Ulančane liste

JEDNOSTRUKO ULANČANE LISTE

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

Obilazak liste (prebrojavanje elemenata)

```
int duz (Elem *lst) {
    Elem *tek;
    int n=0;
    for (tek=lst;tek;tek=tek->sled) n++;
    return n;
}
```

Ispisivanje elemenata liste

```
void pisi (Elem *lst) {
    for (tek=lst;tek;tek=tek->sled)
        printf("%d\n",tek->broj);
}
```

Dodavanje elementa na početak liste

```
Elem* na_pocetak (Elem *lst, int b) {
    Elem *novi=malloc(sizeof(Elem));
    novi->broj=b;
    novi->sled=lst;
    lst=novi;
    return lst;
}
```

Dodavanje elementa na kraj liste

```
Elem* na_kraj (Elem *lst, int b) {
    Elem *tek, *novi=malloc(sizeof(Elem));
    novi->broj=b;
    novi->sled=NULL;
    if (!lst)
        lst=novi;
    else {
        for(tek=lst;tek->sled;tek=tek->sled);
        tek->sled=novi;
    }
    return lst;
}
```

Umetanje elementa u neopadajuću listu

```
Elem* umetni (Elem *lst, int b) {
    Elem *tek=lst, *pret=NULL, *novi;
    while (tek && tek->broj<b) {
        pret=tek; tek=tek->sled;
    }
    novi=malloc(sizeof(Elem));
    novi->broj=b;
    novi->sled=tek;
    if (!pret)
        lst=novi;
    else
        pret->sled=novi;
    return lst;
}
```

Sortiranje liste u neopadajućem poretku (metodom izbora)

```
void sortiraj (Elem *lst) {
    Elem *i, *j;
    int p;
    for (i=lst;i;i=i->sled)
        for (j=lst;j;j=j->sled)
            if (j->broj < i->broj) {
                p=i->broj;
                i->broj=j->broj;
                j->broj=p;
            }
}
```

Brisanje svih elemenata liste

```
void brisi (Elem *lst) {
    Elem *stari;
    while (lst) {
        stari=lst;
        lst=lst->sled;
        free(stari);
    }
}
```

Izostavljanje elementa sa zadatom vrednošću

```
Elem* izostavi (Elem *lst, int b) {
    Elem *tek=lst, *pret=NULL, *stari;
    while (tek) {
        if (tek->broj!=b) {
            pret=tek; tek=tek->sled;
        }
```

```
        else { /* element se izbacuje */
            stari=tek; tek=tek->sled;
            if (!pret) lst=tek;
            else pret->sled=tek;
        free(stari);
    }
    return lst;
}
```

DVOSTRUKO ULANČANE LISTE

```
typedef struct elem {
    int broj;
    struct elem *sled, *pret;
} Elem;

typedef struct {Elem *prvi, *posl} Lista;
```

Prazna lista

```
Lista lst={NULL,NULL};
```

Obilazak liste u napred

```
for (tek=lst.prvi;tek;tek=tek->sled);
```

Obilazak liste u nazad

```
for (tek=lst.posl;tek;tek=tek->pret);
```

Ispisivanje liste u napred

```
void pisi_unapred (Lista lst) {
    Elem *tek;
    for (tek=lst.prvi;tek;tek=tek->sled) {
        printf("%d\n",tek->broj);
}
```

Ispisivanje liste u nazad

```
void pisi_unazad (Lista lst) {
    Elem *tek;
    for (tek=lst.posl;tek;tek=tek->pret) {
        printf("%d\n",tek->broj);
}
```

Nalaženje prve pojave broja

```
Elem* nadji_prvi (Lista lst, int b) {
    Elem *tek;
    for (tek=lst.prvi;tek && tek->broj!=b;tek=tek->sled);
    return tek;
}
```

Nalaženje poslednje pojave broja

```
Elem* nadji_posl (Lista lst, int b) {
    Elem *tek;
    for (tek=lst.posl;tek && tek->broj!=b;tek=tek->pret);
    return tek;
}
```

Dodavanje novog elementa na početak liste

```
Lista na_pocetak (Lista lst, int b) {
    Elem *novi=malloc(sizeof(Elem));
    novi->broj=b;
    novi->sled=lst.prvi;
    novi->pret=NULL;
    if (!lst.posl) lst.posl=novi;
    else lst.prvi->pret=novi;
    lst.prvi=novi;
    return lst;
}
```

Dodavanje novog elementa na kraj liste

```
Lista na_kraj (Lista lst, int b) {
    Elem *novi=malloc(sizeof(Elem));
    novi->broj=b;
    novi->pret=lst.posl;
    novi->sled=NULL;
    if (!lst.prvi) lst.prvi=novi;
    else lst.posl->sled=novi;
    lst.posl=novi;
    return lst;
}
```

Izostavljanje prve pojave zadatog broja

```
void izostavi_prvi (Lista *plst, int b) {
    Elem *tek=plst->prvi;
    while (tek && tek->broj!=b)
        tek=tek->sled;
    if (tek) {
        if (!tek->pret) /* izbacuje se prvi */
            plst->prvi=tek->sled;
```

```
        else tek->pret->sled=tek->sled;
        if (!tek->sled) /* izbacuje se poslednji */
            plst->posl=tek->pret;
        else tek->sled->pret=tek->pret;
        free(tek);
    }
}
```

Brisanje svih elemenata liste

```
void brisi (Lista *plst) {
    Elem *tek=plst->prvi, *stari;
    while (tek) {
        stari=tek;
        tek=tek->sled;
        free(stari);
    }
    plst->prvi=plst->posl=NULL;
}
```