**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 |  | Абдульманов Э.М |
| 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 №1 для первой программы, у которой необходимо определить структурную сложность. Управляющий граф первой программы приведен на Рисунке 1:

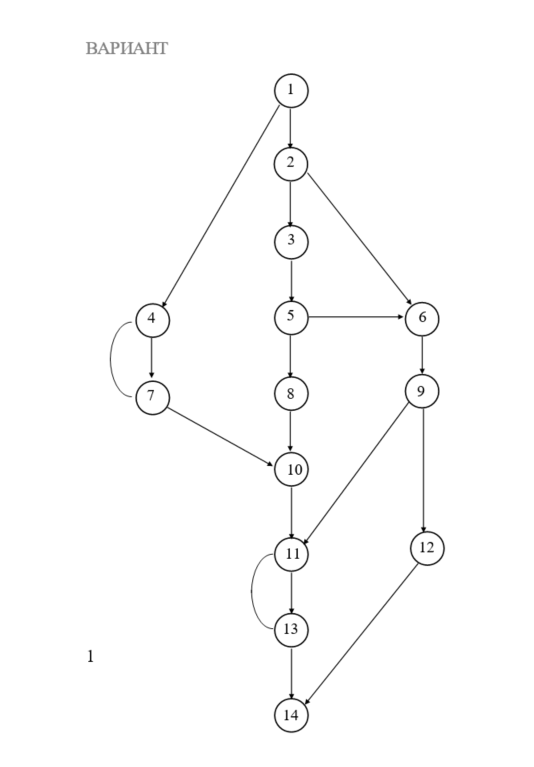


Рисунок 1: Управляющий граф первой программы для определения структурной сложности

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

M1: **1-2-**3-**5**-6-**9**-11-**13**-11-**13**-14 (6 ветвлений)

M2: **1**-**2**-6-**9**-12-14 (3 ветвления)

M3: **1**-4-**7**-4-**7**-10-11-**13**-14 (4 ветвления)

М4: **1**-**2**-3-**5**-8-10-11-**13**-14 (4 ветвления)

Количество маршрутов:

Сложность:

1. Было рассчитано вручную цикломатическое число графа первой программы для дальнейшего применения второго критерия. Для данной программы число вершин в графе равно 14, число дуг – 19, число связных компонент графа 1, тогда: , то есть цикломатическое число равно 7.
2. 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: **1**-**2**-3-**5**-6-**9**-11-**13**-14 (5 ветвлений)

М2: **1-2**-3-**5**-6-**9**-12-14 (4 ветвления)

М3: **1-2**-6-**9**-12-14 (3 ветвления)

М4: **1**-4-7-10-11-**13**-14 (3 ветвления)

М5: **1-2**-3-**5**-8-10-11-**13**-14 (4 ветвления)

М6: 11-**13**-11 (1 ветвление)

М7: 4-**7**-4 (1 ветвление)

Сложность:

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

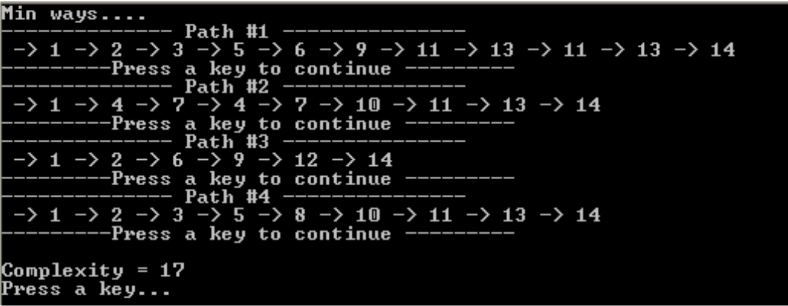


Рисунок 2: Программный расчет структурной сложности первой программы по первому критерию

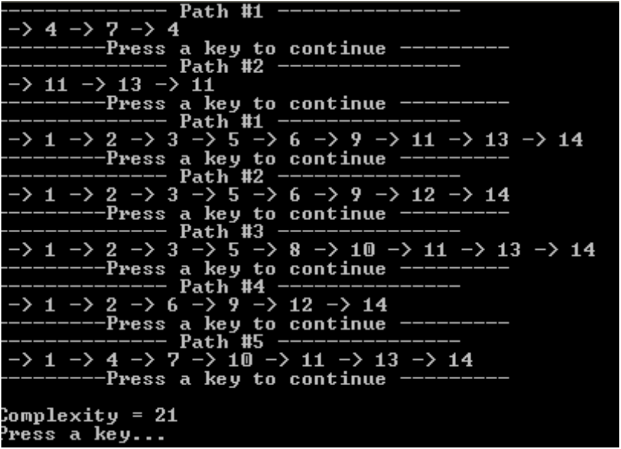


Рисунок 3: Программный расчет структурной сложности первой программы по второму критерию

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

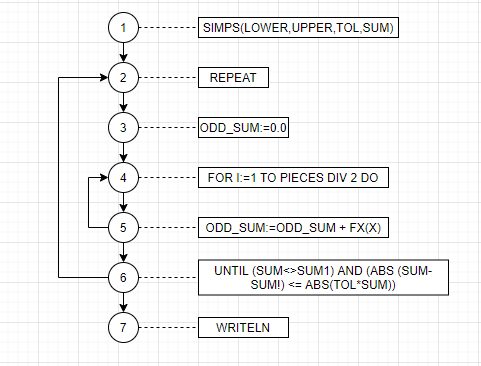


Рисунок 4: Управляющий граф программы из первой лабораторной работы для определения структурной сложности

1. 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-**5**-4-**5-6**-2-3-4-**5-6**-7 (5 ветвлений)

Количество маршрутов:

Сложность:

1. Было рассчитано вручную цикломатическое число графа программы из первой лабораторной работы для дальнейшего применения второго критерия. Для данной программы число вершин в графе равно 7, число дуг – 8, число связных компонент графа 1, тогда: , то есть цикломатическое число равно 3.
2. Для управляющего графа программы из первой лабораторной работы вручную были построены 3 линейно-независимых цикла и линейно – независимых маршрута, после чего была подсчитана структурная сложность по второму критерию.

M1: 1-2-3-4-**5-6**-7 (2 ветвления)

M2: 4-**5**-4 (1 ветвление)

M3: 2-3-4-**5-6**-2 (2 ветвления)

Сложность:

1. 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.

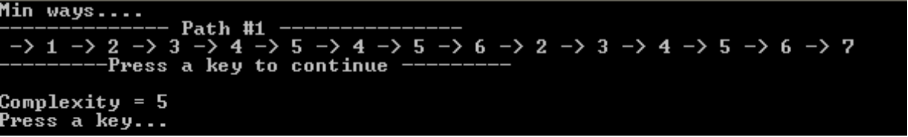


Рисунок 5: Программный расчет структурной сложности программы из первой лабораторной работы по первому критерию

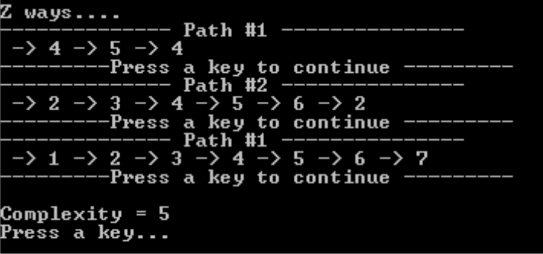


Рисунок 6: Программный расчет структурной сложности программы из первой лабораторной работы по второму критерию

**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 двух критериям: минимальное покрытие дуг графа и выбор маршрутов на основе цикломатического числа графа. Расчеты были проведены как ручным, там и программным способом.

Приложение А.

Код программы на языке Паскаль.

program simp1;

{ integration by Simpson's method }

const tol = 1.0E-6;

var sum,upper,lower : real;

function fx(x: real): real;

begin

fx:=exp(-x/2)

end;

{ function fx }

function dfx(x: real): real;

begin

dfx:=-(exp(-x/2))/2

end;

{ function dfx }

procedure simps(lower,upper,tol : real; var sum : real);

{ numerical integration by Simpson's rule }

{ function is fx, limits are lower and upper }

{ with number of regions equal to pieces }

{ partition is delta\_x, answer is sum }

var i : integer;

x,delta\_x,even\_sum, odd\_sum,end\_sum, end\_cor,sum1 : real;

pieces : integer;

begin

pieces:=2;

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

odd\_sum:=fx(lower+delta\_x);

even\_sum:=0.0;

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

end\_cor:=dfx(lower)-dfx(upper);

sum:=(end\_sum+4.0\*odd\_sum)\*delta\_x/3.0;

repeat

pieces:=pieces\*2;

sum1:=sum;

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

even\_sum:=even\_sum+odd\_sum;

odd\_sum:=0.0;

for i:=1 to pieces div 2 do

begin

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

odd\_sum:=odd\_sum+fx(x)

end; sum:=(7.0\*end\_sum+14.0\*even\_sum+16.00\*odd\_sum+end\_cor\*delta\_x)\*delta\_x/15.0;

until (sum<>sum1) and (abs(sum-sum1)<=abs(tol\*sum))

end; { simps }

begin { main program }

lower:=1.0;

upper:=9.0;

simps(lower,upper,tol,sum);

writeln;

end.