

**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**

Лабораторная работа № 03

Основы языка С#: Полиморфизм (наследование, интерфейсы)по дисциплине «Языки программирования и структуры данных»

Выполнил Сидоров A.O., 6103

Проверил Котенева С.Э.

Самара  
2024

ЗАДАНИЕ

Задание 1.

Описать интерфейс «IVectorable», содержащий следующие элементы:

* индексатор для организации доступа к элементам массива/списка;
* свойство для чтения числа координат вектора Length;
* метод получения модуля вектора GetNorm().

Задание 2.

Сделать классы ArrayVector и LinkListVector реализующими интерфейс IVectorable и привести их в соответствие с описанной структурой наследования. Поменять в классе со статическими методами Vectors типы параметров так, чтобы методы работали со ссылками типа интерфейс IVectorable.

Задание 3.

В классах ArrayVector и LinkListVector переопределить унаследованный от класса Object метод ToString(), который преобразует вектор в строку формата «<число координат вектора><пробел><координаты вектора через пробел>». Например, 5 1 2 3 4 5. В дальнейшем, для вывода информации о векторе использовать вызов данного метода.

Задание 4.

Добавить в индексаторы классов ArrayVector и LinkListVector выброс исключительной ситуации (возможно, IndexOutOfRangeException) в случае выхода параметра индексатора за пределы индексирования координат вектора.

Задание 5.

Протестировать работу приложения в классе Program, разработать адекватный интерфейс пользователя, отлавливать и обрабатывать все возможные исключения. Воспользоваться пользовательским интерфейсом из лабораторной работы 2. (Примечание: в меню предусмотреть пункт, позволяющий сложить вектора разных типов)

Задание 6.

Подготовить отчет о работе.

КОД ПРОГРАММЫ

namespace newlab23

{

class Program

{

public static void Main()

{

bool alive = true;

while (alive == true)

{

Console.WriteLine("\nСеместр 2, Лабораторная работа 2, выполнил студент Артемий Сидоров\n");

Console.WriteLine("1 - использование класса ArrayVector2");

Console.WriteLine("2 - использование класса LinkedListVector");

Console.WriteLine("3 - использование класса Vectors");

Console.WriteLine("4 - завершение работы\n");

string menu = Console.ReadLine()!;

switch (menu)

{

case "1":

{

Console.WriteLine("Введите координаты вектора через пробел");

string[] temp = Console.ReadLine().Split(" ");

ArrayVector vector = new ArrayVector(temp.Length);

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

{

vector[i + 1] = int.Parse(temp[i]);

}

try

{

Console.Write("Ведите индекс элемента вектора, который хотите вывести: ");

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

Console.WriteLine("Элемент с индексом " + j + ": " + vector[j]);

}

catch

{

Console.WriteLine("значение индекса выходит за предел");

}

double d = vector.GetNorm();

Console.WriteLine("Модуль вектора: " + d);

break;

}

case "2":

{

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

string[] temp = Console.ReadLine().Split(" ");

var list = new LinkedListVector(temp.Length);

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

{

list[i + 1] = int.Parse(temp[i]);

}

Console.WriteLine("Список: " + list);

bool alive2 = true;

while (alive2 == true)

{

Console.WriteLine("0 - вернуться в предыдущее меню");

Console.WriteLine("1 - вывод списка и размерности");

Console.WriteLine("2 - вставка в начало");

Console.WriteLine("3 - удаление из начала");

Console.WriteLine("4 - вставка в конец");

Console.WriteLine("5 - удаление из конца");

Console.WriteLine("6 - вставка по индексу");

Console.WriteLine("7 - удаление по индексу");

Console.WriteLine("8 - модуль списка");

Console.WriteLine("9 - получение элемента по индексу");

Console.WriteLine("10 - изменение элемента по индексу");

string menu2 = Console.ReadLine()!;

try

{

switch (menu2)

{

case "0":

{

alive2 = false;

break;

}

case "1":

{

Console.WriteLine("Вывод списка и размерности:");

Console.WriteLine("Список: " + list);

Console.WriteLine("Размерность: " + list.Length);

break;

}

case "2":

{

Console.WriteLine("Вставка в начало");

Console.Write("Введите число, которое вставить: ");

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

list.InsertStart(num);

Console.WriteLine("Список: " + list);

Console.WriteLine("Размерность списка: " + list.Length);

break;

}

case "3":

{

Console.WriteLine("Удаление из начала");

list.DeleteStart();

Console.WriteLine("Список: " + list);

Console.WriteLine("Размерность списка: " + list.Length);

break;

}

case "4":

{

Console.WriteLine("Вставка в конец");

Console.Write("Введите число, которое вставить: ");

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

list.InsertEnd(num);

Console.WriteLine("Список: " + list);

Console.WriteLine("Размерность списка: " + list.Length);

break;

}

case "5":

{

Console.WriteLine("Удаление с конца");

list.DeleteEnd();

Console.WriteLine("Список: " + list);

Console.WriteLine("Размерность списка: " + list.Length);

break;

}

case "6":

{

Console.WriteLine("Вставка по индексу");

Console.Write("Введите индекс: ");

var index = int.Parse(Console.ReadLine()!);

Console.WriteLine("Найден элемент с индексом " + index + ": " + list[index]);

Console.Write("Введите значение (целое число): ");

var value = int.Parse(Console.ReadLine()!);

list.InsertByIndex(index, value);

Console.WriteLine("Вставка произошла успешно, новый список: " + list);

break;

}

case "7":

{

Console.WriteLine("Удаление по индексу");

Console.Write("Введите индекс: ");

var index = int.Parse(Console.ReadLine()!);

list.DeleteByIndex(index);

Console.WriteLine("Удаление произошла успешно, новый список: " + list);

break;

}

case "8":

{

Console.WriteLine("Модуль списка");

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

break;

}

case "9":

{

Console.WriteLine("Получение элемента по индексу");

Console.Write("Введите индекс элемента, который хотите получить: ");

int index = int.Parse(Console.ReadLine()!);

Console.WriteLine("Элемент с индексом " + index + ": " + list[index]);

break;

}

case "10":

{

Console.WriteLine("Изменение элемента по индексу");

Console.WriteLine("Введите индекс элемента, который хотите изменить");

var index = int.Parse(Console.ReadLine()!);

Console.WriteLine("Найден элемент с индексом " + index + ": " + list[index]);

Console.WriteLine("Введите значение (целое число), на которое хотите изменить");

var value = int.Parse(Console.ReadLine()!);

list[index] = value;

Console.WriteLine("Значение изменено, теперь список выглядит так: " + list);

break;

}

default:

{

Console.WriteLine("\nПовторите ввод\n");

break;

}

}

}

catch (IndexOutOfRangeException)

{

Console.WriteLine("Ошибка: индекс вышел за переделы");

}

catch (Exception e)

{

Console.WriteLine("Ошибка: " + e.Message);

}

}

break;

}

case "3":

{

Console.WriteLine("Введите координаты вектора класса ArrayVector через пробел");

string[] temp1 = Console.ReadLine().Split(" ");

ArrayVector vector1 = new ArrayVector(temp1.Length);

for (int i = 0; i < temp1.Length; i++)

{

vector1[i + 1] = int.Parse(temp1[i]);

}

Console.WriteLine("Введите координаты вектора класса LinkedListVector через пробел");

string[] temp2 = Console.ReadLine().Split(" ");

LinkedListVector vector2 = new LinkedListVector(temp2.Length);

for (int i = 0; i < temp2.Length; i++)

{

vector2[i + 1] = int.Parse(temp2[i]);

}

try

{

IVectorable sum = Vectors.Sum(vector1, vector2);

double scalar = Vectors.Scalar(vector1, vector2);

Console.WriteLine("Сумма векторов: " + sum);

Console.WriteLine("Скалярное произведение: " + scalar);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

break;

}

case "4":

{

Console.WriteLine("\nВы завершили работу программы\n");

alive = false;

break;

}

default:

{

Console.WriteLine("\nПовторите ввод\n");

break;

}

}

Console.WriteLine("\n\nНажмите на любую клавишу, чтобы продолжить\n");

Console.ReadKey();

}

}

}

namespace newlab23

{

public class LinkedListVector : IVectorable

{

private Node head;

public LinkedListVector(int length)

{

head = new Node();

var node = head;

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

{

node.next = new Node();

node = node.next;

}

}

public LinkedListVector() : this(5) { }

public int Length

{

get

{

int Length = 0;

Node node = head;

while (node != null)

{

node = node.next;

Length++;

}

return Length;

}

}

public int this[int index]

{

get

{

var node = GetNodeByIndex(index - 1);

return node.value;

}

set

{

var node = GetNodeByIndex(index - 1);

node.value = value;

}

}

public double GetNorm()

{

int sum = 0;

var node = head;

while (node != null)

{

sum += node.value \* node.value;

node = node.next;

}

return Math.Sqrt(sum);

}

public void InsertStart(int value)

{

var node = new Node(value, head);

head = node;

}

public void DeleteStart()

{

if (Length == 0)

{

return;

}

head = head.next;

}

public void InsertEnd(int value)

{

if (head == null)

{

head = new Node(value);

}

else

{

var node = GetNodeByIndex(Length - 1);

node.next = new Node(value);

}

}

public void DeleteEnd()

{

if (head == null)

{

return;

}

if (head.next == null)

{

head = null;

return;

}

var node = GetNodeByIndex(Length - 2);

node.next = null;

}

public void InsertByIndex(int index, int value)

{

if (index == Length)

{

var prevCurrNode = GetNodeByIndex(index - 2);

prevCurrNode.next = new Node(value, prevCurrNode.next);

}

else if (index == 1)

{

InsertStart(value);

}

else

{

var prevCurrNode = GetNodeByIndex(index - 2);

prevCurrNode.next = new Node(value, prevCurrNode.next);

}

}

public void DeleteByIndex(int index)

{

if (index < 1 || index > Length)

{

throw new IndexOutOfRangeException();

}

if (index == Length)

{

DeleteEnd();

}

else if (index == 1)

{

DeleteStart();

}

else

{

var prevCurrNode = GetNodeByIndex(index - 2);

prevCurrNode.next = prevCurrNode.next.next;

}

}

public override string ToString()

{

string res = Length + "";

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

{

res += " " + this[i + 1];

}

return res;

}

private Node GetNodeByIndex(int index)

{

if (index < 0 || index >= Length)

{

throw new IndexOutOfRangeException();

}

var node = head;

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

{

node = node?.next;

}

return node;

}

private class Node

{

public int value;

public Node next;

public Node(int nodeValue = 0, Node? nextNode = null)

{

value = nodeValue;

next = nextNode;

}

}

}

}

namespace newlab23

{

internal class ArrayVector : IVectorable

{

public int[] vector;

public ArrayVector(int size)

{

if (size < 1)

{

throw new Exception("Размерность вектора должна быть больше нуля!");

}

vector = new int[size];

}

public ArrayVector() // конструктор без параметра

{

vector = new int[5];

}

public int this[int i] // индексатор

{

get

{

return vector[i - 1];

}

set

{

vector[i - 1] = value;

}

}

public int Length

{

get

{

return vector.Length;

}

}

public double GetNorm() // метод получения модуля вектора

{

int temp = 0;

foreach (int i in vector)

{

temp += i \* i;

}

return Math.Sqrt(temp);

}

public override string ToString()

{

string res = Length + "";

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

{

res += " " + this[i + 1];

}

return res;

}

}

}

namespace newlab23

{

internal class Vectors

{

public static IVectorable Sum(IVectorable vector1, IVectorable vector2)

{

if (vector1.Length != vector2.Length)

{

throw new Exception("Размерность векторов не совпадает!");

}

IVectorable sum = new ArrayVector(vector1.Length);

for (int i = 1; i < vector1.Length + 1; i++)

{

sum[i] = vector1[i] + vector2[i];

}

return sum;

}

public static double Scalar(IVectorable vector1, IVectorable vector2)

{

if (vector1.Length != vector2.Length)

{

throw new FormatException("Размерность векторов не совпадает!");

}

double scalar = 0;

for (int i = 1; i < vector1.Length + 1; i++)

{

scalar += vector1[i] \* vector2[i];

}

return scalar;

}

public static double GetNormSt(IVectorable vector)

{

return vector.GetNorm();

}

}

}

namespace newlab23

{

public interface Vectorable

{

int this[int index] { get; set; }

int Length { get; }

double GetNorm();

}

}

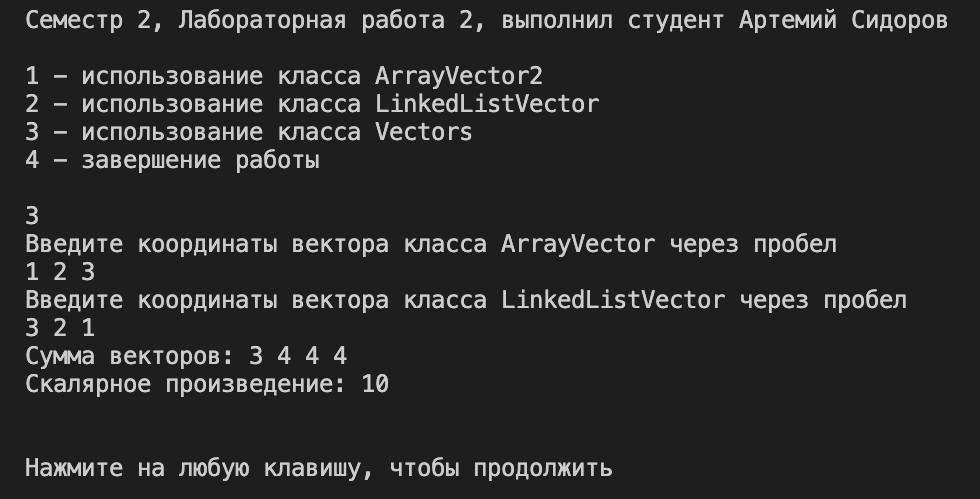
}

Рисунок 1 – Пример работы программы

**ВЫВОДЫ**

В лабораторной работе были использованы возможности языка C#:

* ветвления if;
* цикл с параметром for;
* цикл foreach;
* методы;
* классы;
* индексатор.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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