# 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);
}
```

# Najjednostavnija izvedba stoga

- Implementacija stoga u C-u pomoću polja vrlo je jednostavna: potrebno nam je jedno polje, stack, u koje ćemo ubacivati i iz kojega ćemo izbacivati elemente, te dvije funkcije, push i pop koje ubacuju odnosno izbacuju element s vrha stoga. Slijedi primjer u kojem se stog puni cijelim brojevima, ispisuje se i zatim se skida element po element dok se stog ne isprazni.

```
#include <stdio.h>
#include <stdlib.h>
#define N 20
int stack[N];
int top = 0;      /*vrh stoga u početku postavljen na dno stacka */
void push(int el){
     stack[top++] = el;
     }
int pop(void){
     return stack[--top];
}

int main(){
int i;
   for(i = 0; i < 10; i++)
              push(i + 1);
   for (i = 0; i < N; i++) {
       printf("stack[%i]: %i\n ",i,stack[i]);
      }
   for(i = 0; i < 10; i++)
              printf("%i", pop());
   system("PAUSE");
   return 0;
}
```

# Računanje algebarskih izraza zapisanih u obrnutoj poljskoj notaciji

- Uobičajeni način zapisivanja algebarskih izraza je onaj kod kojega simboli operatora dolaze *između* operanada. Npr. u izrazu 2 + 3 * 5 operatori + i * dolaze između brojeva 2 i 3, odnosno 3 i 5. Redoslijed računanja određen je prioritetom računskih operacija (osim ako nije promijenjen upotrebom zagrada). Tako je 2 + 3 * 5 = 17 jer se prvo obavlja množenje a tek onda zbrajanje.
- Međutim, postoji i drugačiji način zapisivanja izraza, koji se zove **Obrnuta poljska notacija** (engl. **RPN**, **R**everse **P**olish **N**otation). Tu notaciju je prvi uveo poljski matematičar Jan Lukasiewicz. Kod ovog načina, simboli operacija dolaze **nakon** operanada, a računanje se izvodi upravo onim redom kojim su napisane operacije. Zagrade se ne koriste.
- Obrnutu poljsku notaciju koriste gotovo svi HP-ovi kalkulatori, a značajna je i zato jer računala interno upravo tim algoritmom izračunavaju algebarske izraze.

- Evo nekoliko primjera upotrebe RPN-a pri zapisivanju izraza:

- Primjer 1: Izraz 2 + 3 u RPN-u bismo zapisali kao 2 3 +
- Primjer 2: Izraz 2 + 3 * 5  u RPN-u ima oblik 2 3 5 * +
- Primjer 3: Izraz (2 + 3) * 5  u RPN-u ima oblik 2 3 + 5 *

- Želimo sastaviti algoritam kojemu bismo na ulazu dali proizvoljni (korektno zapisani) RPN izraz, a algoritam kao rezultat svog rada treba izračunati vrijednost ulaznog izraza. Rad algoritma je prilično jednostavan - čitamo izraz slijeva nadesno i zatim:
- svaki put kad pročitamo broj, stavimo ga na stog (operacija push).
- svaki put kad pročitamo znak neke operacije, tada skinemo (operacija pop) dva broja sa stoga, nad njima izvršimo traženu operaciju i zatim rezultat te operacije stavimo na stog (operacija push).

- Donji program kao ulaz prima RPN-izraz čiju vrijednost zatim izračunava i ispisuje rezultat. Kao oznaku kraja izraza koristi se znak ";". Stoga prilikom ukucavanja izraze treba unositi ovako

  2 3 5 * + ;

```
#include <stdio.h>
#include <stdlib.h>
#define N 1000

float stack[N];
int top = 0;
void push(float el){
     stack[top++] = el;
}
float pop(){
     return stack[--top];
}

int main(){
 char unos[200];
 float a, b, rez;
 printf("Unesi izraz u RPN obliku: ");
 do {
              scanf("%s", unos);
 switch(unos[0]) {
     case '+':
              b = pop();
              a = pop();
              push(a + b);
              break;
```

```
    case '-':
            b = pop();
            a = pop();
            push(a - b);
            break;
    case '*':
            b = pop();
            a = pop();
            push(a * b);
            break;
    case '/':
            b = pop();
            a = pop();
            push(a / b);
            break;
    case ';':
            rez = pop();
            break;
    default:
            push(atof(unos));
            break;
            }
    } while(unos[0] != ';');
    printf("Rezultat je: %f\n", rez);
    system("PAUSE");
    return 0;
}
```

# Implementacija stoga pomoću polja

- Na predavanjima je uveden apstraktni tip podataka stog i napravljen C pseudokod za jednu njegovu varijantu. Napisati funkcije koje obavljaju zadane operacije na stogu upotrebom polja za primjer s predavanja.

```
#include <stdio.h>
#include <stdlib.h>
#define MAXLENGTH 4

typedef int elementtype;
typedef struct stack * stackptr;
typedef struct stack STACK;

void MakeNull(stackptr s);
int Empty(stackptr s);
void Push(elementtype x, stackptr s);
void Pop(stackptr s);
elementtype Top(stackptr s);
void PrintErrorAndTerminate(char errormsg[]);
```

```
struct stack {
      int top;
      elementtype elements[MAXLENGTH];
};

void MakeNull(stackptr s) {
   s->top = MAXLENGTH;
}

int Empty(stackptr s) {
   if (s->top == MAXLENGTH)
         return 1;
   else
         return 0;
}

void Push(elementtype x, stackptr s) {
      if (s->top == 0)
               PrintErrorAndTerminate("Push: Stack je pun!");
      else
               s->elements[--s->top] = x;
}
```

```
void Pop(stackptr s){
    if (Empty(s))
        PrintErrorAndTerminate("Pop: Stack je prazan!");
    else
        s->top++;
}

elementtype Top(stackptr s)
{
    if (Empty(s))
        PrintErrorAndTerminate("Top: Stack je prazan!");

    return(s->elements[s->top]);
}

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

```
int main(){
 STACK s;

 MakeNull(&s);
Push(1, &s);
Push(2, &s);
Push(3, &s);
Push(4, &s);
 printf("%d\n", Top(&s));
Pop(&s);
 printf("%d\n", Top(&s));
Push(5, &s);
printf("%d\n", Top(&s));
Pop(&s);
printf("%d\n", Top(&s));
Pop(&s);
printf("%d\n", Top(&s));
Pop(&s);
printf("%d\n", Top(&s));
Pop(&s);
system("PAUSE");
return 0;
}
```

# Još jedan primjer stoga ostvarenog poljem

- Stog duljine 6 elemenata se puni generiranim slučajnim brojem ako je on neparan, ukoliko je generiran parni broj, vrh stoga se skine. Ispisati sadržaj stoga nakon svakog koraka te upozoriti ukoliko je stog prazan ili pun.

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

#define MAXSTOG 6

int dodaj (int stavka, int stog[], int n, int *vrh) {
  if (*vrh >= n-1) return 0;
  (*vrh)++;                          // ne pisati  *vrh++; !
  stog[*vrh] = stavka;
  return 1;
}
```

```
int skini (int *stavka, int Stog[], int *vrh) {
      if (*vrh < 0) return 0;
      *stavka = Stog[*vrh];
      (*vrh)--;
      return 1;
}

void main (void) {
  int novi, stari, stog [MAXSTOG];
  int vrh,i,ch;

  vrh = -1; // prazan stog

      printf (“Generiraju se slucajni nenegativni cijeli brojevi.\n");
      printf ("Neparni brojevi upisuju se na stog\n");
      printf ("Parni broj znaci skidanje sa stoga\n");
      printf ("Za obavljanje jednog koraka pritisnuti ENTER, za kraj bilo koji znak\n\n");

      srand ((unsigned) time (NULL));

      while (isspace(getchar())) {
            if (vrh == -1) {
                  printf("prazan stog");
            }
```

```
        else {
                printf ("Stog:");
                for (i=0; i <= vrh; ++i) printf (" %d", stog[i]);
        }
        putchar ('\n');
        novi = rand ();
        if (novi%2) {                   // Neparni se upisuju na stog
                printf ("Dodaj %d\n", novi);
                if (!dodaj (novi, stog, MAXSTOG, &vrh)) printf("Stog je pun!\n");
        } else {                        // Parni broj znaci skidanje sa stoga

                printf ("Skini...");
                if (skini (&stari, stog, &vrh))    {
                        printf ("Skinut %d\n", stari);
                } else {
                        printf("Stog je prazan!\n");
                }
    }
  }
    system("PAUSE");
    exit(0);
}
```