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

№1 «Основы языка C#: Односвязный список»

по дисциплине «Языки программирования и структуры данных»

Выполнил Фадеев А.М. 6101

Проверил Котенёва С.Э.

Самара

2024

ЗАДАНИЕ

Задание 1.

Прочитать теоретический материал.

Задание 2.

Привести класс «вектор» с именем “ArrayVector” к следующей структуре:

– поле – массив элементов целого типа (координаты конца вектора в n-мерном

– пространстве);

– конструктор с параметром – длиной массива;

– конструктор без параметра, задающий длину массива 5;

– индексатор для организации доступа к элементам массива, выбрасывающий

– исключения при вводе некорректного индекса;

– метод GetNorm() вычисления модуля/нормы вектора;

– свойство для чтения числа координат вектора.

Где необходимо выбрасывать, исключения в случае невозможности проведения указанных действий над векторами.

Задание 3.

Описать класс «односвязный список» с именем «LinkedListVector», содержащий координаты конца вектора в n-мерном пространстве в виде динамического односвязного списка. Каждый элемент массива представляет собой отдельный объект «узел» класса «Node», класс Node является внутренним для класса LinkedListVector.

Структура класса «Node»:

– поле – элемент целого типа (по умолчанию = 0);

– поле – ссылка на элемент класса Node (по умолчанию = null);

– Структура класса «LinkedListVector»:

– поле – ссылка на начало списка (на экземпляр класса Node);

– конструктор с параметром – длиной списка;

– конструктор без параметра, задающий длину списка 5;

– индексатор для организации доступа к элементам списка, выбрасывающий исключение при некорректном индексе;

– метод GetNorm() вычисления модуля вектора;

– свойство для чтения числа координат вектора;

а также методы:

– удаления и добавления элемента в конец и в начало;

– удаления и добавления элемента в заданную позицию.

Где необходимо выбрасывать, исключения в случае невозможности проведения указанных действий над векторами.

Задание 4.

Описать класс с именем «Vectors», содержащий следующие публичные статические методы:

– Sum() сложения двух векторов, который принимает в качестве параметра 2 объекта

– ArrayVector и возвращает новый объект ArrayVector;

– Scalar() скалярного произведения двух векторов, который принимает в качестве

– параметра 2 объекта ArrayVector и возвращает целое число;

– GetNormSt() получения модуля/нормы вектора, который принимает в качестве параметра объект ArrayVector и возвращает вещественное число.

Выбрасывать исключения в методах Sum() и Scalar() в случае невозможности проведения указанных действий над векторами (например, FormatException).

Задание 5.

Проверить функциональность классов в методе Main() класса Program. Разработать адекватный пользовательский интерфейс.

Отлавливать все возможные исключения – некорректный ввод пункта меню, некорректный ввод данных, несовпадение длин векторов в

статических методах класса Vectors и т.д. То есть сделать программу правильно реагирующей на предсказуемые ошибки.

Задание 6.

Подготовить отчет о работе.

КОД ПРОГРАММЫ

```
using System.Net.Sockets;

namespace Lab02;

public class LinkedListVector
{
    private Node _head;

    private class Node
    {
        public int Value;
        public Node Next;

        public Node()
        {
            Value = 0;
            Next = null;
        }

        public Node(int value)
        {
            Value = value;
            Next = null;
        }
    }

    public LinkedListVector()
    {
        var r = new Random();

        _head = new Node(r.Next(100));
        Node cur = _head;

        for (int i = 0; i < 5; i++)
        {
            cur.Next = new Node(r.Next(100));
            cur = cur.Next;
        }
    }

    public LinkedListVector(int length)
```

```
    {
        var r = new Random();

        _head = new Node(r.Next(100));
        Node cur = _head;

        for (int i = 0; i < length; i++)
        {
            cur.Next = new Node(r.Next(100));
            cur = cur.Next;
        }
    }

    public int this[int idx]
    {
        get
        {
            if (0 <= idx && idx <= Length)
            {
                Node cur = _head;
                for (int i = 0; i < idx; i++)
                {
                    cur = cur.Next;
                }

                return cur.Value;
            }
            else
            {
                throw new IndexOutOfRangeException("Linked list index out of
range");
            }
        }
        set
        {
            if (0 <= idx && idx <= Length)
            {
                Node cur = _head;
                for (int i = 0; i < idx; i++)
                {
                    cur = cur.Next;
                }

                cur.Value = value;
```

```
            }
            else
            {
                throw new IndexOutOfRangeException("Linked list index out of
range");
            }
        }
    }

    public int Length
    {
        get
        {
            if (_head == null)
            {
                return -1;
            }

            int length = 0;
            Node cur = _head;
            while (cur.Next != null)
            {
                cur = cur.Next;
                length++;
            }

            return length;
        }
    }

    public double GetNorm()
    {
        double acc = 0;
        Node cur = _head;
        for (int i = 0; i < Length; i++)
        {
            acc += Math.Pow(cur.Value, 2);
            cur = cur.Next;
        }

        return Math.Sqrt(acc);
    }

    public void AddToStart(int value)
```

```
    {
        Node tmp = new Node(value);
        tmp.Next = _head;
        _head = tmp;
    }

    public void AddToEnd(int value)
    {
        AddByIndex(Length, value);
    }

    public void AddByIndex(int idx, int value)
    {
        if (0 <= idx && idx <= Length)
        {
            Node cur = _head;
            for (int i = 0; i < idx - 1; i++)
            {
                cur = cur.Next;
            }

            Node tmp = new Node(value);
            tmp.Next = cur.Next;
            cur.Next = tmp;
        }
        else
        {
            throw new IndexOutOfRangeException("Linked list index out of
range");
        }
    }

    public void DeleteFromStart()
    {
        _head = _head.Next;
    }

    public void DeleteFromEnd()
    {
        Node cur = _head;
        for (int i = 0; i < Length - 1; i++)
        {
            cur = cur.Next;
        }
```

```
            cur.Next = null;
        }

        public void DeleteByIndex(int idx)
        {
            if (0 <= idx && idx <= Length)
            {
                Node cur = _head;
                for (int i = 0; i < idx - 1; i++)
                {
                    cur = cur.Next;
                }

                cur.Next = cur.Next.Next;
            }
            else
            {
                throw new IndexOutOfRangeException("Linked list index out of
range");
            }
        }

        public void Log(string message = "")
        {
            if (message != "")
            {
                Console.Write($"{message}: ");
            }
            var cur = _head;
            Console.Write("{");
            while (cur.Next != null)
            {
                if (cur.Next.Next == null)
                {
                    Console.Write(cur.Value);
                }
                else
                {
                    Console.Write(cur.Value + ", ");
                }
                cur = cur.Next;
            }
            Console.WriteLine("}");
```

```
    }
}
using Lab01;

namespace Lab02;

public static class Program
{
    public static void Main(string[] args)
    {
        Greeting();

        string inp;

        while (true)
        {
            Console.WriteLine("Выберете класс для работы:\n\n" +
                              "\t1 - LinkedListVector\n" +
                              "\t2 - ArrayVector\n" +
                              "\t0 - Выход");

            inp = Console.ReadLine();

            switch (inp)
            {
                case "1":
                    TestLinkedListVectorClass();
                    break;
                case "2":
                    TestArrayVectorClass();
                    break;
                case "0":
                    Console.WriteLine("До скорой встречи, до скорой
встречи!");
                    return;
                default:
                    Console.WriteLine("Нет такого пункта в меню");
                    break;
            }

            Console.WriteLine("Нажмите любую клавишу для продолжения...");
            Console.ReadKey();
        }
    }
```

```
public static void TestLinkedListVectorClass()
{
    string inp;
    int length;
    do
    {
        Console.Write("Введите длину вектора: ");
        inp = Console.ReadLine();
    } while (!int.TryParse(inp, out length) || length <= 0);

    var vec = new LinkedListVector(length);
    vec.Log("Созданный вектор");

    while (true)
    {
        Console.WriteLine("Выберете действие:\n\n" +
                          "\t1 - Добавить элемент в начало списка\n" +
                          "\t2 - Добавить элемент в конец списка\n" +
                          "\t3 - Добавить элемент по индексу\n" +
                          "\t4 - Удалить первый элемент\n" +
                          "\t5 - Удалить последний элемент\n" +
                          "\t6 - Удалить элемент по индексу\n" +
                          "\t0 - Выход в меню\n");

        inp = Console.ReadLine();

        switch (inp)
        {
            case "1":
            {
                int value;
                do
                {
                    Console.Write("Введите значение для добавления: ");
                    inp = Console.ReadLine();
                } while (!int.TryParse(inp, out value));

                vec.AddToStart(value);
                vec.Log("Обновленный вектор");
                break;
            }
            case "2":
            {
```

```
                    int value;
                    do
                    {
                        Console.Write("Введите значение для добавления: ");
                        inp = Console.ReadLine();
                    } while (!int.TryParse(inp, out value));

                    vec.AddToEnd(value);
                    vec.Log("Обновленный вектор");
                    break;
                }
                case "3":
                {
                    int value;
                    do
                    {
                        Console.Write("Введите значение для добавления: ");
                        inp = Console.ReadLine();
                    } while (!int.TryParse(inp, out value));

                    int idx;
                    do
                    {
                        Console.Write("Введите позицию для добавления
элемента: ");
                        inp = Console.ReadLine();
                    } while (!int.TryParse(inp, out idx));

                    try
                    {
                        vec.AddByIndex(idx, value);
                        vec.Log("Обновленный вектор");
                    }
                    catch (IndexOutOfRangeException e)
                    {
                        Console.WriteLine("Введенный индекс выходит за рамки
связного списка");
                    }
                    break;
                }
                case "4":
                {
                    vec.DeleteFromStart();
                    vec.Log("Обновленный вектор");
```

```
                    break;
                }
                case "5":
                {
                    vec.DeleteFromEnd();
                    vec.Log("Обновленный вектор");
                    break;
                }
                case "6":
                {
                    int idx;
                    do
                    {
                        Console.Write("Введите индекс элемента который хотите
удалить: ");
                        inp = Console.ReadLine();
                    } while (!int.TryParse(inp, out idx));

                    try
                    {
                        vec.DeleteByIndex(idx);
                        vec.Log("Обновленный вектор");
                    }
                    catch (IndexOutOfRangeException e)
                    {
                        Console.WriteLine("Введенный индекс выходит за рамки
связного списка");
                    }

                    break;
                }
                case "0":
                {
                    return;
                }
                default:
                {
                    Console.WriteLine("Нет такого пункта в меню");
                    break;
                }
            }
            Console.WriteLine("Нажмите любую клавишу для продолжения...");
            Console.ReadKey();
        }
```

```
    }

    public static void TestArrayVectorClass()
    {
        string inp;

        ArrayVector vector = ArrayVector.GetFromUserInput();
        vector.Log();

        while (true)
        {
            ArrayVector vector2;

            Console.WriteLine("Выберете действие:\n\n" +
                              "\t1 - Пересоздать вектор\n" +
                              "\t2 - Получить модуль вектора\n" +
                              "\t3 - Сумма двух векторов\n" +
                              "\t4 - Скалярное произведение двух векторов\n"
+
                              "\t0 - Выход");

            inp = Console.ReadLine();

            switch (inp)
            {
                case "1":
                    vector = ArrayVector.GetFromUserInput();
                    vector.Log("Новый вектор");
                    break;
                case "2":
                    Console.WriteLine($"Модуль вектора равен:
{vector.GetNorm()}");
                    break;
                case "3":
                    vector2 = ArrayVector.GetFromUserInput();
                    vector.Log("Первый вектор");
                    vector2.Log("Второй вектор");
                    Vectors.Sum(vector, vector2).Log("Результат сложения двух
векторов");
                    break;
                case "4":
                    vector2 = ArrayVector.GetFromUserInput();
                    vector.Log("Первый вектор");
                    vector2.Log("Второй вектор");
```

```
                    Console.WriteLine($"Результат скалярного умножения
векторов: {Vectors.ScalarMultiply(vector, vector2)}");
                    break;
                case "0":
                    return;
                default:
                    Console.WriteLine("Нет такого пункта в меню");
                    break;
            }

            Console.WriteLine("Нажмите любую клавишу для продолжения...");
            Console.ReadKey();
        }
    }

    public static void Greeting()
    {
        Console.WriteLine("Языки Программирования и Структуры Данных\n" +
                          "Лабораторная работа #2\n" +
                          "Выполнил студент группы 6101-020302D - Фадеев
Артем");
    }
}
```

```
Языки Программирования и Структуры Данных
Лабораторная работа #2
Выполнил студент группы 6101-020302D - Фадеев Артем
Выберете класс для работы:

        1 - LinkedListVector
        2 - ArrayVector
        0 - Выход
```

Рисунок 1 – Главное меню программы

```
Выберете класс для работы:

        1 - LinkedListVector
        2 - ArrayVector
        0 - Выход
1
Введите длину вектора: 7
Созданный вектор: {42, 12, 14, 35, 34, 58, 58}
Выберете действие:

        1 - Добавить элемент в начало списка
        2 - Добавить элемент в конец списка
        3 - Добавить элемент по индексу
        4 - Удалить первый элемент
        5 - Удалить последний элемент
        6 - Удалить элемент по индексу
        0 - Выход в меню
```

Рисунок 2 – Работа с классом LinkedListVector

```
Выберете класс для работы:

        1 - LinkedListVector
        2 - ArrayVector
        0 - Выход
2
Выберете как хотите заполнить вектор:
1 - Случайно
2 - Ручной ввод
1
Введите длину вектора: 5
{38, 78, 26, 86, 68}
Выберете действие:

        1 - Пересоздать вектор
        2 - Получить модуль вектора
        3 - Сумма двух векторов
        4 - Скалярное произведение двух векторов
        0 - Выход
```

Рисунок 3 – Работа с классом ArrayVector

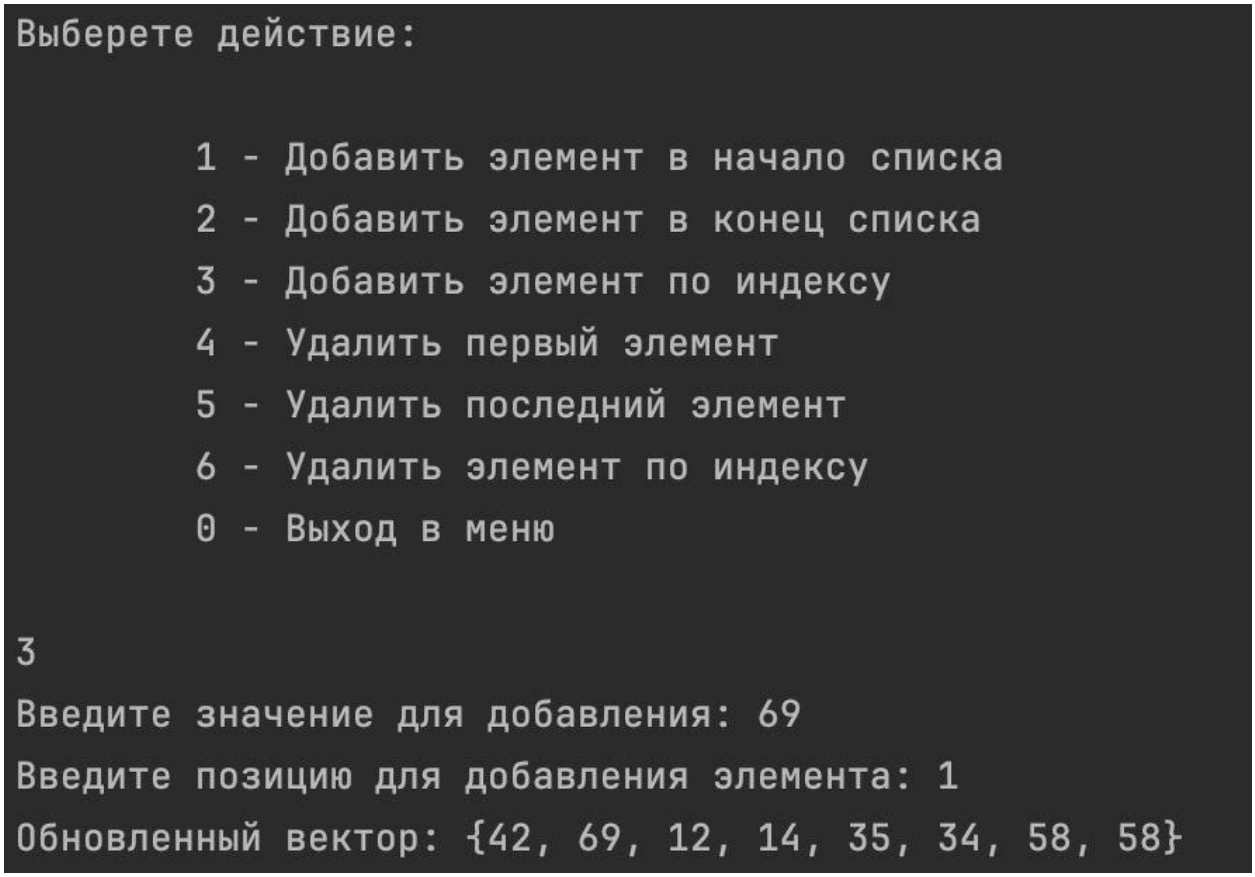

Рисунок 4 – Добавление элемента по индексу

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;
- 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.