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

September, 2019

Vremenske prilike se beleže u nekom periodu na više geografskih lokacija. Za svaku lokaciju se zna njen naziv(niz karekater) i njena lokacija (dva realna broja, koordinate u pravouglom koordinatnom sistemu). Takođe svaka lokacija dobija identifikacioni broj (ceo bro, počev od broja 1). Kada se lokacija dodaje u spisak lokacija koje se prate ona dobija prvi slobodan broj, a kada se uklanja njen broj se oslobađa. Na svakoj lokaciji se beleže temperatura, pritisak, vlažnost, jačina vetra (sve navedene vrednosti su realni brojevi) i pravac vetra (niz karaktera). Priliko očitavanja potaka beleži se tačan datum (u formatu gggg-mm-dd) i vreme (u formatu hh:mm). Napisati program koji iz datoteke učitava podatke od lokacijama, na kojima se beleže vremenske prilike, dodeljujući svakoj lokaciji identifikacioni broj. Podaci o vremenskim prilikama zabeleženim na lokacijama se, takodje, učitavaju iz datoteke. Omogućiti pretraživanje zabeleženih podataka po datumu i vremenskom intervalu.
(4 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 **UcitajLokacije** koja iz datoteke *Lokacije.txt* učitava podatke o lokacijama na kojima se beleže vremenske prilike i formira listu/niz lokacija.
(3 poena, obavezno!)

c) Napisati funkciju **NadjiLokaciju** koja za dati identifikacioni broj lokacije vraća pokazivač na odgovarajuću lokaciju u listi/nizu lokacija. Ukoliko lokacija ne postoji vratiti prazan pokazivač.
(3 poena, obavezno)

d) Napisati funkciju **UcitajPodatke** koja iz datoteke *Podaci.txt* učitava podatke o vremenskim prilikama. Na svim lokacijama podaci se nalaze u istom formatu, najpre identifikacioni broj lokacije, a zatim podaci onim redom kojim su dati u opisu problema.
(4 poena)

e) Napisati funkciju **PronadjiPodatke** koja za dati datum i vremenski interval u toku tog dana kreira listu/niz očitanih podataka sa svih lokacija.
(5 poena)

f) Napisati funkciju **ProsekTemp** koja za dati datum i vremenski interval u toku tog dana određuje prosečnu temperaturu za sve podatke i lokacije očitane u tom periodu.
(3 poena)

g) Napisati funkciju **IzbrišiLokaciju** koja za dati identičikacioni broj lokacije, uklanja lokaciju i sve njene podatke.
(3+2 poena)

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

Strukture podataka i algoritmi 1
Test – max 20 poena

Septembar, 2019

1. **(Obavezno!)** Svi obavezni delovi iz zadatka o merenju vremenskih prilika.
2. Napisati funkciju koja u datu pointersku listu realnih brojeva formira novu liste poemerenjam elementa za k pozicija u desno. Ukoliko je k veće od broja elemenata u listi, lista ostaje nepromenjena.
   Primer. Ulaz: 3 4 2 8 9 k = 2 Izlaz: 8 9 3 4 2
3. Data je struktura

```
struct element{
    int d;
    struct element *par;
};
```

   I niz `struct element *a`. Svaki element niza `a`, može da pokazuje na neki element istog niza. Napisati funkciju koja će za datu poziciju elementa u nizu `a` ispisati vrednost i indeks elementa na koji on pokazuje.
4. Za strukturu i niz `a` iz zadatka 3 napisati funkciju koja će polazeći od prvog elementa niza `a`, koristeći pokazivače, a ne indekse, ići preko elemenata niza `a`, pri čemu će svakom elementu preko koga pređe da smanji vrednost za 1. Ciklus se zaustavlja kada se stigne do elementa čija je vrednost 0 ili kad se stigne do elementa koji ne pokazuje nigde. Pretpostaviti da je niz dobro zadat, pa će bar jedna od uslova biti ispunjen.
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;
            if (!y) break;
            d++;
      }

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

   Ako se kao vrednosti promenljivih `x` i `y` unesu:

   a. 3 17 b. 15 31 c. 16 32

Rešenje studenta

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

typedef struct lokacija {
   char *naziv;
   float x, y;
   int ID;
   struct lokacija *sledeci;
} lokacija;

typedef struct podaci {
   float temperatura;
   float pritisak;
   float vlaznost;
   float jacina;
   char *pravac;
   char *datum;
   char *vreme;
   lokacija *niz_lokacija;
   struct podaci *sledeci;
} podaci;

lokacija* UcitajLokacije (lokacija *lokacije) {
   lokacija *novaL, *pomL;
   int i, n;
   char naziv[30];

   FILE *f = fopen("Lokacije.txt", "r");
   if (f == NULL) {
         printf("Greska\n");
         exit(0);
   }

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

   lokacije = NULL;
   for (i = 0; i < n; i++) {
         novaL = (lokacija*)malloc(sizeof(lokacija));
         novaL->ID = i + 1;

         fgets(naziv, 30, f);
         naziv[strlen(naziv) - 1] = '\0';
         novaL->naziv = (char*)malloc(strlen(naziv) + 1);
         strcpy(novaL->naziv, naziv);

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

         novaL->sledeci = NULL;

         if (lokacije == NULL)
               lokacije = novaL;
         else
               pomL->sledeci = novaL;
         pomL = novaL;
```

```
   }

   fclose(f);
   return lokacije;
}

lokacija *NadjiLokaciju (int ID, lokacija *lokacije) {
   lokacija *trazena = NULL;

   while(lokacije) {
         if(lokacije->ID == ID) {
               trazena = lokacije;
               break;
         }
         lokacije = lokacije->sledeci;
   }

   return trazena;
}

podaci* UcitajPodatke (podaci *pod, lokacija *lokacije) {
   podaci *noviP, *pomP;
   lokacija *pomL;
   int i, n, ID;
   char naziv[30];

   FILE *f = fopen("Podaci.txt", "r");
   if (f == NULL) {
         printf("Greska\n");
         exit(0);
   }

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

   pod = NULL;
   for (i = 0; i < n; i++) {
         noviP = (podaci*)malloc(sizeof(podaci));

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

         pomL = NadjiLokaciju(ID, lokacije);
         noviP->niz_lokacija = pomL;

         fscanf(f,   "%f%f%f%f",   &noviP->temperatura,   &noviP->pritisak,   &noviP-
>vlaznost, &noviP->jacina);
         fgetc(f);

         fgets(naziv, 30, f);
         naziv[strlen(naziv) - 1] = '\0';
         noviP->pravac = (char*)malloc(strlen(naziv) + 1);
         strcpy(noviP->pravac, naziv);

         fgets(naziv, 30, f);
         naziv[strlen(naziv) - 1] = '\0';
         noviP->datum = (char*)malloc(strlen(naziv) + 1);
         strcpy(noviP->datum, naziv);
```

```
		fgets(naziv, 30, f);
		naziv[strlen(naziv)] = '\0';
		noviP->vreme = (char*)malloc(strlen(naziv) + 1);
		strcpy(noviP->vreme, naziv);

		noviP->sledeci = NULL;

		if (pod == NULL)
			pod = noviP;
		else
			pomP->sledeci = noviP;
		pomP = noviP;
	}

	fclose(f);
	return pod;
}

podaci* PronadjiPodatke (char *datum, char *vreme1, char *vreme2, podaci *pod) {
	podaci *noviP, *pomP, *pomNovi, *novi_podaci;
	int sat1, sat2, sat3, minut1, minut2, minut3;

	sat1 = (vreme1[0] - '0')*10 + (vreme1[1] - '0');
	sat2 = (vreme2[0] - '0')*10 + (vreme2[1] - '0');
	minut1 = (vreme1[3] - '0')*10 + (vreme1[4] - '0');
	minut2 = (vreme2[3] - '0')*10 + (vreme2[4] - '0');

	pomP = pod;
	novi_podaci = NULL;

	while(pomP) {
		sat3 = (pomP->vreme[0] - '0')*10 + (pomP->vreme[1] - '0');
		minut3 = (pomP->vreme[3] - '0')*10 + (pomP->vreme[4] - '0');

		if ((strcmp(datum, pomP->datum) == 0) && (sat3 > sat1 && sat3 < sat2) ||
			((sat3 == sat1 && minut3 >= minut1) || (sat3 == sat2 && minut3 <=
minut2))) {
				noviP = (podaci*)malloc(sizeof(podaci));
				noviP->temperatura = pomP->temperatura;
				noviP->pritisak = pomP->pritisak;
				noviP->vlaznost = pomP->vlaznost;
				noviP->jacina = pomP->jacina;
				noviP->pravac = (char*)malloc(strlen(pomP->pravac) + 1);
				strcpy(noviP->pravac, pomP->pravac);
				noviP->datum = (char*)malloc(strlen(pomP->datum) + 1);
				strcpy(noviP->datum, pomP->datum);
				noviP->vreme = (char*)malloc(strlen(pomP->vreme) + 1);
				strcpy(noviP->vreme, pomP->vreme);
				noviP->niz_lokacija = pomP->niz_lokacija;
				noviP->sledeci = NULL;

				if (novi_podaci == NULL)
					novi_podaci = noviP;
				else
					pomNovi->sledeci = noviP;
				pomNovi = noviP;
		}

		pomP = pomP->sledeci;
```

```
   }

   return novi_podaci;
}

void ProsekTemp (char *datum, char *vreme1, char *vreme2, podaci *pod) {
   podaci *trazeni;
   float suma = 0;
   int br = 0;

   trazeni = PronadjiPodatke(datum, vreme1, vreme2, pod);

   while(trazeni) {
         suma += trazeni->temperatura;
         br += 1;
         trazeni = trazeni->sledeci;
   }

   printf("Prosek svih temperatura u datom vremenskom intervalu je %.2f\n",
suma/br);
}

void IzbrisiLokaciju (int ID, podaci **pod, lokacija **lokacije) {
   podaci *prethodniP, *trenutniP;
   lokacija *prethodnaL, *trenutnaL;

   prethodniP = NULL;
   trenutniP = *pod;

   while(trenutniP) {
         if (trenutniP->niz_lokacija->ID == ID) {
               trenutnaL = trenutniP->niz_lokacija;
               prethodnaL = NULL;

               if(trenutnaL != (*lokacije)) {
                     prethodnaL = (*lokacije);
                     while (prethodnaL->sledeci != trenutnaL)
                           prethodnaL = prethodnaL->sledeci;
               }

               if (prethodnaL == NULL) {
                     *lokacije = (*lokacije)->sledeci;
                     free(trenutnaL);
               }
               else {
                     prethodnaL->sledeci = trenutnaL->sledeci;
                     free(trenutnaL);
               }

               if (prethodniP == NULL) {
                     *pod = (*pod)->sledeci;
                     free(trenutniP);
                     trenutniP = *pod;
               }
               else {
                     prethodniP->sledeci = trenutniP->sledeci;
                     free(trenutniP);
                     trenutniP = prethodniP->sledeci;
               }
```

```
            }

            else {
                  prethodniP = trenutniP;
                  trenutniP = trenutniP->sledeci;
            }
      }
}

int main() {
      podaci *pod;
      lokacija *lokacije;
      char string[10];
      int n, ID;
      char *datum, *vreme1, *vreme2;

      lokacije = UcitajLokacije(lokacije);
      pod = UcitajPodatke(pod, lokacije);

      scanf("%s", string);
      getchar();
      datum = (char*)malloc(strlen(string) + 1);
      strcpy(datum, string);

      scanf("%s", string);
      getchar();
      vreme1 = (char*)malloc(strlen(string) + 1);
      strcpy(vreme1, string);

      scanf("%s", string);
      getchar();
      vreme2 = (char*)malloc(strlen(string) + 1);
      strcpy(vreme2, string);

      ProsekTemp(datum, vreme1, vreme2, pod);

      printf("Unesite ID lokacije koju zelite da obrisete: ");
      scanf("%d", &ID);
      IzbrisiLokaciju(ID, &pod, &lokacije);

      return 0;
}
```