Osnovi računarstva II

Uvodne napomene

Tipovi podataka i operacije

Osnovni algoritamski koraci

OSNOVI RAČUNARSTVA II

- Predmetni nastavnik:
  - Prof. dr Vesna Popović-Bugarin – kabinet 322
    - Konsultacije ponedjeljkom od 11:00-13:00h
    - Za kraće konsultacije koristiti e-mail: pvesna@ucg.ac.me
- Saradnici
  - mr Miloš Brajović Računske vježbe + Lab.
  - mr Stefan Vujović Računske vježbe + Lab

CILJEVI PREDMETA

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

STRUKTURA KURSA

I nedjelja	Uvod, Razvoj programskih jezika i algoritmike; Algoritamski koraci i njihovo predstavljanje
II nedjelja	Osnovni algoritmi
III nedjelja	Složenost (vremenska i prostorna) algoritama
IV nedjelja	Uvod u matematičke i inžinjerske programske alate; Predstavljanje podataka
V nedjelja	Elementarne operacije sa matricama i poljima brojeva; 2D grafika,
VI nedjelja	3D grafika, Određivanje osnovnih statističkih veličina

STRUKTURA KURSA

VII nedjelja	Naredbe za kontrolu toka programa, skript fajlovi
VIII nedjelja	***I kolokvijum (01. april 2019.)***
IX nedjelja	Funkcijski fajlovi
X nedjelja	Rad sa polinomima, Interpolacija podataka
XI nedjelja	Rješavanje problema u simboličkom obliku; Alati za simbolička izračunavanja
XII nedjelja	*Vjerski praznik*
XIII nedjelja	Osnove grafičkog korisničkog intefejsa

STRUKTURA KURSA

XIV nedjelja	*Popravni kolokvijum* ***(13. maj 2019.)*** Napredne Octave/MATLAB tehnike; Rekapitulacija gradiva
	Završni ispit (po posebnom rasporedu)

Opterećenje studenata

- **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:
  - Prezentacije sa predavanja;
  - Z. Uskoković, LJ. Stanković, I. Đurović: MATLAB FOR WINDOWS;
  - dodatni materijali vezani za simboličku matematiku, algoritme i grafički korisnički interfejs, koji će biti dostupni studentima preko sajta ETF-a

Provjera znanja

- **Načini provjeravanja znanja**:
  - Laboratorijske vježbe 10 poena
  - Kolokvijum: 40 poena
    - Kolokvijum se radi u pismenoj formi – 90 minuta.
  - Završni ispit: 50 poena
    - Završni ispit se radi u računarskoj sali.
  - Ispit je položen sa 50 i više poena u ukupnom zbiru.

SOFTVER

- Octave – open source (octave.sourceforge.net)
- MATLAB (www.mathworks.com)
- wxMaxima – open source (andrejv.github.com/wxmaxima)
- Maple (www.maplesoft.com)

Procedure: Čovjek i obrada podataka

- 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.
- Pokazuje se, međutim, da je čovjekova sposobnost obrade informacija ograničena.

Čovjek i obrada podataka

- Još u XIX vijeku, pa i ranije, 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.
- 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.

Pascal-ova mehanička mašina za sabiranje i oduzimanje

Jezik računara

- Savremeni elektronski računari rade na principu binarne logike sa alfabetom {0,1}.
- Očigledno je veoma teško 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.

Programski jezici

- Detalje istorijata razvoja programskih jezika studenti će učiti u kursu Osnovi programiranja I.
- 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, za razliku od jezika procedure, moraju biti nedvosmisleni, jer računari mogu da izvršavaju samo nedvosmislene direktive.
- **Slijed koraka koji vode ka rješenju nekog problema naziva se algoritam.**

Algoritmi

- Pojam 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.
- Dakle, programer nije samo prevodilac sa jezika procedure na programski jezik, već i osoba koja osmišljava samu proceduru.

Program

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

Podaci

- U računarskoj terminologiji poznati su:
  - elementarni (osnovni) i
  - složeni (izvedeni) tipovi podataka.
- Elementarni tipovi podataka su:
  - cijeli broj
  - realni 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 tabele.

Podaci – nastavak

- 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 dozvoljene **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.
- Već tokom bavljenja algoritmima vodićemo računa o razlici u tipu slova korišćenih za imenovanje promjenljivih.
- 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 se ne može izvršiti pridruživanje izrazu 3+A. (A i B su imenovani podaci i odnose se na konkretne memorijske lokacije, a izrazu 3+A nema smisla pridruživati memorijsku lokaciju). Sa lijeve strane operatora jednako se očekuje imenovani podatak.
- Sa druge strane, dozvoljeno je (iako matematički nije baš smisleno):

  A=A+B

  A i B se saberu 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← 3).

Aritmetičke operacije

- Operacije se obavljaju u ALU računara koja ima ograničenu dužinu registara.
- U programiranju se ne može podrazumijevati da je sabiranje asocijativna operacija $(a+b)+c = a+(b+c)$
- Iz matematike znamo da je $x+1$ uvijek veće od x ako je x cijeli broj. Razmislite šta će se desiti ako je cijeli broj u registru računara zapisan kao niz jedinica 111...111. i dodamo mu 1 i u skladu sa tim tumačite tvrdnu od upitnosti asocijativnosti u programiranju.

Operacije poređenja

- U našim algoritmima koristićemo sljedeće operacije poređenja:
  - $>$ (veće od, koje je ispunjeno ako je prvi operand veći od drugog);
  - $<$ (manje od, koje je ispunjeno ako je prvi operand manji od drugog);
  - $\geq$ (veće ili jednako, koje je ispunjeno ako je prvi operand veći ili jednak drugom);
  - $\leq$ (manje ili jednako, koje je ispunjeno ako je prvi operand manji ili jednak drugom);
  - $\equiv$ (jednakost, koja je ispunjena ako je prvi operand jednak drugom);
  - $\neq$ (nejednakost, koja je ispunjena ako prvi operand nije jednak drugom).
- U programskim jezicima se uz prva dva koriste operatori >=, <=, ==, <> (~= ili !=)
- Koristićemo $\equiv$ da bi jednakost razlikovali od operatora pridruživanja =.

Logičke operacije

- Programiranje poznaje logičke operacije, tj. operacije Bulove algebre.
- Ove operacije, sa tabelama istinitosti, su prikazane ispod:

operacija "**I**" (AND)

A	B	$A \wedge B$
$\bot$	$\bot$	$\bot$
$\bot$	T	$\bot$
T	$\bot$	$\bot$
T	T	T

operacija "**ILI**" (OR)

A	B	$A \vee B$
$\bot$	$\bot$	$\bot$
$\bot$	T	T
T	$\bot$	T
T	T	T

operacija "**EX-ILI**" (XOR)

A	B	$A \oplus B$
$\bot$	$\bot$	$\bot$
$\bot$	T	T
T	$\bot$	T
T	T	$\bot$

Logičke operacije

operacija "**NE**"

A	⌉A
⊥	T
T	⊥

U programskim jezicima se koriste drugačije oznake, ali ćemo mi tokom rada sa algoritmima koristi matematičke.

Oznake za tačno i netačno takođe nijesu iste u programskim jezicima kao matematičke, ali ćemo po pravilu koristiti matematičke oznake.

Prioritet operacija

- Prioritet operacija u programiranju je isti kao u matematici: množenje i dijeljenje imaju veći prioritet od sabiranja i oduzimanja. To praktično znači da će se u izrazu:

  A+B*C

  prvo obaviti množenje, pa tek onda sabiranje.
- Prioritet se može promijeniti upotrebom malih zagrada:

  (A+B)*C

  gdje se prvo izvrši izraz unutar zagrada, pa tek onda množenje dobijenog rezultata sa brojem C.

Prioritet operacija

- Operacije poređenja 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 kao tip podatka

- U memoriji računara karakteri se, kao i svi drugi podaci, prikazuju preko bitova.
- Za imenovanje promjenljivih tipa karakter važe ista pravila kao i za imenovanje drugih promjenljivih.
- Dekadni ekvivalent zapisanog karaktera se naziva kôdom toga karaktera.
- Dakle, 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.
- Kod po kome se kodiraju karakteri je ASCII kod (mada ima i drugih).

American Standard Code for Information Interchange.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Karakter kao tip podatka

- Tri pravila za kreiranje algoritama koji rade sa ASCII kodom su:
  - Mala slova engleske abecede su poređana u podniz od **a** do **z** (97 do 122 ASCII);
  - Velika slova engleske abecede su poređana u podniz od **A** do **Z** (65 do 90 ASCII);
  - Cifre su poređane u podniz od **0** do **9** (48 do 57 ASCII).
- To da su slova poređana u podniz zapravo znači da je ASCII kod karaktera A za jedan manji od ASCII koda karaktera B, a ovaj za jedan manji od ASCII koda karaktera C.

Karakter kao tip podatka

- Da bismo razlikovali konstante tipa karakter od imena promjenljivih, to ćemo u našim algoritmima (situacija je slična i u programskim jezicima) ove konstante navoditi unutar apostrofa: ‘A’, ‘+’, ‘1’, ‘*’.
- Ostaje da vidimo koje operacije su dozvoljene sa karakterima.
- Dozvoljavamo operacije poređenja. Npr. A>B, gdje su A i B promjenljive tipa karakter, vraća T ako je ASCII kod prvog karaktera veći od ASCII koda drugog karaktera.
- Npr. ako izraz $A \geq$’A’ $\wedge$ A $\leq$’Z’ vraća T to znači da se u promljenljivoj A nalazi neko veliko slovo!

Nizovi i matrice

- Programski jezici često rade sa većom količinom podataka istog tipa.
- Ti podaci se po potrebi mogu smjestiti u niz.
- Elementi niza cijelih brojeva dužine **N** se mogu obilježavati sa: **a[1], a[2], ..., a[N] ili a(1), a(2), ..., a(N).**
- Napomenimo da različiti programski jezici usvajaju drugačije notacije za indeksiranje nizova.
- Kod matrica dimenzija **MxN** elementi su indeksirani kao:
  a[1,1], a[1,2], ..., a[1,N],
  a[2,1], a[2,2], ..., a[2,N],
  ...
  a[M,1], a[M,2], ..., a[M,N].

Operacije sa elementima niza

- 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

- 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.

Algoritamski koraci

- Algoritmi posjeduju 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

- U praksi postoji mnoštvo načina da se algoritmi predstave.
- Odomaćen način je grafički, preko **algoritamske šeme**, koja koristi veliku ljudsku vizuelnu sposobnost (ljudi oko 80% informacija primaju vizuelno).
- Pored ovoga, postoji mnoštvo 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 jezicima.

Početak i kraj algoritma

- Početak i kraj algoritma se u algoritamskoj shemi predstavljaju elipsama sa tekstom START, odnosno END.

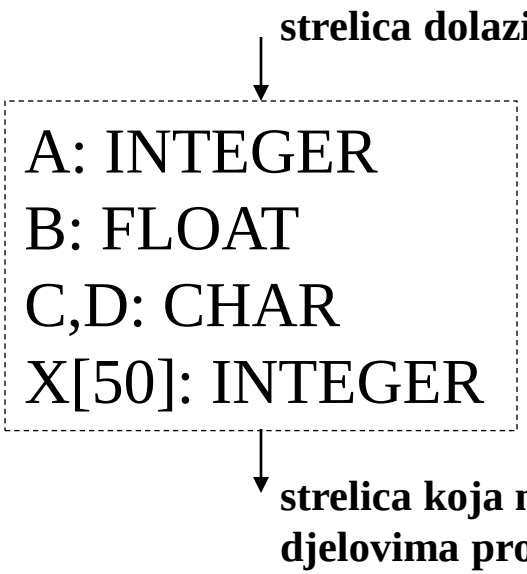
Od START-a počinje izvršavanje programa. Tok izvršavanja programa ilustruju linije (strelice), koje povezuju pojedine djelove algoritma. Tok izvršavanja je po pravilu odozgo na dolje, a ako se desi situacija da naredna naredba koja se izvršava nije ispod, to se označava linijom sa strelicom na kraju usmjerenom prema narednoj naredbi.

Početak i kraj algoritma
Deklaracija promjenljivih

- U našem pseudokodu početak i kraj algoritma će biti naglašeni riječima START i END u prvom i posljednjem redu, respektivno (u tim redovima se ne smije nalaziti ništa osim START, odnosno END).
- Naredni korak u algoritmu je najava svih promjenljivih koje se koriste u programu. Pored imena promjenljive, koje mora biti pravilno, mora se navesti i kojeg je tipa promjenljiva. Ako se najavljuju nizovi i matrice, moraju im se naglasiti dimenzije.
- Ovaj dio se naziva deklaracijom promjenljivih.
- Deklaracija se obavlja nakon START-a, a prije bilo koje druge naredbe u programu.

Deklaracija promjenljivih

- Deklaracija se u našoj shemi obavlja unutar pravougaonika sa isprekidanim ivičnim linijama.



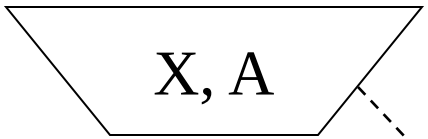
Deklarisano je da će A biti cijeli broj, B realni broj, C i D karakteri (dozvoljeno je odjednom deklarisati više promjenljivih istog tipa) i da ćemo koristiti niz X sa najviše 50 članova.

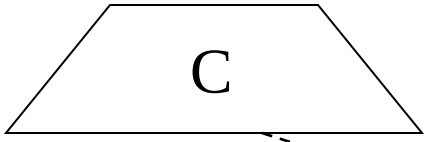
Savjet je da sekciju za deklaraciju na početku ostavite praznu i da je popunjavate kako budete koristili koju promjenljivu u ostatku algoritma.

U pseudokodu sekcija za deklaraciju se prikazuje na isti način, ali bez strelica i isprekidanih linija.

Ulaz i izlaz podataka

- Grafički simboli za ulaz i izlaz podataka su trapezi.
- Kod ulaza, trapez je sa većom gornjom stranicom, a kod izlaza sa većom donjom.





U našem pseudokodu, naredba za ulaz je INPUT, dok je naredba za izlaz OUTPUT.

I jedna i druga naredba su praćene listom promjenljivih koje korisnik unosi ili koje se prikazuju korisniku.

INPUT X, A

OUTPUT C

← pseudokod

Ulaz i izlaz podataka

- Ako imamo niz brojeva, unos i izlaz (prikaz) toga niza se mora obaviti element po element. Npr.
  INPUT X[1], X[7]
- Ako imamo niz karaktera
  CHAR: C[20]
  unos i prikaz se mogu obaviti odjednom, npr.:
  OUTPUT C
- Ako želimo da tokom rada korisnik dobije neko propratno obavještenje, to ćemo raditi navodeći to obavještenje unutar navodnika:
  OUTPUT “NEKO OBAVJESTENJE”

Obrada podataka

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

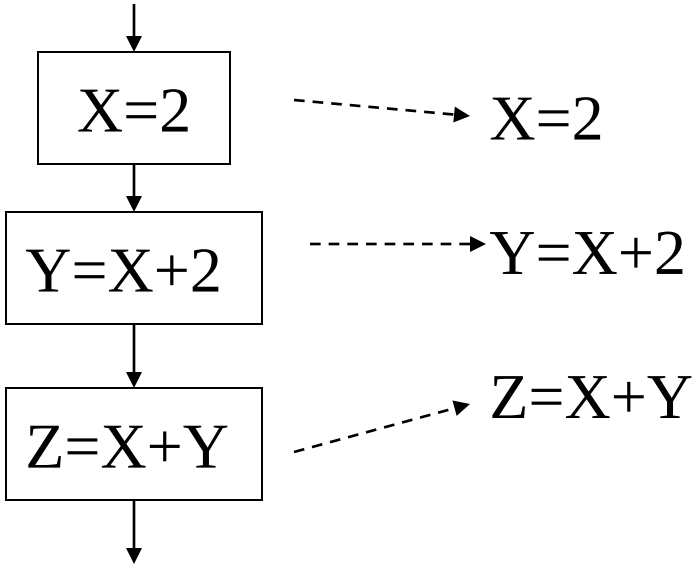
X=A+B

X se izračunava kao ostatak dijeljenja broja A sa brojem B

A+B=C+D

Sekvenca naredbi

- Sekvenca je jedan od tri osnovna algoritamska elementa i predstavlja niz naredbi koje se izvršavaju redom.

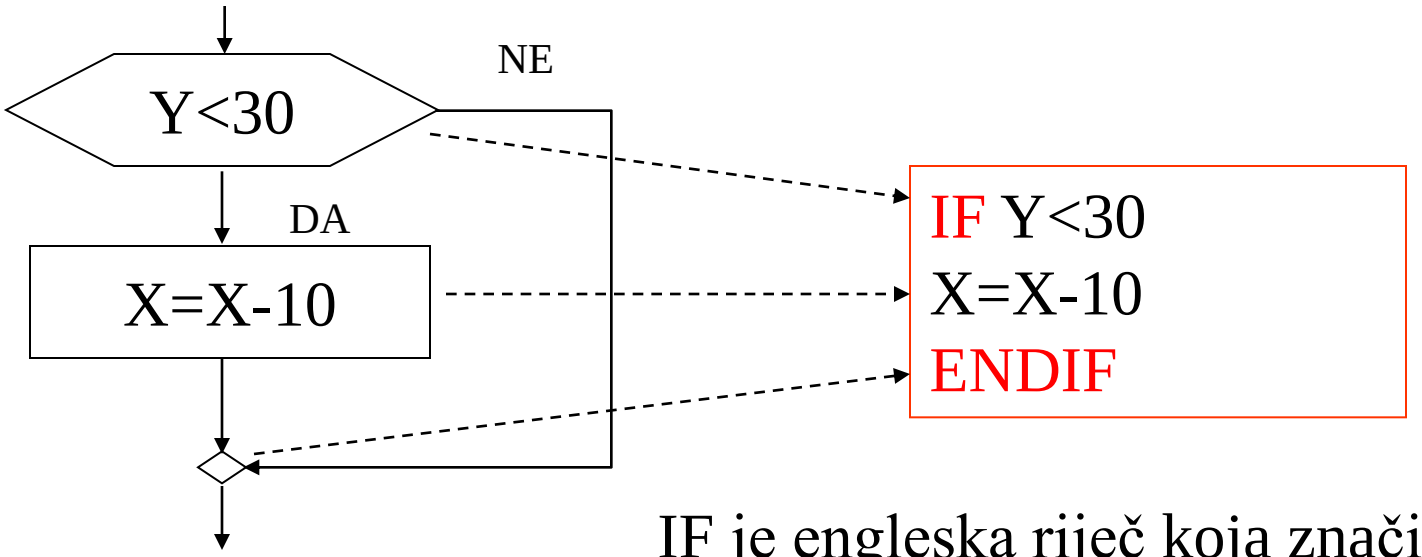


Prikaz sekvence naredbi u algoritamskoj shemi i u pseudokodu.

Sekvenca se u algoritmu označava kao niz pravougaonika povezanih strelicama u smjeru izvršavanja programa (ako nema strelica podrazumijeva se odozgo na dolje), dok se naredbe u pseudokodu upisuju jedna za drugom.

Selekcija

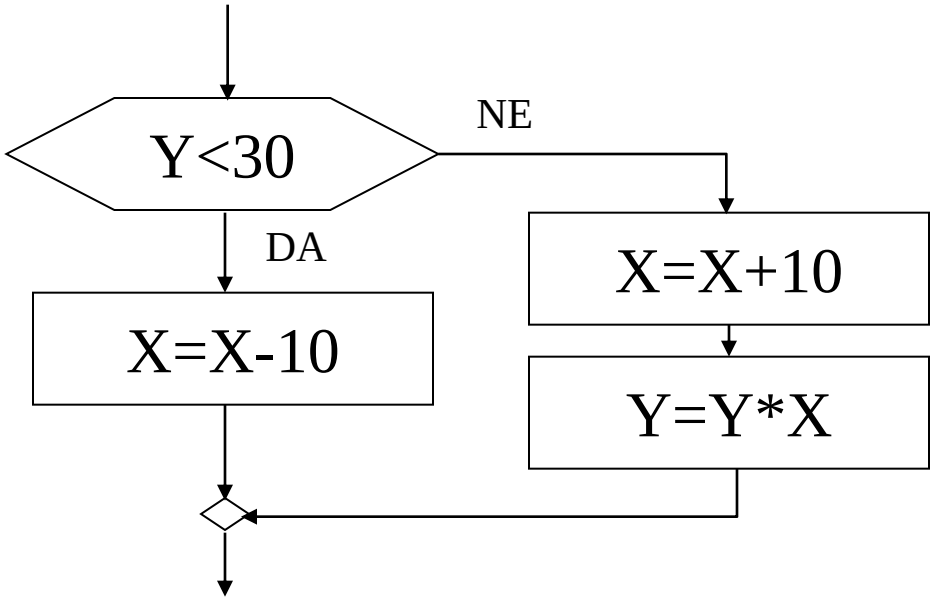
- Kod selekcije se naredbe izvršavaju ako je zadovoljen neki logički uslov.
- Uslov kod selekcije se upisuje unutar romba ili šestougaonika.
- Na primjer, **umanji X za 10 ako je Y manje od 30**, se zapisuje kao:



IF je engleska riječ koja znači AKO.

Selekcija

- Pored prikazane osnovne varijante, selekcija ima i druge varijante.
- Druga varijanta je oblika: "Ako je zadovoljen neki uslov uradi jednu akciju, a ako nije uradi drugu."

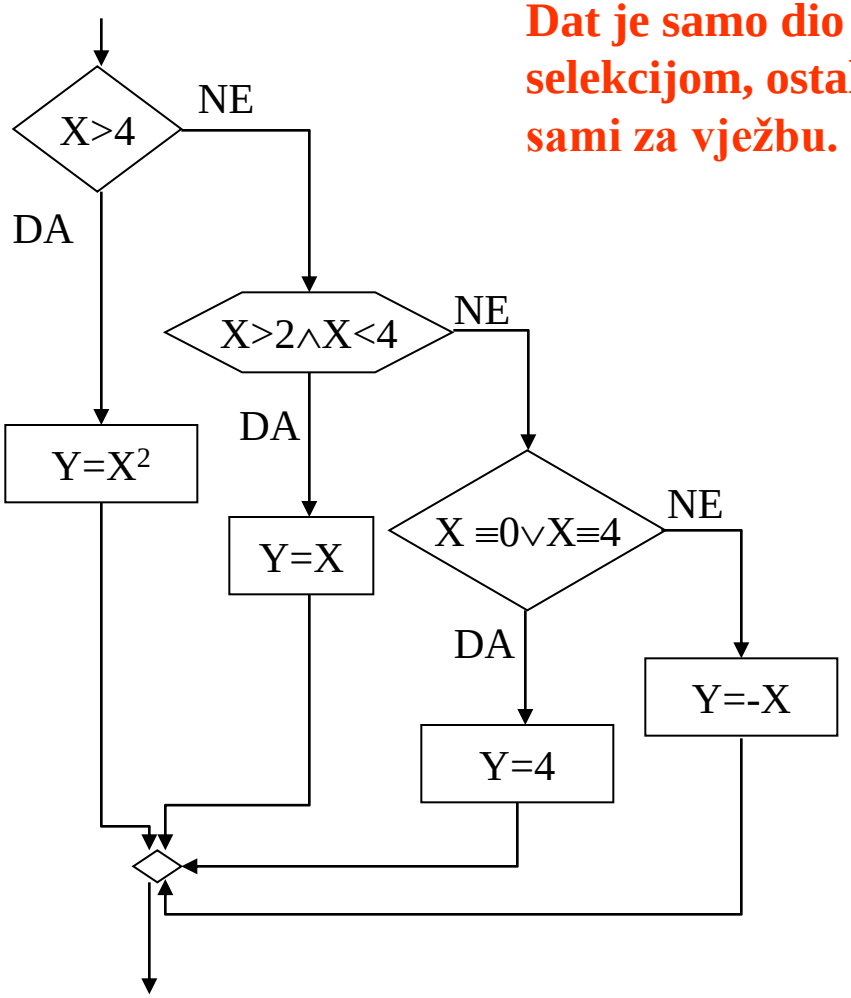



```
IF Y<30
        X=X-10
ELSE
        X=X+10
        Y=Y*X
ENDIF
```

Smisao **ELSE** je INAČE, odnosno, ako IF uslov nije zadovoljen.

Složena selekcija

$$Y = \begin{cases} X^2 & X > 4 \\ X & 2 < X < 4 \\ 4 & X = 0 \text{ ili } X = 4 \\ -X & \text{drugdje} \end{cases}$$



Dat je samo dio sa selekcijom, ostalo dodajte sami za vježbu.

Složena Selekcija

- Jedne naredbe se izvršavaju ako je prvi uslov zadovoljen; druge naredbe se izvršavaju ako prvi uslov nije zadovoljen, a drugi uslov jeste; treće naredbe se izvršavaju ako prva dva uslova nijesu zadovoljena, a treći uslov jeste. Ako nijedan od pobrojanih uslova nije zadovoljen izvršava se INAČE dio.
- Da bi ovo ilustrovali, posmatrajmo slučaj određivanja korijena kvadratne jednačine. Neka je kvadratna jednačina koja se rješava **$AX^2+BX+C=0$** i neka su zadati koeficijenti **A**, **B** i **C**.

Selekcija (nastavak)

- Selekcija može biti još složenija. Jedne naredbe se izvršavaju ako je prvi uslov zadovoljen; druge naredbe se izvršavaju ako prvi uslov nije zadovoljen, a drugi uslov jeste; treće naredbe se izvršavaju ako prva dva uslova nijesu zadovoljena, a treći uslov jeste. Ako nijedan od pobrojanih uslova nije zadovoljen izvršava se INAČE dio.
- Da bi ovo ilustrovali, posmatrajmo slučaj određivanja korijena kvadratne jednačine. Neka je kvadratna jednačina koja se rješava **$AX^2+BX+C=0$** i neka su zadati koeficijenti **A**, **B** i **C**.

Selekcija - Primjer

- Rješenja ove jednačine zavise od diskriminante **B^2-4AC**. Ako je diskriminanta veća od nule i **A≠0** kvadratna jednačina ima dva rješenja:

$$X_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \qquad X_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

- Ako je diskriminanta jednaka nuli, **B^2-4AC≡0**, i **A≠0**, onda postoji jedno rješenje:

X=-B/(2A)

- Ako je diskriminanta manja od nule, a **A≠0**, jednačina nema rješenja u skupu realnih brojeva.

Selekcija - Primjer

- Ako je **A≡0** i **B≠0** postoji jedno rješenje **X=-C/B**.
- Ako je **A≡0** i **B≡0** i **C≠0** nema rješenja.
- Ako je **A≡0** i **B≡0** i **C≡0** rješenje je trivijalno, odnosno, svako **X** je moguće rješenje.

Šta smo danas učili?

- Način organizacije predmeta
- Osnovni tipovi podataka i imenovanje podataka;
- Aritmetičke, logičke i operacije poređenja
- Osnovni pojmovi o nizovima.
- Načini predstavljanja algoritma
- Algoritamski koraci za početak i kraj rada
- Algoritamski koraci za ulaz i izlaz podataka
- Algoritamski korak obrade podataka
- Algoritamski korak selekcije (provjere uslova i donošenja odluke o nastavku izvršavanja programa)