Практическое занятие №15
«Исследование программного кода на предмет ошибок и отклонения от алгоритма»

Цели: Научиться исследовать программный код на предмет ошибок и отклонения от алгоритма.

Теоретические сведения:

При проверке правильности программ и систем рассматриваются процессы верификации, валидации и тестирования ПС, которые регламентированы в стандарте ISO/IEC 12207 жизненного цикла ПО. В некоторой зарубежной литературе процессы верификации и тестирования отождествляются.

Тестирование - это процесс обнаружения ошибок в ПО путем исполнения выходного кода ПС на тестовых данных.

Тестирование можно рассматривать, как процесс семантической отладки (проверки) программы, заключающийся в исполнении последовательности различных наборов контрольных тестов, для которых заранее известен результат. Т.е. тестирование предполагает выполнение программы и получение конкретных результатов выполнения тестов.

Цель тестирования - проверка работы реализованных функций в соответствии с их спецификацией. Методы функционального тестирования подразделяются на статические и динамические.

Динамические методы тестирования используются в процессе выполнения программ. Они базируются на графе, связывающем причины ошибок с ожидаемыми реакциями на эти ошибки. В процессе тестирования накапливается информация об ошибках, которая используется при оценке надежности и качества ПС.

Цель динамического тестирования программ по принципу "черного ящика" - выявление одним тестом максимального числа ошибок с использованием небольшого подмножества возможных входных данных.

Методы "черного ящика" обеспечивают:

· эквивалентное разбиение;

· анализ граничных значений;

· применение функциональных диаграмм, которые в соединении с реверсивным анализом дают достаточно полную информацию о функционировании тестируемой программы.

"Белый ящик" базируется на структуре программы, в случае "черного ящика", о структуре программы ничего неизвестно. Для выполнения тестирования с помощью этих "ящиков" известными считаются выполняемые функции, входы (входные данные) и выходы (выходные данные), а также логика обработки, представленные в документации.

Особенность тестировании программ как «черный ящик» заключается в следующем:

Известны: функции программы.

Исследуется: работа каждой функции на всей области определения.

Как показано на рисунке, основное место приложения тестов «черного ящика» – интерфейс ПО.

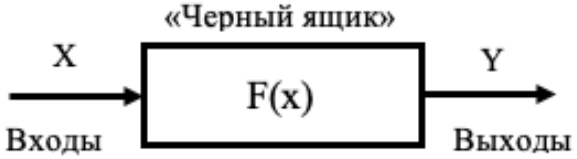

Рисунок – Тестирование «черного ящика»

Эти тесты демонстрируют:

- как выполняются функции программ;
- как принимаются исходные данные;
- как вырабатываются результаты;
- как сохраняется целостность внешней информации.

При тестировании «черного ящика» рассматриваются системные характеристики программ, игнорируется их внутренняя логическая структура.

Методы черного ящика

1. Метод эквивалентных разбиений

Его основу составляют 2 положения:

- Каждый тип должен включать столько различных входных и выходных условий, сколько необходимо для того, что бы минимизировать общее число необходимых тестов.
- Необходимо разбивать входную область программы на конечное число классов эквивалентности

Класс эквивалентности выделяется путем выбора каждого входного условия и разбиением его на две или более групп.

2. Анализ граничных решений (АГР)

Граничные условия – ситуация, возникающая непосредственно на границе, выше или ниже границ входных или выходных элементов класса эквивалентности (КЭ)

АГР предполагает:

- Выбор любого элемента в КЭ в качестве представительного при АГЗ осуществляется таким образом, чтобы проверить тестом каждую границу этого класса.
- При разработке тестов рассматриваются не только входные условия, но и выходные.

3. Метод функциональных диаграмм

Недостатком метод граничных решений и метод эквивалентных разбиений является то, что они не исследуют комбинации входных условий.

Метод функциональных диаграмм помогает создать высоко результирующие тесты.

Функциональная диаграмма представляет собой формальный язык, на который транслируется спецификация, написанная на естественном языке.

Построение тестов осуществляется в несколько этапов:

- спецификация разбивается на несколько участков

• спецификация определяется причиной и следствием

Причины и следствия определяются путем последовательного чтения спецификации, каждой причине и следствию присваивается отдельный номер.

Графы причинно-следственных связей

Диаграммы причинно-следственных связей используются для проектирования тестовых вариантов и обеспечивают формальную запись логических условий и соответствующих действий. Данный способ является разновидностью тестирования «черного ящика». Используется автоматный подход к решению задачи.

На первом шаге способа тестирования, основанного на построении диаграмм причинно-следственных связей, для тестируемой программы (или отдельного тестируемого модуля) перечисляются причины (условия ввода или классы эквивалентности условий ввода) и следствия (действия или условия вывода). Каждой причине и следствию присваивается свой идентификатор.

На втором шаге данного способа тестирования разрабатывается **граф причинно-следственных связей**.

Введем нотацию базовых символов для записи графов причин и следствий. Причины будем обозначать символами c_i, а следствия — символами e_i. Каждый узел графа может находиться в состоянии 0 (состояние отсутствует) или 1 (состояние присутствует).

Функция «тождество» (рис. 1) устанавливает, что если значение есть 1, то и значение есть 1. В противном случае значение есть 0.

Рисунок 1 – Функция «тождество»

Функция «не» (рис. 2) устанавливает, что если значение c_1 есть 1, то значение e_1 есть 0. В противном случае значение есть 1.

Рис. 2 Функция «не»

Функция «или» (рис. 3) устанавливает, что если c_1 или c_2 есть 1, то e_1 есть 1. В противном случае e_1 есть 0.

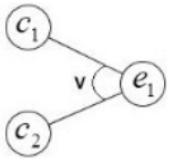

Рис. 3 Функция «или»

Функция «и» (рис. 4) устанавливает, что если и c_1, и c_2 есть 1, то e_1 есть 1. В противном случае есть 0.

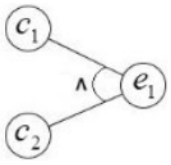

Рис. 4 Функция «и»

На третьем шаге рассматриваемого способа тестирования граф преобразуется в таблицу решений. Порядок генерации таблицы решений [1]:

1) Выбирается некоторое следствие, которое должно быть в состоянии «1».

2) Находятся все комбинации причин (с учетом ограничений), которые устанавливают это следствие в состояние «1». Для этого из следствия прокладывается обратная трасса через граф.

3) Для каждой комбинации причин, приводящих следствие в состояние «1», строится один столбец.

4) Для каждой комбинации причин доопределяются состояния всех других следствий. Они помещаются в тот же столбец таблицы решений.

5) Действия 1–4 повторяются для всех следствий графа. На четвертом шаге данного способа тестирования столбцы таблицы решений преобразуются в тестовые варианты.

Практическое задание:

Порядок выполнения работы:

1. Ознакомиться с теоретическими сведениями по стратегиям тестирования.

2. В соответствии с вариантом задачи, подготовить тесты по методикам стратегии "черного ящика".

3. Предлагаемые тесты свести в таблицу.

Номер теста	Назначение теста	Значения исходных данных	Ожидаемый результат	Реакция программы	Вывод

4. Разработать программу.

5. Выполнить тестирование. Занести в таблицу результаты.

6. Сделать вывод о роли тестирования с использованием стратегии "черного ящика" и возможностях его применения. Сформулировать его достоинства и недостатки.

Варианты заданий:

Задача 1. Разработать программу решения уравнения $ax^2 + bx + c = 0$, где a, b, c - любые вещественные числа.

Задача 2. Разработать программу определения вида треугольника, заданного длинами его сторон: равносторонний, равнобедренный, прямоугольный, разносторонний.

Задача 3. Разработать программу определения вида четырехугольника, заданного координатами вершин на плоскости: квадрат, прямоугольник,

параллелограмм, ромб, равнобедренная трапеция, прямоугольная трапеция, трапеция общего вида, четырехугольник общего вида.

Контрольные вопросы:

1. Назовите формальные методы проверки правильности программ.
2. Какие функции у процесса верификации программ?
3. Назовите основные задачи процесса валидации программ.
4. В чем отличие верификации и валидации?
5. Объясните значения терминов "черный ящик", "белый ящик".