

Osnovi računarstva II

Čas 1

Miloš Daković

Elektrotehnički fakultet – Podgorica

11. februar 2019.

Predmetni nastavnici:

- Prof. dr Ljubiša Stanković
- Prof. dr Miloš Daković

Kabinet na II spratu

Saradnici:

- mr Stefan Vujović
- mr Miloš Brajović

Laboratorija za Digitalnu obradu signala

Studenti će:

- Koristiti računar u rješavanju inženjerskih problema
- Savladati algoritamski pristup rješavanju problema
- Upoznati MATLAB i Octave okruženje
- Upoznati softverska okruženja za rješavanje problema u simboličkom obliku
- Primjenjivati naučeno u toku studija i nakon završetka studija

Način ostvarenja ciljeva:

- Redovno i aktivno praćenje nastave
- Učenje i stalno proširivanje stečenih znanja
- Povezivanje naučenog gradiva sa gradivom iz ostalih disciplina

Ovaj predmet se vrednuje sa 6,5 ECTS kredita.

Nedjeljno opterećenje studenata:

8 sati i 40 minuta

- 2 časa predavanja
- 1 čas računskih vježbi
- 2 časa laboratorijskih vježbi
- 3 sata i 40 minuta samostalnog rada
 - 40 minuta pripreme za predavanja i konsultacije
 - 1 sat samostalnog rada u računarskoj sali
 - 2 sata obnavljanje pređenog gradiva, priprema kolokvijuma i ispita

Osnovna literatura:

- Z. Uskoković, Lj. Stanković, I. Đurović: MATLAB FOR WINDOWS;
- Dodatni materijali vezani za algoritme, simboličku matematiku, i grafički korisnički interfejs, će biti dostupni preko sajta ETF-a

Softver:

- MATLAB <http://www.mathworks.com>
- Octave – open source <http://octave.sourceforge.net>
<http://www.gnu.org/software/octave>
- Maple <http://www.maplesoft.com>
- wxMaxima – open source <http://wxmaxima.sourceforge.net>
<http://andrejv.github.com/wxmaxima>

Ukupno: 15 poena

- Potrebno je uspješno uraditi zadatke na laboratorijskim vježbama. Planirano je ukupno 12 laboratorijskih vježbi, svaka nosi maksimalno 1 poen.
- Domaći zadaci
- Aktivnost na predavanjima
- Aktivnost na računskim vježbama

Kolokvijum i završni ispit

Kolokvijum: 40 poena

- Kolokvijum se radi u pisanoj formi (rješavanje zadataka bez korišćenja računara).
- Termin kolokvijuma je **15. april 2019.**

Završni ispit: 45 poena

- Završni ispit se radi u računarskoj sali.
- Termin završnog ispita određuje prodekan za nastavu. Vjerovatno će to biti 27. maj ili 3. jun 2019.

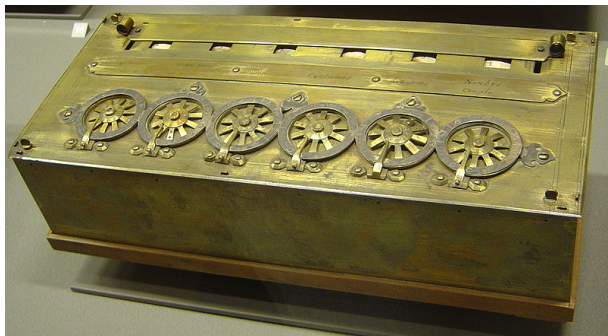
Ispit je položen sa 50 i više poena u ukupnom zbiru.

Procedure: Čovjek i obrada podataka I

- Ljudi u svakodnevnom životu uspješno koriste jezik procedure.
- Pomoću tog jezika opisuju dnevne rutine, upućuju druge ljude u nekom pravcu ili opisuju neke značajne događaje u životu.
- Jezik procedure je neformalizovan, ali obično dovoljno jasan da prenese ključne informacije.
- Čovjekova sposobnost obrade informacija je ograničena.
- Još u XIX vijeku uočeno je da ljudi uspješno obrađuju samo male količine informacija, a da su podložni raznim greškama kako se količina informacija povećava.
- Stoga se došlo na ideju kreiranja nepogrešivih računskih mašina koje bi u zahtjevnim obradama zamijenile ljude.

Procedure: Čovjek i obrada podataka II

- Nacrti prvih takvih mašina su bili plod rada Bebidža (**Charles Babbage**) i Paskala (**Blaise Pascal**). (Paskal je napravio mehaničku mašinu za sabiranje i oduzimanje)



- Revolucija u razvoju računskih mašina je omogućena pojavom poluprovodničkih elektronskih komponenti – tranzistora.

- Savremeni elektronski računari rade na principu binarne logike sa alfabetom $\{0, 1\}$.
- Teško je ljudsku logiku, zasnovanu na procedurama, pretvoriti u binarni zapis direktnim putem.
- Poseban je problem što relativno prosta procedura zapisana binarno može da ima desetine hiljada, pa i milione bita, što je čini nemogućom za održavanje i prepravljanje.
- Stoga su se razvili programski jezici kao posrednici između jezika procedure i jezika koji razumiju računari.
- Program napisan u programskom jeziku podsjeća na jednostavne direktive engleskog jezika, kombinovane sa preciznim matematičkim formulacijama.

- Prije pisanja programa u programskom jeziku treba osmisliti korake u rješavanju problema.
- Ti koraci moraju biti nedvosmisleni, jer računari mogu da izvršavaju samo nedvosmislene direktive.
- Slijed koraka koji vode ka rješenju nekog problema naziva se **algoritam**.
- Termin algoritam potiče od imena persijskog mislioca Abu Abdulah Muhameda bin Musa Al-Kwarezmija, koji je u IX vijeku osmislio postupke za obavljanje osnovnih matematičkih operacija.
- Kada se programer suoči sa nekim problemom, treba da razvije postupak za njegovo rješavanje – algoritam, a zatim da taj algoritam pretvori u kôd (tj. tekst) nekog programa.

Program – algoritam i podaci

- Program je niz instrukcija koje računar može izvršiti i čiji je rezultat rješenje nekog konkretnog problema (zadatka).
- Programski jezik je skup svih dozvoljenih instrukcija i pravila njihovog kombinovanja.
- Po terminologiji koju je uveo Niklaus Wirth, program se sastoji od dvije cjeline:
 - algoritma i
 - podataka.

- U računarskoj terminologiji poznati su: **elementarni** (osnovni) i **složeni** (izvedeni) tipovi podataka.
- Elementarni tipovi podataka su:
 - cijeli broj
 - realni (decimalni) broj
 - karakter (slovni podatak)
- Tri elementarna tipa podataka su realizovana i tumače se hardverski.
- Konkretno ovo znači da se dekadna vrijednost cijelog broja tumači na osnovu njegovog binarnog memorijskog zapisa (npr. $00101011_{(2)} = 43_{(10)}$), a da se negativni cijeli brojevi prikazuju preko dvojnog komplementa (npr. $10101011_{(2)} = -85_{(10)}$). Tumačenje karaktera se obavlja preko ASCII (ili neke druge) tabele.

- Podaci određenog tipa zauzimaju tačno definisanu memoriju. Na primjer karakteri zauzimaju 1 bajt, cijeli brojevi se često zapisuju sa 32 bita, odnosno zauzimaju 4 bajta, realni brojevi se često zapisuju u pokretnom zarezu sa ukupno 64 bita (8 bajtova).
- Podaci imaju domen. Domen predstavlja opseg vrijednosti koje može uzeti promjenljiva određenog tipa. Na primjer ako cjelobrojna promjenljiva zauzima 1 bajt ona ne može imati više od $256 = 2^8$ različitih vrijednosti.
- Kod svakog tipa podataka imamo i operacije koje se nad tim tipom sprovode (npr. sabiranje, oduzimanje, upoređivanje itd). Operacije se izvode uglavnom po matematičkim pravilima.

Imenovanje podataka

- Imena podataka u svim programskim jezicima (što ćemo mi usvojiti za naše algoritme) moraju se sastojati od slova (uključujući i znak underscore ili podvlaka `_`) i cifara, s tim da ime ne smije počinjati cifrom.
- Pojedini programski jezici razlikuju mala i velika slova prilikom imenovanja promjenljivih (za njih kažemo da su „case sensitive“), dok drugi ovu razliku ne poznaju.
- Svakom imenovanom podatku pridružuje se dio radne memorije u kojoj se skladišti vrijednost podatka.

Dodjela vrijednosti

- Podatku A dodijelimo vrijednost 5

$$A = 5$$

- Podatku B dodijelimo vrijednost ' m '

$$B = 'm'$$

- Vrijednost podatka C odredimo kao $2A + 7$

$$C = 2 * A + 7$$

- Sve ove operacije vrše dodjelu vrijednosti imenovanom podatku. Operaciju dodjele vrijednosti obilježavaćemo sa $=$ vodeći računa da to nije matematička jednakost. Na lijevoj strani mora biti imenovani podatak a na desnoj strani izraz, koji kada se „izračuna“ postaje vrijednost imenovanog podatka.

Dodjela vrijednosti – nastavak

- Nije dozvoljeno (iako je matematički korektno):

$$3 + A = B$$

jer na lijevoj strani operatora $=$ nije imenovani podatak.

- Sa druge strane, dozvoljeno je (iako matematički nije baš smisleno):

$$A = A + B$$

A i B se sabere i rezultat smjesti u promjenljivu A .

- U nekim programskim jezicima se koristi $:=$ kao operator dodjele vrijednosti. ($A := 3$)
- U algoritmima se dodjela vrijednosti nekad obilježava sa strelicom ($A \leftarrow 3$)

Aritmetičke operacije

- Sabiranje, množenje, oduzimanje, dijeljenje ...
- Operacije se obavljaju u ALU računara koja ima ograničenu dužinu registara.
- U programiranju se ne može podrazumijevati da je sabiranje asocijativna operacija $(a + b) + c = a + (b + c)$ (ovo će biti demonstrirano kasnije)
- Iz matematike znamo da je $x + 1$ uvijek veće od x ako je x prirodan broj. Razmislite šta će se desiti ako je prirodni broj u registru računara zapisan kao niz jedinica $111...111$ i dodamo mu 1 .

Operacije poređenja

- U našim algoritmima koristićemo sljedeće (binarne) operacije poređenja:
 - $>$ veće od — prvi operand je veći od drugog;
 - $<$ manje od — prvi operand je manji od drugog;
 - \geq veće ili jednako — prvi operand je veći ili jednak od drugog;
 - \leq manje ili jednako — prvi operand je manji ili jednak od drugog;
 - \equiv jednakost — prvi operand je jednak drugom;
 - \neq nejednakost — prvi operand nije jednak drugom.
- U programskim jezicima se koriste operatori: $>=$, $<=$, $==$, $<>$
- Važno je da operator poređenja na jednakost \equiv razlikujemo od operatora pridruživanja (dodjele vrijednosti) $=$.

Logičke operacije

- Programiranje poznaje logičke operacije, tj. operacije Bulove algebre koje ste naučili u prethodnom semestru.
- Operacija negacije, u oznaci $\neg A$ daje vrijednost tačno ako je A netačno i obrnuto.
- Tabele istinitosti operacija i, ili i ekskluzivno ili, su prikazane ispod:

I (AND)

A	B	$A \wedge B$
⊥	⊥	⊥
⊥	⊤	⊥
⊤	⊥	⊥
⊤	⊤	⊤

ILI (OR)

A	B	$A \vee B$
⊥	⊥	⊥
⊥	⊤	⊤
⊤	⊥	⊤
⊤	⊤	⊤

EX-ILI (XOR)

A	B	$A \oplus B$
⊥	⊥	⊥
⊥	⊤	⊤
⊤	⊥	⊤
⊤	⊤	⊥

Prioritet operacija

- Prioritet operacija u programiranju je isti kao u matematici: množenje i dijeljenje imaju veći prioritet od sabiranja i oduzimanja. U izrazu: $A + B * C$ će se prvo obaviti množenje, pa tek onda sabiranje.
- Prioritet se može promijeniti upotrebom malih zagrada: $(A + B) * C$ gdje se prvo izračuna izraz unutar zagrada, pa se tek onda pomnoži sa brojem C .
- Operacije poredjenja se uvijek obavljaju prije logičkih operacija: $x > 2 \wedge x \leq 4$. Za razliku od prethodnog primjera, ovdje zagrade nijesu potrebne.
- Zagrade je poželjno stavljati i tamo gdje se mogu izostaviti, ukoliko izrazi postaju jasniji.
- Ako postoji bilo kakva dilema o prioritetu operacija treba postaviti zagrade bez „ustručavanja“.

Karakter tip podatka

- U memoriji računara karakteri se, kao i svi drugi podaci, čuvaju u obliku niza bitova.
- Numerički ekvivalent zapisanog karaktera se naziva kôdom toga karaktera
- Na osnovu saznanja da je na određenoj memorijskoj lokaciji upisan karakter i na osnovu sadržaja te memorijske lokacije vrši se tumačenje koji je karakter u pitanju.
- Da bi razlikovali konstante tipa karakter od imena promjenljivih, operatora i brojeva u našim algoritmima ćemo ove konstante navoditi unutar apostrofa: 'A', '+', '1', '*'.
- Sa karakter podacima su dozvoljene operacije poređenja, pri čemu se u većini slučajeva poređenje vrši leksikografski vodeći računa da su cifre „ispred“ slova, i da su velika slova „ispred“ malih.

- Programski jezici često rade sa većom količinom podataka istog tipa. Ti podaci se, po potrebi, mogu smjestiti u niz.
- Elementi niza dužine N se mogu obilježiti sa:

$$a(1), a(2), \dots, a(N)$$

$$a[1], a[2], \dots, a[N] \text{ ili}$$

$$a_1, a_2, \dots, a_N$$

- Napomenimo da različiti programski jezici usvajaju drugačije notacije za indeksiranje nizova.
- Kod matrica dimenzija $M \times N$ elementi su indeksirani kao:

$$a(1, 1), a(1, 2), \dots, a(1, N)$$

$$a(2, 1), a(2, 2), \dots, a(2, N)$$

...

$$a(M, 1), a(M, 2), \dots, a(M, N)$$

- Sa elementima niza su dozvoljene sve operacije koje su dozvoljene u radu sa elementarnim podacima tipa kojem pripadaju elementi niza:

$$b(1) = b(2) - b(3)$$

$$a(2, 3) = a(1, 2) - b(1)$$

$$b(1) > 2$$

- Niz karaktera se naziva string.
- Sa članovima niza karaktera mogu da se vrše sve operacije koje se mogu vršiti sa podacima tipa karakter.
- Jedna bitna razlika u odnosu na nizove cijelih i realnih brojeva je ta da se podaci koji čine niz brojeva učitavaju sa tastature računara jedan po jedan, i na isti način prikazuju na ekranu, dok se niz karaktera može učitati i prikazati odjednom.

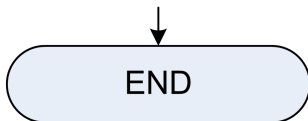
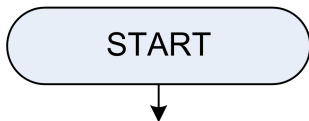
U algoritmima prepoznamo sljedeće korake:

- Početak algoritma;
- Najavu korišćenja promjenljivih (sekcija za deklaraciju);
- Ulaz (unos) podataka;
- Sekvencu (seriju pojedinačnih naredbi, jedna za drugom);
- Selekciju (dio naredbi koje se izvršavaju u zavisnosti od ispunjenja nekog logičkog uslova);
- Ciklus (dio naredbi koji se ponavlja više puta);
- Izlaz (ispis, štampanje) podataka;
- Kraj algoritma.

Predstavljanje algoritama

- Najjednostavniji način za predstavljanje algoritma bi bio tekstualni zapis. Ovakav način u većini slučajeva nije prihvatljiv jer većina govornih jezika nije jednoznačna (istu stvar možemo kazati na više načina a jedna tvrdnja može biti različito tumačena)
- Odomaćen način je grafički, preko algoritamske šeme, koja koristi ljudske vizuelne sposobnosti. Ovdje se algoritamski koraci predstavljaju jednostavnim grafičkim simbolima.
- Pored ovoga, postoji mnogo drugih načina, ali ćemo mi posmatrati još samo pseudokod.
- Pseudokod je sličan tekstu u nekom govornom jeziku (obično engleskom, ali može i našem), a ujedno je i sličan programskim jezicima, mada ne posjeduje komplikovana pravila koja mogu postojati (često i smetati) u programskim i govornim jezicima.

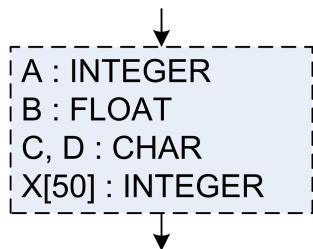
Početak i kraj algoritma



- Početak i kraj algoritma se u algoritamskoj shemi predstavljaju elipsama ili zaobljenim pravougaonicima sa tekstom START, odnosno END.
- Od START-a počinje izvršavanje programa. Tok izvršavanja programa ilustruju orjentisane linije, koje povezuju pojedine djelove algoritma. Tok izvršavanja je po pravilu odozgo na dolje.
- U našem pseudokodu početak i kraj algoritma će biti naglašeni riječima START i END u prvom i posljednjem redu pseudokoda.

Deklaracija promjenljivih

- Deklaracija (najava korišćenja) će se u našoj šemi obilježavati pravougaonikom sa isprekidanim ivičnim linijama. Unutar pravougaonika navodimo imena promjenljivih i odgovarajući tip.

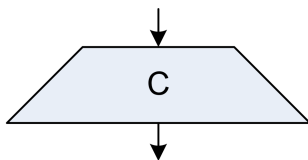
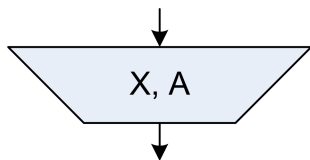


Deklarisano je da će A biti cijeli broj, B realni broj, C i D karakteri (dozvoljeno je odjednom deklarirati više promjenljivih istog tipa) i da ćemo koristiti niz X sa najviše 50 članova (svaki član je cijeli broj).

- U pseudokodu sekcija za deklaraciju se prikazuje na isti način, ali bez strelica i isprekidanih linija.
- **Sekciju za deklaraciju na početku ostavite praznu i dopunjavate je kada god uvedete novu promjenljivu u ostatku algoritma.**

Ulaz i izlaz podataka

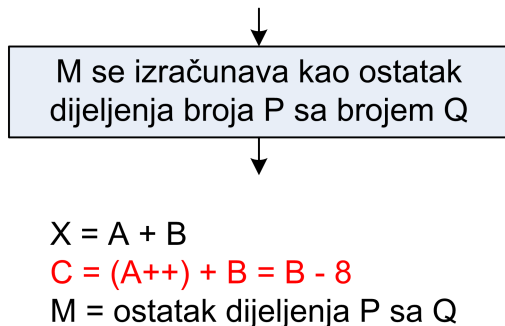
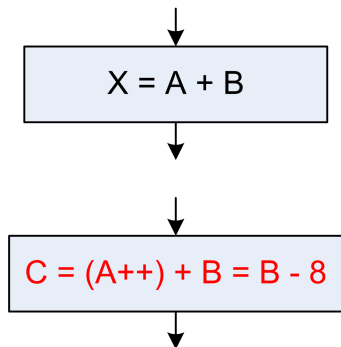
- Grafički simboli za ulaz i izlaz podataka su trapezi (a neki autori koriste i paralelograme). Kod ulaza, je veća gornja osnovica, a kod izlaza donja.
- U našem pseudokodu, naredba za ulaz je INPUT, dok je naredba za izlaz OUTPUT.
- I jedna i druga naredba su praćene sa listom promjenljivih koje korisnik unosi ili koje se prikazuju korisniku.



INPUT X, A
OUTPUT C

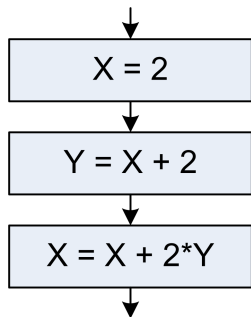
Obrada podataka

- Grafički obradu podataka predstavljamo pravougaonikom.
- Unutar pravougaonika upisujemo o kojoj se operaciji radi. Ovaj opis mora biti nedvosmislen.



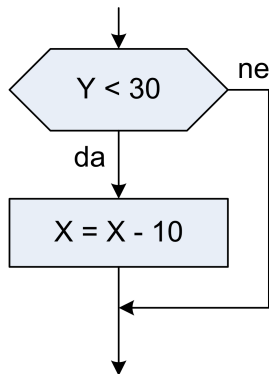
Sekvenca naredbi

- Sekvenca predstavlja niz naredbi koje se izvršavaju redom.
- Sekvenca se u algoritmu označava kao niz algoritamskih koraka povezanih strelicama u pravcu izvršavanja programa (ako nema strelica podrazumjeva se odozgo na dolje), dok se naredbe u pseudokodu upisuju jedna za drugom.
- U primjeru su navedeni koraci obrade podataka, ali jasno je da bilo koji algoritamski korak može biti dio sekvence.



$X = 2$
 $Y = X + 2$
 $X = X + 2*Y$

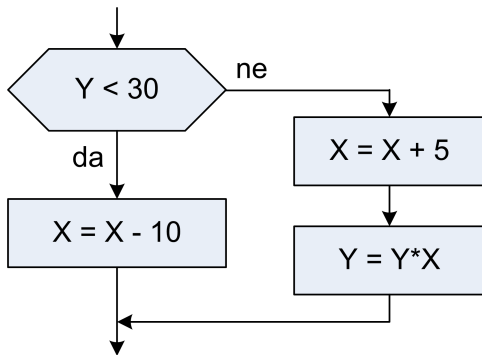
- Kod selekcije se naredbe izvršavaju ako je zadovoljen neki logički uslov.
- Uslov kod selekcije se upisuje unutar romba ili šestougona.
- Primjer: ako je Y manje od 30 tada umanj X za 10
- Uočite da je u pseudokodu neophodan ENDIF



```
IF Y < 30  
    X = X - 10  
ENDIF
```


Selekcija – nastavak

- Druga varijanta selekcije je oblika: „Ako je zadovoljen neki uslov uradi jednu akciju, a ako nije uradi drugu.“
- Uočite ključnu riječ ELSE u pseudokodu.



```
IF Y < 30  
  X = X - 10  
ELSE  
  X = X + 5  
  Y = Y * X  
ENDIF
```

Primjer – kvadratna jednačina

Posmatrajmo realna rješenja kvadratne jednačine

$$Ax^2 + Bx + C = 0$$

gdje su A , B i C poznati koeficijenti.

Rješenja jednačine zavise od njene diskriminante $D = B^2 - 4AC$.

- Ako je diskriminanta pozitivna ($D > 0$) i $A \neq 0$ tada jednačina ima dva rješenja:

$$x_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \qquad x_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

Primjer – kvadratna jednačina

- Ako je diskriminanta jednaka nuli i $A \neq 0$, onda postoji jedno rješenje: $x = -B/(2A)$
- Ako je diskriminanta manja od nule, a $A \neq 0$, jednačina nema rješenja u skupu realnih brojeva.
- Ako je $A \equiv 0$ i $B \neq 0$ postoji jedno rješenje $x = -C/B$.
- Ako je $A \equiv 0$, $B \equiv 0$ i $C \neq 0$ nema rješenja.
- Ako je $A \equiv 0$, $B \equiv 0$ i $C \equiv 0$ rješenje je trivijalno, odnosno, svako x je moguće rješenje.

O svim ovim slučajevima moramo voditi računa prilikom sastavljanja algoritma za rješavanje kvadratne jednačine jer računar neće „pretpostavljati“ ništa, niti će se sam snaći da riješi nestandardne slučajeve kvadratne jednačine.

Laboratorijske vježbe

- Algoritme ćemo raditi na predavanjima i računskim vježbama.
- Dok se bavimo algoritmima na laboratorijskim vježbama će se raditi zadaci sa standardnim softerima: LaTeX, LibreOffice, MS Word i MS Excel.
- Cilj ovog dijela vježbi je da se upoznate sa raznim programskim orkuženjima i steknete vještine potrebne za samostalnu izradu seminarskog ili specijalističkog rada.