POMOĆNI MATERIJALNI ZA PRIPREMU TEORIJSKOG KOLOKVIJUMA IZ PREDMETA APLIKATIVNI PROGRAMI

Podsećanje iz Kompjuterske tehnologije:

Računarski sistem kao svoju glavnu ulogu ima obradu (procesiranje) podataka. Da bi svoj zadatak uspešno izvršio, računarski sistem se sastoji od dve grupe komponenti: hardvera i softvera. **Hardver** čine sve elektronske, tehničke i uopšte fizički opipljive komponente. Za funkcionisanje računara najvažnije su elektronske hardverske komponente, digitalni elektronski sklopovi. Oni se uglavnom nalaze u kućištu (računar u užem smislu). Integrisani su kao sastavni delovi matične ploče, ili postavljeni u njene slotove (priključke). To su pre svega **(mikro)procesor**, čip u kome se vrši sama obrada podataka, i **operativna (radna) memorija**, koja je dovoljno brza pri upisu i čitanju podataka da služi za pamćenje međurezultata u toku obrade. Kako podaci u radnoj memoriji ne ostaju trajno zapisani u njoj i posle isključenja napajanja, to postoji i **spoljna (masovna, diskovna) memorija**, koja, iako zbog nedovoljne brzine čitanja i upisa ne može pratiti procesor u izvršenju obrade podataka, može trajno čuvati podatke. Tako se krajnji rezultati obrade uvek smeštaju u spoljnu memoriju (najčešće *hard-disk*).

Softver predstavlja svu logiku funkcionisanja gotove mašine (hardvera), ugrađenu od strane programerâ, za izvršenje postavljenih zadataka. Ova logika ostvaruje se u vidu *programa*, odnosno nizova instrukcija na određen način saopštenih računaru. Ako se postavljeni zadaci odnose na sâmo funkcionisanje računara, bilo za usklađivanje hardverskih uređaja, ili za podršku pri kreiranju programa za obavljanje poslova korisnika, onda se radi o **sistemskom softveru**. Programi namenski kreirani za obavljanje određenih korisničkih poslova spadaju u grupu **aplikativnog (korisničkog) softvera**.

Najvažniji sistemski softver svakako je *operativni sistem*, koji upravlja logikom funkcionisanja celog računarskog sistema i raspodelom resursâ računara (pre svega procesorom i memorijom, kao i strukturom podataka u diskovnoj memoriji, tzv. *fajl sistemom = sistemom datoteka*).

U grupi aplikativnih programa razlikujemo:

- **aplikativne programe opšte namene**, koji se koriste za obavljanje opštih poslova, npr. obrada teksta (*Microsoft(MS) Word, Corel Word Perfect, LaTex* i dr.); grafički dizajn (*CorelDraw* za vektorisanu grafiku i *Adobe PhotoShop* za bitmapiranu grafiku, *3D Dream, Maya* za prostornu animaciju); kompjutersko projektovanje - CAD alati - za tehničko crtanje *AutoCAD*, za projektovanje i simulaciju električnih kola *OrCAD*, matematiku *MathCAD, MATLAB* i dr.; baze podataka (*MS Access, Oracle, MS SQL Server, MySQL* i dr.) i
- **aplikativne programe posebne namene**, odnosno **namenske programe**, koji su za suviše specijalizovane primene, bilo zbog malog broja korisnika, ili u kojima bi primena opštijih programa

zahtevala mnogo programerskog i projektantskog znanja i angažovanja; tu spadaju npr. *poslovni informacioni sistemi* - svi se zasnivaju na bazama podataka, ali se namenski projektuju i programiraju zbog specifičnosti pojedinih poslovnih logika; *sistemi za podršku odlučivanju* i dr.

Aplikativni programi opšte namene često se grupišu u *programske pakete*, po srodnosti poslova za koje su namenjeni, npr. *MS Office, OpenOffice* i dr. za kancelarijsko poslovanje, *Corel Graphic Suite* i *Adobe* paketi za grafički dizajn i sl.

RAČUNARSKO REŠAVANJE PROBLEMA

"*Programiranje kroz aplikativne programe*", do 23. str.

1. Definisanje problema, razrada matematičkog modela (kod objektno-orijentisanog programiranja kreira se i objektni model);
2. Izbor metode numeričkog rešavanja, pri čemu se uzima u obzir tačnost rešenja, brzina dobijanja rezultata, zahtev za operativnom memorijom, složenost programske realizacije;
3. Razrada algoritma i struktura podataka – *pseudokôd* za (tekstualni) opis algoritma;
4. Realizacija procedura na osnovu algoritma na programskom jeziku – programiranje;
5. Priprema zadatka za računar, unos programa u računarsku memoriju (uglavnom sistemski softver: editori, prevodioci - kompajleri i interpreteri, linkeri, programi za ispravljanje grešaka = debageri, do pojave *integrisanih razvojnih okruženja*);
6. Testiranje i ispravljanje grešaka u programu;
7. Izrada dokumentacije ;
8. Rešavanje zadatka na računaru, obrada i formiranje rezultata.

PROGRAMSKI JEZICI I PRINCIPI PROGRAMIRANJA

Jedini "jezik" koji "razume" računar svodi se na rad hardvera – to je **mašinski jezik**. On se sastoji samo od nula i jedinica koje su grupisane u *mašinske instrukcije* (*operacije*). Raspoređivanjem informacija koje dospevaju u računar i odlučivanjem kako i kada će se izvršiti određena operacija rukovodi *upravljačka jedinica procesora*. Njen rad zasniva se na interpretiranju skupa *instrukcija.* Instrukcije sadrže informacije o operaciji koju treba izvršiti (u svom delu *kôd operacije*) i gde se nalaze *operandi*, podaci koje treba obraditi (u delu instrukcije koji se odnosi na *adresu operanda*). Upravljačka jedinica saopštava i aritmetičko-logičkoj jedinici šta će raditi i gde treba da uzme, odnosno pošalje informaciju. I najsloženija obrada podataka može se svesti na aritmetičke i logičke operacije u *aritmetičko-logičkoj jedinici procesora*. Pored aritmetičkih (+, -, *, /) i logičkih (I,ILI, NE, ekskluzivno ILI, $\Rightarrow$, $\Leftrightarrow$ itd.) operacija, tipične operacije koje računar izvršava pod "komandom" upravljačke jedinice su: prenos podataka – iz interne (*"keš"=cache*) memorije centralnog procesora ili iz njegovih registarau operativnu memoriju i obrnuto; učitavanje

podataka sa ulaza ili slanje podataka na izlazne uređaje; izbor prelaska na jednu ili drugu instrukciju na osnovu vrednosti nekog bita (tzv. *flega*), ili skok na neku instrukciju (kontrola toka izvršavanja programa). Dobijeni rezultat pamti se u nekom od registara procesora, najčešće u *akumulatoru*, ili u operativnoj memoriji. Adresu sledeće instrukcije upravljačka jedinica najčešće nalazi u posebnom registru procesora *programskom brojaču*. Kako se radi o digitalnim elektronskim sklopovima, to se sve osnovne operacije dešavaju sinhronizovano samo u trenucima nailaska jednog ili više radnih taktova iz *generatora takta*, koji time diktira brzinu rada procesora.

Nepregledni nizovi nula i jedinica mašinskih instrukcija sigurno nisu odgovarali programerima, koji su, uz programiranje na ovom hardverskom nivou, najpre morali da posvete pažnju logici rešavanja sve složenijih problema koji su se pred njih postavljali. Da bi sebi olakšali posao i oslobodili se programiranja na nivou hardvera, kreirali su posebne sistemske programe (*prevodioce*) koji će omogućiti prevođenje karakterističnih logičkih struktura na ustaljeni (uvek isti) način u mašinske instrukcije. Time se otvarala mogućnost korišćenja **viših programskih jezika**, pomoću kojih su programeri tada mogli da se posvete implementiranju logike rešavanja problema. Da bi na univerzalan način mogli iskazati logiku koja dovodi do rešenja postavljenog zadatka i učinili je nezavisnom od pojedinih viših programskih jezika koji su kreirani, počeli su koristiti *algoritme*. Algoritam predstavlja niz ponovljenih osnovnih logičkih struktura u redosledu koji predstavlja korake u rešavanju zadatka. *Vrste osnovnih logičkih struktura* su: *sekvenca* (niz instrukcija obrade podataka koje se izvršavaju onim redosledom kako su poređane), *selekcija* (grananje, izbor jedne od više sekvenci obrade zavisno od ispunjenja uslova), *ciklična struktura* (petlja, ponavljanje sekvence obrade do ispunjenja uslova). Ove logičke strukture su se isprva prikazivale grafički, što je kasnije zamenjeno tekstualnim zapisom pomoću tzv. *pseudokôda*. Princip programiranja koji se sastojao od predstavljanja nizova logičkih struktura iz algoritma u višim programskim jezicima zove se **strukturno programiranje**. U skladu sa ovim principom su i **vrste instrukcija u višim programskim jezicima: deklaracione instrukcije** (za "objavljivanje" tipa podataka karakterističnih veličina iz zadatka i samog njihovog postojanja; te veličine se u programima identifikuju simboličkim imenima, i nazivaju *promenljive*, ako se vrednosti u toku izvršenja programa menjaju, ili *konstante*, ako su vrednosti nepromenljive; na osnovu ovih instrukcija sistemski programi rezervišu prema tipu odgovarajući prostor u radnoj memoriji), **instrukcije dodeljivanja** (za izvršavanje traženih operacija i smeštanje rezultata u memoriju na mestu rezervisanom za promenljivu kojoj se dodeljuje vrednost), **instrukcije kontrole toka** (čime se implementiraju logičke strukture grananja i petlji) i **ulazno-izlazne instrukcije** (za preuzimanje podataka sa ulaznih uređaja i slanje rezultata na izlazne uređaje).

Vrste viših programskih jezika: Prvo su samo nizovi 0 i 1 iz kôdova operacija mašinskih instrukcija zamenjeni simboličkim imenima (pomoću prevodioca *asemblera*)**.** Tako je nastao

simbolički mašinski jezik. I dalje se radilo sa mašinskim instrukcijama. Kasnije nastaju "pravi" **viši programski jezici opšte namene** (COBOL, ALGOL, C, BASIC, PASCAL, FORTRAN i dr.). Odlikuju se podrškom različitim tipovima i strukturama podataka, različitim skupovima naredbi i različitim načinima zapisivanja (*sintaksama*) pojedinih logičkih struktura, dodeljivanja vrednosti i pojedinih operacija. Posebne vrste su i **jezici veštačke inteligencije**, **jezici za konkurentno programiranje** (za istovremeno izvršavanje više programskih procesa: ADA, CONCURENT PASCAL i dr.) i **jezici specijalne namene** (npr. SQL jezik za izdvajanje, dodavanje, izmenu, brisanje podataka i definisanje strukture u bazi podataka, za simulaciju rada sistema i električnih kola ...).

Usložnjavanjem i uopštavanjem problema koji su se postavljali pred programere sve više se ispoljavala nemogućnost programiranja zasnovanog na procedurama da izrazi sva međudejstva između pojedinih predmeta, osoba i pojava na koje su se odnosili postavljeni zadaci. Zato su programeri težili stvaranju okruženja koje će podsećati, simulirati realni svet i objekte iz njega, u nadi da će u takvom okruženju prirodnije dolaziti do rešenja složenih savremenih zadataka koji dolaze iz tog okruženja. To je zahtevalo uvođenje novog **objektno-orijentisanog pristupa i principa programiranja**. Analizirajući predmete, osobe, procese i pojave oko sebe, programeri su uočavali grupe objekata srodnih po istim ili sličnim osobinama (svojstvima) i obrascima ponašanja i težili da te zajedničke osobine i obrasce ponašanja programiraju. Tako su ovim uopštavanjem, kao i mogućnošću razlaganja analize na pojedine srodne grupe objekata, smanjivali složenost problema. Kao rezultat tog uopštavanja i razlaganja dobijani su celekupni programirani opisi, planovi, šabloni svojstava i ponašanja grupe srodnih objekata, koji se nazivaju **klase**. Rešavanje konkretnog zadatka sastojalo se dalje u kreiranju konkretnih objekata na osnovu tog šablona, plana, opisa (klase) i ispoljavanju programiranih zajedničkih ponašanja u međudejstvu tih objekata. Kreiranje *apstrakcije* (simboličke predstave, pre svega sa stanovišta konkretnog zadatka) njihovog međudejstva stvara se prirodno okruženje u kome se spontano rešava postavljeni zadatak. Objektno-orijentisano programiranje podrazumeva sledeće **karakteristike**: **enkapsulaciju**, osobinu da su unutrašnji procesi, tj. programirane procedure ponašanja koje utiču na svojstva kreiranih objekata, sakrivene ("ograđene") unutar klase; **nasleđivanje**, karakteristiku koja omogućava kreiranje novih (pod)klasa koje nasleđuju osobine opštijih (super)klasa, sa mogućnošću modifikacije postojećih i dodavanjem novih svojstava i ponašanja (npr. klasa *Student* nasleđuje osobine klase *Osoba* i dodaje nova svojstva: *br. indeksa, god. studija* i posebna ponašanja koja ih modifikuju); **polimorfizam** (više formi, oblika) je mogućnost postojanja ponašanja sa istim imenom koja se različito ispoljavaju (programiraju) u nasleđenim i osnovnim klasama. Objektni jezici: C++, Java...

TIPOVI I STRUKTURE PODATAKA

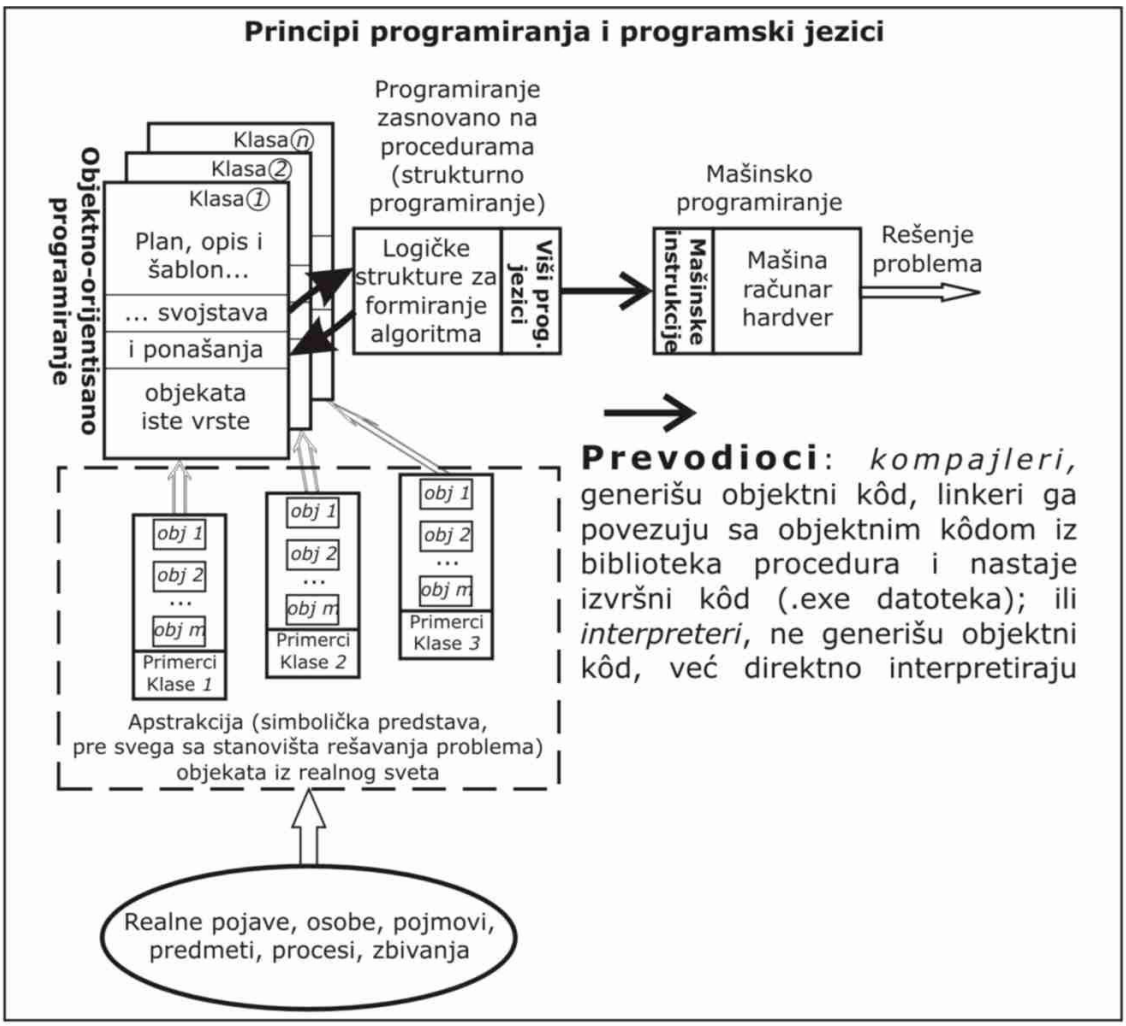
Važan aspekt u razvoju algoritma predstavlja izbor *tipova i struktura podataka* koji će biti korišćeni u razvoju programa. Od toga u značajnoj meri zavisi i izbor programskog jezika koji će biti korišćen za realizaciju samog programa, jer se mora voditi računa o podršci tipovima i strukturama podataka od strane programskih jezika. Poznavanje većeg broja programskih jezika od strane programera pruža mogućnost da se izabere jezik koji najbolje odgovara tipovima i strukturama podataka, koji su specificirani algoritmom. Svaki tip je karakterisan *skupom vrednosti (konstanti)* koje može da dobije promenljiva ili rezultat izvršavanja procedure tog tipa, i *skupom definicija operacija* koje se mogu nad njim vršiti. Broj mogućih vrednosti za podatke određenog tipa naziva se *kardinalni broj*. Neposredno ugrađeni tipovi u okviru programskog jezika zovu se *standardni tipovi*, i to su: celobrojni (**integer**), realni (**real**, **single**, **float**), logički - DA, NE (**boolean**), simbolički (jedan znak: **char**; niz znakova: **string**).

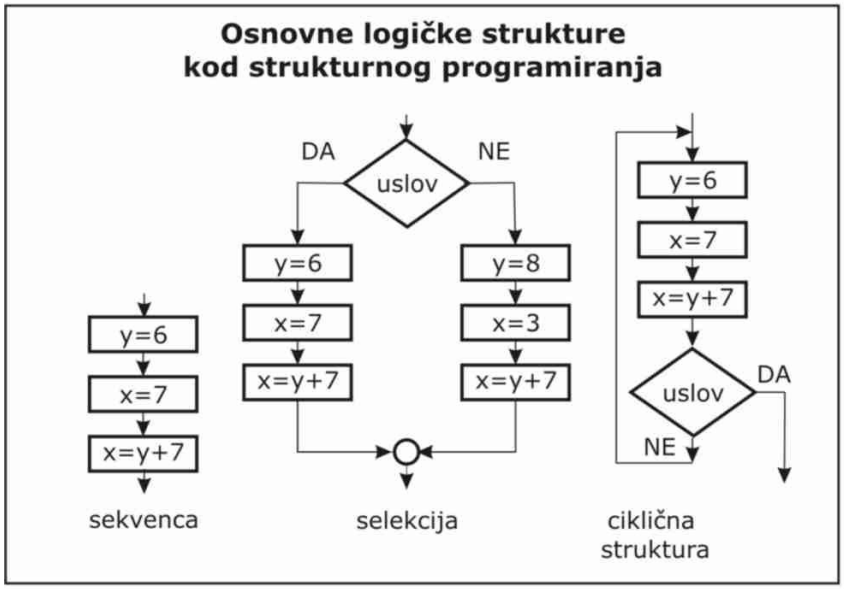
Prema strukturi: *skalarne* i *strukturisane* promenljive: *Niz* = strukturisane promenljive podataka istog tipa; *zapisi* i *skupovi*. *Statičke* (osnovne) i *dinamičke* strukture (*spiskovi*, *redovi*, *stekovi*, *stabla*, *orijentisani grafovi*).

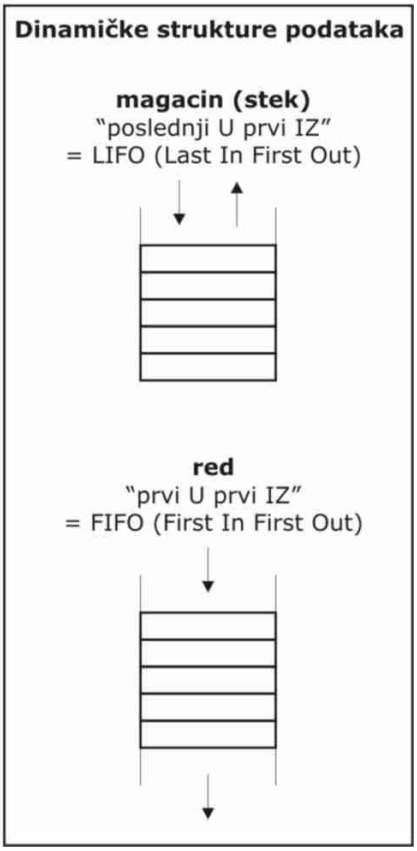
ALATI ZA PROGRAMIRANJE

Danas uglavnom <Računarsko rešavanje problema 5, 6 (često i 7)> integrisani u tzv. ***integrisana razvojna okruženja*** **(integrated development environment)** sa izgrađenom strukturom pojedinih programskih modula: moduli izvornog, programskog kôda, moduli klasa, pregledni moduli – za pregled postojećih klasa i/ili procedura (*Project browser* → svi razvojni alati i moduli integrišu se u *projektu* za rešavanje konkretnog programerskog zadatka, *Object browser*), moduli za kreiranje grafičkog korisničkog interfejsa, moduli za testiranje i otkrivanje grešaka = *Debug*, sa komandama za izvršavanje programa instrukciju po instrukciju = *Step into*, *Step out*, *Step over*, *Run to cursor*, za uvođenje kontrolnih tačaka = *Breakpoints*, uvođenje praćenja vrednosti pojedinih promenljivih = *Watch*-evi, prozori za neposredno probno izvršavanje kôda procedura = *Immediate window*).

I programski paket MS Office može se posmatrati kao sistem sa u nekom smislu razvojnim okruženjem, za programsku integraciju (objedinjeno korišćenje) svih Office programa kroz programski jezik *Visual Basic for Applications* (VBA).


Principi programiranja i programski jezici
Programiranje zasnovano na procedurama (strukturno programiranje)
Mašinsko programiranje
Objektno-orijentisano programiranje
Klasa n
Klasa 2
Klasa 1
Plan, opis i šablon...
... svojstava
i ponašanja
objekata iste vrste
Logičke strukture za formiranje algoritma
Viši prog. jezici
Mašinske instrukcije
Mašina računar hardver
Rešenje problema
obj 1
obj 2
obj m
Primerci Klase 1
Primerci Klase 2
Primerci Klase 3
Prevodioci: kompajleri, generišu objektni kôd, linkeri ga povezuju sa objektnim kôdom iz biblioteka procedura i nastaje izvršni kôd (.exe datoteka); ili interpreteri, ne generišu objektni kôd, već direktno interpretiraju
Apstrakcija (simbolička predstava, pre svega sa stanovišta rešavanja problema) objekata iz realnog sveta
Realne pojave, osobe, pojmovi, predmeti, procesi, zbivanja


Osnovne logičke strukture kod strukturnog programiranja
DA
NE
uslov
y=6
x=7
x=y+7
y=8
x=3
x=y+7
sekvenca
selekcija
ciklična struktura


Dinamičke strukture podataka
magacin (stek)
"poslednji U prvi IZ"
= LIFO (Last In First Out)
red
"prvi U prvi IZ"
= FIFO (First In First Out)