Prikaz liste pomoću polja

- Na predavanjima je prikazan apstraktni tip podataka općenite liste i C pseudokod za operacije definirane na listi u implementaciji pomoću polja. Napisati program u kojem se napravi lista cijelih brojeva maksimalne duljine 100 članova, u listu upisati 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.

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

#define MAXLENGTH 100

typedef int elementtype;
typedef int position;
typedef struct list * listptr;
typedef struct list LIST;

position End(listptr lst);
position MakeNull(listptr lst);
void Insert(elementtype x, position p, listptr lst);
void Delete(position p, listptr lst);
position First(listptr lst);
position Next(position p, listptr lst);
position Previous(position p, listptr lst);
elementtype Retrieve(position p, listptr lst);
void PrintErrorAndTerminate(char errormsg[]);
void PrintList(listptr lst);

struct list{
     elementtype elements[MAXLENGTH];
     int last;
};
```

```
position End(listptr lst) {
    return lst->last + 1;
}

position MakeNull(listptr lst) {
    lst->last = -1;
    return 0;
}

void Insert(elementtype x, position p, listptr lst) {
    position q;

    if(lst->last >= MAXLENGTH - 1)
        PrintErrorAndTerminate("Insert: Lista je do kraja popunjena!");
    else {
        if(p > lst->last + 1 || p < 0)
            PrintErrorAndTerminate("Insert: Pozicija ne postoji!");
        else {
            for(q = lst->last; q >= p; q--)
                lst->elements[q + 1] = lst->elements[q];
            lst->last++;
            lst->elements[p] = x;
        }
    }
}
```

```
void Delete(position p, listptr lst) {
    position q;
    if (p > lst->last || p < 0)
        PrintErrorAndTerminate("Delete: pozicija ne postoji!");
    else   {
        lst->last--;
        for(q = p; q <= lst->last; q++)
            lst->elements[q] = lst->elements[q + 1];
    }
}

position First(listptr lst) {
        return 0;
}

position Next(position p, listptr lst) {
    return ++p;
}

position Previous(position p, listptr lst) {
    return --p;
}
```

```
elementtype Retrieve(position p, listptr lst) {
      if (p >= 0 && p <= lst->last)
            return lst->elements[p];
      else
            PrintErrorAndTerminate("Retrieve: Nepostojeca pozicija!");
            return 0;
}

void PrintErrorAndTerminate(char errormsg[]) {
      printf("%s\n", errormsg);
      exit(-1);
}

void PrintList(listptr lst) {
       position p;
      printf("< ");
      for(p = 0; p < End(lst); p++)
            printf("%d ", Retrieve(p, lst));
      printf(">\n");
}

int main() {
     LIST lst;
      elementtype clan;
      position mjesto, len;
```

```
MakeNull(&lst);
PrintList(&lst);
printf ("Koliko clanova zelite upisati u listu\n");
scanf("%d",&len);
printf("\n");

   srand(time(NULL)*4);
   for (mjesto=0; mjesto < len; mjesto++) {
         clan = (elementtype) 100 * ((float)rand() / (RAND_MAX + 1));
         Insert(clan, mjesto, &lst);
     }
PrintList(&lst);

printf ("Clan koji zelite upisati na zadnje mjesto u listu\n");
scanf("%d",&clan);
printf("\n");
         Insert(clan, End(&lst), &lst);
PrintList(&lst);

printf ("Clan koji zelite upisati u sredinu liste\n");
scanf("%d",&clan);
printf("\n");
```

```
    Insert(clan, (End(&lst) - First(&lst))/2, &lst);
    PrintList(&lst);

    printf ("Koju poziciju zelite izbrisati iz liste\n");
    scanf("%d",&mjesto);
    Delete(mjesto, &lst);
    PrintList(&lst);

    printf("A sad cu izbrisati clan prije i poslije izbrisanog\n");
     if (mjesto > 0) Delete(Previous(mjesto--, &lst),&lst);
     if (mjesto < End(&lst)) Delete(Next(--mjesto,&lst),&lst);
    PrintList(&lst);

     printf("A sad brisem cijelu listu\n");
         for (mjesto=First(&lst); mjesto < End(&lst); mjesto++) {
             Delete(mjesto--, &lst);
             }
    PrintList(&lst);
    system("PAUSE");
    return 0;
}
```

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); }
```

Primjer liste s više ključeva

- U programu učitati matične brojeve (cijeli broj) i prezimena studenata (14+1 znakova). Oblikovati listu po rastućem matičnom broju i listu po abecedi. Podaci su upisani samo jednom! Za zadani matični broj pronaći pripadno prezime.

  Ovdje će se kod oblikovanja liste koristiti adrese pokazivača (adresa adrese čvora) za modificiranje pokazivača na slijedeći čvor.

  - (*glavap)->smbr je pokazivač smbr u čvoru na koji pokazuje *glavap
  - &((*glavap)->smbr) je adresa tog pokazivača

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

struct tip {
  int mbr;
  char prezime[14+1];
};
struct cv {
  struct tip element;
  struct cv *smbr;
  struct cv *sprez;
};
typedef struct cv cvor;
// Dodavanje u listu sortiranu po rastucoj vrijednosti maticnog broja
void DodajMBR (cvor **glavap, cvor *novi) {
  // nadji pokazivac na cvor s elementom vecim od elementa novog cvora
     for (; *glavap && (*glavap)->element.mbr < novi->element.mbr; glavap = &((*glavap)->smbr))
              ;
  // glavap sadrzi adresu trazenog pokazivaca, a *glavap sadrzi vrijednost tog pokazivaca
  novi->smbr = *glavap;          // novi gleda na veceg po kljucu
     *glavap = novi;                                    // pokazivac smbr u cvoru koji prethodi novome
}
```

```
// Dodavanje u listu sortiranu po prezimenu analogno dodavanju po maticnom broju
void DodajPrezime (cvor **glavap, cvor *novi) {
     for (;*glavap && strcmp((*glavap)->element.prezime,novi->element.prezime) < 0;
    glavap = &((*glavap)->sprez))
              ;
  novi->sprez = *glavap;
     *glavap = novi;
}
// Trazenje clana za zadani maticni broj
int TraziMBR (cvor *glava, int mbr, cvor *trazeni) {
  int nasao = 0;
  while (glava) {       // Dok ima clanova liste
    if (glava->element.mbr < mbr) {
      // maticni broj clana u listi manji od trazenoga => trazi dalje
       glava = glava->smbr;
    } else if (glava->element.mbr == mbr) {
      // maticni broj clana u listi jednak trazenom => nasao
      *trazeni = *glava;
      nasao = 1;
                          break;
    } else {  // maticni broj clana u listi veci od trazenog => nema ga
      break;
    }
  }
  return nasao;
}
```

```
void main (void) {
  FILE *fi;              // ulazna datoteka
  int j, mbr;                        // brojac elemenata, maticni broj za pretragu
  struct tip element;                // element koji se dodaje u listu
  cvor *glavambr,                    // glava liste uredjene po mbr
               *glavaprez;           // glava liste uredjene po prezimenu
  cvor *p, *novi;                    // pomocne varijable
  fi = fopen ("UlazZaVisestrukuListu.txt", "r");
  if (!fi) exit (1);
  glavambr = NULL;
  j = 0;
     // citanje ulaznih podataka i dodavanje u listu uredjenu po mbr
  while (fscanf (fi, "%d %s", &element.mbr,
                    &element.prezime) != EOF) {
    printf ("%d. ulazni podatak je %d %s\n",
                           ++j, element.mbr, element.prezime);
      if ((novi = (cvor *) malloc(sizeof(cvor))) != NULL) {
      novi->element = element;
      novi->smbr = NULL;
      novi->sprez = NULL;
      DodajMBR (&glavambr, novi);
    } else {
      printf("Nema vise mjesta\n");
      break;    }
  }
   fclose (fi);
```

```
// ispis po mbr
  p = glavambr;
  printf  ("\nIspis po maticnom broju \n");
  while (p) {
    printf  ("Na adresi %p je %d %s\n", p, p->element.mbr, p->element.prezime);
    p = p->smbr;
  }

  // prolazak kroz listu uredjenu po mbr i povezivanje u listu uredjenu po prezimenu
  glavaprez = NULL;
  novi = glavambr;
  while (novi) {
    DodajPrezime (&glavaprez, novi);
    novi = novi->smbr;
  }

     // ispis po prezimenu
  p = glavaprez;
  printf  (     "\nIspis po prezimenu \n");
  while (p) {
    printf  ("Na adresi %p je %d %s\n",p, p->element.mbr, p->element.prezime);
    p = p->sprez;
  }
```

```
// trazenje clana visestruke liste po MBR
  p = (cvor *) malloc (sizeof(cvor));

  do {
    printf ("Upisite maticni broj >");
    scanf ("%d", &mbr);
    if (TraziMBR (glavambr, mbr, p)) {
      printf  ("Za maticni broj %d prezime je %s\n", mbr, p->element.prezime);
    } else {
      printf  ("Za maticni broj %d prezime nije nadjeno\n", mbr);
      break;
    }
  } while (1);
   system("PAUSE");
   exit (0);
}
```