**ЛАБОРАТОРНАЯ РАБОТА №6**

«Тестирование программ методом «черного ящика»

**Цель работы.** Отработать навыки составления и тестирования программ как «черного ящика».

# Теоретическая часть

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.

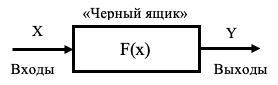


Рисунок 6.1 – Тестирование «черного ящика»

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 ci, а следствия — символами ej. Каждый узел графа может находиться в состоянии 0 (состояние отсутствует) или 1 (состояние присутствует).

Функция «тождество» (рис. 6.2) устанавливает, что если значение есть 1, то и значение есть 1. В противном случае значение есть 0.



Рисунок 6.2 – Функция «тождество»

Функция «не» (рис. 6.3) устанавливает, что если значение c1 есть 1, то значение e1 есть 0. В противном случае значение есть 1.



Рис. 6.3 Функция «не»

Функция «или» (рис. 6.4) устанавливает, что если c1 или c2 есть 1, то e1 есть 1. В противном случае e1 есть 0.

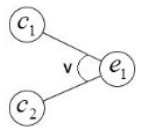


Рис. 6.4 Функция «или»

Функция «и» (рис. 6.5) устанавливает, что если и c1, и c2 есть 1, то e1 есть 1. В противном случае есть 0.

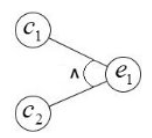


Рис. 6.5 Функция «и»

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:**

Задание

Выполнение работы предусматривает следующую последовательность действий:

1. Определение причин(условий ввода) и следствий;

2. Построение графа причинно-следственных связей;

3. Создание таблиц решений;

4. Построение тестовых вариантов;

5. Оформление результатов тестирования.

В отчет по лабораторной работе входят:

1. Перечень причин и следствий;

2. Граф причинно-следственных связей;

3. Таблица решений;

4. Тестовые варианты;

5. Результаты тестирования.

Варианты задания

Построить таблицу значений функции *y*=*f(x)*, *x* изменяется от *xmin* до *xmax* с шагом *dx*. Проконтролировать правильность ввода *xmin*, *xmax*, *dx* и корректность вычисляемого выражения.

Примечание. В протоколе необходимо указать порядок выполнения операций в соответствии с их приоритетом.

|  |  |
| --- | --- |
| 1. | 2. |
| 3. | 4. |
| 5. | 6. |
| 7. | 8. |
| 9. | 10. |
| 11. | 12. |
| 13. | 14. |
| 15. | 16. |