

## Prikaz liste pomoću pokazivača

- Riješiti isti problem kao u prethodnom zadatku (napraviti listu cijelih brojeva u koju se upisuje proizvoljan broj slučajno generiranih cijelih brojeva, ubaciti novi član na kraj i sredinu liste te izbrisati izabrani član, članove koji se nalaze ispred i nakon prethodno izbrisanog, te na kraju izbrisati cijelu listu. Nakon svake od ovih operacija ispisati sadržaj liste) uz prikaz liste preko pokazivača koji je objašnjen na predavanjima.

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

typedef int elementtype;

struct celltype {
    elementtype element;
    struct celltype * next;
};

typedef struct celltype * LIST;
typedef struct celltype * position;

position End(LIST L) {
    position q = L;
    while(q->next != NULL)
        q = q->next;
    return q;
}

LIST MakeNewNull() {
    LIST L = malloc(sizeof(struct celltype));
    L->next = NULL;
    return L;
}
```

```
void Insert(elementtype x, position p) {
    position temp;
    temp = p->next;
    p->next = malloc(sizeof(struct celltype));
    p->next->element = x;
    p->next->next = temp;
}
```

```
void Delete(position p) {
    position temp;
    temp = p->next;
    p->next = p->next->next;
    free(temp);
}
```

```
position First(LIST L) {
    return L;
}
```

```
position Next(position p) {
    if (p->next != NULL)
        return p->next;
    else
        return p;
}
```

```
position Previous(position p, LIST L) {
    position q = L;
    while(q->next != p)
        q = q->next;
    return q;
}
```

```
elementtype Retrieve(position p) {
    return p->next->element;
}
```

```
void PrintElement(position p) {
    printf("%i ", p->next->element);
}
```

```
void PrintListElements(LIST L) {
    position q = L;
    printf("<");
    while (q->next != NULL) {
        printf("%i ", q->next->element);
        q = q->next;
    }
    printf(">");
}
```

```
void ProcessList(LIST L, void (*pf)(position p, elementtype e)){
    position q = L;
    while (q->next != NULL)
    {
        (*pf)(q, q->next->element);
        q = q->next;
    }
}
```

```
void print_el(position p, elementtype e){
    printf("%i ", e);
}
```

```

int main() {
    LIST ls;
    position pos,p2;
    elementtype clan;
    int len, mjesto, ind;
    ls = MakeNewNull();
    pos = ls;
    PrintListElements(ls);
    printf ("Koliko clanova zelite upisati u listu\n");
    scanf("%d",&len);
    printf("\n");
    srand(time(NULL));
    for (mjesto=0; mjesto < len; mjesto++) {
        clan = (elementtype) 100 * ((float)rand() / (RAND_MAX + 1));
        Insert(clan, pos);
        pos = Next(pos);
    }
    PrintListElements(ls);
    printf("\n");
    printf ("Clan koji zelite upisati na zadnje mjesto u listu\n");
    scanf("%d",&clan);
    printf("\n");
    Insert(clan,End(ls));
    PrintListElements(ls);
    printf("\n");
}

```

```
printf ("Clan koji zelite upisati u sredinu liste\n");
scanf("%d",&clan);
printf("\n");
pos=First(ls);
len=0;
while (pos != End(ls)) {
    printf("Clan na mjestu %d je %d\n",len,Retrieve(pos));
    pos=Next(pos);
    len++;
}
mjesto=len/2;

p2=First(ls);
for (ind=0; ind < mjesto; ind++)
    p2=Next(p2);
Insert(clan,p2);
PrintListElements(ls);
printf("\n");
```

```
printf ("Koju poziciju zelite izbrisati iz liste\n");
scanf("%d",&mjesto);
p2=First(ls);
for (ind=0; ind < mjesto; ind++)
    p2=Next(p2);
printf("Brisem clan s vrijednoscu %d\n",Retrieve(p2));
Delete(p2);
PrintListElements(ls);
printf("\n");
```

```
printf("A sad cu izbrisati clan prije i poslije izbrisanog\n");
if (p2 != First(ls)) Delete(Previous(p2,ls));
PrintListElements(ls);
printf("\n");
if (mjesto != len) {
    pos = Previous(Next(p2),ls);
    Delete(pos);
}
PrintListElements(ls);
printf("\n");
```



```
ProcessList(ls, print_el);  
printf("\n");
```

```
printf("A sad brisem cijelu listu\n");
```

```
p2=First(ls);
```

```
while (p2 != End(ls)) {
```

```
    Delete(p2);
```

```
}
```

```
PrintListElements(ls);
```

```
printf("\n");
```

```
system("PAUSE");
```

```
return 0;
```

```
}
```

## Drugi primjer liste pomoću pokazivača

- Napisati program koji će pročitati niz cijelih brojeva iz datoteke i od njih oblikovati linearnu jednostruko povezanu listu tako da podaci budu u rastućem nizu. Ispisati redom sadržaj liste. Učitavati podatke koji se žele brisati iz liste. Program se završava kad se upiše podatak koji ne postoji u listi.

Upotreba pokazivača na pokazivač:

`glavap` sadrži adresu pokazivača na prvi član liste, tj. `&(cvor*)` ili `&(&(cvor))`

`*glavap` sadrži pokazivač na prvi član liste, tj. `(cvor*)` ili `(&cvor)`

`**glavap` je prvi član liste, tj. `cvor`

```

#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
typedef int tip;
struct cv {
    tip element;
    struct cv *sljed;
};
typedef struct cv cvor;
// Dodavanje u listu sortiranu po rastucoj vrijednosti elementa, vraca 1 ako uspije, inace 0
int dodaj (cvor **glavap, tip element) {
    cvor *novi, *p;
    if ((novi = (cvor *) malloc(sizeof(cvor))) == NULL)
        return 0;
    novi->element = element;
    if (*glavap == NULL || (*glavap)->element >= element) { // Dodavanje na pocetak liste
        novi->sljed = *glavap;
        *glavap = novi;
    } else { // Dodavanje iza postojeceg elementa kad:
//    postojeci cvor nema sljedeceg ili element u sljedecem cvoru je veci ili jednak novome
        for (p = *glavap; p->sljed && (p->sljed)->element < element; p = p->sljed)
            ;
        novi->sljed = p->sljed;
        p->sljed = novi; }
    return 1;
}

```

```

// ispis elemenata liste
void ispisi (cvor *glava) {
    cvor *p;

    for (p = glava; p != NULL; p = p->sljed) {
        printf ("Na adresi %p je %d koji gleda na %p\n",p, p->element, p->sljed);
    }
}

```

```

// trazenje elementa liste
// vraca pokazivac na trazeni element ili NULL ako ga ne nadje
cvor *trazi(cvor *glava, tip element) {
    cvor *p;

    for (p = glava; p != NULL; p = p->sljed) {
        if (p ->element == element)
            return p;
    }
    return NULL;
}

```

```

// brisanje elementa liste po kljucu koristenjem funkcije trazi
int brisi (cvor **glavap, tip element) {
    cvor *p, *pp;

    if ((p = trazi(*glavap, element)) == NULL)
        return 0;

    if (p == *glavap) {          // Brisanje s pocetka liste

        pp = (*glavap)->sljed;
        free (*glavap);
        *glavap = pp;

    } else {                    // Brisanje iza clana liste
        // pronadji prethodni cvor
        for (pp = *glavap; pp->sljed != p; pp = pp->sljed)
            ;
        // Povezi prethodni cvor sa sljedbenikom izbrisanog cvora
        pp->sljed = p->sljed;
        // oslobodi memoriju zauzetu elementom koji se brise
        free (p);
    }
    return 1;
}

```

```

void main (void) {
    int element, j; // element i brojac elemenata
    cvor *glava;           // glava liste
    FILE *fi;              // ulazna datoteka
    fi = fopen ("UlazZaListu.txt", "r");
    if (!fi) exit (1);
    glava = NULL;
    j = 0;
    while (fscanf (fi, "%d", &element) != EOF) {
        printf ("%d. ulazni podatak je %d \n", ++j, element);
        if ((dodaj (&glava, element))) {
            ispisi (glava);
        } else {
            printf ("Nema vise mjesta\n");
            break; }
    }
    fclose (fi);
    printf ("\n");
    do { // trazenje i brisanje elemenata
        ispisi (glava);
        printf ("Upisite element koji se brise >");
        scanf ("%d", &element);
    } while (brisi (&glava, element));
    printf ("Nema trazenog elementa!\n");
    system("PAUSE");
    exit (0); }

```