Структурное тестирование

Структурное тестирование основывается на детальном изучении логики программы и подборе тестов, обеспечивающих максимально возможное число проверяемых операторов, логических ветвлений и условий. Его еще называют «тестирование по маршрутам».
Под **маршрутом** понимают последовательности операторов программы, которые выполняются при конкретном варианте исходных данных. При построении тестов используют следующие критерии:

- покрытие операторов путем выбора набора данных, обеспечивающего выполнение каждого оператора в программе по крайней мере один раз.

- покрытие условий путем подбора наборов данных, обеспечивающих в узлах ветвления с более чем одним условием принятие каждым условием значения "истина" или "ложь" хотя бы по одному разу.

- комбинаторное покрытие условий путем подбора тестов, обеспечивающих в узлах ветвления с более чем одним условием перебор всех возможных сочетаний значений условий в одном узле ветвления.

Считают, что программа проверена полностью, если с помощью тестов удается осуществить выполнение программы **по всем возможным маршрутам передач управления**. Однако нетрудно видеть, что даже в программе среднего уровня сложности число не повторяющихся маршрутов может быть очень велико и, следовательно, полное или исчерпывающее тестирование маршрутов, как правило, невозможно.

Рассмотрим пример простой программы

```
{Программа, вычисляющая выражение

y=1/a +sqrt(25-b*b)

или выдающая сообщение об ошибке}

program example;

{Программа, вычисляющая выражение y=1/a +sqrt(25-b*b)

или выдающая сообщение об ошибке}

var
```

```
y,a,b:real;
begin
writeln('Введите a и b');
readln(a,b);
if(a<>0) and (b*b<=25) then
begin
y:=1/a+sqrt(25-sqr(b));
writeln('Результат: ',y);
end
else
writeln('Некорректные данные!');
end.
```

При структурном тестировании учитывается логика программы.

Примеры тестов.

1) Для тестирования программы с помощью критерия покрытия операторов в ПРОСТЕЙШЕМ СЛУЧАЕ достаточно проверки программы на следующих тестах:

1) a = 1 b = 4 (b<5)

2) a = 1 b = 6 (b>5)

Видно, что в этих тестах ни разу не будет рассмотрен случай, когда a равно 0.

2) Для тестирования программы с помощью критерия покрытия условий можно использовать следующий набор тестов:

1) a = 1 b = 6;

2) a = 0 b = 1.

Видно, что в этих тестах ни разу не будет проверено выполнение вычисления, хотя тестовые наборы и удовлетворяют критерию.

3) Для тестирования программы с помощью критерия комбинаторного покрытия условий тестовые наборы данных могут выглядеть, например, следующим образом

1) a = 1 b = 4;

2) a = 0 b = 4;

3) a = 1 b = 6;

5) a = 0 b = 6.

Структурный подход к тестированию имеет ряд недостатков. Так тестовые наборы, построенные по данной стратегии:

• не обнаруживают пропущенных маршрутов;

• не обнаруживают ошибок, зависящих от обрабатываемых данных, например, в операторе

if (a - b) < eps

пропуск функции абсолютного значения abs проявится только, если a < b;

• не дают гарантии, что программа правильна, например, если вместо сортировки по убыванию реализована сортировка по возрастанию.