**ЗАДАНИЕ 9**

Задание 1. В программе должна выполняться: неявная неоднозначная реализация методов интерфейсов Iy и Iz, вызов функций с явным приведением к типу интерфейса, вызов метода для объекта посредством интерфейсной ссылки.

Листинг программы:

namespace zad1

{

public interface Iy

{

double F0(int w);

double F1(int w);

}

public interface Iz

{

double F0(int w);

double F1(int w);

}

class Program : Iy, Iz

{

public double F0(int w)

{

return w+5;

}

public double F1(int w)

{

return Math.Pow(w, 3);

}

public double IyF0(int w)

{

return ((Iy)this).F0(w);

}

public double IyF1(int w)

{

return ((Iy)this).F1(w);

}

public double IzF0(int w)

{

return ((Iz)this).F0(w);

}

public double IzF1(int w)

{

return ((Iz)this).F1(w);

}

static void Main()

{

Program program = new Program();

int w = 1;

Console.WriteLine("F0 from Iy: " + program.IyF0(w));

Console.WriteLine("F1 from Iy: " + program.IyF1(w));

Console.WriteLine("F0 from Iz: " + program.IzF0(w));

Console.WriteLine("F1 from Iz: " + program.IzF1(w));

}

}

}

Таблица 9.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Задаются программой | F0 from Iy: 6  F1 from Iy: 1  F0 from Iz: 6  F1 from Iz: 1 |

Анализ результатов:

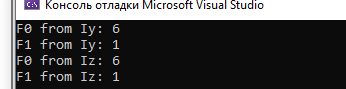


Рисунок 9.1 – Результат работы программы

Задание 2. Выполнить задания, используя для хранения экземпляров разработанных классов массивы. Во всех классах реализовать интерфейс IComparable и перегрузить операции отношения для реализации значимой семантики сравнения объектов по какому-либо полю на усмотрение студента. Описать класс для работы с одномерным массивом строк фиксированной длины. Обеспечить следующие возможности:задание произвольных целых границ индексов при создании объекта; обращение к отдельной строке массива по индексу с контролем выхода за пределы массива; выполнение операций поэлементного сцепления двух массивов с образованием нового массива; вывод на экран элемента массива по заданному индексу и всего массива.

Листинг программы:

namespace zad2

{

internal class Program

{

static void Main(string[] args)

{

Random random = new Random();

int rows1 = random.Next(2, 10);

int cols1 = random.Next(2, 10);

Matrix matrix1 = new Matrix(rows1, cols1);

int rows2 = random.Next(2, 10);

int cols2 = random.Next(2, 10);

Matrix matrix2 = new Matrix(rows2, cols2);

for (int i = 0; i < matrix1.rows; i++)

{

for (int j = 0; j < matrix1.columns; j++)

{

matrix1[i, j] = random.NextDouble() \* 10;

matrix2[i, j] = random.NextDouble() \* 10;

}

}

Console.WriteLine("Матрица matrix1:");

matrix1.Print();

Console.WriteLine("\nМатрица matrix2:");

matrix2.Print();

int newRows = random.Next(5, 15);

int newCols = random.Next(5, 15);

matrix1.Resize(newRows, newCols);

Console.WriteLine($"\nmatrix1 после изменения размера ({newRows}x{newCols}):");

matrix1.Print();

Console.WriteLine("\nПодматрица matrix2 (2-я строка - 1-й столбец):");

matrix2.PrintSubmatrix(1, 1, 0, 0);

}

}

class Matrix : IComparable

{

private double[,] data;

public int rows;

public int columns;

public Matrix(int rows, int columns)

{

if (rows <= 0 || columns <= 0)

{

throw new ArgumentException("Размеры матрицы должны быть больше 0");

}

this.rows = rows;

this.columns = columns;

data = new double[rows, columns];

}

public double this[int row, int column]

{

get

{

if (row < 0 || row >= rows || column < 0 || column >= columns)

{

throw new IndexOutOfRangeException("Индекс выходит за пределы матрицы");

}

return data[row, column];

}

set

{

if (row < 0 || row >= rows || column < 0 || column >= columns)

{

throw new IndexOutOfRangeException("Индекс выходит за пределы матрицы");

}

data[row, column] = value;

}

}

public void Resize(int newRows, int newColumns)

{

if (newRows <= 0 || newColumns <= 0)

{

throw new ArgumentException("Размеры матрицы должны быть больше 0");

}

double[,] newData = new double[newRows, newColumns];

for (int i = 0; i < Math.Min(rows, newRows); i++)

{

for (int j = 0; j < Math.Min(columns, newColumns); j++)

{

newData[i, j] = data[i, j];

}

}

rows = newRows;

columns = newColumns;

data = newData;

}

public void Print()

{

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

{

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

{

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

}

Console.WriteLine();

}

}

public void PrintSubmatrix(int startRow, int endRow, int startColumn, int endColumn)

{

if (startRow < 0 || endRow >= rows || startColumn < 0 || endColumn >= columns ||

startRow > endRow || startColumn > endColumn)

{

throw new ArgumentException("Неверные границы подматрицы");

}

for (int i = startRow; i <= endRow; i++)

{

for (int j = startColumn; j <= endColumn; j++)

{

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

}

Console.WriteLine();

}

}

public int CompareTo(object obj)

{

if (obj == null || !(obj is Matrix))

{

throw new ArgumentException("Невозможно сравнить объект с Matrix");

}

Matrix otherMatrix = (Matrix)obj;

int result = rows.CompareTo(otherMatrix.rows);

if (result == 0)

{

result = columns.CompareTo(otherMatrix.columns);

}

return result;

}

}

}

Таблица 9.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Задаются программой | Матрицы |

Анализ результатов:

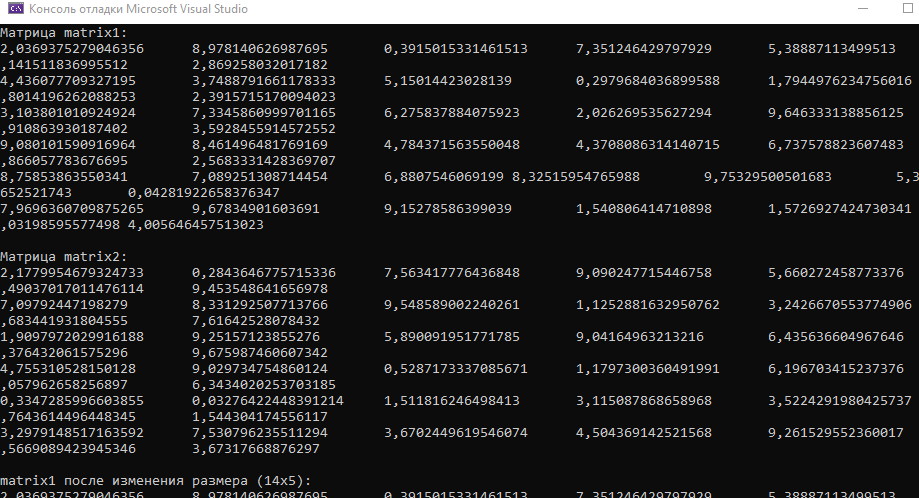


Рисунок 9.2 – Результат работы программы