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»

Студентка гр. 8304 _____ Мельникова О.А.

Преподаватель _____ Кирьянчиков В.А.

Санкт-Петербург
2022

Формулировка задания

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).

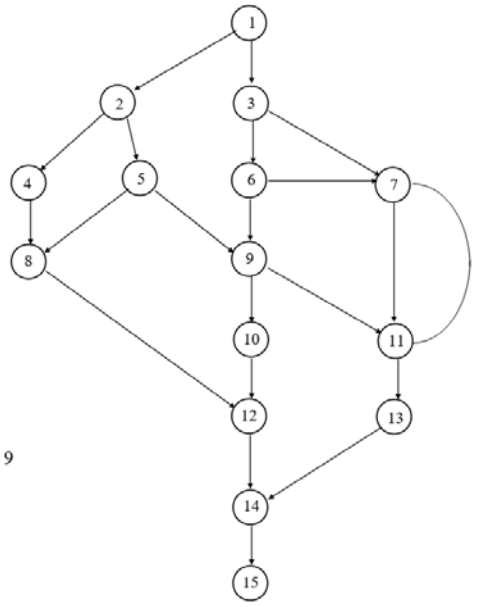
Оцениваемые характеристики структурной сложности:

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

Ход работы

1. Анализ заданной программы (вариант 9)

1.1. Граф программы 1



9

Рисунок 1 – Заданный управляющий граф

1.2. Ручные расчеты

Критерий 1:

M1: **1-2**-4-8-12-14-15 |2 ветвлений

M2: **1-2-5**-8-12-14-15 |3 ветвления

M3: **1-2-5-9**-10-12-14-15 |4 ветвления

M4: **1-3-6-9-11**-13-14-15 |5 ветвления

M5: **1-3**-7-**11**-7-**11**-13-14-15 |4 ветвления

M6: **1-3-6**-7-**11**-13-14-15 |4 ветвления

Количество маршрутов M = 6

Сложность: $S_2 = \sum_{i=1}^{M} \xi_i = 2 + 3 + 4 + 4 + 5 + 4 + 4 = 22$

Критерий 2:

Добавим ребро 15-1 для того чтобы граф стал связным.

$Z = Y - N + 2 * P = 20 - 15 + 2 * 1 = 7$

Количество проверяемых маршрутов равно цикломатическому числу графа.

M1: **1-2-**4-8-12-14-15 |2 ветвлений

M2: **1-2-5-9-11**-13-14-15 |5 ветвления

M3: **1-2-5-**8-12-14-15 |3 ветвления

M4:**1-3-**7-**11-**13-14-15 |3 ветвления

M5:**1-3-6-9-11-**13-14-15 |5 ветвления

M6**: 1-3-6-9**-10-12-14-15 |4 ветвление

M7**: 1-3-6-**7-**11-**13-14-15 |4 ветвление

Сложность: $S_2 = \sum_{i=1}^{M} \xi_i = 2 + 5 + 3 + 3 + 5 + 4 + 4 = 26$

1.3. Автоматические расчёты

Описание структуры графа:
Nodes{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}
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(6,9);
arc(6,7);
arc(7,11);
arc(8,12);
arc(9,10);
arc(9,11);
arc(10,12);
arc(11,13);
arc(11,7);
arc(12,14);
arc(13,14);
arc(14,15);}

Результат анализа структурной сложности:

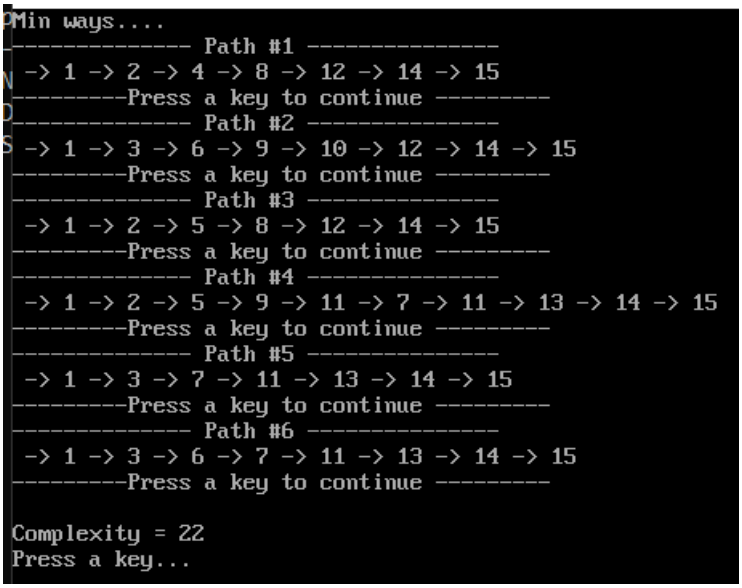


Рисунок 2 - Результат выполнения ways.exe для 1-го критерия

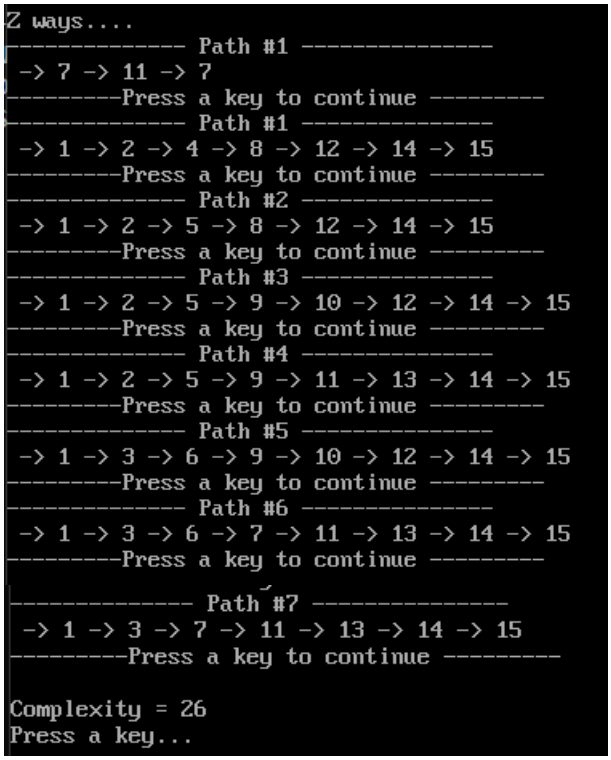


Рисунок 3 - Результат выполнения ways.exe для 2-го критерия

2.Анализ программы из 1-ой лабораторной работы

2.1. Граф программы 2

Из-за цикла for и получающейся структуры, представленной на рис. 4, следующий участок кода был заменен на другой, результат представлен на рис. 5.

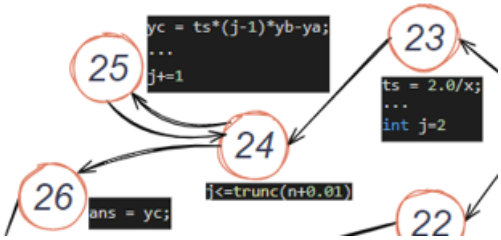


Рисунок 4 – недопустимый вид структуры графа

```
}else{
    ts = 2.0/x;
    ya = y0;
    yb = y1;
    for (int j=2; j<=trunc(n+0.01); j+=1) {
        yc = ts*(j-1)*yb-ya;
        ya = yb;
        yb = yc;
    }
    ans = yc;
}
```

```
else{
    ts = 2.0/x;
    ya = y0;
    yb = y1;
    ans = yc;
    int j=2;
    if(j<=trunc(n+0.01)){
        do{
            yc = ts*(j-1)*yb-ya;
            ya = yb;
            yb = yc;
            ans = yc;
            j+=1;
        }while(j<=trunc(n+0.01))
    }
}
```

Рисунок 5 – измененный участок кода

Код программы представлен в приложении А.

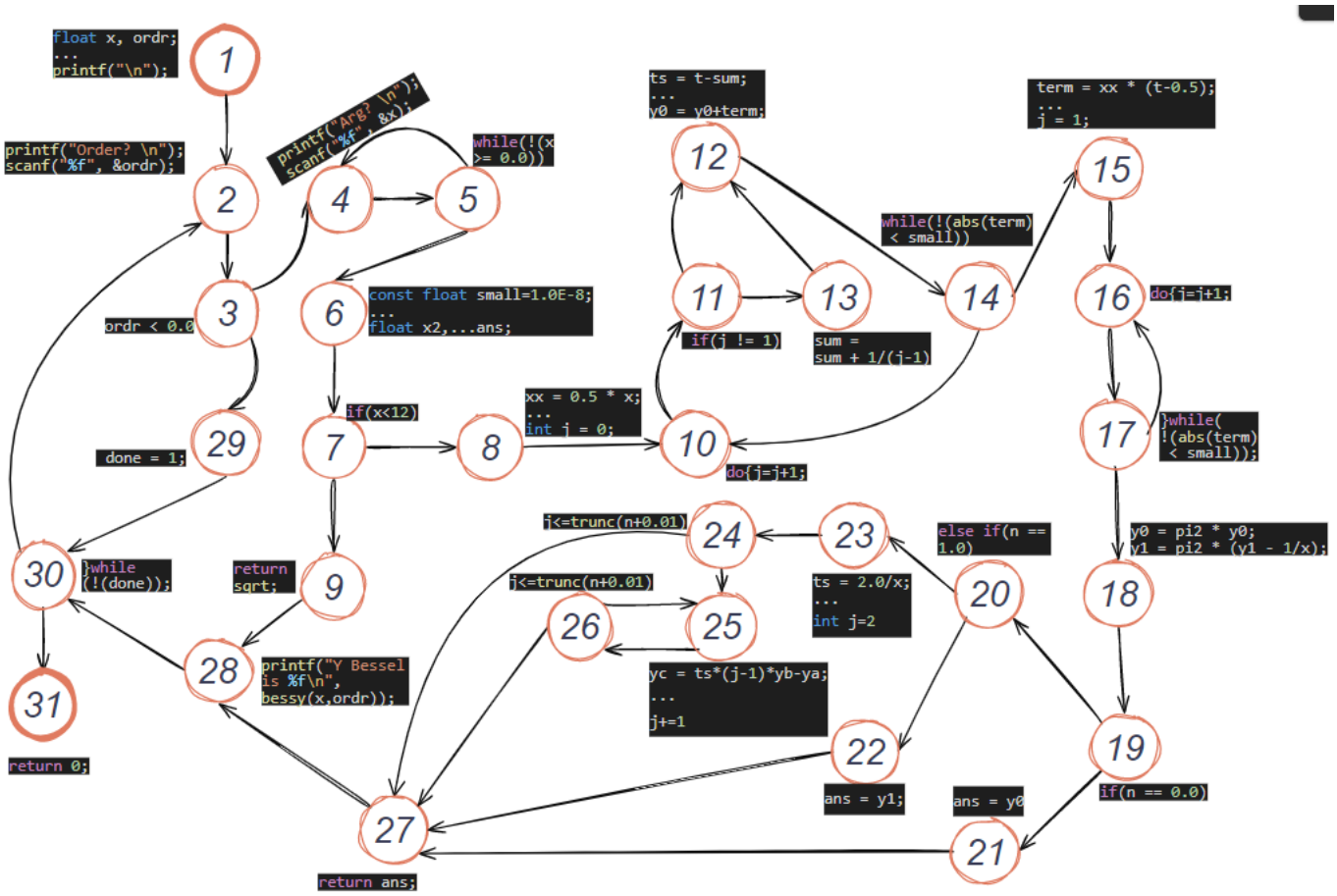


Рисунок 6 – управляющий граф программы 2

2.2. Ручные расчеты

Критерий 1:

M1: 1-2-**3**-4-**5**-4-**5**-6-**7**-8-10-**11**-13-12-**14**-10-**11**-12-**14**-15-16-**17**-16-**17**-18-**19**-21-27-28-**30**-2-**3**-4-**5**-6-**7**-9-28-**30**-2-**3**-4-**5**-6-**7**-8-10-**11**-12-14-15-16-**17**-18-**19**-**20**-23-**24**-25-26-25-26-27-28-**30**-2-**3**-4-**5**-6-**7**-8-10-**11**-12-**14**-15-16-**17**-18-**19**-**20**-23-**24**-27-28-**30**-2-**3**-4-**5**-6-**7**-8-10-**11**-12-**14**-15-16-**17**-18-**19**-**20**-22-27-28-**30**-2-**3**-29-**30**-31

| 49 ветвлений

Количество маршрутов M = 1

Сложность: S=49.

Критерий 2:

$Z = Y - N + 2 * P = 40 - 31 + 2 * 1 = 11$

Количество проверяемых маршрутов равно цикломатическому числу графа.

26-25-26

4-**5**-4

16-**17**-16

1-2-**3**-4-**5**-6-**7**-8-10-**11**-12-**14**-15-16-**17**-18-**19**-**20**-23-**24**-27-28-**30**-31

1-2-**3**-29-**30**-31

1-2-**3**-4-**5**-6-**7**-9-28-**30**-31

1-2-**3**-4-**5**-6-**7**-8-10-**11**-13-12-**14**-15-16-**17**-18-**19**-**20**-22-27-28-**30**-31

1-2-**3**-4-**5**-6-**7**-8-10-**11**-13-12-**14**-15-16-**17**-18-**19**-21-27-28-**30**-31

10-**11**-12-**14**-10

10-**11**-13-12-**14**-10

3-4-**5**-6-**7**-9-28-**30**-2-**3**

Итого 43 ветвления.

2.3. Автоматические расчёты

Описание структуры графа:
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}
Top{1}
Last{31}
Arcs{
arc(1,2);
arc(2,3);
arc(3,4);
arc(3,29);
arc(4,5);
arc(5,4);
arc(5,6);
arc(6,7);
arc(7,9);
arc(7,8);
arc(8,10);
arc(9,28);
arc(10,11);
arc(11,12);
arc(11,13);
arc(12,14);
arc(13,12);
arc(14,10);
arc(14,15);
arc(15,16);
arc(16,17);
arc(17,16);
arc(17,18);
arc(18,19);
arc(19,21);
arc(19,20);
arc(21,27);
arc(20,23);
arc(20,22);
arc(22,27);
arc(23,24);
arc(24,25);
arc(24,27);
arc(25,26);
arc(26,25);
arc(26,27);
arc(27,28);

arc(28,30);
arc(29,30);
arc(30,31);
arc(30,2);}

Результат анализа структурной сложности:

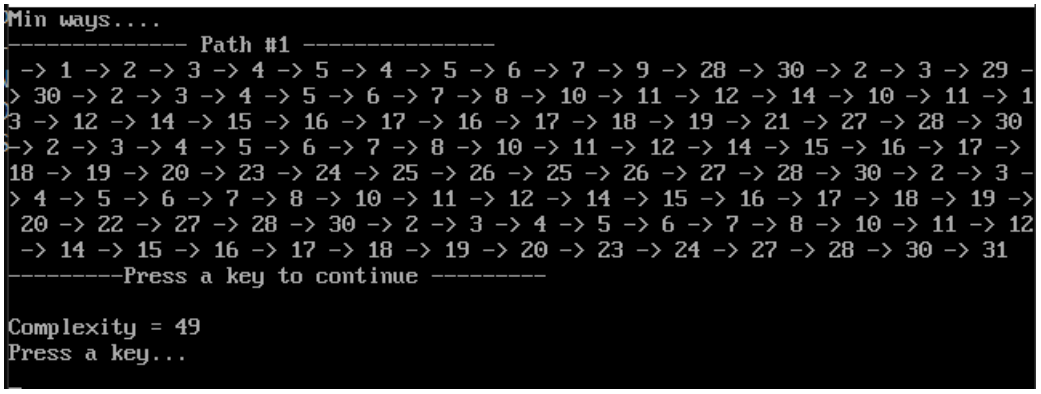


Рисунок 7 - Результат выполнения ways.exe для 1-го критерия

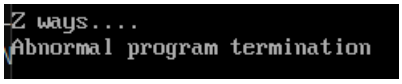


Рисунок 8 - Результат выполнения ways.exe для 2-го критерия

Вывод

В данной лабораторной работе была выполнена оценка структурной сложности двух программ с помощью критериев: минимального покрытия дуг графа и выбора маршрутов на основе цикломатического числа графа. Расчеты были проведены как ручным, так и программным способом.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ИЗ ЛАБОРАТОРНОЙ №1

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

float bessy(float x, float n){
    const float small = 1.0E-8;
  const float euler = 0.57721566;
  const float pi = 3.1415926;
  const float pi2 = 0.63661977;

  float x2, sum, t, ts, term, xx, y0, y1, ya, yb, yc, ans;

    if(x<12){
        xx = 0.5 * x;
        x2 = xx * xx;
        t = log(xx) + euler;
        sum = 0.0;
        term = t;
        y0 = t;
        int j = 0;
        do{
            j = j+1;
            if(j != 1) sum = sum + 1/(j-1);
            ts = t-sum;
            term = -x2 * term / (j*j) * (1-1 / (j*ts));
            y0 = y0+term;
        }while(!(abs(term) < small));
        term = xx * (t-0.5);
        sum = 0.0;
        y1 = term;
        j = 1;
        do{
            j = j+1;
            sum = sum+1/(j-1);
            ts = t-sum;
            term = (-x2 * term) / (j * (j-1)) * ((ts-0.5 / j) / (ts + 0.5 / (j-
1)));
            y1 = y1+term;
        }while(!(abs(term) < small));
        y0 = pi2 * y0;
        y1 = pi2 * (y1 - 1/x);
        if(n == 0.0){
            ans = y0;
        }else if(n == 1.0){
            ans = y1;
        }
        else{
            ts = 2.0/x;
            ya = y0;
            yb = y1;
            ans = yc;
            int j=2;
            if(j<=trunc(n+0.01)){
                do{
                    yc = ts*(j-1)*yb-ya;
                    ya = yb;
                    yb = yc;
                    ans = yc;
```

```
                    j+=1;
                }while(j<=trunc(n+0.01))
            }
        }
        return ans;
    }else{
        return sqrt(2 / (pi*x)) * sin(x - pi/4 - n * pi/2);
    }
}

int main(){
    float x, ordr;
    int done = 0;
    printf("\n");
    do{
        printf("Order? \n");
        scanf("%f", &ordr);
        if(ordr < 0.0){
            done = 1;
        }else{
            do{
                printf("Arg? \n");
                scanf("%f", &x);
            }while(!(x >= 0.0));
            printf("Y Bessel is %f \n", bessy(x,ordr));
        }
    }while(!(done));
    return 0;
}
```