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 _____ Алтухов А.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

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;
- Программа из 1-ой лабораторной работы (управляющий граф составить самостоятельно).

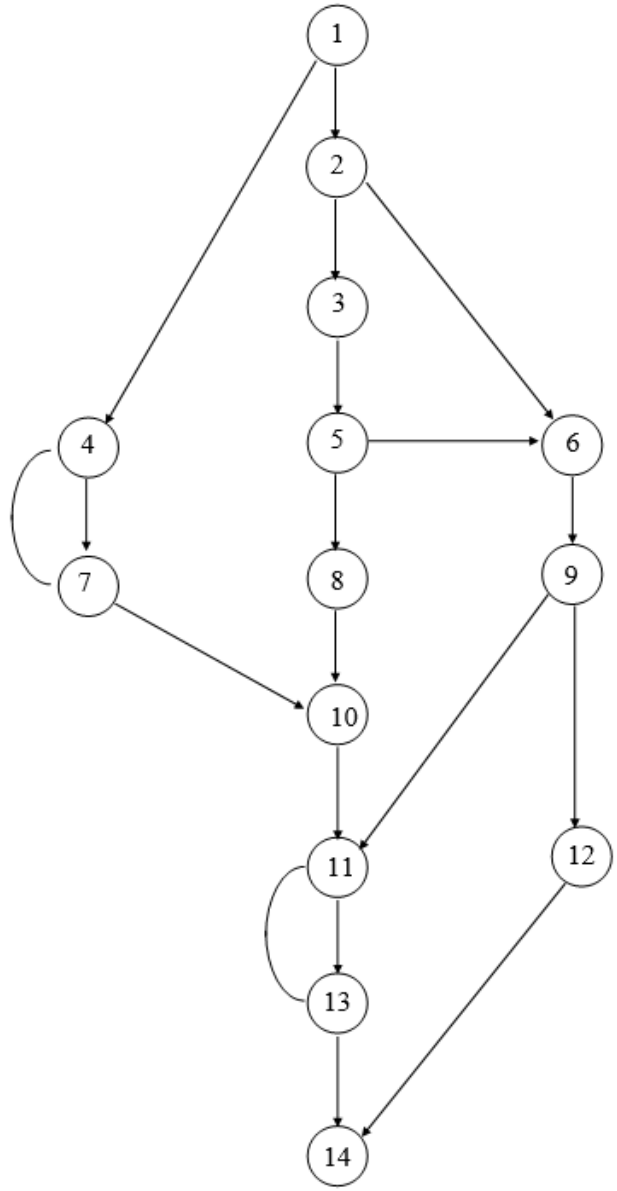
Оцениваемые характеристики структурной сложности:

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

Ход выполнения.

Было проведено оценивание структурной сложности программы с заданной структурой управляющего графа (вариант 1), изображенного на рисунке 1.

ВАРИАНТ



1

Рисунок 1 – Управляющий граф программы, соответствующий варианту 1

Был выполнен расчет структурной сложности заданной программы по первому критерию – минимальное покрытие вершин и дуг управления.

M1: **1**-4-**7**-4-**7**-10-11-**13**-11-**13**-14 S1=5

M2: **1**-**2**-3-**5**-8-10-11-**13**-14 S2=4

M3: **1**-**2**-3-**5**-6-**9**-11-**13**-14 S3=5

M4: **1**-**2**-6-**9**-12-14 S4=3

Сложность программы:

$$S = 5 + 4 + 5 + 3 = 17$$

Был выполнен расчёт структурной сложности этой программы по второму критерию – каждый линейно-независимый цикл и ациклический участок программы.

Полное число вершин N = 14

Количество связывающих дуг Y = 19

Число связных компонент P = 1

Цикломатическое число Z = Y - N + 2 * P = 7

Линейно-независимые циклические маршруты:

M1: 4-**7** S1=1

M2: 11-**13** S2=1

M3: **1**-4-**7**-10-11-**13**-14 S3=3

M4: **1**-**2**-3-**5**-8-10-11-**13**-14 S4=4

M5: **1**-**2**-3-**5**-6-**9**-11-**13**-14 S5=5

M6: **1**-**2**-3-**5**-6-**9**-12-14 S6=4

M7: **1**-**2**-6-**9**-12-14 S7=3

Сложность программы:

$$S = 1 + 1 + 3 + 4 + 5 + 4 + 3 = 21$$

Проведен автоматический расчет сложности программы по обоим критериям с помощью программы ways.exe. Граф в текстовом виде представлен в приложении А.

Результаты работы программы представлены на рисунках 2-3.

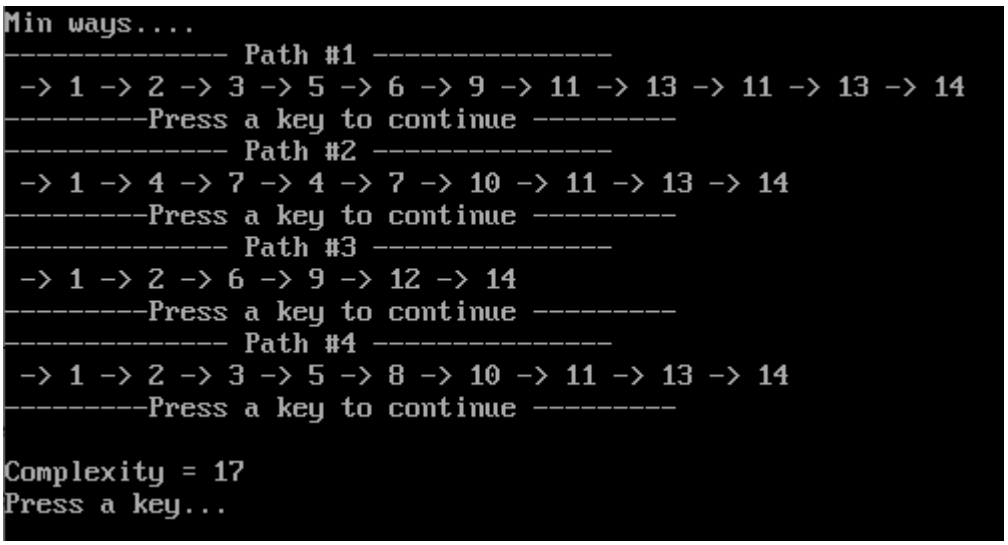


Рисунок 2 – Расчет сложности по первому критерию программой ways.exe

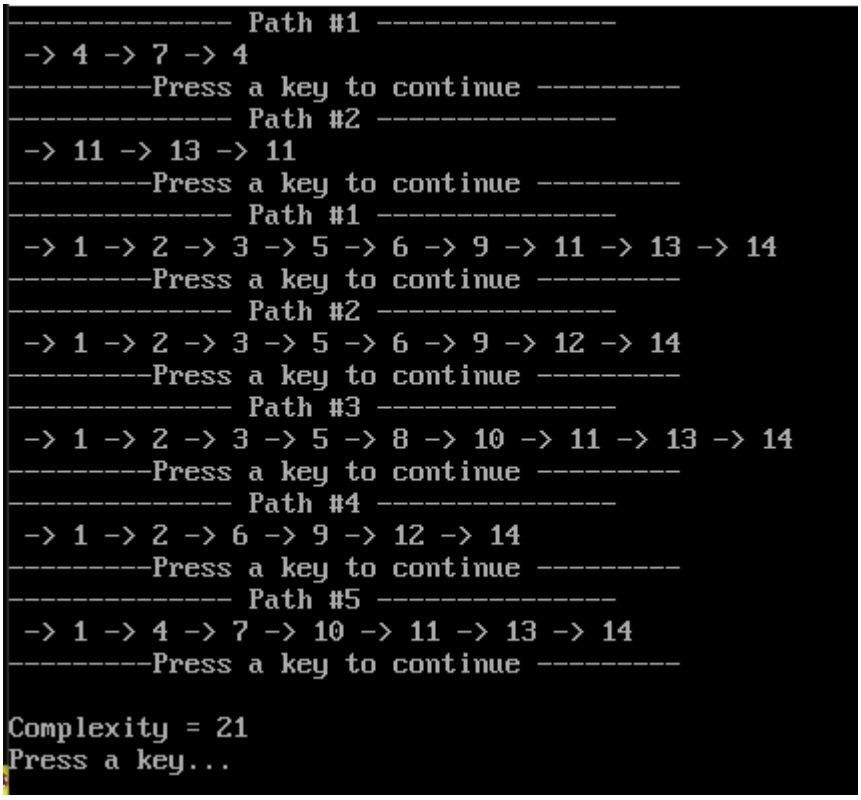


Рисунок 3 – Расчет сложности по второму критерию программой ways.exe

Структурная сложность программы из первой лабораторной работы

Для программы из лабораторной работы получен граф потока, представленный на рисунке 4.

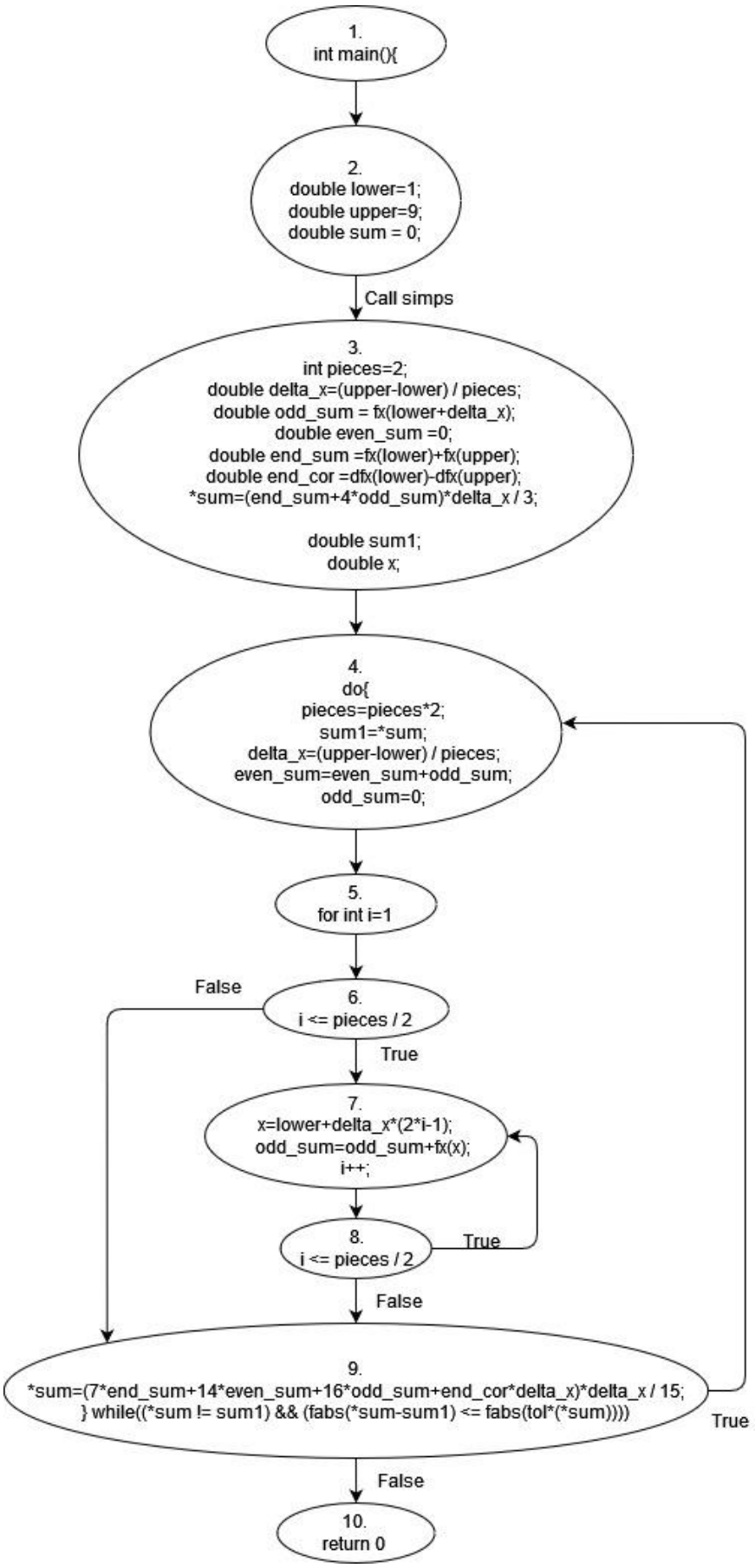


Рисунок 4 – Граф потока для программы из первой лабораторной

Текстовое представление графа приведено в приложении Б. Код программы, для которой составлялся граф, представлен в приложении В.

Был выполнен расчет структурной сложности заданной программы по первому критерию – минимальное покрытие вершин и дуг управления.

M1: 1-2-3-4-5-**6**-**9**-4-5-**6**-7-**8**-7-**8**-**9**-10 S1=6

Сложность программы:

$$S = 6$$

Был выполнен расчёт структурной сложности этой программы по второму критерию – каждый линейно-независимый цикл и ациклический участок программы.

Полное число вершин N = 10

Количество связывающих дуг Y = 12

Число связных компонент P = 1

Цикломатическое число Z = Y - N + 2 * P = 4

Линейно-независимые циклические маршруты:

M1: 7-**8** S1=1

M2: 4-5-**6**-**9** S2=2

M3: 1-2-3-4-5-**6**-**9**-10 S3=2

M4: 1-2-3-4-5-**6**-7-**8**-**9**-10 S4=3

Сложность программы:

$$S = 1 + 2 + 2 + 3 = 8$$

Проведен автоматический расчет сложности программы по обоим критериям с помощью программы ways.exe. Результаты работы программы представлены на рисунках 5-6.

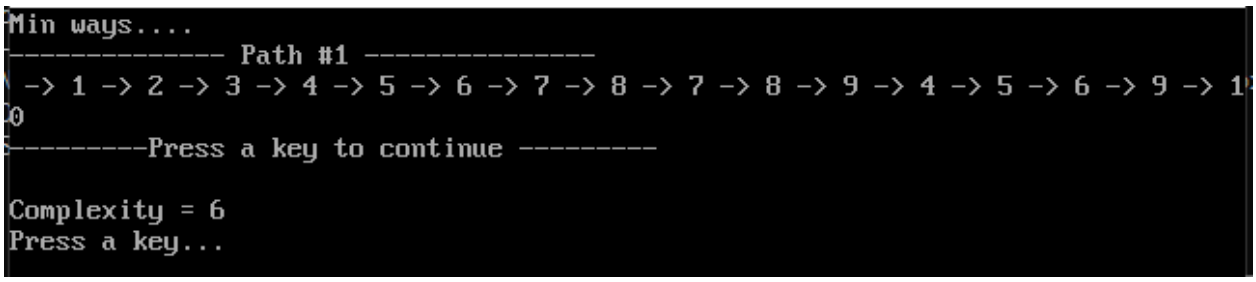


Рисунок 5 – Расчет сложности по первому критерию программой ways.exe

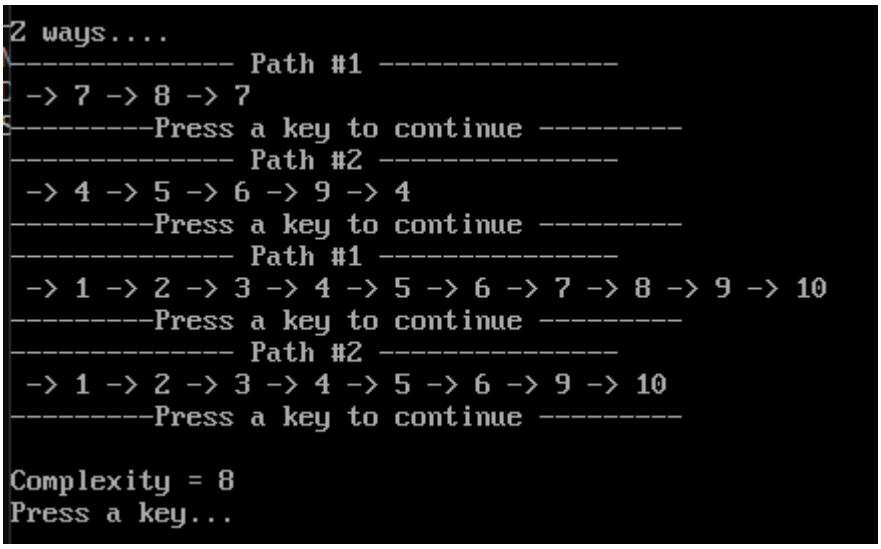


Рисунок 5 – Расчет сложности по второму критерию программой ways.exe

Выводы.

В результате выполнения данной лабораторной работы были изучены критерии оценивания структурной сложности программ. Была проведена оценка структурной сложности (число учитываемых маршрутов проверки программы, цикломатическое число, суммарное число ветвлений по всем маршрутам) для двух программ — соответствующая варианту и программы из первой лабораторной работы, первая оказалась более сложной по структуре.

Приложение А.

Nodes{

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

}

Top{1}

Last{14}

Arcs{

arc(1,2);

arc(1,4);

arc(2,3);

arc(2,6);

arc(3,5);

arc(4,7);

arc(5,6);

arc(5,8);

arc(6,9);

arc(7,4);

arc(7,10);

arc(8,10);

arc(9,11);

arc(9,12);

arc(10,11);

arc(11,13);

arc(12,14);

arc(13,11);

arc(13,14);

}

Приложение Б.

Nodes{
1,2,3,4,5,6,7,8,9,10
}

Top{1}
Last{10}

Arcs{
arc(1,2);
arc(2,3);
arc(3,4);
arc(4,5);
arc(5,6);
arc(6,7);
arc(6,9);
arc(7,8);
arc(8,9);
arc(8,7);
arc(9,4);
arc(9,10);
}

Приложение В.

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

const double tol = 1.0E-6;

double fx(double x) {
   return exp(-x / 2);
}

double dfx(double x) {
   return -(exp(-x / 2)) / 2;
}

double simps(double lower, double upper, double tol, double* sum) {
   int pieces=2;
   double delta_x=(upper-lower) / pieces;
   double odd_sum = fx(lower+delta_x);
   double even_sum =0;
   double end_sum =fx(lower)+fx(upper);
   double end_cor =dfx(lower)-dfx(upper);
   *sum=(end_sum+4*odd_sum)*delta_x / 3;

   double sum1;
   double x;

   do
   {
      pieces=pieces*2;
```

```
        sum1=*sum;
        delta_x=(upper-lower) / pieces;
        even_sum=even_sum+odd_sum;
        odd_sum=0;
        for (int i=1; i <= pieces / 2; i++) {

                x=lower+delta_x*(2*i-1);
                odd_sum=odd_sum+fx(x);

        }
        *sum=(7*end_sum+14*even_sum+16*odd_sum+end_cor*delta_x)*delta_x /
15;
    } while ( (*sum != sum1) && (fabs(*sum-sum1) <= fabs(tol*(*sum))) );
}
int main()
{
    double lower=1;
    double upper=9;
    double sum = 0;
    simps(lower,upper,tol,&sum);
    return 0;
}
```