

**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. 6304  
Преподаватель

\_\_\_\_\_  
\_\_\_\_\_

Корытов П.В.  
Кирияничков В.А.

Санкт-Петербург  
2020

## **1. Цель работы**

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

## **2. Постановка задачи**

Выполнить оценивание структурной сложности двух программ с помощью критериев:

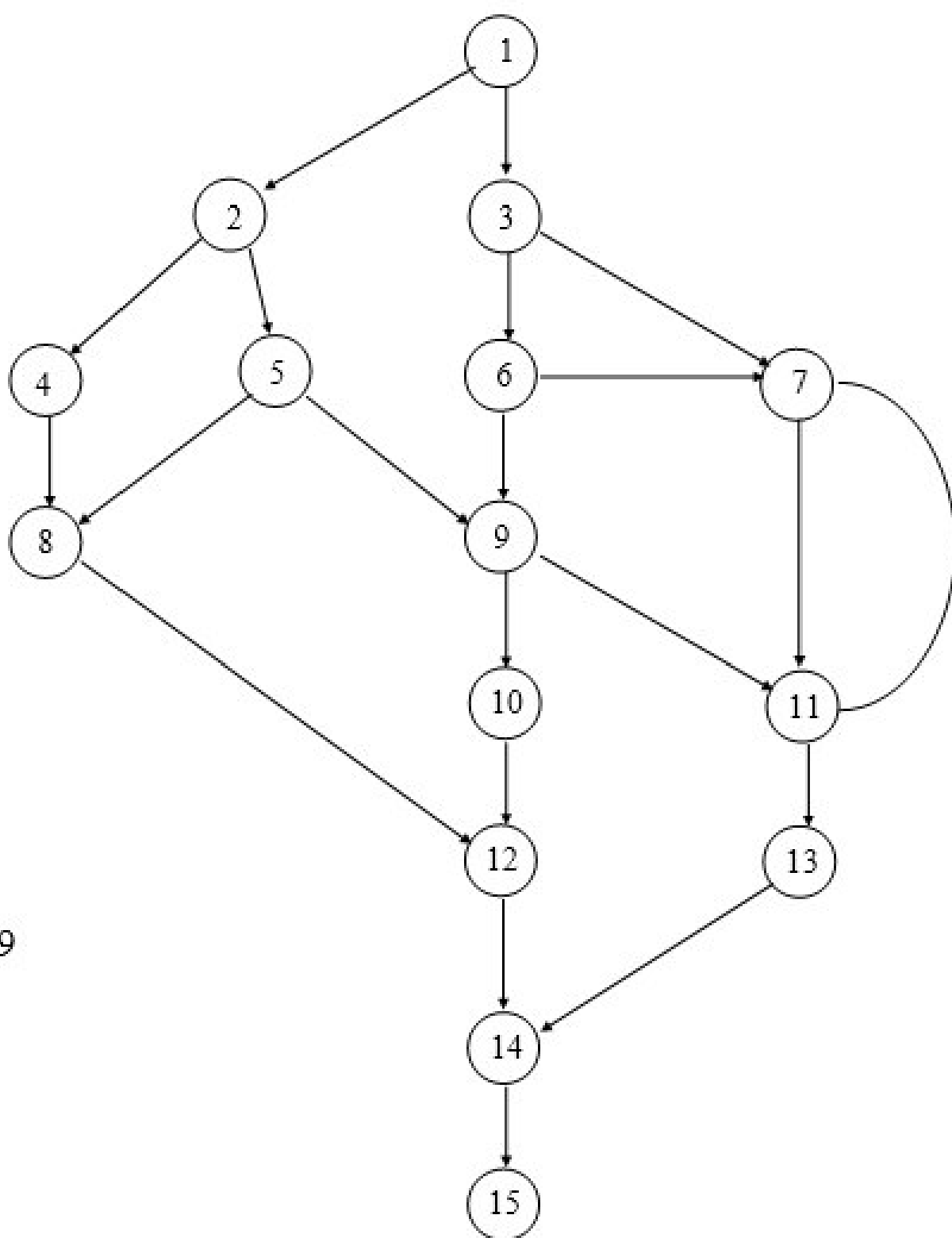
- Минимального покрытия дуг графа;
- Выбора маршрутов на основе цикломатического числа графа.

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

### 3. Ход работы

#### 3.1. Программа из задания

##### 3.1.1. Критерий минимального покрытия

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, 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

Сложность программы:

$$S_2 = \sum_{i=1}^M \xi_i = 2 + 3 + 6 + 4 + 3 + 5 = 23. \quad (3.1)$$

##### 3.1.2. Анализ базовых маршрутов

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

$$Z = Y - N + 2 * \Omega = 8. \quad (3.2)$$

Маршруты:

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, 11, 13, 14, 15;
6. 1, 3, 6, 9, 10, 12, 14, 15;
7. 1, 3, 7, 11, 13, 14, 15;
8. 1, 3, 6, 7, 11, 13, 14, 15;

Сложность программы:

$$S_2 = \sum_{i=1}^M \xi_i = 2 + 2 + 3 + 4 + 5 + 4 + 3 + 4 = 27. \quad (3.3)$$

### 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. Для решения добавлено дополнительная вершина. Лог работы программы в приложении Б.

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

Базовые маршруты:

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: 25

### 3.2. Программа из ЛР1

Произведено построение графа работы для программы из ЛР1 (приложение В). Полученный граф представлен на рис. 2.

#### 3.2.1. Критерий минимального покрытия

Маршруты для минимального покрытия:

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,

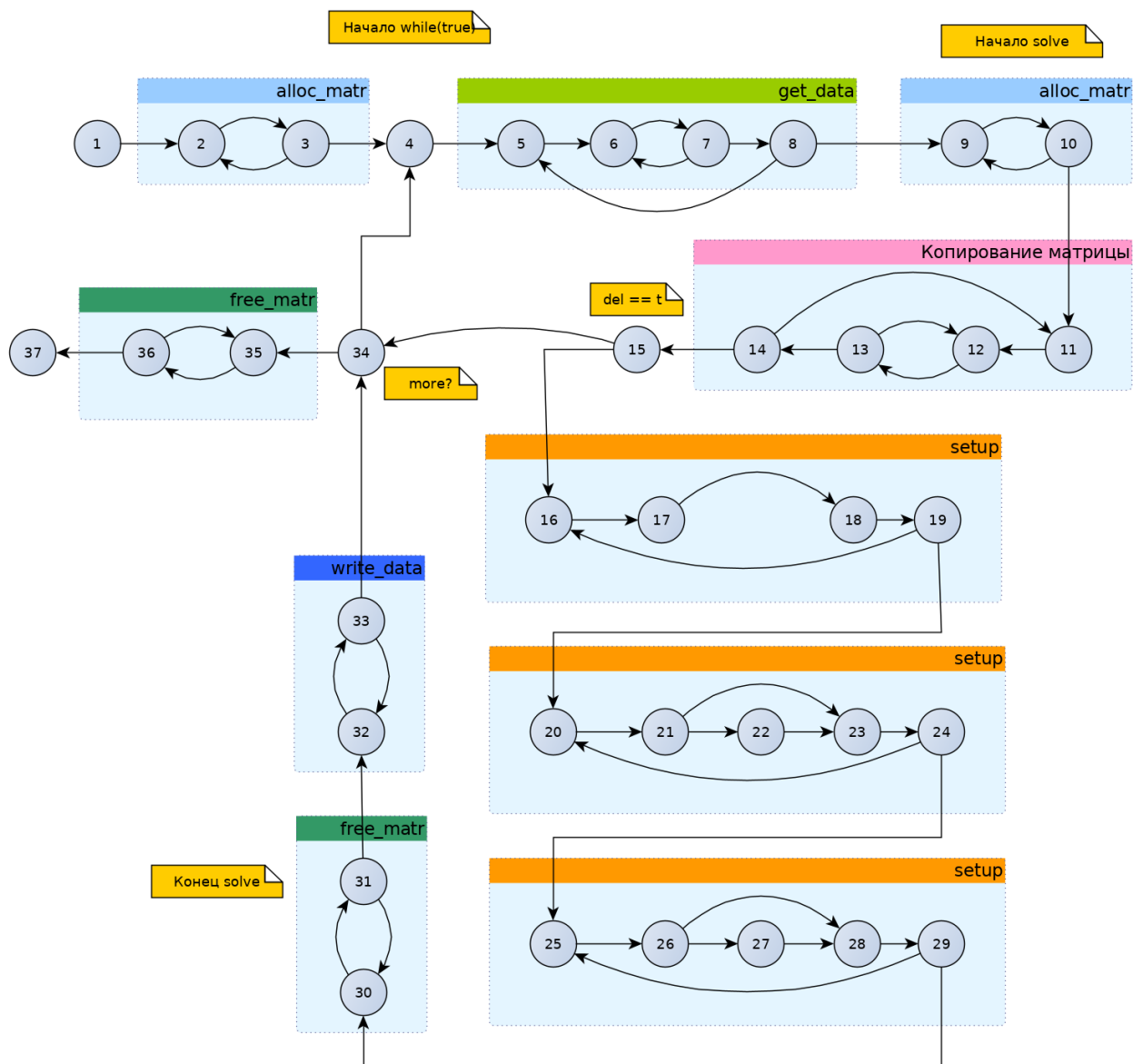


Рисунок 2 – Граф приложения

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.

### 3.2.2. Анализ базовых маршрутов

Число вершин графа  $N = 37$ , число дуг графа  $Y = 52$ , число связных компонент  $\Omega = 1$ . Цикломатическое число  $Z$ :

$$Z = Y - N + 2 * \Omega = 17. \quad (3.4)$$

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.

Вычисление базовых маршрутов заканчивается некорректным завершением программы-анализатора.

## **4. 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

```
1  Nodes { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 }
2
3  Top { 1 }
4
5  Last { 15 }
6
7  Arcs {
8      arc(1, 2);
9      arc(1, 3);
10     arc(2, 4);
11     arc(2, 5);
12     arc(3, 6);
13     arc(3, 7);
14     arc(4, 8);
15     arc(5, 8);
16     arc(5, 9);
17     arc(6, 7);
18     arc(6, 9);
19     arc(7, 11);
20     arc(8, 12);
21     arc(9, 10);
22     arc(9, 16);
23     arc(10, 12);
24     arc(11, 7);
25     arc(11, 16);
26     arc(12, 14);
27     arc(13, 14);
28     arc(14, 15);
29     arc(16, 13);
30 }
```

## ПРИЛОЖЕНИЕ Б

### Лог работы для графа из задания

```
1  Min ways....
2  ----- Path #1 -----
3  -> 1 -> 2 -> 4 -> 8 -> 12 -> 14 -> 15
4  -----Press a key to continue -----
5  ----- Path #2 -----
6  -> 1 -> 3 -> 6 -> 7 -> 11 -> 7 -> 11 -> 16 -> 13 -> 14 -> 15
7  -----Press a key to continue -----
8  ----- Path #3 -----
9  -> 1 -> 2 -> 5 -> 8 -> 12 -> 14 -> 15
10 -----Press a key to continue -----
11 ----- Path #4 -----
12 -> 1 -> 2 -> 5 -> 9 -> 10 -> 12 -> 14 -> 15
13 -----Press a key to continue -----
14 ----- Path #5 -----
15 -> 1 -> 2 -> 5 -> 9 -> 16 -> 13 -> 14 -> 15
16 -----Press a key to continue -----
17 ----- Path #6 -----
18 -> 1 -> 3 -> 7 -> 11 -> 16 -> 13 -> 14 -> 15
19 -----Press a key to continue -----
20 ----- Path #7 -----
21 -> 1 -> 3 -> 6 -> 9 -> 10 -> 12 -> 14 -> 15
22 -----Press a key to continue -----
23
24 Complexity = 25
25 Press a key...
26
27 Z ways....
28 ----- Path #1 -----
29 -> 7 -> 11 -> 7
30 -----Press a key to continue -----
31 ----- Path #1 -----
32 -> 1 -> 2 -> 4 -> 8 -> 12 -> 14 -> 15
```

```

33 -----Press a key to continue -----
34 ----- Path #2 -----
35 -> 1 -> 2 -> 5 -> 8 -> 12 -> 14 -> 15
36 -----Press a key to continue -----
37 ----- Path #3 -----
38 -> 1 -> 2 -> 5 -> 9 -> 10 -> 12 -> 14 -> 15
39 -----Press a key to continue -----
40 ----- Path #4 -----
41 -> 1 -> 2 -> 5 -> 9 -> 16 -> 13 -> 14 -> 15
42 -----Press a key to continue -----
43 ----- Path #5 -----
44 -> 1 -> 3 -> 6 -> 7 -> 11 -> 16 -> 13 -> 14 -> 15
45 -----Press a key to continue -----
46 ----- Path #6 -----
47 -> 1 -> 3 -> 6 -> 9 -> 10 -> 12 -> 14 -> 15
48 -----Press a key to continue -----
49 ----- Path #7 -----
50 -> 1 -> 3 -> 7 -> 11 -> 16 -> 13 -> 14 -> 15
51 -----Press a key to continue -----
52
53 Complexity = 25
54 Press a key...

```

## ПРИЛОЖЕНИЕ В

### Программа из ЛР1

```
1  #include "stdio.h"
2  #include "stdlib.h"
3  #include "stdbool.h"
4
5  #define RMAX 3
6  #define CMAX 3
7
8  float** _alloc_matr(int a, int b) {
9      float** m = (float**)malloc(a * sizeof(float*));
10     for (int i = 0; i < CMAX; i++) {
11         m[i] = (float*)malloc(b * sizeof(float));
12     }
13     return m;
14 }
15
16 void _free_matr(float** m, int a) {
17     for (int i = 0; i < a; i++) {
18         free(m[i]);
19     }
20     free(m);
21 }
22
23
24 /* print out the answers */
25 void print_matr(float** a, float* y) {
26     for (int i = 0; i < RMAX; i++) {
27         for (int j = 0; j < CMAX; j++) {
28             printf("%f ", a[i][j]);
29         }
30         printf(": %f\n", y[i]);
31     }
32 }
```

```

33
34  /* get the values for n, and arrays a,y */
35  void get_data(float** a, float* y) {
36      for (int i = 0; i < RMAX; i++) {
37          printf("Equation %d\n", i);
38          for (int j = 0; j < CMAX; j++) {
39              printf("%d: ", j);
40              scanf("%f", &a[i][j]);
41          }
42          printf("C: ");
43          scanf("%f", &y[i]);
44      }
45      print_matr(a, y);
46      printf("\n");
47  }
48
49  /* pascal program to calculate the determinant of a 3-by-3matrix */
50  float deter(float** a) {
51      return a[0][0] * (a[1][1] * a[2][2] - a [2][1] * a[1][2])
52          - a[0][1] * (a[1][0] * a[2][2] - a [2][0] * a[1][2])
53          + a[0][2] * (a[1][0] * a[2][1] - a [2][0] * a[1][1]);
54  }
55
56  void setup(float** a, float** b, float* coef, float* y, int j,
    ↪ float det) {
57      for (int i = 0; i < RMAX; i++) {
58          b[i][j] = y[i];
59          if (j > 0) {
60              b[i][j-1] = a[i][j-1];
61          }
62      }
63      coef[j] = deter(b) / det;
64  }
65

```

```

66 bool solve(float** a, float* y, float* coef) {
67     float** b = _alloc_matr(RMAX, CMAX);
68     float det = 0;
69     for (int i = 0; i < RMAX; i++) {
70         for (int j = 0; j < CMAX; j++) {
71             b[i][j] = a[i][j];
72         }
73     }
74     det = deter(b);
75     if (det == 0) {
76         printf("ERROR: matrix is singular.");
77         return true;
78     }
79     setup(a, b, coef, y, 0, det);
80     setup(a, b, coef, y, 1, det);
81     setup(a, b, coef, y, 2, det);
82     _free_matr(b, RMAX);
83     return false;
84 }
85
86 void write_data(float* coef) {
87     for (int i = 0; i < CMAX; i++) {
88         printf("%f ", coef[i]);
89     }
90     printf("\n");
91 }
92
93 int main() {
94     float** a = _alloc_matr(RMAX, CMAX);
95     float* y = (float*)malloc(CMAX * sizeof(float));
96     float* coef = (float*)malloc(CMAX * sizeof(float));
97     bool error;
98     char scan;
99     while (true) {

```

```
100         get_data(a, y);
101         error = solve(a, y, coef);
102         if (!error) {
103             write_data(coef);
104         }
105         printf("More? ");
106         scanf(" %c", &scan);
107         if (scan != 'y') {
108             break;
109         }
110     }
111     free(y);
112     free(coef);
113     _free_matr(a, RMAX);
114     return 0;
115 }
```

## ПРИЛОЖЕНИЕ Г

### Граф ЛР1 в нотации программы-анализатора

```
1  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 }  
2  
3  Top { 1 }  
4  
5  Last { 37 }  
6  
7  Arcs {  
8      arc(1, 2);  
9      arc(2, 3);  
10     arc(3, 2);  
11     arc(3, 4);  
12     arc(4, 5);  
13     arc(5, 6);  
14     arc(6, 7);  
15     arc(7, 6);  
16     arc(7, 8);  
17     arc(8, 5);  
18     arc(8, 9);  
19     arc(9, 10);  
20     arc(10, 9);  
21     arc(10, 11);  
22     arc(11, 12);  
23     arc(12, 13);  
24     arc(13, 12);  
25     arc(13, 14);  
26     arc(14, 11);  
27     arc(14, 15);  
28     arc(15, 16);  
29     arc(15, 34);  
30     arc(16, 17);
```



```
31     arc(17, 18);
32     arc(18, 19);
33     arc(19, 16);
34     arc(19, 20);
35     arc(20, 21);
36     arc(21, 22);
37     arc(21, 23);
38     arc(22, 23);
39     arc(23, 24);
40     arc(24, 20);
41     arc(24, 25);
42     arc(25, 26);
43     arc(26, 27);
44     arc(26, 28);
45     arc(27, 28);
46     arc(28, 29);
47     arc(29, 25);
48     arc(29, 30);
49     arc(30, 31);
50     arc(31, 30);
51     arc(31, 32);
52     arc(32, 33);
53     arc(33, 32);
54     arc(33, 34);
55     arc(34, 4);
56     arc(34, 35);
57     arc(35, 36);
58     arc(36, 35);
59     arc(36, 37);
60 }
```

## ПРИЛОЖЕНИЕ Д

### Лог работы программы для графа ЛР1

```
1 Min ways....
2 ----- Path #1 -----
3 -> 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
4 -----Press a key to continue -----
5
6 Complexity = 39
7 Press a key...
8
9 Z ways....
10 Abnormal program termination
```