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

Jun, 2019

Korisnicima je na raspolaganju lista/niz servera sa određenim servisima. Za svaki server se zna njegovo ime (niz karakera), ID (ceo broj), koliko trenutno ima servisa na raspolagnju (ceo broj) i koji su servisi u pitanju – ime (niz karakera), ID (ceo broj) i koliko maksimalno (ceo broj) u jednom tenutku korisnika može biti prijavljeno na taj servis. Podaci o serverima i dostupnim servisima se učitavaju iz datoteke. Korisnik se svojim korisničkim imenom (niz karaktera) prijavljuje na određeni server zadajući ID servera, na koliko servisa se prijavljuje i za svaki servis njegov ID. Ukoliko na željenom servisu trenutno nema mesta, korisnik se stavlja u red za čekanje. Korisnik se može odjaviti sa određenog servisa sljanjem ID servera i ID servisa na koji je prijavljen. Ako se korisnik odjavi sa svih servisa jednog servera, on se automatski odjavljuje i sa čitavog servera i uklanja iz spiska korisnika. Prijave i odjave korisnika se dešavaju „beskonačno", pri čemu pre svake prijave novog korisnika proverava se da li se može realizovati neki zahtev korisnika na čekanju, pa tek potom zahtev novog korisnika.
(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 **UcitajServise** koja iz datoteke *Server.txt* učitava podatke o servisima odgovarajućeg servera i formira listu/niz servisa.
(3 poena, obavezno)

c) Napisati funkciju **UcitajServere** koja iz datoteke *Server.txt* učitava podatke o serverima i formira listu/niz servera, sa odgovarajućim servisima.
(4 poena)

d) Napisati funkciju **NadjiServis** koja za dati ID servera i ID servisa na njemu vraća pokazivač na određeni servis. Ukoliko server ili servis ne postoji vratiti prazan pokazivač.
(3 poena, obavezno)

e) Napisati funkciju **Prijava** koja za korisnika viši prijavu na željeni servis ukoliko je moguće, odnosno smešta ga u red za čekanje ako nije moguće.
(5 poena)

f) Napisati funkciju **OdjavaServisa** koja korisnika odjavljuje sa željenog servisa na nekom serveru
(4 poena)

g) Napisati funkciju **OdjavaServera** koja korisnika odjavljuje sa servera, odnosno briše iz liste/niza korisnika
(4 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

Jun, 2019

1. **(Obavezno!)** Svi obavezni delovi iz zadatka o serverima.
2. Napisati funkciju koja u datu pointersku listu realnih brojeva, iza svakog elementa ubacuje element koji je dvostruko veći.
3. Data je struktura

   struct element{

   int d;

   char *s;

   struct element *par;

   };

   I nizovi struct element *a, *b. Svaki element niza a, može da pokazuje na jedan element niza b i svaki element niza b može da pokazuje na neki elemt niza a, različit on elementa niza a koji na njega pokazuje. Napisati funkciju koja će za datu poziciju elemeta u nizu a ispisati podatke elementa niza b na koji on okazuje.
4. Za strukturu i nizove a i b iz zadatka 3 napisati funkciju koja će polazeći od nego elementa niza a ispisati njegove podatke, zatim podatke elementa niza b na koji on pokazuje, pa potom podatke elementa na koji on pokazuje i tako dalje sve dok se ne stigne do elementa koji ne pokazuje nigde. Pretpostaviti da elementi ne formiraju ciklus.
5. Šta je rezultat sledećeg koda

```
#include <stdio.h>
main()
{
      int x;
      scanf("%d",&x);
      if(x > 20)
            if(25)
                  x &= 024 + 3;
      else
          x = 6;
      printf("%d\n",x);
 }
```

   Ako se kao vrednost promenljive `x` unese:

   a. 20 b. 25 c. 30

Zadatak – rešenje studenta

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

typedef struct servis{
   char *ime;
   int ID;
   int brKorisnika;
   int maxKorisnika;
   struct servis *sledeci;
}Servis;

typedef struct server{
   char *ime;
   int ID;
   int brServisa;
   Servis *servisi;
   struct server *sledeci;
}Server;

typedef struct korisnik{
   char *ime;
   int IDservera;
   int brServisa;
   int *IDservisa;
   int brNaCekanju;
   int brOdjavljenih;
   Servis *servisi;
   struct korisnik *sledeci;
}Korisnik;

Servis *UcitajServise(FILE *ulaz, int n){
   Servis *glava= NULL;
   Servis *pom;
   char ime[20];
   int i;

   for(i=0;i<n;i++){
          Servis *novi= (Servis*)malloc(sizeof(Servis));
          novi->sledeci= NULL;
          novi->brKorisnika= 0;

          fscanf(ulaz, "%s %d %d", ime, &novi->ID, &novi->maxKorisnika);
          fgetc(ulaz);

          novi->ime= (char*)malloc(strlen(ime)+1);
          strcpy(novi->ime,ime);

          if(glava==NULL){
                 glava= novi;
                 pom= glava;
          }
          else{
                 pom->sledeci= novi;
                 pom= pom->sledeci;
          }
   }
   return glava;
}
```

```
Server *UcitajServere(){
   FILE *ulaz= fopen("Server.txt","r");
   Server* glava= NULL;
   Server* pom;
   char ime[20];

   if(ulaz==NULL)
         exit(0);

   while(fscanf(ulaz,"%s",ime)==1){
         fgetc(ulaz);
         Server* novi=(Server*)malloc(sizeof(Server));
         novi->sledeci= NULL;

         novi->ime=(char*)malloc(strlen(ime)+1);
         strcpy(novi->ime,ime);

         fscanf(ulaz, "%d", &novi->ID);
         fgetc(ulaz);

         fscanf(ulaz, "%d", &novi->brServisa);
         fgetc(ulaz);

         novi->servisi= UcitajServise(ulaz,novi->brServisa);


         if(glava==NULL){
               glava= novi;
               pom= glava;
         }
         else{
               pom->sledeci= novi;
               pom= pom->sledeci;
         }
   }
   fclose(ulaz);
   return glava;
}

Servis *NadjiServis(Server **serveri, int IDservera, int IDservisa){
   Servis *trazeni= NULL;
   Server *pom= *serveri;

   while(pom && pom->ID!=IDservera)
         pom= pom->sledeci;

   if(pom!=NULL){
         trazeni= pom->servisi;
         while(trazeni && trazeni->ID!=IDservisa)
               trazeni= trazeni->sledeci;
   }

   return trazeni;
}

void OdjavaServera(Korisnik**, Server**, char*);
```

```
void Prijava(Korisnik **korisnik, Korisnik **korisnici, Korisnik **red, Server **serveri){
   Servis *pom, *servisiKorisnika;
   Korisnik *pomocni= *red;
   Korisnik *pomKorisnik= *korisnici;
   int i, j, n, ind=0;


   while(pomKorisnik && strcmp(pomKorisnik->ime, (*korisnik)->ime)!=0)
          pomKorisnik= pomKorisnik->sledeci;

   if(pomKorisnik!=NULL){
          n= (*korisnik)->brNaCekanju;
          servisiKorisnika= pomKorisnik->servisi;
          while(servisiKorisnika && servisiKorisnika->sledeci != NULL)
                servisiKorisnika= servisiKorisnika->sledeci;
          for(i=0;i<n;i++){
                pom = NadjiServis(serveri, (*korisnik)->IDservera,
                                  (*korisnik)->IDservisa[i]);
                if(pom!=NULL){
                      if(pom->brKorisnika < pom->maxKorisnika){
                            pom->brKorisnika++;

                            Servis *novi= (Servis*)malloc(sizeof(Servis));
                            novi->sledeci= NULL;
                            novi->brKorisnika= pom->brKorisnika;
                     // ovo je samo stanje nakon prijave ovog korisnika, ne i onih posle njega
                            novi->ID= pom->ID;
                            novi->maxKorisnika= pom->maxKorisnika;

                            novi->ime= (char*)malloc(strlen(pom->ime)+1);
                            strcpy(novi->ime,pom->ime);

                            if(pomKorisnik->servisi==NULL){
                                  pomKorisnik->servisi= novi;
                                  servisiKorisnika= pomKorisnik->servisi;
                            }
                            else{
                                  servisiKorisnika->sledeci= novi;
                                  servisiKorisnika= servisiKorisnika->sledeci;
                            }

                            pomKorisnik->brNaCekanju--;
                            (*korisnik)->brNaCekanju= pomKorisnik->brNaCekanju;
                            n-=1;
                            for(j=i;j<n;j++){
                                  pomKorisnik->IDservisa[j]= pomKorisnik->IDservisa[j+1];
                                  (*korisnik)->IDservisa[j]= pomKorisnik->IDservisa[j];
                            }
                            i-=1;
                            if(n==0){
                                  OdjavaServera(red, serveri, (*korisnik)->ime);
                                  free(pomKorisnik->IDservisa);
                            }
                      }
                      else
                            ind= 1;
                }
          }
          if(ind){
                while(pomocni && strcmp(pomocni->ime,(*korisnik)->ime)!=0)
                      pomocni= pomocni->sledeci;

                if(pomocni==NULL){
                      Korisnik *novi=(Korisnik*)malloc(sizeof(Korisnik));
                      novi->sledeci= NULL;
```

```
                        novi->brNaCekanju= (*korisnik)->brNaCekanju;
                        novi->brOdjavljenih= (*korisnik)->brOdjavljenih;

                        novi->ime= (char*)malloc(strlen((*korisnik)->ime)+1);
                        strcpy(novi->ime,(*korisnik)->ime);

                        novi->IDservera = (*korisnik)->IDservera;
                        novi->brServisa = (*korisnik)->brServisa;
                        novi->servisi= NULL;
                        novi->IDservisa=(int*)malloc(sizeof(int)*novi->brServisa);
                        for(i=0;i<novi->brNaCekanju;i++)
                              novi->IDservisa[i]= (*korisnik)->IDservisa[i];

                        if(*red==NULL)
                              *red= novi;
                        else{
                              pomocni= *red;
                              while(pomocni->sledeci != NULL)
                                    pomocni= pomocni->sledeci;
                              pomocni->sledeci= novi;
                        }
                  }
            }
      }

}

void OdjavaServisa(Korisnik **korisnici, Server **serveri, char *ime, int IDservera,
                   int IDservisa){
      Korisnik *pomKorisnik= *korisnici;
      Servis *servisiKorisnika;
      Servis *pomServis, *prethodniServis= NULL;

      while(pomKorisnik && strcmp(pomKorisnik->ime,ime)!=0)
            pomKorisnik= pomKorisnik->sledeci;

      if(pomKorisnik!=NULL){
            pomServis= NadjiServis(serveri, IDservera, IDservisa);
            if(pomServis!=NULL && pomServis->brKorisnika>0){
                  pomServis->brKorisnika--;
                  servisiKorisnika= pomKorisnik->servisi;
                  while(servisiKorisnika && servisiKorisnika->ID!=pomServis->ID){
                        prethodniServis= servisiKorisnika;
                        servisiKorisnika= servisiKorisnika->sledeci;
                  }
                  if(prethodniServis!=NULL)
                        prethodniServis->sledeci= servisiKorisnika->sledeci;
                  else
                        pomKorisnik->servisi= servisiKorisnika->sledeci;
                  free(servisiKorisnika);
            }
            pomKorisnik->brOdjavljenih++;
      }

      if(pomKorisnik->brOdjavljenih==pomKorisnik->brServisa)
            OdjavaServera(korisnici, serveri, ime);
}
```

```
void OdjavaServera(Korisnik **korisnici, Server **serveri, char *ime){
   Korisnik *prethodniKorisnik= NULL, *pomKorisnik= *korisnici;
   Servis *pomServis;
   Servis *servisiKorisnika;

   while(pomKorisnik && strcmp(pomKorisnik->ime,ime)!=0){
         prethodniKorisnik= pomKorisnik;
         pomKorisnik= pomKorisnik->sledeci;
   }

   if(pomKorisnik!=NULL){
         servisiKorisnika= pomKorisnik->servisi;
         while(servisiKorisnika!=NULL){
               pomServis= NadjiServis(serveri, pomKorisnik->IDservera,
                                      servisiKorisnika->ID);
               if(pomServis!=NULL && pomServis->brKorisnika>0)
                     pomServis->brKorisnika--;
               pomServis= servisiKorisnika->sledeci;
               free(servisiKorisnika);
               servisiKorisnika= pomServis;
         }
         if(prethodniKorisnik!=NULL)
               prethodniKorisnik->sledeci= pomKorisnik->sledeci;
         else
               *korisnici= pomKorisnik->sledeci;
         free(pomKorisnik);
   }
}

int main(){
   Server *serveri= UcitajServere();
   Korisnik *korisnici= NULL;
   Korisnik *pom, *pom2;
   Korisnik *red= NULL;
   int ind, n, i;
   int IDservera, IDservisa;
   char ime[20];

   printf("Unesite odgovarajuci broj: (0 - odjava korisnika, 1 - prijava korisnika,
            KRAJ - kraj unosa)\n");
   while(scanf("%d", &ind)==1){
               if(ind==1){
                           printf("Unesite novog korisnika: (redosled unosa: ime,
                                   id servera, broj servisa, id svih servisa)\n");
                           Korisnik *novi=(Korisnik*)malloc(sizeof(Korisnik));
                           novi->sledeci= NULL;
                           novi->brOdjavljenih= 0;

                           scanf("%s%d%d", ime, &novi->IDservera, &novi->brServisa);

                           novi->ime= (char*)malloc(strlen(ime)+1);
                           strcpy(novi->ime,ime);

                           novi->brNaCekanju= novi->brServisa;
                           n= novi->brServisa;
                           novi->IDservisa=(int*)malloc(sizeof(int)*novi->brServisa);
                           for(i=0;i<n;i++){
                                 scanf("%d",&novi->IDservisa[i]);
                           }

                           novi->servisi= NULL;


                           if(korisnici==NULL){
                                 korisnici= novi;
```

```
                                pom= korisnici;
                        }
                        else{
                                pom->sledeci= novi;
                                pom= pom->sledeci;
                        }

                        pom2= red;
                        while(pom2){
                                Prijava(&pom2, &korisnici, &red, &serveri);
                                pom2=pom2->sledeci;
                        }
                        Prijava(&pom, &korisnici, &red, &serveri);
                }
                else
                        if(ind==0){
                                printf("Unesite odgovarajuci broj: (0 - odjava sa servisa,
                                        1 - odjava sa servera)\n");
                                scanf("%d", &ind);
                                if(ind==1){
                                        printf("Ime korisnika koga zelite da odjavite
                                                sa servera:\n");
                                        scanf("%s", ime);
                                        OdjavaServera(&korisnici, &serveri, ime);
                                }
                                else
                                        if(ind==0){
                                                printf("Ime korisnika, ID servera,
                                                        ID servisa:\n");
                                                scanf("%s%d%d", ime, &IDservera, &IDservisa);
                                                OdjavaServisa(&korisnici, &serveri, ime,
                                                                IDservera, IDservisa);
                                        }

                                pom2= red;
                                while(pom2){
                                        Prijava(&pom2, &korisnici, &red, &serveri);
                                        pom2=pom2->sledeci;
                                }
                        }
                printf("Unesite odgovarajuci broj: (0 - odjava korisnika,
                        1 - prijava korisnika, KRAJ - kraj unosa)\n");
        }

        //PROVERA STANJA SERVERA I KORISNIKA NAKON ZAUSTAVLJANJA UNOSA
        pom= korisnici;
        Servis* pom3;
        while(pom){
                printf("\nIme korisnika: %s \nID servera: %d\nBroj servisa na cekanju: %d\n",pom-
>ime, pom->IDservera, pom->brNaCekanju);
                printf("Sevisi na cekanju: ");
                for(i=0;i<pom->brNaCekanju;i++)
                        printf("%d ",pom->IDservisa[i]);
                printf("\n");
                pom3=pom->servisi;
                printf("Trenutno prijavljen na sledece servise: \n");
                while(pom3){
                        printf("Ime:%s - ID:%d\n",pom3->ime, pom3->ID);
                        pom3=pom3->sledeci;
                }
                printf("\n");
                pom= pom->sledeci;
        }

        printf("-------------------------------------------------------------------------");
```

```
    Server *pom1= serveri;
    while(pom1){

           printf("\nIme servera: %s \nID servera: %d\n",pom1->ime, pom1->ID);
           pom3=pom1->servisi;
           while(pom3){
                  printf("Ime:%s - ID:%d - Trenutno korisnika:%d –
                         Dozvoljeno korisnika: %d\n",
                         pom3->ime, pom3->ID, pom3->brKorisnika, pom3->maxKorisnika);
                  pom3=pom3->sledeci;
           }
           pom1= pom1->sledeci;
    }

    return 0;
}
```