ГУАП

КАФЕДРА № 53

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ассистент |  |  |  | С. Ю. Гуков |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| №4.2. РАЗРАБОТКА АЛГОРИТМА ИСЧЕРПЫВАЮЩЕГО ПОИСКА (ПОЛНОГО ПЕРЕБОРА) |
| Вариант 6 (15) |
| по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 5136 |  | , |  | Д. Л. Мягков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

Оглавление:

[Цель. 3](#__RefHeading___Toc11772_844291809)

[Индивидуальное задание. 4](#__RefHeading___Toc11774_844291809)

[Описание алгоритма 5](#__RefHeading___Toc11776_844291809)

[Листинг программы. 8](#__RefHeading___Toc11778_844291809)

[Оценка времени и сложности 29](#__RefHeading___Toc3436_902736657)

[Вывод. 30](#__RefHeading___Toc11780_844291809)

# Цель.

Реализовать алгоритм на языке C/С++, выполняющий поставленную задачу. Вариант задания, пример входных и выходных данных представлен в таблице 8. Глобальные параметры использовать запрещено; допустимо использование дополнительных функций.

* Разработанный алгоритм должен быть реализован в виде цельной программной функции (или нескольких функций) так, чтобы мог быть многократно применением с различными исходными данными и при этом не включал команды, не относящиеся к решаемой задаче, например, ввод и вывод исходных данных на консоль или в файл.
* Произвести теоретическую оценку количества используемых операций разработанного алгоритма.
* Произвести экспериментальную проверку времени работы разработанного алгоритма, определив его класс сложности для среднего случая. Измерить среднее время для Test\_Count повторений при различных размерностях входных данных.

# Индивидуальное задание.

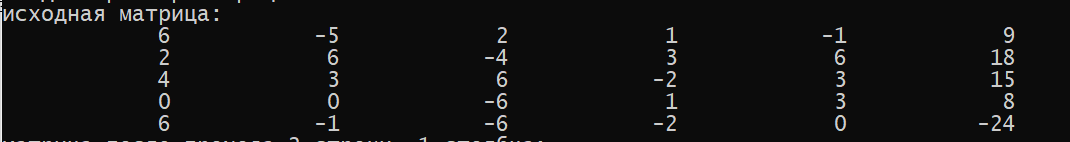
## 

Рисунок 1. Индивидуальное задание

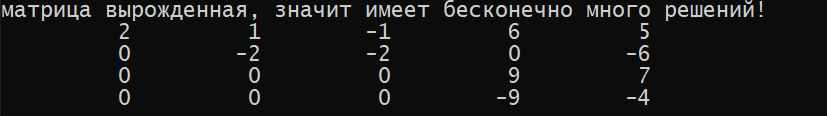
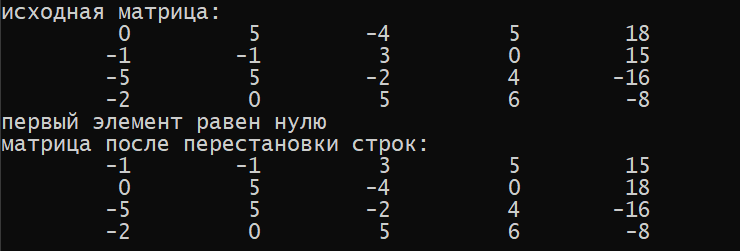
# Описание алгоритма

Создается двумерный массив, заполняется числами.

Метод гаусса. Суть заключается в составлении «лестницы» из нулей и последующем поиске значений каждой из x.

Матрица передается алгоритму в таком виде: 

Дальше составляется «лестница» из нулей. Берется элемент (0,0), если он равен нулю, то ищется такой индекс х, чтобы элемент (0,х) не был равен нулю. Если таких элементов нет, то матрица вырожденная, так как ее определитель равен нулю, значит она имеет бесконечно много решений. И такие матрицы я решил не брать.

Если нашелся такой элемент, то за базовый элемент мы берем его и переставляем строки местами. 

После этого мы начинаем цикл со строки с индексом равным индексу базового элемента (x, y + 1), где х – номер столбца, y – номер строки. (в данном случае базовый элемент (0,0), значит начинаем цикл с (0,1). Потом базовый элемент станет (1,1), значит цикл начнется с (1,2)). Инициализируются две переменные base = matrix[k][k] и element\_i = matrix[i][k], где k,k индекс базового элемента, а i – индекс строки. Если element\_i равен нулю, то мы ничего не делаем, потому что наша задача получить в начале этой строки ноль.

Если не равен, то :

1. если он делится на base без остатка, то мы инициализируем coefficient = element\_i / base и запускаем цикл, где пробегаемся по базовой строке и данной строке и каждый элемент базовой строки мы умножаем на коэффициент и на -1, а потом складываем эти две строки и результат записываем в данную строку.
2. Если не делится без остатка, то проверяем обратное, делится ли base на element\_i без остатка. Если да, то coefficient = base / element\_i, и запускаем цикл, где пробегаемся по базовой строке и данной строке и каждый элемент данной строки мы умножаем на коэффициент, каждый базовый элемент умножаем на -1, а потом складываем эти две строки и результат записываем в данную строку.
3. Если они не делятся без остатка друг на друга, то просто пробегаемся по строкам и каждый элемент базовой строки умножаем на element\_i (первый элемент данной строки), а каждый элемент данной строки умножаем на base (первый элемент базовой строки). Потом умножаем базовую строку на -1 и складываем с данной строкой.

Потом вызывается функция decrease, которая находит общий делитель всех элементов в строке и делит каждый элемент на это значение. Таким образом, мы каждый раз уменьшаем числа в нашей матрице, до максимально маленьких, чтобы не было переполнения типа.

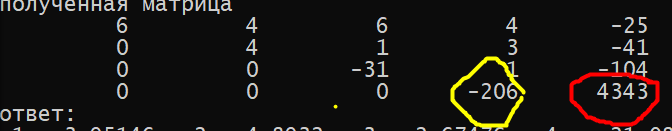
Потом базовый элемент сдвигается и его индекс становится 1,1. Мы снова пробегаемся по строкам, которые лежат ниже базовой строки.

Переносим базовый элемент, пока не будет получена лестница из нулей.

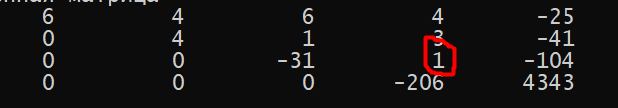
Потом вызывается функция decision. В ней запускается первый цикл, который идет от последней строки до первой. В нем мы инициализируем temp = 0, потом запускаем второй цикл, который пробегается от элемента с индексом на 2 меньше длины строки до индекса строки. То есть в последней строке, которая имеет вид 

Он пробежится от -14575 до -14575, то есть будет всего одна итерация. В следующий раз, будет две итерации, так как строка будет иметь вид 

Каждую итерацию temp складывается с самим собой и matrix[i][j] \* matrix\_x[j], где matrix\_x – вектор x, который изначально заполнен нулями и имеет длину равную количеству строк в матрице. Потом temp = matrix[i][matrix[i].size() - 1] – temp, и matrix\_x[i] = temp / matrix[i][i].

То есть, мы сначала делим красное на желтое и записываем результат в матрицу х, как значение х4. 



После этого мы переходим к третьей строке и умножаем х4 на красное,  потом из последнего столбца вычитаем это и делим на -31. Получаем х3.



И таким образом мы находим все х.

# Листинг программы.

Листинг lab\_4\_1.cpp:

#include <iostream>// библиотека

#include <iomanip>

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include "main\_header.h"

#include "string"

#include "lab\_4\_1\_func.h"

using std::endl;

void lab\_4\_1()

{

setlocale(0, ""); // поддержка кириллицы в консоли (вывод)

SetConsoleCP(1251); // поддержка кириллицы в консоли (ввод)

SetConsoleOutputCP(1251); // поддержка кириллицы в консоли (ввод)

unsigned int N, size;

double average\_time = 0;

cout << "Введите количество измерений: ";

N = get\_number<int>();

cout << "Введите размер матриц или введите ноль, чтобы посмотреть время: ";

size = get\_number<int>();

if (size == 0)

{

for (int size = 1; size < 10; size++)

{

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

{

average\_time += find\_time(size);

}

cout << "среднее время для " << N << " измерений " << "матриц размером " << size << " = " << average\_time / N << "мc" << endl;

}

}

else

{

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

{

average\_time += find\_time(size);

}

cout << "среднее время для " << N << " измерений " << "матриц размером " << size << " = " << average\_time / N << "мc" << endl;

}

}

Листинг lab\_4\_1\_func.cpp:

#include <iostream>// библиотека

#include <iomanip>

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include "lab\_4\_1\_func.h"

#include "string"

#include "vector"

using std::string;

using std::cout;

using std::cin;

using std::vector;

using std::pair;

using std::setw;

using std::make\_pair;

using std::endl;

using std::fixed;

using std::setprecision; //подключение пространства имен std

int find\_time(unsigned int size)

{

int start\_time, end\_time, time = 0;

Value value;

srand(clock());

unsigned int N, P;

N = size;

P = N + 1;

vector <vector <long long>> matrix(N, vector<long long>(P, 0));

vector <double> matrix\_x(N);

fill(matrix);

if (value.debugging == 1)

{

cout << "исходная матрица:" << endl;

print\_vector\_two\_dimensional\_4\_1(matrix);

}

start\_time = clock();

calculation(matrix);

matrix\_x = decision(matrix);

end\_time = clock();

time += end\_time - start\_time;

if (value.debugging == 1)

{

cout << "полученная матрица" << endl;

print\_vector\_two\_dimensional\_4\_1(matrix);

}

if (value.debugging == 1)

{

cout << "ответ: " << endl;

print\_vector\_4\_1(matrix\_x);

for (int i = 0; i < 6; i++)

{

cout << endl;

}

}

matrix.clear();

return time;

}

void calculation(vector <vector <long long>> &matrix)

{

vector <long long> output\_matrix(matrix.size());

long long base, element\_i, coefficient;

int l = 1;

bool final = 1;

Value value;

for (int k = 0; k < matrix.size() - 1; k++)

{

if (matrix[k][k] == 0)

{

l = 1;

if (value.debugging == 1)

{

cout << "первый элемент равен нулю" << endl;

}

for (int i = k + 1; i < matrix.size(); i++)

{

if (matrix[i][k] != 0)

{

for (int j = k; j < matrix[i].size(); j++)

{

swap(matrix[k][j], matrix[i][j]);

}

if (value.debugging == 1)

{

cout << "матрица после перестановки строк:" << endl;

print\_vector\_two\_dimensional\_4\_1(matrix);

}

break;

}

else

l++;

if (l == matrix.size() - k)

{

final = 0;

if (value.debugging == 1)

{

cout << "матрица вырожденная, значит имеет бесконечно много решений!" << endl;

print\_vector\_two\_dimensional\_4\_1(matrix);

}

break;

exit(0);

}

}

}

if (final == 0)

break;

for (int i = k + 1; i < matrix.size(); i++)

{

base = matrix[k][k];

element\_i = matrix[i][k];

if (element\_i != 0)

{

if (element\_i % base == 0)

{

coefficient = element\_i / base;

for (int j = k; j < matrix[i].size(); j++)

{

matrix[k][j] \*= coefficient;

matrix[i][j] += -1 \* matrix[k][j];

}

}

else if (base % element\_i == 0)

{

coefficient = base / element\_i;

for (int j = k; j < matrix[i].size(); j++)

{

matrix[i][j] \*= coefficient;

matrix[i][j] += -1 \* matrix[k][j];

}

}

else

{

for (int j = k; j < matrix[i].size(); j++)

{

matrix[i][j] \*= base;

matrix[k][j] \*= element\_i;

matrix[i][j] += -1 \* matrix[k][j];

}

}

}

if (value.debugging == 1)

{

cout << "матрица после прохода " << i + 1 << " строки, " << k + 1 << " столбца; " << endl;

print\_vector\_two\_dimensional\_4\_1(matrix);

cout << endl;

}

}

decrease(matrix);

}

}

vector <double> decision(vector <vector <long long>> matrix)

{

vector <double> matrix\_x(matrix.size(), 0);

double temp;

for (int i = matrix.size() - 1; i >= 0; i--)

{

temp = 0;

for (int j = matrix[i].size() - 2; j >= i; j--)

{

temp += matrix[i][j] \* matrix\_x[j];

}

temp = matrix[i][matrix[i].size() - 1] - temp;

matrix\_x[i] = temp / matrix[i][i];

}

return matrix\_x;

}

void print\_vector\_4\_1(vector <double> matrix)

{

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

{

cout << "x" << i + 1 << " = " << setw(4) << matrix[i] << "; ";

}

}

void fill(vector <vector <long long>>& matrix)

{

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

{

for (int j = 0; j < matrix[i].size() - 1; j++)

{

matrix[i][j] = rand() % 13 - 6;

}

matrix[i][matrix[i].size() - 1] = (rand() % 3 + 3) \* (rand() % 13 - 6);

}

}

void print\_vector\_two\_dimensional\_4\_1(vector <vector <long long>> matrix)

{

unsigned int N = matrix.size();

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

{

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

{

cout << setw(N \* 3 - 2) << matrix[i][j];

}

cout << endl;

}

}

void decrease(vector <vector <long long>>& matrix)

{

int l;

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

{

for (int s = 2; s <= 122; s++)

{

l = 1;

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

{

if (matrix[i][j] % s != 0)

{

l = 0;

break;

}

}

if (l == 1)

{

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

{

matrix[i][j] /= s;

}

}

}

}

}

Листинг lab\_4\_1\_func.h:

#pragma once

#include <iostream>// библиотека

#include <iomanip>

#include <Windows.h> // библиотека

#include "string"

#include "vector"

using std::string;

using std::cout;

using std::cin;

using std::vector;

struct Value

{

bool debugging = 1;

};

void calculation(vector <vector <long long>> &matrix);

int find\_time(unsigned int size);

void print\_vector\_two\_dimensional\_4\_1(vector <vector <long long>> matrix);

void print\_vector\_4\_1(vector <double> matrix);

vector <double> decision(vector <vector <long long>> matrix);

void decrease(vector <vector <long long>>& matrix);

void fill(vector <vector <long long>>& matrix);

Листинг main.cpp:

#include <iostream>// библиотека

#include <Windows.h> // библиотека

#include "main\_header.h"

#include "various\_functions.h"

int main()

{

setlocale(0, ""); // поддержка кириллицы в консоли (вывод)

SetConsoleCP(1251); // поддержка кириллицы в консоли (ввод)

SetConsoleOutputCP(1251); // поддержка кириллицы в консоли (ввод)

double task\_number;

boolean debugging = true;

if (debugging == true)

{

//lab\_2\_1();

//lab\_5\_1();

//lab\_2\_2();

//lab\_4\_2();

//lab\_4\_1();

//dop\_task\_1();

//lab\_3\_1();

lab\_3\_2();

//lab\_3\_2\_copy();

}

else

{

std::cout

<< "Готовые лабораторные работы: " << std::endl

<< std::endl

<< " 1.1 Поиск макс. суммы двух элементов " << std::endl

<< " 1.2 Цикл с вычислением функции " << std::endl

<< " 1.3 Поиск неповторяющихся остатков от деления " << std::endl

<< " 2.1 Количество скобок " << std::endl

<< " 2.2 Бинарный поиск строки по хэшу " << std::endl

<< " 3.1 Рекурсия " << std::endl

<< " 4.1 Оценка алгоритма (Гаусс) " << std::endl

<< " 4.2 База данных " << std::endl

<< " 5.1 Битовая последовательность " << std::endl

<< std::endl

<< "Введите номер лабораторной работы: ";

while (true)

{

task\_number = get\_number<double>();

if (task\_number == 1.1 ||

task\_number == 1.2 ||

task\_number == 1.3 ||

task\_number == 2.1 ||

task\_number == 2.2 ||

task\_number == 3.1 ||

task\_number == 4.1 ||

task\_number == 4.2 ||

task\_number == 5.1)

{

system("cls");

std::cout << "Лабораторная работа № " << task\_number << std::endl;

break;

}

else std::cout << "Такой лабораторной работы нет, введите другой номер: ";

}

if (task\_number == 1.1)

{

lab\_1\_1();

}

else if (task\_number == 1.2)

{

lab\_1\_2();

}

else if (task\_number == 1.3)

{

lab\_1\_3();

}

else if (task\_number == 2.1)

{

lab\_2\_1();

}

else if (task\_number == 2.2)

{

lab\_2\_2();

}

else if (task\_number == 3.1)

{

lab\_3\_1();

}

else if (task\_number == 4.1)

{

lab\_4\_1();

}

else if (task\_number == 4.2)

{

lab\_4\_2();

}

else if (task\_number == 5.1)

{

lab\_5\_1();

}

}

return 0;

}

Листинг various\_functions.h:

#pragma once

#include <string>

#include <vector>

#include "Point.h"

using std::string;

using std::cout;

using std::cin;

using std::pair;

template <typename T>

T get\_number\_input\_after(string str);

template <typename T>

T input\_value(T a);

template <typename T>

T get\_number();

template <typename T>

bool check(string str);

template <typename T>

bool check\_without\_print(string str);

template <typename T>

T str\_to\_number(string str);

string char\_to\_string(char ch);

Point cursor\_position();

void set\_cursor\_position(Point A);

void erase\_past\_output(Point A);

template <typename T>

string number\_to\_str(T a)

{

string type = typeid(a).name();

string S = {}, s = {};

if (type == "int")

{

if (a < 0)

{

a = a \* -1;

s += '-';

}

while (true)

{

S += (a % 10) + '0';

a /= 10;

if (a == 0)

break;

}

for (int i = S.size() - 1; i != -1; i--)

{

s += S[i];

}

}

return s;

}

template <typename T>

T input\_value(T a)

{

string str;

getline(cin, str);

if (str[0] == '\0')

return a;

return get\_number\_input\_after<T>(str);

}

template <typename T>

T get\_number\_input\_after(string str)

{

while (true) // цикл продолжается до тех пор, пока пользователь не введет корректное значение

{

if (check<T>(str) == 1)

{

return str\_to\_number<T>(str);

}

getline(cin, str);

}

}

template <typename T>

T get\_number()

{

while (true) // цикл продолжается до тех пор, пока пользователь не введет корректное значение

{

string str;

getline(cin, str);

if (check<T>(str) == 1) //проверка на число

{

return str\_to\_number<T>(str); //преобразование строки в число

}

}

}

template <typename T>

bool check(string str)

{

bool k = 1;

if ((str.find('-') < 1 || str.find('-') == -1) && ((str.find('.') < str.size() - 1 && str.find('.') > 0) || str.find('.') == -1))

{

if (str.length() > 1)

{

int count\_minus = 0, count\_point = 0;

for (int i = 0; i < str.length(); i++)

{

if (str[i] == '-')

{

count\_minus++;

}

if (str[i] == '.')

{

count\_point++;

}

}

if ((str[0] == '0' && str[1] != '.') || count\_point > 1 ||

(str[0] == '-' && str[1] == '.') ||

(str[0] == '-' && str[1] == '0' && str[2] != '.') || count\_minus > 1)

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

}

else

if (str == "-")

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

if (k == 1)

{

string s = typeid(T).name();

if (s == "double")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '.' && str[i] != '-')

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

break;

}

}

}

if (s == "int")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '-')

{

k \*= 0;

std::cout << "!!Вы ввели не целое число!!" << std::endl << "Введитe целое число: ";

break;

}

}

}

if (s == "bool")

{

if (str[0] != '0' && str[0] != '1')

{

k \*= 0;

std::cout << "!!Вы ввели не двоичное число!!" << std::endl << "Введитe двоичное число: ";

}

}

}

}

else

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

return k;

}

template <typename T>

bool check\_without\_print(string str)

{

bool k = 1;

// 67642.5423

//проверка на возможность существования такого числа

if ((str.find('-') < 1 || str.find('-') == -1) && ((str.find('.') < str.size() - 1 && str.find('.') > 0) || str.find('.') == -1))

{

//если длина больше

if (str.length() > 1)

{

//проверка на возможность существования такого числа

//после нуля всегда точка или ничего, если ноль первый

//после - если ноль, то должна быть точка или ничего

int count\_minus = 0, count\_point = 0;

for (int i = 0; i < str.length(); i++)

{

if (str[i] == '-')

{

count\_minus++;

}

if (str[i] == '.')

{

count\_point++;

}

}

if ((str[0] == '0' && str[1] != '.') || count\_point > 1 ||

(str[0] == '-' && str[1] == '.') ||

(str[0] == '-' && str[1] == '0' && str[2] != '.') || count\_minus > 1)

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

}

else

if (str == "-")

{

k \*= 0;

}

if (k == 1)

{

string s = typeid(T).name();

if (s == "double")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '.' && str[i] != '-')

{

k \*= 0;

break;

}

}

}

if (s == "int")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '-')

{

k \*= 0;

break;

}

}

}

}

}

else

{

k \*= 0;

}

return k;

}

template <typename T>

T str\_to\_number(std::string str)

{

T a = 0; //само число

int index = 0, k = 1;

if (str[0] == '-')

{

k = -1;

str.erase(0, 1);

}

if (str.find('.') != -1)

{

index = str.find('.');

str.erase(str.find('.'), 1);

}

else

index = str.length();

for (int i = 0; i < str.length(); i++)

{

a = a \* 10 + (str[i] - '0');

}

return k \* (a / pow(10, (str.length() - index)));

}

Листинг various\_functions.cpp:

#include <iostream>// библиотека

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include <vector>

#include "Point.h"

using namespace std;

string char\_to\_string(char ch)

{

string s = {};

s += ch;

return s;

}

Point cursor\_position()

{

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

return { bi.dwCursorPosition.X, bi.dwCursorPosition.Y };

}

void set\_cursor\_position(Point A)

{

COORD coord;

coord.X = A.x;

coord.Y = A.y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);

}

void erase\_past\_output(Point A)

{

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

set\_cursor\_position(A);

string s = "";

for (int i = 0; i < 80; i++)

s += ' ';

for (int i = 0; i < bi.dwCursorPosition.Y - A.y; i++)

cout << s << endl;

set\_cursor\_position(A);

}

Листинг main\_header.h:

#pragma once

void dop\_task\_1();

void lab\_1\_1();

void lab\_1\_2();

void lab\_1\_3();

void lab\_2\_1();

void lab\_2\_2();

void lab\_3\_1();

void lab\_3\_2();

void lab\_3\_2\_copy();

void lab\_4\_1();

void lab\_4\_2();

void lab\_5\_1();

**Результат.**

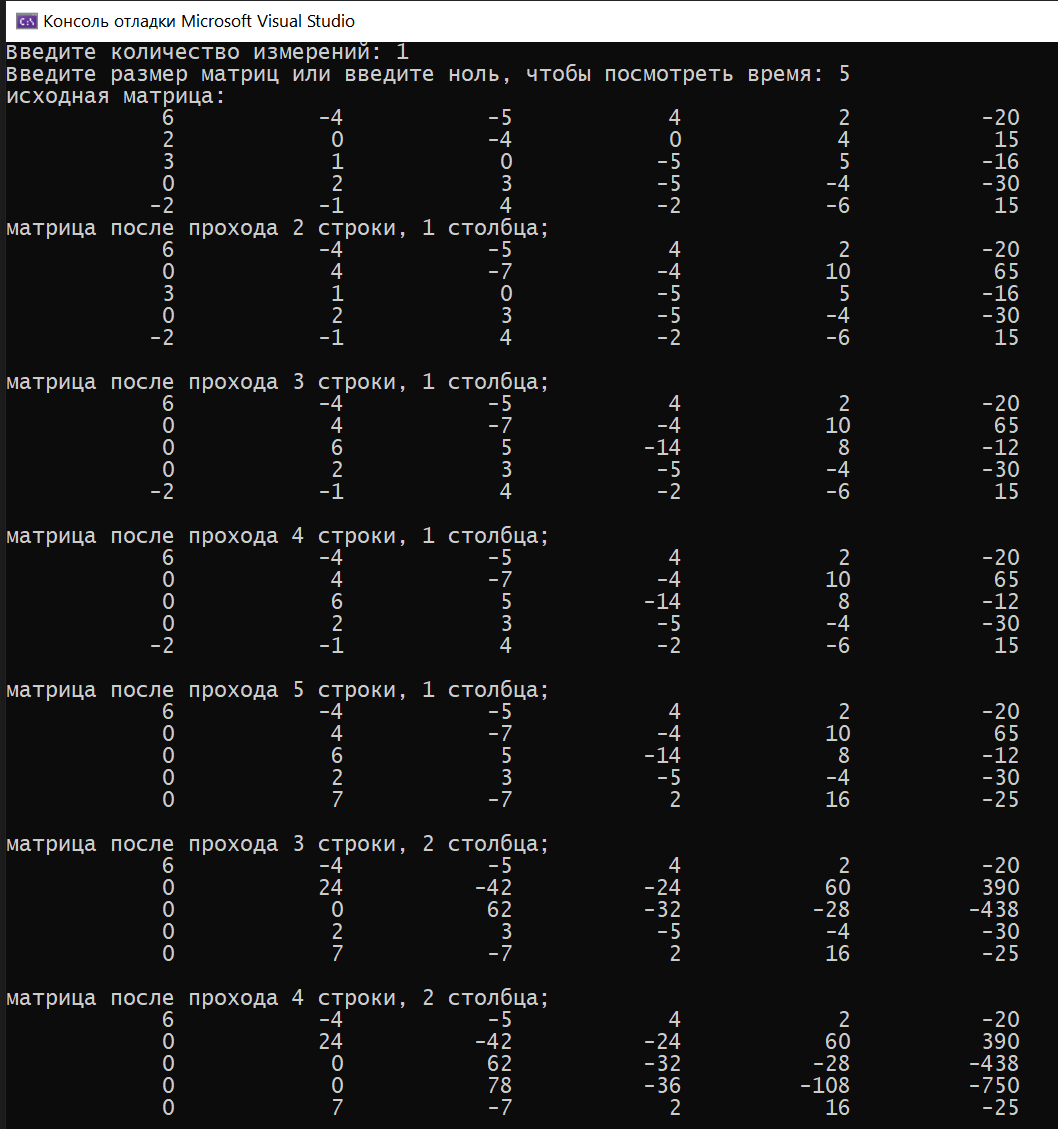


Рисунок 2. Тест 1

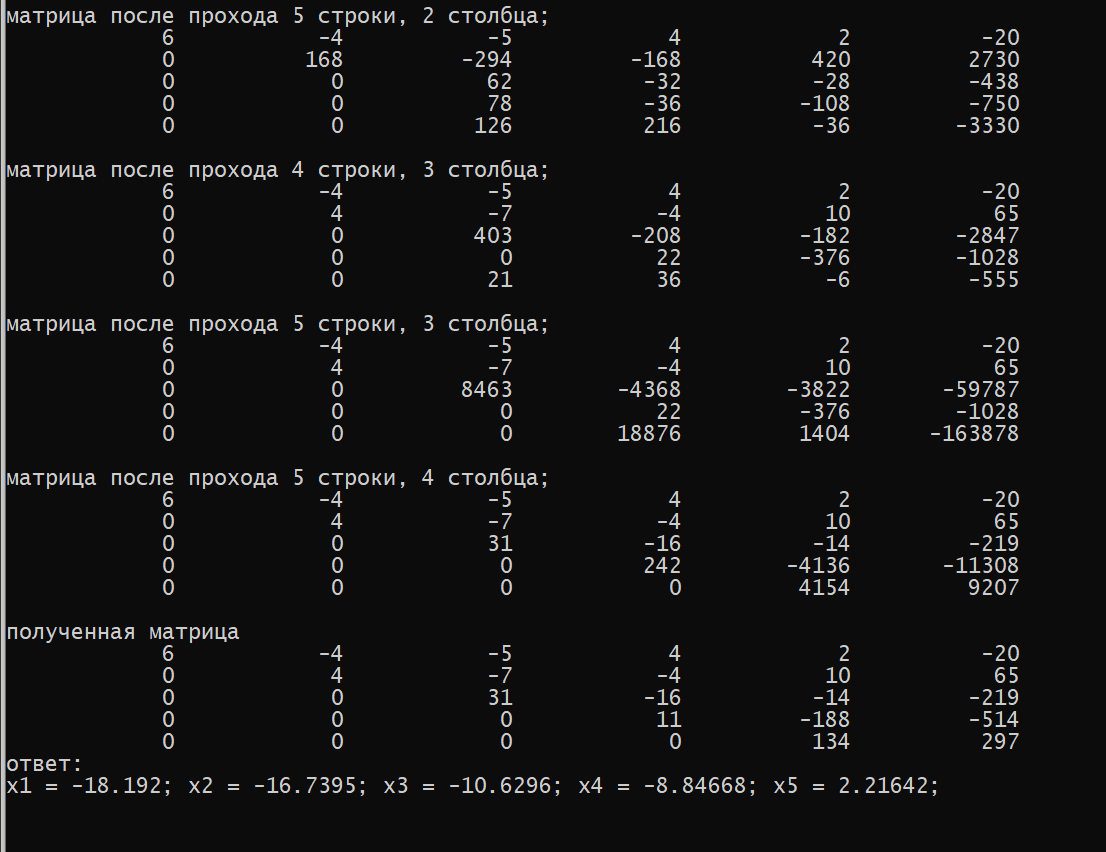


Рисунок 3. Тест 1

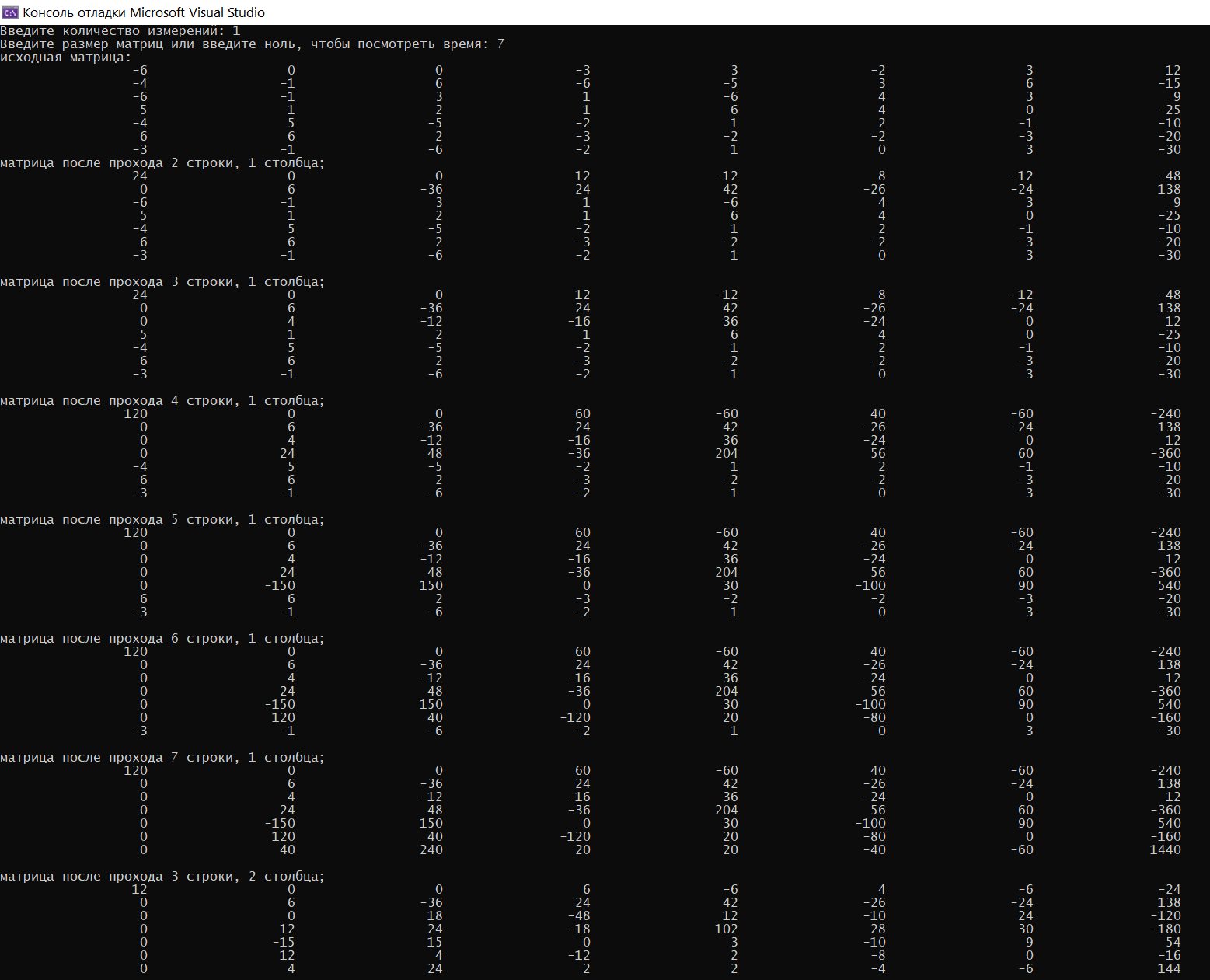


Рисунок 4. Тест 2

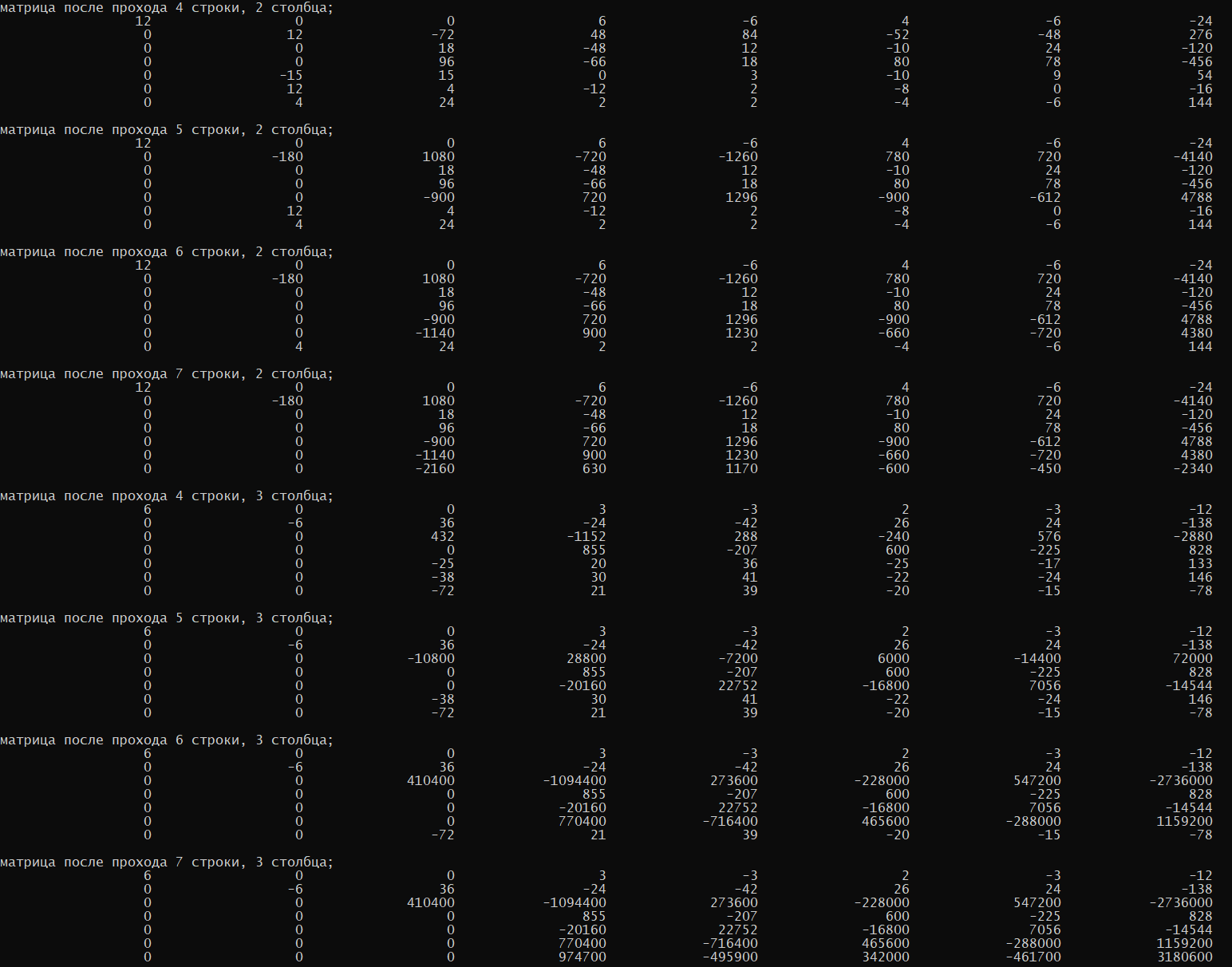
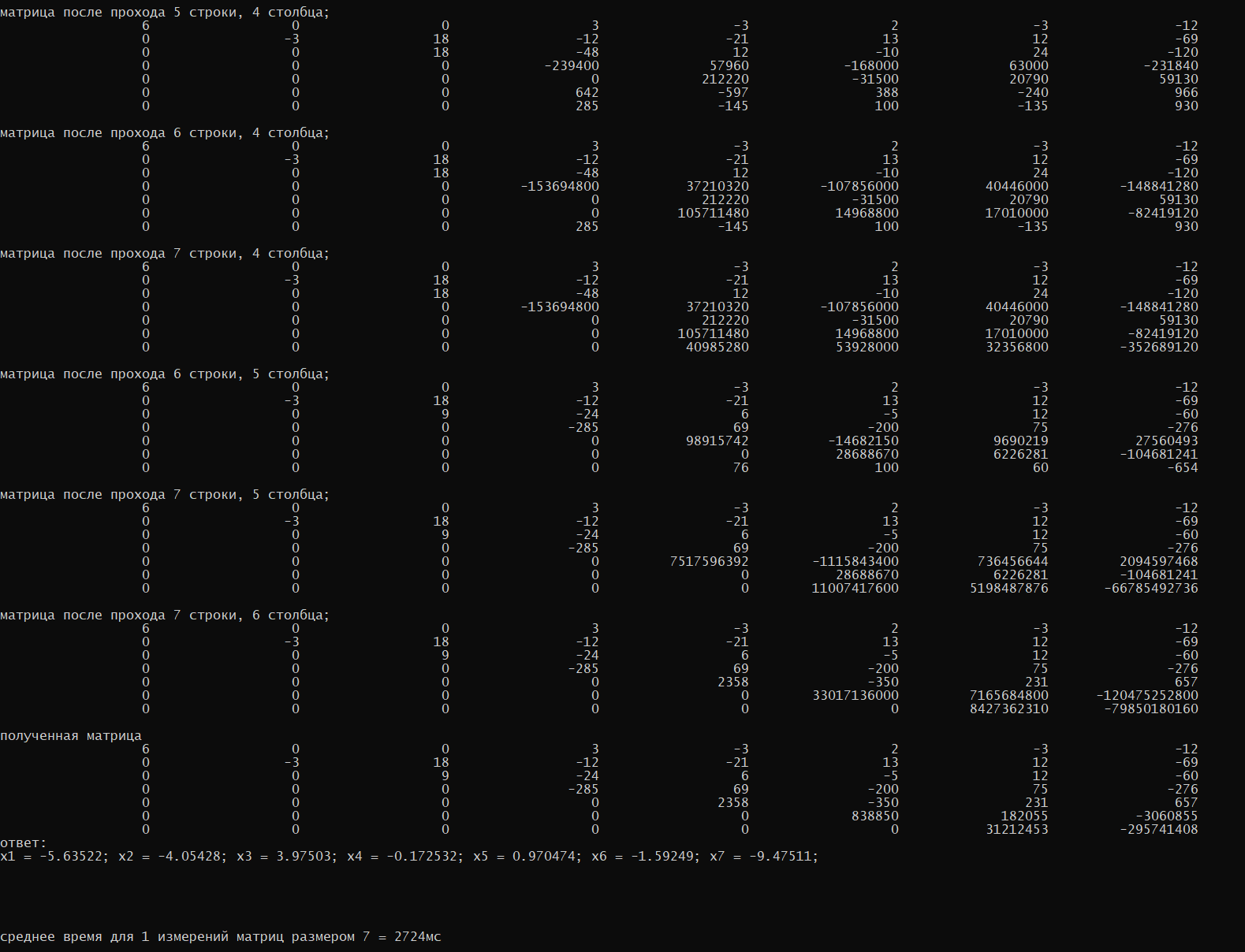


Рисунок 5. Тест 2

Рисунок 6. Тест 2

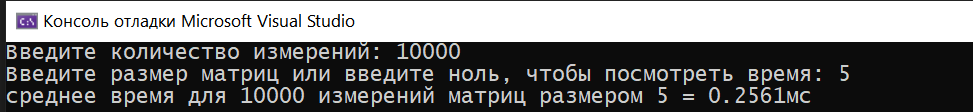
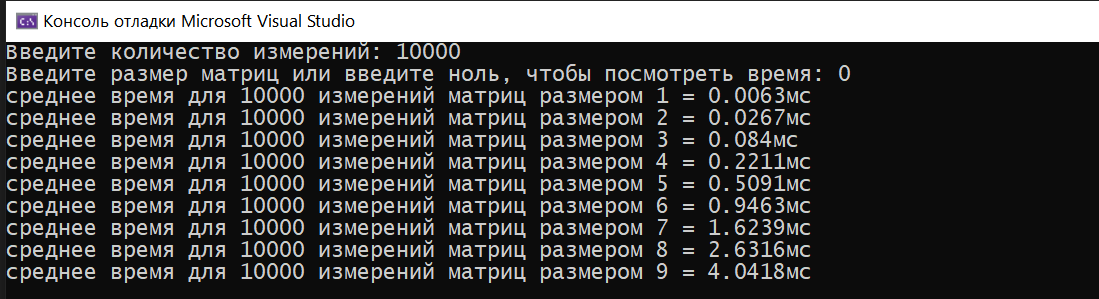


Рисунок 7. Тест 3

# Оценка времени и сложности

Временная сложность:

Рисунок 8. Количество времени при разных исходных данных

Оценка сложности:

Сложность O(n^3). Мы можем найти кубы 1, 2, 3, 4 и т. д., будет 1, 8, 27, 64. Разделим следующий элемент на предыдущий, получим: 8, 3.375, 2,37.

Разделим полученное время:

0.0267 / 0.0063 = 4,238;

0.084 / 0.0267 = 3,146;

0.2211 / 0.084 = 2,63.

Заметим, что цифры получились похожие.

# Вывод.

Вводится размер матрицы. Алгоритм решает ее методом гаусса и полученную матрицу выводит в консоль.