Osnovi programiranja

Beleške sa vežbi

Drugi semestar školske 2005/2006 godine

Smer *Računarstvo i informatika*

Matematički fakultet, Beograd

Jelena Tomašević

May 7, 2006

Sadržaj

1 Programski jezik C **5**

2 Programski jezik C **7**
2.1 Liste 7
2.1.1 Dvosturko povezana kružna lista 16
2.2 Zadaci za vežbu 18

1

Programski jezik C

[1]

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

2

Programski jezik C

2.1 Liste

Primer 1 *Ilustracija rada sa listama.*

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

typedef struct elem {
   int broj;
   struct elem *sled;
} Elem;

/* Citanje brojeva i formiranje liste */
Elem *citaj (void)
{
Elem *lista = NULL, *tekuci = NULL, *novi; int broj;

printf("\n\nUnesite elemente liste - 0 za kraj\n");
scanf ("%d", &broj);

while (broj)
{
    /*Alocira se prostor za novi clan liste*/
    novi =(Elem*)malloc (sizeof (Elem));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom
                        alokacije memorije\n");
        exit(1);
    }

    /* Postavljanje vrednosti */
    novi->broj = broj;
    novi->sled = NULL;

    /* Ukoliko lista nije prazna novi
       element se postavlja iza poslednjeg
```

```
       elemnta liste - na koji pokazuje tekuci */
    if (tekuci != NULL)
        tekuci->sled = novi;
    /* Inace se postavlja da bude pocetak liste */
    else
        lista = novi;

    /*Tekuci se postavlja da pokazuje
      na poslednji element liste */
    /*Ekvivalentno je sa
      tekuci = tekuci ->sledeci */
    tekuci = novi;
    /* Ucitavanje novog elementa liste 0 za kraj */
    scanf ("%d", &broj);
} /* Funkcija vraca pokazivac na pocetak liste */
return lista;
}

/* Ispisivanje liste*/
void pisi (Elem *lista)
{
  while (lista != NULL)
  {
     printf ("%d ", lista->broj),
     lista = lista->sled;
  }
}

/* Oslobadjanje memorije koju lista zauzima*/
void brisi (Elem* lista)
{
  Elem *stari;
  while (lista != NULL)
  {
     stari = lista;
     lista = lista->sled;
     free (stari);
  }
}

/* Izostavljanje (brisanje) zadatog
   broja iz liste. Kako se ovime pocetak
   liste moze izmeniti, vrednost pocetka
   liste je povratna vrednost
   funkcije */
Elem* izost(Elem *lista, int k)
{
   Elem *preth = NULL, *tekuci = lista, *stari;

   while (tekuci != NULL)
      /* Ukoliko tekuci pokazuje na clana
         liste kojeg treba izbaciti */
```

```
      if (tekuci->broj == k)
      {
         stari = tekuci;
         tekuci = tekuci->sled;

         if (preth != NULL )
            preth->sled = tekuci;
        /* Ovaj slucaj odnosi se na izbacivanje
           elementa sa pocetka liste */
         else lista = tekuci;
         free (stari);
      }
      else
      {
      preth = tekuci;
      tekuci = tekuci->sled;
      }
  return lista;
}

main ()
{
/* test glavni program */
Elem *lista;
int k;

lista = citaj ();
printf ("Ucitani lista   = ");
pisi (lista);
putchar ('\n');

printf("Koji element liste zelite da izbacite?\n");
scanf ("%d", &k);
printf ("Izostavlja se = %d\n", k);
printf ("Novi lista      = ");
pisi (lista = izost (lista, k));
printf ("\n");

printf("Oslobadjam memoriju\n");
brisi (lista);

return 0; }
```

Primer 2 *Rad sa listama - celine izdvojene u funkcije.*

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

typedef struct cvor {
    int br;
    struct cvor* sl;
} CVOR;
```

```
/* Pomocna funkcija koja kreira cvor liste
   sa datim sadrzajem.
   Funkcija kreira cvor i postavlja mu sadrzaj
   na dati broj.
   Polje sl ostaje nedefinisano.
   Funkcija vraca pokazivac na kreirani cvor. */

CVOR* napravi_cvor(int br)
{
    CVOR* novi = (CVOR*)malloc(sizeof(CVOR));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom
                    alokacije memorije\n");
        exit(1);
    }
    novi->br = br;
    return novi;
}

/* ------------------------------------------- */

 /* Ispisivanje liste : iterativna verzija */

void ispisi_listu_i(CVOR* l)
{
    CVOR* t;
    for (t = l; t != NULL; t=t->sl)
        printf("%d ", t->br);
}

/* Ispisivanje liste : rekurzivna verzija */
void ispisi_listu_r(CVOR* l)
{
    if (l != NULL)
    {
        printf("%d ", l->br);
        ispisi_listu_r(l->sl);
    }
}

/* Ispisivanje liste unatrag : rekurzivna verzija */
/* Prethodna funkcija se lako modifikuje tako da
   ispisuje listu unazad */
void ispisi_listu_unazad(CVOR* l)
{
    if (l != NULL)
    {
        ispisi_listu_unazad(l->sl);
        printf("%d ", l->br);
    }
}
```

```
/* ------------------------------------------- */
/* Oslobadjanje liste : iterativna verzija */
void oslobodi_listu_i(CVOR* l)
{
    while (l)
    {
        CVOR* tmp = l->sl;
        free(l);
        l = tmp;
    }
}

/* Oslobadjanje liste : rekurzivna verzija */
void oslobodi_listu_r(CVOR* l)
{
    if (l != NULL)
    {
        oslobodi_listu_r(l->sl);
        /* Prvo se oslobadja poslednji element liste */
        free(l);
    }
}

/* ------------------------------------------- */

/* Ubacuje dati broj na pocetak date liste.
   Funkcija pozivaocu eksplicitno vraca pocetak
   rezultujuce liste.*/
CVOR* ubaci_na_pocetak(CVOR* l, int br)
{
    CVOR* novi = napravi_cvor(br);
    novi->sl = l;
    return novi;
}

/* Funkcija vraca pocetak rezultujuce liste, a ubacuje
   cvor na kraj bez pamcenja pokazivaca na kraj */
CVOR* ubaci_na_kraj(CVOR* l, int br) {
    CVOR* novi = napravi_cvor(br);
    novi->sl = NULL;

    if (l == NULL)
        return novi;
    else
    {
        CVOR* t;
        /* Prodjemo do kraja liste */
        for (t = l; t->sl!=NULL; t=t->sl)
            ;
        t->sl = novi;
```

```
        /* Pocetak se nije promenio */
        return l;
    }
}

/* Rekurzivna varijanta prethodne funkcije.
   I ova funkcija vraca pokazivac na pocetak
   rezultujuce liste  */
CVOR* ubaci_na_kraj_rekurzivno(CVOR* l, int br)
{
    if (l == NULL)
    {
        CVOR* novi = napravi_cvor(br);
        novi->sl = NULL;
        return novi;
    }

    l->sl = ubaci_na_kraj_rekurzivno(l->sl, br);
    return l;
}

/* ------------------------------------------- */

/* Kljucna ideja u realizaciji ove funkcije je
   pronalazenje poslednjeg elementa liste ciji je
   kljuc manji od datog elementa br.*/
CVOR* ubaci_sortirano(CVOR* pl, int br)
{
    CVOR* novi = napravi_cvor(br);

    /* U sledeca dva slucaja ne postoji cvor
       ciji je kljuc manji
       od datog broja (br)
       - Prvi je slucaj prazne liste
       - Drugi je slucaj kada je br manji
         od prvog elementa

       U oba slucaja ubacujemo na pocetak liste.
    */
    if (pl == NULL || br < pl->br)
    {
        novi->sl = pl;
        pl = novi;
        return pl;
    }

    /* Krecemo od pocetka i idemo
       dalje sve dok t nije poslednji
       manji element liste ili eventualno
       bas poslednji */
    CVOR* t;
    for(t = pl;
```

```
            t->sl!=NULL && t->sl->br < br;
                                    t=t->sl)
        ;
    novi->sl = t->sl;
    t->sl = novi;

    return pl;
}

main() {
    CVOR* l = NULL;
    CVOR* s = NULL;

    int i;
    for (i = 0; i<5; i++)
        l = ubaci_na_kraj(l, i);
    for (; i<10; i++)
        l = ubaci_na_kraj_rekurzivno(l, i);

    for (i = 0; i < 10 ; i++)
        l = ubaci_na_pocetak(l, i);

    ispisi_listu_i(l);
    putchar('\n');

    ispisi_listu_r(l);
    putchar('\n');

    ispisi_listu_unazad(l);
    putchar('\n');

    oslobodi_listu_i(l);

    s = ubaci_sortirano(s, 5);
    s = ubaci_sortirano(s, 8);
    s = ubaci_sortirano(s, 7);
    s = ubaci_sortirano(s, 6);
    s = ubaci_sortirano(s, 4);

    ispisi_listu_r(s);
    putchar('\n');

    oslobodi_listu_r(s);
}
```

Primer 3 *Program ispisuje broj pojavljivanja za svaku od reči koja se pojavila u tekstu unetom sa standardnog ulaza. Verzija sa (sortiranom) listom.*

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

/* Definicija cvora liste */
typedef struct _cvor
```

```
{
    char ime[80];
    int br_pojavljivanja;
    struct _cvor* sledeci;
} cvor;

/* Funkcija ispisuje listu rekurzivno, pocevsi od poslednjeg
elementa */
void ispisi_listu(cvor* pocetak)
{
    if(pocetak!=NULL)
     {
        ispisi_listu(pocetak->sledeci);
        printf("%s %d\n",pocetak->ime,pocetak->br_pojavljivanja);
     }
}

/* Funkcija koja brise listu */
void obrisi_listu(cvor* pocetak)
{
    if (pocetak!=NULL)
     {  obrisi_listu(pocetak->sledeci);
        free(pocetak);
     }
}

/* Funkcija ubacuje datu rec u listu koja je leksikografski sortirana,
   na odgovarajuce mesto i vraca pokazivac na novi pocetak liste */
cvor* ubaci_sortirano(cvor* pocetak, char* rec)
{
    int cmp;
    /* Ukoliko je lista prazna ubacujemo na pocetak liste*/
    if (pocetak==NULL)
    {    pocetak=(cvor*)malloc(sizeof(cvor));
         if (pocetak == NULL)
         {
         printf("Greska prilikom alokacije memorije!\n");
         exit(1);
         }
         strcpy(pocetak->ime,rec);
         pocetak->br_pojavljivanja=1;
         return pocetak;
    }
    /* Ukoliko lista nije prazna poredimo rec sa elementom u glavi */
    cmp=strcmp(pocetak->ime,rec);
    /* Ukoliko je rec pronadjena samo uvecavamo njen broj pojavljivanja */
    if (cmp==0)
    {   pocetak->br_pojavljivanja++;
        return pocetak;
    }
    /* Ukoliko je rec koju ubacujemo veca od tekuce reci, ubacujemo je
       rekurzivno u rep */
```

```
    else if (cmp>0)
    {    pocetak->sledeci=ubaci_sortirano(pocetak->sledeci,rec);
         return pocetak;
    }
    /* Ukoliko je rec koju ubacujemo manja od tekuce reci, gradimo novi
       cvor i ubacujemo ga ispred pocetka */
    else
    {    cvor* novi=malloc(sizeof(cvor));
         if (novi == NULL) {
                           printf("Greska prilikom alokacije memorije!\n");
                           exit(1);
                           }
         strcpy(novi->ime,rec);
         novi->br_pojavljivanja=1;
         novi->sledeci=pocetak;
         return novi;
    }
}


/* Pomocna funkcija koja cita rec sa standardnog ulaza i vraca
njenu duzinu, odnosno -1 ukoliko se naidje na EOF */

int getword(char word[], int lim)
{
    int c, i=0;
    while (!isalpha(c=getchar()) && c!=EOF)
              ;

    if (c==EOF)
          return -1;
    do
    {   word[i++]=c;
    }while (i<lim-1 && isalpha(c=getchar()));

    word[i]='\0';
    return i;
}

/* Funkcija koja pronalazi datu rec u datoj listi.
   Funkcija vraca pokazivac na cvor u kome je nadjena rec, ili
   NULL ukoliko rec nije nadjena */

cvor* nadji_rec(cvor* lista, char rec[])
{
     if (lista==NULL)
          return NULL;
     if (strcmp(lista->ime,rec)==0)
          return lista;

     return nadji_rec(lista->sledeci,rec);
}
```

```
main() {
   cvor* lista=NULL;
   char procitana_rec[80];
   while(getword(procitana_rec,80)!=-1)
   {     cvor* pronadjen=nadji_rec(lista,procitana_rec);
         if (pronadjen!=NULL)
               pronadjen->br_pojavljivanja++;
         else
               lista=ubaci_sortirano(lista,procitana_rec);
   }
   ispisi_listu(lista);
   obrisi_listu(lista);

}
```

2.1.1 Dvosturko povezana kružna lista

Primer 4 *Napisati funkciju koja omogućava umetanje čvora u dvostruko povezanu kružnu listu kao i izbacivanje čora iz dvostruko povezane kružne liste. Omogućiti i štampanje podataka koje čuva lista.*

```
/* Program implementira deciju razbrajalicu eci-peci-pec i sluzi
da ilustruje rad sa dvostruko povezanim kruznim listama */

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

/* Dvostruko povezana lista */
typedef struct _cvor
{
   int broj;
   struct _cvor* prethodni, *sledeci;
} cvor;

/* Umetanje u dvostruko povezanu listu */
cvor* ubaci(int br, cvor* lista)
{
    cvor* novi=(cvor*)malloc(sizeof(cvor));
    if (novi==NULL)
    {   printf("Greska prilikom alokacije memorije \n");
        exit(1);
    }
    novi->broj=br;

    if (lista==NULL)
    {
       novi->sledeci=novi;
       novi->prethodni=novi;
       return novi;
    }
```

```
    else
    {
       novi->prethodni=lista;
       novi->sledeci=lista->sledeci;
       lista->sledeci->prethodni=novi;
       lista->sledeci=novi;
       return novi;
    }
}

/* Ispis liste */
void ispisi(cvor* lista)
{
    if (lista!=NULL)
    {   cvor* tekuci=lista;
        do
        {   printf("%d\n",tekuci->broj);
            tekuci=tekuci->sledeci;
        } while (tekuci!=lista);
    }
}

/* Izbacivanje datog cvora iz liste */
cvor* izbaci(cvor* lista)
{
    if (lista!=NULL)
    {   cvor* sledeci=lista->sledeci;
        if (lista==lista->sledeci)
        {   printf("Pobednik %d\n",lista->broj);
            free(lista);
            return NULL;
        }

        printf("Ispada %d\n",lista->broj);

        lista->sledeci->prethodni=lista->prethodni;
        lista->prethodni->sledeci=lista->sledeci;
        free(lista);
        return sledeci;
    }
    else return NULL;
}

main()
{
    /* Umecemo petoro dece u listu */
    cvor* lista = NULL;
    lista=ubaci(1,lista);
    lista=ubaci(2,lista);
    lista=ubaci(3,lista);
    lista=ubaci(4,lista);
    lista=ubaci(5,lista);
```

```
    lista=lista->sledeci;

    int smer = 0;
    /* Dok ima dece u listi */
    while(lista!=NULL)
    {   int i;

    /* brojimo 13 slogova u krug i
           u svakom brojanju menjamo smer obilaska*/
        for (i=1; i<=13; i++)
           lista = 1-smer ? lista->sledeci : lista->prethodni;

        lista=izbaci(lista);
        smer = smer ? 0 : 1;
    }
    ispisi(lista);
}
```

2.2 Zadaci za vežbu

Zadatak 1 *Brojeve sa ulaza smeštati u listu sve dok se ne unese nula, a zatim dobijenu listu ispisati na izlaz.*

1. *Zadatak realizovati dodavanjem elemenata liste na početak liste.*
2. *Zadatak realizovati tako da listu koja se formira bude sortirana.*
3. *Zadatak realizovati dodavanjem elemenata liste na kraj liste a listu ispisati unazad.*

Zadatak 2 *Grupa od n plesača (na čijim kostimima su u smeru kazaljke na satu redom brojevi od* 1 *do n) izvodi svoju plesnu tačku tako što formiraju krug iz kog najpre izlazi k-ti plesač (odbrojava se počev od plesača označenog brojem* 1 *u smeru kretanja kazaljke na satu). Preostali plesači obrazuju manji krug iz kog opet izlazi k-ti plesač (odbrojava se pocev od sledećeg suseda prethodno izbačenog, opet u smeru kazaljke na satu). Izlasci iz kruga se nastavljaju sve dok svi plesači ne budu isključeni. Celi brojevi* `n`, `k` *($k < n$) se učitavaju sa standardnog ulaza. Napisati program koji će na standardni izlaz ispisati redne brojeve plesača u redosledu napuštanja kruga.*

PRIMER: za $n = 5$, $k = 3$ redosled izlaska je 3 1 5 2 4.