Programiranje II

Beleške sa vežbi

Smer *Informatika*

Matematički fakultet, Beograd

Sana Stojanović

15.05.08.

Sadržaj

1 Pretraživanje liste, izbacivanje elementa iz liste, ubacivanje elementa u sortiranu listu

1. Napisati program koji formira jednostruko povezanu listu od celih brojeva koji se unose sa standardnog ulaza. Oznaka za kraj unosa je unesena nula. Elemente ubacivati na početak liste.

```
#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. */
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. */
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;
}

/* 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.*/
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;
    }
}

main()
{
    CVOR* l = NULL;     //Pokazivac na pocetak liste postavljamo na NULL
    int br;             //Ceo broj koji ubacujemo u listu

    /* Pravimo petlju koja ce unositi brojeve u listu dok ne unesemo nulu
       (bez unosenja nule) */
    while(1)
    {
        scanf("%d", &br);
        if (br == 0)
```

```
            break;
        ubaci_na_pocetak(&l, br);
    }

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

    oslobodi_listu(l);
}
```

2. Napisati program koji listu dobijenu na način opisan u prethodnom zadatku pretražuje da li se u njoj nalazi ceo broj unesen sa standardnog ulaza pozivom funkcije

   cvor pronadji(cvor* l, int br).*

```
#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. */
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. */
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;
}

/* 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.*/
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;
    }
}

/* Funkcija koja nalazi element liste koji sadrzi trazeni broj.
   Iterativna verzija. */
```

```
CVOR *pronadji(CVOR *l, int br)
{
    CVOR *t;

    /* petlja u kojoj se pomeramo kroz listu dok ne dodjemo do trazenog
       elementa (ako postoji) ili kraja liste (ako ne postoji) */
    for(t=l; t!=NULL && t->br!=br; t=t->sl)
        ;
    return t;
}

/* Funkcija koja nalazi element liste koji sadrzi trazeni broj.
   Rekurzivna verzija.
   Ako je pokazivac na trenutni element NULL, odnosno ako broj nije nadjen,
   ili ako trenutni element sadrzi trazeni broj, vraca se trenutni
   pokazivac, a u suprotnom vraca se ono sto pretraga nadje pocevsi od
   sledeceg elementa. */
CVOR *pronadji_rekurzivno(CVOR *l, int br)
{
    if(l==NULL || l->br==br)
            return l;
    return pronadji_rekurzivno(l->sl,br);
}

main()
{
    CVOR* l = NULL;     //Pokazivac na pocetak liste postavljamo na NULL
    int br;             //Ceo broj koji ubacujemo u listu

    /* Pravimo petlju koja ce unositi brojeve u listu dok ne unesemo nulu
       (bez unosenja nule) */
    while(1)
    {
        scanf("%d", &br);
        if (br == 0)
            break;
        ubaci_na_pocetak(&l, br);
    }

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

    printf("Unesite broj za koji zelite da proverite da li se nalazi u
            listi\n");
    scanf("%d", &br);
```

```
    if (pronadji(l, br) != NULL)
        printf("Element se nalazi u listi.\n");
    else
        printf("Element se ne nalazi u listi.\n");

    oslobodi_listu(l);
}
```

3. Napisati program koji listu dobijenu na način opisan u prethodnom zadatku pretražuje da li se u njoj nalazi ceo broj unesen sa standardnog ulaza i ako se nalazi briše ga iz liste. Za ovo koristiti funkciju *void obrisi(cvor** l, int br).*

```
#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. */
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. */
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;
}

/* 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.*/
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;
    }
}

/* Funkcija koja brise iz liste element koji sadrzi trazeni broj. */
void obrisi(CVOR **pl, int br)
{
    CVOR *t;
```

```
    /* AKo je lista prazna nemamo sta da brisemo. */
    if(*pl == NULL)
        return;

    /* Ako je prvi element koren brisanje je specificno, pa ga
       izvodimo odvojeno od ostalih slucajeva. */
    if((*pl)->br==br)
    {
        /* Cuva se pokazivac na prvi element. */
        CVOR *temp=*pl;

        /* Drugi element postaje prvi. */
        *pl=(*pl)->sl;

        /* Brisemo element koji je do sada bio prvi. */
        free(temp);
        return;
    }

    /* Trazimo element koji prethodi elementu koji sadrzi broj. */
    for(t=*pl; t->sl!=NULL && t->sl->br!=br; t=t->sl)
        ;

    /* Ako smo dosli do kraja, onda se broj ne nalazi u listi,
       pa nemamo ni sta da brisemo. U suprotnom...*/
    if(t->sl!=NULL)
    {
        /* Ako nismo dosli do kraja, petlja je prekinuta jer
           smo nasli prethodnika trazenog cvora, pa ga brisemo. */
           CVOR *temp=t->sl;
           t->sl=temp->sl;
           free(temp);
    }
}

main()
{
    CVOR* l = NULL;     //Pokazivac na pocetak liste postavljamo na NULL
    int br;             //Ceo broj koji ubacujemo u listu

    /* Pravimo petlju koja ce unositi brojeve u listu dok ne unesemo nulu
       (bez unosenja nule) */
    while(1)
    {
        scanf("%d", &br);
        if (br == 0)
```

```
            break;
        ubaci_na_pocetak(&l, br);
    }

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

    printf("Unesite broj koji zelite da izbacite iz liste:\n");
    scanf("%d", &br);
    obrisi(&l, br);

    printf("Nakon izbacivanja elementa iz liste lista izgleda ovako:\n");
    ispisi_listu(l);
    putchar('\n');

    oslobodi_listu(l);
}
```

4. Napisati program koji kreira sortiranu jednostruko povezanu listu. Elementi liste su celi brojevi koji se unose sa standardnog ulaza sa nulom kao oznakom za kraj.

```
/* Ubacivanje na odgovarajuce mesto sortirane 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. */
CVOR* napravi_cvor(int br) {
    CVOR* novi = (CVOR*)malloc(sizeof(CVOR));

    if (novi == NULL)
    {
        printf("Greska prilikom alokacije memorije\n");
        exit(1);
    }
```

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

/* Prilikom ubacivanja elementa u sortiranu listu potrebno je pronaci
   poslednji element liste koji je manji od novog elementa */
void ubaci_sortirano(CVOR** pl, int br)
{
    CVOR* novi = napravi_cvor(br);
    CVOR* t;

    /* U slucaju kada je lista prazna, kao i u slucaju kada je novi
       element manji od svih elemenata liste ubacujemo ga na pocetak */
    if (*pl == NULL || br < (*pl)->br)
    {
        novi->sl = *pl;
        *pl = novi;
        return;
    }

    /* Ako lista nije prazna i ako novi element nije manji od svih
       elemenata liste onda trazimo poslednji element liste koji je manji
       od novog elementa. Krecemo od pocetka liste i krecemo se kroz listu
       sve dok ne nadjemo poslednji element koji je manji od novog elementa
       ili dok ne dodjemo do kraja liste */

    /* Obratiti paznju na redosled uslova t->sl!=NULL i t->sl->br<br.
       Zbog lenjog izracunavanja, redosled MORA biti kao sto je naveden */
    for (t = *pl; t->sl!=NULL && t->sl->br < br; t=t->sl)
        ;

    /* Kada nadjemo takav element, novi element treba ubaciti u listu
       odmah iza njega */
    novi->sl = t->sl;
    t->sl = novi;
}

/* Rekurzivna verzija prethodne funkcije */ void
ubaci_sortirano_rekurzivno(CVOR** pl, int br)
{
    /* Ako je lista bila prazna ili ako je novi element manji od
       svih, ubacujemo ga na pocetak */
    if (*pl == NULL || br < (*pl)->br)
    {
        CVOR* novi = napravi_cvor(br);
```

```
        novi->sl = *pl;
        *pl = novi;
        return;
    }

    /* U suprotnom, element rekurzivno ubacujemo u ostatak liste */
    ubaci_sortirano_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 br;

    /* Pravimo petlju koja ce unositi brojeve u listu dok ne unesemo nulu
       (bez unosenja nule) */
    while(1)
    {
        scanf("%d", &br);
        if (br == 0)
            break;
        ubaci_sortirano(&l, br);
    }

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

```
    oslobodi_listu(l);
}
```

2 Funkcije za rad sa redom, ubacivanje elementa na kraj reda, izbacivanje elementa sa početka reda

1. Napisati program koji implementira red pomoću jednostruko povezane liste. Napisati funkcije za ubacivanje elementa na kraj reda i za izbacivanje elementa sa početka reda.

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

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

/* Pomocna funkcija koja kreira cvor liste sa datim sadrzajem. */
CVOR* napravi_cvor(int br) {
    CVOR* novi = (CVOR*)malloc(sizeof(CVOR));
    if (novi == NULL)
    {
        printf("Greska prilikom alokacije memorije\n");
        exit(1);
    }
    novi->br = br;
    novi->sl = NULL;
    return novi;
}

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

/* Oslobadjanje liste : iterativna verzija */
void oslobodi_listu(CVOR* l)
{
    while (l)
```

```
    {
        CVOR* tmp = l->sl;
        free(l);
        l = tmp;
    }
}

/* Ubacuje dati broj na pocetak liste. */
void ubaci_na_pocetak(CVOR** pl, int br)
{
    CVOR* novi = napravi_cvor(br);
    novi->sl = *pl;
    *pl = novi;
}

/* Funkcija koja dodaje broj u red.
   Kao argumente prima adrese i prvog i poslednjeg elementa liste.
   Ovo je potrebno u slucaju dodavanja prvog elementa redu koji je tada i
   prvi i poslednji. */
void dodaj(CVOR **prvi, CVOR ** poslednji, int br)
{
    CVOR* novi = napravi_cvor(br);
    novi->sl = NULL;

    /* Ako je lista prazna onda su oba pokazivaca postavljena na NULL */
    if(*poslednji==NULL)
    {
        *poslednji = novi;
        *prvi=*poslednji;
    }
    else
    {
        /* U suprotnom vrsimo ubacivanje na kraj liste */
        (*poslednji)->sl = novi;
        *poslednji = novi;
    }
}

/* Funkcija koja dodaje broj u red moze se izvrsiti i pozivom
   funkcije ubaci_na_pocetak.

    void dodaj(CVOR **prvi, CVOR ** poslednji, int br)
    {
        // Ako nema poslednjeg, nema ni reda, pa mozemo
        // izvrsiti ubacivanje na pocetak, sto je u isto
        // vreme i kraj.
```

```
        if(*poslednji==NULL)
        {
            ubaci_na_pocetak(poslednji, br);
                *prvi=*poslednji;
        }
        else
        {
            // U suprotnom vrsimo ubacivanje na pocetak liste
            // na koju pokazuje pokazivac na sledeci element
            // od poslednjeg (sto je naravno prazna lista).
            ubaci_na_pocetak(&(*poslednji)->sl,br);

            // Sada poslednji element postaje upravo ovaj koji
            // smo upravo ubacili.
            *poslednji=(*poslednji)->sl;
        }
    }
*/


/* Funkcija vraca pokazivac na prvi element reda i skida ga sa reda. */
CVOR* izbaci(CVOR **prvi)
{
    CVOR *tmp;

    /* Ako je lista bila prazna vracamo NULL */
    if(*prvi==NULL)
        return NULL;

    /* U suprotnom vracamo pokazivac na prvi element liste i izbacujemo
       ga iz liste */

    /* Cuvamo prvi element u pomocnoj promenljivoj */
    tmp=*prvi;

    /* Menjamo pocetak liste */
    *prvi=tmp->sl;

    /* Vracamo pokazivac u kome je sacuvan izbaceni element liste */
    return tmp;
}

main()
{
    /* Cuvamo pokazivace na prvi i na poslednji element reda.
       Oba pokazivaca su na pocetku NULL */
```

```
    CVOR *prvi = NULL;
    CVOR *poslednji = NULL;

    int i;
    for (i = 1; i<10; i++)
        dodaj(&prvi, &poslednji, i);

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

    for (i = 0; i<3; i++)
    {
        CVOR *t;
        t=izbaci(&prvi);
        printf("Iz reda je izbacen broj %d.\n",t->br);
        free(t);
    }

    ispisi_listu(prvi);

    putchar('\n');

    oslobodi_listu(prvi);
}
```