НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Автоматики и Вычислительной Техники

Кафедра Вычислительной Техники

**ЛАБОРАТОРНАЯ РАБОТА № 1**

по дисциплине «Программирование»

*Разработка классов, создание конструкторов и деструкторов. Использование статических членов класса.*

Группа: АБ-421

Студент: Власов А.И.

Вариант: №4

Преподаватель: Удовиченко А.В.

Новосибирск 2015

***Цель работы:*** Изучить структуру класса, механизм создания и использования, описание членов-данных класса и методов доступа к ним, возможность инициализации объектов класса с помощью конструкторов и уничтожение их с помощью деструкторов.

***Задание на лабораторную работу:*** Постpоить класс для pаботы с квадратными матрицами. Класс должен включать соответствующие поля: порядок, набор коэффициентов. Класс должен обеспечивать пpостейшие функции для pаботы с данными классами: транспонирование матрицы, расчет определителя, вывод матрицы в удобной форме и т.д.

Класс должен обеспечивать набор методов для работы с данными. Создать перегруженные конструкторы: конструктор с параметрами, конструктор копирования, конструктор по умолчанию. Создать деструктор для освобождения памяти. Посмотреть, как вызываются конструкторы и деструкторы.

Составить демонстрационную программу. Для реализации демонстрационной программы использовать отдельный модуль. Программу построить с использованием проекта. Посмотреть работу программы в отладчике, обратить внимание на представление данных. Построить пpогpамму без отладочной информации. Обратить внимание на размер программы.

**Листинг программы:**

**Matrix.h:**

#pragma once // защита от двойного подключения

class Matrix

{

private:

int const DIMENSION = 4; // размерность

int matrix[4][4]; // матрица

int trans\_matrix[4][4]; // транспонированная матрица

public:

Matrix::Matrix() // Конструктор

{}

Matrix::~Matrix() // Деструктор

{

}

void addRandMatrix(); // Генерирует матрицу TODO: cделать, чтобы размерность заносилась самостоятельно

void printMatrix(); // Печатает матрицу

int getIntMatrix(int i, int j); // Возвращает [i][j] элемент матрицы

void getTransMatrix(); // Возвращает транспонированную матрицу

void GetMatr(int \*\*mas, int \*\*p, int i, int j, int m); // Получение матрицы без i-й строки и j-го столбца

int determinant(int \*\*mas, int m); // Вычисляет определитель матрицы

// a3 = a1 + a2

Matrix operator +(Matrix a2); // Перегрузка операторов: суммирование матриц

void operator++(); // Перегрузка операторов: унарное сложение

void operator--(); // Перегрузка операторов: унарное вычитание

void multiplicMatrix(); // Перегрузка операторов: умножение матриц

void subtrMatrix(); // Перегрузка операторов: вычитание матриц

void divisionMatrix(); // Перегрузка операторов: деление матриц

};

#endif

**Matrix.cpp:**

#pragma once

#include <iostream>

#include <cstdlib>

#include <ctime>

#include "Matrix.h"

using namespace std;

void Matrix::addRandMatrix() // Генерирует матрицу TODO: cделать, чтобы размерность заносилась самостоятельно

{

srand(time(NULL)); // Рандом

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

{

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

{

matrix[i][j] = rand() % 100; // заносим случайное число от 0 до 99

}

}

cout << endl;

}

void Matrix::printMatrix() // Печатает матрицу

{

cout << "Ваша матрица: \n" << endl;

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

{

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

{

cout << matrix[i][j] << "\t";

}

cout << endl;

}

cout << endl << endl;

}

int Matrix::getIntMatrix(int i, int j) // Возвращает [i][j] элемент матрицы

{

return matrix[i][j];

}

void Matrix::getTransMatrix() // Возвращает транспонированную матрицу

{

cout << "Ваша транспонированная матрица: \n" << endl;

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

{

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

{

trans\_matrix[i][j] = matrix[j][i];

cout << trans\_matrix[i][j] << "\t";

}

cout << endl;

}

//Присвоим обычной матрице транспонированное значение

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

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

matrix[i][j] = trans\_matrix[i][j];

}

// Получение матрицы без i-й строки и j-го столбца

void Matrix::GetMatr(int \*\*mas, int \*\*p, int i, int j, int m) {

int ki, kj, di, dj;

di = 0;

for (ki = 0; ki < m - 1; ki++) { // проверка индекса строки

if (ki == i) di = 1;

dj = 0;

for (kj = 0; kj < m - 1; kj++) { // проверка индекса столбца

if (kj == j) dj = 1;

p[ki][kj] = mas[ki + di][kj + dj];

}

}

}

int Matrix::determinant(int \*\*mas, int m) { // Вычисляет определитель матрицы

int i, j, d, k, n;

int \*\*p;

p = new int\*[m];

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

p[i] = new int[m];

j = 0; d = 0;

k = 1; //(-1) в степени i

n = m - 1;

if (m<1)

cout << "Определитель вычислить невозможно!";

if (m == 1)

{

d = mas[0][0];

return(d);

}

if (m == 2)

{

d = mas[0][0] \* mas[1][1] - (mas[1][0] \* mas[0][1]);

return(d);

}

if (m>2)

{

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

{

GetMatr(mas, p, i, 0, m);

d = d + k \* mas[i][0] \* determinant(p, n);

k = -k;

}

}

return(d);

}

Matrix Matrix::operator+(Matrix a2)

{

Matrix result;

result.matrix[4][4] = matrix[4][4] + a2.matrix[4][4];

return result;

}

void Matrix::operator++()

{

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

{

matrix[i][j]++;

}

}

void Matrix::operator--()

{

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

{

matrix[i][j]--;

}

}

**Source.cpp:**

#include <iostream>

#include <clocale>

#include "Matrix.h"

#define boolean int // Небольшой костыль

#define false 0

#define true 1

using namespace std;

int main()

{

Matrix matrix;

setlocale(LC\_ALL, "Russian");

boolean finish = false;

boolean exit = false;

cout << "В настоящий момент Вы работаете со следующей матрицей:\n";

matrix.addRandMatrix(); // Инициализируем матрицу

matrix.printMatrix();

while (finish == false)

{

cout << "Выберите необходимый пункт меню, путем ввода числового значения:\n\n";

cout << "1. Сгенерировать матрицу заново\n";

cout << "2. Транспонировать матрицу\n";

cout << "3. Вычислить определитель\n";

cout << "4. Увеличить все элементы матрицы на единицу\n";

cout << "5. Уменьшить все элементы матрицы на единицу\n";

cout << "Введите любое другое число, чтобы закрыть программу...\n\n";

cout << ">> ";

int c = 0; // номер пункта меню

cin >> c;

cout << endl;

switch (c)

{

case 1:

system("cls");

exit = false;

while(exit == false)

{

matrix.addRandMatrix();

matrix.printMatrix();

char answer;

cout << "Продолжить работу (y) или сгенерировать повторно(n)?\n>> ";

cin >> answer;

if (answer == 'y')

{

finish = false; // для наглядности

exit = true;

system("cls");

matrix.printMatrix();

}

else

{

system("cls");

}

}

break;

case 2:

system("cls");

matrix.getTransMatrix(); // Получаем транспонированную матрицу

break;

case 3:

int \*\*mas;

mas = new int\*[4]; // размерность

for (int i = 0; i < 4; i++) {

mas[i] = new int[4];

for (int j = 0; j < 4; j++) {

mas[i][j] = matrix.getIntMatrix(i, j);

}

}

system("cls");

matrix.printMatrix();

cout << "Определитель матрицы равен: " << matrix.determinant(mas, 4) << endl << endl;

break;

case 4:

system("cls");

matrix.printMatrix();

cout << "\nПосле увеличения" << endl;

++matrix;

matrix.printMatrix();

break;

case 5:

cout << "Уменьшить все элементы матрицы на единицу" << endl;

matrix.printMatrix();

cout << "\nПосле уменьшения" << endl;

--matrix;

matrix.printMatrix();

break;

default:

return 0;

}

}

system("pause");

return 0;

}

**Все мои исходники вы можете найти по следующей ссылке:**

**github.com/whalemare/MatrixConsole**

Пример работы программы:

