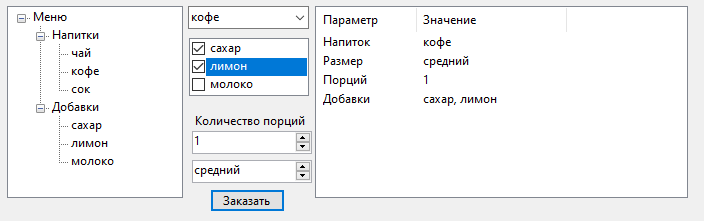


Рисунок 3 – Экранная форма программы

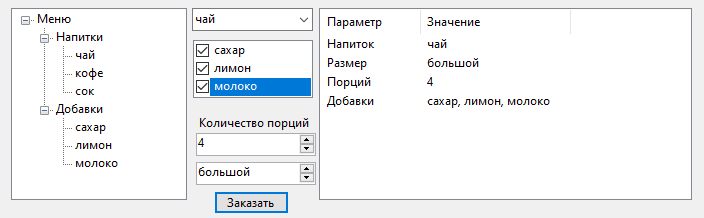


Рисунок 4 – Экранная форма программы

# Программа на C# (задача 2):

using System;

using System.Text;

using System.Windows.Forms;

namespace lab5\_2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

InitializeTreeView();

InitializeListView();

}

private void InitializeTreeView()

{

TreeNode root = new TreeNode("Меню");

TreeNode drinks = new TreeNode("Напитки");

drinks.Nodes.Add("чай");

drinks.Nodes.Add("кофе");

drinks.Nodes.Add("сок");

TreeNode addons = new TreeNode("Добавки");

addons.Nodes.Add("сахар");

addons.Nodes.Add("лимон");

addons.Nodes.Add("молоко");

root.Nodes.Add(drinks);

root.Nodes.Add(addons);

treeView1.Nodes.Add(root);

treeView1.ExpandAll();

}

private void InitializeListView()

{

listView1.View = View.Details;

listView1.Columns.Add("Параметр", 100);

listView1.Columns.Add("Значение", 150);

}

private void button1\_Click\_1(object sender, EventArgs e)

{

listView1.Items.Clear();

string drink = comboBox1.SelectedItem?.ToString() ?? "не выбрано";

string size = domainUpDown1.Text;

int count = (int)numericUpDown1.Value;

// Собираем добавки

StringBuilder addons = new StringBuilder();

foreach (var item in checkedListBox1.CheckedItems)

{

addons.Append(item.ToString() + ", ");

}

if (addons.Length > 0)

addons.Length -= 2; // удаляем последнюю запятую и пробел

// Добавляем в ListView

listView1.Items.Add(new ListViewItem(new[] { "Напиток", drink }));

listView1.Items.Add(new ListViewItem(new[] { "Размер", size }));

listView1.Items.Add(new ListViewItem(new[] { "Порций", count.ToString() }));

listView1.Items.Add(new ListViewItem(new[] { "Добавки", addons.ToString() }));

}

}

}

# Тестирование (задача 3)

Примеры выполнения программы представлены на рисунках 5, 6.

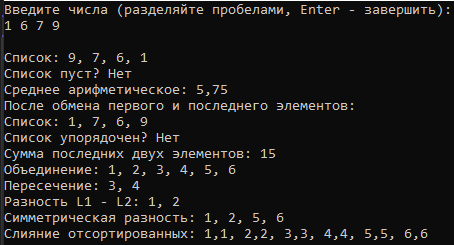


Рисунок 5 – Экранная форма программы

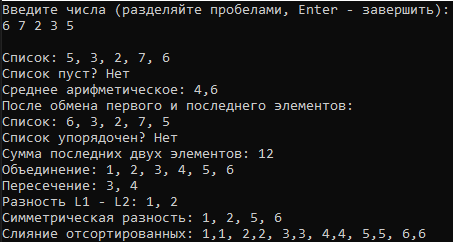


Рисунок 6 – Экранная форма программы

# Программа на C# (задача 3):

using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

static void Main()

{

LinkedList<double> list = new LinkedList<double>();

Console.WriteLine("Введите числа (разделяйте пробелами, Enter — завершить):");

while (true)

{

string input = Console.ReadLine();

if (string.IsNullOrWhiteSpace(input))

break;

var parts = input.Split(' ', StringSplitOptions.RemoveEmptyEntries);

foreach (var part in parts)

{

if (double.TryParse(part, out double value))

list.AddFirst(value);

else

Console.WriteLine($"'{part}' — не число");

}

}

Console.WriteLine("Список: " + GetListString(list));

Console.WriteLine("Список пуст? " + (IsEmpty(list) ? "Да" : "Нет"));

if (!IsEmpty(list))

{

Console.WriteLine("Среднее арифметическое: " + Average(list));

SwapFirstLast(list);

Console.WriteLine("После обмена первого и последнего элементов:");

Console.WriteLine("Список: " + GetListString(list));

Console.WriteLine("Список упорядочен? " + (IsSorted(list) ? "Да" : "Нет"));

if (list.Count >= 2)

Console.WriteLine("Сумма последних двух элементов: " + SumLastTwoElements(list));

else

Console.WriteLine("Недостаточно элементов для вычисления суммы двух последних.");

}

// Демонстрация операций со списками

LinkedList<int> L1 = new LinkedList<int>(new[] { 1, 2, 3, 4 });

LinkedList<int> L2 = new LinkedList<int>(new[] { 3, 4, 5, 6 });

Console.WriteLine("Объединение: " + GetListString(Union(L1, L2)));

Console.WriteLine("Пересечение: " + GetListString(Intersection(L1, L2)));

Console.WriteLine("Разность L1 - L2: " + GetListString(Difference(L1, L2)));

Console.WriteLine("Симметрическая разность: " + GetListString(SymmetricDifference(L1, L2)));

LinkedList<double> sorted1 = new LinkedList<double>(new[] { 1.1, 3.3, 5.5 });

LinkedList<double> sorted2 = new LinkedList<double>(new[] { 2.2, 4.4, 6.6 });

Console.WriteLine("Слияние отсортированных: " + GetListString(MergeSorted(sorted1, sorted2)));

}

static string GetListString<T>(LinkedList<T> list)

{

return string.Join(", ", list);

}

static bool IsEmpty<T>(LinkedList<T> list) => list.Count == 0;

static double Average(LinkedList<double> list)

{

if (list.Count == 0) return 0;

double sum = 0;

foreach (var item in list)

sum += item;

return sum / list.Count;

}

static void SwapFirstLast<T>(LinkedList<T> list)

{

if (list.Count < 2) return;

var first = list.First;

var last = list.Last;

T temp = first.Value;

first.Value = last.Value;

last.Value = temp;

}

static bool IsSorted(LinkedList<double> list)

{

if (list.Count <= 1) return true;

var current = list.First;

while (current.Next != null)

{

if (current.Value > current.Next.Value)

return false;

current = current.Next;

}

return true;

}

static int SumLastTwoElements(LinkedList<double> list)

{

var last = list.Last.Value;

var secondLast = list.Last.Previous.Value;

return (int)last + (int)secondLast;

}

static LinkedList<int> Union(LinkedList<int> L1, LinkedList<int> L2)

{

var set = new HashSet<int>(L1);

foreach (var item in L2)

set.Add(item);

return new LinkedList<int>(set);

}

static LinkedList<int> Intersection(LinkedList<int> L1, LinkedList<int> L2)

{

var set1 = new HashSet<int>(L1);

var result = new LinkedList<int>();

foreach (var item in L2)

if (set1.Contains(item))

result.AddLast(item);

return new LinkedList<int>(result.Distinct());

}

static LinkedList<int> Difference(LinkedList<int> L1, LinkedList<int> L2)

{

var set2 = new HashSet<int>(L2);

var result = new LinkedList<int>();

foreach (var item in L1)

if (!set2.Contains(item))

result.AddLast(item);

return new LinkedList<int>(result.Distinct());

}

static LinkedList<int> SymmetricDifference(LinkedList<int> L1, LinkedList<int> L2)

{

var set1 = new HashSet<int>(L1);

var set2 = new HashSet<int>(L2);

var result = new LinkedList<int>();

foreach (var item in set1)

if (!set2.Contains(item))

result.AddLast(item);

foreach (var item in set2)

if (!set1.Contains(item))

result.AddLast(item);

return result;

}

static LinkedList<double> MergeSorted(LinkedList<double> L1, LinkedList<double> L2)

{

var merged = new List<double>(L1);

merged.AddRange(L2);

merged.Sort();

return new LinkedList<double>(merged);

}

}

# Тестирование (задача 4)

Примеры выполнения программы представлены на рисунках 7, 8.

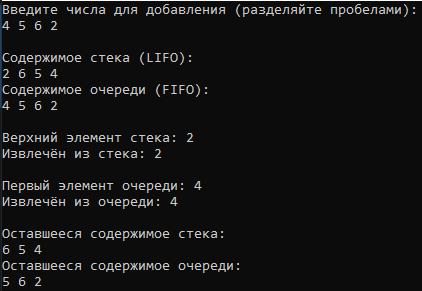


Рисунок 7 – Экранная форма программы

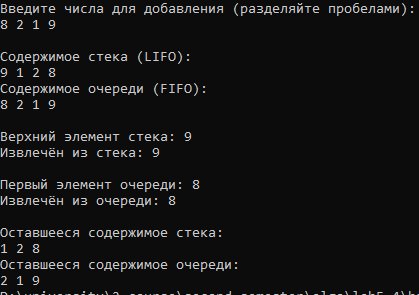


Рисунок 8 – Экранная форма программы

# Программа на C# (задача 4):

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

Stack<int> stack = new Stack<int>();

Queue<int> queue = new Queue<int>();

Console.WriteLine("Введите числа для добавления (разделяйте пробелами):");

string input = Console.ReadLine();

string[] parts = input.Split(' ', StringSplitOptions.RemoveEmptyEntries);

foreach (var part in parts)

{

if (int.TryParse(part, out int number))

{

stack.Push(number);

queue.Enqueue(number);

}

else

{

Console.WriteLine($"'{part}' — не число");

}

}

Console.WriteLine("\nСодержимое стека (LIFO):");

foreach (var item in stack)

{

Console.Write(item + " ");

}

Console.WriteLine("\nСодержимое очереди (FIFO):");

foreach (var item in queue)

{

Console.Write(item + " ");

}

if (stack.Count > 0)

{

Console.WriteLine($"\n\nВерхний элемент стека: {stack.Peek()}");

Console.WriteLine($"Извлечён из стека: {stack.Pop()}");

}

if (queue.Count > 0)

{

Console.WriteLine($"\nПервый элемент очереди: {queue.Peek()}");

Console.WriteLine($"Извлечён из очереди: {queue.Dequeue()}");

}

Console.WriteLine("\nОставшееся содержимое стека:");

foreach (var item in stack)

{

Console.Write(item + " ");

}

Console.WriteLine("\nОставшееся содержимое очереди:");

foreach (var item in queue)

{

Console.Write(item + " ");

}

}

}

# Тестирование (задача 5)

Примеры выполнения программы представлены на рисунке 9.

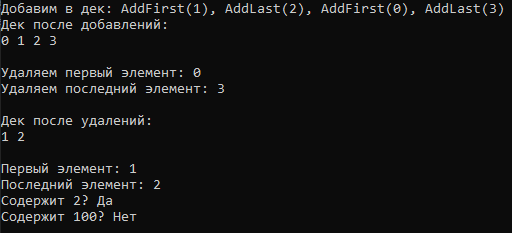


Рисунок 18 – Экранная форма программы

# Программа на C# (задача 5):

using System;

using System.Collections;

using System.Collections.Generic;

namespace Program

{

public class DoublyNode<T>

{

public T Data { get; set; }

public DoublyNode<T> Next { get; set; }

public DoublyNode<T> Previous { get; set; }

public DoublyNode(T data)

{

Data = data;

}

}

public class Deque<T> : IEnumerable<T>

{

private DoublyNode<T> head;

private DoublyNode<T> tail;

private int count;

public void AddFirst(T data)

{

DoublyNode<T> node = new DoublyNode<T>(data);

DoublyNode<T> temp = head;

node.Next = temp;

head = node;

if (count == 0)

tail = head;

else

temp.Previous = node;

count++;

}

public void AddLast(T data)

{

DoublyNode<T> node = new DoublyNode<T>(data);

if (head == null)

head = node;

else

{

tail.Next = node;

node.Previous = tail;

}

tail = node;

count++;

}

public T RemoveFirst()

{

if (count == 0)

throw new InvalidOperationException("Дек пуст");

T output = head.Data;

if (count == 1)

head = tail = null;

else

{

head = head.Next;

head.Previous = null;

}

count--;

return output;

}

public T RemoveLast()

{

if (count == 0)

throw new InvalidOperationException("Дек пуст");

T output = tail.Data;

if (count == 1)

head = tail = null;

else

{

tail = tail.Previous;

tail.Next = null;

}

count--;

return output;

}

public T First

{

get

{

if (IsEmpty)

throw new InvalidOperationException("Дек пуст");

return head.Data;

}

}

public T Last

{

get

{

if (IsEmpty)

throw new InvalidOperationException("Дек пуст");

return tail.Data;

}

}

public int Count => count;

public bool IsEmpty => count == 0;

public void Clear()

{

head = null;

tail = null;

count = 0;

}

public bool Contains(T data)

{

DoublyNode<T> current = head;

while (current != null)

{

if (current.Data.Equals(data))

return true;

current = current.Next;

}

return false;

}

public IEnumerator<T> GetEnumerator()

{

DoublyNode<T> current = head;

while (current != null)

{

yield return current.Data;

current = current.Next;

}

}

IEnumerator IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

}

class Program

{

static void Main()

{

var deque = new Deque<int>();

Console.WriteLine("Добавим в дек: AddFirst(1), AddLast(2), AddFirst(0), AddLast(3)");

deque.AddFirst(1);

deque.AddLast(2);

deque.AddFirst(0);

deque.AddLast(3);

Console.WriteLine("Дек после добавлений:");

PrintDeque(deque);

Console.WriteLine("\nУдаляем первый элемент: " + deque.RemoveFirst());

Console.WriteLine("Удаляем последний элемент: " + deque.RemoveLast());

Console.WriteLine("\nДек после удалений:");

PrintDeque(deque);

Console.WriteLine("\nПервый элемент: " + deque.First);

Console.WriteLine("Последний элемент: " + deque.Last);

Console.WriteLine("Содержит 2? " + (deque.Contains(2) ? "Да" : "Нет"));

Console.WriteLine("Содержит 100? " + (deque.Contains(100) ? "Да" : "Нет"));

}

static void PrintDeque<T>(Deque<T> deque)

{

foreach (var item in deque)

Console.Write(item + " ");

Console.WriteLine();

}

}

}

# Тестирование (задача 6)

Примеры выполнения программы представлены на рисунках 19, 20, 21.

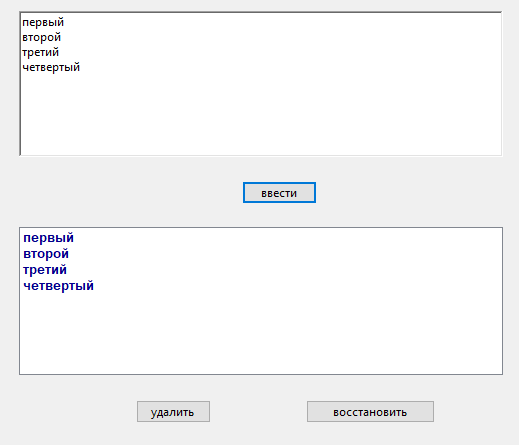


Рисунок 19 – Экранная форма программы

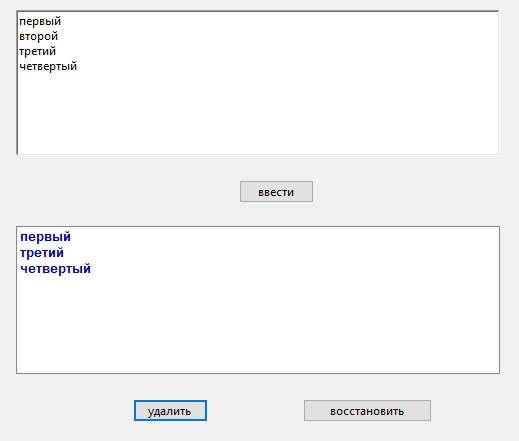


Рисунок 20 – Экранная форма программы

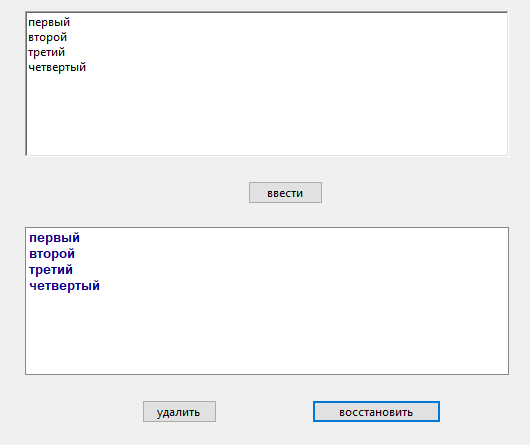


Рисунок 21 – Экранная форма программы

# Программа на C# (задача 6):

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace lab5\_6

{

public partial class Form1 : Form

{

private List<string> originalItems = new List<string>();

private List<string> currentItems = new List<string>();

public Form1()

{

InitializeComponent();

listBox1.ForeColor = System.Drawing.Color.DarkBlue;

listBox1.Font = new System.Drawing.Font("Arial", 10, System.Drawing.FontStyle.Bold);

}

private void button1\_Click(object sender, EventArgs e)

{

string[] lines = richTextBox1.Lines;

originalItems = new List<string>(lines);

currentItems = new List<string>(lines);

UpdateListBox();

}

private void button2\_Click(object sender, EventArgs e)

{

int selectedIndex = listBox1.SelectedIndex;

if (selectedIndex >= 0 && selectedIndex < currentItems.Count)

{

currentItems.RemoveAt(selectedIndex);

UpdateListBox();

}

}

private void button3\_Click(object sender, EventArgs e)

{

currentItems = new List<string>(originalItems);

UpdateListBox();

}

private void UpdateListBox()

{

listBox1.Items.Clear();

foreach (var item in currentItems)

{

listBox1.Items.Add(item);

}

}

}

}

# Тестирование (задача 7)

Примеры выполнения программы представлены на рисунках 22, 23, 24 и 25.

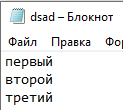


Рисунок 22 – Содержание исходного файла



Рисунок 23 – Экранная форма программы после загрузки файла

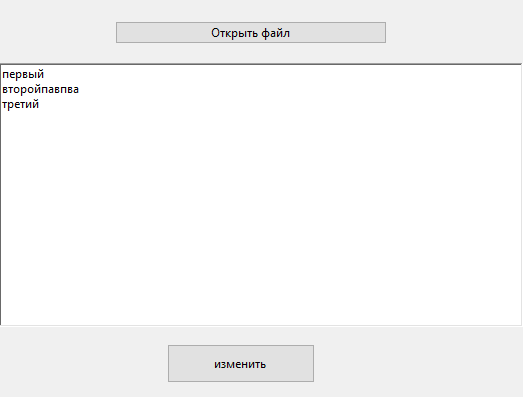


Рисунок 24 – Экранная форма программы после редактирования второго элемента

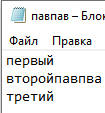


Рисунок 25 – Содержание выходного файла

# Программа на C# (задача 7):

using System;

using System.IO;

using System.Windows.Forms;

namespace lab5\_7

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(richTextBox1.Text))

{

MessageBox.Show("Нет данных для сохранения.");

return;

}

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Title = "Сохранить измененный файл";

saveFileDialog.Filter = "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

File.WriteAllText(saveFileDialog.FileName, richTextBox1.Text);

MessageBox.Show("Файл успешно сохранён!");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при сохранении файла:\n" + ex.Message);

}

}

}

}

private void OpenFile()

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.Title = "Открыть текстовый файл";

openFileDialog.Filter = "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

string[] lines = File.ReadAllLines(openFileDialog.FileName);

richTextBox1.Lines = lines;

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при чтении файла:\n" + ex.Message);

}

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

this.OpenFile();

}

}

}

# Тестирование (задача 8)

Примеры выполнения программы представлены на рисунках 26, 27, 28.



Рисунок 26 – Экранная форма программы, стартовый экран

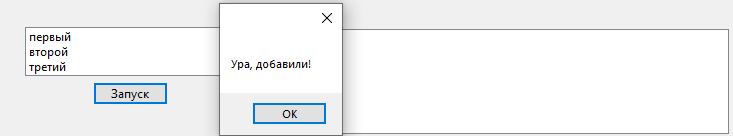


Рисунок 27 – Экранная форма программы, нажали кнопку запуска

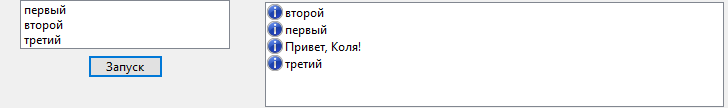


Рисунок 28 – Экранная форма программы, результат работы

# Программа на C# (задача 8):

using System;

using System.Collections.Generic;

using System.IO;

using System.Windows.Forms;

using System.Drawing;

namespace lab5\_8

{

public partial class Form1 : Form

{

private string filePath = Path.Combine(Application.StartupPath, "temp.txt");

public Form1()

{

InitializeComponent();

listView1.View = View.List;

ImageList imageList = new ImageList();

imageList.ImageSize = new Size(16, 16);

imageList.Images.Add(SystemIcons.Information);

listView1.SmallImageList = imageList;

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

List<string> lines = new List<string>();

foreach (var item in listBox1.Items)

{

lines.Add(item.ToString());

}

using (StreamWriter writer = new StreamWriter(filePath))

{

if (lines.Count > 0)

{

writer.WriteLine(lines[0]);

writer.WriteLine("Привет!");

for (int i = 1; i < lines.Count; i++)

{

writer.WriteLine(lines[i]);

}

}

}

List<string> fileLines = new List<string>(File.ReadAllLines(filePath));

fileLines.Sort();

int index = fileLines.BinarySearch("Привет!");

if (index >= 0)

{

fileLines[index] = "Привет, Коля!";

MessageBox.Show("Ура, добавили!");

}

listView1.Items.Clear();

foreach (string line in fileLines)

{

ListViewItem lvi = new ListViewItem(line, 0);

listView1.Items.Add(lvi);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка: " + ex.Message);

}

}

}

}

# Вывод

В ходе выполнения этих задач научились работать с основными структурами данных, реализовывать и использовать двусвязные списки и деки, эффективно считывать и записывать данные из файлов и в файлы, управлять отображением и редактированием данных в элементах управления интерфейса, а также применять алгоритмы сортировки и поиска для обработки коллекций.