Programski jezici i strukture podataka

11

Spregnute strukture podataka

- Kod sekvencijalne realizacije linearne strukture podataka uzastopni elementi su susedni u memoriji.
- Kod spregnute fizičke realizacije linearne strukture podataka, elementi mogu biti bilo gde u memoriji.
- Logički linearni poredak se održava tako što elementi liste sadrže eksplicitnu adresu narednog elementa.

Spregnuta lista

- Često je pogodnije da se elementi urede na proizvoljan način pa da se povežu, nego da se fizički poređaju u sekvencu.
- Struktura koja sadrži pokazivač na objekat istog tipa kao što je ona sama naziva se **samo-upućujuća struktura** podataka.
- Samo-upućujuća struktura podataka predstavlja osnovu za spregnute liste, zovemo je ČVOR
- Upotrebom samo-upućujuće strukture realizuju se različite strukture oblika stabla kao i za spregnute realizacije reda i steka.

Spregnuta lista

- Informacioni sadržaj može biti bilo kog skalarnog ili strukturnog tipa.
- Listi se pristupa preko jednog spoljašnjeg pokazivača koji pokazuje na prvi čvor, zovemo ga **GLAVA** liste.

Spregnuta lista

- Lista je dinamička struktura koja je linearna po svakoj relaciji uspostavljenoj među njenim podacima. Zavisno od broja logičkih veza između čvorova, podataka i uspostavljenih relacija, razlikuju se sledeći tipovi lista:
  - Jednostruko spregnuta lista
  - Dvostruko spregnuta lista
  - Multilista
  - Kružna lista

Jednostruko spregnuta lista (spregnuta realizacija linearne liste)

- Jednostruko spregnuta lista organizuje podatke po jednoj relaciji gde pokazivački deo čvora liste ima informaciju samo o sledećem čvoru po toj relaciji.
- Kao primer mogu poslužiti podaci o studentima (prezime, ime i broj indeksa) uređeni po rastućoj vrednosti broja indeksa. Ima mnogo primena.

Definicija

- Jednostruko spregnuta lista se definiše kao uređeni par:

$$P=(S(P),r(P))$$

sa sledećim osobinama:

- struktura je linearna
- dozvoljen je pristup svakom elementu
- moguće je ukloniti bilo koji element
- dozvoljeno je dodati element na bilo kojoj poziciji

Jednostruko spregnuta lista

- Jednostruko spregnuta lista je veoma fleksibilna struktura.
- Pristup svakom elementu je dozvoljen (prema poziciji, prema informacionom sadržaju, navigacija).
- Element se može dodati i ukloniti sa bilo kog mesta.
- Moguće je sortiranje liste.

Uređena linearna lista

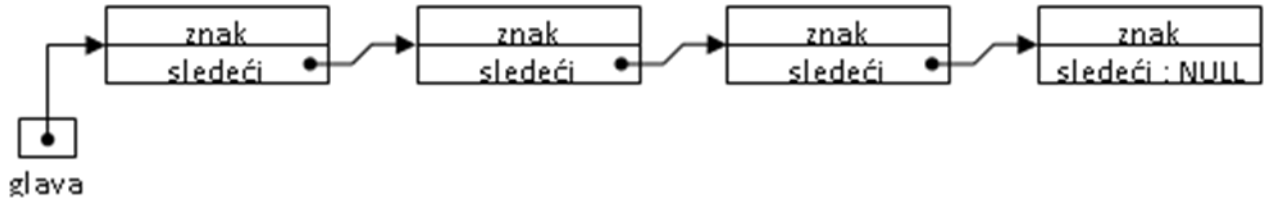
- Ako se po informacionom sadržaju može definisati **funkcija poređenja** i ako su elementi poređani u rastućem ili opadajućem redosledu, za listu se kaže da je **uređena** (u suprotnom kažemo da je **neuređena**).

Karakteristični termini

- Element liste nazivamo **čvor.**
- Lista je **HOMOGENA** ako su svi elementi liste istog tipa.
- Poslednji čvor u listi se naziva **rep.**
- Pokazivačko polje u repu ima vrednost **NULL.**
- Ako je lista **kružna**, pokazivačko polje u repu ukazuje na prvi čvor.

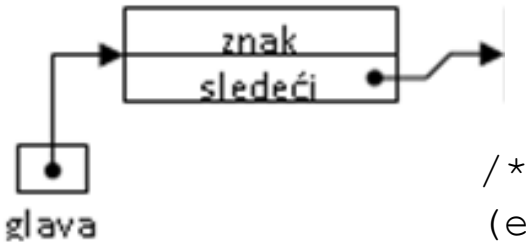
Fizička realizacija

- Fizička realizacija je zavisna od namene i u svakom slučaju je spregnuta.

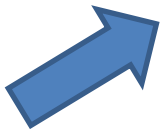


Element liste

- Element liste sastoji se od dva polja:
  - podatak (u ovom slučaju proizvoljan znak)
  - pokazivač na sledeći element liste




```
/* Struktura podataka koja predstavlja slog
(element) liste. */
typedef struct cvor {
                        char znak;
                        struct cvor *sledeci;
                    }Tcvor;
```



Samoupućujuća struktura

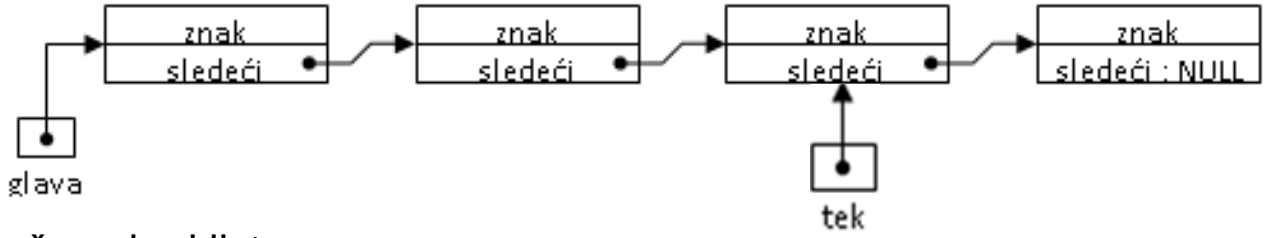
Operacije nad listom

Osnovne operacije koje se mogu izvesti nad listama uključuju sledeće:

- Umetanje čvora u listu
- Brisanje tekućeg čvora iz liste
- Pristup čvoru radi čitanja i upisa

Da bismo mogli pristupati elementima liste potreban nam je pokazivač na prvi element liste, koji se naziva **glava** liste.

Grafička predstava jednostruko spregnute liste u ovom zadatku je:



- NULL označava kraj liste.
- Uopšteno, pokazivačka promenljiva sadrži adresu elementa (sloga liste) na koji pokazuje.

Operacije sa listom su:

1) inicijalizacija liste,
2) unos novog elementa na pocetak liste,
3) listanje liste (prikaz svih elemenata iz liste),
4) brisanje elementa iz liste i
5) brisanje liste (oslobađanje memorije).

1) Inicijalizacija liste predstavlja postavljanje glave liste na NULL.

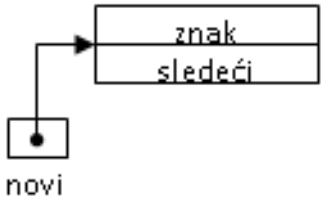
Kada glava liste ima vrednost NULL znači da je lista prazna.

```
*glava=NULL;
```

2) Unos novog elementa u listu može se obaviti dodavanjem elementa na početak liste ili dodavanjem elementa na kraj liste. Dodajemo na kraj.
Sa:

```
novi=(Tcvor *)malloc(sizeof(Tcvor));
```

formira se slog koji je tipa 'Tcvor' i pokazivač 'novi' pokazuje na novoformirani slog.

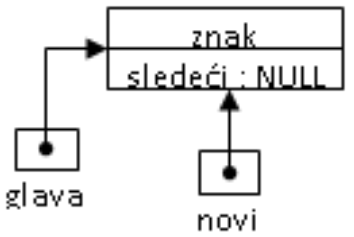


Polja sloga popunjavaju na osnovu sledećih naredbi:

```
c=getc();
novi->znak=c;
novi->sledeci=NULL;
```

- Ako je lista prazna tada se novi element ubacuje kao prvi element liste:

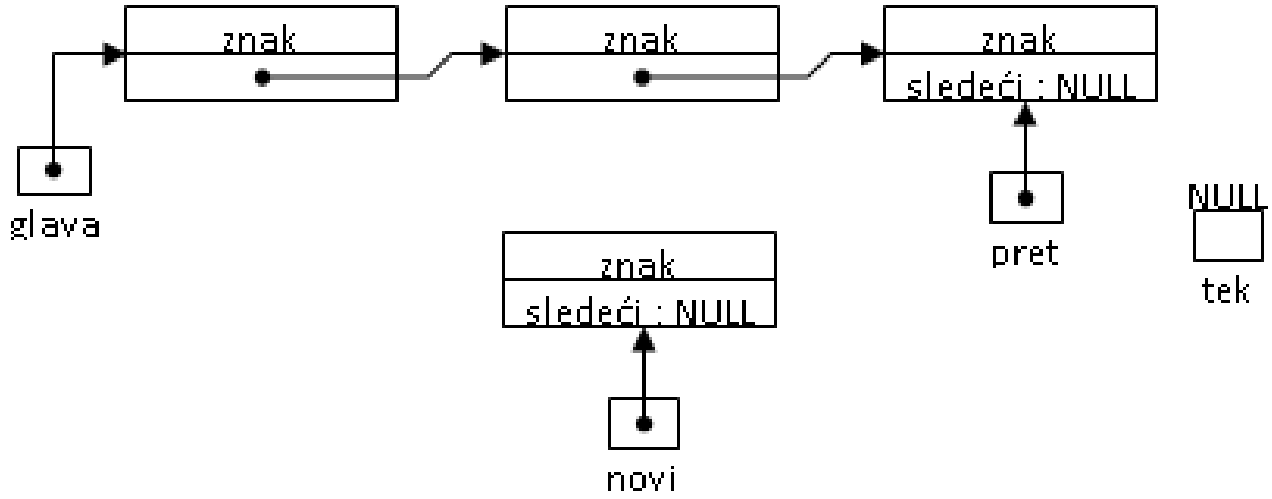
```
if(glava==NULL)
{
        glava=novi;
        return;
}
```



Ako lista već postoji preskočiće se gornja if naredba i unos će se obaviti sledećim delom koda:

```
tek=glava;
pret=glava;
while(tek!=NULL)
{
        pret=tek;
        tek=tek->sledeci;
}
pret->sledeci=novi;
```

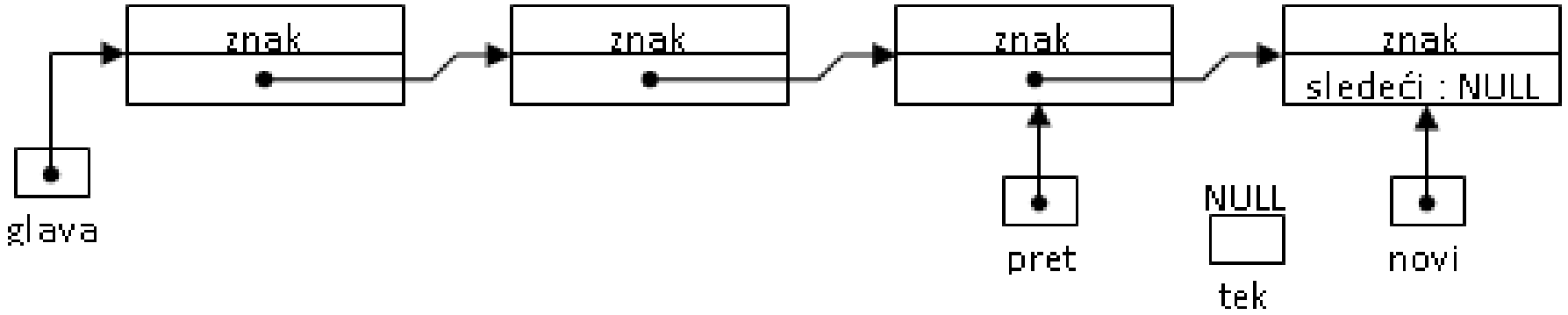
- Nakon prolaska kroz while ciklus imaćemo sledeću situaciju:
  '**tek**' ne pokazuje ni na šta, a '**pret**' koji ga je "pratio" pokazaće na poslednji slog iz liste.



Nakon:

```
pret->sledeci=novi;
```

novi slog će biti povezan kao poslednji element u listi.



3) Listanje liste predstavlja prikazivanje svih elemenata iz liste

- Ako je glava==NULL znači da je lista prazna i nemamo šta prikazati.
- Ako lista nije prazna, novim pokazivačem polazi se od prvog sloga liste (tek=glava) i dok se ne dođe do kraja liste (tek==NULL) prikazuju se slogovi liste.

```
tek=glava;
while(tek!=NULL)
{
    printf("   %c",tek->znak);     /* Sa strelicom (->) se pristupa
                                   elementu sloga (strukture). */
    tek=tek->sledeci;              /* Prelazimo na sledeći element iz liste. */
}
```

4) Brisanje elementa iz liste realizuje se tako što se mora element prvo pronaći (vrši se traženje elementa u listi).

- Traženje elementa liste se može realizovati sledećim delom koda:

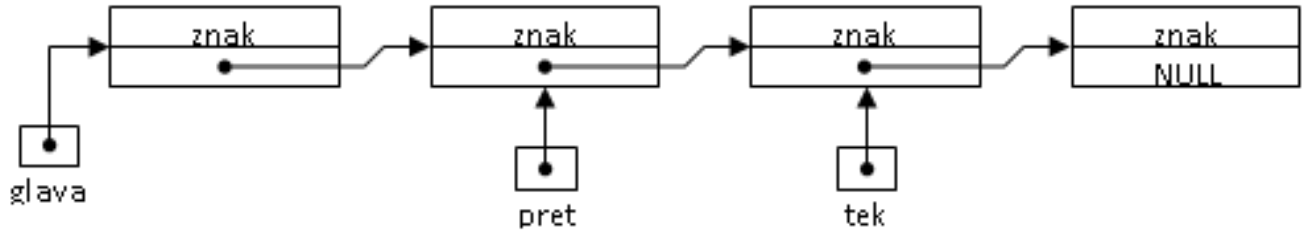
```
tek=glava;
pret=glava;
while(tek!=NULL && (tek->znak != c))
{
        pret=tek;
        tek=tek->sledeci;
}
```

Pri čemu je 'c' promenljiva čija vrednost se traži. Nakon završetka while ciklusa ako pokazivač 'tek' ima vrednost NULL znači da elementa nema u listi. U slučaju da je tek!=NULL element se nalazi u listi, a pokazivač 'pret' pokazuje na njegovog prethodnika iz liste.

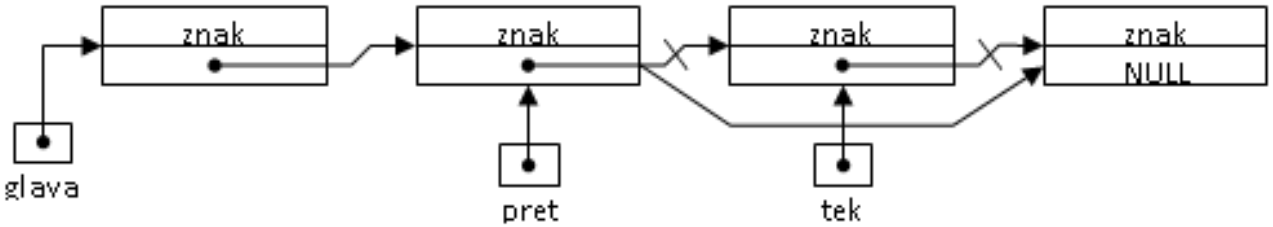
Razlikujemo dva slučaja brisanja:

a) da slog koji se briše nije prvi element liste,
b) da slog koji se briše jeste prvi element liste.

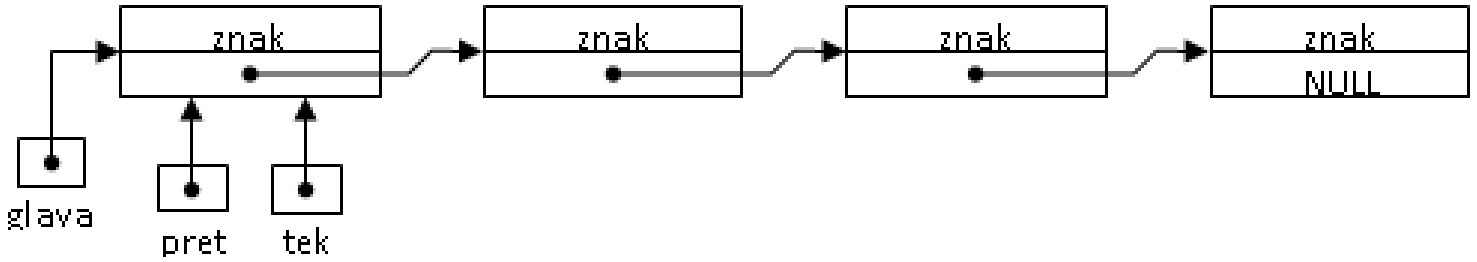
a) Slog koji se briše nije prvi element liste



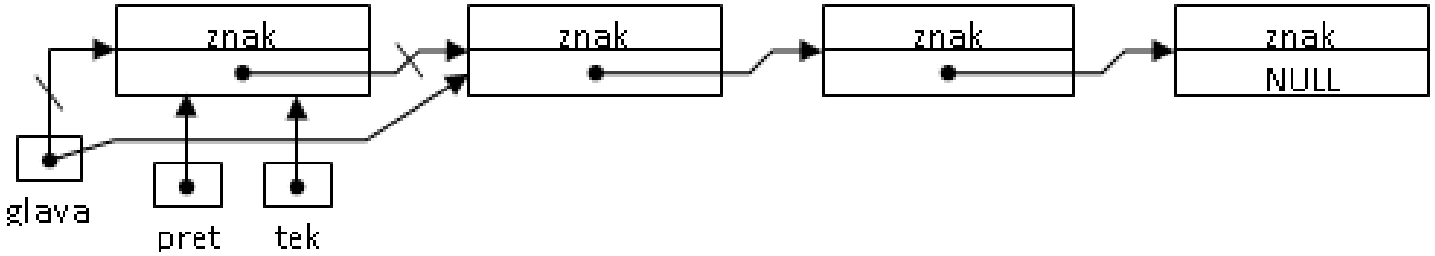

```
pret->sledeci=tek->sledeci;
tek->sledeci=NULL;
```



b) Slog koji se briše jeste prvi element liste

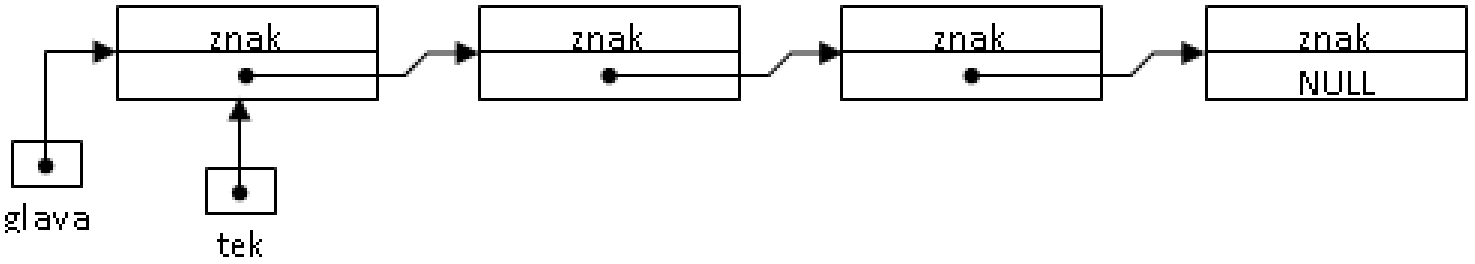


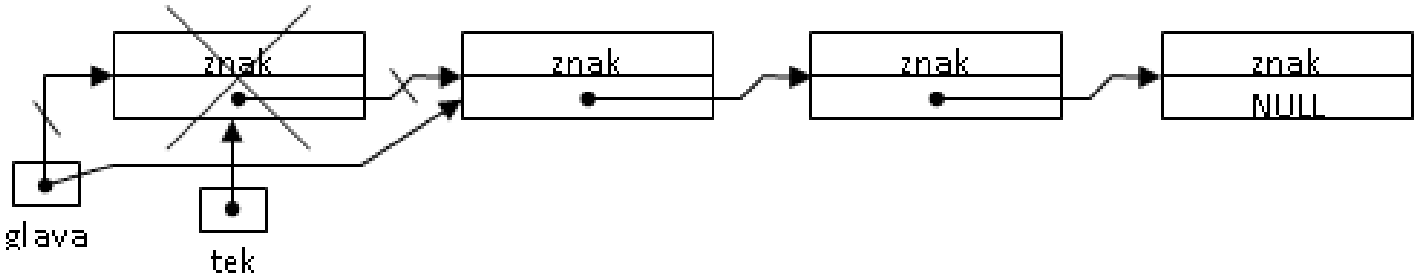

```
glava=tek->sledeci;
```



5) Brisanje liste

```
while(glava!=NULL)
{
        tek=glava;
        glava=tek->sledeci;
        free(tek);
}
```





Još neke operacije sa spregnutim listama

- Inverzija poretka(**z104.c**)
- Konkatenacija(spajanje lista)
- Nalaženje i-tog čvora
- Razbijanje liste u dve liste
- Kopiranje liste