LLENGUATGES DE PROGRAMACIÓ

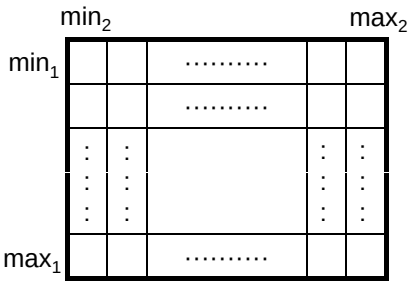
BLOC 1 – SEMINARI 3

TIPUS DE DADES COMPOSTOS: ARRAYS, CADENES DE CARÀCTERS I REGISTRES

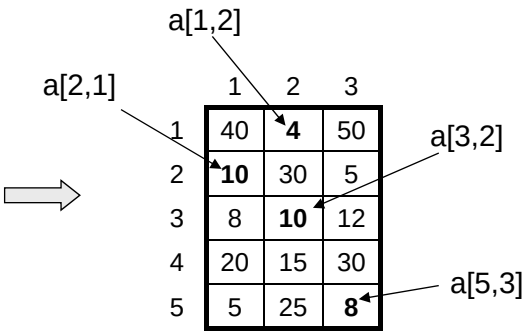
Arrays multidimensionals

Declaració de taules multidimensionals en pseudocodi

tipus
    <nom_tipus_taula> : **taula** [min_1 .. max_1, min_2 .. max_2, ... , min_N .. max_N] **de** <tipus>
fitipus



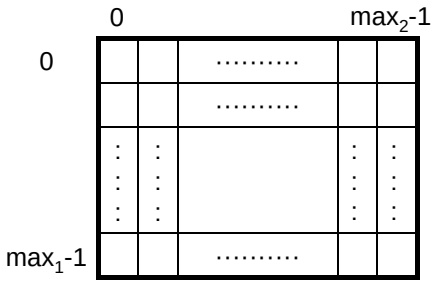

```
tipus
    matriu : taula [1 .. 5, 1 .. 3] de enter
fitipus
var
   a: matriu
fivar
```



Arrays multidimensionals

Declaració d'arrays multidimensionals en llenguatge C

<tipus> <nom_array> [max_1] [max_2] ..... [max_N];

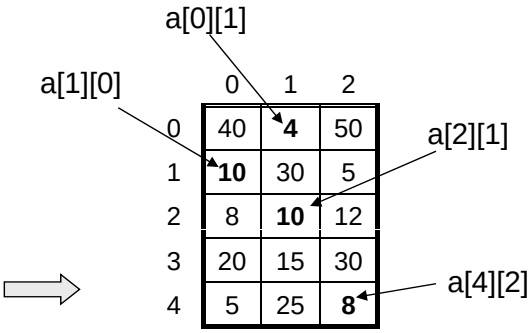


PSEUDOCODI

```
tipus
    matriu : taula [1 .. 5, 1 .. 3] de enter
fitipus
var
   a: matriu
fivar
```

LLENGUATGE C

```
int a[5][3];
```



Inicialització d'arrays multidimensionals

Accés individual als elements de l'array (cas d'un array bidimensional)

PSEUDOCODI

<nom_taula> [<fila> , <columna>]

LLENGUATGE C

<nom_array> [<fila>][<columna>]

Exemple:

a[4,3] ← 30

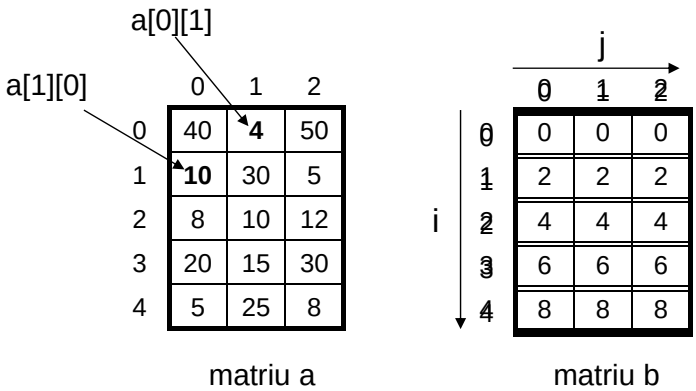
a[3][2] = 30;

Inicialització en la declaració

```
int a[5][3] = { {40,4,50} , {10,30,5} , {8,10,12} , {20,15,30} , {5,25,8} };
```

Inicialització element a element

```
int b[5][3];
int i,j;

for (i=0;i<5;i++)
    for (j=0;j<3;j++)
        b[i][j]=2*i;
```



Exemple: array bidimensional

PSEUDOCODI

```
algorisme LlegirMatriu
   const
      N_FILES ← 5
      N_COL ← 3
   ficonst
   tipus
      matriu: taula
              [1 .. N_FILES, 1 .. N_COL] de enter
   fitipus
   var
      a: matriu
      i, j: enter
   fivar

/* Llegir la matriu  */
   per i ∈ [1 .. N_FILES]
      per j ∈ [1 .. N_COL]
         llegir (a[i,j])
      fiper
   fiper
fialgorisme
```

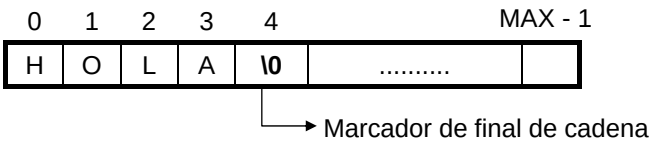
LLENGUATGE C

```
/* LlegirMatriu.c : programa que permet
l'entrada dels elements d'una matriu pel teclat */
#include <stdio.h>

#define N_FILES 5
#define N_COL 3

void main ()
{
int a[N_FILES][N_COL];
int i,j;

/* Llegir la matriu  */
   for (i=0;i<N_FILES;i++)
      for (j=0;j<N_COL;j++)
         scanf ("%d", &a[i][j]);
}
```

Cadenes de caràcters

Definició i declaració

cadena de caràcters = array de char's



PSEUDOCODI

```
const
   MAX=<valor>
ficonst
tipus
   cadena : taula [1 .. MAX] de caràcter
fitipus
var
   <nom> : cadena
fivar
```

LLENGUATGE C

```
#define MAX <valor>

char <nom> [MAX];
```

Exemple:

```
#define MAX 50

char Nom_Alumne[MAX];
```

Inicialització de cadenes de caràcters

Inicialització en la declaració

```
#define MAX 50

char Nom_Alumne1[MAX]={'P','a','u','\0'};
char Nom_Alumne2[MAX]="Pau";
```

Inicialització element a element

```
char Nom_Alumne[MAX];

Nom_Alumne[0]='P';
Nom_Alumne[1]='a';
Nom_Alumne[2]='u';
Nom_Alumne[3]='\0'};
```

Inicialització amb la funció strcpy

```
char Nom_Alumne1[MAX];
char Nom_Alumne2[MAX];

strcpy (Nom_Alumne1,"Pau");
strcpy (Nom_Alumne2,Nom_Alumne1);
```

Entrada / Sortida de cadenes

```
char cadena[30];
```

Entrada

```
scanf ("%s", &cadena);
gets (cadena);
```

Sortida

```
printf ("%s",cadena);
puts (cadena);
```

Funcions de tractaments de cadenes: <string.h>

Funció	Significat
strcpy(<cadena_destí>,<cadena_origen>) int strlen(<cadena>)	Fa la còpia d'una cadena sobre una altra. Retorna la longitud de la cadena.
strcat(<cadena_destí>,<cadena_origen>)	Concatena la segona cadena al final de la primera.
int strcmp(<cadena_1>,<cadena_2>)	Compara les dues cadenes. Retorna 0 si són iguals.
char *strchr(<cadena>, <carácter>)	Busca un caràcter dins d'una cadena.
char *strrchr(<cadena>, <carácter>)	Busca un caràcter dins d'una cadena (comença per final).
char *strstr(<cadena>,<subcadena>)	Busca una subcadena dins d'una altra cadena.
char *strpbrk(<cadena>,<subcadena>)	Busca la primera aparició de qualsevol caràcter d'una subcadena dins d'una altra cadena.
double strtod(<cadena>)	Converteix una cadena de caràcters a un real.
long strtol(<cadena>)	Converteix una cadena de caràcters a un long.
unsigned long strtoul(<cadena>)	Converteix una cadena de caràcters a un unsigned long.

Registres

Registre: Conjunt finit d'elements etiquetats que poden ser de tipus diferent (estructura heterogènia)

alumne

dni	cadena
nom	cadena
cognoms	cadena
grup	enter
nota_examen	real
nota_parcial	real
nota_final	real

PSEUDOCODI

Declaració d'un registre

```
tipus
    TipusAlumne: registre (
        dni : cadena
        nom : cadena
        cognoms : cadena
        grup : enter
        nota_examen : real
        nota_parcial : real
        nota_final : real
    )
    fitipus
    var
       alumne1,alumne2 : TipusAlumne
    fivar
```

Registres

LLENGUATGE C

Registre → Definició de l'estructura de registre + Declaració de variables

Definició de l'estructura d'un registre

Sintaxi C

```
struct <etiqueta_registre>
{
<tipus_1> <nom_camp1>;
<tipus_2> <nom_camp2>;
...
<tipus_N> <nom_campN>;
};
```

Exemple

```
struct Alumne
{
char dni[9];
char nom[25];
char cognoms[50];
int grup;
float nota_examen;
float nota_parcial;
float nota_final;
};
```

Registres

LLENGUATGE C

Declaració de variables registre

a) A la definició

Sintaxi C

```
struct <etiqueta_registre>
{
<tipus_1> <nom_camp1>;
<tipus_2> <nom_camp2>;
...
<tipus_N> <nom_campN>;
} <nom_variable1>, … , <nom_variableN>;
```

Exemple

```
struct Alumne
{
char dni[9];
char nom[25];
char cognoms[50];
int grup;
float nota_examen;
float nota_parcial;
float nota_final;
} alumne1, alumne2;
```

Registres

LLENGUATGE C

Declaració de variables registre

b) Després de la definició

Sintaxi C

```
struct  <etiqueta_registre>
{
<tipus_1>  <nom_camp1>;
<tipus_2>  <nom_camp2>;
...
<tipus_N>  <nom_campN>;
};

struct  <etiqueta_registre> <nom_variable1>;
```

Exemple

```
struct Alumne
{
char dni[9];
char nom[25];
char cognoms[50];
int grup;
float nota_examen;
float nota_parcial;
float nota_final;
};

struct Alumne alumne1, alumne2;
```

Registres

Accés als elements d'un registre

Sintaxi C

```
<nom_variable> . <nom_camp>
```

Exemples

```
alumne1.dni
alumne1.grup

alumne2.nota_parcial
alumne2.nom
```

Registres

Inicialització dels registres

a) En el moment de la declaració

```
struct Alumne alumne1={"12345678W","Pau","Martí Mur",2,0,0,0};
```

b) Camp a camp després de la declaració

```
struct Alumne alumne1;

strcpy(alumne1.dni,"12345678W");
strcpy(alumne1. nom,"Pau");
strcpy(alumne1. cognoms,"Martí Mur");
alumne1. grup=2;
alumne1. nota_examen=0;
alumne1. nota_parcial=0;
alumne1. nota_final=0;
```

c) Fent una còpia d'un altre registre

```
struct Alumne alumne2;

alumne2=alumne1;
```

Definició de nous tipus

typedef

typedef → Permet crear tipus nous a partir de tipus ja existents

```
typedef   <tipus>   <nom_tipus_nou>
```

Exemples

```
typedef   char[25]   Nom_Persona;

int area;
Nom_persona nom;


typedef   float   Numero_Real;

float a;
Numero_Real b;
```

```
typedef struct Alumne
{
char dni[9];
char nom[25];
char cognoms[50];
int grup;
float nota_examen;
float nota_parcial;
float nota_final;
} info_Alumne;

info_Alumne alumne1, alumne2;
int num_alumnes;
```