Programación I

Estructura de datos MATRIZ

TEMAS de la CLASE

1. Definición de tipo de dato Matriz
2. Operaciones frecuentes en el tipo Matriz
3. Ejercitación

Arreglos bidimensionales: Matrices

- Una matriz es una estructura de datos compuesta que permite acceder a cada componente utilizando dos índices (fila y columna).

- Estos índices permiten ubicar un elemento dentro de la estructura. El primer índice indica la fila y el segundo índice indica la columna

Un tipo de dato Matriz es una colección indexada de elementos.

Características importantes:

- Homogénea
- Estática
- Acceso directo
- Lineal

Matrices: Declaración

En Pascal:

Matriz = **Array** [fila, columna] **of** tipo_elementos;

Es posible indexar los elementos por un índice que corresponde a **cualquier tipo ordinal**:

- Entero
- Carácter
- Subrango

Los elementos de un arreglo pueden pertenecer a **cualquier tipo de datos**:

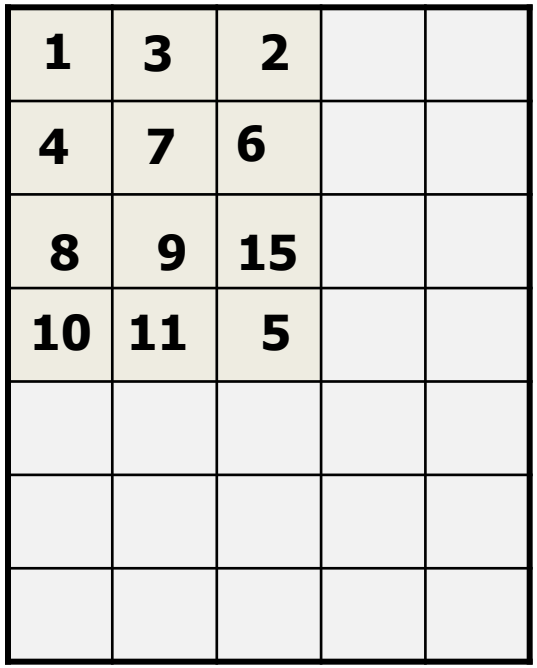
- Enteros
- Reales
- Caracteres
- Registros
- String
- Lógicos
- Otro arreglo

Matrices: Carga de datos

```
Const
  maxfil=50;
  maxcol=40;
Type
   rangof = 0..maxfil;
   rangoc = 0..maxcol;
   matriz = array [1..maxfil, 1..maxcol] of integer;
```

Si dimFila es 4, dimCol es 3 y se leen los siguientes valores:

1 3 2 4 7 6 8 9 15 10 11 5




```
Procedure  Cargar ( var M:matriz; var DimFila: rangof;
                    var DimCol: rangoc);
Var i: rangoF;
     j: rangoc;
  Begin
     readln(DimFila);
     readln(DimCol);
     if Validar(DimFila, DimCol) then
                              For i:=1 to DimFila do
                                 For j:= 1 to DimCol do
                                    readln(M[i,j]);
  End;
```

Esta carga se realiza por filas, pero se podría cargar por columnas…

Matrices: Imprimir

```
Const
  maxfil=50;
  maxcol=40;
Type
   rangof = 0..maxfil;
   rangoc = 0..maxcol;
   matriz = array [1..maxfil, 1..maxcol] of integer;
```

Observar Parámetros

```
Procedure Imprimir (var M: matriz; DimFila: rangof;
                                   DimCol: rangoc);
var i: rangof;
      j: rangoc;
Begin
For i:=1 to DimFila do
    For j:= 1 to DimCol do
        writeln(M[i,j]);
End;
```

1	**3**	**2**		
4	**7**	**6**		
8	**9**	**15**		
10	**11**	**5**		

1
3
2
4
7
6
8
9
15
10
11
5

El recorrido de la matriz está realizado por filas, también podría realizarse por columnas…

Matrices: Buscar un elemento

```
Function Buscar ( M: matriz; DimFila: rangof; DimCol: rangoc; elem: integer): boolean;
 var
    existe : boolean;
    i: rangof;
    j: rangoc;

 Begin
  existe := false;
  i:= 1;
  while ( i<= DimFila) and (not existe) do begin
     j := 1;
     while ( j <= DimCol) and (not existe) do
        if M[i,j] = elem then existe := true
                         else j := j+1;
     if not existe then i := i+1;
  end;
  Buscar := existe;
End;
```

existe-> false->true
i -> 1->2
j -> 1-> 2->3->4-> 1-> 2

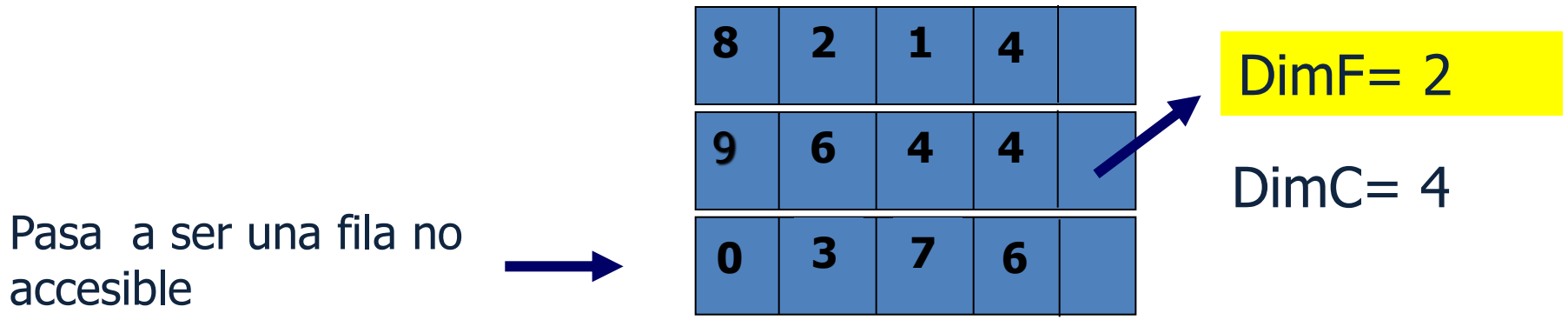
Buscar el elemento 8

1	**3**	**2**		
4	**8**	**6**		
8	**9**	**15**		
10	**11**	**5**		

La búsqueda de la matriz está realizada por filas, también podría realizarse por columnas...

Matrices: Eliminar una fila

Supongamos que necesitamos eliminar la fila 2 de la siguiente matriz.



Matrices: Eliminar una fila

```
Procedure EliminarFila ( var M: matriz; var DimFila: rangoF; DimCol: rangoC;
                         fila: rangoF; var exito:boolean);
 var k: rangoC;  f :rangoF;
begin
  If ( fila >0 and fila < DimFila )  { verifica que el índice es válido, para efectuar la operación}
  then begin
          exito := true;
          For f := fila to DimFila-1 do
               For  k := 1 to DimCol do
                    M [f, k] := M [f+1, k];
               DimFila := DimFila - 1
        end
  else If fila = DimFila then begin
                                 DimFila := DimFila – 1;
                                 exito := true;
                               end
                       else exito := false;
end;
```

Matrices: Suma

```
{solo se pueden sumar matrices de igual dimensión}

Procedure  Sumar (var M1, M2: matriz;
                  var M3: matriz;
                  DimFila: rangof; DimCol: rangoc);
var
     i: rangof;
     j: rangoc;

Begin
   For i:=1 to DimFila do
      For j:= 1 to DimCol do
         M3[i,j] := M1[i,j] + M2[i,j];
end;
```

M1

1	3	2	
4	7	6	
8	9	15	
10	11	5	

+

M2

-1	1	2	
1	2	4	
2	3	-4	
5	5	5	

=

M3

0	4	4	
5	9	10	
10	12	11	
15	16	10	

Matrices: Suma

{solo se pueden sumar matrices de igual dimensión}

```
Procedure  Sumar (var M1, M2: matriz;
                  var M3: matriz;
                  DimFila: rangof;
                  DimCol: rangoc);
var
     i: rangof;
     j: rangoc;

Begin
   For i:=1 to DimFila do
      For j:= 1 to DimCol do
         M3[i,j] := M1[i,j] + M2[i,j];
end;
```

```
Begin
  Cargar(M1, DimFila1, DimCol1);
  Cargar(M2, DimFila2, DimCol2);
  if (DimFila1=DimFila2) and
      (DimCol1=DimCol2)
     then Begin
        Sumar(M1,M2,M3,DimFila1,DimCol1);
        Imprimir(M3,DimFila1,DimCol1);
     end
    else writeln('No se pueden sumar');
end.
```

Matrices: Ejercitación

Ejercicio 1: Implementar un módulo que inserte una fila en una matriz. El módulo debe recibir la matriz, un vector y un valor entero que representa el número de fila donde se insertará el vector.

Ejercicio 2: Un teatro pone a la venta las entradas para la obra "XXX". La sala donde se realizará la obra dispone de 50 filas con 70 butacas por cada fila. Implementar un programa que permita la venta de entradas. Cada cliente indica la ubicación de su interés y si está ocupada, de ser posible se le indicará la ubicación más cercana dentro de la misma fila. Se debe registrar la compra con el dni del cliente.
La venta de entradas finaliza cuando llega el cliente con dni -1 o cuando no quedan más entradas disponibles.
Nota: El cliente puede solicitar hasta dos entradas por vez y sólo se podrán vender si hay dos ubicaciones contiguas libres.