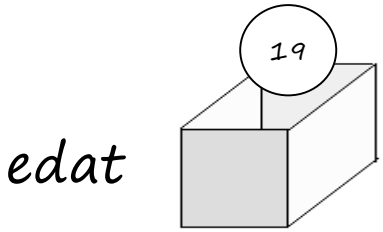
# Tema 5

# Estructures estàtiques. Taules

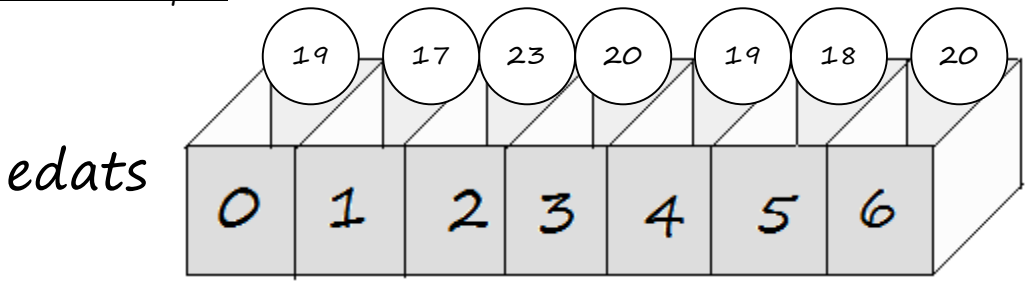


## 1 Introducció

Tots els tipus de dades que hem vist fins ara són tipus bàsics (també anomenats simples, atòmics o elementals). Es caracteritzen perquè les variables d'eixos tipus només poden guardar un únic valor en cada moment.




Ara vorem com guardar molts valors en una sola variable. Eixa variable serà d'un tipus de dades compost.




Per a què servixen estos tipus de dades compostos?

Imaginem que volem un programa que haja de guardar les notes de 100 alumnes per a, després, fer operacions amb elles (màxima, mínima, ordenar-les...).

- ✓ Amb els tipus de dades bàsics necessitaríem **100 variables** i, a més, les operacions sobre elles es farien molt complicades.
- ✓ Amb els tipus de dades compostos necessitarem només **1 variable** (de grandària 100) i les operacions seran més senzilles, utilitzant bucles.

Ja vorem com crear variables compostes, com introduir valors en elles i com accedir a eixos valors.

Este tipus de dades que hem vist s'anomena **taula** o **vector**. Vorem que hi ha més tipus de dades compostos, com poden ser les **matrius** (vector de vectors), registres, etc.

Definició de vector: conjunt finit d'elements del mateix tipus, als quals podem accedir mitjançant un índex i el nom del vector. També s'anomenen taules o arrays.

Exemple: el lloc on podem guardar les notes de 100 alumnes.

# 2 Estructura dels vectors. Implementació en C

## 2.1 Declaració

La forma general de declarar un array és esta:

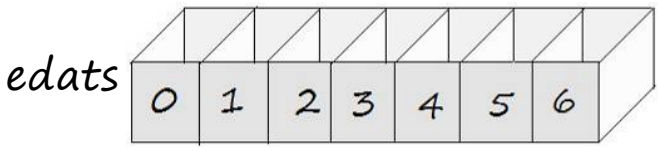
```
tipus_de_dades  nom_vector[ dimensió ]
```

On tenim que:

- ✓ *tipus_de_dades* és un dels tipus coneguts (int, char, float, double)
- ✓ els [ ] no indiquen opcionalitat sinó que cal posar-los obligatòriament.
- ✓ *dimensió* és la quantitat d'elements que té l'array.

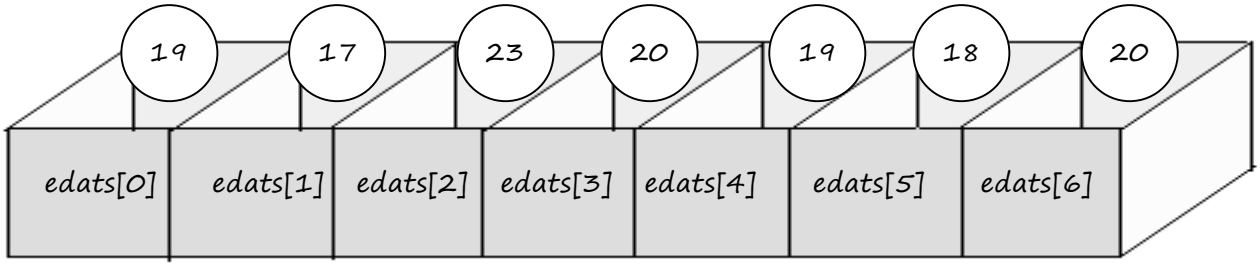
Per exemple:

```
int edats[7];
```



## 2.2 Tractament dels elements

Per a accedir a cada element del vector, ho farem amb el nom del vector i l'índex corresponent tancat entre claudàtors:




Cada element del vector es pot tractar com una variable simple:

- Podem donar valor a un component:
  - notes[3] = 7; // Assignem un valor directament
  - scanf("%i", &notes[4]); // Assignem un valor des de teclat
- O podem consultar el valor d'un component:
  - printf("%i", notes[3]); // Mostrem el valor per pantalla
  - nota_Pep = notes[4]; // Copiem el valor a altra variable
  - if (notes[5] < 5) ... // Utilitzem el valor en una expressió

L'índex del vector és un enter (constant, variable o expressió entera):

```
...
printf("Quina edat vols saber (de l'1 al 7)?");
scanf("%i", &num_persona);
printf("La persona %i té %i anys", num_persona, edats[ num_persona - 1 ];
...
```

## 2.3 Inicialització d'un array

Quan definim un vector, té valors desconeguts. Per tant, si volem usar-lo com a comptadors o acumuladors, haurem d'inicialitzar el vector. Es pot fer de diverses maneres:

### Inicialitzar un array en la declaració

En C es poden inicialitzar els arrays en el moment de declarar-los, igual que es fa amb les variables:

```
int temp[24] = {12, 11, 11, 10, 10, 9, 8, 9, 10, 11, 12, 12,
                13, 14, 14, 14, 15, 14, 13, 14, 13, 13, 12, 12};
```

A partir d'este moment, el primer element del vector temp, és a dir, temp[0], valdrà 12. El següent (temp[1]) valdrà 11, i així amb tots.

Exemple:

```
#include <stdio.h>
int main() {
        int hora;
        int temp[24] = {12, 11, 11, 10, 10, 9, 8, 9, 10, 11, 12, 12,
                        13, 14, 14, 14, 15, 14, 13, 14, 13, 13, 12, 12};
        for (hora=0; hora<24; hora++) {
                printf("La temperatura a les %d era de %d graus.\n", hora, temp[hora]);
        }
}
```

Si inicialitzem **més elements dels que toquen**, donarà error de compilació (però depén del compilador):

```
int notes[3] = {8, 7, 4, 6}  // Error de compilació
```

Ara, bé: si inicialitzem **menys elements** dels que hi ha al vector, no donarà error i, a més, els elements no inicialitzats no tindran valor desconegut, sinó valor 0:

```
int temp[24] = {12, 11};   // El 1r element valdrà 12. El segon 11.
                           // Els altres elements valran 0.
```

Per tant, si volem inicialitzar tot un vector a zeros, una forma ràpida és:

```
int punts[22] = {0};          // Inicialitza tots els elements a 0.
```

Però, i si volem inicialitzar tot el vector a un número diferent de 0? Imaginem que volem guardar l'edat dels 16 alumnes i que tots tinguen 18 anys. Faríem açò?:

```
int edats[16] = {18};
```

No, ja que això posaria 18 anys al primer alumne i 0 anys als altres.

## Inicialitzar un array amb assignacions

Per a assignar un mateix valor a tots els components del vector, haurem d'inicialitzar component a component. Ho podem fer amb un bucle:

```
int edats[16];
int i;
for (i=0; i<16; i++)
  {
    edats[i]=18;
  }
```

Compte! Si sobrepassem per dalt o per baix els límits del vector, el compilador no donarà error però estarem accedint a una dada indeterminada de memòria que no és la que volem. Si per exemple fem:

```
printf("%d", edats[16])
```

... estem accedint a l'element 17 del vector,. que no existix i, per tant, estarà mostrant una cosa que no és la que volem.

## Inicialitzar un array sense indicar la dimensió

Altra forma de declarar un array és sense indicar la quantitat d'elements. Simplement, s'inicialitzen:

```
int notes[] = {7, 8, 6}; // Declarem un vector de 3 elements
```

## 2.4 Recorregut d'arrays

Quines operacions podem fer amb els vectors?

Si suposem que tenim estos 2 vectors:

```
int temp_dilluns[24], temp_dimarts[24];
```

... les següents instruccions donarien error:

```
scanf("%d", &temp_dilluns);        // error !
printf("%d", temp_dilluns);        // error !
temp_dimarts = temp_dilluns;       // error !
temp_dimarts = 10;                 // error !
```

És a dir: no podem llegir tot el vector amb un únic *scanf*, ni podem mostrar el contingut d'un vector tot d'una tacada, ni podem copiar vectors directament, ni podem assignar un valor a tot el vector...

Hem de tindre clar que només té sentit accedir als elements del vector un a un, i no com a conjunt. Per tant, haurem de recórrer els elements de l'array un a un. Com? Amb un bucle. Generalment, un **for**:

```
int temp_dilluns[24], temp_dimarts[24];
int hora;       // farà d'índex del vector

// Llegim de teclat els valors del vector del dilluns
for (hora=0 ; hora<24; hora++) {
        scanf("%d\n", &temp_dilluns[hora]);
}

// Mostrem el vector del dilluns
for (hora=0 ; hora<24; hora++) {
        printf("%d\n", temp_dilluns[hora]);
}

// Copiem el vector del dilluns al del dimarts
for (hora=0 ; hora<24; hora++) {
        temp_dimarts[hora] = temp_dilluns[hora];
}
```

És a dir, la idea és fer servir una variable comptador (*hora*), el valor de la qual variarà entre el rang admés de l'índex del vector (entre 0 i 24-1).

És important els límits del comptador. Vegem què passaria en estos 2 exemples:

```
int num;
int notes[16] = {8,7,7,5,4,5,6,4,8,9,10,5,4,5,8};
for (num=0; num<=16; num++) {
        printf("La nota de l'alumne %d és %d\n.", num, notes[num];
}
```

```
int num;
int notes[16];
float mitja=0;
for (num=0; num<=16; num++) {
        printf("\nDis-me la nota de l'alumne %d:", num);
        scanf("%d", &notes[num];
        mitja += notes[num];
}
mitja = mitja / 24;
printf("\nLa nota mitja és %f\n", mitja);
```

Els 2 programes anteriors estan mal, però un és catastròfic. Quines són les incorreccions? Quin és el catastròfic i per què?

## Exercicis

1. Fes un programa que faça el següent:
    a) Crear un array de nom vector, de 5 elements de tipus enter.
    b) Mostrar el contingut de cada element.
    c) Assignar a cada element el valor de 0.
    d) Tornar a mostrar el contingut de cada element.
    e) Calcular i mostrar la grandària de l'array en bytes.

> Consell: mira el punt 2.8 sobre la representació gràfica d'un vector.

2. Fes un programa que faça el següent:
    a) Crear un array de nom edats, de 10 elements de tipus enter.
    b) Omplir el vector demanant les edats per teclat.
    c) Demanar un número d'alumne i mostrar la seua edat (sense donar error).
    d) Calcular la suma dels n primers elements (paràmetre donat per l'usuari).

3. Programa que guarde els 100 primers números parells en un vector. Després es mostrarà la suma dels últims 30 números guardats.

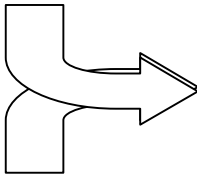
4. Fes un programa que llija les 24 temperatures que s'han produït en un dia i les guarde en un vector. Caldrà mostrar quantes són negatives, quantes positives o zero, la mínima, la màxima i la mitjana. També cal mostrar a quina hora s'ha produït la temperatura mínima (si hi ha moltes, la que es produí primer).

5. Fes un programa que:
    a) Cree el vector origen, inicialitzat amb: {15,22,55,67,33,46,26,54,21,4}
    b) Recórrega el vector posant en un segon vector, de nom destí, tots els elements parells majors de 30.
    c) Mostre per pantalla el contingut del segon vector (només els valors posats).

## 2.5 Vectors paral·lels

Suposem que volem guardar les notes de 15 alumnes (float) i els respectius números d'expedient (int). Per a això necessitarem un vector de 15 enters per a guardar els números d'expedient i altre vector de 15 reals per a guardar les notes.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| notes | 8.0 | 9.3 | 5.6 | 7.3 | 7.0 | 5.4 | 6.7 | 4.2 | 3.8 | 9.4 | 8.4 | 6.4 | 5.0 | 6.2 | 5.8 |



L'alumne que figura en la 3a posició dels vectors ha tret de nota un 5.6 i té el número d'expedient 139.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| expedients | 133 | 137 | 139 | 138 | 136 | 130 | 140 | 142 | 131 | 134 | 138 | 145 | 141 | 147 | 148 |

Diem que eixos **2 vectors són paral·lels** perquè tenen la mateixa quantitat d'elements, i en la mateixa posició de cada vector guarda informació del mateix objecte.

El fet de tindre vectors paral·lels té l'avantatge que, per a mostrar totes les dades, es pot fer en una única passada. Per exemple, suposant que ja tenim guardades les notes i els expedients en eixos vectors, per a imprimir un llistat dels números d'expedient dels alumnes amb les seues notes, seria tan fàcil com fer:

```
for (i=0; i< QUANT_ALUMNES ; i++)

    printf("L'alumne %d ha tret un %d.\n", expedients[i], notes[i]);
```

### Exercicis

6. Fes un programa que demane el número d'expedient, nota i edat de 15 alumnes (amb un únic bucle). Posteriorment, que mostre en format de taula totes les dades. Després, demanar per teclat un número d'expedient i mostrar per pantalla la seua edat i nota.

## 2.6 Utilitat dels vectors

Ja sabem com s'utilitzen, però per a què servixen? No dóna igual tindre una variable tipus vector de 24 enters que 24 variables simples de tipus enter?

Observa este programa:

```
#include <stdio.h>
int main() {
// Declarem 24 variables, per a les temperatures de cada hora del dia
int temp0, temp1, temp2, temp3, temp4,  temp5, temp6, temp7, temp8, temp9,
   temp10, temp11, temp12, temp13, temp14, temp15, temp16, temp17, temp18,
   temp19, temp20, temp21, temp22, temp23;
float mitja;
// Assignem valors a cadascuna
printf("Dis-me la temperatura des de les 0 fins les 23 hores separades per espai");
scanf("%d %d %d %d %db%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d",
   &temp0, &temp1, &temp2, &temp3, &temp4, &temp5, &temp6, &temp7, &temp8,
   &temp9, &temp10, &temp11, &temp12, &temp13, &temp14, &temp15, &temp16,
   &temp17, &temp8, &temp19, &temp20, &temp21, &temp22, &temp23);
// Calculem la mitja
mitja = (temp0 + temp1 + temp2 + temp3 + temp4 + temp5 + temp6  + temp7 + temp8
      + temp9 + temp10 + temp11 + temp12 + temp13 + temp14 + temp15 + temp16
      + temp17 + temp18 + temp19 + temp20 + temp21 + temp22 + temp23) / 24;
printf("\nLa temperatura mitjana és %f\n", mitja);
}
```

No es fa un poc farragós? Imagina si haguérem de guardar 1000 temperatures! El mateix programa es pot simplificar molt:

```
#include <stdio.h>
int main() {
// Declarem 1 variable de 24 elements, un per a cada hora del dia
int temp[24];
float mitja = 0;
int hora;
// Assignem valors a cada element i anem calculant la mitja
for (hora=0; hora<24; hora++) {
        printf("Temperatura de les %d: ", hora);
        scanf("%d", &temp[hora]);
        mitja += temp[hora];
}
// Acabem de calcular la mitja
mitja = mitja / 24;
printf("\nLa temperatura mitjana és %f\n", mitja);
}
```

Un vector és l'estructura idònia per al tractament de llistats (de preus, notes, temperatures...) a fi de poder-los ordenar, buscar, traure mitges, màxims, mínims, etc.

## 2.7 Treballant amb la dimensió del vector

L'últim programa que hem vist és més fàcil d'escriure i de mantindre que l'anterior.

Però encara es pot millorar més. En este segon programa hem indicat en 3 llocs la grandària de l'array (24). I si el programa fóra més gran, probablement caldria escriure esta grandària moltes voltes més. Si al cap d'un temps cal modificar el programa perquè es tinguen en compte 48 temperatures, tenim 2 inconvenients:

- ✓ Molta feina canviant els 24 per 48
- ✓ Possibilitat d'error si ens en deixem algun per canviar

Per això, per a indicar la dimensió del vector, és millor usar una constant simbòlica, per exemple HORES:

```
#include <stdio.h>
#define HORES 24
int main() {
int temp[HORES];
float mitja;
int hora;
for (hora=0; hora< HORES; hora++) {
        printf("Temperatura de les %d: ", hora);
        scanf("%d", &temp[hora]);
        mitja += temp[hora];
}
mitja = mitja / HORES;
printf("\nLa temperatura mitjana és %f\n", mitja);
}
```

Si ara volguérem canviar el programa perquè foren 48 hores, només hauríem de canviar una línia de codi: la del #define.

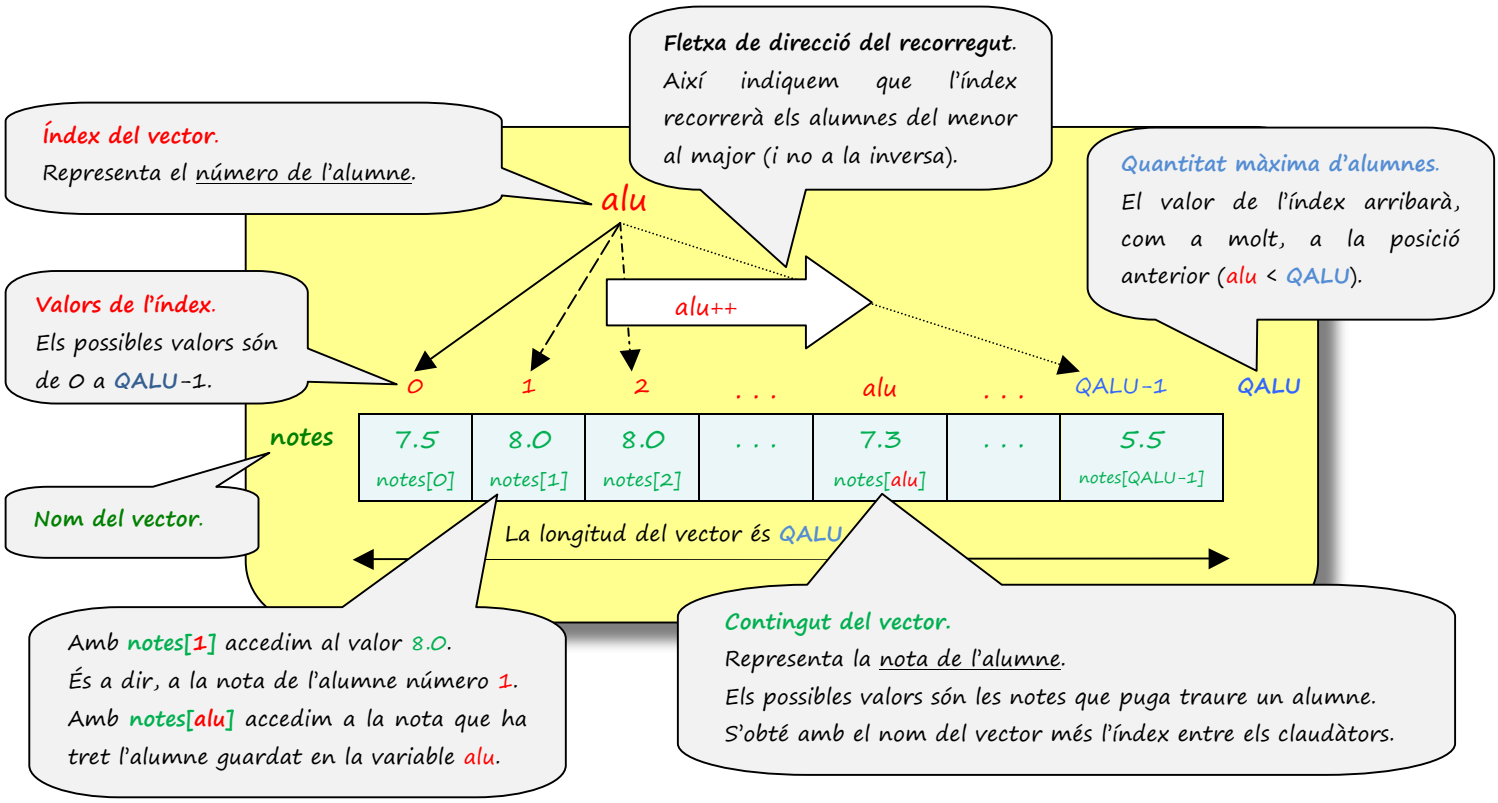
## 2.8 Representació gràfica dels vectors

Davant d'un problema sobre vectors, hem de plantejar-nos el dibuixet del vector amb les variables que li afecten, etc.

Per exemple, volem guardar les notes de 25 alumnes. Suposant estes definicions:

```
# define QALU 25    // Quantitat d'alumnes (majúscules).
...
float notes[QALU];// Variable composta per a guardar les notes de 25 alumnes
int alu;              // Índex (minúscules). Cada volta guardarà el número d'un alumne.
```

... faríem la següent representació gràfica:




A partir del dibuix anterior, podem obtindre més fàcilment el codi que s'encarregarà de recórrer el vector per a guardar totes les notes.

```
for ( alu = 0 ; alu < QALU ; alu++) {
    printf("Dis-me la nota de l'alumne %i:", alu);
    scanf("%f", &notes[alu]);
}
```

# 3 Cadenes de caràcters

En C no hi ha un tipus bàsic per a les cadenes de caràcters, sinó que es representen com a vectors de caràcters, on cada casella és un caràcter (una lletra) i l'última casella està el caràcter nul ( '\0' ):

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| nom | P | e | p | \0 | | | | | | | |

Les cadenes en C es diu que tenen grandària fixa però longitud variable. És a dir:

- ✓ **Grandària fixa**: quan reserve espai per a una cadena, indique la grandària del vector, que ja no variarà al llarg del programa. És a dir, si vull una cadena de 10 caràcters, la reservaré d'11 (una més per al caràcter nul del final).
- ✓ **Longitud variable**: dins d'eixos 10 caràcters podré tindre una cadena de longitud 3 ("Pep"), 10 ("Pep Garcia"), etc.

Per què és tan important el \0? Perquè actua com a finalitzador. Imaginem la cadena "Hola, bon dia":

| H | o | l | a | , | | b | o | n | | d | i | a | \0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Si la sobreescrivim amb "BON DIA", l'array quedarà:

| B | O | N | | D | I | A | \0 | n | | d | i | a | \0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

On acaba la cadena? Fàcil: on està el primer \0.

## 3.1 Declaració i inicialització de cadenes de caràcters

Les cadenes de caràcters es declaren com un array de caràcters:

```
char   nom_cadena[ grandària ] ;
```

Igual que qualsevol vector, també es pot inicialitzar en la declaració:

```
char nom[20] = {'N', 'e', 'u', 's', '\0'};
char cognoms[]= {'G', 'a', 'r', 'c'. 'i", 'a', ' ', 'M', 'a', 'r', 'q', 'u', 'é', 's', '\0'};
```

Però una forma més còmoda d'inicialitzar-lo és la següent:

```
char cognoms[]= "Garcia Marqués";     // El '\0' es posa automàticament
```

Per tant, podem declarar una cadena de caràcters de les següents maneres:

```
#define N 20
char frase1[N] = {'B', 'o', 'n', ' ', 'd', 'i', 'a', '\0'};
char frase2[] =  {'B', 'o', 'n', ' ', 'd', 'i', 'a', '\0'};
char frase3[N] = "Bon dia";
char frase4[] = "Bon dia";
```

Però cal tindre en compte que **estes assignacions només es poden fer en el moment de la declaració**. És a dir, a mitjan programa NO podem fer assignacions com esta: *frase = "Bon dia"* (ja que frase és una variable que conté una adreça de memòria).

## 3.2 Lectura de cadenes de caràcters

Per a llegir una cadena des del teclat ho podem fer de diverses maneres: caràcter a caràcter o de colp. Per exemple, si tenim: *char cadena[100];*

| | |
|---|---|
| scanf("%s", cadena); | Guarda la primera paraula de la frase (fins espai o intró) |
| scanf("%[^\n]", cadena); | Guarda tota la frase (fins intro) |
| gets(cadena); | Guarda tota la frase (fins intro) |

Nota: cal tindre en compte que NO posem el & davant la variable cadena.

## 3.3 Escriptura de cadenes de caràcters

De dos formes:

| |
|---|
| printf(“%s”, cadena); |
| puts(cadena) |

També podríem escriure-ho caràcter a caràcter:

```
int i;
for (i=0; cadena[i]!='\0'; i++)
    printf("%c", cadena[i]);
```

*Exercici*

7. Què haguera passat si la condició del bucle d'escriptura de cadenes de caràcter a caràcter haguera sigut (i<N-1) ?

# 4 Funcions per a la utilització de cadenes

## 4.1 Problemàtica de les cadenes

Com ja hem vist, en C la representació de cadenes es fa amb vectors de caràcters. Per exemple, si vull definir una variable de 50 caràcters com a màxim, ho puc fer amb:

```
char cognoms[51];   // 51 ja que un és per al '\0'
```

Per tant, ja que són vectors, NO PODEM fer directament coses com:

- ✓ Comparar cadenes (s==t), ja que estaríem comparant adreces de memòria.
- ✓ Copiar cadenes (s=t), ja que estaríem copiant una adreça de memòria.
- ✓ Assignar de colp una cadena (s=”Hola”). Donaria error de compilació.

Com les cadenes (o *strings*) s'utilitzen molt en la vida real, C aporta solucions per a treballar còmodament amb elles. Per a tal fi, C disposa de funcions adequades, existents a la biblioteca *string.h*.

## 4.2 La biblioteca string.h

Per a l'ús de cadenes de caràcters en C es disposa de la biblioteca **string.h**, la qual es deu incloure de la forma habitual **#include string.h**. Algunes de les funcions que incorpora esta biblioteca són:

| FUNCIÓ | DESCRIPCIÓ |
|---|---|
| strcat(destí, origen) | Afig la cadena origen al final de la cadena destí |
| strncat(destí, origen, N) | Afig N caràcters de la cadena origen al destí |
| strchr(cadena, caracter) | Busca un caràcter en una cadena, des del principi |
| strrchr(cadena, caracter) | Busca un caràcter en una cadena, des del final |
| strcmp(cadena1, cadena2) | Compara 2 cadenes |
| strncmp(cad1, cad2, N) | Compara els N primers caràcters de cad1 en cad2 |
| strcpy(destí, origen) | Copia la cadena origen a la cadena destí |
| strncpy(destí, origen, N) | Copia N caràcters de la cadena origen en destí |
| strerror(N) | Torna el missatge d'error corresponent al número N |
| strlen(cadena) | Torna la longitud de la cadena |
| strstr(cadena1, cadena2) | Busca la cadena1 dins de la cadena2 |

Podeu buscar informació sobre qualsevol d'estes funcions a la web http://c.conclase.net. Nosaltres ens centrarem en strcmp, strcpy, strcat i strlen.

## 4.3 Comparar cadenes: strcmp

Bàsicament servix per a saber si 2 cadenes són iguals. La sintaxi és:

```
strcmp (cadena1, cadena2)
```

Esta funció rep 2 cadenes i retorna un enter, el qual s'interpreta així:

= 0 → Les cadenes 1 i 2 són idèntiques.

< 0 → La cadena 1 és menor que la cadena 2 (seguint l'ordre ASCII).

> 0 → La dadena 1 és major que la cadena 2 (seguint l'ordre ASCII).

Exemple:

```
char s1[]=''hola'', s2[]=''adéu'', s3[]=''hola'';

// strcmp(s1, s2)   →    1
// strcmp(s2, s1)   →   -1
// strcmp(s1, s3)   →    0

switch (strcmp(s1, s2))
{ case 1:       printf("%s major que %s", s1, s2); break;
  case -1:      printf("%s menor que %s", s1, s2); break;
  case 0:       printf("iguals"); break;
}
```

## 4.4 Copiar cadenes: strcpy

Servix per a assignar una cadena a una variable de tipus cadena. És a dir, com que en C no podem fer directament coses com: **cadena1="hola"** o **cadena1=cadena2**, ho podem fer amb la funció strcpy. La sintaxi és:

```
strcpy (cad_desti,  cad_origen)
```

Esta funció copia la cadena passada com a 2n paràmetre (cad_origen) en la variable de tipus cadena passada com a 1r paràmetre (cad_desti), incloent el '\0'.

Exemple:

```
char s1[]=''bon dia'', s2[]=''bona nit'';
char d[20];
strcpy(d, s1);
printf(''%s\n'', d);    // Mostra ''bon dia''
strcpy(d, s2);
printf(''%s\n'', d);    // Mostra ''bona nit''
```

Notes:

- ✓ Esta funció no comprova si la cadena a copiar cap o no en la variable de destí. És feina del programador. Potser no passe res, però potser sobreescrivim en posicions de memòria assignades a altres variables.

- ✓ La funció també retorna la cadena copiada. Podem fer, per exemple:

```
printf(''%s\n'', strcpy(d, s1));       // Això mostra ''bon dia''
```

## 4.5 Concatenar cadenes: *strcat*

Servix per a afegir una cadena al final d'una altra.

```
strcat (cad_desti, cad_origen)
```

Esta funció copia la cadena passada com a 2n paràmetre en la variable de tipus cadena passada com a segon paràmetre PERÒ a partir del '\0' (sobreescrivint-lo).

Exemple:

```
char cog1[]=''Garcia'', cog2[]=''Burguera'';
char cognoms[50];
strcpy(cognoms, cog1);
strcat(cognoms, '' i '');
strcat(cognoms, cog2);
printf(''%s\n'', cognoms);    // Mostra ''Garcia i Burguera''
```

Notes: les mateixes que en la funció *strcpy*.

## 4.6 Saber la longitud d'una cadena: *strlen*

Com hem vist, tant per a *strcpy* i *strcat* és important saber si la cadena origen cabrà en la cadena destinació. Per tant, cal saber la longitud d'una cadena. Per a això servix la funció *strlen*:

```
strlen (cadena)
```

Esta funció retorna la longitud en caràcters de la cadena que es passa com a paràmetre, sense comptar el '\0'.

Exemple:

```
char s1[]=''hola'';
printf(''%s %d\n'', s1, strlen(s1));          // hola 4
printf(''%d\n'', strlen(''Miquel''));         // 6
```

## Exercicis

8. Suposem declarat un vector char s[41]. Indica l'efecte de les següents instruccions realitzades sobre el vector. Si alguna és errònia indica el motiu.

    a) s=''Bona vesprada'';
    b) s[0]='\0';
    c) if (s=='''') printf(''Cadena buida\n'');
    d) if (strcmp(s, ''Bona nit'')) printf(''Igual o no?\n'');
    e) if (! strcmp(''Bona nit'', s)) printf(''Igual o no?\n'');
    f) strcat(s, ''Bon dia, vesprada, nit, matinada, ...'');
    g) strcpy(s, ''Bon dia'');
    h) strcat(s, '' i bona vesprada'');
    i) printf(''%d %d\n'', strlen(''Bon dia''), strlen(s));
    j) s[0]='\0' ;
    k) printf(''%d\n'', strlen(s));

9. Programa que llija una frase (màxim 255 caràcters) i mostre cada paraula en una línia, indicant la longitud de cada paraula. Nota: cal fer servir un vector auxiliar per a anar guardant les paraules.

10. Imagina que no existira la funció strlen. Fes un programa que llija una cadena de caràcters per teclat i mostre la longitud. Fes altres 3 programes per a simular les funcions strcmp, strcpy i strcat.

11. Una empresa discogràfica realitza una enquesta sobre les cançons d'actualitat. Disposa d'un total de 10 èxits, els quals es poden votar. Cada persona enquestada vota pels seus 5 èxits favorits. El programa haurà de fer el següent:

    a) Enquestar a diverses persones fins dir que no en volem més.
    b) Cada persona dirà quins 5 d'eixos 10 són els seus favorits. Al primer se li sumen 5 punts, al segon 4, i així successivament, al cinqué, 1 punt.
    c) Es mostrarà una llista amb el número de l'èxit juntament amb els punts (de major a menor puntuació). No es trauran aquells que no tinguen vots.

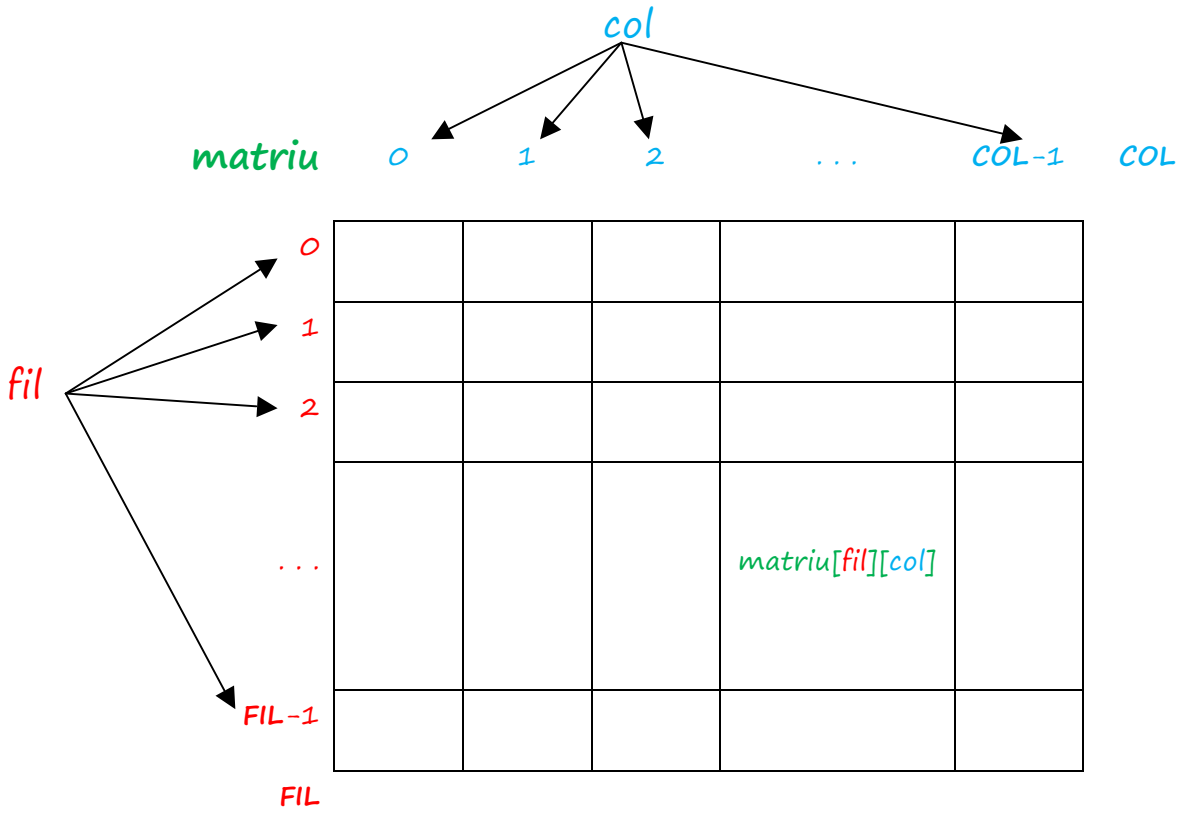
# 5 Matrius

Són com els vectors que hem vist, però en 2 dimensions (anomenats arrays bidimensionals). També poden ser de 3 dimensions (arrays tridimensionals), de 4, etc.

Definició de matriu bidimensional: conjunt finit d'elements del mateix tipus, als quals podem accedir mitjançant **dos índexs** i el nom de la matriu.

Exemple:

```
#define FIL 20        // Quantitat de files
#define COL 10        // Quantitat de columnes
...
float matriu[FIL][COL];    // Per a guardar FIL*COL valors decimals
int fil;        // índex per a les files
int col;      // índex per a les columnes
```




Altra definició de matriu bidimensional: és un vector (unidimensional) on cadascun dels seus elements és un altre vector (unidimensional).

## 5.1 Declaració de matrius i inicialització

La declaració d'una matriu és:

```
tipus_dades  nom_matriu[q_files][q_columnes]  = {llista_de_valors}  ;
```

La inicialització és opcional

Podem vore que, igual que els vectors, en la mateixa declaració poden inicialitzar-se les matrius. Per exemple:

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

... equival a

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

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |

## 5.2 Ús de matrius

S'utilitza per a guardar molta informació del mateix tipus de dades sobre un mateix objecte, sempre que cadascuna d'eixa informació es puga identificar per dos índexs.

En el context dels jocs, les matrius són molt aprofitables per a representar elements bidimensionals, com poden ser taulers d'escacs, cartrons de bingo, 4 en ratlla, enfonsar vaixells, etc.

Una imatge .bmp o .jpg es guarda internament com a tres matrius. Cadascuna d'elles representa un color (RGB). Cada element d'eixa matriu (i,j) representa un píxel del pla de la imatge.

Per a treballar amb estes imatges ho podem fer fàcilment:

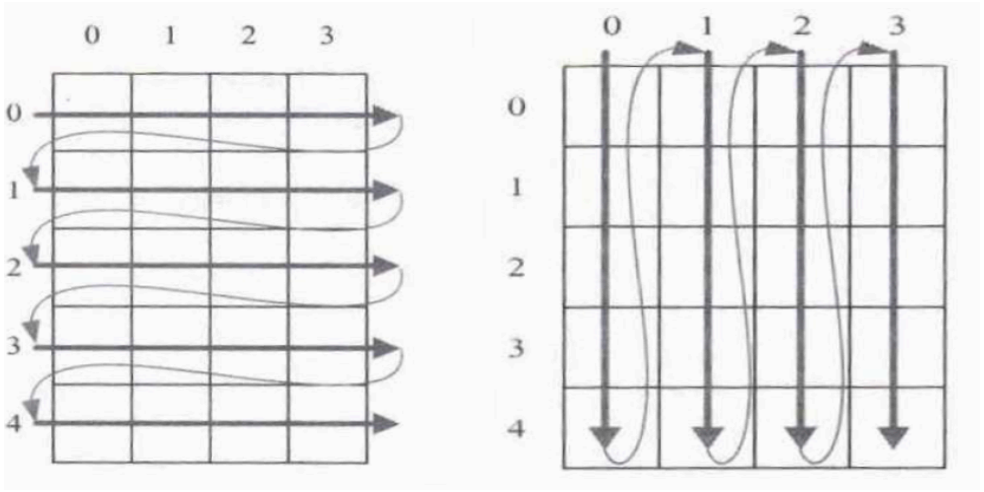
```
roig[150][300] = 10;
```

Amb eixa instrucció estaríem assignant la tonalitat de roig número 10 al píxel de la fila 150 i columna 300.

## 5.3 Recorreguts d'una matriu

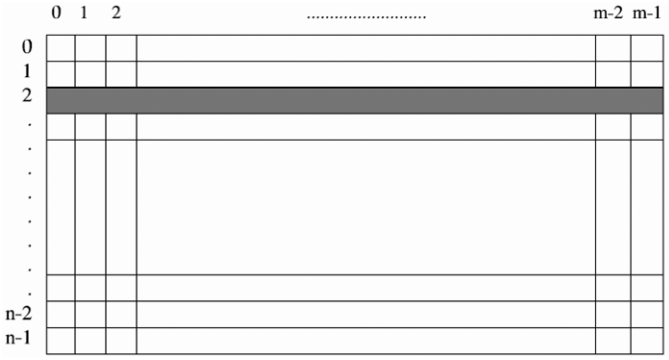
Si volem llegir els elements d'una matriu, o inicialitzar-la, etc, haurem de recórrer-la tota. És a dir, haurem de passar per cadascun dels seus elements. Podem fer eixe recorregut de 2 maneres: per files i per columnes.

| RECORREGUT D'UNA MATRIU BIDIMENSIONAL | |
|---|---|
| PER FILES | PER COLUMNES |
| for (fil = 0 ; fil < **FIL** ; fil++) {<br>  for (col = 0 ; col < **COL** ; col++) {<br>  scanf("%f", &matriu[fil][col]);<br>  }<br>} | for (col = 0 ; col < **COL** ; col++) {<br>  for (fil = 0 ; fil < **FIL** ; fil++) {<br>  scanf("%f", & matriu[fil][col]);<br>  }<br>} |



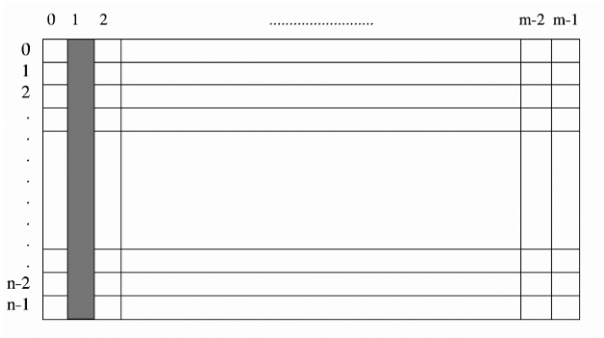

Però si només volem recórrer una fila, per exemple la 2, farem:

```
for (col = 0 ; col < COL ; col++) {
    scanf("%f", & matriu[2][col]);
}
```




I si volem recórrer només una columna, per exemple la 1, farem:

```
for (fil = 0 ; fil < FIL ; fil++) {
    scanf("%f", & matriu[fil][1]);
}
```



## Exercici resolt

12. A un centre meteorològic volem guardar les temperatures d'un mes (30 dies), mesurades 4 voltes al dia (a les 0, 6, 12 i 18 hores de cada dia).

Per a això es guardarà la informació en una matriu. El programa demanarà el dia del qual volem introduir les temperatures i, a continuació demanarà les 4 temperatures corresponents.

Este procés anirà repetint-se fins que introduïm el dia 0. Posteriorment, el programa presentarà un menú perquè l'usuari faça una anàlisi de les dades recol·lectades:

1. Temperatura mitja d'un dia demanat per programa.
2. Temperatura mitja d'una hora demanada per programa.
3. Temperatura mitja del mes.
0. Eixir.

Solució:

```
// A la matriu temperatures anem a guardar 4 temperatures per a cada dia del mes.
// Les temperatures s'han mesurat a les 0, 6, 12 i 18 hores de cada dia.

// INCLUSIO DE LLIBRERIES
#include <stdio.h>
#include <stdlib.h>  // Per a la funcio system()

// DEFINICIO DE CONSTANS
#define DIES 3
#define FRANGES 4

int main(){

// DECLARACIO DE VARIABLES
int i, j, temperatures[DIES+1][FRANGES];  // Un lloc més per als dies per a no usar el dia 0
int dia, hora, franja, opcio;
float total = 0, mitjana;
system("cls");
// RECORREM LA MATRIU PER A INICIALITZAR-LA A 0
```

```
for (i=1; i<=DIES; i++) {
    for (j=0; j<FRANGES; j++)     {
        temperatures[i][j]=0;
    }
}
printf("Array net\n");

// RECORREM LA MATRIU PER A MOSTRAR QUE ESTA INICIALITZADA
for (i=1; i<=DIES; i++) {
    for (j=0; j<FRANGES; j++)     {
        printf("%d ", temperatures[i][j]);
    }
    printf("\n");
}

// INTRODUCCIO DE DADES:
while (1)
{
     // DEMANEM UN DIA CORRECTE (ENTRE 0 I 30)
     do
     {
        printf("Dona'm el dia entre 1 i %i (0 per a acabar): ", DIES);
        scanf("%d", &dia);
        fflush(stdin);
     } while ((dia <0) || (dia > DIES));


     // SI HEM POSAT EL DIA 0, ACABEM:
     if (dia ==0) {
               break;
     }

     // DEMANEM LES TEMPERATURES DEL DIA ESCOLLIT:
     for (j=0; j<FRANGES; j++) {
         printf("Dona'm la temperatura de les %d:", j*6);
         scanf("%d", &temperatures[dia][j]);
         fflush(stdin);
     }
}

// QUAN JA NO VOLEM INTRODUIR MeS DADES (HEM POSAT DIA 0)
```

```
// PROCEDIM A MOSTRAR LES TEMPERATURES QUE HEM POSAT:

for (i=1; i<=DIES; i++) {
    for (j=0; j<FRANGES; j++) {
        printf("%d ", temperatures[i][j]);
    }
    printf("\n");
}

// MENU DE COSES A FER SOBRE LES DADES POSADES:
do {
     // MOSTREM MENU I DEMANEM UNA OPCIO CORRECTA
     do {
         system("cls");
         printf("**********************************\n");
         printf("1.-Mostrar temp.mitjana d'un dia\n");
         printf("2.-Mostrar temp.mitjana d'una hora\n");
         printf("3.-Mostrar temp.mitjana del mes\n");
         printf("0.-Eixir\n");
         printf("**********************************\n");
         printf("Tria opcio: "); scanf("%d", &opcio);
         fflush(stdin);
     } while ((opcio<0) || (opcio>3));

     // EXECUTEM L'OPCIO ESCOLLIDA
     switch (opcio)
     {
          case 1:    // MITJANA DE LES TEMPERATURES D'UN DIA
               do {
                   printf("De quin dia vols calcular la mitjana:");
                   scanf("%d", &dia);
                   fflush(stdin);
                   } while ((dia<=0) || (dia>DIES));
               total = 0;
               for (j=0; j<FRANGES; j++)
                   total = total + temperatures[dia][j];
               mitjana = total / FRANGES;
               printf("La mitjana del dia %d es %5.2f\n", dia, mitjana);
               system("sleep 2");
               break;
```

```
            case 2:     // MITJANA DE LES TEMPERATURES D'UNA HORA
                do {
                    printf("De quina franja vols calcular la mitjana (0 a 3):");
                    scanf("%d", &franja);
                    fflush(stdin);
                    } while ((franja<0) || (franja>3));
                total = 0;
                for (i=1; i<=DIES; i++)
                    total = total + temperatures[i][franja];
                mitjana = total / DIES;
                printf("La mitjana de la franja %d es %5.2f\n", franja, mitjana);
                system("sleep 2");
                break;

            case 3:     // MITJANA DE TOT EL MES
                total = 0;
                for (i=1; i<=DIES; i++)
                    for (j=0; j<FRANGES; j++)
                        total += temperatures[i][j];
                mitjana = total / (DIES * FRANGES);
                printf("La mitjana del mes es %5.2f\n", mitjana);
                system("sleep 2");
                break;

            case 0:     // EIXIR
                printf("Adeu!");
                break;

            default:
               printf(‘‘Opció incorrecta’’);

        } // FINAL DEL SWITCH

    } while (opcio != 0);       // FINAL DEL BUCLE QUE REPETIX EL MENU

}   // FINAL DEL PROGRAMA
```

## Exercici

13. Modifica l'exercici anterior del centre meteorològic afegint estes opcions al menú:

    4. Mostrar temperatura en un dia i hora en concret.

    5. Mostrar la mínima i la màxima d'un dia.

    6. Canviar una temperatura.

Pensa altres possibles opcions que podríes fer i implementa-les.

## 5.4 Matrius de cadenes

És un cas particular de les matrius normal i corrents, només que, si volem guardar per exemple un llistat de noms, ho podem fer amb una matriu:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| P | e | p | \0 | | | | | | |
| M | a | n | o | l | o | \0 | | | |
| M | a | r | i | a | \0 | | | | |
| B | e | r | n | a | t | \0 | | | |
| M | a | r | t | a | \0 | | | | |

Per a declarar eixa matriu farem:

```
char noms[5][10] = {"Pep", "Manolo", "Maria", "Bernat", "Marta"};
```

Què haguera passat si la declaració haguera segut esta?

```
char noms[][] = {"Pep", "Manolo", "Maria", "Bernat", "Marta"};
```

Que la reserva d'espai en memòria seria:

| | | | | | | |
|---|---|---|---|---|---|---|
| P | e | p | \0 | | | |
| M | a | n | o | l | o | \0 |
| M | a | r | i | a | \0 | |
| B | e | r | n | a | t | \0 |
| M | a | r | t | a | \0 | |

### Exercicis

14. Fes un programa que guarde en una matriu les notes de les 3 avaluacions de 15 alumnes de FPR. A continuació trau de cada alumne: el número, la nota mitja i quantes suspeses. També trau de cada avaluació: el número, la nota màxima i la mínima.

15. Modifica el programa anterior perquè es puguen guardar, en la mateixa matriu (ara serà tridimensional), les notes de FPR, SIM i XAL. Mostra quantes avaluacions han suspés en total tots els alumnes.

16. Fes un programa que, a partir d'una posició donada en un tauler d'escacs (x,y) i de quina peça hi ha allí (cavall, alfil...), mostre totes les possibles posicions (x,y) on pot anar eixa peça (tenint en compte que només està eixa peça al tauler).