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 №7

по дисциплине: Исследование операций

тема: Решение полностью целочисленных задач с помощью первого
алгоритма Гомори, а также методом ветвей и границ

Выполнил: ст. группы ПВ-223
Игнатьев Артур Олегович
Проверил:
Вирченко Юрий Петрович

Белгород 2024 г.

Цель работы: Освоить методы отсечения Гомори для полностью целочисленных задач. Изучить алгоритм этого метода. Программно реализовать этот алгоритм.

Задания

1. Изучить возможные постановки задач целочисленного и частично-целочисленного программирования.

2. Ознакомиться с методами решения таких задач, в частности, с методами отсечения и методом ветвей и границ.

3. Выяснить для каких задач применяется первый алгоритм Гомори. Изучить этот алгоритм и написать реализующую его программу для ПЭВМ. Изучить и программно реализовать алгоритм метода ветвей и границ. В качестве тестовых данных использовать, решенную вручную следующую задачу.

Вариант 3

$$z = 9x_1 - 4x_2 + 3x_5 \to \max;$$

$$\begin{cases} 10x_1 + 3x_2 + x_3 = 93, \\ 14x_1 - 5x_2 - x_4 = 26, \\ 2x_1 - 9x_2 - x_5 = 18, \end{cases}$$

$$x_i \geq 0,\ x_i - \textit{целые}\ \left(i = \overline{1,5}\right)$$

Ручной расчет

Решим прямую задачу линейного программирования симплексным методом, с использованием симплексной таблицы.

Определим максимальное значение целевой функции $F(X) = 9x_1 - 4x_2 + 3x_5$ при следующих условиях-ограничений.

$$10x_1 + 3x_2 + x_3 = 93$$

$$14x_1 - 5x_2 - x_4 = 26$$

$$2x_1 - 9x_2 - x_5 = 18$$

Расширенная матрица системы ограничений-равенств данной задачи:

10	3	1	0	0	93
14	-5	0	-1	0	26
2	-9	0	0	-1	18

Приведем систему к единичной матрице методом жордановских преобразований.

1. В качестве базовой переменной можно выбрать x_3.

2. В качестве базовой переменной можно выбрать x_4.

Получаем новую матрицу:

10	3	1	0	0	93
-14	5	0	1	0	-26
2	-9	0	0	-1	18

3. В качестве базовой переменной можно выбрать x_5.

Получаем новую матрицу:

10	3	1	0	0	93
-14	5	0	1	0	-26
-2	9	0	0	1	-18

Поскольку в системе имеется единичная матрица, то в качестве базисных переменных принимаем X = (3,4,5).

Выразим базисные переменные через остальные:

$$x_3 = -10x_1 - 3x_2 + 93$$

$$x_4 = 14x_1 - 5x_2 - 26$$

$$x_5 = 2x_1 - 9x_2 - 18$$

Подставим их в целевую функцию:

$$F(X) = 9x_1 - 4x_2 + 3(2x_1 - 9x_2 - 18)$$

или

$$F(X) = 15x_1 - 31x_2 - 54$$

Среди свободных членов b_i имеются отрицательные значения, следовательно, полученный базисный план не является опорным.

Вместо переменной x_4 следует ввести переменную x_1.

Выполняем преобразования симплексной таблицы методом Жордано-Гаусса.

Базис	B	x_1	x_2	x_3	x_4	x_5
x_3	$\frac{521}{7}$	0	$\frac{46}{7}$	1	$\frac{5}{7}$	0
x_1	$\frac{13}{7}$	1	$-\frac{5}{14}$	0	$-\frac{1}{14}$	0
x_5	$-\frac{100}{7}$	0	$\frac{58}{7}$	0	$-\frac{1}{7}$	1
$F(X_0)$	$-\frac{573}{7}$	0	$-\frac{359}{14}$	0	$\frac{15}{14}$	0

Представим расчет каждого элемента в виде таблицы:

B	x_1	x_2	x_3	x_4	x_5
93-(-26*10):-14	10-(-14*10):-14	3-(5*10):-14	1-(0*10):-14	0-(1*10):-14	0-(0*10):-14
-26 : -14	-14 : -14	5 : -14	0 : -14	1 : -14	0 : -14
-18-(-26*-2):-14	-2-(-14*-2):-14	9-(5*-2):-14	0-(0*-2):-14	0-(1*-2):-14	1-(0*-2):-14

Среди свободных членов bi имеются отрицательные значения, следовательно, полученный базисный план не является опорным.

Вместо переменной x5 следует ввести переменную x4.

Выполняем преобразования симплексной таблицы методом Жордано-Гаусса.

Базис	B	x_1	x_2	x_3	x_4	x_5
x_3	3	0	48	1	0	5
x_1	9	1	$-\frac{9}{2}$	0	0	$-\frac{1}{2}$
x_4	100	0	-58	0	1	-7
$F(X_1)$	-189	0	$\frac{73}{2}$	0	0	$\frac{15}{2}$

Представим расчет каждого элемента в виде таблицы:

B	x_1	x_2	x_3	x_4	x_5
$\frac{521}{7}-\left(-\frac{100}{7}*\frac{5}{7}\right):-\frac{1}{7}$	$0-\left(0*\frac{5}{7}\right):-\frac{1}{7}$	$\frac{46}{7}-\left(\frac{58}{7}*\frac{5}{7}\right):-\frac{1}{7}$	$1-\left(0*\frac{5}{7}\right):-\frac{1}{7}$	$\frac{5}{7}-\left(-\frac{1}{7}*\frac{5}{7}\right):-\frac{1}{7}$	$0-\left(1*\frac{5}{7}\right):-\frac{1}{7}$
$\frac{13}{7}-\left(-\frac{100}{7}*-\frac{1}{14}\right):-\frac{1}{7}$	$1-\left(0*-\frac{1}{14}\right):-\frac{1}{7}$	$-\frac{5}{14}-\left(\frac{58}{7}*-\frac{1}{14}\right):-\frac{1}{7}$	$0-\left(0*-\frac{1}{14}\right):-\frac{1}{7}$	$-\frac{1}{14}-\left(-\frac{1}{7}*-\frac{1}{14}\right):-\frac{1}{7}$	$0-\left(1*-\frac{1}{14}\right):-\frac{1}{7}$
$-\frac{100}{7}:-\frac{1}{7}$	$0:-\frac{1}{7}$	$\frac{58}{7}:-\frac{1}{7}$	$0:-\frac{1}{7}$	$-\frac{1}{7}:-\frac{1}{7}$	$1:-\frac{1}{7}$

Выразим базисные переменные через остальные:

$$x_3 = -48x_2 - 5x_5 + 3$$

$$x_1 = \frac{9}{2x_2} + \frac{1}{2x_5} + 9$$

$$x_4 = 58x_2 + 7x_5 + 100$$

Подставим их в целевую функцию:

$$F(X) = 9\left(\frac{9}{2x_2} + \frac{1}{2x_5} + 9\right) - 4x_2 + 3x_5$$

или

$$F(X) = \frac{73}{2x_2} + \frac{15}{2x_5} + 81$$

$$48x_2 + x_3 + 5x_5 = 3$$

$$x_1 - \frac{9}{2x_2} - \frac{1}{2x_5} = 9$$

$$-58x_2 + x_4 - 7x_5 = 100$$

При вычислениях значение $F_c = 81$ временно не учитываем.

Матрица коэффициентов $A = a(ij)$ этой системы уравнений имеет вид:

A =

0	48	1	0	5
1	-9/2	0	0	-1/2
0	-58	0	1	-7

Базисные переменные это переменные, которые входят только в одно уравнение системы ограничений и притом с единичным коэффициентом.

Экономический смысл дополнительных переменных: дополнительные переменные задачи ЛП обозначают излишки сырья, времени, других ресурсов, остающихся в производстве данного оптимального плана.

Решим систему уравнений относительно базисных переменных: x_3, x_1, x_4

Полагая, что свободные переменные равны 0, получим первый опорный план:

$$X_0 = (9{,}0{,}3{,}100{,}0)$$

Базисное решение называется допустимым, если оно неотрицательно.

Базис	B	x_1	x_2	x_3	x_4	x_5
x_3	3	0	48	1	0	5
x_1	9	1	$-\frac{9}{2}$	0	0	$-\frac{1}{2}$
x_4	100	0	-58	0	1	-7
$F(X_0)$	0	0	$-\frac{73}{2}$	0	0	$-\frac{15}{2}$

Переходим к основному алгоритму симплекс-метода.

Итерация №0.

1. Проверка критерия оптимальности.

Текущий опорный план неоптимален, так как в индексной строке находятся отрицательные коэффициенты.

2. Определение новой базисной переменной.

В качестве ведущего выберем столбец, соответствующий переменной x2, так как это наибольший коэффициент по модулю.

3. Определение новой свободной переменной.

Вычислим значения Di по строкам как частное от деления: $\frac{b_i}{a_{i2}}$ и из них выберем наименьшее: $\min(3 : 48, -, -) = \frac{1}{16}$

Следовательно, 1-ая строка является ведущей.

Разрешающий элемент равен (48) и находится на пересечении ведущего столбца и ведущей строки.

Базис	B	x_1	x_2	x_3	x_4	x_5	min
x_3	3	0	48	1	0	5	$\frac{1}{16}$
x_1	9	1	$-\frac{9}{2}$	0	0	$-\frac{1}{2}$	-
x_4	100	0	-58	0	1	-7	-
$F(X_1)$	0	0	$-\frac{73}{2}$	0	0	$-\frac{15}{2}$	0

4. Пересчет симплекс-таблицы.

Формируем следующую часть симплексной таблицы. Вместо переменной x_3 в план 1 войдет переменная x_2.

Строка, соответствующая переменной x_2 в плане 1, получена в результате деления всех элементов строки x_3 плана 0 на разрешающий элемент РЭ = 48. На месте разрешающего элемента получаем 1. В остальных клетках столбца x_2 записываем нули.

Таким образом, в новом плане 1 заполнены строка x_2 и столбец x_2. Все остальные элементы нового плана 1, включая элементы индексной строки, определяются по правилу прямоугольника.

Для этого выбираем из старого плана четыре числа, которые расположены в вершинах прямоугольника и всегда включают разрешающий элемент РЭ.

$$\text{НЭ} = \text{СЭ} - \frac{\text{А} * \text{В}}{\text{РЭ}}$$

СТЭ - элемент старого плана, РЭ - разрешающий элемент (48), А и В - элементы старого плана, образующие прямоугольник с элементами СТЭ и РЭ.

Представим расчет каждого элемента в виде таблицы:

B	x_1	x_2	x_3	x_4	x_5
$3 : 48$	$0 : 48$	$48 : 48$	$1 : 48$	$0 : 48$	$5 : 48$
$9 - \left(3 * -\frac{9}{2}\right) : 48$	$1 - \left(0 * -\frac{9}{2}\right) : 48$	$-\frac{9}{2} - \left(48 * -\frac{9}{2}\right) : 48$	$0 - \left(1 * -\frac{9}{2}\right) : 48$	$0 - \left(0 * -\frac{9}{2}\right) : 48$	$-\frac{1}{2} - \left(5 * -\frac{9}{2}\right) : 48$
$100 - (3 * -58) : 48$	$0 - \left(0 * -\frac{73}{2}\right) : 48$	$-58 - (48 * -58) : 48$	$0 - (1 * -58) : 48$	$1 - (0 * -58) : 48$	$-7 - (5 * -58) : 48$
$0 - \left(3 * -\frac{73}{2}\right) : 48$	$0 - \left(0 * -\frac{73}{2}\right) : 48$	$-\frac{73}{2} - \left(48 * -\frac{73}{2}\right) : 48$	$0 - \left(1 * -\frac{73}{2}\right) : 48$	$0 - \left(0 * -\frac{73}{2}\right) : 48$	$-\frac{15}{2} - \left(5 * -\frac{73}{2}\right) : 48$

Получаем новую симплекс-таблицу:

Базис	B	x_1	x_2	x_3	x_4	x_5
x_2	$\frac{1}{16}$	0	1	$\frac{1}{48}$	0	$\frac{5}{48}$
x_1	$\frac{297}{32}$	1	0	$\frac{3}{32}$	0	$-\frac{1}{32}$
x_4	$\frac{829}{8}$	0	0	$\frac{29}{24}$	1	$-\frac{23}{24}$
$F(X_1)$	$\frac{73}{32}$	0	0	$\frac{73}{96}$	0	$-\frac{355}{96}$

Итерация №1.

1. Проверка критерия оптимальности.

Текущий опорный план неоптимален, так как в индексной строке находятся отрицательные коэффициенты.

2. Определение новой базисной переменной.

В качестве ведущего выберем столбец, соответствующий переменной x_5, так как это наибольший коэффициент по модулю.

3. Определение новой свободной переменной.

Вычислим значения D_i по строкам как частное от деления: $\frac{b_i}{a_{i5}}$ и из них выберем наименьшее:

$$\min\left(\frac{1}{16}:\frac{5}{48}, -, -\right) = \frac{3}{5}$$

Следовательно, 1-ая строка является ведущей.

Разрешающий элемент равен $(\frac{5}{48})$ и находится на пересечении ведущего столбца и ведущей строки.

Базис	B	x1	x2	x3	x4	x5	min
x_2	$\frac{1}{16}$	0	1	$\frac{1}{48}$	0	$\frac{5}{48}$	$\frac{3}{5}$
x_1	$\frac{297}{32}$	1	0	$\frac{3}{32}$	0	$-\frac{1}{32}$	-
x_4	$\frac{829}{8}$	0	0	$\frac{29}{24}$	1	$-\frac{23}{24}$	-
$F(X_2)$	$\frac{73}{32}$	0	0	$\frac{73}{96}$	0	$-\frac{355}{96}$	0

4. Пересчет симплекс-таблицы.

Формируем следующую часть симплексной таблицы. Вместо переменной x_2 в план 2 войдет переменная x_5.

Строка, соответствующая переменной x_5 в плане 2, получена в результате деления всех элементов строки x_2 плана 1 на разрешающий элемент РЭ $= \frac{5}{48}$. На месте разрешающего элемента получаем 1. В остальных клетках столбца x_5 записываем нули.

Таким образом, в новом плане 2 заполнены строка x_5 и столбец x_5. Все остальные элементы нового плана 2, включая элементы индексной строки, определяются по правилу прямоугольника.

Представим расчет каждого элемента в виде таблицы:

B	x_1	x_2	x_3	x_4	x_5
$\frac{1}{16}:\frac{5}{48}$	$0:\frac{5}{48}$	$1:\frac{5}{48}$	$\frac{1}{48}:\frac{5}{48}$	$0:\frac{5}{48}$	$\frac{5}{48}:\frac{5}{48}$
$\frac{297}{32}-\left(\frac{1}{16} * -\frac{1}{32}\right):\frac{5}{48}$	$1-\left(0 * -\frac{1}{32}\right):\frac{5}{48}$	$0-\left(1 * -\frac{1}{32}\right):\frac{5}{48}$	$\frac{3}{32}-\left(\frac{1}{48} * -\frac{1}{32}\right):\frac{5}{48}$	$0-\left(0 * -\frac{1}{32}\right):\frac{5}{48}$	$-\frac{1}{32}-\left(\frac{5}{48} * -\frac{1}{32}\right):\frac{5}{48}$
$\frac{829}{8}-\left(\frac{1}{16} * -\frac{23}{24}\right):\frac{5}{48}$	$0-\left(0 * -\frac{23}{24}\right):\frac{5}{48}$	$0-\left(1 * -\frac{23}{24}\right):\frac{5}{48}$	$\frac{29}{24}-\left(\frac{1}{48} * -\frac{23}{24}\right):\frac{5}{48}$	$1-\left(0 * -\frac{23}{24}\right):\frac{5}{48}$	$-\frac{23}{24}-\left(\frac{5}{48} * -\frac{23}{24}\right):\frac{5}{48}$
$\frac{73}{32}-\left(\frac{1}{16} * -\frac{355}{96}\right):\frac{5}{48}$	$0-\left(0 * -\frac{355}{96}\right):\frac{5}{48}$	$0-\left(1 * -\frac{355}{96}\right):\frac{5}{48}$	$\frac{73}{96}-\left(\frac{1}{48} * -\frac{355}{96}\right):\frac{5}{48}$	$0-\left(0 * -\frac{355}{96}\right):\frac{5}{48}$	$-\frac{355}{96}-\left(\frac{5}{48} * -\frac{355}{96}\right):\frac{5}{48}$

Получаем новую симплекс-таблицу:

Базис	B	x_1	x_2	x_3	x_4	x_5
x_5	$\frac{3}{5}$	0	$\frac{48}{5}$	$\frac{1}{5}$	0	1
x_1	$\frac{93}{10}$	1	$\frac{3}{10}$	$\frac{1}{10}$	0	0
x_4	$\frac{521}{5}$	0	$\frac{46}{5}$	$\frac{7}{5}$	1	0
$F(X_2)$	$\frac{9}{2}$	0	$\frac{71}{2}$	$\frac{3}{2}$	0	0

1. Проверка критерия оптимальности.

Среди значений индексной строки нет отрицательных. Поэтому эта таблица определяет оптимальный план задачи.

Окончательный вариант симплекс-таблицы:

Базис	B	x_1	x_2	x_3	x_4	x_5
x_5	$\frac{3}{5}$	0	$\frac{48}{5}$	$\frac{1}{5}$	0	1
x_1	$\frac{93}{10}$	1	$\frac{3}{10}$	$\frac{1}{10}$	0	0
x_4	$\frac{521}{5}$	0	$\frac{46}{5}$	$\frac{7}{5}$	1	0
$F(X_3)$	$\frac{9}{2}$	0	$\frac{71}{2}$	$\frac{3}{2}$	0	0

Оптимальный план можно записать так:

$$x_1 = \frac{93}{10}, x_2 = 0, x_3 = 0, x_4 = \frac{521}{5}, x_5 = \frac{3}{5}$$

$$F(X) = 9 * \frac{93}{10} - 4 * 0 + 3 * \frac{3}{5} = \frac{171}{2}$$

Метод Гомори.

В полученном оптимальном плане присутствуют дробные числа.

По 1-у уравнению с переменной x_5, получившей нецелочисленное значение в оптимальном плане с наибольшей дробной частью $\frac{3}{5}$, составляем дополнительное ограничение:

$$q_1 - q_{11} \bullet x_1 - q_{12} \bullet x_2 - q_{13} \bullet x_3 - q_{14} \bullet x_4 - q_{15} \bullet x_5 \leq 0$$

$$q_1 = b_1 - [b_1] = \frac{3}{5} - 0 = \frac{3}{5}$$

$$q_{11} = a_{11} - [a_{11}] = 0 - 0 = 0$$

$$q_{12} = a_{12} - [a_{12}] = \frac{48}{5} - 9 = \frac{3}{5}$$

$$q_{13} = a_{13} - [a_{13}] = \frac{1}{5} - 0 = \frac{1}{5}$$

$$q_{14} = a_{14} - [a_{14}] = 0 - 0 = 0$$

$$q_{15} = a_{15} - [a_{15}] = 1 - 1 = 0$$

Дополнительное ограничение имеет вид:

$$\frac{3}{5} - \frac{3}{5x_2} - \frac{1}{5x_3} \leq 0$$

Преобразуем полученное неравенство в уравнение:

$$\frac{3}{5} - \frac{3}{5x_2} - \frac{1}{5x_3} + x_6 = 0$$

коэффициенты которого введем дополнительной строкой в оптимальную симплексную таблицу.

Поскольку двойственный симплекс-метод используется для поиска минимума целевой функции, делаем преобразование F(x) = -F(X).

Базис	B	x_1	x_2	x_3	x_4	x_5	x_6
x_5	$\frac{3}{5}$	0	$\frac{48}{5}$	$\frac{1}{5}$	0	1	0
x_1	$\frac{93}{10}$	1	$\frac{3}{10}$	$\frac{1}{10}$	0	0	0
x_4	$\frac{521}{5}$	0	$\frac{46}{5}$	$\frac{7}{5}$	1	0	0
x_6	$-\frac{3}{5}$	0	$-\frac{3}{5}$	$-\frac{1}{5}$	0	0	1
$F(X_0)$	$-\frac{9}{2}$	0	$-\frac{71}{2}$	$-\frac{3}{2}$	0	0	0

1. Проверка критерия оптимальности.

План 0 в симплексной таблице является псевдопланом, поэтому определяем ведущие строку и столбец.

2. Определение новой свободной переменной.

Среди отрицательных значений базисных переменных выбираем наибольший по модулю.

Ведущей будет 4-ая строка, а переменную x_6 следует вывести из базиса.

3. Определение новой базисной переменной.

Минимальное значение θ соответствует 3-му столбцу, т.е. переменную x3 необходимо ввести в базис.

На пересечении ведущих строки и столбца находится разрешающий элемент (РЭ), равный $(-\frac{1}{5})$.

Базис	B	x_1	x_2	x_3	x_4	x_5	x_6
x_5	$\frac{3}{5}$	0	$\frac{48}{5}$	$\frac{1}{5}$	0	1	0
x_1	$\frac{93}{10}$	1	$\frac{3}{10}$	$\frac{1}{10}$	0	0	0
x_4	$\frac{521}{5}$	0	$\frac{46}{5}$	$\frac{7}{5}$	1	0	0
x_6	$-\frac{3}{5}$	0	$-\frac{3}{5}$	$-\frac{1}{5}$	0	0	1
$F(X_0)$	$-\frac{9}{2}$	0	$-\frac{71}{2}$	$-\frac{3}{2}$	0	0	0
0		–	$-\frac{71}{2}:\left(-\frac{3}{5}\right) = \frac{355}{6}$	$-\frac{3}{2}:\left(-\frac{1}{5}\right) = \frac{15}{2}$	–	–	–

4. Пересчет симплекс-таблицы.

Выполняем преобразования симплексной таблицы методом Жордано-Гаусса.

Базис	B	x_1	x_2	x_3	x_4	x_5	x_6
x_5	0	0	9	0	0	1	1
x_1	9	1	0	0	0	0	$\frac{1}{2}$
x_4	100	0	5	0	1	0	7
x_3	3	0	3	1	0	0	-5
$F(X_0)$	0	0	-31	0	0	0	$-\frac{15}{2}$

B	x_1	x_2	x_3	x_4	x_5	x_6
$\frac{3}{5} - \left(-\frac{3}{5} * \frac{1}{5}\right) : -\frac{1}{5}$	$0 - \left(0 * \frac{1}{5}\right) : -\frac{1}{5}$	$\frac{48}{5} - \left(-\frac{3}{5} * \frac{1}{5}\right) : -\frac{1}{5}$	$\frac{1}{5} - \left(-\frac{1}{5} * \frac{1}{5}\right) : -\frac{1}{5}$	$0 - \left(0 * \frac{1}{5}\right) : -\frac{1}{5}$	$1 - \left(0 * \frac{1}{5}\right) : -\frac{1}{5}$	$0 - \left(1 * \frac{1}{5}\right) : -\frac{1}{5}$
$\frac{93}{10} - \left(-\frac{3}{5} * \frac{1}{10}\right) : -\frac{1}{5}$	$1 - \left(0 * \frac{1}{10}\right) : -\frac{1}{5}$	$\frac{3}{10} - \left(-\frac{3}{5} * \frac{1}{10}\right) : -\frac{1}{5}$	$\frac{1}{10} - \left(-\frac{1}{5} * \frac{1}{10}\right) : -\frac{1}{5}$	$0 - \left(* \frac{1}{10}\right) : -\frac{1}{5}$	$0 - \left(0 * \frac{1}{10}\right) : -\frac{1}{5}$	$0 - \left(1 * \frac{1}{10}\right) : -\frac{1}{5}$
$\frac{521}{5} - \left(-\frac{3}{7} * \frac{7}{5}\right) : -\frac{1}{5}$	$0 - \left(0 * \frac{7}{5}\right) : -\frac{1}{5}$	$\frac{46}{5} - \left(-\frac{3}{5} * \frac{7}{5}\right) : -\frac{1}{5}$	$\frac{7}{5} - \left(-\frac{1}{5} * \frac{7}{5}\right) : -\frac{1}{5}$	$1 - \left(0 * \frac{7}{5}\right) : -\frac{1}{5}$	$0 - \left(0 * \frac{7}{5}\right) : -\frac{1}{5}$	$0 - \left(1 * \frac{7}{5}\right) : -\frac{1}{5}$
$-\frac{3}{5} : -\frac{1}{5}$	$0 : -\frac{1}{5}$	$-\frac{3}{5} : -\frac{1}{5}$	$-\frac{1}{5} : -\frac{1}{5}$	$0 : -\frac{1}{5}$	$0 : -\frac{1}{5}$	$1 : -\frac{1}{5}$
$-\frac{9}{2} - \left(-\frac{3}{5} * -\frac{3}{2}\right) : -\frac{1}{5}$	$0 - \left(0 * -\frac{3}{2}\right) : -\frac{1}{5}$	$-\frac{71}{2} - \left(-\frac{3}{5} * -\frac{3}{2}\right) : -\frac{1}{5}$	$-\frac{3}{2} - \left(-\frac{1}{5} * -\frac{3}{2}\right) : -\frac{1}{5}$	$0 - \left(0 * -\frac{3}{2}\right) : -\frac{1}{5}$	$0 - \left(0 * -\frac{3}{2}\right) : -\frac{1}{5}$	$0 - \left(1 * -\frac{3}{2}\right) : -\frac{1}{5}$

Решение получилось целочисленным. Нет необходимости применять метод Гомори.

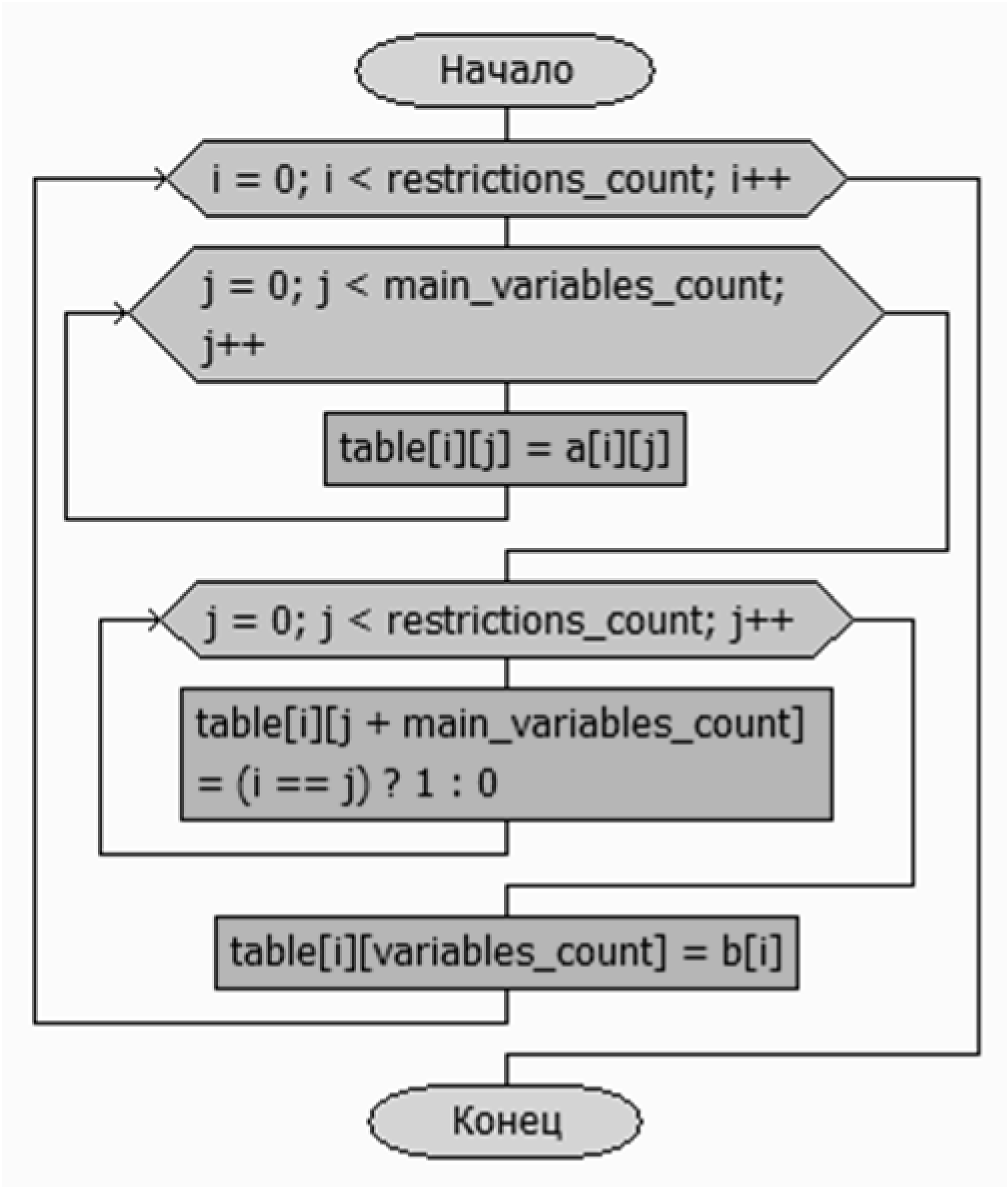
Оптимальный целочисленный план можно записать так:

$$x_1 = 9, x_2 = 0, x_3 = 3, x_4 = 100, x_5 = 0$$

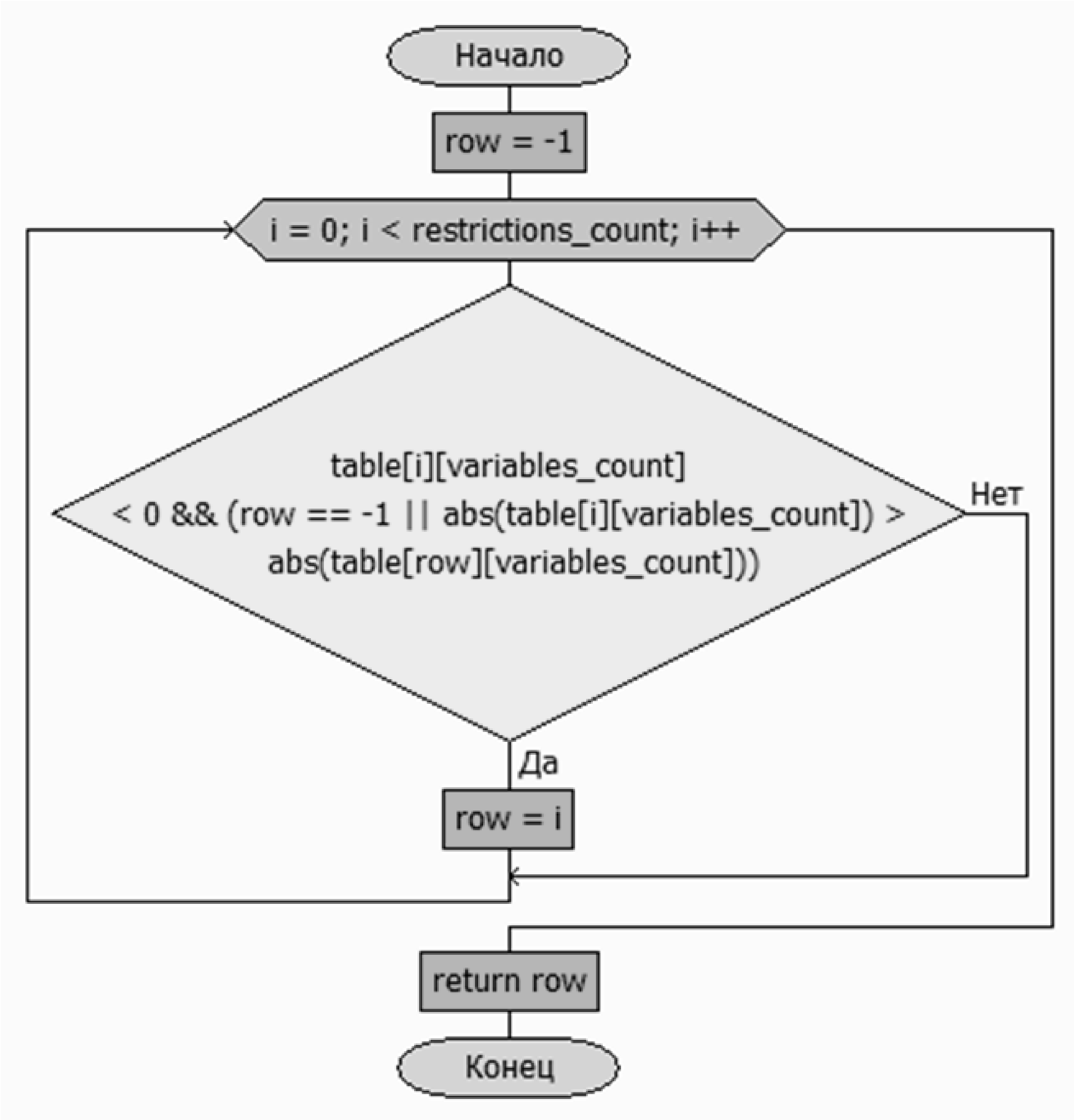
$$F(X) = 9 * 9 - 4 * 0 + 3 * 0 = 81$$

Блок – схемы:

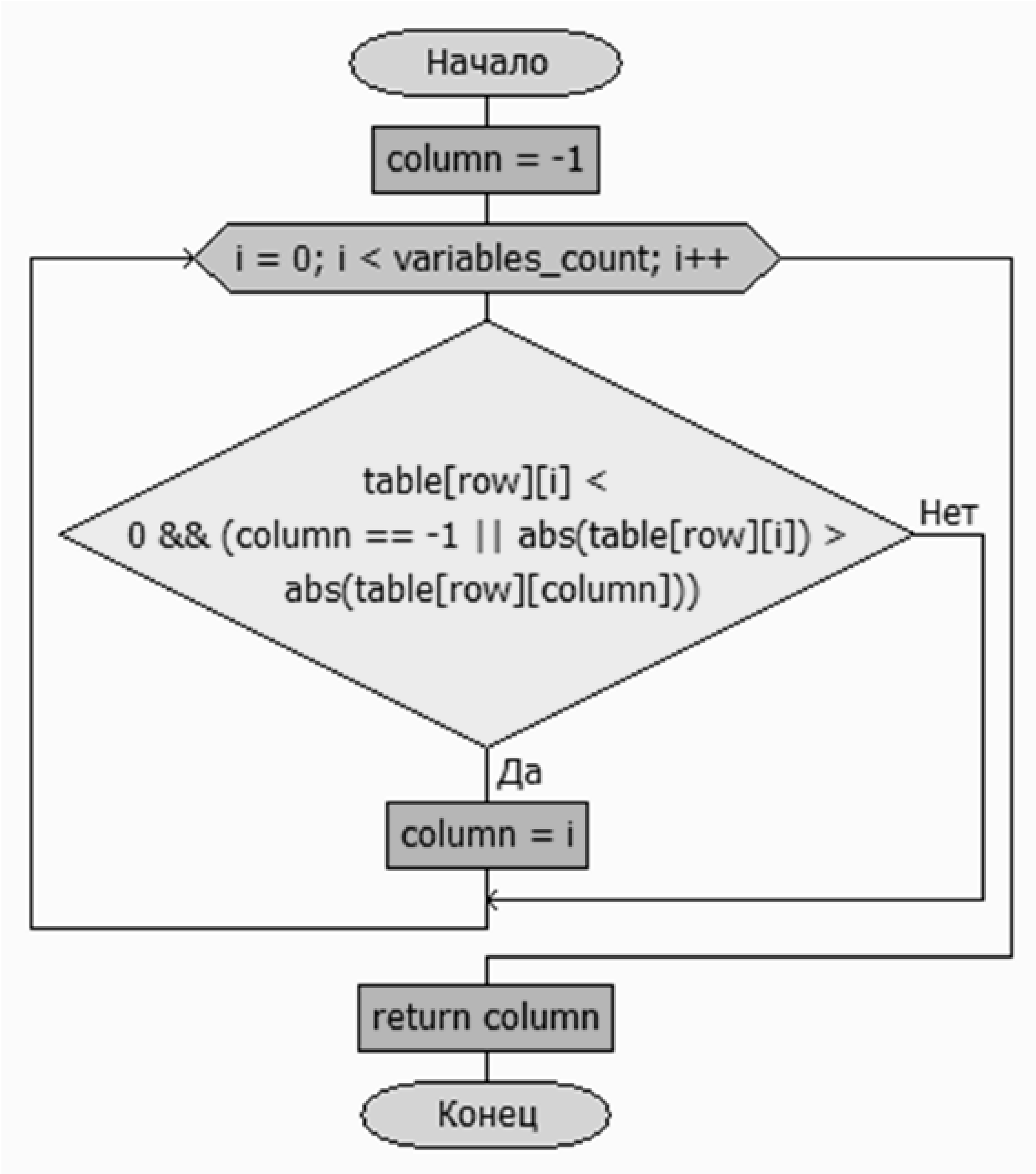
Функция Init_Table



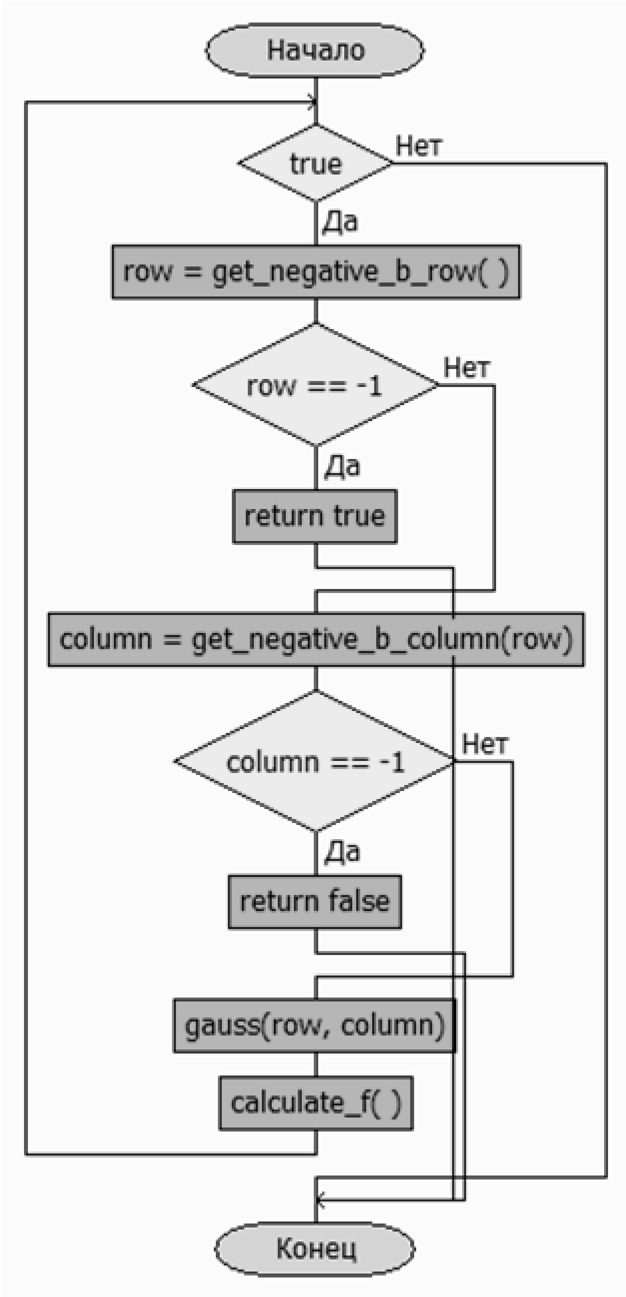
Функция get_negative_b_row



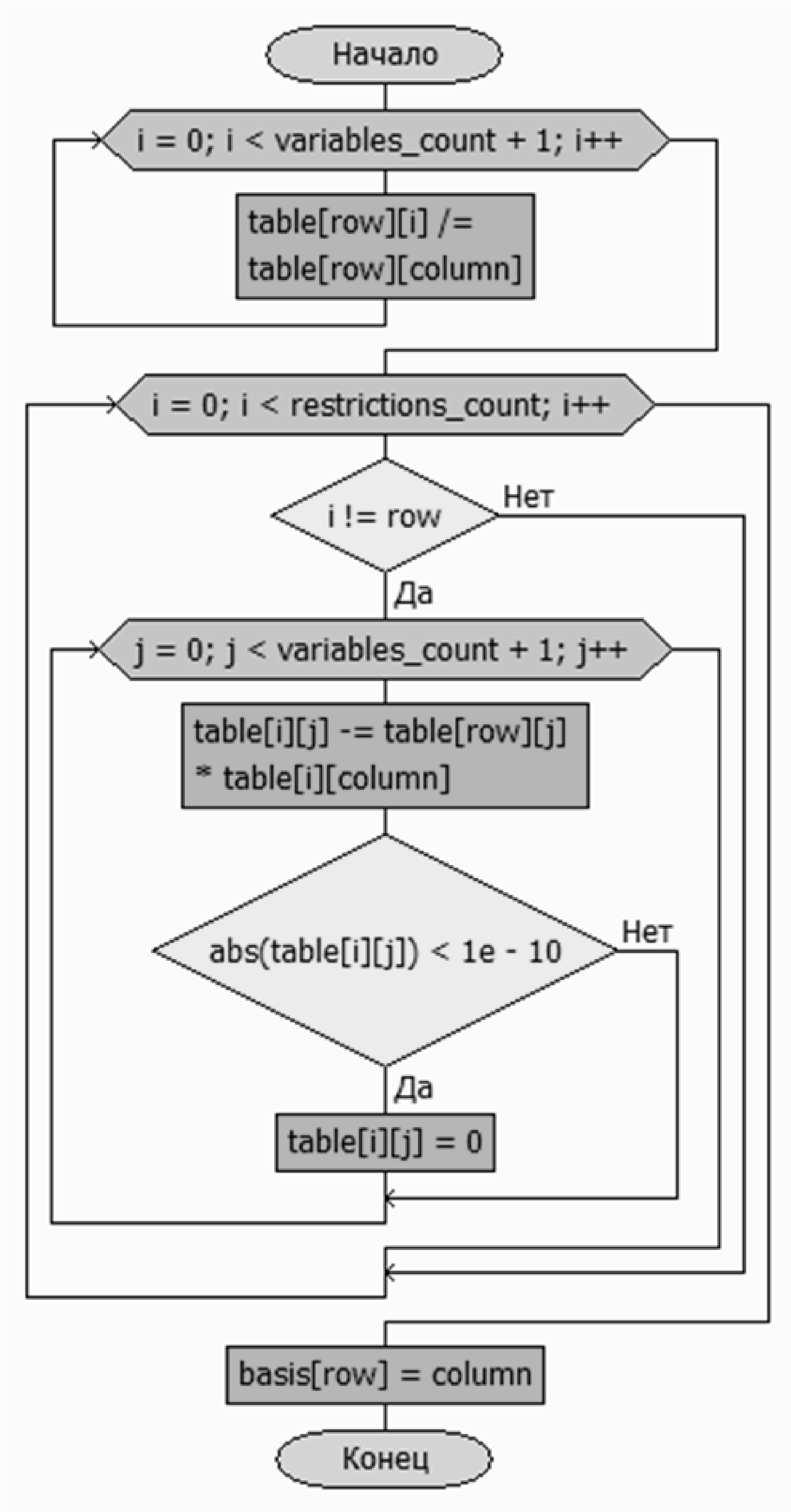
Функция get_negative_b_column



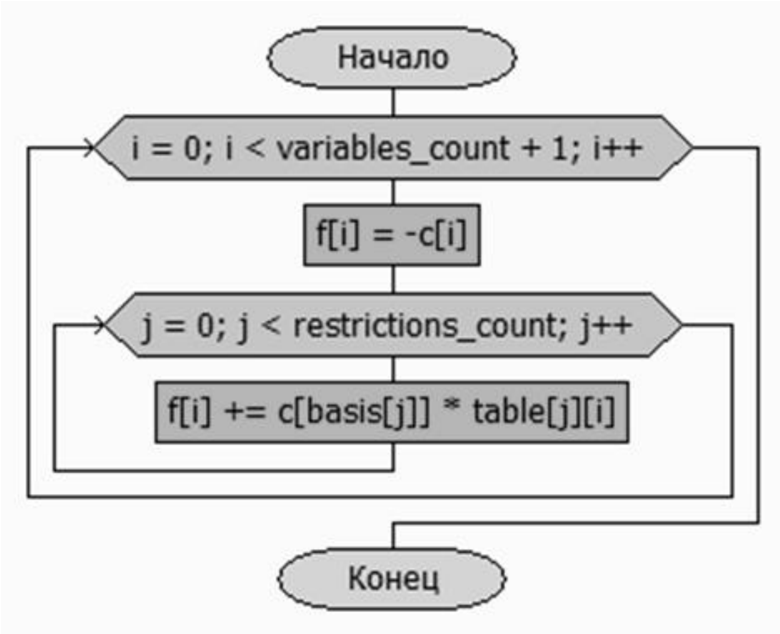
Функция remove_negative_b



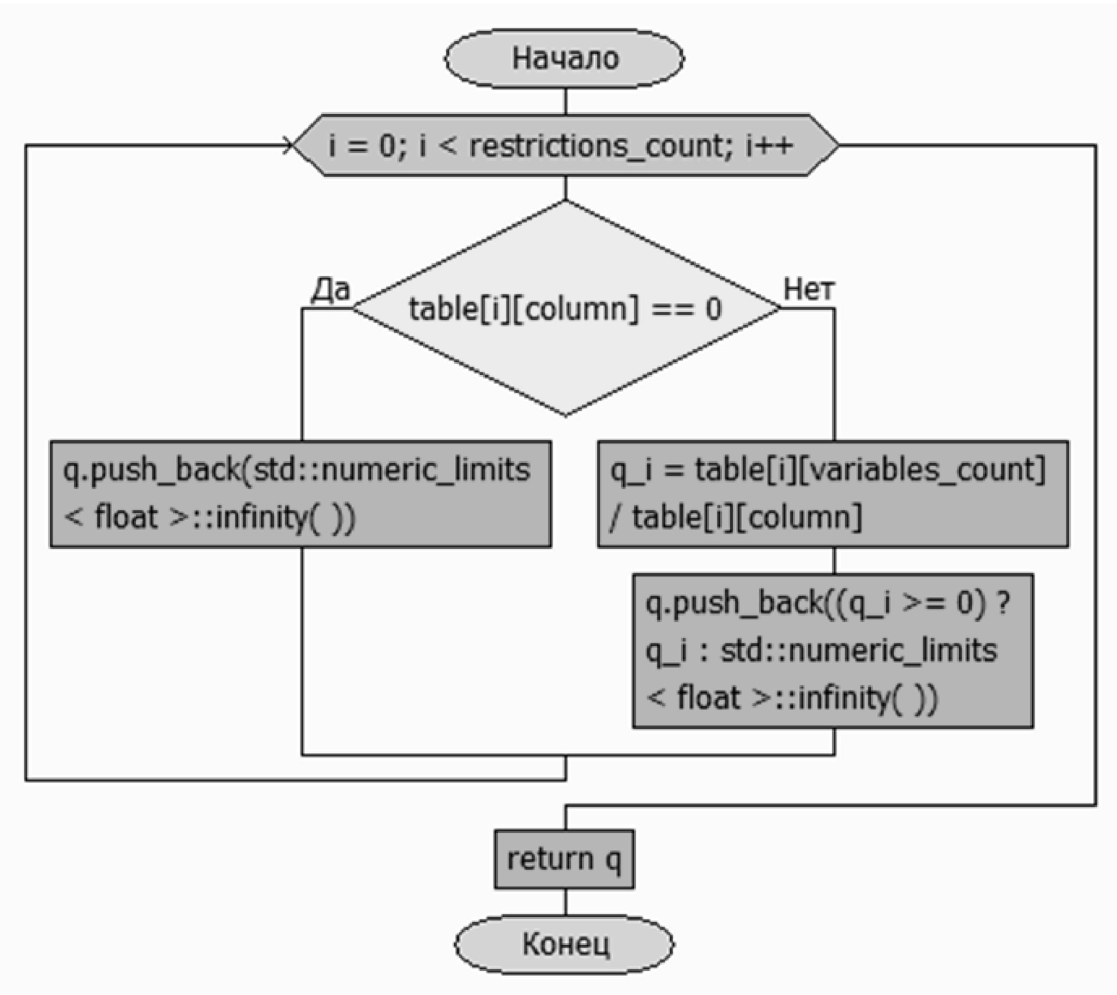
Функция gauss



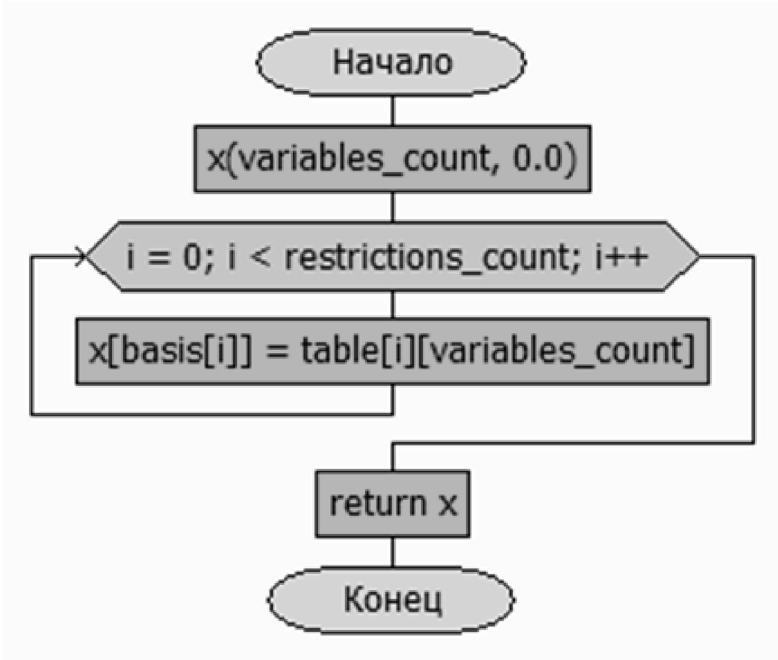
Функция calculate_f



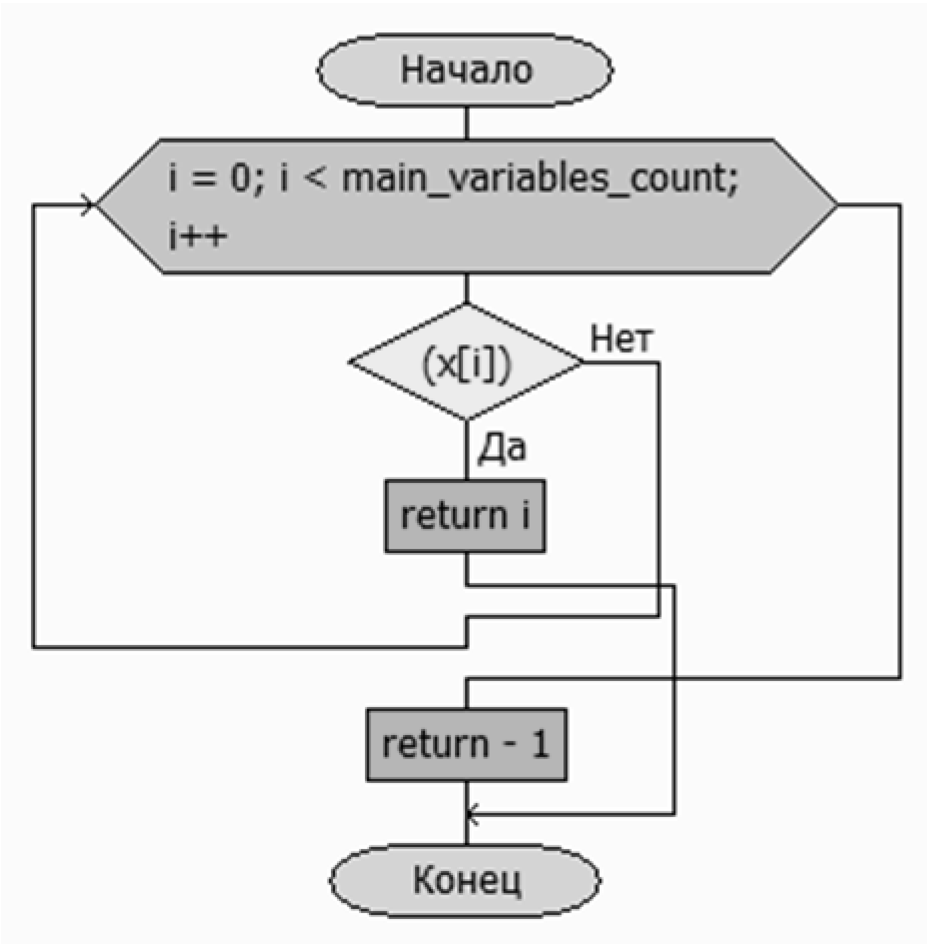
Функция get_relations



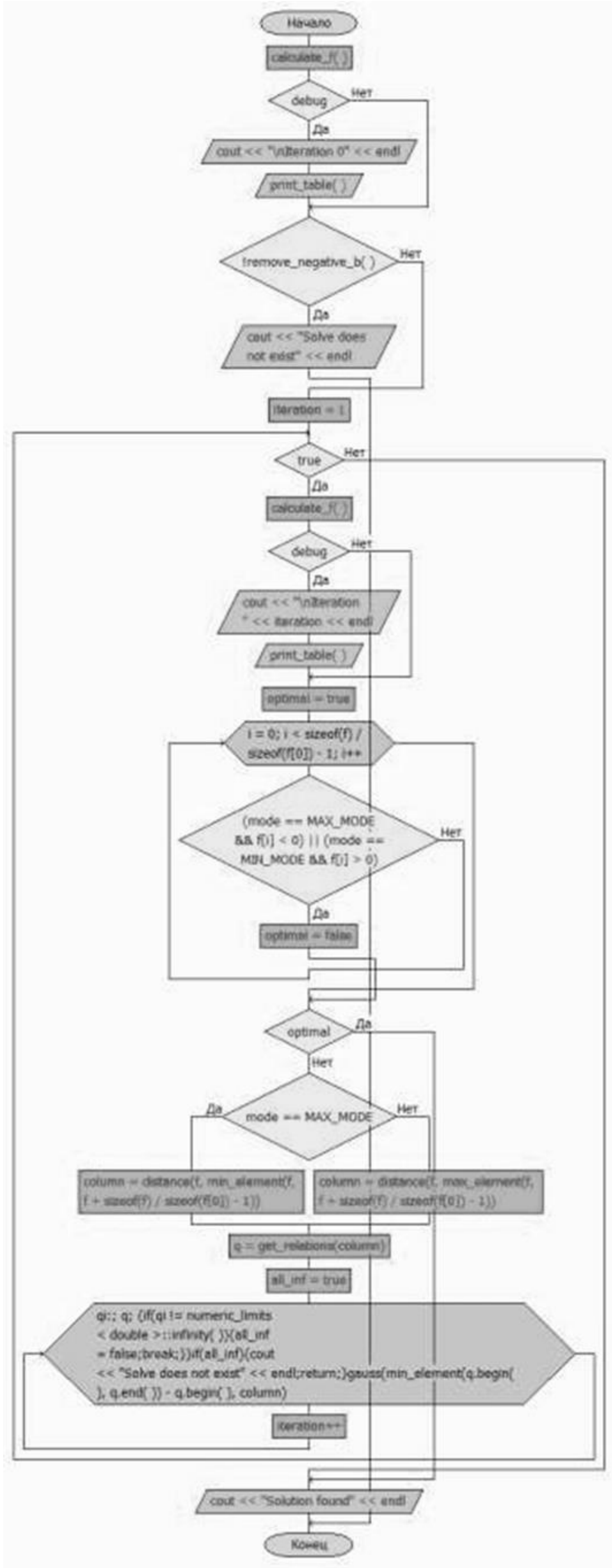
Функция get_solve



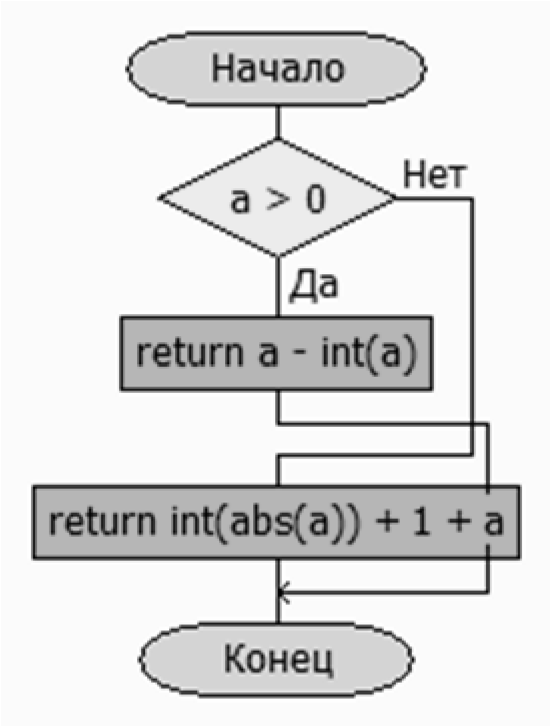
Функция get_first_real



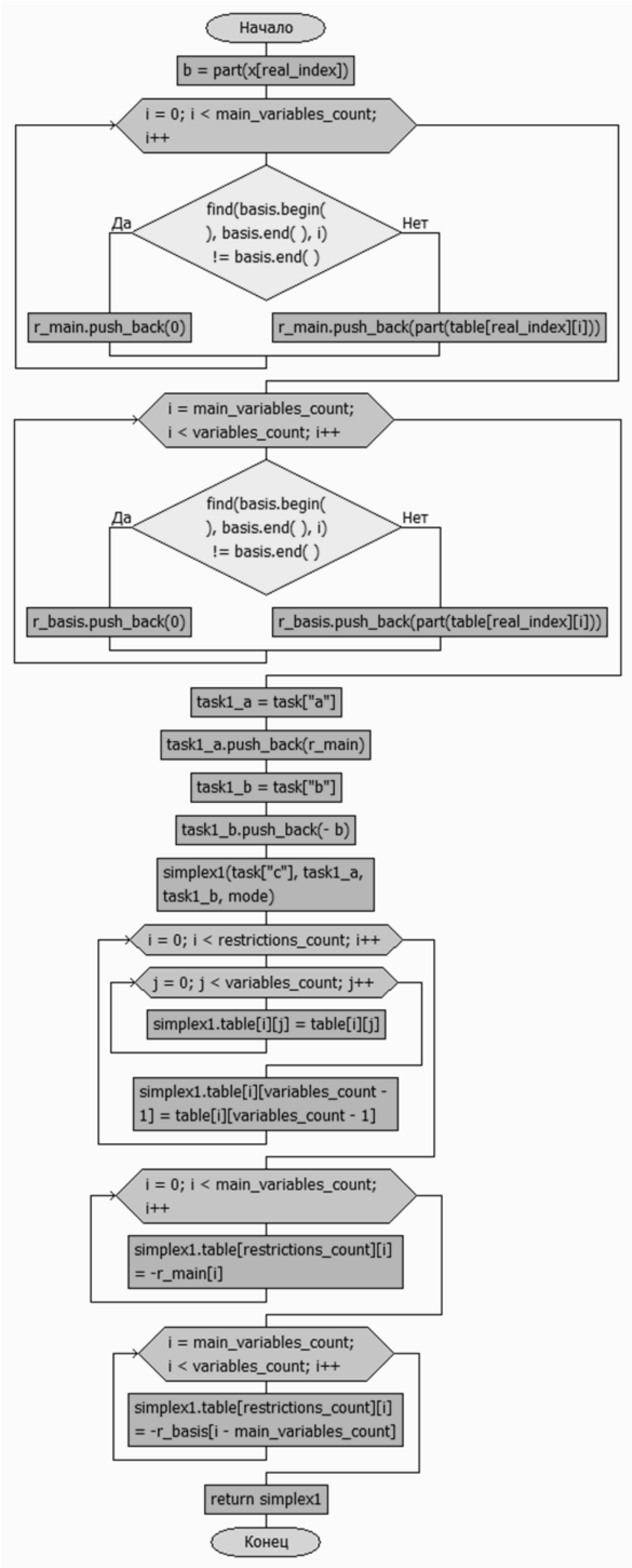
Функция solve



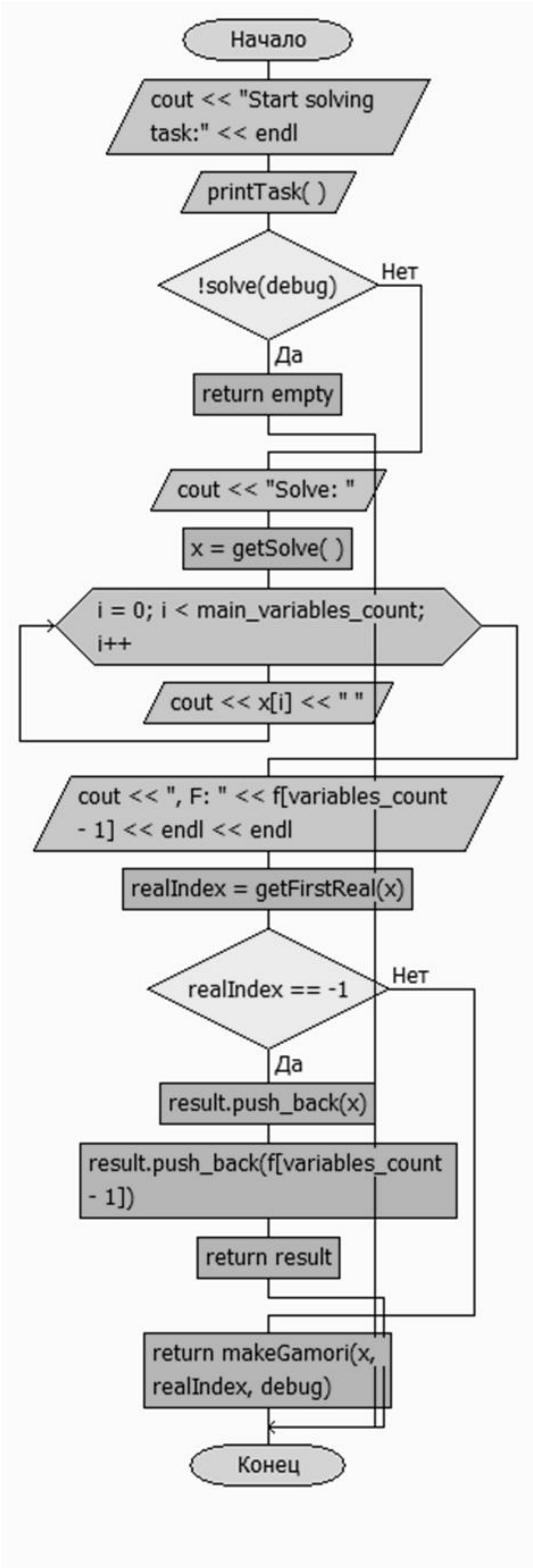
Функция part



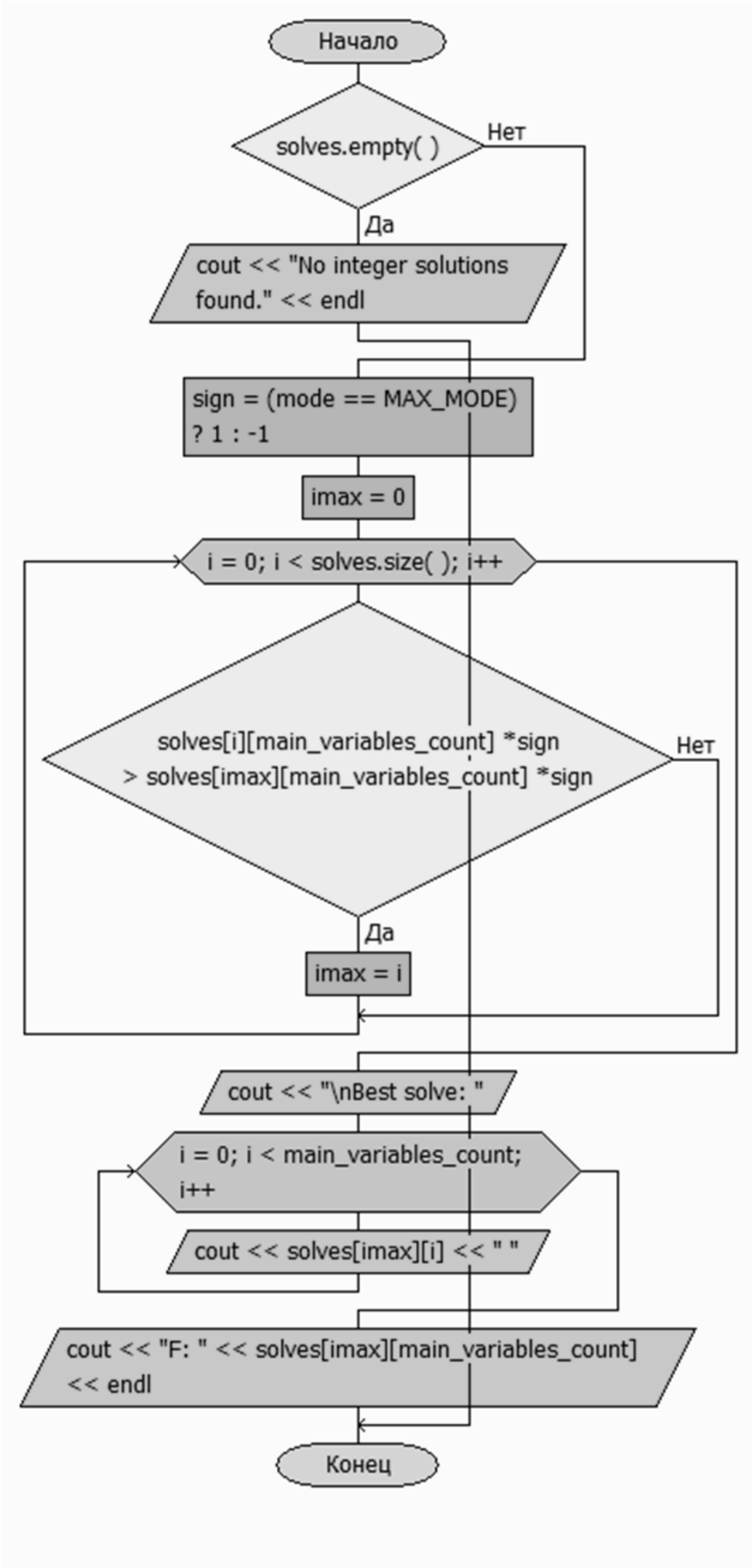
Функция make_gamori



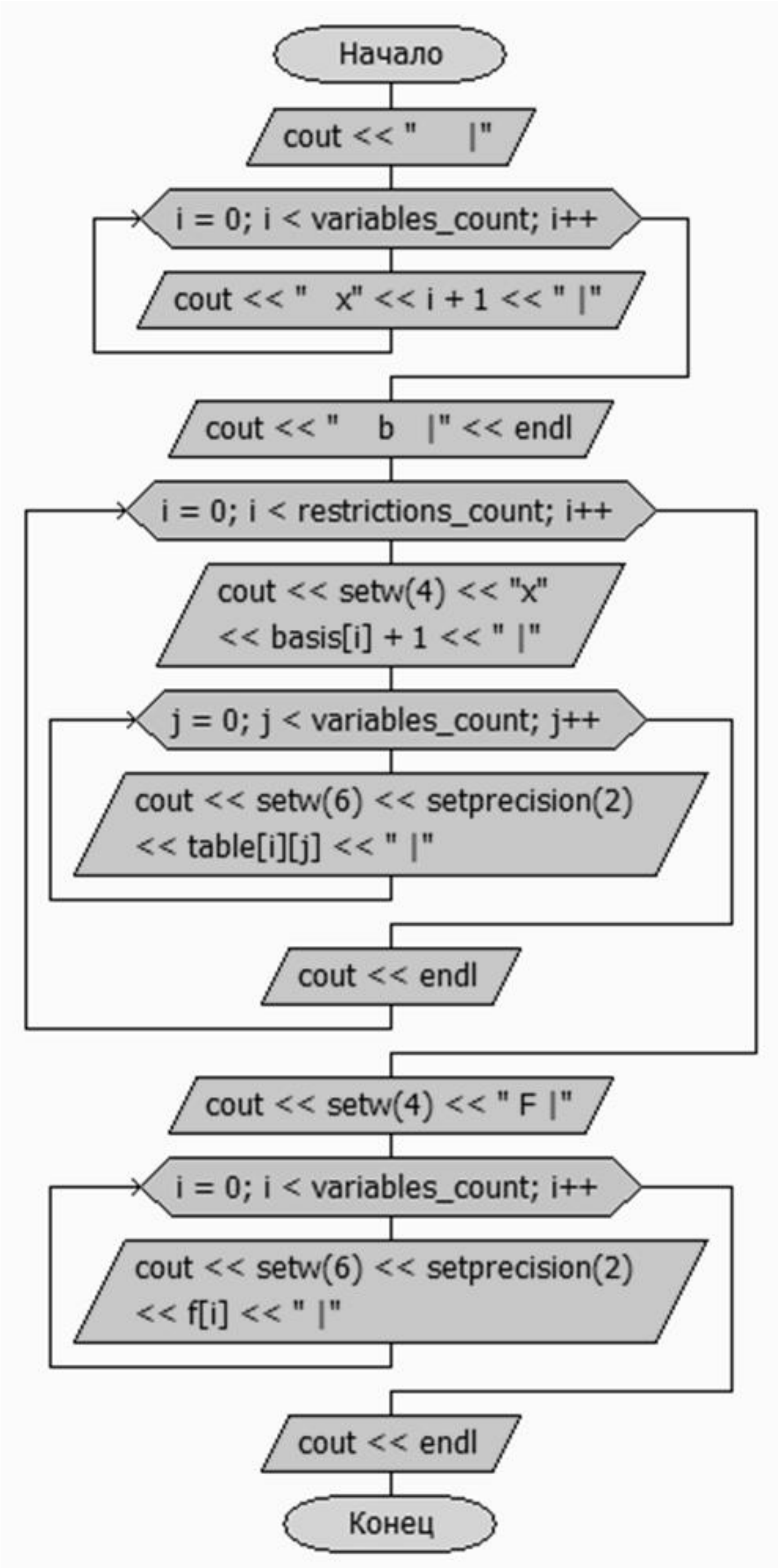
Функция solve_integer



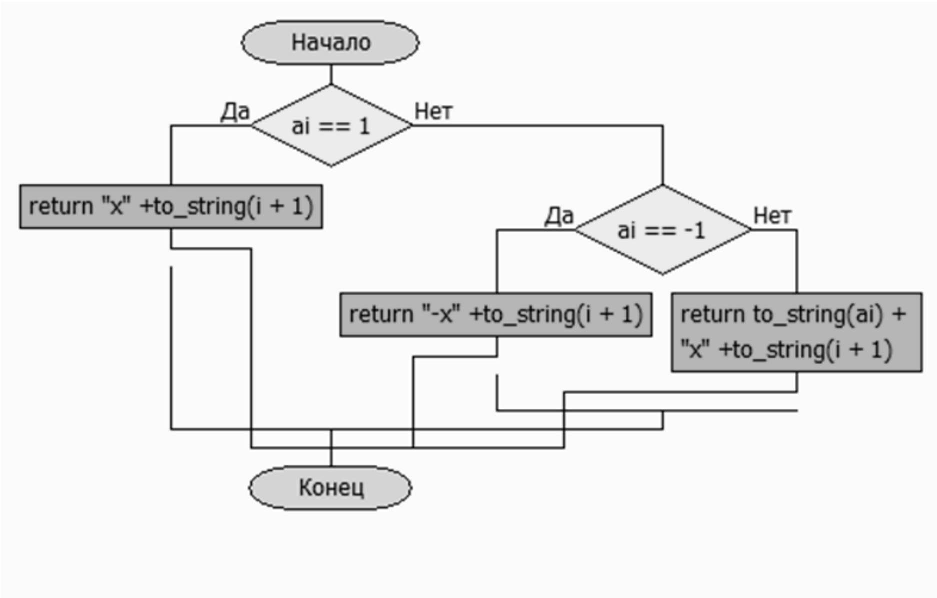
Функция print_best_solve



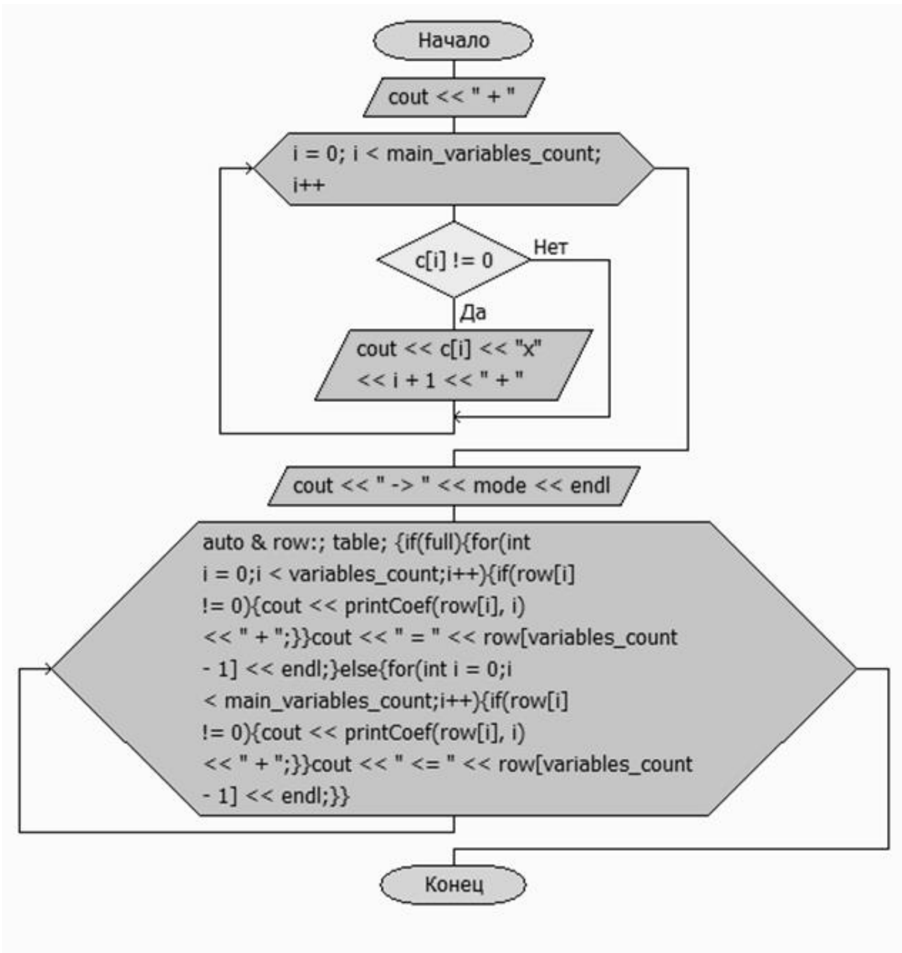
Функция print_table



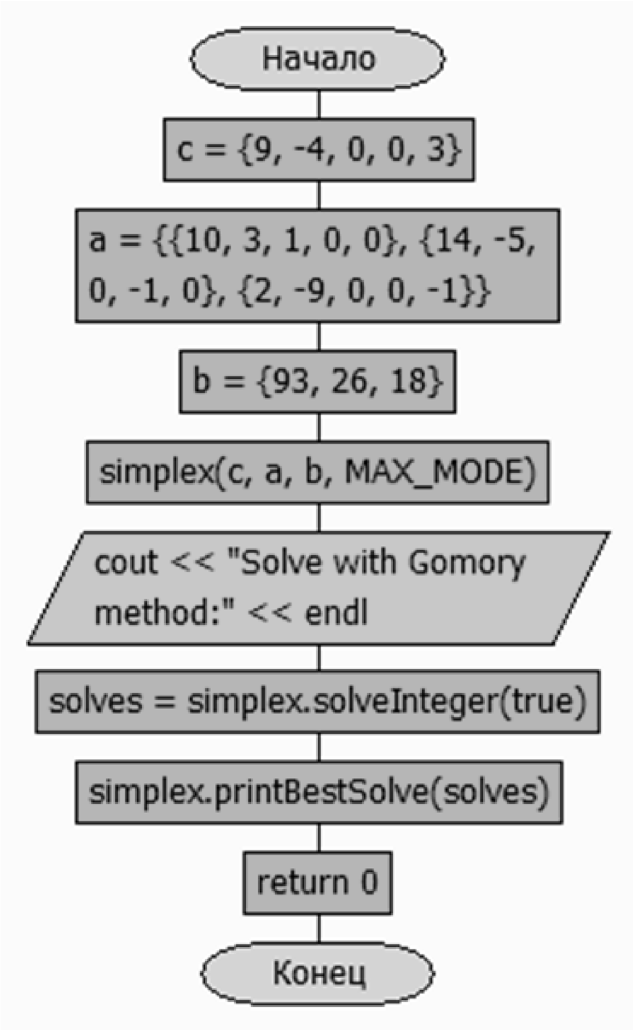
Функция print_coef



Функция print_task



Функция main



Код программы:

Алгоритм Гомори

```
import numpy as np
from copy import deepcopy
from itertools import product

MAX_MODE = 'MAX'  # режим максимизации
MIN_MODE = 'MIN'  # режим минимизации

class SimplexMethod:
    def __init__(self, c, a, b, mode):
        self.main_variables_count = a.shape[1]  # количество переменных
        self.restrictions_count = a.shape[0]  # количество ограничений
        self.variables_count = self.main_variables_count +
self.restrictions_count  # количество переменных
        self.mode = mode  # запоминаем режим работы
        self.c = np.concatenate([c, np.zeros((self.restrictions_count +
1))])  # коэффициенты функции
        self.f = np.zeros((self.variables_count + 1))  # значения функции F
        self.basis = [i + self.main_variables_count for i in
range(self.restrictions_count)]  # индексы базисных переменных
        self.task = {'a': deepcopy(a), 'b': deepcopy(b), 'c': deepcopy(c)}
# сохраняем начальную задачу
        self.init_table(a, b)  # инициализация таблицы

    def init_table(self, a, b):
        self.table = np.zeros((self.restrictions_count,
self.variables_count + 1))  # коэффициенты таблицы
        for i in range(self.restrictions_count):
            for j in range(self.main_variables_count):
                self.table[i][j] = a[i][j]
            for j in range(self.restrictions_count):
                self.table[i][j + self.main_variables_count] = int(i == j)
            self.table[i][-1] = b[i]

    # получение строки с максимальным по модулю отрицательным значением b
    def get_negative_b_row(self):
        row = -1
        for i, a_row in enumerate(self.table):
            if a_row[-1] < 0 and (row == -1 or abs(a_row[-1]) >
abs(self.table[row][-1])):
                row = i
        return row

    # получение столбца с максимальным по модулю элементом в строке
    def get_negative_b_column(self, row):
        column = -1
        for i, aij in enumerate(self.table[row][:-1]):
            if aij < 0 and (column == -1 or abs(aij) >
abs(self.table[row][column])):
                column = i
        return column

    # удаление отрицательных свободных коэффициентов
    def remove_negative_b(self):
        while True:
            row = self.get_negative_b_row()  # ищем строку, в которой
находится отрицательное b
            if row == -1:  # если не нашли такую строку
```

```
                return True  # то всё хорошо
            column = self.get_negative_b_column(row)  # ищем разрешающий
столбец
            if column == -1:
                return False  # не удалось удалить
            self.gauss(row, column)  # выполняем исключение гаусса
            self.calculate_f()

    # выполнение шага метода гаусса
    def gauss(self, row, column):
        self.table[row] /= self.table[row][column]
        for i in range(self.restrictions_count):
            if i != row:
                self.table[i] -= self.table[row] * self.table[i][column]
            for j in range(self.variables_count):
                if abs(self.table[i][j]) < 1e-10:
                    self.table[i][j] = 0
        self.basis[row] = column  # делаем переменную базисной

    # расчёт значений F
    def calculate_f(self):
        for i in range(self.variables_count + 1):
            self.f[i] = -self.c[i]
        for j in range(self.restrictions_count):
            self.f[i] += self.c[self.basis[j]] * self.table[j][i]

    # расчёт симплекс-отношений для столбца column
    def get_relations(self, column):
        q = []
        for i in range(self.restrictions_count):
            if self.table[i][column] == 0:
                q.append(np.inf)
            else:
                q_i = self.table[i][-1] / self.table[i][column]
                q.append(q_i if q_i >= 0 else np.inf)
        return q

    # получение решения
    def get_solve(self):
        x = np.zeros((self.variables_count))
        # заполняем решение
        for i in range(self.restrictions_count):
            x[self.basis[i]] = self.table[i][-1]
        return x  # возвращаем полученное решение

    # получение первого вещественного решения
    def get_first_real(self, x):
        return next((i for i, xi in
enumerate(x[:self.main_variables_count]) if xi != int(xi)), -1)

    # решение
    def solve(self, debug):
        self.calculate_f()
        if debug:
            print('\nIteration 0')
            self.print_table()
        if not self.remove_negative_b():
            print('Solve does not exist')
            return False
        iteration = 1
        while True:
```

```
            self.calculate_f()
            if debug:
                print('\nIteration', iteration)
                self.print_table()
            if all(fi >= 0 if self.mode == MAX_MODE else fi <= 0 for fi in
self.f[:-1]):  # если план оптимален
                break  # то завершаем работу
            column = (np.argmin if self.mode == MAX_MODE else
np.argmax)(self.f[:-1])  # получаем разрешающий столбец
            q = self.get_relations(column)  # получаем симплекс-отношения
для найденного столбца
            if all(qi == np.inf for qi in q):  # если не удалось найти
разрешающую строку
                print('Solve does not exist')  # сообщаем, что решения нет
                return False
            self.gauss(np.argmin(q), column)  # выполняем исключение гаусса
            iteration += 1
        return True  # решение есть

    # получение дробной части
    def part(self, a):
        if a > 0:
            return a - int(a)
        return int(abs(a)) + 1 + a

    def make_gamori(self, x, real_index, debug):
        b = self.part(x[real_index])
        r_main = [0 if i in self.basis else
self.part(self.table[real_index][i]) for i in
range(self.main_variables_count)]
        r_basis = [0 if i in self.basis else
self.part(self.table[real_index][i]) for i in
range(self.main_variables_count, self.variables_count)]
        # добавляем условия
        task1_a = np.append(self.task['a'], np.array([r_main]), 0)
        task1_b = np.append(self.task['b'], -b)
        simplex1 = SimplexMethod(self.task['c'], task1_a, task1_b,
self.mode)
        for i in range(self.restrictions_count):
            for j in range(self.variables_count):
                simplex1.table[i][j] = self.table[i][j]
                simplex1.table[i][-1] = self.table[i][-1]
        for i in range(self.main_variables_count):
            simplex1.table[-1][i] = -r_main[i]
        for i in range(self.main_variables_count, self.variables_count):
            simplex1.table[-1][i] = -r_basis[i - self.main_variables_count]
        return simplex1.solve_integer(debug)

    # получение целочисленных решений
    def solve_integer(self, debug=False):
        print('Start solving task:')
        self.print_task()
        # если решение не было найдено, то добавляем пустое решение
        if not self.solve(debug):
            return []
        print("Solve:", self.get_solve()[:self.main_variables_count], ',
F:', self.f[-1], '\n')
        x = self.get_solve()
        real_index = self.get_first_real(x)
        # если решение содержало только целые числа, то возвращаем решение
        if real_index == -1:
```

```
            return [(x, self.f[-1])]
        return self.make_gamori(x, real_index, debug)

    # вывод лучшего решения
    def print_best_solve(self, solves):
        sign = 1 if self.mode == MAX_MODE else -1
        imax = 0
        for i, solve in enumerate(solves):
            if solve[1] * sign > solves[imax][1] * sign:
                imax = i
        x, f = solves[imax]
        print('\nBest solve: ', x[:self.main_variables_count], 'F:', f)

    # вывод симплекс-таблицы
    def print_table(self):
        print('  |' + ''.join([' x%-3d |' % (i + 1) for i in
range(self.variables_count)]) + ' b |')
        for i in range(self.restrictions_count):
            print('%4s |' % ('x' + str(self.basis[i] + 1)) + ''.join(['
%6.2f |' % aij for j, aij in enumerate(self.table[i])]))
        print(' F |' + ''.join([' %6.2f |' %

 aij for aij in self.f]))
        # print(' x |' + ''.join([' %6.2f |' % xi for xi in
self.get_solve()]))

    # вывод коэффициента
    def print_coef(self, ai, i):
        if ai == 1:
            return 'x%d' % (i + 1)
        if ai == -1:
            return '-x%d' % (i + 1)
        return '%.2fx%d' % (ai, i + 1)

    # вывод задачи
    def print_task(self, full=True):
        print(' + '.join(['%.2fx%d' % (ci, i + 1) for i, ci in
enumerate(self.c[:self.main_variables_count]) if ci != 0]), '-> ',
self.mode)
        for row in self.table:
            if full:
                print(' + '.join([self.print_coef(ai, i) for i, ai in
enumerate(row[:self.variables_count]) if ai != 0]), '=', row[-1])
            else:
                print(' + '.join([self.print_coef(ai, i) for i, ai in
enumerate(row[:self.main_variables_count]) if ai != 0]), '<=', row[-1])

def main():
    c = np.array([9, -4, 0, 0, 3])
    a = np.array([
        [10, 3, 1, 0, 0],
        [14, -5, 0, -1, 0],
        [2, -9, 0, 0, -1]
    ])
    b = np.array([93, 26, 18])
    simplex = SimplexMethod(c, a, b, MAX_MODE)
    print("Solve with Gomory method:")
    solves = simplex.solve_integer(True)
    simplex.print_best_solve(solves)
```

```
if __name__ == '__main__':
    main()
```

Алгоритм ветвей и границ

```
import heapq


def branch_and_bound(graph, start_node):
    n = len(graph)
    min_heap = []
    for i in range(1, n):
        heapq.heappush(min_heap, (graph[start_node][i], [start_node, i]))

    while min_heap:
        current_cost, current_path = heapq.heappop(min_heap)
        last_node = current_path[-1]

        if len(current_path) == n:
            if graph[last_node][start_node] != 0:
                return current_cost + graph[last_node][start_node],
current_path
        else:
            for i in range(n):
                if i not in current_path and graph[last_node][i] != 0:
                    heapq.heappush(min_heap, (current_cost +
graph[last_node][i], current_path + [i]))
```

Результат работы программы:

```
Solve with Gomory method:
Start solving task:
9.00x1 + -4.00x2 + 3.00x5 -> MAX
10.00x1 + 3.00x2 + x3 + x6 = 93.0
14.00x1 + -5.00x2 + -x4 + x7 = 26.0
2.00x1 + -9.00x2 + -x5 + x8 = 18.0

Iteration 0
| x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | b |
x6 | 10.00 | 3.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 93.00 |
x7 | 14.00 | -5.00 | 0.00 | -1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 26.00 |
x8 | 2.00 | -9.00 | 0.00 | 0.00 | -1.00 | 0.00 | 0.00 | 1.00 | 18.00 |
F | -9.00 | 4.00 | 0.00 | 0.00 | -3.00 | 0.00 | 0.00 | 0.00 | 0.00 |

Iteration 1
| x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | b |
x6 | 10.00 | 3.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 93.00 |
x7 | 14.00 | -5.00 | 0.00 | -1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 26.00 |
x8 | 2.00 | -9.00 | 0.00 | 0.00 | -1.00 | 0.00 | 0.00 | 1.00 | 18.00 |
F | -9.00 | 4.00 | 0.00 | 0.00 | -3.00 | 0.00 | 0.00 | 0.00 | 0.00 |

Iteration 2
| x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | b |
x6 | 0.00 | 6.57 | 1.00 | 0.71 | 0.00 | 1.00 | -0.71 | 0.00 | 74.43 |
x1 | 1.00 | -0.36 | 0.00 | -0.07 | 0.00 | 0.00 | 0.07 | 0.00 | 1.86 |
x8 | 0.00 | -8.29 | 0.00 | 0.14 | -1.00 | 0.00 | -0.14 | 1.00 | 14.29 |
F | 0.00 | 0.79 | 0.00 | -0.64 | -3.00 | 0.00 | 0.64 | 0.00 | 16.71 |

Optimal plan: x1 = 9, x2 = 0, x3 = 3, x4 = 100, x5 = 0
Target function: 81
```

Вывод: Освоение методов отсечения Гомори для полностью целочисленных задач является важным аспектом в решении комбинаторных оптимизационных проблем. Метод отсечения Гомори используется для улучшения решений задач целочисленного программирования путем добавления дополнительных целочисленных ограничений, что повышает качество найденного решения. Этот метод основан на выявлении и добавлении так называемых "гомори-неравенств" к существующим ограничениям, что улучшает эффективность поиска оптимального целочисленного решения. Реализация этого алгоритма в программе позволяет проводить более глубокий и эффективный поиск оптимального решения в целочисленном программировании, что улучшает результаты решения различных оптимизационных задач.