DEFINICION

Un arreglo es una estructura de datos homogénea formada por varios elementos **todos** del mismo **tipo** los cuales están almacenados **consecutivamente** en memoria y se referencia con un **nombre común.**

Cuando hablamos de estructura homogénea nos referimos a que **todos sus elementos deben ser del mismo tipo**, consecutivamente a que existe un primer elemento, un segundo elemento y así sucesivamente y **todos comparten un mismo nombre**.

Es una estructura estática por definición, una vez que se declara se define su tamaño, lo que significa que conocemos con antelación cuantas posiciones deben ser recorridas, sea para cargar o llenar un arreglo o para acceder a sus elementos.

Como cada elemento del arreglo comparte un mismo nombre, lo que hace posible ubicar un elemento respecto a otro, **es su posición relativa** dentro del mismo arreglo, lo que nos permite acceder a un contenido o elemento en particular. Esta posición relativa funciona usando una variable llamada **índice** que nos permite acceder a una posición.

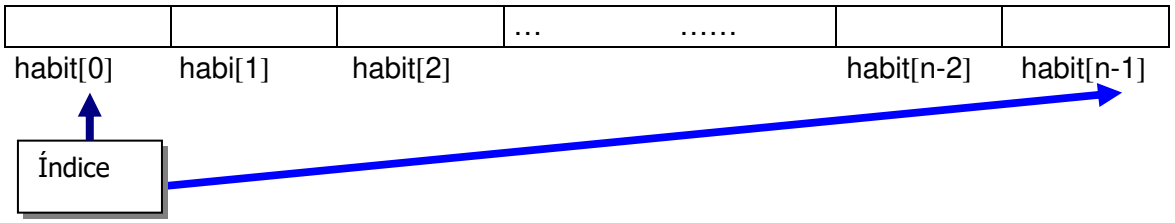
Cada elemento puede ser accedido directamente por el nombre de la variable array seguido de un índice encerrado entre paréntesis.

El valor de cada índice se expresa como una constante entera, variable entera o expresión entera.

REPRESENTACIÓN GRAFICA

Como cada elemento de un arreglo se almacena en posiciones consecutivas de memoria, se puede representar gráficamente un arreglo de n elementos como una tabla de n casillas.

Nombre del arreglo: habit(20)



Los arreglos pueden ser de 3 tipos a saber: Arreglos unidimensionales, bidimensionales o multidimensionales. En este entorno sólo veremos los dos primeros.

1. Arreglos unidimensionales o de una dimensión o vectores

Definición:

Se refiere a arreglos que tienen una sola dimensión y se manejan usando un solo índice. Se les conoce como tabla, vector, es decir que un vector puede ser columna.

1.1 Declaración de un arreglo unidimensional

Los arreglos se declaran como las variables ordinarias, excepto que deben tener una

especificación de tamaño para indicarle al compilador el número de elementos de ha de tener.

Formato

[Tipo de dato nombre del arreglo (tamaño)]

Tipo: indica el valor de los elementos del arreglo. Puede ser cualquier tipo.

Nombre del arreglo: es el identificador que da nombre al arreglo.

Tamaño: constante que indica el número de elementos del arreglo y que debe conocerse en tiempo de compilación, en este caso antes de usarse.

Ejemplos:

flotante lista(100) cadena nombre (40) entero empleado (50)

Indica al compilador la cantidad de almacenamiento de memoria necesario para guardar 100 elementos de tipo entero para la variable lista ó 40 elementos para la variable nombre.
Suponga que queremos almacenar las edades de los 50 empleados de una empresa. Lo cual crearía un arreglo en memoria de 50 elementos, ya que son 50 empleados tal:

entero empleado(50)

36	29	25		35	40
0	**1**	**2**	**3**	**4**	**..... 15 ..... 50**

Al definir **entero empleado (50),** estamos declarando un vector de 50 elementos de tipo entero, todos con el nombre empleado. Sus posiciones estarían referidas por un **índice** que va de 0 a 50. Significa que según lo mostrado, el contenido de empleado(0) es 36 y el contenido de empleado(1) es 29 o sea el elemento cero del arreglo, o dicho de otra forma, el elemento que está en la primera posición física del arreglo tiene como contenido la cantidad 36. Dicho de otra forma, el primer empleado registrado tiene una edad de 36 años. Podemos conceptualizar este almacenamiento así:

NOMBRE DEL ARREGLO: empleado **tipo de dato**: entero

ELEMENTO (CONTENIDO)

40

POSICIÓN: 15

El número de índices determina la dimensionalidad del arreglo.

X(1) Arreglo unidimensional.

B(3,2) Arreglo bidimensional.

Tercero(2,3,5) Arreglo multidimensional

Ejemplo: Arreglo unidimensional llamado datos con tres elementos los cuales se identifican como:

flotante datos (3)

datos (0) datos (1) datos (2)

Reglas para Manejo de Índices

1. Un subíndice debe ser un número entero, pero también pueden ser el valor de una variable. Ejemplo:

```
entero i , entero tab(5)
 i = 2
 tab (3 ) , tab( i )
```

2. Un subíndice debe tomar solamente valores positivos, enteros.

```
KK (10 )        AA (1000)
H(-5) incorrecto X
```

3. Un subíndice no puede ser una variable con subíndice.

```
entero j, s
KK ( j (s) ) incorrecto
```

4. Un identificador que representa un arreglo no debe usarse sin subíndice:

```
entero tab (10), val
-----------
Leer(dato)
 tab(i) = dato
-------------------------
 val = tab INCORRECTO
```

5. El nombre de la tabla tal como aparece en el enunciado debe tener exactamente el mismo número de subíndices que en cualquier otra parte del programa.

Ejemplos:

```
entero tab(10)
X tab(i, j) ← ERROR
```

1.2 Inicialización de un arreglo de una dimensión

Para inicializar un arreglo se debe hacer referencia **a la posición que nos interese, a través del**

nombre_del arreglo (índice).

Ejemplo: si nos interesa inicializar la primera posición del arreglo a con un valor de 2.5 sería:

```
a (0) = 2.5,   a(i) = 2.5    //considerando que i tiene un valor de 1
```

Mientras, si lo que deseamos es acceder al elemento, es decir, al contenido que tiene esa posición:

n = a (0) ó n = a (i)

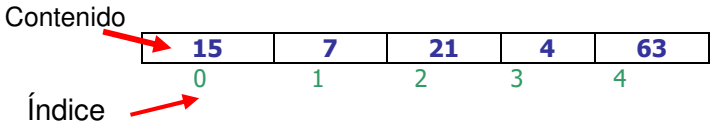
Donde **n** es la variable a quien se le asigna el elemento del vector, por lo tanto, debe ser declarada del mismo tipo de dato que el vector.

1.3 Procesamiento de arreglos Unidimensionales

Las operaciones de asignación, de comparación o recorrido se realizan elemento por elemento, para ello se utilizan ciclos.

a. Operaciones con Arreglos

Como en toda estructura de datos podemos realizar diferentes operaciones. Dado un arreglo llamado **entero vec(5) :**



Entre las operaciones más frecuentes con los arreglos tenemos:

- **ASIGNACIÓN**: La asignación de valores a un elemento del vector se realizará con la operación de asignación.
  vec (4) = 45 Almacena 45 en la posición vec(4)

Si se desea asignar valores a todos los elementos del vector se deben usar estructuras de repetición (Para, Mientras, Hasta que).

Operaciones matemáticas:

suma	=	vec(0) + vec(1)	suma = 22
vec (2)	=	vec (4) + 2	vec(2) = 65
vec(3)	=	vec (1) + vec (4)	vec(3) = 70
suma	=	suma + vec (3)	suma = 92

- **Recorrer un arreglo**: no es más utilizar un índice que se mueve desde el primero hasta el último elemento o dependiendo del último al primer elemento.

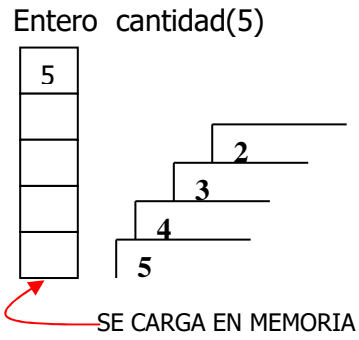
```
entero i
entero table(5)
 Para i = 0,4,1
      table(i) =i * i
 FinPara
```

- **Lectura y Escritura :**

Se puede acceder a los elementos de un vector para introducir datos en él, lo cual se conoce como **cargar un vector** desde el teclado.

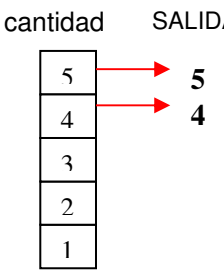
- Cargar un vector (Solicitando los datos desde el teclado)

```
Para i=0,4, 1
   Leer (dato)
   cantidad(i) = dato
Fin Para
```



- Escritura

Significa que se va a acceder a los elementos de un vector para visualizar o mostrar su contenido.




```
Para i = 0,4,1
    Imprimir (cantidad (i) )
Fin Para
```

1.4 Paso de Arreglos Unidimensionales a Funciones

El paso de arreglos a métodos se puede realizar de dos formas

- Un elemento del arreglo (**paso por valor**).
- El arreglo completo (**paso por referencia**).

Todos los ejemplos que hemos visto anteriormente, han sido trabajados con el concepto de **paso por valor**. Recordemos que cuando se pasa un parámetro por valor a un método, el parámetro será una copia del valor transferido.
Ejemplo. Si tenemos este segmento del algoritmo:

```
entero n
n = 25
Saludar (n)   /* llamada a la función,  mandando el valor 25 */
 Imprimir (n)

   :

Saludar( entero numero) /* parámetro  por valor : número que recibe 25  */
    { Imprimir (numero)  /* imprimirá  25  */
       numero = 34    /* modifica el valor 25 que tenía numero  */
        Imprimir (numero)   /* imprimirá  34  */
     }
```

El parámetro ***número*** del método *saludar*, recibe una copia del argumento *n* de la llamada, en este ejemplo el valor 25. Por lo cual, cualquier cambio que se realice en la variable *numero* sólo afectará a la copia (numero) y no al original (n). Así cuando se imprima *n*, una vez ejecutado el método *saludar* esta variable mantendrá su valor de 25.

1.4.1 Pasando un elemento del arreglo

Se debe especificar el elemento del arreglo que se desea enviar a la función. Esto se logra ubicando el nombre del arreglo y los subíndices del elemento que se desea enviar.

Por ejemplo:

```
_______
entero vect(3)
entero r

r = proc(vec(2))
---
---
publico entero  Proc (entero x)
   {
     entero  suma = 5   /* x  recibe el valor del elemento 2 del vector vec */
        suma  = suma + x
        retornar suma
 }
```

1.4.2 Definición de Paso por Referencia.

Toda variable que es declarada en un programa, tiene en memoria un espacio o celda para guardar data. Esta celda o espacio tiene un **único número** que la identifica y este número se conoce como **dirección de memoria**.
El paso por referencia es enviar la **dirección de memoria** de una variable a un parámetro. Esto ocasiona que cualquier cambio al parámetro, en la función, afecta a la variable original.

Existen estructuras de datos que presentan características que permiten recibir una **dirección de memoria**, tales como los **arreglos y los objetos**.

1.4.2.1 Pasar un arreglo completo a una función: Para hacer esto, en la llamada a la función, se pone el nombre del arreglo sin índice y en la función se definirá el parámetro formal con paréntesis en blanco, el cual recibirá la dirección de memoria.
Al pasar un arreglo como argumento a una función, el nombre del arreglo pasa la **dirección de memoria** (o referencia) del primer elemento del vector, lo cual permitirá acceder a ese arreglo en la función.

Ejemplo: Si declaramos un vector de 3 elementos, **vec(3)** estamos reservando memoria para 3 valores. El nombre del arreglo **vec** guardará la dirección de memoria del primer elemento.

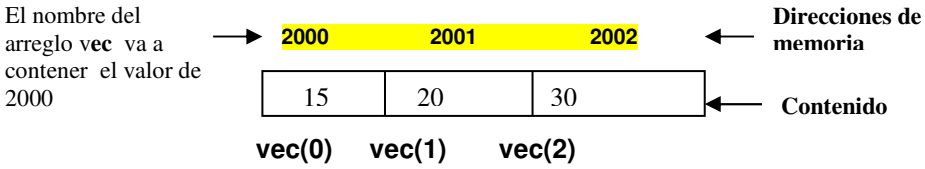
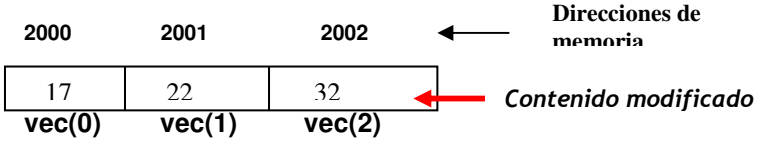


Figura 1: Representación gráfica de la dirección de memoria

```
Proc(vect)    /* Llamada al método proc  */
---
Proc (entero tab( ))      /* tab recibe la dirección del  primer elemento de vec : 2000 */
{
 entero i, suma
   para i = 0, 2,1
       tab(i)  = tab(i) + 2
   fin para
}
```

En este ejemplo, el nombre del arreglo ***vec*** guarda la dirección de memoria **de su primer elemento,** el cual sería **2000,** tal como se observa en la figura 1. Por ello, ***tab*** recibe esa dirección o referencia y, a partir de allí, se puede acceder a los elementos de ***vec***.

Si imprimiéramos el vector ***vec*** después del llamado de la función P***roc***, veríamos que los elementos contienen su valor incrementado en 2.



La memoria y su contenido modificado

1.4.2.2. Retornar un elemento del arreglo

También es paso por valor. Se debe especificar el elemento del arreglo que se

desea devolver al programa principal. Esto se logra ubicando el nombre del arreglo y los subíndices del elemento que se desea enviar desde la función.

```
entero  Proceso(flotante tab( )){
   --------------
    Retornar (tab(3))
  }
INICIO.
  flotante tab(10)
  entero r
  -----
  r = Proceso(tab)
  -----
FIN.
```

2. Arreglos Multidimensionales

Los arreglos multidimensionales son comúnmente llamados ***matrices***. En este punto veremos solamente el manejo de arreglos **bidimensionales**, pero recordemos que se pueden manejar matrices de 3, 4 o más dimensiones, por ejemplo: mata (2,2) es un arreglo bidimensional, ventas (3,4,2) tridimensional, donde el primer arreglo hace referencia a fila columna y el segundo ejemplo hace referencia a fila- columnas - profundidad.

2. 1 Arreglo Bidimensional

Un arreglo bidimensional es aquel que se representa a través de dos dimensiones y se concibe como una matriz de filas y columnas. Llamamos *filas* a las líneas horizontales y *columnas* a las verticales. Para trabajar con arreglos de 2 dimensiones se requieren de 2 subíndices para poder acceder a cualquier elemento.

Declaración:

[tipo nombre_nombre (tamaño,tamaño)]

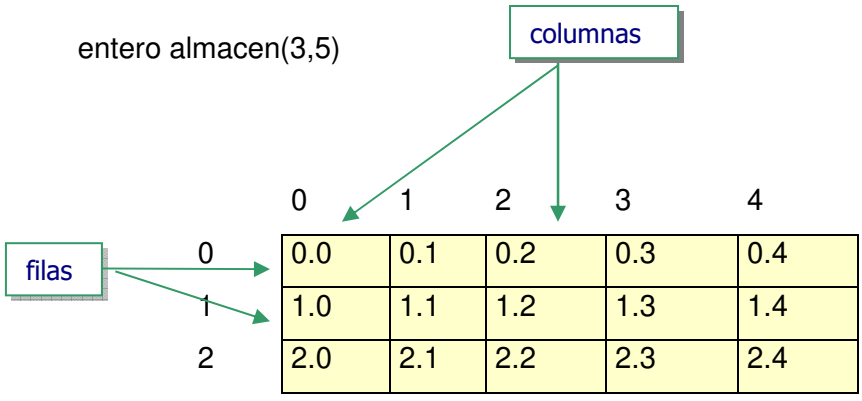
tipo: es el tipo de datos del arreglo

nombre: nombre que lo identifica.

tamaño: número d elementos.

Para declarar un arreglo d de enteros bidimensionales 10,20 se escribirá:

entero almacen(3,5)



Ejemplos de declaración de arreglo bidimensional:

entero mat (4,4) /* **mat** tiene 4 filas y 4 columnas y maneja 16 elementos*/

flotante total (5,4) /* **total** tiene 5 filas y 4 columnas y maneja 20 elementos*/

booleano mat (3,8) /***mat** tiene 3 filas y 8 columnas y maneja 24 elementos */

2.1 Cómo acceder los elementos de un arreglo bidimensional.

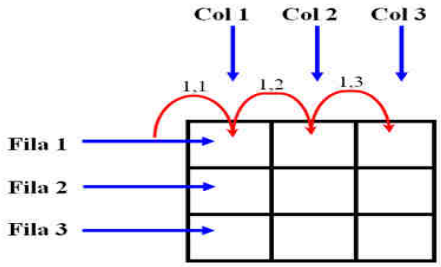
Para acceder a los elementos de una matriz se requiere de 2 variables subíndices, una para la fila y otra para la columna.

Si se desea acceder al valor 1.3 de la matriz **almacén**, se debe indicar la fila y la columna en donde se encuentra almacenado el valor, esto sería,
almacén (1,3).

2.2 Manipulación de Matrices

Para cualquier operación que se realice sobre una matriz, se debe indicar el tipo de recorrido que desea hacerse sobre ella. Para esto se requiere de estructuras repetitivas con el fin de manejar los subíndices del arreglo. Para una matriz se requiere de dos ciclos anidados, uno que controle las filas y otro que controle las columnas. Luego el usuario debe establecer como desea recorrer la matriz, esto es, por fila o por columna.

Recorrido por fila: Cuando se recorre a la matriz por fila, se accede a los elementos de la primera fila, luego los elementos de la segunda y así sucesivamente hasta recorrer todo el arreglo.
Por ejemplo:



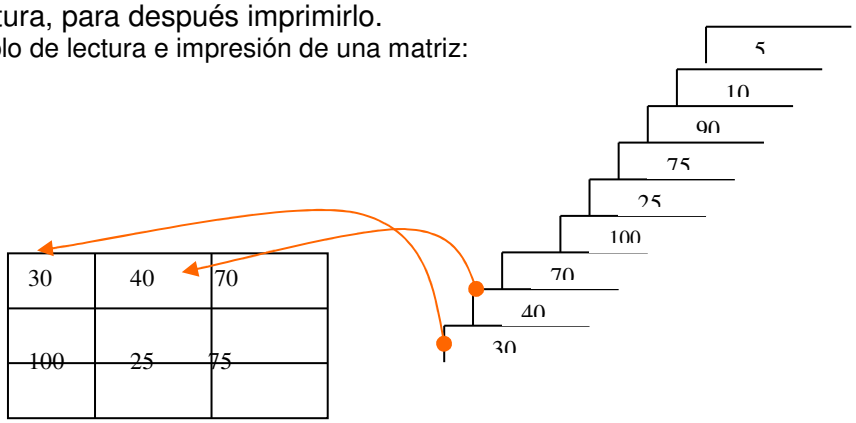

```
Para f = 1,3,1
 Para c = 1,3,1
    Imprimir ( mat (f,c))
 Fin para c
Fin para f
```

. Representación gráfica del recorrido por fila

En el diagrama se observa que en el recorrido por fila, el ciclo que controla las columnas varía mucho más rápido que las filas.
Recorrido por Columnas: En este recorrido se accede al contenido de los elementos de la primera columna, luego los elementos de la segunda columna y así sucesivamente hasta recorrer todo el arreglo.
En este recorrido el ciclo que controla las filas varía muchos más rápido que el de las columnas.

Lectura e impresión de una matriz.

Una vez definido el tipo de recorrido se procede a cargar el arreglo, a través de la lectura, para después imprimirlo.
Ejemplo de lectura e impresión de una matriz:



Representación gráfica de cargar una matriz desde registros de entrada por fila

Los ejemplos a continuación reflejan el recorrido por fila y otro por columna

```
/* Se carga por fila*/
    entero mat (3,3).
    entero f, c,  num
     Para f = 0, 2,1
       Para c = 0,2,1
          Leer(num)
          mat (f,c) = num
       Fin para
    Fin para

  /* Se imprime por columna*/
  Para c = 0,2,1
    Para f = 0,2,1
      Imprimir ( mat ( f, c) )
    Fin para
  Fin para
```

Operaciones Matemáticas:

```
/* Sumar todo los elementos de una matriz */
entero mat (3,3)
entero f, c,  tot = 0
Para f = 0,2,1
   Para c = 0,2,1
    tot = mat (f,c) + tot
  Fin para c
Fin para f
```

3. Paso de arreglos Bidimensionales a funciones

Para hacer esto, en la llamada a la función, se pone el nombre del arreglo sin índices y en la función se definirá el parámetro formal con el nombre del arreglo y paréntesis en blanco en el elemento fila y el tamaño en la columna. De ésta forma se recibirá la dirección de memoria.
Al pasar un arreglo como argumento a una función, el nombre del arreglo pasa la **dirección de memoria** (o referencia) del primer elemento del arreglo, lo cual permitirá acceder a ese arreglo en la función.

```
Lectura_D(entero arr1( )(4))
{
  entero filas, col
  Para filas = 0, 4, 1)
     Para col = 0, 3, 1)
          Leer( dato)
          arr1(filas)(col) = dato
     Fin Para
  FinPara
Retornar ()
}
INICIO.
{
  entero arr(5)(4)
   -----
   Lectura_D(arr)
  Imprimir("----")
FIN.
```

3.3.1 Paso de un elemento del arreglo (es paso por valor)

```
 Proceso (entero x)
{------
 Retornar()
 }
INICIO.
 ------
  Proceso (arr(1)(3))
------
FIN.
```

3.2 Retornando un elemento del arreglo

```
entero Proceso( )
{
  entero  a(3)(5)
  ------
  Retornar (a(0)(2) )
}

INICIO.
  Entero result
  entero r
  -----
  r = Proceso( )
  -----
FIN.
```