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.

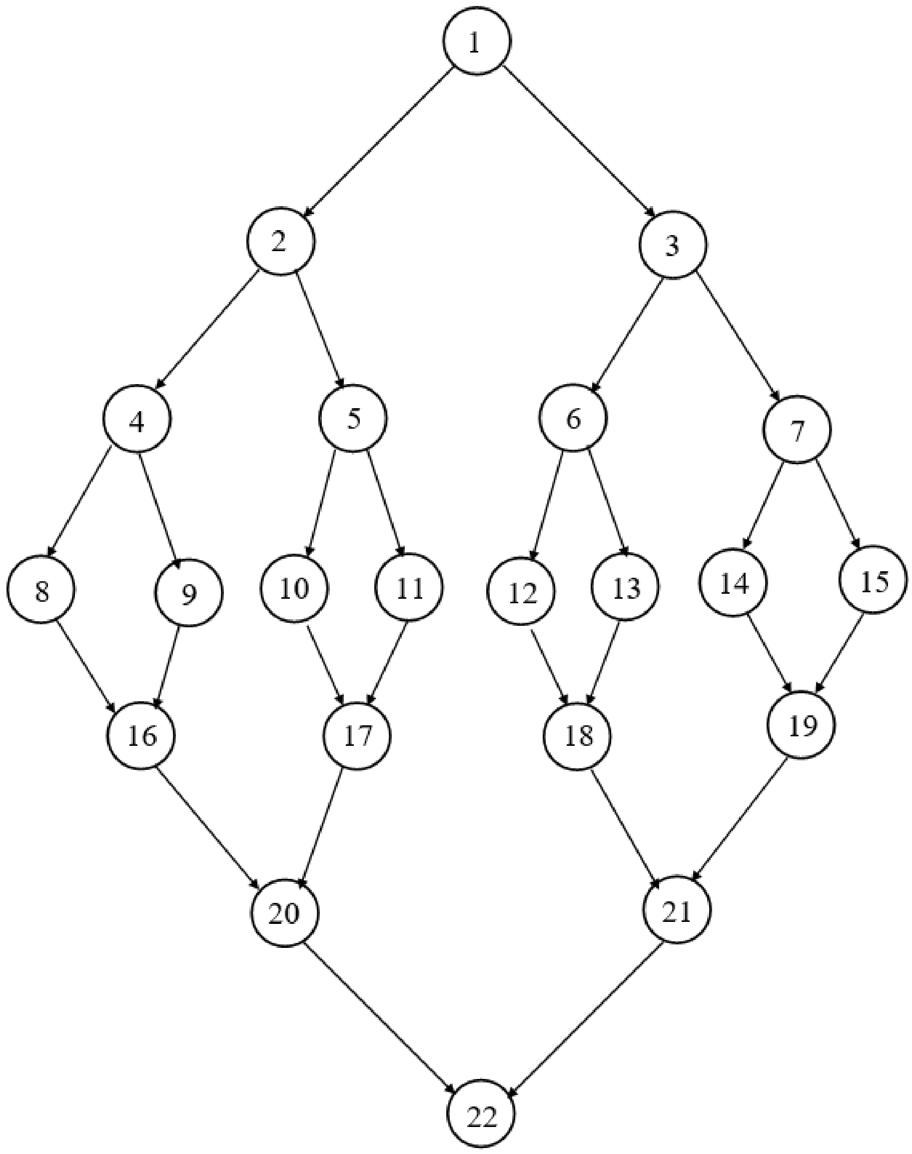
Варианты программ:

- 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.

1.1. Вручную

Ветвления в вершинах 1-7.

Минимальный набор маршрутов:

M1: **<u>1</u>** – **<u>2</u>** – **<u>4</u>** – 8 – 16 – 20 – 22; = 3
M2: **<u>1</u>** – **<u>2</u>** – **<u>4</u>** – 9 – 16 – 20 – 22; = 3
M3: **<u>1</u>** – **<u>2</u>** – **<u>5</u>** – 10 – 17 – 20 – 22; = 3
M4: **<u>1</u>** – **<u>2</u>** – **<u>5</u>** – 11 – 17 – 20 – 22; = 3
M5: **<u>1</u>** – **<u>3</u>** – **<u>6</u>** – 12 – 18 – 21 – 22; = 3

M6: <u>**1**</u> – <u>**3**</u> – <u>**6**</u> – 13 – 18 – 21 – 22; = 3
M7: <u>**1**</u> – <u>**3**</u> – <u>**7**</u> – 14 – 19 – 21 – 22; = 3
M8: <u>**1**</u> – <u>**3**</u> – <u>**7**</u> – 15 – 19 – 21 – 22; = 3

$$\mathbf{S}_2 = \sum_{\mathbf{i=1}}^{\mathbf{M}} \xi_{\mathbf{i}} = 8 * 3 = 24$$

1.2. С помощью программы ways.exe

Граф для программы:

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

M1: **<u>1</u>** – **<u>2</u>** – **<u>4</u>** – 8 – 16 – 20 – 22;
M2: **<u>1</u>** – **<u>3</u>** – **<u>6</u>** – 12 – 18 – 21 – 22;
M3: **<u>1</u>** – **<u>2</u>** – **<u>5</u>** – 10 – 17 – 20 – 22;
M4: **<u>1</u>** – **<u>2</u>** – **<u>4</u>** – 9 – 16 – 20 – 22;
M5: **<u>1</u>** – **<u>2</u>** – **<u>5</u>** – 11 – 17 – 20 – 22;
M6: **<u>1</u>** – **<u>3</u>** – **<u>7</u>** – 14 – 19 – 21 – 22;
M7: **<u>1</u>** – **<u>3</u>** – **<u>6</u>** – 13 – 18 – 21 – 22;
M8: **<u>1</u>** – **<u>3</u>** – **<u>7</u>** – 15 – 19 – 21 – 22;

$$S_2 = 24$$

1.3. Сравнение результатов

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

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

2.1. 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:

$$\mathbf{Z = Y - N + 2\times P =} 28 - 22 + 2\times 1 = 8$$

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).

$$\mathbf{Z} = \mathbf{n_в} + \mathbf{1} = 7 + 1 = 8$$

Набор базовых маршрутов:

M1: **1** – **2** – **4** – 8 – 16 – 20 – 22; = 3
M2: **1** – **2** – **4** – 9 – 16 – 20 – 22; = 3
M3: **1** – **2** – **5** – 10 – 17 – 20 – 22; = 3
M4: **1** – **2** – **5** – 11 – 17 – 20 – 22; = 3
M5: **1** – **3** – **6** – 12 – 18 – 21 – 22; = 3
M6: **1** – **3** – **6** – 13 – 18 – 21 – 22; = 3
M7: **1** – **3** – **7** – 14 – 19 – 21 – 22; = 3
M8: **1** – **3** – **7** – 15 – 19 – 21 – 22; = 3

$$\mathbf{S}_2 = \sum_{\mathbf{i}=1}^{\mathbf{M}} \xi_{\mathbf{i}} = 8 * 3 = 24$$

2.2. С помощью программы ways.exe

Маршруты:

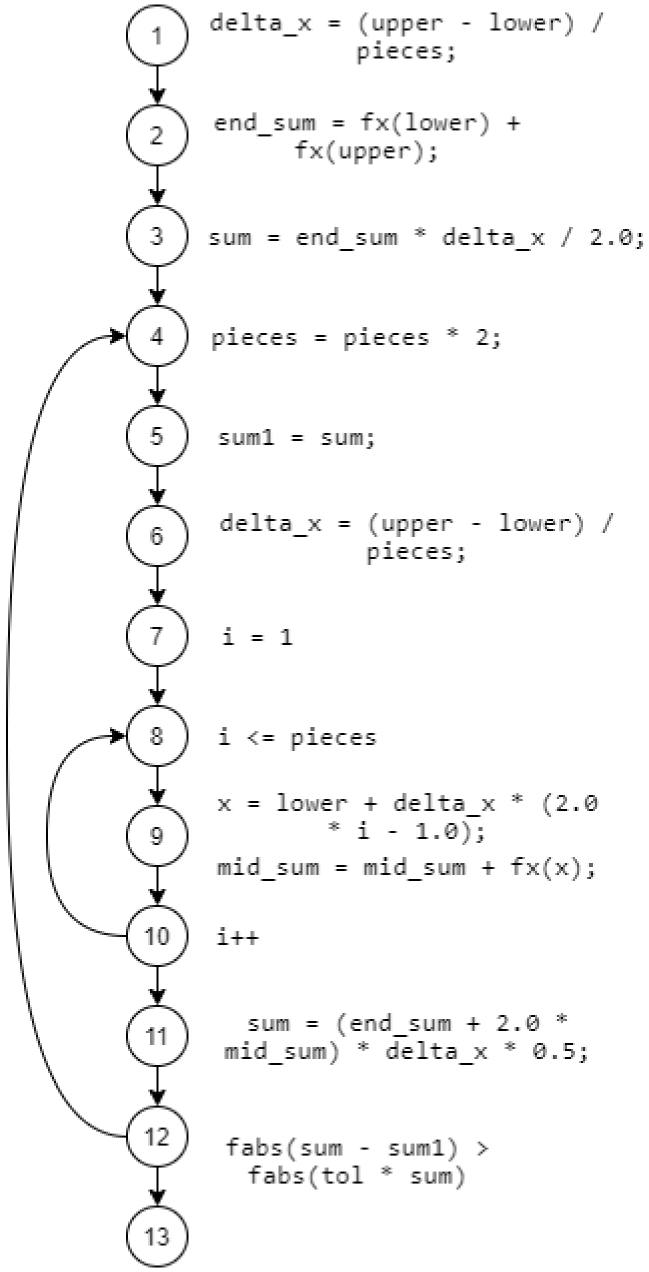
M1: **1** – **2** – **4** – 8 – 16 – 20 – 22;
M2: **1** – **2** – **4** – 9 – 16 – 20 – 22;
M3: **1** – **2** – **5** – 10 – 17 – 20 – 22;
M4: **1** – **2** – **5** – 11 – 17 – 20 – 22;
M5: **1** – **3** – **6** – 12 – 18 – 21 – 22;
M6: **1** – **3** – **6** – 13 – 18 – 21 – 22;
M7: **1** – **3** – **7** – 14 – 19 – 21 – 22;
M8: **1** – **3** – **7** – 15 – 19 – 21 – 22;

$$S_2 = 24$$

2.3.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.

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 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)

$$S_2 = 5$$

3.2. С помощью программы ways.exe

Граф для программы:

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

$$S_2 = 5$$

3.3. Сравнение результатов.

Сложность ручного и программного расчетов совпадают, маршруты отличаются.

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

4.1. 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:

$$\mathbf{Z = Y - N + 2 \times P} = 14 - 13 + 2 \times 1 = 3$$

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

$$\mathbf{Z = n_в + 1} = 2 + 1 = 3$$

Набор маршрутов:

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

$$S_2 = 5$$

4.2. С помощью программы ways.exe.

Набор маршрутов:

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

$$S_2 = 5$$

4.3.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);
}
```