ГУАП

КАФЕДРА № 53

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

ПРЕПОДАВАТЕЛЬ

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

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| №3.1.РЕКУРСИВНЫЙ АЛГОРИТМ |
| Вариант 2 (15) |
| по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |

РАБОТУ ВЫПОЛНИЛ

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

Санкт-Петербург 2022

**Оглавление:**

[Цель. 3](#_Toc96548818)

[Индивидуальное задание 4](#_Toc96548819)

[Описание алгоритма 5](#_Toc96548821)

[Блок-схема алгоритма. 6](#_Toc96548822)

[Листинг программы. 10](#_Toc96548823)

[Результат. 14](#_Toc96548824)

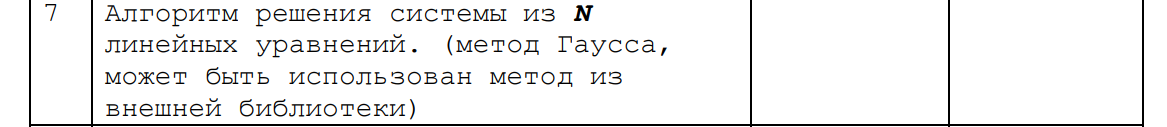
[Вывод. 16](#_Toc96548825)

# 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.

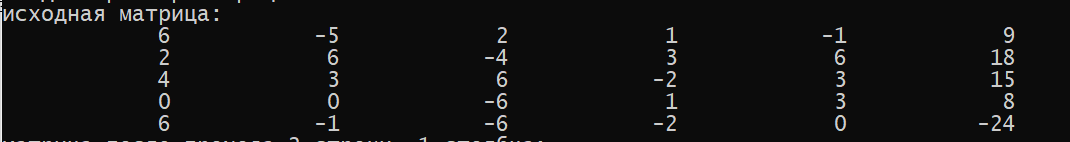
# Индивидуальное задание.

Рисунок 1. Индивидуальное задание

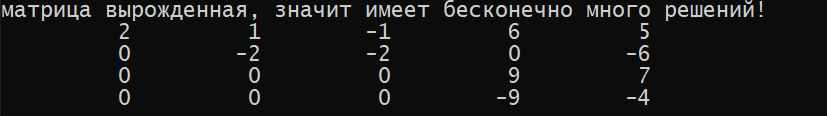
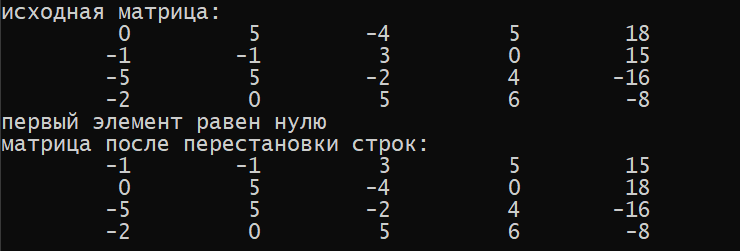
# Описание алгоритма

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.

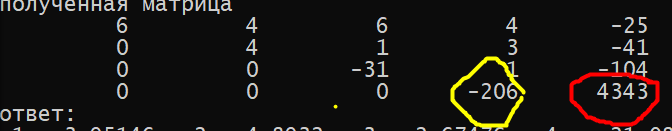
Потом базовый элемент сдвигается и его индекс становится 1,1. Мы снова пробегаемся по строкам, которые лежат ниже базовой строки.

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

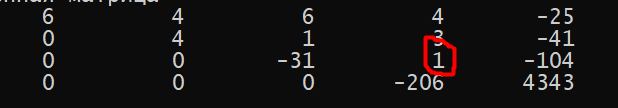
Потом вызывается функция decision. В ней запускается первый цикл, который идет от последней строки до первой. В нем мы инициализируем temp = 0, потом запускаем второй цикл, который пробегается от элемента с индексом на 2 меньше длины строки до индекса строки. То есть в последней строке, которая имеет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

# Блок-схема алгоритма.

# 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 namespace std; //подключение пространства имен std

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, n, size;

double average\_time = 0;

cout << "Введите количество измерений: ";

N = get\_number\_int();

cout << "Введите количество матриц в измерении: ";

n = get\_number\_int();

cout << "Введите размер матриц: ";

size = get\_number\_int();

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

{

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

}

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

}

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 namespace std; //подключение пространства имен std

int find\_time(unsigned int n, unsigned int size)

{

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

Values value;

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

{

srand(clock());

unsigned int N, P;

if (size == 0)

{

N = rand() % 5 + 2;

}

else 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 == 0)

{

cout << "исходная матрица:" << endl;

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

}

if (value.debugging == 1)

{

cout << "ответ: " << endl;

print\_vector(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;

boolean final = 1;

Values value;

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

{

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

{

l = 1;

if (value.debugging == 0)

{

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

}

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

{

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

{

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

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

else

l++;

}

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

{

final = 0;

if (value.debugging == 0)

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

print\_vector\_two\_dimensional(matrix);

break;

exit(0);

}

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

{

if (value.debugging == 0)

{

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

print\_vector\_two\_dimensional(matrix);

}

break;

}

}

}

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 == 0)

{

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

print\_vector\_two\_dimensional(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(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(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 namespace std;

struct Values

{

boolean debugging = 0;

};

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

int find\_time(unsigned int n, unsigned int size);

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

void print\_vector(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);

# Результат.

# Вывод.

В ходе выполнения лабораторной работы были освоены:

* массивы
* условные операторы
* циклы
* функции
* вызов функций из другого файла
* обработка ввода символов
* рекурсия
* динамические массивы
* форматированный вывод

Программа принимает число и выводит количество единиц. Программа делит число 2 пока число не станет равно нулю.