Programiranje 2

Beleške sa vežbi

Školska 2006/2007 godina

Matematički fakultet, Beograd

Jelena Tomašević

April 4, 2007

Sadržaj

1

Programski jezik C

[1]

1.1 Pokazivači na funkcije

Primer 1 *Program demonstrira upotrebu pokazivača na funkcije.*

```
#include <stdio.h>

int kvadrat(int n) { return n*n; }

int kub(int n) { return n*n*n; }

int parni_broj(int n) { return 2*n; }

/* Funkcija izracunava sumu od 1 do n f(i), gde je f data funkcija.
   int (*f) (int) u argumentu funkcije sumiraj je pokazivac na funkciju sa imenom f,
   koja kao argument prima promenljivu tipa int i vraca kao rezultat vrednost tipa int */

int sumiraj(int (*f) (int), int n) {
    int i, suma=0;
    for (i=1; i<=n; i++)
        suma += (*f)(i);

    return suma;
}

main(){

/*  U pozivu funkcije sumiraj, kvadrat, kub i parni_broj su adrese funkcija pa operator & nije
    neophodan, iz istog razloga zbog kojeg on nije bio potreban ni za ime niza.*/

printf("Suma kvadrata brojeva od jedan do 3 je %d\n", sumiraj(kvadrat,3));
printf("Suma kubova brojeva od jedan do 3 je %d\n", sumiraj(kub,3));
printf("Suma prvih pet parnih brojeva je %d\n", sumiraj(parni_broj,5));
}
```

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

```
 /*
 Izlaz:
 Suma kvadrata brojeva od jedan do 3 je 14
 Suma kubova brojeva od jedan do 3 je 36
 Suma prvih pet parnih brojeva je 30
 */
```

Podsetimo se sada pokazivača na tip void:

```
void *pp;
```

Svaki pokazivač se može pretvoriti u tip void * i opet vratiti u svoj prvobitni tip bez gubitka informacije. Njemu se dakle može dodeliti da pokazuje na int, ili na char ili na proizvoljan tip ali je to neophodno eksplicitno naglasiti svaki put kada želimo da koristimo ono na šta on pokazuje.

Primer 2 *Upotreba pokazivača na prazan tip.*

```
#include<stdio.h>

main()
{
void *pp;
int x=2;
char c='a';

pp = &x;
*(int *)pp = 17;    /* x postaje 17*/
printf("\n adresa od x je %p", &x);
printf("\n%d i %p",*(int*)pp,(int * )pp);

pp = &c;
printf("\n adresa od c je %p", &c);
printf("\n%c i %p",*(char*)pp,(char * )pp);

}

/*
 adresa od x je 0012FF78
17 i 0012FF78
 adresa od c je 0012FF74
a i 0012FF74

*/
```

1.2 qsort – funkcija iz standardne biblioteke

Prototip funkcije qsort iz stdlib.h je:

```
void qsort(void *niz, int duzina_niza, int velicina_elementa_niza,
    int (*poredi)(const void*, const void*) )
```

Ova funkcija sortira niz niz[0], niz[1],...,niz[duzina_niza - 1] elemenata veličine velicina_elementa_niza. Funkcija poredi vrši poređenje dva elementa niza, vraća pozitivnu vrednost ako je prvi element veći od drugog, 0 ako su jednaki i negativnu vrednost ako je prvi manji. Tada će se niz sortirati u

rastućem poretku. Modifikacijom ove funkcije niz se može sortirati u opadajućem poretku (ukoliko vraća pozitivnu vrednost ako je prvi manji, 0 ako su jednaki i negativnu vrednost ako je prvi veći).

Primer 3 *Upotrebom qsort funkcije iz standardne biblioteke izvršiti sortiranje niza celih i niza realnih 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);
    printf("Sortirani niz celih brojeva je:\n");
    for(i=0; i<n; i++)
        printf("%d\t",niz[i]);

    printf("\n\nSortirani niz realnih brojeva je:\n");

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

}
```

Primer 4 *Izvršiti sortiranje niza reči, leksikografski odnosno po dužini, korišćenjem ugradjene qsort funkcije.*

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

/* Funkcija koja vrsi leksikografsko poredjenje dve reci.
   Vraca kao rezultat >0 ukoliko je prva rec veca, 0 ako su jednake
   i <0 ako je prva rec manja. Sortiranje ce biti u rastucem redosledu.*/
int poredi_leksikografski(const void* a, const void* b)
{    return strcmp(*(char**)a,*(char**)b);   }

/* Funkcija koja vrsi poredjenje po duzini dve reci, opadajuce!!!*/
int poredi_po_duzini(const void* a, const void* b)
{     return strlen(*(char**)b)-strlen(*(char**)a); }


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

    /* Sortiramo leksikografski i ispisujemo rezultat */
    qsort((void*)a, n, sizeof(char*), poredi_leksikografski);
    for (i=0; i<n; i++)
        printf("%s ", a[i]);
    putchar('\n');


    /* Sortiramo po duzini i ispisujemo rezultat */
    qsort((void*)a, n, sizeof(char*), poredi_po_duzini);
    for (i=0; i<n; i++)
        printf("%s ", a[i]);
    putchar('\n');
}
/*
Izlaz:
Dinja Jabuka Kruska Lubenica Sljiva
Lubenica Kruska Jabuka Sljiva Dinja
*/
```

1.3 bsearch – funkcija iz standardne biblioteke

Prototip funkcije qsort iz stdlib.h je:

```
void *bsearch (const void *kljuc, const void *niz, int duzina_niza, int velicina_elementa_niza,
    int (*poredi)(const void*, const void*) )
```

Ova funkcija u nizu niz[0], niz[1],...,niz[duzina_niza - 1] elemenata veličine velicina_elementa_niza traži element koji se poklapa sa *kljuc. Funkcija poredi mora vratiti vrednost ¿0 ako je prvi argument (ključ pretrage) veći od drugog argumenta (koji je element niza), 0 ako su jednaki i ¡0 ako je prvi manji. Elementi u nizu niz moraju biti u rastućem redosledu. Funkcija bsearch vraća pokazivač na pronađeni element ili NULL ako takav ne postoji.

Primer 5 *Binarno pretraživanje - korišćenje ugrađene bsearch funkcije.*

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

/* Vrsi se poredjenje podatka a po kome se pretrazuje niz sa elementom niza b.*/
int poredi(const void* a, const void *b)
{
    return *(int*)a-*(int*)b;
}

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

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

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

Primer 6 ***Binarna pretraga niza struktura - pretraga studenata po broju indeksa***

Datoteka sadrži podatke o uspehu studenata na kolokvijumima iz osnova programiranja. Prva linija datoteke sadrži broj studenata (¡1000), a zatim svaka sledeća linija sadrži ime i prezime odredjenog studenta, njegov broj indeksa (u obliku korisničkog imena na alas-u npr. mr01123) i broj poena na prvom i na drugom kolokvijumu. Redosled studenata u datoteci je odredjen na osnovu njihovog broja indeksa, i to tako da su na početku datoteke navedeni stariji studenti sa manjim brojevima indeksa (npr. mv02234 je ispred mn03123 jer je stariji, a mr03123 je ispred ml03234 jer ima manji broj indeksa).

a) Definisati strukturu podataka za čuvanje podataka o studentima

b) Napraviti funkciju koja poredi dva indeksa u skladu sa poretkom opisanim u uvodu zadatka

c) Napisati funkciju

int binary_search_stdlib(char indeks, ...)*

koja pozivom bibliotecke funkcije bsearch za dati indeks studenta vraća broj poena koje je ostvario na prvom plus drugom kolokvijumu.

d) Napisati program koji učitava podatke o studentima iz datoteke čije je ime navedeno kao argument komandne linije, zatim koristeći funkcije iz prethodnog dela zadatka ispisuje ukupan broj poena za svakog studenata čiji indeks korisnik unese sa standardnog ulaza, sve dok ne unese reč kraj za kraj.

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

/* Definicija strukture za cuvanje podataka o jednom studentu */
typedef struct _student
{
    char ime[15];
```

```
    char prezime[15];
    char indeks[8];
    int  kol_1;
    int  kol_2;
} student;

/* Studenti se smestaju u niz koji nema vi\v{s}e od 1000 elemenata.
   Pretpostavlja se da je niz sve vreme sortiran kako bi se
   moglo vrsiti binarno pretrazivanje */
student studenti[1000];
int br_stud;

/* Funkcija vrsi poredjenje indeksa. Manjim se smatraju stariji studenti
   sa manjim brojem indeksa. Funkca vraca negativnu vrednost ukoliko je
   prvi indeks manji, pozitivnu ukoliko je veci, a 0 ukoliko su indeksi
   jednaki. Informacije o smerovima se u potpunosti zanemaruju.
*/
int poredi_indekse(char* indeks_1, char* indeks_2)
{
    /* Izdvaja se informacija o godini upisa i
       broju indeksa za oba studenta*/
    char s_godina_1[3], s_godina_2[3];
    int  godina_1, godina_2;
    char s_broj_1[4], s_broj_2[4];
    int  broj_1, broj_2;


    /*mr99123 upisan je 1999 a mr01123 upisan je 2001*/

    /* Godina upisa je zapisana u 3 i 4 karakteru indeksa.*/
    strncpy(s_godina_1, indeks_1+2, 2);
    godina_1 = atoi(s_godina_1);
    if (godina_1 < 10) godina_1 += 100;
    godina_1 += 1900;

    strncpy(s_godina_2, indeks_2+2, 2);
    godina_2 = atoi(s_godina_2);
    if (godina_2 < 10) godina_2 += 100;
    godina_2 += 1900;

    /* Prvo poredimo godine */
    if (godina_1 < godina_2)
        return -1;

    if (godina_1 > godina_2)
        return 1;

    /* Ukoliko su godine jednake, izdvajamo brojeve indeksa
       koji su zapisani u 5. 6. i 7. karakteru indeksa */
    strncpy(s_broj_1, indeks_1+4, 3);
    broj_1 = atoi(s_broj_1);
    strncpy(s_broj_2, indeks_2+4, 3);
```

```
    broj_2 = atoi(s_broj_2);

    /* Umesto ovoga, prolazi i return strcmp(indeks_1+4, indeks_2+4); */

    /* Poredimo godine upisa */
    return broj_1-broj_2;
}


/* Funkcija poredjenja koja je potrebna prilikom poziva
   ugradjene funkcije bsearch.
   Vrsi se poredjenje prvog argumenta indeks koji je kljuc pretrage i tipa je string
   sa elementom niza koji je tipa struktura student */
int poredi(const void* indeks, const void* indeks_2)
{
    return poredi_indekse(*(char**)indeks, ((student*)indeks_2)->indeks);
}

int binary_search_stdlib(char* indeks)
{
    student *s = (student*)bsearch((const void*)&indeks, (const void*)studenti,
                                                br_stud, sizeof(student), poredi);
    return s==NULL ? -1 : s->kol_1+s->kol_2;
}

/* Glavni program */
main(int argc, char* argv[])
{
    FILE* datoteka;
    int i;
    char indeks[8];

    /* Proveravamo prisutnost argumenata komandne linije */
    if (argc<2)
    {   printf("Upotreba %s : ime_datoteke\n",argv[0]);
        return -1;
    }


    /* Otvaramo datoteku */
    if ((datoteka = fopen(argv[1], "r")) == NULL)
    {   printf("Greska prilikom otvaranja datoteke %s\n",argv[1]);
        return -1;
    }

    /* Ucivamo broj studenata i alociramo niz */
    fscanf(datoteka, "%d", &br_stud);

    /* Ucitavamo podatke o studentima */
    for (i = 0; i < br_stud; i++)
    {   fscanf(datoteka, "%s",studenti[i].ime);
        fscanf(datoteka, "%s",studenti[i].prezime);
```

```
        fscanf(datoteka, "%s",studenti[i].indeks);
        fscanf(datoteka, "%d", &studenti[i].kol_1);
        fscanf(datoteka, "%d", &studenti[i].kol_2);
    }

    /* Ispisujemo poene za svakog studenta kojeg korisnik trazi */
    printf("Unesi broj indeksa : ");
    scanf("%s",indeks);
    while (strcmp(indeks,"kraj") != 0)
    {
        printf("%s ",indeks);
        printf("Broj poena : %d\n",binary_search_stdlib(indeks));
        printf("Unesi broj indeksa : ");
        scanf("%s",indeks);
    }

    return 0;

}
```

Primer 7

1.4 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).*