Федеральное агентство Российской Федерации по образованию

ГОУ ВПО «Тульский государственный университет»

Кафедра электронных вычислительных машин

Лабораторная работа № 1

Языки программирования. «Классы»

Выполнил: студент

группы 230701

Гребецкая Е.В.

Проверил: Савин Н.И.

Тула 2010

# Введение

**Задание**

Реализовать класс «вещественная матрица», который позволяет осуществлять основные операции с вещественными прямоугольными матрицами.

# **Техническое задание**

Написать программу, реализующую тип данных «вещественная матрица» и работу с ними. Класс должен реализовывать следующие операции над матрицами:

• сложение, вычитание, умножение, деление (+, -, \*, /)

• операции сравнения на равенство/неравенство;

• операции вычисления транспонированной матрицы

## Требования к разработке

Microsoft Visual Studio 2008 или новее.

Наличие версий операционной системы Microsoft Windows 98, Windows (Me), Windows 2000, Windows ХР, Windows Vista, Windows 7.

## Требования к эксплуатации

Microsoft Windows 98/Me/2000/ХР/Vista/7.

## Технические характеристики

Время и эффективность работы программы зависит от эксплуатационных характеристик пользователя.

## Эксплуатационные характеристики

Для того чтобы программа работала быстро и эффективно не требуется мощных компьютеров и современных операционных систем. Ниже приведены минимальные параметры компьютера, которые нужны для работы:

* Центральный процессор: Intel Pentium 166 MHz (рекомендуется P2 400 MHz)
* Оперативная память: 128 Mb (рекомендуемая 256 Mb)
* Памяти на жестком диске: 1 Мб.
* Монитор с разрешением VGA и выше

## Входные параметры:

элементы матрицы вводятся с клавиатуры

## Выходные параметры:

Вывод в консоль результата выполнения программы.

# Теоретические положения

## транспонированная вещественная матрица

## Справка по программным методам

Класс-это производный структурированный тип, введенный программистом на основе уже существующих типов. Механизм классов позволяет создавать типы в полном соответствии с принципами абстракции данных, т. е класс задает некоторую структурированную совокупность типизированных данных и позволяет определить набор операций над этими данными. Простейшим образом класс можно определить с помощью инструкции:

Ключ\_класса имя\_класса {список\_компонентов};

где ключ\_класса – одно из служебных слов class, struct, union, имя\_класса – произвольно выбираемый разработчиком идентификатор, список\_компонентов – определения и описания типизированных данных и принадлежащих классу функций.

Конструктор – специальная функция класса, которая вызывается автоматически при создании объекта типа класса. Имя конструктора совпадает с именем класса, не возвращает никакого результата, даже void. Компилятор гарантирует единственный вызов конструктора для одного объекта.

Деструктор – специальная функция класса, которая вызывается автоматически при уничтожении объекта. Имя деструктора совпадает с именем класса, перед которым ставится символ “~”. Компилятор гарантирует единственный вызов деструктора для одного объекта. Деструктор не может иметь параметров и поэтому не может быть перегружен.

При создании автоматической переменной деструктор вызывается автоматически при выходе из области видимости, то есть за рамки блока, в котором определена переменная. Для динамических переменных дела обстоят совсем по-другому, для освобождения занятой памяти используется оператор delete, который и вызывает деструктор.

## Справка по математическим методам:

Прямоугольная матрица – математический элемент, в котором в виде таблицы записываются числа, называемые элементами матрицы, которые образую т прямоугольную область. Элементы, расположенные в ряд по горизонтали, называются строками матрицы, по вертикали - столбцами матрицы.

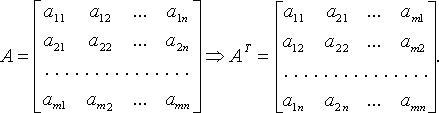
Действия с матрицами:

* Сложение: сложение матриц происходит поэлементно, например, элемент a11 складывается с элементом b11, a21 с b21 и т.д.
* Вычитание происходит поэлементно, аналогично сложению.
* Умножение матриц:

Произведением матрицы размером на матрицу размером называется матрица размером у которой:



* Транспонирование матрицы:



* Сравнение матриц: если элементы одной матрици равны элементам другой матрицы, то матрицы равны.

# Алгоритмические положения

В данной программе имеется файл matrix.cpp в котором в секции private представлены типы размеров матрицы и указателей, а в секции public констуктор класса, а также представлены функции, которые будут осуществляться при работе с этим классом.

Сначала срабатывает конструктор, выделяющий память под указатели и элементы.

Matrix::Matrix(int n, int m)

{int i,j;

Matrix::n=n;

Matrix::m=m;

a=new double\*[n]; // выделение памяти под указатели

for(i=0;i<n;i++)

a[i]=new double[m]; // выделение памяти под элемент

c=new double\*[n];

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

c[i]=new double[m];

-числа, однако число строк и число столбцов должно совпадать с прописанными в закрытой секции m и n.

{puts("\nВведите элементы матрицы:");

for(i=0;i<n;i++)

for(j=0;j<m;j++)

{cout<<"a["<<i<<"]["<<j<<"]=";

cin>>a[i][j];}

cout<<endl;}};

# Программная реализация

# 

# Тестирование

При тестировании программы выполнение программы прошло успешно.

# Приложение

# Код программы

#include "stdafx.h"

#include <iostream>

using namespace std;

class Matrix{

private:

int n,m;

## //размеры матрицы

double \*\*a, \*\*c;

### // указатели

public:

Matrix::Matrix(int n, int m)

{int i,j;

Matrix::n=n;

Matrix::m=m;

a=new double\*[n]; // выделение памяти под указатели

for(i=0;i<n;i++)

a[i]=new double[m]; // выделение памяти под элемент

c=new double\*[n];

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

c[i]=new double[m];

// Заполнение массива элементами с клавиатуры

{puts("\nВведите элементы матрицы:");

for(i=0;i<n;i++)

for(j=0;j<m;j++)

{cout<<"a["<<i<<"]["<<j<<"]=";

cin>>a[i][j];}

cout<<endl;}};

Matrix::~Matrix()

{for(int i=0;i<n;i++)

delete []a[i];// освобождение памяти

delete []a;};

//Операция присваивания

Matrix &Matrix::operator =(const Matrix &A)

{for(int i=0; i<n; i++)

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

a[i][j]=A.c[i][j];

return \*this;}

//Сложение матриц

Matrix & Matrix::operator +(Matrix &A)

{for(int i=0; i<n; i++)

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

c[i][j]=a[i][j]+A.a[i][j];

return \*this;}

//Вычитание матриц

Matrix & Matrix::operator -(Matrix &A)

{for(int i=0; i<n; i++)

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

c[i][j]=a[i][j]-A.a[i][j];

return \*this;}

// Умножение матриц

Matrix & Matrix::operator \*(Matrix &A)

{

double s=0;

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

{

int z=0;

while(z<A.m)

{

c[i][z]=0;

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

{

s=GetI(i,j)\*A.GetI(j,z);

c[i][z]+=s;

}

z++;

}

}

return \*this;

}

//Сравнение матриц

Matrix & Matrix::operator ==(Matrix &A)

{int f=0;

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

{for(int j=0; j<m; j++)

if(a[i][j]!=A.a[i][j]){ cout<<"\nМатрицы не равны\n";f=1; break;}

if(f==1) break;}

if(f==0) cout<<"\nМатрицы равны\n";

return \*this;}

//Транспонирование матрицы

void Matrix::Transp()

{c=new double\*[m];

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

c[i]=new double[n]; // захват памяти под элемент

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

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

c[j][i]=a[i][j];

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

delete []a[i];// освобождение памяти

delete []a;

a=new double\*[m];

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

a[i]=new double[n]; // захват памяти под элемент

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

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

a[i][j]=c[i][j];

Matrix::n=m;

Matrix::m=n;

};

void main()

{**}**