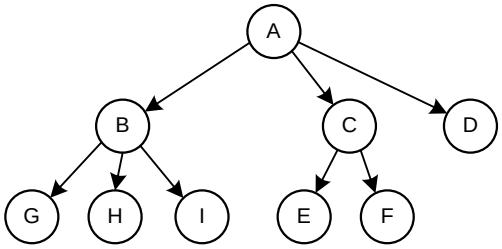
Stabla

Uvod

Stabla se koriste za predstavljanje hijerarhijske strukture neke kolekcije podataka. Stablo se sastoji od jednog ili više čvorova koji zadovoljavaju sledeće uslove:

1. Postoji jedan čvor koji predstavlja koren stabla i
2. Ostali čvorovi su podeljeni u *n* disjunktnih skupova čvorova, tako da svaki skup predstavlja stablo.

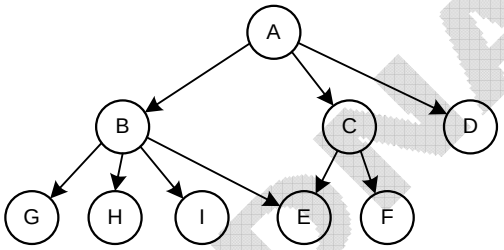
Na Slici ### je prikazan primer stabla:



Slika ### Šematski prikaz stabla

Struktura prikazana na prethodnoj slici je stablo iz razloga što je to skup čvorova {A, B, C, D, E, F, G, H, I}, pri čemu je čvor A koren stabla, a ostali čvorovi su podeljeni u tri disjunktna skupa {B, G, H, I}, {C, E, F} i {D}. Svaki od ova tri skupa je takođe stablo, zato što zadovoljava oba uslova navedena u definiciji.

Struktura prikazana na Slici ### nije stablo, zato što činjenica da je čvor E podeljen onemogućava razdvajanje čvorova od B do I u disjunktne skupove.



Slika ### Šematski prikaz strukture koja nije stablo

Stepen čvora stabla

Stepen čvora stabla je broj podstabala kojima je ovaj čvor koren. Drugim rečima, stepen je broj naslednika (dece) čvora. Ukoliko je stepen čvora nula, takav čvor nazivamo **list stabla**.

Stepen stabla

Stepen stabla se definiše kao maksimum stepena svih čvorova stabla.

Nivo čvora

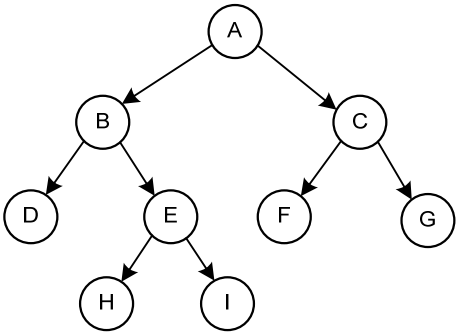
Nivo čvora definišemo tako što uzimamo da je nivo korena stabla jednak 1, a zatim uvećavamo ovaj broj za jedan pri svakom skoku od korena ka podstablima. Tako, nivo svih naslednika korena će biti 2, nivo njihovih naslednika 3 itd. **Dubinu stabla** zatim definišemo kao maksimalni nivo čvora u stablu.

Binarno stablo

Binarno stablo je specijalni slučaj stabla u kome ni jedan čvor nema stepen veći od 2. To praktično znači da svaki čvor ima najviše dva naslednika. Dakle, binarno stablo je skup od jednog ili više čvorova takvih da:

1. Postoji jedan čvor koji je koren stabla i
2. Preostali čvorovi su podeljeni u dva disjunktna skupa, tako da svaki skup predstavlja binarno stablo. Ova dva skupa se nazivaju levo i desno podstablo.

Na Slici ### je prikazano binarno stablo:

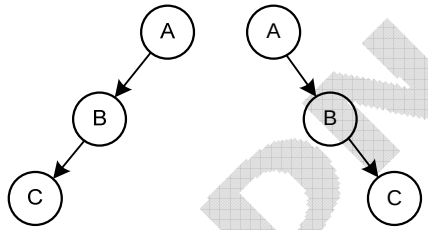


Slika ### Binarno stablo

Za binarno stablo važi:

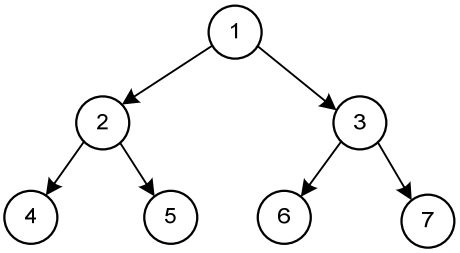
1. Maksimalni broj čvorova na *i*-tom nivou je 2^{i-1}
2. Ako je *k* dubina binarnog stabla, onda je maksimalan broj čvorova koje stablo može da ima je $2^k - 1 = 2^{k-1} + 2^{k-2} + ... + 2^0$

Napomenimo da mogu postojati i iskošena stabla, kao što je to prikazano na Slici ###:



Slika ### Iskošena binarna stabla

Puno binarno stablo je binarno stablo dubine *k* koje sadrži $2^k - 1$ čvorova. Ukoliko je broj čvorova manji od $2^k - 1$, binarno stablo je nepotpuno. Primer punog binarnog stabla je prikazan na Slici ###:

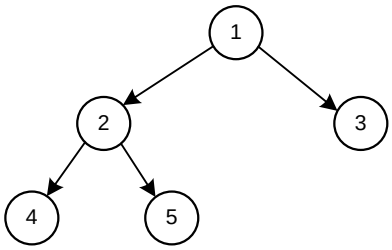


Slika ### Puno binarno stablo

Na prethodnoj slici smo čvorove stabla označili brojevima od 1 do 7. Ukoliko je binarno stablo puno njegovi čvorovi se mogu numerisati brojevima od 1 do $2^k - 1$ tako što bi se počelo od korena, a zatim numerisao jedan po jedan nivo sa leva na desno.

Kompletno binarno stablo dubine *k* je stablo sa *n* čvorova u kome bi se ovih *n* čvorova moglo numerisati brojevima od 1 do *n*, kao kada bi to bilo prvih *n* čvorova u punom stablu dubine *k*.

Na Slici ### je prikazano kompletno binarno stablo dubine *k*=3:

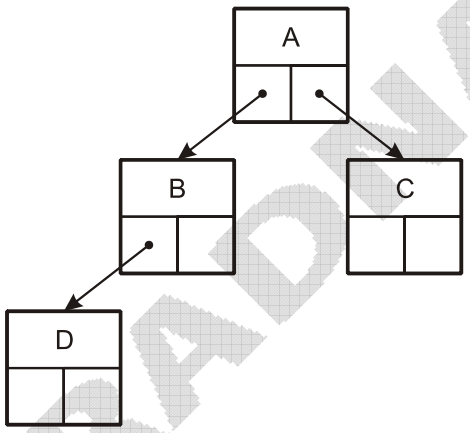


Slika ### Kompletno binarno stablo

Predstavljanje binarnog stabla

Binarno stablo može biti predstavljeno korišćenjem niza tako što bi vrednost u svakom čvoru bila pridružena elementu niza sa indeksom koji je jednak numeraciji tog čvora u odgovarajućem punom binarnom stablu. Međutim, s obzirom na ozbiljne nedostatke, nećemo detaljnije opisivati ovaj način predstavljanja binarnog stabla. Umesto toga, pokazaćemo kako je moguće predstaviti binarno stablo korišćenjem povezanih struktura.

Binarno stablo je moguće veoma efikasno predstaviti korišćenjem povezanih struktura, tako što bi svaki čvor bio predstavljen strukturom sa tri promenljive: podatak koji se čuva, pokazivač na levo podstablo i pokazivač na desno podstablo, kao što je prikazano na Slici ###:



Slika ### Predstavljanje binarnog stabla korišćenjem povezanih struktura

Ukoliko čvor nema nekog od naslednika, pokazivač na tog naslednika bi u tom slučaju bio NULL.

Deklaracija strukture za predstavljanje čvora binarnog stabla u programskom jeziku C bi mogla da izgleda ovako:

```
struct cvor
{
   int podatak;
   struct cvor *levi,*desni;
};
```

Obilazak binarnog stabla

Pod obilaskom binarnog stabla podrazumevamo prolazak kroz sve njegove čvorove određenim redosledom, pri čemu je istovremeno moguće izvršiti i određenu operaciju nad podacima koji su u njima zapisani.

U zavisnosti od redosleda obrade određenog čvora i njegovog levog i desnog podstabla, obilazak stabla se može izvršiti na šest načina:

Skraćenica	**Opis**
LPD	obilazak levog podstabla, obrada podatka, obilazak desnog podstabla
LDP	obilazak levog podstabla, obilazak desnog podstabla, obrada podatka
PLD	obrada podatka, obilazak levog podstabla, obilazak desnog podstabla
PDL	obrada podatka, obilazak desnog podstabla, obilazak levog podstabla
DLP	obilazak desnog podstabla, obilazak levog podstabla, obrada podatka
DPL	obilazak desnog podstabla, obrada podatka, obilazak levog podstabla

pri čemu

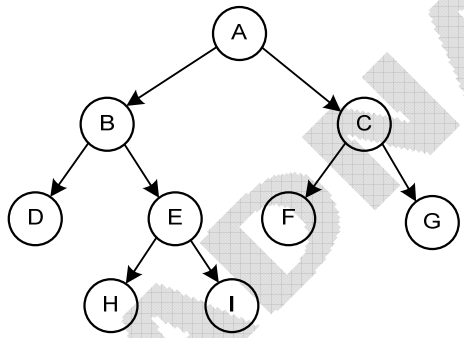
L – označava obilazak levog podstabla,

D – označava obilazak desnog podstabla,

P – označava obradu podatka u trenutnom čvoru.

Tako, na primer LPD označava redosled u kome krećemo od korena, obilazimo levo podstablo, obrađujemo podatak u korenu i na kraju obilazimo desno podstablo. S obzirom na to da su levo i desno podstablo takođe binarna stabla, potpuno ista procedura se koristi i za njihov obilazak.

Redosled LPD se naziva još i **inorder**, redosled LDP **postorder**, a redosled PLD **preorder**. Ostala tri redosleda se nikada ne koriste. Ukoliko prilikom obilaska stabla štampamo podatke koji se nalaze u čvorovima, u zavisnosti od toga da li koristimo inorder, preorder ili postorder, dobijamo rezultate kao što je prikazano na Slici ###:



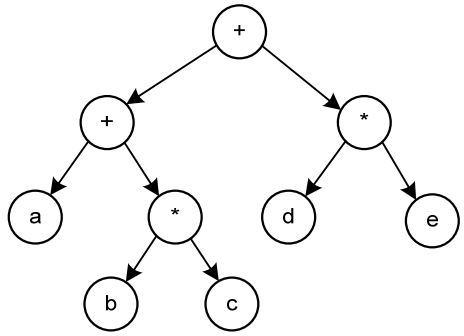
Inorder: D,B,H,E,I,A,F,C,G

Preorder: A,B,D,E,H,I,C,F,G

Postorder: D,H,I,E,B,F,G,C,A

Slika ### Redosled obilaska binarnog stabla

Tipičan primer korišćenja binarnog stabla je predstavljanje aritmetičkih izraza.



Inorder: a+b*c+d*e

Preorder: ++a*bc*de

Postorder: abc*+de*+

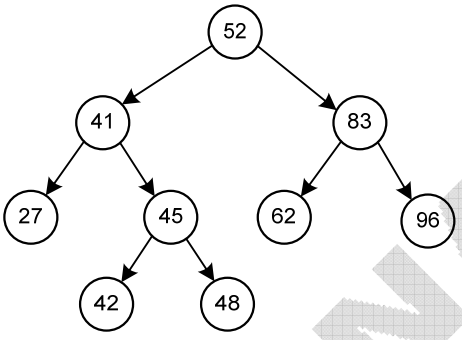
Slika ### Aritmetički izraz predstavljen binarnim stablom

Binarno stablo za pretraživanje

Binarno stablo za pretraživanje je binarno stablo koje ima sledeće karakteristike:

1. Čvorovima binarnog stabla su pridružene vrednosti po kojima se obavlja pretraživanje.
2. Za bilo koji čvor važi da svi čvorovi u levom podstablu imaju vrednost manju od njega, a svi čvorovi u desnom podstablu imaju vrednost veću od njega. I levo i desno stablo su takođe binarna stabla za pretraživanje.

Primer binarnog stabla za pretraživanje je prikazan na Slici ###



Slika ### Binarno stablo za pretraživanje

Binarno stablo za pretraživanje je u osnovi binarno stablo, tako da se može obići u inorder, preorder ili postorder redosledu. Ukoliko binarno stablo za pretraživanje obiđemo u inorder redosledu i pri tome odštampamo vrednosti u čvorovima, dobićemo vrednosti sortirane u rastućem redosledu.

Binarno stablo za pretraživanje je veoma važna struktura podataka. Posmatrajmo, na primer, problem pretraživanja niza. Ukoliko je niz sortiran, pretraživanje možemo znatno ubrzati korišćenjem binarnog pretraživanja. Međutim, ukoliko želimo da izvršimo promene u nizu kao što su dodavanje i brisanje elementa, to će zahtevati pomeranje određenog broja elemenata svaki put. Ovaj problem možemo rešiti korišćenjem povezanih lista, zato što one omogućavaju dodavanje i brisanje elemenata vrlo jednostavnim premeštanjem nekoliko pokazivača. Međutim, problem sa povezanim listama je to što one ne omogućavaju direktan pristup pojedinim elementima, već samo sekvencijalno kretanje kroz listu, element po element. Rešenje ovih problema leži upravo u binarnim stablima. Ukoliko se elementi uređenog niza ili liste smeste u čvorove binarnog stabla za pretraživanje, nalaženje odgovarajućeg ključa u binarnom stablu zahteva $O(\log n)$ koraka.

Kreiranje binarnog stabla za pretraživanje

Pretpostavimo da svaki čvor binarnog stabla sadrži celobrojni podatak, kao i pokazivače na levo i desno podstablo. U tom slučaju struktura koja opisuje čvor binarnog stabla bi mogla da izgleda ovako:

```
struct cvor
{
   int podatak;
   struct cvor *levi,*desni;
};
```

U nastavku je dat kompletan program za kreiranje binarnog stabla za pretraživanje:

```
#include <stdio.h>
#include <stdlib.h>

struct cvor
{
   int podatak;
   struct cvor *levi, *desni;
};

/* funkcija za dodavanje novog cvora u binarno stablo za pretrazivanje */
struct cvor *dodaj(struct cvor *p,int vrednost)
{
   struct cvor *pom1,*pom2,*novi;

   /* kreiranje novog cvora */
   novi = (struct cvor *) malloc(sizeof(struct cvor));
   if(novi == NULL)
   {
      printf("Greska pri alociranju memorije.\n");
      exit(0);
   }
   novi->podatak = vrednost;
   novi->levi = novi->desni = NULL;

   if(p == NULL)
   {
      p = novi;
   }
   else
   {
      /* prolazak kroz stablo kako bi se pronasao cvor cije ce dete biti novi cvor */
      pom1 = p;
      while(pom1 != NULL)
      {
         pom2 = pom1;
         if( vrednost < pom1->podatak )
            pom1 = pom1->levi;
         else
            pom1 = pom1->desni;
      }

      if( vrednost < pom2->podatak )
         pom2->levi = novi;     /* dodavanje novog cvora kao levog deteta */
      else
         pom2->desni = novi;    /* dodavanje novog cvora kao desnog deteta */
   }

   return(p);
}

/* funkcija za stampanje binarnog stabla u inorder redosledu */
void stampaj_inorder(struct cvor *p)
{
   if(p != NULL)
   {
      stampaj_inorder(p->levi);
      printf("%d\t",p->podatak);
      stampaj_inorder(p->desni);
   }
}

void main()
{
   struct cvor *koren = NULL;
   int i,n,x;
```

```
   printf("Unesite broj cvorova:\n");
   scanf("%d",&n);

   for(i=0; i<n; i++)
   {
      printf("Unesite vrednost:\n");
      scanf("%d",&x);
      koren = dodaj(koren,x);
   }

   stampaj_inorder(koren);
}
```

Za kreiranje binarnog stabla za pretraživanje korišćena ju funkcija *dodaj*, koja kreira kreira novi čvor, dodeljuje mu prosleđenu vrednost i dodaje ga u postojeće stablo. Prva provera koju funkcija obavlja jeste provera da li je stablo prazno. Ukoliko jeste, novi čvor se dodaje kao koren stabla. Ako stablo nije prazno otpočinje se sa kretanjem kroz stablo korišćenjem pomoćnog pokazivača *pom1*. Podatak u trenutnom čvoru upoređuje se sa novom vrednošću. U slučaju da je nova vrednost manja od vrednosti u trenutnom čvoru, kretanje se nastavlja kroz levo podstablo. U suprotnom kretanje se nastavlja kroz desno podstablo. Kretanje se vrši sve do trenutka dok se ne dođe do kraja stabla, odnosno dok pokazivač na odgovarajuće dete nije NULL. Sve vreme kretanja čuva se pokazivač na roditeljski čvor *pom2*, kako bi se na njega na kraju dodao novi cvor kao dete.

Kada se dospe do kraja stabla, novi čvor se dodaje kao levo ili desno dete poslednjem čvoru, u zavisnosti od toga da li je nova vrednost manja ili veća od vrednosti poslednjeg čvora.

U glavnom programu zadaje se broj čvorova, kao i vrednosti u njima. Unete vrednosti se dodaju u drvo korišćenjem prethodno opisane funkcije *dodaj*. Na kraju se vrši štampanje čvorova stabla korišćenjem funkcije *stampaj_inorder*, koja štampanje obavlja tokom obilaska stabla u inorder redosledu.

Pretraživanje u binarnom stablu za pretraživanje

Pretraživanje u binarnom stablu predstavlja proces u kome se traži čvor u stablu, koji zadovoljava unapred definisane kriterijume. Najčešće je to pretraživanje u cilju nalaženja čvora koji sadrži određeni podatak koji nazivamo **ključ**.

Da bismo pronašli određeni ključ u datom binarnom stablu za pretraživanje, krećemo od korena stabla i upoređujemo ključ sa podatkom u čvoru. Ukoliko su ključ i podatak jednaki, funkcija vraća pokazivač na taj čvor. Ako je ključ manji od vrednosti u čvoru, isti proces ponavljamo za levo podstablo. U suprotnom, proces ponavljamo za desno podstablo. Pretraga se prekida onog trenutka kada se pronađe traženi čvor ili kada je podstablo koje posmatramo prazno. Ukoliko je posmatrano podstablo prazno, funkcija vraća NULL, kao znak da traženi čvor nije pronađen.

U nastavku je data funkcija za pretraživanje u binarnom stablu za pretraživanje:

```
struct cvor
{
   int podatak;
   struct cvor *levi, *desni;
};

      ...

/* Funkcija za pretrazivanje u binarnom stablu za pretrazivanje */
struct cvor *trazi( struct cvor *p,int kljuc)
{
   struct cvor *pom;

   pom = p;
   while( pom != NULL)
   {
      if(pom->podatak == kljuc) return(pom);
      else
         if(pom->podatak > kljuc)
            pom = pom->levi;
         else
```

```
            pom = pom->desni;
   }

   return(NULL);
}
```

Određivanje broja čvorova u binarnom stablu

Broj čvorova bilo kog stabla koje nije list jednak je zbiru broja čvorova u njegovom levom i desnom podstablu, koji se uvećava za jedan kako bi se uračunao i koren stabla. Ukoliko je stablo list (nema potomaka), onda je broj čvorova u njemu jednak jedan. Ovaj opis ukazuje na potrebu rekurzivnog prolaska kroz stablo sve dok se ne dođe do njegovih listova.

Naredna rekurzivna funkcija vrši prebrojavanje čvorova u binarnom stablu.

```
struct cvor
{
   int podatak;
   struct cvor *levi, *desni;
};

      ...

int broj(struct cvor *p)
{
   if( p == NULL)  return(0);
   else
      if( p->levi == NULL && p->desni == NULL)
         return(1);
      else
         return( 1 + ( broj(p->levi) + broj(p->desni) ) );
}
```

Zamena mesta levim i desnim podstablima

Elegantan metod za zamenu mesta levim i desnim podstablima unutar binarnog stabla je korišćenje rekurzivnog algoritma. Funkcija zamenjuje mesta levom i desnom podstablu, a zatim na njima rekurzivno primenjuje isti postupak.

```
struct cvor
{
   int podatak;
   struct cvor *levi, *desni;
};

      ...

void zameni(struct cvor *p)
{
   struct cvor *pom;

   if( p != NULL)
   {
      pom = p->levi;
      p->levi = p->desni;
      p->desni = pom;

      zameni(p->levi);
      zameni(p->desni);
   }
}
```

Primenom ovog algoritma na binarno stablo za pretraživanje, koje bi pri inorder štampanju dalo rastući niz, dobilo bi se binarno stablo koje bi pri inorder štampanju dalo opadajući niz.

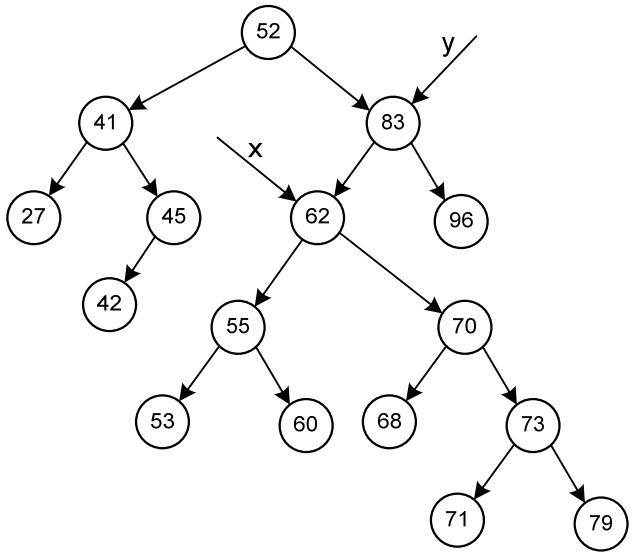
Brisanje čvora iz binarnog stabla za pretraživanje

Označimo pokazivač na čvor koji želimo da obrišemo sa *x* (skraćeno: čvor *x*), a pokazivač na čvor koji je njegov roditelj sa *y*. Nakon brisanja čvora *x*, potrebno je izvršiti reorganizaciju njegovih potomaka kako bi oni i dalje formirali binarno stablo za pretraživanje, a zatim novoformirano podstablo povezati na

odgovarajući način sa čvorom *y*, u zavisnosti od toga da li je čvor *x* bio levi ili desni potomak čvora *y*. U slučaju da je čvor *x* bio koren stabla, onda podstablo formirano od njegovih potomaka postaje novi koren. Označimo pokazivač na podstablo formirano od potomaka čvora *x* sa *q*.

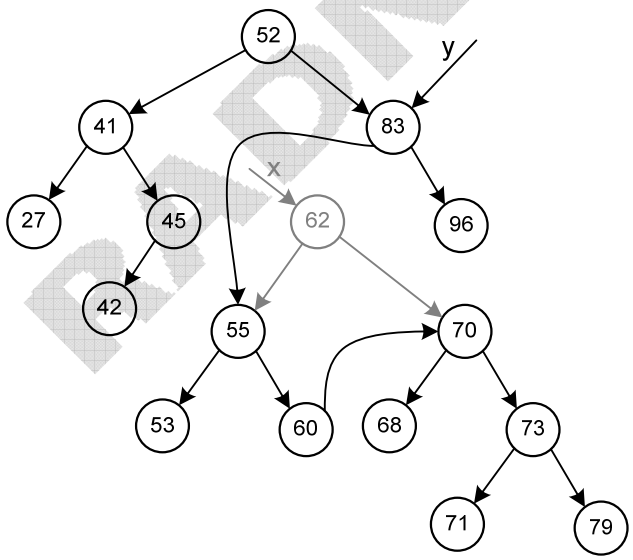
Brisanje čvora koji ima dva potomka

Posmatrajmo binarno stablo prikazano na Slici ###:



Slika ### Binarno stablo pre brisanja čvora na koji pokazuje *x*

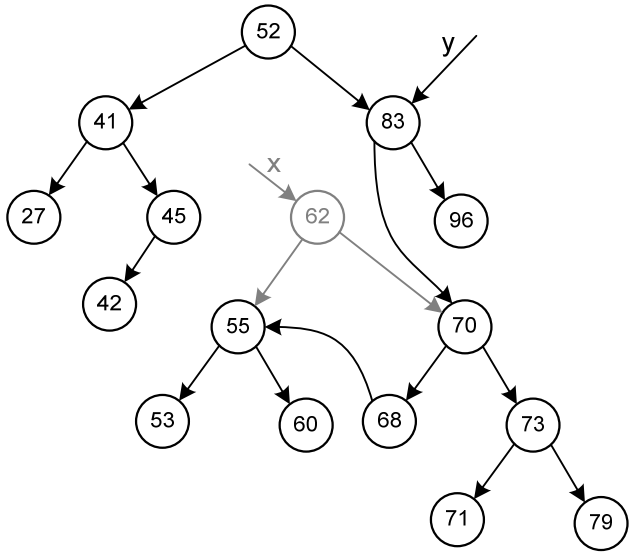
Korišćenjem pomoćnog pokazivača *pom,* prolazimo kroz levo podstablo čvora *x*, kako bismo pronašli njegov desni list, a zatim postavljamo desni pokazivač ovog lista da pokazuje na desno podstablo čvora *x*, kao što je pokazano na Slici ###. Nakon toga, pokazivač *q* se postavlja da pokazuje na levo podstablo čvora *x*, a čvor *x* se briše iz memorije. Ukoliko pokazivač *y* nije NULL, ostaje samo da se pokazivač *q* postavi za odgovarajućeg potomka čvora *y*. Ako pokazivač na čvor *y* jeste NULL (obrisan je koren), onda novi koren postaje pokazivač *q*.



Slika ### Binarno stablo nakon brisanja čvora na koji pokazuje *x*

Drugi način brisanja čvora je da se prolazi kroz desno podstablo čvora *x* kako bi se pronašao njegov levi list, a zatim se levi pokazivač tog lista postavlja da pokazuje na levo podstablo čvora *x*, kao što je prikazano na

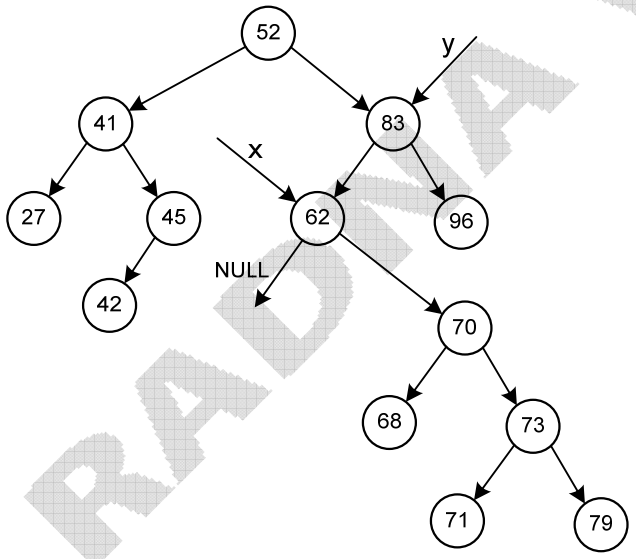
Slici ###. Naravno, na kraju je potrebno obrisati čvor *x* iz memorije. Nakon toga, pokazivač *q* se postavlja da pokazuje na desno podstablo čvora *x*, a čvor *x* se briše iz memorije. Ukoliko pokazivač *y* nije NULL, ostaje samo da se pokazivač *q* postavi za odgovarajućeg potomka čvora *y*. Ako pokazivač na čvor *y* jeste NULL (obrisan je koren), onda novi koren postaje pokazivač *q*.



Slika ### Binarno stablo nakon brisanja čvora na koji pokazuje *x*

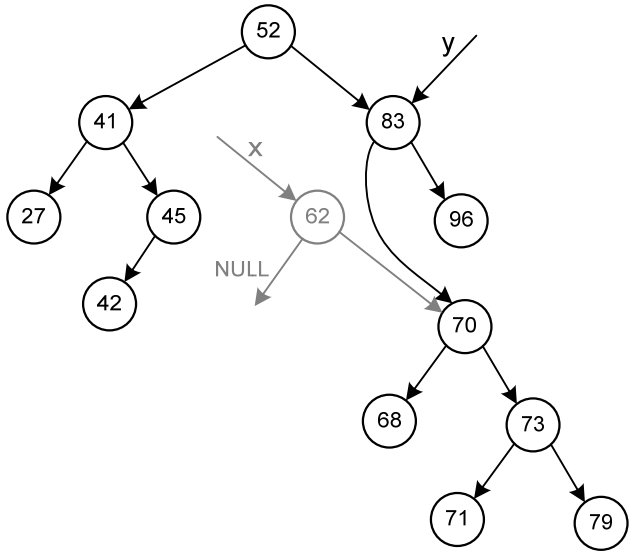
Brisanje čvora koji ima jednog potomka

Posmatrajmo binarno stablo prikazano na Slici ###:



Slika ### Binarno stablo pre brisanja čvora na koji pokazuje *x*

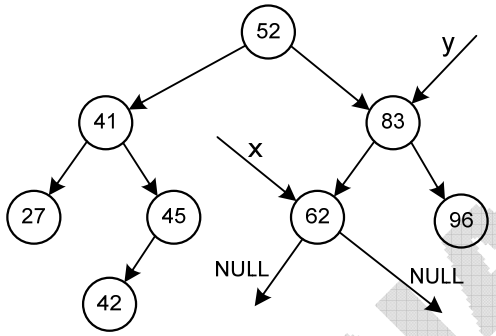
Ukoliko želimo da obrišemo čvor *x*, dovoljno je da postavimo pokazivač *q* da pokazuje na podstablo čvora *x* koje postoji, a zatim iz memorije obrišemo čvor *x*, kao što je prikazano na Slici ###. Ukoliko pokazivač *y* nije NULL, ostaje samo da se pokazivač *q* postavi za odgovarajućeg potomka čvora *y*. Ako pokazivač na čvor *y* jeste NULL (obrisan je koren), onda novi koren postaje pokazivač *q*.



Slika ### Binarno stablo nakon brisanja čvora na koji pokazuje *x*

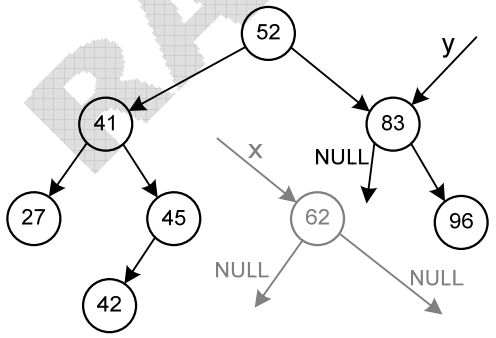
Brisanje čvora koji nema ni jednog potomka

Posmatrajmo binarno stablo na Slici ###:



Slika ### Binarno stablo pre brisanja čvora na koji pokazuje *x*

Da bismo obrisali čvor *x* iz binarnog stabla, potrebno je samo da pokazivaču *q* dodelimo NULL, a zatim, ukoliko pokazivač *y* nije NULL, ostaje samo da se pokazivač *q* postavi za odgovarajućeg potomka čvora *y*. Ako pokazivač na čvor *y* jeste NULL (obrisan je koren), onda novi koren postaje pokazivač *q*.



Slika ### Binarno stablo nakon brisanja čvora na koji pokazuje *x*

Primer

```
#include <stdio.h>
#include <stdlib.h>

struct cvor
{
   int podatak;
   struct cvor *levi, *desni;
};

/* Funkcija koja vraca pokazivac na cvor sa trazenom vrednoscu i pokazivac na njegovog roditelja */
struct cvor *pronadji(struct cvor *p, int kljuc, struct cvor **y)
{
   struct cvor *pom;

   if( p == NULL)  return(NULL);

   pom = p;
   *y = NULL;
   while( pom != NULL)
   {
      if(pom->data == kljuc)    return(pom);
      else
      {
         *y = pom; /* pamcenje roditelja */
         if( kljuc < pom->data )
            pom = pom->levi;
         else
            pom = pom->desni;
      }
   }

   return(NULL);
}

/* Funkcija za brisanje cvora koji sadrzi zadatu vrednost */
struct cvor *obrisi(struct cvor *p,int vrednost)
{
   struct cvor *x, *y, *q, *pom,;

   x = pronadji(p,vrednost,&y);

   if( x == NULL)
   {
      printf("Cvor ne postoji.\n");
      return(p);
   }

   if( x->levi != NULL && x->desni != NULL) /* brisanje cvora koji ima oba potomka */
   {
      pom = x->levi;
      while(pom->desni != NULL) pom = pom->desni;
      pom->desni=x->desni;
      q = x->levi;
   }
   else if(x->levi == NULL && x->desni !== NULL) /* brisanje cvora koji ima samo desnog potomka */
   {
      q = x->desni;
   }
   else if( x->levi != NULL && x->desni == NULL) /* brisanje cvora koji ima samo levog potomka */
   {
      q = x->levi;
   }
   else if(x->levi == NULL && x->desni == NULL) /* brisanje cvora koji nema ni jednog potomka */
   {
      q = NULL;
   }

   if(y!=NULL)
   {
      if(y->levi == x)
         y->levi = q;
      else
```

```
            y->desni = q;
    }
    else   p = q;

    free(x);
    return(p);
}

/* funkcija za dodavanje novog cvora u binarno stablo za pretrazivanje */
struct cvor *dodaj(struct cvor *p,int vrednost)
{
    struct cvor *pom1,*pom2,*novi;

    /* kreiranje novog cvora */
    novi = (struct cvor *) malloc(sizeof(struct cvor));
    if(novi == NULL)
    {
       printf("Greska pri alociranju memorije.\n");
       exit(0);
    }
    novi->podatak = vrednost;
    novi->levi = novi->desni = NULL;

    if(p == NULL)
    {
       p = novi;
    }
    else
    {
       /* prolazak kroz stablo kako bi se pronasao cvor cije ce dete biti novi cvor */
       pom1 = p;
       while(pom1 != NULL)
       {
          pom2 = pom1;
          if( vrednost < pom1->podatak )
             pom1 = pom1->levi;
          else
             pom1 = pom1->desni;
       }

       if( vrednost < pom2->podatak )
          pom2->levi = novi;      /* dodavanje novog cvora kao levog deteta */
       else
          pom2->desni = novi;     /* dodavanje novog cvora kao desnog deteta */
    }

    return(p);
}

/* funkcija za stampanje binarnog stabla u inorder redosledu */
void stampaj_inorder(struct cvor *p)
{
    if(p != NULL)
    {
       stampaj_inorder(p->levi);
       printf("%d\t",p->podatak);
       stampaj_inorder(p->desni);
    }
}

void main()
{
    struct cvor *koren = NULL;
    int n,i,x;

    printf("Unesite broj cvorova stabla:\n");
    scanf("%d",&n);

    for(i=0; i<n; i++)
    {
       printf("Unesite vrednost:\n");
       scanf("%d",&x);
       koren = dodaj(koren,x);
    }
```

```
    printf("Stablo pre brisanja je:\n");
    stampaj_inorder(koren);
    printf("\n");

    printf("Unesite vrednost koju treba obrisati:\n");
    scanf("%d",&x);

    koren=obrisi(koren,n);

    printf("Stablo posle brisanja je:\n");
    stampaj_inorder(koren);
}
```