****

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

Институт (факультет) Информатики и кибернетики

Кафедра Программных систем

**ОТЧЁТ**

**по лабораторной работе**

№7 «Делегаты»  
по дисциплине «Основы программирования»

Выполнил Мананников М.А. 6103- 020302D

Проверил Котенева Светлана Эдуардовна

Самара  
2022

ЗАДАНИЕ

Задание 1

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

* Описать делегат без параметров, возвращающий void;
* в классе Program описать статическими методами все пункты меню, сигнатура методов совпадает с сигнатурой, заданной делегатом;
* запросить у пользователя траекторию выполнения программы – набор пунктов меню, 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.

Задание 3

Подготовить отчет о работе.

**КОДПРОГРАММЫ**

using System;

namespace ЛР\_05

{

class Program

{

delegate void Del();

static void Main(string[] args)

{

Del del = null;

Console.WriteLine("Лабораторная работа №7: Делегаты\nВыполнил Мананников Максим, студент группы 6103-020302D");

Console.WriteLine("Задайте траекторию программы:\n0 - Исполнение траектории\n1 - Задание №1(ЛР №2)\n2 - Задание №2 (ЛР №2)\n3 - Задание №3 (ЛР №4)\n4 - Задание №4 (ЛР №5)\n5 - Выход");

int a = 1;

while (a == 1)

{

switch (Console.ReadLine())

{

case "1":

del += Task1;

break;

case "2":

del += Task2;

break;

case "3":

del += Task3;

break;

case "4":

del += Task4;

break;

case "5":

a = 0;

break;

case "0":

del();

break;

default:

Console.WriteLine("Ошибка ввода! Повторите попытку!");

break;

}

}

Console.ReadKey();

}

static void Task1()

{

while (true)

{

try

{

int number;

Console.Write("Введите количество координат: ");

number = int.Parse(Console.ReadLine());

ArrayVector v1 = new ArrayVector(number);

Console.WriteLine("Введите массив:");

int digit;

while (true)

{

try

{

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

{

digit = int.Parse(Console.ReadLine());

v1.SetElement(i, digit);

}

break;

}

catch (Exception)

{

Console.WriteLine("Ошибка ввода! Повторите попытку!");

}

}

try

{

Console.WriteLine();

Console.Write("Выберите какой элемент хотите вывести: ");

int i = int.Parse(Console.ReadLine());

Console.WriteLine("Результат: " + v1.GetElement(i - 1));

}

catch (Exception)

{

Console.WriteLine("Ошибка! Введите другой индекс!");

}

Console.WriteLine();

Console.WriteLine("Вывод вектора: " + v1.ToString());

Console.WriteLine("Модуль списка: " + v1.GetNorm());

break;

}

catch (Exception)

{

Console.WriteLine("Ошибка! Попробуйте ещё раз!");

}

}

}

static void Task2()

{

while (true)

{

try

{

Console.Write("Введите длину списка: ");

int length = int.Parse(Console.ReadLine());

LinkedListVector vector = new LinkedListVector(length);

Console.WriteLine();

Console.WriteLine("Введите элементы списка: ");

for (LinkedListVector.Node node = vector.Head; node != null; node = node.Link)

{

node.value = Convert.ToInt32(Console.ReadLine());

}

try

{

Console.WriteLine();

Console.Write("Выберите элемент, который хотите вывести: ");

int i = int.Parse(Console.ReadLine());

double a1 = vector[i];

Console.Write("Результат: " + a1);

}

catch

{

throw new Exception("Ошибка! Введите другой индекс!");

}

Console.WriteLine();

Console.WriteLine("Вывод списка: " + vector.ToString());

Console.WriteLine("Модуль списка: " + vector.GetNorm());

break;

}

catch (Exception)

{

Console.WriteLine();

Console.WriteLine("Ошибка ввода! Повторите попытку!");

}

}

}

static void Task3()

{

Console.WriteLine("Выберите, каким образом хотите работать с векторами:\n1 - ArrayVector и ArrayVector\n2 - LinkedListVector и LinkedListVector");

Console.WriteLine("3 - ArrayVector и LinkedListvector\n4 - LinkedListVector и ArrayVector\n0 - Выход");

while (true)

{

switch (Console.ReadLine())

{

case "1":

try

{

Console.Write("Введите количество координат первого вектора: ");

int length1 = int.Parse(Console.ReadLine());

ArrayVector arrayVector1 = new ArrayVector(length1);

Console.WriteLine();

Console.Write("Введите количество координат второго вектора: ");

int length2 = int.Parse(Console.ReadLine());

ArrayVector arrayVector2 = new ArrayVector(length2);

try

{

Console.WriteLine();

Console.WriteLine("Введите первый вектор:");

for (int i = 0; i < length1; i++)

{

arrayVector1.SetElement(i, int.Parse(Console.ReadLine()));

}

Console.WriteLine();

Console.WriteLine("Введите второй вектор:");

for (int i = 0; i < length2; i++)

{

arrayVector2.SetElement(i, int.Parse(Console.ReadLine()));

}

}

catch (Exception)

{

Console.WriteLine("Ошибка ввода!");

}

Console.WriteLine();

try

{

Vectors.Sum(arrayVector1, arrayVector2);

}

catch (Exception)

{

Console.WriteLine("Операция сложения невозможна!");

}

Console.WriteLine();

try

{

Console.WriteLine(Vectors.Scalar(arrayVector1, arrayVector2));

}

catch (Exception)

{

Console.WriteLine("Операция скалярного произведения невозможна!");

}

Console.WriteLine();

Console.Write("Модуль первого вектора: " + Vectors.GetNorm(arrayVector1));

Console.WriteLine();

Console.Write("Модуль второго вектора: " + Vectors.GetNorm(arrayVector2));

Console.WriteLine();

Console.WriteLine("Вывод списка: " + arrayVector1.ToString());

Console.WriteLine("Вывод списка: " + arrayVector2.ToString());

break;

}

catch (Exception)

{

Console.WriteLine();

Console.WriteLine("Ошибка ввода! Повторите попытку!");

}

break;

case "2":

try

{

Console.Write("Введите количество координат первого вектора: ");

int length1 = int.Parse(Console.ReadLine());

LinkedListVector linkedListVector1 = new LinkedListVector(length1);

Console.WriteLine();

Console.Write("Введите количество координат второго вектора: ");

int length2 = int.Parse(Console.ReadLine());

LinkedListVector linkedListVector2 = new LinkedListVector(length2);

try

{

Console.WriteLine();

Console.WriteLine("Введите первый вектор:");

for (LinkedListVector.Node node = linkedListVector1.Head; node != null; node = node.Link)

{

node.value = Convert.ToInt32(Console.ReadLine());

}

Console.WriteLine();

Console.WriteLine("Введите второй вектор:");

for (LinkedListVector.Node node = linkedListVector2.Head; node != null; node = node.Link)

{

node.value = Convert.ToInt32(Console.ReadLine());

}

}

catch (Exception)

{

Console.WriteLine("Ошибка ввода!");

}

Console.WriteLine();

try

{

Vectors.Sum(linkedListVector1, linkedListVector2);

}

catch (Exception)

{

Console.WriteLine("Операция сложения невозможна!");

}

Console.WriteLine();

try

{

Console.WriteLine(Vectors.Scalar(linkedListVector1, linkedListVector2));

}

catch (Exception)

{

Console.WriteLine("Операция скалярного произведения невозможна!");

}

Console.WriteLine();

Console.Write("Модуль первого вектора: " + Vectors.GetNorm(linkedListVector1));

Console.WriteLine();

Console.Write("Модуль второго вектора: " + Vectors.GetNorm(linkedListVector2));

Console.WriteLine();

Console.WriteLine("Вывод списка: " + linkedListVector1.ToString());

Console.WriteLine("Вывод списка: " + linkedListVector2.ToString());

break;

}

catch (Exception)

{

Console.WriteLine();

Console.WriteLine("Ошибка ввода! Повторите попытку!");

}

break;

case "3":

try

{

Console.Write("Введите количество координат первого вектора: ");

int length1 = int.Parse(Console.ReadLine());

ArrayVector arrayVector = new ArrayVector(length1);

Console.WriteLine();

Console.Write("Введите количество координат второго вектора: ");

int length2 = int.Parse(Console.ReadLine());

LinkedListVector linkedList = new LinkedListVector(length2);

try

{

Console.WriteLine();

Console.WriteLine("Введите первый вектор:");

for (int i = 0; i < length1; i++)

{

arrayVector.SetElement(i, int.Parse(Console.ReadLine()));

}

Console.WriteLine();

Console.WriteLine("Введите второй вектор:");

for (LinkedListVector.Node node = linkedList.Head; node != null; node = node.Link)

{

node.value = Convert.ToInt32(Console.ReadLine());

}

}

catch (Exception)

{

Console.WriteLine("Ошибка ввода!");

}

Console.WriteLine();

try

{

Vectors.Sum(arrayVector, linkedList);

}

catch (Exception)

{

Console.WriteLine("Операция сложения невозможна!");

}

Console.WriteLine();

try

{

Console.WriteLine(Vectors.Scalar(arrayVector, linkedList));

}

catch (Exception)

{

Console.WriteLine("Операция скалярного произведения невозможна!");

}

Console.WriteLine();

Console.Write("Модуль первого вектора: " + Vectors.GetNorm(arrayVector));

Console.WriteLine();

Console.Write("Модуль второго вектора: " + Vectors.GetNorm(linkedList));

Console.WriteLine();

Console.WriteLine("Вывод списка: " + arrayVector.ToString());

Console.WriteLine("Вывод списка: " + linkedList.ToString());

break;

}

catch (Exception)

{

Console.WriteLine();

Console.WriteLine("Ошибка ввода! Повторите попытку!");

}

break;

case "4":

try

{

Console.Write("Введите количество координат первого вектора: ");

int length1 = int.Parse(Console.ReadLine());

LinkedListVector linkedList1 = new LinkedListVector(length1);

Console.WriteLine();

Console.Write("Введите количество координат второго вектора: ");

int length2 = int.Parse(Console.ReadLine());

ArrayVector arrayVector1 = new ArrayVector(length2);

try

{

Console.WriteLine();

Console.WriteLine("Введите первый вектор:");

for (LinkedListVector.Node node = linkedList1.Head; node != null; node = node.Link)

{

node.value = Convert.ToInt32(Console.ReadLine());

}

Console.WriteLine();

Console.WriteLine("Введите второй вектор:");

for (int i = 0; i < length2; i++)

{

arrayVector1.SetElement(i, int.Parse(Console.ReadLine()));

}

}

catch (Exception)

{

Console.WriteLine("Ошибка ввода!");

}

Console.WriteLine();

try

{

Vectors.Sum(linkedList1, arrayVector1);

}

catch (Exception)

{

Console.WriteLine("Операция сложения невозможна!");

}

Console.WriteLine();

try

{

Console.WriteLine(Vectors.Scalar(linkedList1, arrayVector1));

}

catch (Exception)

{

Console.WriteLine("Операция скалярного произведения невозможна!");

}

Console.WriteLine();

Console.Write("Модуль первого вектора: " + Vectors.GetNorm(linkedList1));

Console.WriteLine();

Console.Write("Модуль второго вектора: " + Vectors.GetNorm(arrayVector1));

Console.WriteLine();

Console.WriteLine("Вывод списка: " + linkedList1.ToString());

Console.WriteLine("Вывод списка: " + arrayVector1.ToString());

break;

}

catch (Exception)

{

Console.WriteLine();

Console.WriteLine("Ошибка ввода! Повторите попытку!");

}

break;

case "0":

return;

default:

Console.WriteLine("Ошибка ввода! Повторите попытку!");

break;

}

}

}

static void Task4()

{

Console.WriteLine("Введите количество векторов: ");

int count = int.Parse(Console.ReadLine());

if (count > 0)

{

IVectorable[] mass\_vec = new IVectorable[count];

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

{

while (true)

{

Console.WriteLine("Выберите вектор, с которым хотите работать:\n1 - ArrayVector\n2 - LinkedListVector\n0 - Выход");

switch (Console.ReadLine())

{

case "1":

try

{

while (true)

{

Console.WriteLine();

Console.WriteLine("Вектор номер " + (i + 1) + "\n");

Console.Write("Введите размерность вектора: ");

int count\_vec = int.Parse(Console.ReadLine());

if (count\_vec > 0)

{

Console.WriteLine("Введите координаты вектора: ");

mass\_vec[i] = new ArrayVector(count\_vec);

for (int j = 0; j < count\_vec; j++)

{

mass\_vec[i][j] = int.Parse(Console.ReadLine());

}

Console.WriteLine();

break;

}

else

{

throw new Exception("Количество должно быть больше нуля!");

}

}

}

catch (Exception)

{

Console.WriteLine("Ошибка ввода! Повторите попытку!");

}

break;

case "2":

try

{

while (true)

{

Console.WriteLine();

Console.WriteLine("Вектор номер " + (i + 1) + "\n");

Console.Write("Введите размерность вектора: ");

int count\_vec = int.Parse(Console.ReadLine());

if (count\_vec > 0)

{

Console.WriteLine("Введите координаты вектора: ");

mass\_vec[i] = new LinkedListVector(count\_vec);

for (int j = 0; j < count\_vec; j++)

{

mass\_vec[i][j] = int.Parse(Console.ReadLine());

}

Console.WriteLine();

break;

}

else

{

throw new Exception("Количество координат должно быть больше нуля!");

}

}

}

catch (Exception)

{

Console.WriteLine("Ошибка ввода! Повторите попытку!");

}

break;

case "0":

return;

default:

Console.WriteLine("Ошибка ввода! Повторите попытку!");

break;

}

break;

}

}

try

{

IVectorable max = mass\_vec[0];

IVectorable min = mass\_vec[0];

int cheking = 0;

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

{

cheking = min.CompareTo(mass\_vec[i]);

if (cheking < 0)

{

min = mass\_vec[i];

}

}

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

{

if (mass\_vec[i].Length > 0)

{

cheking = max.CompareTo(mass\_vec[i]);

if (cheking > 0)

{

max = mass\_vec[i];

}

}

}

Array.Sort(mass\_vec, new Sravnenie());

Console.WriteLine("Отсортированный список векторов по модулю (возр): ");

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

{

Console.WriteLine("Вектор № " + (i + 1) + " ");

Console.WriteLine(mass\_vec[i].ToString() + "\n");

Console.Write("Модуль вектора: ");

Console.WriteLine(mass\_vec[i].GetNorm() + "\n");

}

Console.WriteLine("Вектора с максимальным кол-вом координат: ");

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

{

if (mass\_vec[i].Length == max.Length)

{

Console.WriteLine(mass\_vec[i].ToString() + "\n");

}

}

Console.WriteLine("Вектора с минимальным кол-вом координат: ");

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

{

if (mass\_vec[i].Length == min.Length)

{

Console.WriteLine(mass\_vec[i].ToString() + "\n");

}

}

Console.WriteLine();

while (true)

{

Console.WriteLine("Выберите, что хотите сделать:\n1 - Сравнить на равенство\n2 - Создать клон\n0 - Выход");

switch (Console.ReadLine())

{

case "1":

if (mass\_vec.Length != 1)

{

Console.Write("Введите номера векторов, которые нужно сравнить: ");

try

{

Console.Write("Первый: ");

int nomer1 = int.Parse(Console.ReadLine());

nomer1--;

Console.Write("Второй: ");

int nomer2 = int.Parse(Console.ReadLine());

nomer2--;

if (nomer1 != nomer2)

{

if (mass\_vec[nomer1].Equals(mass\_vec[nomer2]))

{

Console.WriteLine("Векторы равны!");

}

else

{

Console.WriteLine("Вектора не равны!");

}

}

else

{

Console.WriteLine("Вы получили один и тот же вектор!");

}

}

catch

{

throw new Exception("Ошибка ввода! Повторите попытку!");

}

}

break;

case "2":

try

{

Console.Write("Введите номер вектора, для которого нужно создать клон: ");

int nomer = int.Parse(Console.ReadLine());

nomer--;

IVectorable vec = (IVectorable)mass\_vec[nomer].Clone();

Console.Write("Исходный вектор: ");

Console.WriteLine(mass\_vec[nomer].ToString() + "\n");

Console.Write("Клон исходного вектора: ");

Console.WriteLine(mass\_vec[nomer].ToString() + "\n");

Console.Write("Введите число, на которое хотите заменить первую координату: ");

int num = int.Parse(Console.ReadLine());

vec[0] = num;

Console.WriteLine();

Console.Write("Клонированный вектор: " + vec.ToString());

Console.WriteLine();

}

catch (Exception)

{

Console.WriteLine();

Console.WriteLine("Такого номера вектора нет! Повторите попытку!");

}

break;

case "0":

return;

default:

Console.WriteLine("Ошибка ввода! Повторите попытку!");

break;

}

}

}

catch (Exception)

{

Console.WriteLine("Ошибка ввода! Повторите попытку!");

}

}

else

{

throw new Exception("Ошибка ввода! Повторите попытку!");

}

}

static void Task5()

{

int i = 1;

while (i != 0)

{

try

{

Console.Write("Введите длину списка: ");

int length = int.Parse(Console.ReadLine());

LinkedListVector vector = new LinkedListVector(length);

Console.WriteLine();

Console.WriteLine("Введите элементы списка: ");

for (LinkedListVector.Node node = vector.Head; node != null; node = node.Link)

{

node.value = Convert.ToInt32(Console.ReadLine());

}

double mod1 = vector.GetNorm();

Console.WriteLine("Модуль списка: " + mod1);

Console.WriteLine();

Console.Write("Выберите действие:\n1 - Добавление в список(начало/конец)\n2 - Удаление из списка(начало/конец)\n0 - Выход");

switch (Console.ReadLine())

{

case "1":

Console.Write("Введите элемент, который нужно добавить в начало: ");

int number1 = int.Parse(Console.ReadLine());

vector.AddElenentInBegin(number1);

Console.WriteLine();

Console.Write("Введите элемент, который нужно добавить в конец: ");

int number2 = int.Parse(Console.ReadLine());

Console.WriteLine();

vector.AddElementInEnd(number2);

break;

case "2":

Console.WriteLine("Удаление элемента в начале списка:");

vector.DeleteElementInBegin();

Console.WriteLine();

Console.WriteLine("Удаление элемента в конце списка:");

vector.DeleteElementInEnd();

Console.WriteLine();

break;

case "0":

return;

default:

Console.WriteLine("Ошибка ввода! Повторите попытку!");

break;

}

}

catch (Exception)

{

Console.WriteLine();

Console.WriteLine("Ошибка ввода! Повторите попытку!");

}

break;

}

}

}

}

using System;

namespace ЛР\_05

{

class ArrayVector : IVectorable, ICloneable, IComparable

{

int[] arr;

public ArrayVector(int len) // конструктор с параметрами

{

arr = new int[len];

}

public ArrayVector() // конструктор без парам (по умолчанию 5)

{

arr = new int[5];

}

public void SetElement(int index, int digit) // метод установки элемента массива

{

arr[index] = digit;

}

public int Length

{

get

{

return arr.Length;

}

}

public int GetElement(int index)

{

return arr[index];

}

public int this[int index] // индексатор для организации доступа к элементам массива, выбрасывающий искл. при некорректном индексе

{

get

{

try

{

return GetElement(index);

}

catch (IndexOutOfRangeException)

{

Console.WriteLine("Ошибка! Выход за границу массива!");

return 0;

}

}

set

{

try

{

SetElement(index, value);

}

catch (IndexOutOfRangeException) // объект исключения

{

Console.WriteLine("Ошибка! Выход за границу массива!");

}

}

}

public double GetNorm() // модуль вектора

{

double modul = 0;

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

{

modul += Math.Pow(arr[i], 2);

}

modul = Math.Sqrt(modul);

return modul;

}

public override string ToString()

{

string str = Length + " ";

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

{

if (arr[i] != 0)

{

str += arr[i] + " ";

}

}

return str;

}

public override bool Equals(object obj) // переопреление метода Equals класса Object

{

int count = 0;

if (obj is IVectorable vectorable)

{

if (vectorable.Length == arr.Length) // если равны по длине

{

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

{

if (vectorable[i] == arr[i]) // если равны покоординатно

{

count += 1;

}

}

if (arr.Length == count) // если равны по кол-ву координат

{

return true;

}

else

{

return false;

}

}

else

{

return false;

}

}

else

{

return false;

}

}

public override int GetHashCode() // получение хэш-кода

{

return base.GetHashCode();

}

public object Clone() // клонирование объектов

{

ArrayVector array = new ArrayVector(Length);

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

{

array[i] = this[i];

}

return array;

}

public int CompareTo(object obj) // метод сравнения по числу координат

{

if (obj is IVectorable vectorable)

{

if (Length == arr.Length)

{

return 0;

}

else if (Length > arr.Length)

{

return 1;

}

else

{

return -1;

}

}

else

{

throw new Exception("Невозможно сравнить!");

}

}

}

}

using System;

namespace ЛР\_05

{

class Vectors

{

public static ArrayVector Sum(IVectorable vec1, IVectorable vec2)

{

if (vec1.Length == vec2.Length)

{

ArrayVector result = new ArrayVector();

int sum;

Console.Write("Сумма векторов: ");

for (int i = 0; i < vec1.Length; i++)

{

sum = vec1[i] + vec2[i];

result[i] = sum;

Console.Write(result[i] + " ");

}

return result;

}

else

{

throw new FormatException("Количество координат должно быть одинаково!");

}

}

public static double Scalar(IVectorable vec1, IVectorable vec2)

{

double result = 0;

if (vec1.Length == vec2.Length)

{

Console.Write("Скалярное произведние: ");

for (int i = 0; i < vec1.Length; i++)

{

result += vec1[i] \* vec2[i];

}

}

else

{

throw new FormatException("Количество координат должно быть одинаково!");

}

return result;

}

public static double GetNorm(IVectorable vec)

{

double len = 0;

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

{

len += Math.Pow(vec[i], 2);

}

len = Math.Sqrt(len);

return len;

}

}

}

using System;

namespace ЛР\_05

{

interface IVectorable : IComparable, ICloneable

{

int this[int num] { get; set; } // индексатор

int Length { get; } // свойство для чтения

double GetNorm(); // получение модуля вектора

}

}

using System;

using System.Collections.Generic;

namespace ЛР\_05

{

class LinkedListVector : IVectorable, IComparable, ICloneable

{

public Node Head;

public class Node

{

public Node link; // null

public int value; // целое число

public Node() // констуктор по умолчанию

{

value = 0;

link = null;

}

public Node Link // свойство

{

get { return link; }

set { link = value; }

}

}

public int Length // свойство для чтения элентов

{

get

{

int i = 0;

Node node = Head;

while (node != null)

{

node = node.link;

i++;

}

return i;

}

}

public LinkedListVector(int len) // конструктор с парамаметром

{

try

{

if (len != 0)

{

Head = new Node();

Node node = Head;

for (int i = 1; i < len; i++)

{

node.Link = new Node();

node = node.Link;

}

}

}

catch (Exception)

{

Console.WriteLine("Длина не может равняться 0!");

}

}

public LinkedListVector() : this(5) { }

public Node GetElement(int num)

{

Node node = Head;

if (num > -1 && num < Length)

{

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

{

node = node.Link; //добавление ссылки

}

}

else

{

throw new IndexOutOfRangeException(); // принудительный выброс объекта исключения

}

return node;

}

public double GetNorm() // длина вектора (модуль)

{

double result = 0;

Node node = Head;

while (node != null)

{

result += Math.Pow(node.value, 2);

node = node.Link;

}

result = Math.Sqrt(result);

return result;

}

public int this[int num] // индексатор

{

get { return GetElement(num).value; }

set { GetElement(num).value = value; }

}

public override string ToString() // переопределение метода ToString класса Object

{

string str = Length + " ";

Node node = Head;

while (node != null)

{

str += node.value + " ";

node = node.Link;

}

return str;

}

public override bool Equals(object obj) // переопреление метода Equals класса Object

{

int count = 0;

if (obj is IVectorable vectorable)

{

if (vectorable.Length == Length) // если равны по длине

{

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

{

if (vectorable[i] == GetElement(i).value) // если равны покоординатно

{

count += 1;

}

}

if (Length == count) // если равны по кол-ву координат

{

return true;

}

else

{

return false;

}

}

else

{

return false;

}

}

else

{

return false;

}

}

public override int GetHashCode()

{

return base.GetHashCode();

}

public object Clone() // клонирование объектов

{

LinkedListVector list = new LinkedListVector(Length);

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

{

list[i] = this[i];

}

return list;

}

public int CompareTo(object obj)

{

if (obj is IVectorable vectorable)

{

if (Length == vectorable.Length)

{

return 0;

}

else if (Length > vectorable.Length)

{

return 1;

}

else

{

return -1;

}

}

else

{

throw new Exception("Невозможно сравнить!");

}

}

// ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ

public void AddElenentInBegin(int value)

{

Node newnode = new Node(); // создание нового объекта

newnode.value = value;

newnode.link = Head;

Head = newnode; // переписваивание главного элемента новым элементом

Node node = Head;

while (node != null)

{

Console.WriteLine(node.value);

node = node.link;

}

}

public void AddElementInEnd(int value)

{

Node newnode = new Node(); // создание нового объекта

newnode.value = value;

Node node = Head;

for (int i = 0; i < Length - 1; i++)

{

node = node.link; // добавление ссылки элементу

}

node.link = newnode; // присваивание ссылки новому элементу

node = Head;

while (node != null)

{

Console.WriteLine(node.value);

node = node.link;

}

}

public void DeleteElementInBegin()

{

Node node = Head;

node = node.link;

Head = node;

while (node != null)

{

Console.WriteLine(node.value);

node = node.link;

}

}

public void DeleteElementInEnd()

{

Node node = Head;

for (int i = 0; i < Length - 2; i++)

{

node = node.link;

}

node.link = null;

node = Head;

while (node != null)

{

Console.WriteLine(node.value);

node = node.link;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

namespace ЛР\_05

{

class Sravnenie : IComparer<IVectorable>

{

public int Compare([AllowNull] IVectorable vec1, [AllowNull] IVectorable vec2)

{

if (vec1.GetNorm() == vec2.GetNorm())

{

return 0;

}

else if (vec1.GetNorm() < vec2.GetNorm())

{

return 1;

}

else

{

return -1;

}

}

}

}

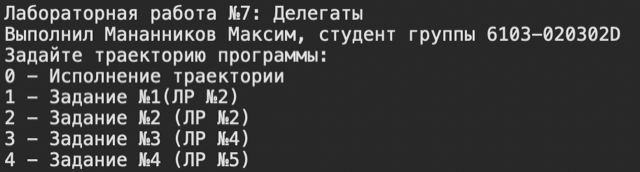


Рисунок 1 - Начало работы программы

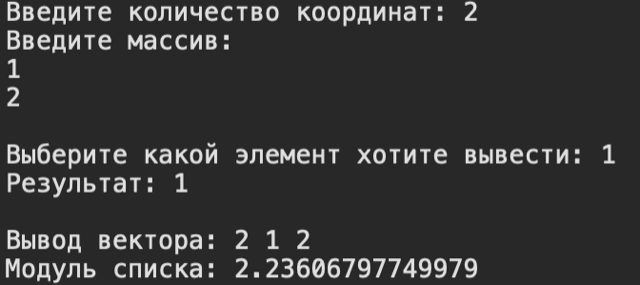


Рисунок 2 - Пример выполнения задания 1

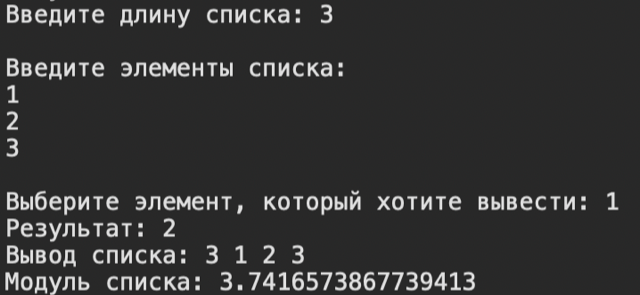


Рисунок 3 - Пример выполнения задания 2

**ВЫВОДЫ**

В лабораторной работе были использованы конструкции языка:

* условия;
* циклы;
* классы;
* конструкторы классов;
* различные методы классов;
* оператор switch;
* оператор try-catch
* делегаты.

**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.
2. Шилдт, Герберт. С# 4.0 : полное руководство / Герберт Шилдт ; [пер. с англ. и ред. И. В. Берштейна]. – Москва [и др.] : Вильямс, 2015. - 1056 с.