Osnovi programiranja

Beleške sa vežbi

Smer *Računarstvo i informatika*
Matematički fakultet, Beograd

Jelena Tomašević

February 27, 2006

Sadržaj

1

[1]

1.1 Sortiranje

Niz može biti sortiran ili uređjen u opadajućem, rastućem, neopadajućem i nerastućem poretku. Dato je nekoliko algoritama za sortiranje niza koji se unosi sa ulaza u nerastućem poretku odnosno tako da važi da je `niz[0] >= niz[1] >= ... niz[n]`. Jednostavnom modifikacijom svakim od ovih algoritama niz se može sortirati i u opadajućem, rastućem ili neopadajućem poretku.

Primer 1 *Selection sort*
U prvom prolazu se razmenjuju vrednosti a[0] sa onim članovima ostatka niza koji su veći od njega. Na taj način će se posle prvog prolaza kroz niz a[0] postaviti na najveći element niza.

```
#include<stdio.h>
#define MAXDUZ 100

int main()
{
   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

   /* Dimenzija niza, pomocna i brojacke promenljive */
   int n,pom,i,j;

   printf("Unsite dimenziju niza\n");
   scanf("%d",&n);

   if (n>MAXDUZ)
   {
        printf("Nedozvoljena vrednost za n\n");
        exit(1);
   }

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
```

[1] Zasnovano na primerima sa sajta `http://www.matf.bg.ac.yu/~milena`

```
   }

   /*Sortiranje*/
   for(i=0; i<n-1; i++)
      for(j=i+1; j<n; j++)
         if(a[i]<a[j])
         {
            pom=a[i];
            a[i]=a[j];
            a[j]=pom;
         }

   /* Ispis niza */
   printf("Sortirani niz:\n");
   for(i=0; i<n; i++)
      printf("%d\t",a[i]);

   putchar('\n');

   return 0;

}
```

Primer 2 *Selection sort 2*
Modifikacija prethodnog rešenja radi dobijanja na efikasnosti. Ne vrše se zamene svaki put već samo jednom, kada se pronađe odgovarajući element u nizu sa kojim treba izvršiti zamenu tako da u nizu bude postavljen trenutno najveći element na odgovarajuće mesto.

```
#include<stdio.h>
#define MAXDUZ 100

int main()
{

   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

     /* Dimenzija niza, indeks najveceg elementa
   u i-tom prolazu,pomocna i brojacke promenljive */
   int n,ind,pom,i,j;

   printf("Unsite dimenziju niza\n");
   scanf("%d",&n);

   if (n>MAXDUZ)
   {
        printf("Nedozvoljena vrednost za n\n");
        exit(1);
   }

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
```

```
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

   /*Sortiranje - bez stalnih zamena vec se
   pronalazi indeks trenutno najveceg clana niza*/
   for(i=0; i<n-1; i++)
   {
       for(ind=i,j=i+1; j<n; j++)
          if(a[ind]<a[j])
             ind=j;

       /* Vrsi se zamena onda kada na i-tom mestu
        nije najveci element. Tada se na i-to mesto
        postavlja najveci element koji se nalazio na
        mestu ind. */
       if(i != ind)
       {
          pom=a[ind];
          a[ind]=a[i];
          a[i]=pom;
       }
   }
   /* Ispis niza */
   printf("Sortirani niz:\n");
   for(i=0; i<n; i++)
      printf("%d\t",a[i]);

   return 0;

}
```

Primer 3 *bbsort1*
Algoritam sortiranja buble sort poredi dva susedna elementa niza i ako su pogrešno raspoređeni zamenjuje im mesta. Posle poređenja svih susednih parova najmanji od njih će isplivati na kraj niza. Zbog toga se ovaj metod naziva metod mehurića. Da bi se najmanji broj nesortiranog dela niza doveo na svoje mesto treba ponoviti postupak.

```
#include<stdio.h>
#define MAXDUZ 100

int main()
{
   /* Dimenzija niza, pomocna promenljiva
   i brojacke promenljive */
   int n,pom,i,j;

   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

   printf("Unsite dimenziju niza\n");
   scanf("%d",&n);
```

```
   if (n>MAXDUZ)
   {
        printf("Nedozvoljena vrednost za n\n");
        exit(1);
   }

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

   /*Sortiranje */
   for(i=n-1; i>0; i--)
      for(j=0; j<i; j++)
         if(a[j]<a[j+1])
         {
            pom=a[j];
            a[j]=a[j+1];
            a[j+1]=pom;
         }

   /* Ispis niza */
   printf("Sortirani niz:\n");
   for(i=0; i<n; i++)
      printf("%d\t",a[i]);

   /* Stampa prazan red */
   putchar('\n');

   /*Regularan zavrsetak rada programa */
   return 0;

}
```

Primer 4 *bbsort2*
Unapredjujemo prethodni algoritam kako bismo obezbedli da se ne vrse provere onda kada je niz već sortiran nego da se u tom slučaju prekine rad.

```
#include<stdio.h>
#define MAXDUZ 100

int main()
{
   /* Dimenzija niza, pomocna promenljiva
    i brojacke promenljive */
   int n,pom,i,j;

   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

   /* Promenljiva koja govori da li je izvrsena
```

```
   zamena u i-tom prolazu kroz niz pa ako nije
   sortiranje je zavrseno jer su svaka dva
   susedna elementa niza u odgovarajucem poretku */
   int zam;

   printf("Unsite dimenziju niza\n");
   scanf("%d",&n);

   if (n>MAXDUZ)
   {
        printf("Nedozvoljena vrednost za n\n");
        exit(1);
   }

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

   /*Sortiranje */
   for(zam=1,i=n-1; zam && i>0; i--)
      for(zam=0,j=0; j<i; j++)
         if(a[j]<a[j+1])
         {
            /* Zamena odgovarajucih clanova niza */
            pom=a[j];
            a[j]=a[j+1];
            a[j+1]=pom;

            /* Posto je u i-tom prolazu
            izvrsena bar ova zamena zam
            se postavlja na 1 sto
            nastavlja sortiranje */
            zam=1;
         }

   /* Ispis niza */
   printf("Sortirani niz:\n");
   for(i=0; i<n; i++)
      printf("%d\t",a[i]);

   return 0;

}
```

Primer 5 *isort*
Insert sort, u svakom trenutku je početak niza sortiran a sortiranje se vrši tako što se jedan po jedan element niza sa kraja ubacuje na odgovarajuće mesto.

```
#include<stdio.h>
#define MAXDUZ 100
```

```
int main()
{
   /* Dimenzija niza, pomocna
   i brojacke promenljive */
   int n,pom,i,j;

   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

   printf("Unsite dimenziju niza\n");
   scanf("%d",&n);

   if (n>MAXDUZ)
   {
        printf("Nedozvoljena vrednost za n!\n");
        exit(1);
   }

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

   /*Sortiranje*/
   for(i=1; i<n; i++)
      for(j=i; (j>0) && (a[j]>a[j-1]); j--)
      {
         pom=a[j];
         a[j]=a[j-1];
         a[j-1]=pom;
      }

   /* Ispis niza */
   printf("Sortirani niz:\n");
   for(i=0; i<n; i++)
      printf("%d\t",a[i]);

   putchar('\n');

   return 0;

}
```

Primer 6 *Binarno pretrazivanje*

```
#include<stdio.h>
#define MAXDUZ 100

int main()
{
```

```
/* Dimenzija niza,pomocna i brojacke
   promenljive */
int n,pom,i,j;

/* Niz od maksimalno MAXDUZ elemenata*/
int a[MAXDUZ];

/* Elemet koji se trazi i pozicija
na kojoj se nalazi- ukoliko je u nizu*/
int x,pozicija;

/* Pomocne promenljive za pretragu */
int donji, gornji, srednji;

printf("Unsite dimenziju niza\n");
scanf("%d",&n);

/* Unos clanova niza */
for(i=0; i<n; i++)
{
   printf("Unesite %d. clan niza\n",i+1);
   scanf("%d",&a[i]);
}

/*Sortiranje*/
for(i=0; i<n-1; i++)
   for(j=i+1; j<n; j++)
      if(a[i]>a[j])
      {
         pom=a[i];
         a[i]=a[j];
         a[j]=pom;
      }
/* Unos elementa binarne pretrage */
printf("Unesite element koji se trazi\n");
scanf("%d",&x);

donji = 0;
gornji = n-1;
pozicija = -1;

while(donji<=gornji)
{
   srednji = (donji + gornji)/2;
   if(a[srednji] == x)
   {
      pozicija = srednji;
      break;
   }
   else
      if(a[srednji] < x)
         donji = srednji + 1;
```

```
         else
            gornji = srednji -1;
   }


   /* Ispis rezultata */
   if(pozicija == -1)
      printf("Trazeni broj se ne nalazi u nizu!\n");
   else
      printf("Broj %d se nalazi na %d poziciji
       sortiranog niza! \n",x,pozicija+1);

   putchar('\n');

   return 0;
}
```

Primer 7 *Sabiranje dva velika broja, njihovo poređenje, unos i ispis, množenje velikog broja cifrom.*

```
#include<stdio.h>
#define MAXDUZ 1000

int unos_broja(int cifre[], int maxduz)
   {
     int brcifara=0;
     char c;

     c=getchar();
     while ( brcifara < maxduz && c >= '0' && c <= '9')
     {
        cifre[brcifara++]=c-'0';
        c=getchar();
     }

     return brcifara;
   }

void obrni(int cifre[],int brcifara)
{
   int i,pom;

   for (i=0; i<brcifara/2; i++)
   {
      pom=cifre[i];
      cifre[i]=cifre[brcifara-i-1];
      cifre[brcifara-i-1]=pom;
   }
}

void ispisi(int cifre[],int brcifara)
{  int i;
```

```
   putchar('\n');
   for (i=brcifara-1; i>=0; i--)
      printf("%d",cifre[i]);
      /* ili
      putchar(cifre[i]+'0');
      */
    putchar('\n');
}

int jednaki(int cifre1[],int cifre2[],
            int brcifara1, int brcifara2)
{
   int i;
   if (brcifara1 != brcifara2) return 0;

   for (i=0; i<brcifara1; i++)
     if (cifre1[i] != cifre2[i]) return 0;
   return 1;
}


int veci(int cifre1[], int brcifara1,
        int cifre2[], int brcifara2)
{
   int i;
   if (brcifara1>brcifara2) return 1;
   if (brcifara1<brcifara2) return 0;

   for (i=brcifara1-1; i>=0; i--)
   {
      if (cifre1[i]<cifre2[i]) return 0;
      if (cifre1[i]>cifre2[i]) return 1;
   }

   return 0;
}


int saberi(int cifre1[], int brcifara1,
           int cifre2[], int brcifara2,
           int cifre[])
{
    int brcifara=0;
    int i,pom,pamtim=0;

    for(i=0; i<brcifara1 || i<brcifara2; i++)
    {
        pom =((i < brcifara1)? cifre1[i] : 0 )
            +((i < brcifara2)? cifre2[i] : 0)
            + pamtim;

        cifre[i] = pom%10;
```

```
        pamtim = pom/10;
    }
    if (pamtim)
    {
       cifre[i]=pamtim;
       brcifara=i+1;
    }
    else brcifara=i;

    return brcifara;
}

int pomnozic(int c,int cifre[],
             int brcifara, int pcifre[])
{
   int pbrcifara=0;
   int i,pamtim=0;
   for (i=0; i<brcifara; i++)
   {
      pcifre[i]=(cifre[i]*c+pamtim)%10;
      pamtim=(cifre[i]*c+pamtim)/10;
   }
   pbrcifara=brcifara;
   if (pamtim)
   {
   pcifre[pbrcifara]=pamtim;
   pbrcifara++;
   }

   return pbrcifara;
}


int main()
{
int d1,d2,d;
int broj1[MAXDUZ], broj2[MAXDUZ], zbir[MAXDUZ];
d1=unos_broja(broj1,MAXDUZ);
d2=unos_broja(broj2,MAXDUZ);

obrni(broj1,d1);
obrni(broj2,d2);
d=saberi(broj1,d1,broj2,d2,zbir);
ispisi(zbir,d);
return 0;
}
```