Osnovi programiranja

Beleške sa vežbi

Drugi semestar školske 2005/2006 godine

Smer *Računarstvo i informatika*

Matematički fakultet, Beograd

Jelena Tomašević

April 17, 2006

Sadržaj

1

Programski jezik C

[1]

[1] Preuzeto sa sajta http://www.matf.bg.ac.yu/∼milena

2

Programski jezik C

2.1 qsort

Primer 1 *Implementacija funkcije qsort.*

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

void printarray(int v[], int left, int right)
{
    int i;
    for (i=left; i<=right; i++)
            printf("%d  ",v[i]);
    putchar('\n');
}

void swap(int v[], int i, int j)
{
    int tmp=v[i];
    v[i]=v[j];
    v[j]=tmp;
}

/* qsort:  sortira v[left]...v[right] u rastucem poretku */
void qsort(int v[], int left, int right)
{
    int i, last;

    /* ne radi nista ako niz sadrzi  */
    /* manje od dva elementa */
    if (left >= right)
        return;
    /* prebaci element particioniranja (pivot)*/
    /* u v[left] */
    swap(v, left, (left + right)/2);
    last = left;

    /* partition */
```

```
    for (i = left + 1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++last, i);

    /* restore partition  elem */
    swap(v, left, last);

    /* Sortiraj preostala dva dela niza */
    qsort(v, left, last-1);
    qsort(v, last+1, right);
}

main()
{
    int array[]={8, 3, 2, 6, 5, 7, 4, 9, 1};
    int n=sizeof(array)/sizeof(int);

    printarray(array, 0, n-1);
    qsort(array, 0, n-1);
    printarray(array, 0, n-1);

}
```

2.2 Sortiranje — generička funkcija

Sortiranje niza celih brojeva (jedan od algoritama)

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

Sortiranje iz programa mozemo da izdvojimo u funkciju:

```
void sort_int(int a[], int n)
{
  for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        if(a[i]<a[j])
        {
        int pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Sortiranje niza realnih brojeva:

```
void sort_float(float a[], int n)
```

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

Razlike:

- prvi argument funkcije;
- pomoćna promenljiva;
- poređenje.

Sortiranje studenata po oceni ukoliko je data struktura student:

```
typedef struct _student {
   char ime[MAX_IME];
   char prezime[MAX_IME];
   int ocena;
} student;

void sort_po_oceni(student a[], int n)
{
  for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        if(a[i].ocena < a[j].ocena)
        {
        student pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Sortiranje studenta po prezimenu:

```
void sort_po_prezimenu(student a[], int n)
{
  for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        if(strcmp(a[i].prezime, a[j].prezime)<0)
        {
        student pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Sortiranje studenta po imenu:

```
void sort_po_imenu(student a[], int n)
{
  for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        if(strcmp(a[i].ime, a[j].ime)<0)
        {
        student pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Kako da napravimo jednu funkciju koja sortira studente bez obzira na kriterijum?
Prvo moramo da izdvojimo funkciju poređenja:

```
int poredi_po_oceni(student st1, student st2)
{
return st1.ocena - st2.ocena;
}

int poredi_po_prezimenu(student st1, student st2)
{
return strcmp(st1.prezime, s2.prezime);
}

int poredi_po_imenu(student st1, student st2)
{
return strcmp(st1.ime, st2.ime);
}
```

Funkcija poređenja vraća 0 ukoliko su elementi jednaki, broj manji od nule ukoliko je prvi manji od drugog i broj veći od nule ukolikoje prvi veći od drugog.

```
void sort_po_imenu(student a[], int n)
{
for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        /*if(poredi_po_prezimenu(a[i], a[j])<0)*/
        /*if(poredi_po_oceni(a[i], a[j])<0)*/
        if(poredi_po_imenu(a[i], a[j])<0)
        {
        student pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Sada možemo da dodamo još jedan argument funkciji sortiranja i tako da dobijemo jednu funkciju umesto tri:

```
void sort_studente(student a[], int n,
                     int (*f)(student, student))
{
for(i=0; i<n-1; i++)
```

```
    for(j=i+1; j<n; j++)
        if((*f)(a[i], a[j])<0)
        {
        student pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Šta dalje? Kako da dobijemo jednu funkciju sortiranja bez obzira na tip elemenata niza? Teba da rešimo sledeće stvari:

- razmena mesta elemenata ne sme da zavisi od tipa elemenata koji se razmenjuju.
- potpis funkcije poređenja ne sme da zavisi od tipa elemenata koji se porede kako bi on bio jedinstven.
- prvi argument funkcije ne sme da zavisi od tipa elemenata niza.

Da bi smo razmenili dva elementa potrebna nam je pomoćna promenljiva u kojoj privremeno čuvamo neku vrednost. Ako ne znamo tip elementa onda ne možemo da napravimo pomoćnu promenljivu. Ali zato mozemo da koristeći funkciju malloc odvojimo neko mesto u memoriji za smestanje elementa koji nam u datoj situaciji treba. Koliko je to mesto? Nekada 4 bajta, npr za int, a nekada dosta veće, npr za studenta. Kako funkcija sortiranja zna koliko mesta treba da odvoji? Znace tako sto ćemo joj tu veličinu proslediti kao argument. Sada, dakle umesto pomoćne promenljive, imamo blok u memoriji, a umesto naredbe dodele koristićemo funkciju `memcpy` koja kopira deo memorije sa jednog mesta na drugo mesto.

Dakle, razmenu ćemo da radimo na sledeći način:

```
void* tmp = malloc(size);
memcpy(tmp, adresa_itog, size);
memcpy(adresa_itog, adresa_jtog, size);
memcpy(adresa_jtog, tmp, size);
free(tmp);
```

Potpis funkcije poređenja ne sme da zavisi od tipa elemenata koji se porede. To se može postići koristeći pokazivač na tip void.

Na primer, poređenje dva cela broja:

```
int poredi_br(void* a, void* b) {
    int br_a = *(int*)a;
    int br_b = *(int*)b;

    return br_a-br_b;
}
```

Na primer, poređenje dva realna broja:

```
int poredi_br(void* a, void* b) {
    float br_a = *(float*)a;
    float br_b = *(float*)b;

    if (br_a > br_b) return 1;
    else if (br_a < br_b) return -1;
    else return 0;
}
```

Na primer, poređenje dva studenta po oceni

```
int poredi_br(void* a, void* b) {
    student student1 = *(studnet*)a;
    student student2 = *(studnet*)b;

    return student1.ocena-student2.ocena;
}
```

Sada funkcija poređenja ima uvek potpis

```
int poredi(void* a, void* b)
```

i može se kao parametar proslediti našoj funkciji sortiranja.

Primer 2

```
/* Genericka funkcija sortiranja -
   nezavisna od tipa elemenata niza
   koji se sortira */
#include <stdlib.h>

void sort(void* a, int n, int size,
             int (*poredi)(void*, void*))
{
  int i, j;
  for (i = 0; i<n-1; i++)
    for (j = i+1; j<n; j++)
    {
        void* adresa_itog = (char*)a+i*size;
        void* adresa_jtog = (char*)a+j*size;

        if (poredi(adresa_itog, adresa_jtog)<0)
        {
            void* tmp = malloc(size);
            memcpy(tmp, adresa_itog, size);
            memcpy(adresa_itog, adresa_jtog, size);
            memcpy(adresa_jtog, tmp, size);
            free(tmp);
        }

    }
}


int poredi_br(void* a, void* b) {
    int br_a = *(int*)a;
    int br_b = *(int*)b;

    return br_a-br_b;
}

int poredi_float(void* a, void* b) {
    float br_a = *(float*)a;
    float br_b = *(float*)b;
```

```
    if (br_a > br_b) return 1;
    else if (br_a < br_b) return -1;
    else return 0;
}

main() {
    int a[] = {8, 2, 1, 9, 3, 7, 6, 4, 5};
    float b[] = {0.3, 2, 5, 5.8, 8}
    int n = sizeof(a)/sizeof(int);
    int nf = sizeof(b)/sizeof(float);
    int i;

    sort(a, n, sizeof(int), &poredi_br);

    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    putchar('\n');

    sort(b, nf, sizeof(float), &poredi_float);

    for (i = 0; i < n; i++)
        printf("%f ", b[i]);
    putchar('\n');
}
```

2.3 qSort funkcija iz standardne biblioteke

Primer 3 *qSort-Upotreba.*

```
/* Ilustracija upotrebe funkcije qsort iz stdlib.h
   Sortira se niz celih brojeva.
*/

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

/* const znaci da ono na sta pokazuje a (odnosno b)
   nece biti menjano u funkciji */
int poredi(const void* a, const void* b)
{
    return *((int*)a)-*((int*)b);
}

int poredi_float(const void* a, const void* b)
{
    float br_a = *(float*)a;
    float br_b = *(float*)b;

    if (br_a > br_b) return 1;
    else if (br_a < br_b) return -1;
    else return 0;
```

```
}


main()
{
    int i;
    int niz[]={3,8,7,1,2,3,5,6,9};
    float nizf[]={3.0,8.7,7.8,1.9,2.1,3.3,6.6,9.9};

    int n=sizeof(niz)/sizeof(int);
    qsort((void*)niz, n, sizeof(int),&poredi);
    for(i=0; i<n; i++)
        printf("%d",niz[i]);


    n=sizeof(nizf)/sizeof(float);
    qsort((void*)nizf, n, sizeof(float),&poredi_float);
    for(i=0; i<n; i++)
            printf("%f",nizf[i]);

}
```

Primer 4 *Binarno pretraživanje - korišćenje ugrađene bsearch funkcije.*

```
/* Funkcija ilustruje koriscenje ugradjene funkcije bsearch */
#include <stdlib.h>

int poredi(const void* a, const void *b)
{
    return *(int*)a-*(int*)b;
}

main()
{
    int x=-1;
    int niz[]={1,2,3,4,5,6,7,8,9,10,11,12};

    int* elem=(int*)bsearch((void*)&x,(void*)niz,
        sizeof(niz)/sizeof(int),sizeof(int),&poredi);

if (elem==NULL)
    printf("Element nije pronadjen\n");
else
    printf("Element postoji
                na poziciji %d\n",elem-niz);
}
```

2.4 Generičko sortiranje reči

Primer 5 *Sortiranje reči. Ako se sortira niz stringova, onda svaki element je sam po sebi pokazivač tipa* `char *`*, te funkcija poređenja tada prima podatke tipa* `char **` *koji se konvertuju u svoj tip i derefenciraju radi dobijanja podataka tipa* `char *`*.*

```
/* Ilustracija upotrebe funkcije qsort iz stdlib.h
   Sortira se niz reci i to ili leksikografski
   ili po duzini
*/

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

int poredi(const void* a, const void* b)
{
    char *s1 = *(char **)a;
    char *s2 = *(char **) b;
    return strcmp(s1, s2);

    /* Prethodno je ekvivalentno sa:
    return strcmp(*(char**)a,*(char**)b); */
}

int poredi_po_duzini(const void* a, const void* b)
{
    char *s1 = *(char **) a;
    char *s2 = *(char **) b;
    return strlen(s1) - strlen(s2);
    /* Prethodno je ekvivalentno sa:
    return strlen(*(char**)b)-strlen(*(char**)a); */
}

main()
{
    int i;
    char* nizreci[]= {"Jabuka","Kruska","Sljiva","Dinja","Lubenica"};

    qsort((void*)nizreci,5,
            sizeof(char*),&poredi_po_duzini);

    for (i=0; i<5; i++)
        printf("%s\n",nizreci[i]);

    qsort((void*)nizreci,5,
            sizeof(char*),&poredi);

    for (i=0; i<5; i++)
         printf("%s\n",nizreci[i]);
}
```

2.5 Zadaci za vežbu:

Zadatak 1 *Napisati program koji sa standardnog ulaza ucitava 2 stringa, s i t (duzine ¡=20), sortira nizove njihovih karaktera (biblioteckom qsort funkcijom) i ispituje i stampa da li su s i t*

anagrami (npr. vrata, vatra).

Zadatak 2 *Napisati program koji sa standardnog ulaza ucitava prvo ceo broj n (n¡=10) a zatim niz S od n stringova (maksimalna duzina stringa je 20), sortira niz S (biblioteckom funkcijom qsort) i proverava da li u njemu ima identicnih stringova.*

Zadatak 3 *Napisati program u kome se prvo inicijalizuje staticki niz struktura osoba sa clanovima ime i prezime(uredjen u rastucem poretku prezimena) sa ¡=10 elemenata, a zatim se ucitava jedan karakter i pronalazi(sa bsearch) i stampa jedna struktura iz niza osoba cije prezime pocinje tim kakrakterom(ako takva postoji).*