

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

Николаева М. А.

Преподаватель

Кирияничков В. А.

Санкт-Петербург

2022

Задание.

Выполнить оценивание структурной сложности двух программ с помощью критериев:

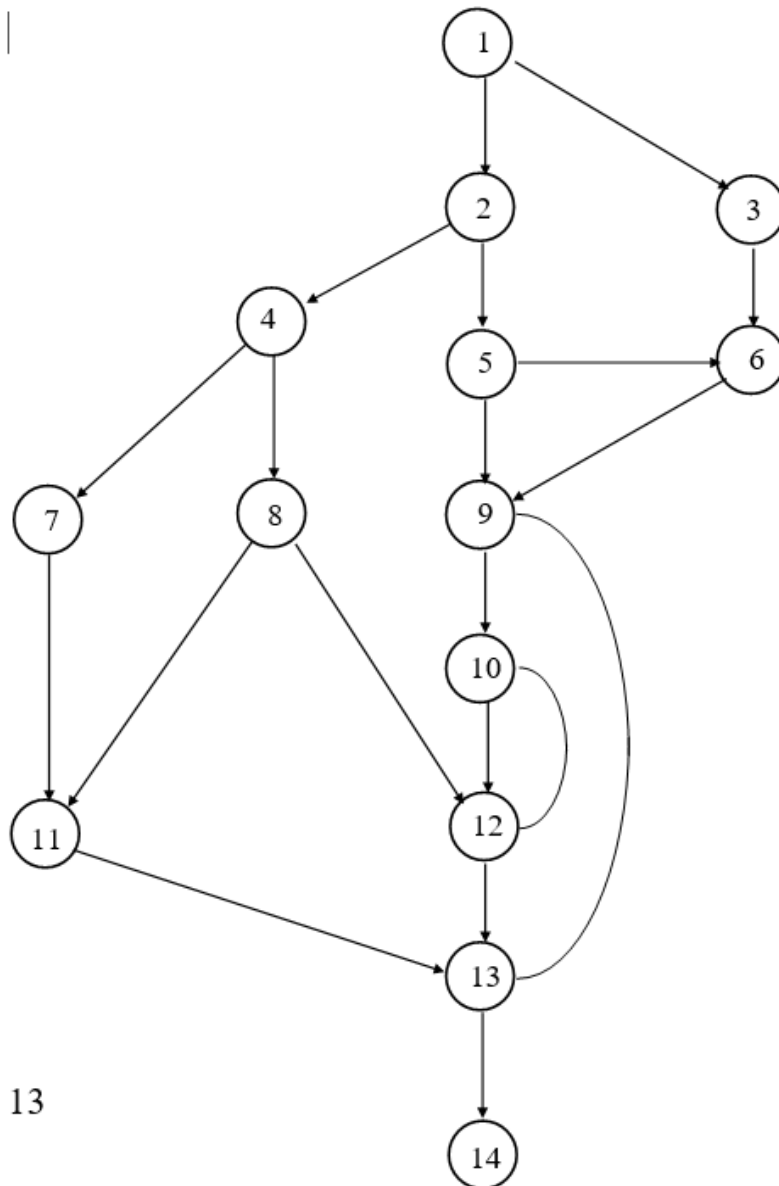
- Минимального покрытия вершин и дуг графа управления;
- Выбора маршрутов на основе цикломатического числа графа.

Варианты программ:

- Программа с заданной с заданной структурой управляющего графа, выбираемой из файла `zadan_struct.doc` в соответствии с номером в списке группы (рисунок 1).
- 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.



13

Рисунок 1 – Граф из файла zadan_struct.doc

Оценка структурной сложности программы из файла zadan_struct.doc.

Оценивание структурной сложности с помощью критерия минимального покрытия дуг графа **вручную**.

Ветвления: 1, 2, 4, 5, 8, 12, 13. Всего ветвлений – 7.

Минимальный набор путей (жирным выделены ветвления):

- 1-2-4-7-11-13-14 (4 ветвления);
- 1-2-4-8-11-13-14 (5 ветвлений);
- 1-2-4-8-12-13-14 (6 ветвлений);
- 1-3-6-9-10-12-13-14 (3 ветвления);

- 1-2-5-6-9-10-12-10-12-13-9-10-12-13-14 (8 ветвлений);
- 1-2-5-9-10-12-13-14 (5 ветвлений).

Итого сложность равна $4 + 5 + 6 + 3 + 8 + 5 = 31$.

Оценивание структурной сложности с помощью критерия на основе цикломатического числа **вручную**.

Число вершин в графе – 14, число ребер – 20. Для того, чтобы граф стал связным (из каждой вершины существовал путь в любую другую) достаточно добавить одно ребро $(14-1)$. Таким образом, цикломатическое число графа равно $20 - 14 + 2 * 1 = 8$. Значит необходимо рассмотреть 8 линейно-независимых циклов и путей.

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

Итого сложность равна $1 + 2 + 4 + 5 + 6 + 3 + 5 + 5 = 31$.

Оценивание структурной сложности с помощью критерия минимального покрытия дуг графа **программно** представлено на рисунке 2. Структура графа для программы представлена в приложении Б.

```

Min ways....
----- Path #1 -----
-> 1 -> 2 -> 4 -> 7 -> 11 -> 13 -> 9 -> 10 -> 12 -> 10 -> 12 -> 13 -> 14
-----Press a key to continue -----
----- Path #2 -----
-> 1 -> 3 -> 6 -> 9 -> 10 -> 12 -> 13 -> 14
-----Press a key to continue -----
----- Path #3 -----
-> 1 -> 2 -> 5 -> 6 -> 9 -> 10 -> 12 -> 13 -> 14
-----Press a key to continue -----
----- Path #4 -----
-> 1 -> 2 -> 4 -> 8 -> 11 -> 13 -> 14
-----Press a key to continue -----
----- Path #5 -----
-> 1 -> 2 -> 4 -> 8 -> 12 -> 13 -> 14
-----Press a key to continue -----
----- Path #6 -----
-> 1 -> 2 -> 5 -> 9 -> 10 -> 12 -> 13 -> 14
-----Press a key to continue -----

Complexity = 31
Press a key...

```

Рисунок 2 - Минимальное покрытие дуг

Оценивание структурной сложности с помощью критерия на основе цикломатического числа **программно** представлено на рисунке 3.

```

Z ways....
----- Path #1 -----
-> 10 -> 12 -> 10
-----Press a key to continue -----
----- Path #2 -----
-> 9 -> 10 -> 12 -> 13 -> 9
-----Press a key to continue -----
----- Path #1 -----
-> 1 -> 2 -> 4 -> 7 -> 11 -> 13 -> 14
-----Press a key to continue -----
----- Path #2 -----
-> 1 -> 2 -> 4 -> 8 -> 11 -> 13 -> 14
-----Press a key to continue -----
----- Path #3 -----
-> 1 -> 2 -> 4 -> 8 -> 12 -> 13 -> 14
-----Press a key to continue -----
----- Path #4 -----
-> 1 -> 2 -> 5 -> 6 -> 9 -> 10 -> 12 -> 13 -> 14
-----Press a key to continue -----
----- Path #5 -----
-> 1 -> 2 -> 5 -> 9 -> 10 -> 12 -> 13 -> 14
-----Press a key to continue -----
----- Path #6 -----
-> 1 -> 3 -> 6 -> 9 -> 10 -> 12 -> 13 -> 14
-----Press a key to continue -----

Complexity = 31
Press a key...

```

Рисунок 3 – Цикломатическое число

Оценка структурной сложности программы из 1-ой лабораторной.

Код программы из первой лабораторной представлен в приложении А. Граф программы представлен на рисунке 4.

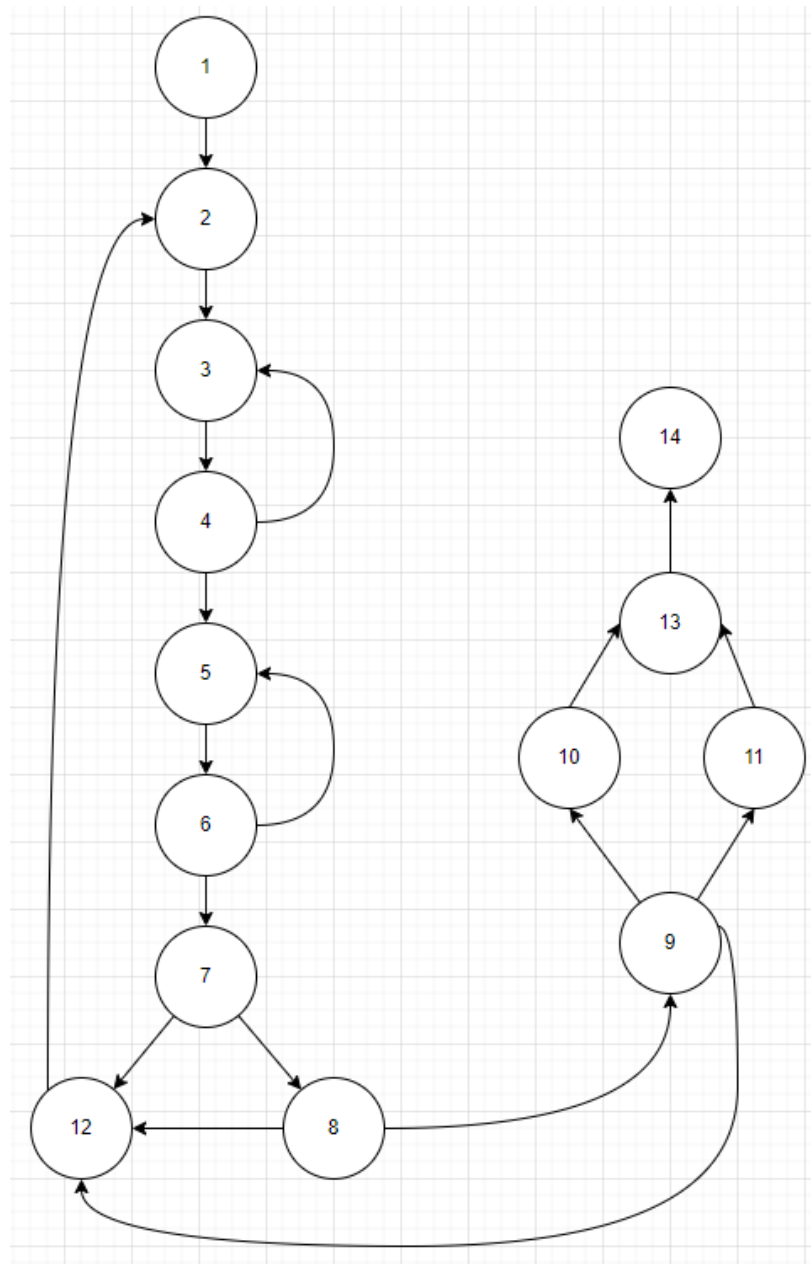


Рисунок 4 - Граф программы на языке СИ

Ключевые узлы и переходы графа:

- 3-4-3 – Первый FOR;
- 5-6-5 – Второй FOR;
- 2-3-4-5-6-7-12-2 – DO-while
- 7-12– Проверка на $n \leq 4$;

- 7-8 – Проверка на $n > 4$;
- 8-12 – Проверка на $t[nn + 1] == 0.0$;
- 8-9 – Проверка на $t[nn + 1] != 0.0$;
- 9-10 – Проверка на $(fabs(t[ntra+1]-t[nn+1]) \leq fabs(t[nn+1]*tol)) \parallel (fabs(t[nn-1]-t[j]) \leq fabs(t[j]*tol))$;
- 9-11 – Проверка на $n > 15$;
- 9-12 – Если предыдущие проверки не прошли;

Оценивание структурной сложности с помощью критерия минимального покрытия дуг графа **вручную**.

Ветвления: 4, 6, 7, 8, 9. Всего ветвлений – 5.

Минимальный набор путей (жирным выделены ветвления):

- 1-2-3-**4**-3-**4**-5-**6**-5-**6**-**7**-12-2-3-**4**-5-**6**-**7**-**8**-12-2-3-**4**-5-**6**-**7**-**8**-**9**-12-2-3-**4**-5-**6**-**7**-**8**-**9**-10-13-14 (19 ветвлений);
- 1-2-3-**4**-5-**6**-**7**-**8**-**9**-11-13-14 (5 ветвлений).

Итого сложность равна $19 + 5 = 24$.

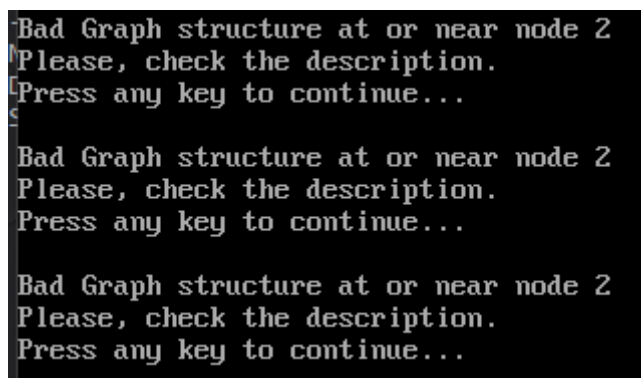
Оценивание структурной сложности с помощью критерия на основе цикломатического числа **вручную**.

Число вершин в графе – 14, число ребер – 19. Для того, чтобы граф стал связным (из каждой вершины существовал путь в любую другую) достаточно добавить одно ребро (14-1). Таким образом, цикломатическое число графа равно $19 - 14 + 2 * 1 = 7$. Значит необходимо рассмотреть 7 линейно-независимых циклов и путей.

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

Итого сложность равна $1 + 1 + 3 + 4 + 5 + 5 + 5 = 24$.

Оценивание структурной сложности с помощью критерия минимального покрытия дуг графа и с помощью критерия на основе цикломатического числа **программно** не удалось, т.к. программа выдала ошибку в вершине «2». Результаты попытки представлены на рисунке 5. Структура графа для программы представлена в приложении В.



```
Bad Graph structure at or near node 2
Please, check the description.
Press any key to continue...

Bad Graph structure at or near node 2
Please, check the description.
Press any key to continue...

Bad Graph structure at or near node 2
Please, check the description.
Press any key to continue...
```

Рисунок 5 - Неудачный запуск программы

Выводы.

В результате выполнения данной лабораторной работы были изучены критерии оценивания структурной сложности программ. Была проведена оценка структурной сложности двух программ: соответствующая варианту и из первой лабораторной работы.

По результатам оценки можно сделать заключение, что программный и ручной способы совпадают с точностью до порядка путей.

ПРИЛОЖЕНИЕ А. ПРОГРАММА НА ЯЗЫКЕ СИ

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

float fx (float x)
{
    return (1.0 / x);
}

float romb (float lower, float upper, float tol)
{
    int nx [16];
    float t [136];

    bool done = false;
    bool error = false;
    int pieces = 1;
    nx[1] = 1;
    float delta_x = (upper - lower) / pieces;
    float c = (fx(lower) + fx(upper)) * 0.5;
    t[1] = delta_x * c;
    int n = 1;
    int nn = 2;
    float sum = c;

    float fotom,x;
    int l,i,j,k,nt,ntra;
    do
    {
        n = n+1;
        fotom = 4;
        nx[n] = nn;
        pieces = pieces * 2;
        l = pieces - 1;
        delta_x = (upper - lower) / pieces;

        int ll = (l+1)/2;
        for(int ii = 1; ii <= ll; ii++)
        {
            i = ii * 2 - 1;
            x = lower + i * delta_x;
            sum = sum + fx(x);
        }

        t[nn] = delta_x * sum;

        ntra = nx[n-1];
        k = n-1;

        for(int m = 1; m <= k; m++)
```

```

    {
        j = nn+m;
        nt = nx[n - 1] + m - 1;
        t[j] = (fotom * t[j - 1] - t[nt]) / (fotom-1.0);
        fotom = fotom * 4;
    }

    if (n > 4)
    {
        if (t[nn + 1] != 0.0) {
            if ((fabs(t[ntra+1]-t[nn+1])<=fabs(t[nn+1]*tol))
                || (fabs(t[nn-1]-t[j])<=fabs(t[j]*tol)))
            {
                done = true;
            } else
            if (n>15) {
                done = true;
                error = true;
            }
        }
        nn = j+1;
    } while (!done);

    return (t[j]);
}

int main()
{
    const float tol = 1.0E-4;
    float lower = 1.0;
    float upper = 9.0;
    float sum = romb(lower,upper,tol);

    return 0;
}

```

ПРИЛОЖЕНИЕ Б. СТРУКТУРА ГРАФА ИЗ ФАЙЛА

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

```
Top{1}
```

```
Last{14}
```

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

ПРИЛОЖЕНИЕ В. СТРУКТУРА ГРАФА НА ОСНОВЕ ПРОГРАММЫ

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

```
Top{1}
```

```
Last{14}
```

```
Arcs{
```

```
arc(1, 2);
```

```
arc(2, 3);
```

```
arc(3, 4);
```

```
arc(4, 3);
```

```
arc(4, 5);
```

```
arc(5, 6);
```

```
arc(6, 5);
```

```
arc(6, 7);
```

```
arc(7, 8);
```

```
arc(7, 12);
```

```
arc(8, 12);
```

```
arc(8, 9);
```

```
arc(9, 10);
```

```
arc(9, 11);
```

```
arc(9, 12);
```

```
arc(10, 13);
```

```
arc(11, 13);
```

```
arc(12, 2);
```

```
arc(13, 14);
```

```
}
```