Федеральное государственное бюджетное образовательное учреждение «Сибирский государственный университет телекоммуникаций и информатики»

Кафедра ПМиК

Лабораторная работа №2

по дисциплине

«Алгоритмы и вычислительные методы оптимизации»

Выполнил: студент III курса

Группы ИП-813

Бурдуковский И.А.

Проверила:

Галкина М.Ю.

Новосибирск 2021

Оглавление

[Задание на лабораторную работу 3](#_Toc40794109)

[Текст программы 4](#_Toc40794110)

[Результаты тестирования 17](#_Toc40794111)

# Задание на лабораторную работу

Написать программу, находящую все базисные решения системы линейных уравнений методом Жордана-Гаусса. Программа должна выводить промежуточные матрицы после каждого шага исключений и все найденные базисные решения.

# Текст программы

**Zhordan-Gauss.cpp**

#include <fstream>

#include <conio.h>

#include <iomanip>

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <iostream>

#include <sstream>

using namespace std;

struct Fraction

{

long long numerator;

long long denominator = 1;

};

void print(Fraction \*\*arr,int n,int m)

{

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

{

for (int j=0;j<n-1;j++)

{

if(arr[i][j].denominator==1 || arr[i][j].numerator==0){

cout << arr[i][j].numerator << " ";

}

else

cout<<arr[i][j].numerator << "/" << arr[i][j].denominator<<" ";

}

cout<<"| ";

if(arr[i][n-1].denominator==1 || arr[i][n-1].numerator==0){

cout<<arr[i][n-1].numerator<<endl;

}

else

cout<<arr[i][n-1].numerator << "/" << arr[i][n-1].denominator<<endl;

}

cout<<endl;

}

//рекурсивное нахождение НОДа

long long NOD(long long a, long long b)

{

if(b == 0) return a;

else return NOD(b, a % b);

}

//сокращение простых дробей

long long divide(Fraction \*\*arr,int n,int m){

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

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

if(arr[i][j].denominator != 0)

if(arr[i][j].numerator % arr[i][j].denominator == 0){

arr[i][j].numerator /= arr[i][j].denominator;

arr[i][j].denominator = 1;

}

else{

long long d = NOD(arr[i][j].denominator,arr[i][j].numerator);

if(d!=0){

arr[i][j].denominator /= d;

arr[i][j].numerator /= d;

}

}

}

}

}

//выписать значения всех X

void PrintXAnswer(Fraction \*\*arr, int n, int m){

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

if(arr[i][i].numerator==0){

continue;

}

cout << "x" << i+1 << " = ";

if(arr[i][n-1].numerator==0){

cout << "0 ";

}

else{

cout << arr[i][n-1].numerator;

if (arr[i][n-1].denominator != 1){

cout << "/" << arr[i][n-1].denominator;

}

}

for(int j = i+1; j < n-1; j++){

if (arr[i][j].numerator == 0){

continue;

}

if (arr[i][j].numerator < 0){

if (arr[i][j].denominator != 1){

cout << " + " << arr[i][j].numerator \* -1 << "/" << arr[i][j].denominator;

}

else{

cout << " + " << arr[i][j].numerator \* -1;

}

}

else{

if (arr[i][j].denominator < 0){

cout << " + " << arr[i][j].numerator << "/" << arr[i][j].denominator\*-1;

}

else{

if (arr[i][j].denominator != 1){

cout << " - " << arr[i][j].numerator << "/" << arr[i][j].denominator;

}

else{

cout << " - " << arr[i][j].numerator;

}

}

}

cout << " \* x" <<j+1;

}

cout << endl;

}

}

//перенос пустых строк в низ матрицы

void MoveZeroLinesBot(Fraction \*\*arr, int n, int m){

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

int count = 0;

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

if(arr[i][j].numerator != 0){

break;

}

count++;

}

if(count == n){

for(int v = i+1; v < m; v++){

int count\_2 = 0;

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

if(arr[v][j].numerator != 0){

break;

}

count\_2++;

}

if(count\_2 != n){

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

arr[i][j].numerator = arr[v][j].numerator;

arr[i][j].denominator = arr[v][j].denominator;

arr[v][j].numerator = 0;

}

break;

}

}

}

}

}

//расчёт ранга матрицы

int CountRang(Fraction \*\*arr, int n, int m){

int rang = 0;

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

{

int count = 0;

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

if(arr[i][j].numerator != 0){

break;

}

count++;

}

if (count != m)

rang++;

}

return rang;

}

//расчёт факториала

long long fact(long long n)

{

if (n<=1)

return 1;

return n\*fact(n-1);

}

//расчёт количества комбинаций

long long CombinNum(int rang, int X\_count)

{

if (rang == X\_count)

return 1;

return fact(X\_count) / (fact(X\_count - rang) \* fact(rang));

}

bool NextSequence(int \*a, int X\_count, int rang)

{

for (int i = rang - 1; i >= 0; --i)

if (a[i] < X\_count - rang + i + 1)

{

++a[i];

for (int j = i + 1; j < rang; ++j)

a[j] = a[j - 1] + 1;

return true;

}

return false;

}

void Jordan\_Gauss(Fraction \*\*arr,int n,int m){

MoveZeroLinesBot(arr,n,m);

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

//если элемент главной диагонали == 0, то скипаем

if(arr[i][i].numerator == 0){

continue;

}

//если элемент главной диаганали - отрицательное число

if(arr[i][i].numerator/arr[i][i].denominator < 0){

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

arr[i][j].numerator \*= -1;

}

}

//если элемент главной диаганали не = 1

if(arr[i][i].numerator/arr[i][i].denominator != 1 || arr[i][i].numerator % arr[i][i].denominator != 0){

long long num = arr[i][i].numerator;

long long den = arr[i][i].denominator;

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

arr[i][j].numerator \*= den;

arr[i][j].denominator \*= num;

}

}

//cout << endl;

divide(arr,n,m);

print(arr,n,m);

//зануляем элемент выше и ниже главной диагонали

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

//если элемент на главной диаганали - скипаем

if(v == i){

continue;

}

//иначе зануляем

for(int j = i+1; j < n; j++){

//умножаем множитель на элемент ведущей строки

long long num = arr[v][i].numerator \* arr[i][j].numerator;

long long den = arr[v][i].denominator \* arr[i][j].denominator;

//cout << num<< "/" << den<< " - " << arr[v][j].numerator << "/" << arr[v][j].denominator << " - ";

//если у уменьшаемого и вычитаемого разные знаменатели - домнажаем на знаменатели друг друга

if (den != arr[v][j].denominator){

long long buf = den;

num \*= arr[v][j].denominator;

den \*= arr[v][j].denominator;

arr[v][j].numerator \*= buf;

arr[v][j].denominator \*= buf;

}

//cout << num<< "/" << den<<endl;

arr[v][j].numerator -= num;

}

//зануляем элементы

arr[v][i].numerator = 0;

arr[v][i].denominator = 0;

//проверка на решаемость СЛАУ (все х==0, когда результат !=0)

if(arr[v][n-1].numerator != 0){

int zero\_counter = 0;

for (int j = 0; j <n-1;j++){

if (arr[v][j].numerator!=0){

break;

}

zero\_counter++;

}

if(zero\_counter==n-1){

divide(arr,n,m);

cout << endl;

print(arr,n,m);

cout << "There are no resolves here!"<<endl;

return;

}

}

}

divide(arr,n,m);

cout << endl;

print(arr,n,m);

}

MoveZeroLinesBot(arr,n,m);

print(arr,n,m);

//находим ранг матрицы

int rang = CountRang(arr,n,m);

int combination\_num = CombinNum(rang,n-1);

cout << "Matrix rang:\t" << rang << endl << "X count:\t" << n-1 << endl << "Combination num:\t" << combination\_num << endl;

//генерируем сочетания

int \*buf\_comb = new int[rang];

int \*comb = new int[rang \* combination\_num];

int combin\_index = rang;

//генерируем перввую последовательность

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

{

buf\_comb[i]=i+1;

comb[i]=i+1;

}

//генерируем остальные последовательности

while(NextSequence(buf\_comb, n-1, rang))

{

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

{

comb[combin\_index] = buf\_comb[i];

combin\_index++;

}

}

cout << endl << "Combin sequence:" << endl;

for(int i = 0; i < rang \* combination\_num; i++){

cout << comb[i] << " ";

}

cout << endl << endl;

cout << "--------------------" << endl << endl;

//ищем все базисные решения

for(int i = 0; i < combination\_num; i++){

int break\_flag = 0;

cout << "#" << i+1 << " [";

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

cout << " " << comb[i\*rang + j];

}

cout << " ]" << endl << endl;

//буфферная матрица

Fraction \*\*buf = new Fraction\*[m];

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

{

buf[v]=new Fraction[n];

}

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

{

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

{

buf[v][j].numerator = arr[v][j].numerator;

buf[v][j].denominator = arr[v][j].denominator;

}

}

print(buf, n, m);

int v\_memmory = -1;

//обрабатываем случай комбинации

for(int v = 0; v < rang; v++){

//если элемент комбинации равен 0, то проверяем может ли этот базис вообще существовать

if(buf[v][comb[i\*rang + v]-1].numerator == 0){

int count = 0;

//если элементы в каждом столбце, соответствующему комбинации равны 0, то базиса нет

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

if(buf[v][comb[i\*rang + j]-1].numerator == 0){

count++;

}

}

if(count == rang){

cout << "This option can't exist" << endl;

break\_flag = 1;

break;

}

//иначе вернёмся к этой строке потом

else{

//не запоминаем строку, если была пропущена ещё одна ранеее

if(v\_memmory != -1)

continue;

//иначе запоминаем строку, к которой потом вернёмся

v\_memmory = v;

//cout << endl << "v\_memmory: " << v\_memmory+1 << endl;

continue;

}

}

//cout << endl << v+1 << " - " << comb[i\*rang + v] << endl;

//print(buf,n,m);

//если элемент - отрицательное число

if(buf[v][comb[i\*rang + v]-1].numerator / buf[v][comb[i\*rang + v]-1].denominator < 0){

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

buf[v][k].numerator \*= -1;

}

// print(buf,n,m);

}

//если ведущий элемент не = 1

if(buf[v][comb[i\*rang + v]-1].numerator / buf[v][comb[i\*rang + v]-1].denominator != 1 || buf[v][comb[i\*rang + v]-1].numerator % buf[v][comb[i\*rang + v]-1].denominator != 0){

long long num = buf[v][comb[i\*rang + v]-1].numerator;

long long den = buf[v][comb[i\*rang + v]-1].denominator;

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

buf[v][k].numerator \*= den;

buf[v][k].denominator \*= num;

}

divide(buf,n,m);

print(buf,n,m);

}

//зануляем элементы выше и ниже ведущего элемента

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

if (v == j){

continue;

}

//если нашли строку с не нулевым элементом в ведущем столбце

if(buf[j][comb[i\*rang + v]-1].numerator != 0){

//зануляем

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

if(k == comb[i\*rang + v]-1){

continue;

}

//умножаем множитель на ведущий элемент

long long num = buf[j][comb[i\*rang + v]-1].numerator \* buf[v][k].numerator;

long long den = buf[j][comb[i\*rang + v]-1].denominator \* buf[v][k].denominator;

//cout << num << " / " << den << endl;

//если у уменьшаемого и вычитаемого разные знаменатели - домнажаем на знаменатели друг друга

if (den != buf[j][k].denominator && buf[j][k].denominator != 0 && buf[j][k].numerator != 0 && den != 0){

long long buffer = den;

num \*= buf[j][k].denominator;

den \*= buf[j][k].denominator;

buf[j][k].numerator \*= buffer;

buf[j][k].denominator \*= buffer;

}

//cout << num<< "/" << den<<endl;

//cout << num << " / " << den << endl<< endl ;

buf[j][k].numerator -= num;

if (buf[j][k].denominator==0)

buf[j][k].denominator = den;

}

//зануляем элементы

buf[j][comb[i\*rang + v]-1].numerator = 0;

buf[j][comb[i\*rang + v]-1].denominator = 0;

divide(buf,n,m);

print(buf,n,m);

}

}

//если мы пропустили строку, к которой потом вернёмся

if(v\_memmory != -1){

//если элемент не поменялся, то продолжаем дальше преобразовать, к строке вернёмся позже

if(buf[v][comb[i\*rang + v\_memmory]-1].numerator == 0)

continue;

//иначе возвращаемся к пропущеной строке

v = v\_memmory-1;

v\_memmory = -1;

}

}

divide(buf,n,m);

//если такой базис не возможен

if(break\_flag){

cout << "--------------------" << endl << endl;

continue;

}

//выводим базисное решение

cout << "X" << i+1 << " = (";

int index = 0;

for(int j = 0; j < n-1; j++){

cout << " ";

if (j == comb[i\*rang + index]-1){

if(buf[index][n-1].numerator == 0){

cout << "0";

}

else{

cout << buf[index][n-1].numerator;

if(buf[index][n-1].denominator != 1){

cout << "/" << buf[index][n-1].denominator;

}

}

index++;

}

else{

cout << "0";

}

cout << ";";

}

cout << ")" << endl;

cout << "--------------------" << endl << endl;

}

return;

}

int main()

{

int n,m;

string filename;

cout << "Enter file name:" << endl;

cin >> filename;

ifstream input(filename);

if(!input){

cout << "Invalid file name" << endl;

return 0;

}

input >> n;

input >> m;

Fraction \*\*arr=new Fraction\*[m];

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

{

arr[i]=new Fraction[n];

}

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

{

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

{

input>>arr[i][j].numerator;

}

}

print(arr,n,m);

Jordan\_Gauss(arr,n,m);

return 0;

}

# Результаты тестирования



