Grau en Enginyeria en Informàtica

Programació I

Tipus de dades estructurades. Arrays

Objectius / competències

1. Comprendre la diferència entre tipus de dades simples i estructurades.
2. Conèixer els tipus de dades estructurades: array unidimensional i bidimensional.
3. Aprendre a usar arrays d'una i dues dimensions amb llenguatge C.

Índex

Recordatori: tipus de dades simples

- Totes les variables amb les quals hem treballat fins ara són de tipus simple
- Una variable de tipus simple només pot contenir un valor cada vegada
  - Exemple: si x és de tipus enter, podem assignar a x un únic valor cada vegada:

```
x= 7;
x = 10;
x= 2000;
…
```

Tipus de dades estructurades

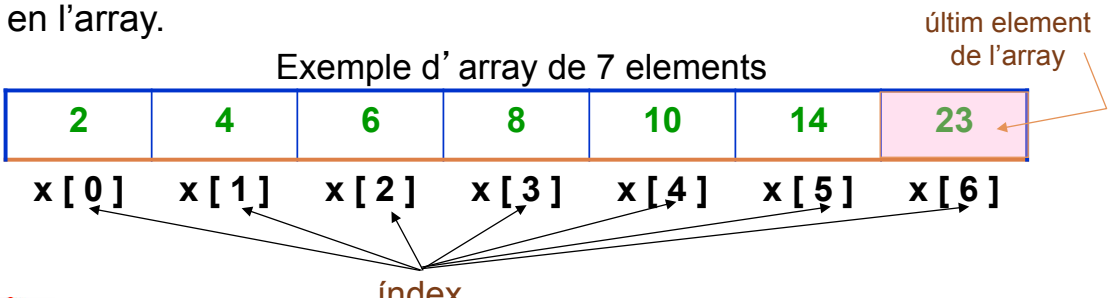
- Una variable de tipus estructurat consisteix en una col·lecció de dades de tipus simple
- Un tipus estructurat pot emmagatzemar mes d'un element (valor) alhora
  - Tipus array
    - Tots els elements que emmagatzema una variable de tipus array han de ser del mateix tipus
  - Tipus registre
    - Una variable de tipus registre pot emmagatzemar elements de diferent tipus
- Exemple: considerem una variable z que emmagatzemarà els números premiats en la Bonoloto. Per tant, emmagatzemarem 6 valors cada vegada

```
z =  (1, 4, 6, 24, 13, 2);
z = (3, 9, 12, 15, 23, 27);
```

En llenguatge C s'utilitza el tipus **struct** equivalent al tipus registre (*record*) d'altres llenguatges .

El tipus *array*

- És una estructura de dades en la qual s'emmagatzema una col·lecció de dades finita, homogènia i ordenada d'elements
  - Finita: ha de determinar-se quin serà el nombre màxim d'elements que es podran emmagatzemar en l'array
  - Homogènia: tots els elements han de ser del mateix tipus
  - Ordenada: es pot determinar quin és el n-èssim element de l'array
- Per a referir-se a un determinat element d'un array s'haurà d'utilitzar un **índex** (tancat entre claudàtors) que especifique la seua posició relativa en l'array.



En llenguatge C, el primer element de l'array es troba en la **posició** (índex) **zero**.

Exercicis

Donat l'array següent

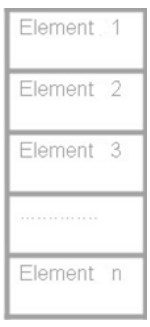
P= {pep, joan, lola, maria, rafa, carles, ferran},

indiqueu:

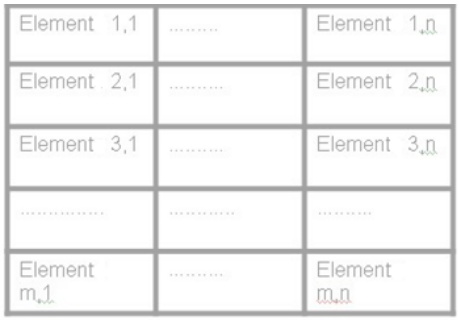
- P[3] = **maria**
- Com accediries a l'element 'ferran'? **7**
- Quants elements té ? **P[6]**

Classificació dels arrays

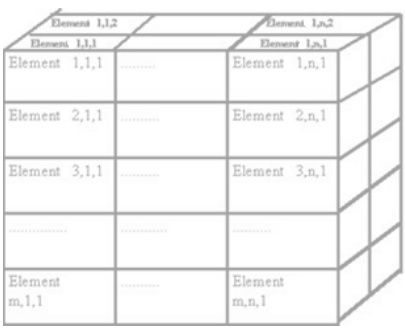
- Els arrays es classifiquen, segons el nombre de dimensions, en:
  - Unidimensionals (vectors)
  - Bidimensionals (matrius)
  - Multidimensionals, quan té tres dimensions o més
- La dimensió d'un array és el nombre d'índexs utilitzats per a referenciar un qualsevol dels seus elements



Array unidimensional



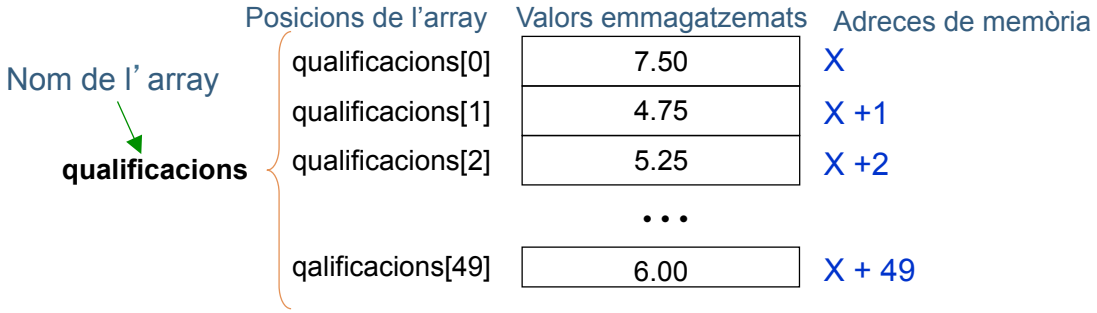
Array bidimensional



Array multidimensional

Arrays unidimensionals

- Un array d'una dimensió és un tipus de dades estructurat en què els elements s'emmagatzemen en posicions contigües de memòria, a cadascuna de les quals es pot accedir directament mitjançant un índex.
- Suposem que volem emmagatzemar la nota de l'examen de Programació 1 de 50 estudiants; per a fer-ho necessitarem:
  1. Reservar 50 posicions de memòria
  2. Donar un nom a l'array
  3. Associar una posició en l'array a cadascun dels 50 estudiants
  4. Assignar les puntuacions a cadascuna d'aquestes posicions



Declaració d' un array

Per a poder utilitzar una variable de tipus array (unidimensional) primer hem de declarar la

Sintaxi:

```
tipus_elements  nom_array [ nre_elem] ;
```

tipus_elements: indica el tipus de cada element de l'array; tots els elements són del mateix tipus

nom_array: indica el nom de l'array; pot ser qualsevol identificador vàlid

[nre_elem] : indica el nombre **màxim** d'elements de l'array; ha de ser un valor constant numèric enter

Exemple: float qualificacions[50];

Inicialització i accés a un array

- Igual que qualsevol altre tipus de variable, abans d'utilitzar un array hem d'inicialitzar el seu contingut.
- Una possible manera d'inicialitzar un array és accedint a cadascun dels seus components utilitzant un bucle i assignar-los un valor.
- Per accedir a una posició d'un array utilitzem la sintaxi següent :

```
nom_array [index] ;
```

Exemple d'accés a la qualificació de l'alumne que ocupa la posició 5 en l'array: qualificacions[4];

Utilitzar valors d'índexs **fora del rang** comprès per la mida de l'array provoca errors indesitjables en l'execució del nostre programa.

Exemple 1 d'inicialització d'un *array*

Si es coneixen els valors que prenen les components de l'array en definir-lo, podem definir i assignar valors simultàniament:

```
// exemple inicialització array
#include <iostream>
using namespace std;


main () {
      int vectorA [4] = {1, 5, 3, 9}
      int vectorB  [] = {1, 5, 3, 9};
      int vectorC [10] ={1, 5, 3, 9};
}
```

Pren el nombre de valors com a grandària del vector.

És possible inicialitzar de manera parcial l'array.

Exemple 2 d'inicialització d'un *array*

Podem inicialitzar un array fent que l'usuari introduïsca les dades per teclat, de la manera següent:

```
// exemple inicialització array
#include <iostream>
using namespace std;

void inicialitzar_Array(float qualificacions[ ]);

main () {
     float  qualificacions[50];

     inicialitzar_Array(qualificacions);
}

// procediment per inicialitzar el array
void inicialitzar_Array(float qualificacions[ ])
{
   int i;

   for ( i=0 ; i < 50 ; i++ ) {
      cout << "Introdueix la qualificació " << i << ":";
      cin >> qualificacions[i];
   }
}
```

En llenguatge C, el pas de paràmetres dels arrays sempre és **per referència.**

En llenguatge C, les funcions no poden retornar un tipus array. Per a modificar un array, ha de ser passat com a paràmetre.

Cerca lineal d'un element en un array

Si els elements de l'array NO estan ordenats

Per a buscar un element en un array podem utilitzar el que es denomina ***cerca lineal***

- Recorrem l'array des de la primera posició accedint a posicions consecutives fins a trobar l'element buscat.

```
// Cerca lineal d'un element
// funció per a buscar un element "elem" en un array amb TAM_MAX elements
// Torna la posició de "elem" en l'array si el troba o -1 si no el troba
int Cerca_Lineal(int nom_array[], int elem)
{
   int     pos;
   bool  trobat;

   pos = 0;
   trobat = false;
   // acabem la cerca si s'arriba al final de l'array o si s'ha trobat l'element
   while ( pos < TAM_MAX  &&   ! trobat) {
        if  (nom_array[pos] == elem)
             trobat = true;
        else
             pos = pos +1;
    }
    if (! trobat)
         pos = -1;

   return(pos);
}
```

Cerca binària d'un element en un *array*

14

Si els elements de l'array estan ORDENATS

Per a buscar un element en un array ordenat podem utilitzar el que es denomina ***cerca binària o dicotòmica***

- Reduïm la cerca dividint en meitats, de manera que es va delimitant l'interval de cerca depenent del valor que busquem.

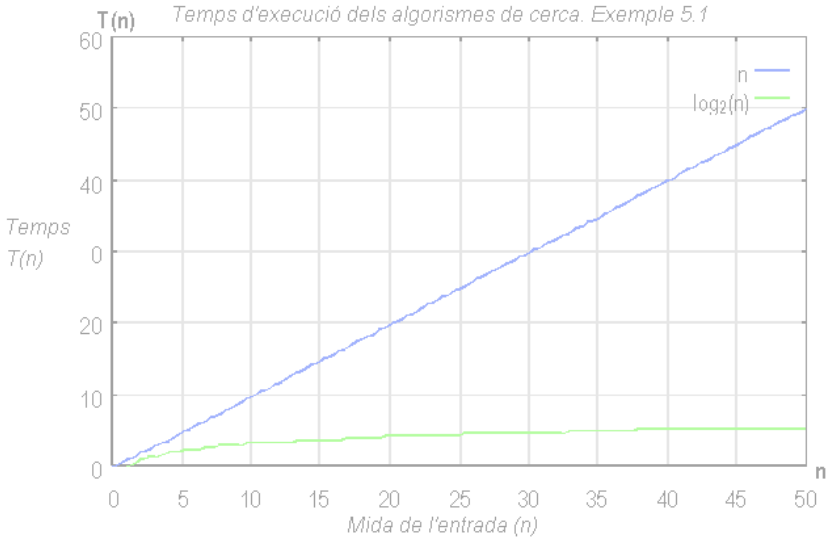
```
// Cerca binària d'un element en un array amb TAM_MAX elements ordenats de manera creixent
int Cerca nom_array[], int elem)  {
     int      pos_inici, pos_fin, pos_mitjana;
     bool  trobat;

     // [pos_inici, pos_fin] = interval actual de cerca
     pos_inici= 0;   // primera posició de l'array
     pos_fi n= TAM_MAX -1;   // última posició de l'array
     trobat= false;
     while ( pos_inici <= pos_fi   &&   ! trobat) {
          pos_mitjana = (pos_inici + pos_fin) / 2;   // posició intermèdia de l'array
          if  (elem == nom_array[pos_mitjana])  // element trobat en la posició pos_mitjana
               trobat = true;
          else if (elem > nom_array[pos_mitjana] )
               pos_inici = pos_mitjana +1;   // l'element s'ha de buscar en la meitat superior
          else
               pos_fin = pos_mitjana -1;   // l'element s'ha de buscar en la meitat inferior
     }
     if (! trobat)
          pos_mitjana = -1;

     return(pos_mitjana);
}
```

Cerca (cost temporal)

15

Cerca lineal: temps d'execució lineal

Cerca binària: temps d'execució logarísmic



Cadenes de caràcters

- Una cadena de caràcters (o ***string***) és una seqüència finita de caràcters consecutius
- Utilitzarem **arrays de caràcters**
- En un array de caràcters podrem emmagatzemar
  - Paraules
  - Frases
  - Noms de persona, noms de ciutats…
  - Codis alfanumèrics
  - Etc.

En llenguatge C++ existeix el tipus string, encara que en P1 no l'usarem.

Representació de cadenes de caràcters en C

- En llenguatge C, una cadena de caràcters s'escriu entre cometes dobles

"hola"

- En llenguatge C, totes les cadenes de caràcters han d'acabar amb el caràcter nul'\0', que ha d'emmagatzemar-se en l'array a continuació de l'últim caràcter de la cadena

'h'	'o'	'l'	'a'	'\0'					
0	1	2	3	4	5	6	7	8	9

- Exemple: la cadena "hola"
  - s'ha emmagatzemat en un array de caràcters de grandària 10
  - està formada per 4 caràcters (té longitud 4), però ocupa en l'array 5 caràcters (perquè s'emmagatzema també el caràcter '\0')

```
char cad[10]="hola";
```

Funcions de C per a fer servir cadenes de caràcters

Funció	Descripció	Us
cin.getline(*cadena*, *MIDA***)**	Lectura d'una cadena de caràcters per teclat fins a final de línia o la grandària màxima especificada pel paràmetre MIDA (enter positiu). La seqüència de caràcters llegida s'emmagatzema en el paràmetre cadena (array de caràcters, com a mínim, d'una grandària determinada pel paràmetre MIDA) .	Com a procediment
strcpy(*cadena_destinació, cadena_origen***)**	Còpia de cadenes. Copia el contingut de *cadena_origen* en *cadena_destinació*	Com a procediment
strcmp(*cadena1, cadena2***)**	Comparació alfabètica de cadenes Si *cadena1* < *cadena2* Aleshores torna un nombre < 0 Si *cadena1* == *cadena2* Aleshores torna 0 Si *cadena1* > *cadena2* Aleshores torna un nombre > 0	Com a funció
Strlen(*cadena***)**	Retorna un tipus int que indica la longitud de la cadena de caràcters especificada com a paràmetre, és a dir, el nombre de caràcters vàlids d'aquest array (fins al caràcter especial de finalització de cadena '\0', sense incloure aquest).	Com a funció

Exemple de cadena de caràcters

Funció que torna la longitud d'una cadena de caràcters

```
// torna la longitud d'una cadena de caràcters
// aquesta funció és equivalent a la predefinida en C strlen( )
int longitud_Cadena(char  cad[ ])
{
     int len;

     len = 0;
     while (cad[len] != '\0')
       len++;

     return(len);
}
```

Exemples d'arrays unidimensionals (I)

- Procediment que imprimeix per pantalla el contingut d'un array d'elements de tipus double.

```
// imprimeix per pantalla els elements d'un
// array de tipus double
void print_Array(double a[], int len)
{
     int i;
      for (i=0; i < len; i++)
          cout << "[" << i << "] = " << a[i] << endl;
}
```

- Funció que calcula la mitjana de notes d'alumnes.

```
// disposem de "len" notes de tipus float
float calcula_Mitjana(float a[], int len)
{
     int     i;
     float  suma;

     suma = 0.0;
     for (i=0; i < len; i++)
        suma = suma + a[i];

     // suposem len > 0
     return(suma / len);
}
```

Exemples d'arrays unidimensionals (II)

- Donat un array d'enters, retornar el major valor, el nombre d'ocurrències d'aquest valor, i la posició de la primera i última aparició en la qual es troba emmagatzemada

```
void Ocurrencies(int  v[ ], int &major, int &num_ocur, int &pos_pri, int &pos_ult)
{
  int i;

  maJor = v[0]; //  inicialment el nombre major serà el que està en la primera posició
  num_ocur = 1;
  pos_pri = 0;
  pos_ult = 0;

  // recórrer la taula: des de la segona posició fins la posició final (constant LMAX)
  for (i=1; i < LMAX; i++) {
    if (v[i] > major) {  // trobem un nou nombre major
       major = v[i];
       num_ocur = 1;
       pos_pri = i;
       pos_ult = i;
    }
    else if (v[i] == mayor) {
       // trobem una nova ocurrència del nombre major fins al moment
       num_ocur = num_ocur +1;
       pos_ult = i;
    }
  }
}
```

Exemples d'arrays unidimensionals (III)

- Donat un array d'enters, moveu tots els seus elements una posició a la dreta. El desplaçament serà circular, és a dir, l'últim element passarà a ser el primer

```
void moure_En_Circular (int  v[ ])
{
   int i, ult;

   // guardar el valor de l'última posició de la taula
   ult = v[LMAX-1];

   // moure tots els elements una posició a la dreta, excepte l'últim
   for (i=LMAX-1; i > 0; i--)
      v[i] = v[i-1];

   // actualitzar la primera posició amb el valor que teníem en la última
   v[0] = ult;
}
```

Algorismes d'ordenació d'arrays

- Es interesant i habitual l'operació de ordenació en un array
  - Exemple: mantenir ordenat el nostre vector de qualificacions per a poder consultar ràpidament les cinc millors notes. Per a fer-ho hauríem d'ordenar el nostre vector de major a menor (en ordre decreixent) i accedir a les cinc primeres posicions del vector.
- Hi ha molts algorismes per ordenar els elements d'un array. Un algorisme eficient per a ordenar els elements d'un array és l'algorisme d'inserció directa:
  - Cada element es compara amb els que estan a la seua esquerra i s'insereix en la seua posició adequada
    - la posició adequada s'obté quan es troba un element major (si l'ordre és decreixent) que el pretenem inserir o quan s'aconsegueix l'extrem esquerre de l'array.
  - En la cerca de la posició adequada, cada element menor (si l'ordre és decreixent) es desplaça una posició a la dreta.

Exemple d'ordenació d'arrays (I)

L'ordenació per inserció directa pot comparar-se amb l'ordenació d'una mà de cartes

- Cada vegada que agafem una carta la inserim en la seua posició correcta entre les que ja tenim ordenades a la mà.

- La inserció divideix l'array en dues parts:
  - La primera (que representa les cartes que tenim a la mà) està ordenada i creix en grandària a mesura que avança l'ordenació.
  - La segona (que representa les cartes que anem afegint a les que hi ha a la taula) està sense ordenar, i conté els elements que anem a anar inserint en la primera part de l'array. Aquesta segona part va decreixent a mesura que avança l'ordenació.

Traça d'ordenació d'array per inserció directa

Array inicial

Part ordenada

Part desordenada

Array ordenat

84	**69**	**76**	**86**	**94**	**91**
84	**69**	**76**	**86**	**94**	**91**
84	**69**	**76**	**86**	**94**	**91**
84	**76**	**69**	**86**	**94**	**91**
86	**84**	**76**	**69**	**94**	**91**
94	**86**	**84**	**76**	**69**	**91**
94	**91**	**86**	**84**	**76**	**69**

Implementació de l'algorisme d'inserció directa


```
// exemple ordenació array en ordre DECREIXENT amb el mètode d'INSERCIÓ DIRECTA
void Ordenar_Array( int elem[ ], int num_elem)
{
    int     esq;    // posició a la esquerra de l'element inserint en la part ordenada
    int     dta;    // posició del primer element de la part actualment desordenada
    int     actual; // primer element de la part actualment desordenada que s'ha d'inserir
    bool    troba_lloc;

    // Inicialment la part ordenada (part esquerra) està formada només per la primera posició ,
    // per tant, comencem a buscar el primer element de la part desordenada (part dreta)
    // des de la segona posició (índex igual a 1)
   for (der = 1; der < num_elem; der++)  {
        actual = elem[der];
        esq = der -1;
        troba_lloc = false;
        while ( izq >= 0   &&   ! troba lloc) {
            if  (actual > elem[esq]) {
                elem[esq+1] = elem[esq];  // els elements menors es desplacen cap a la dreta
                esq = esq -1;
            }
            else
                troba_lloc = true;
        }
        // inserim el primer element que estava en la part desordenada en el lloc
        // adequat dins de la part actualment ordenada
        elem[esq+1] = actual;
    }
}
```

Un altre algorisme d'ordenació d'arrays

- Un altre mètode d'ordenació d'arrays és l'algorisme de selecció directa:
  - Pas 1: buscar i seleccionar d'entre tots els elements que encara no estiguen ordenats el major de tots (si es un ordre decreixent).
  - Pas 2: intercanviar les posicions d'aquest element amb el que hi ha en l'extrem esquerre dels desordenats.

Traça d'ordenació d'array per selecció directa

Array inicial

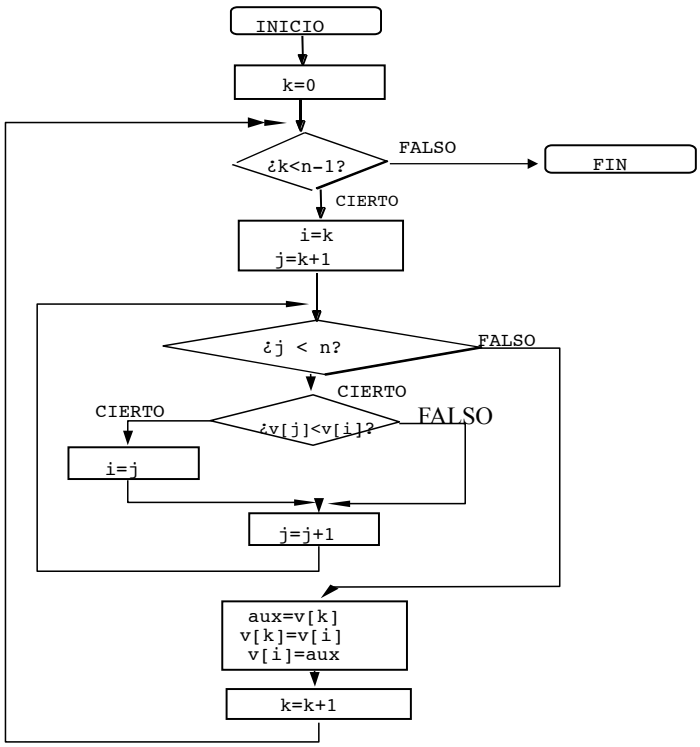
Part ordenada

Part desordenada

Array ordenat

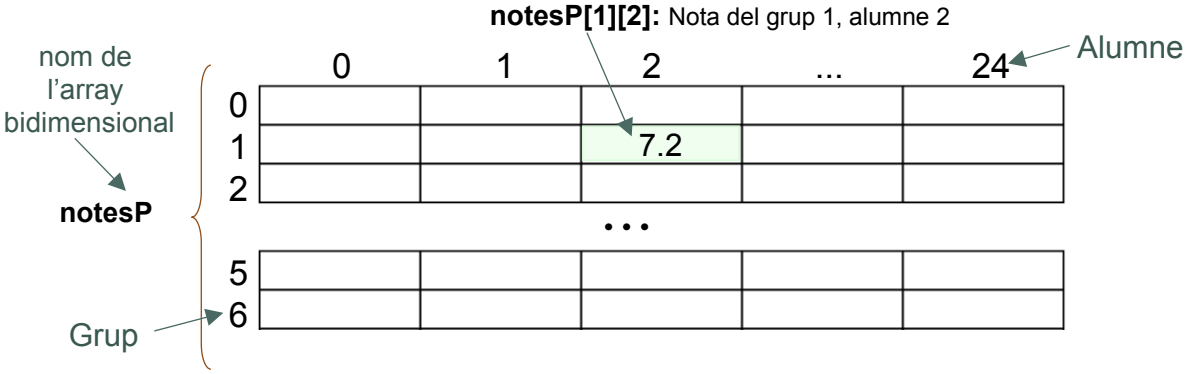
84	69	76	86	94	91
84	69	76	86	94	91
94	69	76	86	84	91
94	91	76	86	84	69
94	91	86	76	84	69
94	91	86	84	76	69
94	91	86	84	76	69

Algorisme de selecció directa (ordre creixent)



Arrays bidimensionals (matrius)

- Es necessiten 2 índexs per accedir a un qualsevol dels seus elements.
- Suposem que volem emmagatzemar la nota de l'examen de 7 grups de P1, cadascun dels quals té 25 alumnes.



Declaració d'un *array* bidimiensional

- Per a poder utilitzar una variable de tipus *array* bidimensional, primer hem de declarar la
- Sintaxi:

```
tipus  nom_array [n_elemF] [n_elemC] ;
```

- tipus: indica el tipus de cada element del array; tots els elements de l'array són del mateix tipus
- nom_array: indica el nom de l'array
- [n_elemF] : indica el nombre de "files" del array (primera dimensió)
- [n_elemC] : indica el nombre de "columnes" de l'array (segona dimensió)

Inicialització i accés a un array bidimensional

- Una possible manera d'inicialitzar un array bidimensional és accedint a cadascun dels seus components utilitzant dos bucles (un per a cada dimensió) i assignar-los un valor
- Per a accedir a una posició d'un array bidimensional utilitzem aquesta sintaxi:

```
nom [índexF] [índexC] ;
```

  - nom: indica el nom de l'array
  - [índexF] : indica la posició de la primera dimensió (**fila**) de l'array a al qual volem accedir; ha de ser sempre un valor comprès entre 0..índexF-1
  - [índexC] : indica la posició de la segona dimensió (**columna**) de l'array al qual volem accedir; ha de ser sempre un valor comprès entre 0..índexC-1
- Exemples:
  - notasFP [6] [24]; *// denota la nota de l'alumne 24 del grup 6*
  - notasFP [6]; *// denota totes les notes del grup6 (array unidimensional associat a la fila 6*

Exemple 1 d'inicialització d'un array bidimensional

- Si es coneixen els valors, es pot inicialitzar d'aquesta manera:

```
// exemple inicialització d'array bidimensional
#include <iostream>
using namespace std;

const int  N_FILAS = 4;
const int  N_COL = 2;

main () {
     float  matDD [N_FILAS][N_COL] = { {3.6, 6.7},
                                       {2.9, 7.6},
                                       {8.9, 9.3},

                                       {1.9, 0.2},
                                     };

      int  mat [][N_COL] = { {3, 6},
                             {9, 7},
                             {8, 3},
                             {1, 0},
                           };
}
```

En llenguatge C no és necessari especificar la grandària de la primera dimensió d'un array.

Exemple 2 d'inicialització d'un *array* bidimensional

- Podem inicialitzar un array fent que l'usuari introduïsca les dades per teclat, de la manera següent:

```
// exemple inicialització d'array bidimensional
#include <iostream>
using namespace std;

const int  N_FILAS = 10;
const int  N_COLUMNAS = 20;
void inicialitzar_Matriu(float matriu[ ][N_COLUMNES]);

main () {
     float  matriu[N_FILAS][N_COLUMNAS];

     inicialitzar_Matriu(matriu);
}

// procediment  per inicialitzar la matriu
void inicialitzar_Matriu(float matriz[ ][N_COLUMNAS])
{
     int  i, j;

       for ( i=0 ; i < N_FILAS ; i++ ) {
             cout << "Fila " << i << ":" << endl;
          for ( j=0; j < N_COLUMNES; j++) {
               cout << "columna " << j << ":";
                cin >> matriu[i][j];
          }
       }
}
```

En llenguatge C no és necessari especificar la grandària de la primera dimensió d'un array.

Exemples d'arrays bidimensionals (I)

- Donats 25 alumnes, dels quals es coneixen les notes de 7 assignatures, calcula la nota mitjana de les assignatures per a cadascun dels alumnes i imprimeix-les per pantalla

```
#include <iostream>
using namespace std;

const int  N_ALUMNES = 25;
const int N_ASIGNATURES = 7;
void  imprimeix_Mitjana_Alumnes(float notes[ ] [N_ASSIGNATURES]);

main () {
  float notes [N_ALUMNES][N_ASSIGNATURES];

  imprimeix_Mitjana_Alumnes (notes);
}

// calcula la mitjana de notes per a cada alumne i les imprimeix per pantalla
void imprime_Media_Alumnes(float notes[ ] [N_ASSIGNATURES]) {
  int i;

  for (int i=0; i< N_ALUMNES; i++)
    cout << "L'alumne " << i << " té de mitjana "<< calcula_Mitjana(notes[i], N_ASSIGNATURES) << endl;
}
```

utilitzem la funció d'un dels exemples anteriors

Exemples d'arrays bidimensionals (II)

- Donada una matriu quadrada d'enters, imprimeix en l'ordre següent els elements de la diagonal, els elements del triangle superior (per damunt de la diagonal) i els de el triangle inferior (per sota de la diagonal), tot això amb un recorregut per files i columnes.

```
// Versió que recorre tres vegades la matriu
void Imprimeix_Matriu_3 (int  matriu[ ][LMAX])
{   int i, j;

   //  Imprimeix diagonal
   for (i=0; i < LMAX; i++)  // recórrer files
      for (j=0; j < LMAX; j++) // recórrer columnes
         if (i == j)
            cout << matriz[i][j];
   //  Imprimeix triangle superior
   for (i=0; i < LMAX; i++)
      for (j=0; j < LMAX; j++)
         if (j > i)
            cout << matriz[i][j];
   //  Imprimeix triangle inferior
   for (i=0; i < LMAX; i++)
      for (j=0; j < LMAX; j++)
         if (j < i)
            cout << matriz[i][j];
}
```

```
// Versió que recorre una sola vegada  la matriu
void Imprimeix_Matriu_1(int  matriu[][LMAX])
{   int i, j;

   //  Imprimir diagonal
   for (i=0; i < LMAX; i++)   // recórrer files
      cout << matriu[i][i];

   //  Imprimeix triangle superior
   for (i=0; i < LMAX-1; i++)
      for (j=i+1; j < LMAX; j++)
         cout << matriu[i][j];

   //  Imprimeix triangle inferior
   for (i=1; i < LMAX; i++)
      for (j=0; j < i; j++)
         cout << matriu[i][j];
}
```

Tipus de dades definides pel programador en C

- S'utilitza la paraula reservada typedef perquè l'usuari definisca tipus de dades estructurades (tals com arrays i structs).
- És útil crear nous tipus de dades per millorar la llegibilitat dels programes.

```
// Definició de tipus de dades
typedef   int     Tenters[20];
typedef   float   Tnotes[50];
typedef   char    TCadena[30];
typedef   int     TMatriu[3][3];

// Declaració de variables
Tnotes     notesFP1, notesFP2;
TCadena    nomAlumne1, nomAlumne2;
TMatriu    matriu1, matriu2;
```

```
// Si no definim tipus de dades,
// la declaració de variables de tipus
// array seria:
float   notesFP1[50];
float   notesFP2[50]
char    nomAlumne1[30];
char    nomAlumne2[30];
int     matriu1[3][3];
int     matriu2[3][3];
```

Bibliografía Recomendada

Fundamentos de Programación
Jesús Carretero, Félix García, y otros
Thomson-Paraninfo 2007. ISBN: 978-84-9732-550-9

- ✓ Capítol 8

Problemas Resueltos de Programación en Lenguaje C
Félix García, Alejandro Calderón, y otros
Thomson (2002) ISBN: 84-9732-102-2

- ✓ Capítol 6

Resolución de Problemas con C++
Walter Savitch
Pearson Addison Wesley 2007. ISBN: 978-970-26-0806-6

- ✓ Capítol 10 (excepte apartat 10.4)
- ✓ Capítol 11 (apartat 11.1)