Strukture podataka i algoritmi 1
(zadatak - max 30 poena)

Jul, 2023

Asocijacija koja okuplja botaničke bašte i uređene parkove u cilju zaštite biljnog sveta formira bazu biljnih vrsta koje ima svaka od članica. Za svaku članicu se učitava njen ID (ceo broj), naziv (niz karaktera), lokacija (niz karaktera) i površina koju zauzima (realan broj u m^2). Za svaku biljnu vrstu se unosi njen ID (ceo broj), naziv (niz karaktera), da li se radi o visokom ili niskom rastinju (unosi se karakter V ili N), na koliko različitih lokacija se nalazi ta biljna vrsta i za svaku lokaciju ID članice asocijacije članica asocijacije kojoj pripada ta lokacija i broj komada (ceo broj) ukoliko se radi o visokom rastinju, odnosno površina (realan broj u m^2) ukoliko se radi u niskom rastinju. Jedna članica asocijacije može da ima istu biljnu vrstu na više lokacija. Ukoliko se kao ID članice, prilikom unosa biljnih vrsta, unese nepostojeći broj, ta lokacija briše. Odrediti po jednu biljnu vrstu, odvojeno za visoko i nisko rastinje, koje su najmanje zastupljene. Usred elementarnih nepogoda došlo je do uništenja biljaka. Podaci o nastaloj šteti se unose tako što se unosi ID biljne vrsta, ID članice i realna broj koji predstavlja procenat uništenja na tom mestu. Ukoliko je neka biljna vrsta uništena na prostoru koji pripada jednoj članica, ta biljna vrste se uklanja iz evidencije biljnih vrsta koje ta članica asocijacije ima. Ukoliko je došlo potpunog uništenje neke biljne vrste, tj. nema nije više ni kod jedne članice, ta biljna vrsta se briše i spiska biljnih vrsta i ispisuje se poruka o ugroženoj biljnoj vrsti.
(4 poena, obavezno)

Za rešavanje problema napisati sledeće funkcije:

a) Definisati sve potrebne složene tipove podataka neophodne za rešavanje opisanog problema.
(3 poena, obavezno)
b) Napisati funkciju **UcitajClanice** koja iz datoteke *Clanice.txt* učitava podatke o članicama asocijacije i formira listu/niz članica.
(3 poena, obavezno)
c) Napisati funkciju **NadjiClanicu** koja za datu oznaku vraća pokazivač na traženu članicu.
(2 poena)
d) Napisati funkciju **UcitajBiljke** koja iz datoteke *Biljke.txt* učitava podatke o biljnim vrstama i formira listu/niz biljaka.
(3 poena + 2 poena)
e) Napisati funkciju **MinBiljke** koja određuje koje su dve biljne vrste najmanje rasprostranjene.
(4 poena)
f) Napisati funkciju **Steta** koja prihvata informacije o šteti i ažurira podatke o biljnim vrstama.
(3+1 poena)
g) Napisati funkciju **UgrozeneBiljke** koja određuje i iz liste biljnih vrsta uklanja one koje su potpuno uništene.
(3 poena + 2 poena)

Pri učitavanju podataka podrazumevati da su svi podaci korektno zadati.
Dozvoljeno je proširivanje struktura i definisanje novih, kao i definisanje drugih funkcija. Za korišćenje unija dobijaju se bonus poeni.
Maksimalan broj poena se dobija samo u slučaju da postoji veza između struktura koje čuvaju biljke i struktura koje čuvaju članice asocijacije, tj. ako postoje pokazivači "na jednu" ili "na obe strane".

Zadatak se može rešavati korišćenjem povezanih lista, nizova ili kombinacijom.
Zadatak rešiti bez korišćenja globalnih promenljivih i bez unapred definisanih dužina korišćenih nizova (izuzetak može da bude pomoćni niz za učutavanje stringova)**.**
Za funkcije koje nisu definisane mora da postoji deklaracija funkcije i kratak opis koji deo problema funkcija treba da reši i mogu se pozivati kao da postoji kompletna definisana funkcija.

Rešenje studenta

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


struct Clanica
{
    int ID;
    char *naziv;
    char *lokacija;
    float povrsina;
    struct Clanica *sledeca;
};

struct clan_vrst
{
    struct Clanica *lokacija;
    union{
        int broj;
        float povrsina;
    } br_pov;
    struct clan_vrst *sledeca;
};

struct vrste{
    int ID;
    char *naziv;
    char tip;
    int br_lok;
    struct clan_vrst *lista;
    float ukupno;
    struct vrste *sledeca;
};

struct Clanica* UcitajClanice(FILE *in,int *n){
    struct Clanica *glava=NULL;
    struct Clanica *nov,*poslednji;
    int i,d;
    char rec[50];
    fscanf(in, "%d", n);
    for (i = 0; i < *n; i++)
    {
        nov=(struct Clanica*)malloc(sizeof(struct Clanica));
        fscanf(in, "%d\n",&nov->ID);
        fgets(rec, 50, in);
        d=strlen(rec);
        rec[d-2]='\0';
        nov->naziv=(char*)malloc(d);
        strcpy(nov->naziv,rec);
        fgets(rec, 50, in);
        d=strlen(rec);
        rec[d-2]='\0';
        nov->lokacija=(char*)malloc(d);
        strcpy(nov->lokacija,rec);
        fscanf(in, "%f",&nov->povrsina);
        nov->sledeca=NULL;

        if (glava==NULL)
        {
            glava=nov;
            poslednji=nov;
        }
        else {
```

```
            poslednji->sledeca=nov;
            poslednji=nov;
        }
    }
    return glava;
}

struct Clanica* NadjiClanicu(struct Clanica *glava, int id){
    while (glava && glava->ID!=id) {
        glava=glava->sledeca;
    }
    return glava;
}

struct vrste* UcitajBiljke(FILE *in, int *m,struct Clanica *clanice){
    struct vrste *glava=NULL;
    struct vrste *nov, *posledji;
    struct clan_vrst *nov_cv;
    int i,j,d,cl_id;
    int visoko;
    float nisko;
    char rec[50];
    fscanf(in, "%d\n", m);
    for (i=0; i<*m; i++) {
        nov=(struct vrste*)malloc(sizeof(struct vrste));
        fscanf(in, "%d\n",&nov->ID);
        fgets(rec, 50, in);
        d=strlen(rec);
        rec[d-2]='\0';
        nov->naziv=(char*)malloc(d);
        strcpy(nov->naziv, rec);
        fscanf(in, "%c",&nov->tip);
        nov->ukupno=0;
        fscanf(in, "%d\n",&nov->br_lok);
        nov->lista=NULL;
        for(j=0;j<nov->br_lok;j++){
            fscanf(in, "%d\n",&cl_id);
            struct clan_vrst *temp;
            struct Clanica *temp2;
            temp2=NadjiClanicu(clanice, cl_id);
            if (nov->tip=='V')
            {
                fscanf(in, "%d\n",&visoko);
            }
            else {
                fscanf(in, "%f\n",&nisko);
            }
            if (temp2)
            {
                temp=nov->lista;
                while (temp && temp->lokacija!=temp2) {
                    temp=temp->sledeca;
                }
                if (temp)
                {
                    if (nov->tip=='V')
                    {
                        temp->br_pov.broj+=visoko;
                        nov->ukupno+=visoko;
                    }
                    else {
                        temp->br_pov.povrsina+=nisko;
                        nov->ukupno+=nisko;
                    }
                }
```

```
                else {
                    nov_cv=(struct clan_vrst*)malloc(sizeof(struct clan_vrst));
                    nov_cv->lokacija=temp2;
                    if (nov->tip=='V')
                    {
                        nov_cv->br_pov.broj=visoko;
                        nov->ukupno+=visoko;
                    }
                    else {
                        nov_cv->br_pov.povrsina=nisko;
                        nov->ukupno+=nisko;
                    }
                    nov_cv->sledeca=nov->lista;
                    nov->lista=nov_cv;
                }
            }
        }
        nov->sledeca=NULL;
        if (glava==NULL)
        {
            glava=nov;
            posledji=nov;
        }
        else {
            posledji->sledeca=nov;
            posledji=nov;
        }

    }
    return glava;
}

void MinBiljke(struct vrste *glava){
    struct vrste *min_V,*min_N;
    struct vrste *temp;
    temp=glava;
    while (temp && temp->tip!='V') {
        temp=temp->sledeca;
    }
    min_V=temp;
    while (temp) {
        if (temp->tip=='V' && temp->ukupno<min_V->ukupno)
        {
            min_V=temp;
        }
        temp=temp->sledeca;
    }
    temp=glava;
    while (temp && temp->tip!='N') {
        temp=temp->sledeca;
    }
    min_N=temp;
    while (temp) {
        if (temp->tip=='N' && temp->ukupno<min_N->ukupno)
        {
            min_N=temp;
        }
        temp=temp->sledeca;
    }
    if (min_V) {
        printf("Visko rastinje %s\n",min_V->naziv);
    }
    if (min_N)
    {
        printf("Nisko rastinje %s\n",min_N->naziv);
```

```
    }
}

void Steta(struct vrste *glava)
{
    struct vrste *temp_v;
    struct clan_vrst *temp_c;

    int i,k,id_c,id_v;
    float pr;
    int visoko;
    float nisko;

    scanf("%d",&k);
    for (i=0; i < k; i++)
    {
        scanf("%d %d %f",&id_c,&id_v,&pr);
        temp_v=glava;
        while (temp_v && temp_v->ID!=id_v) {
            temp_v=temp_v->sledeca;
        }
        if (temp_v)
        {
            temp_c=temp_v->lista;
            while (temp_c && temp_c->lokacija->ID!=id_c) {
                temp_c=temp_c->sledeca;
            }
            if (temp_c)
            {
                if (temp_v->tip=='V')
                {
                    visoko=(int)(temp_c->br_pov.broj*pr/100);
                    temp_c->br_pov.broj-=visoko;
                    temp_v->ukupno-=visoko;
                }
                else {
                    nisko=temp_c->br_pov.povrsina*pr/100;
                    temp_c->br_pov.povrsina-=nisko;
                    temp_v->ukupno-=nisko;
                }
            }
        }
    }
}

struct vrste* UgrozeneBiljke(struct vrste *glava)
{
    struct vrste *temp,*prethodni;
    struct clan_vrst *temp2;
    while (glava && !glava->ukupno) {
        temp=glava;
        glava=glava->sledeca;
        while (temp->lista) {
            temp2=temp->lista;
            temp->lista=temp->lista->sledeca;
            free(temp2);
        }
        free(temp);
    }
    temp=glava->sledeca;
    prethodni=glava;
    while (temp) {
        if (!temp->ukupno)
        {
            while (temp->lista) {
```

```
                temp2=temp->lista;
                temp->lista=temp->lista->sledeca;
                free(temp2);
            }
            prethodni->sledeca=temp->sledeca;
            free(temp);
            temp=prethodni->sledeca;
        }
        else {
            temp=temp->sledeca;
            prethodni=prethodni->sledeca;
        }
    }
    return glava;
}

int main(){
    struct Clanica *clanice;
    struct vrste *biljke;
    FILE *in;
    int n,m;
    in=fopen("clanice.txt", "r");
    clanice=UcitajClanice(in,&n);
    fclose(in);
    in=fopen("biljke.txt", "r");
    biljke=UcitajBiljke(in, &m, clanice);
    fclose(in);
    MinBiljke(biljke);
    Steta(biljke);
    biljke=UgrozeneBiljke(biljke);
}
```