ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"

Факультет КНТ

Кафедра ПИ

Лабораторная работа №1

по курсу: «Введение в .Net и Java»

по теме: «Создание консольного приложения .NET с использованием базовых концепций ООП»

Выполнил:

ст. гр. ПИ-19а

Саевский О.В.

Проверил:

Артёменко О.Г.

Ногтев Е.А.

ДОНЕЦК – 2021

ЗАДАНИЕ

Описать класс, реализующий тип данных «вещественная матрица» и работу с ними. Класс должен реализовывать следующие операции над матрицами:

• сложение, вычитание, умножение (как на матрицу, так и на число);

• операции сравнения на равенство/неравенство;

• операции вычисления обратной и транспонированной матрицы;

• методы вычисления детерминанта и нормы;

• методы, реализующие проверку типа матрицы (квадратная, диагональная, нулевая, единичная);

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Menu.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApp1

{

class Menu

{

string text\_menu =

"1. Добавить значение к матрице (число)\n" +

"2. Добавить значение к матрице (матрица)\n" +

"3. Вычитание от матрицы(числа)\n" +

"4. Вычитание от матрицы(матрицы)\n" +

"5. Умножение на матрицу(числа)\n" +

"6. Умножение на матрицу(матрицы)\n" +

"7. Сравнить матрицы\n" +

"8. Перевести в обратную\n" +

"9. Перевести в транспонированную\n" +

"0. Вычислить детерминанту\n" +

"q. Вычислить норму\n" +

"w. Указать тип матрицы\n" +

"Esc. Выход\n";

public Menu() { CreateMatrix(); }

private void CreateMatrix()

{

char key = '1';

while (key != 27)

{

Console.WriteLine(

"1. Создать матрицу по умолчанию\n" +

"2. Создать матрицу вручную\n" +

"Esc - Выход");

key = Console.ReadKey(true).KeyChar; ;

switch (key)

{

case '1':

ZagMatrix mat1 = new ZagMatrix(1);

View(mat1);

break;

case '2':

ZagMatrix mat2 = new ZagMatrix(2);

View(mat2);

break;

}

}

}

private int[] GiveSize()

{

int[] size = new int[2];

while (true)

{

try

{

Console.WriteLine("Укажите размер матрицы:");

Console.Write("Количество столбцов:");

size[0] = Convert.ToInt32(Console.ReadLine());

Console.Write("Количество строк:");

size[1] = Convert.ToInt32(Console.ReadLine());

if (size[0] < 1 || size[1] < 1)

throw new Exception("<1");

break;

}

catch (Exception ex)

{

Console.WriteLine("Значение введено неверно. Введите корректное значение.");

}

}

return size;

}

private float InputFloat()

{

while (true)

{

try

{

float a = float.Parse(Console.ReadLine());

return a;

}

catch (Exception ex)

{

Console.WriteLine("Значение введено неверно. Введите корректное значение.");

}

}

}

private float[][] InputMatrix(int[] size)

{

float[][] matrix = new float[size[0]][];

for (int i = 0; i < size[0]; i++)

{

matrix[i] = new float[size[1]];

}

for (int i = 0; i < size[0]; i++)

{

for (int j = 0; j < size[1]; j++)

{

Console.Write("Введите значение matrix[" + i + "][" + j + "]: ");

matrix[i][j] = InputFloat();

}

}

return matrix;

}

public void View(ZagMatrix matrix)

{

char key = '0';

int[] size = GiveSize();

float[][] mat = InputMatrix(size);

Matr mat1 = new Matr(size, mat);

while (key != 27)

{

Console.WriteLine(mat1.ToString());

Console.WriteLine(text\_menu);

key = Console.ReadKey(true).KeyChar;

switch (key)

{

case '1':

Console.Write("Укажите значение: ");

matrix.Plus(InputFloat(), ref mat1);

break;

case '2':

int[] size2 = GiveSize();

float[][] matr2 = InputMatrix(size2);

Matr mat2 = new Matr(size2, matr2);

matrix.Plus2(ref mat1, ref mat2);

break;

case '3':

Console.Write("Укажите значение: ");

matrix.Diff(InputFloat(), ref mat1);

break;

case '4':

int[] size3 = GiveSize();

float[][] mat13 = InputMatrix(size3);

Matr mat3 = new Matr(size3, mat13);

matrix.Diff2(ref mat1, ref mat3);

break;

case '5':

Console.Write("Укажите значение: ");

matrix.Mult(InputFloat(), ref mat1);

break;

case '6':

int[] size4 = GiveSize();

float[][] mat14 = InputMatrix(size4);

Matr mat4 = new Matr(size4, mat14);

matrix.Mult2(ref mat1, ref mat4);

break;

case '7':

int[] size5 = GiveSize();

float[][] mat15 = InputMatrix(size5);

Matr mat5 = new Matr(size5, mat15);

matrix.Comprasion(ref mat1, ref mat5);

break;

case '8':

matrix.Reverse(ref mat1);

break;

case '9':

matrix.Trans(ref mat1);

break;

case '0':

matrix.Determ(mat1);

break;

case 'q':

matrix.Norm(mat1);

break;

case 'w':

matrix.Type(mat1);

break;

}

}

}

}

}

Matrix.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApp1

{

class Matrix:Interface

{

public Matrix()

{}

static Matrix() { Console.WriteLine("Статический конструктор вызван\n"); }

public Matrix(string text)

{

Console.WriteLine(text);

}

private int[] GiveSize()

{

int[] size = new int[2];

Console.WriteLine("Укажите размер матрицы:");

Console.Write("Количество столбцов:");

size[0] = Convert.ToInt32(Console.ReadLine());

Console.Write("Количество строк:");

size[1] = Convert.ToInt32(Console.ReadLine());

return size;

}

public virtual void Plus(float v, ref Matr mat)

{

for(int i = 0; i < mat.height; i++)

{

for(int j = 0; j < mat.width; j++)

{

mat.matrix[i][j] += v;

}

}

}

public virtual void Plus2(ref Matr mat1,ref Matr mat2)

{

if (mat1.height == mat2.height && mat1.width == mat2.width)

{

for (int i = 0; i < mat1.height; i++)

{

for (int j = 0; j < mat1.width; j++)

{

mat1.matrix[i][j] += mat2.matrix[i][j];

}

}

}

}//сложение

public virtual void Diff(float v,ref Matr mat)

{

for (int i = 0; i < mat.height; i++)

{

for (int j = 0; j < mat.width; j++)

{

mat.matrix[i][j] -= v;

}

}

}

public virtual void Diff2(ref Matr mat1,ref Matr mat2)

{

if (mat1.height == mat2.height && mat1.width == mat2.width)

{

for (int i = 0; i < mat1.height; i++)

{

for (int j = 0; j < mat1.width; j++)

{

mat1.matrix[i][j] -= mat2.matrix[i][j];

}

}

}

}//сложение

public virtual void Mult(float v,ref Matr mat)

{

for (int i = 0; i < mat.height; i++)

{

for (int j = 0; j < mat.width; j++)

{

mat.matrix[i][j] = v;

}

}

}

public virtual void Mult2(ref Matr mat1, ref Matr mat2)

{

if (mat1.width == mat2.height)

{

float[][] matrix3 = new float[mat1.height][];

for (int i = 0; i < mat1.height; i++)

{

matrix3[i] = new float[mat2.width];

}

int height = mat1.height;

int width = mat1.width;

int width2 = mat2.width;

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

{

for (int j = 0; j < width2; j++)

{

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

{

matrix3[i][j] += mat1.matrix[i][k] \* mat2.matrix[k][j];

}

}

}

mat1.matrix = matrix3;

mat1.width = width2;

}

}

public virtual void Comprasion(ref Matr mat1, ref Matr mat2)

{

if(mat1.height == mat2.height && mat1.width == mat2.width)

{

for(int i = 0; i < mat1.height; i++)

{

for(int j = 0; j < mat2.width; j++)

{

if(mat1.matrix[i][j] != mat2.matrix[i][j])

{

Console.WriteLine("\nМатрицы не равны.");

return;

}

}

}

Console.WriteLine("\nМатрицы равны");

return;

}

Console.WriteLine("\nМатрицы не равны.");

}//Сравнение

public virtual void Reverse(ref Matr mat)

{

float det = Determ(mat);

if (mat.height != mat.width)

{

Console.WriteLine("Матрица не квадратная");

return;

}else if(det == 0)

{

Console.WriteLine("Определитель = 0.");

return;

}

Trans(ref mat);

for(int i = 0; i < mat.height; i++)

{

for(int j = 0; j < mat.width; j++)

{

mat.matrix[i][j] /= det;

}

}

}//Перевод в обратную

public virtual float[][] Trans(ref Matr mat)

{

float[][] matrix2 = new float[mat.width][];

for(int i = 0; i < mat.width; i++)

{

matrix2[i] = new float[mat.height];

}

for (int i = 0; i < mat.height; i++)

{

for(int j = 0; j < mat.width; j++)

{

matrix2[j][i] = mat.matrix[i][j];

}

}

mat.matrix = matrix2;

int temp = mat.width;

mat.width = mat.height;

mat.height = temp;

return mat.matrix;

}//Перевод в транспонированную

public virtual float Determ(Matr mat)

{

if (mat.height != mat.width)

{

Console.WriteLine("Определитель может быть только в квадратной матрице");

return 0;

}

float det = 1;

int EPS = 0;

float[][] a = new float[mat.width][];

for(int i = 0; i < mat.width; i++)

{

a[i] = new float[mat.width];

for (int j = 0; j < mat.width; j++)

a[i][j] = mat.matrix[i][j];

}

float[][] b = new float[1][];

b[0] = new float[mat.width];

for (int i = 0; i < mat.width; ++i)

{

//присваиваем k номер строки

int k = i;

//идем по строке от i+1 до конца

for (int j = i + 1; j < mat.width; ++j)

//проверяем

if (Math.Abs(a[j][i]) > Math.Abs(a[k][i]))

//если равенство выполняется то k присваиваем j

k = j;

//если равенство выполняется то определитель приравниваем 0 и выходим из программы

if (Math.Abs(a[k][i]) < EPS)

{

det = 0;

break;

}

//меняем местами a[i] и a[k]

b[0] = a[i];

a[i] = a[k];

a[k] = b[0];

//если i не равно k

if (i != k)

//то меняем знак определителя

det = -det;

//умножаем det на элемент a[i][i]

det \*= a[i][i];

//идем по строке от i+1 до конца

for (int j = i + 1; j < mat.width; ++j)

//каждый элемент делим на a[i][i]

a[i][j] /= a[i][i];

//идем по столбцам

for (int j = 0; j < mat.width; ++j)

//проверяем

if ((j != i) && (Math.Abs(a[j][i]) > EPS))

//если да, то идем по k от i+1

for (k = i + 1; k < mat.width; ++k)

a[j][k] -= a[i][k] \* a[j][i];

}

Console.WriteLine("Определитель: " + det.ToString());

return det;

}//Детерминанта матрицы

public virtual float Norm(Matr mat)

{

float max = float.MinValue;

float temp;

for (int i = 0; i < mat.height; i++)

{

temp = 0;

for (int j = 0; j < mat.width; j++)

{

temp = temp + mat.matrix[i][j];

}

if (temp > max) max = temp;

}

Console.WriteLine("Норма: " + max);

return max;

}//Норма матрицы

public virtual void Type(Matr mat)

{

if (mat.height == mat.width)

Console.WriteLine("Матрица квадратная");

bool diag = true, ed = true, nul = true;

for(int i = 0; i < mat.height; i++)

{

for (int j = 0; j < mat.height; j++)

{

if (i != j && diag == true && mat.matrix[i][j] != 0)

{

diag = false;

ed = false;

}

if (i == j && mat.matrix[i][j] != 1)

ed = false;

if (mat.matrix[i][j] != 0)

nul = false;

}

}

if(diag == true)

Console.WriteLine("Матрица диагональная");

if(ed == true)

Console.WriteLine("Матрица единичная");

if(nul == true)

Console.WriteLine("Матрица нулевая");

}//Тип матрицы

}

}

Interface.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApp1

{

interface Interface

{

void Plus(float type, ref Matr mat);//сложение

void Plus2(ref Matr mat1, ref Matr mat2);//сложение

void Diff(float type, ref Matr mat);//Вычитание

void Diff2(ref Matr mat1, ref Matr mat2);

void Mult(float type, ref Matr mat);//Умножение

void Mult2(ref Matr mat1, ref Matr mat2);

void Comprasion(ref Matr mat1, ref Matr mat2);//Сравнение

void Reverse(ref Matr mat1);//Перевод в обратную

float[][] Trans(ref Matr mat1);//Перевод в транспонированную

float Determ(Matr mat1);//Детерминанта матрицы

float Norm(Matr mat1);//Норма матрицы

void Type(Matr mat1);//Тип матрицы

}

}

Matr.cs

using System;

public class Matr

{

public Matr(int[] size, float[][] matr)

{

matrix = matr;

height = size[0];

width = size[1];

}

public float[][] matrix;

public int height, width;

public override string ToString()

{

string line = "\nМатрица:\n";

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

{

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

{

line += matrix[i][j] + " ";

}

line += "\n";

}

line += "\nДлина: " + width + " Высота: " + height + "\n";

return line;

}

}

ZagMatr.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApp1

{

class ZagMatrix : Matrix

{

public ZagMatrix(int i)

{

if (i == 1)

matrix = new Matrix();

else if (i == 2)

matrix = new Matrix("Конструктор с параметрами вызван");

}

Matrix matrix;

public override void Plus(float type, ref Matr mat)

{

Console.WriteLine("Заглушка Plus");

matrix.Plus(type, ref mat);

}//сложение

public override void Plus2(ref Matr mat1, ref Matr mat2)

{

Console.WriteLine("Заглушка Plus2");

matrix.Plus2(ref mat1, ref mat2);

}//сложение//сложение

public override void Diff(float type, ref Matr mat)

{

Console.WriteLine("Заглушка Diff");

matrix.Diff(type, ref mat);

}//сложение//Вычитание

public override void Diff2(ref Matr mat1, ref Matr mat2)

{

Console.WriteLine("Заглушка Diff2");

matrix.Diff2(ref mat1, ref mat2);

}//сложение

public override void Mult(float type, ref Matr mat)

{

Console.WriteLine("Заглушка Mult");

matrix.Mult(type, ref mat);

}//сложение//Умножение

public override void Mult2(ref Matr mat1, ref Matr mat2)

{

Console.WriteLine("Заглушка Mult2");

matrix.Mult2(ref mat1, ref mat2);

}//сложение

public override void Comprasion(ref Matr mat1, ref Matr mat2)

{

Console.WriteLine("Заглушка Comprasion");

matrix.Comprasion(ref mat1, ref mat2);

}//сложение//Сравнение

public override void Reverse(ref Matr mat1)

{

Console.WriteLine("Заглушка Reverse");

matrix.Reverse(ref mat1);

}//сложение//Перевод в обратную

public override float[][] Trans(ref Matr mat1)

{

Console.WriteLine("Заглушка Trans");

return matrix.Trans(ref mat1);

}//сложение//Перевод в транспонированную

public override float Determ(Matr mat1)

{

Console.WriteLine("Заглушка Determ");

return matrix.Determ(mat1);

}//сложение//Детерминанта матрицы

public override float Norm(Matr mat1)

{

Console.WriteLine("Заглушка Norm");

return matrix.Norm(mat1);

}//сложение//Норма матрицы

public override void Type(Matr mat1)

{

Console.WriteLine("Заглушка Type");

matrix.Type(mat1);

}//сложение//Тип матрицы

}

}

Program.cs

позволяющее осуществить проверку всех методов класса.\*/

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

Menu menu = new Menu();

}

}

}

Примеры работы программы:





