

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

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

Студент гр. 7304

Моторин Е.В

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

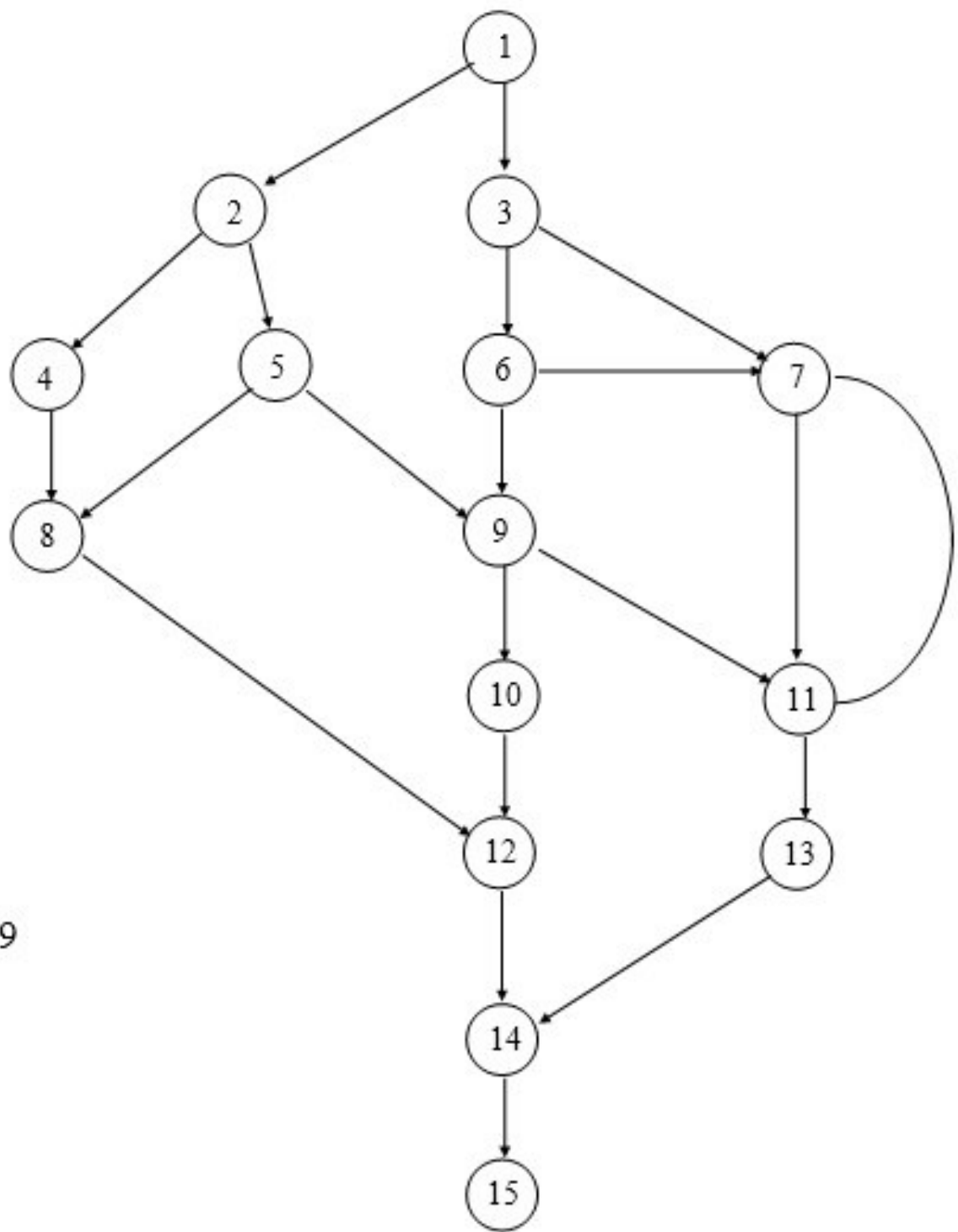
## Цель работы

Изучить применение метрик структурной сложности программ — критерия минимального покрытия и анализа базовых маршрутов.

## Задание

Выполнить оценивание структурной сложности двух программ с помощью критериев:

- Минимального покрытия дуг графа;
- Выбора маршрутов на основе цикломатического числа графа.
- Варианты программ:
- Программа с заданной преподавателем структурой управляющего графа;
- 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.



9

Рисунок 1 – Заданный граф

## Ход работы:

### 3.1. Программа из задания

#### 3.1.1. Критерий минимального покрытия

Маршруты для минимального покрытия (РАСЧИТЫВАЛОСЬ ДЛЯ ЗАДАННОГО ИЗНАЧАЛЬНО ГРАФА, Т.К. ДО ПРОГРАММНОГО ЗАПУСКА ПРОБЛЕМ С ГРАФОМ НЕ БЫЛО НАЙДЕНО):

1. 1, 2, 4, 8, 12, 14, 15
2. 1, 2, 5, 8, 12, 14, 15
3. 1, 2, 5, 9, 11, 7, 11, 13, 14, 15
4. 1, 3, 6, 9, 10, 12, 14, 15
5. 1, 3, 7, 11, 13, 14, 15
6. 1, 3, 6, 7, 11, 13, 14, 15

Количество маршрутов  $M = 6$

$$\text{Сложность: } S_2 = \sum_{i=1}^M \xi_i = 2 + 3 + 6 + 4 + 3 + 4 = 22$$

РУЧНОЙ ПЕРЕСЧЁТ ДЛЯ ИСПРАВЛЕННОГО ГРАФА (ЕСЛИ ИМЕННО ЭТО НУЖНО СДЕЛАТЬ)

1. 1, 2, 4, 8, 12, 14, 15
2. 1, 2, 5, 8, 12, 14, 15
3. 1, 2, 5, 9, 10, 12, 14, 15
4. 1, 2, 5, 9, 16, 13, 14, 15
5. 1, 3, 6, 7, 11, 7, 11, 16, 13, 14, 15
6. 1, 3, 7, 11, 16, 13, 14, 15
7. 1, 3, 6, 9, 10, 12, 14, 15

Количество маршрутов  $M = 7$

Сложность:  $S_2 = \sum_{i=1}^M \xi_i = 2 + 3 + 4 + 4 + 5 + 3 + 4 = 25$

### 3.1.2. Анализ базовых маршрутов

Число вершин графа  $N = 15$ , число дуг графа  $Y = 21$ , число связных компонент  $\Omega = 1$ . Цикломатическое число  $Z$ :

$$Z = Y - N + 2 * \Omega = 8.$$

Маршруты:

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

Сложность:  $S_2 = 1 + 2 + 3 + 4 + 6 + 4 + 3 + 4 = 26$

С ДОПОЛНИТЕЛЬНОЙ ВЕРШИНОЙ 16

Число вершин графа  $N = 16$ , число дуг графа  $Y = 22$ , число связных компонент  $\Omega = 1$ . Цикломатическое число  $Z$ :

$$Z = Y - N + 2 * \Omega = 8.$$

Маршруты:

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

Сложность:  $S_2=1+2+3+4+4+4+3+4=25$

### 3.1.3. Программный анализ

Граф задан в нотации приложенной программы. Файл с описанием — в приложении А.

С графом в приведенном описании возникла ошибка работы из-за ребра 9–11. На рис. 2 приведён результат работы программы для случая с исходным графом. Для решения была добавлена дополнительная вершина. Лог работы программы в приложении Б. Началом графа осталась вершина 1, концом – 15, независимо от того, что вершин в графе стало 16 для того, чтобы не перестраивать граф (вершина 16 была добавлена в середину). Во время ручных расчётов этой ошибки в вершинах 9-11 обнаружено не было и успешно были посчитаны маршруты минимального покрытия и базовые маршруты.

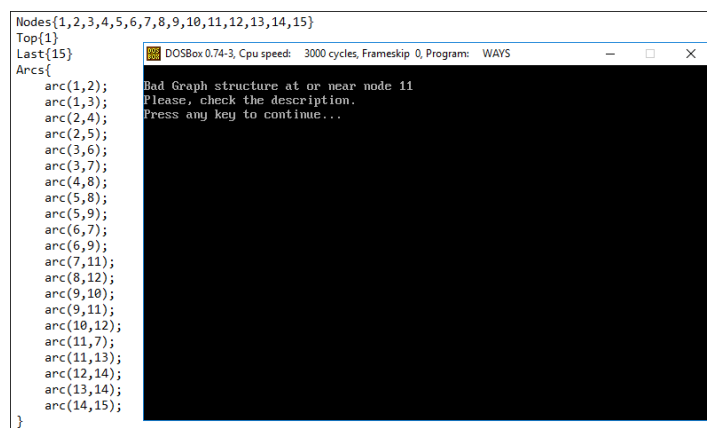
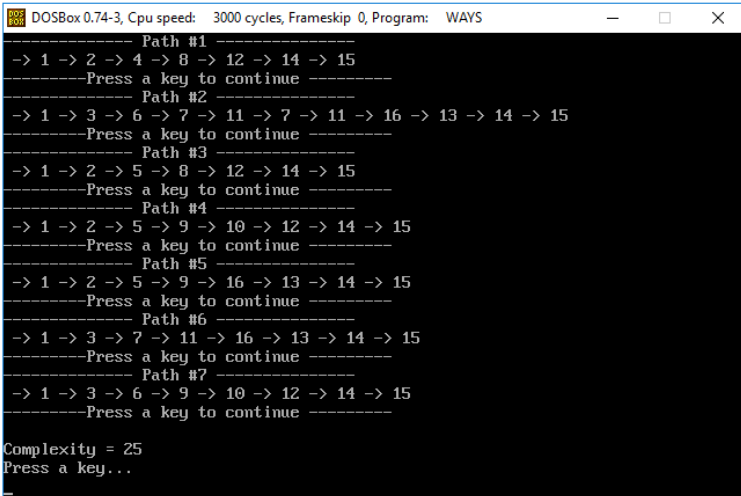


Рисунок 2 – Результат работы программы с неверным графом

На рис. 3 приведет результаты работы программы для графа из приложения

А.

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



```
Path #1
-> 1 -> 2 -> 4 -> 8 -> 12 -> 14 -> 15
Press a key to continue
Path #2
-> 1 -> 3 -> 6 -> 7 -> 11 -> 7 -> 11 -> 16 -> 13 -> 14 -> 15
Press a key to continue
Path #3
-> 1 -> 2 -> 5 -> 8 -> 12 -> 14 -> 15
Press a key to continue
Path #4
-> 1 -> 2 -> 5 -> 9 -> 10 -> 12 -> 14 -> 15
Press a key to continue
Path #5
-> 1 -> 2 -> 5 -> 9 -> 16 -> 13 -> 14 -> 15
Press a key to continue
Path #6
-> 1 -> 3 -> 7 -> 11 -> 16 -> 13 -> 14 -> 15
Press a key to continue
Path #7
-> 1 -> 3 -> 6 -> 9 -> 10 -> 12 -> 14 -> 15
Press a key to continue
Complexity = 25
Press a key...
```

Рисунок 3 – Результат работы программы с правильным графом

К тому же, на рис. 4 приведено задание к ЛР из методических указаний, где сказано, что состав и содержание маршрутов должны отличаться, поэтому отличается и сложность этих маршрутов (по большей части из-за того, что поменялась структура графа с добавлением новой вершины в области вершин 9-11).

Следует отметить, что приведенные наборы маршрутов являются лишь одними из возможных вариантов их построения.

Для каждой из заданных программ построение маршрутов и оценивание структурной сложности следует выполнять двумя способами:

- 1) «вручную», непосредственно по управляющему графу программы;
- 2) автоматически с помощью программы ways.exe (для контроля результата, полученного «вручную»).

**Состав и содержание маршрутов, полученных ручным и автоматическим способами должны отличаться.**

Рисунок 4 – Задание из методических указаний для ЛР 2

Маршруты для минимального покрытия:

1. 1, 2, 4, 8, 12, 14, 15;
2. 1, 2, 5, 8, 12, 14, 15;
3. 1, 2, 5, 9, 10, 12, 14, 15;
4. 1, 2, 5, 9, 16, 13, 14, 15;

5. 1, 3, 6, 7, 11, 7, 11, 16, 13, 14, 15;
6. 1, 3, 6, 9, 10, 12, 14;
7. 1, 3, 7, 11, 16, 13, 14, 15.

Сложность: 25

1. 7, **11**, 7;
2. **1**, **2**, 4, 8, 12, 14, 15;
3. **1**, **2**, **5**, 8, 12, 14, 15;
4. **1**, **2**, **5**, **9**, 10, 12, 14, 15;
5. **1**, **2**, **5**, **9**, 16, 13, 14, 15;
6. **1**, **3**, **6**, 7, **11**, 16, 13, 14, 15;
7. **1**, **3**, **6**, **9**, 10, 12, 14, 15;
8. **1**, **3**, 7, **11**, 16, 13, 14, 15;

Сложность:  $S_2=1+2+3+4+4+4+4+3=25$

### 3.2. Программа из ЛР1

Произведено построение графа работы для программы из ЛР1 (приложение В). Полученный граф представлен на рис. 5.

#### 3.2.1. Критерий минимального покрытия

Маршруты для минимального покрытия:

1, 2, **3**, 2, **3**, 4, 5, 6, 7, 6, 7, **8**, 5, 6, 7, **8**, 9, **10**, 9, **10**, 11, 12, **13**, 12, **13**, **14**, 11, 12, **13**, **14**, **15**, 16, 17, 18, **19**, 16, 17, 18, **19**, 20, **21**, 22, 23, **24**, 20, **21**, 23, **24**, 25, **26**, 27, 28, **29**, 25, **26**, 28, **29**, 30, **31**, 30, **31**, 32, **33**, 32, **33**, **34**, 4, 5, 6, 7, **8**, 9, **19**, 11, 12, **13**, **14**, **15**, **34**, 35, **36**, 35, **36**, 37

Сложность: 39.





Маршруты:

1. 2, **3**, 2;
2. 6, 7, 6;
3. 5, 6, 7, **8**, 5;
4. 9, **10**, 9;
5. 12, **13**, 12;
6. 11, 12, **13**, **14**, 11;
7. 16, 17, 18, **19**, 16;
8. 20, **21**, 22, 23, **24**, 20;
9. 25, **26**, 27, 28, **29**, 25;
10. 30, **31**, 30;
11. 32, **33**, 32;
12. 35, **36**, 35;
13. 4, 5, 6, 7, **8**, 9, **10**, 11, 12, **13**, **14**, 15, 16, 17, 18, **19**, 20, **21**,  
22, 23, **24**, 25, **26**, 27, 28, **29**, 30, **31**, 32, **33**, 34, 4;
14. 4, 5, 6, 7, **8**, 9, **10**, 11, 12, **13**, **14**, 15, 34, 4;
15. 1, 2, **3**, 4, 5, 6, 7, **8**, 9, **10**, 11, 12, **13**, **14**, 15, 16, 17, 18, **19**,  
20, **21**, 22, 23, **24**, 25, **26**, 27, 28, **29**, 30, **31**, 32, **33**, 34, 35, **36**, 37;
16. 1, 2, **3**, 4, 5, 6, 7, **8**, 9, **10**, 11, 12, **13**, **14**, 15, 16, 17, 18, **19**,  
20, **21**, 23, **24**, 25, **26**, 27, 28, **29**, 30, **31**, 32, **33**, 34, 35, **36**, 37;
17. 1, 2, **3**, 4, 5, 6, 7, **8**, 9, **10**, 11, 12, **13**, **14**, 15, 16, 17, 18, **19**, 20, **21**, 22, 23, **24**, 25,  
**26**, 28, **29**, 30, **31**, 32, **33**, 34, 35, **36**, 37.

Сложность: 75

### 3.2.3. Программный анализ

Граф задан в нотации программы.

Результаты в приложении Г. Лог работы программы в приложении Д. Результаты вычисления минимального покрытия соответствуют полученным в п.3.2.1. Результаты теоретического расчёта минимального покрытия с программным совпадают.

Вычисление базовых маршрутов заканчивается некорректным завершением программы-анализатора. При ручном расчёте сложность базовых маршрутов составила 75.

### Выводы

Изучено применение:

- критерия минимального покрытия,
- анализа базовых маршрутов для оценки структурной сложности программ.

Проведено ручное и автоматизированное вычисление метрик для двух примеров, получены похожие результаты.

Установлено, что программа-анализатор не отличается устойчивостью.

## ПРИЛОЖЕНИЕ А

### Граф из задания

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

Top{1}

Last{15}

Arcs{

arc(1,2);

arc(1,3);

arc(2,4);

arc(2,5);

arc(3,6);

arc(3,7);

arc(4,8);

arc(5,8);

arc(5,9);

arc(6,7);

arc(6,9);

arc(7,11);

arc(8,12);

arc(9,10);

arc(9,16);

arc(10,12);

arc(11,7);

arc(11,16);

arc(12,14);

arc(13,14);

arc(14,15);

arc(16,13);

}

## ПРИЛОЖЕНИЕ Б

### Лог работы для графа из задания

Min ways....

```
----- Path #1 -----
->1->2->4->8->12->14->15
-----Press a key to continue -----
----- Path #2 -----
->1->3->6->7->11->7->11->16->13->14->15
-----Press a key to continue -----
----- Path #3 -----
->1->2->5->8->12->14->15
-----Press a key to continue -----
----- Path #4 -----
->1->2->5->9->10->12->14->15
-----Press a key to continue -----
----- Path #5 -----
->1->2->5->9->16->13->14->15
-----Press a key to continue -----
----- Path #6 -----
->1->3->7->11->16->13->14->15
-----Press a key to continue -----
----- Path #7 -----
->1->3->6->9->10->12->14->15
-----Press a key to continue -----
Complexity = 25
Press a key...
```

Z ways....

```
----- Path #1 -----
->7->11->7
-----Press a key to continue -----
----- Path #1 -----
->1->2->4->8->12->14->15
-----Press a key to continue -----
----- Path #2 -----
->1->2->5->8->12->14->15
-----Press a key to continue -----
----- Path #3 -----
->1->2->5->9->10->12->14->15
-----Press a key to continue -----
----- Path #4 -----
```

->1->2->5->9->16->13->14->15  
-----Press a key to continue -----  
----- Path #5 -----  
->1->3->6->7->11->16->13->14->15  
-----Press a key to continue -----  
----- Path #6 -----  
->1->3->6->9->10->12->14->15  
-----Press a key to continue -----  
----- Path #7 -----  
->1->3->7->11->16->13->14->15  
-----Press a key to continue -----

Complexity = 25

Press a key...

## ПРИЛОЖЕНИЕ В

### Программа из ЛР1

```
#include "stdio.h"
#include "stdlib.h"
#include "stdbool.h"
#define RMAX 3
#define CMAX 3

float** _alloc_matr(int a, int b) {
    float** m = (float**)malloc(a * sizeof(float*));
    for (int i = 0; i < CMAX; i++) {
        m[i] = (float*)malloc(b * sizeof(float));
    }
    return m;
}

void _free_matr(float** m, int a) {
    for (int i = 0; i < a; i++) {
        free(m[i]);
    }
    free(m);
}

/* print out the answers */
void print_matr(float** a, float* y) {
    for (int i = 0; i < RMAX; i++) {
        for (int j = 0; j < CMAX; j++) {
            printf("%f  ", a[i][j]);
        }
        printf(": %f\n", y[i]);
    }
}

/* get the values for n, and arrays a,y */
void get_data(float** a, float* y) {
    for (int i = 0; i < RMAX; i++) {
        printf("Equation %d\n", i);
        for (int j = 0; j < CMAX; j++) {
            printf("%d:  ", j);
            scanf("%f", &a[i][j]);
        }
    }
}
```

```

    printf("C: ");
    scanf("%f", &y[i]);
}
print_matr(a, y);
printf("\n");
}

/* pascal program to calculate the determinant of a 3-by-3matrix */
float deter(float** a) {
    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(float** a, float** b, float* coef, float* y, int j, float det) {
    for (int 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;
}

int solve(float** a, float* y, float* coef) {
    float** b = _alloc_matr(RMAX, CMAX);
    float det = 0;
    for (int i = 0; i < RMAX; i++) {
        for (int j = 0; j < CMAX; j++) {
            b[i][j] = a[i][j];
        }
    }
    det = deter(b);
    if (det == 0) {
        printf("ERROR: matrix is singular.");
        return 1;
    }
    setup(a, b, coef, y, 0, det);
    setup(a, b, coef, y, 1, det);
    setup(a, b, coef, y, 2, det);
    _free_matr(b, RMAX);
    return 0;
}

```



```

void write_data(float* coef) {
    for (int i = 0; i < CMAX; i++) {
        printf("%f ", coef[i]);
    }
    printf("\n");
}

int main() {
    float** a = _alloc_matr(RMAX, CMAX);
    float* y = (float*)malloc(CMAX * sizeof(float));
    float* coef = (float*)malloc(CMAX * sizeof(float));
    int error;
    char scan;
    while (1) {
        get_data(a, y);
        error = solve(a, y, coef);
        if (!error) {
            write_data(coef);
        }
        printf("More? ");
        scanf(" %c", &scan);
        if (scan != 'y') {
            break;
        }
    }
    free(y);
    free(coef);
    _free_matr(a, RMAX);
    return 0;
}

```

## ПРИЛОЖЕНИЕ Г

### Граф ЛР1 в нотации программы-анализатора

Nodes{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,34, 35, 36, 37 }

Top{1}

Last{37}

Arcs{

arc(1, 2);  
arc(2, 3);  
arc(3, 2);  
arc(3, 4);  
arc(4, 5);  
arc(5, 6);  
arc(6, 7);  
arc(7, 6);  
arc(7, 8);  
arc(8, 5);  
arc(8, 9);  
arc(9, 10);  
arc(10, 9);  
arc(10, 11);  
arc(11, 12);  
arc(12, 13);  
arc(13, 12);  
arc(13, 14);  
arc(14, 11);  
arc(14, 15);  
arc(15, 16);  
arc(15, 34);  
arc(16, 17);  
arc(17, 18);  
arc(18, 19);  
arc(19, 16);  
arc(19, 20);

arc(20, 21);  
arc(21, 22);  
arc(21, 23);  
arc(22, 23);  
arc(23, 24);  
arc(24, 20);  
arc(24, 25);  
arc(25, 26);  
arc(26, 27);  
arc(26, 28);  
arc(27, 28);  
arc(28, 29);  
arc(29, 25);  
arc(29, 30);  
arc(30, 31);  
arc(31, 30);  
arc(31, 32);  
arc(32, 33);  
arc(33, 32);  
arc(33, 34);  
arc(34, 4);  
arc(34, 35);  
arc(35, 36);  
arc(36, 35);  
arc(36, 37);

}

## ПРИЛОЖЕНИЕ Д

### Лог работы программы для графа ЛР1

Min ways....

----- Path #1 -----

->1->2->3->2->3->4->5->6->7->6->7->8->5  
->6->7->8->9->10->9->10->11->12->13->12  
->13->14->11->12->13->14->15->16->17->18->  
19->16->17->18->19->20->21->22->23->24->20  
->21->23->24->25->26->27->28->29->25->26->  
28->29->30->31->30->31->32->33->32->33->34  
->4->5->6->7->8->9->10->11->12->13->14->  
15->34->35->36->35->36->37

-----Press a key to continue -----

Complexity = 39

Press a key...

Z ways....

Abnormal program termination