Liste

Uvod

Pri rešavanju mnogih problema potrebne su nam dinamičke liste, čiju veličinu ne moramo znati u trenutku kompajliranja, već je možemo definisati i u toku rada programa. **Povezana lista** je struktura podataka koja se koristi za modeliranje ovakvih dinamičkih lista.

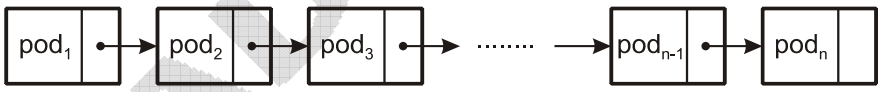
Koncept

Nizovi su u memoriji predstavljeni korišćenjem sekvencijalnog mapiranja, tako da su elementi niza podjednako udaljeni jedan od drugog. Međutim, ovakav pristup ima sledeće nedostatke:

- Umetanje novog ili brisanje postojećeg elementa iz niza je vremenski veoma skupa operacija, jer zahteva pomeranje određenog broja postojećih elemenata.
- U slučaju korišćenja statičkih nizova neophodno je poznavanje maksimalnog broja elemenata još u trenutku pisanja programa. Pored ovog nedostatka, dodatni problem je i to što bez obzira na stvaran broj elemenata koji će biti smešten u niz, program uvek alocira maksimalnu veličinu niza, što dovodi do bespotrebnog trošenja memorije.
- Korišćenje dinamičkih nizova rešava problem poznavanja maksimalnog broja elemenata u trenutku pisanja programa, ali uvodi i neke nove probleme. Svaka promena veličine niza zahteva dodatnu alokaciju i dealokaciju memorije, kao i kopiranje sadržaja elemenata u novi memorijski prostor, što drastično pogoršava performanse programa.

Da bi se prevazišli navedeni problemi uvodi se koncept povezanih elemenata. U ovakvom pristupu nije neophodno da elementi budu međusobno na podjednakom rastojanju. Umesto toga elemente možemo smestiti bilo gde u memoriji, a zatim ih povezati tako da svaki element (osim prvog) bude povezan sa prethodnim elementom u listi. To se može postići tako što se u svaki element upiše i adresa njegovog sledbenika, što zahteva da svaki element bude u mogućnosti da pored svojih podataka čuva i adresu sledećeg elementa. Zbog toga svaki element mora biti slog koji sadrži dva dela: jedan koji čuva neophodne podatke (nazovimo ga **podatak** ili **data field**) i drugi koji čuva adresu sledbenika (**veza** ili **link**).

Dakle, **povezana lista** je lista elemenata koji su proizvoljno raspoređeni u memoriji i koji su međusobno povezani **linkom**, tj. smeštanjem adrese svakog elementa (osim prvog) u njegovog prethodnika.



Slika ### Povezana lista

Definisanje

Za realizaciju koncepta povezanih lista u programskom jeziku Pascal koriste se slogovi i pokazivači (pointeri). Element liste koja će sadržati celobrojne podatke se definiše na sledeći način:

```
type pokelement=^element;
     element=record
               podatak:integer;
               sledeci:pokelement;
             end;
```

Prethodne linije koda definišu slogovni tip *element* koji će predstavljati element liste. Svaki element liste sadrži celobrojni podatak i pokazivač na sledeći element liste. Podatak unutar liste može biti i bilo kog drugog prostog ili složenog tipa.

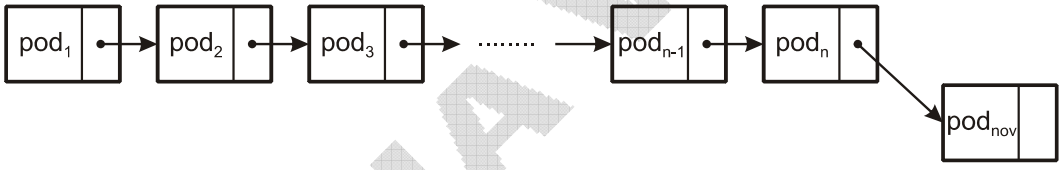
Dodavanje elementa u listu

Da bismo uopšte mogli da baratamo listom, potrebno je da nam njen prvi element (ukoliko postoji) uvek bude poznat. Kada znamo prvi element, korišćenjem pokazivača na njegovog sledbenika lako dolazimo do sledećeg elementa u listi. Pošto lista može biti i prazna, umesto da pamtimo prvi element liste, koristimo pokazivač na prvi element, koji nazivamo **glava liste**. Dakle, glava liste je pokazivač na prvi element liste, a u slučaju da je lista prazna vrednost ovog pokazivača je NIL. Veoma je bitno u svakom trenutku imati informaciju o glavi liste, jer je samo tako moguće pristupiti listi. Iz tog razloga prilikom pisanja programa treba voditi računa da uvek postoji bar jedan pokazivač na početak liste, koji se neće koristiti u druge svrhe, čime bi se došlo u opasnost da se izgubi informacija o početku liste.

Prilikom dodavanja novog elementa u listu moramo razlikovati dva slučaja:

1. **Lista je bila prazna** pre dodavanja novog elementa
2. **Lista nije bila prazna** pre dodavanja novog elementa

Pre dodavanja elementa u praznu listu, glava liste je pokazivač koji ima vrednost NIL (ne pokazuje ni na jedan element). Nakon dodavanja novog elementa, glava liste postaje pokazivač na taj element. U drugom slučaju lista nije prazna i glava pokazuje na prvi element. Da bi se dodao element na kraj liste, potrebno je proći kroz sve elemente liste kako bi se pronašao poslednji element, a zatim iza njega dodao novi element. Dodavanjem novog elementa na kraj liste očigledno ne dolazi do promene glave liste. Imajući u vidu navedene postupke dodavanja novog elementa, jasno je da se algoritmi u ova dva slučaja drastično razlikuju, pa ih iz tog razloga i u programu treba razdvojiti.



Slika ### Dodavanje elementa u listu

Prolazak kroz listu

U radu sa listama je često potrebno kretati se kroz listu, od elementa do elementa. Najčešće je u pitanju obrada podataka u listi, pretraga liste kako bi se našao odgovarajući element ili traženje kraja liste. U svim ovim slučajevima algoritam je sličan. Uvodi se pomoćni pokazivač koji je inicijalno jednak glavi liste, odnosno pokazuje na prvi element liste ukoliko ona nije prazna. Nakon provere da li pokazivač pokazuje na neki element (ima vrednost različitu od NIL) vrši se obrada podatka u tom element (štampa, upoređivanje, račun...). Po završetku obrade podatka, pomoćni pokazivač dobija vrednost pokazivača na sledeći element, a čitav postupak se ponavlja sve dok pomoćni pokazivač ima nenultu vrednost, tj. dok pokazuje na neki element. Kada pomoćni pokazivač dobije vrednost NIL, to znači da smo došli do kraja liste.

Sledeći isečak koda pokazuje prolazak kroz listu čiji je početak definisan pokazivačem *glava* i štampanje podataka zapisanih u svim njenim elementima:

```
var pom:pokelement;
     ...
pom:=glava;
while pom<>NIL do
  begin
    write(pom^.podatak,'   ');
    pom:=pom^.sledeci;
  end;
```

Primer

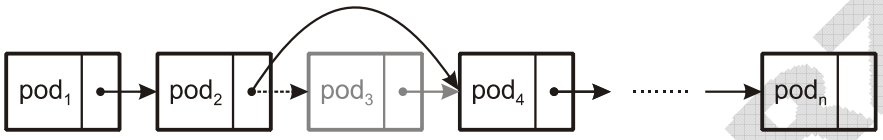
Listing programa za kreiranje i štampanje povezane liste:

```
program Kreiranje;
type   pokelement=^element;
       element=record
                   podatak:integer;
                   sledeci:pokelement;
               end;

{ funkcija za dodavanje novog elementa na kraj liste }
function dodaj(p:pokelement; n:integer):pokelement;
var pom:pokelement;
begin
   { ako je postojeca lista prazna onda se dodaje novi element kao pocetni }
   if p=NIL then
      begin
         { kreiranje novog elementa }
         new(p);

         { novom elementu se dodeljuje prosledjeni podatak }
         p^.podatak := n;
         p^.sledeci := NIL;
      end
   else
      begin
         pom := p;

         { prolazi se kroz postojecu listu da bi se dobio pokazivac na poslednji element }
         while pom^.sledeci <> NIL do   pom := pom^.sledeci;

         { kreiranje novog elementa }
         new(pom^.sledeci);

         { novom elementu se dodeljuje prosledjeni podatak, a njegova adresa se upisuje kao link
         prethodnog elementa }
         pom := pom^.sledeci;
         pom^.podatak := n;
         pom^.sledeci := NIL;
      end;

   dodaj:=p;
end;

{ procedura za stampanje liste }
procedure stampaj_listu( p:pokelement );
var pom:pokelement;
begin
   pom := p;
   if p <> NIL then
      begin
         repeat
            write(pom^.podatak,'   ');
            pom:=pom^.sledeci;
         until pom = NIL;

         writeln;
      end
   else     writeln('Lista je prazna.');

   writeln;
end;

var    n,i,x:integer;
       glava:pokelement;

begin
   glava := NIL ;

   writeln('Unesi broj elemenata u listi:');
   readln(n);
```

```
   for i:=1 to n do
      begin
         writeln('Unesi ',i,'. element:');
         readln(x);
         glava := dodaj( glava, x );
      end;

   writeln('Kreirana lista je:');
   stampaj_listu( glava );
end.
```

Brisanje elementa iz liste

Da bismo obrisali određeni element iz liste potrebno je da znamo njegovu poziciju u listi. Jedan od načina je da navedemo njegov redni broj, uz pretpostavku da su elementi numerisani redom od 1 do n. Kada znamo redni broj elementa, krećemo se kroz listu kako bismo pronašli pokazivač na traženi element. Pored toga, potrebno je da znamo i pokazivač na prethodni element u listi, kako bismo njegovom linku dodelili pokazivač na element koji sledi elementu koji brišemo. Nakon toga možemo da obrišemo traženi element, odnosno da oslobodimo memoriju koju on zauzima.



Slika ### Brisanje elementa iz liste

Primer

Listing programa za kreiranje liste, brisanje zadatog elementa i štampanje povezane liste pre i posle brisanja:

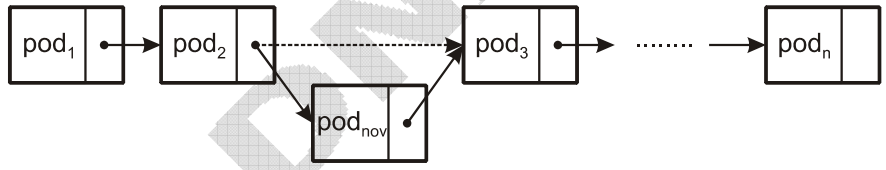
```
program Brisanje;
type   pokelement=^element;
       element=record
                    podatak:integer;
                    sledeci:pokelement;
                end;

{ funkcija za dodavanje novog elementa na kraj liste }
function dodaj(p:pokelement; n:integer):pokelement;
var pom:pokelement;
begin
   { ako je postojeca lista prazna onda se dodaje novi element kao pocetni }
   if p=NIL then
      begin
         { kreiranje novog elementa }
         new(p);

         { novom elementu se dodeljuje prosledjeni podatak }
         p^.podatak := n;
         p^.sledeci := NIL;
      end
   else
      begin
         pom := p;

         { prolazi se kroz postojecu listu da bi se dobio pokazivac na poslednji element }
         while pom^.sledeci <> NIL do   pom := pom^.sledeci;

         { kreiranje novog elementa }
         new(pom^.sledeci);

         { novom elementu se dodeljuje prosledjeni podatak, a njegova adresa se upisuje kao link
         prethodnog elementa }
         pom := pom^.sledeci;
         pom^.podatak := n;
```

```
         pom^.sledeci := NIL;
      end;

   dodaj:=p;
end;

{ procedura za stampanje liste }
procedure stampaj_listu( p:pokelement );
var pom:pokelement;
begin
   pom := p;
   if p <> NIL then
      begin
         repeat
            write(pom^.podatak,'   ');
            pom:=pom^.sledeci;
         until pom = NIL;

         writeln;
      end
   else  writeln('Lista je prazna.');

   writeln;
end;

{ funkcija za odredjivanje duzine liste }
function duzina( p:pokelement ):integer;
var broj:integer;
begin
   broj := 0 ;
   while p <> NIL do
      begin
         broj := broj+1;
         p := p^.sledeci;
      end;

   duzina := broj;
end;

{ funkcija za brisanje zadatog elementa iz liste }
function obrisi( p:pokelement; redni_broj:integer ):pokelement;
var   prethodni, trenutni:pokelement;
      i:integer;
begin
   if p = NIL then
      writeln('Lista je prazna.')
   else
      begin
         if redni_broj > duzina(p) then
            writeln('Greska.')
         else
            begin
               prethodni := NIL;
               trenutni := p;
               i := 1;
               while i < redni_broj do
                  begin
                     prethodni := trenutni;
                     trenutni := trenutni^.sledeci;
                     i := i+1;
                  end;
               if prethodni = NIL then
                  begin
                     p := trenutni^.sledeci;
                     dispose( trenutni );
                  end
               else
                  begin
                     prethodni^.sledeci := trenutni^.sledeci;
                     dispose( trenutni );
                  end;
            end;
      end;
```

```
   obrisi := p;
end;

var    n,i,x:integer;
       glava:pokelement;

begin
   glava := NIL;

   writeln('Unesi broj cvorova u listi:');
   readln(n);

   for i:=1 to n do
      begin
         writeln('Unesi ',i,'. element:');
         readln(x);
         glava := dodaj( glava, x );
      end;

   writeln('Lista pre brisanja elementa:');
   stampaj_listu( glava );

   writeln('Unesi redni broj elementa:');
   readln(n);
   glava := obrisi(glava , n );

   writeln('Lista posle brisanja elementa:');
   stampaj_listu( glava );
end.
```

Umetanje novog elementa iza određenog elementa u listi

Da bismo umetnuli novi element na određeno mesto u listi, neophodno je da znamo redni broj elementa u listi iza koga će novi element biti umetnut. Pretpostavljamo da su svi elementi liste numerisani rednim brojevima od 1 do *n*. Kada znamo broj elementa, krećemo se kroz listu kako bismo dobili pokazivač na element sa zadatim rednim brojem. Zatim kreiramo novi element, a njegovom linku dodeljujemo pokazivač na element koji sledi iza elementa sa navedenim rednim brojem. Nakon toga, elementu sa zadatim rednim brojem postavljamo link tako da pokazuje novi element.



Slika ### Umetanje novog elementa iza određenog elementa u listi

Primer

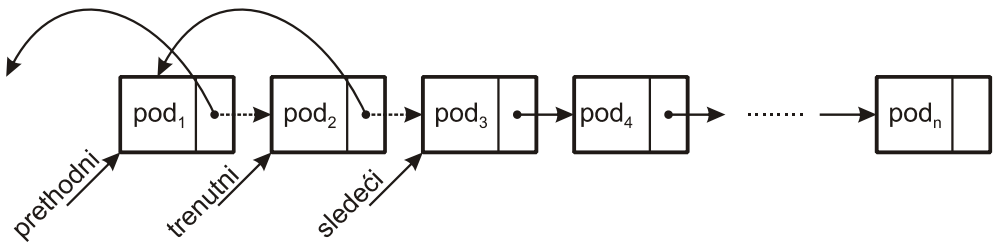
```
program Umetanje;
type   pokelement=^element;
       element=record
                    podatak:integer;
                    sledeci:pokelement;
                end;

{ funkcija za dodavanje novog elementa na kraj liste }
function dodaj(p:pokelement; n:integer):pokelement;
var pom:pokelement;
begin
   { ako je postojeca lista prazna onda se dodaje novi element kao pocetni }
   if p=NIL then
      begin
         { kreiranje novog elementa }
         new(p);
```

```
         { novom elementu se dodeljuje prosledjeni podatak }
         p^.podatak := n;
         p^.sledeci := NIL;
      end
   else
      begin
         pom := p;

         { prolazi se kroz postojecu listu da bi se dobio pokazivac na poslednji element }
         while pom^.sledeci <> NIL do   pom := pom^.sledeci;

         { kreiranje novog elementa }
         new(pom^.sledeci);

         { novom elementu se dodeljuje prosledjeni podatak, a njegova adresa se upisuje kao link
         prethodnog elementa }
         pom := pom^.sledeci;
         pom^.podatak := n;
         pom^.sledeci := NIL;
      end;

   dodaj:=p;
end;

{ funkcija za odredjivanje duzine liste }
function duzina( p:pokelement ):integer;
var broj:integer;
begin
   broj := 0 ;
   while p <> NIL do
      begin
         broj := broj+1;
         p := p^.sledeci;
      end;

   duzina := broj;
end;

{ funkcija za umetanje novog elementa iza odredjenog elementa u listi }
function umetni( p:pokelement; redni_broj, vrednost:integer ):pokelement;
var   novi, pom: pokelement;
      i:integer;
begin
   if ( redni_broj < 0 ) or ( redni_broj > duzina(p) ) then
      begin
         writeln('Greska! Zadati element ne postoji.');
         exit;
      end;

   if redni_broj = 0 then
      begin
         new(novi);
         novi^.podatak := vrednost;
         novi^.sledeci := p;
         p := novi ;
      end
   else
      begin
         pom := p ;
         i := 1;
         while i < redni_broj do
            begin
               i := i+1;
               pom := pom^.sledeci;
            end;

         new(novi);
         novi^.podatak := vrednost;
         novi^.sledeci := pom^.sledeci;
         pom^.sledeci := novi;
      end;

   umetni := p;
end;
```

```
{ procedura za stampanje liste }
procedure stampaj_listu( p:pokelement );
var pom:pokelement;
begin
   pom := p;
   if p <> NIL then
      begin
         repeat
            write(pom^.podatak,'   ');
            pom:=pom^.sledeci;
         until pom = NIL;

         writeln;
      end
   else     writeln('Lista je prazna.');

   writeln;
end;

var   n,i,x:integer;
      glava:pokelement;

begin
   glava := NIL;

   writeln('Unesi broj cvorova u listi:');
   readln(n);

   for i:=1 to n do
      begin
         writeln('Unesi ',i,'. element:');
         readln(x);
         glava := dodaj( glava, x );
      end;

   writeln('Lista pre umetanja elementa:');
   stampaj_listu( glava );

   writeln('Unesi redni broj elementa iza koga zelis da novi element bude umetnut:');
   readln(n);

   writeln('Unesi vrednost novog elementa:');
   readln(x);

   glava := umetni( glava, n, x );

   writeln('Lista posle umetanja elementa:');
   stampaj_listu( glava );
end.
```

Okretanje redosleda u listi

Da bismo izvršili okretanje redosleda u listi neophodno je da za svaki element znamo pokazivače na njegovog prethodnika i sledbenika. Kada su nam ovi pokazivači poznati, onda možemo link trenutnog elementa da postavimo tako da pokazuje na prethodnika. Nakon toga trenutni element proglašavamo za prethodnika, a njegovog sledbenika za trenutni.



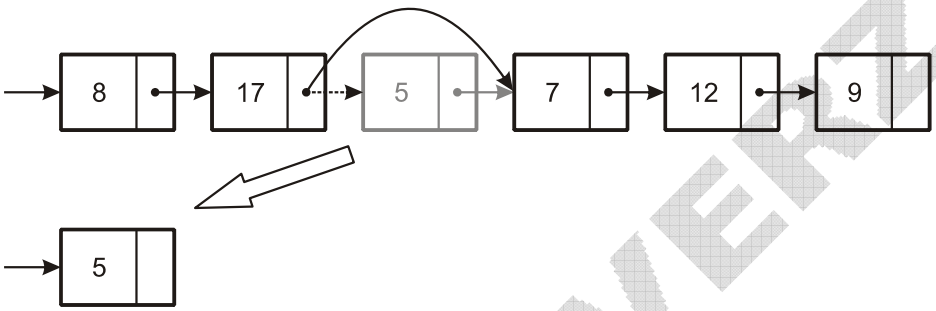
Slika ### Okretanje redosleda u listi

Deo koda koji bi obavljao navedenu proceduru mogao bi da izgleda ovako:

```
prethodni := NIL;
while trenutni <> NIL do
   begin
      sledeci := trenutni^.sledeci;
      trenutni^.sledeci := prethodni;
      prethodni := trenutni;
      trenutni := sledeci;
   end
```

Sortiranje liste

Da bismo sortirali listu, prvo prolazimo kroz nju kako bismo pronašli element sa najmanjom vrednošću podatka. Nakon toga uklanjamo taj element iz liste i dodajemo ga na kraj druge liste, koja je inicijalno bila prazna. Ovaj proces ponavljamo sve dok početna lista ne postane prazna. Na kraju ostaje samo da funkcija vrati pokazivač na listu u koju su prebačeni svi elementi.



Slika ### Sortiranje liste

Primer

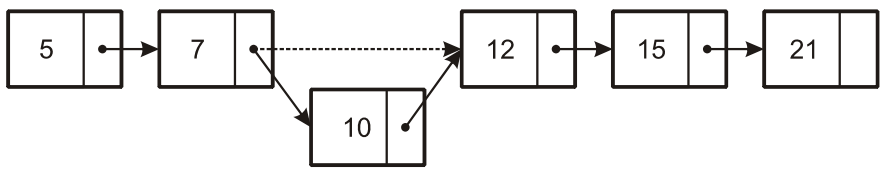
```
program Sortiranje;
type   pokelement=^element;
       element=record
                    podatak:integer;
                    sledeci:pokelement;
                end;

{ funkcija za dodavanje novog elementa na kraj liste }
function dodaj(p:pokelement; n:integer):pokelement;
var pom:pokelement;
begin
   { ako je postojeca lista prazna onda se dodaje novi element kao pocetni }
   if p=NIL then
      begin
         { kreiranje novog elementa }
         new(p);

         { novom elementu se dodeljuje prosledjeni podatak }
         p^.podatak := n;
         p^.sledeci := NIL;
      end
   else
      begin
         pom := p;

         { prolazi se kroz postojecu listu da bi se dobio pokazivac na poslednji element }
         while pom^.sledeci <> NIL do   pom := pom^.sledeci;

         { kreiranje novog elementa }
         new(pom^.sledeci);
```

```
         { novom elementu se dodeljuje prosledjeni podatak, a njegova adresa se upisuje kao link
         prethodnog elementa }
         pom := pom^.sledeci;
         pom^.podatak := n;
         pom^.sledeci := NIL;
      end;

   dodaj:=p;
end;

{ procedura za stampanje liste }
procedure stampaj_listu( p:pokelement );
var pom:pokelement;
begin
   pom := p;
   if p <> NIL then
      begin
         repeat
            write(pom^.podatak,'   ');
            pom:=pom^.sledeci;
         until pom = NIL;

         writeln;
      end
   else  writeln('Lista je prazna.');

   writeln;
end;

{ funkcija za okretanja redosleda u listi }
function okreni(p:pokelement):pokelement;
var prethodni, trenutni, sledeci:pokelement;
begin
   prethodni := NIL;
   trenutni := p;

   while trenutni <> NIL do
      begin
         sledeci := trenutni^.sledeci;
         trenutni^.sledeci := prethodni;
         prethodni := trenutni;
         trenutni := sledeci;
      end;

   okreni := prethodni;
end;

{ funkcija za sortiranje liste u rastucem redosledu }
function sortiraj(p:pokelement):pokelement;
var pom1,pom2,min,prethodni,q:pokelement;
begin
   q := NIL;

   while p <> NIL do
      begin
         prethodni := NIL;
         pom1 := p;
         min := pom1;

         pom2 := p^.sledeci;
         while pom2 <> NIL do
            begin
               if pom2^.podatak < min^.podatak then
                  begin
                     min := pom2;
                     prethodni := pom1;
                  end;
               pom1 := pom2;
               pom2 := pom2^.sledeci;
            end;

         if prethodni = NIL then
            p := min^.sledeci
         else
```

```
            prethodni^.sledeci := min^.sledeci;

         min^.sledeci := NIL;

         if q = NIL then
            q := min  { premesta element sa najmanjom vrednoscu podatka
                      iz liste p na pocetak liste q }
         else
            begin
               pom1 := q;

               { prolazi kroz listu q da bi pronasao njen poslednji element }
               while pom1^.sledeci <> NIL do pom1 := pom1^.sledeci;

               pom1^.sledeci := min;    { premesta element sa najmanjom vrednoscu podatka
                                          iz liste p na kraj liste q }
            end;
      end;

   sortiraj := q;
end;

var   n,x,i:integer;
      glava:pokelement;

begin
   glava := NIL;

   writeln('Unesi broj cvorova u listi:');
   readln(n);

   for i:=1 to n do
      begin
         writeln('Unesi ',i,'. element:');
         readln(x);
         glava := dodaj( glava, x );
      end;

   writeln('Lista pre sortiranja:');
   stampaj_listu( glava );

   glava := sortiraj(glava);

   writeln('Sortirana lista je:');
   stampaj_listu( glava );

   glava := okreni(glava);

   writeln('Okrenuta lista je:');
   stampaj_listu( glava );
end.
```

Umetanje novog elementa u sortiranu listu

Da bismo umetnuli novi element u listu koja je prethodno sortirana, upoređujemo redom podatke u listi sa vrednošću podatka novog elementa. Ovaj proces se ponavlja sve dok ne dobijemo pokazivač na element koji se nalazi ispred elementa čiji je podatak veći od podatka u novom elementu.



Slika ### Umetanje elementa u sortiranu listu

Primer

```
program UmetanjeSort;
type   pokelement=^element;
       element=record
                   podatak:integer;
                   sledeci:pokelement;
                end;

{ funkcija za dodavanje novog elementa na kraj liste }
function dodaj(p:pokelement; n:integer):pokelement;
var pom:pokelement;
begin
   { ako je postojeca lista prazna onda se dodaje novi element kao pocetni }
   if p=NIL then
      begin
         { kreiranje novog elementa }
         new(p);

         { novom elementu se dodeljuje prosledjeni podatak }
         p^.podatak := n;
         p^.sledeci := NIL;
      end
   else
      begin
         pom := p;

         { prolazi se kroz postojecu listu da bi se dobio pokazivac na poslednji element }
         while pom^.sledeci <> NIL do   pom := pom^.sledeci;

         { kreiranje novog elementa }
         new(pom^.sledeci);

         { novom elementu se dodeljuje prosledjeni podatak, a njegova adresa se upisuje kao link
         prethodnog elementa }
         pom := pom^.sledeci;
         pom^.podatak := n;
         pom^.sledeci := NIL;
      end;

   dodaj:=p;
end;

{ procedura za stampanje liste }
procedure stampaj_listu( p:pokelement );
var pom:pokelement;
begin
   pom := p;
   if p <> NIL then
      begin
         repeat
            write(pom^.podatak,'   ');
            pom:=pom^.sledeci;
         until pom = NIL;

         writeln;
      end
   else  writeln('Lista je prazna.');

   writeln;
end;

{ funkcija za sortiranje liste u rastucem redosledu }
function sortiraj(p:pokelement):pokelement;
var pom1,pom2,min,prethodni,q:pokelement;
begin
   q := NIL;

   while p <> NIL do
      begin
         prethodni := NIL;
         pom1 := p;
         min := pom1;
```

```
         pom2 := p^.sledeci;
         while pom2 <> NIL do
            begin
               if pom2^.podatak < min^.podatak then
                  begin
                     min := pom2;
                     prethodni := pom1;
                  end;
               pom1 := pom2;
               pom2 := pom2^.sledeci;
            end;

         if prethodni = NIL then
            p := min^.sledeci
         else
            prethodni^.sledeci := min^.sledeci;

         min^.sledeci := NIL;

         if q = NIL then
            q := min  { premesta element sa najmanjom vrednoscu podatka
                        iz liste p na pocetak liste q }
         else
            begin
               pom1 := q;

               { prolazi kroz listu q da bi pronasao njen poslednji element }
               while pom1^.sledeci <> NIL do pom1 := pom1^.sledeci;

               pom1^.sledeci := min;    { premesta element sa najmanjom vrednoscu podatka
                                          iz liste p na kraj liste q }
            end;
      end;

   sortiraj := q;
end;

{ funkcija za umetanje novog elementa u sortiranu listu }
function sort_dodaj(p:pokelement; n:integer):pokelement;
var trenutni, prethodni, novi:pokelement;
begin
   trenutni := p;
   prethodni := NIL;

   while trenutni^.podatak < n do
      begin
         prethodni := trenutni;
         trenutni := trenutni^.sledeci;
      end;

   new(novi);

   if prethodni = NIL then { element ce biti umetnut na pocetak liste }
      begin
         novi^.podatak := n;
         novi^.sledeci := p;
         p := novi;
      end
   else
      begin
         novi^.podatak := n;
         novi^.sledeci := prethodni^.sledeci;
         prethodni^.sledeci := novi;
      end;

   sort_dodaj := p;
end;

var   n,x,i:integer;
      glava:pokelement;

begin
   glava := NIL;
```

```
   writeln('Unesi broj cvorova u listi:');
   readln(n);

   for i:=1 to n do
      begin
         writeln('Unesi ',i,'. element:');
         readln(x);
         glava := dodaj( glava, x );
      end;

   writeln('Lista pre sortiranja:');
   stampaj_listu( glava );

   glava := sortiraj(glava);

   writeln('Sortirana lista je:');
   stampaj_listu( glava );

   writeln('Unesi vrednost novog elementa:');
   readln(x);

   glava := sort_dodaj( glava, x );

   writeln('Lista posle umetanja elementa je:');
   stampaj_listu( glava );
end.
```