Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Математическое обеспечение и применение ЭВМ»

ОТЧЕТ

по лабораторной работе №4

по курсу «Объектно-ориентированное программирование» на тему

«LINQ»

Выполнили: студенты группы 21ВП2

Лакеева Софья

Малькова Анастасия

Приняли: к.т.н. Афонин А.Ю.

к.т.н. Иванчуков А.Г.

Пенза 2023

**Название**

LINQ.

**Цель работы**

Научиться использовать делегаты и события.

**Лабораторное задание**

В лабораторной работе требуется создать обобщенный класс CollectionType. Определить в классе конструкторы, деструктор, методы добавления и удаления элементов, другие необходимые методы и, если требуется, перегруженные операции. Определить индексаторы и свойства. CollectionType можно реализовать на основе стандартных коллекций (List, Stack, Array и т.д.).

Возьмите созданный тип (класс) из предыдущих лабораторных работ и реализуйте в нем интерфейс IComparable. Используйте данный класс в качестве параметра вашего обобщенного класса. Создайте несколько коллекций. Выполните сохранение в файл, сортировку, LINQ-запросы в соответствии с вариантом. Выполните несколько сложных LINQToObject запросов (минимум 5) к коллекции объектов, используя одновременно более трех операций (пример: where + select + orderBy, first + any + min).

Создайте обобщённую стандартную коллекцию из пространства имен System.Collections указанную в варианте со строками и выполните вводвывод, сохранение в файл, поиск строк, содержащих определенное значение, подсчет количества строк длины n, сортировку в возрастающем и убывающем порядке.

Вариант:

1, создать массив объектов CollectionType, запросы – найти коллекции размера n; найти максимальную и минимальную коллекцию в массиве по количеству элементов. Обобщенная коллекция – LinkedList.

**Описание метода решения задачи**

Лабораторная работа была выполнена с использованием LINQ, методов строк, а также встроенных классов для записи данных в файл.

**Листинг (текст программы)**

Файл «CollectionType.cs»

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinformsUI

{

public class CollectionType<T> : IEnumerable

{

/// <summary>

/// Список обобщенной коллекции

/// </summary>

private List<T> \_list;

/// <summary>

/// Размер коллекции

/// </summary>

public int Count { get { return \_list.Count; } }

/// <summary>

/// Конструктор по умолчанию

/// </summary>

public CollectionType()

{

\_list = new List<T>();

}

/// <summary>

/// Метод добавления объекта в коллекцию

/// </summary>

/// <param name="item">Объект для добавления</param>

public void Add(T item)

{

\_list.Add(item);

}

/// <summary>

/// Метод удаления объекта из коллекции

/// </summary>

/// <param name="index">Индекс объекта в коллекции</param>

public void RemoveAt(int index)

{

\_list.RemoveAt(index);

}

/// <summary>

/// Копирует значения другой коллекции, удаляя ранее имеющиеся

/// </summary>

/// <param name="other">Коллекция из которой нужно скопировать</param>

public void Copy(CollectionType<T> other)

{

\_list.Clear();

for (int i = 0; i < other.\_list.Count; i++)

{

\_list.Add(other.\_list[i]);

}

}

/// <summary>

///

/// </summary>

public void Sort()

{

\_list.Sort();

}

public IEnumerator GetEnumerator()

{

return \_list.GetEnumerator();

}

/// <summary>

/// Метод получения списка из объекта

/// </summary>

/// <returns>Лист с объектами внутри</returns>

public List<T> GetList()

{

return \_list;

}

/// <summary>

/// Индексатор

/// </summary>

/// <param name="index">Индекс объекта в коллекции</param>

/// <returns>Объект по указанному индексу</returns>

public T this[int index]

{

get => \_list[index];

set => \_list[index] = value;

}

}

}

Файл «LINQWindow.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics.Metrics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WinformsUI

{

public partial class LINQWindow : Form

{

private CollectionType<Library> \_libraries = new CollectionType<Library>();

private LinkedList<string>[] \_lists;

public LINQWindow()

{

InitializeComponent();

\_libraries.Add(new Library("Британская библиотека", "not cool library 1", 100, 10, "public", false, 2.0m));

\_libraries.Add(new Library("Российская национальная библиотека", "not cool library 2", 200, 20, "public", false, 2.5m));

\_libraries.Add(new Library("Национальная библиотека Испании", "not cool library 3", 300, 30, "public", false, 3.0m));

\_libraries.Add(new Library("Александрийская библиотека", "not cool library 4", 400, 40, "public", false, 3.5m));

\_libraries.Add(new Library("Баварская государственная библиотека", "not cool library 5", 500, 50, "public", false, 4.0m));

\_libraries.Add(new Library("Никольская библиотека", "cool library 1", 600, 60, "public", true, 5.0m));

\_libraries.Add(new Library("Зареченская библиотека", "cool library 2", 700, 70, "public", true, 5.0m));

\_libraries.Add(new Library("Ртищевская библиотека", "cool library 3", 800, 80, "public", true, 5.0m));

}

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

{

Close();

}

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

{

consoleLog.Clear();

}

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

{

foreach (var library in \_libraries)

{

Utils.FillOutputLog(library.ToString(), consoleLog);

}

Utils.FillOutputLog("=========================", consoleLog);

}

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

{

CollectionType<Library> libs = new CollectionType<Library>();

libs.Copy(\_libraries);

libs.Sort();

Utils.FillOutputLog("До сортировки: ", consoleLog);

foreach (var library in \_libraries)

{

Utils.FillOutputLog(library.ToString(), consoleLog);

}

Utils.FillOutputLog("После сортировки: ", consoleLog);

foreach (var library in libs)

{

Utils.FillOutputLog(library.ToString(), consoleLog);

}

Utils.FillOutputLog("=========================", consoleLog);

}

private async void writeToFileLibCollectionButton\_Click(object sender, EventArgs e)

{

StringBuilder stringBuilder = new StringBuilder();

foreach (var library in \_libraries)

{

stringBuilder.AppendLine(library.ToString());

}

using (StreamWriter writer = new StreamWriter("libs.txt", false))

{

await writer.WriteLineAsync(stringBuilder.ToString());

}

}

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

{

Utils.FillOutputLog("LINQ Libraries", consoleLog);

Library lib = \_libraries.GetList().Where(library => library.BooksNumber > 400).Reverse().First();

Utils.FillOutputLog("Where(library => library.BooksNumber > 400).Reverse().First()", consoleLog);

Utils.FillOutputLog(lib.ToString(), consoleLog);

Utils.FillOutputLog("============================================", consoleLog);

bool result = \_libraries.GetList().Skip(2).Take(5).All(library => library.ReadingRoomsCount > 0);

Utils.FillOutputLog("Skip(2).Take(5).All(library => library.ReadingRoomsCount > 0)", consoleLog);

Utils.FillOutputLog(result.ToString(), consoleLog);

Utils.FillOutputLog("============================================", consoleLog);

int count = \_libraries.GetList().SkipWhile(library => library.Rating < 3.0m).TakeWhile(library => !library.WithWiFi).Count();

Utils.FillOutputLog("SkipWhile(library => library.Rating < 3.0m).TakeWhile(library => !library.WithWiFi).Count()", consoleLog);

Utils.FillOutputLog(count.ToString(), consoleLog);

Utils.FillOutputLog("============================================", consoleLog);

int sum = \_libraries.GetList().OrderBy(library => library.Name).Take(3).Sum(library => library.ReadingRoomsCount);

Utils.FillOutputLog("OrderBy(library => library.Name).Take(3).Sum(library => library.ReadingRoomsCount)", consoleLog);

Utils.FillOutputLog(sum.ToString(), consoleLog);

Utils.FillOutputLog("============================================", consoleLog);

bool result2 = \_libraries.GetList().OrderByDescending(library => library.BooksNumber).Skip(4).Any(library => library.Rating > 3.0m);

Utils.FillOutputLog("OrderByDescending(library => library.BooksNumber).Skip(4).Any(library => library.Rating > 3.0m)", consoleLog);

Utils.FillOutputLog(result2.ToString(), consoleLog);

Utils.FillOutputLog("============================================", consoleLog);

}

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

{

Random random = new Random();

LinkedList<string>[] list = new LinkedList<string>[random.Next(1, (int)arrayLengthTextBox.Value)];

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

{

list[i] = new LinkedList<string>();

}

for (int k = 0; k < list.Length; k++)

{

int collectionLength = random.Next(1, (int)collectionLengthTextBox.Value);

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

{

int length = random.Next((int)fromLengthTextBox.Value, (int)toLengthTextBox.Value);

StringBuilder stringBuilder = new StringBuilder();

for (int j = 0; j < length; j++)

{

stringBuilder.Append(Convert.ToChar(Convert.ToInt32(Math.Floor(25 \* random.NextDouble())) + 65));

}

list[k].AddFirst(stringBuilder.ToString());

}

}

\_lists = list;

}

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

{

if (\_lists.Any(element => element.Count == 0))

{

Utils.MessageBox(IntPtr.Zero, "Массив не был создан!", "Ошибка", 0);

return;

}

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

{

for (int j = 0; j < \_lists[i].Count; j++)

{

Utils.FillOutputLog($"array line: {i}, linkedList element: {j}, value: {\_lists[i].ElementAt(j)}", consoleLog);

}

}

}

private async void executeArrayCollectionOperationsButton\_Click(object sender, EventArgs e)

{

if (\_lists.Any(element => element.Count == 0))

{

Utils.MessageBox(IntPtr.Zero, "Массив не был создан!", "Ошибка", 0);

return;

}

if (searchValueTextBox.Text == "")

{

Utils.MessageBox(IntPtr.Zero, "Введите значение для поиска!", "Ошибка", 0);

return;

}

consoleLog.Clear();

Utils.FillOutputLog("1) СОХРАНЕНИЕ В ФАЙЛ", consoleLog);

StringBuilder stringBuilder = new StringBuilder();

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

{

for (int j = 0; j < \_lists[i].Count; j++)

{

stringBuilder.Append($"array line: {i}, linkedList element: {j}, value: {\_lists[i].ElementAt(j)}");

}

}

using (StreamWriter writer = new StreamWriter("strings.txt", false))

{

await writer.WriteLineAsync(stringBuilder.ToString());

}

Utils.FillOutputLog("Сохранение в файл произошло успешно", consoleLog);

Utils.FillOutputLog("2) ПОИСК СТРОК ПО ЗНАЧЕНИЮ", consoleLog);

Utils.FillOutputLog($"По значению {searchValueTextBox.Text} найдено:", consoleLog);

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

{

for (int j = 0; j < \_lists[i].Count; j++)

{

if (\_lists[i].ElementAt(j).Contains(searchValueTextBox.Text))

{

Utils.FillOutputLog($"array line: {i}, linkedList element: {j}, value: {\_lists[i].ElementAt(j)}", consoleLog);

}

}

}

Utils.FillOutputLog("3) ПОИСК СТРОК ПО ДЛИНЕ", consoleLog);

int length = (int)stringLengthTextBox.Value;

int count = 0;

foreach (var item in \_lists)

{

count += item.Where(str => str.Length == length).Count();

}

Utils.FillOutputLog($"По длине {stringLengthTextBox.Value} найдено: {count} строк", consoleLog);

int collectionsLength = \_lists.Count(list => list.Count == (int)collectionsLengthNTextBox.Value);

Utils.FillOutputLog($"Количество коллекций длины n: {collectionsLength}", consoleLog);

int max = \_lists.Max(list => list.Count);

Utils.FillOutputLog($"Максимальная длина коллекции: {max}", consoleLog);

int min = \_lists.Min(list => list.Count);

Utils.FillOutputLog($"Минимальная длина коллекции: {min}", consoleLog);

Utils.FillOutputLog("4) СОРТИРОВКА В ВОЗРАСТАЮЩЕМ ПОРЯДКЕ", consoleLog);

foreach (var item in \_lists)

{

item.OrderBy(str => str);

}

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

{

for (int j = 0; j < \_lists[i].Count; j++)

{

Utils.FillOutputLog($"array line: {i}, linkedList element: {j}, value: {\_lists[i].ElementAt(j)}", consoleLog);

}

}

Utils.FillOutputLog("4) СОРТИРОВКА В УБЫВАЮЩЕМ ПОРЯДКЕ", consoleLog);

foreach (var item in \_lists)

{

item.OrderByDescending(str => str);

}

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

{

for (int j = 0; j < \_lists[i].Count; j++)

{

Utils.FillOutputLog($"array line: {i}, linkedList element: {j}, value: {\_lists[i].ElementAt(j)}", consoleLog);

}

}

Utils.FillOutputLog("======================================", consoleLog);

}

}

}

**Пояснительный текст к программе**

Класс CollectionType является обобщенным классом-коллекцией, в основе которой лежит коллекция List. Новая форма с названием «LINQWindow» предоставляет интерфейс для работы с коллекцией объектов класса Library и массива коллекций строк LinkedList.

**Результаты работы программы**

Результаты работы программы приведены на рисунке 1

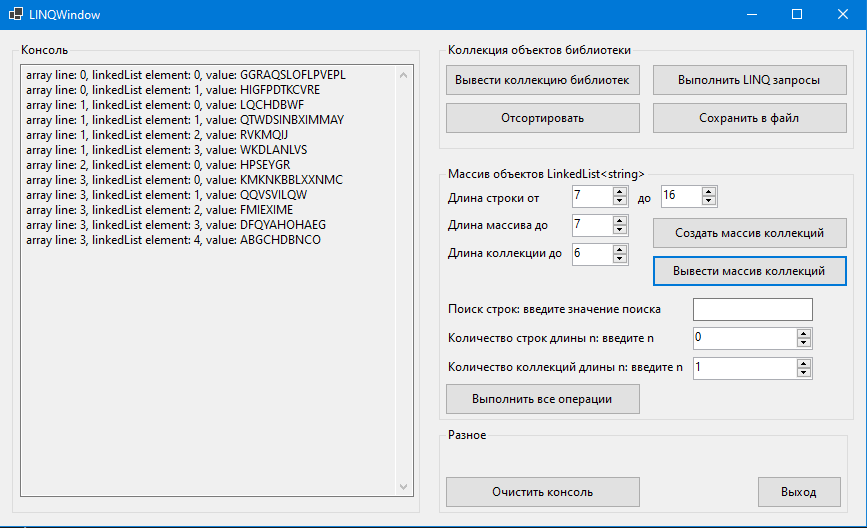


Рисунок 1 – результат работы программы

**Выводы**

Были освоены навыки работы с LINQ. Также освоены навыки работы с файлами и строками.