**Лаболаторная работа №5**

**Реализация прототипа**

**Цель:** разработать документацию на ПП, разработать и отладить код программы на выбранном инструменте разработки, разработать план тестирования программы с определением значений параметров (качественных характеристик системы) и провести тестирование программы.

1. **Документация на программный продукт**

* **Архитектурная/проектная документация**

Нашей командой разработчиков был разработан программный продукт под названием «Матричный Калькулятор». В ходе разработки программного продуктка мы использовали пакет Miscrosoft Visual Studio 2013 и язык программирования С++ в частности, так как на данный момент у нас было больше всего опыта работы именно с этим пакетом, а также его функциональные возможностей было достаточно для выполнения ТЗ. Для разработки данного программного продукта мы использовали процедурную парадигму программирования. Выбор такого подхода к разработке программного продукта обусловлен тем, что наш нынешний уровень знаний не очень высок; тем, что проект является учебным, и тем, что такой подход полностью удовлетворяет ТЗ. В ходе разработки было решено не разбивать программный код на отдельные модули, так как его размер был небольшим.

* **Техническая документация**

В качестве техническое документации будет представлен код программы, содержащий кометарий, которые достаточно ясно дают понять, что делает код.

#include "stdafx.h"

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <fstream>

using namespace std;

int \*\*form\_matr\_ref(int &n, int &m);

int \*\*form\_matr(int n, int m);

int \*\*form\_matr\_sqr(int n);

void zapolnenie(int \*\*matr, int n, int m);

void print(int \*\*matr, int n, int m);

int \*\*multy\_chislo(int \*\*matr, int n, int m);

int \*\*multy\_matr(int \*\*multy, int \*\*matr1, int \*\*matr2, int n1, int m1, int n2, int m2);

int \*\*multy\_sqr(int \*\*matr, int n);

int \*\*sum\_matr(int \*\*sum, int \*\*matr1, int \*\*matr2, int n, int m);

int \*\*transp(int \*\*matr, int n, int m);

int det(int \*\*matr, int m);

void get\_matr(int \*\*mas, int \*\*p, int i, int j, int m);

int input();

bool error(int n);

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

ofstream fout("Calc.txt");

fout.close();

int operacia, n, m, k, l;

cout << "\tВас приветствует программа 'Матричный калькулятор'!";

cout << " Для взаимодействия\nс программой выберите действие и нажмите Enter." << endl;

while (true) {

cout << "\tВыберите операцию: " << endl;

cout << "1 - нахождение транспонированной матрицы" << endl;

cout << "2 - нахождение определителя (только для квадратных матриц)" << endl;

cout << "3 - возведение матрицы в квадрат (только для квадратных матриц)" << endl;

cout << "4 - возведение матрицы в степень (только для квадратных матриц)" << endl;

cout << "5 - умножение матрицы на число" << endl;

cout << "6 - сложение матриц (только для одинаковых матриц)" << endl;

cout << "7 - умножение матрицы A на матрицу B" << endl;

cout << "8 - справка" << endl;

cout << "0 - выход" << endl << endl;

cout << "\t\tВаш выбор: ";

operacia = input();

if (operacia == 1)

{

ofstream fout("Calc.txt", ios::app);

cout << "Введите размерность матрицы" << endl;

int \*\*matr = form\_matr\_ref(n, m); cout << endl;

if (matr == NULL) continue;

zapolnenie(matr, n, m); cout << endl;

cout << "Ваша матрица: " << endl;

fout << "Ваша матрица: " << endl;

print(matr, n, m);

matr = transp(matr, n, m);

cout << "Транспонированная матрица: " << endl;

fout << "Транспонированная матрица: " << endl;

print(matr, m, n); cout << endl; fout << endl;

fout.close();

}

else if (operacia == 2)

{

ofstream fout("Calc.txt", ios::app);

cout << "Только для квадратных матриц" << endl;

cout << "Введите размерность матрицы NxN: "; n = input();

if (error(n)) continue;

int \*\*matr = form\_matr\_sqr(n); cout << endl;

zapolnenie(matr, n, n);

cout << "Ваша матрица: " << endl;

fout << "Ваша матрица: " << endl;

print(matr, n, n);

int a = det(matr, n);

cout << "Определитель: " << a << endl << endl;

fout << "Определитель: " << a << endl << endl;

fout.close();

}

else if (operacia == 3)

{

ofstream fout("Calc.txt", ios::app);

cout << "Только для квадратных матриц" << endl;

cout << "Введите размерность матрицы NxN: "; n = input();

if (error(n)) continue;

int \*\*matr = form\_matr\_sqr(n); cout << endl;

zapolnenie(matr, n, n);

cout << "Ваша матрица: " << endl;

fout << "Ваша матрица: " << endl;

print(matr, n, n);

matr = multy\_sqr(matr, n);

cout << "Результат возведение матрицы в квадрат: " << endl;

fout << "Результат возведение матрицы в квадрат: " << endl;

print(matr, n, n); cout << endl; fout << endl;

fout.close();

}

else if (operacia == 4)

{

ofstream fout("Calc.txt", ios::app);

int exponent;

cout << "Только для квадратных матриц" << endl;

cout << "Введите размерность матрицы NxN: "; n = input();

if (error(n)) continue;

cout << "Введите степень, в которую нужно возвести матрицу: ";

exponent = input();

int \*\*matr = form\_matr\_sqr(n); cout << endl;

zapolnenie(matr, n, n);

cout << "Ваша матрица: " << endl;

fout << "Ваша матрица: " << endl;

print(matr, n, n);

if (exponent > 0)

{

if (exponent == 1)

{

cout << "Результат возведения матрицы в степень " << exponent << " :" << endl;

fout << "Результат возведения матрицы в степень " << exponent << " :" << endl;

print(matr, n, n);

}

else if (exponent == 2)

{

matr = multy\_sqr(matr, n);

cout << "Результат возведения матрицы в степень " << exponent << " :" << endl;

fout << "Результат возведения матрицы в степень " << exponent << " :" << endl;

print(matr, n, n);

}

else

{

int \*\*matr\_e = form\_matr\_sqr(n);

int \*\*temp1 = form\_matr\_sqr(n);

temp1 = matr;

temp1 = multy\_sqr(temp1, n);

for (int i = 0; i < exponent - 2; i++)

{

matr\_e = multy\_matr(matr\_e, temp1, matr, n, n, n, n);

temp1 = matr\_e;

}

cout << "Результат возведения матрицы в степень " << exponent << ":" << endl;

fout << "Результат возведения матрицы в степень " << exponent << " :" << endl;

print(temp1, n, n);

}

}

else {

cout << "\tОшибка! Возведение в степень меньше еденицы невозможно. " << endl;

fout << "\tОшибка! Возведение в степень меньше еденицы невозможно. " << endl;

}

cout << endl;

fout << endl;

fout.close();

}

else if (operacia == 5)

{

ofstream fout("Calc.txt", ios::app);

cout << "Введите размерность матрицы" << endl;

int \*\*matr = form\_matr\_ref(n, m); cout << endl;

if (matr == NULL) continue;

zapolnenie(matr, n, m); cout << endl;

cout << "Ваша матрица: " << endl;

fout << "Ваша матрица: " << endl;

print(matr, n, m);

matr = multy\_chislo(matr, n, m);

cout << "Результат умножения матрицы на число: " << endl;

fout << "Результат умножения матрицы на число: " << endl;

print(matr, n, m); cout << endl; fout << endl;

fout.close();

}

else if (operacia == 6)

{

ofstream fout("Calc.txt", ios::app);

cout << "Только для одинаковых матриц" << endl;

cout << "Введите размерность матрицы NxM: ";

n = input();

m = input();

if (error(n) || error(m)) continue;

int \*\*matr1 = form\_matr(n, m); cout << endl;

zapolnenie(matr1, n, m); cout << endl;

cout << "A:" << endl;

fout << "A:" << endl;

print(matr1, n, m);

int \*\*matr2 = form\_matr(n, m); cout << endl;

zapolnenie(matr2, n, m); cout << endl;

cout << "B:" << endl;

fout << "B:" << endl;

print(matr2, n, m);

int \*\*sum = form\_matr(n, m); cout << endl;

sum = sum\_matr(sum, matr1, matr2, n, m);

cout << "Результат сложения матрицы А и матрицы В: " << endl;

fout << "Результат сложения матрицы А и матрицы В: " << endl;

print(sum, n, m); cout << endl; fout << endl;

fout.close();

}

else if (operacia == 7)

{

ofstream fout("Calc.txt", ios::app);

cout << " Только для матриц, у которых количество столбцев матрицы А равно количеству\nстрок матрицы В" << endl;

int \*\*matr1 = form\_matr\_ref(n, m); cout << endl;

if (matr1 == NULL) continue;

zapolnenie(matr1, n, m);

cout << "A:" << endl;

fout << "A:" << endl;

print(matr1, n, m);

int \*\*matr2 = form\_matr\_ref(k, l); cout << endl;

if (matr2 == NULL) continue;

zapolnenie(matr2, k, l);

cout << "B:" << endl;

fout << "B:" << endl;

print(matr2, k, l);

if (m == k) //

{

int \*\*multy = form\_matr(n, l); cout << endl;

multy = multy\_matr(multy, matr1, matr2, n, m, k, l);

cout << "Результат умножения матрицы А на матрицу В: " << endl;

fout << "Результат умножения матрицы А на матрицу В: " << endl;

print(multy, n, l);

}

else {

cout << "\tОшибка! Умножение невозможно. " << endl;

fout << "\tОшибка! Умножение невозможно. " << endl;

}

cout << endl; fout << endl;

fout.close();

}

else if (operacia == 8) {

cout << "\t\tОбщие сведенья" << endl;

cout << "\tПри вводе размерности матрицы NxM Вам нужно вначале ввести"

<< "количество\nстрок N и нажать Enter, a затем количество столбцев M и нажать Enter."

<< " Для\nквадратных матриц размерность N вводится аналогично предыдущему пункту."

<< "Елементы матрицы вводятся построчно аналогично вводу размерности. "

<< "Если при вводе\nразмерности матрицы вы введете отрицательное число, нуль или "

<< "какой-либо символ, отличный от числа, то вы получите сообщение об ошибке"

<< "(если при вводе\nразмерности вы укажите вещестенное число, то программа округлит это число "

<< "до\nцелого в меньшую сторону). Если при вводе елементов матрицы вы введете\nкакой-либо символ, "

<< "отличный от числа, вы получите сообщение об ошибке." << endl;

cout << "\t\tИнформация о выполнении операций с матрицами" << endl;

cout << "\t1.Для нахождения транспонированное матрицы введите данные с учетом\nвышеизложенных инструкций" << endl;

cout << "\t2.Для нахождения определителя матрицы введите данные с учетом\nвышеизложенных инструкций. "

<< "Cтоит заметить, что вычисление определителя\nосуществимо только для квадратных матриц." << endl;

cout << "\t3.Для нахождения результата возведения матрицы в квадрат введите\nданные с учетом "

<< "вышеизложенных инструкций. Cтоит заметить, что вычисление\nопределителя осуществимо только"

<< "для квадратных матриц." << endl;

cout << "\t4.Для нахождения результата возведения матрицы в cтепень введите\nданные с учетом "

<< "вышеизложенных инструкций. Cтоит заметить, что вычисление\nопределителя осуществимо только"

<< "для квадратных матриц и для целого\nнеотрицательного показателя степени." << endl;

cout << "\t5.Для нахождения результата произведения матрицы и числа введите\nданные с учетом "

<< "вышеизложенных инструкций." << endl;

cout << "\t6.Для нахождения суммы двух матрицы введите данные с учетом\nвышеизложенных инструкций для "

<< "двух матриц. Cтоит заметить, что сложение матриц\nосуществимо только для одинаковых матриц." << endl;

cout << "\t7.Для нахождения результата произведения двух матрицы введите данные с\nучетом "

<< "вышеизложенных инструкций для двух матриц. Cтоит заметить, что операция\nпроизведения осуществима "

<< "только для матриц, у которых количество строк первой\nматрицы равно количеству столбцев"

<< " второй матрицы " << endl;

cout << endl;

}

else if (operacia == 0)

return 0;

else

{

cout << "Произошла ошибка. Повторите ввод: ";

input();

}

}

}

//Функция, которая запрашивает размерность матрицы у пользователя

//и формирует соответствующую матрицу

int \*\*form\_matr\_ref(int &n, int &m)

{

cout << "Введите количество строк: "; n = input();

cout << "Введите количество столбцев: "; m = input();

if (error(n) || error(m)) return NULL;

int \*\*matr = new int \*[n];

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

{

matr[i] = new int[m];

}

return matr;

}

//Функция формирования матрицы произвольного размера

int \*\*form\_matr(int n, int m)

{

int \*\*matr = new int \*[n];

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

{

matr[i] = new int[m];

}

return matr;

}

//Функция формирования квадратной матрицы

int \*\*form\_matr\_sqr(int n)

{

int \*\*matr = new int \*[n];

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

{

matr[i] = new int[n];

}

return matr;

}

//Функция заполенения матрицы

void zapolnenie(int \*\*matr, int n, int m)

{

cout << "Формируем матрицу:" << endl;

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

{

cout << "Введите элементы " << i + 1 << " строки:" << endl;

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

{

matr[i][j] = input();

}

}

}

//Функция вывода матрицы на экран

void print(int \*\*matr, int n, int m)

{

ofstream fout("Calc.txt", ios::app);

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

{

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

{

cout << setw(5) << matr[i][j];

fout << setw(5) << matr[i][j];

}

cout << endl; fout << endl;

}

fout.close();

}

//Функция нахождения результата умножения двух матриц

int \*\*multy\_chislo(int \*\*matr, int n, int m)

{

int x;

cout << "Введите число, на которое нужно умножить матрицу: ";

x = input();

cout << endl;

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

{

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

{

matr[i][j] = matr[i][j] \* x;

}

}

return matr;

}

//Функция нахождения произведения двух матриц

int \*\*multy\_matr(int \*\*multy, int \*\*matr1, int \*\*matr2, int n1, int m1, int n2, int m2)

{

int i = 0, j = 0;

for (int a = 0; a < n1; a++) // закрепили строку 1 матрицы

{

j = 0;

for (int d = 0; d < m2; d++) // закрепили столбец 2 матрицы

{

int sum = 0;

for (int b = 0, c = 0; b < m1, c < n2; b++, c++)

// умножаем строку 1 на столбец 2

{

sum += matr1[a][b] \* matr2[c][d];

}

multy[i][j] = sum;

j++;

}

i++;

}

return multy;

}

//Функция возведения матрицы в квадрат

int \*\*multy\_sqr(int \*\*matr, int n)

{

int \*\*sqr = form\_matr\_sqr(n);

int i = 0, j = 0;

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

{

j = 0;

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

{

int sum = 0;

for (int b = 0, c = 0; b < n, c < n; b++, c++)

{

sum += matr[a][b] \* matr[c][d];

}

sqr[i][j] = sum;

j++;

}

i++;

}

return sqr;

}

//Функция нахождения суммы двух матриц

int \*\*sum\_matr(int \*\*sum, int \*\*matr1, int \*\*matr2, int n, int m)

{

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

{

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

{

sum[i][j] = matr1[i][j] + matr2[i][j];

}

}

return sum;

}

//Функция нахождения транспонированой матрицы

int \*\*transp(int \*\*matr, int n, int m)

{

int \*\*temp = form\_matr(m, n);

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

{

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

{

temp[j][i] = matr[i][j];

}

}

return temp;

}

//Функция вычисления определителя

int det(int \*\*matr, int m)

{

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 << "\tОпределитель вычислить невозможно!";

if (m == 1)

{

d = matr[0][0];

return(d);

}

if (m == 2)

{

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

return(d);

}

if (m > 2)

{

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

{

get\_matr(matr, p, i, 0, m);

//cout << matr[i][j] << endl;

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

k = -k;

}

}

return(d);

}

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

//Функция используется внутри функции det()

void get\_matr(int \*\*matr, 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] = matr[ki + di][kj + dj];

}

}

}

//Функция проверки корректности ввода

int input()

{

while (true)

{

int val;

cin >> val;

if (cin.fail())

{

cin.clear();

cin.ignore(32767, '\n');

cout << "Произошла ошибка. Повторите ввод: ";

}

else

{

cin.ignore(32767, '\n');

return val;

}

}

}

//Функция вывода ошибки о некорректном вводе

//размерности матриц

bool error(int n) {

if (n < 1) {

cout << "Ошибка ввода! Введена некорректная размерность матрицы!" << endl;

return 1;

}

else return 0;

}

* **Пользовательская документация (руководство пользователя)**

Руководство пользователя выполнено со ГОСТ 19.505-79, ГОСТ 19.105-77, ГОСТ 19.105-78.

1. Назначение программы

Функциональное назначение

Функциональным назначением программы является предоставление пользователю возможности выполнять математические операции с матрицами.

Эксплутационное назначение

Программа предназначена для студентов, которым нужно призвести действия над матрицами, а также программистов, математиков, физиков для сокращения времени расчетов и рационализации времени работы. Чтобы использовать программ, пользователь должен быть знаком с понятием матрицы и действиями, которые можно выполнить с ними.

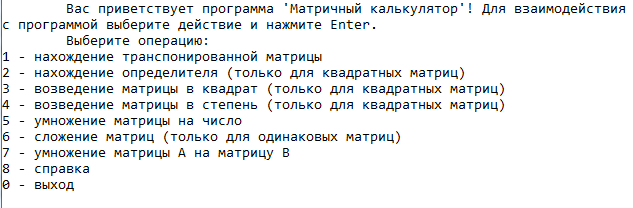
Состав функций

1. Ввод исходных данных
2. Вывод обработанных данных
3. Сохранение обработанных данных в текстовый файл
4. Условия выполненям программы

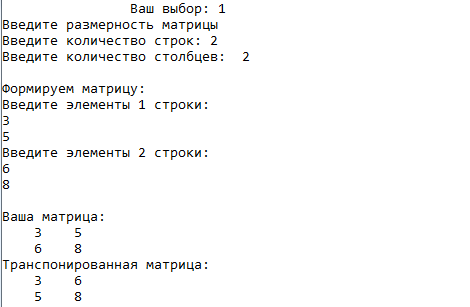
Из-за простоты программного продукта, для его выполнения подойдет любой компьютер из тех, что были выпущены за последние 20 лет. Данный программный продукт подходит для всех операционных систем.

1. Выполнение программы

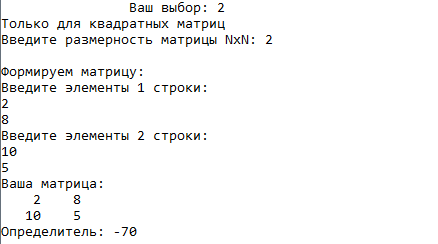
Чтобы начать использовать программный продукт, требуется открыть файл Calculator.exe и изучить меню программы.



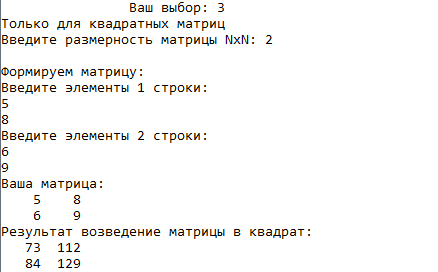
1. Если Вам нужно найти транспонированную матрицу, то нажмите на клавиатуре цифру 1, а затем Enter. Далее Вам будет предложенно ввести количество строк и столбцев матрицы, а затем элементы построчно. Ввод данных происходит точно таким же образом, как и при взаимодействии с меню. Если в результате ввода вы введете какое-либо значение, отлично от числа, то вы получите сообщение об ошибке, и вам нужно будет повторить ввод. Также если при вводе размерности матрицы вы укажите число, меньше еденицы, то вы получите сообщение об ошибке, и программа вернется в главное меню (если при вводе размерности вы укажите вещестенное число, то программа округлит это число до целого в меньшую сторону). В конечном итоге вы увидите изначальную матрицу и транспонированную матрицу.



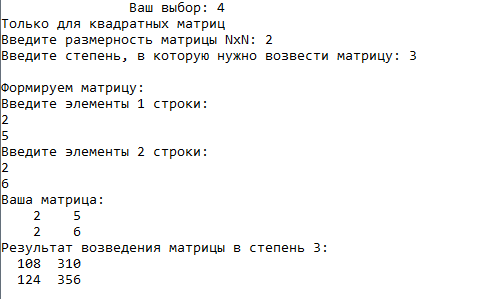
1. Если Вам нужно найти определитель матрицы, то нажмите на клавиатуре цифру 2, а затем Enter. Далее вам будет предложенно ввести размерность матрицы (стоит заметить, что вычисление определителя осуществимо только для квадратных матриц). Ввод исходных данных происходит аналогично первому пункту. Если в результате ввода вы введете какое-либо значение, отличное от числа, то вы получите сообщение об ошибке, и вам нужно будет повторить ввод. Также если при вводе размерности матрицы вы укажите число, меньше еденицы, то вы получите сообщение об ошибке, и программа вернется в главное меню (если при вводе размерности вы укажите вещестенное число, то программа округлит это число до целого в меньшую сторону). В конечном итоге вы увидите изначальную матрицу и ее определитель.



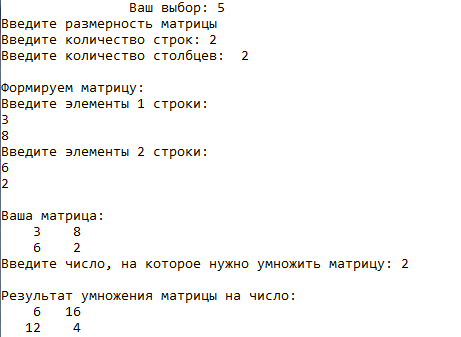
1. Если Вам нужно возвести матрицу в квадрат, то нажмите на клавиатуре цифру 3, а затем Enter. Далее вам будет предложенно ввести размерность матрицы (стоит заметить, что возведение матрицы в квадрат осуществимо только для квадратных матриц). Ввод исходных данных происходит аналогично первому пункту. Если в результате ввода вы введете какое-либо значение, отличное от числа, то вы получите сообщение об ошибке, и вам нужно будет повторить ввод. Также если при вводе размерности матрицы вы укажите число, меньше еденицы, то вы получите сообщение об ошибке, и программа вернется в главное меню (если при вводе размерности вы укажите вещестенное число, то программа округлит это число до целого в меньшую сторону). В конечном итоге вы увидите изначальную матрицу и изначальную матрицу, возведенную в квадрат.



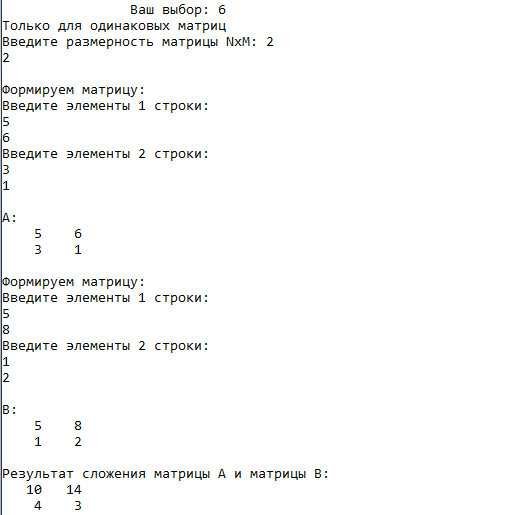
1. Если Вам нужно возвести матрицу в степень, то нажмите на клавиатуре цифру 4, а затем Enter. Далее вам будет предложенно ввести размерность матрицы и искомую степень (стоит заметить, что возведение матрицы в степент осуществимо только для квадратных матриц и для целого неотрицательного показателя степени). Ввод исходных данных происходит аналогично первому пункту. Если в результате ввода вы введете какое-либо значение, отличное от числа, то вы получите сообщение об ошибке, и вам нужно будет повторить ввод. Также если при вводе размерности матрицы или искомой степени вы укажите число, меньше еденицы, то вы получите сообщение об ошибке, и программа вернется в главное меню (если при вводе размерности вы укажите вещестенное число, то программа округлит это число до целого в меньшую сторону). В конечном итоге вы увидите изначальную матрицу и изначальную матрицу, возведенную в искомую степень.



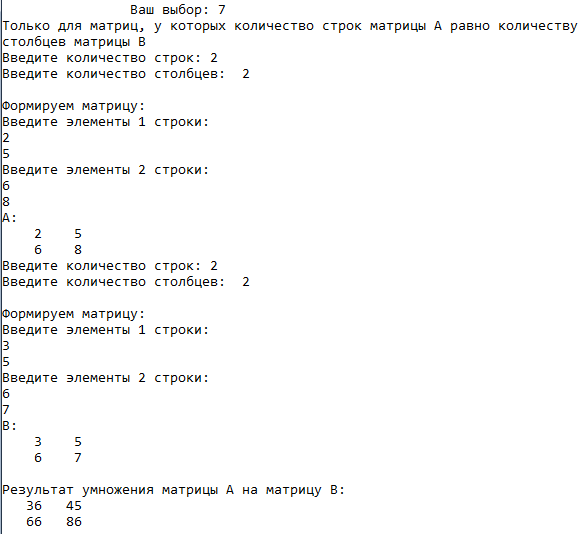
1. Если Вам нужно умножить матрицу на число, то нажмите на клавиатуре цифру 5, а затем Enter. Далее вам будет предложенно ввести размерность матрицы. Ввод исходных данных происходит аналогично первому пункту. Если в результате ввода вы введете какое-либо значение, отличное от числа, то вы получите сообщение об ошибке, и вам нужно будет повторить ввод. Также если при вводе размерности матрицы вы укажите число, меньше еденицы, то вы получите сообщение об ошибке, и программа вернется в главное меню (если при вводе размерности вы укажите вещестенное число, то программа округлит это число до целого в меньшую сторону). В конечном итоге вы увидите изначальную матрицу и произведение изначальной матрицы и числа.



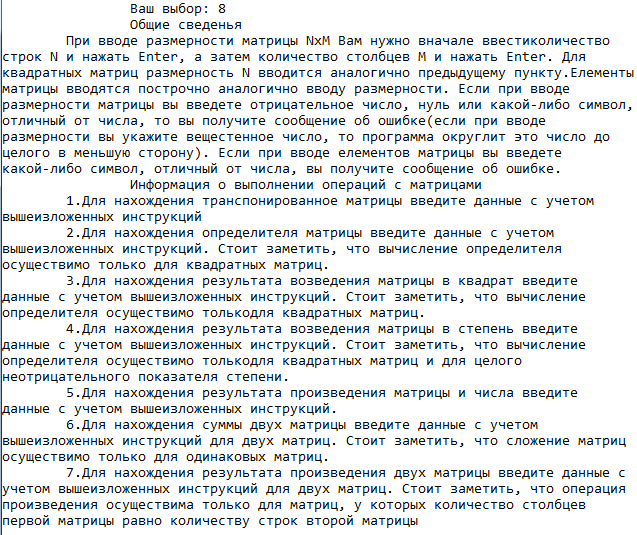
1. Если Вам нужно найти сумму двух матриц, то нажмите на клавиатуре цифру 6, а затем Enter. Далее вам будет предложенно ввести размерность первой матрицы и ее элементы, а затем размерность второй матрицы и ее елементы (стоит заметить, что сложение матриц осуществимо только для одинаковых матриц). Ввод исходных данных происходит аналогично первому пункту. Если в результате ввода вы введете какое-либо значение, отличное от числа, то вы получите сообщение об ошибке, и вам нужно будет повторить ввод. Также если при вводе размерности матрицы вы укажите число, меньше еденицы, то вы получите сообщение об ошибке, и программа вернется в главное меню (если при вводе размерности вы укажите вещестенное число, то программа округлит это число до целого в меньшую сторону). В конечном итоге вы увидите результат сложения двух матриц.



1. Если Вам нужно найти произведение двух матриц, то нажмите на клавиатуре цифру 7, а затем Enter. Далее вам будет предложенно ввести размерность первой матрицы и ее элементы, а затем размерность второй матрицы и ее елементы (стоит заметить, что умножение матриц осуществимо только матриц, у которых количество столбцев матрицы А равно количеству строк матрицы В). Ввод исходных данных происходит аналогично первому пункту. Если в результате ввода вы введете какое-либо значение, отличное от числа, то вы получите сообщение об ошибке, и вам нужно будет повторить ввод. Также если при вводе размерности матрицы вы укажите число, меньше еденицы, или введете недопустимую размерность для умножения матриц, то вы получите сообщение об ошибке, и программа вернется в главное меню (если при вводе размерности вы укажите вещестенное число, то программа округлит это число до целого в меньшую сторону). В конечном итоге вы увидите результат произведения двух матриц.



1. Если Вам нужно прочесть краткое руководство пользователя, то нажмите на клавиатуре цифру 8, а затем Enter.



1. Если Вам нужно выйти из программы, то нажмите на клавиатуре цифру 0, а затем Enter. Если в результате ввода вы введете какое-либо значение, отличное от числа, то вы получите сообщение об ошибке, и вам нужно будет повторить ввод. После этого программа завершит свою работу.
2. **Разработка и отладка кода программы с использованием инструментов Microsft Visual Studio 2013**

В ходе работы над проектом нашей командой разработчиков с помощью пакета Microsft Visual Studio 2013 был разработан и отлажен программный код. Для отладки мы использовали встроенный в Microsft Visual Studio 2013 отладчик и другие инструменты этой среды.

1. **Разработка плана тестирования программы и проведения тестрования**

**План тестирования программы**

В ходе этапа тестрования планируется постестировать программный продукт «Матричный Калькулятор», который является консольным приложением для систем Microsoft Windows.

В ходе тестирования необходимо протестировать все функции программы, а именно: функцию формирования пустого массива для последующего его заполнения, функцию заполнения массива (матрицы) с клавиатуры, функцию печати массива (матрицы) на экран, функцию произведения матрицы и числа, функцию произведения матриц, функцию возведения матрицы в квадрат, функцию сложения матриц, функцию транспонирования матриц, функцию нахождения определителя и вспомогтельную для нее функцию получения матрицы без i-й строки и j-го столбца, функцию проверки корректности ввода, а ткаже правильность выполнения операций калькулятора в целом. Также необходимо провести тестирование работы всех компонентов целиком; провести тестирование интерфейса пользователя.

Так как разрабатываемый нами программный продукт является учебным проектом небольшого размера, который не может нести какие-либо риски, некоторые из видов тестов можно опустить. Также из-за небольшого размера программного продукта нет необходимо в специальных программах для проведения тестов, достаточно будет ручного тестирования разработчиками. Учитывая все это, было решено провести тестирование классов эквивалентности, тестирование черного ящика и тестирование удобства пользователя.

Согласно нашему плану, тестирование должно проходить после реализации какого-либо рабочего компонента программного продукта, а также в самом конце разработки, когда на выходе будет уже готовый программный продукт.

Критерием начала тестирования является завершенность разработки части требуемого функционала или всего требуемого функционала, а критерием окончания тестирования является состояние программного продукта, при котором все найденные ошибки были устранены, и при котором он стал удовлетворять требования ТЗ и критерии качества.

**Тестирование класснов эквивалентностей**

 Это техника, при которой мы разделяем функционал (часто диапазон возможных вводимых значений) на группы эквивалентных по своему влиянию на систему значений. Такое разделение помогает убедиться в правильном функционировании целой системы — одного класса эквивалентности, проверив только один элемент этой группы.

Признаки эквивалентности тестов:

* направлены на поиск одной и той же ошибки;
* если один из тестов обнаруживает ошибку, другие скорее всего, тоже её обнаружат;
* если один из тестов не обнаруживает ошибку, другие, скорее всего, тоже её не обнаружат;
* тесты используют схожие наборы входных данных;
* для выполнения тестов мы совершаем одни и те же операции;
* тесты генерируют одинаковые выходные данные или приводят приложение в одно и то же состояние;
* все тесты приводят к срабатыванию одного и того же блока обработки ошибок;
* ни один из тестов не приводит к срабатыванию блока обработки ошибок

*Алгоритм использования:*

1. Определить классы эквивалентности.
2. Выбрать одного представителя от каждого класса эквивалентности.
3. Выполнени тестов.

**Тестирование «черного ящика»**

Тестирование **«черного ящика»** — это стратегия или метод тестирования, базируется только лишь на тестировании по функциональной спецификации и требованиям, при этом не смотря во внутреннюю структуру кода и без доступа к базе данных. Фактически мы знаем какой должен быть результат при определенном наборе данных, которые подаються на вход. Результат проверяем с юзер интерфейса на уровне простого пользователя.

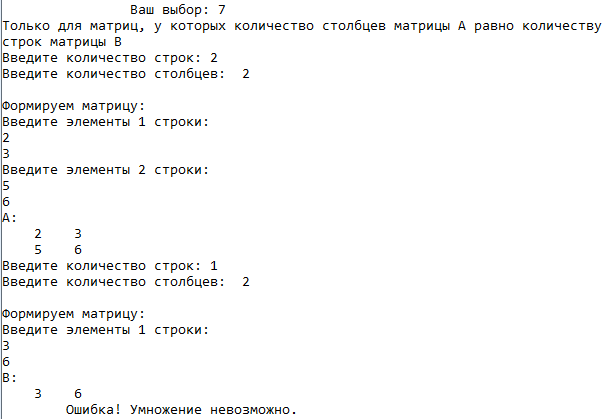
*Приемы тестирования черного ящика:*

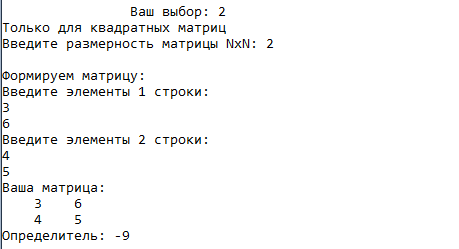
1. Эквивалентное разбиение.
2. Анализ граничных значений.
3. Анализ причино-следственных связей.
4. Предположение об ошибке.

**Результаты тестирования**

В результате тестирования программного продукта с помощью тестирования классов эквивалентности была выявлена и устранена ошибка, из-за который при вводе цифрового значения размерности матрицы пользователь вводил значение, меньшее 1, программа завершала работу с критической ошибкой. После исправления этой ошибки, программа в подобной ситуации сообщает о некорректном вводе и выходит в главное меню.

В проведения тестирования черного ящика были протестированы операции, выполняемые над матрицами. Ошибок при выполнении операций не было обнаруженно. Вот некоторые из результатов тестирования:





Также в результате тестирования был усовершенствован интерфейс пользователя, а именно: его понятность, удобность, простота.

**Вывод:** выполнив лабораторную работу, мы разработали документацию на ПП, разработали и отладили код программы с помощью пакета Miscrosoft Visual Studio, разработали план тестирования программы с определением значений параметров и провели тестирование программы.