Лабораторная работа №1

Тема: Стратегия «черного ящика».

Справочные данные.

Одним из способов проверки программ является стратегия тестирования, называемая стратегией "черного ящика". В этом случае программа рассматривается как "черный ящик" и такое тестирование имеет целью выяснения обстоятельств, в которых поведение программы не соответствует спецификации.

Для обнаружения всех ошибок в программе необходимо выполнить исчерпывающее тестирование, т.е. тестирование на всех возможных наборах данных. Для тех же программ, где исполнение команды зависит от предшествующих ей событий, необходимо проверить и все возможные последовательности.

Очевидно, что построение исчерпывающего входного теста для большинства случаев невозможно. Поэтому, обычно выполняется "разумное" тестирование, при котором тестирование программы ограничивается прогонами на небольшом подмножестве всех возможных входных данных. Естественно при этом целесообразно выбрать наиболее подходящее подмножество (подмножество с наивысшей вероятностью обнаружения ошибок).

Правильно выбранный тест подмножества должен обладать следующими свойствами:

1)уменьшать, причем более чем на единицу число других тестов, которые должны быть разработаны для достижения заранее определенной цели «приемлемого» тестирования;

2)покрывать значительную часть других возможных тестов, что в некоторой степени свидетельствует о наличии или отсутствии ошибок до и после применения этого ограниченного множества значений входных данных.

Стратегия "черного ящика" включает в себя следующие методы формирования тестовых наборов:

– эквивалентное разбиение;
– анализ граничных значений;
– анализ причинно-следственных связей;
– предположение об ошибке.

Эквивалентное разбиение.

Основу метода составляют два положения:

1.Исходные данные программы необходимо разбить на конечное число классов эквивалентности, так чтобы можно было предположить, что каждый тест, являющийся представителем некоторого класса, эквивалентен любому другому тесту этого класса. Иными словами, если тест какого-либо класса обнаруживает ошибку, то предполагается, что все другие тесты этого класса эквивалентности тоже обнаружат эту ошибку и наоборот.

2.Каждый тест должен включать по возможности максимальное количество различных входных условий, что позволяет минимизировать общее число необходимых тестов.

Первое положение используется для разработки набора "интересных" условий, которые должны быть протестированы, а второе - для разработки минимального набора тестов.

Разработка тестов методом эквивалентного разбиения осуществляется в два этапа:

– выделение классов эквивалентности;
– построение тестов.

Выделение классов эквивалентности.

Классы эквивалентности выделяются путем выбора каждого входного условия (обычно это предложение или фраза из спецификации) и разбиением его на две или более групп. Для этого используется таблица следующего вида:

Входное условие	Правильные классы эквивалентности	Неправильные классы эквивалентности

Правильные классы включают правильные данные, неправильные классы - неправильные данные.

Выделение классов эквивалентности является эвристическим процессом, однако при этом существует ряд правил:

- Если входные условия описывают область значений (например «целое данное может принимать значения от 1 до 999»), то выделяют один правильный класс $1 \le X \le 999$ и два неправильных $X<1$ и $X>999$.
- Если входное условие описывает число значений (например, «в автомобиле могут ехать от одного до шести человек»), то определяется один правильный класс эквивалентности и два неправильных (ни одного и более шести человек).
- Если входное условие описывает множество входных значений и есть основания полагать, что каждое значение программист трактует особо (например, «известные способы передвижения на АВТОБУСЕ, ГРУЗОВИКЕ, ТАКСИ, МОТОЦИКЛЕ или ПЕШКОМ»), то определяется правильный класс эквивалентности для каждого значения и один неправильный класс (например «на ПРИЦЕПЕ»).
- Если входное условие описывает ситуацию «должно быть» (например, «первым символом идентификатора должна быть буква»), то определяется один правильный класс эквивалентности (первый символ - буква) и один неправильный (первый символ - не буква).
- Если есть любое основание считать, что различные элементы класса эквивалентности трактуются программой неодинаково, то данный класс разбивается на меньшие классы эквивалентности.

Построение тестов.

Этот шаг заключается в использовании классов эквивалентности для построения тестов. Этот процесс включает в себя:

- Назначение каждому классу эквивалентности уникального номера.
- Проектирование новых тестов, каждый из которых покрывает как можно большее число непокрытых классов эквивалентности, до тех пор, пока все правильные классы не будут покрыты (только не общими) тестами.
- Запись тестов, каждый из которых покрывает один и только один из непокрытых неправильных классов эквивалентности, до тех пор, пока все неправильные классы не будут покрыты тестами.

Разработка индивидуальных тестов для неправильных классов эквивалентности обусловлено тем, что определенные проверки с ошибочными входами скрывают или заменяют другие проверки с ошибочными входами.

Недостатком метода эквивалентных разбиения в том, что он не исследует комбинации входных условий.

Анализ граничных значений.

Граничные условия - это ситуации, возникающие на, выше или ниже границ входных классов эквивалентности. Анализ граничных значений отличается от эквивалентного разбиения следующим:

- Выбор любого элемента в классе эквивалентности в качестве представительного при анализе граничных условий осуществляется таким образом, чтобы проверить тестом каждую границу этого класса.
- При разработке тестов рассматриваются не только входные условия (пространство входов), но и пространство результатов.

Применение метода анализа граничных условий требует определенной степени творчества и специализации в рассматриваемой проблеме. Тем не менее, существует несколько общих правил этого метода:

- Построить тесты для границ области и тесты с неправильными входными данными для ситуаций незначительного выхода за границы области, если входное условие описывает область значений (например, для области входных значений от -1.0 до +1.0 необходимо написать тесты для ситуаций -1.0, +1.0, -1.001 и +1.001).
- Построить тесты для минимального и максимального значений условий и тесты, большие и меньшие этих двух значений, если входное условие удовлетворяет дискретному ряду значений. Например, если входной файл может содержать от 1 до 255 записей, то проверить 0,1,255 и 256 записей.
- Использовать правило 1 для каждого выходного условия. Причем, важно проверить границы пространства результатов, поскольку не всегда границы входных областей представляют такой же набор условий, как и границы выходных областей. Не всегда также можно получить результат вне выходной области, но, тем не менее, стоит рассмотреть эту возможность.
- Использовать правило 2 для каждого выходного условия.
- Если вход или выход программы есть упорядоченное множество (например, последовательный файл, линейный список, таблица), то сосредоточить внимание на первом и последнем элементах этого множества.
- Попробовать свои силы в поиске других граничных условий.

Анализ граничных условий, если он применен правильно, является одним из наиболее полезных методов проектирования тестов. Однако следует помнить, что граничные условия могут быть едва уловимы и определение их связано с большими трудностями, что является недостатком этого метода. Второй недостаток связан с тем, что метод анализа граничных условий не позволяет проверять различные сочетания исходных данных.

Анализ причинно-следственных связей.

Метод анализа причинно-следственных связей помогает системно выбирать высокорезультативные тесты. Он дает полезный побочный эффект, позволяя обнаруживать неполноту и неоднозначность исходных спецификаций.

Для использования метода необходимо понимание булевской логики (логических операторов - и, или, не). Построение тестов осуществляется в несколько этапов.

1) Спецификация разбивается на «рабочие » участки, так как таблицы причинно-следственных связей становятся громоздкими при применении метода к большим спецификациям. Например, при тестировании компилятора в качестве рабочего участка можно рассматривать отдельный оператор языка.

2) В спецификации определяются множество причин и множество следствий. Причина есть отдельное входное условие или класс эквивалентности входных условий. Следствие есть выходное условие или преобразование системы. Каждым причине и следствию приписывается отдельный номер.

3) На основе анализа семантического (смыслового) содержания спецификации строится таблица истинности, в которой последовательно перебираются все возможные комбинации причин и определяются следствия каждой комбинации причин. Таблица снабжается примечаниями, задающими ограничения и описывающими комбинации причин и/или следствий, которые являются невозможными из-за синтаксических или внешних ограничений. Аналогично, при необходимости строится таблица истинности для класса эквивалентности.

Примечание. При этом можно использовать следующие приемы:

- По возможности выделять независимые группы причинно-следственных связей в отдельные таблицы.
- Истина обозначается "1". Ложь обозначается "0". Для обозначения безразличных состояний условий применять обозначение "X", которое предполагает произвольное значение условия (0 или 1).

4) Каждая строка таблицы истинности преобразуется в тест. При этом:

- по возможности следует совмещать тесты из независимых таблиц;
- для классов эквивалентности входных условий дополнительно необходимо Недостаток метода - неадекватно исследует граничные условия.

Предположение об ошибке.

Часто программист с большим опытом выискивает ошибки "без всяких методов". При этом он подсознательно использует метод "предположение об ошибке". Процедура метода предположения об ошибке в значительной степени основана на интуиции. Основная идея метода состоит в том, чтобы перечислить в некотором списке возможные ошибки или ситуации, в которых они могут появиться, а затем на основе этого списка составить тесты. Другими словами, требуется перечислить те специальные случаи, которые могут быть не учтены при проектировании.

Пример.

Пусть необходимо выполнить тестирование программы, определяющей точку пересечения двух прямых на плоскости. Попутно, она должна определять параллельность прямой одной их осей координат.

В основе программы лежит решение системы линейных уравнений:

$Ax+By=C$ и $Dx+Ey=F$.

1. Используя **метод эквивалентных разбиений**, получаем для всех коэффициентов один правильный класс эквивалентности (коэффициент - вещественное число) и один неправильный (коэффициент - не вещественное число). Откуда можно предложить 7 тестов:

1) все коэффициенты - вещественные числа;
2) -7) поочередно каждый из коэффициентов - не вещественное число.

2. По **методу граничных условий**:

можно считать, что для исходных данных граничные условия отсутствуют (коэффициенты - "любые" вещественные числа);

для результатов - получаем, что возможны варианты: единственное решение, прямые сливаются (множество решений), прямые параллельны (отсутствие решений). Следовательно, можно предложить тесты, с результатами внутри области:

1) результат - единственное решение ($\delta \neq 0$);
2) результат - множество решений ($\delta = 0$ и $\delta x = \delta y = 0$);
3) результат - отсутствие решений ($\delta = 0$, но $\delta x \neq 0$ или $\delta y \neq 0$);

и с результатами на границе:

1) δ = 0,01;

2) δ = -0,01;

3) δ=0, δx=0,01, δy =0;

4) δ=0, δy=-0,01, δx=0.

3. По **методу анализа причинно-следственных связей**:

Определяем множество условий.

а) для определения типа прямой:

a=0, b=0, c=0 - для определения типа и существования первой прямой

d=0, e=0, f=0 - для определения типа и существования второй прямой

б) для определения точки пересечения:

δ= 0

δx = 0

δy = 0

Выделяем три группы причинно-следственных связей (определение типа и существования первой линии, определение типа и существования второй линии, определение точки пересечения) и строим таблицы истинности.

A=0	B=0	C=0	Результат
0	0	X	прямая общего положения
0	1	0	прямая, параллельная оси OX
0	1	1	ось OX
1	0	0	прямая, параллельная оси OУ
1	0	1	ось OУ
1	1	X	множество точек плоскости

Такая же таблица строится для второй прямой.

δ = 0	δx = 0	δy = 0	Ед. реш.	Мн.реш.	Реш. нет
0	X	X	1	0	0
1	0	X	0	0	1
1	X	0	0	0	1
1	1	1	0	1	0

Каждая строка этих таблиц преобразуется в тест. При возможности (с учетом независимости групп) берутся данные, соответствующие строкам сразу двух или всех трех таблиц.

В результате к уже имеющимся тестам добавляются

1) проверки всех случаев расположения обеих прямых - 6 тестов по первой прямой вкладываются в 6 тестов по второй прямой так, чтобы варианты не совпадали, - 6 тестов;

2) выполняется отдельная проверка несовпадения условия δx = 0 или δy = 0 (в зависимости от того, какой тест был выбран по методу граничных условий) - тест также можно совместить с предыдущими 6 тестами;

4. По методу предположения об ошибке добавим тест:

Все коэффициенты - нули.

Всего получили 20 тестов по всем четырем методикам. Если еще попробовать вложить независимые проверки, то возможно число тестов можно еще сократить. (Не забудьте для каждого теста заранее указывать результат!).

Варианты заданий:

№	Задача
1	Разработать программу решения квадратного уравнения , где a, b, c - любые вещественные числа.

2	Разработать программу определения суммарной длины тени, которую отбрасывают на ось OX отрезки, параллельные этой оси и заданные координатами x начала и конца отрезка.
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.
4	Пользователь должен ввести свой e-mail. Необходимо разработать программу, которая проверит корректность вводимого пользователем адреса.
5	Пользователь заполняет анкету и в одной из граф пишет свою дату рождения. Разработать программу, которая определяет знак зодиака пользователя по дате рождения.
6	Пользователь вводит число. Разработать программу, которая определяет, что он ввел год, а также рассматривает и определяет тип года (високосный или нет).
7	Пользователь вводит IP адрес (версия протокола 4). Разработать программу, которая проверяет корректность введенных данных, т.е. принимает или нет введенные данные.
8	Разработать программу определения вида треугольника, заданного длинами его сторон: равносторонний, равнобедренный, прямоугольный, разносторонний.
9	Программа считает сумму последовательности 15 целых чисел без максимального элемента последовательности.
10	Программа из последовательности 15 целых чисел выводит предпоследнее максимальное значение элемента.
11	Программа считает сумму последовательности 15 целых чисел без минимального элемента последовательности.
12	Программа из последовательности 15 целых чисел выводит предпоследнее минимальное значение элемента.
13	Программа из последовательности 15 целых чисел выводит минимальное значение элемента и предпоследний по величине элемент.
14	Программа из последовательности 15 целых чисел выводит максимальное значение элемента и второй по величине элемент.
15	Программа из последовательности 15 целых чисел выводит максимальное значение элемента, устанавливает, сколько раз это значение встречается.
16	Программа из последовательности 15 целых чисел выводит максимальное значение элемента, минимальное значение элемента и их сумму.