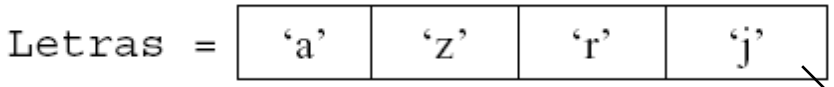
ARREGLOS (ARRAYS)

Arreglos(Arrays)

Definición:
Un Array es un conjunto de variables del mismo tipo que tienen el mismo nombre y se diferencian en el índice.



0 1 2 3

Cada elemento del array está numerado y a este número se lo denomina índice

A cada dato almacenado se le denomina elemento del array o ítem

Los elementos del array se numeran consecutivamente comenzando con 0,1,2,,….
Los índices permiten localizar cada elemento del array.

letras[0] es el elemento que está en la posición 0 y su valor es a
letras[3] es el elemento que está en la posición 3 y su valor es j

Los elementos almacenados en el array pueden ser de cualquier tipo: tipos simples como: int, char, Bool, float ó tipos definidos por el programador como por ejemplo las estructuras.

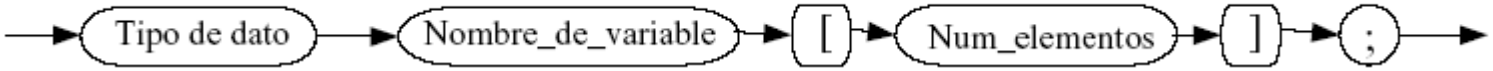
Arreglos(Arrays)

Declaración de un array

Se declara de forma similar a cualquier otro tipo de datos, solo que hay indicarle al compilador el Número de elementos que forma el array; tamaño ó longitud del array.

Sintaxis

<tipo_de_dato> <nombre_de_variable> [número_de_elementos];

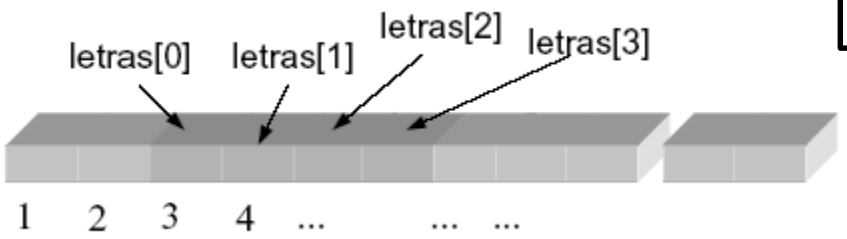


Ejemplos:

char letras[4]; //variable letras de tipo array de 4 elementos de tipo char
int edades[10]; // variable edades de tipo array de 10 elementos de tipo entero.

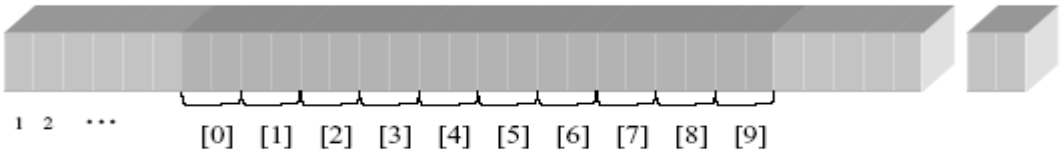
El compilador reserva espacio en memoria en posiciones contiguas.

El array letras ocupa 4 bytes de memoria



Arreglos(Arrays)

Cuando declaramos el array edades lo que hacemos es reservar un espacio de memoria para 10 Variables del tipo int.

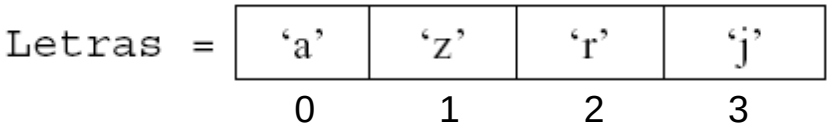


El array ocupa 20 bytes Consecutivos de memoria (2 bytes por elemento)

Si se quiere saber el número de bytes que se necesita para almacenar un array en memoria, se Puede utilizar la función sizeof(), (pertenece a librería stdio.h)
Ej.: sizeof(edades) = 20.

Acceso a los elementos de un array

Para acceder a los elementos de un array hay que poner el nombre y el índice del elemento al que se quiere acceder ó referenciar.




```
…
Int primera = 1, k = 3;
Char letras [4];
…
Cout<< letras [0]   //visualiza a la letra ´a´
Cout <<letras [2]   //visualiza a la letra ´r´
Cin>> letras [0]
Letras [2] = ´R´;
…
Letras [0] = letras [primera+2]
Cout <<letras [k-2];  //visualiza la letra ´z´
..
```

Podemos visualizar el valor de los elementos de un array

Modificar el valor de los elementos de un array

Acceder a los elementos utilizando fórmulas

Arreglos(Arrays)

Se puede acceder a todos los elementos, utilizando uno de los tipos de sentencia de bucle como el **for**

```
for ( int j=0; j <num_elementos ; j++)
        procesar_elemento [j];
```

Variable de control j para recorrer cada uno de los elementos

```
…
char letras[4];
…
for  (int j=0;j<a;j++)
    {
      cout<< “Introduzca el elemento:” <<j+1;
      cin>> letras[j];
      cout << endl;
     }
  …
```

Pasada	j	Salida por pantalla
1	0	Introduzca el elemento: 1
2	1	Introduzca el elemento: 2
3	2	Introduzca el elemento: 3
4	3	Introduzca el elemento: 4

C++ no comprueba que los índices del array estén dentro del rango permitido.

Si escribimos cout<< letras [4]; no tendremos error de compilación, pero el programa no funcionará correctamente.

Arreglos(Arrays)

Inicialización de un Array

Antes de empezar a utilizar una variable de tipo Array hay que asignar valores a cada uno de sus elementos. Tenemos varias formas de inicializar un array:

1. Inicialización de la declaración:

…
char letras [4]= { ‘a’,’z’,’t’,’j’};

Los valores se encierran entre llaves y se separan por comas

…
int edades [] = {10, 20, 30, 40, 50,60};
char saludo [] { ‘h’,’o’,’l’,’a’};
char saludo [] {“hola”};
…

C++ permite omitir el tamaño del array cuando se inicializa. El compilador reserva memoria para un array de enteros de tamaño 6.

Los array de caracteres se pueden inicializar de éstas 2 formas

2.Inicialización elemento a elemento en el cuerpo del programa:

Letras [0]=’a’;
Letras [1]=’z’;
Letras [2]=’r’;
Letras [3]=’j’;

3. Inicialización mediante una sentencia FOR:

```
 Int edades [6];
Int valor= 10;
For (int i =0; i<6 ; i++)
    {
       Edades[i] = valor;
      valor = valor + 10;  }
```

edades =	10	20	30	40	50	60

Arreglos(Arrays) Multidimensionales

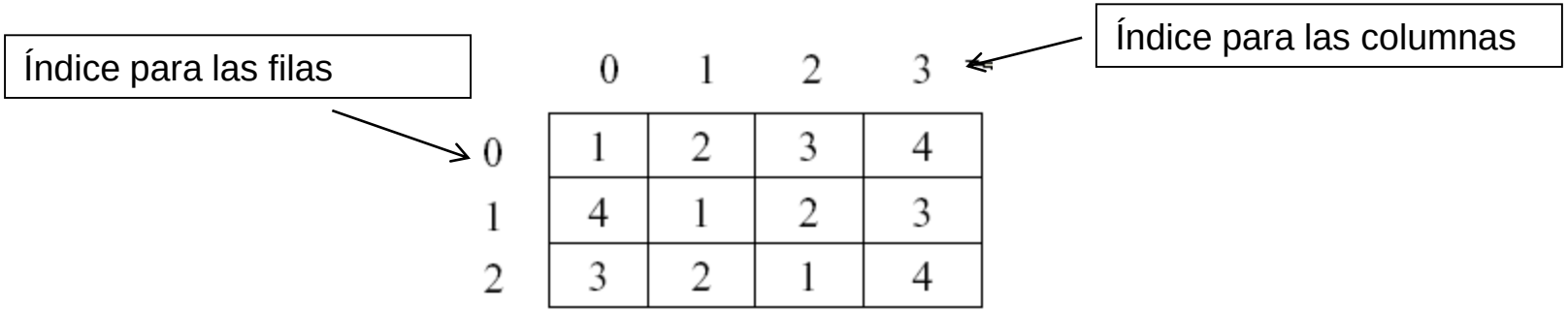
Definición

Los Arrays vistos anteriormente se conocen como Arrays unidimensionales y se caracterizan por tener un solo índice. También se conocen como listas de elementos y se corresponden con el concepto de ***vector***.

Los arrays multidimensionales son aquellos que tienen más de una dimensión y por lo tanto tienen más de un índice. Los más utilizados son los de dos dimensiones, conocidos con el nombre de tablas. Se corresponden con el concepto ***matriz.***

C++ permite trabajar con arrays de tantas dimensiones como requieran las aplicaciones (3, 4 ó más dimensiones).

Una array de dos dimensiones se corresponde con una tabla con varias filas y varias columnas.



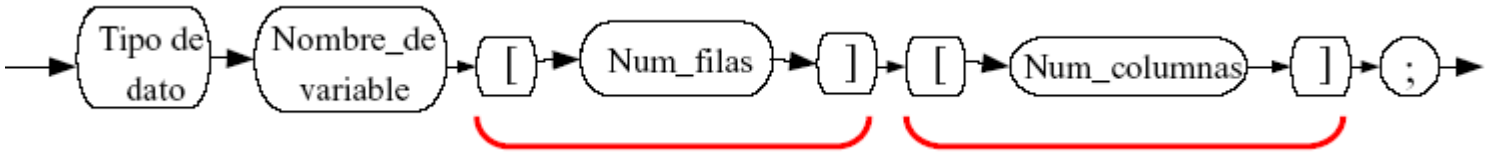
Cada elemento almacenado en el array está identificado por dos índices, sus coordenadas, las filas y las columnas en la que se encuentras dicho elemento. Ambos índices se numeran consecutivamente comenzando con 0, 1, 2 …

Arreglos(Arrays) Multidimensionales

Se declara de forma similar al tipo de dato array de una dimensión, solo que hay que indicarle al compilador el tamaño de cada uno de los índices, es decir, el número de filas y el número de columnas.

Sintaxis

<tipo_de_dato> <nombre_de_variable> [número_de_filas] [número_de_columnas];



Ejemplos:

```
char  tablero[8][8];   // variable llamada tablero de tipo array de dos dimensiones (8 filas y
                          8 columnas). Alamacena 64 elementos de tipo char.

int   matriz[3][4];    // variable llamada matriz de tipo array de dos dimensiones (3 filas y 4
                          columnas). Almacena 12 elementos de tipo entero.
```

Al igual que en los arrays de una dimensión, el compilador reserva espacio en memoria en posiciones contiguas. Primero se almacenan los datos de la primera fila, luego los de la segunda…

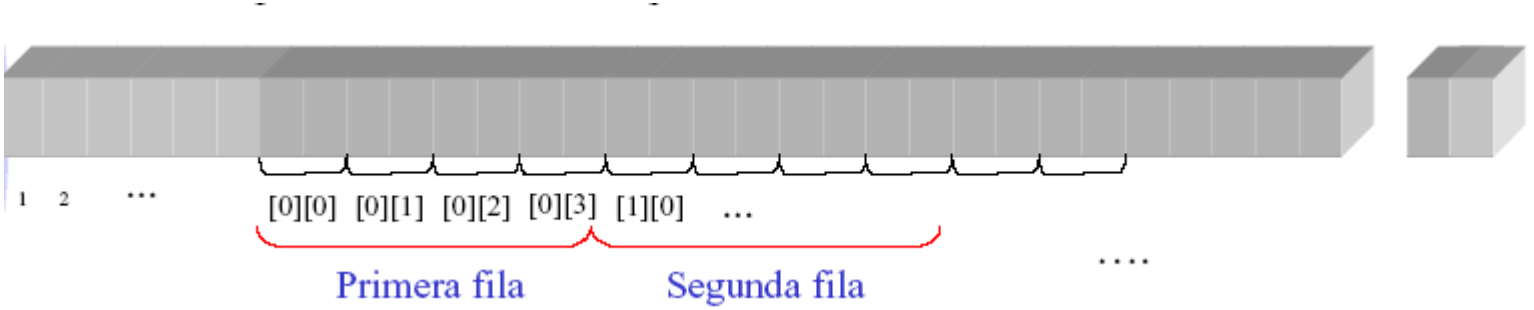
```
int  matriz[3][4];
```

	0	1	2	3
0	1	2	3	4
1	4	1	2	3
2	3	2	1	4

El array **matriz** ocupa 24 bytes consecutivos de memoria

Arreglos(Arrays) Multidimensionales

Cuando se declara el array matriz lo que hacemos es reservar un espacio de memoria para 12 variables de tipo int.



Acceso a los elementos

Se puede acceder a los elementos de un array bidimensional de forma similar a como se hace para Arrays de una dimensión. Hay que poner el nombre y los índices(fila y columna) del elemento al que queremos acceder ó referenciar

```
int j = 0;
char valor;
int matriz[3] [4];
char cuadrado[3] [3];
cout << matriz[0] [3];
cin >> matriz[1] [j+1];
cuadro[2] [2] = 'R';
Valor = cuadro[2] [1] ;
```

Podemos visualizar el valor de los elementos de un array

Modificar el valor de los elementos de un array

Acceder a los elementos para extraer valores

Arreglos(Arrays) Multidimensionales

Para acceder a elementos de un array bidimensional, utilizaremos un doble bucle como por ej.: utilización de for.

```
for (int i = 0; i< num_filas; i++)
    for (int j = 0 ;  j< num_columnas  ; j++)
            procesar_elemento [i][j];
```

Variable de control **i** para recorrer las filas

Variable de control **j** para recorrer las columnas

Sintaxis

```
 int matriz[3][4];
…
 for (int I = 0 ; I < 3 ; i++)
    for (int j = 0 ; j < 4 ; j++)
    {
        Cout<< “El elemento  ” << i+1 << j+1 << “es: ”;
        Cout << matriz [i][j] << endl;
     }
```

Arreglos(Multidimensional)

Inicialización

Al igual que los arrays de una dimensión, tenemos varias formas de inicializar un Array.

1. Inicialización de la declaración:

Los valores se encierran entre llaves y se separan por comas

```
...
    Int matriz [3][4] = { 1,2,3,4,4,1,2,3,3,2,1,4 }

    int valores [2] [3] = {{10, 20,30}, {0, 1,2,}};
    char cuadro [3] [3] = {
                          { 'B','N','N'}
                          { 'N','N','N'}
                           { 'B','B','B'}
                        };
...
```

	0	1	2	3
0	1	2	3	4
1	4	1	2	3
2	3	2	1	4

	0	1	2
0	10	20	30
1	0	1	2

	0	1	2
0	B	N	N
1	N	N	N
2	B	B	B

2.Inicialización elemento a elemento en el cuerpo del programa:

Poco práctico como ocurría en Arrays de una dimensión.

```
Matriz [0][0] = 1;
Matriz [0][1] = 2;
Matriz [0][2] = 3;
…
Matriz [2][3] = 4;
```

Arreglos(Multidimensional)

3. Inicialización mediante una doble sentencia FOR:

```
Int valores [2][3]
Int dato = 10;
For (int i = 0; i<2 ; i++)
    For (int j = 0; j=3 ; j++)
    {
        Valores [i][j] = dato;
        Datos = dato + 10;
    }
```

Variable de control **i** para recorrer las filas

Variable de control **j** para recorrer las columnas

	0	1	2
0	10	20	30
1	40	50	60

Importante

- Si un array se declara globalmente, y no se inicializa en dicha declaración, el compilador se encarga de inicializarlo automáticamente con un valor por defecto.
- Si no se declaran globalmente, pero se inicializan uno o más elementos pero no todos, el compilador no inicializa automáticamente el resto de elementos con un valor por defecto.
- Si el array almacena elementos de **tipo Entero o real: el valor por defecto es 0.**
- Si el array almacena Elementos de **tipo Carácter**: **el valor por defecto es el carácter nulo '/0'.**

Arreglos(Multidimensional)

Ejemplo de Inicialización Global y Local

Declaración de Variables Globales

```
#include <iostream.h>
Int matriz[3][4];
Char cuadro [3][4];
```

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0

	0	1	2
0	'\0'	'\0'	'\0'
1	'\0'	'\0'	'\0'
2	'\0'	'\0'	'\0'

Declaración de Variables Locales

```
main ()
{
   Float   ventas[4] = {3.2};
   Int otrocuadro[3][3] = {7,6};
```

3.2	0	0	0
0	1	2	3

	0	1	2
0	7	6	0
1	0	0	0
2	0	0	0

```
…..
…..
.....

}
```