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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Качество и метрология программного обеспечения»

Тема: Анализ структурной сложности графовых моделей программ

Студент гр. 8304 _____ Мешков М.А.

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.

1. Было проведено оценивание структурной сложности программы с заданной структурой управляющего графа (вариант 10) — см. рис. 1.

Вычисление сложности программы по критерию минимального покрытия вершин и дуг графа управления:

m1: **1** — **2** — 4 — 10 — 13 — **15** — 16 S1=3

m2: **1** — **2** — 5 — **8** — 10 — 13 — **15** — 16 S2=4

m3: **1** — **3** — **6** — **11** — 13 — **15** — 16 S3=5

m4: **1** — **2** — 5 — **8** — **11** — 12 — 14 — **15** — 12 — 14 — **15** — 16 S4=6

m5: **1** — **3** — **6** — 9 — 12 — 14 — **15** — 16 S5=4

m6: **1** — **3** — 7 — 9 — 12 — 14 — **15** — 16 S6=3

S=25

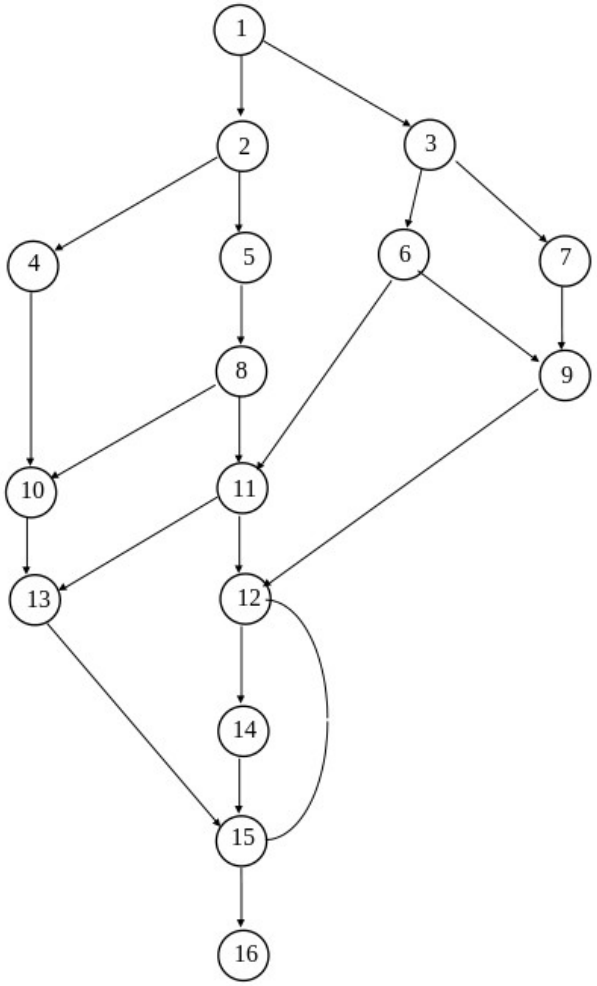

Рисунок 1 — Управляющий граф программы в соответствии с вариантом

Вычисление сложности программы с выбором маршрутов на основе цикломатического числа графа:

Цикломатическое число **Z = 22 - 16 + 2×1 = 8**

m1: 12 — 14 — **15** — 12 S1=1

m2: **1** — **2** — 4 — 10 — 13 — **15** — 16 S2=3

m3: **1** — **2** — 5 — **8** — 10 — 13 — **15** — 16 S3=4

m4: **1** — **3** — **6** — **11** — 13 — **15** — 16 S4=5

m5: **1** — **2** — 5 — **8** — **11** — 12 — 14 — **15** — 16 S5=5

m6: **1** — **3** — **6** — 9 — 12 — 14 — **15** — 16 S6=4

m7: **1** — **3** — 7 — 9 — 12 — 14 — **15** — 16 S7=3

M8: **1** — **3** — **6** — **11** — 12 — 14 — **15** — 16 S8=5

S=30

Также было проведено оценивание структурной сложности заданной программы с помощью программы ways (вход программы см. в приложении А) — см. рис 2 и 3.

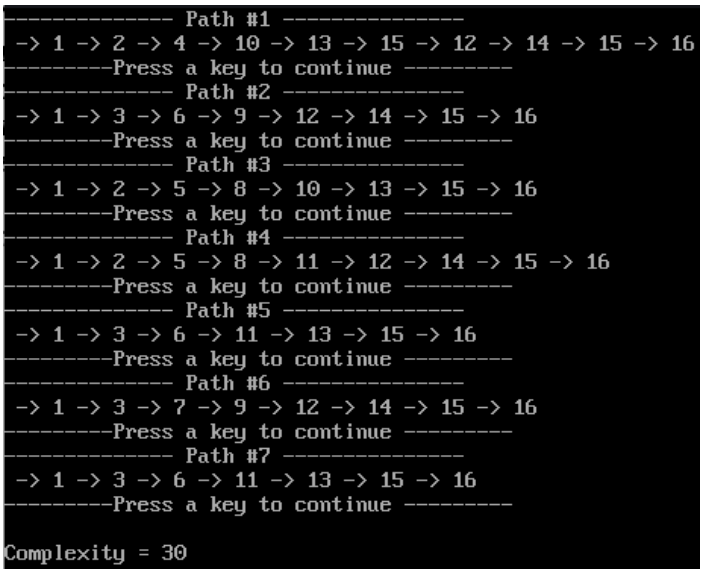

Рисунок 2 — Результат программы ways по критерию минимального покрытия

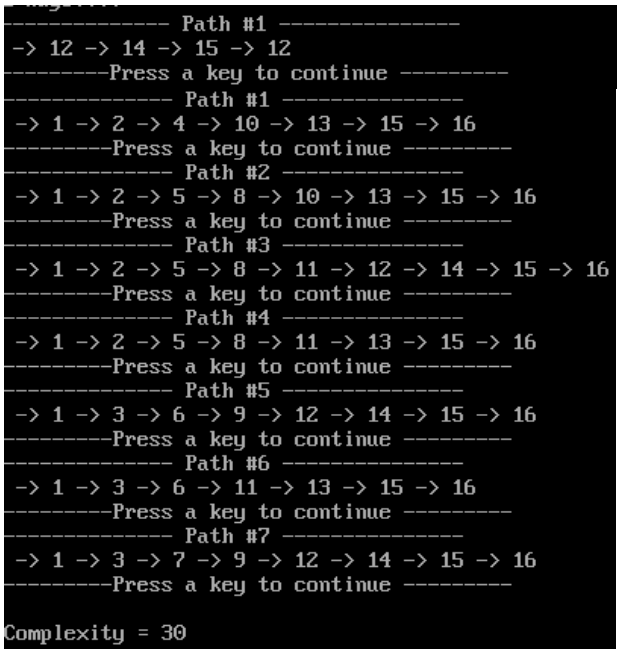

Рисунок 3 — Результат программы ways по путям по цикломатическому числу

Видно что результат программы ways по критерию минимального покрытия (рис. 2) некорректен, т. к. маршруты 5 и 7 одинаковы. А результат программы по путям по цикломатическому числу (рис. 3) совпал с ручным расчетом.

2. Для программы из 1-ой лабораторной работы был составлен управляющий граф — см. рис. 4 (см. код программы в приложении B с комментариями, отражающими соответствие операторов и вершин).

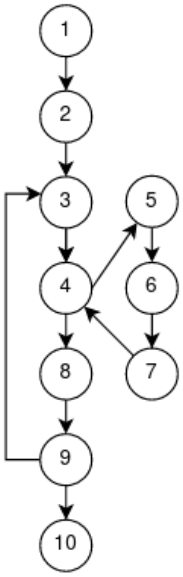

Рисунок 4 — Управляющий граф программы из лаб. работы 1

Вычисление сложности программы по критерию минимального покрытия вершин и дуг графа управления:

m1: 1 — 2 — 3 — **4** — 5 — 6 — 7 — **4** — 8 — **9** — 3 — **4** — 8 — **9** — 10 S1=5

S=5

Вычисление сложности программы с выбором маршрутов на основе цикломатического числа графа:

Цикломатическое число **Z = 11 - 10 + 2×1 = 3**

m1: 3 — **4** — 8 — **9** — 3 S1=2

m2: 5 — 6 — 7 — **4** — 5 S2=1

m3: 1 — 2 — 3 — **4** — 8 — **9** — 10 S3=2

S=5

Оценивание структурной сложности заданной программы с помощью программы ways провести не удалось, т. к. программа зависает (вход программы см. в приложении C).

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. УПРАВЛЯЮЩИЙ ГРАФ ПРОГРАММЫ В СООТВЕТСТВИИ С ВАРИАНТОМ

```
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(3,7);
arc(4,10);
arc(5,8);
arc(6,9);
arc(6,11);
arc(7,9);
arc(8,10);
arc(8,11);
arc(9,12);
arc(10,13);
arc(11,12);
arc(11,13);
arc(12,14);
arc(13,15);
arc(14,15);
arc(15,12);
arc(15,16);
}
```

ПРИЛОЖЕНИЕ В. ПРОГРАММА НА ЯЗЫКЕ СИ ИЗ ЛАБ. РАБОТЫ 1

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

double fx(double x) {
   return 1.0 / x; // (6)
}

void simps(double lower, double upper, double tol, double *sum) {
   int pieces = 2; // (2)
   double
      delta_x = (upper-lower)/pieces,
      odd_sum = fx(lower+delta_x),
      even_sum = 0.0,
      end_sum = fx(lower) + fx(upper); // (2)
   *sum = (end_sum + 4.0 * odd_sum)*delta_x/3.0; // (2)
   printf("%5d %f\n", pieces, *sum); // (2)
   double sum1; // (2)
   do {
      pieces *= 2; // (3)
      sum1 = *sum; // (3)
      delta_x = (upper-lower)/pieces; // (3)
      even_sum = even_sum + odd_sum; // (3)
      odd_sum = 0.0; // (3)
      for (int i = 1; i <= pieces/2; i++) { // (4)
         double x = lower+delta_x*(2*i-1); // (5)
         odd_sum += fx(x); // (7)
      }
      *sum = (end_sum + 4.0*odd_sum + 2.0*even_sum)*delta_x/3.0; // (8)
```

```
        printf("%5d %f\n", pieces, *sum); // (8)
    } while (*sum != sum1 && fabs(*sum-sum1) >= fabs(tol**sum)); // (9)
}

int main() {
    const double tol = 1.0e-6; // (1)
    double
        lower = 1.0,
        upper = 9.0,
        sum; // (1)
    simps(lower, upper, tol, &sum);
    printf("\narea=%f\n", sum); // (10)
}
```

ПРИЛОЖЕНИЕ С. УПРАВЛЯЮЩИЙ ГРАФ ПРОГРАММЫ ИЗ ЛАБ. РАБОТЫ 1

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(4,8);

arc(5,6);

arc(6,7);

arc(7,4);

arc(8,9);

arc(9,3);

arc(9,10);

}