МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Качество и метрология программного обеспечения»
Тема: «Анализ структурной сложности графовых моделей программ»

Студент гр. 8304 _____ Мухин А. М.

Преподаватель _____ Кирьянчиков В. А.

Санкт-Петербург
2022

Задание.

Выполнить оценивание структурной сложности двух программ с помощью критериев:

- Минимального покрытия вершин и дуг графа управления;
- Выбора маршрутов на основе цикломатического числа графа.

Варианты программ:

- Программа с заданной структурой управляющего графа, выбираемой из файла zadan_struct.doc в соответствии с номером в списке группы (рисунок 1).
- Программа из 1-ой лабораторной работы (управляющий граф составить самостоятельно).

Оцениваемые характеристики структурной сложности:

- Число учитываемых маршрутов проверки программы для заданного критерия;
- Цикломатическое число;
- Суммарное число ветвлений по всем маршрутам – оценка структурной сложности.

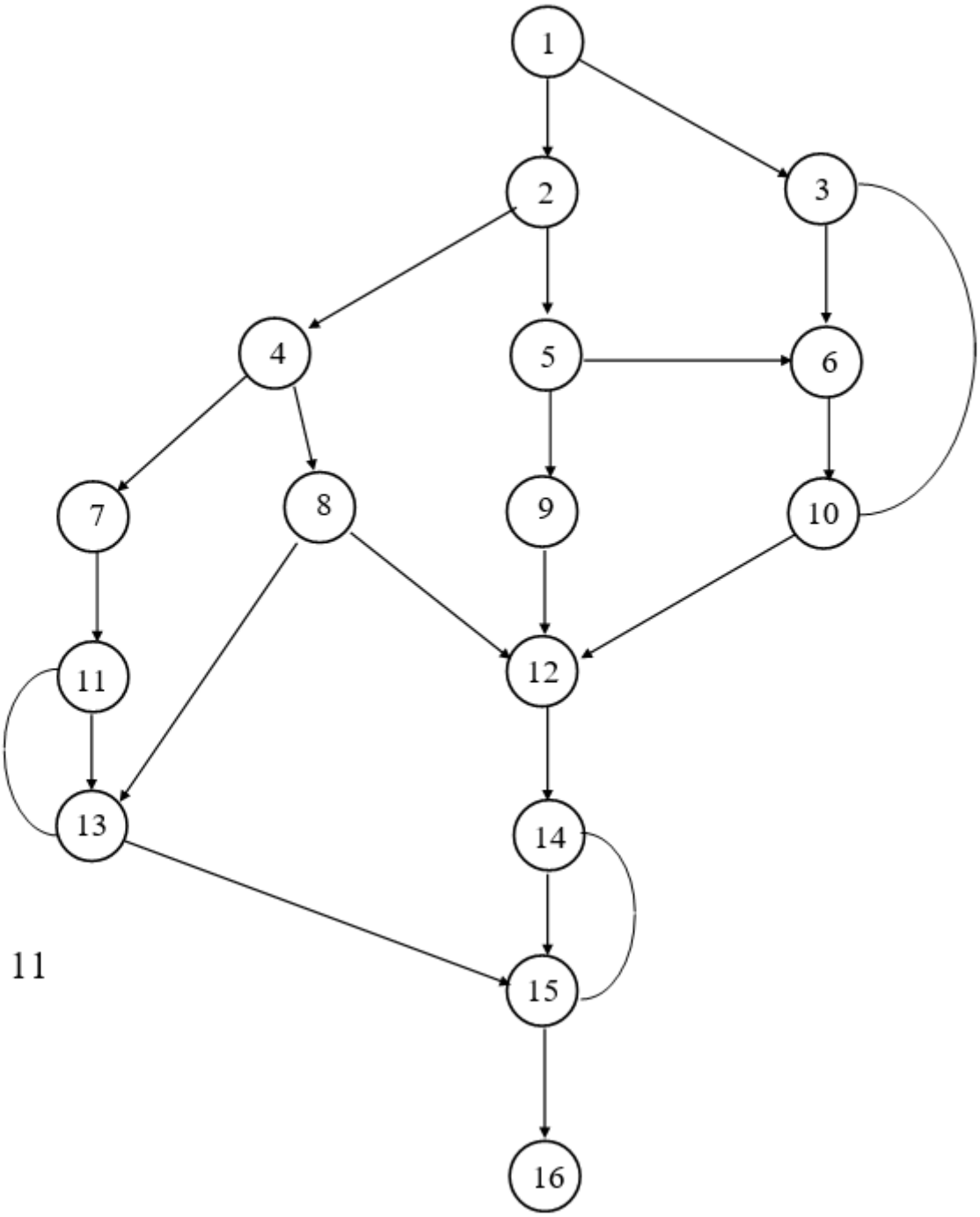

Рисунок 1 – Граф из файла zadan_struct.doc

Оценка структурной сложности программы из файла zadan_struct.doc.

Оценивание структурной сложности с помощью критерия минимального покрытия дуг графа **вручную**.

Ветвления: 1, 2, 4, 5, 8, 10, 13, 15. Всего ветвлений – 8.

Минимальный набор путей (жирным выделены ветвления):

- **1**-**2**-**4**-7-11-**13**-11-**13**-**15**-14-**15**-16 (7 ветвлений);
- **1**-**2**-**4**-**8**-**13**-**15**-16 (6 ветвлений);
- **1**-**2**-**5**-9-12-14-**15**-16 (4 ветвления);
- **1**-**2**-**5**-6-**10**-12-14-**15**-16 (5 ветвлений);
- **1**-**2**-**4**-**8**-12-14-**15**-16 (5 ветвлений);
- **1**-3-6-**10**-3-6-**10**-12-14-**15**-16 (4 ветвления).

Итого сложность равна 7 + 6 + 4 + 5 + 5 + 4 = 31.

Оценивание структурной сложности с помощью критерия на основе цикломатического числа **вручную**.

Число вершин в графе – 16, число ребер – 23. Для того, чтобы граф стал связным (из каждой вершины существовал путь в любую другую) достаточно добавить одно ребро (16-1). Таким образом, цикломатическое число графа равно $23-16+2*1=9$. Значит необходимо рассмотреть 9 линейно-независимых циклов и путей.

- 3-6-**10**-3 (1 ветвление);
- 11-**13**-11 (1 ветвление);
- 14-**15**-14 (1 ветвление);
- **1**-**2**-**4**-**8**-**13**-**15**-16 (6 ветвлений);
- **1**-**2**-**4**-7-11-**13**-**15**-16 (5 ветвлений);
- **1**-**2**-**4**-**8**-12-14-**15**-16 (5 ветвлений);
- **1**-**2**-**5**-6-**10**-12-14-**15**-16 (5 ветвлений);
- **1**-3-6-**10**-12-14-**15**-16 (3 ветвления);
- **1**-**2**-**5**-9-12-14-**15**-16 (4 ветвления).

Итого сложность равна 1 + 1 + 1 + 6 + 5 + 5 + 5 + 3 + 4 = 31.

Оценивание структурной сложности с помощью критерия минимального покрытия дуг графа **программно** представлено на рисунке 2. Структура графа для программы представлена в приложении Б.

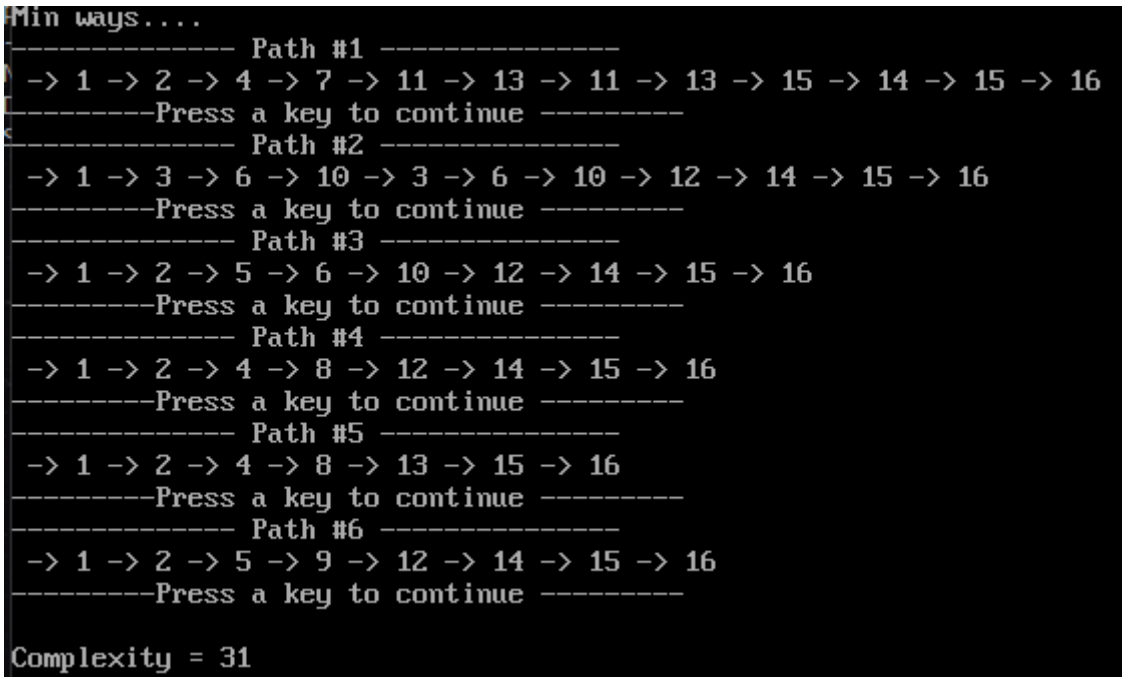

Рисунок 2 - Минимальное покрытие дуг

Оценивание структурной сложности с помощью критерия на основе цикломатического числа **программно** представлено на рисунке 3.

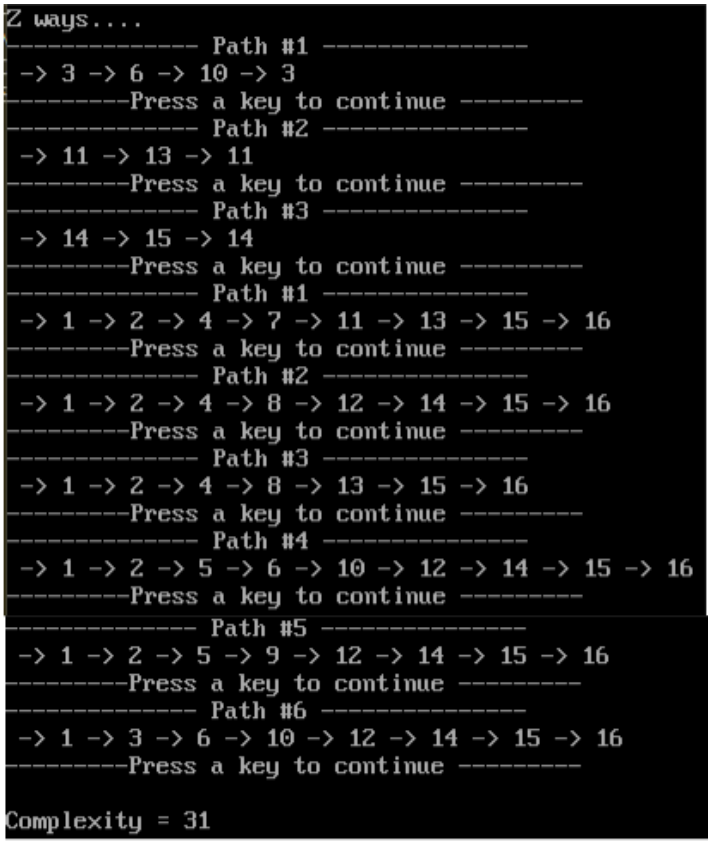

Рисунок 3 – Цикломатическое число

Оценка структурной сложности программы из 1-ой лабораторной.

Код программы из первой лабораторной представлен в приложении А. Граф программы представлен на рисунке 4.

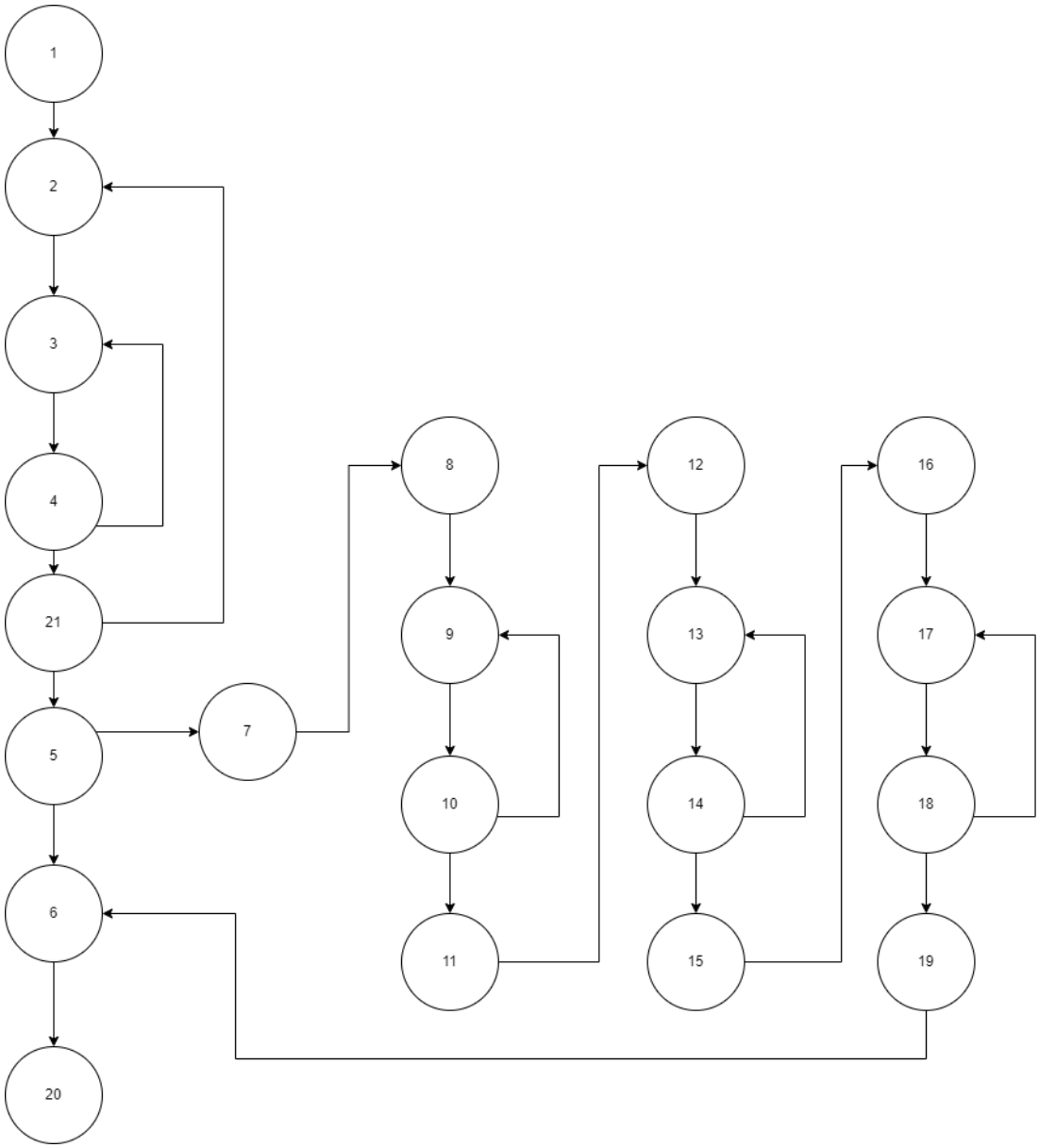

Рисунок 4 - Граф программы на языке СИ

Ключевые узлы и переходы графа:

- 2-3-4-3-4-21-2 – FOR внутри solve;
- 5 – Проверка определителя матрицы;
- 9-10-9 – FOR внутри setup при первом вызове функции;
- 13-14-13 – FOR внутри setup при втором вызове функции;
- 17-18-17 – FOR внутри setup при третьем вызове функции.

Оценивание структурной сложности с помощью критерия минимального покрытия дуг графа **вручную**.

Ветвления: 4, 5, 10, 14, 18, 21 Всего ветвлений – 5.

Минимальный набор путей (жирным выделены ветвления):

- 1-2-3-**4**-**21**-**5**-7-8-9-**10**-9-**10**-11-12-13-**14**-13-**14**-15-16-17-**18**-17-**18**-19-6-20 (9 ветвлений);
- 1-2-3-**4**-**21**-2-3-**4**-3-**4**-**21**-**5**-6-20 (6 ветвлений).

Итого сложность равна 9 + 6 = 15.

Оценивание структурной сложности с помощью критерия на основе цикломатического числа **вручную**.

Число вершин в графе – 21, число ребер – 26. Для того, чтобы граф стал связным (из каждой вершины существовал путь в любую другую) достаточно добавить одно ребро (20-1). Таким образом, цикломатическое число графа равно $26 - 21 + 2 * 1 = 7$. Значит необходимо рассмотреть 6 линейно-независимых циклов и путей.

- 3-**4**-3 (1 ветвление);
- 2-3-**4**-**21**-2 (2 ветвления);
- 9-**10**-9 (1 ветвление);
- 13-**14**-13 (1 ветвление);
- 17-**18**-17 (1 ветвление);
- 1-2-3-**4**-**21**-**5**-7-8-9-**10**-11-12-13-**14**-15-16-17-**18**-19-6-20 (6 ветвлений);
- 1-2-3-**4**-**21**-**5**-6-20 (3 ветвления).

Итого сложность равна 1 + 2 + 1 + 1 + 1 + 6 + 3 = 15.

Оценивание структурной сложности с помощью критерия минимального покрытия дуг графа **программно** представлено на рисунке 5. Структура графа для программы представлена в приложении В.

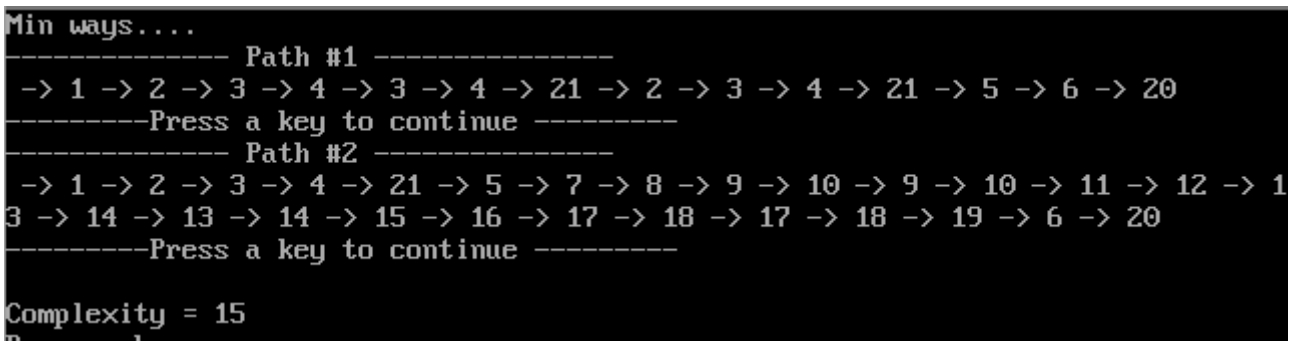

Рисунок 5 - Минимальное покрытие дуг

Оценивание структурной сложности с помощью критерия на основе цикломатического числа **программно** представлено на рисунке 6.

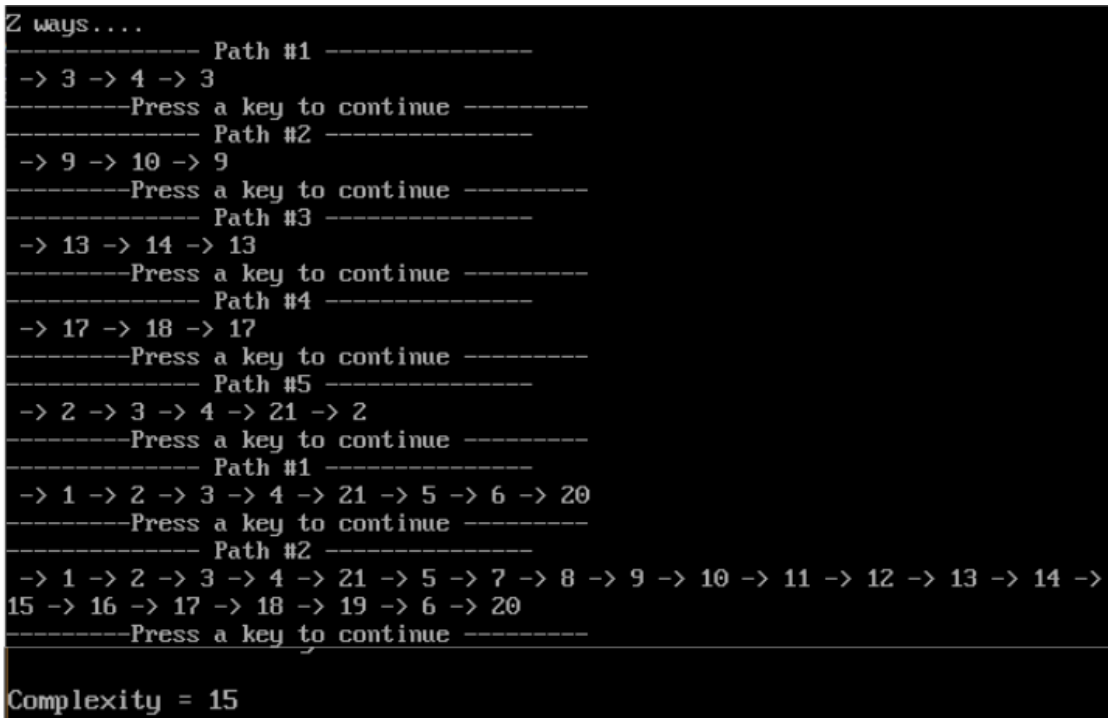

Рисунок 6 - Цикломатическое число

Выводы.

В результате выполнения данной лабораторной работы были изучены критерии оценивания структурной сложности программ. Была проведена оценка структурной сложности двух программ: соответствующая варианту и из первой лабораторной работы.

По результатам оценки можно сделать заключение, что программный и ручной способы совпадают с точностью до порядка путей.

ПРИЛОЖЕНИЕ А. ПРОГРАММА НА ЯЗЫКЕ СИ

```
#include <stdio.h>
#include <stdlib.h>

#define RMAX 3
#define CMAX 3


void get_data(double a[RMAX][CMAX], double y[CMAX]) {
    a[0][0] = 1;
    a[0][1] = -43;
    a[0][2] = 19;
    y[0] = 81;
    a[1][0] = 145;
    a[1][1] = -134;
    a[1][2] = 99;
    y[1] = 12;
    a[2][0] = 325;
    a[2][1] = 991;
    a[2][2] = -199;
    y[2] = 213;
}


double deter(double a[RMAX][CMAX]) {
    return(a[0][0] * (a[1][1] * a[2][2] - a[2][1] * a[1][2])
         - a[0][1] * (a[1][0] * a[2][2] - a[2][0] * a[1][2])
         + a[0][2] * (a[1][0] * a[2][1] - a[2][0] * a[1][1]));
}


void setup(double a[RMAX][CMAX], double b[RMAX][CMAX], double
y[CMAX], double coef[CMAX], int j, double det) {
    int i;
    for (i = 0; i < RMAX; i++) {
         b[i][j] = y[i];
         if (j > 0) {
              b[i][j - 1] = a[i][j - 1];
         }
    }
    coef[j] = deter(b) / det;
}


void solve(double a[RMAX][CMAX], double y[CMAX], double
coef[CMAX]) {
    double b[RMAX][CMAX];
    int i, j;
    double det;

    for (i = 0; i < RMAX; i++) {
         for (j = 0; j < CMAX; j++) {
```

```
                b[i][j] = a[i][j];
            }
        }

        det = deter(b);
        if (det == 0) {
            return;
        }
        else {
            setup(a, b, y, coef, 0, det);
            setup(a, b, y, coef, 1, det);
            setup(a, b, y, coef, 2, det);
        }
    }


    int main() {
        double a[RMAX][CMAX];
        double y[CMAX];
        double coef[CMAX];

        get_data(a, y);
        solve(a, y, coef);

        return 0;
    }
```

ПРИЛОЖЕНИЕ Б. СТРУКТУРА ГРАФА ИЗ ФАЙЛА

```
Nodes{1, 2, 3,4,5,6,7,8,9,10,11,12,13,14,15,16}

Top{1}
Last{16}

Arcs{
arc(1,2);
arc(1,3);
arc(2,4);
arc(2,5);
arc(3,6);
arc(4,7);
arc(4,8);
arc(5,6);
arc(5,9);
arc(6,10);
arc(7,11);
arc(8,12);
arc(8,13);
arc(9,12);
arc(10,12);
arc(10,3);
arc(11,13);
arc(12,14);
arc(13,11);
arc(13,15);
arc(14,15);
arc(15,14);
arc(15,16);
}
```

ПРИЛОЖЕНИЕ В. СТРУКТУРА ГРАФА НА ОСНОВЕ ПРОГРАММЫ

```
Nodes{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21}

Top{1}
Last{20}

Arcs{
arc(1,2);
arc(2,3);
arc(3,4);
arc(4,3);
arc(4,21);
arc(21,5);
arc(21,2);
arc(5,6);
arc(5,7);
arc(6,20);
arc(7,8);
arc(8,9);
arc(9,10);
arc(10,9);
arc(10,11);
arc(11,12);
arc(12,13);
arc(13,14);
arc(14,13);
arc(14,15);
arc(15,16);
arc(16,17);
arc(17,18);
arc(18,17);
arc(18,19);
arc(19,6);
}
```