ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА (САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт Информатики и кибернетики
Кафедра Программных систем

ОТЧЁТ

по лабораторной работе

№6 «Язык программирования C#: Делегаты»

по дисциплине «Языки программирования и структуры данных»

Выполнил Фадеев А.М. 6101

Проверил Котенёва С.Э.

Самара

2024

ЗАДАНИЕ

Задание 0.

Прочитать теоретический материал.

Задание 1.

Организовать работу меню в программе из лабораторной работы 5 с помощью механизма делегатов:

– описать делегат без параметров, возвращающий void;

– в классе Program все пункты меню реализовать как статические методы, сигнатура методов совпадает с сигнатурой, заданной делегатом;

– запросить у пользователя траекторию выполнения программы – набор пунктов меню и добавить в делегат, в зависимости от набора пунктов меню, введенного пользователем, список статических методов, выполняющих соответствующие пункты меню;

– в методе Main() вызывать делегат на исполнение.

Задание 2.

Протестировать работу приложения в классе Program, разработать адекватный интерфейс пользователя. Необходимо отлавливать и обрабатывать все возможные исключения.

Воспользоваться пользовательским интерфейсом из лабораторной работы 5.

Задание 3.

Подготовить отчет о работе.

КОД ПРОГРАММЫ

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace JIP06
{
    interface IVectorable
    {
        int this[int index] { get; set; }
        int Length { get; }
        double GetNorm();
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace JIP06
{
    class ArrayVector : IVectorable, IComparable, ICloneable
    {
        private int[] vector;

        public ArrayVector()
        {
            int len = 5;
            vector = new int[len];
        }
        public ArrayVector(int len)
        {
            if (len > 0)
            {
                vector = new int[len];
            }
            else
            {
                throw new Exception("Invalid length");
            }
        }

        public int this[int i]
        {
            get
            {
                if (0 <= i && i < Length) return vector[i];
                else throw new IndexOutOfRangeException("Vector index out of
range");
            }
            set
            {
                if (0 <= i && i < Length) vector[i] = value;
                else throw new IndexOutOfRangeException("Vector index out of
range");
            }
        }

        public int Length
        {
            get { return vector.Length; }
        }
```

```
        public override string ToString()
        {
            string lenStr = Length.ToString();
            string vectorStr = string.Join(" ", Array.ConvertAll<int,
string>(vector, x => x.ToString()));
            return string.Format("{0, -13}  {1}", lenStr, vectorStr);
        }

        public override bool Equals(object obj)
        {
            if (obj.GetType() is IVectorable || Length !=
((IVectorable)obj).Length)
            {
                return false;
            }
            else
            {
                for (int i = 0; i < Length; i++)
                {
                    if (vector[i] != ((IVectorable)obj)[i]) return false;
                }
            }
            return true;
        }

        public override int GetHashCode()
        {
            int hash = 0;
            for (int i = 0; i < Length; i++)
            {
                hash += vector[i];
            }
            hash *= Length;
            return hash;
        }

        public int CompareTo(object obj)
        {
            if (obj.GetType() is IVectorable)
            {
                throw new Exception("Object is not IVectorable");
            }
            else
            {
                if (Length < ((IVectorable)obj).Length)
                {
                    return -1;
                }
                else if (Length > ((IVectorable)obj).Length)
                {
                    return 1;
                }
                return 0;
            }
        }

        public Object Clone()
        {
            ArrayVector clone = new ArrayVector(Length);
            for (int i = 0; i < Length; i++)
            {
                clone[i] = vector[i];
            }
            return clone;
        }
```

```
        public double GetNorm()
        {
            double norm = 0;
            for (int i = 0; i < vector.Length; i++)
            {
                norm += Math.Pow(vector[i], 2);
            }
            return Math.Sqrt(norm);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace JIP06
{
    class LinkedListVector : IVectorable, IComparable, ICloneable
    {
        private Node start;

        public class Node
        {
            public int value = 0;
            public Node link = null;
        }

        public LinkedListVector()
        {
            int len = 5;
            start = new Node();
            Node curNode = start;
            for (int i = 0; i < len - 1; i++)
            {
                curNode.link = new Node();
                curNode = curNode.link;
            }
        }

        public LinkedListVector(int len)
        {
            if (len > 0)
            {
                start = new Node();
                Node curNode = start;
                for (int i = 0; i < len - 1; i++)
                {
                    curNode.link = new Node();
                    curNode = curNode.link;
                }
            }
            else
            {
                throw new Exception("Invalid length");
            }
        }

        public int this[int index]
        {
            get
            {
                if (0 <= index && index < Length)
                {
                    Node curNode = start;
```

```
                    for (int i = 0; i < index; i++)
                    {
                        curNode = curNode.link;
                    }
                    return curNode.value;
                }
                else
                {
                    throw new IndexOutOfRangeException("Vector index out of
range");
                }
            }
            set
            {
                if (0 <= index && index < Length)
                {
                    Node curNode = start;
                    for (int i = 0; i < index; i++)
                    {
                        curNode = curNode.link;
                    }
                    curNode.value = value;
                }
                else
                {
                    throw new IndexOutOfRangeException("Vector index out of
range");
                }
            }
        }

        public int Length
        {
            get
            {
                int length = 0;
                if (start == null)
                {
                    return length;
                }
                else
                {
                    Node curNode = start;
                    length++;
                    while (curNode.link != null)
                    {
                        curNode = curNode.link;
                        length++;
                    }
                    return length;
                }
            }
        }

        public override string ToString()
        {
            string lenStr = Length.ToString();
            string vectorStr;
            Node curNode = start;
            vectorStr = curNode.value.ToString();
            while (curNode.link != null)
            {
                curNode = curNode.link;
                vectorStr += " " + curNode.value.ToString();
            }
            return string.Format("{0, -13}  {1}", lenStr, vectorStr);
```

```
        }

        public override bool Equals(object obj)
        {
            if (obj.GetType() is IVectorable || Length !=
((IVectorable)obj).Length)
            {
                return false;
            }
            Node cureNode = start;
            for (int i = 0; i < Length; i++)
            {
                if (cureNode.value != ((IVectorable)obj)[i]) return false;
                cureNode = cureNode.link;
            }
            return true;
        }

        public override int GetHashCode()
        {
            int hash = 0;
            Node cureNode = start;
            for (int i = 0; i < Length; i++)
            {
                hash += cureNode.value;
                cureNode = cureNode.link;
            }
            hash *= Length;
            return hash;
        }

        public int CompareTo(object obj)
        {
            if (obj.GetType() is IVectorable)
            {
                throw new Exception("Object is not IVectorable");
            }
            else
            {
                if (Length < ((IVectorable)obj).Length)
                {
                    return -1;
                }
                else if (Length > ((IVectorable)obj).Length)
                {
                    return 1;
                }
                return 0;
            }
        }

        public Object Clone()
        {
            LinkedListVector clone = new LinkedListVector(Length);
            Node curNode = start;
            for (int i = 0; i < Length; i++)
            {
                clone[i] = curNode.value;
                curNode = curNode.link;
            }
            return clone;
        }

        public double GetNorm()
        {
            double norm = 0;
```

```
            Node curNode = start;
            while (curNode.link != null)
            {
                norm += Math.Pow(curNode.value, 2);
                curNode = curNode.link;
            }
            norm += Math.Pow(curNode.value, 2);
            return Math.Sqrt(norm);
        }

        public void AddTop(int newElemValue)
        {
            Node temp = new Node();
            temp.value = newElemValue;
            temp.link = start;
            start = temp;
        }

        public void AddEnd(int newElemValue)
        {
            Node curNode = start;
            for (int i = 0; i < Length - 1; i++)
            {
                curNode = curNode.link;
            }
            curNode.link = new Node();
            curNode.link.value = newElemValue;
        }

        public void AddByIndex(int index, int newElemValue)
        {
            if (index == 0)
            {
                AddTop(newElemValue);
            }
            else if (index == Length)
            {
                AddEnd(newElemValue);
            }
            else if (0 < index && index < Length)
            {
                Node curNode = start;
                for (int i = 0; i < index - 1; i++)
                {
                    curNode = curNode.link;
                }
                Node temp = new Node();
                temp.value = newElemValue;
                temp.link = curNode.link;
                curNode.link = temp;
            }
            else
            {
                throw new IndexOutOfRangeException("Vector index out of
range");
            }
        }

        public void DeleteTop()
        {
            start = start.link;
        }

        public void DeleteEnd()
        {
            if (Length == 1)
```

```
            {
                start = null;
            }
            else
            {
                Node curNode = start;
                for (int i = 0; i < Length - 2; i++)
                {
                    curNode = curNode.link;
                }
                curNode.link = null;
            }
        }

        public void DeleteByIndex(int index)
        {
            if (index == 0)
            {
                DeleteTop();
            }
            else if (index == Length - 1)
            {
                DeleteEnd();
            }
            else if (0 < index && index < Length - 1)
            {
                Node curNode = start;
                for (int i = 0; i < index - 1; i++)
                {
                    curNode = curNode.link;
                }
                curNode.link = curNode.link.link;
            }
            else
            {
                throw new IndexOutOfRangeException("Vector index out of
range");
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace JP06
{
    class VectorsComparer : IComparer<IVectorable>
    {
        public int Compare(IVectorable vector1, IVectorable vector2)
        {
            if (vector1.GetNorm() < vector2.GetNorm())
            {
                return -1;
            }
            else if (vector1.GetNorm() > vector2.GetNorm())
            {
                return 1;
            }
            return 0;
        }
    }
}
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.IO;

namespace JIP06
{
    class Vectors
    {
        public static ArrayVector Sum(IVectorable vector1, IVectorable
vector2)
        {
            if (vector1.Length == vector2.Length)
            {
                int len = vector1.Length;
                ArrayVector res = new ArrayVector(len);
                for (int i = 0; i < len; i++)
                {
                    res[i] = vector1[i] + vector2[i];
                }
                return res;
            }
            else
            {
                throw new Exception("Different length");
            }
        }
        public static int Scalar(IVectorable vector1, IVectorable vector2)
        {
            int scalar = 0;
            int len = 0;
            if (vector1.Length == vector2.Length)
            {
                len = vector1.Length;
                for (int i = 0; i < len; i++)
                {
                    scalar += vector1[i] * vector2[i];
                }
                return scalar;
            }
            else
            {
                throw new Exception("Different length");
            }
        }
        public static double GetNormSt(IVectorable vector)
        {
            double norm = 0;
            for (int i = 0; i < vector.Length; i++)
            {
                norm += Math.Pow(vector[i], 2);
            }
            return Math.Sqrt(norm);
        }
        public static void OutputVector(IVectorable vector, Stream output)
        {
            byte[] buffer = BitConverter.GetBytes(vector.Length);
            output.Write(buffer, 0, buffer.Length);
            for (int i = 0; i < vector.Length; i++)
            {
                buffer = BitConverter.GetBytes(vector[i]);
                output.Write(buffer, 0, buffer.Length);
            }
        }
        public static IVectorable InputVector(Stream input)
        {
            byte[] buffer = new byte[4];
```

```
            input.Read(buffer, 0, buffer.Length);
            int len = BitConverter.ToInt32(buffer, 0);
            ArrayVector vector = new ArrayVector(len);

            for (int i = 0; i < len; i++)
            {
                input.Read(buffer, 0, buffer.Length);
                vector[i] = BitConverter.ToInt32(buffer, 0);
            }
            return vector;
        }
        public static void WriteVector(IVectorable vector, TextWriter output)
        {
            output.WriteLine(vector.ToString());
        }
        public static IVectorable ReadVector(TextReader input)
        {
            ArrayVector vector;
            string[] nums = input.ReadLine().Split(' ');
            int num;
            int counter = 0;
            int i = 0;
            while (!Int32.TryParse(nums[i], out num)) i++;
            vector = new ArrayVector(Int32.Parse(nums[i]));

            for (i++ ; i < nums.Length; i++)
            {
                if (Int32.TryParse(nums[i], out num) && counter <
vector.Length)
                {
                    vector[counter] = Int32.Parse(nums[i]);
                    counter++;
                }
            }
            return vector;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;

namespace ЛР06
{
    delegate void ExecutionPath(List<IVectorable> vectors);
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Лабораторная работа №6");
            Console.WriteLine("Выполнил студент Дьячков Дмитрий 6101-
020302D\n");

            List<IVectorable> vectors = new List<IVectorable>();
            //bool flag = true;
            //while (flag)
            //{
            //    flag = MainMenu(vectors);
            //}

            Console.WriteLine("Выберите пункт меню: ");
            Console.WriteLine("1 - Создать вектор ArrayVecor");
            Console.WriteLine("2 - Создать вектор LinkedList");
            Console.WriteLine("3 - Клонировать вектор");
            Console.WriteLine("4 - Редактировать вектор");
```

```
            Console.WriteLine("5 - Высчитать модуль вектора");
            Console.WriteLine("6 - Проверить вектора на равенство");
            Console.WriteLine("7 - Сумма двух векторов");
            Console.WriteLine("8 - Скалярное произведение двух векторов");
            Console.WriteLine("9 - Вывести вектора с максимальным и
минимальным количеством координат");
            Console.WriteLine("10 - Отсортировать вектора по количеству
координат");
            Console.WriteLine("11 - Отсортировать вектора по модулю");
            Console.WriteLine("12 - Удалить вектор");

            Console.WriteLine("Введите пункты меню через пробел:");
            string sTrajectory = Console.ReadLine();
            Console.Clear();
            string[] trajectory = sTrajectory.Split(' ');

            // создание делегата
            ExecutionPath execution = null;
            for (int i = 0; i < trajectory.Length; i++)
            {
                if (trajectory[i] == "1")
                {
                    execution += VectorsInfo;
                    execution += CreateArrayVector;
                    execution += ConsoleClear;
                }
                else if (trajectory[i] == "2")
                {
                    execution += VectorsInfo;
                    execution += CreateLinkedListVector;
                    execution += ConsoleClear;
                }
                else if (trajectory[i] == "3")
                {
                    execution += VectorsInfo;
                    execution += CloneVector;
                    execution += ConsoleClear;
                }
                else if (trajectory[i] == "4")
                {
                    execution += VectorsInfo;
                    execution += EditVector;
                    execution += ConsoleClear;
                }
                else if (trajectory[i] == "5")
                {
                    execution += VectorsInfo;
                    execution += VectorNorm;
                    execution += ConsoleClear;
                }
                else if (trajectory[i] == "6")
                {
                    execution += VectorsInfo;
                    execution += IsEquals;
                    execution += ConsoleClear;
                }
                else if (trajectory[i] == "7")
                {
                    execution += VectorsInfo;
                    execution += Sum;
                    execution += ConsoleClear;
                }
                else if (trajectory[i] == "8")
                {
                    execution += VectorsInfo;
                    execution += Mult;
```

```
                    execution += ConsoleClear;
                }
                else if (trajectory[i] == "9")
                {
                    execution += VectorsInfo;
                    execution += MinMaxLength;
                    execution += ConsoleClear;
                }
                else if (trajectory[i] == "10")
                {
                    execution += VectorsInfo;
                    execution += SortByLength;
                    execution += ConsoleClear;
                }
                else if (trajectory[i] == "11")
                {
                    execution += VectorsInfo;
                    execution += SortByNorm;
                    execution += ConsoleClear;

                }
                else if (trajectory[i] == "12")
                {
                    execution += VectorsInfo;
                    execution += Delete;
                    execution += ConsoleClear;
                }
            }
            execution += VectorsInfo;

            execution(vectors);
            Console.WriteLine("Программа завершена");

            Console.ReadKey();
        }


        //МЕНЮ:


        static bool MainMenu(List<IVectorable> vectors)
        {
            string choice;
            if (vectors.Count > 0)
            {
                Console.WriteLine("Список векторов:");
                VectorsInfo(vectors);
                Console.WriteLine();
            }
            Console.WriteLine("Выберите пункт меню: ");
            Console.WriteLine("(1) - Добавление, редактирование и удаление
векторов");
            Console.WriteLine("(2) - Взаимодействие с файловой системой");
            Console.WriteLine("(3) - Сортировка векторов");
            Console.WriteLine("4 - Высчитать модуль вектора");
            Console.WriteLine("5 - Проверить вектора на равенство");
            Console.WriteLine("6 - Скалярное произведение двух векторов");
            Console.WriteLine("7 - Вывести вектора с максимальным и
минимальным количеством координат");
            Console.WriteLine("Enter - Завершить программу");

            Console.Write("Выбор: ");
            choice = Console.ReadLine();
            Console.WriteLine();

            bool flag;
```

```
    switch (choice)
    {
        case "1":
            flag = true;
            while (flag)
            {
                flag = MoodificationMenu(vectors);
            }
            break;
        case "2":
            flag = true;
            while (flag)
            {
                flag = FileMenu(vectors);
            }
            break;
        case "3":
            flag = true;
            while (flag)
            {
                flag = SortMenu(vectors);
            }
            break;
        case "4":
            VectorNorm(vectors);
            break;
        case "5":
            IsEquals(vectors);
            break;
        case "6":
            Mult(vectors);
            break;
        case "7":
            MinMaxLength(vectors);
            break;
        default:
            return false;
    }
    Console.Clear();
    return true;
}

static bool MoodificationMenu(List<IVectorable> vectors)
{
    Console.Clear();

    string choice;
    if (vectors.Count > 0)
    {
        Console.WriteLine("Список векторов:");
        VectorsInfo(vectors);
        Console.WriteLine();
    }
    Console.WriteLine("Выберите пункт меню: ");
    Console.WriteLine("1 - Создать ветор ArrayVector");
    Console.WriteLine("2 - Создать вектор LinkedList");
    Console.WriteLine("3 - Добавить клон вектора");
    Console.WriteLine("4 - Добавить сумму векторов");
    Console.WriteLine("5 - Редактировать вектор");
    Console.WriteLine("6 - Удалить вектор");
    Console.WriteLine("Enter - выход в главное меню");

    Console.Write("Выбор: ");
    choice = Console.ReadLine();
    Console.WriteLine();
```

```
            switch (choice)
            {
                case "1":
                    CreateArrayVector(vectors);
                    break;
                case "2":
                    CreateLinkedListVector(vectors);
                    break;
                case "3":
                    CloneVector(vectors);
                    break;
                case "4":
                    Sum(vectors);
                    break;
                case "5":

                    break;
                case "6":
                    Delete(vectors);
                    break;
                default:
                    return false;
            }
            Console.Clear();
            return true;
        }

        static bool FileMenu(List<IVectorable> vectors)
        {
            Console.Clear();

            string choice;
            if (vectors.Count > 0)
            {
                Console.WriteLine("Список векторов:");
                VectorsInfo(vectors);
                Console.WriteLine();
            }
            FileList();
            Console.WriteLine();

            Console.WriteLine("Выберите пункт меню: ");
            Console.WriteLine("1 - Создать файл данных");
            Console.WriteLine("2 - Создать текстовый файл");
            Console.WriteLine("3 - Добавить в файл вектора");
            Console.WriteLine("4 - Извлечь вектора из файла");
            Console.WriteLine("5 - Удалить файл");
            Console.WriteLine("Enter - выход в главное меню");

            Console.Write("Выбор: ");
            choice = Console.ReadLine();
            Console.WriteLine();

            string pathB = "//mscs-fs/data/314_user_data/!  C# 2023-2024/2
семестр - C#. Отчеты/6101/Дьячков Д/ЛР05/ЛР05/ЛР05/FileStream/{0}.bin";
            string pathT = "//mscs-fs/data/314_user_data/!  C# 2023-2024/2
семестр - C#. Отчеты/6101/Дьячков Д/ЛР05/ЛР05/ЛР05/FileStream/{0}.txt";
            string path = "//mscs-fs/data/314_user_data/!  C# 2023-2024/2
семестр - C#. Отчеты/6101/Дьячков Д/ЛР05/ЛР05/ЛР05/FileStream";
            string fileName;
            string fullPath;
            switch (choice)
            {
                case "1":
                    Console.Write("Введите имя файла: ");
                    fileName = Console.ReadLine();
```

```
                    fullPath = String.Format(pathB, fileName);

                    try
                    {
                        FileStream cFileStream = new FileStream(fullPath,
FileMode.Create);
                        cFileStream.Close();
                        Console.WriteLine("Файл создан");
                        Console.ReadKey();
                    }
                    catch (Exception)
                    {
                        Console.WriteLine("Имя содержит недопустимые знаки");
                        Console.ReadKey();
                    }
                    break;
                case "2":
                    Console.Write("Введите имя файла: ");
                    fileName = Console.ReadLine();
                    fullPath = String.Format(pathT, fileName);

                    try
                    {
                        FileStream cFileStream = new FileStream(fullPath,
FileMode.Create);
                        cFileStream.Close();
                        Console.WriteLine("Файл создан");
                        Console.ReadKey();
                    }
                    catch (Exception)
                    {
                        Console.WriteLine("Имя содержит недопустимые знаки");
                        Console.ReadKey();
                    }
                    break;
                case "3":
                    Console.Write("Введите имя файла: ");
                    fileName = Console.ReadLine();
                    if (Path.GetExtension(fileName) == ".bin")
                    {
                        fullPath = String.Format(path + "/" + fileName);
                        OutputStream(vectors, fullPath);
                    }
                    else if (Path.GetExtension(fileName) == ".txt")
                    {
                        fullPath = String.Format(path + "/" + fileName);
                        OutputVector(vectors, fullPath);
                    }
                    else
                    {
                        string file = String.Format("{0}.*", fileName);
                        DirectoryInfo di = new DirectoryInfo(path);
                        foreach (FileInfo f in di.GetFiles(file))
                        {
                            if (Path.GetExtension(Convert.ToString(f)) ==
".bin")
                            {
                                fullPath = String.Format(pathB, fileName);
                                OutputStream(vectors, fullPath);
                            }
                            else if (Path.GetExtension(Convert.ToString(f))
== ".txt")
                            {
                                fullPath = String.Format(pathT, fileName);
                                OutputVector(vectors, fullPath);
                            }
```

```
                        }
                    }
                    break;
                case "4":
                    Console.Write("Введите имя файла: ");
                    fileName = Console.ReadLine();
                    if (Path.GetExtension(fileName) == ".bin")
                    {
                        fullPath = String.Format(path + "/" + fileName);
                        InputStream(vectors, fullPath);
                    }
                    else if (Path.GetExtension(fileName) == ".txt")
                    {
                        fullPath = String.Format(path + "/" + fileName);
                        InputVector(vectors, fullPath);
                    }
                    else
                    {
                        string file = String.Format("{0}.*", fileName);
                        DirectoryInfo di = new DirectoryInfo(path);
                        foreach (FileInfo f in di.GetFiles(file))
                        {
                            if (Path.GetExtension(Convert.ToString(f)) ==
".bin")
                            {
                                fullPath = String.Format(pathB, fileName);
                                InputStream(vectors, fullPath);
                            }
                            else if (Path.GetExtension(Convert.ToString(f))
== ".txt")
                            {
                                fullPath = String.Format(pathT, fileName);
                                InputVector(vectors, fullPath);
                            }
                        }
                    }
                    break;
                case "5":
                    Console.Write("Введите имя файла: ");
                    fileName = Console.ReadLine();
                    if (Path.GetExtension(fileName) == ".bin")
                    {
                        fullPath = String.Format(path + "/" + fileName);
                        File.Delete(fullPath);
                    }
                    else if (Path.GetExtension(fileName) == ".txt")
                    {
                        fullPath = String.Format(path + "/" + fileName);
                        File.Delete(fullPath);
                    }
                    else
                    {
                        string file = String.Format("{0}.*", fileName);
                        DirectoryInfo di = new DirectoryInfo(path);
                        foreach (FileInfo f in di.GetFiles(file))
                        {
                            if (Path.GetExtension(Convert.ToString(f)) ==
".bin")
                            {
                                fullPath = String.Format(pathB, fileName);
                                File.Delete(fullPath);
                            }
                            else if (Path.GetExtension(Convert.ToString(f))
== ".txt")
                            {
                                fullPath = String.Format(pathT, fileName);
```

```
                            File.Delete(fullPath);
                        }
                    }
                }
                break;
            default:
                return false;
        }
        Console.Clear();
        return true;
    }

    static bool SortMenu(List<IVectorable> vectors)
    {
        Console.Clear();

        string choice;
        if (vectors.Count > 0)
        {
            Console.WriteLine("Список векторов:");
            VectorsInfo(vectors);
            Console.WriteLine();
        }
        Console.WriteLine("Выберите пункт меню: ");
        Console.WriteLine("1 - Отсортировать вектора по количеству
координат");
        Console.WriteLine("2 - Отсортировать вектора по модулю");
        Console.WriteLine("Enter - Выйти в главное меню");

        Console.Write("Выбор: ");
        choice = Console.ReadLine();
        Console.WriteLine();

        switch (choice)
        {
            case "1":
                SortByLength(vectors);
                break;
            case "2":
                SortByNorm(vectors);
                break;
            default:
                return false;
        }
        Console.Clear();
        return true;
    }


    //ОСНОВНЫЕ МЕТОДЫ:


    static void CreateArrayVector(List<IVectorable> vectors)
    {
        Console.WriteLine("\nСоздание Array вектора\n");
        int len;
        int value;

        try
        {
            OutputInput("Введите размерность вектора: ", out len);
            ArrayVector vector = new ArrayVector(len);

            for (int i = 0; i < len; i++)
            {
```

```
                    string messege = String.Format("Введите координату {0}:
", i + 1);
                    OutputInput(messege, out value);
                    vector[i] = value;
                }
                vectors.Add(vector);
            }
            catch
            {
                Console.WriteLine("Размерность должна быть положительной");
                Console.ReadKey();
            }
        }

        static void CreateLinkedListVector(List<IVectorable> vectors)
        {
            Console.WriteLine("\nСоздание LinkedList вектора\n");
            int len;
            int value;

            try
            {
                OutputInput("Введите размерность вектора: ", out len);
                LinkedListVector vector = new LinkedListVector(len);

                for (int i = 0; i < len; i++)
                {
                    string messege = String.Format("Введите координату {0}:
", i + 1);
                    OutputInput(messege, out value);
                    vector[i] = value;
                }
                vectors.Add(vector);
            }
            catch
            {
                Console.WriteLine("Размерность должна быть положительной");
                Console.ReadKey();
            }
        }

        static void CloneVector(List<IVectorable> vectors)
        {
            Console.WriteLine("\nКлонирование\n");
            int numV;
            OutputInput("Введите номер вектора, который хотите клонировать:
", out numV);
            try
            {
                if (vectors[numV - 1] is ArrayVector)
                {
                    vectors.Add((IVectorable)((ArrayVector)vectors[numV -
1]).Clone());
                }
                else
                {
                    vectors.Add((IVectorable)((LinkedListVector)vectors[numV
- 1]).Clone());
                }
            }
            catch (Exception)
            {
                Console.WriteLine("Введён неверный номер");
            }
        }
```

```
        static void Sum(List<IVectorable> vectors)
        {
            Console.WriteLine("\nСумма векторов\n");
            int numV1, numV2;
            OutputInput("Введите номер первого вектора: ", out numV1);
            OutputInput("Введите номер второго вектора: ", out numV2);
            try
            {
                vectors.Add(Vectors.Sum(vectors[numV1 - 1], vectors[numV2 -
1]));
                Console.WriteLine("Полученный вектор добавлен в список");
                Console.ReadKey();
            }
            catch (Exception)
            {
                Console.WriteLine("Вектора должны быть одинаковой длины");
                Console.ReadKey();
            }
        }

        static void EditVector(List<IVectorable> vectors)
        {
            Console.WriteLine("\nРедактирование вектора\n");
            int num;
            OutputInput("Введите номер вектора: ", out num);
            try
            {
                if (vectors[num - 1].GetType() == typeof(ArrayVector))
EditArrayVector((ArrayVector)vectors[num - 1], num);
                else EditLinkedList((LinkedListVector)vectors[num - 1], num,
vectors);
            }
            catch (Exception)
            {
                Console.WriteLine("Введён неверный номер");
                Console.ReadKey();
            }
        }

        static void EditArrayVector(ArrayVector vector, int numV)
        {
            bool flag = true;
            while (flag)
            {
                Console.Clear();
                Console.WriteLine("Редактируемый вектор:");
                Console.WriteLine("{0, -5}{1, -7}{2, -15}{3}", "№", "Тип",
"Размерность", "Координаты");
                Console.WriteLine("{0, -5}{1, -7}{2}", numV, "Arr",
vector.ToString());
                Console.WriteLine();

                try
                {
                    int num;
                    int newValue;
                    Console.Write("Введите номер элемента или нажмите Enter,
чтобы звершить редактирование: ");
                    flag = Int32.TryParse(Console.ReadLine(), out num);
                    if (flag)
                    {
                        OutputInput("Введите новое значение: ", out
newValue);
                        vector[num - 1] = newValue;
                    }
                }
```

```
                catch (Exception)
                {
                    Console.WriteLine("Введён недопустимый номер элемента");
                    Console.ReadKey();
                }
            }
        }

        static void EditLinkedList(LinkedListVector vector, int numV,
List<IVectorable> vectors)
        {
            bool flag = true;
            while (flag)
            {
                Console.Clear();
                Console.WriteLine("Редактируемый вектор:");
                Console.WriteLine("{0, -5}{1, -7}{2, -15}{3}", "№", "Тип",
"Размерность", "Координаты");
                Console.WriteLine("{0, -5}{1, -7}{2}", numV, "List",
vector.ToString());
                Console.WriteLine();

                Console.WriteLine("Введите номер элемента или нажмите Enter,
чтобы звершить редактирование: ");
                Console.WriteLine("1 - Изменить элемент по номеру");
                Console.WriteLine("2 - Добавить элемент в начало");
                Console.WriteLine("3 - Добавить элемент в конец");
                Console.WriteLine("4 - Добавить элемент по номеру");
                Console.WriteLine("5 - Удалить первый элемент");
                Console.WriteLine("6 - Удалить последний элемент");
                Console.WriteLine("7 - Удалить элемент по номеру");

                Console.Write("Выбор: ");
                string choice = Console.ReadLine();
                int value = 0;
                int index = 0;

                switch (choice)
                {
                    case "1":
                        OutputInput("Ведите номер: ", out index);
                        OutputInput("Ведите значение: ", out value);
                        try
                        {
                            vector[index - 1] = value;
                        }
                        catch (Exception)
                        {
                            Console.WriteLine("Введён неверный номер");
                            Console.ReadKey();
                        }
                        break;
                    case "2":
                        OutputInput("Ведите значение: ", out value);
                        vector.AddTop(value);
                        break;
                    case "3":
                        OutputInput("Ведите значение: ", out value);
                        vector.AddEnd(value);
                        break;
                    case "4":
                        OutputInput("Ведите номер: ", out index);
                        OutputInput("Ведите значение: ", out value);
                        try
                        {
                            vector.AddByIndex(index - 1, value);
```

```
                }
                catch (Exception)
                {
                    Console.WriteLine("Введён неверный номер");
                    Console.ReadKey();
                }
                break;
            case "5":
                vector.DeleteTop();
                if (vector.Length == 0)
                {
                    vectors.Remove(vector);
                    flag = false;
                }
                break;
            case "6":
                vector.DeleteEnd();
                if (vector.Length == 0)
                {
                    vectors.Remove(vector);
                    flag = false;
                }
                break;
            case "7":
                OutputInput("Ведите номер: ", out index);
                try
                {
                    vector.DeleteByIndex(index - 1);
                }
                catch (Exception)
                {
                    Console.WriteLine("Введён неверный номер");
                    Console.ReadKey();
                }
                if (vector.Length == 0)
                {
                    vectors.Remove(vector);
                    flag = false;
                }
                break;
            default:
                flag = false;
                break;
        }
    }
}

static void Delete(List<IVectorable> vectors)
{
    Console.WriteLine("\nУдаление вектора\n");
    int num;
    OutputInput("Введите номер вектора: ", out num);
    try
    {
        vectors.RemoveAt(num - 1);
    }
    catch (Exception)
    {
        Console.WriteLine("Введён неверный номер");
        Console.ReadKey();
    }
}

static void OutputStream(List<IVectorable> vectors, string fullPath)
{
    Console.WriteLine("\nДобавление в файл данных\n");
```

```
            Console.Write("Введите номера векторов через пробел: ");
            List<int> nums = new List<int>();
            foreach (string n in Console.ReadLine().Split(' '))
            {
                int num;
                bool parsed = Int32.TryParse(n, out num);
                if (parsed && 0 <= num && num <= vectors.Count)
                {
                    nums.Add(num);
                }
            }
            using (FileStream aFileStream = new FileStream(fullPath,
FileMode.Append))
            {
                aFileStream.Seek(0, SeekOrigin.End);
                for (int i = 0; i < nums.Count; i++)
                {
                    Vectors.OutputVector(vectors[nums[i] - 1], aFileStream);
                }
            }

            if (nums.Count > 0)
            {
                Console.WriteLine("Вектора добавлены в файл");
                Console.ReadKey();
            }
            else
            {
                Console.WriteLine("Векторов с такими номерами нет");
                Console.ReadKey();
            }
        }

        static void OutputVector(List<IVectorable> vectors, string fullPath)
        {
            Console.WriteLine("\nДобавление в текстовый файл\n");
            Console.Write("Введите номера векторов через пробел: ");
            List<int> nums = new List<int>();
            foreach (string n in Console.ReadLine().Split(' '))
            {
                int num;
                bool parsed = Int32.TryParse(n, out num);
                if (parsed && 0 <= num && num <= vectors.Count)
                {
                    nums.Add(num);
                }
            }
            using (TextWriter writer = File.AppendText(fullPath))
            {
                for (int i = 0; i < nums.Count; i++)
                {
                    Vectors.WriteVector(vectors[nums[i] - 1], writer);
                }
            }

            if (nums.Count > 0)
            {
                Console.WriteLine("Вектора добавлены в файл");
                Console.ReadKey();
            }
            else
            {
                Console.WriteLine("Векторов с такими номерами нет");
                Console.ReadKey();
            }
        }
```

```
        static void InputStream(List<IVectorable> vectors, string fullPath)
        {
            Console.WriteLine("\nИзвлечение из файла данных");
            try
            {
                using (FileStream rFileStream = new FileStream(fullPath,
FileMode.Open))
                {
                    while (rFileStream.Position != rFileStream.Length)
                    {

vectors.Add((ArrayVector)Vectors.InputVector(rFileStream));
                    }
                }

                Console.WriteLine("Вектора добавлены в список");
                Console.ReadKey();
            }
            catch (Exception)
            {
                Console.WriteLine("Файл не найден");
                Console.ReadKey();
            }
        }

        static void InputVector(List<IVectorable> vectors, string fullPath)
        {
            Console.WriteLine("\nИзвлечение из текстового файла");
            try
            {
                using (TextReader reader = File.OpenText(fullPath))
                {
                    while (reader.Peek() >= 0)
                    {
                        vectors.Add((ArrayVector)Vectors.ReadVector(reader));
                    }
                }

                Console.WriteLine("Вектора добавлены в список");
                Console.ReadKey();
            }
            catch (Exception)
            {
                Console.WriteLine("Файл не найден");
                Console.ReadKey();
            }
        }

        static void VectorNorm(List<IVectorable> vectors)
        {
            Console.WriteLine("\nВычисление длины\n");
            int numV;
            OutputInput("Введите номер вектора: ", out numV);
            try
            {
                Console.WriteLine("Модуль вектора №{0}: {1}", numV,
vectors[numV - 1].GetNorm());
                Console.ReadKey();
            }
            catch (Exception)
            {
                Console.WriteLine("Введён неверный номер");
                Console.ReadKey();
            }
        }
```

```
        static void IsEquals(List<IVectorable> vectors)
        {
            Console.WriteLine("\nПроверка на равенство\n");
            int numV1, numV2;
            OutputInput("Введите номер первого вектора: ", out numV1);
            OutputInput("Введите номер второго вектора: ", out numV2);
            try
            {
                if (vectors[numV1 - 1].Equals(vectors[numV2 - 1]))
                {
                    Console.WriteLine("Вектора равны");
                }
                else
                {
                    Console.WriteLine("Вектора не равны");
                }
                Console.ReadKey();
            }
            catch (Exception)
            {
                Console.WriteLine("Введён неверный номер");
                Console.ReadKey();
            }
        }

        static void Mult(List<IVectorable> vectors)
        {
            Console.WriteLine("\nУмножение векторов\n");
            int numV1, numV2;
            OutputInput("Введите номер первого вектора: ", out numV1);
            OutputInput("Введите номер второго вектора: ", out numV2);
            try
            {
                Console.WriteLine("Значение: {0}",
Vectors.Scalar(vectors[numV1 - 1], vectors[numV2 - 1]));
                Console.ReadLine();
            }
            catch (Exception)
            {
                Console.WriteLine("Вектора должны быть одинаковой длины");
                Console.ReadKey();
            }
        }

        static void MinMaxLength(List<IVectorable> vectors)
        {
            Console.WriteLine("\nМинимальная/Максимальная длина\n");
            try
            {
                int minL = vectors[0].Length;
                int maxL = vectors[0].Length;
                for (int i = 0; i < vectors.Count(); i++)
                {
                    if (minL > vectors[i].Length)
                    {
                        minL = vectors[i].Length;
                    }
                    if (maxL < vectors[i].Length)
                    {
                        maxL = vectors[i].Length;
                    }
                }
                string type;
                Console.WriteLine("Вектора с минимальным количеством
координат:");
```

```
                Console.WriteLine("{0, -7}{1, -15}{2}", "Тип", "Размерность",
"Координаты");
                for (int i = 0; i < vectors.Count(); i++)
                {
                    if (vectors[i].Length == minL)
                    {
                        if (vectors[i].GetType() == typeof(ArrayVector)) type
= "Arr";
                        else type = "List";
                        Console.WriteLine("{0, -7}{1}", type,
vectors[i].ToString());
                    }
                }
                Console.WriteLine("Вектора с максимальным количеством
координат:");
                Console.WriteLine("{0, -7}{1, -15}{2}", "Тип", "Размерность",
"Координаты");
                for (int i = 0; i < vectors.Count(); i++)
                {
                    if (vectors[i].Length == maxL)
                    {
                        if (vectors[i].GetType() == typeof(ArrayVector)) type
= "Arr";
                        else type = "List";
                        Console.WriteLine("{0, -7}{1}", type,
vectors[i].ToString());
                    }
                }
                Console.ReadKey();
            }
            catch (Exception)
            {
                Console.WriteLine("Нет векторов для вывода");
                Console.ReadKey();
            }
        }

        static void SortByLength(List<IVectorable> vectors)
        {
            IVectorable temp;
            for (int i = 0; i < vectors.Count(); i++)
            {
                for (int j = 0; j < vectors.Count() - 1 - i; j++)
                {
                    if (((IComparable)vectors[j]).CompareTo(vectors[j + 1]) >
0)
                    {
                        temp = vectors[j];
                        vectors[j] = vectors[j + 1];
                        vectors[j + 1] = temp;
                    }
                }
            }
        }

        static void SortByNorm(List<IVectorable> vectors)
        {
            IVectorable temp;
            VectorsComparer comparer = new VectorsComparer();
            for (int i = 0; i < vectors.Count(); i++)
            {
                for (int j = 0; j < vectors.Count() - 1 - i; j++)
                {
                    if (comparer.Compare(vectors[j], vectors[j + 1]) > 0)
                    {
                        temp = vectors[j];
```

```
                        vectors[j] = vectors[j + 1];
                        vectors[j + 1] = temp;
                    }
                }
            }
        }


        //ВСПОМОГАТЕЛЬНЫЕ МЕТОДЫ:


        static void ConsoleClear(List<IVectorable> vectors)
        {
            Console.Clear();
        }

        static void VectorsInfo(List<IVectorable> vectors)
        {
            Console.WriteLine("{0, -5}{1, -7}{2, -15}{3}", "№", "Тип",
"Размерность", "Координаты");
            string type;
            for (int i = 0; i < vectors.Count; i++)
            {
                if (vectors[i].GetType() == typeof(ArrayVector)) type =
"Arr";
                else type = "List";
                Console.WriteLine("{0, -5}{1, -7}{2}", i + 1, type,
vectors[i].ToString());
            }
        }

        static void FileList()
        {
            DirectoryInfo di = new DirectoryInfo("//mscs-
fs/data/314_user_data/!  C# 2023-2024/2 семестр - C#. Отчеты/6101/Дьячков
Д/ЛР05/ЛР05/ЛР05/FileStream");
            Console.WriteLine("Список файлов: ");
            foreach (FileInfo file in di.GetFiles("*.bin"))
            {
                Console.WriteLine(file);
            }
            foreach (FileInfo file in di.GetFiles("*.txt"))
            {
                Console.WriteLine(file);
            }
        }

        static void OutputInput(string output, out int input)
        {
            bool inputCompleted = false;
            do
            {
                Console.Write(output);
                inputCompleted = Int32.TryParse(Console.ReadLine(), out
input);
                if (!inputCompleted)
                {
                    Console.WriteLine("\nВведённые данные некорректны,
повторите ввод\n");
                }
            } while (!inputCompleted);
        }
    }
}
```

```
Выберете действие:

        1  - Сумма векторов
        2  - Скалярное умножение
        3  - Умножение на число
        4  - Рассчитать модуль вектора
        5  - Добавить вектор в список
        6  - Удалить вектор из списка
        7  - Клонировать вектор
        8  - Вывести вектора с минимальным и максимальным количеством координат
        9  - Отсортировать вектора по количеству координат
        10 - Отсортировать вектора по модулю
        11 - Сравнить вектора
        0  - Выход

Введите траекторию выполнения программы через пробел:
```

Рисунок 1 – Главное меню программы

```
Введите траекторию выполнения программы через пробел: 5 5 1
Выберете тип вектора:

        1 - Вектор
        2 - Связный список

1
Введите длину вектора: 3
1:      Array: 3 76 62 88
Выберете тип вектора:

        1 - Вектор
        2 - Связный список

2
Введите длину связного списка: 3
```

Рисунок 2 – Создание векторов

```
1:      Array: 3 76 62 88
2: LinkedList: 3 44 46 17
Введите индекс первого вектора: 1
Введите индекс второго вектора: 2
Результат сложения 1-го и 2-го векторов: 3 120 108 105
1:      Array: 3 76 62 88
2: LinkedList: 3 44 46 17
Прорамма завершила работу . . .
```

Рисунок 3 – Сумма векторов, завершение программы

ВЫВОДЫ

В лабораторной работе были использованы конструкции языка:

– форматированный вывод информации на консоль;
– оператор switch;
– условные операторы;
– функции;
– классы;
– конструкторы класса;
– поля класса;
– статические и динамические методы класса;
– интерфейсы;
– индексаторы;
– байтовые и символьные потоки;
– делегаты;
– конструкция try-catch.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Павловская Т.А. C#. Программирование на языке высокого уровня. Учебник для вузов [Текст]/Т.А. Павловская. – СПб.: Питер, 2007. – 432 с.