

# Zadaci za zadaću

- Napisati funkcije za dodavanje i skidanje elementa u stog realiziran poljem u koje stane najviše MAXZ zapisa. U glavnom programu izvesti učitavanje podataka za element stoga s tastature kao i ispisivanje elementa stoga koji je skinut (npr. unijeti nekoliko elemenata u stog, a zatim skinuti i ispisati svaki element sve dok stog ne ostane prazan). Element stoga je zapis o artiklu koji se prodaje u dućanu:
  - šifra artikla (int)
  - naziv artikla (25 znakova)
  - cijena (float)

Rok za predaju zadaće: 22. 12 2006.

- U memoriji oblikovati vezanu listu (pomoću pokazivača). U pojedini čvor liste upisati:
  - matični broj osobe (int)
  - prezime (30 znakova)
  - ime (30 znakova)
  - godina rođenja (short)
  - pokazivač na sljedeći čvor

Unos podataka ostvariti učitavanjem iz ulazne datoteke. Elemente liste složiti po ključu matični broj, tako da su uzlazno sortirani. Napisati funkciju koja će iz liste ispisati sve zapise o osobama rođenim određene godine koja se unosi tijekom izvršavanja programa.

Rok za predaju zadaće: 22. 12 2006.

# Implementacija stoga pomoću pokazivača

- Napisati program za realizaciju stoga pomoću pokazivača: iz ulazne datoteke se čitaju podaci i upisuju u stog. Ispisati vrijednosti elemenata u stogu i lokacije na kojima se nalaze. Nakon toga skidati podatke iz stoga sve dok ne ostane prazan.

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
typedef int tip;
struct cv {
    tip element;
    struct cv *sljed;
};
typedef struct cv cvor;

cvor *dodaj (cvor *vrh, tip element) {
    cvor *novi;
    if ((novi = (cvor *) malloc(sizeof(cvor))) != NULL) {
        novi->element = element;
        novi->sljed = vrh;
        printf("Na adresu %p dodao sam %d, a sljedeci je %p\n",novi, element, vrh);
    }
    return novi;        // vrati pokazivac na novi cvor
}
```

```

cvor *skini (cvor *vrh, int *element) {
    cvor *pom;

    *element = vrh->element;
    printf ("S adrese %p ", vrh);
    pom = vrh->sljed;
    free (vrh);                // oslobodi vrh
    return pom;                // vrati novi vrh
}

```

```

void main (void) {
    FILE *fi;
    int j;
    int element;              // element stoga
    cvor *vrh, *p;// pokazivac na vrh i pomocni pokazivac

    fi = fopen ("UlazZaStog.txt", "r");
    if (fi) {
        vrh = NULL;
        j = 0;
        while (fscanf (fi, "%d", &element) != EOF) {
            j++;
            if ((p = dodaj (vrh, element)) != NULL) {
                vrh = p;
                printf ("%d. ulazni podatak je %d\n", j, vrh->element);
            }
        }
    }
}

```

```
    else {
        printf("Nema vise mjesta za stog\n");
        break;
    }
}
fclose (fi);
p = vrh;
    // Skidanje elemenata sa stoga
while (vrh) {
    vrh = skini (vrh, &element);
    printf ("skinuo sam element %d\n", element);
}
}
else {
    printf ("Nema ulazne datoteke\n");
    exit (1);
}
system("PAUSE");
exit(0);
}
```

## Izvedba reda pomoću polja

- Napisati program u kojem se red realizira upotrebom polja. Efikasan način realizacije reda statičkom strukturom je jednodimenzionalno polje zadane podatkovne strukture koje se koristi cirkularno. Koriste se dva indeksa (ulaz i izlaz), a cirkularnost se ostvaruje uporabom operatora modulo ( $\%$ ) (vidjeti predavanja). Ulazni podaci čitaju se iz datoteke dok se red ne popuni. Ukoliko ima još elemenata u ulaznoj datoteci, izbrišu se postojeći elementi iz reda, te se upisuju novi dokle god ima podataka u ulaznoj datoteci. Prije završetka izvršavanja programa izbrisati sve elemente reda. Ispisati broj elemenata u redu i element na kojem se obavlja operacija nakon svake operacije.

```

#include <stdlib.h>
#include <stdio.h>
#define MAXRED 10
typedef int tip;

// dodaje element u polje red od max n clanova, mijenja ulaz, tj straznji kraj
// vraca 1 ako ima mjesta u redu, inace 0
int DodajURed (tip element, tip red[], int n, int izlaz, int *ulaz) {
    if (((*ulaz+1) % n) == izlaz) return 0;
    (*ulaz)++;
    *ulaz %= n;
    red [*ulaz] = element;
    return 1;
}

// logicki uklanja element iz polja red od max n clanova, mijenja izlaz, tj prednji kraj
// vraca 1 ako ima clanova u redu, inace 0
int SkiniIzReda (tip *element, tip red[], int n, int *izlaz, int ulaz) {
    if (ulaz == *izlaz) return 0;
    (*izlaz) ++;
    *izlaz %= n;
    *element = red[*izlaz];
    return 1;
}

```

```

// vraca broj elemenata u redu
int prebroji (int n, int izlaz, int ulaz) {
    if (ulaz >= izlaz) {
        return (ulaz - izlaz);           // standardno
    } else {
        return (ulaz - izlaz + n);      // cirkularnost
    }
}

void main (void) {
    int red[MAXRED];
    int element, ulaz, izlaz;
    FILE *fi;

    ulaz = 0; izlaz = 0;
    fi = fopen ("UlazZaRed.txt", "r");
    if (fi) {
        while (fscanf (fi, "%d", &element) != EOF) {
            if ((DodajURed (element, red, MAXRED, izlaz, &ulaz))) {
                printf ("U red dodan element %d\n", element);
                printf ("Broj elemenata u redu je %d\n",prebroji (MAXRED, izlaz, ulaz));
            } else {
                printf ("Nema vise mjesta u redu\n\n\n");
            }
        }
    }
}

```

```

// uklanjanje iz reda
while (SkiniIzReda (&element, red, MAXRED, &izlaz, ulaz)) {
    printf ("Iz reda skinut element %d\n", element);
    printf ("Broj elemenata u redu je %d\n",prebroji (MAXRED, izlaz, ulaz));
    }
    printf("\n");
}
}
fclose (fi);

// uklanjanje preostalih elemenata
printf("\n\n");
while (SkiniIzReda (&element, red, MAXRED, &izlaz, ulaz)) {
    printf ("Iz reda skinut element %d\n", element);
    printf ("Broj elemenata u redu je %d\n",prebroji (MAXRED, izlaz, ulaz));
}

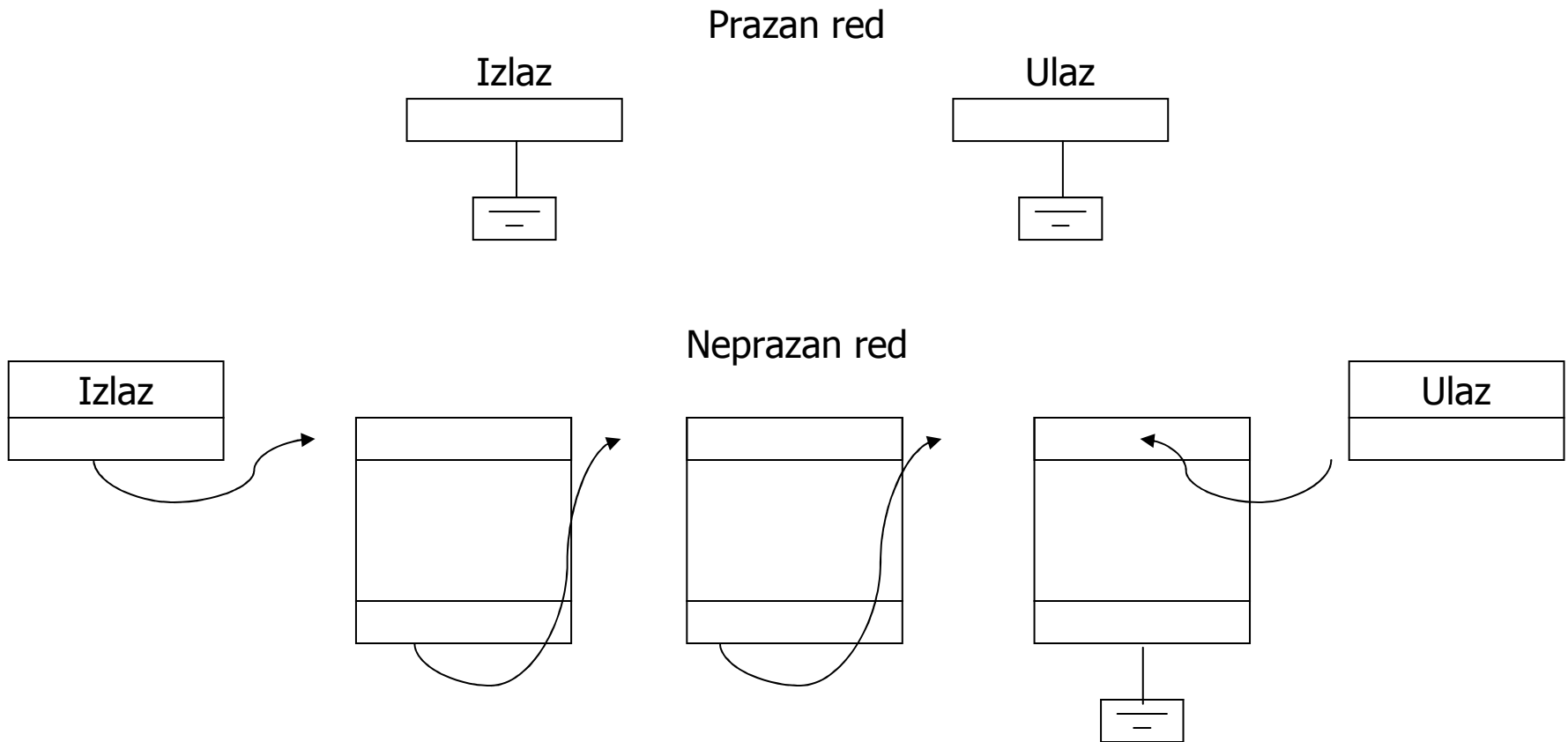
} else {
    printf ("Nema ulazne datoteke\n");
    exit (1);
}
system("PAUSE");
exit (0);
}

```



# Izvedba reda pomoću pokazivača

- Napisati program za izvedbu reda pomoću pokazivača. Upotrijebiti generator slučajnih brojeva za punjenje reda: neparni broj se upisuje na kraj reda (ulaz), a ako je generiran parni broj briše se iz reda prvi broj na početku reda (izlaz).



```

#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <ctype.h>
struct cv {
    int element;
    struct cv *sljed;
};
typedef struct cv cvor;

// dodaje element u red, vraca 1 ako uspije, inace 0
int DodajURed (int element, cvor **ulaz, cvor **izlaz) {
    cvor *novi;
    if (novi = malloc (sizeof (cvor))) {
        novi->element = element;
        novi->sljed = NULL;
        if (*izlaz == NULL) {
            *izlaz = novi;          // ako je red bio prazan
        } else {
            (*ulaz)->sljed = novi; // inace, stavi na kraj
        }
        *ulaz = novi;      //zapamti zadnjeg
        return 1;
    }
    return 0;
}

```

```

// uklanja element iz reda, vraca 1 ako uspije, inace 0
int SkiniIzReda (int *element, cvor **ulaz, cvor **izlaz) {
    cvor *stari;
    if (*izlaz) {
        // ako red nije prazan
        *element = (*izlaz)->element; // element koji se skida
        stari = *izlaz; // zapamti trenutni izlaz
        *izlaz = (*izlaz)->sljed; // novi izlaz
        free (stari); // oslobodi memoriju skinutog
        if (*izlaz == NULL) *ulaz = NULL; // prazan red
        return 1;
    }
    return 0;
}

```

```

// vraca broj elemenata u redu
int Prebroji (cvor *izlaz) {
    int n;
    for (n = 0; izlaz; izlaz = izlaz->sljed) {
        printf ("%d -> ", izlaz->element);
        n++ ;
    }
    printf ("NULL\n");
    return n;
}

```

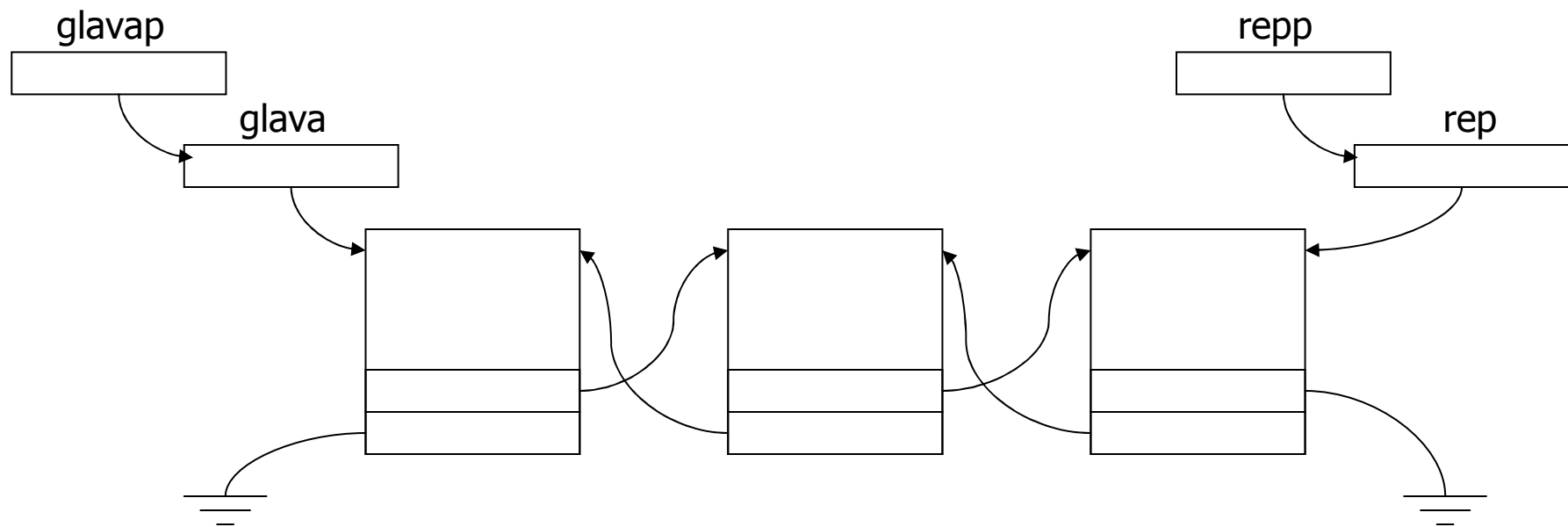
```

void main (void) {
    int broj;
    cvor *ulaz = NULL;          // krajevi
    cvor *izlaz = NULL;
    printf ("Generiraju se slucajni nenegativni cijeli brojevi.\n");
    printf ("Neparni brojevi upisuju se u red, a parni broj znaci skidanje iz reda\n");
    printf ("Za obavljanje jednog koraka pritisnuti ENTER, za kraj bilo koji znak\n");
    srand ((unsigned) time (NULL));
    while (isspace(getchar())) {
        broj = rand ();
        if (broj%2) {           // Neparne upisujemo u red
            printf ("U red se upisuje broj %d\n", broj);
            if (!DodajURed (broj, &ulaz, &izlaz))
                printf("Nema vise memorije\n");
        } else {               // Parni broj znaci skidanje iz reda
            if (SkiniIzReda (&broj, &ulaz, &izlaz)) {
                printf ("Iz reda je skinut podatak %d\n", broj);
            } else {
                printf("Red je prazan\n");
            }
        }
    }
    printf ("Broj elemenata u redu: %d\n", Prebroji (izlaz));
}
system("PAUSE");
}

```

## Izvedba reda kao dvostruko povezane liste

- Radi bržeg traženja u oba smjera kretanja po listi, ona može biti dvostruko povezana. Svaki čvor osim elementa s podacima, sadrži pokazivač na sljedeći čvor i pokazivač na prethodni čvor.
- Lista ima *glavu* i *rep*, što je prikladno za izvedbu reda.
- funkcije za dodavanje i skidanje rukuju s pokazivačima na glavu (*glavap*) i rep (*repp*)



- Napisati program za realizaciju reda kao dvostruko povezane općenite liste s funkcijama koje upisuju novi element na kraj reda i skidaju prvi element na čelu reda. Također napisati funkciju za dvostruko povezanu listu koja skida proizvoljan element iz liste tražeći vrijednost elementa koji skida. Punjenje i pražnjenje reda ostvariti upotrebom generatora slučajnih brojeva gdje se neparni broj upisuje na kraj reda, a parni broj znači skidanje prvog člana u redu. Ispisati red nakon svake operacije. Na kraju obrisati član reda čija se vrijednost učita s tastature (to nije funkcija za red nego za listu).

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

```
struct cv2 {
    int element;
    struct cv2 *sljed;
    struct cv2 *preth;
};
typedef struct cv2 cvor2;
```

```

// dodavanje u red realizirano dvostruko povezanom listom
// funkcija vraca 1 ako uspije, inace 0
int DodajURed (int element, cvor2 **glavap, cvor2 **repp) {
    cvor2 *novi;

    if (novi = malloc (sizeof (cvor2))) {
        novi->element = element;
        novi->sljed = NULL;
        novi->preth = NULL;

        if (*glavap == NULL) { // Ako je red bio prazan
            *glavap = novi; *repp = novi;
        }
        else { // inace, stavi na kraj
            (*repp)->sljed = novi;
            novi->preth = *repp;
            *repp = novi;
        }
        return 1;
    }
    return 0;
}

```

```

// skidanje iz reda, funkcija vraca 1 ako uspije, inace 0
int SkiniIzReda (int *element, cvor2 **glavap, cvor2 **repp) {
    cvor2 *stari;

    if (*repp) {        // provjera da li je red prazan
        *element = (*glavap)->element; // vrati element
        if (*glavap == *repp) {        // Ako je samo jedan clan
            stari = *glavap;
            *glavap = NULL; *repp = NULL;
        } else {                    //inace, povezi ih
            (*glavap)->sljed->preth = NULL; // prvi u redu nema prethodnika
            stari = *glavap;
            *glavap = stari->sljed; // nova glava je sljedbenik stare
        } free (stari);
        return 1;
    }
    return 0;
}

// ispis reda
void IspisiRed (cvor2 *glava) {
    for (; glava; glava = glava->sljed)
        printf ("%d ", glava->element);
    printf ("\n");
}

```



```

// brisanje iz reda clana sa zadanim kljucem
int BrisiIzReda (cvor2 **glavap, cvor2 **repp, int element) {
    cvor2 *pom;

    if (*glavap) { // neprazan red
        for (pom = *glavap; pom && (pom->element != element); pom = pom->sljed)
            ;
        if (pom) { // Ako je nadjen,
            if (pom == *glavap) { // ako je prvi
                *glavap = pom->sljed;
                if (pom->sljed) { // ako nije jedini
                    pom->sljed->preth = NULL;
                } else { //ako jest jedini
                    *glavap = NULL; *repp = NULL;
                }
            } else if (pom == *repp) { // ako je zadnji, ali ne i jedini
                (*repp)->preth->sljed = NULL; // predzadnji postaje zadnji
                *repp = (*repp)->preth;
            } else { // nije ni prvi ni zadnji
                pom->preth->sljed = pom->sljed;
                pom->sljed->preth = pom->preth;
            } free (pom);
            return 1;
        }
    }
    return 0; // Nije nadjen ili lista prazna
}

```

```

void main (void) {
    cvor2 *glava = NULL;    // glava reda
    cvor2 *rep = NULL;     // rep reda
    int broj;

    printf ("Generiraju se slucajni nenegativni cijeli brojevi.\n");
    printf ("Neparni brojevi upisuju se u red, a parni broj simulira skidanje iz reda\n");
    printf ("Za obavljanje jednog koraka pritisnuti ENTER, za kraj bilo koji znak\n");

    srand ((unsigned) time (NULL));
    while (isspace(getchar())) {
        broj = rand ();
        if (broj%2) { // Neparne upisujemo u red
            printf ("U red se upisuje broj %d\n", broj);
            if (!DodajURed (broj, &glava, &rep))
                printf("Nema vise memorije\n");
        } else {
            // Parni broj znaci skidanje iz reda
            if (SkiniIzReda (&broj, &glava, &rep)) {
                printf ("Iz reda je skinut podatak %d\n", broj);
            } else {
                printf("Red je prazan\n");
            }
        }
    }
    IspisiRed (glava);
}

```

```
// brisanje iz reda bilo kojeg clana
printf("\nSad cemo red tretirati kao listu, brise se bilo koji clan\n");
while (1) {
    IspisiRed (glava);
    printf ("Upisite podatak koji se brise iz reda >");
    scanf ("%d", &broj);
    if (!BrisiIzReda (&glava, &rep, broj))
        break;
}
system("PAUSE");
exit (1);
}
```

## Zadatak za zadaću

- Strukturu podataka red realizirati uz upotrebu pokazivača. Svaki element reda sadrži naziv (20 znakova) i vrijednost (realni broj). Treba napisati funkciju koja dodaje i funkciju koja briše element iz reda, u kojima treba ispisati element na kojem se obavlja operacija. Također napisati funkciju koja ispiše sve elemente u redu.

Rok za predaju zadaće: 10. 01. 2007.