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. 6304 Рыбин А.С.

Преподаватель Кирьянчиков В.А.

Санкт-Петербург

2020

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;
- 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

Выполняется **вариант 14**

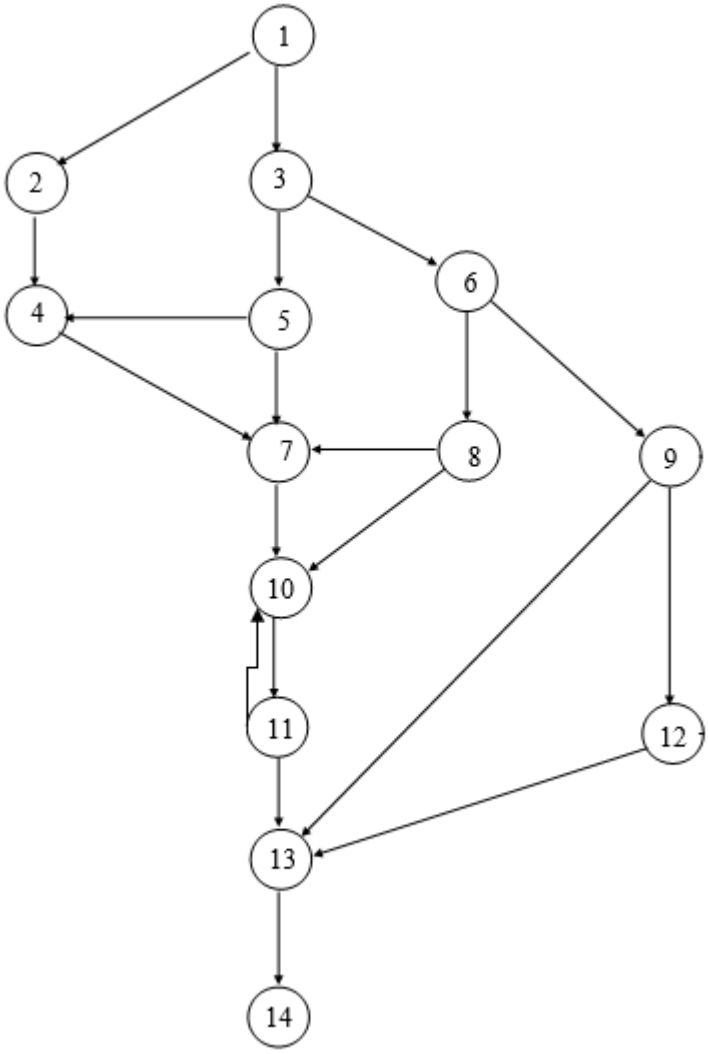

Рисунок 1 – Граф выполнения программы

Над графом проведены следующие модификации:

- Ненаправленная дуга 9-12 удалена, т.к. при направлении 12-9 программа ways.exe сообщает о некорректной структуре графа в вершине 9, а при направлении 9-12 дуга будет дублировать уже существующую.
- Ненаправленная дуга 10-11 ориентирована из 11 в 10, т.к. в противном случае она будет дублировать уже существующую дугу 10-11.

1. Оценивание структурной сложности программы с помощью критерия минимального покрытия дуг графа

1.1. Ручной подсчёт

Ветвления в вершинах: 1, 3, 5, 6, 8, 9, 11.

Минимальный набор путей:

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

Сложность равна **29**.

1.2. Программный расчёт

Граф для программы:

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

Top{1}

Last{14}

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

Минимальный набор путей:

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

Сложность равна **29**.

1.3. Сравнение результатов

Программный результат от ручного отличается двумя маршрутами. Результаты расчёта сложности ручным и программным способом совпадают. Таким образом выбор сложность программы по критерию минимального покрытия дуг не зависит от выбора маршрутов для расчёта при условии выполнения требований критерия для маршрутов.

2. Оценивание структурной сложности первой программы с помощью критерия на основе цикломатического числа.

2.1. Ручной подсчёт

Количество рёбер – 20.

Количество вершин – 14.

До полносвязного графа требуется добавить 1 ребро из вершины 14 в вершину 1.

Цикломатическое число равно = 20 – 14 + 2*1 = 8.

Ветвления в вершинах: 1, 3, 5, 6, 8, 9, 11.

Набор путей:

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

Сложность равна **29**.

2.2. Программный расчёт

Пути:

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

Сложность равна **29**.

2.3. Сравнение результатов

Программный и ручной расчёт полностью совпадают по сложности и выбранным для расчёта маршрутам (за исключением порядка их выбора).

3. Оценивание структурной сложности программы из л/р 1 (алгоритм линеаризации данных) с помощью критерия минимального покрытия дуг графа

В граф дополнительно введены вершины 6 и 12 для корректной работы программы ways.exe.

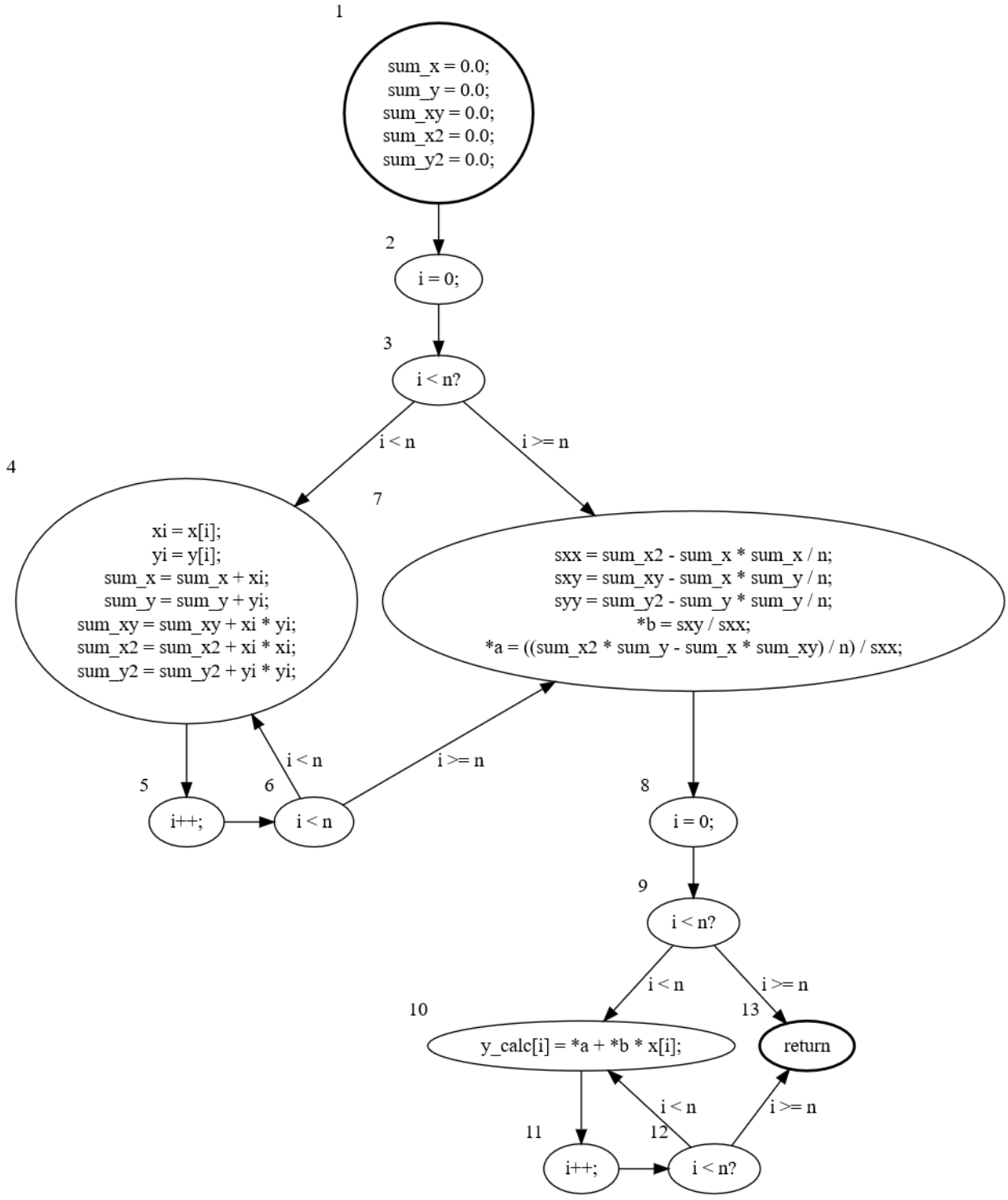

Рисунок 2 – Граф выполнения программы линеаризации данных

3.1. Ручной подсчёт

Ветвления в вершинах: 3, 6, 9, 12.

Минимальный набор путей:

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

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

Сложность равна **10**.

3.2. Программный расчёт

Граф для программы:

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

Top{1}

Last{13}

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

Минимальный набор путей:

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

2) 1-2-3-7-8-9-13.

Сложность равна **8**.

3.3. Сравнение результатов

Программный результат от ручного отличается двумя маршрутами. Сложность, рассчитанная программно на две единицы меньше чем подсчитанная вручную.

4. Оценивание структурной сложности программы из л/р 1 (алгоритм линеаризации данных) с помощью критерия на основе цикломатического числа.

4.1. Ручной подсчёт

Количество рёбер – 16.

Количество вершин – 13.

Для связного графа требуется добавить 1 ребро из вершины №13 в вершину №1.

Цикломатическое число равно = 16 – 13 + 2 * 1 = 5.

Ветвления в вершинах: 3, 6, 9, 12.

Набор путей:

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

Сложность равна **10**.

4.2. Программный расчёт

Пути:

1) 4-5-6-4;
2) 10-11-12-10;
3) 1-2-3-4-5-6-7-8-9-10-11-12-13;
4) 1-2-3-4-5-6-7-8-9-13;
5) 1-2-3-7-8-9-13.

Сложность равна **11**.

4.3. Сравнение результатов

Программный результат от ручного отличается одним маршрутом. Сложность, рассчитанная программно на единицу больше чем подсчитанная вручную.

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.

ПРИЛОЖЕНИЕ А

ГРАФ ИСПОЛНЕНИЯ ПРОГРАММЫ ВАРИАНТ 14 ДЛЯ ПРОГРАММЫ WAYS.EXE

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

Top{1}

Last{14}

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

ПРИЛОЖЕНИЕ Б

ГРАФ ИСПОЛНЕНИЯ ПРОГРАММЫ ЛИНЕАРИЗАЦИИ ДАННЫХ ДЛЯ ПРОГРАММЫ WAYS.EXE

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

Top{1}

Last{13}

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

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ВИЗУАЛИЗАЦИИ ГРАФА ИСПОЛНЕНИЯ ПРОГРАММЫ ЛИНЕАРИЗАЦИИ ДАННЫХ

```
digraph G {
    1[xlabel="1",label="sum_x = 0.0;\nsum_y = 0.0;\nsum_xy = 0.0;\nsum_x2 = 0.0;\nsum_y2 = 0.0;",style=bold];

    2[xlabel="2",label="i = 0;"];

    3[xlabel="3",label="i < n?"];

    4[xlabel="4",label="xi = x[i];\nyi = y[i];\nsum_x = sum_x + xi;\nsum_y = sum_y + yi;\nsum_xy = sum_xy + xi * yi;\nsum_x2 = sum_x2 + xi * xi;\nsum_y2 = sum_y2 + yi * yi;"];

    5[xlabel="5",label="i++;"];

    6[xlabel="6",label="i < n"];

    7[xlabel="7",label="sxx = sum_x2 - sum_x * sum_x / n;\nsxy = sum_xy - sum_x * sum_y / n;\nsyy = sum_y2 - sum_y * sum_y / n;\n*b = sxy / sxx;\n*a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;"];

    8[xlabel="8",label="i = 0;"];

    9[xlabel="9",label="i < n?"];

    10[xlabel="10",label="y_calc[i] = *a + *b * x[i];"];

    11[xlabel="11",label="i++;"];

    12[xlabel="12",label="i < n?"]

    13[xlabel="13",label="return",style=bold];

    1 -> 2;
    2 -> 3;
    3 -> 4 [label="i < n"];
    4 -> 5;
    5 -> 6;
    3 -> 7 [label="i >= n"];
    6 -> 4 [label="i < n"];
    6 -> 7 [label="i >= n"];
    7 -> 8;
    8 -> 9;
    9 -> 10 [label="i < n"];
    9 -> 13 [label="i >= n"];
    10 -> 11;
    11 -> 12;
    12 -> 10 [label="i < n"];
    12 -> 13 [label="i >= n"];


    {rank=same; 4;7}
    {rank=same; 5;6}
    {rank=same; 10;13}
    {rank=same; 11;12}
}
```

ПРИЛОЖЕНИЕ Г
ГРАФ ИСПОЛНЕНИЯ ПРОГРАММЫ ЛИНЕАРИЗАЦИИ ДАННЫХ