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 _____ Дементьев М.Е.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Задание

Выполнить оценивание структурной сложности двух программ с помощью критериев:

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

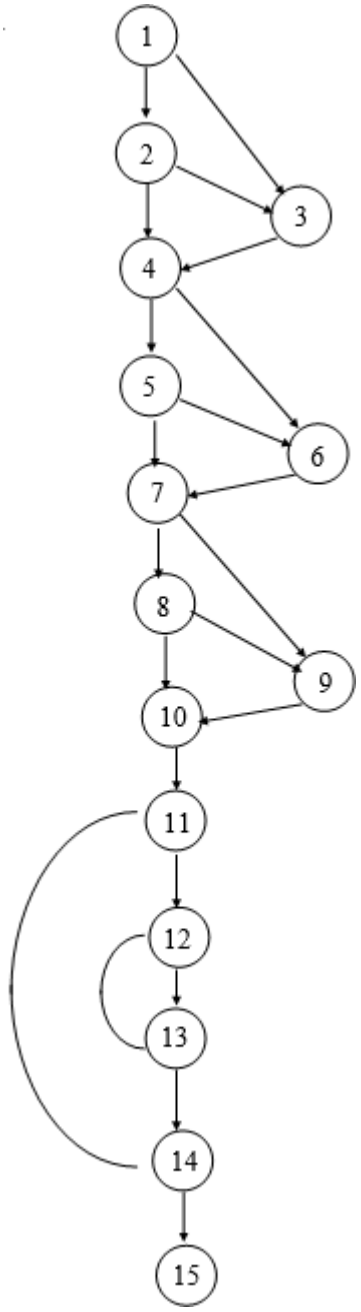
Варианты программ:

- 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 4.



Ход работы

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

1.1. Вручную

Ветвления в вершинах 1,2,4,5,7,8,13,14.

Минимальный набор маршрутов:

M1: **1** – **2** – **4** – **5** – **7** – **8** – 10 – 11 – 12 – **13** – 12 – **13** – **14** – 11 – 12 – **13** – **14** – 15 ; = 11

М2: **1** – **2** – 3 – **4** – **5** – 6 – **7** – **8** – 9 – 10 – 11 – 12 – **13** – **14** – 15 ; = 8
М3: **1** – 3 – **4** – 6 – **7** – 9 – 10 – 11 – 12 – **13** – **14** – 15 ; = 5

$$\mathbf{S}_2 = \sum_{i=1}^{M} \xi_i = 24$$

1.2. С помощью программы ways.exe

Граф для программы:

```
Nodes{
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
}
Top{1}
Last{15}
Arcs{
arc(1,2);
arc(1,3);
arc(2,4);
arc(2,3);
arc(3,4);
arc(4,5);
arc(4,6);
arc(5,7);
arc(5,6);
arc(6,7);
arc(7,8);
arc(7,9);
arc(8,10);
arc(8,9);
arc(9,10);
arc(10,11);
arc(11,12);
arc(12,13);
arc(13,12);
arc(13,14);
arc(14,11);
arc(14,15);
}
```

Минимальный набор маршрутов:

M1: **1** – **2** – **4** – **5** – **7** – **8** – 10 – 11 – 12 – **13** – 12 – **13** – **14** – 11 – 12 – **13** – **14** – 15 ; = 11

M2: **1** – 3 – **4** – 6 – **7** – 9 – 10 – 11 – 12 – **13** – **14** – 15 ; = 5

M3: **1** – **2** – 3 – **4** – **5** – 6 – **7** – **8** – 9 – 10 – 11 – 12 – **13** – **14** – 15 ; = 8

$S_2 = 24$

1.3. Сравнение результатов

Маршруты и сложность ручного и программного расчетов совпадают, ее совпадает только порядок.

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

2.1. Вручную

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 того, чтобы получить из исходного графа, граф в котором каждая вершина доступна из любой другой, можно добавить ребро из вершины №15 в вершину №1(P = 1). Цикломатическое число графа:

$$\mathbf{Z = Y - N + 2 \times P} = 22 - 15 + 2 \times 1 = 9$$

Для структурированных программ определение цикломатического числа может осуществляться на основе количества вершин, в которых происходит ветвление: $\mathbf{Z = n_в + 1} = 8 + 1 = 9$.

Набор базовых маршрутов:

M1: 12 – **13** – 12; = 1

M2: 11 – 12 – **13** – **14** – 11; = 2

M3: **1** – **2** – **4** – **5** – **7** – **8** – 9 – 10 – 11 – 12 – **13** – **14** – 15; = 8

M4: **1** – **2** – **4** – **5** – **7** – **8** – 10 – 11 – 12 – **13** – **14** – 15; = 8

M5: **1** – **2** – **4** – **5** – **7** – 9 – 10 – 11 – 12 – **13** – **14** – 15; = 7

M6: **1** – **2** – **4** – **5** – 6 – **7** – 9 – 10 – 11 – 12 – **13** – **14** – 15; = 7
M7: **1** – **2** – **4** – 6 – **7** – 9 – 10 – 11 – 12 – **13** – **14** – 15; = 6
M8: **1** – 3 – **4** – 6 – **7** – 9 – 10 – 11 – 12 – **13** – **14** – 15; = 5
M9: **1** – **2** – 3 – **4** – 6 – **7** – 9 – 10 – 11 – 12 – **13** – **14** – 15; = 6

$$\mathbf{S_2} = \sum_{i=1}^{M} \xi_i = 50$$

2.2. С помощью программы ways.exe

Маршруты:

M1: 12 – **13** – 12;
M2: 11 – 12 – **13** – **14** – 11;

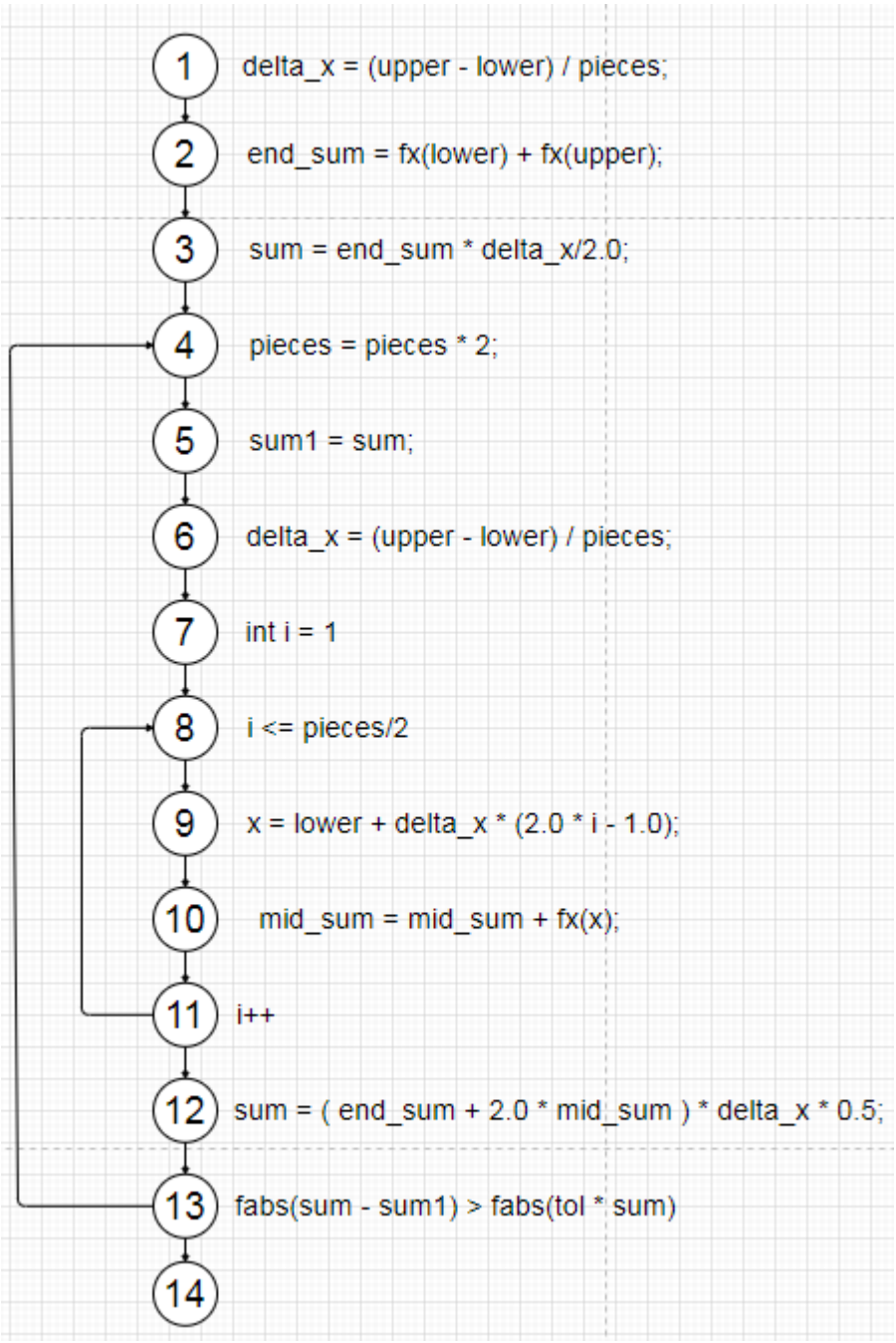
M3: **1** – 2 – **4** – **5** – 7 – **8** – 10 – 11 – 12 – **13** – **14** – 15;
M4: **1** – 2 – **4** – **5** – 7 – **8** – 9 – 10 – 11 – 12 – **13** – **14** – 15;
M5: **1** – 2 – **4** – **5** – 7 – 9 – 10 – 11 – 12 – **13** – **14** – 15;
M6: **1** – 2 – **4** – **5** – 6 – 7 – 9 – 10 – 11 – 12 – **13** – **14** – 15;
M7: **1** – 2 – **4** – 6 – 7 – 9 – 10 – 11 – 12 – **13** – **14** – 15;
M8: **1** – 2 – 3 – **4** – 6 – 7 – 9 – 10 – 11 – 12 – **13** – **14** – 15;
M9: **1** – 3 – **4** – 6 – 7 – 9 – 10 – 11 – 12 – **13** – **14** – 15;

$$S_2 = 50$$

2.3. Сравнение результатов.

Так как цикломатическое число графа меньше 10, то модули легко проверяемы и число ошибок в таких модулях будет минимальным. Маршруты и сложность ручного и программного расчетов совпадают, не совпадает только порядок.

3. 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:



3.1.Вручную

Ветвления в вершинах 11, 13.

Минимальный набор маршрутов содержит единственный путь:

1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – **11** – 8 – 9 – 10 – **11** – 12 – **13** – 4 – 5 – 6 – 7 – 8 – 9 – 10 – **11** – 8 – 9 – 10 – 12 – **13** – 14 (5 ветвлений)

$$S_2 = 5$$

3.2. С помощью программы ways.exe

Граф для программы:

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

```
}
Top{1}
Last{14}
Arcs{
arc(1,2);
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(11,8);
arc(11,12);
arc(12,13);
arc(13,4);
arc(13,14);
}
```

Минимальный набор маршрутов содержит единственный путь:

1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – **11** – 8 – 9 – 10 – **11** – 12 – **13** – 4 – 5 – 6 – 7 – 8 – 9 – 10 – **11** – 8 – 9 – 10 – 12 – **13** – 14 (5 ветвлений)

$$S_2 = 5$$

3.3. Сравнение результатов.

Сложность и маршруты ручного и программного расчетов совпадают.

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

4.1. Вручную

Количество рёбер – 15.

Количество вершин – 14.

Для того, чтобы получить из исходного графа, граф в котором каждая вершина доступна из любой другой, можно добавить ребро из вершины № 14 в начальную вершину №1 (P = 1). Цикломатическое число графа:

$$\mathbf{Z = Y - N + 2 \times P} = 15 - 14 + 2 \times 1 = 3$$

$$\mathbf{Z = n_в + 1} = 2 + 1 = 3$$

Набор маршрутов:

M1: 8 – 9 – 10 – **11** – 8;
M2: 4 – 5 – 6 – 7 – 8 – 9 – 10 – **11** – 12 – **13** – 4;
M3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – **11** – 12 – **13** – 14;

$S_2 = 5$

4.2. С помощью программы ways.exe.

Набор маршрутов:

M1: 8 – 9 – 10 – **11** – 8;
M2: 4 – 5 – 6 – 7 – 8 – 9 – 10 – **11** – 12 – **13** – 4;
M3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – **11** – 12 – **13** – 14;

$S_2 = 5$

4.3. Сравнение результатов.

Сложность и маршруты ручного и программного расчетов совпадают.

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>


double sum = 0.0;
double upper, lower;
const double tol = 1.0E-6;
double fx(double x) {
	return 1.0 / sqrt(x);

}

void trap2(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;
	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;
	} while (fabs(sum - sum1) > fabs(tol * sum));
}

int main() {
	lower = 1.0;
	upper  =  9.0;
	trap2(lower, upper, tol);
}
```