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

Jun, 2020

Učenici osmog razreda polažu malu maturu onlajn preko platforme IMIMatura. Svi testovi se polažu istog dana, ali učenici imaju priliku da izaberu u koliko sati će raditi koji test, a ako žele, mogu istovremeno polagati više testova. Izrada svakog testa počinje na pun sat, a dužina trajanja zavisi od predmeta. Za svaki predmet je definisano koliko učenika i jednom trenutku može da polaže test. Svaki učenik prilikom prijave za polaganje unosi svoj ID (ceo broj), ime, prezime, mesto (nizovi karatktera), broj predmeta koji polaže (ceo broj) i za svaki predmet koji polaže ID predmeta (ceo broj) i vreme u koje želi da počne izradu testa (ceo broj koji označava pun sat i ima vrednosti od 8 do 22). Za predmete iz kojih se polažu testovi definisani su ID (ceo broj), naziv predmeta (niz karaktera), maksimalna dužina trajanja izražena u minutima (ceo broj) i kapacitet, odnosno broj učenika koji u jednom trenutku mogu da polažu test. Ukoliko učenik želi da polaganje testa iz nekog predmeta započne u satu kada su kapaciteti već popunjeni, za njega se početak polaganja pomera za prvi naredni sad kada ima mesta. Redosled prijava koja je izvršena unapred se uzima kao prioritet pri popunjavanju kapaciteta testova. Revidiranje stanja se vrši na svaki pun sat kada se unošenjem ID-jeva predmeta može štampati spisak učenika koji u tom trenutku polažu test ili započinju polaganje testa iz nekog predmeta. Napisati program koji učitava podatke, raspoređuje učenike po predmetima, omogućava pregled stanja na svaki pun sat.
(5 poena)

Za rešavanje problema napisati sledeće funkcije:

a) Definisati sve potrebne složene tipove podataka neophodne za rešavanje opisanog problema.
(3 poena, obavezno!)
b) Napisati funkciju **UcitajPredmete** koja iz datoteke *Predmeti.txt* učitava podatke o predmetima i formira listu/niz predmeta.
(3 poena, obavezno)
c) Napisati funkciju **NadjiPredmet** koja za dati ID predmeta vraća pokazivač na određeni predmet.
(3 poena, obavezno)
d) Napisati funkciju **UcitajUcenike** koja iz datoteke *Ucenici.txt* učitava podatke o učenicima i formira listu/niz učenika.
(4+2 poena)
e) Napisati funkciju **Rasporedjivanje** koja rasporedjuje učenike po predmetima.
(6 poena)
f) Napisati funkciju **KrajTesta** koja odjavljuje učenika sa određenog predmeta ukoliko mu je isteklo vreme za izradu testa.
(4 poena)

Pri učitavanju podataka podrazumevati da su svi podaci korektno zadati: ID-jevi jedinstveni i korektni, vremena u zadatom intervalu…
Funkcije **UcitajUcenike, Rasporedjivanje** i **KrajTesta** možete rešiti i drugačijom implementacijom funkcija. Dozvoljeno je proširivanje struktura i definisanje novih.

Zadatak se može rešavati korišćenjem povezanih lista, nizova ili kombinacijom.
Zadatak rešiti bez korišćenja globalnih promenljivih i bez unapred definisanih dužina korišćenih nizova.

Rešenje studenta

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

/* String makro-i za duzinu */
#define IME_LEN 20
#define PREZ_LEN 25
#define MESTO_LEN 30
#define NAZIV_LEN 16
#define REQ_LEN 255

/* Granice */
#define POCETAK 8
#define KRAJ 22

typedef struct ucenik
{
	int id;
	char *ime;
	char *prezime;
	char *mesto;
	int brPredmeta;
	int *predmetiID;
	int *vremePolaganjaSat;
	struct ucenik *sledeci;
} Ucenik;

typedef struct ucesnik
{
	Ucenik *ptr;
	struct ucesnik* sledeci;
}Ucesnik;

typedef struct predmet
{
	int id;
	char *naziv;
	int trajanjeMin;
	int kapacitet;
	int trenPrijavljeno;
	Ucesnik *ucesnici; /* lista*/
	struct predmet *sledeci;
} Predmet;

/* input_predmet: ime fajla u kome se nalaze podaci o predmetima */
const char * const input_predmet = "Predmeti.txt";

/* input_ucenik: ime fajla u kome se nalaze podaci o prijavama ucenika */
const char * const input_ucenik = "Ucenici.txt";

/* errMsgAlloc: poruka o gresci pri neuspesnom alociranju memorije */
char const * const errMsgAlloc = "neuspesno alociranje memorije";

/* errMsgLoad: poruka o gresci pri neuspesnom ucitavanju fajla */
char const * const errMsgLoad = "neuspesno ucitavanje fajla";

void ProveriVrednostPtr( void *ptr, char const * const msg );
void StampajUcesnike( char *naziv_predmeta, Ucesnik *ucesnici );
int UcitajUcenike( Ucenik **ucenici );
int UcitajPredmete( Predmet **predmeti );
void DodajUcenika( Ucenik **head, Ucenik *u );
void DodajPredmet( Predmet **head, Predmet *p );
void DodajUcesnika( Ucesnik **head, Ucesnik *u );
Predmet *NadjiPredmet( Predmet *predmeti, int id );
void Rasporedi( Predmet *predmeti, Ucenik *ucenici, int sat, int IDs[], int idCount );
void KrajTesta( Predmet *predmeti, int sat );

int main()
{
	Predmet *predmeti = NULL;
```

```
	Ucenik *ucenici = NULL;

	int i, j, k, brojPredmeta;;

	brojPredmeta = UcitajPredmete( &predmeti );
	UcitajUcenike( &ucenici );

	int *IDs = (int *) malloc( brojPredmeta * sizeof( int ) );
	ProveriVrednostPtr( (void *) IDs, errMsgAlloc );

	char req[REQ_LEN];
	for( i = POCETAK; i <= KRAJ; i++ )
	{
		printf("\nVreme: %d\n", i );
		printf("Unesite ID-jeve predmeta (u jednoj liniji):\n");

		fgets( req, REQ_LEN, stdin );
		req[ strlen(req) - 1 ] = '\0';

		j = 0;
		k = 0;

		IDs[k] = 0;
		while( req[j] )
		{
			if( req[j] >= '0' && req[j] <= '9')
				IDs[k] = 10 * IDs[k] + req[j] - '0';
			else if( req[j] == ' ' )
			{
				if( req[j+1] != ' ' && req[j+1] != '\t' )
					IDs[++k] = 0;
				else
					++j;
			}
			else
			{
				fprintf( stderr ,"ERROR: nepoznat karakter je ucitan prilikom unosa ID-
jeva\n");
				exit( EXIT_FAILURE );
			}

			++j;
		}

		Rasporedi( predmeti, ucenici, i, IDs, k+1 );
	}

	return 0;
}

/* ProveriVrednostPtr: proverava da li je vrednost pointera NULL,
   ukoliko jeste, ispisuje poruku o gresci i obustavlja rad programa */
void ProveriVrednostPtr( void *ptr, char const * const msg )
{
	if( ptr == NULL )
	{
		fprintf( stderr, "ERROR: %s\n", msg );
		exit( EXIT_FAILURE );
	}
}

void DodajPredmet( Predmet **head, Predmet *p )
{
	if( *head == NULL )
	{
		*head = p;
		return;
	}

	Predmet *trenutni = *head;

	while( trenutni->sledeci != NULL )
		trenutni = trenutni->sledeci;
```

```
	trenutni->sledeci = p;
}

void DodajUcenika( Ucenik **head, Ucenik *u )
{
	if( *head == NULL )
	{
		*head = u;
		return;
	}

	Ucenik *trenutni = *head;

	while( trenutni->sledeci != NULL )
		trenutni = trenutni->sledeci;

	trenutni->sledeci = u;
}

void DodajUcesnika( Ucesnik **head, Ucesnik *u )
{
	if( *head == NULL )
	{
		*head = u;
		return;
	}

	Ucesnik *trenutni = *head;

	while( trenutni->sledeci != NULL )
		trenutni = trenutni->sledeci;

	trenutni->sledeci = u;
}

int UcitajPredmete( Predmet **predmeti )
{
	FILE *f = fopen( input_predmet, "r" );
	ProveriVrednostPtr( (void *) f, errMsgLoad );

	int i, brPred;

	fscanf( f, "%d", &brPred );
	fgetc( f );

	Predmet *trenutni = NULL;
	char naziv[NAZIV_LEN+2];

	for( i=0; i < brPred; i++ )
	{
		trenutni = (Predmet *) malloc( sizeof( Predmet ) );
		ProveriVrednostPtr( (void *) trenutni, errMsgAlloc );

		fscanf( f, "%d", &trenutni->id );
		fgetc( f );

		fgets( naziv, NAZIV_LEN + 2, f );
		naziv[ strlen(naziv) - 1 ] = '\0';
		trenutni->naziv = (char *) malloc( ( strlen(naziv) + 1 ) * sizeof( char ) );
		ProveriVrednostPtr( (void *) trenutni->naziv, errMsgAlloc );
		strcpy( trenutni->naziv, naziv );

		fscanf( f, "%d", &trenutni->trajanjeMin );
		fgetc( f );

		fscanf( f, "%d", &trenutni->kapacitet );
		fgetc( f );

		trenutni->ucesnici = NULL;
		trenutni->sledeci = NULL;

		trenutni->trenPrijavljeno = 0;
```

```
                DodajPredmet( predmeti, trenutni );
        }

        fclose( f );

        return brPred;
}

/* NadjiPredmet: vraca pokazivac na predmet sa prosledjenim
 * ID-jem ukoliko takav postoji, u suprotnom vraca NULL */
Predmet *NadjiPredmet( Predmet *p, int id )
{
        while( p != NULL && p->id != id )
                p = p->sledeci;

        if( p == NULL )
        {
                printf( "Predmet sa datim id-jem ne postoji\n");
                return NULL;
        }
        else
                return p;
}

int UcitajUcenike( Ucenik **ucenici )
{
        FILE *f = fopen( input_ucenik, "r" );
        ProveriVrednostPtr( (void *) f, errMsgLoad );

        int i, j, brUcen;
        fscanf( f, "%d", &brUcen );
        fgetc( f );

        Ucenik *trenutni = NULL;
        char ime[IME_LEN+2], prezime[PREZ_LEN+2], mesto[MESTO_LEN+2];

        for( i = 0; i < brUcen; i++ )
        {
                trenutni = (Ucenik *) malloc( sizeof( Ucenik ) );
                ProveriVrednostPtr( (void *) trenutni, errMsgAlloc );

                fscanf( f, "%d", &trenutni->id );
                fgetc( f );

                fgets( ime, IME_LEN + 2, f );
                ime[ strlen( ime ) - 1 ] = '\0';
                trenutni->ime = (char *) malloc( ( strlen(ime) + 1 ) * sizeof( char ) );
                ProveriVrednostPtr( (void *) trenutni->ime, errMsgAlloc );
                strcpy( trenutni->ime, ime );

                fgets( prezime, PREZ_LEN + 2, f );
                prezime[ strlen( prezime ) - 1 ] = '\0';
                trenutni->prezime = (char *) malloc( ( strlen(prezime) + 1 ) * sizeof( char ) );
                ProveriVrednostPtr( (void *) trenutni->prezime, errMsgAlloc );
                strcpy( trenutni->prezime, prezime );

                fgets( mesto, MESTO_LEN + 2, f );
                mesto[ strlen( mesto ) - 1 ] = '\0';
                trenutni->mesto = (char *) malloc( ( strlen(mesto) + 1 ) * sizeof( char ) );
                ProveriVrednostPtr( (void *) trenutni->mesto, errMsgAlloc );
                strcpy( trenutni->mesto, mesto );

                fscanf( f, "%d", &trenutni->brPredmeta );
                fgetc( f );

                trenutni->predmetiID = (int *) malloc( trenutni->brPredmeta * sizeof(int) );
                ProveriVrednostPtr( (void *) trenutni->predmetiID, errMsgAlloc );

                trenutni->vremePolaganjaSat = (int *) malloc( trenutni->brPredmeta * sizeof(int) );
                ProveriVrednostPtr( (void *) trenutni->vremePolaganjaSat, errMsgAlloc );

                for(j = 0; j < trenutni->brPredmeta; j++)
```

```
		{
			fscanf( f, "%d", &trenutni->predmetiID[j] );
			fgetc( f );

			fscanf( f, "%d", &trenutni->vremePolaganjaSat[j] );
			fgetc( f );

		}

		trenutni->sledeci = NULL;

		DodajUcenika( ucenici, trenutni );
	}

	fclose( f );

	return brUcen;
}
/* Rasporedi: rasporedjuje ucenike na predmete;
 *
 * Prvo proverava da li su odjavljeni svi ucenici
 * koji su trebali do datog vremena 'sat' da zavrse test,
 * nakon toga dodaje nove prijave na predmet i na kraju
 * stampa sve prijavljene ucenike na predmetima ciji su
 * ID-jevi prosledjeni */
void Rasporedi( Predmet *predmeti, Ucenik *ucenici, int sat, int IDs[], int idCount )
{
	int i;
	Ucesnik *u = NULL;
	Predmet *trenPredmet = NULL;

	KrajTesta( predmeti, sat );

	/* Dodaje sve ucenike u listu ucesnika odredjenog predmeta ako
	 * njegov kapacitet nije prepunjen, u suprotnom, vreme polaganja
	 * ucenika se odlaze na naredni sat */
	while( ucenici != NULL )
	{
		for(i = 0; i < ucenici->brPredmeta; i++)
		{
			if( ucenici->vremePolaganjaSat[i] == sat )
			{
				trenPredmet = NadjiPredmet( predmeti, ucenici->predmetiID[i] );
				if( trenPredmet == NULL )
				{
					printf("ERROR: predmet sa ID-jem %d ne postoji\n", ucenici-
>predmetiID[i] );
					continue;
				}
				else if( trenPredmet->trenPrijavljeno < trenPredmet->kapacitet )
				{
					u = (Ucesnik *) malloc( sizeof( Ucesnik ) );
					ProveriVrednostPtr( (void *) u, errMsgAlloc );

					u->ptr = ucenici;
					u->sledeci = NULL;

					DodajUcesnika( &trenPredmet->ucesnici, u );
					trenPredmet->trenPrijavljeno++;
				}
				else
					ucenici->vremePolaganjaSat[i]++;
			}
		}
		ucenici = ucenici->sledeci;
	}

	/* Stampa ucenike koji trenutno rade ili zapocinju
	 * rad test iz predmeta sa prosledjenim ID-jevima */
	for( i = 0; i < idCount; i++ )
	{
		trenPredmet = NadjiPredmet( predmeti, IDs[i] );
		StampajUcesnike( trenPredmet->naziv, trenPredmet->ucesnici );
```

```
	}
}

/* StampajUcesnike: Stampa ucenike koji trenutno
 * rade ili zapocinju rad test iz datog predmeta */
void StampajUcesnike( char *naziv_predmeta, Ucesnik *ucesnici )
{
	printf("\n%s:\n", naziv_predmeta );
	printf("=================\n");

	if( ucesnici == NULL )
	{
		printf("Nema prijavljenih\n\n");
		return;
	}

	printf("%-6s%-*s%-*s%-*s\n", "ID",
	                                          IME_LEN, "Ime",
	                                       PREZ_LEN, "Prezime",
	                                       MESTO_LEN, "Mesto" );
	printf("-----------------------------------------------------------------\n");
	while( ucesnici != NULL )
	{
		printf("%-6d%-*s%-*s%-*s\n", ucesnici->ptr->id,
		                                                IME_LEN, ucesnici->ptr->ime,
		                                                PREZ_LEN, ucesnici->ptr->prezime,
		                                                MESTO_LEN, ucesnici->ptr->mesto );
		ucesnici = ucesnici->sledeci;
	}
}

/* KrajTesta: odjavljuje ucenika sa
 * predmeta kada se test zavrsi */
void KrajTesta( Predmet *predmeti, int sat )
{
	int i;
	Predmet *trenPredmet = predmeti;
	Ucesnik *trenUcesnik = NULL, *prethUcesnik;

	while( trenPredmet != NULL )
	{
		trenUcesnik = trenPredmet->ucesnici;
		prethUcesnik = NULL;

		while( trenUcesnik != NULL )
		{
			i = 0;
			/* trazi indeks elemenata koji se odnosi na trenutni predmet;
			 * sigurno ce postojati id koji jeste jednak id-ju predmeta jer
			 * je ucesnik upisan na dati predmet */
			while( trenUcesnik->ptr->predmetiID[i] != trenPredmet->id )
				++i;

			if( trenUcesnik->ptr->vremePolaganjaSat[i]*60 + trenPredmet->trajanjeMin <= sat*60 )
			{
				trenUcesnik = trenUcesnik->sledeci;
				if( prethUcesnik != NULL )
					prethUcesnik->sledeci = trenUcesnik;
				else
					trenPredmet->ucesnici = trenUcesnik;

				trenPredmet->trenPrijavljeno--;
			}
			else
			{
				prethUcesnik = trenUcesnik;
				trenUcesnik = trenUcesnik->sledeci;
			}
		}
		trenPredmet = trenPredmet->sledeci;
	}
}
```

Strukture podataka i algoritmi 1
Test – max 20 poena

Jun, 2020

Ime i prezime	Broj indeksa	Broj poena

1. (1.5) Šta je rezultat sledećih kodova?

```
int a = 27, b = 13, c = 18;
printf("%d",(a & b ) | c);
```

27

```
int a = 28;
printf("%d", (a << 4) ^ 10);
```

458

```
int s = 0;
for (b = 15;  b; b >>= 1)
    s += b;
printf("%d",s);
```

26

2. (2.0) Sortiraj niz brojeva 17 3 22 13 78 28 30 54 47 6 koristeći Bubble sort i ispisati svaki korak prilikom sortiranja brojeva.

```
3   17   13   22   28   30   54   47    6   78
3   13   17   22   28   30   47    6   54   78
3   13   17   22   28   30    6   47   54   78
3   13   17   22   28    6   30   47   54   78
3   13   17   22    6   28   30   47   54   78
3   13   17    6   22   28   30   47   54   78
3   13    6   17   22   28   30   47   54   78
3    6   13   17   22   28   30   47   54   78
```

3. (2.5) Data je struktura:

```
struct node{
  float x;
  struct node* next;
};
struct node* head;
```

Formirana je lista na čiji prvi element pokazuje promenljiva head i čiji elementi imaju vrednosti 1.2 2.3 4.5 3.2 10.0 7.8 12.1. Napisati niz komandi kojima se dobija lista 1.2 2.3 4.5 5.3 19.2 14.3 3.2 10.0 7.8 12.1.

```
struct node *temp, *novi;
  temp = head->next->next;
  for(i=0;i<3;i++){
        novi=(struct node*)malloc(sizeof(struct node));
        scanf("%f",&novi->x);
        novi->next=temp->next;
        temp->next=novi;
        temp=novi;
  }
```

4. (2.5) Za strukturu iz zadatka 3 napisati funkciju koja dodaje element ispred (levo od) elementa koji je prvi jednak vrednosti koja se dodaje.

Primer: 1 9 5 14 3, dodaje se broj 5 Izlaz: 1 9 5 5 7 14 3

```
struct node* dodaj(struct node *p, float val){
  struct node *temp, *novi;
  novi=(struct node*)malloc(sizeof(struct node));
  novi->x = val;
  if (p->x == val){
        novi->next = p;
        p = novi;
        return p;
  }
  temp = p;
  while(temp->next && temp->next->x != val)
        temp =temp->next;
  if(temp->next){
        novi->next=temp->next;
        temp->next=novi;
  }
}
```

5. (1.5) Napisati uslovni izraz koji odgovara sledećem kodu

```
if(x%4==0)
  y = x * 4;
else if(x%5==0)
  y = sqrt(x, 2);
else
  y = x * x;
```

```
y = (x%4)? (x%5 ? x*x : sqrt(x)) : x*4;
```

6. (1.0) Zaokružiti izraze/izraz koji su/je ekvivalatan/ni sa izrazom `a[i][j][k][l]`

a. `(*(*(*(*a+i)+j)+k)+l)` b. `*(*(*(*(a+i)+j)+k)+l)`

c. `*(((a+i)+j)+k+l)` d. `(*(a+i)+j+k+l)`

7. (1.5) Koliko memorijskog prostora zauzima promenljiva koja je tipa `struct geom_figura`?

```
struct geom_figura {
  int tip;
  double koord[2];
  union {
      struct { double r; } kruznica;
      struct { double a; } kvadrat;
      struct { double a, b; } pravougaonik;
      struct { double a, b, c; } trougao;
  } figura;
};
```

```
sizeof(struct geon_figura) <= sizeof(int) + sizeof(double) + sizeof(figura) =
          = sizeof(int) + 2*sizeof(double) + 3*sizeof(double)
```

8. (2.5) Napisati program pomeri koji izvršava bitovsku operaciju pomeranja u levo za dva zadata broja x i y, tako što broj x pomeri u levo y puta. Brojevi x i y su predati prilikom pokretanja programa na sledeći način:

```
./pomeri 4 5
```

```
main(int argc, char **argv)
{
      int x,y;
      x = atoi(argv[1]);
      y = atoi(argv[2]);
      printf("%d %d %d",x,y, x<<y);
}
```

9. (1.0) Šta je rezultat sledećeg koda?

```
#include <stdio.h>
main(){
    int w=1;
    float y=4.5;
    if (y/w) printf("%d\n",(int)(w--?y:5*w));
    w?printf("Test 1\n"):printf("%d\n",(2+w)/++w);
}
```

4
2

10. (1.0) Šta je rezultat sledećeg koda?

```
#include <stdio.h>
#define PODELI(x,y) (x/y)
main() {
    int w=8;
    float y=4,0;
    printf("%.2f\n", PODELI (w + (int)y, w - y));
}
```

4.00

11. (1.5) Šta je rezultat sledećeg koda

```
#include <stdio.h>
main(){
      int x, y, d = 0;
      scanf("%d%d", &x, &y);

      while (x>0)
      {
            y >>= --x/2;
            printf("%d %d\n",x, y);
            if (!y) continue;
            d++;
      }

      printf("%d\n",d); }
```

Ako se kao vrednosti promenljivih x i y unesu:

5 17	12 32	6 31
5	1	2

12. (1.5) Šta je rezultat sledećeg koda?

```
char *str[] = { "februar", "mart", "april", "maj"};
char **s1 = str;
int n = sizeof(str)/sizeof(str[0])-1;
while(n--)
printf("%c\n",*(++(*s1++)));
```

e
a
p