Programiranje 2

Beleške sa vežbi

Školska 2006/2007 godina

Matematički fakultet, Beograd

Jelena Tomašević

March 26, 2007

Sadržaj

1

Programski jezik C

[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++)
   {
```

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

```
      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;
         }

   /* 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;

}
```

1.2 Linearna i binarna pretraga niza

Primer 6 *Linearno pretraživanje*

```
#include <stdio.h>
/* Funkcija proverava da li se dati element x nalazi
u datom nizu celih brojeva.
Funkcija vraca poziciju u nizu na
kojoj je x pronadjen
odnosno -1 ukoliko elementa nema.
*/
int linearna_pretraga(int niz[], int br_elem, int x)
{
    int i;
    for (i = 0; i<br_elem; i++)
        if (niz[i] == x)
            return i;
    /* nikako else */
    return -1;
}

main()
{
    /* Inicijalizacija niza moguca je
    na ovaj nacin*/
    int a[] = {4, 3, 2, 6, 7, 9, 11};
    /* Da bi smo odredili koliko clanova
    ima niz mozemo koristiti operator
    sizeof*/
    int br_elem = sizeof(a)/sizeof(int);
    int x;
    int i;
    printf("Unesite broj koji trazimo : ");
    scanf("%d",&x);
    i = linearna_pretraga(a, br_elem, x);
    if (i == -1)
        printf("Element %d nije nadjen\n",x);
    else
        printf("Element %d je nadjen na poziciji %d\n",x, i);
}
```

Primer 7 *Binarna pretraga niza*

```
/* Binarna pretraga niza celih brojeva - iterativna verzija*/

#include <stdio.h>

/* Funkcija proverava da li se element x javlja unutar niza
celih brojeva a.
Funkcija vraca poziciju na kojoj je element nadjen odnosno
-1 ako ga nema.
!!!!! VAZNO !!!!!
Pretpostavka je da je niz a uredjen po velicini
*/
int binarna_pretraga(int a[], int n, int x)
```

```
{
    /* Pretrazujemo interval [l, d] */
    int l = 0;
    int d = n-1;
    /* Sve dok interval [l, d] nije prazan */
    while (l <= d)
    {
        /* Srednja pozicija intervala [l, d] */
        int s = (l+d)/2;
        /* Ispitujemo odnos x i a[srednjeg elementa] */
        if (x == a[s])
            /* Element je pronadjen */
            return s;
        else if (x < a[s])
                {
                 /* Pretrazujemo interval [l, s-1] */
                 d = s-1;
                }
            else
            {
                /* Pretrazujemo interval [s+1, d] */
                l = s+1;
            }
    }
    /* Element nije nadjen */
    return -1;
}

main()
{
    int a[] = {3, 5, 7, 9, 11, 13, 15};
    int x;
    int i;
    printf("Unesi element koji trazimo : ");
    scanf("%d",&x);
    i = binarna_pretraga(a, sizeof(a)/sizeof(int), x);
    if (i==-1)
        printf("Elementa %d nema\n", x);
    else
        printf("Pronadjen na poziciji %d\n", i);
}
```

Zadaci za vežbu:(Zasnovano na primerima sa sajta `http://www.matf.bg.ac.yu/~milan`)

Zadatak 1 *a) Napisati funkcije za sortiranje niza u rastućem poretku za razne tipove elemenata (float,char, long) i raznim metodama (select, insert, bubble). Na primer, prototip funkcije za sortiranje niza celih brojeva metodom select sorta može biti:*

`void int_select_sort(int a[], int n)`

Objediniti sve te funkcije sortiranja u jedan fajl "sort.c" (bez main funkcije), a zatim kreirati zaglavlje "sort.h" koje će sadržati deklaracije (prototipove) svih napisanih funkcija za sortiranje.

b) Napisati funkcije koje generišu niz slučajnih brojeva različitih tipova (float,char, long) dužine n, iz opsega [l, d].

Prototip funkcije za generisanje niza od n celih brojeva iz opsega [l,d] može biti:

```
void random_int_niz(int a[], int n, int l, int d)
```

Napomena: Broj celobrojnog tipa (npr. tipa long) iz opsega [l, d] se može generisati sledećim izrazom:

```
a= l + (long)rand() % (d-l);
```

Slično, broj realnog tipa (npr. float) iz opsega [l, d] se može generisati sledećim izrazom:

```
a= l + ((float)rand()/RAND_MAX) *(d-l);
```

Inicijalizacija početnih vrednosti (semena) algoritma za generisanje slučajnih brojeva ostvaruje se pozivom funkcije:

```
srand(time(NULL));
```

Objedniti sve te u jedan fajl "random.c", a zatim kreirati zaglavlje random.h koje će sadržati deklaracije (prototipove) svih napisanih funkcija za generisanje slučajnih brojeva.

c) U fajlu "main.c" napisati main() funkciju, uz uključenje zaglavlja "sort.h" i "random.h" direktivom `#include`*. U okviru funkcije main(), generisati niz od 15 slučajnih brojeva u intervalu od -10000 do 10000, sortirati nekom od funkcija i ispisati tako dobijeni sortirani niz.*

d) Odvojeno prevesti svaki fajl sa izvornim kodom. Na kraju sve dobijene objektne fajlove povezati u jedan izvršni fajl sort.

Uputstvo: Program prevesti naredbama:

```
gcc -c -o sort.o sort.c
gcc -c -o random.o random.c
gcc -c -o main.o main.c
gcc -o sort main.o sort.o random.o
```

e) Proširiti funkciju main() iz prethodnog primera tako da omogućava korisniku izbor da li želi da ručno unese brojeve za sortiranje, ili želi da se niz date dužine generiše na slučajan način. Ovo se na primer može uraditi argumentima komandne linije: npr. ako se program pozove sa:

```
./sort -r 10
```

tada ce se na slučajan način generisati niz od deset brojeva, dok ako se navede:

```
./sort -i
```

tada se od korisnika očekuje da brojeve unosi na standardnom ulazu.

Implementirati i opciju -c:

```
./sort -c 2 -1 5 67 2
```

kojom se specificira da se niz unosi sa komandne linije nakon opcije -c.

f) Prethodni program obogatiti još i mogućnošću rada sa skupovima predstavljenim preko sortiranih nizova. Napisati funkcije za izračunavanje unije, preseka, razlike i pripadanja, objediniti ih u fajl "skup.c" a zatim kreirati zaglavlje "skup.h".