UNIVERZITET
METROPOLITAN
BEOGRAD

KI102 - OSNOVE PROGRAMIRANJA

Tehnike rešavanja problema, Osnove programiranja

Lekcija 01

PRIRUČNIK ZA STUDENTE

KI102 - OSNOVE PROGRAMIRANJA

Lekcija 01

TEHNIKE REŠAVANJA PROBLEMA, OSNOVE PROGRAMIRANJA

UVOD

U okviru ove lekcije naučićemo sledeće:

Računar (computer) u najgrubljem prevodu znači: onaj koji računa, ili onaj koji preračunava. Kao što vidite, to je veoma kratka definicija za nešto što je, u samo nekoliko poslednjih decenija, promenilo način života savremenog društva.

Računari (ili kompjuteri) su danas zastupljeni u svim sferama našeg života: plaćanje računa, vožnja automobila, korišćenje telefona, kupovina, itd. Zapravo, bilo bi mnogo lakše nabrojati one sfere života u kojima računari nemaju primenu. Vi ste se najverovatnije familijarizovali sa računarom kroz korišćenje video igara, alata za kucanje teksta, veb pretraživača, i ostalih programa. Međutim, ono što treba da znate je da ovaj predmet nema za cilj da vas nauči kako da koristite računar. Ideja ovog i svakog sledećeg predmeta na kursu je da vas nauči kako da računar isprogramirate tako da on uradi upravo ono što vi želite.

Računar nam omogućava da postavljene zadatke obavimo što brže, efikasnije i tačnije nego što bi te iste zadatke uradili sami i ručno — mada neke čak ne možemo da rešimo ni ručno. Kako bi od moćne mašine kao što je računar napravili jedan koristan alat, neophodno je da ga isprogramiramo. To znači da mi moramo da specificiramo računaru šta želimo da on uradi za nas i na koji način. To radimo kroz programiranje.

Računar naime nije inteligentan. On ne može da analizira problem i da onda osmisli rešenje za njega. Čovek (programer) je taj koji mora da analizira problem, da razvije skup instrukcija za postavljeni problem, a tek onda da prosledi računaru te instrukcije koje on treba da izvrši. Jednom kada napišemo program koji se izvršava na računaru, računar taj program može da izvši nebrojano puta i to veoma brzo i konzistentno, i na taj način nas oslobađa od ponavljanja dosadnih poslova i zadataka.

Da bi napisali program koji će računar da izvrši, moramo da sprovedemo proces koji se sastoji iz nekoliko faza i koraka. U okviru ovog kursa ćemo se upravo baviti tim koracima, kao i metodama koje treba koristiti da bi se postavljeni problem rešio što efikasnije.

Poglavlje 1

Računar, programiranje, softver

ŠTA JE PROGRAMIRANJE?

Da bi računar izvšio ono što tražimo od njega, on mora biti isprogramiran.

Ovaj predmet, a i ceo ovaj kurs, je o programiranju. Šta je ustvari programiranje? Termin programiranje znači kreiranje i razvoj softvera, koji se takođe naziva i program. Prostije rečeno, softver se sastoji iz instrukcija koje govore kompjuteru—ili drugom računarskom uređaju—šta on treba da radi.

Softver je svuda oko nas, čak i u uređajima za koje mislite da ih nikada nećete koristiti. Naravno, vama je poznato da softver postoji u računarima, ali softver igra bitnu ulogu u procesu upravljanja avionom, automobilima, radu mobilnih telefona, pa čak i tostera. Na personalnom računaru, uglavnom koristite alat za procesiranje reči u cilju pisanja dokumenata, pregledač veba (web browser) da surfujete ineternetom, imejl (email) programe da šaljete poruke. Programeri kreiraju softver uz pomoć moćnih alata koji se nazivaju programski jezici.

Ljudsko ponašanje i ljudske misli su uobičajeno okarakterisane logičnom sekvencom postupaka koji mogu biti primenjeni na neki objekat. Još od ranog detinstva vas uče kako da se ponašate, kako da izvršavate razne stvari; takođe ste sebe naučili da očekujete i odgovarajuće ponašanje od drugih ljudi. Puno stvari koje radite svakodnevno ustvari radite automatski. Na sreću, nije neophodno da svesno razmišljate o svakom koraku koji je uključen u taj proces, već to radite kao da rukom okrećete stranicu knjige:

1. Podigni ruku.
2. Pomeri ruku na desnu stranu knjige.
3. Uhvatite gornju-desnu ivicu stranice.
4. Pomerite ruku s desna na levo sve dok je stranica postavljena tako da možete da čitate ono što je na sledećoj strani.
5. Sklonite ruku sa stranice.

Zamislite samo koliko neurona morate da aktivirate i koliko mišića treba da pokrenete, sve u odgovarajućem i unapred defisanom redosledu, da bi ste pomerili ruku i šaku. A vi to sve radite potpuno nesvesno.

Sve što vi sada radite nesvesno je nešto što ste nekada morali da naučite kako se radi. Samo pokušajte da posmatrate kako beba skoncentrisano pomera jednu nogu ispred druge dok uči da hoda. Zatim zamislite grupu petogodišnjaka kako igraju igru „šuge“.

Šire posmatrano, oblast matematike nikada ne bi mogla biti razvijena bez logične sekvence koraka manipulisanja simbola u cilju rešavanja problema ili dokazivanja neke teoreme. Masovna proizvodnja, takođe, ne bi mogla biti izvedena bez operacija koje se izvršavaju nad pojedinačnim delovima i u unapred definisanom redosledu. Naša cela civilizacija se, takođe, bazira na jasno definisanom redosledu stvari i postupaka.

Mi kreiramo redosled, istovremeno svesno i nesvesno, kroz proces koji se naziva programiranje. Ovaj predmet, a i ceo kurs, se bavi programiranjem jednog specifičnog alata a to je *elektronski računar*.

ŠTA JE RAČUNAR?

Računar je elektronski uređaj koji memoriše i obrađuje podatke.

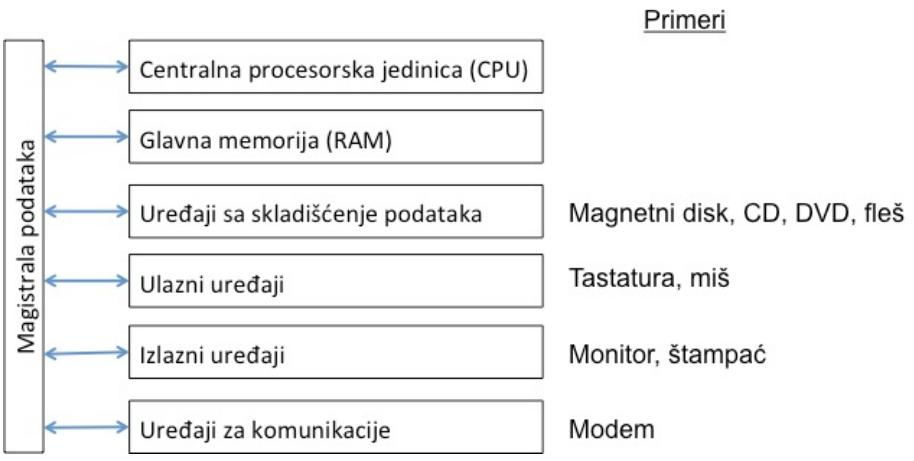
Računar čini hardver i softver. Hardver čini fizički deo računara, znači ono što se vidi, a softver je nevidljiv, jer ga čine programske instrukcije koje kontrolišu rad hardvera i koje omogućavaju da hardver izvrši određeni zadatak.

Računar čine sledeće osnovne komponente:

- ***Centralna*** procesorska jedinica (CPU-Central Processing Unit)
- Glavna memorija (main memory)
- Uređaj za skladištenje podataka (storage device)
- Uređaji za unos podataka (input devices)
- Uređaji za prikazivanje podataka (output devices)
- Uređaji za komunikaciju (communication devices)

Komponente računara su međusobno povezane sistemom koji se naziva: *magistrala podataka* (data bus). Preko magistrale podataka komponente računara razmenjuju podatke. Kod personalnih računara (PC-Personal Computer) magistrala podataka je ugrađena u osnovnu elektronsku ploču računara (motherboard) koja sadrži eleketronska kola koja povezuju sve elemente računara.

Slika 1 prikazuje osnovnu strukturu računara.



Slika 1.1 Osnovna struktura računara

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PODATAK, INFORMACIJA, PROGRAMIRANJE

Računar nam omogućava da zadatke izvršimo efikasnije, brže i tačnije nego što bi to uradili ručno

Ključna reč u definiciji računara je ustvari *podatak* (data). Računari naime manipulišu podacima. Kada pišete program (ili plan) za vaš računar, vi ustvari definišete osobine podataka i operacije koje vršite nad njima. Ove operacije se zatim međusobno kombinuju, ukoliko je potrebno, da bi se rešio neki problem.

Podatak je informacija predstavljena u formi koju računar može da koristi—na primer, brojevi ili slova.

Informacija je bilo koje znanje koje može biti preneto, uključujući apstraktne ideje ili koncepte kao što je na primer izjava: “Zemlja je okrugla.”

Podaci mogu da se jave u različitim oblicima: slova, reči, celi brojevi, decimalni brojevi, datumi, vremena, koordinate na mapama, itd. Bilo koji tip informacije može virtuelno biti predstavljen kao podatak, ili kao kombinacija podataka i operacija nad njima. Za svaku vrstu podataka na računaru se kaže da ima specifičan *tip podatka* (data type). Na primer, ako kažemo da su dva podatka tipa *Integer*, mi znamo da su ovi podaci zapisani u memoriji i da možemo da primenimo aritmetičke operacije nad njima.

Kao i program koncerta, gde je tačno izlistano koje će pesme biti izvedene i koji će ih izvođači izvoditi, tako i kompjuterski program sadrži listu tipova podataka koji će biti korišćeni i sekvencu koraka koje računar izvršava nad podacima.

Od sada pa nadalje, kada koristimo reči *programiranje* i *program*, mislićemo na pojmove *programiranje računara* i *računarski program*.

Programiranje - Planiranje izvršavanja zadataka ili događaja
Elektronski računar - Programabilni uređaj koji može da čuva, prima i procesira podatke
Podatak (Data) - Informacije u obliku koji računar može ra koristi
Informacija Bilo koje znanje koje može biti saopšteno
Tip podatka - Specifikacija o tome kako je informacija predstavljena u računaru kao podataka i skupp operacija koje mogu biti primenjene pri radu sa tim podatkom
Programiranje računara - Proces specificiranja tipova podatka i operacija koje računar treba da primerni na podacima kako bi rešio postavljeni problem
Računarski program - Specifikacija tipova podataka i instrukcija za izvršavanje operacija koje računar koristi kako bi rešio postavljeni problem

Računar nam omogućava da zadatke izvršimo efikasnije, brže i tačnije nego što bi to uradili ručno—ako ih čak možemo i rešiti ručno. Da bi od moćne mašine kao što je računar načinili koristan alat, moramo da je isprogramiramo. To znači da moramo da specificiramo tačno šta želimo da ona uradi za nas i na koji način. To upravo radimo kroz *programiranje*.

ŠTA JE SOFTVER?

Softver obuhvata programe, tj. listu instrukcija koje procesor treba da izvrši i podatke koje računar koristi.

Softver obuhvata programe i podatke koje računar koristi; nalazi se na nekom uređaju, na primer, na disku, ali je sam po sebi neopipljiv.

Programi su lista instrukcija koje procesor treba da izvrši.

Podaci mogu biti bilo koje informacije koje program koristi. To mogu biti brojčani podaci, tekst, zvučni ili neki drugi podaci. Razlika između programa i podataka nije baš tako jasna kako bismo možda zaključili.

I programi, i podaci se u memoriji računara predstavljaju na isti način. Elektronika memorije računara ne pravi razliku između programa i podataka. Upravo ta činjenica da se i programi, i podaci skladište na isti način, jedna je od najvažnijih ideja u računarskim naukama. Računarski sistemi svoju memoriju koriste za ono što potrebe nalažu.

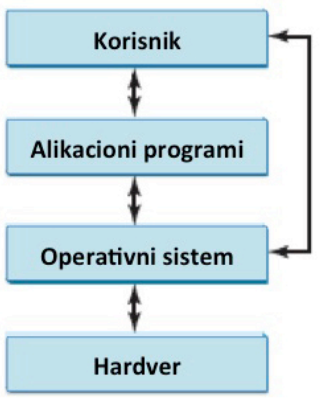
Postoje dva tipa softvera (programa), Slika-2.

1. **Aplikacioni programi** (aplikacije) su programi koje ljudi koriste za obavljanje nekog konkretnog posla. Računari i postoje upravo zato što ljudi imaju potrebu da koriste takve programe.
2. **Sistemski softver** (programi) omogućavaju da hardver i softver jednog računarskog sistema rade kako treba.

Najvažniji sistemski program je **operativni sistem**. Ovaj program je uvek prisutan kada računar radi i on koordinira rad svih hardverskih i softverskih komponenti sistema.

Operativni sistem je odgovoran za pokretanje aplikacionih programa; za pronalaženje resursa koji su potrebni za rad tih programa. Tokom rada nekog aplikacionog programa operativni sistem rukuje detaljima vezanim za hardver koji su, tom prilikom - potrebni.

Na primer, kada na tastaturi unesete neki karakter, operativni sistem određuje kojem programu je to namenjeno.



Slika 1.2 Osnovni tipovi softvera - aplikacioni i operativni

Poglavlje 2

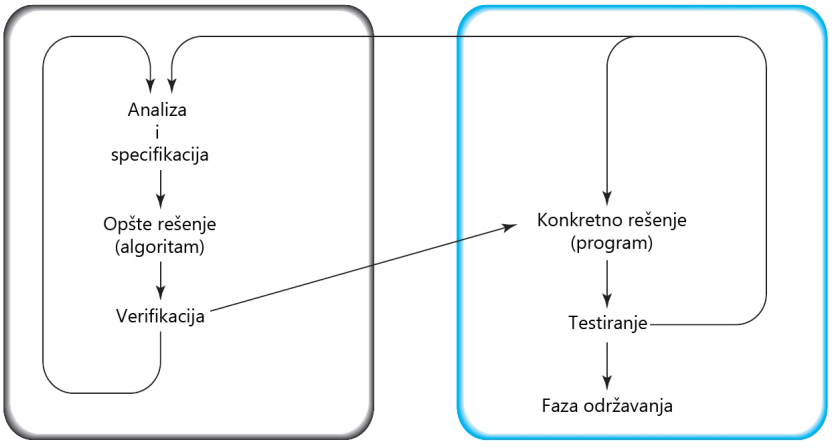
Rešavanje problema. Kako pišemo programe?

KAKO PIŠEMO PROGRAME?

Moramo da izvršimo proces koji se sastoji iz dve faze: rešavanje problema i implementacija

Računar naime nije inteligentan. On ne može da analizira problem i da zatim osmisli rešenje za njega. Čovek (programer) je taj koji mora da analizira problem, da razvije skup instrukcija za postavljeni problem, a tek onda da prosledi računaru te instrukcije koje on treba da izvrši. Koja je onda svrha računara ako on ne može da reši naš problem? Naime, jednom kada napišemo program koji se izvršava na računaru, računar taj program može da izvrši nebrojano puta i to veoma brzo i konzistentno, i na taj način nas oslobađa od ponavljanja dosadnih poslova i zadataka.

Da bi napisali program koji će računar izvršavati, moramo da izvršimo proces koji se sastoji iz dve faze: *rešavanje problema* (problem solving) i *implementacija* (implementation), Slika-1.



Slika 2.1 Proces programiranja

Faza rešavanja problema

1. Analiza i specifikacija. Razumevanje problema i kako rešenje treba tačno da se ponaša.
2. Opšte rešenje (algoritam). Specifikacija neophodnih tipova podataka i logične sekvence koraka koja rešava problem.
3. Verifikacija. Pažljivo praćenje koraka u izvršavanju da bi videli da li formirano programsko rešenje rešava postavljeni problem.

Faza implementacije

1. Konkretno rešenje (Program). Prevođenje algoritma (opšteg rešenja) u konkretan programski jezik.
2. Testiranje. Provera da li računar pravilno prati zadate instrukcije. U tu svrhu ručno proveriti rezultate. Ako pronađete greške onda analizirajte program i algoritam da bi otkrili uzrok greške. A zatim izvršiti korekcije.

Nakon što je program korektno napisan, i izvršeno osnovno testiranje, ulazi se u treću fazu: *održavanje* (maintenance).

Faza održavanja
1. Korišćenje. Koristiti program.
2. Održavanje. Modifikovati program da bi ispratili zahteve tržišta, zahteve korisnika, ili da bi korigovali bilo koju grešku koja je otkrivena u fazi korišćenja.

ANALIZA PROBLEMA I PROJEKTOVANJE ALGORITMA

Algoritam je pismeni ili usmeni opis sekvence akcija koje se primenjuju nad objektima.

Programer započinje proces programiranja analiziranjem problema, cepkanjem problema na manje celine koje su lakše za rešavanje, i razvojem opšteg rešenja za svaki deo slagalice, pri čemu se to rešenje naziva *algoritam* (algorithm).

Rešenja za svaki deo se spajaju u jednu celinu koja predstavlja program za rešavanje originalnog problema. Proces razumevanja i analize problema zahteva mnogo više vremena nego što se to može zaključiti sa Slike-1. Ova dva procesa predstavljaju srž procesa programiranja.

> Algoritam - Instrukcije za rešavanje problema ili podproblema u konačnom vremenu korišćenjem konačne količine podataka

Ako naš računarski program i projektovani algoritam deluju slično, to je zbog toga što je program ustvari jedan algoritam napisan da bi se izvršio od strane računara.

Algoritam je pismeni ili usmeni opis logične sekvence akcija koje se primenjuju nad odgovarajućim objektima.

Mi, u realnom životu, koristimo algoritme svakodnevno. Recepti, instrukcije, smernice, samo su neki od primera algoritama koji se na izvršavaju na računaru. Na primer, kada startujete vaš auto, vi izvršavate proceduru korak-po-korak. Algoritam može da bude predstavljen na sledeći način:

> 1. Ubacite ključ.
> 2. Proverite da li je menjač izbačen iz brzine.
> 3. Okrenite ključ u desno do pozicije startovanja (start).
> 4. Ako se motor startuje u roku od 6 sekundi, pustiti ključ da se vrati sam u poziciju paljenja (ignition).

5. Ako se auto ne upali u roku od 6 sekundi, pustiti ključ i skloniti nogu sa pedale gasa, sačekati 10 sekundi, ponoviti korake od Koraka 3 do 5, ali ne ponavljajte više od 5 puta.
6. Ako se auto ne upali, pozovite servis.

Bez navedene instrukcije *"ali ne više od 5 puta "* u Koraku 5, mogli bi ste beskonačno da pokušavate da upalite automobil. Zašto? Zato što ako nešto nije u redu sa automobilom, onda ponavljanje koraka od Koraka 3 do 5 iznova i iznova sigurno neće pokrenuti motor automobila. Ovaj tip situacije koja se nikad ne završava je poznat kao *beskonačna petlja* (infinite loop). Ako izostavimo instrukciju *"ali ne više od 5 puta"* iz Koraka 5, procedura se ne bi poklapala sa definicijom jednog algoritma. Naime, algoritam mora da se završi u konačnom vremenskom intervalu za sve moguće uslove. 

Nakon razvoja opšteg rešenja, programer testira algoritam, obrađuje svaki korak ručno, korišćenjem papira i olovke. Ako algoritam ne radi, programer ponavlja proces rešavanja problema, analizirajući problem ispočetka i osmišljava novi, korigovani algoritam. Kada je programer zadovoljan sa algoritmom, on ili ona prevodi algoritam u programski jezik.

PREVOĐENJE ALGORITMA U PROGRAMSKI JEZIK

Prevođenje algoritma u programski jezik se naziva kodiranje algoritma

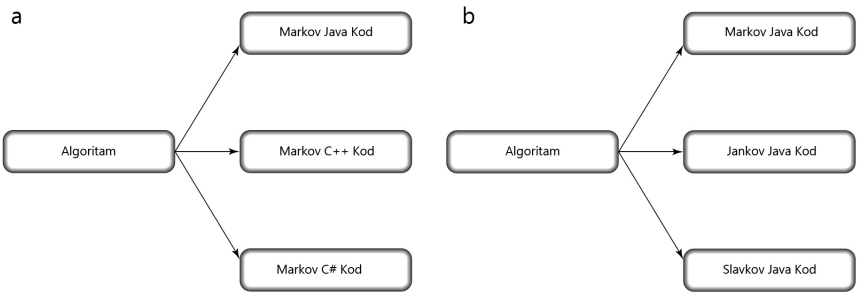
Programski jezik je uprošćena verzija engleskog jezika (sa matematičkim simbolima kao dodacima) koja se pridržava striktnog skupa gramatičkih pravila. Engleski jezik je ionako suviše komplikovan i dvosmislen da bi računari mogli jasno da ga razumeju. Programski jezici su znatno prostiji jer su njihov fond reči i gramatika znatno ograničeni i redukovani.

Iako programski jezik ima prostu formu, on često nije lak za korišćenje. Pokušajte nekome da date instrukcije da dođe do najbližeg aerodroma korišćenjem ograničenog rečnika od ne više od 25 reči, i primetićete problem. Programiranje vas stoga tera da pišete veoma proste i precizne instrukcije.

Prevođenje algoritma u programski jezik je naziva *kodiranje* (coding) algoritma. Proizvodi prevođenja—kod za sve algoritme u problemu— su testirani spajanjem svih struktura u jedinstvenu celinu i izvršavanjem (executing) programa na računaru. Ako program ne uspe da proizvede željeni rezultat, programer mora da ga testira i otkloni greške (debug) — što znači da mora da otkrije šta je loše i da onda koriguje program, ili da koriguje jedan ili više algoritama, kako bi popravio rad programa. Kombinacija procesa kodiranja i testiranja algoritama se naziva *implementacija* (implementation).

Kod (code) je proizvod prevođenja algoritma u odgovarajući programski jezik. Termin kod može da se odnosi na kompletan program ili na jedan deo programa.

Ne postoji jedinstven način da se implementira algoritam. Na primer, jedan isti algoritam može da se prevede na nekoliko različitih programskih jezika. Svaki proces prevođenja rezultira u različitoj implementaciji:



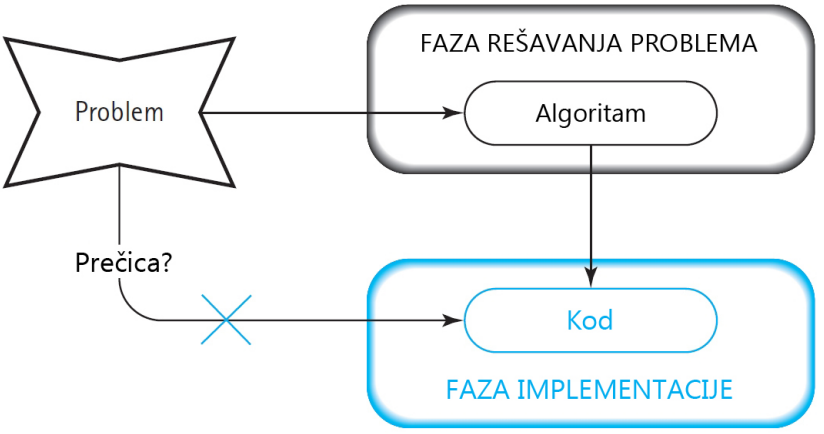
Slika 2.2 Razlike u implementaciji algoritama a) Algoritam prevedene na različite programske jezike, b) Algoritam preveden od strane različitih osoba

Čak i kada dvoje ljudi prevode algoritam u isti programski jezik, najverovatnije će se desiti da će se proizvesti različiti kodovi (Slika-2b). Zbog čega je to tako? Pa, svaki programski jezik dozvoljava programeru neku fleksibilnost i slobodu da prevede algoritam na način kako on misli da je najbolje ili najlakše za njega. Dajući ljudima ovu fleksibilnost, oni usvajaju sopstveni stil (styles) u pisanju programa, kao što postoje slobodni stilovi u pisanju kratkih priča ili eseja. Nakon što steknete neko programersko iskustvo, bićete sposobni da razvijete svoj stil pisanja programa (kodiranja). U okviru ovog predmeta, i celog kursa, biće vam preporučena najpoželjnija programerska praksa kada je u pitanju usvajanje dobrog programerskog stila.

PREČICE U PROGRAMIRANJU?

Dobra stara izreka: 'Preko preče, naokolo bliže' je primenljiva i na programiranje. Stoga prvo dobro razmilite o problemu, a tek onda kodirajte!

Neki ljudi pokušavaju da ubrzaju proces programiranja prelazeći direktno iz faze definisanja problema na kodiranje programa (Slika-3). Ova prečica deluje veoma primamljivo za programere i na prvu loptu deluje da ona može da uštedi puno vremena. Međutim, iz puno razloga koji će vam postati očigledni kako stičete nova znanja iz programiranja, ovaj tip prečice u stvari će zahtevati više vremena i napora. Detaljan razvoj opšteg rešenja pre nego što počnete da pišete program će vam veoma pomoći da lakše savladate problem, da usredsredite misli na rešavanje problema, i da izbegnete greške. Ukoliko ne utrošite više vremena na početku da bi bolje osmislili i doterali vaš algoritam, utrošićete sigurno puno više vremena da bi ste kasnije korigovali greške i ispravili vaš program.



Slika 2.3 Prečice u programiranju?

Stoga prvo razmilite dobro, a tek onda kodirajte! Što ranije počnete da kodirate program, to će vam kasnije više vremena trebati da napišete program koji radi.

ODRŽAVANJE SOFTVERA

Od trenutka kada je softver pušten u upotrebu počinje faza održavanja softvera. Ona može da oduzme više vremena nego sam proces implementacije koda i testiranja.

Jednom kad je program pušten u upotrebu, veoma često je potrebno modifikovati ga. Modifikacije (izmene) mogu da uključuju ispravljanje grešaka koje su otkrivene u toku faze korišćenja programa ili izmene programa kako bi se izašlo u susret novim zahtevima korisnika programa. Nakon svake izmene u programu neophodno je ponoviti faze rešavanja problema i implementacije za onaj deo programa koji je izmenjen (da ne bi program postao nestabilan u tom delu). Ova faza procesa programiranja se naziva *održavanje* (maintenance) i u njoj se, kod većine napisanih programa, troši i najviše vremena.

Na primer, program koji je realizovan i kreiran za nekoliko meseci može da bude održavan i po nekoliko godina. Stoga je nekad poželjnije utrošiti više vremena u startu da bi pažljivo razvili početno problemsko rešenje i programsku implementaciju.

Faze rešavanja problema, implementacije i održavanja zajedno čine *zivotni ciklus* (life cycle) programa.

Kao dodatak na rešavanje problema, realizaciju algoritma i održavanje programa, pisanje dokumentacije je jedan veoma bitan deo procesa programiranja. Dokumentacija uključuje pisano objašnjenje problema koji se rešava i opis rešenja, komentare koji su uključeni u sam kod, i korisničko uputstvo koje opisuje kako se koristi program. Pošto je uobičajeno da veliki broj ljudi radi na istom programu u toku dugog vremenskog perioda, neophodno je da svaki od tih ljudi bude u mogućnosti da čita i razume kod – što je i svrha pisanja dokumentacije.

Dokumentacija - Pisani dokument koji uključuje tekst ili komentare koji olakšavanju drugima da bolje razumeju program, da ga lakše koriste, i po potrebi lakše unesu izmene u njega

PRIMERI

Navodimo primere realnih algoritama iz svakogdnevnog zivota

Primeri realnih problema

Primer 1: Ako gledate na TV-u vesti i želite da prebacite na sportski kanal neophodno je da uradite nešto, tj. neophodno je da pritisnete odgovarajuće dugme na daljinskom upravljaču. Ovo je jedan tip rešavanja problema.

Primer 2: Jednog ponedeljka ujutru, student se sprema da ide u školi ali nije još uvek pokupio/la one knjige i skripte koje su neophodne za predavanja. Stoga je uzimanje (nabaljanje) knjiga i skripti jedan vid rešavanja problema.

Primer 3: Ako vas neko pita koje je sada vreme: ? Gledanje u sat i saopštavanje informacije o tačnom vremenu je jedan vid rešavanja problema.

Primer 4: Neki sudenti sa klase planiraju da idu na piknik i dogovaraju se da podele treoškove medju njima. Stoga je sračunavanje ukupnog troška i količine novca koju svaki od studenta treba da izdvoji za piknik ustvari jedan tip rešavanja problema.

Zadatak za samostalni rad: projektovati algoritme za postavljene probleme.

Zadatak 1. Odrediti prosečnu ocenu za sve studente u razredu (klasi).
1. Ulaz: učitati ocene svih studenata ... kucanjem na tastaturi ili učitavanjem ocena sa USB fleša ili sa har diska.
2. Obrada: sabrati sve ocene i pronaći prosečnu ocenu.
3. Izlaz: oštampati poruku na monitoru, poslati štampaču, ili sačuvati na USB fleš ili hard disk ... ili kombinacija bilo čega navedenog.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 3

Pokazni primeri

POKAZNI PRIMER: ANALIZA PROBLEMA I PROJEKTOVANJE ALGORITMA

Na prostom primeru prodaje kabla pokazujemo kako se vrši projektovanje algoritma

Projektovanje algoritma na konkretnom primeru prodaje kabla:

1. Opis problema. Radite kao službenik u jednoj firmi koja se bavi izvozom kablova. Većina kablova koji se nalaze u skladištu su izmereni u metrima. Međutim, postoje i mušterije iz Sjedinjenih Država koje naručuju kablove u stopama. Potrebno je da izračunate dužinu u stopama svih kablova u skladištu. Potrebno je napisati program za ovu konverziju.

2. Analiza. Vrednosti se moraju konvertovati iz jednog sistema merenja u drugi. Opis problema kaže da dobijate kablove sa dužinom naznačenom u metrima, stoga je ulazna vrednost u metrima. Potrebno je izračunati ekvivalentnu vrednost u stopama, što je izlaz problema.

Prvo je potrebno znati odnos metara i stopa. Prvo je potrebno znati odnos metra i stope. Na primer, u tabelama konverzije se može naći podatak da je metar jednak 39.37 inča. Takođe, poznato je da je stopa jednaka 12 inča.

3. Projektovanje. Sledeći korak je projektovanje algoritma. Počinjemo sa zapisivanjem liste tri osnovna koraka, tj. potproblema:

```
1. Pročitaj dužinu kabla u metrima
2. Konvertuj pročitanu dužinu u stope
3. Prikaži dužinu kabla u stopama
```

Dalje odlučujemo da li je potrebno osnovne korake razbiti na manje korake. Korake 1 i 3 ne treba menjati. Izmena koraka dva nam daje jasnije instrukcije kako se vrši konverzija:

```
2.1 Konvertuj dužinu kabla iz metara u inče.
2.2 Konvertuj dužinu kabla iz inča u stope.
```

Kompletan algoritam je onda sledeći:

```
1. Pročitaj dužinu kabla u metrima
    2.1 Konvertuj dužinu kabla iz metara u inče.
    2.2 Konvertuj dužinu kabla iz inča u stope.
3. Prikaži dužinu kabla u stopama
```

4. Implementacija. Sledeći korak jeste implementacija. Prevodimo algoritam zapisan u vidu liste koraka u kod nekog od programskih jezika.

5. Testiranje i verifikacija. Poredimo rezultate programa sa onim vrednostima koje dobijemo ručno ili pomoću kalkulatora. Treba imati na umu da programi mogu da sadrži greške. Testiranje možete probati tek kada prevedete program na programski jezik.

POKAZNI PRIMERI 1

Navodimo primere projektovanja algoritama radi boljeg uvida u samu analizu problema.

Nedeljna plata radnika

Pretpostavimo da programer treba da osmisli algoritam koji određuje nedeljnu platu radnika. Ako ovo sračunavamo ručno, ona bi se postupak računanja mogao napisati na sledeći način:

1. Pogledati kolika je plata radnika po radnom satu.
2. Odrediti broj odrađenih sati u toku jedne sedmice.
3. Ako je broj radnih sati manji ili jednak 40, onda pomnožiti sate sa platom po satu da bi sesračunala osnovna plata.
4. Ako je broj radnih sati veći od 40, pomnožiti 40 sa platom po satu da bi se sračunala osnovna plata, a zatim razliku izmedju radnih sati i broja 40 pomnožiti sa 1.5 da bi se sračunala plata prekovremenog rada.
5. Sabrati osnovnu platu i platu prekovremenog rada (ukoliko postoji) da bi sračunala ukupna nedeljna plata.

Koraci koje računar izvršava su često koraci koje bi ste i vi koristili ako bi ste pokušali da izračunavnanje izvršite ručno.

Cena taksi usluge

Algoritam za računanje cene taksi usluge bi mogao da se napiše na sledeći način:

1. Definisati ulazne veličine: cenu "starta" S, cenu po pređenom kilometru C, broj pređenih kilometara L i popust u procentima P
2. Izračunati cenu bez popusta T1 kao T1=S+C*L
3. Izračunati cenu sa popustom T2 kao T2=T1-T1*P/1000.
4. Prikazazi izlazne veličine: cenu bez pousta T1 i cenu sa popustom T2.

Konvertovanje temperature iz Celzijusa u Farenhajte

Treba da konvertujemo temperaturu zadatu u stepenima Celizijusa u Farenhajte. Algoritam za rešavanje ovog problema je sledeći:

1. Pročitaj vrednost temperature u Celzijusima
2. Primeni formulu za konvertovanje
3. Prikaži vrednost temperature u Farenhajtima

U ovom primeru, algoritam se sastoji od samo tri koraka.

POKAZNI PRIMERI 2

Navodimo primere projektovanja algoritama radi boljeg uvida u samu analizu problema

Problem 1: Odrediti površinu kruga poluprečnikar.
Ulaz u algoritam**:** poluprečnik r kružnice.
Očekivani izlaz **:** površina kružnice

Algoritam:

```
Korak1: Read\input the Radius r of the Circle
Korak2: Area  <- PI*r*r // calculation of area
Korak3: Print Area
```

Problem 2: Napisati algoritam koji učitava dva broja i određuje njihovu zbir.

Ulaz u algoritam: Prvi broj num1 i drugi broj num2.

Očekivani izlaz: Zbir dva uneta broja.

Algoritam:

```
Korak1: Start
Korak2: Read\input the first num1.
Korak3: Read\input the second num2.
Korak4: Sum  <- num1+num2 // calculation of sum
Korak5: Print Sum
Korak6: End
```

Problem 3. Napisati algoritam koji konvertuje dužinu iz stopa (feet) u centimetre (cm). Pseudokod:

```
Input the length in feet (Lft)
Calculate the length in cm (Lcm)
   by multiplying LFT with 30
Print length in cm (LCM)
```

Problem 4. Write an algorithm that will read the two sides of a rectangle and calculate its area. Pseudocode

```
Input the width (W) and Length (L)
    of a rectangle
```

```
Calculate the area (A) by multiplying L with W
Print A
```

Problem 5: Napisati algoritam koji izračunava vrednost 2 na 4.

Algoritam:

```
Korak 1: Base <- 2
Korak 2: Product <- Base
Korak 3: Product <- Product * Base
Korak 4: Product <- Product * Base
Korak 5: Product <- Product * Base
Korak 6: PrintProduct
```

ZADACI ZA SAMOSTALNI RAD

Na osnovu svega obrađenog dati odgovore na postavljena pitanja i rešiti zadatke za samostalni rad.

Zadatak 1. Napisati sekvencu iskaza koje opisuju postupak izvrsavanja telefonskog poziva. Da li telefonski poziv moze buiti obavljen bez kontrolnih struktura?

Zadatak 2:
Objasniti zašto sledeća lista koraka ne predstavlja jedan algoritam, a zatim listu pretvoriti u algoritam.
Pranje kose:
1. Isperi.
2. Nanesi šampon.
3. Ponovi.

Zadatak 3. Napisati algoritme za sledeće instrukcije:

- Pravljenje sendviča sa džemom ili buterom
- Sastavljanje delića slagalice
- Igranje igre muzičke stolice
- Zamena točka na automobilu
- Povratak od škole do kuće
- Kupovina namirnica (od ulaska u radnju do izlaska iz radnje)

ZAKLJUČAK

Na osnovu svega obrađenog možemo zaključiti sledeće:

Televizor je za nas "crna" i nepoznata kutija. Jedino o čemu razmišljamo je kako ćemo da ga upalimo, da se uvalimo u fotelju i da ga gledamo. Znamo da je TV sredstvo komunikacije koje se koristi da bi poboljšalo naše živote. Tako i računari postaju uobičajena stvar kao i televizori, jednostavno: sastavni deo naših života. I slično kao kod televizora, računari se baziraju na složenim principima ali su dizajnirani tako da su laki za korišćenje.

Računari nisu baš inteligentni; mora im se precizno reći šta da urade. Stoga su prave računarske greške veoma retke (obično usled otkaza delova ili električnih kvarova). Pošto mi govorimo računarima šta treba da urade, najveći deo grešaka koje prave računari su ustvari ljudske greške.

Programiranje računara je proces planiranja sekvence koraka u računaru koji se primenjuju na podatke. Programiranje uključuje faze rešavanja problema i implementacije. Nakon analize problema, mi razvijamo i testiramo opšte rešenje (algoritam). Ovo opšte rešenje zatim postaje konkretno rešenje — naš program — onda kada ga napišemo u programskom jeziku visokog nivoa. Nakon što se isprave sve greške u kodu ili bagovi (bugs) koji se otkriju u toku testiranja prorama, naš program je spreman za upotrebu.

Onog trenutka kada počnemo da koristimo naš program počinje *faza održavanja programa* (maintenance phase). Održavanje programa uključuje korigovanje grešaka otkrivenih u toku korišćenja programa, od strane korisnika) i izmenu programa kako bi se zadovljile potrebe korisnika za dodatnim izmenama.

Rekli smo da je rešavanje problema sastavni deo procesa programiranja. Iako možda imate malo iskustva u programiranju, sigurni smo da imate puno iskustva u rešavanju probolema. Ključna stvar u programiranju je da za trenutak zastanete sa kodiranjem, da dobro razmislite o strategijama koje koristite da bi ste rešili problem, a zatim i da iskoristite neku od strategija kako bi ste konstruisali primenljiv algoritam. Neke od ovih strategija mogu da budu: postavljanje pitanja, potraga za stvarima sa kojima ste familijarizovani, rešavanje po analogiji, podela problema na manje podprobleme, korišćenje već poznatih rešenja za manje probleme kako bi rešili veliki problem, spajanje rešenja, i preformulacija problema kako bi ste prebrodili mentalnu blokadu koja eventualno može da vam se javi.

Računari se danas veoma puno koriste u nauci, inženjerstvu, poslovanju, državnoj upravi, medicini, proizvodnji dobara, i u umetnosti. Učenje programiranja u Javi vam može omogućiti da ovaj moćan alat koristite još efikasnije.