Testiranje petlji

Testiranje petlji

(prema: Boris Beizer Software Testing Techniques)

Jednostruke petlje

- Treba primeniti sledeće testove (n je maksimum mogućih iteracija petlje):
  1. Preskočiti potpuno telo petlje,
  2. Jedna iteracija petlje,
  3. m iteracija petlje, gde je m<n,
  4. n - 1, n, n + 1 iteracija.

Testiranje petlji

- Maksimum iteracija često se ne određuje na osnovu uslova ostanka u petlji, nego i na osnovu drugih operacija unutar petlje
- **Primer**: program za računanje n!

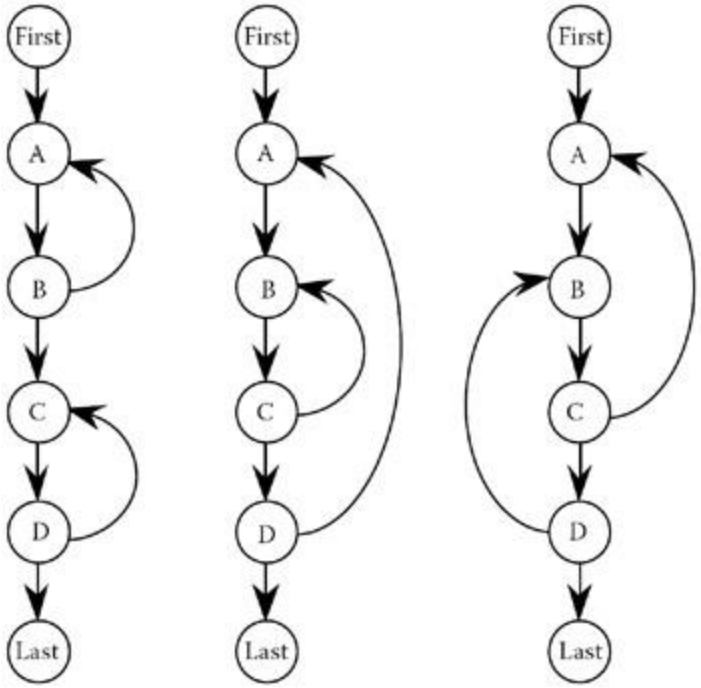
```
1 #include <stdio.h>
2 main()
3 {
4     int i, n, f;
5     printf (“n = “);
6     scanf (“%d”, &n);
7     if(n<0) {
8         printf (“Invalid: %d\n”, n);
9         n=-1;
10    } else {
11        f = 1;
12        for (i = 1; i <= n; i++) {
13            f *= i;
14        }
15        printf(“d! = %d\n”, n, f);
16    }
17    return n;
18 }
```

n	n!
0	1
1	1
2	2
3	6
4	24
5	120
6	720
7	5,040
8	40,320
9	362,880
10	3,628,800
11	39,916,800
12	479,001,600
13	6,227,020,800
14	87,178,291,200
15	1,307,674,368,000

Za 32bitni int, maxint=2,147,483,647 maksimalni broj iteracija da ne dođe do prekoračenja u f je 12

Testiranje petlji

- Klasifikacija višestrukih petlji:



- Nadovezane ugneždene nestrukturirane

Testiranje petlji

Ugneždene petlje

- Za testiranje ugneždenih petlji ne može se primeniti prethodno opisana tehnika za proste petlje, pošto bi to dovelo do geometrijskog povećanja broja test primera.
- Jedan pristup za ugneždene petlje bi bio:
  1. Početi sa najugneždenijom petljom.
  2. Sprovesti testove za jednostruku petlju na najugneždenijoj petlji držeći ostale petlje na minimumu iteracija. Dodatno testirati za nelegalne vrednosti (izvan opsega, izuzetke).
  3. Preći na okružujuću petlju, sprovodeći testove opisane u koraku 2. za nju dok se okružujuće petlje drže na minimumu iteracija, a ugneždena petlja ili petlje na tipičnim vrednostima iteracija.
  4. Ponavljati korak 3. sve dok se sve petlje ne istestiraju.

Testiranje petlji

- **Nadovezane petlje**
- Petlje koje slede jedna za drugom mogu se tretirati kao jednostuke ako su nezavisne jedna od druge. U suprotnom slučaju (na primer, ako je brojač petlje isti za obe), primeniti pristup kao za ugneždene petlje.

Testiranje petlji

Nestrukturirane petlje

- Ova vrsta petlji (na primer, nisu ugneždene nego se “seku”) je indikator lošeg stila programiranja. Ovakve petlje ne treba testirati nego ih eliminisati restrukturiranjem programa.

LCSAJ testiranje

Uvod

- **LCSAJ (Linear Code Sequence And Jump)** se definiše kao trojka (brojeva linija izvornog koda):
  - Početak linearne sekvence izvršnih naredbi (početak programa ili odredište nekog skoka)
  - Kraj linearne sekvence naredbi (kraj programa ili linija sa skokom) i
  - Odredišna linija na koju se kontrola toka prenosi na kraju linearne sekvence skokom (a ne sekvencijalno)
- Napomena: Vrlo slična tehnika je **dd-path testing** (decision-decision path testing): putanja izvršavanja kroz graf toka između dve odluje koja ne uključuje uslovne čvorove

Testiranje LCSAJ sekvenci

- Pronaći sve LCSAJ sekvence u programu i pokriti ih testovima (ako je moguće)

Primer

- Razmatramo sledeći deo programa koji kategorizuje pozitivne cele brojeve na proste i složene, i određuje delioce onih koji su složeni.

Program

```
1   READ (Num);
2    WHILE NOT End of File DO
3          Prime := TRUE;
4          FOR Factor := 2 TO Num DIV 2 DO
5                IF Num - (Num DIV Factor)*Factor = 0 THEN
6                      WRITE (Factor, ` is a factor of', Num);
7                      Prime := FALSE;
8                ENDIF;
9          ENDFOR;
10         IF Prime = TRUE THEN
11               WRITE (Num, ` is prime');
12         ENDIF;
13         READ (Num);
14  ENDWHILE;
15  WRITE (`End of prime number program');
```

Objašnjenje programa

Kod u liniji 5 izračunava ostatak pri deljenju broja(Num) deliocem(Factor).

Na primer: Factor=5, Num=13

izraz će imati vrednost 3

Zbog jednostavnosti unosa, program ne čini proveru pri unosu

Određivanje LCSAJ sekvenci

- Počinje se od liste grana u programu (navedeni su i uslovi koji treba da budu zadovoljeni da bi se grana izvršila)

(2->3) :		NOT End of File
(2->15) :	SKOK	End of File
(4->5) :		Num DIV 2 >= 2
(4->10) :	SKOK	Num DIV 2 <2
(5->6) :		IF naredba u liniji 5 je true
(5->9) :	SKOK	IF naredba u liniji 5 je false
(9->5) :	SKOK	sledeće izvršavanje FOR petlje
(9->10) :		završavanje FOR petlje
(10->11) :		Prime=true
(10->13) :	SKOK	Prime=false
(14->2) :	SKOK	uvek se izvršava

Određivanje LCSAJ sekvenci

- Pomoću prethodne liste nalazimo LCSAJ početne tačke. To je početak programa (linija 1), zatim linije na koje kontrola skače, ukoliko to nije sa prethodne naredbe (linije 2, 5, 9, 10, 13, 15)

Određivanje LCSAJ sekvenci

- Postoji šest LCSAJ sekvenci koji počinju od linije 1 (gledamo gde sve počev od 1 može da se izvrši skok):

  (1,2,15), (1,4,10), (1,5,9), (1,9,5), (1,10,13) i (1,14,2).

- Npr. da bi se izvršilo (1,14,2) moraju svih 5 grana da se izvrše sekvencijalno- (2->3), (4->5), (5->6), (9->10), (10->11)

Određivanje LCSAJ sekvenci

- LCSAJ sekvence sa početkom u 2, isti nastavak kao od 1:

  (2,2,15), (2,4,10), (2,5,9), (2,9,5), (2,10,13) i (2,14,2)

- Itd, razmatramo i ostale sekvence sa početkom u 5,9,10, 13, 15

Tabela svih LCSAJ sekvenci

LCSAJ		
START LINE	FINISH LINE	JUMP TO LINE
1	2	15
1	4	10
1	5	9
1	9	5
1	10	13
1	14	2
2	2	15
2	4	10
2	5	9
2	9	5
2	10	13
2	14	2

LCSAJ		
START LINE	FINISH LINE	JUMP TO LINE
5	5	9
5	9	5
5	10	13
5	14	2
9	9	5
9	10	13
9	14	2
10	10	13
10	14	2
13	14	2
15	15	exit

Test primer

- Inicijalni test ulaz se sastoji od jednog prostog broja (5), jednog složenog (6) i specijalnog slučaja broja (2). Svi brojevi se unose odjednom

Tabela test primera sa očekivanim izlazima

Test Case	Input	Expected Outcome
1	5	5 is prime
	6	2 is a factor of 6
		3 is a factor of 6
	2	2 is prime
		End of prime number program

Tabela LCSAJ sekv. i pokrivenost grana osnovnim test primerom

LCSAJ			
START LINE	FINISH LINE	JUMP TO LINE	TIMES EXECUTED
1	2	15	0 ***
1	4	10	0 ***
1	5	9	1
1	9	5	0 ***
1	10	13	0 ***
1	14	2	0 ***
2	2	15	1
2	4	10	1
2	5	9	0 ***
2	9	5	1
2	10	13	0 ***
2	14	2	0 ***

Nastavak tabele

5	5	9	0 ***
5	9	5	0 ***
5	10	13	1
5	14	2	0 ***
9	9	5	0 ***
9	10	13	0 ***
9	14	2	1
10	10	13	0 ***
10	14	2	1
13	14	2	1
15	15	exit	1

Number of LCSAJs	23
Number executed	9
Number not executed	14
LCSAJ Coverage	39%

Preduslovi za ostale test primere

- treba da obezbede maksimalno moguće pokrivanje preostalih LCSAJ sekv.

LCSAJ	Uslovi
(1, 2, 15)	Odmah na početku End of file
(1, 4, 10)	Prvi broj na listi manji od 4
(1, 9, 5)	Prvi broj na listi veći od 5
(1, 10, 13)	Prvi broj na listi 4
(2, 5, 9)	Dodatni broj veći od 4 koji nije prvi na listi
(2, 10, 13)	Broj 4 koji nije prvi na listi
(5, 5, 9)	Broj veci od 6
(5, 9, 5)	Dodatni složeni broj veći od 7
(9, 9, 5)	Broj veći od 7
(9, 10, 13)	Dodatni složeni broj veći od 8

Neizvršive LCSAJ sekvence

- Posle izvršavanja linije 7, prost=False. Grana (10->11) zahteva da je prime=True. Prema tome, sledeća 3 LCSAJs čija linearna sekvenca sadrži deo od 7. do 11. linije su neizvršivi: (1,14,2), (2,14,2) i (5,14,2).

Neizvršive LCSAJ sekvence

- LCSAJ (10,10,13) je takođe neizvršiv, pošto zahteva da na liniji 10 prime=False. Opet, da bi LCSAJ počeo na liniji 10, potrebno je da skoči sa linije 4 na 10, samim ti da prime bude postavljeno na True na liniji 3

Dodatni test primeri

Test Case	Input	Expected Output	LCSAJs executed
2	<none>	End of prime number program	(1, 2, 15)
3	2	2 is prime	(1, 4, 10)
	4	2 is a factor of 4	(2, 10, 13)
		End of prime number program	
4	8	2 is a factor of 8	(1, 9, 5)
		4 is a factor of 8	(5, 5, 9)
		End of prime number program	

Dodatni test primeri

5	4	2 is a factor of 4	(1, 10, 13)
	11	11 is prime	(2, 5, 9)
			(9, 9, 5)
			(5, 5, 9)
			(9, 10, 13)
		End of prime number program	

Posle primene dodatnih testova pokriveno je 19 od 23 LCSAJ sekvenci, odnosno 83%