Primjer baratanja s podatcima u binarnom stablu traženja

- Napisati funkciju koja upisuje u čvor BST podatke tipa: naziv proizvoda (char 15) i cijena (float). Uređaj između zapisa u čvorovima određuje naziv proizvoda (manje znači bliže početku abecede). Napisati funkciju koja ispiše zapise u BST po algoritmu INORDER, te funkciju koja izračuna prosječnu cijenu proizvoda u cjeniku.

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

typedef struct {
  char artikl[15+1];
  float cijena;
} el;

typedef struct cv {
  el element;
  struct cv *lijevo;
  struct cv *desno;
} cvor;

typedef struct {
  float suma;
  int broj;
} pros;
```

```
// upisuje u stablo podatke: lijevo manji, desno veci
cvor *upis (cvor *glava, el element) {
  int smjer;
  if (glava == NULL) {
    glava = (cvor *) malloc (sizeof (cvor));
            if (glava) {
                        glava->element = element;
                        glava->lijevo = glava->desno = NULL;
            } else {
                        printf ("Nema dovoljno memorije!\n");
            }
  }
    else if ((smjer = strcmp (element.artikl, glava->element.artikl)) < 0) {
    glava->lijevo = upis (glava->lijevo, element);
  }
    else if (smjer > 0) {
    glava->desno  = upis (glava->desno, element);
  }
    else {
    printf ("Podatak '%s' vec postoji!\n", element.artikl);
  }
  return glava;
}
```

```
// ispis inorder
void ispisin (cvor *glava) {
  if (glava != NULL) {
    ispisin (glava->lijevo);
    printf ("%-15s %6.2f\n", glava->element.artikl, glava->element.cijena);
    ispisin (glava->desno);
  }
}

// sumiranje cijena i brojanje elemenata
void prosjek (cvor *glava, pros *prs) {
  if (glava != NULL) {
            prs->suma += glava->element.cijena;
            prs->broj++;
    prosjek (glava->lijevo, prs);
    prosjek (glava->desno, prs);
  }
}

void main(void) {
  FILE *fi;
  int j;
  cvor *glava;                    // pokazivac na korijen
  el element;                     // sadrzaj cvora
  pros prs;                       // broj elemenata i suma cijena
```

```
  prs.suma = 0.; prs.broj = 0;
  fi = fopen ("UlazZaProsjekUStablu.txt", "r");
  if (!fi) {
    printf ("Nema ulaznih podataka\n");
              exit (1);
  }
  j = 1;
  glava = NULL;
  while (fscanf (fi, "%s %f", element.artikl, &element.cijena) != EOF) {
    printf ("%2d. ulazni podatak je %-15s %6.2f\n", j++, element.artikl, element.cijena);
    glava = upis (glava, element);
  }
  fclose (fi);
     // ispis, racun sume cijena i broja elemenata
     getchar ();
     ispisin (glava);
     getchar ();
     prosjek (glava, &prs);
     if (prs.broj) {
          printf ("Suma=%6.2f, Broj cvorova=%d, Prosjek=%6.2f\n",
                              prs.suma, prs.broj, prs.suma / prs.broj);
              getchar ();
  }
     exit (0);
}
```

Nešto složeniji primjer binarnog stabla traženja

Zadatak: Zadani su podaci o studentu: matični broj long, ime 25 znakova, prezime 25 znakova i godina rođenja short. Uz uporabu dinamičke strukture binarnog sortiranog stabla (tj. binarno stablo traženja) potrebno je napisati funkcije za:

a) pohranu podataka o studentima uz uvjet da je omogućeno brže pronalaženje studenata po prezimenu

b) pronalaženje studenta po prezimenu

c) ispis čvorova po INORDER algoritmu

d) ispis čvorova na zadanoj razini

e) izračunavanje dubine stabla i najmanje razine listova

f) ispis listova stabla

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

struct podaci {
  long matBroj;
  char ime[25+1];
  char prezime[25+1];
  short godRod;
};
typedef struct podaci Podaci;

struct cvor {
  Podaci * student;
  struct cvor * lijevo;
  struct cvor * desno;
};
typedef struct cvor Cvor;
```

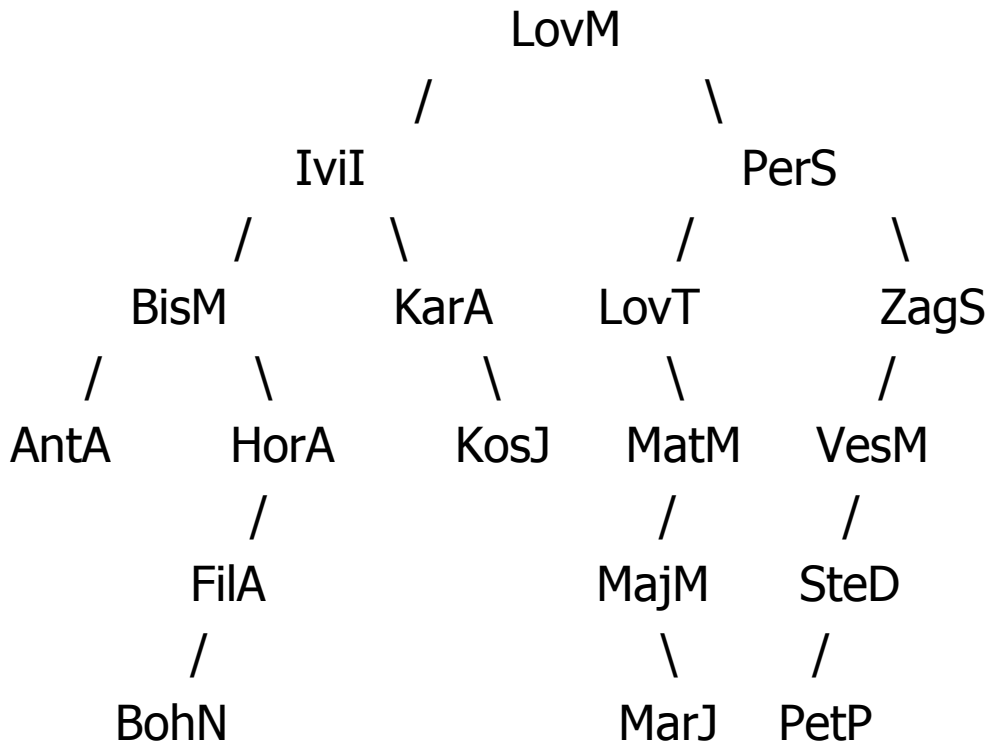
```
// funkcija dodaje podatak u binarno sortirano stablo, kljuc je prezime
Cvor * dodajPrezime (Cvor * korijen, Podaci * student) {
  int smjer;
  if (korijen == NULL) {
    korijen = (Cvor *) malloc (sizeof(Cvor));
    if (korijen) {
      korijen->student = student;
      korijen->lijevo = korijen->desno = NULL;    }
  } else if ((smjer = strcmp(student->prezime, korijen->student->prezime)) < 0)
           korijen->lijevo = dodajPrezime (korijen->lijevo, student);
        else if (smjer > 0)
               korijen->desno = dodajPrezime (korijen->desno, student);
             else
               printf("Podatak %s vec postoji!\n", student->prezime);
  return korijen;}

// funkcija pretrazuje binarno stablo trazenja po kljucu prezime
Cvor * potraziPrezime (Cvor * korijen, char * prezime) {
  int smjer;
  if (korijen) {
    if ((smjer = strcmp(prezime, korijen->student->prezime)) < 0)
      return potraziPrezime (korijen->lijevo, prezime);
    else if (smjer > 0)
           return potraziPrezime (korijen->desno, prezime); }
  return korijen; }
```

```
// funkcija inorder ispisuje zadano binarno stablo
void inOrder (Cvor * korijen) {
  if (korijen) {
    inOrder (korijen->lijevo);
    printf("%s, %s %d %d\n", korijen->student->prezime, korijen->student->ime,
      korijen->student->matBroj,korijen->student->godRod);
    inOrder (korijen->desno);
  }
}

// funkcija ispisuje sve cvorove zadane dubine
// poziv ispis(korijen, 1, n) gdje je n trazena dubina
void ispisi (Cvor * korijen, int trenutnaDubina, int trazenaDubina) {
  if (korijen) {
    if (trenutnaDubina == trazenaDubina)
      printf("%d %s %s %d\n",korijen->student->matBroj, korijen->student->ime,
      korijen->student->prezime,korijen->student->godRod);
    ispisi (korijen->lijevo, trenutnaDubina + 1, trazenaDubina);
    ispisi (korijen->desno, trenutnaDubina + 1, trazenaDubina);
  }
}
```

```
// funkcija trazi listove s najvecom i najmanjom razinom
void dubine(Cvor * korijen, int trenDub, int * maxDub, int * minDub) {
  if (korijen) {
    if (!korijen->lijevo & !korijen->desno) {
      if (*maxDub == 0 || trenDub > *maxDub)
        *maxDub = trenDub;
      if (*minDub == 0 || trenDub < *minDub)
        *minDub = trenDub;
    }
    else {
      dubine(korijen->lijevo, trenDub + 1, maxDub, minDub);
      dubine(korijen->desno, trenDub + 1, maxDub, minDub);
     }
  }
}

// funkcija za zadano binarno stablo ispisuje listove
void ispisiListove(Cvor * korijen) {
  if (korijen) {
    if (!korijen->lijevo & !korijen->desno)
      printf("%s %s %d %d; ", korijen->student->ime, korijen->student->prezime,
            korijen->student->matBroj,korijen->student->godRod);
    ispisiListove(korijen->lijevo);
    ispisiListove(korijen->desno);
  }
}
```

```
void main () {
  FILE * fUl;
  char buf[256], trazim[25+1];
  Podaci * student;
  Cvor * korijenPrezime = NULL;
  Cvor * trazeni;
  int minDubina = 0, maxDubina = 0;
  int i;

  if ((fUl = fopen("studenti.txt", "r")) == NULL) {
    printf("Ne mogu otvoriti 'studenti.txt'\n");
    exit(1);
  }
  while (fgets(buf, 256, fUl)) {
    student = (Podaci *) malloc(sizeof(Podaci));
    sscanf(buf, "%ld;%[^;];%[^;];%d", &(student->matBroj), student->ime,
      student->prezime, &(student->godRod));
    korijenPrezime = dodajPrezime(korijenPrezime, student);
  }
  fclose(fUl);

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

```
while (1){
      printf("Upisite prezime koje se trazi, kraj za prekid trazenja\n");
      scanf("%s",trazim);
      if (strcmp(trazim,"kraj")==0) break;
      if(trazeni = potraziPrezime(korijenPrezime, trazim))
        printf("Pronasao: %s, %s %ld %d\n", trazeni->student->prezime,
        trazeni->student->ime, trazeni->student->matBroj, trazeni->student->godRod);
      else
              printf("Nije nadjen student %s\n",trazim);
}

  printf("\nIspis listova:\n");
  ispisiListove(korijenPrezime);

  dubine(korijenPrezime, 1, &maxDubina, &minDubina);
  printf("\nMaxDubina=%d  MinDubina=%d\n", maxDubina, minDubina);

  for (i=1; i<=maxDubina; i++) {
    printf("%d razina:\n", i);
    ispisi(korijenPrezime, 1, i);
  }
 system("PAUSE");
}
```


LovM
IviI
PerS
BisM
KarA
LovT
ZagS
AntA
HorA
KosJ
MatM
VesM
FilA
MajM
SteD
BohN
MarJ
PetP

Operacije sa skupovima

- Napisati funkciju koja će pronaći i ispisati presjek dvaju skupova cijelih brojeva, te ustanoviti složenost algoritma.
- Razmotrit ćemo dva slučaja: kada su elementi skupova nesortirani i kada su sortirani
- Za nesortirane elemente potrebno je usporediti svaki element iz prvog skupa sa svakim iz drugog - funkcija presjek1()
- Složenost funkcije je O(NA*NB)
- Za sortirane elemente: uspoređuje se element iz prvog skupa s elementom iz drugog i pomiče se kursor kod onog skupa čiji je element manji – funkcija presjek2()
- Složenost funkcije O(max(NA,NB)), ali radi samo za sortirane nizove

```
#include <stdio.h>
#include <stdlib.h>
#define NA 10
#define NB 15

int presjek1(int *a, int *b, int **p) {
    int i, j, k = 0; // k – broj zajedničkih elemenata
    for (i = 0; i < NA; i ++)
        for (j = 0; j < NB; j ++)
            if (a[i] == b[j]) {
                *p = realloc(*p, (k + 1) * sizeof(int));
                (*p)[k] = a[i];  // paziti: *p[k] znači *(p[k])!!
                ++ k; }
    return k; }

int presjek2(int *a, int *b, int **p) {
    int i = 0, j = 0, k = 0;
    while (i < NA && j < NB) {
        if (a[i] == b[j]) {
            *p = realloc(*p, (k + 1) * sizeof(int));
            (*p)[k] = a[i];
            ++ k; ++ i; ++ j;
        } else if (a[i] < b[j]) {
            ++ i; } else {
            ++ j;} }
    return k; }
```

```
void InsertionSort(int a[], int n) {
   int j,k,i;
   for(j = 1; j < n; j++) {
            k = a[j];
             i = j - 1;
            while (a[i] > k && i >= 0) {
                        a[i + 1] = a[i];
                        i--; }
            a[i + 1] = k; }
 }

void main() {
     int a[NA] = {1,8,2,22,19,17,21,11,72,7};
     int b[NB] = {2,19,3,34,72,99,35,0,5,14,7,31,25,67,21};
     int *p = NULL, i, n;

      printf(" Presjek nesortiranih skupova:\n");
     n = presjek1(a, b, &p);
     printf ("Presjek: ");
     for (i = 0; i < n; i++)
            printf ("%d ", p[i]);
```

```
    printf(" Presjek sortiranih skupova:\n");
    InsertionSort(a,NA);
    InsertionSort(b,NB);
    printf("Polje a nakon sortiranja: ");
   for (i = 0; i < NA; i ++)
    printf ("%d ", a[i]);
   printf ("\n");
    printf("Polje b nakon sortiranja: ");
   for (i = 0; i < NB; i ++)
    printf ("%d ", b[i]);
   printf ("\n");

    n = presjek2(a, b, &p);
   printf ("Presjek: ");
   for (i = 0; i < n; i ++)
   printf ("%d ", p[i]);
   system("PAUSE");
}
```