Universidad Centroamericana “José Simeón Cañas”
Facultad de Ingeniería y Arquitectura

Tutorías - Fundamentos de Programación Ciclo 01/2021
Tutor: Ernesto José Canales Guillén

Arreglos en C++

Arrays (Areglos)

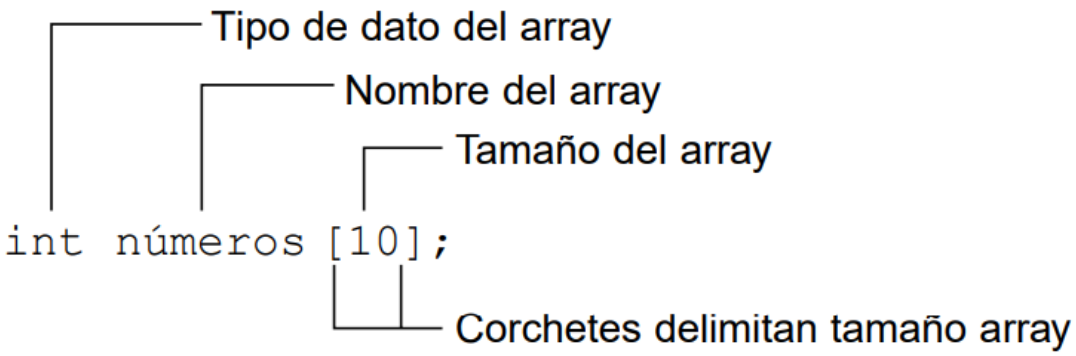
Un array o arreglo (lista o tabla) es una secuencia de objetos del mismo tipo. Los objetos se llaman elementos del array y se numeran consecutivamente 0, 1, 2, 3... El tipo de elementos almacenados en el array puede ser cualquier tipo de dato de C++, incluyendo estructuras definidas por el usuario, como se describirá más adelante. Normalmente, el array se utiliza para almacenar tipos tales como char, int o float.

Cada item del array se denomina elemento. Los elementos de un array se numeran, como ya se ha comentado, consecutivamente 0, 1, 2, 3... Estos números se denominan valores índice o subíndice del array.

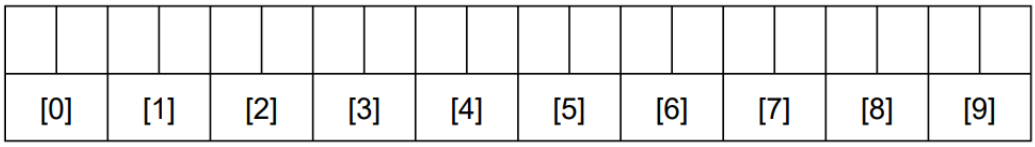
Declaración de un array

Un array se declara de modo similar a otros tipos de datos, excepto que se debe indicar al compilador el tamaño o longitud del array. Para indicar al compilador el tamaño o longitud del array se debe hacer seguir al nombre, el tamaño encerrado entre corchetes. La sintaxis para declarar un array de una dimensión determinada es:

```
tipo nombreArray[numeroDeElementos];
```



Esta declaración hace que el compilador reserve espacio suficiente para contener diez valores enteros. En C++ los enteros ocupan, normalmente, 2 bytes, de modo que un array de diez enteros ocupa 20 bytes de memoria. La Figura muestra el esquema de un array de diez elementos; cada elemento puede tener su propio valor.



Acceso a los elementos de un array

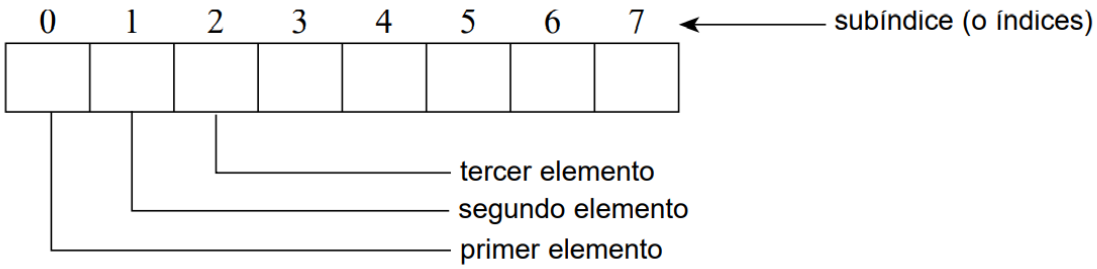
Gran parte de la utilidad de un array proviene del hecho que se pueda acceder a los elementos de dicho array de modo individual. El método para acceder a un elemento es utilizar un subíndice o un índice.

```
nombreArray[n]
```

número del elemento dentro del array

El primer elemento del array tiene de índice 0, el siguiente 1, y así sucesivamente.

```
int edad[8];
```



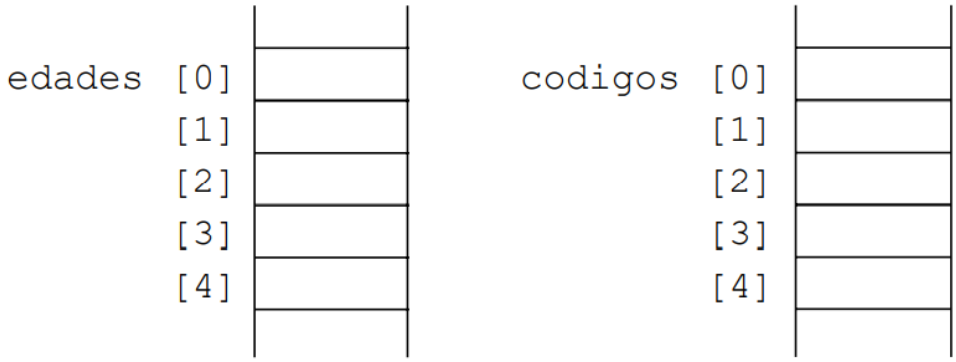
En los programas se pueden referenciar elementos utilizando fórmulas para los subíndices. Mientras que el subíndice puede evaluar a un entero, se puede utilizar una constante, una variable o una expresión para el subíndice.

```
edad[4]
ventas[total + 5]
bonos[mes]
salario[mes[i] * 5]
```

Almacenamiento en memoria de los arrays

Los elementos de los arrays se almacenan en bloques contiguos.

```
int edades[5];
char codigos[5];
```



El tamaño de los arrays (sizeof)

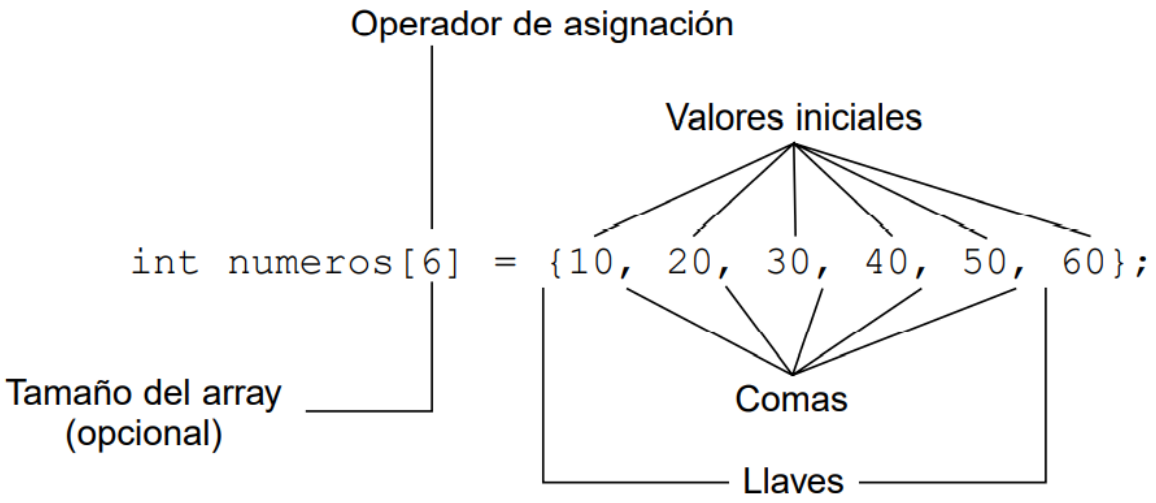
La función sizeof() devuelve el número de bytes necesarios para contener su argumento. Si se usa sizeof() para solicitar el tamaño de un array, esta función devuelve el número de bytes reservados para el array completo.

```
n = sizeof(edades);
```

Inicialización (iniciación) de un array

Se deben asignar valores a los elementos del array antes de utilizarlos, tal como se asignan valores a variables.

```
precios[0] = 10;
precios[1] = 20;
precios[3] = 30;
precios[4] = 40;
...
```



Omitir tamaño del array

No es necesario especificar el tamaño de la matriz. Pero si no lo hace, solo será tan grande como los elementos que se inserten en él:

```
string cars[] = {"Volvo", "BMW", "Ford"}; // size of array is always 3
```

Esto está completamente bien. Sin embargo, el problema surge si desea espacio adicional para elementos futuros. Entonces tienes que sobrescribir los valores existentes:

```
string cars[] = {"Volvo", "BMW", "Ford"};

string cars[] = {"Volvo", "BMW", "Ford", "Mazda", "Tesla"};
```

Arrays de caracteres y cadenas de texto

Una cadena de texto es un conjunto de caracteres, tales como «ABCDEFG». C++ soporta cadenas de texto utilizando un array de caracteres que contenga una secuencia de caracteres.

Es importante comprender la diferencia entre un array de caracteres y una cadena de caracteres. Las cadenas contienen un carácter nulo al final del array de caracteres.

“Las cadenas se deben almacenar en arrays de caracteres, pero no todos los arrays de caracteres contienen cadenas”

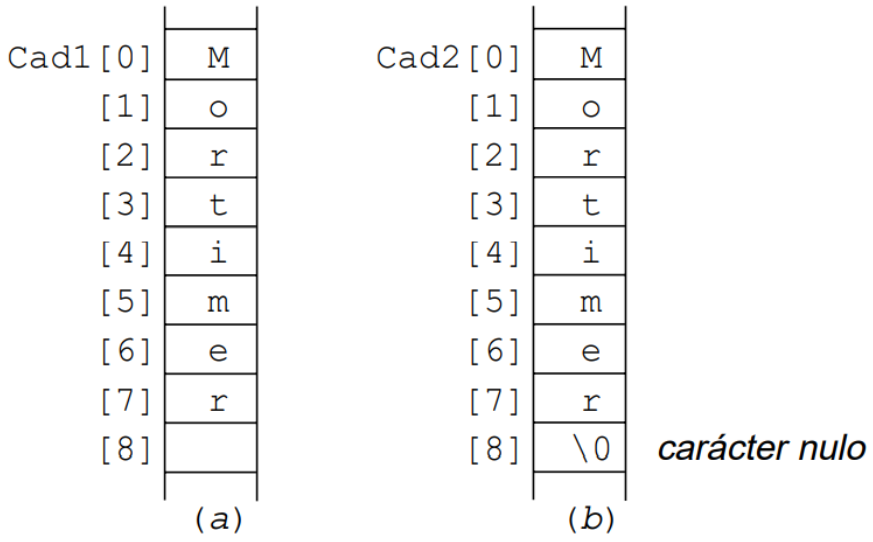


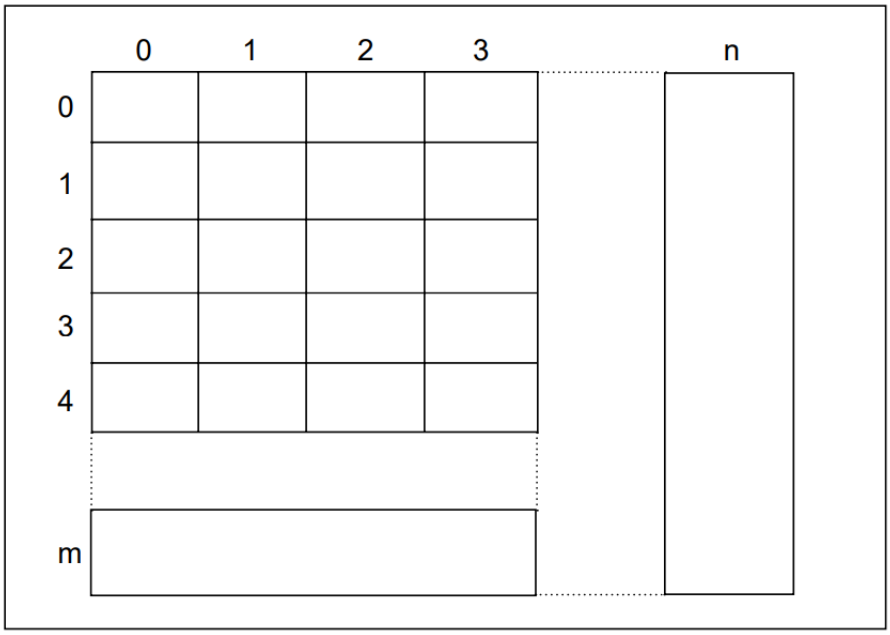
Figura 7.7. (a) Array de caracteres; (b) cadena.

Las cadenas se señalan incluyendo un carácter al final de la cadena: el carácter nulo (\0), cuyo valor en el código ASCII es 0.

Arrays multidimensionales

Los arrays vistos anteriormente se conocen como arrays unidimensionales (una sola dimensión) y se caracterizan por tener un solo subíndice. Estos arrays se conocen también por el término listas. Los arrays multidimensionales son aquellos que tienen más de una dimensión y, en consecuencia, más de un índice. Los arrays más usuales son los de dos dimensiones, conocidos también por el nombre de tablas o matrices. Sin embargo, es posible crear arrays de tantas dimensiones como requieran sus aplicaciones, esto es, tres, cuatro o más dimensiones.

Un array de dos dimensiones equivale a una tabla con múltiples filas y múltiples columnas:



La sintaxis para la declaración de un array de dos dimensiones es:

<tipo de datoElemento> <nombre array> [<NúmeroDeFilas<] [<NúmeroDeColumnas>]

Algunos ejemplos de declaración de tablas son:

```
char Pantalla[25][80];
int puestos[6][8];
int equipos[4][30];
int matriz[4][2];
```

Un array de dos dimensiones en realidad es un array de arrays. Es decir, es un array unidimensional, y cada elemento no es un valor entero de coma flotante o carácter, sino que cada elemento es otro array.

Los elementos de los arrays se almacenan en memoria de modo que el subíndice más próximo al nombre del array es la fila y el otro subíndice, la columna.

Elemento	Posición relativa de memoria
`tabla[0][0]`	0
`tabla[0][1]`	2
`tabla[1][0]`	4
`tabla[1][1]`	6
`tabla[2][0]`	8
`tabla[2][1]`	10
`tabla[3][0]`	12
`tabla[3][1]`	14

Inicialización de arrays multidimensionales

Los arrays multidimensionales se pueden inicializar, al igual que los de una dimensión, cuando se declaran. La inicialización consta de una lista de constantes separadas por comas y encerradas entre llaves, como en:

1.
```
int tabla1[2][3] =  {51, 52, 53, 54, 55, 56};
```

o bien en los formatos:

```
int tabla[2][3]=      {{51, 52, 53},
                       {54, 55, 56} };
int tabla[2][3]=      {{51, 52, 53}, {54, 55, 56}};
int tabla[2][3]=      {
                          {51, 52, 53}
                          {54, 55, 56}
                      };
```

2.
```
int tabla2[3][4] =   {
                          {1, 2, 3, 4},
                          {5, 6, 7, 8},
                          {9, 10, 11, 12}
                      };
```

tabla[2][3]

		0	1	2	*Columnas*
Filas	0	*51*	*52*	*53*	
	1	*54*	*55*	*56*	

tabla[3][4]

		0	1	2	3	*Columnas*
Filas	0	*1*	*2*	*3*	*4*	
	1	*5*	*6*	*7*	*8*	
	2	*9*	*10*	*11*	*12*	

Acceso a los elementos de arrays bidimensionales

Se puede acceder a los elementos de arrays bidimensionales de igual forma que a los elementos de un array unidimensional. La diferencia reside en que en los elementos bidimensionales deben especificarse los índices de la fila y la columna:

```
Tabla[2][3] = 4.5;
Resistencias[2][4] = 50;
AsientosLibres[5][12] = 5;
```

Se puede acceder a los elementos de arrays bidimensionales mediante bucles anidados:

```
for (int IndiceFila = 0; IndiceFila < NumFilas; ++IndiceFila)
   for (int IndiceCol = 0; IndiceCol < NumCol; ++IndiceCol)
      Procesar elemento[IndiceFila] [IndiceCol]
```

Arrays de más de dos dimensiones

C++ proporciona la posibilidad de almacenar varias dimensiones, aunque raramente los datos del mundo real requieren más de dos o tres dimensiones. El medio más fácil de dibujar un array de tres dimensiones es imaginar un cubo.

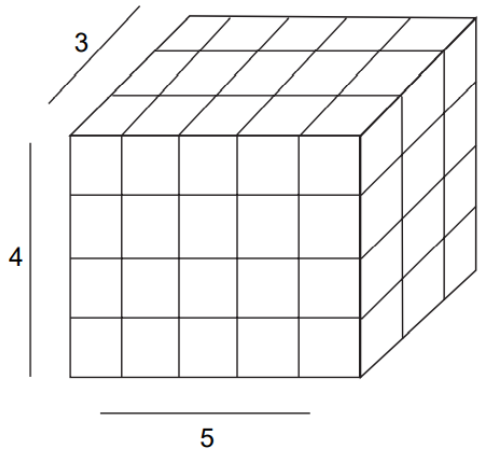


Figura 7.11. Un array de tres dimensiones (4 × 5 × 3).

Un array tridimensional se puede considerar como un conjunto de arrays bidimensionales combinados juntos para formar, en profundidad, una tercera dimensión. El cubo se construye con filas (dimensión vertical), columnas (dimensión horizontal) y planos (dimensión en profundidad). Por consiguiente, un elemento dado se localiza especificando su plano, fila y columna.

```
int equipos[3][15][10];
```

Una aplicación práctica

El array libro tiene tres dimensiones [PAGINAS] [LINEAS] [COLUMNAS], que definen el tamaño del array. El tipo de datos del array es char, ya que los elementos son caracteres. ¿Cómo se puede acceder a la información del libro? El método más fácil es mediante bucles anidados. Dado que el libro se compone de un conjunto de páginas, el bucle más externo será el bucle de página; y el bucle de columnas el bucle más interno. Esto significa que el bucle de filas se insertará entre los bucles página y columna. El código siguiente permite procesar el array.

```
for (int Pagina = 0; Pagina < PAGINAS; ++Pagina)
   for (int Linea = 0; Linea < LINEAS; ++Linea)
      for (int Columna = 0; Columna < COLUMNAS; ++Columna)
         <procesar Libro[Pagina][Linea][Columna]>
```