МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №1

по дисциплине: Исследование операций

тема: «Исследование множества опорных планов системы ограничений задачи линейного программирования (задачи ЛП) в канонической форме»

Выполнил: ст. группы ПВ-233
Ситников Алексей Павлович
Проверил:
Вирченко Юрий Петрович

Белгород 2025 г.

Цель работы: изучить метод Гаусса-Жордана и операцию замещения, а также освоить их применение к отысканию множества допустимых базисных видов системы линейных уравнений, и решению задачи линейного программирования простым перебором опорных решений.

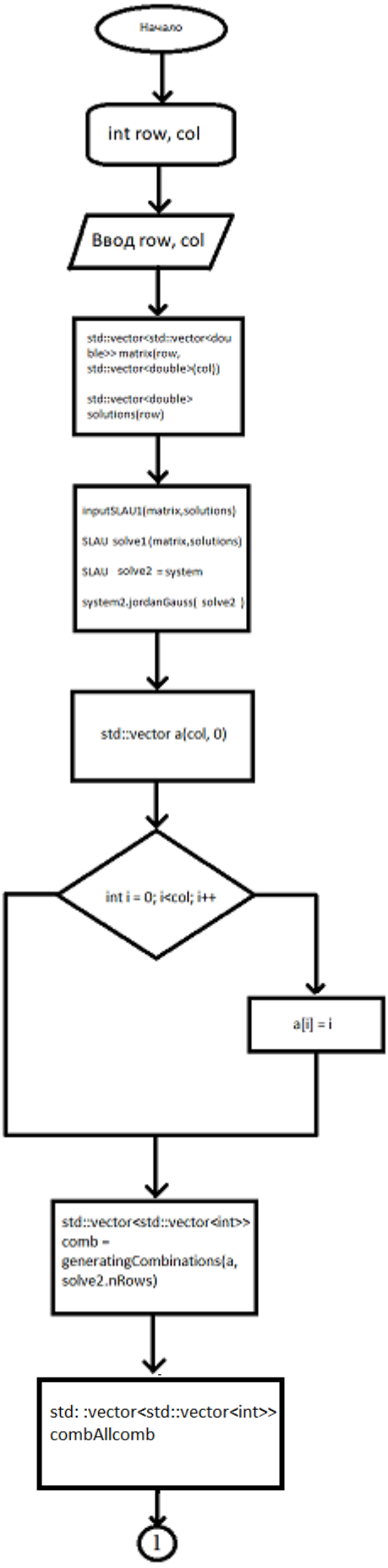
Задание:

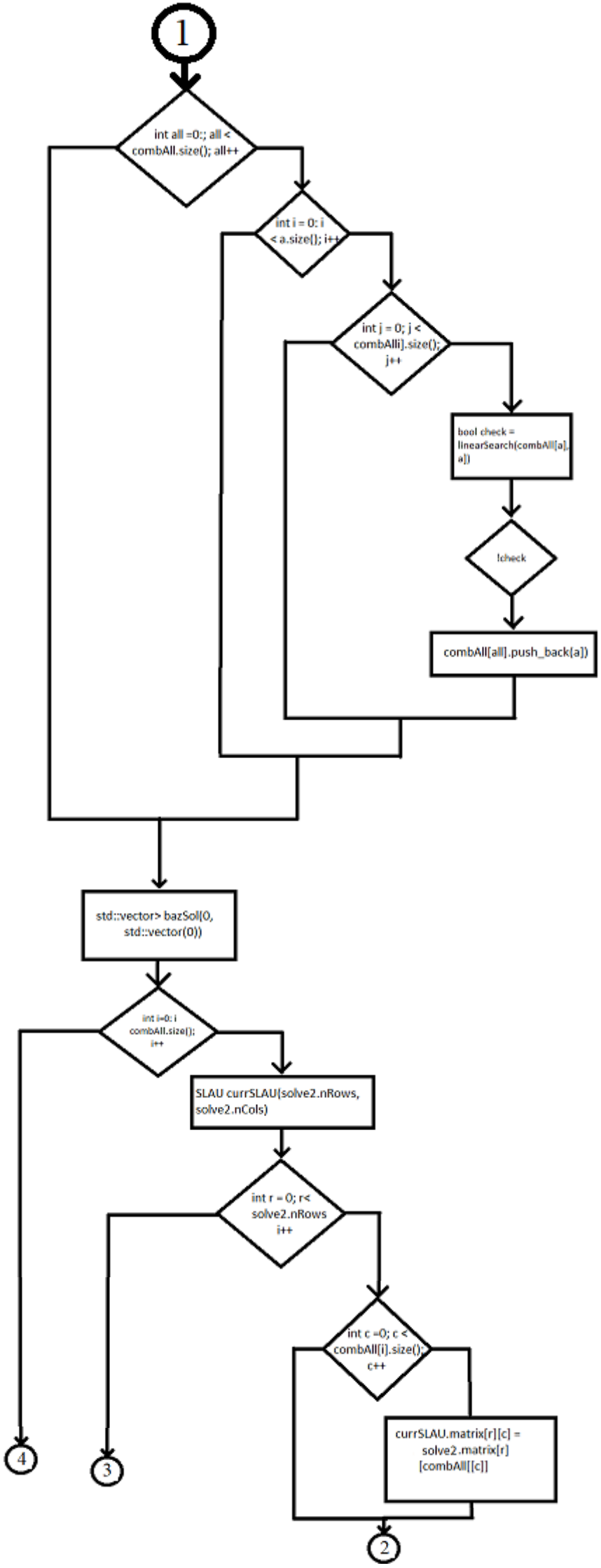
1. Составить программу для отыскания всех базисных видов системы линейных уравнений.
2. Организовать отбор опорных планов среди всех базисных решений, а также нахождение оптимального опорного плана методом прямого перебора. Целевая функция выбирается произвольно.
3. Решить задачу под вариантом 13.

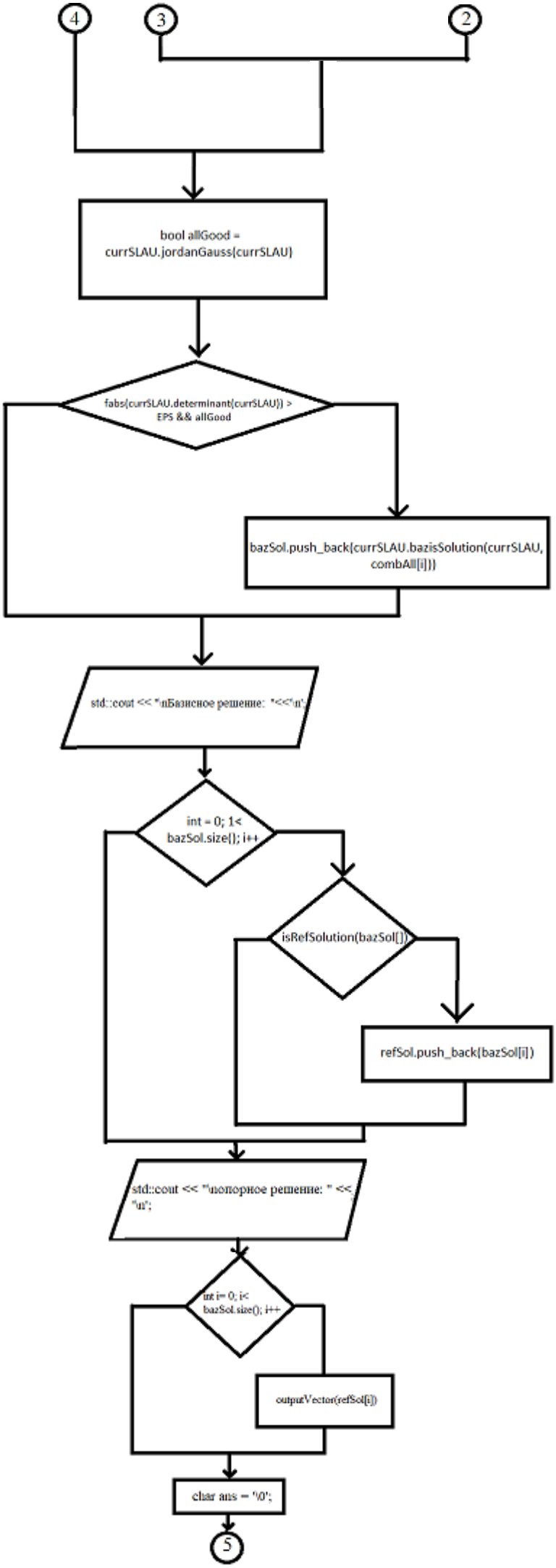
Вариант 13

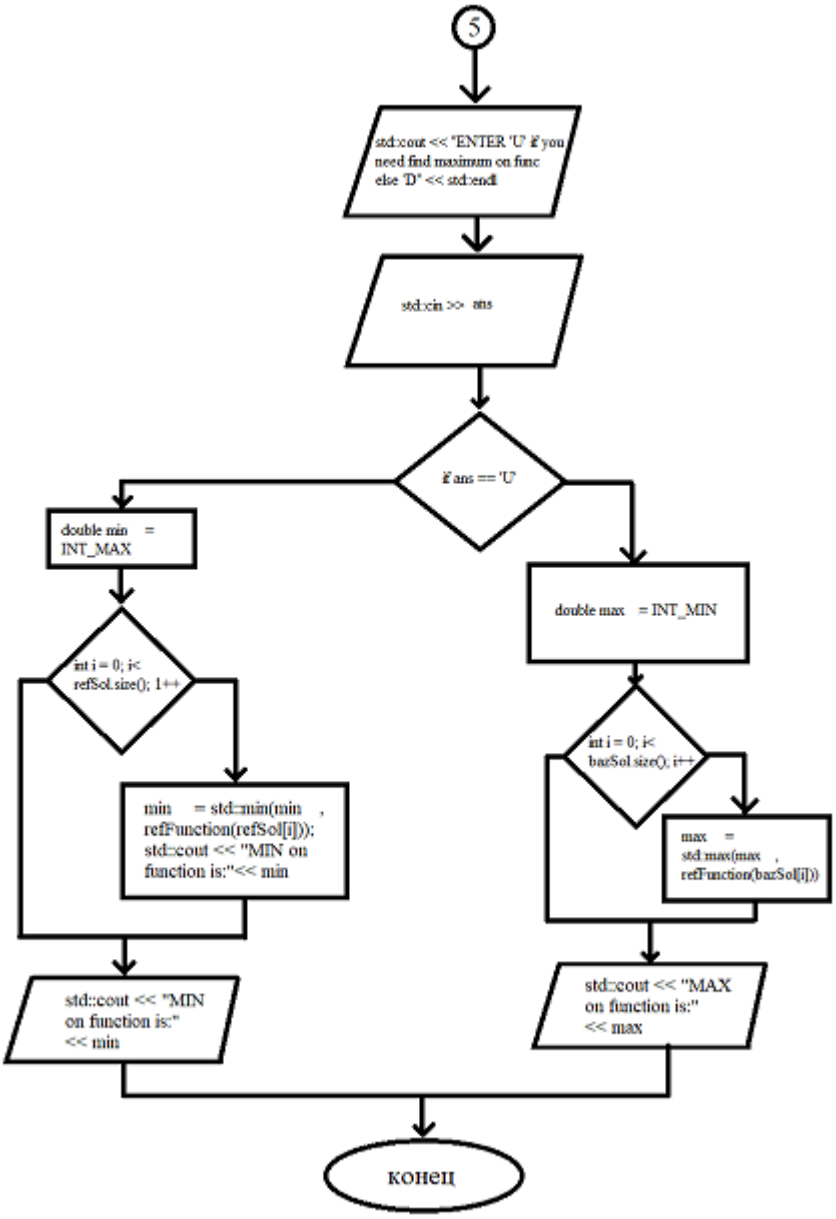
13.
$$\begin{cases} 8x_1 + x_2 - x_3 + 2x_5 + 3x_6 = 8 \\ x_1 - 5x_2 + 4x_3 + 6x_4 - x_6 = 9 \\ 4x_1 + 3x_2 - 2x_3 + 9x_4 + x_5 + 7x_6 = 1 \\ 5x_1 - 2x_2 + 2x_3 + 15x_4 + x_5 + 6x_6 = 10 \end{cases}$$

Блок-схема файла main.cpp:




1
int all =0:; all < combAll.size(); all++
int i = 0: i < a.size(); i++
int j = 0; j < combAlli].size(); j++
bool check = linearSearch(combAll[a], a])
!check
combAll[all].push_back(a])
std::vector> bazSol(0, std::vector(0))
int i=0: i combAll.size(); i++
SLAU currSLAU(solve2.nRows, solve2.nCols)
int r = 0; r< solve2.nRows i++
int c =0; c < combAll[i].size(); c++
currSLAU.matrix[r][c] = solve2.matrix[r] [combAll[[c]]
4
3
2


4
3
2
bool allGood = currSLAU.jordanGauss(currSLAU)
fabs(currSLAU.determinant(currSLAU)) > EPS && allGood
bazSol.push_back(currSLAU.bazisSolution(currSLAU, combAll[i]))
std::cout << "\nБазисное решение: "<<"\n";
int = 0; 1< bazSol.size(); i++
isRefSolution(bazSol[])
refSol.push_back(bazSol[i])
std::cout << "\nопорное решение: " << "\n";
int i= 0; i< bazSol.size(); i++
outputVector(refSol[i])
char ans = '\0';
5



Код:

```
#include <iostream>
#include <vector>
#include <cmath>
#include <windows.h>

#define min_value 2.2e-10
typedef struct SLAU {
// хранение матрицы элементов
    std::vector<std::vector<double>> matrix;
// хранение решений уравнений
    std::vector<double> solutions;
// количество строк и столбцов
    int nRows;
    int nCols;
// Конструкторы //
    explicit SLAU(int nr, int nc) {//конструктор структуры SLAU
        nRows = nr, nCols = nc; //задаём значения полям структуры
        matrix.resize(nRows);//выделяем память
        solutions.resize(nRows);
        for (int i = 0; i < nRows; i++) {
            matrix[i].resize(nCols);
        }
    }
    //конструктор второй, через прямое задание матрицы и решения
```

```
    template<typename Type>
    explicit SLAU(std::vector<std::vector<Type>> m,
std::vector<Type>Solutions) {
        nRows = m.size(), nCols = m[0].size();
        matrix.resize(nRows);
        solutions.resize(nRows);
        for (int i = 0; i < nRows; i++)
            matrix[i].resize(nCols);
        for (int i = 0; i < nRows; i++) {
            for (int j = 0; j < nCols; j++)
                matrix[i][j] = m[i][j];
            solutions[i] = Solutions[i];
        }
    }
// Прибавляет строку from к строке to
    void plusRows(int from, int to) {
        for (int i = 0; i < nCols; i++)
            matrix[to][i] += matrix[from][i];
        solutions[to] += solutions[from];
    }
// Прибавляет к строке индекс row строку b
    void plusRowsStr(SLAU &source, int row, std::vector<double> b) {
        for (int i = 0; i < nCols; i++)
            matrix[row][i] += b[i];
    }
// Умножает элементы строки row на коэффициент k
    void mulRowsOnNum(int row, double k) {
        for (int i = 0; i < nCols; i++)
            matrix[row][i] *= k;
        solutions[row] *= k;
    }
// Возвращает индекс строки с ненулевым элементом после строки с индексом
index
// Если такой строки нет возвращает -1.
    int findRowWithNoZeroPos(int index) {
        int numRow = -1;
        int i = index;
        while (numRow < 0 && i < matrix.size()) {
            if (fabs(matrix[i][index]) > min_value) {
                numRow = i;
                return numRow;
            } else
                i++;
        }
        return numRow;
    }
// делает на главной диагонали все элементы ненулевыми
    int checkMainDiagonal(SLAU &source) {
        for (int i = 0; i < source.nRows && source.nCols; i++)
            if (fabs(source.matrix[i][i]) < min_value) {
                if (findRowWithNoZeroPos(i) != -1) {
                    plusRows(findRowWithNoZeroPos(i), i);
                    return 1;
                } else
                    return 0;
            }
        return 1;
    }
// находит и удаляет пустую строку если такая есть
    void deleteEmptyRow(SLAU &source) {
        for (int i = 0; i < source.nRows; i++) {
            int j = 0;
            while (j < nCols && fabs(matrix[i][j]) < min_value)
                j++;
```

```
            if (j == nCols && fabs(solutions[i]) < min_value) {
                nRows--;
                for (int q = i; q < solutions.size() - 1; q++) {
                    matrix[q] = matrix[q + 1];
                    solutions[q] = solutions[q + 1];
                }
                matrix.resize(nRows);
                solutions.resize(nRows);
                i--;
            }
        }

        for(int i = 0; i < source.nRows-1; i++){
            for(int j = i+1; j < source.nRows; j++){
                 int flag = 1;
                 for(int g = 0; g < source.nCols; g++){
                     if(source.matrix[i][g] != source.matrix[j][g]){
                         flag = 0;
                         break;
                     }
                 }
                 if(flag){
                     nRows--;
                     for (int q = i; q < solutions.size() - 1; q++) {
                         matrix[q] = matrix[q + 1];
                         solutions[q] = solutions[q + 1];
                     }
                     matrix.resize(nRows);
                     solutions.resize(nRows);
                 }
            }
        }

    }
// рассчитываем определитель матрицы
    double determinant(SLAU source) {
        int i = 0;
        double determ = matrix[i][i];
        if (fabs(determ) < min_value)
            return 0;
        i++;
        while (i < nRows && i < nCols) {
            if (fabs(matrix[i][i]) > min_value) {
                determ *= matrix[i][i];
                i++;
            } else
                return 0;
        }
        return determ;
    }
// приводим систему уравнений к ступенчатому виду
    bool jordanGauss(SLAU &source) {
// удаляем пустые строки
        deleteEmptyRow(source);
// проверка наличия диагонали с ненулевыми элементами
        if (!checkMainDiagonal(source))
            return false;
// применение метода Гаусса-Жордана
        for (int i = 0; i < nRows; i++) {
            double mainEl = matrix[i][i];
            double k = 1 / mainEl;
// умножение строки на обратный коэффициент главного элемента
            mulRowsOnNum(i, k);
// приведение остальных строк к нулю
```

```
            for (int j = i + 1; j < nRows; j++) {
                if (fabs(matrix[j][i]) > min_value) {
                    double koef = (-1 / matrix[j][i]);
                    std::vector<double> strKoef(nCols);
// вычисление коэффициентов для операции замещения
                    for (int qq = 0; qq < nCols; qq++)
                        strKoef[qq] = matrix[i][qq] / koef;
// применение операции замещения
                    plusRowsStr(source, j, strKoef);
                    solutions[j] += solutions[i] / koef;
                }
            }
// удаление пустых строк и повторная проверка диагонали
            deleteEmptyRow(source);
            if (!checkMainDiagonal(source))
                return false;
        }
// обратный ход метода Гаусса-Жордана
        deleteEmptyRow(source);
        if (!checkMainDiagonal(source))
            return false;
        for (int i = 1; i < nRows; i++) {
            for (int j = i - 1; j >= 0; j--) {
                if (fabs(matrix[j][i]) > min_value) {
                    double koef = (-1 / matrix[j][i]);
                    std::vector<double> strKoef(nCols);
                    for (int k = 0; k < nCols; k++)
                        strKoef[k] = matrix[i][k] / koef;
                    plusRowsStr(source, j, strKoef);
                    solutions[j] += solutions[i] / koef;
                }
            }
            deleteEmptyRow(source);
            if (!checkMainDiagonal(source))
                return false;
        }
        if (!checkMainDiagonal(source)) {
            std::cout << "ERROR";
            return false;
        }
        return true;
    }
    std::vector<double> bazisSolution(SLAU source, std::vector<int>sequens) {
        std::vector<double> sol(sequens.size(), 0);
        // заполнение вектора значениями переменных из решения системы
        for (int i = 0; i < nRows; i++) {
            sol[sequens[i]] = solutions[i];
        }
        return sol;
    }
} SLAU;

void print_matrix(SLAU matrix, std::vector<int> comb){
    std::cout << "Базисные переменные: ";
    for(int i = 0; i < matrix.nRows; i++){
        std::cout << comb[i] + 1;
        if(i+1 != matrix.nRows){
            std::cout << ", ";
        }
    }
    std::cout << std::endl;
    std::cout << "Свободные переменные: ";
    for(int i = matrix.nRows; i < matrix.nCols; i++){
        std::cout << comb[i] + 1;
```

```
        if(i+1 != matrix.nCols){
            std::cout << ", ";
        }
    }
    std::cout << std::endl;
    for(int i = 0; i < matrix.nRows; i++){
        int flag = 0;
        for(int j = 0; j < matrix.nCols; j++){
            if(std::fabs(matrix.matrix[i][j]) > min_value){
                if(flag){
                    std::cout << " + ";
                }
                std::cout << matrix.matrix[i][j] << " * x" << j+1;
                flag = 1;
            }
        }
        std::cout << " = " << matrix.solutions[i] << std::endl;
    }

}

void print_answer(std::vector<double> ans){
    std::cout << "Найденные базисные переменные: ";
    for(int i = 0; i < ans.size(); i++){
        if(ans[i] != 0){
            std::cout << "x" << i+1 << " = " << ans[i] << "; ";
        }
    }
    std::cout << std::endl << std::endl;
}

// проверка на то, является ли базисным решение
bool isRefSolution(std::vector<double> sequens) {
// проверка значений на отрицательность
    for (int i = 0; i < sequens.size(); i++)
        if (sequens[i] < 0)
            return false;
    return true;
}
template<typename Type>
// вывод вектора
void outputVector(std::vector<Type> a) {
    std::cout << "{";
    for (int i = 0; i < a.size(); i++) {
        std::cout << a[i];
        if (i != a.size() - 1)
            std::cout << ", ";
    }
    std::cout << "}" << std::endl;
}
template<typename Type>
// ввод значений матрицы и вектора решений
void inputSLAU1(std::vector<std::vector<Type>> &matrix, std::vector<Type>
&Solutions) {
    for (int i = 0; i < matrix.size(); i++) {
        std::cout << "Введите коэффициенты уравнения " << i + 1 << std::endl;
        for (int j = 0; j < matrix[i].size(); j++)
            std::cin >> matrix[i][j];
        std::cout << "Введите решение уравнения" << std::endl;
        std::cin >> Solutions[i];
    }
}
// Создает сочетания из n по k и записывает их в вектор векторов
void generatingCombinations_(std::vector<int> &a, int k, std::vector<int>
```

```
&combination, std::vector<bool> &indicator, int sizeCombination, int fix,
std::vector<std::vector<int>> &solut) {
    for (int x = fix; x < a.size(); x++) {
        if (indicator.at(x)) {
            combination.at(sizeCombination) = a.at(x);
            indicator.at(x) = false;
            if (sizeCombination + 1 < k) {
                generatingCombinations_(a, k, combination,
indicator,sizeCombination + 1, x, solut);
                indicator.at(x) = true;
            } else {
                solut.push_back(combination);
                indicator.at(x) = true;
            }
        }
    }
}
std::vector<std::vector<int>> generatingCombinations(std::vector<int> &a, int
k) {
    static std::vector<int> combination(k);
    static std::vector<bool> indicator(a.size(), true);
    std::vector<std::vector<int>> ans(0, std::vector<int>(0));
    generatingCombinations_(a, k, combination, indicator, 0, 0, ans);
    return ans;
}
// линейный поиск элемента в векторе
bool linearSearch(std::vector<int> a, int b) {
    for (int i = 0; i < a.size(); i++)
        if (a[i] == b)
            return true;
    return false;
}
// Функция 1*x1 + 2*x2 + ... n*xn
double refFunction(std::vector<double> a) {
    double solution = 0;
    for (int i = 0; i < a.size(); i++)
        solution += (i+1)*a[i];
    return solution;
}

int main() {
    SetConsoleOutputCP(CP_UTF8);
    printf("Введите количество уравнений и количество неизвестных через
пробел\n");
    int row, col;
    std::cin >> row >> col;
    std::vector<std::vector<double>> matrix(row, std::vector<double>(col));
    std::vector<double> solutions(row);
    inputSLAU1(matrix, solutions);
    SLAU solve1(matrix, solutions);
    SLAU solve2 = solve1;
    if(!solve2.jordanGauss(solve2)){
        return 10;
    }
    std::vector<int> a(col, 0);
    for (int i = 0; i < col; i++)
        a[i] = i;
// генерация всех сочетаний без повторений
    std::vector<std::vector<int>> comb = generatingCombinations(a,
solve2.nRows);
// записываем полностью вектор, где первые row элементов не базисные
    std::vector<std::vector<int>> combAll = comb;
    for (int all = 0; all < combAll.size(); all++) {
        for (int i = 0; i < a.size(); i++) {
```

```
            for (int j = 0; j < combAll[j].size(); j++) {
                bool check = linearSearch(combAll[all], a[i]);
                if (!check)
                    combAll[all].push_back(a[i]);
            }
        }
    }
    std::vector<std::vector<double>> bazSol(0, std::vector<double>(0));
    std::cout << "\nВсе базисные виды системы:\n\n";
    for (int i = 0; i < combAll.size(); i++) {
        SLAU currSLAU(solve2.nRows, solve2.nCols);
        for (int r = 0; r < solve2.nRows; r++) {
            for (int c = 0; c < combAll[i].size(); c++)
                currSLAU.matrix[r][c] = solve2.matrix[r][combAll[i][c]];
            currSLAU.solutions[r] = solve2.solutions[r];
        }
        bool F = currSLAU.jordanGauss(currSLAU);
        print_matrix(currSLAU, combAll[i]);
        if (fabs(currSLAU.determinant(currSLAU)) > min_value && F) {
            bazSol.push_back(currSLAU.bazisSolution(currSLAU, combAll[i]));
            print_answer(bazSol[i]);
        }
    }
    std::vector<std::vector<double>> refSol(0, std::vector<double>(0));
    for (int i = 0; i < bazSol.size(); i++)
        if (isRefSolution(bazSol[i]))
            refSol.push_back(bazSol[i]);

    std::cout << "\nопорные решения: " << '\n';
    for (int i = 0; i < refSol.size(); i++)
        outputVector(refSol[i]);
    char ans = '\0';
    std::cout << "Введите 'U' если хотите найти максимум функции или
'D'\nесли хотите найти минимум" << std::endl;
    std::cin >> ans;
    std::cout << "Целевая функция:\n";
    for(int i = 0; i < solve2.nCols; i++){
        if(i != 0){
            std::cout << "+ ";
        }
        std::cout << i+1 << "*x" << i+1 << " ";
    }
    printf("\n");
    int flag_first = 0;
    if (ans == 'U') {
        double max = INT_MIN;
        for (int i = 0; i < refSol.size(); i++) {
            max = std::max(max, refFunction(refSol[i]));
        }
        for(int i = 0; i < refSol.size(); i++){
            if(refFunction(refSol[i]) == max){
                for(int j = 0; j < refSol[i].size(); j++){
                    if(refSol[i][j] != 0){
                        if(flag_first!=0){
                            std::cout << " + ";
                        }
                        flag_first = 1;
                        std::cout << refSol[i][j] << " * " << j+1;
                    }
                }
                break;
            }
        }
        std::cout << "= " << max << std::endl;
```

```
        std::cout << "Максимум функции = " << max;
    } else{
        double min = INT_MAX;
        for (int i = 0; i < refSol.size(); i++)
            min = std::min(min, refFunction(refSol[i]));

        for(int i = 0; i < refSol.size(); i++){
            if(refFunction(refSol[i]) == min){
                for(int j = 0; j < refSol[i].size(); j++){
                    if(refSol[i][j] != 0){
                        if(flag_first!=0){
                            std::cout << " + ";
                        }
                        flag_first = 1;

                        std::cout << refSol[i][j] << " * " << j+1;
                    }
                }
                break;
            }
        }
        std::cout << "= " << min << std::endl;
        std::cout << "Минимум функции = " << min;
    }
    return 0;
}
```

Запуск программы:

Введите количество уравнений и количество неизвестных через пробел
4 6
Введите коэффициенты уравнения 1
1 -5 4 6 0 -1
Введите решение уравнения
9
Введите коэффициенты уравнения 2
8 1 -1 0 2 3
Введите решение уравнения
8
Введите коэффициенты уравнения 3
4 3 -2 9 1 7
Введите решение уравнения
1
Введите коэффициенты уравнения 4
5 -2 2 15 1 6
Введите решение уравнения
10

Все базисные виды системы:

1)
Базисные переменные: 1, 2, 3
Свободные переменные: 4, 5, 6
1 * x1 + 0.714286 * x4 + 0.238095 * x5 + 0.571429 * x6 = 1.2381
1 * x2 + 17.5714 * x4 + -0.142857 * x5 + 7.85714 * x6 = -0.142857
1 * x3 + 23.2857 * x4 + -0.238095 * x5 + 9.42857 * x6 = 1.7619
Найденные базисные переменные: x1 = 1.2381; x2 = -0.142857; x3 = 1.7619;

2)
Базисные переменные: 1, 2, 4
Свободные переменные: 3, 5, 6
1 * x1 + -0.0306748 * x4 + 0.245399 * x5 + 0.282209 * x6 = 1.18405
1 * x2 + -0.754601 * x4 + 0.0368098 * x5 + 0.742331 * x6 = -1.47239
1 * x3 + 0.0429448 * x4 + -0.0102249 * x5 + 0.404908 * x6 = 0.0756646
Найденные базисные переменные: x1 = 1.18405; x2 = -1.47239; x4 = 0.0756646;

3)
Базисные переменные: 1, 2, 5
Свободные переменные: 3, 4, 6
1 * x1 + 1 * x4 + 24 * x5 + 10 * x6 = 3
1 * x2 + -0.6 * x4 + 3.6 * x5 + 2.2 * x6 = -1.2
1 * x3 + -4.2 * x4 + -97.8 * x5 + -39.6 * x6 = -7.4
Найденные базисные переменные: x1 = 3; x2 = -1.2; x5 = -7.4;

4)
Базисные переменные: 1, 2, 6
Свободные переменные: 3, 4, 5
1 * x1 + -0.0606061 * x4 + -0.69697 * x5 + 0.252525 * x6 = 1.13131
1 * x2 + -0.833333 * x4 + -1.83333 * x5 + 0.0555556 * x6 = -1.61111
1 * x3 + 0.106061 * x4 + 2.4697 * x5 + -0.0252525 * x6 = 0.186869
Найденные базисные переменные: x1 = 1.13131; x2 = -1.61111; x6 = 0.186869;

5)
Базисные переменные: 1, 3, 4
Свободные переменные: 2, 5, 6
1 * x1 + -0.0406504 * x4 + 0.243902 * x5 + 0.252033 * x6 = 1.2439
1 * x2 + -1.3252 * x4 + -0.0487805 * x5 + -0.98374 * x6 = 1.95122
1 * x3 + 0.0569106 * x4 + -0.00813008 * x5 + 0.447154 * x6 = -0.00813008

Найденные базисные переменные: x1 = 1.2439; x3 = 1.95122; x4 = -0.00813008;

6)
Базисные переменные: 1, 3, 5
Свободные переменные: 2, 4, 6
1 * x1 + 1.66667 * x4 + 30 * x5 + 13.6667 * x6 = 1
1 * x2 + -1.66667 * x4 + -6 * x5 + -3.66667 * x6 = 2
1 * x3 + -7 * x4 + -123 * x5 + -55 * x6 = 1
Найденные базисные переменные: x1 = 1; x3 = 2; x5 = 1;

7)
Базисные переменные: 1, 3, 6
Свободные переменные: 2, 4, 5
1 * x1 + -0.0727273 * x4 + -0.563636 * x5 + 0.248485 * x6 = 1.24848
1 * x2 + -1.2 * x4 + 2.2 * x5 + -0.0666667 * x6 = 1.93333
1 * x3 + 0.127273 * x4 + 2.23636 * x5 + -0.0181818 * x6 = -0.0181818
Найденные базисные переменные: x1 = 1.24848; x3 = 1.93333; x6 = -0.0181818;

8)
Базисные переменные: 1, 4, 5
Свободные переменные: 2, 3, 6
1 * x1 + -6.66667 * x4 + 5 * x5 + -4.66667 * x6 = 11
1 * x2 + 0.277778 * x4 + -0.166667 * x5 + 0.611111 * x6 = -0.333333
1 * x3 + 27.1667 * x4 + -20.5 * x5 + 20.1667 * x6 = -40
Найденные базисные переменные: x1 = 11; x4 = -0.333333; x5 = -40;

9)
Базисные переменные: 1, 4, 6
Свободные переменные: 2, 3, 5
1 * x1 + -0.380165 * x4 + 0.256198 * x5 + 0.231405 * x6 = 1.7438
1 * x2 + -0.545455 * x4 + 0.454545 * x5 + -0.030303 * x6 = 0.878788
1 * x3 + 1.34711 * x4 + -1.01653 * x5 + 0.0495868 * x6 = -1.98347
Найденные базисные переменные: x1 = 1.7438; x4 = 0.878788; x6 = -1.98347;

10)
Базисные переменные: 1, 5, 6
Свободные переменные: 2, 3, 4
1 * x1 + -4.54545 * x4 + 3.72727 * x5 + 7.63636 * x6 = 8.45455
1 * x2 + 18 * x4 + -15 * x5 + -33 * x6 = -29
1 * x3 + 0.454545 * x4 + -0.272727 * x5 + 1.63636 * x6 = -0.545455

Найденные базисные переменные: x1 = 8.45455; x5 = -29; x6 = -0.545455;

11)
Базисные переменные: 2, 3, 4
Свободные переменные: 1, 5, 6
1 * x1 + -24.6 * x4 + -6 * x5 + -6.2 * x6 = -30.6
1 * x2 + -32.6 * x4 + -8 * x5 + -9.2 * x6 = -38.6
1 * x3 + 1.4 * x4 + 0.333333 * x5 + 0.8 * x6 = 1.73333
Найденные базисные переменные: x2 = -30.6; x3 = -38.6; x4 = 1.73333;

12)
Базисные переменные: 2, 3, 5
Свободные переменные: 1, 4, 6
1 * x1 + 0.6 * x4 + 18 * x5 + 8.2 * x6 = 0.6
1 * x2 + 1 * x4 + 24 * x5 + 10 * x6 = 3
1 * x3 + 4.2 * x4 + 3 * x5 + 2.4 * x6 = 5.2
Найденные базисные переменные: x2 = 0.6; x3 = 3; x5 = 5.2;

13)
Базисные переменные: 2, 3, 6
Свободные переменные: 1, 4, 5
1 * x1 + -13.75 * x4 + 7.75 * x5 + -3.41667 * x6 = -17.1667
1 * x2 + -16.5 * x4 + 11.5 * x5 + -4.16667 * x6 = -18.6667
1 * x3 + 1.75 * x4 + 1.25 * x5 + 0.416667 * x6 = 2.16667
Найденные базисные переменные: x2 = -17.1667; x3 = -18.6667; x6 = 2.16667;

14)
Базисные переменные: 2, 4, 5
Свободные переменные: 1, 3, 6
1 * x1 + -0.15 * x4 + -0.75 * x5 + 0.7 * x6 = -1.65
1 * x2 + 0.0416667 * x4 + 0.0416667 * x5 + 0.416667 * x6 = 0.125
1 * x3 + 4.075 * x4 + -0.125 * x5 + 1.15 * x6 = 4.825
Найденные базисные переменные: x2 = -1.65; x4 = 0.125; x5 = 4.825;

15)
Базисные переменные: 2, 4, 6
Свободные переменные: 1, 3, 5
1 * x1 + -2.63043 * x4 + -0.673913 * x5 + -0.608696 * x6 = -4.58696
1 * x2 + -1.43478 * x4 + 0.0869565 * x5 + -0.362319 * x6 = -1.62319
1 * x3 + 3.54348 * x4 + -0.108696 * x5 + 0.869565 * x6 = 4.19565
Найденные базисные переменные: x2 = -4.58696; x4 = -1.62319; x6 = 4.19565;

16)
Базисные переменные: 2, 5, 6
Свободные переменные: 1, 3, 4
1 * x1 + -0.22 * x4 + -0.82 * x5 + -1.68 * x6 = -1.86
1 * x2 + 3.96 * x4 + -0.24 * x5 + -2.76 * x6 = 4.48
1 * x3 + 0.1 * x4 + 0.1 * x5 + 2.4 * x6 = 0.3
Найденные базисные переменные: x2 = -1.86; x5 = 4.48; x6 = 0.3;

17)
Базисные переменные: 3, 4, 5
Свободные переменные: 1, 2, 6
1 * x1 + 0.2 * x4 + -1.33333 * x5 + -0.933333 * x6 = 2.2
1 * x2 + 0.0333333 * x4 + 0.0555556 * x5 + 0.455556 * x6 = 0.0333333
1 * x3 + 4.1 * x4 + -0.166667 * x5 + 1.03333 * x6 = 5.1
Найденные базисные переменные: x3 = 2.2; x4 = 0.0333333; x5 = 5.1;

18)
Базисные переменные: 3, 4, 6
Свободные переменные: 1, 2, 5
1 * x1 + 3.90323 * x4 + -1.48387 * x5 + 0.903226 * x6 = 6.80645
1 * x2 + -1.77419 * x4 + 0.129032 * x5 + -0.44086 * x6 = -2.21505
1 * x3 + 3.96774 * x4 + -0.16129 * x5 + 0.967742 * x6 = 4.93548
Найденные базисные переменные: x3 = 6.80645; x4 = -2.21505; x6 = 4.93548;

19)
Базисные переменные: 3, 5, 6
Свободные переменные: 1, 2, 4
1 * x1 + 0.268293 * x4 + -1.21951 * x5 + 2.04878 * x6 = 2.26829
1 * x2 + 4.02439 * x4 + -0.292683 * x5 + -2.26829 * x6 = 5.02439
1 * x3 + 0.0731707 * x4 + 0.121951 * x5 + 2.19512 * x6 = 0.0731707
Найденные базисные переменные: x3 = 2.26829; x5 = 5.02439; x6 = 0.0731707;

20)
Базисные переменные: 4, 5, 6
Свободные переменные: 1, 2, 3
1 * x1 + 0.130952 * x4 + -0.595238 * x5 + 0.488095 * x6 = 1.10714
1 * x2 + 4.32143 * x4 + -1.64286 * x5 + 1.10714 * x6 = 7.53571
1 * x3 + -0.214286 * x4 + 1.42857 * x5 + -1.07143 * x6 = -2.35714
Найденные базисные переменные: x4 = 1.10714; x5 = 7.53571; x6 = -2.35714;

опорные решения:
{1, 0, 2, 0, 1, 0}
{0, 0.6, 3, 0, 5.2, 0}
{0, 0, 2.2, 0.0333333, 5.1, 0}
{0, 0, 2.26829, 0, 5.02439, 0.0731707}
Введите 'U' если хотите найти максимум функции или 'D'
если хотите найти минимум
U
Целевая функция:
1*x1 + 2*x2 + 3*x3 + 4*x4 + 5*x5 + 6*x6
0.6 * 2 + 3 * 3 + 5.2 * 5= 36.2
Максимум функции = 36.2
Process finished with exit code 0

Аналитическое решение

13. $$\begin{cases} 8x_1 + x_2 - x_3 + 2x_5 + 3x_6 = 8 \\ x_1 - 5x_2 + 4x_3 + 6x_4 - x_6 = 9 \\ 4x_1 + 3x_2 - 2x_3 + 9x_4 + x_5 + 7x_6 = 1 \\ 5x_1 - 2x_2 + 2x_3 + 15x_4 + x_5 + 6x_6 = 10 \end{cases}$$

поменяли местами строки 1 и 2.

$$\begin{pmatrix} 1 & -5 & 4 & 6 & 0 & -1 & 9 \\ 8 & 1 & -1 & 0 & 2 & 3 & 8 \\ 4 & 3 & -2 & 9 & 1 & 7 & 1 \\ 5 & -2 & 2 & 15 & 1 & 6 & 10 \end{pmatrix}$$

умножили первую строку на -8 и сложили с 2, умножили первую строку на -4 и сложили с 3, умножили первую строку на -5 и сложили с 4.

$$\begin{pmatrix} 1 & -5 & 4 & 6 & 0 & -1 & 9 \\ 0 & 41 & -33 & -48 & 2 & 11 & -64 \\ 0 & 23 & -18 & -15 & 1 & 11 & -35 \\ 0 & 23 & -18 & -15 & 1 & 11 & -35 \end{pmatrix}$$

Убрали четвёртую строку так как она такая же, как и третья, затем разделили вторую строку на 41, затем умножили её на -23 и сложили с 3.

$$\begin{pmatrix} 1 & -5 & 4 & 6 & 0 & -1 & 9 \\ 0 & 1 & -\frac{33}{41} & -\frac{48}{41} & \frac{2}{41} & \frac{11}{41} & -\frac{64}{41} \\ 0 & 0 & -18 + \frac{33 * 23}{41} & -15 + \frac{48 * 23}{41} & 1 - \frac{46}{41} & 11 - \frac{11 * 23}{41} & -35 + \frac{64 * 23}{41} \end{pmatrix}$$

Вычислили третью строку.

$$\begin{pmatrix} 1 & -5 & 4 & 6 & 0 & -1 & 9 \\ 0 & 1 & -\frac{33}{41} & -\frac{48}{41} & \frac{2}{41} & \frac{11}{41} & -\frac{64}{41} \\ 0 & 0 & \frac{21}{41} & \frac{489}{41} & -\frac{5}{41} & \frac{198}{41} & \frac{37}{41} \end{pmatrix}$$

Умножили третью строку на $\frac{41}{21}$.

$$\begin{pmatrix} 1 & -5 & 4 & 6 & 0 & -1 & 9 \\ 0 & 1 & -\frac{33}{41} & -\frac{48}{41} & \frac{2}{41} & \frac{11}{41} & -\frac{64}{41} \\ 0 & 0 & 1 & \frac{489}{21} & -\frac{5}{21} & \frac{198}{21} & \frac{37}{21} \end{pmatrix}$$

Умножили третью строку на $\frac{33}{41}$ и сложили со второй, затем умножили третью строку на -4 и сложили с первой.

$$\begin{pmatrix} 1 & -5 & 0 & -\frac{1830}{21} & \frac{20}{21} & -\frac{813}{21} & \frac{41}{21} \\ 0 & 1 & 0 & \frac{15129}{861} & -\frac{123}{861} & \frac{6765}{861} & -\frac{123}{861} \\ 0 & 0 & 1 & \frac{489}{21} & -\frac{5}{21} & \frac{198}{21} & \frac{37}{21} \end{pmatrix}$$

Умножили вторую строку на -5 и сложили с первой.

$$\begin{pmatrix} 1 & 0 & 0 & \frac{615}{861} & \frac{205}{861} & \frac{492}{861} & \frac{1066}{861} \\ 0 & 1 & 0 & \frac{15129}{861} & -\frac{123}{861} & \frac{6765}{861} & -\frac{123}{861} \\ 0 & 0 & 1 & \frac{489}{21} & -\frac{5}{21} & \frac{198}{21} & \frac{37}{21} \end{pmatrix}$$

Сократили дроби.

$$\begin{pmatrix} 1 & 0 & 0 & \frac{5}{7} & \frac{5}{21} & \frac{4}{7} & \frac{26}{21} \\ 0 & 1 & 0 & \frac{123}{7} & -\frac{1}{7} & \frac{55}{7} & -\frac{1}{7} \\ 0 & 0 & 1 & \frac{163}{7} & -\frac{5}{21} & \frac{66}{7} & \frac{37}{21} \end{pmatrix}$$

Базисные	свободные	Базисное решение
1, 2, 3	4, 5, 6	$\{\frac{26}{21}, -\frac{1}{7}, \frac{37}{21}, 0, 0, 0\}$
2, 3, 4	1, 5, 6	$\{0, -\frac{153}{5}, -\frac{193}{5}, \frac{26}{15}, 0, 0\}$
2, 3, 5	1, 4, 6	$\{0, \frac{3}{5}, 3, 0, \frac{26}{5}, 0\}$
2, 3, 6	1, 4, 5	$\{0, -\frac{103}{6}, -\frac{56}{3}, 0, 0, \frac{13}{6}\}$
1, 3, 4	2, 5, 6	$\{\frac{51}{41}, 0, \frac{80}{41}, -\frac{1}{123}, 0, 0\}$
1, 3, 5	2, 4, 6	$\{1, 0, 2, 0, 1, 0\}$
1, 3, 6	2, 4, 5	$\{\frac{206}{165}, 0, \frac{29}{15}, 0, 0, -\frac{1}{55}\}$
1, 2, 4	3, 5, 6	$\{\frac{80117}{37653}, -\frac{240}{163}, 0, \frac{37}{489}, 0, 0\}$
1, 2, 5	3, 4, 6	$\{\frac{911}{231}, -\frac{6}{5}, 0, 0, -\frac{37}{5}, 0\}$
1, 2, 6	3, 4, 5	$\{\frac{1438}{693}, -\frac{29}{18}, 0, 0, 0, \frac{37}{198}\}$
1, 4, 5	2, 3, 6	$\{11, 0, 0, -\frac{1}{3}, -40, 0\}$
1, 4, 6	2, 3, 5	$\{\frac{5}{3}, 0, 0, \frac{11}{13}, 0, -\frac{17}{9}\}$

1, 5, 6	2, 3, 4	$\{\frac{169}{20}, 0, 0, 0, -29, \frac{6}{11}\}$
2, 4, 5	1, 3, 6	{0, -1.65, 0, 0.125, 4.8, 0}
2, 4, 6	1, 3, 5	{0, -4.5, 0, -1.6, 0, 4}
2, 5, 6	1, 3, 4	{0, -1.86, 0, 0, 4.48, 0.3}
3, 4, 5	1, 2, 6	$\{0, 0, 2.2, \frac{1}{30}, 5.1, 0\}$
3, 4, 6	1, 2, 5	{0, 0, 6.8, -2.2, 0, 4.9}
3, 5, 6	1, 2, 4	{0, 0, 2.26, 0, 5.02, 0.07}
4, 5, 6	1, 2, 3	{0, 0, 0, 1.1, 7.5, -2.4}

Целевая функция: $x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$

0.6*2 + 3*3 + 5.2*5 = 36.2 (Максимум)

Решение вычисленное программой совпало с вычисленным аналитически.

Вывод: метод Гаусса-Жордана является эффективным инструментом для решения систем линейных уравнений и нахождения базисных решений. Реализация метода на языке C++ продемонстрировала его применимость для решения практических задач. Метод простого перебора опорных решений позволяет находить оптимальные решения в задачах линейного программирования, хотя и имеет ограничения по производительности для больших систем.