НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КПІ»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

ЛАБОРАТОРНА РОБОТА №1

*з дисципліни*

*«Об’єктно-орієнтоване програмування»*

**ТЕМА: «C# .Net. Основні синтаксичні конструкції мови програмування C#»**

Підготував: студент групи КП-42

Лисогор Дмитро Юрійович

Перевірила:

Заболотня Тетяна Миколаївна

Київ 2015

**Варіант №10**

Обчислити всі норми вектора: (AB – C)(X+4Y)

**Послідовність виконання завдання**

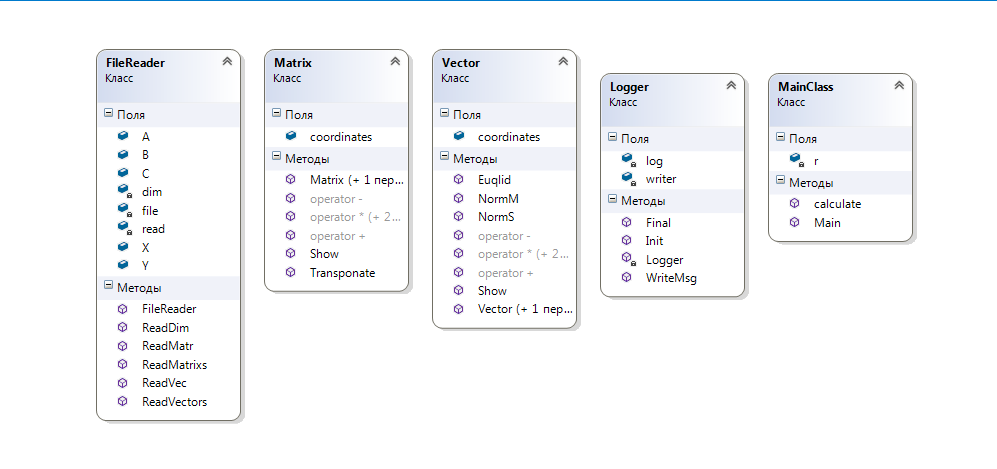
1. Створити відповідні класи для роботи з векторами, матрицями і.т.п.

2. Введення та виведення значень елементів векторів та матриць організувати за допомогою текстових файлів. Створити відповідні методи для роботи з текстовими файлами.

3. Створити файли вхідних даних, вміст яких дозволить якнайбільш повно продемонструвати коректність роботи програми.

4. Забезпечити запис до log-файлу коментарів щодо роботи програми, а також кінцевих результатів її роботи у певному форматі.

5. Підготувати протокол про виконану роботу. Протокол має містити діаграму розроблених класів.



Vector.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OOp

{

public class Vector

{

public double[] coordinates;

public Vector(params double[] crd)

{

coordinates = new double[crd.GetLength(0)];

for (int i = 0; i < crd.GetLength(0); i++)

{

coordinates[i] = crd[i];

}

}

public Vector(Vector prev)

{

coordinates = new double[prev.coordinates.GetLength(0)];

for (int i = 0; i < coordinates.GetLength(0); i++)

{

coordinates[i] = prev.coordinates[i];

}

}

public double Euqlid()

{

double result = 0;

for (int i = 0; i < coordinates.GetLength(0); i++)

{

result += coordinates[i] \* coordinates[i];

}

result = Math.Sqrt(result);

return result;

}

public double NormS()

{

double result = 0;

for (int i = 0; i < coordinates.GetLength(0); i++)

{

result += Math.Abs(coordinates[i]);

}

return result;

}

public double NormM()

{

double result = 0;

for (int i = 0; i < coordinates.GetLength(0); i++)

{

if (result < coordinates[i])

result = coordinates[i];

}

return result;

}

public void Show()

{

for (int i = 0; i < coordinates.GetLength(0); i++)

Console.Write(coordinates[i] + " ");

}

public static Vector operator \*(Vector vec, double val)

{

Vector vect = new Vector(vec);

for (int i = 0; i < vect.coordinates.GetLength(0); i++)

{

vect.coordinates[i] \*= val;

}

return vect;

}

public static Vector operator \*(double val, Vector vec)

{

Vector vect = new Vector(vec);

for (int i = 0; i < vect.coordinates.GetLength(0); i++)

{

vect.coordinates[i] \*= val;

}

return vect;

}

public static double operator \*(Vector left, Vector right)

{

double result = 0;

if (left.coordinates.GetLength(0) != right.coordinates.GetLength(0))

return -1;

for (int i = 0; i < left.coordinates.GetLength(0); i++)

{

result += left.coordinates[i] \* right.coordinates[i];

}

result = Math.Sqrt(result);

return result;

}

public static Vector operator +(Vector left, Vector right)

{

Vector result = new Vector(left);

for (int i = 0; i < result.coordinates.GetLength(0); i++)

{

result.coordinates[i] += right.coordinates[i];

}

return result;

}

public static Vector operator -(Vector left, Vector right)

{

Vector result = new Vector(left);

for (int i = 0; i < result.coordinates.GetLength(0); i++)

{

result.coordinates[i] -= right.coordinates[i];

}

return result;

}

}

}

Matrix.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OOp

{

public class Matrix

{

public double[,] coordinates;

public Matrix(int dim, params double[] coord)

{

if (dim != Math.Sqrt(coord.Length))

throw new Exception("Wrong dims");

coordinates = new double[dim, dim];

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

{

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

{

coordinates[i, j] = coord[i \* dim + j];

}

}

}

public Matrix(Matrix prev)

{

int dim = prev.coordinates.GetLength(0);

coordinates = new double[dim, dim];

coordinates = (double[,])prev.coordinates.Clone();

}

public Matrix Transponate(Matrix m)

{

Matrix result = new Matrix(m);

int dim = m.coordinates.GetLength(0);

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

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

result.coordinates[i, j] = m.coordinates[j, i];

return result;

}

public static Matrix operator +(Matrix left, Matrix right)

{

if (left.coordinates.GetLength(0) != right.coordinates.GetLength(0))

throw new Exception("Wrong dims");

int dim = left.coordinates.GetLength(0);

Matrix res = new Matrix(left);

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

{

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

{

res.coordinates[i, j] += right.coordinates[i, j];

}

}

return res;

}

public static Matrix operator -(Matrix left, Matrix right)

{

if (left.coordinates.GetLength(0) != right.coordinates.GetLength(0))

throw new Exception("Wrong dims");

int dim = left.coordinates.GetLength(0);

Matrix res = new Matrix(left);

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

{

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

{

res.coordinates[i, j] -= right.coordinates[i, j];

}

}

return res;

}

public static Matrix operator \*(Matrix left, double val)

{

int dim = left.coordinates.GetLength(0);

Matrix res = new Matrix(left);

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

{

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

{

res.coordinates[i, j] \*= val;

}

}

return res;

}

public static Matrix operator \*(Matrix l, Matrix r)

{

if (l.coordinates.GetLength(0) != r.coordinates.GetLength(0))

throw new Exception("Wrong dims");

int dim = l.coordinates.GetLength(0);

Matrix result = new Matrix(l);

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

{

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

{

result.coordinates[i, j] = 0;

for (int k = 0; k < dim; k++)

{

result.coordinates[i, j] += l.coordinates[i, k] \* r.coordinates[k, j];

}

}

}

return result;

}

public static Vector operator \*(Matrix M, Vector V)

{

if (M.coordinates.GetLength(0) != V.coordinates.GetLength(0))

throw new Exception("Wrong dims");

int dim = M.coordinates.GetLength(0);

Vector Result = new Vector(V);

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

{

Result.coordinates[i] = 0;

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

{

Result.coordinates[i] += M.coordinates[i, j] \* V.coordinates[j];

}

}

return Result;

}

public void Show()

{

int dim = coordinates.GetLength(0);

for (int i = 0; i < dim; i++, Console.WriteLine())

{

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

Console.Write(coordinates[i, j] + " ");

}

}

}

}

FileReader.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace OOp

{

public class FileReader

{

int dim;

public Vector X, Y;

public Matrix A, B, C;

FileStream file;

StreamReader read;

public FileReader()

{

Logger.WriteMsg("FileReader Started");

file = new FileStream("F:\\Films\\OOP\_LAB1\\read.txt", FileMode.Open, FileAccess.Read);

read = new StreamReader(file);

ReadDim();

ReadVectors();

ReadMatrixs();

Logger.WriteMsg("FileReader finished");

}

public int ReadDim()

{

Logger.WriteMsg("Reading dimension");

try

{

string input = read.ReadLine();

string pattern = "([0-9]+)";

Match m = Regex.Match(input, pattern);

Int32.TryParse(m.Value, out dim);

return dim;

}

catch (Exception e)

{

Logger.WriteMsg("Failed to read dimension");

Logger.Final();

Console.WriteLine("Exception caught in reading dimension");

Console.WriteLine(e);

Console.ReadKey();

Environment.Exit(0);

return -1;

}

}

public Vector ReadVec(Vector vec)

{

Logger.WriteMsg("Reading single vector");

string input = read.ReadLine();

string pattern = @"-?\d+[,]?";

try

{

double[] arr = new double[dim];

int cnt = 0;

foreach (Match m in Regex.Matches(input, pattern))

{

arr[cnt++] = double.Parse(m.Value);

}

if (cnt != dim)

throw new Exception();

vec = new Vector(arr);

return vec;

}

catch (Exception e)

{

Logger.WriteMsg("Failed to read vector");

Logger.Final();

Console.WriteLine("Exception caught in reading vector");

Console.WriteLine(e);

Console.ReadKey();

Environment.Exit(0);

return null;

}

}

public Matrix ReadMatr(Matrix matr)

{

Logger.WriteMsg("Reading single matrix");

string input = read.ReadLine();

string pattern = @"-?\d+[,]?";

try

{

double[] arr = new double[dim \* dim];

int cnt = 0;

foreach (Match m in Regex.Matches(input, pattern))

{

arr[cnt++] = double.Parse(m.Value);

}

if (cnt != dim \* dim)

throw new Exception();

matr = new Matrix(dim, arr);

return matr;

}

catch (Exception e)

{

Logger.WriteMsg("Failed to read matrix");

Logger.Final();

Console.WriteLine("Exception caught in reading matrix");

Console.WriteLine(e);

Console.ReadKey();

Environment.Exit(0);

return null;

}

}

public void ReadVectors()

{

X = ReadVec(X);

Y = ReadVec(Y);

Logger.WriteMsg("Vectors reading finished");

}

public void ReadMatrixs()

{

A = ReadMatr(A);

B = ReadMatr(B);

C = ReadMatr(C);

Logger.WriteMsg("Matrixs reading finished");

}

}

}

Logger.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OOp

{

public class Logger

{

static FileStream log;

static StreamWriter writer;

private Logger()

{ }

public static void Init()

{

log = new FileStream("F:\\Films\\OOP\_LAB1\\log.txt", FileMode.Truncate, FileAccess.Write);

writer = new StreamWriter(log);

WriteMsg("Logging started");

}

public static void WriteMsg(string m)

{

//Console.WriteLine (System.DateTime.Now.ToString () + " " + m);

writer.WriteLine(System.DateTime.Now.ToString() + " " + m);

}

public static void Final()

{

WriteMsg("Logging finished");

writer.Flush();

log.Close();

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OOp

{

class MainClass

{

static FileReader r;

public static void calculate()

{

Logger.WriteMsg("Calculations started");

Vector MainRes = new Vector(0);

MainRes = ((r.A \* r.B - r.C) \* (r.X + 4d \* r.Y));

Console.Write("(AB - C)(X + 4Y) = ");

MainRes.Show();

Console.WriteLine();

Console.WriteLine("Norm Euqlid = " + MainRes.Euqlid().ToString());

Console.WriteLine("Norm Sum = " + MainRes.NormS().ToString());

Console.WriteLine("Norm MaxAbs = " + MainRes.NormM().ToString());

Logger.WriteMsg("Calculations finished");

}

public static void Main(string[] args)

{

Logger.Init();

try

{

r = new FileReader();

Console.Write("X = ");

r.X.Show();

Console.WriteLine();

Console.Write("Y = ");

r.Y.Show();

Console.WriteLine();

Console.WriteLine();

Console.WriteLine("A = ");

r.A.Show();

Console.WriteLine();

Console.WriteLine("B = ");

r.B.Show();

Console.WriteLine();

Console.WriteLine("C = ");

r.C.Show();

MainClass.calculate();

Logger.Final();

}

catch (Exception e)

{

Console.WriteLine("Exception caught in main program");

Console.Write(e);

Console.ReadKey();

Logger.Final();

System.Environment.Exit(-1);

}

Console.ReadKey();

}

}

}

Log.txt

17.10.2015 8:25:24 Logging started

17.10.2015 8:25:24 FileReader Started

17.10.2015 8:25:24 Reading dimension

17.10.2015 8:25:24 Reading single vector

17.10.2015 8:25:24 Reading single vector

17.10.2015 8:25:24 Vectors reading finished

17.10.2015 8:25:24 Reading single matrix

17.10.2015 8:25:24 Reading single matrix

17.10.2015 8:25:24 Reading single matrix

17.10.2015 8:25:24 Matrixs reading finished

17.10.2015 8:25:24 FileReader finished

17.10.2015 8:25:24 Calculations started

17.10.2015 8:25:24 Calculations finished

17.10.2015 8:25:24 Logging finished