Laborator 7

Transmiterea parametrilor prin valoare vs. transmiterea parametrilor prin pointeri

```
void swap( int a, int b )
{
	int temp=a;
	a=b;
	b=temp;
}
```

Cand apelam functia, e.g. swap(c, d) (c si d sunt alte variabile de tip int), astfel 'a' va contine o copie a lui 'c' si 'b' o copie a lui 'd'. Dupa executarea functiei valorile lui 'c' si 'd' vor ramane neschimbate.

Rescriem functia swap :

```
void swap( int* a, int* b )
{
  int temp=*a;
  *a=*b;
  *b=temp;
}
```

Apelul va fi urmatorul ; swap(&c, &d)

Insa cand folosim pointeri, nu se mai face o copie a variabilelor 'c' si 'd', ci se paseaza adresa lor astfel dupa executarea functiei 'c' si 'd' vor avea valoarea modificata, daca este cazul

Problema:

1. Se da o matrice n x m. Sa se afiseze suma maxima pe linii si pe coloane.

Functii cu numar variabile de argumente

O functie cu numar variabile de argument poate fi creata cu ajutorul header-ului ***stdarg.h*** si a macro-urilor definite in acesta: ***va_list, va_start(), va_arg()*** si ***va_end()***.

Prototipul unei functii cu numar variabil de parametrii este:

tip_functie nume_functie(tip_argument nume_argument, …);

Astfel, (***cel putin***) primul argument este intotdeauna fix, vizibil, restul argumentelor fiind declarate prin *mecanismul elipsa*, reprezentat de cele 3 puncte dupa virgula.

Cele 4 constructii definite in header si utilizate sunt:

- ***va_list*** : variabila pointer, de tip *void**, folosita pentru parcurgerea listei de argumente
- ***void va_start(va_list obiect, NumeArgument)*** : initializeaza parcurgerea la primul element
    - Parametrii:
        - *obiect* : variabila de tip va_list care va parcurge lista de argumente
        - *NumeArgument* : numele ultimului argument vizibil din definitia functiei
- ***tip va_arg(va_list obiect, tip)*** : ofera urmatorul argument
    - Parametrii:
        - *obiect* : variabila de tip va_list care va parcurge lista de argumente; devine urmatorul argument din lista
        - *tip* : tipul de date al argumentului
- ***void va_end(va_list obiect)*** : termina actiunea de parcurgere a listei, eliberand si memoria astfel
    - Parametrii:
        - *obiect :* variabila de tip va_list folosita la initializare, prin apelarea va_start()

```
void test(int nr_param, ...)

{

    /* Pentru a avea acces la parametri functiei, avem nevoie de o variabila de tipul "va_list".  E necesar sa includem
<stdarg.h> pentru asta.  Practic aceasta variabila este un pointer care va indica rand pe rand catre parametri
variabili ai functiei. */
```

```
    va_list ap;


    int i, n;

    /* Primul pas este sa initializam variabila de mai sus folosind macro-ul "va_start". Varibila se initializeaza
relativ la ultimul parametru fix. In cazul nostru avem un singur parametru fix, deci nu avem de ales. */

    va_start(ap, nr_param);

    /* Parametri ficsi pot fi accesati direct, asa cum suntem obisnuiti. Noi folosim singurul parametru fix pentru
a parcurge intr-o bucla parametri variabili. */

    for (i=0; i<nr_param; i++)

    {

        /* Pentru a afla valoarea unui parametru variabil, folosim macro-ul "va_arg". La fiecare apel al acestui macro
se  returneaza un parametru variabil (cel spre care indica pointer-ul "ap" definit mai sus). Dupa apel, in mod
automat variabila "ap" este setata sa indice spre urmatorul  parametru variabil. */

        n = va_arg(ap, int);

        }


    /* Dupa ce am parcurs toti parametri variabili, apelam macro-ul  "va_end". Acesta face operatii necesare pentru
curatarea  memoriei. */

    va_end(ap);

}
```

Exemplu de apel : `test(7, 0, 3, 1, 6, 1, 4, 1);`

Probleme:

1. Scrieti o functie cu numar variabil de argumente care primeste ca prim parametru un numar N, iar ca urmatorii N parametri primeste N numere reale. Functia va returna suma celor N numere reale. Verificati functia prin cateva apeluri ale ei cu diverse seturi de parametri.

2. Scrieti o functie cu numar variabil de argumente care primeste ca prim parametru un numar N, iar ca urmatorii N parametri primeste N numere intregi. Functia va afisa pe ecran pentru fiecare argument caracterul * daca numarul este perfect, # daca nu.

3. Scrieti o functie cu numar variabil de argumente care primeste ca prim parametru un numar N, o matrice N x N iar ca urmatorii N parametri primeste N numere reale. Functia va afisa pe ecran pentru fiecare coloana a matricei 1 in cazul in care cele N numere reale sunt o permutare a coloanei, 0 daca nu.

---

4. Se da o matrice n x m, sa se afiseze numarul maxim de valori de 1 alaturate existente in matrice.
Ex : n=3, m=4 ;
1 1 0
0 0 1
1 1 1 se va afisa 4.