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

Jul, 2020

Na takmičenju za orijentaciju u prostoru postavlja se poligon sa fiksnim brojem tačaka, pri čemu se za svaku tačku zna njen ID (ceo broj), njene koordinate u prostoru (*x*, *y* - celi brojevi) i koliko je vremena potrebno za rešavanje problema postavljenog na toj tački (ceo broj). Ekipe koje se prijavljuju za ovo takmičenje imaju svoje nazive (niz karaktera), svoj ID (ceo broj), a od organizatora dobijaju u kom vremenskom trenutku će krenuti (ceo broj) i kojim redom će obilaziti tačke (niz celih brojeva – IDjevi tačaka). Svaka ekipa mora da obiđe sve tačke zadatim redom. Kada ekipa stigne na neku tačku i na njoj se već nalazi neka ekipa, ona moraju da sačekaju da prethodna ekipa završi sa rešavanjem zadatog problema, pa tek onda da rešava problem. Na tački može da se formira i red, pri čemu ekipe prolaze tačku onim redom kojim su stigle. Pobednik je ekipa koja za najkraće vreme obiđe ceo poligon, uzimajući u obzir vremana potrebna da se stigne iz jedne tačke u drugu, vremena potrebna da se reše zadaci i vremena čekanje na tačkama, Odnosno gleda se vrema od trenutka polaska do trenutka završavanja problema na poslednjoj tački. Pri određivanju vremena da se iz jedne tačke pređe u drugu, uzima se da se ekipa kreće po najkraćoj liniji i da joj je za prelazak jedinične dužine potrebna jedna jedinica vremena. Ekipa polazi od prve tačke koju je dobila, tako da ne troši vreme na stizanje do te tačke. Napisati program koji učitava podatke, simulira trku i ispiuje rang listu po vremenima.
(3 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 **UcitajTacke** koja iz datoteke *Tacke.txt* učitava podatke o tačkama i formira listu/niz tačaka.
(3 poena, obavezno)
c) Napisati funkciju **NadjiTačku** koja za dati ID tačke vraća pokazivač na određenu tačku.
(3 poena, obavezno)
d) Napisati funkciju **UcitajEkipe** koja iz datoteke *Ekipe.txt* učitava podatke o ekipama i formira listu/niz ekipa.
(4 poena)
e) Napisati funkciju **Rastojanje** koja za dve tačke određuje najkraće rastojanje među njima.
(2 poena)
f) Napisati funkciju **Trka** koja koja simulira trku.
(6+1 poena)
g) Napisati funkciju **RangLista** koja ispisuje rang listu ekipa.
(3+2 poena)

Pri učitavanju podataka podrazumevati da su svi podaci korektno zadati
Dozvoljeno je proširivanje struktura i definisanje novih, kao i definisanje dodatnih funkcija.

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

Rastojanje dve tačke $T_1(x_1, y_1)$ i $T_2(x_2, y_2)$ može da se računa po obrascu $d(T_1, T_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Rešenje studenta

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

#define NAZIV_LEN 16 /* String makro za naziv ekipe */

typedef struct tacka {
    int id;
    int x;
    int y;
    int vreme;
    int zauzeto; //0 ako nije, 1 ako jeste
    struct tacka *sledeci;
} Tacka;

typedef struct ekipa {
    char *naziv;
    int id;
    int start;
    int *IDs;
    Tacka *tacke;
    float prolaznoVreme;
    struct ekipa *sledeci;
} Ekipa;

void DodajTacku(Tacka **head, Tacka *t)
{
      if (*head==NULL)
      {
            *head=t;
            return;
      }

      Tacka *trenutna=*head;

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

      trenutna->sledeci=t;
}

void DodajEkipu(Ekipa **head, Ekipa *e)
{
      if (*head==NULL)
      {
            *head=e;
            return;
      }

      Ekipa *trenutna=*head;

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

      trenutna->sledeci=e;
}

int UcitajTacke(Tacka **tacke)
{
      FILE *f=fopen("Tacke.txt","r");
      if (f==NULL)
      {
```

```
		printf("greska kod ucitavanja datoteke!!!");
		exit(1);
	}

	int i,brTacaka;

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

	Tacka *trenutna=NULL;

	for(i=0;i<brTacaka;i++)
	{
		trenutna=(Tacka*)malloc(sizeof(Tacka));
		if (trenutna==NULL)
		{
			printf("greska kod alokacije");
			exit(1);
		}

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

		fscanf(f,"%d %d",&trenutna->x, &trenutna->y);
		fgetc(f);

		fscanf(f,"%d",&trenutna->vreme);
		fgetc(f);

		trenutna->zauzeto=0;
		trenutna->sledeci=NULL;

		DodajTacku(tacke,trenutna);
	}

	fclose(f);

	return brTacaka;
}

Tacka *NadjiTacku(Tacka *t, int id)
{
	while (t!=NULL && t->id != id)
	     t=t->sledeci;
	if (t==NULL)
	{
		printf("Tacka sa datim id-jem ne postoji\n");
		return NULL;
	}
	else
		 return t;
}

int UcitajEkipe(Ekipa **ekipe,int brojTacaka)
{
	FILE *f=fopen("Ekipe.txt","r");
	if (f==NULL)
	{
		printf("greska kod ucitavanja datoteke!!!");
		exit(1);
	}

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

```
	Ekipa *trenutna=NULL;
	char naziv[NAZIV_LEN+2];

	for (i=0;i<brEkipa;i++)
	{
		trenutna=(Ekipa*)malloc(sizeof(Ekipa));
		if (trenutna==NULL)
		{
			printf("greska kod alokacije");
			exit(1);
		}

		fgets(naziv, NAZIV_LEN+2,f);
		naziv[strlen(naziv)-1] = '\0';
		trenutna->naziv = (char*)malloc( (strlen(naziv)+1) * sizeof(char));
		if (trenutna->naziv==NULL)
		{
			printf("greska kod alokacije");
			exit(1);
		}
		strcpy(trenutna->naziv,naziv);

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

		fscanf(f,"%d",&trenutna->start);
		fgetc(f);

		trenutna->IDs=(int*)malloc( brojTacaka*sizeof(int));
		if (trenutna->IDs==NULL)
		{
			printf("greska kod alokacije");
			exit(1);
		}

		for (j=0;j<brojTacaka;j++)
		{
			fscanf(f,"%d",&trenutna->IDs[j]);
			fgetc(f);
		}

		trenutna->prolaznoVreme=0.0;
		trenutna->sledeci=NULL;
		trenutna->tacke=NULL;

		DodajEkipu(ekipe,trenutna);
	}

	fclose(f);

	return brEkipa;
}

float rastojanje(float x, float y, float a, float b){
	return sqrt((x-a)*(x-a)+(y-b)*(y-b));
}

void Trka(Tacka *tacke, int brojTacaka, Ekipa *ekipe)
{
   int i;
   Tacka *trenTacka = NULL;
   Tacka *prethTacka = NULL;
   Tacka *point=NULL;
   Ekipa *ekipe1=ekipe;
```

```
   while (ekipe1->sledeci!=NULL)
   {
          for (i=0;i<brojTacaka;i++)
          {
                 trenTacka=NadjiTacku(tacke, ekipe1->IDs[i]);
                 if (trenTacka==NULL)
                 {
                        printf("ne postoji tacka sa ovim ID-jem");
                        continue;
                 }
                 else if (trenTacka->zauzeto==0)
                 {
                        DodajTacku(&ekipe1->tacke, trenTacka);
                        trenTacka->zauzeto=1;
                        ekipe1->prolaznoVreme+=trenTacka->vreme;
                        if (i!=0)
                        {
                               prethTacka=NadjiTacku(tacke, ekipe1->IDs[i-1]);
                               if (prethTacka==NULL)
                               {
                                      printf("ne postoji tacka sa ovim ID-jem");
                                      continue;
                               }
                               else ekipe1->prolaznoVreme+=rastojanje(prethTacka->x,prethTacka-
>y,trenTacka->x,trenTacka->y);
                        }
                        //posto trenutna tacka nije zauzeta ekipa nece utrositi vreme cekajuci
da neka druga ekipa zavrsi zadatak na toj tacki
                 }
                 else
                 {
                        ekipe1->prolaznoVreme+=trenTacka->vreme;
                        if (i!=0)
                        {
                               prethTacka=NadjiTacku(tacke, ekipe1->IDs[i-1]);
                               if (prethTacka==NULL)
                               {
                                      printf("ne postoji tacka sa ovim ID-jem");
                                      continue;
                               }
                               else ekipe1->prolaznoVreme+=rastojanje(prethTacka->x,prethTacka-
>y,trenTacka->x,trenTacka->y);
                        }
                        Ekipa *ekipe2=ekipe;
                        while (ekipe2->sledeci!=NULL)
                        {
                               Tacka *point=NadjiTacku(ekipe2->tacke, ekipe1->IDs[i]);
                               if (point==NULL)
                               {
                                      printf("ne postoji tacka sa ovim ID-jem");
                                      continue;
                               }
                               else
                                  if (point->zauzeto==1) break;
                               ekipe2=ekipe2->sledeci;
                        }
                        ekipe1->prolaznoVreme+=(ekipe2->start+ekipe2->prolaznoVreme+point-
>vreme)-(ekipe1->start+ekipe1->prolaznoVreme);
                 }
          }
          ekipe1=ekipe1->sledeci;
   }
}
```

```
void RangLista(Ekipa *e)
{
	Ekipa *pom1,*pom2,*min,*prethodna,*e1;

	e1 = NULL;
	while(e != NULL)
	{
		prethodna = NULL;
		min = pom1 = e;
		pom2 = e->sledeci;

		while (pom2!=NULL)
		{
			if (pom2->prolaznoVreme<min->prolaznoVreme)
			{
				min=pom2;
				prethodna=pom1;
			}
			pom1=pom2;
			pom2=pom2->sledeci;

			if (prethodna==NULL) e=min->sledeci;
			else
			   prethodna->sledeci=min->sledeci;
			min->sledeci=NULL;

			if (e1==NULL)
			   e1=min;
			else
			{
				pom1=e1;
				while (pom1->sledeci!=NULL)
				   pom1=pom1->sledeci;
				pom1->sledeci=min;
			}
		}
	}
	if (e1==NULL)
	{
		printf("Nema prijavljenih\n\n");
		return;
	}
	while (e1!=NULL)
	{
		printf("%s %d %f\n",e1->naziv,e1->id,e1->prolaznoVreme);
		e1=e1->sledeci;
	}
}

int main()
{
	Tacka *tacke=NULL;
	Ekipa *ekipe=NULL;
	int brojTacaka, brojEkipa;

	brojTacaka=UcitajTacke(&tacke);
	brojEkipa=UcitajEkipe(&ekipe,brojTacaka);
	RangLista(ekipe);

	return 0;
}
```

Strukture podataka i algoritmi 1
Test – max 20 poena

Jul, 2020

Ime i prezime	Broj indeksa	Broj poena

1. (1.5) Šta je rezultat sledećih kodova?

```
int a = 25;
printf("%d",a | 14);
```

31

```
int a = 23;
printf("%d",a >> 2);
```

5

```
int b, s = 1;
for (b = 3;  b<100 ; b <<= 2)
      s *= b;
printf("%d",s);
```

1728

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

```
Manji niz u 0 iteraciji          17
Veci niz u 0 iteraciji

Manji niz u 1 iteraciji          3
Veci niz u 1 iteraciji           17

Manji niz u 2 iteraciji          3 17 22
Veci niz u 2 iteraciji

Manji niz u 3 iteraciji          3 13
Veci niz u 3 iteraciji           22 17

Manji niz u 4 iteraciji          3 13 17 22 78
Veci niz u 4 iteraciji

Manji niz u 5 iteraciji          3 13 17 22 28
Veci niz u 5 iteraciji           78

Manji niz u 6 iteraciji          3 13 17 22 28 30
Veci niz u 6 iteraciji           78

Manji niz u 7 iteraciji          3 13 17 22 28 30 54
Veci niz u 7 iteraciji           78

Manji niz u 8 iteraciji          3 13 17 22 28 30 47
Veci niz u 8 iteraciji           78 54

Manji niz u 9 iteraciji          3 6
Veci niz u 9 iteraciji           78 54 47 30 28 22 17 13


Na kraju                         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 dodaje tri elementa 1.3 0.2 11.3, tako da se svaki od ovih elemenata dodaje iza (desno od) prvog elementa čija je vrednost veća od elementa koji se dodaje. Na ovaj način dobiće se lista 1.2 0.2 2.3 1.3 4.5 3.2 10.0 7.8 12.1 11.3 .

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

4. (2.5) Za strukturu iz zadatka 3 napisati funkciju koja dodaje element -1.0 ispred (levo od) elementa čija je cifra desetica manja od cifara desetica svih ostalih elemenata liste. Ukoliko postoji više elemenata sa istom cifrom desetica (minimalnom) novi element dodati ispred prvog takvog elementa.

Primer: 25.5 18.24 105.4 418.0 1004.7 Izlaz: 25.5 18.24 -1.0 105.4 418.0 1004.7

```
struct node* dodaj(struct node *p){
  struct node *temp, *novi, *preth;
  int min;
  novi=(struct node*)malloc(sizeof(struct node));
  novi->x = -1.0;
  temp = p;
  preth = NULL;
  min = ((int)temp->x % 100)/10;
  while(temp->next){
    if(((int)temp->next->x%100)/10 < min){
          min = ((int)temp->next->x%100)/10;
          preth = temp;
    }
    temp = temp->next;
  }

  if(preth){
    novi->next = preth->next;
    preth->next = novi;
  }
  else{
    novi->next = p;
    p = novi;
  }
  return p;
}
```

5. (1.0) Napisati uslovni izraz koji odgovara sledećem kodu

```
if(!x)
  y = x * 4;
else if(x/4)
  y = sqrt(x);
else
  y = x * x;
```

```
y=!x? x*4:(x/4?sqrt(x):x*x);
```

6. (2.5) Napisati deo koda koji korišćenjem **isključivo pokazivačke aritmetike** (nije dozvoljeno koristiti uglaste zagrade) ispisuje elemente matrice celih brojeva `A` koja ima `n` vrsta i `m` kolona. Svaku vrstu matrice ispisati u posebnom redu, a elemente jedne vrste razdvojiti razmakom.

```
int i,j;
for(i=0; i<n; i++){
  for(j=0; j<m; j++)
        printf("%d ",*(*(a+i)+j));
  printf("\n");
}
```

7. (1.0) Koliko najmanje memorijskog prostora (u bajtovima) zauzima promenljiva koja je tipa struct student ukoliko promenljiva tipa char zauzima 1 bajt, int 4 bajta i double 8 bajtova?

```
struct student
{
    char ime[20];
    int godina;
    double prosek;
    union
    {
        struct { char imeOca[20]; char imeMajke[20]; } saRoditeljima;
        struct { char imeDoma[20]; int samofinansiranje;} studentskiDom;
        struct { char adresa[20]; double kirija; } privatno;
    } mestoStanovanja;
};
```

```
72
```

8. (2.5) Napisati program `palindrom` koji proverava da li je argument koji je poslat preko komandne linije palindrom:

```
   ./palindrom anavolimilovana
main(int argc, char** argv){
  int i = 0, n = strlen(argv[1]);
  while (i < n/2 && argv[1][i] == argv[1][n-i-1])
    i++;
  if (i >= n/2)
    printf("Jeste palindrom\n");
  else
    printf("Nije palindrom\n");
}
```

9. (1.0) Šta je rezultat sledećeg koda?

```
#include <stdio.h>
main(){
      int w=5;
      float y=5.5;
      if ((int)(y/w)) printf("%.2f\n",(float)(--w-4?2*y:w/4));
      w-- - 4?printf("Prosao uslov\n"):printf("%.2f\n",((float)(6+w)/2));
}
```

```
1.00
4.50
```

10. (1.0) Šta je rezultat sledećeg koda?

```
#include <stdio.h>
#define STAR(x,y,z) (x*y+z)
main() {
    int a = 1, b = 5;
    float c = 7.5;
    printf("%.2f\n", STAR(a+b, a-b, b+c));
}
```

```
13.5
```

11. (1.0) Šta je rezultat sledećeg koda

```
#include <stdio.h>
main(){
    int x, y, d = 0;
    scanf("%d%d", &x, &y);
    while (x>0){
        x >>= 2;
        if (x & y) break;
        d++;
     }
    printf("%d\n",d);}
```

Ako se kao vrednosti promenljivih x i y unesu:

46 17	192 3
0	2

12. (1.5) Šta je rezultat sledećeg koda?

```
struct cvor{
      int d;
      char s[10];
};
main(){
      struct cvor a[] = {'a', "Pera" , 'b', "Mika", 'c', "Laza" };
      struct cvor *ptr=a;
      int n = sizeof(a)/sizeof(*a);
      while(--n)
            printf("%d %c\n",ptr->d,*((++ptr)->s));
}
```

```
98 M
99 L
```