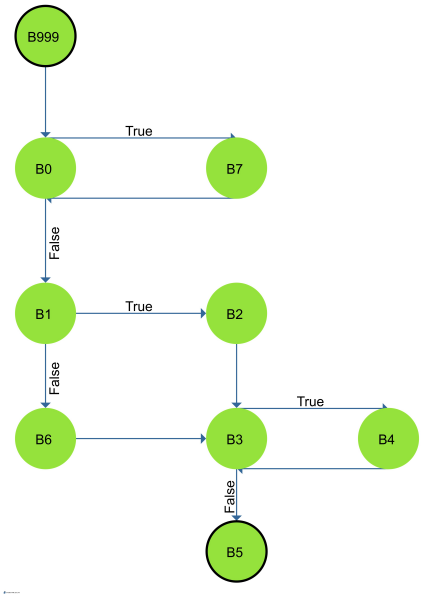
Strukturální testování - řídicí tok

Skupina: **10**
Řešitelé: **Stanke Michal**, **Timr Marek**, **Voříšek Lukáš**

Zadání úlohy

Následující zdrojový Java kód funkce převeďte na graf řídicího toku. Metodou hlavních cest sestrojte sadu testovacích cest, které plně pokryjí daný kód.

```
public int fnc() {
  B999();
  while( B0() ) {
    B7();
  }
  if( B1() ) then {
    B2();
  }
  else {
    B6();
  }
  while( B3() ) {
    B4();
  }
  B5();
}
```



obrázek 1 - Model řídícího toku

Hledání hlavních cest

1. Nalezni cesty délky 0 (uzly).
2. Kombinuj cesty délky 0 do cest délky 1 (hrany).
3. Kombinuj cesty délky 1 do cest délky 2.
4. atd.

Značení:
- ! ... cesta nemůže být prodloužena
- * ... cesta tvoří smyčku

Cesty délky 1

- 999
- 0
- 1
- 2
- 3
- 4
- 5!
- 6
- 7

Cesty délky 2

- 999→0
- 0→1
- 0→7
- 1→2
- 1→6
- 2→3
- 3→4
- 3→5!
- 4→3
- 6→3
- 7→0

Cesty délky 3

- 999→0→1
- 999→0→7!
- 0→1→2
- 0→1→6
- 0→7→0*
- 1→2→3
- 1→6→3
- 2→3→4!
- 2→3→5!
- 3→4→3*
- 4→3→4*
- 4→3→5!
- 6→3→4!
- 6→3→5!
- 7→0→1
- 7→0→7*

Cesty délky 4

- 999→0→1→2
- 999→0→1→6
- 0→1→2→3
- 0→1→6→3
- 1→2→3→4!
- 1→2→3→5!
- 1→6→3→4!
- 1→6→3→5!
- 7→0→1→2
- 7→0→1→6

Cesty délky 5

- 999→0→1→2→3
- 999→0→1→6→3
- 0→1→2→3→4!
- 0→1→2→3→5!
- 0→1→6→3→4!
- 0→1→6→3→5!
- 7→0→1→2→5
- 7→0→1→6→3

Cesty délky 6

- 999→0→1→2→3→4!
- 999→0→1→2→3→5!
- 999→0→1→6→3→4!
- 999→0→1→6→3→5!
- 7→0→1→2→5→4!
- 7→0→1→2→5→5!
- 7→0→1→6→3→4!
- 7→0→1→6→3→5!

Konečné cesty

Seznam níže obshuje všechny cesty, které jsou konečné, nebo vedou na cyklus.

1. B5!
2. B3→B5!
3. B999→B0→B7!
4. B0→B7→B0*
5. B2→B3→B4!
6. B2→B3→B5!
7. B3→B4→B3*
8. B4→B3→B4*
9. B4→B3→B5!
10. B6→B3→B4!
11. B6→B3→B5!
12. B1→B2→B3→B4!
13. B1→B2→B3→B5!
14. B1→B6→B3→B4!
15. B1→B6→B3→B5!
16. B0→B1→B2→B3→B4!
17. B0→B1→B2→B3→B5!
18. B0→B1→B6→B3→B4!
19. B0→B1→B6→B3→B5!
20. B999→B0→B1→B2→B3→B4!
21. B999→B0→B1→B2→B3→B5!
22. B999→B0→B1→B6→B3→B4!
23. B999→B0→B1→B6→B3→B5!
24. B7→B0→B1→B2→B5→B4!
25. B7→B0→B1→B2→B5→B5!
26. B7→B0→B1→B6→B3→B4!
27. B7→B0→B1→B6→B3→B5!

Hlavní cesty

Z předchozího seznamu konečných cest jsme vynechali podcesty jiných jednoduchých cest a tak jsme získali seznam hlavních cest, který je vypsán v následující tabulce.

Id cesty	Cesty:
1	999→0→7!
2	0→7→0*
3	3→4→3*
4	4→3→4*
5	4→3→5!
6	999→0→1→2→3→4!
7	999→0→1→2→3→5!
8	999→0→1→6→3→4!
9	999→0→1→6→3→5!
10	7→0→1→2→5→4!
11	7→0→1→2→5→5!
12	7→0→1→6→3→4!
13	7→0→1→6→3→5!

Testovací cesty

Nyní z předchozích hlavních cest zkonstruujeme cesty testovací, které mají tu vlastnost, že začínají v počátečním uzlu (999) a končí v koncovém (5). Tyto cesty nám zaručí spolehlivé pokrytí možných variant a určují tak, které průběhy musí tester vyzkoušet, aby bylo zaručeno, že se systém chová správně.

Označení testovací cesty (z jakých hlavních cest byla složena):	**Cesta:**
7	999→0→1→2→3→5!
9	999→0→1→6→3→5!
6+3+5	999→0→1→2→3→4→3→5!
8+5	999→0→1→6→3→4→3→5!
1+2+11	999→0→7→0→1→2→5→5!
1+2+11	999→0→7→0→1→2→5→5!
1+13	999→0→7→0→1→6→3→5!
1+10+5	999→0→7→0→1→2→5→4→3→5!
1+12+5	999→0→7→0→1→6→3→4→3→5!
6+4+5	999→0→1→2→3→4→3→4→3→5!

Pro ilustraci si z tabulky výše vybereme například testovací cestu označenou jako **9**. To pro testera znamená, že obrazovky, funkcionalitu či flow musí zkontrolovat ve zmíněných krocích. V našem případě tedy musí být zkontrolován průběh funkce `fnc` (vrací navzdory zadání void) tak, že bude zavolána sekvence funkcí `B999`, `B0`, `B1`, `B6`, `B3`, `B5`. Tedy například víme, že funkce `B0` vrátí nepravdu (v tomto konkrétním případě), neboť neproběhne zavolání funkce `B7`.