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 _____ Чешуин Д. И.

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.

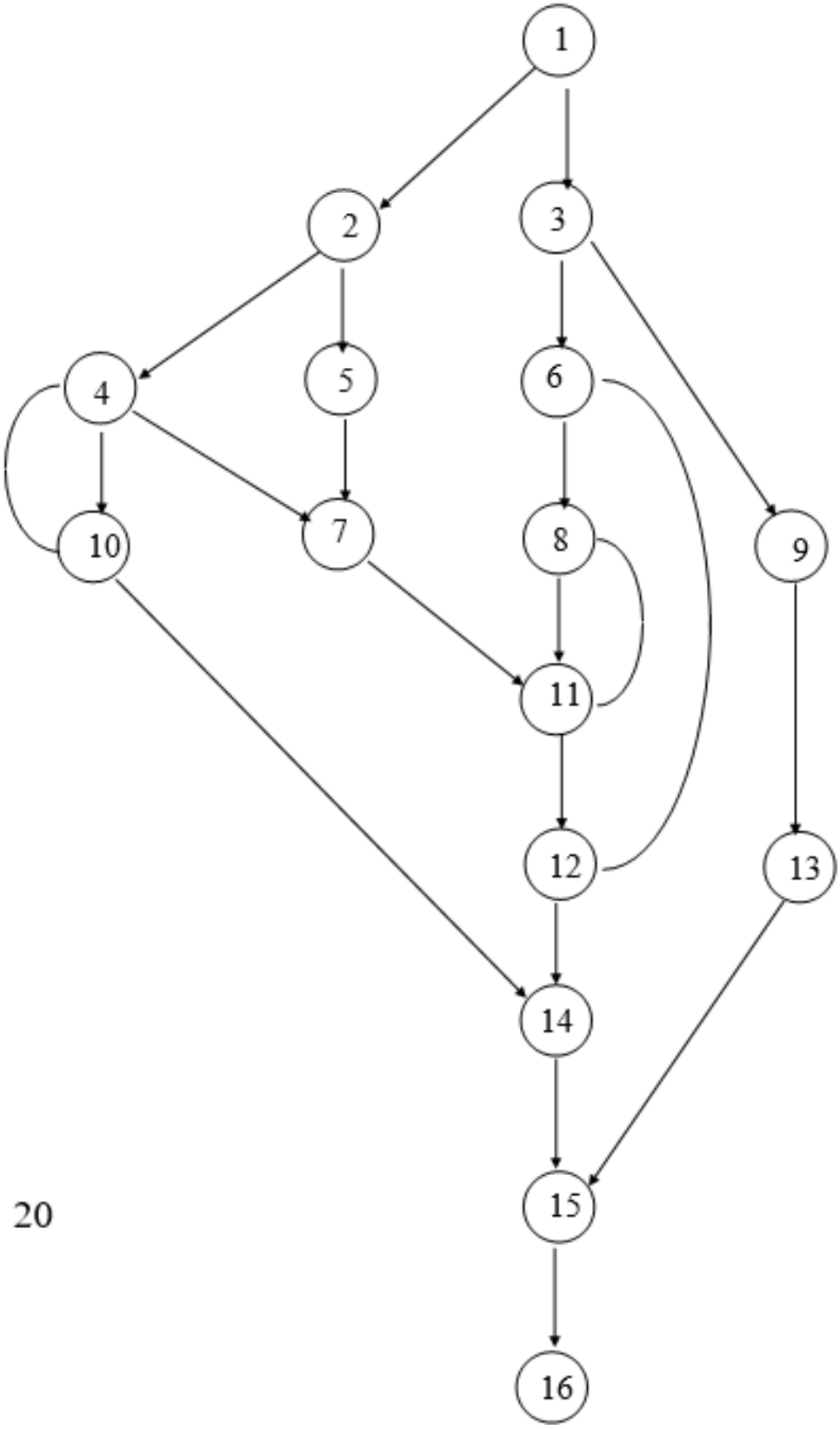

Рисунок 1 – Граф из файла zadan_struct.doc

Оценка структурной сложности программы из файла.

Было выполнено оценивание структурной сложности с помощью критерия минимального покрытия дуг графа вручную.

Ветвления в графе (номера вершин): 1, 2, 3, 4, 10, 11, 12. Всего ветвлений – 7.

Минимальный набор путей (жирным выделены ветвления):

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

Итого сложность равна $6 + 8 + 4 + 4 + 2 = 24$.

Было проведено оценивание структурной сложности с помощью критерия на основе цикломатического числа вручную.

Число вершин в графе – 16, число ребер – 22. 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 $22 - 16 + 2 * 1 = 8$. Значит необходимо рассмотреть 8 линейно-независимых циклов и путей.

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

Итого сложность равна $1 + 1 + 2 + 4 + 5 + 4 + 2 + 4 = 23$.

Оценивание структурной сложности с помощью критерия минимального покрытия дуг графа, а также с помощью критерия на основе цикломатического числа программно не удалось, программа вывела ошибку и затем зависла. Результат представлен на рисунке 2. Структура графа для программы представлена в приложении Б.

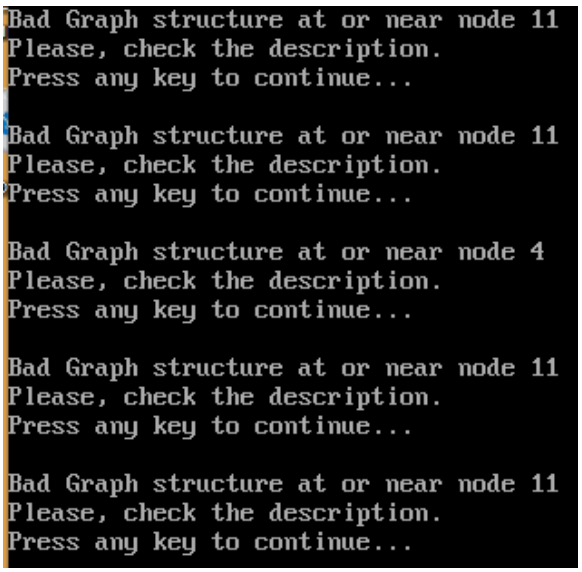

Рисунок 2 – Результат работы программы

Оценка структурной сложности программы из 1-ой лабораторной.

По коду программы из 1-ой лабораторной работы был составлен граф управления программы. Исходный код представлен в приложении А. Граф программы представлен на рисунке 3.

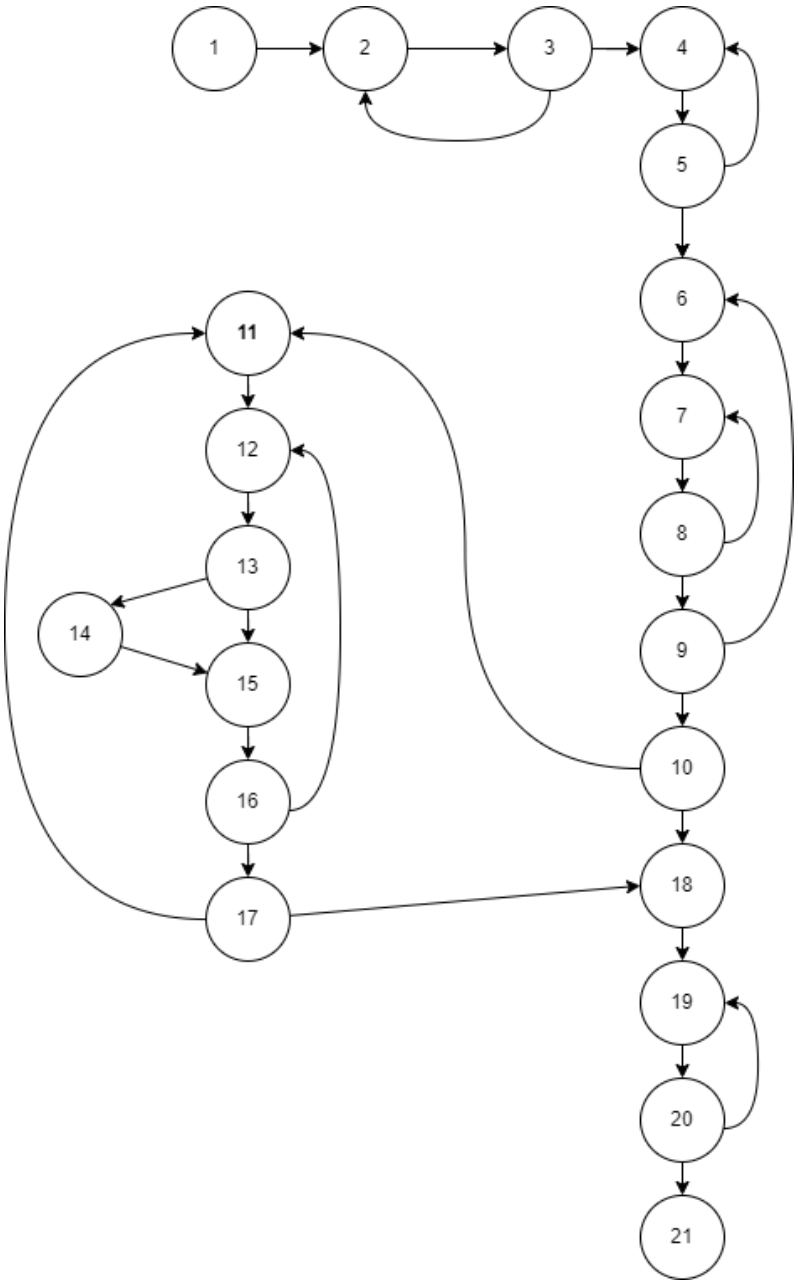

Рисунок 3 - Граф управления программы

Ключевые узлы и дуги графа:

- 2-3-2 – Генерация массива данных;
- 4-5-4 – Первый цикл FOR в linfit;
- 7-8-7 – Вложенный цикл FOR в solve;
- 6-7-8-9-6 – Внешний цикл FOR в solve;
- 19-20 – Последний цикл FOR в linfit;
- 10 – Проверка определителя на 0;
- 13 – Проверка внутри setup на $j > 0$;

- 17 – Вызов нескольких setup подряд;
- 12-13-15-16-12 - Цикл FOR в setup.

Было проведено оценивание структурной сложности с помощью критерия минимального покрытия дуг графа вручную.

Ветвления: 3, 5, 8, 9, 10, 13, 16, 17, 20. Всего ветвлений – 9.

Минимальный набор путей (жирным выделены ветвления):

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

Итого сложность равна $20 + 6 = 26$.

Было проведено оценивание структурной сложности с помощью критерия на основе цикломатического числа вручную.

Число вершин в графе – 21, число ребер – 29. Для того, чтобы граф стал связным (из каждой вершины существовал путь в любую другую) достаточно добавить одно ребро (21-1). Таким образом, цикломатическое число графа равно $29 - 21 + 2 * 1 = 9$. Значит необходимо рассмотреть 9 линейно-независимых циклов и путей.

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

Итого сложность равна $1 + 1 + 1 + 2 + 1 + 2 + 3 + 6 + 9 + 9 = 35$.

Было выполнено оценивание структурной сложности с помощью критерия минимального покрытия дуг графа программно. Результат представлен на рисунке 4.

Было выполнено оценивание структурной сложности с помощью критерия на основе цикломатического числа программно. Результат представлен на рисунке 5. Структура графа для программы представлена в приложении В.

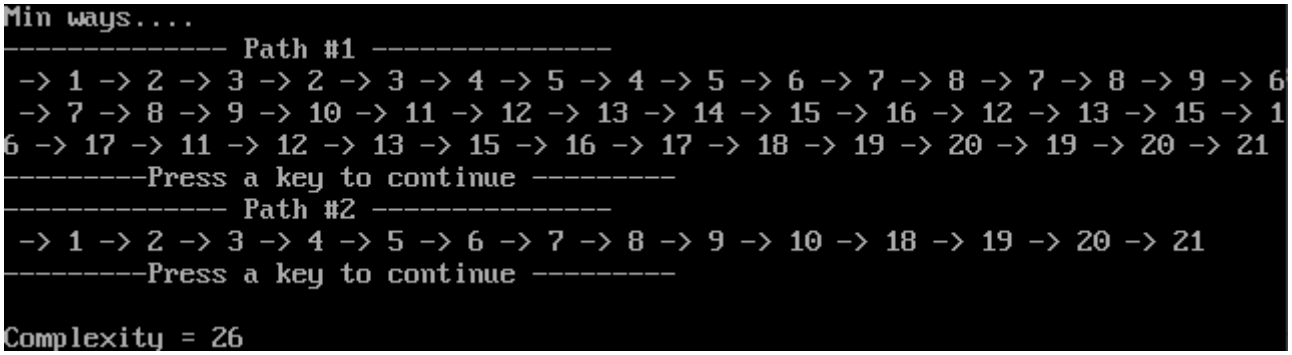

Рисунок 4 - Оценивание структурной сложности с помощью критерия минимального покрытия программно

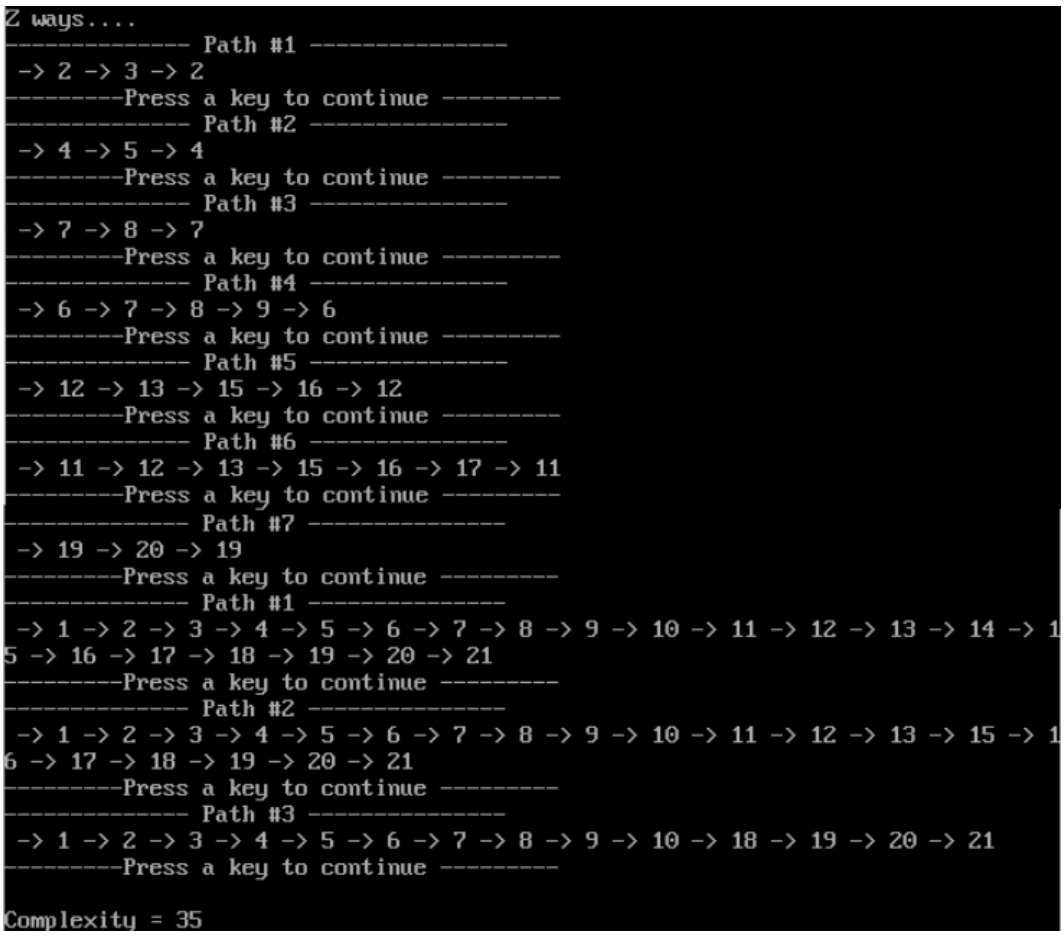

Рисунок 5 - Оценивание структурной сложности с помощью критерия на основе цикломатического числа программно

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 для графа на основе программы совпадают с точностью до порядка путей, ручной и программный способы для графа из файла сравнить не удалось, т.к. программа выдала ошибку.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ НА ЯЗЫКЕ СИ

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

#define MAXR 9
#define MAXC 3

void get_data(double x[MAXR], double y[MAXR]) {
    int i;
    for (i = 0; i < MAXR; i++) {
         x[i] = (double)i + 1;
    }
    y[0] = 2.07;
    y[1] = 8.6;
    y[2] = 14.42;
    y[3] = 15.8;
    y[4] = 18.92;
    y[5] = 17.96;
    y[6] = 12.98;
    y[7] = 6.45;
    y[8] = 0.27;
}

double deter(double a[MAXC][MAXC]) {
    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[MAXC][MAXC], double b[MAXC][MAXC], double
y[MAXC], double coef[MAXC], int j, double det) {
    int i;
    for (i = 0; i < MAXC; 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[MAXC][MAXC], double y[MAXC], double
coef[MAXC]) {
    double b[MAXC][MAXC];
    int i, j;
    double det;

    for (i = 0; i < MAXC; i++) {
```

```
            for (j = 0; j < MAXC; 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);
        }
    }

    void linfit(double x[MAXR], double y[MAXR], double y_calc[MAXR],
double coef[MAXC], double* corel_coef) {
        double sum_x, sum_y, sum_xy, sum_x2, sum_y2;
        double xi, yi, x2, sum_x3, sum_x4, sum_2y, srs;
        int i;
        double a[MAXC][MAXC];
        double g[MAXC];

        sum_x = 0.0;
        sum_y = 0.0;
        sum_xy = 0.0;
        sum_x2 = 0.0;
        sum_y2 = 0.0;
        sum_x3 = 0.0;
        sum_x4 = 0.0;
        sum_2y = 0.0;

        for (i = 0; i < MAXR; i++) {
            xi = x[i];
            yi = y[i];
            x2 = xi * xi;
            sum_x = sum_x + xi;
            sum_y = sum_y + yi;
            sum_xy = sum_xy + xi * yi;
            sum_x2 = sum_x2 + x2;
            sum_y2 = sum_y2 + yi * yi;
            sum_x3 = sum_x3 + xi * x2;
            sum_x4 = sum_x4 + x2 * x2;
            sum_2y = sum_2y + x2 * yi;
        }

        a[0][0] = MAXR;
        a[1][0] = sum_x;
        a[0][1] = sum_x;
        a[2][0] = sum_x2;
        a[0][2] = sum_x2;
        a[1][1] = sum_x2;
```

```
        a[2][1] = sum_x3;
        a[1][2] = sum_x3;
        a[2][2] = sum_x4;
        g[0] = sum_y;
        g[1] = sum_xy;
        g[2] = sum_2y;

        solve(a, g, coef);
        srs = 0.0;

        for (i = 0; i < MAXR; i++) {
             y_calc[i] = coef[0] + coef[1] * x[i] + coef[2] * pow(x[i],
2);
             srs += pow(y[i] - y_calc[i], 2);
        }
        *corel_coef = sqrt(1.0 - srs / (sum_y2 - pow(sum_y, 2) /
MAXR));
    }

    int main() {
        double x[MAXR];
        double y[MAXR];
        double y_calc[MAXR];
        double coef[MAXC];
        double corel_coef;

        get_data(x, y);
        linfit(x, y, y_calc, coef, &corel_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(3,9);
arc(4,7);
arc(4,10);
arc(5,7);
arc(6,8);
arc(7,11);
arc(8,11);
arc(9,13);
arc(10,4);
arc(10,14);
arc(11,8);
arc(11,12);
arc(12,6);
arc(12,14);
arc(13,15);
arc(14,15);
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{21}

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