

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"
профиль "Программное обеспечение средств
вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

Лабораторная работа №1 по дисциплине
Алгоритмы и вычислительные методы оптимизации

Вариант ____

Выполнили:

студенты гр.ИП-312

_____/Дорогин Н. С./
ФИО студента

_____/Грязин А. В./
ФИО студента

«18» февраля 2026 г.

Новосибирск 2026 г.

Задание

Написать программу, находящую решение системы линейных уравнений методом Жордана-Гаусса с выбором главного элемента в столбце.

Входные данные

На вход программе подаются коэффициенты системы линейных уравнений (считываются из файла в виде матрицы размера $m * (n+1)$):

[illegible]

Требования к программе

Программа должна:

- выводить промежуточные матрицы после каждого шага исключений и решение системы;
- работать для различных случаев решений системы: система имеет единственное решение, система имеет бесконечно много решений, система не имеет решения.

Результаты работы программы

```
≡ matrix_1.txt X
Лабы > Лаба2 > matrix > ≡ matrix_1.txt
1   3 2 5 4 | 3
2   1 -1 -1 -4 | -2
3   4 1 4 0 | 2
```

Скриншот 1. Матрица для первого теста

```
-----
Лабораторная работа №2
МЕТОД ЖОРДАНА-ГАУССА
-----

Введите имя файла с матрицей (или exit, чтобы выйти):
matrix_1
Исходная матрица:
   3   2   5   4 |   3
   1  -1  -1  -4 |  -2
   4   1   4   0 |   2
```

Скриншот 2. Тест 1. Вывод исходной матрицы

```
-----
ШАГ 1:
Меняем строки 2 и 0 местами:
   4   1   4   0 |   2
   1  -1  -1  -4 |  -2
   3   2   5   4 |   3
Главный элемент (0,0): 4
Делим 0 строку на 4:
   1  (1/4)   1   0 |  (1/2)
   1  -1  -1  -4 |  -2
   3   2   5   4 |   3
Зануляем элементы над и под 1 в столбце 0:
   1  (1/4)   1   0 |  (1/2)
   0  (-5/4)  -2  -4 | (-5/2)
   0  (5/4)   2   4 |  (3/2)
```

Скриншот 3. Тест 1. Шаг 1

```
-----
ШАГ 2:
Главный элемент (1,1): (-5/4)
Делим 1 строку на (-5/4):
      1      (1/4)      1      0 |      (1/2)
      0      1      (8/5)  (16/5) |      2
      0      (5/4)      2      4 |      (3/2)
Зануляем элементы над и под 1 в столбце 1:
      1      0      (3/5)  (-4/5) |      0
      0      1      (8/5)  (16/5) |      2
      0      0      0      0 |      -1
-----
```

Скриншот 4. Тест 1. Шаг 2

```
-----
ШАГ 3:
Главный элемент (2,2): 0

Нулевая левая часть при ненулевой правой!
Система не имеет решений!

Введите имя файла с матрицей (или exit, чтобы выйти):

```

Скриншот 5. Тест 1. Шаг 3

```
≡ matrix_2.txt X
Лабы > Лаба2 > matrix > ≡ matrix_2.txt
      1      4 -3 -2 1 | -2
      2      3 -1 -2 0 | 1
      3      2 1 -2 -1 | 4
```

Скриншот 6. Матрица для второго теста

```

Введите имя файла с матрицей (или exit, чтобы выйти):
matrix_2
Исходная матрица:
      4      -3      -2      1 |      -2
      3      -1      -2      0 |      1
      2       1      -2     -1 |      4
-----
ШАГ 1:
Главный элемент (0,0): 4
Делим 0 строку на 4:
      1  (-3/4)  (-1/2)  (1/4) |  (-1/2)
      3      -1      -2      0 |      1
      2       1      -2     -1 |      4
Зануляем элементы над и под 1 в столбце 0:
      1  (-3/4)  (-1/2)  (1/4) |  (-1/2)
      0  (5/4)   (-1/2)  (-3/4) |  (5/2)
      0  (5/2)           -1  (-3/2) |      5
-----

```

Скриншот 7. Тест 2. Вывод исходной матрицы и шаг 1

```

-----
ШАГ 2:
Меняем строки 2 и 1 местами:
      1  (-3/4)  (-1/2)  (1/4) |  (-1/2)
      0  (5/2)           -1  (-3/2) |      5
      0  (5/4)   (-1/2)  (-3/4) |  (5/2)
Главный элемент (1,1): (5/2)
Делим 1 строку на (5/2):
      1  (-3/4)  (-1/2)  (1/4) |  (-1/2)
      0       1  (-2/5)  (-3/5) |      2
      0  (5/4)   (-1/2)  (-3/4) |  (5/2)
Зануляем элементы над и под 1 в столбце 1:
      1       0  (-4/5)  (-1/5) |      1
      0       1  (-2/5)  (-3/5) |      2
      0       0       0       0 |      0
-----

```

Скриншот 8. Тест 2. Шаг 2

ШАГ 3:

Главный элемент (2,2): 0

Столбец 2 не входит в базис.

Система имеет бесконечное множество решений.

Общий вид:

$$x_1 = 1 + 0 + (4/5)*x_3 + (1/5)*x_4$$

$$x_2 = 2 + 0 + (2/5)*x_3 + (3/5)*x_4$$

Скриншот 9. Тест 2. Шаг 3

```
≡ matrix_3.txt X
Лабы > Лаба2 > matrix > ≡ matrix_3.txt
 1  1 4 0 -1 | 5
 2  2 -3 1 1 | 3
 3  1 0 2 -1 | 3
 4  0 2 -3 2 | 3
```

Скриншот 10. Матрица для третьего теста

```
Введите имя файла с матрицей (или exit, чтобы выйти):
matrix_3
Исходная матрица:
 1  4  0 -1 | 5
 2 -3  1  1 | 3
 1  0  2 -1 | 3
 0  2 -3  2 | 3
```

Скриншот 11. Тест 3. Вывод исходной матрицы

ШАГ 1:

Меняем строки 1 и 0 местами:

$$\begin{array}{cccc|c} 2 & -3 & 1 & 1 & 3 \\ 1 & 4 & 0 & -1 & 5 \\ 1 & 0 & 2 & -1 & 3 \\ 0 & 2 & -3 & 2 & 3 \end{array}$$

Главный элемент (0,0): 2

Делим 0 строку на 2:

$$\begin{array}{cccc|c} 1 & (-3/2) & (1/2) & (1/2) & (3/2) \\ 1 & 4 & 0 & -1 & 5 \\ 1 & 0 & 2 & -1 & 3 \\ 0 & 2 & -3 & 2 & 3 \end{array}$$

Зануляем элементы над и под 1 в столбце 0:

$$\begin{array}{cccc|c} 1 & (-3/2) & (1/2) & (1/2) & (3/2) \\ 0 & (11/2) & (-1/2) & (-3/2) & (7/2) \\ 0 & (3/2) & (3/2) & (-3/2) & (3/2) \\ 0 & 2 & -3 & 2 & 3 \end{array}$$

Скриншот 12. Тест 3. Шаг 1

ШАГ 2:

Главный элемент (1,1): (11/2)

Делим 1 строку на (11/2):

$$\begin{array}{cccc|c} 1 & (-3/2) & (1/2) & (1/2) & (3/2) \\ 0 & 1 & (-1/11) & (-3/11) & (7/11) \\ 0 & (3/2) & (3/2) & (-3/2) & (3/2) \\ 0 & 2 & -3 & 2 & 3 \end{array}$$

Зануляем элементы над и под 1 в столбце 1:

$$\begin{array}{cccc|c} 1 & 0 & (4/11) & (1/11) & (27/11) \\ 0 & 1 & (-1/11) & (-3/11) & (7/11) \\ 0 & 0 & (18/11) & (-12/11) & (6/11) \\ 0 & 0 & (-31/11) & (28/11) & (19/11) \end{array}$$

Скриншот 13. Тест 3. Шаг 2

ШАГ 3:

Меняем строки 3 и 2 местами:

$$\begin{array}{cccc|c} 1 & 0 & (4/11) & (1/11) & (27/11) \\ 0 & 1 & (-1/11) & (-3/11) & (7/11) \\ 0 & 0 & (-31/11) & (28/11) & (19/11) \\ 0 & 0 & (18/11) & (-12/11) & (6/11) \end{array}$$

Главный элемент (2,2): $(-31/11)$

Делим 2 строку на $(-31/11)$:

$$\begin{array}{cccc|c} 1 & 0 & (4/11) & (1/11) & (27/11) \\ 0 & 1 & (-1/11) & (-3/11) & (7/11) \\ 0 & 0 & 1 & (-28/31) & (-19/31) \\ 0 & 0 & (18/11) & (-12/11) & (6/11) \end{array}$$

Зануляем элементы над и под 1 в столбце 2:

$$\begin{array}{cccc|c} 1 & 0 & 0 & (13/31) & (83/31) \\ 0 & 1 & 0 & (-11/31) & (18/31) \\ 0 & 0 & 1 & (-28/31) & (-19/31) \\ 0 & 0 & 0 & (12/31) & (48/31) \end{array}$$

Скриншот 14. Тест 3. Шаг 3

ШАГ 4:

Главный элемент (3,3): (12/31)

Делим 3 строку на (12/31):

1	0	0	(13/31)		(83/31)
0	1	0	(-11/31)		(18/31)
0	0	1	(-28/31)		(-19/31)
0	0	0	1		4

Зануляем элементы над и под 1 в столбце 3:

1	0	0	0		1
0	1	0	0		2
0	0	1	0		3
0	0	0	1		4

Система имеет единственное решение:

x1 = 1

x2 = 2

x3 = 3

x4 = 4

Введите имя файла с матрицей (или exit, чтобы выйти):

exit

Завершение работы...

PS C:\Users\Danik\Desktop\Учёба\ABMO\sibsutis_09.03.01_IP_3_course_AVMO>

Скриншот 15. Тест 3. Шаг 4. Выход из программы.

Исходный код программы

```
import math

class Fract:
    def __init__(self, upper, lower=1):
        if lower == 0:
            raise ValueError("Знаменатель не может быть 0!")
        if lower < 0:
            upper = -upper
            lower = -lower
        self.upper = upper
        self.lower = lower
        self.reduce()

    def __str__(self):
        if self.upper % self.lower == 0:
            return str(self.upper // self.lower)
        else:
            return f'({self.upper}/{self.lower})'

    def reduce(self):
        gcd = math.gcd(self.upper, self.lower)
        self.upper //= gcd
        self.lower //= gcd
```

```

def __mul__(self, x):
    if isinstance(x, Fract):
        return Fract(self.upper * x.upper, self.lower * x.lower)
    elif isinstance(x, (int, float)):
        return Fract(self.upper * x, self.lower)
    else:
        raise TypeError("Ошибка умножения дроби!")

def __truediv__(self, x):
    if isinstance(x, Fract):
        return Fract(self.upper * x.lower, self.lower * x.upper)
    elif isinstance(x, (int, float)):
        return Fract(self.upper, self.lower * x)
    else:
        raise TypeError("Ошибка деления дроби!")

def __add__(self, x):
    if isinstance(x, Fract):
        new_upper = self.upper * x.lower + x.upper * self.lower
        new_lower = self.lower * x.lower
        return Fract(new_upper, new_lower)
    elif isinstance(x, (int, float)):
        return Fract(self.upper + x * self.lower, self.lower)
    else:
        raise TypeError("Ошибка сложения дробей!")

def __sub__(self, x):
    if isinstance(x, Fract):
        new_upper = self.upper * x.lower - x.upper * self.lower
        new_lower = self.lower * x.lower
        return Fract(new_upper, new_lower)
    elif isinstance(x, (int, float)):
        return Fract(self.upper - x * self.lower, self.lower)
    else:
        raise TypeError("Ошибка вычитания из дроби!")

def __eq__(self, x):
    if isinstance(x, Fract):
        return self.upper * x.lower == x.upper * self.lower
    elif isinstance(x, (int, float)):
        return self.upper == x * self.lower
    return False

def __ne__(self, x):
    return not self.__eq__(x)

def __gt__(self, x):
    if isinstance(x, Fract):
        return self.upper * x.lower > x.upper * self.lower
    elif isinstance(x, (int, float)):
        return self.upper > x * self.lower
    raise TypeError("Ошибка сравнения!")

def __ge__(self, x):
    if isinstance(x, Fract):
        return self.upper * x.lower >= x.upper * self.lower
    elif isinstance(x, (int, float)):
        return self.upper >= x * self.lower
    raise TypeError("Ошибка сравнения!")

def __lt__(self, x):
    if isinstance(x, Fract):
        return self.upper * x.lower < x.upper * self.lower
    elif isinstance(x, (int, float)):

```

```

        return self.upper < x * self.lower
        raise TypeError("Ошибка сравнения!")

def __le__(self, x):
    if isinstance(x, Fract):
        return self.upper * x.lower <= x.upper * self.lower
    elif isinstance(x, (int, float)):
        return self.upper <= x * self.lower
    raise TypeError("Ошибка сравнения!")

def __abs__(self):
    return Fract(abs(self.upper), self.lower)

def parse_matrix(matrix_str):
    rows = matrix_str.strip().split('\n')
    left_matrix = []
    right_vector = []
    for row in rows:
        left, right = row.strip().split('|')
        left_parts = left.strip().split()
        left_part = []
        for num in left_parts:
            if '/' in num:
                up, low = map(int, num.split('/'))
                left_part.append(Fract(up, low))
            else:
                left_part.append(Fract(int(num)))
        right_str = right.strip()
        if '/' in right_str:
            up, low = map(int, right_str.split('/'))
            right_part = Fract(up, low)
        else:
            right_part = Fract(int(right_str))

        left_matrix.append(left_part)
        right_vector.append(right_part)

    return left_matrix, right_vector

def print_matrix(left_matrix, right_vector):
    for i in range(len(left_matrix)):
        row_str = ""
        for j in range(len(left_matrix[i])):
            row_str += f"{str(left_matrix[i][j]):>8} "
        row_str += f"| {str(right_vector[i]):>8}"
        print(row_str)

def Jordan_Gauss(left, right):
    n = len(left)
    m = len(left[0])

    cur_row = 0
    j_case = 0

    for col in range(min(n, m)):
        print('-' * 50)
        print(f'ШАГ {col + 1}:')
        main_row = cur_row
        for i in range(col, n):
            if abs(left[i][col]) > abs(left[main_row][col]):
                main_row = i

        if main_row != cur_row:
            left[cur_row], left[main_row] = left[main_row], left[cur_row]

```

```

        right[cur_row], right[main_row] = right[main_row],
right[cur_row]
        print(f'Меняем строки {main_row} и {cur_row} местами:')
        print_matrix(left, right)
        main_row = cur_row
        main = left[main_row][col]
        print(f'Главный элемент ({main_row},{col}): {main}')

    if(main == 0):
        zeros = True
        for c in range(m):
            if left[main_row][c] != 0:
                zeros = False
        if(zeros and right[main_row] != 0):
            j_case = 3
        else:
            print(f'Столбец {col} не входит в базис.')
            j_case = 1
        continue

    for j in range(col, m):
        left[main_row][j] = left[main_row][j] / main
        right[main_row] = right[main_row] / main
        print(f'Делим {main_row} строку на {main}:')
        print_matrix(left, right)

    for i in range(n):
        if i != main_row:
            d = left[i][col]
            if d.upper != 0:
                for j in range(main_row, m):
                    left[i][j] = left[i][j] - (left[main_row][j] * d)
                    right[i] = right[i] - (right[main_row] * d)
        print(f'Зануляем элементы над и под 1 в столбце {col}:')
        print_matrix(left, right)
        print()
        cur_row += 1

    if(j_case == 0):
        print("Система имеет единственное решение:")
        for i in range(len(left)):
            print(f'x{i+1} = {right[i]}')
    elif (j_case == 1):
        print("Система имеет бесконечное множество решений.")
        print("Общий вид:")
        row = 0
        col = 0
        while(col < m and row < n):
            o_zeros = True
            for z in range(m):
                if left[row][z] != 0:
                    o_zeros = False
            if(o_zeros and right[row] == 0):
                row += 1
                continue

            dec = f'x{col+1} = '
            if (right[row] != 0):
                dec += str(right[row])
            else:
                dec += '0'

            for s in range(m):
                if s != col:

```

```

        k = left[row][s] * (-1)
        if k > 0:
            if abs(k) == 1:
                dec += f' + x{s+1}'
            else:
                dec += f' + {k}*x{s+1}'
        elif k < 0:
            if abs(k) == 1:
                dec += f' - x{s+1}'
            else:
                dec += f' - {abs(k)}*x{s+1}'
        else:
            dec += f' + 0'

    print(dec)
    col += 1
    row += 1
else:
    print("\nНулевая левая часть при ненулевой правой!")
    print("Система не имеет решений!")

if __name__ == "__main__":
    path = ".\Лабы\Лаба2\matrix"
    print(80 * "-")
    print( (20 * " ") + "Лабораторная работа №2" + (20 * " ") )
    print( (20 * " ") + "МЕТОД ЖОРДАНА-ГАУССА" + (20 * " ") )
    print(80 * "-")
    while(1):
        print("\nВведите имя файла с матрицей (или exit, чтобы выйти): ")
        name = input()

        if (name == "exit"):
            print("Завершение работы...")
            break

        if not name.__contains__(".txt"):
            name = name + ".txt"
        f_matrix = open(f'{path}\\{name}', "r")
        s_matrix = f_matrix.read()
        f_matrix.close()

        m_left, m_right = parse_matrix(s_matrix)

        print("Исходная матрица:")
        print_matrix(m_left, m_right)

        Jordan_Gauss(m_left, m_right)

```