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)

ИИТУС
БГТУ им. В.Г. Шухова

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №1

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

Выполнил: ст. группы ПВ-223
Дмитриев Андрей
Проверил:
Вирченко Ю.П.

Белгород, 2024 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 в соответствии с вариантом вручную (подготовить тестовые данные).

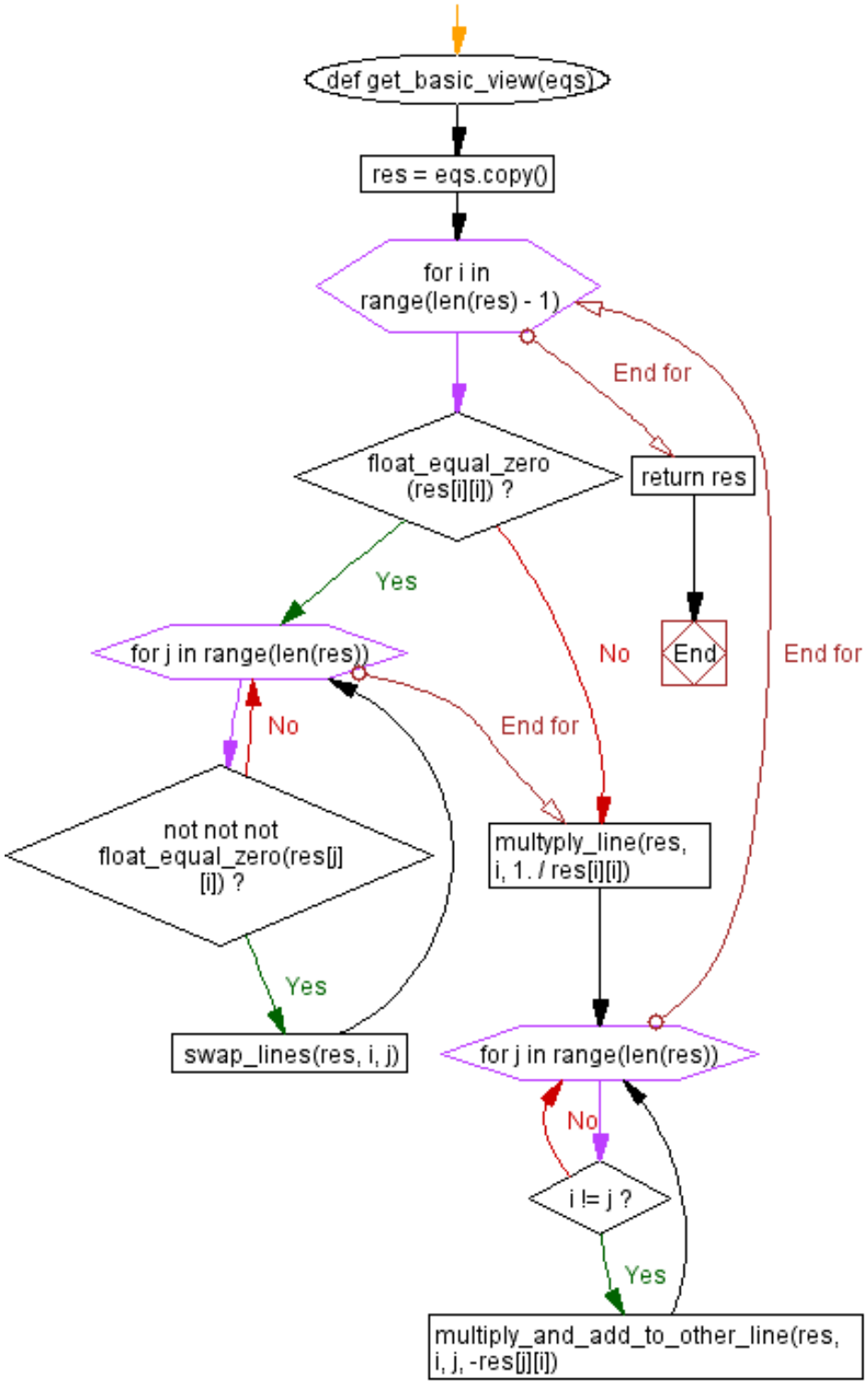
Вариант 2:

Задание:

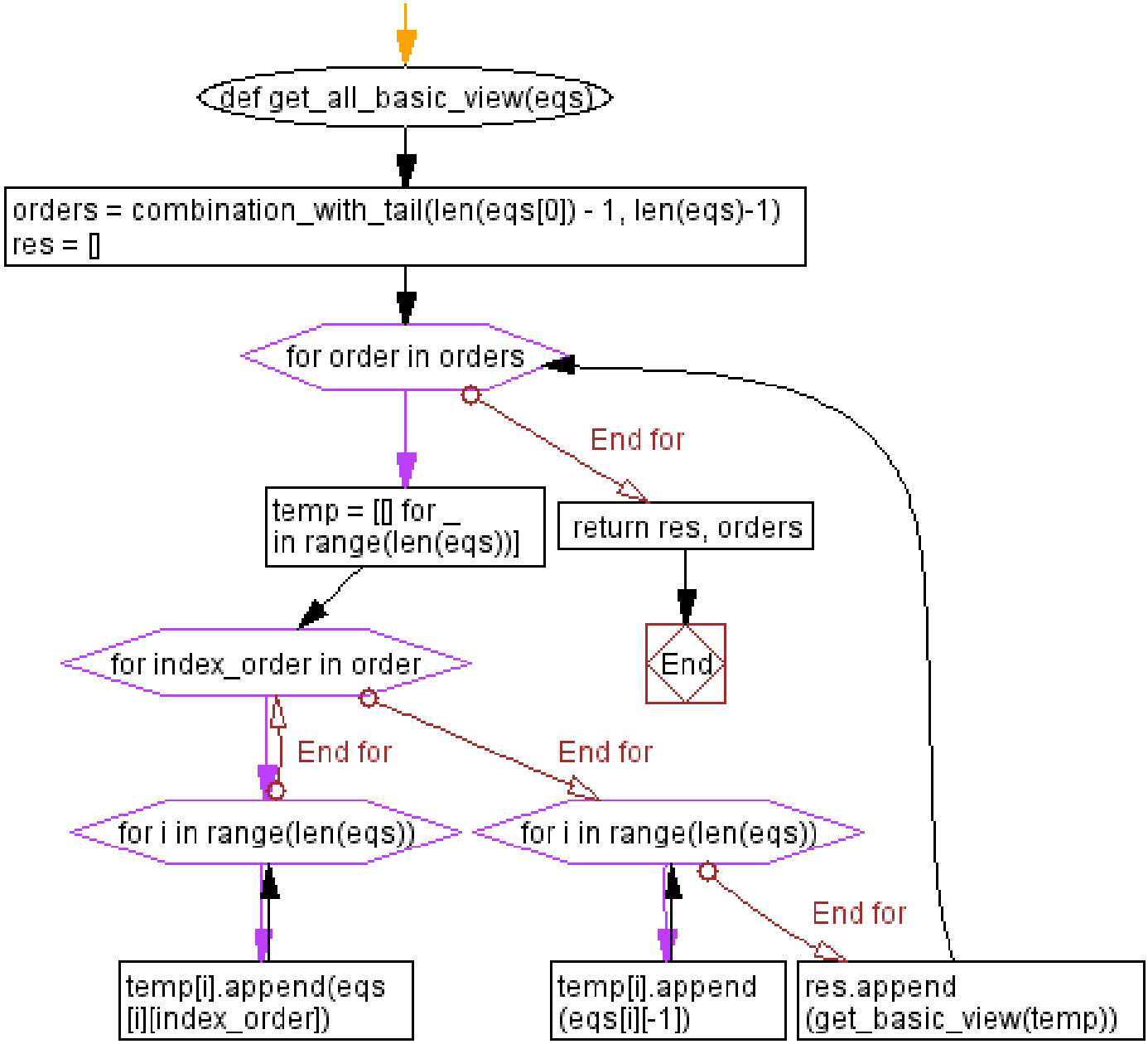
$$\begin{cases} 6x_1 - 2x_2 + 2x_3 + 5x_5 = 6 \\ x_1 - 2x_2 + 3x_3 + 4x_4 - x_5 = 2 \\ 4x_1 - 7x_2 + 2x_3 + x_4 + 3x_5 = 0 \\ 2x_1 - 4x_2 + 6x_3 + 8x_4 - 2x_5 = 4 \end{cases}$$

Блок схема:

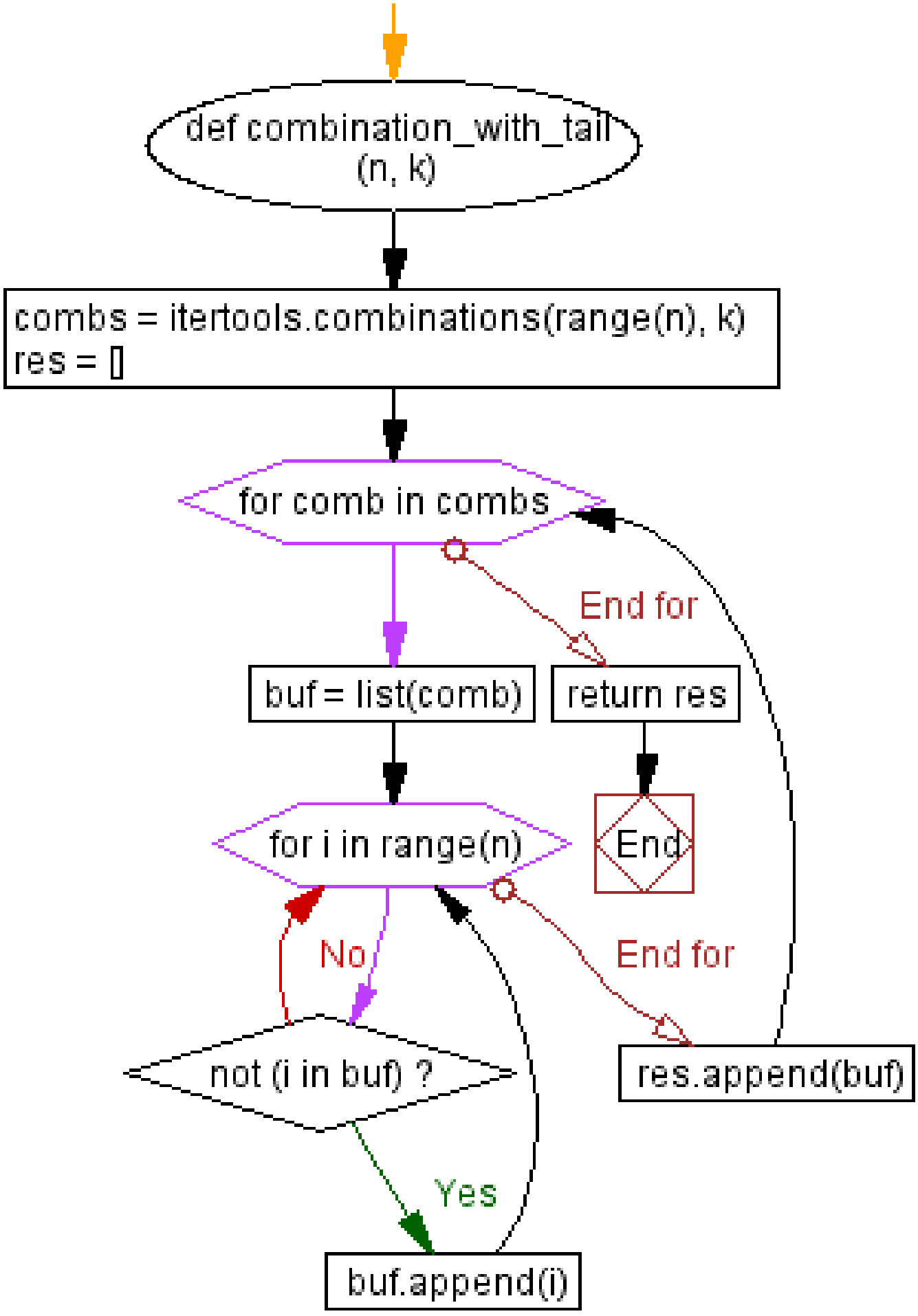
Нахождение одного базового вида:



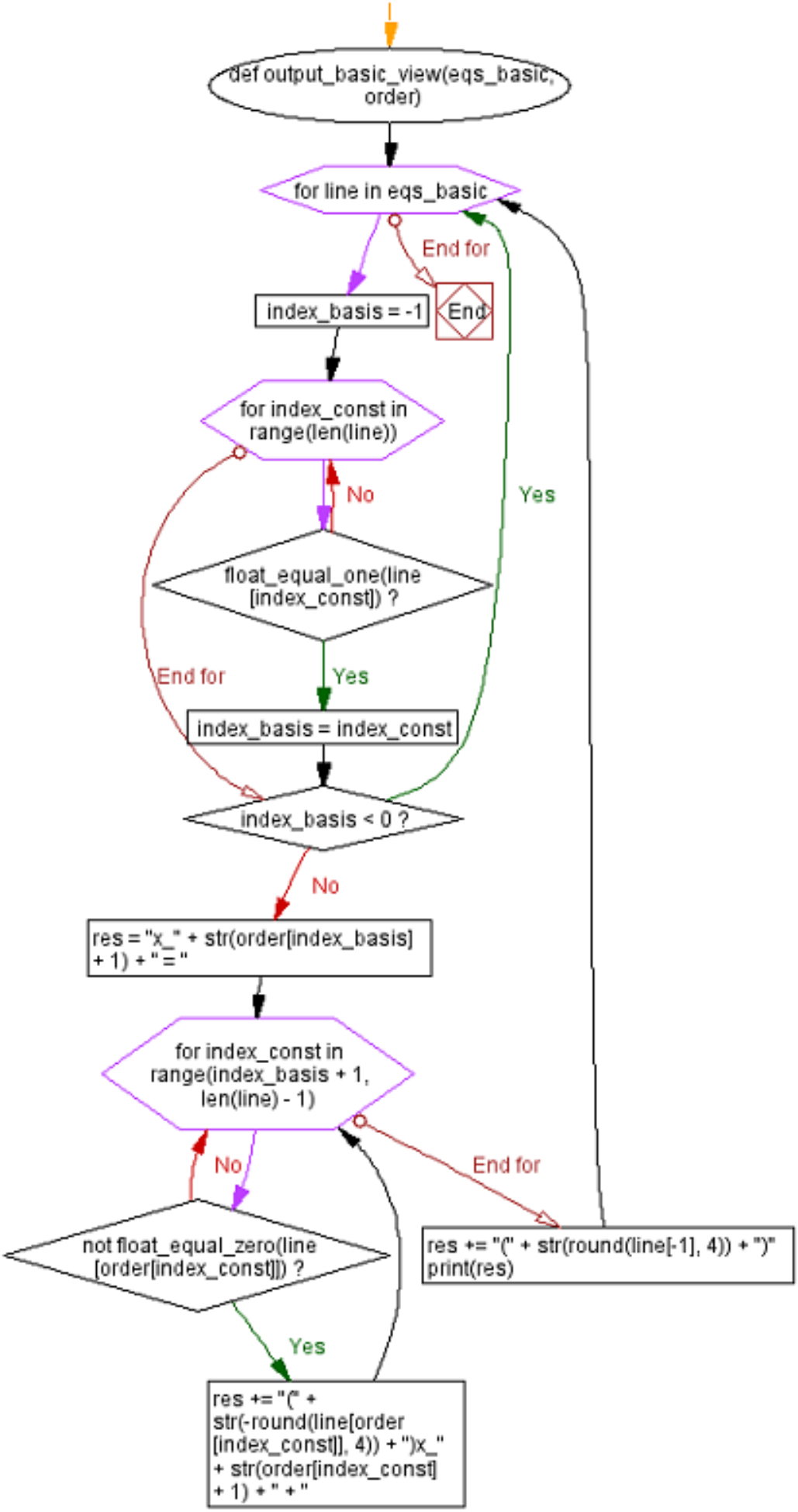
Нахождение всех базисных видов:



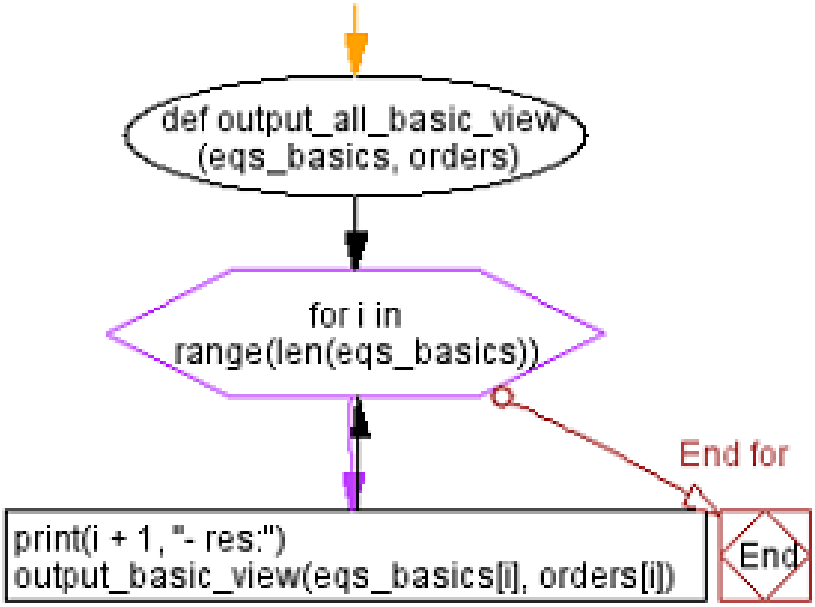
Вспомогательная рекурсивная функция для получения всех сочетаний:



Вывод базого вида:



Вывод всех базисных видов:



Листинги кода:

```
def get_basic_view(eqs):
    res = eqs.copy()
    # Главный цикл для построчного прохода
    for i in range(len(res) - 1):
        # Условие которое запускает цикл для
        # поиска колонки с ненулевым значением
        # по диагонали
        if float_equal_zero(res[i][i]):
            for j in range(len(res)):
                if not not not float_equal_zero(res[j][i]):
                    swap_lines(res, i, j)
        # Приводим к единице на диагонали
        multyply_line(res, i, 1. / res[i][i])
        # Приводим к нулям в текущей колонке
        for j in range(len(res)):
            if i != j:
                multiply_and_add_to_other_line(res, i, j, -res[j][i])

    return res


def get_all_basic_view(eqs):
    # Получаем все перестановки
    orders = combination_with_tail(len(eqs[0]) - 1, len(eqs)-1)

    # Формируем все базисные виды, ассоциируя
    # их с сгенерированными порядками
    res = []
    for order in orders:
        temp = [[] for _ in range(len(eqs))]
        for index_order in order:
            for i in range(len(eqs)):
                temp[i].append(eqs[i][index_order])

        for i in range(len(eqs)):
            temp[i].append(eqs[i][-1])

        res.append(get_basic_view(temp))

    return res, orders
```

```
def combination_with_tail(n, k):
    # Формируем комбинации
    combs = itertools.combinations(range(n), k)
    # Приписываем остальную часть
    res = []
    for comb in combs:
        buf = list(comb)
        for i in range(n):
            if not (i in buf):
                buf.append(i)

        res.append(buf)

    return res



def output_basic_view(eqs_basic, order):
    for line in eqs_basic:
        # Производим поиск единицы для
        # определения его в базисе
        index_basis = -1
        for index_const in range(len(line)):
            if float_equal_one(line[index_const]):
                index_basis = index_const
                break
        # Если переменная не найдена -
        # значит пропускаем итерацию
        if index_basis < 0:
            continue
        # Вывод с опорой на заданный порядок
        res = "x_" + str(order[index_basis] + 1) + " = "
        for index_const in range(index_basis + 1, len(line) - 1):
            if not float_equal_zero(line[order[index_const]]):
                res += "(" + str(-round(line[order[index_const]], 4)) + ")x " +
str(order[index_const] + 1) + " + "
        res += "(" + str(round(line[-1], 4)) + ")"

        print(res)


def output_all_basic_view(eqs_basics, orders):
    for i in range(len(eqs_basics)):
        print(i + 1, "- результат:")
        output_basic_view(eqs_basics[i], orders[i])
```

Тестовые данные:

```
def test_one_iter():
    eqs = [
        [6., -2., 2., 0., 5., 6.],
        [1., -2., 3., 4., -1., 2.],
        [4., -7., 2., 1., 3., 0.],
        [2., -4., 6., 8., -2., 4.],
    ]

    output_basic_view(get_basic_view(eqs), [0, 1, 2, 3, 4])
```

Результат программы:

```
T:\2kurs2sem\OperRes\lab1\venv\Scripts\py
Получение первого базиса:
x_1 = (0.5)x_4 + (-1.0595)x_5 + (0.9762)
x_2 = (0.0238)x_5 + (0.8095)
x_3 = (-1.5)x_4 + (0.7024)x_5 + (0.881)

Process finished with exit code 0
```

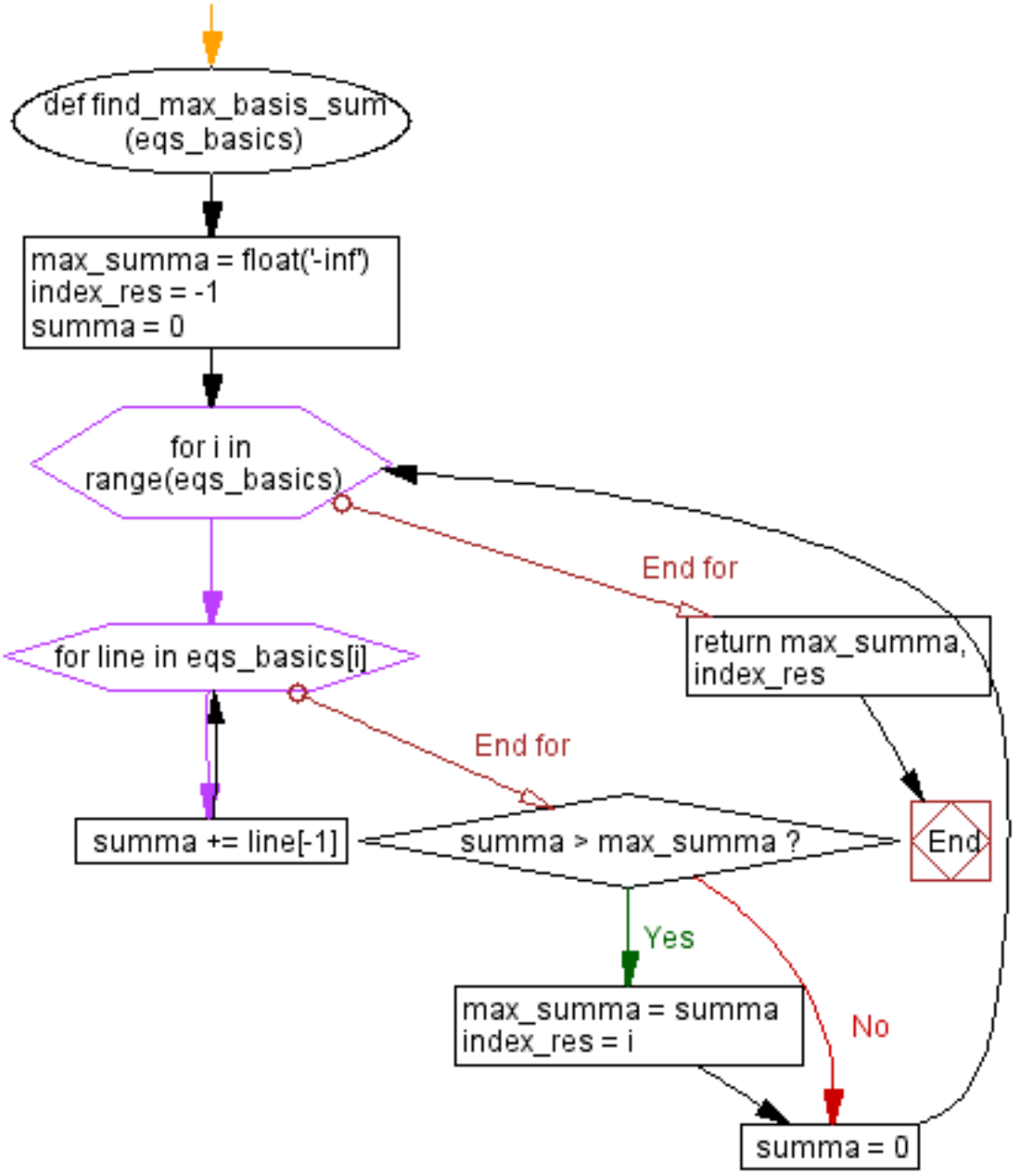
Тестовые данные:

```
def test_shaffle():
    eqs = [
        [6., -2., 2., 0., 5., 6.],
        [1., -2., 3., 4., -1., 2.],
        [4., -7., 2., 1., 3., 0.],
        [2., -4., 6., 8., -2., 4.],
    ]

    temp = get_all_basic_view(eqs)
    output_all_basic_view(temp[0], temp[1])
```

Результат программы:

```
T:\2kurs2sem\OperRes\lab1\venv\Scripts\python.exe T:/2kurs2sem/OperRes/lab1/main.py
Получение всех базисов:
1 - результат:
x_1 = (0.5)x_4 + (-1.0595)x_5 + (0.9762)
x_2 = (0.0238)x_5 + (0.8095)
x_3 = (-1.5)x_4 + (0.7024)x_5 + (0.881)
2 - результат:
x_1 = (-0.3333)x_4 + (-0.8254)x_5 + (1.2698)
x_2 = (0.0238)x_5 + (0.8095)
x_4 = (-1.0)x_3 + (0.4683)x_5 + (0.5873)
3 - результат:
x_1 = (-1.7627)x_5 + (-1.5085)x_4 + (2.3051)
x_2 = (0.0508)x_5 + (0.0339)x_4 + (0.7797)
x_5 = (-1.0)x_3 + (1.4237)x_4 + (-1.2542)
4 - результат:
x_1 = (0.3333)x_4 + (-0.3333)x_2 + (-0.8333)x_5 + (1.0)
x_4 = (-0.6667)x_2 + (0.4583)x_5 + (0.25)
5 - результат:
x_1 = (0.5)x_5 + (-44.5)x_4 + (37.0)
x_3 = (-1.5)x_5 + (-1.0)x_2 + (29.5)x_4 + (-23.0)
x_5 = (42.0)x_4 + (-34.0)
6 - результат:
x_1 = (-34.6667)x_4 + (-0.3333)x_5 + (29.3333)
x_4 = (-0.6667)x_5 + (-1.0)x_2 + (-15.3333)
x_5 = (-1.0)x_3 + (-34.0)
7 - результат:
x_2 = (-1.0)x_1 + (0.0238)x_5 + (0.8095)
x_3 = (-3.0)x_4 + (-2.4762)x_5 + (3.8095)
x_4 = (2.119)x_5 + (-1.9524)
8 - результат:
x_2 = (0.0112)x_5 + (-1.0)x_1 + (-0.0225)x_4 + (0.8315)
x_3 = (-1.1685)x_5 + (-0.6629)x_4 + (1.5281)
x_5 = (-0.9438)x_4 + (0.9213)
9 - результат:
x_2 = (-0.0288)x_4 + (-0.0096)x_5 + (-1.0)x_1 + (0.8462)
x_4 = (-0.8558)x_5 + (1.3077)
x_5 = (-1.0)x_3 + (1.5385)
10 - результат:
x_3 = (-3.0)x_4 + (-104.0)x_5 + (-1.0)x_1 + (88.0)
x_4 = (89.0)x_5 + (-1.0)x_2 + (-74.0)
x_5 = (-34.0)

Process finished with exit code 0
```

Нахождение оптимального опорного плана методом прямого перебора:



Листинг кода:

```
def find_max_basis_sum(eqs_basics):
    # Поиск максимальной суммы свободных членов
    max_summa = float('-inf')
    index_res = -1
    summa = 0
    for i in range(eqs_basics):
        for line in eqs_basics[i]:
            summa += line[-1]

        if summa > max_summa:
            max_summa = summa
            index_res = i

        summa = 0

    return max_summa, index_res
```

Тестовые данные:

```
def test_get_plan():
    eqs = [
        [6., -2., 2., 0., 5., 6.],
        [1., -2., 3., 4., -1., 2.],
        [4., -7., 2., 1., 3., 0.],
        [2., -4., 6., 8., -2., 4.],
    ]

    temp = find_max_basis_sum(get_all_basic_view(eqs)[0])
    print(temp[0], "- максимальная сумма на позиции:", temp[1])
```

Результат программы:

```
T:\2kurs2sem\OperRes\lab1\venv\Scripts\python.exe T:/2kurs2sem/OperRes/lab1/main.py
1.8076923076923086 - максимальная сумма на позиции: 107


Process finished with exit code 0
```

Вычисление вручную:

$$\begin{cases} 6x_1 - 2x_2 + 2x_3 + 5x_5 = 6 \\ x_1 - 2x_2 + 3x_3 + 4x_4 - x_5 = 2 \\ 4x_1 - 7x_2 + 2x_3 + x_4 + 3x_5 = 0 \\ 2x_1 - 4x_2 + 6x_3 + 8x_4 - 2x_5 = 4 \end{cases} \quad \left| \begin{array}{cccccc} 6 & -2 & 2 & 0 & 5 & 6 \\ 1 & -2 & 3 & 4 & -1 & 2 \\ 4 & -7 & 2 & 1 & 3 & 0 \\ 2 & -4 & 6 & 8 & -2 & 4 \end{array} \right|$$

Уравнения 1 и 2 поменяем местами.

$$\left| \begin{array}{cccccc} 1 & -2 & 3 & 4 & -1 & 2 \\ 6 & -2 & 2 & 0 & 5 & 6 \\ 4 & -7 & 2 & 1 & 3 & 0 \\ 2 & -4 & 6 & 8 & -2 & 4 \end{array} \right|$$

К уравнению 2 прибавляем уравнение 1, умноженное на -6.

$$\left| \begin{array}{cccccc} 1 & -2 & 3 & 4 & -1 & 2 \\ 0 & 10 & -16 & -24 & 11 & -6 \\ 4 & -7 & 2 & 1 & 3 & 0 \\ 2 & -4 & 6 & 8 & -2 & 4 \end{array} \right|$$

К уравнению 3 прибавляем уравнение 1, умноженное на -4.

1	-2	3	4	-1	2
0	10	-16	-24	11	-6
0	1	-10	-15	7	-8
2	-4	6	8	-2	4

К уравнению 4 прибавляем уравнение 1, умноженное на -2.

1	-2	3	4	-1	2
0	10	-16	-24	11	-6
0	1	-10	-15	7	-8
0	0	0	0	0	0

Уравнения 2 и 3 поменяем местами.

1	-2	3	4	-1	2
0	1	-10	-15	7	-8
0	10	-16	-24	11	-6

К уравнению 3 прибавляем уравнение 2, умноженное на -10.

1	-2	3	4	-1	2
0	1	-10	-15	7	-8
0	0	84	126	-59	74

Найдём базисные решения:

$x_3 = 37/42 - 3/2x_4 + 59/84x_5 = 0,8809 – 1,5x_4 + 0,7023x_5$

$x_2 = 17/21 + 1/42x_5 = 0,8095 + 0,0238x_5$

$x_1 = 41/42 + 1/2x_4 - 89/84x_5 = 0,9761 + 0,5x_4 - 1,0595x_5$

Вывод: В ходе лабораторной работы изучили метод Гаусса-Жордана, написав на основе метода алгоритм поиска всех базисных видов. Также познакомились с методом перебора для определения оптимального опорного плана. Сравнили рукописное решение с получившимся программно, результаты совпали.

GitHub: https://github.com/AnDreV133