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 №3**

по дисциплине: Исследование операций

тема: «Модификации симплекс метода. Методы искусственного базиса

и больших штрафов»

Выполнил: ст. группы ПВ-233

Ситников Алексей Павлович

Проверил:

Вирченко Юрий Петрович

Белгород 2025 г.

**Цель работы:** изучение методов искусственного базиса и больших штрафов решения задач ЛП в канонической форме, не подготовленных

к работе симплекс-методом в чистом виде.

Вариант 13

**Задания для подготовки к работе**

1. Изучить метод и алгоритм искусственного базиса и составить программу решения задачи ЛП этим методом.

2. Изучить метод и алгоритм больших штрафов и составить программу решения задачи ЛП этим методом.

3. Запрограммировать изученные алгоритмы и отладить соответствующие программы. В рамках подготовки тестовых данных решить следующую задачу:

Изображение выглядит как текст, рукописный текст, Шрифт, каллиграфия

Контент, сгенерированный ИИ, может содержать ошибки.

Аналитическое решение методом искусственного базиса:  
Сформулируем вспомогательную задачу и добавим в первое уравнение y1, а во второе y2.

Cоставим первую симплекс-таблицу:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 | y1 | y2 |
| y1 | 10 | 1 | 5 | -3 | -4 | 2 | 1 | 1 | 0 |
| y2 | 32 | 3 | 9 | -5 | -7 | 4 | 2 | 0 | 1 |
| f | -42 | -4 | -14 | 8 | 11 | -6 | -3 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 | y1 | y2 |
| x2 | 2 |  | 1 |  |  |  |  |  | 0 |
| y2 | 14 |  | 0 |  |  |  |  |  | 1 |
| f | -14 |  | 0 |  |  |  |  |  | 0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 | y1 | y2 |
| x1 | 10 | 1 | 5 | -3 | -4 | 2 | 1 | 1 | 0 |
| y2 | 2 | 0 | -6 | 4 | 5 | -2 | -1 | -3 | 1 |
| f | -2 | 0 | 6 | -4 | -5 | 2 | 1 | 4 | 0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 | y1 | y2 |
| x1 |  |  |  |  |  |  |  |  |  |
| x4 |  |  |  |  |  |  |  |  |  |
| f | 0 |  | 0 |  |  |  |  |  | 1 |

Завершили подготовку, переходим к целевой функции.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 |
| x1 |  |  |  |  |  |  |  |
| x4 |  |  |  |  |  |  |  |
| z |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 |
| x6 | 58 | 5 | 1 | 1 | 0 | 2 | 1 |
| x4 |  | 1 | -1 | 1 | 1 | 0 | 0 |
| z |  | 43 | 6 | 9 | 0 | 17 | 0 |

z max = . Координаты точки максимума: x1 = 0, x2 = 0, x3 = 0, x4 = 12,

x5 = 0, x6 = 58.

Аналитическое решение методом больших штрафов.

Вводим искусственные переменные:

Выражаем базисные переменные:

Пусть М = 20.

zM = – M (y1 + y2)

zM =

Для этой задачи можно составить симплекс-таблицы:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 | y1 | y2 |
| y1 | 10 | 1 | 5 | -3 | -4 | 2 | 1 | 1 | 0 |
| y2 | 32 | 3 | 9 | -5 | -7 | 4 | 2 | 0 | 1 |
| zM | -820 | -82 | -277 | 156 | 215 | -119 | -68 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 | y1 | y2 |
| x2 | 2 |  | 1 |  |  |  |  |  | 0 |
| y2 | 14 |  | 0 |  |  |  |  |  | 1 |
| zM | -286 |  | 0 |  |  |  |  |  | 0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 | y1 | y2 |
| x1 | 10 | 1 | 5 | -3 | -4 | 2 | 1 | 1 | 0 |
| y2 | 2 |  | -6 |  |  |  |  | -3 | 1 |
| zM | -20 |  | 133 |  |  |  |  |  | 0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 | y1 | y2 |
| x1 |  |  |  |  |  |  |  |  |  |
| x4 |  |  |  |  |  |  |  |  |  |
| zM |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Баз. переменная | Св. член | x1 | x2 | x3 | x4 | x5 | x6 | y1 | y2 |
| x6 |  |  |  |  |  |  |  |  |  |
| x4 |  |  |  |  |  |  |  |  |  |
| zM |  |  |  |  |  |  |  |  |  |

zM = . Координаты точки максимума: x1 = 0, x2 = 0, x3 = 0, x4 = 12,

x5 = 0, x6 = 58.

Решения методов совпали.

Метод искусственного базиса.

Блок-схема модуля main:

Изображение выглядит как текст, снимок экрана, Шрифт, чек

Контент, сгенерированный ИИ, может содержать ошибки.

Код программы:

#include <iostream>  
#include <vector>  
#include <cmath>  
#include <windows.h>  
#include <climits>  
  
#define **min\_value** 2.2e-10  
  
//вывод решения  
void OutputSolve(std::vector<std::vector<double>> &matrix, std::vector<int> &baz){  
 std::cout **<<** "Максимум функции = " **<<** matrix**[**matrix.size()-1**][**0**] <<** std::endl;  
 std::cout **<<** "Координаты точки максимума:" **<<** std::endl **<<** '<';  
 for(int i = 1; i < matrix**[**0**]**.size() ; i++){  
 int f = -1;  
 for(int j = 0; j < matrix.size()-1; j++){  
 if(baz**[**j**]** == i){  
 f = j;  
 }  
 }  
 if(f!=-1){  
 std::cout **<<** "x" **<<** i **<<** " = " **<<** matrix**[**f**][**0**]**;  
 }  
 else{  
 std::cout **<<** "x" **<<** i **<<** " = " **<<** 0;  
 }  
 if(i != matrix**[**0**]**.size()-1){  
 std::cout **<<** "; ";  
 }  
 }  
 std::cout **<<** ">\n";  
}  
  
//функция для ввода данных  
std::vector<std::vector<double>> InputData(int n, int r, std::vector<int> &baz){  
 std::vector<std::vector<double>> matrix(r, std::vector<double>(n));  
 for (int i = 0; i < r - 1; i++) {//пробегаемся по строкам  
 std::cout **<<** "Введите коэффициенты для уравнения " **<<** i + 1 **<<** std::endl;  
 for (int j = 1; j < n; j++) {//пробегаемся по столбцам  
 std::cin **>>** matrix**[**i**][**j**]**;  
 }  
 std::cout **<<** "Введите свободный член для уравнения " **<<** i + 1 **<<** std::endl;  
 std::cin **>>** matrix**[**i**][**0**]**;  
 }  
 std::cout **<<** "Введите коэффициенты для целевой функции начиная с свободного члена\n";  
 for (int i = 0; i < n; i++) {  
 std::cin **>>** matrix**[**r - 1**][**i**]**;  
 matrix**[**r - 1**][**i**]** \*= -1;  
 }  
 //проверяем чтобы b >= 0  
 for(int i = 0; i < r-1; i++){  
 if(matrix**[**i**][**0**]** < 0){  
 for(int j = 0; j < n; j++){  
 matrix**[**i**][**j**]** \*= -1.;  
 }  
 }  
 }  
 //находим базисные переменные  
 for(int i = 1; i < n; i++){  
 double sum = 0;  
 int flag = -1;  
 int indx;  
 for(int j = 0; j < r-1; j++){  
 if(matrix**[**j**][**i**]** == 1.) {  
 flag = j;  
 indx = i;  
 sum++;  
 }  
 else if(fabs(matrix**[**j**][**i**]**) > **min\_value**){  
 sum += matrix**[**j**][**i**]**;  
 if(flag!=-1){  
 break;  
 }  
 }  
 }  
 if(flag != -1 && sum == 1){  
 baz**[**flag**]** = indx;  
 }  
 }  
 return matrix;  
}  
  
  
//Находим минимальный коэффициент в строке целевой функции  
int FindMinIndex(std::vector<double> &v){  
 double min = (double)**INT\_MAX** - 1;//переменная, которая будет хранить минимальное число  
 int index = 0;//переменная, которая будет хранить индекс минимального числа  
 for(int i = 1; i < v.size(); i++){//пробегаемся по массиву  
 if(min > v**[**i**]** && fabs(v**[**i**]**) > **min\_value**){  
 min = v**[**i**]**;  
 index = i;  
 }  
 }  
 return index;//передаём индекс  
}  
  
int FindMinStr(int index\_col, std::vector<std::vector<double>> &matrix, bool \*flag){  
 int index\_string;  
 double min\_val = (double)**INT\_MAX** - 1;  
 for(int i = 0; i < matrix.size()-1; i++){  
 if(matrix**[**i**][**index\_col**]** > 0){//число должно быть > 0  
 \*flag = true;//хотя бы одно число найдено, вероятно решение есть  
 if(matrix**[**i**][**0**]**/matrix**[**i**][**index\_col**]** < min\_val){//находим минимальное значение  
 min\_val = matrix**[**i**][**0**]**/matrix**[**i**][**index\_col**]**;  
 index\_string = i;  
 }  
 }  
 }  
 return index\_string;//индекс найденной строки  
}  
  
  
void simpleks(std::vector<std::vector<double>> &matrix, std::vector<int> &baz){  
  
 int index\_col = FindMinIndex(matrix**[**matrix.size() - 1**]**);//нахождение минимального элемента в последней строке  
  
 while (matrix**[**matrix.size() - 1**][**index\_col**]** < 0){//пока в последней строке минимальное число меньше 0  
 bool flag = false;// для проверки есть ли в столбце положительные числа  
 int index\_string = FindMinStr(index\_col, matrix, &flag);//находим подходящую строку, (index\_string, index\_col) - разрешающий элемент  
 if(!flag){//если в столбце все числа отрицательные  
 std::cout **<<** "целевая функция неограниченна на области допустимых значений переменных, решения нет" **<<** std::endl;  
 return;  
 }  
  
 double K = 1/matrix**[**index\_string**][**index\_col**]**;//коэффициент для получения единицы в разрешающем элементе  
 for(int i = 0; i < matrix**[**0**]**.size(); i++){  
 matrix**[**index\_string**][**i**]** \*= K;//умножаем всю строку разрешающего элемента на этот коэффициент  
 }  
 for(int i = 0; i < matrix.size(); i++){//проходимся п строкам  
 if(i != index\_string){//если строка не та в которой разрешающий элемент  
 std::vector<double> temp(matrix**[**0**]**.size());//временная строка  
 for(int j = 0; j < matrix**[**0**]**.size(); j++){  
 temp**[**j**]** = matrix**[**index\_string**][**j**]** \* matrix**[**i**][**index\_col**]**;//заполняем её разрешающей умноженной на коэффициент который нужно обнулить  
 }  
 for(int j = 0; j < matrix**[**0**]**.size(); j++){  
 matrix**[**i**][**j**]** -= temp**[**j**]**;//вычитаем строку  
 }  
 }  
 }  
  
 baz**[**index\_string**]** = index\_col;//меняем базисные элементы  
 index\_col = FindMinIndex(matrix**[**matrix.size() - 1**]**);//снова находим минимальный элемент в последней строке  
 }  
}  
  
  
void solveF(std::vector<std::vector<double>> &matrix, std::vector<std::vector<double>> &F, int r, int n, std::vector<int> &baz){  
 int index\_col = FindMinIndex(F**[**r-1**]**);//нахождение минимального элемента в последней строке  
 while (F**[**r-1**][**index\_col**]** < 0){//пока в последней строке минимальное число меньше 0  
 bool flag = false;// для проверки есть ли в столбце положительные числа  
 int index\_string = FindMinStr(index\_col, F, &flag);//находим подходящую строку, (index\_string, index\_col) - разрешающий элемент  
 if(!flag){//если в столбце все числа отрицательные  
 std::cout **<<** "целевая функция неограниченна на области допустимых значений переменных, решения нет" **<<** std::endl;  
 exit(1);  
 }  
 double K = 1/F**[**index\_string**][**index\_col**]**;//коэффициент для получения единицы в разрешающем элементе  
 for(auto &i : F**[**index\_string**]**){  
 i \*= K;//умножаем всю строку разрешающего элемента на этот коэффициент  
 }  
 std::vector<double> t(n);  
 for(int i = 0; i < r; i++){//проходимся по строкам  
 if(i != index\_string){//если строка не та в которой разрешающий элемент  
 for(int j = 0; j < F**[**index\_string**]**.size(); j++){  
 t**[**j**]** = F**[**index\_string**][**j**]** \* F**[**i**][**index\_col**]**;//заполняем её разрешающей умноженной на коэффициент который нужно обнулить  
 }  
 for(int j = 0; j < F**[**i**]**.size(); j++){  
 F**[**i**][**j**]** -= t**[**j**]**;//вычитаем строку  
 }  
 }  
 }  
  
 baz**[**index\_string**]** = index\_col;//меняем базисные элементы  
 index\_col = FindMinIndex(F**[**r-1**]**);//снова находим минимальный элемент в последней строке  
 }  
 //подготавливаем симплекс-таблицу для целевой функции z  
 for(int i = 0; i < r-1; i++){  
 for(int j = 0; j < n; j++) {  
 matrix**[**i**][**j**]** = F**[**i**][**j**]**;  
 }  
 }  
  
 for(int i = 0; i < baz.size(); i++){  
 matrix**[**r-1**][**0**]** += matrix**[**i**][**0**]** \* -1 \* matrix**[**r-1**][**baz**[**i**]]**;  
 }  
 //избавляемся от базисных переменных в целевой функции  
 std::vector<double> t(n);  
 matrix**[**matrix.size()-1**][**0**]** = 0;  
 for(int i = 0; i < r-1; i++){  
 for(int j = 0; j < n; j++) {  
 t**[**j**]** = matrix**[**i**][**j**]** \* -1 \* matrix**[**r-1**][**baz**[**i**]]**;  
 }  
 for(int j = 0; j < n; j++){  
 matrix**[**r-1**][**j**]** += t**[**j**]**;  
 }  
 }  
}  
  
void creatTableF(std::vector<std::vector<double>> &F, std::vector<std::vector<double>> &matrix, std::vector<int> &baz, int r, int n){  
 int countY = 0;  
 for(int i = 0; i < baz.size(); i++){  
 if(baz**[**i**]** == 0){//добавляем y при необходимости  
 baz**[**i**]** = ((countY++)+1)\*-1;  
 F**[**i**]**.push\_back(1);  
 F**[**r-1**]**.push\_back(0);  
 for(int j = 0; j < r-1; j++){  
 if(j!= i){  
 F**[**j**]**.push\_back(0);  
 }  
 }  
 }  
 }  
 for(int i = 0; i < n; i++){  
 double sum = 0;  
 for(int j = 0; j < r-1; j++){  
 if(baz**[**j**]** < 0){  
 sum -= matrix**[**j**][**i**]**;  
 }  
 }  
 F**[**r-1**][**i**]** = sum;  
 }  
}  
  
int main(){  
  
 SetConsoleOutputCP(**CP\_UTF8**);  
 int n, r;  
 std::cout **<<** "Введите через пробел количество неизвестных и количество уравнений\n";  
 std::cin **>>** n **>>** r;  
 n++;  
 r++;  
 std::vector<std::vector<double>> matrix;//создаём матрицу для таблицы  
 std::vector<int> baz(r-1);//создаём вектор базисных переменных  
 matrix **=** InputData(n, r, baz);//заполняем таблицу  
 std::vector<std::vector<double>> F = matrix;//создаём таблицу для решения задачи F  
 creatTableF(F, matrix, baz, r, n);//заполняем таблицу  
 solveF(matrix, F, r, n, baz);//решаем F задачу  
 simpleks(matrix, baz);//решаем основную задачу  
 OutputSolve(matrix, baz);//выводим решение  
 return 0;  
}

Вывод программы:  
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Метод больших штрафов.

Блок-схема:  
Изображение выглядит как текст, диаграмма, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Код программы:

#include <iostream>  
#include <vector>  
#include <windows.h>  
#include <cmath>  
  
#define **e** 1e-15  
  
//вывод решения  
void OutputSolve(std::vector<std::vector<double>> &matrix, std::vector<int> &baz, int y\_count){  
 std::cout **<<** "Максимум функции = " **<<** matrix**[**matrix.size()-1**][**0**] <<** std::endl;  
 std::cout **<<** "Координаты точки максимума:" **<<** std::endl **<<** '<';  
 for(int i = 1; i < matrix**[**0**]**.size() - y\_count; i++){  
 int f = -1;  
 for(int j = 0; j < matrix.size()-1; j++){  
 if(baz**[**j**]** == i){  
 f = j;  
 }  
 }  
 if(f!=-1){  
 std::cout **<<** "x" **<<** i **<<** " = " **<<** matrix**[**f**][**0**]**;  
 }  
 else{  
 std::cout **<<** "x" **<<** i **<<** " = " **<<** 0;  
 }  
 if(i != matrix**[**0**]**.size()-1 - y\_count){  
 std::cout **<<** "; ";  
 }  
 }  
 std::cout **<<** ">\n";  
}  
std::vector<std::vector<double>> input(int r, int n, std::vector<int> &baz, int \*Y\_count){  
 std::vector<std::vector<double>> matrix(r+1, std::vector<double>(n+1));;  
 int sizeBefore = (int)matrix**[**0**]**.size();  
 for(int i = 0; i < matrix.size() - 1; i++){  
 std::cout **<<** "Введите коэффициенты для уравнения " **<<** i+1 **<<** std::endl;  
 for(int j = 1; j < matrix**[**0**]**.size(); j++){  
 std::cin **>>** matrix**[**i**][**j**]**;  
 }  
 std::cout **<<** "Введите свободный член для уравнения " **<<** i+1 **<<** std::endl;  
 std::cin **>>** matrix**[**i**][**0**]**;  
 }  
 std::cout **<<** "введите коэффициенты целевой функции начиная с свободного члена\n";  
 for(int i = 0; i < matrix**[**0**]**.size(); i++){  
 std::cin **>>** matrix**[**matrix.size()-1**][**i**]**;  
 }  
  
 for(int i = 0; i < r-1; i++){  
 if(matrix**[**i**][**0**]** < 0){  
 for(int j = 0; j < n; j++){  
 matrix**[**i**][**j**]** \*= -1.;  
 }  
 }  
 }  
  
  
 //ищем базисные переменные  
 for(int i = 1; i < matrix**[**0**]**.size(); i++) {  
 int index = 0;  
 int flag = 0;  
 for (int j = 0; j < matrix.size() - 1; j++) {  
 if (fabs(matrix**[**j**][**i**]**) > **e** && matrix**[**j**][**i**]** != 1) {  
 flag = 0;  
 break;  
 } else {  
 flag += matrix**[**j**][**i**]** == 1 ? 1 : 0;  
 index = matrix**[**j**][**i**]** == 1 ? j : index;  
 }  
 }  
 if (flag == 1) {  
 baz**[**index**]** = i;  
 }  
 }  
  
 for(int i = 0; i < baz.size(); i++){  
 if(baz**[**i**]** == 0){//если базисная переменная y  
 baz**[**i**]** = -(i+1);  
 matrix**[**i**]**.push\_back(1.);  
 for(int j = 0; j < matrix.size(); j++){  
 if(j!=i){  
 matrix**[**j**]**.push\_back(0.);  
 }  
 }  
  
 }  
 }  
 int M;  
 std::cout **<<** "Введите М\n";  
 std::cin **>>** M;//вводим М  
 std::vector<double> temp(matrix**[**0**]**.size());//создаём временную строку где выражаем базисные переменные  
 for(int i = 0; i < baz.size(); i++){  
 for(int j = 0; j < sizeBefore; j++){//заполняем строку  
 temp**[**j**]** = matrix**[**i**][**j**]**;  
 }  
 int indx = baz**[**i**]**;  
 if(baz**[**i**]** < 0){  
 indx = (baz**[**i**]** \* -1) + sizeBefore-1;  
 }  
 temp**[**indx**]** = 0.;  
 temp**[**0**]**\*=-1.;  
 if(indx > sizeBefore-1){//если базисная переменная это y  
  
 for(int j = 0; j < sizeBefore; j++){  
 temp**[**j**]** \*= M;  
 }  
 }  
 else{//если базисная переменная не y  
 for(int j = 0; j < sizeBefore; j++){  
 temp**[**j**]** \*= matrix**[**matrix.size()-1**][**baz**[**i**]]** \* -1;  
 }  
 }  
 for(int j = 0; j < sizeBefore; j++){//изменяем елевую функцию  
 matrix**[**matrix.size()-1**][**j**]** += temp**[**j**]**;  
 }  
 }  
 for(int j = 1; j < sizeBefore; j++){//приводим значения целевой функции к нужным для симплекс-таблицы  
 matrix**[**matrix.size()-1**][**j**]** \*= -1;  
 }  
 for(int i : baz){  
 if(i > 0){  
 matrix**[**matrix.size()-1**][**i**]** = 0;//обнуляем базисные переменные в целевой функции  
 }  
 }  
 \*Y\_count = (int)matrix**[**0**]**.size() - sizeBefore;//запоминаем количество y  
 return matrix;  
}  
  
//Находим минимальный коэффициент в строке целевой функции  
int FindMinIndex(std::vector<double> &v, int f){  
 double min = (double)**INT\_MAX** - 1;//переменная, которая будет хранить минимальное число  
 int index = 0;//переменная, которая будет хранить индекс минимального числа  
 for(int i = 1; i < f; i++){//пробегаемся по массиву  
 if(min > v**[**i**]** && fabs(v**[**i**]**) > **e**){  
 min = v**[**i**]**;  
 index = i;  
 }  
 }  
 return index;//передаём индекс  
}  
  
int FindMinStr(int index\_col, std::vector<std::vector<double>> &matrix, bool \*flag){  
 int index\_string;  
 double min\_val = (double)**INT\_MAX** - 1;  
 for(int i = 0; i < matrix.size()-1; i++){  
 if(matrix**[**i**][**index\_col**]** > 0){//число должно быть > 0  
 \*flag = true;//хотя бы одно число найдено, вероятно решение есть  
 if(matrix**[**i**][**0**]**/matrix**[**i**][**index\_col**]** < min\_val){//находим минимальное значение  
 min\_val = matrix**[**i**][**0**]**/matrix**[**i**][**index\_col**]**;  
 index\_string = i;  
 }  
 }  
 }  
 return index\_string;//индекс найденной строки  
}  
  
  
void simpleks(std::vector<std::vector<double>> &matrix, std::vector<int> &baz, int y\_count){  
  
 int index\_col = FindMinIndex(matrix**[**matrix.size() - 1**]**, (int)matrix**[**0**]**.size() - y\_count);//нахождение минимального элемента в последней строке  
  
 while (matrix**[**matrix.size() - 1**][**index\_col**]** < 0){//пока в последней строке минимальное число меньше 0  
 bool flag = false;// для проверки есть ли в столбце положительные числа  
 int index\_string = FindMinStr(index\_col, matrix, &flag);//находим подходящую строку, (index\_string, index\_col) - разрешающий элемент  
 if(!flag){//если в столбце все числа отрицательные  
 std::cout **<<** "целевая функция неограниченна на области допустимых значений переменных, решения нет" **<<** std::endl;  
 return;  
 }  
  
 double K = 1/matrix**[**index\_string**][**index\_col**]**;//коэффициент для получения единицы в разрешающем элементе  
 for(int i = 0; i < matrix**[**0**]**.size(); i++){  
 matrix**[**index\_string**][**i**]** \*= K;//умножаем всю строку разрешающего элемента на этот коэффициент  
 }  
 for(int i = 0; i < matrix.size(); i++){//проходимся п строкам  
 if(i != index\_string){//если строка не та в которой разрешающий элемент  
 std::vector<double> temp(matrix**[**0**]**.size());//временная строка  
 for(int j = 0; j < matrix**[**0**]**.size(); j++){  
 temp**[**j**]** = matrix**[**index\_string**][**j**]** \* matrix**[**i**][**index\_col**]**;//заполняем её разрешающей умноженной на коэффициент который нужно обнулить  
 }  
 for(int j = 0; j < matrix**[**0**]**.size(); j++){  
 matrix**[**i**][**j**]** -= temp**[**j**]**;//вычитаем строку  
 }  
 }  
 }  
  
 baz**[**index\_string**]** = index\_col;//меняем базисные элементы  
 index\_col = FindMinIndex(matrix**[**matrix.size() - 1**]**, matrix**[**0**]**.size() - y\_count);//снова находим минимальный элемент в последней строке  
 }  
}  
  
  
  
  
int main() {  
 SetConsoleOutputCP(**CP\_UTF8**);  
 int r, n;  
 std::cout **<<** "Введите количество неизвестных и количество уравнений\n";  
 std::cin **>>** n **>>** r;  
  
 std::vector<int> baz(r);  
 int y\_count;  
 std::vector<std::vector<double>> matrix = input(r, n, baz, &y\_count);  
 simpleks(matrix, baz, y\_count);

OutputSolve(matrix, baz, y\_count);//выводим решение  
 return 0;  
}

Вывод программы:  
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Вывод:** результатом выполненной работы являются две программы для решения задач линейного программирования, в которых реализован метод искусственного базиса и метод больших штрафов.