****

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

Институт (факультет) Информатики

Кафедра Программных систем

ОТЧЁТ

по лабораторной работе №06

Программы с использованием цикловпо дисциплине «Основы программирования»

Выполнил Боряков Никита Сергеевич, группа 6102-020302D

Проверил Котенёва Светлана Эдуардовна

Самара  
2021

ЗАДАНИЕ

Задание 1

В класс Vectors добавить следующие методы:

* записи вектора в байтовый поток:

void OutputVector(Vector v, Stream out);

* чтения вектора из байтового потока:

Vector InputVector(Stream in).

Записанный вектор должен представлять собой последовательность чисел, первым из которых является размерность вектора, а остальные числа являются значениями координат вектора.

Проверить возможности методов в классе Program, в качестве байтового потока используя файловый поток (создать файл данных в текущей папке).

Задание 2

В класс Vectors добавить следующие методы:

* записи вектора в байтовый поток:

void OutputVector(Vector v, Stream out);

* чтения вектора из байтового потока:

Vector InputVector(Stream in).

Записанный вектор должен представлять собой последовательность чисел, первым из которых является размерность вектора, а остальные числа являются значениями координат вектора.

Проверить возможности методов в классе Program, в качестве байтового потока используя файловый поток (создать файл данных в текущей папке).

Задание 3

Модифицировать классы ArrayVector и LinkListVector таким образом, чтобы они были сериализуемыми.

Продемонстрировать возможности сериализации в классе Program, записав в файл объект, затем считав и сравнив с исходным, для чего вывести содержимое обоих объектов на экран, можно также использовать метод Equals(), унаследованный от класса Object.

Задание 4

Протестировать работу приложения в классе Program, доработав интерфейс пользователя из лабораторной работы 4.

Задание 5

Подготовить отчет о работе.

КОД ПРОГРАММЫ

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

namespace lab\_work\_0206

{

interface IVectorable : IComparable, ICloneable

{

int this[int i]

{

get;

set;

}

int Length { get; }

double GetNorm();

}

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Лабораторная работа №5.\n" +

"Выполнил студент группы 6102 Боряков Никита\n ");

Console.WriteLine("Выберете режим работы\n" +

"1 - Array Vector\n" +

"2 - Vectors\n" +

"3 - LinkedListVector\n" +

"4 - Массив типа интерфейс\n" +

"5 - Выход из программы\n");

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

while (m != 5)

{

switch (m)

{

case 1:

{

Console.Clear();

Console.Write("Введите кол-во пространств:");

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

ArrayVector v = new ArrayVector(count);

v.EnterVector();

Console.Write("Введите кол-во пространств:");

int count1 = int.Parse(Console.ReadLine());

LinkedListVector v1 = new LinkedListVector(count1);

v1.AddElement(count1);

Console.WriteLine(v.Equals(v1));

Console.Clear();

Console.WriteLine(v.Equals(v1));

Console.WriteLine(v.ToString());

Console.WriteLine(v.coordinates);

Console.WriteLine(v.GetHashCode());

Console.WriteLine(" 1 - \"Чтение массива\"\n " +

"2 - \"Получение модуля вектора\"\n " +

"3 - \"Выход из программы\"\n ");

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

while (k != 3)

{

switch (k)

{

case 1:

{

v.ToString();

Console.ReadLine();

break;

}

case 2:

{

Console.WriteLine(v.ToString());

Console.WriteLine("Длина вектора = " + v.GetNorm());

Console.ReadLine();

break;

}

}

Console.Clear();

Console.WriteLine(" 1 - \"Чтение элемента массива\"\n " +

"2 - \"Получение модуля вектора\"\n " +

"3 - \"Выход из программы\"\n ");

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

}

break;

}

case 2:

{

Console.Clear();

Console.Write("Введите кол-во пространств первого вектора:");

int count1 = int.Parse(Console.ReadLine());

ArrayVector v1 = new ArrayVector(count1);

v1.EnterVector();

Console.Write("Введите кол-во пространств второго вектора:");

int count2 = int.Parse(Console.ReadLine());

LinkedListVector v2 = new LinkedListVector(count1);

v2.AddElement(count2);

Console.Clear();

Console.WriteLine(v1.ToString());

Console.WriteLine(v2.ToString());

Console.WriteLine(" 1 - \"Сложение векторов\"\n " +

"2 - \"Скалярное произведение векторов\"\n " +

"3 - \"Получение модуля вектора 1\"\n " +

"4 - \"Получение модуля вектора 2\"\n " +

"5 - \"Выход в главное меню\"\n ");

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

while (k != 7)

{

switch (k)

{

case 1:

{

if (Vectors.Сontrol(v1, v2))

{

Console.WriteLine("Сумма = " + Vectors.Sum(v1, v2));

}

else

{

Console.WriteLine("Данные векторы нельзя сложить!");

}

Console.ReadLine();

break;

}

case 2:

{

if (Vectors.Сontrol(v1, v2))

{

Console.WriteLine("Произведение = " + Vectors.Scalar(v1, v2));

}

else

{

Console.WriteLine("Данные векторы нельзя перемножить!");

}

Console.ReadLine();

break;

}

case 3:

{

Console.WriteLine("Модуль вектора 1 = " + Vectors.GetNorm(v1));

Console.ReadLine();

break;

}

case 4:

{

Console.WriteLine("Модуль вектора 2 = " + Vectors.GetNorm(v2));

Console.ReadLine();

break;

}

}

Console.Clear();

Console.WriteLine(" 1 - \"Сложение векторов\"\n " +

"2 - \"Скалярное произведение векторов\"\n " +

"3 - \"Получение модуля вектора 1\"\n " +

"4 - \"Получение модуля вектора 2\"\n " +

"5 - \"Выход в главное меню\"\n ");

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

}

break;

}

case 3:

{

Console.Write("Введите кол-во элементов списка: ");

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

LinkedListVector lis = new LinkedListVector(x);

Console.WriteLine("Ваш список: {0} ", lis);

Console.WriteLine(" 1 - \"Метод получения элемента списка по индексу\"\n " +

"2 - \"Метод получения модуля\"\n " +

"3 - \"Метод удаления элемента в конце\"\n " +

"4 - \"Метод добавления элемента в конец\"\n " +

"5 - \"Метод удаления элемента в начале\"\n " +

"6 - \"Метод добавления элемента в начало\"\n " +

"7 - \"Метод удаления элемента по индексу\"\n " +

"8 - \"Метод добавления элемента по индексу\"\n " +

"9 - \"Выход в главное меню\"\n ");

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

while (k != 9)

{

switch (k)

{

case 1:

{

Console.WriteLine("Введите индекс элемента массива: ");

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

Console.WriteLine("Элемент: {0}", lis[ind]);

Console.ReadLine();

break;

}

case 2:

{

Console.WriteLine("Длина вектора: {0} ", lis.GetNorm());

Console.ReadLine();

break;

}

case 3:

{

lis.DeletElementEnd();

Console.WriteLine(lis);

Console.ReadLine();

break;

}

case 4:

{

lis.AddElementEnd();

Console.WriteLine(lis);

Console.ReadLine();

break;

}

case 5:

{

lis.DeleteElementStart();

Console.WriteLine(lis);

Console.ReadLine();

break;

}

case 6:

{

lis.AddElementStart();

Console.WriteLine(lis);

Console.ReadLine();

break;

}

case 7:

{

lis.DeleteInd();

Console.WriteLine(lis);

Console.ReadLine();

break;

}

case 8:

{

lis.AddInd();

Console.WriteLine(lis);

Console.ReadLine();

break;

}

}

Console.Clear();

Console.WriteLine(" 1 - \"Метод получения элемента списка по индексу\"\n " +

"2 - \"Метод получения модуля\"\n " +

"3 - \"Метод удаления элемента в конце\"\n " +

"4 - \"Метод добавления элемента в конец\"\n " +

"5 - \"Метод удаления элемента в начале\"\n " +

"6 - \"Метод добавления элемента в начало\"\n " +

"7 - \"Метод удаления элемента по индексу\"\n " +

"8 - \"Метод добавления элемента по индексу\"\n " +

"9 - \"Выход в главное меню\"\n ");

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

}

break;

}

case 4:

{

Console.Write("\nВведите количество векторов: ");

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

IVectorable[] arr = new IVectorable[length];

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

{

Console.WriteLine("\n1 - тип ArrayVector\n" +

"2 - тип LinkedListVector\n");

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

switch (v)

{

case 1:

{

Console.Clear();

Console.Write("Введите кол-во пространств {0}ого вектора: ", i + 1);

int count1 = int.Parse(Console.ReadLine());

arr[i] = new ArrayVector(count1);

Console.Write("Введите координаты:\n\n");

for (int j = 0; j < count1; j++)

{

Console.Write("Значение {0}ого элемента: ", j + 1);

arr[i][j] = int.Parse(Console.ReadLine());

}

break;

}

case 2:

{

Console.Write("Введите кол-во пространств {0}ого вектора: ", i + 1);

int count2 = int.Parse(Console.ReadLine());

arr[i] = new LinkedListVector(count2);

for (int j = 0; j < count2; j++)

{

Console.Write("Значение {0}ого элемента: ", j + 1);

arr[i][j] = int.Parse(Console.ReadLine());

}

// Console.WriteLine(v2.ToString());

break;

}

}

}

Console.WriteLine("\nВыберете режим работы: ");

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

while (n != 7)

{

switch (n)

{

case 1:

{

IVectorable max = arr[0];

IVectorable min = arr[0];

int value;

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

{

value = min.CompareTo(arr[i]);

if (value < 0)

{

min = arr[i];

}

}

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

{

value = max.CompareTo(arr[i]);

if (value > 0)

{

max = arr[i];

}

}

Console.WriteLine("Сортировка массивов векторов по возрастанию");

Array.Sort(arr, new Additional());

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

{

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

}

Console.WriteLine("Минимальное количество координат в векторе");

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

{

if (arr[i].Length == min.Length)

{

Console.WriteLine(arr[i].ToString());

Console.WriteLine();

}

}

Console.WriteLine("Максимальное количество координат в векторе");

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

{

if (arr[i].Length == max.Length)

{

Console.WriteLine(arr[i].ToString());

Console.WriteLine();

}

}

Console.WriteLine("Введите индекс векторов");

Console.Write("Первый индекс: ");

int index1 = int.Parse(Console.ReadLine());

Console.Write("Второй индекс: ");

int index2 = int.Parse(Console.ReadLine());

if (index1 != index2)

{

if (arr[index1].Equals(arr[index2]))

Console.WriteLine("Данные векторы равны");

else

Console.WriteLine("Данные векторы не равны");

Console.WriteLine();

}

else

{

Console.WriteLine("Введенный вектор один и тот же");

}

Console.WriteLine("Клонирование векторов");

Console.WriteLine("Выберете вектор который вы хотите клонировать");

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

IVectorable vector = (IVectorable)arr[index].Clone();

Console.WriteLine(arr[index].ToString());

Console.WriteLine();

Console.WriteLine(vector.ToString());

Console.WriteLine();

break;

}

case 2:

{

Console.WriteLine("Ввод вектора в байтовый поток\n");

FileStream file = new FileStream("test.dat", FileMode.Append, FileAccess.Write);

Vectors vectors = new Vectors();

vectors.OutputVector(arr, file);

file.Close();

Console.ReadLine();

break;

}

case 3:

{

Console.WriteLine("Чтение вектора из байтового потока\n");

FileStream file = new FileStream("test.dat", FileMode.Open, FileAccess.Read);

Vectors vector = new Vectors();

IVectorable[] \_arr = vector.InputVector(file);

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

{

Console.WriteLine(\_arr[i].ToString());

}

Console.ReadLine();

break;

}

case 4:

{

Console.WriteLine("Ввод вектора в символьный поток\n");

StreamWriter file = new StreamWriter("test.txt");

Vectors vector = new Vectors();

vector.WriteVector(arr, file);

file.Close();

Console.ReadLine();

break;

}

case 5:

{

StreamReader file = new StreamReader("test.txt");

Vectors vector = new Vectors();

IVectorable[] \_arr = vector.ReadVector(file);

Console.WriteLine("Чтение вектора из байтового потока\n");

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

{

Console.WriteLine(\_arr[i].ToString());

}

Console.ReadLine();

break;

}

case 6:

{

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

int \_length = Convert.ToInt32(Console.ReadLine());

IVectorable vector1 = new ArrayVector(\_length);

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

for (int i = 0; i < \_length; i++)

{

vector1[i] = Convert.ToInt32(Console.ReadLine());

}

BinaryFormatter binFormat = new BinaryFormatter(); // Использование бинарного формата

using (FileStream stream = new FileStream("lab.bin", FileMode.Append, FileAccess.Write)) // Создания объекта файла

{

binFormat.Serialize(stream, vector1); // Использование метода для сериализации

}

Console.WriteLine("\nВектор успешно сохранен в файл user.bin посредством метода сериализации");

Console.ReadLine();

break;

}

}

Console.Clear();

Console.WriteLine("\nВыберете режим работы: ");

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

}

Console.ReadLine();

break;

}

}

Console.Clear();

Console.WriteLine("Выберете режим работы\n" +

"1 - Array Vector\n" +

"2 - Vectors\n" +

"3 - LinkedListVector\n" +

"4 - Массив типа интерфейс\n" +

"5 - Выход из программы\n");

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

}

}

}

}}

Изображение выглядит как текст

Автоматически созданное описание Рисунок 1 – Главное меню программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Задания

ВЫВОДЫ

В лабораторной работе были использованы конструкции языка:

* ветвление;
* конструктор;
* свойства;
* форматированный вывод информации на консоль;
* цикл с условием;
* switch;
* интерфейсы;
* класс FileStream;
* класс StreamWriter;
* класс StreamReader;
* класс TextWriter;
* класс TextReader;
* класс Stream;
* классы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОНИКОВ

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.