Содержание

Введение......................................................................................3

Краткие теоретические сведения..............................................5

Основные определения.................................................................................5

Действия над матрицами..............................................................................6

Свойства матричных операций.....................................................................8

Понятие об определителях...........................................................................9

Свойства определителей.............................................................................10

Решение систем линейных алгебраических уравнений матричным способом.................................................................................................................11

Разработка программного средства.........................................12

Разработка функций для выполнения матричных операций...................12

Разработка методов для взаимодействия с формой.................................15

Разработка графического интерфейса пользователя.................................16

Приложение................................................................................18Список использованной литературы........................................23

ВВЕДЕНИЕ

Высшая алгебра представляет собой далеко идущее, но вполне естественное обобщение основного содержания школьного курса алгебры. Центральное место в школьной алгебре, бесспорно, занимает вопрос о решении уравнений. Важной составляющей этого вопроса является рассмотрение систем двух, трёх и более уравнений с несколькими неизвестными.

Это направление получило развитие в курсе высшей алгебры. Раздел алгебры, занимающийся рассмотрением данного вопроса, носит название «Основы линейной алгебры» и имеет исходной задачей изучение произвольных систем уравнений первой степени или, как говорят, линейных уравнений. Для решения таких систем в том случае, когда число уравнений равно числу неизвестных, разрабатывается аппарат теории определителей. Этого аппарата уже недостаточно, однако, для изучения таких систем линейных уравнений, у которых число неизвестных не равно числу уравнений, - случай непривычный с точки зрения элементарной алгебры, но очень важный для приложений. Оказалось необходимым, в частности, разрабатывать теорию матриц, то есть систем чисел, расположенных в квадратные или прямоугольные таблицы из нескольких строк и столбцов. Эта теория оказалась очень глубокой и нашла приложения далеко за пределами теории систем линейных уравнений.

К середине XIX в. матрицы стали самостоятельными объектами математических исследований. К этому времени были сформулированы правила сложения и умножения матриц. Основную роль в их разработке сыграли работы Гамильтона, Кэли и Сильвестра. Современное обозначение матрицы предложил Кэли в 1841 году. Исследования Вейерштрасса и Фробениуса далеко продвинули теорию матриц, обогатив её новым содержанием.

Матрицы применяются не только для решения систем уравнений. В физике и других прикладных науках матрицы являются средством записи данных и их преобразования. В программировании они имеют важную роль в написании программ (их ещё называют массивами). Широкое применение матрицы нашли в технике. Например, любая картинка на экране – это двумерная матрица, элементами которой являются цвета точек. В психологии понимание термина сходно с его определением в математике, но вместо математических объектов подразумеваются некие “психологические объекты”, например, тесты. Кроме того, матрицы имеют широкое применение в экономике, биологи, химии и даже в маркетинге.

Калькулятор матриц облегчает изучение данной темы в математическом контексте. Он имеет функции сложения, умножения матриц, поиска обратной матрицы, вычисления определителя, возведения матрицы в натуральную степень и т.д. Кроме того в нём реализована возможность решения матричных уравнений, которые являются одним из способов решения систем линейных алгебраических уравнений. Калькулятор довольно прост в использовании и может применяться как для непосредственных вычислений, так и для самоконтроля.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Матрица – прямоугольная таблица чисел размера m×n, где m – число строк, n – число столбцов таблицы. Матрицы обозначают заглавными буквами латинского алфавита. Элементы матрицы обозначают с помощью строчных латинских символов с двойными индексами, причём первый индекс – номер строки, а второй – номер столбца, в которых находится элемент. Например, элемент а13 находится в первой строке и в третьем столбце некоторой заданной матрицы.

Если в матрице число строк равно числу столбцов, то матрица называется **квадратной**. Число строк или столбцов такой матрицы называют её *порядком*.

Матрица, в которой число строк не равно числу столбцов называется *прямоугольной*.

Матрица, состоящая только из одной строки (столбца) называется *вектор-строкой* (*вектор-столбцом*).

Если все элементы матрицы равны нулю, то матрица является *нулевой*. Её обозначают, как О или 0.

**Главной диагональю** матрицы называют ей диагональ, идущей из верхнего левого угла в правый нижний.

1 2 3

А = 0 3 0

5 0 1

Квадратная матрица, у которой все элементы, лежащие ниже главной диагонали равны нулю, называется *треугольной* матрицей.

а11 а12 а13

А = 0 а22 а23

0 0 а33

­Квадратная матрица, у которой все элементы, кроме быть может стоящих на главной диагонали равны нулю, называется *диагональной*.

Например, матрицы В и С являются диагональными.

1 0 0 1 0 0

В = 0 3 0 , С = 0 0 0

0 0 4 0 0 5

Диагональная матрица, у которой все элементы на главной диагонали равны 1, называется *единичной* и обозначается символом Е. Например, единичная матрица третьего порядка имеет вид

1 0 0

Е = 0 1 0

1. 0 1

**Действия над матрицами.**

1. Равенство матриц

Две матрицы А и B называются равными, если они имеют одинаковое число строк и столбцов и их соответствующие элементы аij = bij. Например,

а11 а12

а21  а22

b11 b12

b21 b22

пусть даны матрицы А = и В = . Говорят, что А = В, если

а11 = b11, a12 = b12, a21 = b21, a22 = b22.

1. Транспонирование

Рассмотрим произвольную матрицу А из m строк и n столбцов. Ей можно сопоставить такую матрицу В из n строк и m столбцов, у которой каждая строка является столбцом матрицы А с тем же номером (следовательно, каждый столбец является строкой матрицы А с тем же номером). Итак, если

a11 … am1

a12 … am2

. . . . .

a1n … amn

а11 а12 … а1n

. . . . . .

am1 am2 … amn

А = , то B = . Эту матрицу В называют

*транспонированной матрицей А*, а переход от А к В – *транспонированием.*

Транспонированную матрицуобозначают АТ. Связь между матрицей А и её транспонированной можно записать в виде аTij = aji.

1. Сложение матриц.

Пусть матрицы А и В состоят из одинакового числа строк и столбцов, т.е. имеют одинаковые размеры. Тогда для того, чтобы сложить матрицы А и В нужно к элементам матрицы А прибавить соответствующие элементы матрицы В. Таким образом суммой двух матриц А и В является матрица С, которая определяется по правилу

a11 + b11 a12 + b12 a13 + b13

a21 + b21 a22 + b22 a23 + b23

a31 + b31 a32 + b32 a33 + b33

b11 b12 b13

b21 b22 b23

b31 b32 b33

a11 a12 a13

a21 a22 a23

a31 a32 a33

А + В = + = = C

или сij = aij + bij.

1. Умножение матрицы на число.

Чтобы умножить матрицу А на число k, нужно каждый элемент матрицы А умножить на число k. Таким образом, произведение матрицы А на число k есть новая матрица, которая определяется по правилу

a11 \* k a12 \* k a13 \* k

a21 \* k a22 \* k a23 \* k

a31 \* k a32 \* k a33 \* k

a11 a12 a13

a21 a22 a23

a31 a32 a33

kA = k \* = или cij = kaij.

1. Умножение матриц.

Для того, чтобы можно было перемножить две матрицы, количество строк второй матрицы должно быть равно количеству столбцов первой. То есть можно умножать матрицы A\*B с размерами Am×n и Bn×k , при этом B на А умножить нельзя.

Умножение матриц осуществляется по следующему правилу.

a11 a12 ... a1n

a21 a22 ... a2n

. . . . . .

am1 am2 ... amn

c11 c12 ... c1k

c21 c22 ... c2k

. . . . . .

cm1 cm2 ... cmk

b11 b12 ... b1k

b21  b22 ... b2k

. . . . . . .

bn1 bn2 ... bnk

× = , где Cij =.

**Свойства матричных операций.**

1. Коммутативность сложения

А + В = В + А

1. Ассоциативность сложения

(А + В) + С = А + (В + С)

1. Дистрибутивность умножения на константу относительно сложения матриц

k\*(A + B) = k\*A + k\*B

1. Для любой матрицы А существует противоположная матрица (-А), такая, что А + (-А) = О, где О – нулевая матрица
2. Умножение матриц в общем случае некоммутативно

АВ ≠ ВА

1. Ассоциативность умножения

k(AB) = (kA)B = A(kB)

(AB)C = A(BC)

1. Дистрибутивность умножения матриц относительно сложения

А(В + С) = АВ + АС

(А + В)С = АС + ВС

1. При транспонировании транспонированной матрицы получается исходная матрица

(АT)T = A

1. Ассоциативность умножения транспонированной матрицы на константу

(kAT) = k(AT)

1. Дистрибутивность транспонирования относительно сложения

(А + В)Т = АТ + ВТ

1. Дистрибутивность транспонирования относительно умножения

(АВ)Т = ВТАТ

**Понятие об определителях**

Определителем n-го порядка матрицы называется алгебраическая сумма всевозможных произведений элементов, взятых по одному и только по одному из каждой строки и каждого столбца матрицы. Знак каждого слагаемого определяется числом инверсий в перестановках, составленных из первых и вторых индексов сомножителей aij: если сумма инверсий чётная, то слагаемое берётся со знаком «+», если она нечётная, то слагаемое берётся со знаком «-». Таким образом, по определению

a11 a12 ... a1n

a21 a22 ... a2n

|A| = . . . . = (-1)s + tai1j1ai2j2...ainjn, i = 1,n, j = 1,n,

an1 an2 ... ann

где s – число инверсий в перестановке первых индексов, а t – число инверсий в перестановке вторых индексов.

Минором элемента aij называется определитель (n-1)-го порядка, полученный после вычёркивания в исходном определителе i-й строки и j-го столбца. (Обозначается Мij)

Алгебраическим дополнением элемента aij определителя |A| называется число Aij = (-1)i+jMij.

Теорема Лапласа(о вычислении определителя).

Определитель |A| равен сумме произведений элементов строки(столбца) матрицы А на соответствующие алгебраические дополнения элементов этой строки(столбца).

**Свойства определителей.**

1. Определитель не меняется при транспонировании.
2. Если одна из строк (один из столбцов) определителя состоит из нулей, то определитель равен нулю.
3. Если один определитель получен из другого путём перестановки двух строк, то все члены первого определителя будут членами и во втором, но с обратным знаком, то есть от перестановки двух строк определитель лишь меняет знак.
4. Определитель, содержащий две одинаковые строки (столбца) равен нулю.
5. Если все элементы некоторой строки (столбца) определителя умножить на некоторое число k, то сам определитель умножится на k.
6. Определитель, содержащий две пропорциональные строки (два пропорциональных столбца) равен нулю.
7. Если все элементы i-й строки определителя n-го порядка представлены в виде суммы двух слагаемых:

aij = bj + cj , j = 1, ..., n,

то определитель равен сумме двух определителей, у которых все строки, кроме i-й, ­— такие же, как и в заданном определителе, а i-я строка в одном из слагаемых состоит из элементов bj, в другом — из элементов cj.

1. Если одна из строк определителя есть линейная комбинация его других строк, то определитель равен нулю.
2. Определитель не меняется, если к элементам одной его строки прибавить соответственные элементы другой строки, умноженные на одно и то же число.

**Решение систем линейных алгебраических уравнений матричным способом.**

Пусть дана система линейных алгебраических уравнений вида:

a11x1 + a12x2 + a13x3 = b1

a21x1 + a22x2 + a23x3 = b2

a31x1 + a32x2 + a33x3 = b3

Данная система в матричной форме запишется как A∙X = B, где

x1

x2

x3

b1

b2

b3

a11 a12 a13

a21 a22 a23

a31 a32 a33

А = , B = , X =

A – основная матрица системы

В – матрица свободных членов

Х – матрица-столбец неизвестных

Пусть для матрицы А существует обратная матрица А-1. Умножим обе части равенства на А-1 слева.

А-1∙А∙Х = А-1∙В

Е∙Х = А-1∙В

Х = А-1∙ В

Решение системы линейных алгебраических уравнений матричным методом определяется по формуле Х = А­-1∙В. Другими словами решение системы находится с помощью обратной матрицы А-1. Так как обратная матрица существует тогда и только тогда, когда определитель матрицы отличен от нуля, систему можно решать матричным методом только, когда определитель основной матрицы не равен нулю.

Разработка программного средства

**Разработка функций для выполнения матричных операций.**

В отдельном файле программы Functions.h создаём класс Functions и структуру Mat с полями rows (содержит число строк матрицы), colons (содержит число столбцов матрицы) и data (содержит элементы матрицы).

В классе Functions объявляем статические методы Sum (сумма), Diferance (разность), Mult (умножение матриц), Scalar (умножение матрицы на число), Transpose (транспонирование матрицы), Determinant (определитель), Addition (алгебраическое дополнение), Inverse (обратная матрица), Power (возведение в степень).

В файле Functions.cpp реализуем вышеуказанные методы.

Функции Sum и Diferance принимают в качестве параметров 2 матрицы matrix1 и matrix2 и возвращают матрицу-результат их сложения либо вычитания.

Алгоритм суммирования двух матриц:

В системе вложенных циклов каждому элементу поля data аргумента matrix1 прибавляем элемент поля data аргумента matrix2 с такими же индексами. Возвращаем матрицу matrix1.

Аналогичный вид имеет алгоритм функции Diferance.

Функция Mult принимает в качестве параметров 2 матрицы matrix1 и matrix2 и возвращает результат произведения этих матриц – новую матрицу.

Алгоритм произведения матриц:

• Объявляем матрицу res, в которой будем хранить результат.

• Создаём переменную temp, в которой будем хранить промежуточные значения.

• В системе вложенных циклов в переменную temp записываем сумму произведений элементов строки первой матрицы на элементы столбца второй матрицы.

• Значение переменной temp присваиваем соответствующему элементу матрицы res.

• После прохождения всех итераций возвращаем res.

Функция Scalar принимает в качестве параметров матрицу matrix и целое число k, а возвращает матрицу-результат произведения.

Алгоритм умножения матрицы на число:

В системе вложенных циклов каждый элемент матрицы умножаем на число k, передаваемое как параметр. Возвращаем полученную матрицу.

Функция Transpose принимает параметром матрицу matrix, а возвращает транспонированную матрицу.

Алгоритм транспонирования матрицы:

• Создаём матрицу res, в которой будет храниться результат операции.

• В системе вложенных циклов каждому элементу rij матрицы res присваиваем значение элемента mji матрицы, переданной через параметр.

• Возвращаем res.

Функция Determinant принимает в качестве параметров массив matrix и её порядок ord, а возвращает целое число – определитель матрицы.

Алгоритм вычисления определителя:

• Если порядок равен 1 возвращаем первый элемент матрицы, иначе идём на следующий шаг.

• Создаём переменную temp, в которой будем хранить промежуточные значения суммы.

• В цикле создаём массив, в котором на одну строку и на один столбец меньше, чем в исходном.

• В системе вложенных циклов записываем в новый массив элементы исходного без i-й строки и i-го столбца.

• К значению переменной temp прибавляем умноженное на значение элемента из первой строки и i-го столбца и на (-1)i значение рекурсивного вызова фунции Determinant, в которую в качестве параметра передаём новый массив и порядок на единицу меньший.

• Возвращаем temp.

Функция Inverse принимает в качестве параметра матрицу matrix, а возвращает транспонированную матрицу алгебраических дополнений элементов исходной матрицы.

Алгоритм поиска обратной матрицы:

В цикле каждому элементу матрицы вызывается метод Addition, который возвращает алгебраическое дополнение этого элемента, и записывается в новую матрицу res. Затем для матрицы res вызываем метод Transpose, результат выполнения которого является возвращаемым значением функции Inverse.

Функция Power принимает в качестве аргументов матрицу matrix и целое число – значение степени n. Возвращаемым значением является матрица – результат возведения матрицы matrix в степень n.

Алгоритм возведения матрицы в натуральную степень:

• Если n < 0, выводим сообщение о невозможности возведения матрицы в отрицательную степень.

• Если n = 0, возвращаем единичную матрицу.

В остальных случаях:

• В цикле выполняем умножение матриц, пока счётчик не достигнет значения n.

• Возвращаем результат этого произведения.

**Разработка методов для взаимодействия с формой.**

В отдельном файле Matrixs.h объявим класс Matrixs. В классе объявим private поля matix1, matrix2, result, которые будут хранить матрицы.

Краткая характеристика методов класса Matrixs.

• AddMatrix1 принимает список компонент TextBox, размеры матрицы и на их основе создаёт массив чисел, который вместе с размерами матрицы помещает в поле matrix1.

• AddMatrix2 принимает список компонент TextBox, размеры матрицы и на их основе так же создаёт массив чисел, который вместе с размерами матрицы помещает в поле matrix2.

• Perform принимает параметром компонент TextBox, в котором содержится информация о знаке арифметической операции и в соответствии с этой информацией записывает в поле result матрицу – результат вычислений.

• Transponse выполняет транспонирование матрицы, результат записывает в матрицу result.

• Invers выполняет поиск обратной матрицы, результат записывает в матрицу result.

• Power принимает параметром целое число n – показатель степени. Выполняет возведение матрицы в степень, результат записывает в матрицу result.

• AXB и XAB выполняют решения матричных уравнений вида AX = B и XA = B соответственно. Результат записывается в поле result.

• Result принимает в качестве параметра списо компонент TextBox и заполняет его значениями матрицы result. Возвращает заполненный список компонент TextBox.

**Разработка графического интерфейса пользователя.**

В конструкторе формы создаём объекты компнонентов графического интерфейса (рис.1).

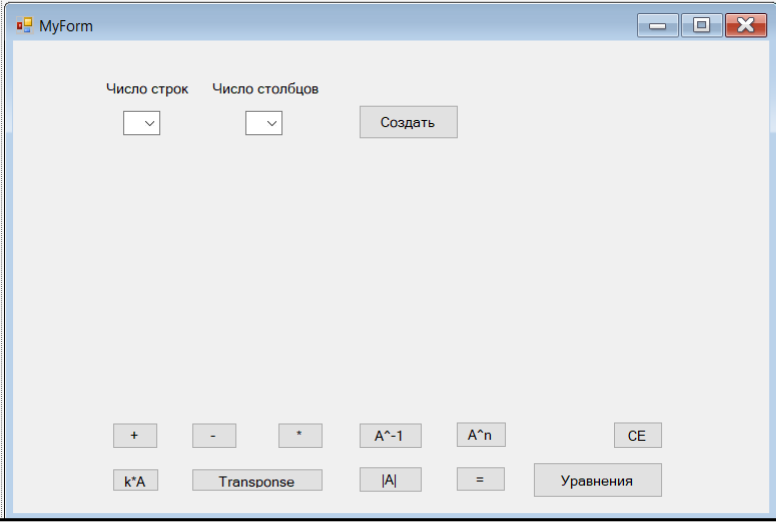


рис.1

Устанавливаем обработчик события при изменении индекса в компоненте ComboBox по следующей логике:

1) Программы удаляет с формы раннее созданные компоненты TextBox;

2) Создаются новые компоненты TextBox в зависимости от выбранных значений.

Обработчики события нажатия на кнопки «+» и «-» имеют схожую структуру:

1. В компонент TextBox sign, предназначенных для ввода символа, записывается «+» или «-»;
2. Создаётся новая матрица из компонент TextBox такого же размера, как предыдущая;
3. Создаётся компонент equal, предназначенный для записи знака «=».

Обработчик события нажатия на кнопку «\*» открывает новую форму для ввода размера второй матрицы, причём число строк второй матрицы уже зафиксировано. После ввода размера создаётся матрица компонент TextBox, а в компонент sign записывается «\*».

При нажатии на кнопку k\*A появляется новый компонент TextBox, который предназначен для ввода числа, на которое нужно умножить матрицу. В компонент sign записывается «\*».

При нажатии на кнопку «=» создаётся новая матрица, исходя из размеров предыдущих и знака арифметической операции. Происходит вычисление значений новой матрицы и их запись в созданные компоненты TextBox.

Обработчики события нажатия на кнопки «A^-1», «Transponse», «|A|» в отличие от предыдущих создают новую матрицу не для ввода значений, для вывода результата. Сначала считываются значения первой матрицы, затем происходит вычисление результата операции и вывод его на экран при помощи компонент TextBox. Таким образом, кнопку «=» нажимать не нужно, операция выполняется сразу.

Обработчик события нажатия на кнопку «A^n» открывает новое окно, в котором требуется ввести показатель степени, в которую нужно возвести матрицу. Выполняется возведение в степень и результат отображается в компонентах TextBox.

Для очистки окна описан обработчик события нажатия на кнопку «СЕ», для ввода размеров матрицы с клавиатуры – кнопки «Создать».

Все обработчики объявлены в файле MyForm.h, а их реализация находится в файле MyForm.cpp.

Аналогично создано окно формы для решения матричных уравнений, который вызывается нажатием на кнопку «Уравнения». Отличие состоит в том, что объекты новой формы создаются динамически из кода программы, а не используя конструктор формы. Класс формы для решения уравнений описан в файле EquationForm.h, а его методы реализованы в файле EquationForm.cpp.

**Приложение**

Исходный код программы.

Functions.h

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::Collections::Generic;

struct Mat

{

int rows;

int colons;

int\*\* data;

};

ref class Functions

{

public:

Functions();

static int Convertation(String^ str);

static Mat\* Sum(Mat\* matrix1, Mat\* matrix2);

static Mat\* Diferance(Mat\* matrix1, Mat\* matrix2);

static Mat\* Mult(Mat\* matrix1, Mat\* matrix2);

static Mat\* Scalar(Mat\* matrix, int p);

static Mat\* Transpose(Mat\* matrix);

static int Determinant(int \*\*arr, int ord);

static int Addition(int \*\*arr, int ord, int ind1, int ind2 );

static Mat\* Inverse(Mat\* matrix);

static Mat\* Power(Mat\* matrix, int n);

};

Functions.cpp

#include "Functions.h"

#include <math.h>

Mat\* InitArr(Mat\* mat, int rows, int colons)

{

mat->data = new int\*[rows];

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

{

mat->data[i] = new int[colons];

}

mat->rows = rows;

mat->colons = colons;

return mat;

}

void DeleteArr(int \*\*&arr, int size)

{

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

delete[] arr[i];

delete[] arr;

}

Mat\* Functions::Sum(Mat\* matrix1, Mat\* matrix2)

{

int M = matrix1->rows, N = matrix1->colons;

Mat\* res = new Mat;

res = InitArr(res, M, N);

int k = 0;

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

{

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

{

res->data[i][j] = matrix1->data[i][j] + matrix2->data[i][j];

}

}

return res;

}

Mat\* Functions::Diferance(Mat\* matrix1, Mat\* matrix2)

{

int M = matrix1->rows, N = matrix1->colons;

Mat\* res = new Mat;

res = InitArr(res, M, N);

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

{

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

{

res->data[i][j] = matrix1->data[i][j] - matrix2->data[i][j];

}

}

return res;

}

Mat\* Functions::Mult(Mat\* matrix1, Mat\* matrix2)

{

int M = matrix1->rows, N = matrix2->colons;

Mat\* res = new Mat;

res = InitArr(res, M, N);

int temp;

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

{

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

{

temp = 0;

for (int h = 0; h < matrix1->colons; h++)

{

temp += matrix1->data[i][h] \* matrix2->data[h][j];

}

res->data[i][j] = temp;

}

}

return res;

}

Mat\* Functions::Scalar(Mat\* matrix, int p)

{

for (int i = 0; i < matrix->rows; i++)

{

for (int j = 0; j < matrix->colons; j++)

{

matrix->data[i][j] \*= p;

}

}

return matrix;

}

Mat\* Functions::Transpose(Mat\* matrix)

{

Mat\* res = new Mat;

int M = matrix->colons, N = matrix->rows;

res = InitArr(res, M, N);

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

{

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

{

res->data[i][j] = matrix->data[j][i];

}

}

return res;

}

int Functions::Determinant(int \*\*arr, int ord)

{

if (ord == 1)

return arr[0][0];

else

{

int i = 0;

int tmp = 0;

while (i < ord)

{

int \*\*temp = new int\*[ord - 1];

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

temp[j] = new int[ord - 1];

for (int j = 0; j < ord - 1; j++)

{

for (int k = 0; k <= ord - 1; k++)

{

if (k < i)

{

temp[j][k] = arr[j + 1][k];

}

if (k > i)

temp[j][k - 1] = arr[j + 1][k];

}

}

tmp += arr[0][i] \* pow(-1, i) \* Determinant(temp, ord - 1);

i++;

}

return tmp;

}

}

int Functions::Addition(int \*\*arr, int ord, int ind1, int ind2)

{

if (ord == 1)

return arr[0][0];

else

{

int \*\*temp = new int\*[ord - 1];

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

temp[i] = new int[ord - 1];

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

{

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

{

if (i < ind1)

{

if (j < ind2)

temp[i][j] = arr[i][j];

if (j > ind2)

temp[i][j - 1] = arr[i][j];

}

if (i > ind1)

{

if (j < ind2)

temp[i - 1][j] = arr[i][j];

if (j > ind2)

temp[i - 1][j - 1] = arr[i][j];

}

}

}

return Determinant(temp, ord - 1);

}

}

Mat\* Functions::Inverse(Mat\* matrix)

{

int ord = matrix->rows;

Mat\* res = new Mat;

res = InitArr(res, ord, ord);

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

{

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

{

res->data[i][j] = pow(-1, i+j) \* Addition(matrix->data, ord, i, j);

}

}

return Transpose(res);

}

int Functions::Convertation(String^ str)

{

int x;

try

{

x = Convert::ToInt32(str);

}

catch (FormatException^)

{

Exception^ x = gcnew Exception("Введены не все значения");

throw x;

return NULL;

}

x = Convert::ToInt32(str);

return x;

}

Mat\* Functions::Power(Mat\* matrix, int n)

{

if (n < 0)

{

MessageBox::Show("Возведение в отрицательную степень невозможно");

return matrix;

}

if (n == 0)

{

for (int i = 0; i < matrix->rows; i++)

{

for (int j = 0; j < matrix->colons; j++)

{

if (i == j)

matrix->data[i][j] = 1;

else

matrix->data[i][j] = 0;

}

}

return matrix;

}

int i = 1;

Mat\* M = matrix;

while (i < n)

{

M = Mult(M, matrix);

i++;

}

return M;

}

Список использованной литературы.

1. Жевняк Р. М., Карпук А. А. Ж45 Высшая математика Ч.1 [учебное пособие для втузов] – Мн.: Выш.шк., 1984. – 223с. ил.
2. Курош А. Г. Курс высшей алгебры – изд. «Наука», Москва 1968.