Objectiu de la pràctica:

- **Assolir els coneixements necessaris sobre estructures d'informació basades en sèries de dades del mateix tipus: *vectors i matrius en C/C++*.**

Conceptes bàsics: VECTORS I MATRIUS

Un vector o array -arreglos en algunes traduccions- es una seqüència d'objectes del mateix tipus emmagatzemats de forma seguida en memòria. El tipus d'objecte emmagatzemat en l'array pot ser qualsevol tipus definit en C/C++.

Adoptarem el conveni d'usar ***typedef*** per a crear tipus que seran els vectors i matrius. Així

```
typedef    int vector20[20];  //defineix un vector de 20 enters
vector20  vector_meu; //declare un vector de 20 enters.
```

L'accés als elements d'un vector es fa mitjançant un índex. Els elements estan ordenats tot i començant des de 0.

Ex. `vector_meu[3]=45;  //assigna el valor 45 a la quarta component`

Analogament passa amb matrius o arrays bidimensionals:

```
typedef  int  matriu10x4[10][4];
matriu10x4  matriu;
matriu_meua[1][3]=32;
```

Els vectors i matrius són passats sempre per referència com arguments d'una funció. L' única diferència amb els tipus simples és que no s'usa el '***&***' atés que només caldrà simplement el nom del vector o matriu.

INICI D'UN VECTOR.

Com tota variable, un vector pot prendre valors inicials després de la seva declaració. El format general és:

tipus identificador_variable [amplada] = { llista_de_valors };

La llista de valors és una llista separada per comes " , ", de constants que són del mateix tipus que el tipus base del vector.

Exemple:

```
int num[5] = {0,1,2,3,4};
```

En aquest exemple la primera constant de valor 0 l'emmagatzemarà a la primera posició del vector, la segona constant de valor 1 a la segona posició del vector, i així successivament fins completar les cinc constants. El compilador les situarà en posicions contigües de memòria.

No és necessari posar valor inicial a tot el vector, en aquest cas el compilador posarà zeros a tots aquells elements que no li hem assignat cap valor. Per exemple:

```
int num[5] = {0,1,2};
```

En aquest exemple el compilador assignarà zeros als dos últims elements del vector.També és possible al posar els valors inicials al vector, no declarar la seva grandària. El compilador la determina calculant el nombre de valors enumerats.

Així, l'assignació:

```
int num[] = {0,1,2,3,4};
```

fa que la dimensió de la variable *num* siga 5.

Per posar valors inicials als vectors multidimensionals ho farem d'una forma semblant a la feta pels vectors unidimensionals. Són exemples de posar valors inicials les següents declaracions d'assignació:

```
int tab[2][5] = {{0,1,2,3,4},{5,6,7,8,9}};
```

equivalent a:

```
int tab[2][5] = {0,1,2,3,4,5,6,7,8,9};
```

i també equivalent a:

```
int tab[2][5] ={
                  {0,1,2,3,4},
                  {5,6,7,8,9}
                 };
```

En aquest exemple declarem una matriu d'enters de dues files per cinc columnes. Així, com en els vectors unidimensionals, el posar valor inicial ho podíem fer en forma parcial, ara també és possible, a condició de començar pel principi de cada índex del vector.

No obstant és necessari anar en compte perquè en les declaracions incompletes les confusions són fàcils com mostra el següent exemple:

```
int taula1[2][4] = {1,2,3,4,5,6,7,8};

/* valors inicials complets, correspon a:

1 2 3 4
5 6 7 8

*/

int taula2[2][4] = {1,2,3};

/* valors inicials incomplets, correspon a:

1 2 3 0
0 0 0 0

*/

int taula3[2][4] = {{1},{2,3}};

/* valors inicials incomplets, correspon a:

1 0 0 0
2 3 0 0

*/

/* valors inicials a un vector sense dimensions. La dimensió indeterminada serà sempre la
primera */
```

```
int taula4[][2] = {{1,2},{3,4},{5,6}};

/* valors inicials que correspon a:

1 2
3 4
5 6

*/
```

(*)EXERCICI 1.
Feu un programa que:

a) Lliga una seqüència de 20 valors numèrics reals i ho emmagatzeme en un vector.
b) Ens mostre per pantalla el valor màxim, així com la posició que ocupa en el vector. En el cas d'estar repetit el valor màxim es mostrarà el menor índex dels valors màxims.

(*)EXERCICI 2.
Compila i executa el programa ***incògnita.cpp.*** Què fa el programa?.

(*)EXERCICI 3
Feu un programa que "suavitze"un vector introduït per l'usuari. Entenem per "suavitzar", substituir cada element per la mitjana aritmètica dels elements més propers, és a dir :

```
uS[i]->(u[i-1]+u[i]+u[i+1])/3        si i diferent de 0 i n-1
uS[0]->(u[0]+u[1])/2
uS[n-1]=(u[n-2]+u[n-1])/2
```

Per exemple, el vector ***u*** de la primera columna de la taula s'ha de convertir en el vector u^S de la segona columna.

u	u^S
14	9,5
5	6,666666667
1	5,333333333
10	7,666666667
12	7,666666667
1	6,333333333
6	2,666666667
1	2,666666667
1	5,333333333
14	7,333333333
7	10,5

Caldrà definir una funció (procediment) *suavitzar,* amb prototipus:

```
typedef double vector[TAM];
void suavitzar(vector a, int n); // n serà la mida del vector de  a de reals
```

Hauràs de passar-li a la funció el vector que prèviament has introduït pel teclat i la funció tornarà per pantalla el vector original i el seu vector "suavitzat".

(*)EXERCICI 4

Calculeu la matriu transposada d'una matriu . Una matriu transposada d'una altra és el resultat de canviar de la matriu original els valors de les files pels de les columnes.

Hom podrà definir de forma opcional un prototipus:

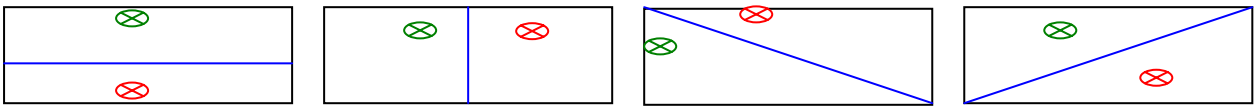
```
const int MAX_F = 100;
const int MAX_C = 100;
typedef matriz double matriz[MAX_F][MAX_C];
void transposada(matriz a, int f,int c);
```

Aquesta funció rebria la matriu original introduïda pel teclat i el nombre de les seues columnes i files com uns enters *f* i *c* i haurà de calcular la seua transposada. El programa principal haurà de mostrar per pantalla la matriu original i la seua transposada

(*)EXERCICI 5

En una matriu quadrada (nxn) d'enters hom defineix quatre eixos de simetria - vegeu la figura-.Per a un element donat (i,j) hi haurà , per cada eix, un element de la matriu (i*, j*) que ocupe una posició simètrica amb l'anterior respecte a l'esmentat eix.
Hom demana una funció que per a un element (i, j) de la matriu, determine quins valors, dels quatre possibles simètrics, són iguals a (i, j). El resultat serà tornat en un vector de quatre elements, un per simetria, que indicarà amb un valor 1, o *true,* que ambdos simètrics seran iguals o, amb el valor 0 – *false*- quan seran diferents

Element (i, j) ⊗ ***Element simètric*** ⊗



SIMETRIA 0 ***SIMETRIA1*** ***SIMETRIA2*** ***SIMETRIA3***

AJUDA: Per exemple, si hi ha la matriu:

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4
4,1	4,2	4,3	4,4

Els elements simètrics de l'element (1,2) -color verd- serien els quatre elements amb roig.

1. Trobeu una relació entre els índex dels elements per a determinar les simetries. Per exemple, per a una simetria :

   (1,2)→(1,3).....i=i ; j*=n-j+1 // els índex en C/C++ comencen per 0... ull!*

2. El prototipus opcional de la funció podria ser:

void simetries(vectort b, int i, int j, matriu m, int f, int c); ;// b[0]=1 si hi hi ha simetria0…

m : representa la matriu sobre la que calculem simetries
b: seria el vector que recolliria els 4 tipus de simetria possibles:0,1,2,3
i,j: element sobre el que calculem simetries

Feu un programa que demane per pantalla la introducció d'una matriu quadrada de nxn elements -n=4 com l'exemple-, l'element (i, j) i s'obtinga el vector b i un missatge indicant els elements simètrics iguals.
Caldrà fer el programa amb les màximes funcions possibles, per exemple: la introducció de la matriu, l'obtenció de resultats i el càlcul de les simetries serien algunes de les funciones per a fer.