МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

**«Вологодский государственный университет»**

**Институт математики, естественных и компьютерных наук**

**Информатика и вычислительная техника**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

Создание классов на языке C#.

Дисциплина: «Объектно-ориентированное программирование»

Направление подготовки: 09.03.01. Информатика и вычислительная техника

|  |  |
| --- | --- |
| Руководитель | Сорокин А.Н. |
| Выполнили студенты | Пчелкина О.С. |
| Группа, курс | ВМ-41 |
| Дата сдачи | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата защиты | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись преподавателя)* |

Вологда

2022 г.

**Цель работы**: научиться разрабатывать классы на языке C#.

# Вариант 1

1. Создать базовый класс (абстрактный класс или интерфейс) с виртуальными функциями ввода и вывода.

2. Создать производные классы: класс «одномерный массив», класс «матрица».

В производных классах определить: скрытые поля, конструктор по умолчанию, конструктор с параметрами (с проверкой на корректность), метод для установки значений полей (с проверкой на корректность), метод для получения значений полей, методы для ввода и вывода.

3. Разработать демонстрационный класс. В этом классе определить массив или коллекцию ссылок на объекты базового класса, методы для добавления и удаления экземпляров классов, метод для вывода всех элементов.

4. Продемонстрировать работоспособность всех методов классов.

5. Разработать программу, которая демонстрировала бы возможности сериализации. Для проверки введите данные с клавиатуры, сохраните информацию в файл (требуется сохранить несколько объектов разных классов из задания 1), затем прочитайте данные в цикле и выведите информацию на экран (в этом цикле неизвестно количество и порядок объектов в файле).

Код программы:

[DataContract]

public class FileInfo

{

[DataMember]

[JsonProperty("Array")]

public List<int[]> Array { get; set; } = new();

[DataMember]

[JsonProperty("Matrix")]

public List<int[,]> Matrix { get; set; } = new();

}

public class Demonstration

{

public List<IClass> Classes = new();

public void AddClass(IClass classAdd) => Classes.Add(classAdd);

public void RemoveClass(IClass classToRemove) => Classes.Remove(classToRemove);

public FileInfo Output()

{

var fileInfo = new FileInfo();

foreach(var item in Classes)

{

if(item.GetType() == typeof(Array))

fileInfo.Array.Add(((Array)item).\_array);

else fileInfo.Matrix.Add(((Matrix)item).\_matrix);

}

return fileInfo;

}

}

public interface IClass

{

public void Input();

public string Output();

}

public class Array : IClass

{

private int \_size = 5;

public int[] \_array;

public Array() => \_array = new int[\_size];

public Array(int size) => \_size = size;

public void Input() => \_array = Console.ReadLine().Split(' ').Select(s => int.Parse(s)).ToArray();

public string Output()

{

var str = string.Join(" ", \_array);

return str;

}

public void SetSize(int size) => \_size = size;

public int GetSize() => \_size;

}

public class Matrix : IClass

{

private int \_row = 2;

private int \_column = 3;

public int[,] \_matrix;

public Matrix() => \_matrix = new int[\_row, \_column];

public Matrix(int row, int column)

{

\_row = row;

\_column = column;

\_matrix = new int[row, column];

}

public void Input()

{

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

{

for(int j = 0; j < \_column; j++)

\_matrix[i, j] = int.Parse(Console.ReadLine());

}

}

public string Output()

{

var size = \_row \* \_column;

var array = new int[size];

var k = 0;

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

{

for(int j = 0; j < \_column; j++)

{

array[k] = \_matrix[i, j];

Console.Write($"{\_matrix[i, j]}\t");

k++;

}

Console.WriteLine();

}

return string.Join(" ", array);

}

public void SetRow(int row)

{

\_row = row;

}

public void SetColumn(int column)

{

\_column = column;

\_matrix = new int[\_row, \_column];

}

public int GetRow() => \_row;

public int GetColumn() => \_column;

}

Программа, демонстрирующая работу класса:

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Одномерный массив!");

var array = new Array();

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

array.SetSize(int.Parse(Console.ReadLine()));

Console.WriteLine("Введите массив");

array.Input();

Console.WriteLine($"Размерность массива: {array.GetSize()}");

Console.WriteLine($"Массив: {array.Output()}");

Console.WriteLine();

var array2 = new Array();

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

array2.SetSize(int.Parse(Console.ReadLine()));

Console.WriteLine("Введите массив");

array2.Input();

Console.WriteLine($"Размерность массива: {array2.GetSize()}");

Console.WriteLine($"Массив: {array2.Output()}");

Console.WriteLine();

Console.WriteLine("Матрица!");

var matrix = new Matrix();

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

matrix.SetRow(int.Parse(Console.ReadLine()));

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

matrix.SetColumn(int.Parse(Console.ReadLine()));

Console.WriteLine("Введите матрицу");

matrix.Input();

Console.WriteLine($"Количество строк: {matrix.GetRow()}");

Console.WriteLine($"Количество столбцов: {matrix.GetColumn()}");

Console.WriteLine($"Матрица");

matrix.Output();

Console.WriteLine();

Console.WriteLine("Матрица!");

var matrix2 = new Matrix();

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

matrix2.SetRow(int.Parse(Console.ReadLine()));

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

matrix2.SetColumn(int.Parse(Console.ReadLine()));

Console.WriteLine("Введите матрицу");

matrix2.Input();

Console.WriteLine($"Количество строк: {matrix2.GetRow()}");

Console.WriteLine($"Количество столбцов: {matrix2.GetColumn()}");

Console.WriteLine($"Матрица");

matrix2.Output();

Console.WriteLine();

Console.WriteLine($"Добавление экземпляров классов");

var demostration = new Demonstration();

demostration.AddClass(array);

demostration.AddClass(matrix);

demostration.AddClass(array2);

demostration.AddClass(matrix2);

var path = @"C:\users\olpch\lab6.txt";

var settings = JsonConvert.SerializeObject(demostration.Output(), Formatting.Indented);

File.WriteAllText(path, settings);

var fileInfo = JsonConvert.DeserializeObject<FileInfo>(File.ReadAllText(path));

Console.WriteLine($"Данные из файла");

int i = 0;

foreach(var item in fileInfo.Array)

{

i++;

Array array1 = new Array(item.Length);

array1.\_array = item;

Console.WriteLine($"Массив {i}: {array1.Output()}");

}

i = 0;

foreach(var item in fileInfo.Matrix)

{

i++;

Matrix matrix1 = new Matrix(item.GetLength(0), item.GetLength(1));

matrix1.\_matrix = item;

Console.WriteLine($"Матрица {i}: ");

matrix1.Output();

}

Console.WriteLine();

Console.WriteLine($"Удаление экземпляров классов");

demostration.RemoveClass(array);

var settings2 = JsonConvert.SerializeObject(demostration.Output(), Formatting.Indented);

File.WriteAllText(path, settings2);

var fileInfo2 = JsonConvert.DeserializeObject<FileInfo>(File.ReadAllText(path));

Console.WriteLine($"Данные из файла");

i = 0;

foreach(var item in fileInfo2.Array)

{

i++;

Array array1 = new Array(item.Length);

array1.\_array = item;

Console.WriteLine($"Массив {i}: {array1.Output()}");

}

i = 0;

foreach(var item in fileInfo2.Matrix)

{

i++;

Matrix matrix1 = new Matrix(item.GetLength(0), item.GetLength(1));

matrix1.\_matrix = item;

Console.WriteLine($"Матрица {i}: ");

matrix1.Output();

}

Console.WriteLine();

}

}

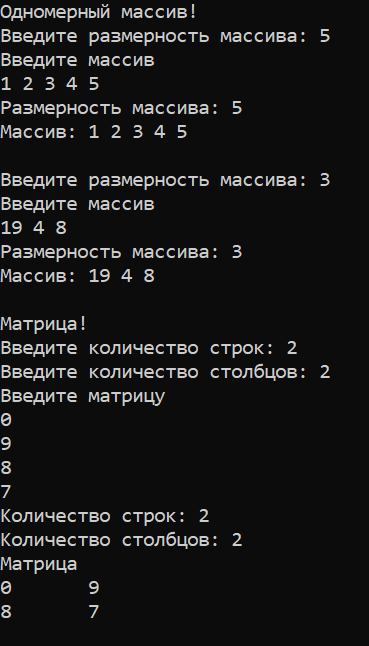
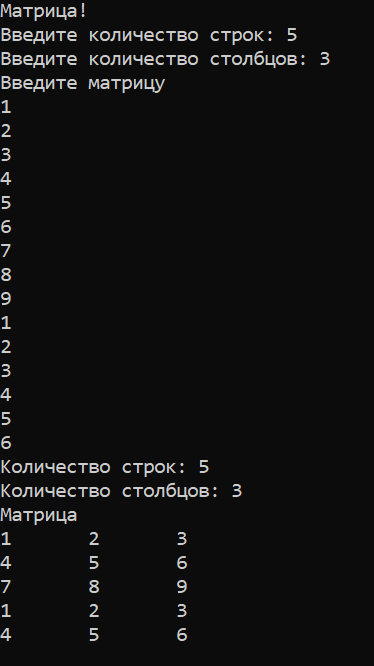
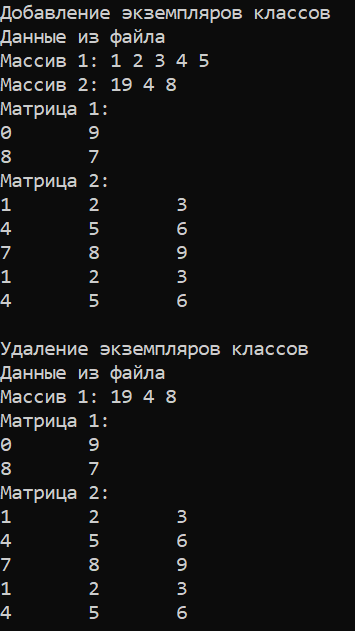
  

Рис. 1 – Работа программы

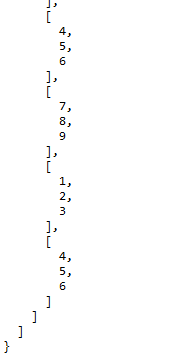
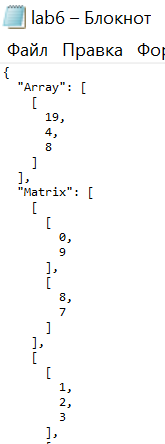


Рис. 2 – Данные файла

# Контрольные вопросы

1. Опишите элементы классов, специфичные для языка C#.

*В С# определены несколько специфических разновидностей членов класса. Это - переменные экземпляра, статические переменные, константы, методы, конструкторы, деструкторы, индексаторы, события, операторы и свойства.*

2. Что понимается под термином «сериализация»? Что такое глубокая и поверхностная сериализация?

*Сериализация — процесс перевода структуры данных в последовательность байтов.*

*Сериализация может быть поверхностной, когда сериализуется на одном шаге единственный объект, и глубокой, когда, начиная с корневого объекта, сериализуется совокупность объектов, связанных взаимными ссылками (граф объектов).*

**Вывод:** в ходе лабораторной работы ознакомились с основными принципами ООП, научились создавать классы, изучили работу конструкторов, деструкторов, перегрузки операторов на языке C#. Разработали классы на языке С# и программу для демонстрации работы класса.