**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 |  | Григорьев И.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.

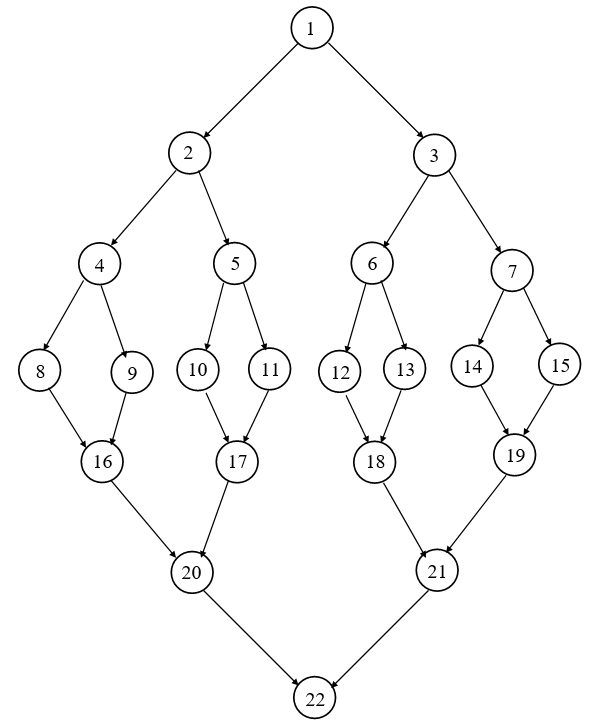
Варианты программ:

* 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.

Вариант 7.



**Ход работы**

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

Ветвления в вершинах 1-7.

Минимальный набор маршрутов:

М1: **1** – **2** – **4** – 8 – 16 – 20 – 22; = 3

М2: **1** – **2** – **4** – 9 – 16 – 20 – 22; = 3

М3: **1** – **2** – **5** – 10 – 17 – 20 – 22; = 3

М4: **1** – **2** – **5** – 11 – 17 – 20 – 22; = 3

М5: **1** – **3** – **6** – 12 – 18 – 21 – 22; = 3

М6: **1** – **3** – **6** – 13 – 18 – 21 – 22; = 3

М7: **1** – **3** – **7** – 14 – 19 – 21 – 22; = 3

М8: **1** – **3** – **7** – 15 – 19 – 21 – 22; = 3

 = 8 \* 3 = 24

* 1. С помощью программы ways.exe

Граф для программы:

Nodes{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22}

Top{1}

Last{22}

Arcs{

arc(1,2);

arc(1,3);

arc(2,4);

arc(2,5);

arc(3,6);

arc(3,7);

arc(4,8);

arc(4,9);

arc(5,10);

arc(5,11);

arc(6,12);

arc(6,13);

arc(7,14);

arc(7,15);

arc(8,16);

arc(9,16);

arc(10,17);

arc(11,17);

arc(12,18);

arc(13,18);

arc(14,19);

arc(15,19);

arc(16,20);

arc(17,20);

arc(18,21);

arc(19,21);

arc(20,22);

arc(21,22);}

Минимальный набор маршрутов:

М1: **1** – **2** – **4** – 8 – 16 – 20 – 22;

М2: **1** – **3** – **6** – 12 – 18 – 21 – 22;

М3: **1** – **2** – **5** – 10 – 17 – 20 – 22;

М4: **1** – **2** – **4** – 9 – 16 – 20 – 22;

М5: **1** – **2** – **5** – 11 – 17 – 20 – 22;

М6: **1** – **3** – **7** – 14 – 19 – 21 – 22;

М7: **1** – **3** – **6** – 13 – 18 – 21 – 22;

М8: **1** – **3** – **7** – 15 – 19 – 21 – 22;

S2 = 24

* 1. Сравнение результатов

Маршруты и сложность ручного и программного расчетов совпали, отличается только порядок нумерации маршрутов.

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

Количество рёбер – 28.

Количество вершин – 22.

Второй критерий рассматривает все маршруты, отличающиеся хотя бы одной дугой или вершиной (базовые маршруты), требует проверки каждого линейно-независимого цикла и каждого линейно-независимого ациклического участка программы. При этом количество проверяемых маршрутов равно цикломатическому числу.

Вычисление цикломатического числа осуществляется по величинам, определяемым по максимально связанному графу. Для превращения исходного графа в граф, у которого любая вершина доступна из любой другой, достаточно добавить одну дугу из конечной вершины № 22 в начальную вершину №1 (P = 1). Цикломатическое число графа:

**Z = Y - N + 2×P =** 28 - 22 + 2×1 = 8

Также цикломатическое число данного графа можно определить путем подсчета числа вершин, в которых происходит ветвление (т.к. программа является правильно структурированной: не имеет циклов с несколькими выходами, переходов внутрь циклов или условных операторов и принудительных выходов из внутренней части циклов или условных операторов).

**Z = nв +1 =** 7 + 1 **=** 8

Набор базовых маршрутов:

М1: **1** – **2** – **4** – 8 – 16 – 20 – 22; = 3

М2: **1** – **2** – **4** – 9 – 16 – 20 – 22; = 3

М3: **1** – **2** – **5** – 10 – 17 – 20 – 22; = 3

М4: **1** – **2** – **5** – 11 – 17 – 20 – 22; = 3

М5: **1** – **3** – **6** – 12 – 18 – 21 – 22; = 3

М6: **1** – **3** – **6** – 13 – 18 – 21 – 22; = 3

М7: **1** – **3** – **7** – 14 – 19 – 21 – 22; = 3

М8: **1** – **3** – **7** – 15 – 19 – 21 – 22; = 3

 = 8 \* 3 = 24

* 1. С помощью программы ways.exe

Маршруты:

М1: **1** – **2** – **4** – 8 – 16 – 20 – 22;

М2: **1** – **2** – **4** – 9 – 16 – 20 – 22;

М3: **1** – **2** – **5** – 10 – 17 – 20 – 22;

М4: **1** – **2** – **5** – 11 – 17 – 20 – 22;

М5: **1** – **3** – **6** – 12 – 18 – 21 – 22;

М6: **1** – **3** – **6** – 13 – 18 – 21 – 22;

М7: **1** – **3** – **7** – 14 – 19 – 21 – 22;

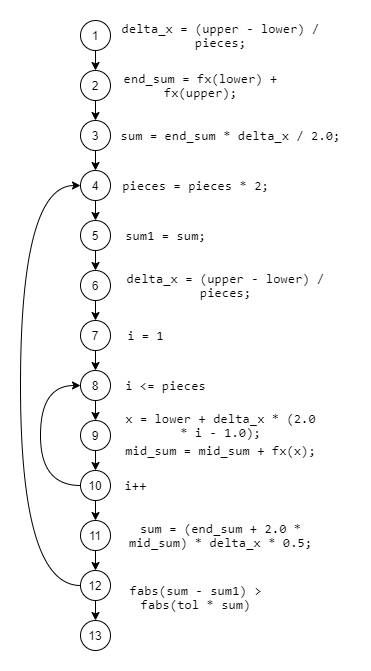
М8: **1** – **3** – **7** – 15 – 19 – 21 – 22;

S2 = 24

* 1. Сравнение результатов.

Маршруты и сложность ручного и программного расчетов совпадают и не отличаются от расчетов минимального покрытия дуг графа. Цикломатическое число графа меньше 10, значит модули легко проверяемы и число ошибок в таких модулях будет минимальным.

1. Оценивание структурной сложности второй программы (из л/р 1) с помощью критерия минимального покрытия дуг графа. Код программы представлен в приложении А. Управляющий граф программы:



* 1. Вручную

Ветвления в вершинах 10, 12.

Минимальный набор маршрутов содержит единственный путь:

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

S2 = 5

* 1. С помощью программы ways.exe

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(6,7);

arc(7,8);

arc(8,9);

arc(9,10);

arc(10,11);

arc(11,12);

arc(12,13);

arc(12,4);

arc(10,8);

}

Минимальный набор маршрутов содержит единственный путь:

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

S2 = 5

* 1. Сравнение результатов.

Сложность ручного и программного расчетов совпадают, маршруты отличаются.

1. Оценивание структурной сложности второй программы (из л/р 1) с помощью критерия на основе цикломатического числа.
   1. Вручную

Количество рёбер – 14.

Количество вершин – 13.

Для превращения исходного графа в максимально связанный достаточно добавить одну дугу из конечной вершины № 13 в начальную вершину №1 (P = 1). Цикломатическое число графа:

**Z = Y - N + 2×P =** 14 - 13 + 2×1 = 3

Также цикломатическое число данного графа можно определить путем подсчета числа вершин, в которых происходит ветвление (т.к. программа является правильно структурированной):

**Z = nв +1 =** 2 + 1 **=** 3

Набор маршрутов:

М1: 8 – 9 – **10** – 8;

М2: 4 – 5 – 6 – 7 – 8 – 9 – **10** – 11 – **12** – 4;

М3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – **10** – 11 – **12** – 13;

S2 = 5

* 1. С помощью программы ways.exe.

Набор маршрутов:

М1: 8 – 9 – **10** – 8;

М2: 4 – 5 – 6 – 7 – 8 – 9 – **10** – 11 – **12** – 4;

М3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – **10** – 11 – **12** – 13;

S2 = 5

* 1. Сравнение результатов.

Сложность ручного и программного расчетов совпадают, маршруты отличаются.

**Выводы**

В ходе выполнения лабораторной работы дана оценка структурной сложности двух программ, вычисленная вручную и с помощью программы ways.exe. Изучены критерии минимального покрытия дуг и выбора маршрутов на основе цикломатического числа управляющего графа программы.

**ПРИЛОЖЕНИЕ А**

**Код программы**

#include <stdio.h>

#include <math.h>

const double tol = 1.0E-6;

double sum, upper, lower;

double fx(double x) {

return 1.0 / x;

}

void trapez(double lower, double upper, double tol) {

int pieces = 1;

double x, delta\_x, end\_sum, mid\_sum, sum1;

delta\_x = (upper - lower) / pieces;

end\_sum = fx(lower) + fx(upper);

sum = end\_sum \* delta\_x / 2.0;

// printf(" 1 %.20f\n", sum);

mid\_sum = 0.0;

do {

pieces = pieces \* 2;

sum1 = sum;

delta\_x = (upper - lower) / pieces;

for (int i = 1; i <= pieces / 2; i++)

{

x = lower + delta\_x \* (2.0 \* i - 1.0);

mid\_sum = mid\_sum + fx(x);

}

sum = (end\_sum + 2.0 \* mid\_sum) \* delta\_x \* 0.5;

// printf(" %i %.20f\n", pieces, sum);

} while (fabs(sum - sum1) > fabs(tol \* sum));

}

int main() {

lower = 1.0;

upper = 9.0;

trapez(lower, upper, tol);

//printf("area = %.20f", sum);

}