Megoldás

Feladat 2.

Statikus tesztelés – Statikus analízis

A megoldásnál a feladatkiírásnak megfelelően a megvalósított funkcióval nem kell foglalkozni, úgy kell elemeznünk a kódot, mintha fogalmunk sem lenne róla, hogy mit KELLENE valójában csinálnia. Pont mintha egy statikus analízis eszköz vagy compiler elemezné. Tehát nem tudjuk figyelembe venni azokat az észrevételeket, amelyek arra vonatkoznak, hogy egy probléma miatt nem fog jól működni az összehasonlítás, de nem is fut hibára.

Lehet, hogy meglepő egyébként, de a kód az alábbiakban felsorolt, általunk szándékosan elhelyezett hibákon kívül stabilan működik egy létező alkalmazásban a specifikációban szereplő inputokra. ☺

Nem tudjuk elfogadni azokat az észrevételeket, amelyek az inputok nem megfelelő ellenőrzésére vonatkoznak (a funkcionalítás vizsgálata nélkül nem lehet eldönteni az eljárás inputjainak értékkészletét). Statikus analízisnél ez nem érdekes, de amúgy a függvény leírásában is szerepel: “Egyeb formatumu verzioszamok kezelesevel nem kell foglalkozni, a fuggveny akar hibara is futhat.”.

A nyelv szándékosan fiktív, hogy ne legyenek előnyben azok, akik esetleg pont a feladat programnyelvéhez értenek. Sajnos akik azt nehezményezték, hogy miért nem egy igazi nyelvet használtunk, nem gondolták át, hogy igazi nyelv esetén elég egyszerű lett volna bemásolni egy eszközbe és megpróbálni lefordítani. Nem adtuk meg a nyelv pontos szintaktikáját, de a kódból közel 100%-ig kikövetkeztethető. Itt nem fogadtunk el semmi olyan hibát, ami egy létező programozási nyelvben megszokott – szintaktikailag az általunk használttól eltérő módon követelné meg a kód megírását. Elfogadjuk viszont azokat szintaktikai problémákat, amelyek teljesen egyértelműek a kódrészletből. Feltételezések, problémák, amelyeket nem fogadtunk el és így nem díjaztunk ponttal:

- // nem használható egy soros megjegyzés jelzéseként (szinte minden nyelvben megengedett)
- A for ciklust belül vagy a végén egy Next vagy Step paranccsal léptetni kellene
- Minden változó deklarációját ; zárja (van olyan nyelv, ahol vesszővel kell elválasztani)
- EgeszSzamTipus és hasonló típusoknál hosszt kellene megadni
- Feltételben lévő = jel értékadást jelent (egyértelmű a szintaktikából, hogy itt := az értékadás)

- Összetett feltételek nagy zárójelben megadandóak (sok nyelvben ez nem kötelező, pl. Pascal)
- Exit eljárásból lép ki, nem ciklusból (nálunk egyértelmű a megjegyzésből, hogy ciklusból)
- Else előtti utasításnál is kötelező a ; parancs végén (nálunk nem, ld. Pascal)
- Nullával kéne kezdődjön az indexelés tömböknél, ezért túlindexelések jelentkezhetnek (sok nyelvben 1-től, sőt, van olyan nyelv, ahol állítható, pl. Visual Basic)
- Megjegyzésben található elírások, nem egyértelmű részek

A kódban szereplő meghívott függvényekre és eljárásokra vonatkozó észrevételeket sem tudjuk hibának elfogadni, mindegyiket úgy próbáltuk megfogalmazni, hogy egyértelmű legyen a működése és a nyelv részének tekintettük őket.

Az alábbi hibákat tekintettük statikus analízis során megtalálható problémának.

1. iCheck változó nem kerül megfelelően inicializálásra használat előtt.

2. Értékadás i változónak a program végén felesleges, utána már nem használt a változó.

3. Felesleges ismételt értékadás: sTemp := SzovegbolKimasolHonnanMennyit(sPart, 1, iPos - 1);

4. Változó j soha nem használt.

5. sPart := ''; előtti Exit utasítás érintetlen kódot eredményez.

6. Szintaktikai hiba: endlf a végén endif helyett.

7. aVersionList2 változó nem kerül inicializálásra.

8. Tömbök felszabadítása túl hamar történik, utána még megpróbáljuk használni őket.

9. A két feldolgozás között üres blokkok találhatóak, felesleges if szerkezet.

10. iPos deklarálatlan változó.

11. sVersion2 paraméter soha nem használt a kódban.

12. Az egész rutin duplán van, ki kellene emelni egy függvénybe egy verziószám feldolgozását (jobb eszközök bizony a kód duplikálását is jelzik).

13. Numerikus konstansok a kódban nem ajánlottak: ki kell emelni őket a kód elejére vagy központi helyre, pl. 100 és 4 (Magic numbers). Itt azt is elfogadtuk, ha valaki lehetséges túlindexelési problémát jelzett a 100 miatt.

Minden megtalált hiba 7 pont, a beadott megoldás formátumára (anomáliák pontos megfogalmazására és könnyű javíthatóságára) további 4 pontot, statikus analízis lényegének megfelelő ismeretére további maximum 5 pontot adtunk. A statikus analízis lényege a vezérlési és adatfolyam elemzés, aki ezeket a fogalmakat ismeri és jól használta, ők kaphattak 5 pontot utóbbira. A fogalmak ismerete a megtalált hibákból eldönthető volt.

Ha valaki nem ott jelezte a hibát, ahol egy eszköz tenné, azt elfogadtuk. Ilyenek például:

- Második nagy blokk elején sVersion2 helyett sVersion1 használata: ez önmagában nem hiba, sokan funkcionális problémának jelezték. Pedig nemcsak funkcionális hiba, hanem automatikusan felismerhető hiba is: sVersion2 paraméter soha nem használt. Emiatt igazából a függvény paraméterlistájánál kéne jelezni.
- Második nagy blokk elején aVersionList2 helyett aVersionList1 újrainicializálása: ez önmagában nem hiba, a tényleges hiba, hogy a rossz inicializálás miatt pár sorral lentebb aVersionList2 változót inicializálatlanul próbáljuk használni.

Nem kellett mindkét lemásolt nagy blokkban külön jelezni a hibákat, ha valaki az egyikben megtette, már megkapta a pontot arra a részre.

Összesen maximum 100 pont.

Az alábbiakban láthatóak a hibák, egyetlen előfordulási helyet jelezve.

```
/*
VersionCheck: 2 verzioszam kozul visszaadja a nagyobbat.
A verzioszamok szovegesen adottak, csak szamot es pontot tartalmazhatnak, minimum 1 es maximum 4
egysegbol allhatnak.
Az egysegekben talalhato al verzioszamok maximum 4 szamjegybol allhatnak.
Ervenyes bemeneti formatumok a verzioszamokra:
        8
        8.2
        8.3.0
        8.3.1.4
        1234.1234.1234.1234
Egyeb formatumu verzioszamok kezelesevel nem kell foglalkozni, a fuggveny akar hibara is futhat.
A fuggvenynek megadhato, hogy hany szint melyen vizsgalja a verzioszamokat osszehasonlistaskor.
Lehetseges ertekei:
        1: csak az elso    egysegig hasonlitja, a tobbit nullanak tekinti
        2: csak a masodik  egysegig hasonlitja, a tobbit nullanak tekinti
        3: csak a harmadik egysegig hasonlitja, a tobbit nullanak tekinti
        4: csak a negyedik egysegig hasonlitja, a tobbit nullanak tekinti
A fuggveny felbontja a szovegkent adott elso verzioszamot egy szamokat tartalmazo tombbe, majd a
masodik verzioszamra is megteszi ugyanezt.
Vegul osszehasonlitja a ket tombot.
A fuggveny visszateresi erteke, hogy melyik verzioszam nagyobb:
        0: megegyeznek
        1: az elso
        2: a masodik
*/

function VersionCheck(sVersion1 SzovegesTipus, sVersion2 SzovegesTipus, iDeep EgeszSzamTipus) return
EgeszSzamTipus
declare
   aVersionList1 DinamikusVektorTipus,
   aVersionList2 DinamikusVektorTipus,
   i EgeszSzamTipus,
   j EgeszSzamTipus,
   sPart SzovegesTipus,
   iCheck EgeszSzamTipus,
   sTemp SzovegesTipus;
begin
 //1. verzioszam feldolgozasa
 sPart := sVersion1;
 InicializalHosszraIndexTollg(aVersionList1, 1, 4);
 for i := 1..4
   if (i > iDeep) //ha elertuk a megadott szintet, akkor nem kell tovabb vizsgalnunk
     aVersionList1( i ) := 0;
     exit; //kilepes a ciklusbol
   endif;
   iPos := SzovegKeresMitMiben( '.', sPart );
   if (iPos > 0)  //talaltunk meg pontot a verzioszamban
```

Comment [g1]: Nem használt változó (parameter)

Comment [g2]: Nem használt változó

Comment [g3]: Deklarálatlan változó

```
    sTemp := SzovegbolKimasolHonnanMennyit( sPart, 1, iPos - 1 );
    aVersionList1( i ) := SzovegSzammaAlakitas(sTemp);
    sTemp := SzovegbolKimasolHonnanMennyit( sPart, 1, iPos - 1 );
    sPart := SzovegbolKimasolHonnanMennyit( sPart, iPos + 1, 100 );
   else
    sTemp := sPart;
    aVersionList1( i ) := SzovegSzammaAlakitas(sTemp);
    exit; //kilepes a ciklusbol
    sPart := '';
   endif;
  endfor;
  if (sPart > '')
    if (sPart < '    ')
    endif;
  endif;
  Felszabadit(aVersionList1);

  //2. verzioszam feldolgozasa
  sPart := sVersion1;
  InicializalHosszraIndexTolIg(aVersionList1, 1, 4);
  for i := 1..4
   if (i > iDeep) //ha elertuk a megadott szintet, akkor nem kell tovabb vizsgalnunk
    aVersionList2( i ) := 0;
    exit; //kilepes a ciklusbol
   endif;
   iPos := SzovegKeresMitMiben( '.', sPart );
   if (iPos > 0)  //talaltunk meg pontot a verzioszamban
    sTemp := SzovegbolKimasolHonnanMennyit( sPart, 1, iPos - 1 );
    aVersionList2( i ) := SzovegSzammaAlakitas(sTemp);
    sTemp := SzovegbolKimasolHonnanMennyit( sPart, 1, iPos - 1 );
    sPart := SzovegbolKimasolHonnanMennyit( sPart, iPos + 1, 100 );
   else
    sTemp := sPart;
    aVersionList2( i ) := SzovegSzammaAlakitas(sTemp);
    exit; //kilepes a ciklusbol
    sPart := '';
   endif;
  endfor;
  Felszabadit(aVersionList2);

  //Szamma alakitott verzioszamok osszehasonlitasa
  i := 0;
  while ( i < iDeep ) and ( iCheck = 0 )
   i := i + 1;
   if (aVersionList2( i ) > aVersionList1( i ))
    iCheck := 2
   elseif (aVersionList2( i ) < aVersionList1( i ))
    iCheck := 1;
```

Comment [g4]: Ismételt értékadás azonos értékkel

Comment [g5]: Többször használt numerikus konstans (magic numbers)

Comment [g6]: Érintetlen kód

Comment [g7]: Üres blokk, felesleges

Comment [g8]: Kódismétlések

Comment [g9]: Felszabadított változó

Comment [g10]: Inicializálatlan változó

Comment [g11]: Inicializálatlan változó

Comment [g12]: Felszabadított változó

```
  endIf;
 endwhile;
 i := 0;

 return iCheck;
endfunction;
```

Comment [g13]: Ismeretlen kulcsszó

Comment [g14]: Felesleges értékadás, már nem használt változó