Programiranje II

Beleške sa vežbi

Smer *Informatika*

Matematički fakultet, Beograd

Sana Stojanović

08.05.08.

Sadržaj

1 Osnovne funkcije za rad sa listama

1.1 Ubacivanje elementa na početak jednostruko povezane liste, ispisivanje liste, oslobađanje liste

1. Napisati program koji kreira jednostruko povezanu listu. Elemente, cele brojeve od 1 do 10, ubacivati na početak liste.

   Napisati:

   - **funkciju koja kreira jedan čvor liste**,
     `CVOR* napravi_cvor(int br)`
   - **funkciju za ubacivanje broja na početak liste pomoću pokazivača na početak liste**,
     `void ubaci_na_pocetak(CVOR** pl, int br)`
   - **funkciju za štampanje liste**,
     `void ispisi_listu(CVOR* l)`
   - **funkciju za oslobađanje liste**,
     `void oslobodi_listu(CVOR* l)`.

```
/* Ubacivanje na pocetak jednostruko povezane liste - verzija sa
   pokazivacem na pocetak liste.
   Ispis i oslobadjanje liste realizovani iterativno. */

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

/* Struktura koju koristimo za cuvanje jednog cvora liste */
typedef struct cvor
{
    int br;
    struct cvor* sl;    //cak i kada koristimo typedef, na ovom mestu
                        //je neophodno koriscenje struct cvor
} 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;     //pomocni pokazivac

    //odvajamo prostor potreban za smestanje jednog cvora liste
```

```
    novi = (CVOR*)malloc(sizeof(CVOR));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom alokacije memorije\n");
        exit(1);
    }

    //Postavljamo polje na vrednost koja je predata funkciji
    novi->br = br;
    novi->sl = NULL;    //ova naredba nije neophodna ali malo olaksava
                        //pisanje narednih funkcija

    //Kreiran cvor vracamo kao povratnu vrednost funkcije
    return novi;
}


/* Funkcija koja ubacuje dati broj na pocetak liste.
   Prenosi se pokazivac na pocetak liste kako bi mogla da se izmeni
   vrednost pocetka liste. */

void ubaci_na_pocetak(CVOR** pl, int br)
{
    //funkcija napravi_cvor kreira novi cvor...
    CVOR* novi = napravi_cvor(br);

    //...koji sada povezujemo sa starom listom tako da novi cvor
    //predstavlja novi pocetak liste
    novi->sl = *pl;
    *pl = novi;
}

/* Zbog prenosa po vrednosti, sledeca funkcija ne radi ispravno */
/* void ubaci_na_pocetak(CVOR* l, int br) {
    CVOR* novi = napravi_cvor(br);
    novi->sl = l;
    l = novi;  /* Ovde se menja lokalna kopija pokazivaca l, a
              ne l iz funkcije pozivaoca (main) */
} */


/* Ispisivanje liste : iterativna verzija */
void ispisi_listu(CVOR* l)
{
    CVOR* t;
```

```
    //petlja pomocu koje se setamo kroz celu listu
    for (t = l; t != NULL; t=t->sl)
        printf("%d ", t->br);
}


/* Oslobadjanje liste : iterativna verzija.
   Prilikom oslobadjanja liste moramo voditi racuna o redosledu
   oslobadjanja pokazivaca (ako oslobodimo pokazivac l nakon toga
   ne mozemo osloboditi ostatak liste) */

void oslobodi_listu(CVOR* l)
{
    //dok ima elemenata u listi
    while (l)
    {
        //Pomocni pokazivac na ostatak liste
        CVOR* tmp = l->sl;

        //oslobadjamo pocetak liste
        free(l);

        //iterativna promena da bismo oslobodili jedan po jedan cvor
        //ostatka liste
        l = tmp;
    }
}

/* Sledeca funkcija je neispravna */
/* void oslobodi_listu(CVOR* l)
{
    CVOR* t;
    for (t = l; t!=NULL; t = t->sl)
        free(t);
        /* Ovde se unistava sadrzaj cvora na koji ukazuje t.
           Korak petlje t = t->sl nece moci da se izvrsi */

}
*/


main()
{
    CVOR* l = NULL;     //Pokazivac na pocetak liste postavljamo na NULL
    int i;              //Brojac koji cemo koristiti da generisemo brojeve
                        //za ubacivanje u listu
```

```
    //for petlja koja nam sluzi samo kao generator brojeva koje
    //cemo ubacivati u listu
    for (i = 0; i<10; i++)
        ubaci_na_pocetak(&l, i);

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

    oslobodi_listu(l);
}
```

1.2 Ubacivanje elementa na početak liste (sa eksplicitnim vraćanjem novog početka liste), rekurzivna funkcija za ispis liste, rekurzivna funkcija za oslobađanje liste

1. Napisati program koji kreira jednostruko povezanu listu. Elemente, cele brojeve od 1 do 10, ubacivati na početak liste. Napisati:
   - funkciju koja kreira jedan čvor liste,
     `CVOR* napravi_cvor(int br)`
   - **funkciju za ubacivanje broja na početak liste sa eksplicitnim vraćanjem novog početka liste**,
     `CVOR* ubaci_na_pocetak(CVOR* l, int br)`
   - **rekurzivnu funkciju za štampanje liste**,
     `void ispisi_listu(CVOR* l)`
   - **rekurzivnu funkciju za oslobađanje liste**,
     `void oslobodi_listu(CVOR* l)`.

```
/* Ubacivanje na pocetak jednostruko povezane liste - verzija sa
   eksplicitnim vracanjem novog pocetka liste.
   Ispis i oslobadjanje liste su realizovani rekurzivno
*/

#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;
    novi->sl = NULL;
    return novi;
}

/* 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;
}

/* Ispisivanje liste : rekurzivna verzija */
void ispisi_listu(CVOR* l)
{
    //Ako ima elemenata u listi...
    if (l != NULL)
    {
        //...ispisi tekuci
        printf("%d ", l->br);

        //...i rekurzivno ispisi ostatak liste
        ispisi_listu(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 : rekurzivna verzija */
void oslobodi_listu(CVOR* l)
{
    //Ako ima elemenata u listi...
    if (l != NULL)
    {
        //...prvo oslobadjamo ostatak liste
        oslobodi_listu(l->sl);

        //Prvo se oslobadja poslednji element liste
        printf("Oslobadjam %d\n", l->br);

        //...pa tek onda pocetak
        free(l);
    }
}

main()
{
    CVOR* l = NULL;
    int i;

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

    printf("Ispis liste: \n");
    ispisi_listu(l);
    putchar('\n');

    printf("Ispis liste u obrnutom poretku: \n");
    ispisi_listu_unazad(l);
    putchar('\n');

    oslobodi_listu(l);
}
```

1.3 Ubacivanje elementa na kraj jednostruko povezane liste

1. Napisati program koji kreira jednostruko povezanu listu. Napisati:
   - funkciju koja kreira jedan čvor liste,
     `CVOR* napravi_cvor(int br)`
   - **iterativnu i rekurzivnu funkciju za ubacivanje broja na kraj liste pomoću pokazivača na početak liste**,
     `void ubaci_na_kraj(CVOR** pl, int br)`
   - funkciju za štampanje liste,
     `void ispisi_listu(CVOR* l)`
   - funkciju za oslobađanje liste,
     `void oslobodi_listu(CVOR* l)`.

```
/* Ubacivanje na kraj jednostruko povezane liste, verzija sa
   pokazivacem na pocetak liste.
   - iterativna i rekurzivna verzija */

#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;
    novi->sl = NULL;
    return novi;
}
```

```
/* Ubacuje dati broj na kraj liste.
   Pokazivac na pocetak liste se prenosi preko pokazivaca, umesto po
   vrednosti, kako bi mogla da mu se izmeni vrednost.
   Iterativna verzija funkcije */

void ubaci_na_kraj(CVOR** pl, int br)
{
    //Kreiramo novi cvor
    CVOR* novi = napravi_cvor(br);

    //Kako ce on predstavljati novi kraj liste postavljamo polje
    //sledeci na NULL
    novi->sl = 0;

    //Ako je lista bila prazna sada se sastoji od samo jednog elementa
    if (*pl == NULL)
        *pl = novi;
    else
    {
        //Inace, pronalazimo poslednji element liste
        CVOR* t;
        for (t=*pl; t->sl!=NULL; t=t->sl)
            ;

        //Uvezujemo novi cvor na kraj liste
        t->sl = novi;
    }

}

/* Rekurzivna varijanta prethodne funkcije */
void ubaci_na_kraj_rekurzivno(CVOR** pl, int br)
{
    if (*pl == NULL)
    {
        CVOR* novi = napravi_cvor(br);
        novi->sl = 0;
        *pl = novi;
    }
    else
        //Ako lista nije prazna ubacujemo element na kraj ostatka
        //liste
        ubaci_na_kraj_rekurzivno( &((*pl)->sl) ,br);
}
```

```
/* Ispisivanje liste : iterativna verzija */
void ispisi_listu(CVOR* l)
{
    CVOR* t;
    for (t = l; t != NULL; t=t->sl)
        printf("%d ", t->br);
}


/* Iterativna verzija funkcije koja oslobadja listu */
void oslobodi_listu(CVOR* l)
{
    while (l)
    {
        CVOR* tmp = l->sl;
        free(l);
        l = tmp;
    }
}

main()
{
    CVOR* l = NULL;
    int i;

    for (i = 0; i<5; i++)
        ubaci_na_kraj(&l, i);

    for (; i<10; i++)
        ubaci_na_kraj_rekurzivno(&l, i);

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

    oslobodi_listu(l);
}
```

1.4 Ubacivanje elementa na kraj liste sa eksplicitnim vraćanjem novog početka liste

1. Napisati program koji kreira jednostruko povezanu listu. Napisati:

   - funkciju koja kreira jedan čvor liste,
     `CVOR* napravi_cvor(int br)`

- **iterativnu i rekurzivnu funkciju za ubacivanje broja na kraj liste sa eksplicitnim vraćanjem novog početka liste**,
  `CVOR* ubaci_na_pocetak(CVOR* l, int br)`
- funkciju za štampanje liste,
  `void ispisi_listu(CVOR* l)`
- funkciju za oslobađanje liste,
  `void oslobodi_listu(CVOR* l)`.

```
/* Ubacivanje na kraj jednostruko povezane liste
  - verzija sa eksplicitnim vracanjem pocetka liste
  - iterativna i rekurzivna verzija */

#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;
    novi->sl = sl;
    return novi;
}

/* Funkcija vraca pocetak rezultujuce liste */
CVOR* ubaci_na_kraj(CVOR* l, int br)
{
    CVOR* novi = napravi_cvor(br);
    novi->sl = NULL;

    if (l == NULL)
```

```
        return novi;
    else
    {
        CVOR* t;
        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;
}

/* Ispisivanje liste : iterativna verzija */
void ispisi_listu(CVOR* l)
{
    CVOR* t;
    for (t = l; t != NULL; t=t->sl)
        printf("%d ", t->br);
}


/* Iterativna verzija funkcije koja oslobadja listu */
 void oslobodi_listu(CVOR* l)
 {
    while (l)
    {
        CVOR* tmp = l->sl;
        free(l);
        l = tmp;
    }
```

```
}

main()
{
    CVOR* l = 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);

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

    oslobodi_listu(l);
}
```