

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)



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ 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-223
Пахомов Владислав Андреевич

Проверили:
проф. Вирченко Юрий Петрович

Белгород 2024 г.

Лабораторная работа №1

Исследование множества опорных планов системы ограничений задачи линейного программирования (задачи ЛП) в канонической форме.

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.

Вариант 10

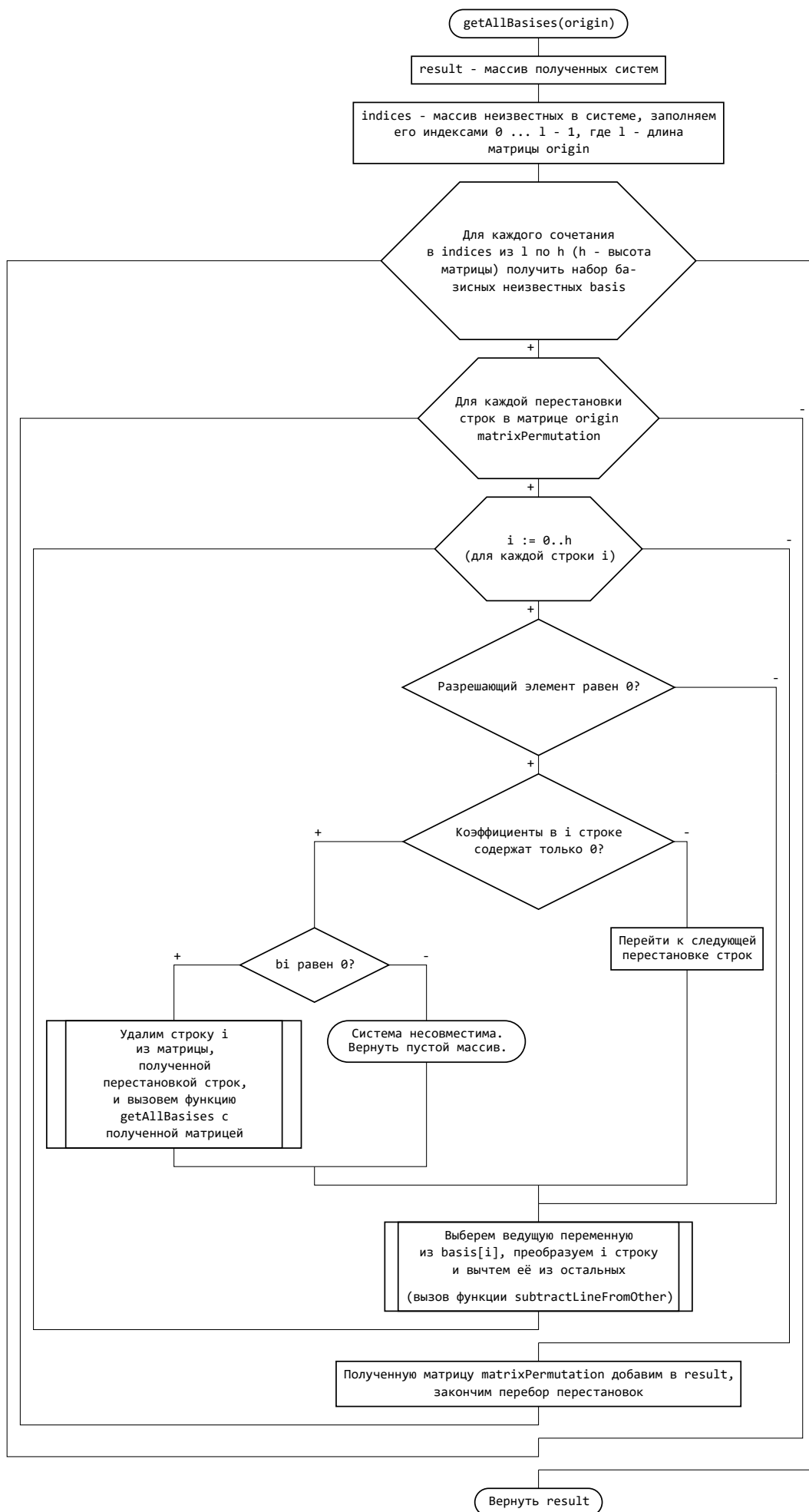
$$\begin{cases} x_1 - 4x_2 + 8x_3 + 9x_4 - 3x_5 - x_6 = 87 \\ 8x_1 + x_2 - 3x_3 + 4x_4 + 5x_5 + 6x_6 = 11 \\ 4x_1 + x_3 + 3x_4 - 2x_5 - 5x_6 = 17 \\ -3x_1 - 4x_2 + 7x_3 + 6x_4 - x_5 + 4x_6 = 70 \end{cases}$$

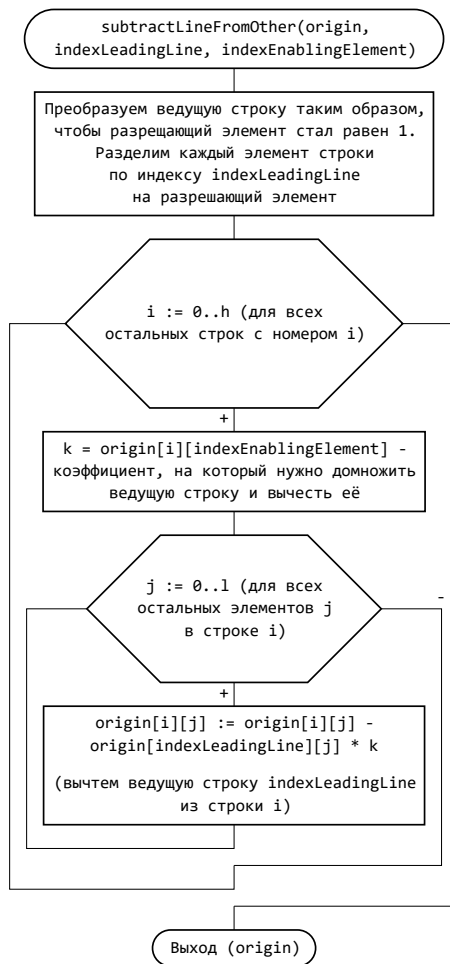
Задание 1

Составить программу для отыскания всех базисных видов системы линейных уравнений.

Блок-схемы:







Исходный код:

```

void subtractLineFromOther(std::vector<std::vector<double>>& origin, int indexLeadingLine, int indexEnablingElement) {
    // Преобразуем ведущую строку таким образом, чтобы разрежающий элемент стал равен 1.
    double originEnablingElement = origin[indexLeadingLine][indexEnablingElement];
    for (int i = 0; i < origin[indexLeadingLine].size(); i++)
        // Разделим каждый элемент строки по индексу indexLeadingLine
        // на разрежающий элемент
        origin[indexLeadingLine][i] /= originEnablingElement;

    // Для всех остальных строк с номером i
    for (int i = 0; i < origin.size(); i++) {
        if (i == indexLeadingLine) continue;

        // k - коэффициент, на который нужно домножить ведущую строку и вычесть её
        double k = origin[i][indexEnablingElement];

        // Для всех остальных элементов j в строке i
        for (int j = 0; j < origin[indexLeadingLine].size(); j++)
            // Вычтем ведущую строку indexLeadingLine из строки i
            origin[i][j] -= origin[indexLeadingLine][j] * k;
    }
}

// Вспомогательная СД
struct Basis {
    std::vector<int> indices;
    std::vector<std::vector<double>> matrix;

```

```

};

std::vector<Basis> getAllBasises(std::vector<std::vector<double>> origin) {
    // result - массив полученных систем
    std::vector<Basis> result;

    // indices - массив неизвестных в системе, заполняем
    // его индексами 0 ... L - 1, где L - длина
    // матрицы origin
    std::vector<int> indices;
    for (int i = 0; i < origin[0].size() - 1; i++)
        indices.push_back(i);

    /*
    Для каждого сочетания в indices из L по h (h - высота матрицы)
    получим набор базисных неизвестных basis.
    */
    for (auto basis : getCombinations(indices, origin.size())) {
        // Для каждой перестановки строк в матрице origin matrixPermutation
        for (auto matrixPermutation : getPermutations(origin)) {
            auto copyMatrixPermutation = matrixPermutation;
            bool badPermutation = false;

            // Для каждой строки в матрице (ввод счётчика строк i)
            for (int i = 0; i < matrixPermutation.size(); i++) {
                // Разрешающий элемент равен 0?
                if (std::abs(matrixPermutation[i][basis[i]]) < EPS) {
                    bool allZeros = true;
                    for (int j = 0; (j < matrixPermutation[i].size() - 1) && allZeros; j++) {
                        if (std::abs(matrixPermutation[i][j]) > EPS) {
                            // Неудачное расположение строк, необходимо
                            // перейти к другой перестановке
                            badPermutation = true;
                            allZeros = false;
                            break;
                        }
                    }
                }
            }

            // Коэффициенты в i строке содержат только 0?
            if (!allZeros) {
                // Перейти к следующей перестановке строк
                break;
            }

            // b_i равен 0?
            if (std::abs(matrixPermutation[i].back()) > EPS) {
                // Система несовместима. Вернуть пустой массив.
                return {};
            }

            /*
            Удалить строку i из матрицы, полученной перестановкой строк,
            и вызвать функцию getAllBasises с полученной матрицей.
            */
        }
    }
}

```

```

        if (allZeros) {
            copyMatrixPermutation.erase(copyMatrixPermutation.begin() + i);
            return getAllBasises(copyMatrixPermutation);
        }

        break;
    }

    // Выберем ведущую переменную из basis[i], преобразуем i строку и вычтем её из остальных
    subtractLineFromOther(matrixPermutation, i, basis[i]);
}

if (badPermutation) continue;

// Полученную матрицу matrixPermutation добавим в result, закончим перебор перестановок
result.push_back({basis, matrixPermutation});
break;
}
}

return result;
}

```

Ссылка на репозиторий

```

#include <iostream>
#include <iomanip>
#include <windows.h>

#include "../libs/alg/lab1/task1.hpp"

int main() {
    SetConsoleOutputCP(CP_UTF8);
    // Инициализируем матрицу
    std::vector<std::vector<double>> matrix = {
        {1, -4, 8, 9, -3, -1, 87},
        {8, 1, -3, 4, 5, 6, 11},
        {4, 0, 1, 3, -2, -5, 17},
        {-3, -4, 7, 6, -1, 4, 70}};

    // Получаем все базисы для матрицы
    auto res = getAllBasises(matrix);

    // Выводим базисы
    std::cout <<
        "\n===== \n";
    for (auto& matrix : res) {
        std::cout << "Выбранные базисные переменные: ";
        for (auto& bas : matrix.indices) {
            std::cout << "x" << (bas + 1) << " ";
        }
        std::cout << "\n\nПолученная система: " << std::endl;
    }
}

```


Полученная система:

a1	a2	a3	a4	a5	a6	b
-1.7987	-0.655844	1	0	0	1.35065	6.05844
1.48701	0.0584416	0	1	0	-0.506494	4.91558
-0.668831	-0.24026	0	0	1	2.41558	1.9026

=====

Выбранные базисные переменные: x2 x5 x6

Полученная система:

a1	a2	a3	a4	a5	a6	b
-4.33333	1	-2	-5.33333	0	0	-38.3333
6.58974	0	0.0769231	4.97436	1	0	26.8205
-3.4359	-0	-0.230769	-2.58974	-0	1	-14.1282

=====

Выбранные базисные переменные: x2 x4 x6

Полученная система:

a1	a2	a3	a4	a5	a6	b
2.73196	1	-1.91753	0	1.07216	0	-9.57732
1.32474	0	0.0154639	1	0.201031	0	5.39175
-0.00515464	0	-0.190722	0	0.520619	1	-0.164948

=====

Выбранные базисные переменные: x2 x4 x5

Полученная система:

a1	a2	a3	a4	a5	a6	b
2.74257	1	-1.52475	0	0	-2.05941	-9.23762
1.32673	0	0.0891089	1	0	-0.386139	5.45545
-0.00990099	0	-0.366337	0	1	1.92079	-0.316832

=====

Выбранные базисные переменные: x2 x3 x6

Полученная система:

a1	a2	a3	a4	a5	a6	b
167	1	0	124	26	0	659
85.6667	0	1	64.6667	13	0	348.667
16.3333	0	0	12.3333	3	1	66.3333

=====

Выбранные базисные переменные: x2 x3 x5

Полученная система:

a1	a2	a3	a4	a5	a6	b
25.4444	1	0	17.1111	0	-8.66667	84.1111
14.8889	0	1	11.2222	0	-4.33333	61.2222
5.44444	0	0	4.11111	1	0.333333	22.1111

=====

Выбранные базисные переменные: x2 x3 x4

Полученная система:

a1	a2	a3	a4	a5	a6	b
2.78378	1	0	0	-4.16216	-10.0541	-7.91892
0.027027	0	1	0	-2.72973	-5.24324	0.864865
1.32432	0	0	1	0.243243	0.0810811	5.37838

=====

Выбранные базисные переменные: x1 x5 x6

Полученная система:

a1	a2	a3	a4	a5	a6	b
1	-0.230769	0.461538	1.23077	0	0	8.84615
-0	1.52071	-2.9645	-3.13609	1	0	-31.4734
0	-0.792899	1.35503	1.63905	0	1	16.2663

Выбранные базисные переменные: x1 x4 x6

Полученная система:

a1	a2	a3	a4	a5	a6	b
1	0.366038	-0.701887	0	0.392453	0	-3.50566
0	-0.484906	0.945283	1	-0.318868	0	10.0358
0	0.00188679	-0.19434	0	0.522642	1	-0.183019

Выбранные базисные переменные: x1 x4 x5

Полученная система:

a1	a2	a3	a4	a5	a6	b
1	0.364621	-0.555957	0	0	-0.750903	-3.36823
0	-0.483755	0.826715	1	0	0.610108	9.92419
0	0.00361011	-0.371841	0	1	1.91336	-0.350181

Выбранные базисные переменные: x1 x3 x6

Полученная система:

a1	a2	a3	a4	a5	a6	b
1	0.00598802	0	0.742515	0.155689	0	3.94611
0	-0.512974	1	1.05788	-0.337325	0	10.6168
0	-0.0978044	0	0.205589	0.457086	1	1.88024

Выбранные базисные переменные: x1 x3 x5

Полученная система:

a1	a2	a3	a4	a5	a6	b
1	0.0393013	0	0.672489	0	-0.340611	3.30568
0	-0.585153	1	1.20961	0	0.737991	12.0044
0	-0.213974	0	0.449782	1	2.18777	4.11354

Выбранные базисные переменные: x1 x3 x4

Полученная система:

a1	a2	a3	a4	a5	a6	b
1	0.359223	0	0	-1.49515	-3.61165	-2.84466
0	-0.00970874	1	0	-2.68932	-5.14563	0.941748
0	-0.475728	0	1	2.2233	4.86408	9.14563

Выбранные базисные переменные: x1 x2 x6

Полученная система:

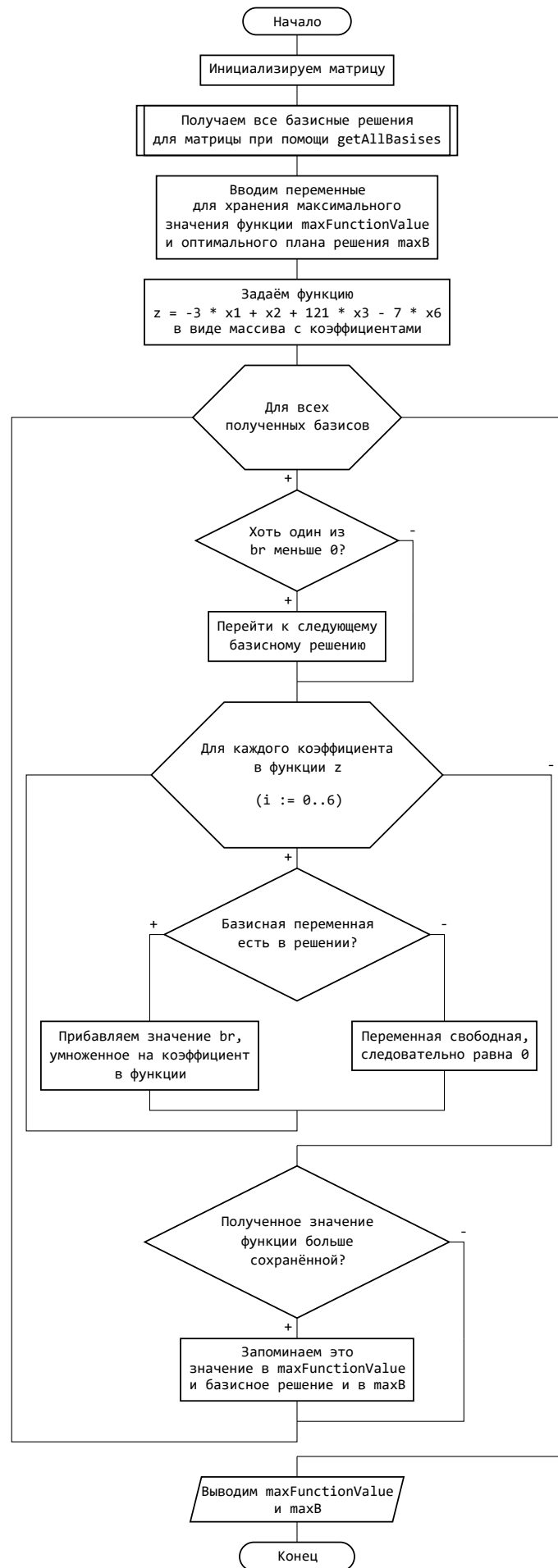
a1	a2	a3	a4	a5	a6	b
1	0	0.0116732	0.754864	0.151751	0	4.07004
-0	1	-1.94942	-2.06226	0.657588	0	-20.6965

0	0	-0.190661	0.00389105	0.521401	1	-0.143969
=====						
Выбранные базисные переменные: x1 x2 x5						
Полученная система:						
a1	a2	a3	a4	a5	a6	b
1	0	0.0671642	0.753731	0	-0.291045	4.11194
-0	1	-1.70896	-2.06716	0	-1.26119	-20.5149
0	0	-0.365672	0.00746269	1	1.91791	-0.276119
=====						
Выбранные базисные переменные: x1 x2 x4						
Полученная система:						
a1	a2	a3	a4	a5	a6	b
1	0	37	0	-101	-194	32
0	1	-103	0	277	530	-97
0	0	-49	1	134	257	-37
=====						
Выбранные базисные переменные: x1 x2 x3						
Полученная система:						
a1	a2	a3	a4	a5	a6	b
1	0	0	0.755102	0.183673	0.0612245	4.06122
-0	1	0	-2.10204	-4.67347	-10.2245	-19.2245
-0	-0	1	-0.0204082	-2.73469	-5.2449	0.755102
=====						

Задание 2

Организовать отбор опорных планов среди всех базисных решений, а также нахождение оптимального опорного плана методом прямого перебора. Целевая функция выбирается произвольно.

Блок-схемы:



Исходный код:

```

#include <iostream>
#include <iomanip>
#include <windows.h>
#include <limits>
#include <algorithm>

#include "../libs/alg/lab1/task1.tpp"

int main() {
    SetConsoleOutputCP(CP_UTF8);
    // Инициализируем матрицу
    std::vector<std::vector<double>> matrix = {
        {1, -4, 8, 9, -3, -1, 87},
        {8, 1, -3, 4, 5, 6, 11},
        {4, 0, 1, 3, -2, -5, 17},
        {-3, -4, 7, 6, -1, 4, 70}};

    // Получаем все базисные решения для матрицы при помощи getAllBasises
    auto res = getAllBasises(matrix);

    // Вводим переменные для хранения максимального значения функции и оптимального плана решения
    double maxFunctionValue = std::numeric_limits<double>::min();
    std::vector<double> maxB;
    // Задаём функцию  $z = -3 * x_1 + x_2 + 121 * x_3 - 7 * x_6$ 
    std::vector<double> function = {-3, 1, 121, 0, 0, -7};

    // Для всех полученных базисов
    for (auto &basis : res) {
        bool isAllBsMoreOrEqualToZero = true;
        for (int i = 0; i < basis.matrix.size() && isAllBsMoreOrEqualToZero; i++) {
            if (basis.matrix[i].back() < EPS)
                isAllBsMoreOrEqualToZero = false;
        }

        // Хотя один из br меньше 0?
        if (!isAllBsMoreOrEqualToZero) {
            // Перейти к следующему базисному решению
            continue;
        }

        double z = 0;
        std::vector<double> B;
        // Для каждого коэффициента в функции z
        for (int i = 0; i < function.size(); i++) {
            // Базисная переменная есть в решении?
            if (std::find(basis.indices.begin(), basis.indices.end(), i) != basis.indices.end()) {
                for (int j = 0; j < basis.matrix.size(); j++) {
                    if (std::abs(basis.matrix[j][i] - 1.0) < EPS) {
                        // Прибавляем значение br, умноженное на коэффициент в функции
                        z += function[i] * basis.matrix[j].back();
                        B.push_back(basis.matrix[j].back());
                        break;
                    }
                }
            }
        }
    }
}

```

```

    } else {
        // Переменная свободная, следовательно равна 0
        B.push_back(0);
    }
}

// Полученное значение функции больше сохранённой?
if (z > maxFunctionValue) {
    // Запоминаем его и базисное решение
    maxFunctionValue = z;
    maxB = B;
}

// Выводим значение z и базисное решение
std::cout << "Zmax: " << maxFunctionValue << "\n" << "Оптимальное решение: ";
for (int i = 0; i < maxB.size(); i++) {
    std::cout << maxB[i] << " ";
}
}

```

Ссылка на репозиторий

Результаты выполнения программы:

```

Zmax: 42383.3
Оптимальное решение: 0 659 348.667 0 0 66.3333

```

Задание 3

Решить одну из следующих ниже задач вручную (подготовить тестовые данные).

Решение: Построим расширенную матрицу:

$$\begin{pmatrix} 1 & -4 & 8 & 9 & -3 & -1 & 87 \\ 8 & 1 & -3 & 4 & 5 & 6 & 11 \\ 4 & 0 & 1 & 3 & -2 & -5 & 17 \\ -3 & -4 & 7 & 6 & -1 & 4 & 70 \end{pmatrix}$$

В качестве базисных переменных выберем x_2 , x_3 и x_5 . Выберем ведущий элемент x_2 в первый строке:

$$\begin{pmatrix} 1 & \parallel -4 \parallel & 8 & 9 & -3 & -1 & 87 \\ 8 & 1 & -3 & 4 & 5 & 6 & 11 \\ 4 & 0 & 1 & 3 & -2 & -5 & 17 \\ -3 & -4 & 7 & 6 & -1 & 4 & 70 \end{pmatrix}$$

Разделим ведущую строку на -4. Вычтем 1 строку из 2, домножив её на -1 и сложив; 3 строка останется неизменной; из 4, домножив на 4 и сложив:

$$\begin{pmatrix} -\frac{1}{4} & \|1\| & -2 & -2\frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -21\frac{3}{4} \\ 8\frac{1}{4} & 0 & -1 & 6\frac{1}{4} & 4\frac{1}{4} & 5\frac{3}{4} & 32\frac{3}{4} \\ 4 & 0 & 1 & 3 & -2 & -5 & 17 \\ -4 & 0 & -1 & -3 & 2 & 5 & -17 \end{pmatrix}$$

Во второй строке выберем ведущим элементом x_3 :

$$\begin{pmatrix} -\frac{1}{4} & 1 & -2 & -2\frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -21\frac{3}{4} \\ 8\frac{1}{4} & 0 & \| -1 \| & 6\frac{1}{4} & 4\frac{1}{4} & 5\frac{3}{4} & 32\frac{3}{4} \\ 4 & 0 & 1 & 3 & -2 & -5 & 17 \\ -4 & 0 & -1 & -3 & 2 & 5 & -17 \end{pmatrix}$$

Разделим ведущую строку на -1. Вычтем 2 строку из 1, домножив её на 2 и сложив; из 3, домножив на -1 сложив; из 4, сложив:

$$\begin{pmatrix} -16\frac{3}{4} & 1 & 0 & -14\frac{3}{4} & -7\frac{3}{4} & -11\frac{1}{4} & -87\frac{1}{4} \\ -8\frac{1}{4} & 0 & \|1\| & -6\frac{1}{4} & -4\frac{1}{4} & -5\frac{3}{4} & -32\frac{3}{4} \\ 12\frac{1}{4} & 0 & 0 & 9\frac{1}{4} & 2\frac{1}{4} & \frac{3}{4} & 49\frac{3}{4} \\ -12\frac{1}{4} & 0 & 0 & -9\frac{1}{4} & -2\frac{1}{4} & -\frac{3}{4} & -49\frac{3}{4} \end{pmatrix}$$

В третьей строке выберем ведущим элементом x_5 :

$$\begin{pmatrix} -16\frac{3}{4} & 1 & 0 & -14\frac{3}{4} & -7\frac{3}{4} & -11\frac{1}{4} & -87\frac{1}{4} \\ -8\frac{1}{4} & 0 & 1 & -6\frac{1}{4} & -4\frac{1}{4} & -5\frac{3}{4} & -32\frac{3}{4} \\ 12\frac{1}{4} & 0 & 0 & 9\frac{1}{4} & \|2\frac{1}{4}\| & \frac{3}{4} & 49\frac{3}{4} \\ -12\frac{1}{4} & 0 & 0 & -9\frac{1}{4} & -2\frac{1}{4} & -\frac{3}{4} & -49\frac{3}{4} \end{pmatrix}$$

Разделим ведущую строку на $2\frac{1}{4}$. Вычтем 3 строку из 1, домножив её на $7\frac{3}{4}$ и сложив; из 2, домножив на $4\frac{1}{4}$ и сложив; из 4, домножив на $2\frac{1}{4}$ и сложив:

$$\begin{pmatrix} 25\frac{4}{9} & 1 & 0 & 17\frac{1}{9} & 0 & -8\frac{2}{3} & 84\frac{1}{9} \\ 14\frac{8}{9} & 0 & 1 & 11\frac{2}{9} & 0 & -4\frac{1}{3} & 61\frac{2}{9} \\ 5\frac{4}{9} & 0 & 0 & 4\frac{1}{9} & \|1\| & \frac{1}{3} & 22\frac{1}{9} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Так как в 4 строке только нули и b_4 тоже равен 0, строчку можем убрать:

$$\begin{pmatrix} 25\frac{4}{9} & 1 & 0 & 17\frac{1}{9} & 0 & -8\frac{2}{3} & 84\frac{1}{9} \\ 14\frac{8}{9} & 0 & 1 & 11\frac{2}{9} & 0 & -4\frac{1}{3} & 61\frac{2}{9} \\ 5\frac{4}{9} & 0 & 0 & 4\frac{1}{9} & \|1\| & \frac{1}{3} & 22\frac{1}{9} \end{pmatrix}$$

Получили систему уравнений со свободными переменными x_1, x_4, x_6 и базисными x_2, x_3 и x_5 :

$$\begin{cases} x_2 = 84\frac{1}{9} - (25\frac{4}{9}x_1 + 17\frac{1}{9}x_4 - 8\frac{2}{3}x_6) \\ x_3 = 61\frac{2}{9} - (-14\frac{8}{9}x_1 + 11\frac{2}{9}x_4 - 4\frac{1}{3}x_6) \\ x_5 = 22\frac{1}{9} - (5\frac{4}{9}x_1 + 4\frac{1}{9}x_4 + \frac{1}{3}x_6) \end{cases}$$

Вывод: в ходе лабораторной работы разработали и отладили программу, находящую базисные решения системы уравнений методом Гаусса-Жордана.