Strukture podataka i algoritmi 1

Jun, 2014

Na mreži *IMIGame* namenjenoj za igranje različitih tipova igara, svaki korisnik u jednom trenutku može da igra jednu igru. Svaka igra može da postoji na više servera i na svakom od servera može da je igra različit broj igrača. Igra može da donese različit broj poena na različitim serverima. Kada jednu igru igra veći broj igrača od maksimalnog dozvoljenog, ta igra se gasi zbog preopterećenja i igrači se prebacuju na druge najmanje opterećene servere sa tom igrom, ako takvi postoje. Ukoliko ne postoje drugi serveri sa istom igrom, igrači se prebacuju na druge igre istog tipa. Ako ni to nije moguće igrači završavaju igru. Igranje jedne igre donosi odgeđen broj poena. Ukoliko igrač pređe na drugi server i/ili drugu igru njegovom broju poena se dodaju poeni koje donosi ta igra.

Napisati program u kome se učitavaju podaci o igrama, podaci o igračima i svi igrači se rasporedjuju po igrama. Potom se gase sve igre koje su preopterećene i igrači preraspoređuju na druge servere i/ili igre. Na kraju se ispisuje rang lista igrača koji su završili igru, tj. igrača koji nisu zakačeni ni na jednu igru.
(3 poena)

Za rešavanje problema napisati sledeće funkcije:

a) Napisati funkciju **UcitajIgre** koja iz datoteke učitava podatke o igrama i to ukupan broj igara i za svaku igru ime, tip, ime servera (tekstualni podaci), maksimalan broj igrača i broj poena koje igra donosi (celobrojni podaci).
(3 poena)

b) Napisati funkciju **UbaciIgrača** koja za zadatu igru i njen tip odredjuje na koji server i/ili igru igrač može da se zakači. Svaki igrač najpre pokušava da se zakači na navedenu igru na najmanje opterećenom serveru, ukoliko ta igra ne postoji, igrač se kači na igru istog tipa koja je najmanje opterećena. Ako ne postoji ni navedena igra ni navedeni tip igre, igrač ostaje nezakačen, tj. igrač završava igru.
(5 poena)

c) Napisati funkciju **UcitajIgrace** koja iz datoteke učitava podatke o igračima i to ukupan broj igrača i za svakog igrača njegovo ime, ime igre koju želi da igra i tip te igre i formira niz igrača, pri čemu je svaki igrač zakačen na neku igru, ukoliko je to moguće, tj. ima pokazivač na odgovarajući element u nizu igara.
(3 poena)

d) Napisati funkciju **UgasiIgru** koja izabacuje iz niza igara zadatu igru na zadatom serveru. Igrače koji su igrali navedenu igru koja se gasi preraspodeliti na druge servere i/ili igre. Igračima koji se rasporedjuju na nove servere i/ili igre se poeni koje donosi nova igra dodaju na već osvojene poene. Igrač može pri ovome i da završi igru, ukoliko ne postoji igra na koju može da se zakači.
(6 poena)

e) Napisati funkciju **Oprerecenje** koja će gasiti sve preoterećene igre, tj. igre na koje je zakačen veći broj igrača od dozvoljenog makismuma, i igrače prebacivati na nove servere i/ili igre ako je to moguće, sve dok postoje preopterećene igre.
(6 poena)

f) Napisati funkciju **Poeni**, koja formira rang listu igrača koji su završili igru prema osvojenim poenima.
(4 poena)

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

typedef struct igra
          {
          char ime[50];
          char tip[50];
          char server[50];
          int max;
          int poeni;
          int br_igraca;
          } igra;

typedef struct igrac
          {
          char ime[50];
          char ime_igre[50];
          char tip_igre[50];
          struct igra* igra;
          int poeni;
          int ind;
          } igrac;

void UcitajIgre(igra* a[],int* n,char dat[]);
igra* UbaciIgraca(char ime[],char tip[],igra a[],int n);
void UcitajIgrace(igrac* a[],int* n,char dat[],igra b[],int m);
void UgasiIgru(char ime[],char server[],igra a[],int* n,igrac b[],int m);
void Opterecenje(igra a[],int* n,igrac b[],int m);
igrac* Poeni(igrac a[],int n,int* k);

main(int argc,char* argv[])
{
     igra* a;
     igrac* b;
     igrac* rang;
     int n=0,m=0,k=0,i;

     if (argc!=3)
     {
     printf("Greska! Upotreba: %s dat_igre dat_igraci\n",argv[0]);
     return 1;
     }

     UcitajIgre(&a,&n,argv[1]);
     UcitajIgrace(&b,&m,argv[2],a,n);
     Opterecenje(a,&n,b,m);
     rang=Poeni(b,m,&k);

     for (i=0;i<k;i++)
          printf("%s%d\n\n",rang[i].ime,rang[i].poeni);
}

void UcitajIgre(igra* a[],int* n,char dat[])
{
     int i;
     FILE* f=fopen(dat,"r");
```

```
      if (f==NULL)
      {
            printf("Greska pri otvaranju datoteke %s!\n",dat);
            exit (1);
      }
      fscanf(f,"%d",n);
      fgetc(f); /* ucitava \n posle podatka tipa int */
      *a=(igra*)malloc((*n)*sizeof(igra));
      for (i=0;i<(*n);i++)
      {
            fgets((*a)[i].ime,50,f);
            fgets((*a)[i].tip,50,f);
            fgets((*a)[i].server,50,f);
            fscanf(f,"%d%d",&(*a)[i].max,&(*a)[i].poeni);
            fgetc(f);
            (*a)[i].br_igraca=0;
      }
      fclose(f);
}

igra* UbaciIgraca(char ime[],char tip[],igra a[],int n)
{
      int tren,indeks,i,ind=0;

      for (i=0;i<n;i++)
            if (strcmp(ime,a[i].ime)==0)
            {
                  ind=1;
                  break;
            }
      if (ind==1)
      {
            tren=a[i].max-a[i].br_igraca;
            indeks=i;
            i++;
            for (;i<n;i++)
                  if ((strcmp(ime,a[i].ime)==0)&&
                        (tren<(a[i].max-a[i].br_igraca)))
                  {
                        tren=a[i].max-a[i].br_igraca;
                        indeks=i;
                  }
      a[indeks].br_igraca++;
      return(&a[indeks]);
      }
      ind=0;
      for (i=0;i<n;i++)
            if (strcmp(tip,a[i].tip)==0)
            {
                  ind=1;
                  break;
            }
      if (ind==1)
      {
            tren=a[i].max-a[i].br_igraca;
            indeks=i;
            i++;
```

```
            for (;i<n;i++)
                  if ((strcmp(tip,a[i].tip)==0)&&
                        (tren<(a[i].max-a[i].br_igraca)))
                  {
                        tren=a[i].max-a[i].br_igraca;
                        indeks=i;
                  }
            a[indeks].br_igraca++;
            return(&a[indeks]);
            }
      return(NULL);
}

void UcitajIgrace(igrac* a[],int* n,char dat[],igra b[],int m)
{
      int i;
      FILE* f=fopen(dat,"r");
      if (f==0)
      {
            printf("Greska pri otvaranju datoteke %s!\n",dat);
            exit (1);
      }
      fscanf(f,"%d",n);
      fgetc(f);
      *a=(igrac*)malloc((*n)*sizeof(igrac));
      for (i=0;i<(*n);i++)
      {
            fgets((*a)[i].ime,50,f);
            fgets((*a)[i].ime_igre,50,f);
            fgets((*a)[i].tip_igre,50,f);
            (*a)[i].igra=UbaciIgraca((*a)[i].ime_igre,(*a)[i].tip_igre,b,m);
            if ((*a)[i].igra!=NULL) (*a)[i].poeni=(*a)[i].igra->poeni;
            else (*a)[i].poeni=0;
            (*a)[i].ind=0;
      }
      fclose(f);
}

void UgasiIgru(char ime[],char server[],igra a[],int* n,igrac b[],int m)
{
      int i,indeks;
      char tip[50];

      for (i=0;i<(*n);i++)
            if ((strcmp(ime,a[i].ime)==0)&&(strcmp(server,a[i].server)==0))
            {
                  indeks=i;
                  break;
            }
      strcpy(tip,a[indeks].tip);
      for (i=0;i<m;i++)
            if (b[i].igra==&a[indeks])
                  b[i].ind=1;
      a[indeks]=a[(*n)-1];
      for (i=0;i<m;i++)
            if (b[i].igra==&a[(*n)-1])
                  b[i].igra=&a[indeks];
```

```
	(*n)--;
	for (i=0;i<m;i++)
		if (b[i].ind==1)
		{
			b[i].igra=UbaciIgraca(ime,tip,a,*n);
			if (b[i].igra!=NULL) b[i].poeni+=b[i].igra->poeni;
			b[i].ind=0;
		}
}

void Opterecenje(igra a[],int* n,igrac b[],int m)
{
	int i,ind=1,k;

	while(ind)
	{
		ind=0;
		for (i=0;i<(*n);i++)
			if ((a[i].max-a[i].br_igraca)<0) ind=1;
		for (i=0;i<(*n);i++)
			if ((a[i].max-a[i].br_igraca)<0)
				UgasiIgru(a[i].ime,a[i].server,a,n,b,m);
	}
}

igrac* Poeni(igrac a[],int n,int* k)
{
	int i,j=0;
	igrac* b;
	igrac t;

	for (i=0;i<n;i++)
		if (a[i].igra==NULL) (*k)++;
	b=(igrac*)malloc((*k)*sizeof(igrac));
	for (i=0;i<n;i++)
		if (a[i].igra==NULL) b[j++]=a[i];

	for (i=0;i<(*k)-1;i++)
		for (j=i+1;j<(*k);j++)
			if (b[i].poeni<b[j].poeni)
			{
				t=b[i];
				b[i]=b[j];
				b[j]=t;
			}
	return(b);
}
```