Loop Testing

MADHESWARI.K
AP/CSE

Loop Testing?

- Loop testing is a white box testing technique that focuses exclusively on the validity of loop constructs.
- Loop Structures are most frequently used Control Structures.

Goal of loop testing

The goal of *loop testing* is to test while-do, repeat-until, (or do-while) and any other loops in a program thoroughly - by trying to ensure that each is executed at

- Minimal (n),
- typical,
- and (if this is defined) maximal values (m)

- - and to try to ``break'' the program, by trying to have a loop executed with a
  - fewer than minimum, (n-1)
  - as well as a larger than maximal,(m+1)
  - number of iterations. (m-n)

Types of Loops

1.Simple loops-*Simple Loops* are loops whose loop bodies contain no other loops

2. Nested loops-*Nested Loops* are combinations of loops such that each is contained inside the loop body of the next.

3. Concatenated loops

4. Unstructured loops: this should be redesigned into structured loops.

Guidelines for Loop Testing

- The ``guidelines'' for loop testing consist of lists of test cases that should be included for each type of loop.

Guidelines for Simple Loops

- Try to design a test in which the loop body isn't executed at all.
- Try to design a test in which the loop body is executed exactly once.
- Try to design a test in which the loop body is executed exactly twice.
- Design a test in which a loop body is executed some ``typical'' number of times.

If there is an upper bound, *n*, on the number of times the loop body can be executed, then the following cases should also be applied.

- Design a test in which the loop body is executed exactly *n*-1 times.
- Design a test in which the loop body is executed exactly *n* times.
- Try to design a test causing the loop body to be executed exactly *n*+1 times.

Example

```
//range 0 to 10
int n=10
for ( int i=0; i<=n;i++)
{
print (“welcome”);
}
```

Test case for simple loop testing

Test case ID	Value of I range(n-) (0-10)	Expected outcome	Actual results
1	n-1=-1 Value of I is -1	loop body isn't executed at all.	
2	Value of I is 10	Executed once Welcome	
3	Value of I is 9	Executed twice Welcome welcome	
4	Value of I is 6	Executed 5 times	
5	Value of I is 1	Executed 10 times	
6	Value of I is 0	Executed 11 times	
7	Value of I is 11	loop body isn't executed at all.	

Guidelines for Nested Loops

- start at the inner most loop. Set all other loops to minimal non zero values.
- Conduct simple loop test for inner most loop while holding outer loops at their minimum iteration parameter values.
- Work outward, conducting tests for the next loop, but keep all other outer loops to minimal values and nested loops to typical values.
- Continue this until all loops have been tested.

Example

```
//range 0 to 10
int n=10
for ( int i=0; i<=n;i++)
{
for ( int j=1; i<=5;j++)
{
print (“welcome”);
}
}
```

Test case for Nested loop testing

Test case ID	Value of I range(n-) (0-10)	Expected outcome	Actual results
1			
2			
3			
4			
5			
6			
7			

Guidelines for Concatenated Loops

If the loops are ``independent,'' so that the number of iterations used for one loop doesn't depend on the number of iterations used for any other(s), then it is sufficient to apply the guidelines for simple loops to each of the loops in the sequence.

On the other hand, if the number of iterations used for one loop *does* depend on the number of iterations used for another, then the following guidelines should be used instead.

- Conduct ``simple loop tests'' for the *bottom* (or ``*last*'') loop in the sequence, holding the number of iterations of the higher (or ``previous'') loops at minimal values.
- Work up toward the top loop, considering each loop in turn, and applying ``simple loop tests'' for each loop in turn, keeping the number of iterations of upper loops at minimal values and keeping the number of iterations of lower loops at typical values.

Test case for concatenated loop testing

Test case ID	Value of I range(n-) (0-10)	Expected outcome	Actual results
1			
2			
3			
4			
5			
6			
7			

Unstructured loops

- This class of loops should be redesigned to reflect the use of structured programming constructs.