2 Лабораторная работа №2
Исследование способов структурного тестирования программного обеспечения

2.1. Цель работы

Исследовать основные подходы к структурному тестированию программного обеспечения. Приобрести практические навыки построения графа потоков управления и определения независимых ветвей программы.

2.2. Постановка задачи

Был выдан вариант 14.

Варианты заданий соответствуют заданиям по лабораторной работе No1. По варианту задаются требования к программам. Для каждой из них необходимо: 1) написать программу, выполняющую заданные действия; 2) построить граф потоков управления; 3) вычислить цикломатическое число для построенного графа потоков управления; 4) определить независимые ветви программы.

2.3. Ход выполнения работы

2.3.1.Граф потоков управления

По составленной в прошлой лабораторной работе программе были составлены графы потоков управления (рисунки 2.1 и 2.2).

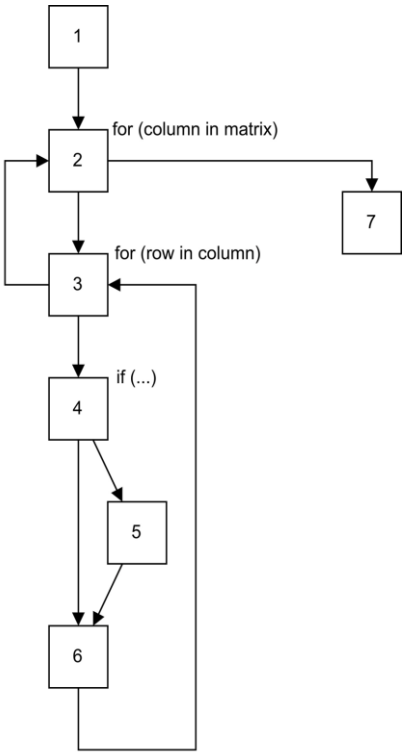

Рисунок 2.1 — Граф потоков управления для программы в задании 1

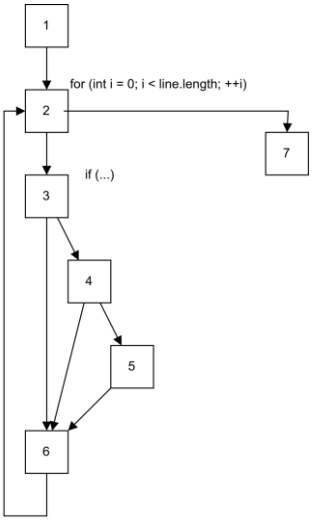

Рисунок 2.2 — Граф потоков управления для программы в задании 3

Для задачи 2 невозможно составить граф по причине того, что метод, выдающий конечный результат, состоит из вызова одной стандартной функции.

2.3.2.Результаты расчета цикломатического числа

Для первой задачи цикломатическое число рассчитано в соответствии с формулой (2.1).

$$C(G_1) = 9 - 7 + 2 = 4 \quad (2.1)$$

Для третьей задачи — по формуле (2.2).

$$C(G_3) = 9 - 7 + 2 = 4 \quad (2.2)$$

2.3.3.Независимые ветви управления

Для первой задачи были получены следующие независимые ветви:

1. 1, 2, 7;
2. 1, 2, 3, 2, 7;
3. 1, 2, 3, 4, 5, 6, 3, 2, 7;
4. 1, 2, 3, 4, 6, 3, 2, 7.

Для третьей задачи получены следующие независимые ветви:

1. 1, 2, 7;
2. 1, 2, 3, 6, 2, 7;
3. 1, 2, 3, 4, 6, 2, 7;
4. 1, 2, 3, 4, 5, 6, 2, 7.

Выводы

При выполнении данной лабораторной работы были исследованы способы тестирования программного обеспечения, а именно тестирование ветвей. Были получены практические навыки нахождения цикломатического числа, позволяющего вычислить количество независимых ветвей, построения графа потоков управления, позволяющего найти эти независимые ветви.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое граф потоков управления?

Граф потока управления (англ. control flow graph, CFG) — в теории компиляции – множество всех возможных путей исполнения программы, представленное в виде графа.

Метод тестирования ветвей основывается на графе потоков управления программы. Этот граф представляет собой скелетную модель всех ветвей программы. Граф потоков управления состоит из узлов, соответствующих ветвлениям решений, и дуг, показывающих поток управления. Если в 1 программе нет операторов безусловного перехода, то создание графа — достаточно простой процесс. При построении графа потоков все последовательные операторы (операторы присвоения, вызова процедур и ввода-вывода) можно проигнорировать. Каждое ветвление операторов условного перехода (if – then – else или case) представлено отдельной ветвью, а циклы обозначаются стрелками, концы которых замкнуты на узле с условием цикла.

2. Назовите основные принципы структурного тестирования ПО?

Цель структурного тестирования — удостовериться, что каждая независимая ветвь программы выполняется хотя бы один раз. Независимая ветвь программы — это ветвь, которая проходит, по крайней мере, по одной новой дуге графа потоков. В терминах программы это означает ее выполнение при новых условиях. Если все независимые ветви выполняются, то можно быть уверенным в том, что:

1)каждый оператор выполняется по крайней мере один раз;

2)каждая ветвь выполняется при условиях, принимающих как истинные, так и ложные значения.

3. Как вычисляется цикломатическое число графа?

Количество независимых ветвей в программе можно определить, вычислив цикломатическое число графа потоков управления программы. Цикломатическое число любого связанного графа G вычисляется по формуле: C (G) = количество дуг - количество узлов + 2.

4. Дайте определение независимых ветвей программы?

Независимая ветвь программы – это ветвь, которая проходит по крайней мере по одной новой дуге графа потоков. Независимые ветви программы – это ветви выполнения программы, которые не зависят от других ветвей выполнения в том смысле, что изменения в одной ветви не повлияют на другие ветви. Каждая ветвь программы может быть выполнена независимо от других ветвей, и результаты ее выполнения не влияют на результаты выполнения других ветвей.

Например, в условном операторе if-else две ветви кода будут независимыми, если выполнение одной из них не зависит от значения условия, проверяемого в другой ветви. Если выполнение одной ветви зависит от выполнения другой ветви, то эти ветви не будут независимыми.