

9. Algoritmi

Nakon ovog predavanja vi ćete biti u stanju da:

1. definišete termin *algoritam*
2. navedete osnovne karakteristike algoritama
3. opišete šta se podrazumeva pod *nizom* (sekvencom)
4. opišete *if... then* i *if... then ... else* konstrukciju
5. opišete *do...while* i *while...do* konstrukciju
6. navedete različite načine izražavanja algoritama
7. objasnite šta se podrazumeva pod terminom "varijabla"
8. objasnite šta se podrazumeva pod pojmom *tip varijable (data type)*
9. objasnite šta su to pravila za davanje imena varijablama
10. opišete strategiju projektovanja algoritama

Šta je to algoritam?

Poznati programerski istraživač i profesor Niklaus Wirth je dao ovakvu definiciju:

$$\mathbf{Programs = Algorithms + Data}$$

Još u prvom predavanju vam je predstavljen pojam programa i podataka, ali šta je sad to algoritam?

Algoritam je osnovno rešenje problema na kome se zasniva program ili plan za izradu programa, ili se može reći da je algoritam:

"Efektivna procedura za rešavanje problema u konačnom broju koraka."

Efektivna zapravo znači da su koraci koje procedura definiše izvodljivi i jasno definisani. Veoma je važno da je broj tih koraka konačan tj. da se procedura odvija u konačnom vremenu. Dobro projektovan program mora uvek dati odgovor, taj odgovor nam se ne mora uvek svideti, ali on mora biti dat u konačnom vremenu. Ponekad se dešava da nema odgovora. To je u računarskoj teoriji poznato kao HALTING problem. Međutim, dobro projektovani programi moraju imati rešenje i za takve slučajeve, tj. moraju garantovati završetak u konačnom broju koraka.

Osnovne elementi (konstruktori) algoritama

Evo najpre jednog jednostavnog algoritma za spremanje čaja:

1. Ako u čajniku nema vode napunite čajnik vodom
2. Stavite čajnik na šporet i uključite odgovarajuću ringlu.
3. Ako šolja za čaj nije prazna ispraznite je.
4. Stavite lišće čaja u šolju za čaj.

5. Ako voda u čajniku nije provrela idite na korak 5, ako jeste idite na korak 6.
6. Isključite ringlu.
7. Sipajte vodu iz čajnika u šolju (pazite da ne prelijete).

Zapažamo da ovaj algoritam ima više koraka, da neki od koraka (1,3 i 5) sadrže donošenje odluka, da jedan korak (5) sadrži ponavljanje u kome se izvršava proces čekanja na vodu da provri.

Algoritam sadrži tri elementa:

1. Sekvencu – niz (proces)
2. Odlučivanje (selekcija)
3. Ponavljanje (repeticija, iteracija, ciklus, petlja)

Godine 1964 matematičari Corrado Bohm i Guiseppe Jacopini pokazali su da se svaki algoritam može izraziti pomoću sekvence, odluke i ponavljanja. To je poznato kao **teorema o programskoj strukturi** i predstavljalo je veoma važan korak ka strukturalnom programiranju koje je danas u upotrebi.

Sekvenca (niz operacija)

Sekvenca znači da se svaki korak sekvence izvršava u unapred datom redosledu – onako kako se pojavljuju u sekvenci, jedan za drugim. U predhodnom algoritmu svaki korak mora biti urađen u zadatom redosledu. Ako bi promenili redosled koraka to bi moglo da dovede do pogrešnih (ponekad i tragikomičnih) rezultata.

Odluka (Selekcija)

U algoritmima rezultat odluke je ili *tačno* ili *netačno*, nema ništa između. Rezultat odluke se bazira na nekoj tvrdnji (logičkom iskazu) koja može da ima vrednost tačno ili netačno, na primer:

if danas je sreda then uzmi platu

je odluka koja ima sledeći oblika:

if tvrdnja then proces

Tvrdnja (iskaz) je neka rečenica koja može biti tačna ili netačna, tako je tačno da danas je sreda ili je netačno da danas je sreda. To ne može biti istovremeno i tačno i netačno, niti nešto treće. Ako je tvrdnja tačna tada se izvršava proces koji sledi iza reči *then*. Ako tvrdnja nije tačna prelazi se na sledeću instrukciju bez izvršavanja procesa posle reči *then*.

Odluka može da ima i nešto složeniji oblik:

*if tvrdnja
then proces1
else proces2*

to je oblik *if ... then ... else ...*. Ovo znači da ako je tvrdnja tačna tada se izvršava proces1 a ako je netačna proces2.

U onom prvom obliku odluke *if tvrdnja then proces* u else delu nema procesa pa zato ni else nije potrebno.

Ponavljjanje (Iteracija)

Ponavljjanje ima dva oblika: *do...while* ciklus i *while..do* ciklus.

Repeat ciklus se koristi za ponavljanje procesa ili niza procesa sve dok tvrdnja iz uslova ponavljanja ne postane tačna. Repeat ciklus ima sledeći oblik:

do
Proces1
Proces2
.....
ProcesN

while *tvrdnja*

Evo jednog primera

do
Sipaj vodu u čajnik
while *čajnik nije pun*

Proces je *Sipaj vodu u čajnik*, a uslov za nastavak ciklusa *čajnik nije pun*.

do...while ciklus obavi (bar jednom) proces pre testiranja da li je uslov završetka ciklusa ispunjen. Šta će se desiti u predhodnom primeru ako je čajnik već bio pun kada je ciklus započet? Doći će do neželjenog procesa koji će izazvati prelivanje vode iz čajnika.

Za takve slučajeve pogodnije je koristiti *while...do* ciklus:

while *čajnik nije pun*
Sipaj vodu u čajnik

Pošto se odluka da li je čajnik pun ili ne donosi pre sipanja vode, mogućnost prelivanja je eliminisana.

Različiti načini izražavanja algoritama

Jedan način izražavanja algoritma smo videli maločas – taj način ćemo zvati forma deskriptivnih koraka. Tokom predavanja proučićemo četiri različite forme izražavanja algoritama:

1. Koračna-forma
2. Pseudokod
3. Dijagram toka
4. Nassi-Schneiderman (NS) forma
5. Jackson-ovi strukturni dijagrami (JSD) forma

Prve dve su pisane forma. Primer pravljenja čaja je tipičan primer Koračne forme (Step-Form) gde smo za izražavanje algoritma koristili prirodan jezik. Problem sa prirodnim jezikom je u tome što ponekad može biti neprecizan. Tako se može desiti da ono što jedan čovek napiše drugi pročita sa sasvim drugim tumačenjem. Pseudokod je takođe vrlo sličan prirodnom jeziku ali mnogo precizniji i sa ograničenim rečnikom.

Poslednja tri načina izražavanja algoritama su grafički, to jest u njima se koristi mešavina grafičkih simbola i pisanih reči da se predstave sekvenca, odluka i ponavljanje.

U sledećem predavanju ćemo se baviti ovim različitim načinima izražavanja algoritama, a sada ćemo najpre proučiti dve važne teme.

Šta su to varijable?

Pošto je $Programs = Algorithms + Data$ vraćamo se na temu podataka. Kako smo već rekli podatak je simbolički prikazana vrednost koja u programskom kontekstu dobija i značenje – znači u programu se podatak transformiše u informaciju. Pitanje glasi: Kako se podaci predstavljaju u programima?

Skoro svaki program sadrži podatke, a podaci se obično “sadrže” u varijablama. Tako varijablu možemo shvatiti kao kontejner za vrednosti koje mogu da se menaju tokom izvršavanja programa. Na primer, u našem primeru za pravljenje čaja nivo vode u čajniku je jedna varijabla, temperatura vode je varijabla, a i količina lišća je, takođe, varijabla.

Svakoj varijabli u programu se daje posebno ime, na primer:

- Nivo_Vode
- Temperatura_Vode
- Količina_Listova_Čaja

i u svakom datom trenutku vrednost koja je predstavljena varijablom Nivo_Vode može biti različita od vrednosti iste varijable u nekom drugom trenutku. Instrukcija:

- *If čajnik ne sadrži vodu then napuni čajnik*

može biti napisana i ovako:

- *If Nivo_Vode je 0 then napuni čajnik*

ili

- *If Nivo_Vode = 0 then napuni čajnik*

U nekom trenutku Nivo_Vode će dostići dozvoljeni maksimum, ma koliki on bio, i tada je čajnik pun.

Varijable i tipovi podataka

Podaci koji se koriste u algoritmima mogu biti različitog tipa. Najprostiji tipovi podataka su:

- numerički podaci kao što su 12, 11.45, 901.
- alfabetski (slovni) podaci (karakter) kao što su 'A', 'Z' ili 'Ovo je alfabetski niz'.
- logički podaci sa TRUE, FALSE vrednostima.

Davanje imena varijablama

Uvek treba da se trudite da varijablama u algoritmu date smisljena imena što će algoritam (i program) učiniti čitljivijim i razumljivijim. To je posebno važno kod velikih i kompleksnih programa.

U algoritmu za pravljenje čaja koristili smo prirodan jezik. Pokazaćemo kako se mogu koristiti imena varijabli za varijable tog algoritma. U desnoj koloni smo izabrali imena varijabli koja iako kraća od originalnih ne umanjuju značenje. Donja crta u nazivu varijable treba da označi da sve reči predstavljaju jednu jedinstvenu celinu kojom se predstavlja data varijabla.

1. Ako u čajniku nema vode napunite čajnik vodom	1. <i>If čajnik_prazan</i> then napuni čajnik
2. Stavite čajnik na šporet i uključite odgovarajuću ringlu	2. Stavite čajnik na šporet i uključite odgovarajuću ringlu
3. Ako šolja za čaj nije prazna ispraznite je.	3. <i>If šolja_nije_prazna</i> then isprazni šolju
4. Stavite lišće čaja u šolju za čaj.	4. Stavite lišće čaja u šolju za čaj.
5. Ako voda u čajniku nije provrela idi na korak 5	5. <i>If voda_ne_vri</i> then idi na korak 5
6. Isključite ringlu	6. Isključite ringlu

Ne postoje neka stroga pravila kao treba davati imena varijablama, ali postoje određene konvencije i preporuke. Dobro je da se usvojite neku od tih preporuka i da je onda dosledno koristite.

Uz put slična preporuka se može dati i za davanje imena procesima, takođe. To sve čini vaš program čitljivijim i razumljivijim, a time pogodnijim za održavanje i dalje unapređivanja - što znači da pomaže produžetku životnog ciklusa programa.

Strategija projektovanja algoritama

Sada kada znamo ponešto o algoritmima u stanju smo da razmatramo i strategiju za projektovanje algoritama. Evo jedne takve strategije koja može da bude korisna:

Korak 1: Istraživački korak

1. Identifikovati procese
2. Identifikovati glavne odluke
3. Identifikovati ponavljanja
4. Identifikovati varijable

Korak 2: Izrada preliminarnog (grubog) algoritma

1. Izrada algoritma "višeg nivoa"
2. Proći kroz algoritam misaonom simulacijom. Ako simulacija otkrije probleme ispraviti algoritam.

Korak 3: Izrada finalnog (detaljnog) algoritma

1. Do detalja razraditi "grubi" algoritam napravljen u koraku 2.
2. Grupišite procese koji se mogu grupisati
3. Grupišite varijable koje se mogu grupisati
4. Testirajte algoritam simulacijom korak po korak.

Ovu strategiju ćemo stalno koristiti, ali za momenat ćemo se baviti samo opisom navedenih koraka.

Prvi korak – Istraživanje – zahteva pažljivo proučavanje definicije problema koji se rešava, kao i detaljnu analizu termina i pojmova koji se u toj definiciji koriste. U primeru sa pravljenjem čaja uočavamo niz procesa – punjenje čajnika, stavljanje na ringlu i tako dalje. Tu su takođe odluke i varijable.

Drugi korak – Preliminarni algoritam – je prvi pokušaj rešavanja problema. taj prvi pokušaj može biti i veoma grub, ponekad i nepravilan, ali će korak 2.2 otkriti sve nedostatke i pomoći da se dođe do korektnog rešenja.

Treći korak – Finalizacija – je i najteži, zahteva iskustvo, strpljenje i preciznost. takođe je za uspešnu realizaciju ovog koraka potrebno vladanje nekim programerskim veštinama koje se stiču samo dugotrajnim praktikovanjem – vežbanjem. U pragmatičnom delu ovog predmeta, kada se izučava specifičan programski jezik (Visual Basic ili C++) takva praktična znanja i veštine biće razvijene do novoa samostalnog razvoja programa od ideje do gotovog izvršnog programa.

Sada ćemo analizirati jedan nešto složeniji problem i algoritam kojim se on rešava.

Problem stabilnih parova

Pretpostavimo da u jednom gradu na jugu Srbije imamo n mladića i n devojaka. Svi se međusobno poznaju, jer to je jedno malo mesto. Svaki mladić ima rang listu svih devojaka iz tog gradića, tako da se na toj listi na prvom mestu nalazi devojka koja mu se najviše sviđa, a na poslednjem n -tom mestu devojka koja mu se najmanje sviđa. Istu takvu listu imaju i sve devojke, ali sa rang listom svih n mladića. Te liste su nam unapred poznate na neki način (Radio Milevom).

Cilj nam je da oženimo momke i devojke tako da svi budu sretni, a da brakovi budu stabilni. Videćemo uskoro šta pod tim podrazumevamo.

Definicija. Skup brakova je stabilan ako ne postoji par mladić-devojka koji se jedno drugom sviđaju više nego njihovi bračni drugovi.

Na primer, pretpostavimo da su Marko i Jovana u jednom, a Petar i Milica u drugom braku. Na nesreću Jovani se više sviđa Petar nego Marko, a Petru se sviđa više Jovana nego Milica. Ovo znači da bi Jovana i Petar bili srećniji da su zajedno. To može da bude razlog za bračnu prevaru, pa se njihovi brakovi mogu smatrati nestabilnim.

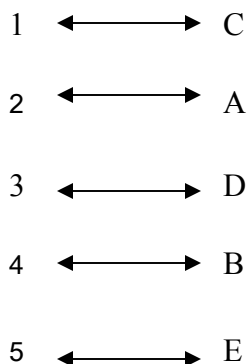
Matematički se može dokazati da uvek postoji stabilno uparivanje mladića i devojaka, takvo da ne postoji ni jedan nestabilan par. (Takođe se može matematički dokazati da ne postoji stabilno uparivanje ako bi dozvolili da se bračno uparuju osobe istog pola.).

Veoma je interesantno da ovaj algoritam u SAD-u ima jednu veoma praktičnu primenu. Tamo se svake godine svršeni studenti medicine upućuju (mladići) na praksu u bolnice (devojke) upravo korišćenjem ovog algoritma. Inače, algoritam su prvi put opisali D. Gale i L.S. Shapley 1962 godine.

Hajdemo sada da problem razmotrimo na jednom konkretnom slučaju 5 devojaka i 5 mladića. Mladiće ćemo označiti brojevima 1-5, a devojke slovim A-E. Sledeća tabela pokazule njihove rang liste.

mladići	devojke
1: CBEAD	A : 35214
2: ABECD	B : 52143
3: DCBAE	C : 43512
4: ACDBE	D : 12345
5: ABDEC	E : 23415

Pokušajmo najpre jednu prostu (grabljivu) strategiju. Krenimo redom od prvog mladića i svima dajmo devojku koja je najviše kotirana na njegovoj listi, a još uvek je “raspoloživa”. To bi nam dalo sledeći rezultat uparivanja:



Da proverimo stabilnost ovako napravljenih parova. Mladići 1, 2 i 3 su dobili svoje omiljene devojke, tako da im neće pasti na pamet da jure za drugima. Mladić 4 može biti problematičan jer mu se više sviđa devojka A od one koju je dobio (B), ali je devojka A njega stavila na poslednje mesto, tako da to nije problem. Međutim mladiću 4 se više sviđa i devojka C one koju je dobio, a devojci C se on takođe više sviđa od onog kojeg je ona dobila (1). Znači mladić 4 i devojka C su potencijalni brakolomci. Uparivanje je, dakle, nestabilno. Mogli bi sada da pokušamo da nekom kombinatorikom to popravimo, ali bi pri tom mogli da formiramo druge nestabilne parove, i da vrteći se u krug ne nađemo rešenje. I to za samo 5 parova. A, šta bi bilo da je 100-tinak (ili više) parova u pitanju. Izgleda kao nemoguć zadatak. Ali, generalno rešenje postoji i to veoma elegantno.

Algoritam

Sada ćemo opisati jedan algoritam koji se odvija u nekoliko dana i koji dovodi do stabilnog uparivanja.

Svakod dana se ponavlja sledeći ritual (scenario):

Jutro: Svaka devojka izađe na svoj balkon. Svaki mladić dođe ispod balkona devojke koja je na prvom mestu njegove rang liste i peva joj serenadu. Ako mladiću nije ostala ni jedna devojka u listi, on ostaje kod kuće (i radi domaće zadatke iz programiranja).

Popodne: Svaka devojka koja ispod svog balkona ima udvarače (koji pevaju serenade), jednom od njih koji je najviši na njenoj rang listi kaže: “Možda, dođi sutra.”, a svim ostalima kaže: “Nikad se neću udati za tebe. Šetaj”.

Uveče: Svaki mladić koji je dobio “korpu” iz svoje liste briše devojku kod koje nema nikakve šanse.

Kada posle nekoliko dana ujutro ispod svakog balkona bude samo po jedan mladić, devojke uzimaju te mladiće i tako se formiraju parovi. Tako formirani parovi biće stabilni u skladu sa našom definicijom stabilnosti. Tu je i kraj našeg algoritma. Algoritam sadrži sve tri vrste koraka: sekvencu, selekciju i repeticiju.

Siže predavanja

U ovom predavanju ste saznali o:

- algoritmima i njihovim osnovnim elementima – sekvenci, odluci i ponavljanju
- različitim načinima prikaza algoritama
- varijablama, tipovima varijabli i konvencijama za davanje imena varijablama
- strategiji projektovanja algoritama

Algoritam je zapravo plan (scenario) kako će problem biti rešen, a skoro svi algoritmi imaju iste osobine i sastavljeni su od istih elemenata (teorema o strukturi). Postoji više načina za izražavanje (prikazivanje) algoritama, a neki od tih načina su ovde pomenuti. Svaki algoritam koristi podatke koji se mogu menjati tokom rada algoritma. Za projektovanje (izradu) algoritma dobro je imati strategiju. Jedna moguća strategija prikazana je u ovom predavanju.

Sada ste već spremni za projektovanje programa, ali pre toga u narednom predavanju bavićemo se jednom temom koja projektovanje programa stavlja u širi kontekst projektovanja softverskih sistema i softversko inženjerstvo uopšte.

Pitanja

1. Definišite pojam *algoritam*.
2. Šta se podrazumeva kad kažemo da je neki algoritam *efektivan*?
3. Kakvo je značenje reči *konačan* u vezi sa algoritmima?
4. Navedite tri osnovna elementa algoritama.
5. Koja su to dva konstruktora odluke?
6. Šta je to tvrdnja?

7. Koja su to dva konstruktora ponavljanja?
8. Navedite četiri načina prikazivanja algoritama.
9. Šta je to varijabla?
10. Dajte neki primer varijable iz svakodnevnog života.
11. Šta je to tip podatka?
12. Navedite tri osnovna tipa podatka koji se mogu pojaviti u algoritmima.
13. U predavanju je prikazana jedna strategija za projektovanje algoritama. Navedite osnovne korake te strategije. Možete li napraviti neku svoju strategiju - opišite.