Programiranje 2 – ispit, 8. 2. 2024.

Rješenja zadataka

Zadatak 1 (2+20 bodova) Karta pozicija je reprezentirana dvodimenzionalnim poljem dimenzija $n \times n$. Svaka pozicija na koju možemo stati je označena znakom '.', dok su pozicije na koje ne možemo stati (preko kojih ne možemo prijeći) označene znakom '*'. Želimo odrediti najmanji mogući broj pomaka od zadane pozicije `start` do ciljne pozicije `kraj` tako da poštujemo navedena ograničenja i posjećujemo svaku poziciju maksimalno jednom. Po karti se možemo kretati za jednu poziciju sjeverno (npr. pomak: $(1,0) \to (0,0)$), za jednu poziciju južno (npr. pomak $(1,0) \to (2,0)$), za jednu poziciju zapadno (npr. pomak $(0,1) \to (0,0)$) i za jednu poziciju istočno (npr. pomak $(0,0) \to (0,1)$). Gornja lijeva točka mape ima koordinate $(0,0)$, a donja desna (n,n).

a) Konceptualno objasnite kako biste riješili navedeni problem. Koju tehniku programiranja biste koristili? Razložite objašnjenje u cjeline koje možete implementirati.

b) Napišite funkciju **brojPomaka**, koja računa najmanji broj pomaka koje trebamo napraviti da bismo došli od početne pozicije `start` do ciljne pozicije `kraj`. Obavezno napišite naredbu poziva funkcije **brojPomaka**.

Napomena: dozvoljeno je korištenje zaglavlja `limits.h`, te jednog dodatnog jednodimenzionalnog globalnog polja dimenzije n^2.

Primjer: karta dimenzija 4×4

```
  0 1 2 3
0 . * * .
1 . . * .
2 . . . .
3 . * . .
```

Najkraći put između $(0,0)$ i $(3,2)$ je duljine 5, npr.: $(0,0) \to (1,0) \to (1,1) \to (2,1) \to (2,2) \to (3,2)$.

Rješenje:

```
#include <stdio.h>
#include <limits.h>

typedef struct _poz{
  int x, y;
}pozicija;

char ploca[4][4] = {{'.','.','.','.'},{'.','*','*','.'},{'*','.','.','.'},{'.','.','.','*'}};
pozicija niz[16];


int brojKoraka(char p[][4], int d, pozicija n[], pozicija start, pozicija kraj){

   if(start.x == kraj.x && start.y == kraj.y) return d;

   int imaPoteza = 0;

   pozicija p1,p2,p3,p4;

   p1.x = start.x -1;
   p1.y = start.y;

   p2.x = start.x + 1;
   p2.y = start.y;

   p3.x = start.x;
   p3.y = start.y +1;

   p4.x = start.x;
   p4.y = start.y - 1;
```

```
int b1 = INT_MAX, b2 = INT_MAX, b3 = INT_MAX, b4 = INT_MAX;

if(p1.x>=0 && p1.y>=0 && p1.x<=3 && p1.y<=3){
     int pronadjen = 0;

     if(p[p1.x][p1.y] == '*') pronadjen = 1;

   if(pronadjen == 0){
     for(int i=0;i<d;i++)
         if(n[i].x == p1.x && n[i].y == p1.y){
             pronadjen = 1;
             break;
         }
   }
    if(pronadjen == 0){
     imaPoteza = 1;
     n[d] = start;
     b1 = brojKoraka(p,d+1,n,p1,kraj);
    }
}

if(p2.x>=0 && p2.y>=0 & p2.x<=3 && p2.y<=3){
     int pronadjen = 0;

     if(p[p2.x][p2.y] == '*') pronadjen = 1;

     if(pronadjen == 0){
     for(int i=0;i<d;i++)
         if(n[i].x == p2.x && n[i].y == p2.y){
             pronadjen = 1;
             break;
         }
     }

     if(pronadjen == 0){
     imaPoteza = 1;
     n[d] = start;
     b2 = brojKoraka(p,d+1,n,p2,kraj);
     }
}

if(p3.x>=0 && p3.y>=0 && p3.x<=3 && p3.y<=3){
     int pronadjen = 0;

     if(p[p3.x][p3.y] == '*') pronadjen = 1;


     if(pronadjen == 0){
     for(int i=0;i<d;i++)
          if(n[i].x == p3.x && n[i].y == p3.y){
             pronadjen = 1;
             break;
         }
     }

    if(pronadjen == 0){
     imaPoteza = 1;
     n[d] = start;
     b3 = brojKoraka(p,d+1,n,p3,kraj);
    }
}

if(p4.x>=0 && p4.y>=0 && p4.x<=3 && p4.y<=3){
     int pronadjen = 0;
```

```
        if(p[p4.x][p4.y] == '*') pronadjen = 1;

        if(pronadjen == 0){
        for(int i=0;i<d;i++)
             if(n[i].x == p4.x && n[i].y == p4.y){
                pronadjen = 1;
                break;
            }
        }
       if(pronadjen == 0){
        imaPoteza = 1;
        n[d] = start;
        b1 = brojKoraka(p,d+1,n,p4,kraj);
       }
   }

   if(imaPoteza == 0) return INT_MAX;

   int min;

   min = (b1>=b2)? b2:b1;
   min = (min>=b3) ? b3: min;
   min = (min>=b4) ? b4: min;

   return min;

}

int main(void){

pozicija start = {0,1}, kraj = {2,1};

printf("Start: (%d, %d)\n", start.x, start.y);
printf("Kraj: (%d, %d)\n", kraj.x, kraj.y);

printf("Najkraci put: %d",brojKoraka(ploca,0,niz,start, kraj));

return 0;

}
```

Programiranje 2 – ispit, 8. 2. 2024.

Rješenja zadataka

Zadatak 2 (2+20 bodova) Napišite funkciju `char* sort(char* s, int* br)` koja prima string te vraća podstring najveće duljine koji je sortiran. Takav podstring mora biti minimalno duljine 2. Za string kažemo da je sortiran ako su njegovi znakovi uzlazno sortirani gdje ne razlikujemo velika i mala slova (npr. string `"abCcf"` je sortiran, dok `"abCcA"` nije). Ukoliko postoji više podnizova koji zadovoljavaju navedena svojstva, funkcija vraća onaj podstring koji je najmanji po leksikografskom uređaju. Ako polazni string ne sadrži sortirani podstring funkcija vraća prazan string. Preko varijabilnog argumenta funkcija vraća broj podstringova duljine veće ili jednake 2 koji su sortirani.
Napomena. Možete definirati dodatne (pomoćne) funkcije. Dopušteno je korištenje jednog niza za spremanje najvećeg podstringa (taj niz morate obavezno dinamički alocirati). Smijete koristiti funkcije iz `ctype.h`, `string.h` i `stdlib.h`. Možete pretpostaviti da funkcija `sort` prima string duljine barem 2.

Rješenje:

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

int check_sort(char* s, int i, int j){
    int sort=1;
    int k;
    for(k=i ; k<j ; k++){
        if(tolower(s[k+1])-tolower(s[k])<0){
            sort=0;
            break;
        }
    }
    return sort;
}

char* sort(char* s, int* br){
    *br=0;
    int i, j, k;
    int max=0;
    int n=strlen(s);
    int smax_i;
    for(i=0 ; i<=n-2 ; i++){
        for(j=i+1 ; j<n ; j++){
            if(check_sort(s, i, j)){
                (*br)++;
                if(j-i+1>max){
                    max=j-i+1;
                    smax_i=i;
                }
                else if(j-i+1==max){
                    int par=0;
                    for(k=0 ; k<max ; k++){
                        if(s[i+k]<s[smax_i+k]){
                            par=1;
                            break;
                        }
                    }
                    if(par==1){
                        smax_i=i;
                    }
                }
            }
            else {
                break;
            }
        }
    }
```

```
    char* smax=(char*)malloc((max+1)*sizeof(char));
    if(max>0){
        for(k=0 ; k<max ; k++){
            smax[k]=s[smax_i+k];
        }
    }
    smax[max]='\0';
    return smax;
}

int main(){
    char s[11]={'a', 'b', 'B', 'a', 'C', 'e', 'f', 'e', '\0'};
    int br;
    printf("%s\n", sort(s, &br));
    printf("%d\n", br);
    return 0;
}
```

Programiranje 2 – ispit, 8. 2. 2024.

Rješenja zadataka

Zadatak 3 (15 bodova) Pretpostavimo da imamo učitan rječnik brojeva `char** rjecnik`, koji sadrži n stringova proizvoljne veličine. Napišite program koji kao argument komandne linije prima cijeli broj koji ima vrijednost 1 ukoliko vršimo uzlazno sortiranje, a -1 ukoliko vršimo silazno sortiranje rječnika. Program prvo ispisuje sve stringove iz rječnika koristeći razmak kao separator. Zatim, ovisno o vrijednosti ulaznog argumenta, program treba sortirati rječnik ulazno/silazno koristeći implementaciju Quick Sort algoritma iz datoteke zaglavlja `stdlib.h`. Pripazite da izbjegnete upozorenja prevodioca uzrokovana konverzijama tipova argumenata funkcija.

Napomena: smijete koristiti funkcije deklarirane u datotekama zaglavlja `string.h` i `stdlib.h`. U ovom programu ne treba alocirati/dealocirati memoriju.

Rješenje:

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

int usporediU(const void *a, const void *b){;
  return strcmp(*((char **)a), *((char **)b));
return 0;
}


int usporediS(const void *a, const void *b){;
  return -strcmp(*((char **)a), *((char **)b));
return 0;
}

int main(int argc, char** argv){

char **rjecnik = NULL;
char *s;
int numStr = 0;

int smjer = atoi(argv[1]);

s = (char*) malloc (50*sizeof(char));
scanf("%s",s);
while(strcmp(s,"kraj")){
    rjecnik = (char**) realloc(rjecnik,numStr++);
    rjecnik[numStr-1] = s;
    s = (char*) malloc (50*sizeof(char));
    scanf("%s",s);
}

free(s);

for(int i=0;i<numStr;i++)
    printf("%s ",rjecnik[i]);
printf("\n\n");


if(smjer == 1)
    qsort(rjecnik, numStr, sizeof(char*), usporediU);
else if(smjer == -1)
    qsort(rjecnik, numStr, sizeof(char*), usporediS);

printf("Ispis sort\n");
for(int i=0;i<numStr;i++)
    printf("%s ",rjecnik[i]);
printf("\n\n");
```

```
for(int i=0;i<numStr;i++)
    free(rjecnik[i]);

free(rjecnik);



return 0;

}
```

Programiranje 2 – ispit, 8. 2. 2024.

Rješenja zadataka

Zadatak 4 (2+9+12 bodova) Popis knjiga u knjižnici sprema se u **vezanu listu** tako da je lista sortirana uzlazno leksikografski prema naslovu knjige. Za svaku knjigu pamtimo naslov knjige (string duljine najviše 50), ime i prezime autora (string duljine najviše 30) te jedinstveni identifikacijski broj knjige (prirodni broj).

(a) Deklarirajte tip podatka `knjiga` koji pamti navedene podatke o knjizi. Dodatno, varijabla `book` tog tipa se treba moći definirati sa `knjiga book;`.

(b) Napišite funkciju

```
knjiga* dodaj_knjigu(knjiga* knjiznica, char* naslov, char* autor, int id)
```

koja u vezanu listu `knjiznica` dodaje knjigu s podacima prosljeđenim u argumentima funkcije tako da lista `knjiznica` ostane sortirana abecedno prema naslovu knjige. Ukoliko već postoji knjiga s naslovom `naslov`, novu knjigu treba dodati na **početak** bloka knjiga s naslovom `naslov`. Funkcija treba vratiti pokazivač na početak ažurirane liste `knjiznica`.

(c) Napišite funkciju

```
knjiga* izdvoji_autora(knjiga** knjiznica, char* autor)
```

koja stvara vezanu listu svih knjiga iz vezane liste `knjiznica` kojima je autor jednak stringu `autor` iz argumenta funkcije te sukladno tome mijenja vezanu listu `knjiznica`. Stvaranje tražene vezane liste treba provesti **bez alociranja dodatne memorije**, to jest preslagivanjem postojećih elemenata vezane liste `knjiznica`. Funkcija treba vratiti pokazivač na početak kreirane vezane liste koja također treba biti sortirana uzlazno leksikografski prema naslovu knjige. Pokazivač na početak ažurirane vezane liste `knjiznica` se šalje preko **varijabilnog argumenta** funkcije `izdvoji_autora`. Vezana lista `knjiznica` nakon poziva funkcije `izdvoji_autora` ne sadrži niti jednu knjigu kojoj je autor jednak `autor`.

Napomena. Pripazite na slučajeve kada je vezana lista `knjiznica` prazna. Nije dozvoljeno korištenje pomoćnih polja, niti dodatnih varijabli u strukturi `knjiga`.

Primjer. Vezana lista `knjiznica`:

$$\{\texttt{Harry Potter, J. K. Rowling, 73692}\} \to \{\texttt{Harry Potter, J. K. Rowling, 73645}\} \to \{\texttt{Mali princ, A. de Saint-Exupéry, 12345}\} \to \texttt{NULL}$$

nakon poziva funkcije `dodaj_knjigu(knjiznica,"Kronike iz Narnije","C.S. Lewis", 86490)` izgleda ovako:

$$\{\texttt{Harry Potter, J. K. Rowling, 73692}\} \to \{\texttt{Harry Potter, J. K. Rowling, 73645}\} \to \{\texttt{Kronike iz Narnije, C.S. Lewis, 86490}\} \to \{\texttt{Mali princ, A. de Saint-Exupéry, 12345}\} \to \texttt{NULL}.$$

Nakon poziva funkcije `izdvoji_autora(&knjiznica,"J. K. Rowling")`, lista izdvojenih knjiga izgleda ovako:

$$\{\texttt{Harry Potter, J. K. Rowling, 73692}\} \to \{\texttt{Harry Potter, J. K. Rowling, 73645}\} \to \texttt{NULL},$$

a lista `knjiznica` izgleda ovako:

$$\{\texttt{Kronike iz Narnije, C.S. Lewis, 86490}\} \to \{\texttt{Mali princ, A. de Saint-Exupéry, 12345}\} \to \texttt{NULL}.$$

Rješenje:

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

typedef struct _knjiga{
    char naslov[51];
    char autor[31];
    int id;
    struct _knjiga* next;
}knjiga;

knjiga* dodaj_knjigu(knjiga* knjiznica, char* naslov, char* autor, int id)
{
    knjiga *nova, *pom=knjiznica, *prosli;
```

```
    nova=(knjiga*)malloc(sizeof(knjiga));

    strcpy(nova->naslov,naslov);
    strcpy(nova->autor,autor);
    nova->id=id;

    while(pom!=NULL && strcmp(pom->naslov,naslov)<0)
    {
        prosli=pom;
        pom=pom->next;
    }

    if(pom==knjiznica) // dodajemo na pocetak liste - tu je ukljucen slucaj i kada je knjiznica prazna
    {
        nova->next=knjiznica;
        knjiznica=nova;

        return knjiznica;
    }

    else
    {
        prosli->next=nova;
        nova->next=pom; // ili NULL pointer ako smo sa while petljom došli do kraja liste ili knjiga koja ima >

        return knjiznica;
    }

}

knjiga* izdvoji_autora(knjiga** knjiznica, char* autor)
{
    knjiga* first=NULL, *last, *pom, *prosli;

    if(*knjiznica==NULL) return NULL;

    for(pom=*knjiznica;pom;pom=pom->next)
    {
        if(!strcmp(pom->autor,autor))
        {
            if(first==NULL)
                last=first=pom;
            else
                last=last->next=pom;

            if(pom==*knjiznica) *knjiznica=pom->next; //azuriranje pocetka liste knjiznica

            else prosli->next=pom->next; //preslagivanje strjelica u "sredini"
        }
        else prosli=pom;
    }

    if(first) last->next=NULL;

    return first;
}

//nije dio zadatka

int main()
{
    knjiga* knjiznica=NULL, *pom, *izdvojeni;

    knjiznica=dodaj_knjigu(knjiznica, "Kronike iz Narnije", "C.S. Lewis", 86490);
    knjiznica=dodaj_knjigu(knjiznica, "Harry Potter", "J. K. Rowling", 73645);
    knjiznica=dodaj_knjigu(knjiznica, "Harry Potter", "J. K. Rowling", 74625);
    knjiznica=dodaj_knjigu(knjiznica, "Alisa u zemlji cudesa", "L. Carroll", 98841);
```

```
    knjiznica=dodaj_knjigu(knjiznica, "Mali princ", "A. de Saint-Exupéry", 12345);
    knjiznica=dodaj_knjigu(knjiznica, "Zov kukavice", "J. K. Rowling", 84656);
    knjiznica=dodaj_knjigu(knjiznica, "Cetiristo dana", "L. Carroll", 84656);

    for(pom=knjiznica;pom;pom=pom->next)
        printf("%s: %d  ", pom->naslov, pom->id);

    printf("\n\n");

    izdvojeni=izdvoji_autora(&knjiznica,"J. K. Rowling");

    for(pom=izdvojeni;pom;pom=pom->next)
        printf("%s: %d  ", pom->naslov, pom->id);

    printf("\n\n");

    for(pom=knjiznica;pom;pom=pom->next)
        printf("%s: %d  ", pom->naslov, pom->id);

    printf("\n\n");

    return 0;
}
```

Programiranje 2 – ispit, 8. 2. 2024.

Rješenja zadataka

Zadatak 5 (2+20 bodova) Podaci o jednom učeniku se čuvaju u strukturi `ucenik` koja se sastoji od sljedećih podataka: ime učenika (string od maksimalno 50 znakova) i broj bodova (int).

(a) Definirajte strukturu `ucenik` koja pamti navedene podatke o jednom učeniku. Struktura mora biti definirana tako da bude moguća deklaracija oblika `ucenik u;` te smije sadržavati samo zadane podatke.

(b) Napišite funkciju `void azuriraj(FILE* in, char ime[], int bodovi)` koja kao argument prima pokazivač `in` na binarnu datoteku već otvorenu za čitanje i pisanje, ime učenika i broj bodova. Binarna datoteka se sastoji od struktura `ucenik` sortiranih silazno po broju bodova. Funkcija treba ažurirati binarnu datoteku na sljedeći način. Učeniku imena `ime` se broj bodova uveća za ulaznu varijablu `bodovi`. Nakon toga datoteka mora ostati sortirana silazno. Ako neki učenici imaju jednak broj bodova poredak nije bitan. Možete pretpostaviti da je ulazna varijabla `bodovi` > 0 i da se učenik imena `ime` sigurno nalazi u datoteci. Također, možete pretpostaviti da su sva imena različita.

Primjer:

Početna binarna datoteka `in`:
`{Marko, 12}, {Ana, 8}, {Marija, 8}, {Zdenko, 8}, {Mia, 7}`

Poziv: `azuriraj(in,"Zdenko",2)`

Završna binarna datoteka `in`:
`{Marko, 12}, {Zdenko, 10}, {Ana, 8}, {Marija, 8}, {Mia, 7}`

Poziv: `azuriraj(in,"Ana",1)`

Završna binarna datoteka `in`:
`{Marko, 12}, {Zdenko, 10}, {Ana, 9}, {Marija, 8}, {Mia, 7}`

Napomena: Nije dozvoljeno korištenje dodatnih nizova i datoteka. Dozvoljeno je korištenje zaglavlja `stdio.h`, `stdlib.h` i `string.h`.

Rješenje:

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

typedef struct
{
    char ime[51];
    int bodovi;
}ucenik;

void azuriraj(FILE* in, char ime[], int bodovi)
{
    ucenik temp, tempu;
    int poz, pozu;
    while(fread(&temp,sizeof(ucenik),1,in)==1)
    {
        poz = ftell(in);
        if(strcmp(temp.ime,ime)==0)
            temp.bodovi+=bodovi;

        if(strcmp(temp.ime,ime)) continue;
        pozu = poz-2*sizeof(ucenik);
        if(pozu>=0)
        {
            fseek(in,pozu,SEEK_SET);
            fread(&tempu,sizeof(ucenik),1,in);
            if(temp.bodovi > tempu.bodovi)
            {
                fseek(in,pozu,SEEK_SET);
                fwrite(&temp,sizeof(ucenik),1,in);
```

```
                fseek(in, poz-sizeof(ucenik),SEEK_SET);
                fwrite(&tempu,sizeof(ucenik),1,in);
                while(1)
                {
                    pozu = pozu-sizeof(ucenik);
                    if(pozu<0)
                        return;

                    fseek(in,pozu,SEEK_SET);
                    fread(&tempu,sizeof(ucenik),1,in);
                    if(tempu.bodovi>=temp.bodovi)
                        return;

                    fseek(in, pozu, SEEK_SET);
                    fwrite(&temp,sizeof(ucenik),1,in);
                    fseek(in,pozu+sizeof(ucenik),SEEK_SET);
                    fwrite(&tempu,sizeof(ucenik),1,in);
                }
                return;
            }
        }
        fseek(in,poz-sizeof(ucenik),SEEK_SET);
        fwrite(&temp,sizeof(ucenik),1,in);
        return;
    }
}
```