STRUKTURE PODATAKA I ALGORITMI 1

VEŽBE 12

Nikola Bačanin
Nikola Rnjak

SORTIRANJE LISTE

- Napisati program koji sortira listu celih brojeva u neopadajućem poretku. Pri sortiranju liste, dozvoljeno je menjati samo pokazivače na sledeći element, **bez menjanja vrednosti elemenata.**

SORTIRANJE LISTE

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

typedef struct element
{
	int broj;
	struct element *sledeci;
} Element;

Element* alociraj(int broj)
{
	Element *novi = (Element*)malloc(sizeof(Element));
	novi->broj = broj;
	novi->sledeci = NULL;
	return novi;
}
```

SORTIRANJE LISTE

```
void dodaj(Element **p, Element *novi)
{
	if (*p == NULL)
		*p = novi;
	else
	{
		Element *pom = *p;
		while (pom->sledeci != NULL)
			pom = pom->sledeci;

		pom->sledeci = novi;
	}
}
```

SORTIRANJE LISTE

```
void sortiraj(Element **p)
{
	Element *pp = *p;
	Element *q = NULL;

	while (pp != NULL)
	{
		Element *min_pr = NULL;
		Element *min = pp;
		Element *pom_pr = pp;
		Element *pom = pp->sledeci;
```

SORTIRANJE LISTE

```
while (pom)
{
        if (pom->broj < min->broj)
        {
                min_pr = pom_pr;
                min = pom;
        }

        pom_pr = pom;
        pom = pom->sledeci;
}
```

```
            if (min_pr == NULL)
                  pp = pp->sledeci;
            else
                  min_pr->sledeci = min->sledeci;

            min->sledeci = NULL;

            dodaj(&q, min);
      }

      *p = q;
}
```

ZADATAK

- Potrebno je demonstrirati igru "**Slučajno najbolji**" u kojoj učestvuje **n** igrača. Za svakog IGRAČA se pamte sledeći podaci:
  - ime i prezime igrača (više reči)
  - ID igrača (ceo broj)
  - država (više reči)
  - runda (ceo broj)
  - niz celih izabranih brojeva (neće biti više od 20 rundi)
- Igra se organizuje tako što se prvo unosi broj igrača N, a zatim podaci o njima na osnovu kojih se kreira igra. Igrači kako se prijavljuju za igru zauzimaju poslednju poziciju na startnoj listi. Za svakog igrača se najpre unosi ime i prezime, ID se dodeljuje kao redni broj prijave (prvi igrač ima ID 1), država, u samom startu sigurno učestvuje u prvoj rundi tako da mu je na startu broj runde 1 i odmah prijavljuje svoj izabrani ceo broj koji je prvi element niza izabranih brojeva.
- Ispisati kreiranu prvu rundu igre sa **svim** podacima o prijavljenim igračima u formatu:

  Petar Petrovic (ime i prezime)

  2 (ID)

  Srbija (Država)

  3 (broj runde)

  5 3 2 (izabrani brojevi u rundama)

ZADATAK

- Igra se nastavlja tako što u drugu rundu odlaze igrači koji imaju jedinstven izabrani broj iz prve runde. Ispisati igrače druge runde.
- U drugoj rundi za preostale igrače se povećava broj runde i svaki od preostalih igrača ponovo prijavljuje svoj izabrani broj (za njih se unose brojevi), koji se beleži na odgovaraće mesto u nizu izabranih brojeva. Zatim u treću rundu odlaze svi igrači koji su u drugoj (prethodnoj) rundi prijavili jedinstven broj. Igra se nastavlja sve dok na kraju ne ostanu tri, ili manje igrača.
- Ispisivati igrače svake runde.
- **BONUS**: Napisati funkciju kojom se određuje država iz koje dolazi najviše igrača.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct lista_igre
{
    char ime_prezime[50];
    int ID;
    char drzava[50];
    int runda;
    int izabrani_brojevi[20];
    struct lista_igre *sledeci;
}IGRAC;

IGRAC *alociraj(void)
{
    IGRAC *novi;
    novi = (IGRAC*) malloc (sizeof(IGRAC));
    return novi;
}
```

```
void ispis_liste(IGRAC *glava)
{   int i;
    if(glava == NULL)
        printf("lista je prazna\n");
        else
        {   printf("LISTA\n");
            while (glava != NULL)
            {
                printf("%s\n",glava->ime_prezime);
                printf("%d\n",glava->ID);
                printf("%s\n",glava->drzava);
                printf("%d\n",glava->runda);
                for(i=0;i<glava->runda;i++)
                        printf("%d  ",glava-
>izabrani_brojevi[i]);
                printf("\n");
                glava = glava->sledeci;
            }
        }
}
```

```
IGRAC *ucitaj_novi(int redni_br)
{
    IGRAC *novi;
    novi = alociraj();
    fgets(novi->ime_prezime,50,stdin);//unosimo ime i skidamo
'\n'
    novi->ime_prezime[strlen(novi->ime_prezime)-1] = '\0';
    novi->ID = redni_br;//dodeljujemo redni broj

    fgets(novi->drzava,50,stdin); //unosimo drzavu i skidamo '\n'
    novi->drzava[strlen(novi->drzava)-1] = '\0';

    novi->runda = 1;//za rundu uvek ide 1

    scanf("%d",&novi->izabrani_brojevi[0]);//unosimo prvi
izabrani
    getchar();
    novi->sledeci = NULL;
    return novi;
}
```

```
IGRAC *formiraj_listu(int n)
{
    IGRAC *glava = NULL;
    IGRAC *kraj,*novi;
    int i;
    for(i=0;i<n;i++)
    {
        novi = ucitaj_novi(i+1);
        if(glava == NULL) //ako je prazna
        {
            glava = novi;
            kraj = novi;
        }else //ako lista nije prazna
            {
                kraj->sledeci = novi;
                kraj = novi;
            }
    }
    return glava;
}
```

ZADATAK

```
IGRAC *ucitaj_izabrani_broj(IGRAC *glava)
{
    IGRAC *pom;
    pom = glava;
    while(pom != NULL)
    {
        scanf("%d",&pom->izabrani_brojevi[pom->runda]); // ucitavamo
novi izabrani broj na poslednje mesto
        pom->runda++;//uvecavamo rundu
        pom = pom->sledeci;
    }
    return glava;
}
IGRAC *nadji_jedinstvene(IGRAC *glava, int *n)
{   //pom1 ide kroz listu, pom2 trazi sve elemente koji imaju
izabrani broj isti kao pom1
    IGRAC *pom1,*pom2,*pom3,*pom11; //pom11 ide iza pom1, iza pom2
ide pom3
    int indikator = 0;//indikator se menja ako pronadje bar jednog sa
istim izabranim brojem, 0 ako ne nadje  1 ako nadje
    pom1 = glava;
    pom11 = NULL;
```

```
while(pom1 != NULL)
        {
            pom3 = pom1;
            pom2 = pom1->sledeci;
            indikator = 0;
            while(pom2 != NULL)
                {
                    if(pom1->izabrani_brojevi[pom1->runda-1] ==
pom2->izabrani_brojevi[pom2->runda-1])
                        {
                            pom3->sledeci = pom2->sledeci;
                            free(pom2);
                            pom2 = pom3->sledeci;
                            indikator = 1;
                            (*n)--;
                        } else{
                                pom3 = pom2;
                                pom2 = pom2->sledeci;
                            }
                }
```

```
if(indikator == 1)
                    {
                        if(pom1 == glava) //ako prvi nije jedinstven
                            {
                                glava = glava->sledeci;
                                free(pom1);
                                pom1 = glava;
                            }else //ako je negde u sredini
                                 {
                                    pom11->sledeci = pom1->sledeci;
                                    free(pom1);
                                    pom1 = pom11->sledeci;
                                 }
                        (*n)--;
                    }else{
                            pom11 = pom1;
                            pom1 = pom1->sledeci;
                          }
   } //kraj od glavne while petlje
   return glava;}
```

```
int main()
{

    IGRAC *lista;
    int n;

    scanf("%d",&n);
    getchar();//kupimo '\n' nakon scanf();

    lista = formiraj_listu(n);
    ispis_liste(lista);

    while (n > 3)
    {
        lista = nadji_jedinstvene(lista,&n);
        ispis_liste(lista);
        if(n > 3) lista = ucitaj_izabrani_broj(lista);
    }
    return 0;
}
```