Programozás alapjai vizsgazárthelyi dolgozat 2011. január 12.

1. Függvények, függvénytípus, függvényhívási mechanizmus. Rekurzió. függvény pointer.

2. C preprocesszor direktívák, makroszimbólumok és makroeljárások.

Definiáljon függvényeket a következő feladatokra:

a. Visszatérő értéke 1, ha a paraméterként megadott A egész típusú, N elemű tömb elemei Fibonacci sorozatot alkotnak, különben 0. (a harmadik elemtől kezdődően minden elem értéke a megelőző két elem összege)

b. Paraméterek: az X és Y hosszú egész típusú, N ill. M elemű tömb. Visszatérő értéke az X tömb Y-ban nem szereplő elemeinek száma.

c. Paramétere az SZ karaktertömb. Törli SZ-ből a helyköz és tabulátor karaktereket. Visszatérő értéke a tömörített SZ tömb karaktereinek száma.

d. Kiír a standard outputra egy 8*8-as sakktáblát, a * és a helyköz karakterek alkalmazásával. A bal felső karakter * legyen. Visszatérő értéke nincs.

e. Visszatérő értéke 1, ha a paraméterként megadott N*N-es egész elemű tömb főátlón kivüli elemei zérusok, különben 0.

f. Dinamikusan helyet foglal a memóriában egy N elemű, hosszú egész típusú tömb számára. Visszatérő értéke a tömb címe, sikertelenség esetén NULL.

```
/*****************************************************************************
 *
 * Programozás alapjai 2011 január 12-ei vizsga kidolgozott tételei. Feladat leírása a
 * forrás végén
 *
 * Kidolgozta: Pozsgay Máté <matthew.linux@gmail.com>
 * Utolsó módosítás: 2011.01.22
 * Verzió: 1.0
 * ***************************************************************************/

#include <stdio.h>      //ki-be menet kezeléséhez
#include <stdlib.h>     //memóriafoglaláshoz
#include <time.h>       //idő lekérdezéséhez (véletlen véletlenszámokhoz)
#include <string.h>     //szövegkezeléshez

int afeladat(int *t, int n){
        int i=2; //kettőről indítjuk a ciklust, hogy össze tudjuk adni a nulladik, és az első elemet
        while(i<n && t[i]==t[i-2]+t[i-1])          //addig nézzük, amíg nem futunk ki a ciklusból
ÉS a jelenlegi elem megegyezik az előző kettő összegével
                i++;    //lépünk
        return !(i<n);  //Az (i<n) egy logikai kifejezés. C-ben a hamis állítás a nulla, az igaz a
nem nulla ( általában 1-es). Azt vizsgájuk, hogy kifutottunk-e a tömbből. Ha az állítás igaz,
akkor igen, tehát valamelyik elemnél elromlott a tulajdonság, ez azt jelenti, hogy 1-est kell
visszaadnunk, emiatt meg kell a '!'-el negálni a feltételt, hisz nekünk pont ellentétes
viselkedésre van szükségünk
}

int bfeladat(int *x, int *y, int n, int m){
        int i, j, s=0;
        for(i=0;i<n; i++){      //külső ciklussal végigmegyünk az X tömb minden elemén
                j=0;    //a balső ciklusban a második tömb minden elemén, ahol
                while(j<m && y[j]!=x[i])          //addig vizsgálódunk, amíg ki nem futunk a
tömbből ÉS az első tömb aktuálisan vizsgált eleme egyenl lesz egy másik Y beli elemmel
                        j++;    //lépünk a ciklusban
```

```
		if(j>=m)	//ha kifutottunk a ciklusból, akkor találtunk egy olyan elemet, ami nincs benne az Y-ban, de benne van az X-ben
			s++;	//megszámoljuk
	}
	return s;	//visszaadjuk a megszámolt találatokat
}

int cfeladat(char *sz){
		int i=0, j;	//elindulunk a szöveg elejéről
		while(i<strlen(sz)){	//addig megyünk, amíg el nem érjük a szöveg végét
			if(sz[i] == '\t' || sz[i] == ' '){	//Ha tabulátort, vagy szóközt találunk
				j=i;	//akkor egy másik ciklusban indulunk az aktuális pozíciótól
				while(sz[j] != '\0'){	//addig megyünk, amíg el nem érjük a szöveg végét
					sz[j]=sz[j+1];	//és a következő karaktert mindig ráírjuk a jelenlegire
					j++;	//lépünk
				}
			}
			i++;//lépünk a külső ciklusban is
		}
	return strlen(sz);	//visszaadjuk a módosított szöveg hosszát
}

void dfeladat(){
	int i, j, n=8;	//n=8 azt jelenti, hogy 8*8-as sakktáblát készítünk. Nem kötelező, n helyére beírhatjuk mindenhova a 8-ast, de így generikusabb.
	for(i=0; i<n; i++){	//készíteni kell n darab sort
		for(j=0;j<n; j++)	//minden sorban n darab karaktert fogunk beírni
			if((i+j)%2==0)	//ha i és j összege páros szám, akkor
				printf("*");		//'*'-t írunk
			else
				printf(" ");		//különben nullát
		printf("\n");		//minden sor után berakunk egy sortörést
	}
}

int efeladat(int **t, int n){
	int i, j;
	for(i=0; i<n; i++)		//egy mátrixban megyünk végig, minden során
		for(j=0; j<n; j++)	//és minden sor minden elemén
			if(i!=j && t[i][j]!=0)	//ha találunk egy olyan elemet, ami nem a főűtló ÉS nem nulla
				return 0;	//akkor egyből visszatérünk 0-van
	return 1;	//ha mindenhol teljesült a feltétel, akkor nyugodt szívvel mondhatjuk, hogy 1-est kell visszaadni
}

void *ffeladat(int n){
	return malloc(n*sizeof(int));	//malloc pont azt csinálja, amire nekünk szükségünk van, csak meg tudjuk neki mondani, hogy int méreteket kell lefoglalni.
}

int main(){
	int elemszam=10, i, j;
	srand(time(NULL));
	printf("\n2011.01.12-ei vizsgafeladatok megoldásai\nKészítette: Pozsgay Máté <matthew.linux@gmail.com>\n\nElső feladat:\n\tAdott a következő sorozat: ");

//---------------------------------ELSŐ FELADAT----------------------------
	int *it=malloc(elemszam * sizeof(int));
	it[0]=it[1]=1;
	printf("1, 1, ");
```

```
	for(i=2; i<elemszam; i++){
		it[i]=it[i-1]+it[i-2];
		printf("%d, ", it[i]);
	}
	printf("\n\tEz a sorozat Fibonacchi sorozat, ha a következő szám egy, nulla ha nem:
%d", afeladat(it, elemszam));

//---------------------------------MÁSODIK FELADAT-------------------------
	int *it2=malloc(elemszam * sizeof(int));
	printf("\n\nMásodik feladat\n\tAdott az alábbi két tömb:\n\t");
	for(i=0; i<elemszam; i++){
		it[i]=rand()%10;
		printf("%d, ", it[i]);
	}
	printf("\n\t");
	for(i=0; i<elemszam; i++){
		it2[i]=rand()%10;
		printf("%d, ", it2[i]);
	}
	printf("\n\tA másodikban nem szereplő, de az elsőben szereplő elemek száma: %d",
bfeladat(it, it2, elemszam, elemszam));
//---------------------------------HARMADIK FELADAT-------------------------
	char *sz="Ez egy szöveg, melyben sok szóköz van.";
	char *str=malloc(strlen(sz) * sizeof(char));
	strcpy(str, sz);
	printf("\n\nHarmadik feladat:\n\tA \"%s\" szöveg (hossza: %d, tömörített hossz: %d)
tömörítve így néz ki: \"%s\"", sz, (int)strlen(sz), cfeladat(str), str);

//---------------------------------NEGYEDIK FELADAT-------------------------
	printf("\n\nNegyedik feladat:\n\tA 8*8-as sakktábla így néz ki:\n");
	dfeladat();

//---------------------------------ÖTÖDIK FELADAT----------------------------
	int **im=malloc(elemszam * sizeof(*im));
	for(i=0; i<elemszam; i++)
		im[i]=malloc(elemszam * sizeof(*im));
	printf("\n\nÖtödik feladat\n\tAdott a következő mátrix:");
	for(i=0; i<elemszam; i++){
		printf("\n\t\t");
		for(j=0; j<elemszam; j++){
			if(i==j)
				im[i][j]=rand()%10;
			else
				im[i][j]=0;
			printf("%d ", im[i][ j]);
		}
	}
	printf("\n\tHa a fenti mátrix főátlón kívüli elemei nullák, akkor a következő szám egy,
különben nulla: %d", efeladat(im, elemszam));

//---------------------------------HATODIK FELADAT-------------------------
	int *nt=ffeladat(elemszam);
	printf("\n\nHatodik feladat:\n\t A foglalt memória kezdőcíme: %p", nt);

//---------------------------------FELSZABADÍTÁS----------------------------
	free(it);//kultúrált programozó lévén visszaadjuk az operációs rendszernek azt, ami az
övé
	free(it2);
	free(nt);
	free(str);
	free(im);	//lehet, hogy az elemeit is fel kéne szabadítani???
	return 0;
}
```

/*****FELADATOK*****
 * a. Visszatérő értéke 1, ha a paraméterként megadott A egész típusú , N elemű tömb elemei Fibonacci sorozatot alkotnak, különben 0. (a harmadik elemtől kezdődően minden elem értéke a megelőző két elem összege)
 * b. Paraméterek: az X és Y hosszú egész típusú, N ill. M elemű tömb. Visszatérő értéke az X tömb Y-ban nem szereplő elemeinek száma.
 * c. Paramétere az SZ karaktertömb. Törli SZ-ből a helyköz és tabulátor karaktereket. Visszatérő értéke a tömörített SZ tömb karaktereinek száma.
 * d. Kiír a standart outputra egy 8*8-as sakktáblát, a * és a helyköz karakterek alkalmazásával. A bal felső karakter * legyen. Visszatérő értéke nincs.
 * e. Visszatérő értéke 1, ha a paraméterként megadott N*N-es egész elemű tömb főátlón kivüli elemei zérusok, különben 0.
 * f. Dinamikusan helyet foglal a memóriában egy N elemű, hosszú egész típusú tömb számára. Visszatérő értéke a tömb címe , sikertelenség esetén NULL.
Csak szabad szoftver felhasználásával készült/
//EOF