Белорусско-Российский университет

Кафедра ПОИТ

Дисциплина ООПП

Отчет по лабораторной работе №3

«ПЕРЕГРУЗКА МЕТОДОВ И ОПЕРАЦИЙ.»

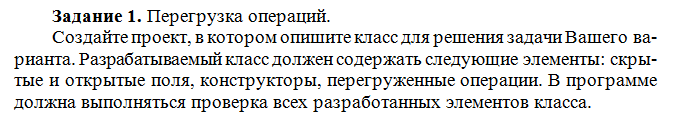
Выполнил студент группы АСОИ-181

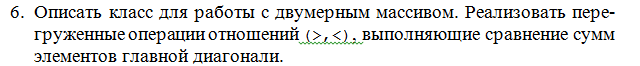
Самусев Д.А.

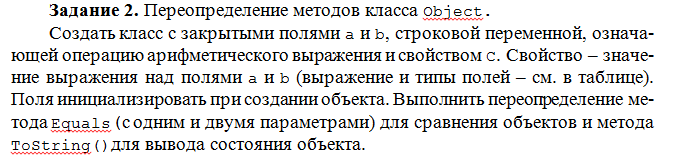
Проверил \_\_\_\_\_\_ Горбатенко Н.Н.

Могилёв 2020г

Цель работы: освоить приемы создания операций класса и их использование.





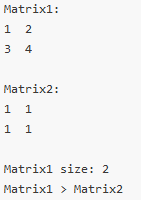




Код программы :

using System;  
  
namespace Lab3  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 var matrix1 = new Matrix();  
   
 double[,] array = new double[2,2];  
 array[0, 0] = 1;  
 array[0, 1] = 1;  
 array[1, 0] = 1;  
 array[1, 1] = 1;  
 var matrix2 = new Matrix(array);   
  
 string matrix1AsString = matrix1.GetMatrixAsString();  
 string matrix2AsString = matrix2.GetMatrixAsString();  
 Console.WriteLine($"Matrix1: \n{matrix1AsString}");  
 Console.WriteLine($"Matrix2: \n{matrix2AsString}");  
  
 Console.WriteLine($"Matrix1 size: {matrix1.Size}");  
   
 var result = matrix1 > matrix2;  
 string message = result ? "Matrix1 > Matrix2" : "Matrix1 > Matrix1";   
   
 Console.WriteLine(message);  
   
 Console.ReadKey();  
 }  
 }  
}

namespace Lab3  
{  
 public class Matrix  
 {  
 private double[,] \_matrix;  
  
 public int Size;  
   
 public Matrix()  
 {  
 \_matrix = new double[2, 2];  
 Size = 2;  
 InitializeMatrix();  
 }  
  
 public Matrix(double[,] matrix)  
 {  
 \_matrix = matrix;  
 Size = matrix.GetLength(0);  
 }  
  
 public string GetMatrixAsString()  
 {  
 string result = string.Empty;  
  
 for (var row = 0; row < \_matrix.GetLength(0); row++)  
 {  
 for (var column = 0; column < \_matrix.GetLength(1); column++)  
 {  
 result += $"{\_matrix[row, column]} ";  
 }  
  
 result += "\n";  
 }  
  
 return result;  
 }  
  
 public static bool operator > (Matrix matrix1, Matrix matrix2)  
 {  
 return GetMainDiagonalSum(matrix1.\_matrix) > GetMainDiagonalSum(matrix2.\_matrix);  
 }  
   
 public static bool operator < (Matrix matrix1, Matrix matrix2)  
 {  
 return !(matrix1 > matrix2);  
 }  
  
 private static double GetMainDiagonalSum(double[,] matrix)  
 {  
 double sum = 0;  
   
 for (var index = 0; index < matrix.GetLength(0); index++)  
 {  
 sum += matrix[index, index];  
 }  
  
 return sum;  
 }  
   
 private void InitializeMatrix()  
 {  
 double element = 1;  
  
 for (var row = 0; row < \_matrix.GetLength(0); row++)  
 {  
 for (var column = 0; column < \_matrix.GetLength(1); column++)  
 {  
 \_matrix[row, column] = element;  
 element++;  
 }  
 }  
 }  
 }  
}



namespace Lab3

{

public class ObjectChild

{

private int \_a;

private int \_b;

private string \_operation;

public int C

{

get

{

switch (\_operation)

{

case "+":

return \_a + \_b;

case "-":

return \_a - \_b;

case "\*":

return \_a \* \_b;

}

return 0;

}

}

public ObjectChild()

{

\_a = 1;

\_b = 2;

\_operation = "+";

}

public override bool Equals(object obj)

{

var cast = obj as ObjectChild;

return this.C == cast.C;

}

public override string ToString()

{

return $"a: {\_a}, b: {\_b}, operation: {\_operation}";

}

}

}