

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. 7304

Шарапенков И.И.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы:

Изучение структурной сложности графовых моделей программ и метрик ее оценки.

Задание:

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, 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:

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

Ход работы

1. Был выбран вариант №18 для первой программы, у которой необходимо определить структурную сложность. Управляющий граф первой программы приведен на Рисунке 1:

ВАРИАНТ

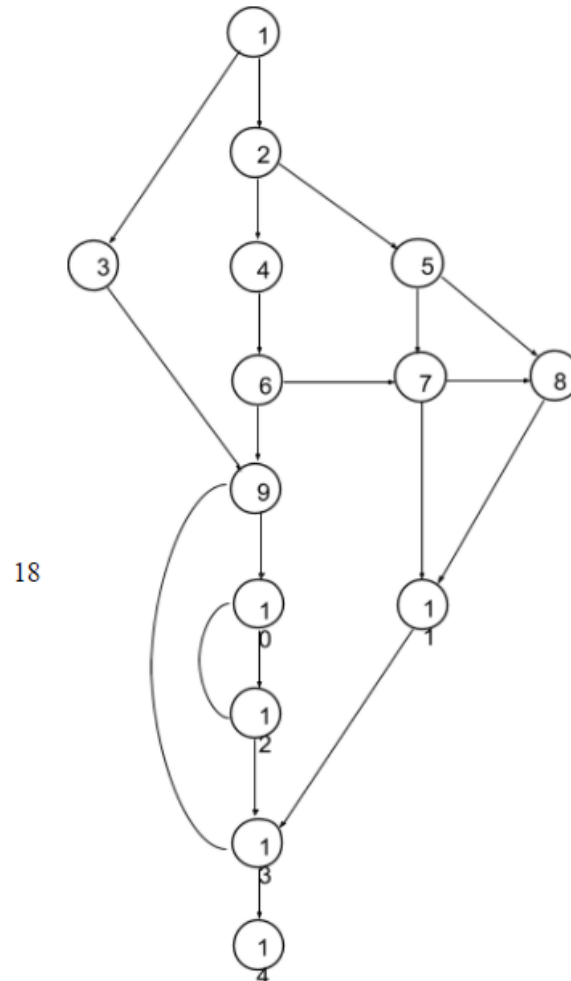


Рисунок 1: Управляющий граф первой программы для определения структурной сложности

2. Была рассчитана структурная сложность первой программы вручную по первому критерию, а именно по минимальному покрытию вершин и дуг графа управления.

М1: 1-3-9-10-12-10-12-13-9-10-12-13-14 (6 ветвлений)

М2: **1-2-4-6-9-10-12-13-14** (5 ветвлений)

М3: **1-2-5-8-11-13-14** (4 ветвления)

М4: **1-2-5-7-8-11-13-14** (5 ветвлений)

М5: **1-2-4-6-7-11-13-14** (5 ветвлений)

Количество маршрутов: $M = 5$

Сложность: $S = 6 + 5 + 4 + 5 + 5 = 25$

3. Было рассчитано вручную цикломатическое число графа первой программы для дальнейшего применения второго критерия. Для данной программы число вершин в графе равно 14, число дуг – 20, число связанных компонент графа 1, тогда: $Z = Y - N + 2 * P = 20 - 14 + 2 * 1 = 8$, то есть цикломатическое число равно 8.

4. Для управляющего графа первой программы вручную были построены 8 линейно-независимых циклов и линейно-независимых маршрутов, после чего была подсчитана структурная сложность по второму критерию.

М1: **10-12** (1 ветвление)

М2: **9-10-12-13** (2 ветвления)

М3: **1-3-9-10-12-13-14** (3 ветвления)

М4: **1-2-4-6-9-10-12-13-14** (5 ветвлений)

М5: **1-2-4-6-7-11-13-14** (5 ветвлений)

М6: **1-2-5-7-8-11-13-14** (5 ветвлений)

М7: **1-2-5-8-11-13-14** (4 ветвления)

М8: **1-2-4-6-7-8-11-13-14** (5 ветвлений)

Сложность: $S = 1 + 2 + 3 + 5 + 5 + 5 + 4 + 5 = 30$

5. Для первой программы была подсчитана структурная сложность по двум критериям с помощью программы ways.exe. Результаты расчета структурной сложности по первому критерию приведены на Рисунке 2, по второму критерию на Рисунке 3:

```

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

Complexity = 25
Press a key...

```

Рисунок 2: Программный расчет структурной сложности первой программы по
первому критерию

```

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

----- Path #6 -----
-> 1 -> 3 -> 9 -> 10 -> 12 -> 13 -> 14
-----Press a key to continue -----

Complexity = 30
Press a key...

```

Рисунок 3: Программный расчет структурной сложности первой программы по
второму критерию

6. Для программы из первой лабораторной работы был составлен управляющий граф. Данный управляющий граф представлен на Рисунке 4:

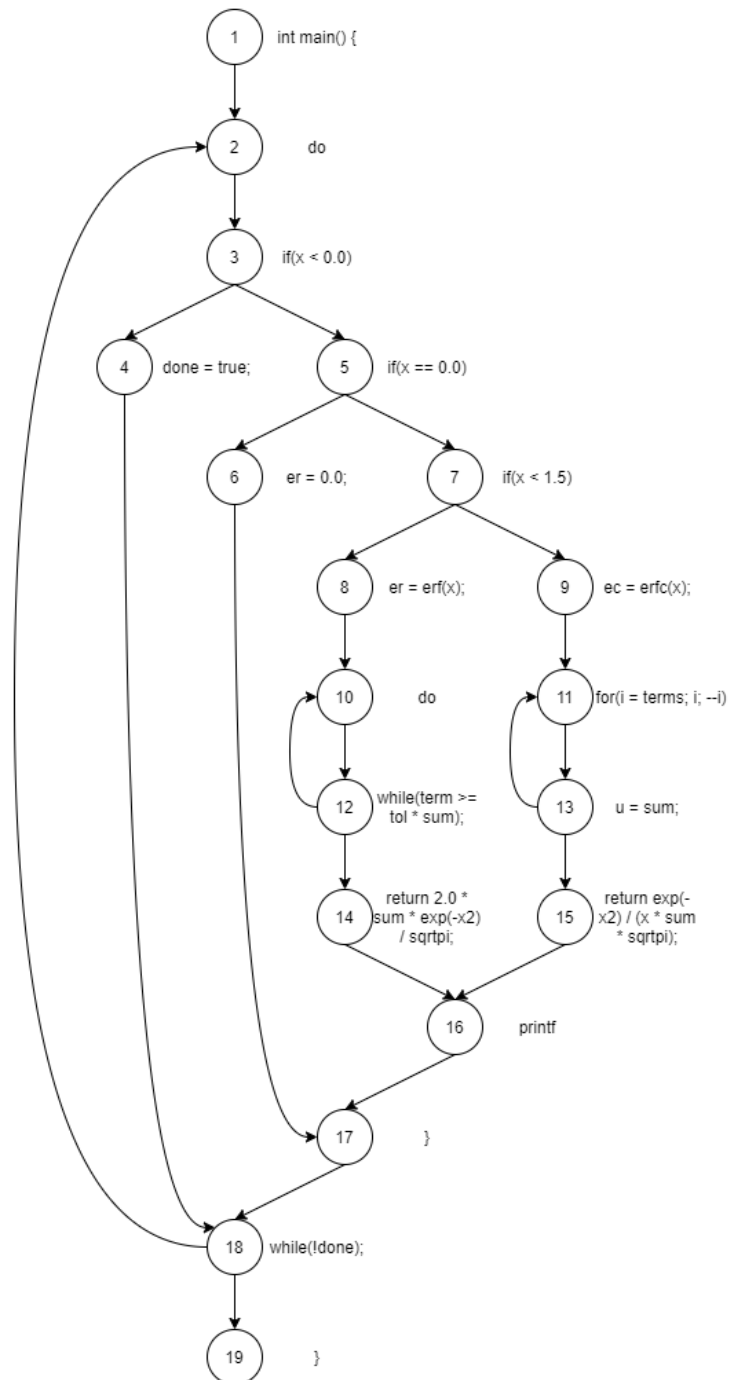


Рисунок 4: Управляющий граф программы из первой лабораторной работы для определения структурной сложности

7. Была рассчитана структурная сложность программы из первой лабораторной работы вручную по первому критерию, а именно по минимальному покрытию вершин и дуг графа управления.

M1:

1-2-3-4-18-2-3-5-6-17-18-2-3-5-7-8-10-12-10-12-14-16-17-18-2-3-5-7-9-11-13-11-13-15-16-17-18-19 (17 ветвлений)

Количество маршрутов: $M = 1$

Сложность: $S = 17$

8. Было рассчитано вручную цикломатическое число графа программы из первой лабораторной работы для дальнейшего применения второго критерия. Для данной программы число вершин в графе равно 19, число дуг – 24, число связных компонент графа 1, тогда: $Z = Y - N + 2 * P = 24 - 19 + 2 * 1 = 7$, то есть цикломатическое число равно 7.

9. Для управляющего графа программы из первой лабораторной работы вручную были построены 7 линейно-независимых цикла и линейно – независимых маршрута, после чего была подсчитана структурная сложность по второму критерию.

M1: 10-12 (1 ветвление)

M2: 11-13 (1 ветвление)

M3: 2-3-4-18 (2 ветвления)

M4: 1-2-3-5-6-17-18-19 (3 ветвления)

M5: 1-2-3-5-7-8-10-12-14-16-17-18-19 (5 ветвлений)

M6: 1-2-3-5-7-9-11-13-15-16-17-18-19 (5 ветвлений)

M7: 1-2-3-4-18-19 (2 ветвления)

Сложность: $S = 1 + 1 + 2 + 3 + 5 + 5 + 2 = 19$

10. Для программы из первой лабораторной работы была подсчитана структурная сложность по двум критериям с помощью программы

ways.exe. Результаты расчета структурной сложности по первому критерию приведены на Рисунке 5, по второму критерию на Рисунке 6.

```
Min ways....
----- Path #1 -----
-> 1 -> 2 -> 3 -> 4 -> 18 -> 2 -> 3 -> 5 -> 6 -> 17 -> 18 -> 2 -> 3 -> 5 -> 7 -> 8 -> 10 -> 12 -> 10 -> 12 -> 14 -> 16 -> 17 -> 18 -> 2 -> 3 -> 5 -> 7 -> 9 -> 11 -> 13 -> 11 -> 13 -> 15 -> 16 -> 17 -> 18 -> 19
-----Press a key to continue -----
Complexity = 17
Press a key...
```

Рисунок 5: Программный расчет структурной сложности программы из первой лабораторной работы по первому критерию

```
Z ways....
----- Path #1 -----
-> 10 -> 12 -> 10
-----Press a key to continue -----
----- Path #2 -----
-> 11 -> 13 -> 11
-----Press a key to continue -----
----- Path #3 -----
-> 2 -> 3 -> 4 -> 18 -> 2
-----Press a key to continue -----
----- Path #1 -----
-> 1 -> 2 -> 3 -> 4 -> 18 -> 19
-----Press a key to continue -----
----- Path #2 -----
-> 1 -> 2 -> 3 -> 5 -> 6 -> 17 -> 18 -> 19
-----Press a key to continue -----
----- Path #3 -----
-> 1 -> 2 -> 3 -> 5 -> 7 -> 8 -> 10 -> 12 -> 14 -> 16 -> 17 -> 18 -> 19
-----Press a key to continue -----
----- Path #4 -----
-> 1 -> 2 -> 3 -> 5 -> 7 -> 9 -> 11 -> 13 -> 15 -> 16 -> 17 -> 18 -> 19
-----Press a key to continue -----
Complexity = 19
Press a key...
```

Рисунок 6: Программный расчет структурной сложности программы из первой лабораторной работы по второму критерию

Выводы.

В ходе выполнения лабораторной работы были изучены методы оценки структурной сложности программы на основе его управляющего графа, была рассчитана структурная сложность двух программ по двум критериям: минимальное покрытие дуг графа и выбор маршрутов на основе цикломатического числа графа. Расчеты были проведены как ручным, так и программным способом.

ПРИЛОЖЕНИЕ А.

КОД ПРОГРАММЫ НА ЯЗЫКЕ С.

```
#INCLUDE <STDIO.H>
#include <MATH.H>

typedef int bool;

#define TRUE 1
#define FALSE 0

double x, er, ec;
bool done;

double erf(double x) {
    const double sqrtpi = 1.7724538;
    const double tol = 1.0E-4;

    double x2, sum, sum1, term;
    int i;

    x2 = x * x;
    sum = x;
    term = x;
    i = 0;
    do {
        i = i + 1;
        sum1 = sum;
        term = 2.0 * term * x2 / (1.0 + 2.0 * i);
        sum = term + sum1;
    } while (term >= tol * sum);
    return 2.0 * sum * exp(-x2) / sqrtpi;
}

double erfc(double x) {
    const double sqrtpi = 1.7724538;
    const int terms = 12;

    double x2, u, v, sum;
    int i;
    x2 = x * x;
    v = 1.0 / (2.0 * x2);
    u = 1.0 + v * (terms + 1.0);
    for (i = terms; i; --i) {
        sum = 1.0 + i * v / u;
        u = sum;
    }
    return exp(-x2) / (x * sum * sqrtpi);
}

int main() {
    bool done = FALSE;
    double x;
    do {
        printf("Arg? ");
        scanf("%lf", &x);
        if (x < 0.0) {
            done = TRUE;
        } else {
            if (x == 0.0) {
                er = 0.0;
            }
        }
    } while (!done);
}
```

```

        EC = 1.0;
    } ELSE {
        IF (X < 1.5) {
            ER = ERF (X);
            EC = 1.0 - ER;
        } ELSE {
            EC = ERFC (X);
            ER = 1.0 - EC;
        }
    }
    PRINTF ("X= %8.4LF ERF= %12.8LF ERFC= %12LF\n", X, ER, EC);
}
} WHILE (!DONE);
}

```