PROGRAMSKE UPRAVLJAČKE STRUKTURE

Dragan dr Dinu

PROGRAMSKE UPRAVLJAČKE STRUKTURE

- *Programske upravljačke strukture su mehanizmi za definisanje redosleda izvođenja operatora i izraza nad datim tipovima podataka ...*

- *Rešenje bilo kog problema, koji je po svojoj prirodi rešiv pomoću računara, može se izraziti kao superpozicija sledećih struktura:*
  - ***sekvence***,
  - ***selekcije*** *i*
  - ***iteracije***

SELEKCIJA

Dragan dr Dinu

ISKAZI SELEKCIJE

- Omogućava da se određene akcije naredbe izvrše uslovno u zavisnosti od vrednosti određenog iskaza

- Iskazi selekcije (grananje):
  - **if**
  - **switch**

IF SELEKCIJA ...

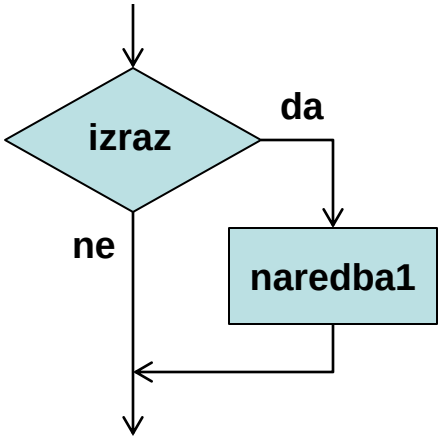
- Koju granu (od moguće dve) izvršiti?

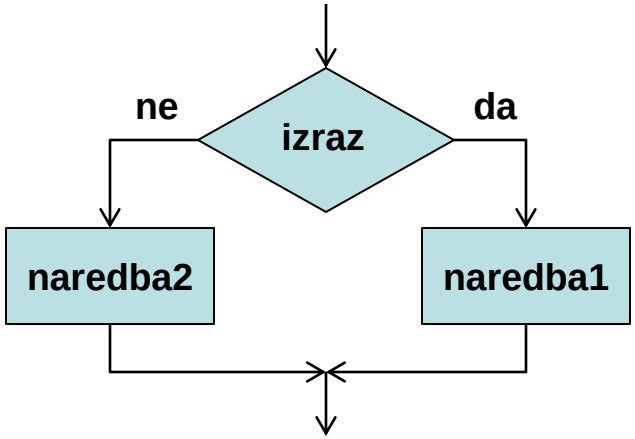
Nepotpuna if selekcija

Ako je **izraz** tačan
    naredba1 se izvršava

Potpuna if selekcija

Ako je **izraz** tačan
    naredba1 se izvršava
u suprotnom, ako je **izraz** netačan
    naredba2 se izvršava





... IF SELEKCIJA ...

- Kako to izgleda u C kodu:

Nepotpuna if selekcija

```
if (izraz)
   naredba1
```

Potpuna if selekcija

```
if (izraz)
   naredba1
else
   naredba2
```

... IF SELEKCIJA

- Ili blok naredbi:

Nepotpuna if selekcija

```
if (izraz)
{
  naredba11
  ...
  naredba1n
}
```

Potpuna if selekcija

```
if (izraz)
{
  naredba11
  ...
  naredba1n
}
else
{
  naredba21
  ...
  naredba2m
}
```

UGNJEŽDAVANJE IF SELEKCIJA …

- Može se napraviti ugnježdavanje if selekcija:

Ugnježdene if selekcije

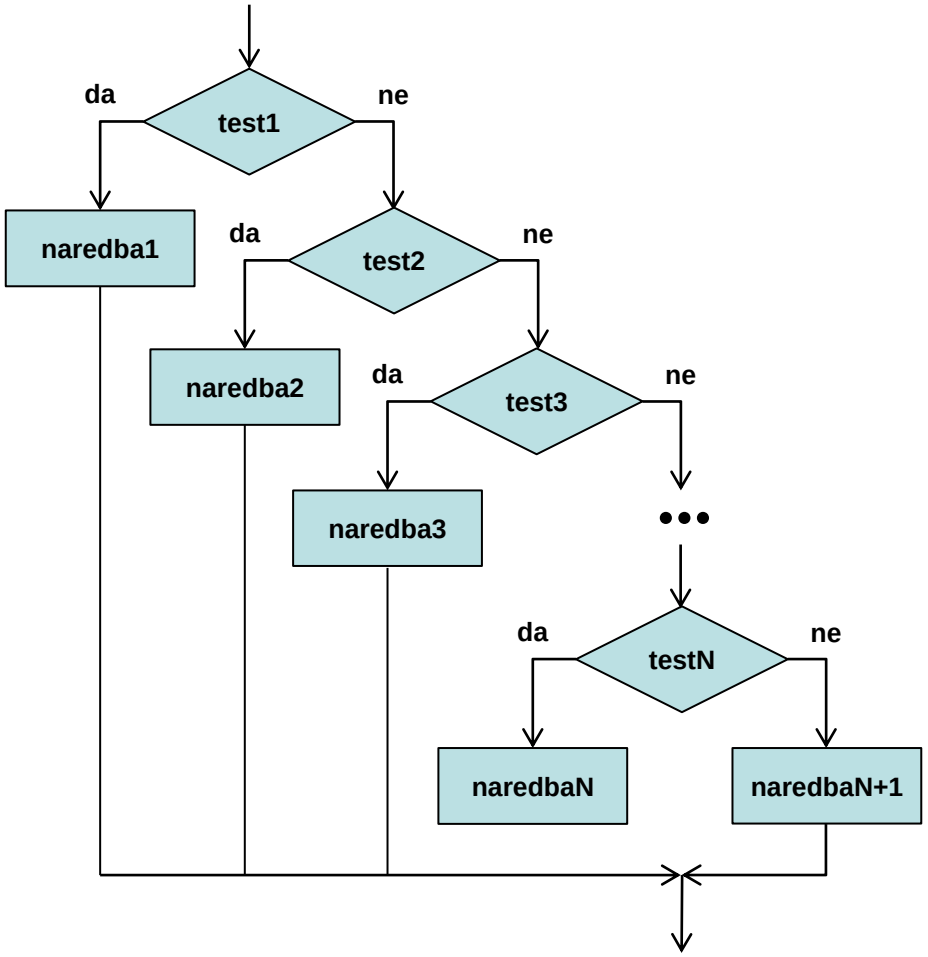
```
if (boolean_izraz1)
  naredba1
else
  if (boolean_izraz2)
     naredba2
  else
     naredba3
```

Ugnježdene if selekcije (kompaktan zapi)

```
if (boolean_izraz1)
  naredba1
else if (boolean_izraz2)
  naredba2
else
  naredba3
```

… UGNJEŽDAVANJE IF SELEKCIJA …


```
if (test1)
  naredba1
else if (test2)
  naredba2
else if (test3)
  naredba3

    ...

else if (testN)
  naredbaN
else
  naredbaN-1
```



… UGNJEŽDAVANJE IF SELEKCIJA

- Kako rešiti sledeći problem:
  Ako je **x > 3** prikaži “gore” a ako je **x < 0** prikaži “dole”

```
int x = 0;
scanf("%d", &x);
if (x >= 0)
   if (x > 3)
      printf("gore");
   else
     printf("dole");
```

```
int x = 0;
scanf("%d", &x);
if (x >= 0)
{
   if (x > 3)
      printf("gore");
}
else
   printf("dole");
```

```
int x = 0;
scanf("%d", &x);
if (x > 3)
   printf("gore");
else
   if (x < 0)
      printf("dole");
```

- **else** se uvek vezuje za poslednji **if**

IF SELEKCIJA PRIMERI

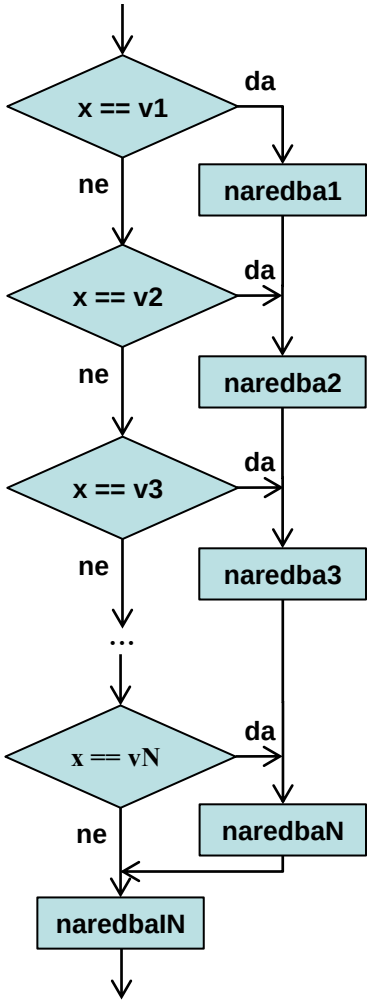
- U prethodnim primerima dodati logičku proveru unetih vrednosti (da li su možda manje od nule i šta tada raditi?)

- Zadatak 1: implementirati program koji ispisuje koji je od dva broja veći (odrediti tipove podataka, potrebnu proveru, ...)

SWITCH SELEKCIJA

- Koju granu (od više njih) izvršiti?
- Izbor se vrši na osnovu celobrojne vrednosti

```
switch (a)
{
    case v1: naredba1;
    case v2: naredba2;
    case v3: naredba3;
    ...
   case vN: naredbaN;
   default:   naredbaIN;
};
```



SWITCH SELEKCIJA PRIMER ...

- Program za ispis uspeha na osnovu unete ocene

```
int ocena;
printf("\n Program za ispis uspeha na osnovu unete ocene!\n\n");
printf("Unesite ocenu: ");
scanf("%d", &ocena);
switch (ocena)
{
   case 5: printf("\nOdlican!\n");
   case 4: printf("\nVrlo dobar!\n");
   case 3: printf("\nDobar!\n");
   case 2: printf("\nDovoljan!\n");
   case 1: printf("\nNedovoljan!\n");
}
return 0;
```

... SWITCH SELEKCIJA PRIMER ...

- Kako da se izvrši samo jedan case?

```
int ocena;

printf("\n Program za ispis uspeha na osnovu unete ocene!\n\n");
printf("Unesite ocenu: ");
scanf("%d", &ocena);

switch (ocena)
{
   case 5: printf("\nOdlican!\n");
           break;
   case 4: printf("\nVrlo dobar!\n");
           break;
   case 3: printf("\nDobar!\n");
           break;
   case 2: printf("\nDovoljan!\n");
           break;
   case 1: printf("\nNedovoljan!\n");
}

return 0;
```

... SWITCH SELEKCIJA PRIMER ...

- Šta je **BREAK**?
  - Kada se naiđe na **break**, blok naredbi se napušta bez obzira na trenutnu vrednost
  - Program nastavlja sa izvršavanjem prve naredbe iza bloka
  - Ukoliko se pozove u ugnježdenim blokovima, **break** izlazi samo iz unutrašnjeg bloka
  - Koristiti vrlo obazrivo i samo u krajnjoj nuždi

... SWITCH SELEKCIJA PRIMER

- Šta ako uneti broj nije iz opsega 1 - 5?

```
int ocena;

printf("\n Program za ispis uspeha na osnovu unete ocene!\n\n");
printf("Unesite ocenu: ");
scanf("%d", &ocena);

switch (ocena)
{
   case 5:  printf("\nOdlican!\n");
            break;
   case 4:  printf("\nVrlo dobar!\n");
            break;
   case 3:  printf("\nDobar!\n");
            break;
   case 2:  printf("\nDovoljan!\n");
            break;
   case 1:  printf("\nNedovoljan!\n");
            break;
   default: printf("\nOcena mora biti između 1 i 5.\n");
}

return 0;
```

ZADATAK IZ IF I SWITCH SELEKCIJA

- Implementirati program za računanje kvadratne jednačine:

$$a \cdot x^2 + b \cdot x + c = 0$$

- http://sh.wikipedia.org/wiki/Kvadratna_jednačina
- Promenljive:

  a, b, c, d, x1, x2, y1, y2

- Šta ako je **a == 0** ili **b == 0** ?
- Šta ako je **d == 0** ili **d < 0** ili **d > 0** ?
- U zavisnosti od vrste korena prikazati rezultat i odgovarajući tekst (**switch?**)

ITERACIJA

ISKAZI ITERACIJE ...

- Nazivaju se još i **programskim petljama** jer vrte naredbe u krug (petlju)
- **Omogućavaju ponavljanje određenih naredbi ili blokova više puta (telo petlje)**
- C programski jezik omogućava **fleksibilne načine** za određivanje broja ponavljanja petlji ili donošenje odluke o izlasku iz petlje

... ISKAZI ITERACIJE

- Koliko puta će se iteracija izvršavati određuje se na osnovu toga da li je određeni izraz (**uslov izvršavanja iteracije**) tačan ili netačan

- U zavisnosti od toga da li se uslov izvršavanja iteracije proverava na početku ili na kraju, razlikuju se iteracije sa izlaskom na vrhu, na dnu, ili u sredini

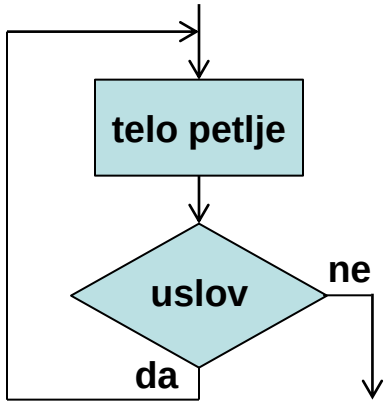
- Iskazi iteracije u C programskom jeziku:
  - **do-while** iskaz (izlazak na dnu)
  - **while** iskaz (izlazak na vrhu)
  - **for** iskaz (izlazak na vrhu)

DO WHILE ITERACIJA

- Iteracija sa izlaskom na dnu, uslov iteracije se izračunava nakon tela petlje
- Telo petlje se izvršava **NAJMANJE** jednom, čak i kada je uslov izvršavanja iteracije odmah netačan (jednak nuli)

```
do
  naredba
while (uslov);
```

```
do
{
  naredba1
  ...
  naredbaN
}
while (uslov);
```



DO WHILE ITERACIJA PRIMER

- Proširite primer za ispis uspeha na osnovu unete ocene tako da se korisniku ne dozvoli unos ni jednog drugog broja osim onog iz opsega 1 - 5

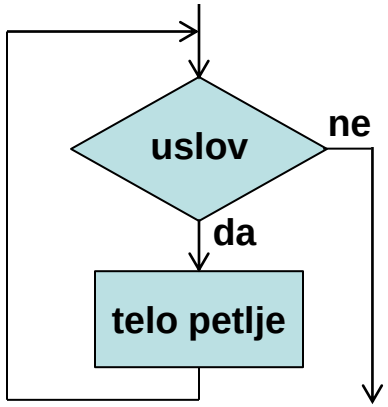
```
int ocena;

printf("\n Program za ispis uspeha na osnovu unete ocene (1 -5)!\n\n");
printf("Unesite ocenu: ");

do
 {
   scanf("%d", &ocena);
   if ((ocena < 1) || (ocena > 5))
      printf("\n\nOcena mora biti iz intervala 1-5. Unesite ponovo ocenu: ");
} while ((ocena < 1) || (ocena > 5));

switch (ocena)
{
   case 5: printf("\nOdlican!\n"); break;
   case 4: printf("\nVrlo dobar!\n"); break;
   case 3: printf("\nDobar!\n"); break;
   case 2: printf("\nDovoljan!\n"); break;
   case 1: printf("\nNedovoljan!\n"); break;
}

return 0;
```

WHILE ITERACIJA

- Iteracija sa izlaskom na vrhu, uslov se izvršava pre tela petlje
- Telo petlje se izvršava sve dok je uslov izvršavanja iteracije ispunjen (jednak jedinici), kada to nije više slučaj izlazi se iz iteracije
- Telo petlje se izvršava nula, jednom ili više puta

```
while (uslov)
  naredba
```

```
while (uslov)
{
  naredba1
  ...
  naredbaN
}
```



WHILE ITERACIJA PRIMER

- Napisati program za računanje faktorijala celog broja koji definiše korisnik

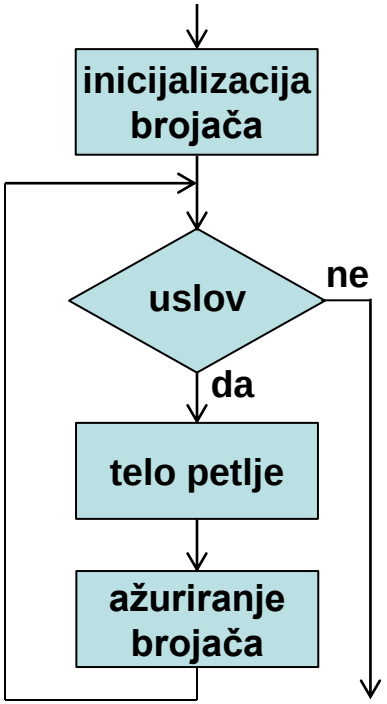
```
int broj = 1, i = 1, fakt = 1;

printf("\n Program za racunanje faktorijala unetog broja.\n\n");
printf("Unesite broj: ");

do
 {
    scanf("%d", &broj);
    if (broj < 1)
        printf("\n\nBroj mora biti veći od 0. Unesite ponovo broj: ");
} while (broj < 1);

while (i <= broj)
{
    fakt *= i;
    i++;        //uzrokuje da (i <= broj) konvergira nuli
}

printf("\n\nFaktorijal broja %d je: %d. \n", broj, fakt);

return 0;
```

FOR ITERACIJA

- Iteracija sa izlaskom na vrhu, uslov se izvršava pre tela petlje
- Telo petlje se izvršava nula, jednom ili više puta
- Slično WHILE iteraciji ali se izvršavanje kontroliše brojačkom promenljivom

```
for (inic_br; uslov; ažur_br)
  naredba
```

```
for (inic.br;uslov;ažur.br)
{
  naredba1
  ...
  naredbaN
}
```



FOR ITERACIJA PRIMER ...

- Kod za prikazivanje prvih 100 prirodnih brojeva:

```
int i;
for (i = 1; i <= 100; i++)
{
   printf("\n%d\n", i);
}
```

- Brojač je promenljiva **i**
- Brojač se inkrementira (ili dekrementira) nakon svake iteracije
- Može i ovako:

```
for (int i = 1; i <= 100; i++)
{
   printf("\n%d\n", i);
}
```

- Šta su prednosti i mane ovog pristupa?

... FOR ITERACIJA PRIMER

- Napisati kod za računanje sume prvih N prirodnih brojeva, pri čemu se N zadaje na početku programa od strane korisnika (problem rešiti pomoću **for** iteracije)

```
int N = 1;
int i, suma = 0;

printf("\n Program za racunanje sume prvih N prirodnih brojeva.\n\n");
printf("Unesite N: ");

do
{
    scanf("%d", &N);
    if (N < 1)
    printf("\n\nN mora biti veći od 0. Unesite ponovo N: ");
} while (N < 1);

for (i = 1; i <= N ; i++)
    suma += i;

printf("\n\nSuma prvih %d prirodnih brojeva je: %d. \n", N, suma);

return 0;
```

FOR I WHILE ITERACIJE

- Postoji ekvivalencija između FOR i WHILE
- U C programskom jeziku FOR se može konvertovati u WHILE i obrnuto
- U zavisnosti od prirode problema zavisi i koja je iteracija će se upotrebiti
- Ako nešto treba obaviti određen poznat broj puta koristi se FOR
- Ako se čeka da se neki uslov desi (a to nije unapred određeno brojem ponavljanja) bolje je koristiti WHILE
- Primer: odrediti sumu prvih N prirodnih brojeva, pri čemu se N zadaje na početku programa od strane korisnika i koristiti `while` iteraciju

FOR I WHILE ITERACIJE PRIMER

- Napisati kod za računanje sume prvih N prirodnih brojeva, pri čemu se N zadaje na početku programa od strane korisnika (problem rešiti pomoću **WHILE** iteracije)

```
int N = 1, i, suma = 0;

printf("\n Program za racunanje sume prvih N prirodnih brojeva.\n\n");
printf("Unesite N: ");

do
{
    scanf("%d", &N);
    if (N < 1)
    printf("\n\nN mora biti veći od 0. Unesite ponovo N: ");
} while (N < 1);

i = 1;
while(i <= N)
{
    suma += i;
    i++;
};

printf("\n\nSuma prvih %d prirodnih brojeva je: %d. \n", N, suma);

return 0;
```

UGNJEŽDAVANJE ITERACIJA

- Bilo koja upravljačka struktura može sadržati unutar sebe drugu neku upravljačku strukturu (u prethodnim primerima se unutar **`do-while`** iteracije nalazila **`if`** selekcija)

- Tako se, u zavisnosti od problema, unutar **`for`** iteracija može naći **`while`**, **`do-while`**, i sl.

- Java ne nameće nikakvo ograničenje po pitanju broja ugnježdenih upravljačkih kontrola

  – Sa praktične strane, pak, previše ugnježdenih struktura je teško ispratiti, razumeti i testirati

UGNJEŽDAVANJE FOR ITERACIJA

- Ugnježdene **for** upravljačke strukture se često koriste u za rešavanje nekoliko vrsta problema (prolazak kroz elemente matrice, sortiranje, ...)

- Kod ugnježdenih **for** upravljačkih struktura mora se voditi računa da svaka **for** iteracija mora imati svoj brojač (ne može se upotrebiti isti brojač, jer će unutrašnja iteracija pokvariti stanje brojača spoljašnjeg brojača)

```
for (i = 0; i < br_vrst; i++)     - spoljašnja iteracija
  for (j = 0; j < br_kolona; j++) - unutrašnja iteracija
```

CONTINUE I GOTO

- **continue** naredba nastavlja izvršavanja iteracije sa narednim prolazom uz preskakanje svih naredbi tela tekuće iteracije (while, for i do while)

```
for(i=0;i < N;i++)
{                                   pocetak tela for iteracije;
   naredba1; naredba2;
   if (X[i] % 2)                    i-ta vrednost nije deljiva sa 2
      continue;
   naredba3; naredba4;
}                                   na sledeci prolaz ako i < N
```

- Program nastavlja izvršavanje ulaskom u naredni ciklus
- Isti efekat može da se postigne korišćenjem **IF-ELSE** selekcije

- Naredba **goto** usmerava programski tok na navedenu labelu, nestrukturirano programiranje, **ZABRANJENO**