PROGRAMIRANJE (C) - Rješenja

10. srpnja 2002.

1. Napišite program koji učitava dvije matrice A i B dimenzija mxn ($max\{m,n\} \leq 20$) i provjerava da li je $A^TB = B^TA$. Ne smijete koristiti dodatna polja.
   Rj:

```
#include<stdio.h>

#define MAX 20

main(){
    int i,j,m,n, A[MAX][MAX], B[MAX][MAX];
    int uvjet=1;

    printf("Unesite m i n:");
    scanf("%d %d", &m,&n);
    while( ! (m>0 && m<=MAX)  ){
        printf("Greska, m mora biti veci od 0, a manji od %d \n", MAX);
        scanf("%d",&m);
        }
    while( ! (n>0 && n<=MAX)  ){
        printf("Greska, n mora biti veci od 0, a manji od %d \n", MAX);
        scanf("%d",&n);
    }

    for (i=0; i<m ; i++)
        for (j=0; j<n; j++){
            printf("A[%d][%d]=", i+1, j+1);
            scanf("%d", &A[i][j]);
        }

    for (i=0; i<m ; i++)
        for (j=0; j<n; j++){
            printf("B[%d][%d]=", i+1, j+1);
            scanf("%d", &B[i][j]);
        }

    for (i=0; i<n && uvjet; i++)
        for (j=0; j<n && uvjet; j++){
            int c1 = 0; int c2=0;
            int k;
            for (k=0;k<m;k++){
                c1 += A[k][i] * B[k][j];
                c2 += B[k][i] * A[k][j];
            }
            if (c1!=c2) uvjet=0;
        }

    if (!uvjet){
        printf("Matrice A i B ne zadovoljavaju uvjet\n");
    }
    else printf("Matrice A i B zadovoljavaju uvjet\n");
}
```

2. Napišite funkciju čiji je prototip: `char* nadjiZnak(char *niz, char z, int *br);`
   Funkcija treba vratiti pokazivač na prvo pojavljivanje znaka *z* u nizu *niz*, te odrediti *br* - broj pojavljivanja zadanog znaka u nizu *niz*. Ako se znak *z* ne pojavljuje u nizu *niz*, funkcija treba vratiti NULL pokazivač. Nije dozvoljeno korištenje funkcija iz `<stdio.h>`.
   Rj:

```
char* nadjiznak(char *niz, char z, int *br){
    char *temp = NULL;

    *br=0;
    while(*niz){
        if (*niz==z){
            if (*br==0) temp=niz;
            (*br)++;
        }
        niz++;
    }
    return temp;
}
```

3. Napišite funkciju čiji je prototip:
`void prepisi (char *imeUlazneDatoteke, char *imeIzlazneDatoteke);`
Vaš potprogram treba iz ulazne datoteke u izlaznu datoteku prepisati sve znakove, ali na način da sva mala slova pretvori u velika i obrnuto te umjesto svakog procitanog broja u izlaznu datoteku napise broj uvecan za zbroj njegovih znamenki.
Napomena: Brojevi u ulaznoj datoteci ne moraju biti prazninom odvojeni od ostalih znakova. Predznak broja možete zanemariti.
Npr. ako se u ulaznoj datoteci nalazio tekst "Ab!cDe.F251 G-22f", u izlaznoj datoteci treba biti zapisano "aB!CdE.f259 g-26F".
(Dodatnih 10 bodova dobit ćete ako ne koristite funkcije iz `<ctype.h>`. Napomena: A ima manji ASCII znak od a)
Rj:

```
void prepisi(char *imeUlazneDatoteke, char *imeIzlazneDatoteke){
    FILE *fin, *fout;
    char c;int temp_broj=0; int broj_nezapisan=0; int zbroj_znamenki=0;

    if ( (fin=fopen(imeUlazneDatoteke, "rt"))==NULL){
        printf("Ne mogu otvoriti datoteku %s\n", imeUlazneDatoteke);
        return;
    }
    if ( (fout=fopen(imeIzlazneDatoteke, "wt"))==NULL){
        printf("Ne mogu otvoriti datoteku %s\n", imeIzlazneDatoteke);
        return;
    }

    while ( (c=fgetc(fin))!=EOF ){
        if (c>='0' && c<='9'){
            temp_broj= temp_broj*10 + c - '0';
            broj_nezapisan=1;
            zbroj_znamenki += c-'0';
        }
        else{
            if (broj_nezapisan){
                    fprintf(fout, "%d", temp_broj+zbroj_znamenki);
                    temp_broj=0; broj_nezapisan=0;zbroj_znamenki=0;
            }
            if (c>='A' && c<='Z')
                fputc(c+'a'-'A', fout);
            else if (c>='a' && c<='z')
                fputc(c-('a'-'A'), fout);
            else fputc(c,fout);
        }
    }
    if (broj_nezapisan){
        fprintf(fout, "%d", temp_broj+zbroj_znamenki);
        temp_broj=0; broj_nezapisan=0;zbroj_znamenki=0;
    }
    fclose(fin);fclose(fout);
}
```

4. Definirana je struktura:
   struct *kompleksni_broj* {
   float re;
   float im; };
   Napišite funkciju koja kao argument prima polje kompleksnih brojeva i duljinu polja te ga sortira silazno po modulu.
   Rj:

```
double compl_abs(struct kompleksni_broj compl){
    return sqrt(compl.re * compl.re + compl.im * compl.im);
}

void f(struct kompleksni_broj kba[], int n){
    int i,j; struct kompleksni_broj temp;
    for (i=0; i<n-1; i++)
        for (j=i+1; j<n ; j++)
            if ( compl_abs(kba[i]) < compl_abs(kba[j]) ){
                temp = kba[i];
                kba[i] = kba[j];
                kba[j] = temp;
            }
}
```

5. Zadana je funkcija `int sumascii(char* s1, char* s2)` koja vraća zbroj ASCII vrijednosti znakova u prvom i drugom nizu znakova. Napišite funkciju čiji je prototip:
`struct student** makehash(struct student *lista, int n);`
Funkcija kao argumente prima pokazivač na početak vezane liste studenata i broj n. Svaki element strukture student sadrzi ime, prezime te prosjek ocjena studenta. Funkcija treba vratiti pokazivač na polje pokazivaca velicine n. Povratno polje treba biti ovako oblikovano:
Na i-tom mjestu u povratnom polju bit će adresa početka nove vezane liste koja će sadržavati sve one studente iz početne liste čiji je ostatak pri dijeljenju sume ascii vrijednosti znakova u imenu i prezimenu brojem n jednak i. Ako nema takvih studenata na i-tom mjestu u polju pokazivač mora biti NULL.
(Dodatnih 10 bodova dobit ćete ako elementi u pojedinoj vezanoj listi budu složeni po prosjeku.)
Rj:

```
struct student** makehash(struct student *lista, int n){
    struct student **pa;
    int i, ost;
    struct student *novi;

    pa = (struct student**) malloc(n*sizeof(struct student*));
    for (i=0; i<n ; i++){
        pa[i] = NULL;
    }

    while(lista!=NULL){
        ost = sumascii(lista->ime, lista->prezime) % n;

        novi = (struct student*) malloc(sizeof(struct student));
        strcpy(novi->ime,lista->ime);
        strcpy(novi->prezime,lista->prezime);
        novi->prosjek = lista->prosjek;

        novi->next = pa[ost];
        pa[ost] = novi;

        lista = lista->next;
    }
    return pa;
}
```

Rj2 (za dodatne bodove) (studenti se slažu po prosjeku)

```
struct student** makehash(struct student *lista,
int n){
    struct student **pa;
    int i, ost;
    struct student *novi;

    pa = (struct student**) malloc(n*sizeof(struct student*));
    for (i=0; i<n ; i++){
        pa[i] = NULL;
    }
    while(lista!=NULL){
        ost = sumascii(lista->ime, lista->prezime) % n;

        novi = (struct student*) malloc(sizeof(struct student));
        strcpy(novi->ime,lista->ime);
        strcpy(novi->prezime,lista->prezime);
        novi->prosjek = lista->prosjek;

        if (pa[ost]==NULL){
            novi->next=NULL;
            pa[ost] = novi;
        }
        else{
            struct student *prethodni, *trenutni;
            prethodni = trenutni =  pa[ost];

            while (trenutni->next!=NULL && trenutni->prosjek>novi->prosjek){
                prethodni=trenutni;
                trenutni = trenutni->next;
            }
            novi->next = prethodni->next;
            prethodni->next = novi;
        }

        lista = lista->next;
    }
    return pa;
}
```

Boris Milašinović