

## 11 Testiranje algoritama (programa)

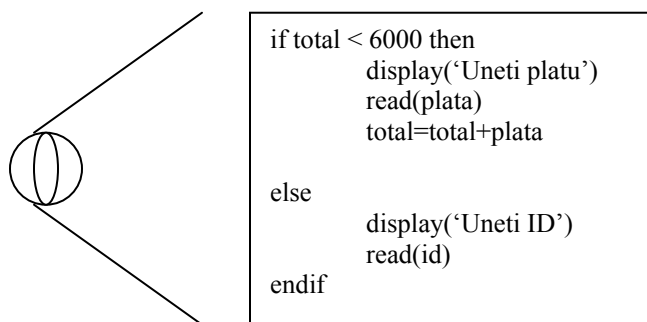
Dokazati da algoritam obavlja zadatak ili rešava problem koji je postavljen nije nimalo lak zadatak. Iako postoje matematičke metode za takvo dokazivanje (koje se nazivaju verifikacija programa) u praksi se najčešće takve metode ne primenjuju. Umesto toga pristupa se testiranju programa. Izuzev za veoma jednostavne programe, testiranjem se ne može dokazati ispravnost programa. Testiranjem se pokušavaju pronaći greške u programu. No, bez obzira koliko se grešaka pronađe, uvek postoji mogućnost da ih ima još. I pored toga što testiranje ne garantuje odsustvo grešaka, testirani programi su bolji nego netestirani.

Kao što je već rečeno u vezi sa životnim ciklusom softvera, a i kod strategije razvoja algoritama, testiranje je veoma važna faza u razvoju svakog algoritma – programa.

Nažalost, testiranjem se ne može dokazati ispravnost programa, to jest ne može se dokazati potpuno odsustvo grešaka, ali se mogu otkriti neke od grešaka ako se primeni pravilan postupak testiranja nad pažljivo odabranim testnim podacima.

Cilj testiranja je da se pronađu okolnosti pod kojima program (algoritam) daje pogrešne rezultate ili ne da je rezultat uopšte (halting problem). Ako ne možemo da pronađemo takve okolnosti tada možemo imati razumno verovanje da program ispravno funkcioniše. Pravljenje dobrog test plana kojim se potvrđuje ispravnost programa je veoma težak zadatak. Kada programi imaju hiljade linija koda (koraka) teško je testirati svaku moguću »putanju« pri njegovom izvršavanju kako bi se proverilo da program ispravno radi u svim mogućim slučajevima. Postoji nekoliko test strategija koje povećati šansu za pronalaženje grešaka (bagova) u programima.

### Testiranje »bele kutije« (staklene kutije)



Ovu vrstu testiranja može da obavlja ili programer koji je napisao program ili neko drugi kome je specijalnost testiranje softvera. Testiranje se naziva »bela kutija« zato što je onom ko testira poznat algoritam ili programski kod.

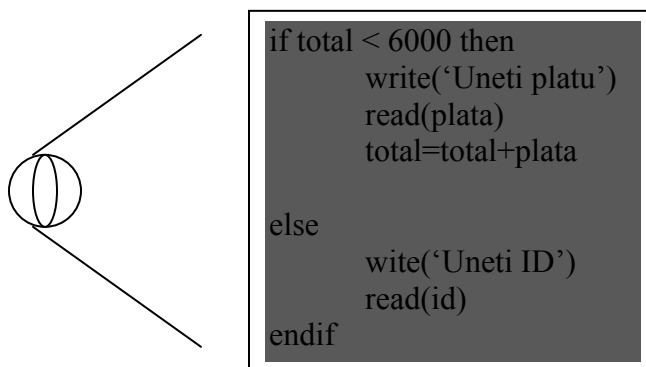
Posmatrajmo sledeći primer programa napisanog u pseudokodu:

```
if podatak < 100 then
    print('Loša ocena')
else
    if podatak < 200 then
        display('Moglo je bolje')
    else
        if podatak < 300 then
            display('Dobro')
        else
            display('Vrlo dobro')
        endif
    endif
endif
```

Pošto osoba koja testira program može da vidi program onda je u stanju da proveri saki deo programa tako što će testirati program za razne vrednosti podatka: za manje od 100, između 100 i 200, između 200 i 300, i na kraju za 300 ili više od 300.

Za testiranje programa pored uobičajenih biraju i takozvane granične vrednosti. Granične vrednosti su one vrednosti podataka kod kojih se mogu očekivati pogrešni rezultati (nula, ekstremne vrednosti itd.)

### Testiranje »crne kutije«



Testiranje “crne kutije” se tako naziva jer je programski kod ili algoritam nevidljiv osobi koja vrši testiranje. Test se sprovodi tako što se programu (crnoj kutiji) daju različiti podaci i posmatra ponašanje algoritma. Da bi se postigao efekat testiranja, kod ove vrste testova se test podaci pripremaju pre izrade programa, imajući na umu samo programske zahteve date u specifikaciji programa.

Za ovu vrstu testa se obično planiraju tri vrste testnih podataka :

- Ispravni podaci
- Neispravni podaci
- Granične vrednosti podataka

Za predhodni primer, ispravni podaci bi bili recimo 34, 150, 250, neispravni recimo ako umesto cifara damo slova, a graničnim slučajevima se mogu smatrati podaci 99,100,101,199,200,201,299,300 i 301.

### **Alfa testiranje**

Softver može biti testiran bilo kojom od predhodne dve metode. Uobičajeno je da se nakon testiranja program pre puštanja u prodaju, da na upotrebu ograničenom broju (obično programera) u istoj kompaniji. Oni koristeći program (simulirajući krajnjeg korisnika) pronalaze eventualne greške koje su zaostale posle testiranja, a takođe mogu da daju i sugestije za unapređenje programa

### **Beta testiranje**

Nakon alfa testiranja uobičajena je procedura da se softver da na upotrebu ograničenom broju spolasnih korisnika i da se od njih dobije odziv. Odziv može da sadrži pronađene greške ili takođe sugestije za funkcionalno ili neko drugo unapređenje softvera .

### **Tehnike testiranja**

Jedna od najčešće primenjivanih tehnika testiranja kada znamo algoritam (»bela kutija«) je primena tabele za testiranje (»Trace Table«). Ovde se nad testnim podacima vrši simuliranje rada algoritma, a za svaki korak algoritma se u tabeli vrši odgovarajuća izmena podataka definisana tim korakom.

Pokazaćemo to na jednom jednostavnom primeru.

Posmatrajmo sledeći algoritam (dat u prirodnom jeziku):

1. Postaviti **pokazivač (indeks) na tekuće slovo** na vrednost 0.
2. Postaviti **pokazivač (indeks) na poslednje slovo** na vrednost koja odgovara ukupnom broju slova na stranici.
3. Postaviti **brojač** na 0.
4. Učitati slovo na koje pokazuje **pokazivač (indeks) na tekuće slovo**.
5. Ako je učitano **slovo** samoglasnik dodati 1 na **brojač**.

6. Povećati **pokazivač (indeks) na tekuće slovo** za 1.
7. Ako je **pokazivač (indeks) na tekuće slovo**  $\leq$  **pokazivač (indeks) na poslednje slovo** idi na korak 4.

Ovim programom se očigledno prebrojavaju samoglasnici u tekstu ispisanom na nekoj stranici.

Kao testne podatke uzmimo da se na stranici nalazi sledeći tekst:

### Taj tekst

Tekst ima 9 slova (uočite da smo i razmak (blanko znak) računali kao slovo, jer i to je jedan ASCII znak).

Sledeća tabela ilustruje kako se u svakom od koraka gornjeg algoritma menjaju vrednosti podataka (promenljivih) koje se pojavljuju u algoritmu.

Korak	pokazivač na tekuće slovo	pokazivač na poslednje slovo	brojač	slovo	da li je slovo samoglasnik	da li je pokazivač na tekuće slovo $\leq$ pokazivač na poslednje slovo
pre početka algoritma	x	x	x	x	x	x
1	1	x	x	x	x	x
2	1	9	x	x	x	x
3	1	9	0	x	x	x
4	1	9	0	T	x	x
5	1	9	0	T	ne	x
6	2	9	0	T	ne	x
7	2	9	0	T	ne	da
4	2	9	0	a	ne	da
5	2	9	1	a	da	da
6	3	9	1	a	da	da
7	3	9	1	a	da	da
4	3	9	1	j	da	da
5	3	9	1	j	ne	da
6	4	9	1	j	ne	da
7	4	9	1	j	ne	da
4	4	9	1	'	ne	da
5	4	9	1	'	ne	da
6	5	9	1	'	ne	da
7	5	9	1	'	ne	da
4	5	9	1	t	ne	da
5	5	9	1	t	ne	da
6	6	9	1	t	ne	da
7	6	9	1	t	ne	da
4	6	9	1	e	ne	da
5	6	9	2	e	da	da
6	7	9	2	e	da	da
7	7	9	2	e	da	da

4	7	9	2	k	da	da
5	7	9	2	k	ne	da
6	8	9	2	k	ne	da
7	8	9	2	k	ne	da
4	8	9	2	s	ne	da
5	8	9	2	s	ne	da
6	9	9	2	s	ne	Da
7	9	9	2	s	ne	Da
4	9	9	2	t	ne	Da
5	9	9	2	t	ne	Da
6	10	9	2	t	ne	Da
7	10	9	2	t	ne	Ne

Ovakve tabele nam pokazuju neke interesantne stvari o algoritmima. Iz tabele se jasno vidi da promenljive (podaci – varijable) ne menjaju vredosti sve dok se ne naiđe na korak u kojem se nad njima vrši neka operacija. Druga važna činjenica je da tabela postaje veoma velika kada algoritam sadrži ponavljanje, posebno nad većim skupom podataka. Veličina tabele se može smanjiti ako se kod repeticije uradi samo jedan ciklus u posebnoj tabeli a u glavnoj tabeli pokaže samo prvo i poslednje ponavljanje. Tabela takođe pokazuje i značaj inicijalizacije promenljivih na početku programa kako bi se izbegle situacije da algoritam radi sa pogrešnim podacima ( pogledajte inicijalno stanje pre početka rada algoritma gde promenljive imaju nepoznatu vrednost označenu sa x).

### Pitanja

1. Šta su testni podaci?
2. Šta je testiranje »bele kutije«?
3. Šta je testiranje »crne kutije«?
4. Šta je alfa test?
5. Šta je beta test?
6. Šta su testne tabele (Trace table)?
7. Kada testna tabela ima mnogo više redova nego što algoritam ima koraka?