



República Bolivariana de Venezuela
Aldea Universitaria Liceo Fray Pedro de Agreda

Prof. Elías Cisneros (cisneros.elias@gmail.com)




Lenguaje C++
Arreglos unidimensionales y multidimensionales



Usted es libre de:

- Copiar, distribuir y comunicar públicamente la obra .
- Hacer obras derivadas .

Bajo las condiciones siguientes:

	Reconocimiento. Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).
	No comercial. No puede utilizar esta obra para fines comerciales.
	Compartir bajo la misma licencia. Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.



Contenido

Arreglos.....	1
Arreglos unidimensionales.....	3
Asignación entre arreglos.....	4
Arreglos multidimensionales	6
Algunas operaciones con arreglos.....	9
Operaciones con arreglos unidimensionales.....	11
Operaciones con arreglos multidimensionales	11
Arreglos de caracteres multidimensionales.....	14
Inicialización de arreglos en C++	16
Ejercicios propuestos.....	18
Bibliografía.....	21

Arreglos

Un arreglo o array es una colección de variables relacionadas a las que se hace referencia por medio de un nombre común. Los arreglos pueden tener una o varias dimensiones. En el lenguaje C++ un arreglo se le conoce como un tipo de dato compuesto. Los arreglos pueden tener una o varias dimensiones.

float arreglo[6]; Representación gráfica de
 un arreglo de **una**
 dimensión

1	arreglo[0]
2	arreglo[1]
3	arreglo[2]
4	arreglo[3]
5	arreglo[4]
6	arreglo[5]



Int array[4][4]

Representación gráfica de un arreglo de **dos** dimensiones

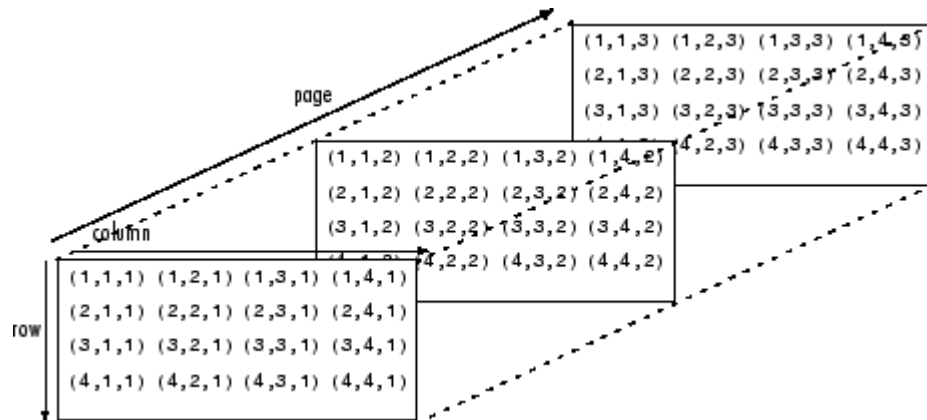
A: array [0..3,0..3] of char.

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

Memory

15	A[3,3]
14	A[3,2]
13	A[3,1]
12	A[3,0]
11	A[2,3]
10	A[2,2]
9	A[2,1]
8	A[2,0]
7	A[1,3]
6	A[1,2]
5	A[1,1]
4	A[1,0]
3	A[0,3]
2	A[0,2]
1	A[0,1]
0	A[0,0]

Int array[4][4][3]
Representación gráfica de un arreglo de **tres** dimensiones

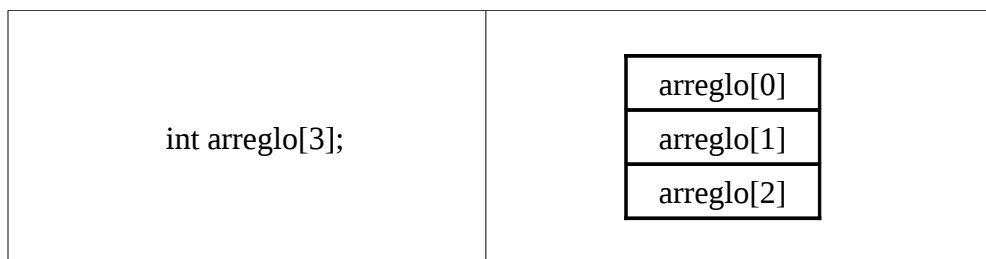




Arreglos unidimensionales

Un arreglo de una dimensión es una lista de variables, todas de un mismo tipo a las que se hace referencia por medio de un nombre común. Una variable individual del arreglo se llama elemento del arreglo. Para declarar un arreglo de una sola dimensión se usa el formato general:

```
tipo_dato identificador[tamaño];
```



Un elemento del arreglo se accede indexando el arreglo por medio de un número del elemento. En C++ todos los arreglos empiezan en 0, esto quiere decir que si se desea acceder al primer elemento del arreglo debe usar el índice igual a 0. Para indexar un arreglo se especifica el índice del elemento que interesa dentro de un corchete, ejemplo;

```
valor = arreglo[1];
```

Recuerde que los arreglos empiezan en 0, de manera que el índice 1 se refiere al segundo elemento. Para asignar el valor a un elemento de un arreglo, ponga el elemento en el lado izquierdo de una sentencia de asignación.

```
mi_arreglo[0] = 100;
```

C++ almacena arreglos de una sola dimensión en una localización de memoria contigua con el primer elemento en la posición más baja. De esta manera, `mi_arreglo[0]` es adyacente a `mi_arreglo[1]`, que es adyacente a `mi_arreglo[2]` y así sucesivamente. Puede usar el valor de un elemento de un arreglo donde quiera que usaría una variable sencilla o una constante.

Ejemplo 1. Arreglo de una dimensión

Declaración

```
int arreglo[3]; // forma un arreglo de una dimensión y de tres elementos
```

Nombre del arreglo

```
arreglo
```

Nombre de los elementos

```
arreglo[0] → primer elemento
```



arreglo[1] → segundo elemento
arreglo[2] → tercer elemento

Ejemplo 2, el siguiente programa carga el arreglo sqrs con los cuadrados de los números del 1 al 10 y luego los visualiza.

```
1. using namespace std;
2. #include <iostream>
3. int main()
4. {
5.     int sqrs[10];
6.     int i;
7.     for (i=1;i<11;i++) {
8.         sqrs[i-1]=i*i;
9.     }
10.    for (i=0;i<10;i++) {
11.        cout<<sqrs[i]<<endl;
12.    }
13.    return 0;
14. }
```

La forma como se almacenan los valores en el arreglo es la siguiente:

```
sqrs[0] = 1*1
sqrs[1] = 2*2
sqrs[2] = 3*3
sqrs[3] = 4*4
sqrs[4] = 5*5
sqrs[5] = 6*6
sqrs[6] = 7*7
sqrs[7] = 8*8
sqrs[8] = 9*9
sqrs[9] = 10*10
```

Asignación entre arreglos

En C++ no se puede asignar un arreglo completo a otro arreglo. Por ejemplo, este fragmento es incorrecto.

```
char a1[10], a2[10];
.
.
.
a2=a1; // Es incorrecto
```

Si desea copiar los valores de todos los elementos de un arreglo a otro debe hacerlo copiando cada elemento por separado. Por ejemplo, el siguiente programa carga a1 con los números 1 a 10 y después los copia en a2. **Ejemplo 3**.



<pre>1. using namespace std; 2. #include <iostream> 3. int main() 4. { 5. int a1[10], a2[10]; 6. int i; 7. //Inicialización de a1 8. for (i=0; i<10;i++) 9. a1[i]=i+1; 10. //Copiar en a2 11. for (i=0; i<10;i++) 12. a2[i]=a1[i] 13. //Mostrar a2 14. for (i=0; i<10;i++) 15. cout<<a2[i]<<endl; 16. return 0; 17. }</pre>	<p>La forma como quedarán los valores en el arreglo es la siguiente:</p> <pre>a1[0] = a2[0] a1[1] = a2[1] a1[2] = a2[2] a1[3] = a2[3] a1[4] = a2[4] a1[5] = a2[5] a1[6] = a2[6] a1[7] = a2[7] a1[8] = a2[8] a1[9] = a2[9]</pre>
--	---

Los arreglos prestan mucha utilidad cuando es necesario manejar lista de información. Por ejemplo, este programa lee la temperatura al mediodía, durante todos los días de un mes y luego informar la temperatura promedio mensual así como el día más caluroso y el más frío. **Ejemplo 4.**

```
1. using namespace std;
2. #include <iostream>
3. int main()
4. {
5.     int temp[31],min, max, media;
6.     int dias;
7.     cout<<"Cuantos días tiene el mes"<<endl;
8.     cin>>dias;
9.     for(int i=0;i<dias;i++){
10.     cout<<"Introduzca la temperatura de mediodía del día"<<i+1<<":"<<endl;
11.     cin>>temp[i];
12. }
13. // Hallar la media
14. media=0;
15. for(int i=0;i<dias;i++){
16.     media=media+temp[i];
17. }
```



```

18. cout<<"Temperatura media: "<<media/dias<<endl;
19. //Hallar min y max
20. min=60;// Temperatura minima de la tierra es -90 Grados centígrados
21. max=-90; // Temperatura máxima de la tierra es 60 Grados centígrados
22. for(int i=0;i<dias;i++){
23. if(min>temp[i]) min=temp[i];
24. if(max<temp[i]) max=temp[i];
25. }
26. cout<<"Temperatura mínima: "<<min<<endl;
27. cout<<"Temperatura máxima: "<<max<<endl;
28. return 0;
29. }

```

Arreglos multidimensionales

Es una estructura de datos estática y de un mismo tipo de datos, y de longitud fija que almacena datos de forma matricial. El almacenamiento de los datos en la memoria se realiza de forma secuencial y son accedidos mediante índices. Los arreglos multidimensionales son también conocidos como matrices.

Se llama matriz de **orden** "m×n" a un conjunto rectangular de elementos dispuestos en filas "m" y en columnas "n", siendo m y n números naturales. Una matriz de orden 3x4 se muestra a continuación.

M3x4

Las matrices se denotan con letras mayúsculas: A, B, C, ... y los elementos de las mismas con letras minúsculas y subíndices que indican el lugar ocupado: a, b, c, ... Un elemento genérico que ocupe la fila i y la columna j se escribe i,j. Si el elemento genérico aparece entre paréntesis también representa a toda la matriz: A (i,j). Por ejemplo, siendo M una matriz de 3 filas y 4 columnas, la representación gráfica de sus posiciones sería la siguiente:

M 3x4

Filas = 3, columnas = 4

		<i>columnas</i>			
		<i>c0</i>	<i>c1</i>	<i>c2</i>	<i>c3</i>
<i>filas</i>	<i>f0</i>	<i>m[f0,c0]</i>	<i>m[f0,c1]</i>	<i>m[f0,c2]</i>	<i>m[f0,c3]</i>
	<i>f1</i>	<i>m[f1,c0]</i>	<i>m[f1,c1]</i>	<i>m[f1,c2]</i>	<i>m[f1,c3]</i>
	<i>f2</i>	<i>m[f2,c0]</i>	<i>m[f2,c1]</i>	<i>m[f2,c2]</i>	<i>m[f2,c3]</i>



Matrices cuadradas

Una matriz cuadrada es una matriz que tiene el mismo número de filas y columnas. La matriz que se muestra a continuación es de orden 3x3.

$$\begin{pmatrix} 1 & -3 & 8 \\ 2 & 0 & 0 \\ 0 & 1 & -1 \end{pmatrix}$$

Declaración de arreglos multidimensionales

La sintaxis es la siguiente:

tipo_dato identificador [dimensión1] [dimensión2] ... [dimensiónN] ; Donde N es un número natural positivo.

Ejemplo Arreglo de dos dimensiones de orden 2x3.

```
char m[2][3];
```

	<i>c0</i>	<i>c1</i>	<i>c2</i>
<i>f0</i>	a	x	w
<i>f1</i>	b	y	10

Declaración

```
char m[2][3]; // forma una tabla de dos filas y tres columnas  
// cada fila es un arreglo de una dimensión  
// la declaración indica que hay dos arreglos de una dimensión
```

Nombre del grupo

m → indica la localización del grupo en la memoria

Nombre de las filas

m[0] → primera fila → indica la localización de la fila dentro del grupo

m[1] → segunda fila → indica la localización de la fila dentro

--	--	--



del grupo



Nombre de los elementos

m[0][0] → primer elemento
m[0][1] → segundo elemento
m[0][2] → tercer elemento
m[1][0] → cuarto elemento
m[1][1] → quinto elemento
m[1][2] → sexto elemento

m[0][0]	m[0][1]	m[0][2]
m[1][0]	m[1][1]	m[1][2]

Haciendo referencia a algunos elementos obtendríamos lo siguiente:

- m[0][0] = a
- m[1][1] = y
- m[1][2] = 10
- m[0][2] = w

Ejemplo 5. Llenado de un arreglo de enteros de dimensión 3x2. En este ejemplo el llenado lo realiza el usuario, en otros ejemplos se verá como realizar llenado de matrices mediante asignación automática, cálculos de operaciones, etc.

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.     int matriz [3][2];
6.     int valor;
7.     for(int i=0;i<3;i++)
8.     {
9.         for(int j=0; j<2;j++)
10.        {
11.            cout<<"Ingrese el valor de la matriz en la posicion ["<<i<<","<<j<<"]"<<endl;
12.            cin>>valor;
13.            matriz[i][j] = valor;
14.        }
```



```
15. }
16. // Imprimiendo el arreglo en formato matricial
17. for(int i=0;i<3;i++)
18. {
19.     cout<<"|";
20.     for(int j=0; j<2;j++)
21.     {
22.         cout<<"\t"<<matriz[i][j]<<"\t";
23.     }
24.     cout<<"|"<<endl;
25. }
26. return 0;
27. }
```

Algunas operaciones con arreglos

A continuación un ejemplo de operaciones de llenado automático de dos arreglos multidimensionales o matrices de orden 3x2. La matriz se llena automáticamente, la matriz A se llena con valores de un contador que va desde 0 a hasta 9. La matriz B se llena con un acumulador que inicia desde 10 y va incrementado y almacenando su valor a razón de 3. Adicionalmente se utiliza una opción para indicar al usuario si desea continuar trabajando en el programa. Revise el **Ejemplo 6**:

```
1. /*
2. Autor: Elias Cisneros
3. Fecha: 19-07-2009
4. correo:cisneros.elias@gmail.com
5. */
6. #include <iostream>
7. #include <string.h>
8. using namespace std;
9. int main()
10. {
11.     int matriz_A [3][2],matriz_B [3][2];
12.     int valores_A=0, valores_B=10;
13.     char opcion[2];
14.     int comparacion=0;
15.     cout<<"Bienvenido al programa de Cadenas Multidimensionales (Presione Enter)"<<endl;
16.     getchar();
17.     do
18.     {
19.         valores_A=0, valores_B=10;
20.         //Llenado arreglo A y B
```



```
21.     for(int i=0;i<3;i++)
22.     {
23.         for(int j=0; j<2;j++)
24.         {
25.             matriz_A[i][j] = valores_A;
26.             valores_A     = valores_A+1;
27.             matriz_B[i][j] = valores_B;
28.             valores_B     = valores_B+3;
29.         }
30.     }
31.     // Imprimiendo el arreglo A en formato matricial
32.     cout<<"Matriz A "<<endl;
33.     for(int i=0;i<3;i++)
34.     {
35.         cout<<"|";
36.         for(int j=0; j<2;j++)
37.         {
38.             cout<<"\t"<<matriz_A[i][j]<<"\t";
39.         }
40.         cout<<"|"<<endl;
41.     }
42.     // Imprimiendo el arreglo B en formato matricial
43.     cout<<"Matriz B "<<endl;
44.     for(int i=0;i<3;i++)
45.     {
46.         cout<<"|";
47.         for(int j=0; j<2;j++)
48.         {
49.             cout<<"\t"<<matriz_B[i][j]<<"\t";
50.         }
51.         cout<<"|"<<endl;
52.     }
53.     cout<<"Para continuar el programa presione Si, para salir No: "<<endl;
54.     gets(opcion);
55.     // Transformación de cadena a mayusculas
56.     int tamano =strlen(opcion);
57.     for(int i=0;i<tamano;i++)
58.         opcion[i]= toupper(opcion[i]);
59.     comparacion = strcmp(opcion,"SI");
60.
61.     }while(comparacion==0);
62.     cout<<"Fin"<<endl;
```



```
63. return 0;  
64. }
```

Operaciones con arreglos unidimensionales

Suma y Resta

Los arreglos deben tener el mismo tamaño y la suma se realiza elemento a elemento. Por ejemplo $C = A + B$. Donde A, B y C son arreglos de enteros de tamaño 3.

$$\begin{array}{c} C \\ \hline c00=a00+b00 \\ \hline c01=a01+b01 \\ \hline c02=a02+b02 \\ \hline \end{array} = \begin{array}{c} A \\ \hline a00 \\ \hline a01 \\ \hline a02 \\ \hline \end{array} + \begin{array}{c} B \\ \hline b00 \\ \hline b01 \\ \hline b02 \\ \hline \end{array}$$

```
int A[3],B[3],C[3];
```

```
for (int j=0;j<3;j++)  
{  
    C[j]=A[j]+B[j];  
}
```

Operaciones con arreglos multidimensionales

En matemáticas, una matriz es una tabla de números consistente en cantidades abstractas que pueden sumarse y multiplicarse. Las matrices se utilizan para describir sistemas de ecuaciones lineales, realizar un seguimiento de los coeficientes de una aplicación lineal y registrar los datos que dependen de varios parámetros. Las matrices se describen en el campo de la teoría de matrices. Pueden sumarse, multiplicarse y descomponerse de varias formas, lo que también las hace un concepto clave en el campo del álgebra lineal. Las matrices son utilizadas ampliamente en la computación, por su facilidad y liviandad para manipular información. En este contexto, son la mejor forma para representar grafos, y son muy utilizadas en el cálculo numérico.



$$\begin{bmatrix} 1 & 3 & 2 \\ 1 & 0 & 0 \\ 1 & 2 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 5 \\ 7 & 5 & 0 \\ 2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1+1 & 3+0 & 2+5 \\ 1+7 & 0+5 & 0+0 \\ 1+2 & 2+1 & 2+1 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 7 \\ 8 & 5 & 0 \\ 3 & 3 & 3 \end{bmatrix}$$

Propiedades

* Asociativa

Dadas las matrices $m \times n$ A, B y C

$$A + (B + C) = (A + B) + C$$

* Conmutativa

Dadas las matrices $m \times n$ A y B

$$A + B = B + A$$

* Existencia de matriz cero o matriz nula

$$A + 0 = 0 + A = A$$

* Existencia de matriz opuesta con $-A = [-a_{ij}]$

$$A + (-A) = 0$$

Suma y resta

Los arreglos deben tener el mismo orden y la suma se realiza elemento a elemento. Por ejemplo sean A, B y C arreglos de números punto flotante de orden 2×3 . Entonces la operación $C = A+B$ sería:

```
float A[3][3],B[3][3],C[3][3];
```

$$\begin{array}{ccc}
 C & = & A & + & B \\
 \begin{array}{|c|c|c|} \hline c_{00}=a_{00}+b_{00} & c_{01}=a_{01}+b_{01} & c_{02}=a_{02}+b_{02} \\ \hline c_{10}=a_{10}+b_{10} & c_{11}=a_{11}+b_{11} & c_{12}=a_{12}+b_{12} \\ \hline c_{20}=a_{20}+b_{20} & c_{21}=a_{21}+b_{21} & c_{22}=a_{22}+b_{22} \\ \hline \end{array} & = & \begin{array}{|c|c|c|} \hline a_{00} & a_{01} & a_{02} \\ \hline a_{10} & a_{11} & a_{12} \\ \hline a_{20} & a_{21} & a_{22} \\ \hline \end{array} & + & \begin{array}{|c|c|c|} \hline b_{00} & b_{01} & b_{02} \\ \hline b_{10} & b_{11} & b_{12} \\ \hline b_{20} & b_{21} & b_{22} \\ \hline \end{array}
 \end{array}$$

```
for (int i=0;i<2;++)  
{
```



```

for (int j=0;j<3;j++)
{
    C[i][j]=A[i][j]+B[i][j];
}
}

```

Producto por un escalar

Dada una matriz A y un escalar c, su producto cA se calcula multiplicando el escalar por cada elemento de A (i.e. (cA)[i, j] = cA[i, j]).

Ejemplo

$$2 \begin{bmatrix} 1 & 8 & -3 \\ 4 & -2 & 5 \end{bmatrix} = \begin{bmatrix} 2 \times 1 & 2 \times 8 & 2 \times -3 \\ 2 \times 4 & 2 \times -2 & 2 \times 5 \end{bmatrix} = \begin{bmatrix} 2 & 16 & -6 \\ 8 & -4 & 10 \end{bmatrix}$$

Propiedades

Sean A y B matrices y c y d escalares.

- * Clausura: Si A es matriz y c es escalar, entonces cA es matriz.
- * Asociatividad: (cd)A = c(dA)
- * Elemento Neutro: 1·A = A
- * Distributividad:
 - o De escalar: c(A+B) = cA+cB
 - o De matriz: (c+d)A = cA+dA

Ejemplo de producto de un escalar por una matriz

Realizar la operación M=2*S donde M y S son arreglos de dimensión dos de orden 2x2.

```
float M[2][2], S[2][2];
```

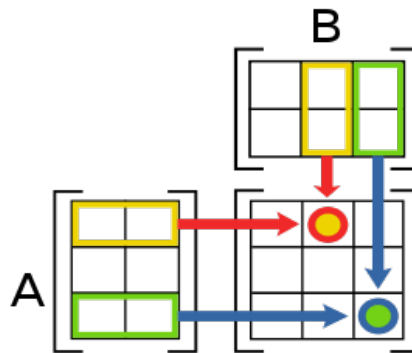
$$\begin{matrix}
 & M & = & 2 & * & S \\
 \begin{bmatrix} m_{00}=2*s_{00} & m_{10}=2*s_{10} \\ m_{01}=2*s_{01} & m_{11}=2*s_{11} \end{bmatrix} & = & 2 & * & \begin{bmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{bmatrix}
 \end{matrix}$$



Producto de matrices

El producto de dos matrices se puede definir sólo si el número de columnas de la matriz izquierda es el mismo que el número de filas de la matriz derecha. Si A es una matriz $m \times n$ y B es una matriz $n \times p$, entonces su producto matricial AB es la matriz $m \times p$ (m filas, p columnas) dada por:

$$(AB)[i,j] = A[i,1] B[1,j] + A[i,2] B[2,j] + \dots + A[i,n] B[n,j] \text{ para cada par } i \text{ y } j.$$



Por ejemplo:

$$\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} (1 \times 3 + 0 \times 2 + 2 \times 1) & (1 \times 1 + 0 \times 1 + 2 \times 0) \\ (-1 \times 3 + 3 \times 2 + 1 \times 1) & (-1 \times 1 + 3 \times 1 + 1 \times 0) \end{bmatrix} = \begin{bmatrix} 5 & 1 \\ 4 & 2 \end{bmatrix}$$

Arreglos de caracteres multidimensionales

Los arreglos de cadenas, que a menudo se conocen como tablas de cadenas son comunes en la programación en C++. Una tabla de cadenas de dos dimensiones es creada como otro cualquier arreglo de dos dimensiones. No obstante, la forma como se conceptualizará será levemente diferente. Por ejemplo:



```
char nombres[10][50]
```

Leidys\0
Henry\0
Luis\0
Alexis\0
José\0
Estrella\0
Alberto\0
Ducyelis\0
Angel\0
Joan\0

Esta sentencia especifica una tabla que puede contener hasta de 10 cadenas, cada una de hasta 50 caracteres de longitud (incluyendo el carácter de fin de cadena). Para acceder a una cadena dentro de esa tabla se especifica solamente el primer índice. Por ejemplo para introducir una cadena desde el teclado en la tercera cadena de nombres, se utilizaría la siguiente sentencia:

```
gets(nombres[2]);
```

De la misma manera, para dar salida a la primera cadena se utilizaría la sentencia

```
cout<<nombres[0];
```

La declaración que sigue crea una tabla de tres dimensiones con tres listas de cadenas. Cada lista tiene cinco cadenas de longitud, y cada cadena puede almacenar 80 caracteres.

```
char animales[2][5][80];
```

Para acceder a una cadena concreta en este caso, debe especificar las primeras dos dimensiones. Por ejemplo, para acceder a la segunda cadena de la tercera lista especifique `animales[2][1]`. En el siguiente ejemplo se utilizan dos listas (nombres y apellidos) con capacidad de almacenar 5 cadenas de 30 caracteres cada una.

Ejemplo 7.

1. `#include <iostream>`
2. `using namespace std;`
3. `int main()`



```
4. {
5.     int cant_estud = 5;
6.     char nombres[5][30], apellidos[5][30];
7.     cout << "\t\tPrograma de demostración de arreglos de cadenas" << endl;
8.     for( int i=0;i<cant_estud;i++)
9.     {
10.        cout<<"Estudiante "<<i+1<<": "<<endl;
11.        cout<<"Ingrese el Nombre: ";
12.        gets(nombres[i]);
13.        cout<<"Ingrese el Apellido : ";
14.        gets(apellidos[i]);
15.        cout<<" "<<endl;
16.    }
17.    cout << "\t\tCarga realizada..." << endl;
18.    for( int i=0;i<cant_estud;i++)
19.    {
20.        cout<<"Estudiante "<<i+1<<" es : "<<nombres[i]<<" "<<apellidos[i]<<endl;
21.    }
22.    return 0;
23. }
```

Inicialización de arreglos en C++

Tal como a otros tipos de variables, puede darle valores iniciales a los elementos de una arreglo. Esto se consigue especificando una lista de valores que tendrán los elementos del arreglo.

```
tipo_dato identificador [tamaño] = {lista de valores};
```

La lista de valores es un conjunto de valores separados por comas que son de un tipo de dato compatible con el tipo de dato del arreglo. La primera constante será colocada en la primera posición del arreglo, la segunda constante en la segunda posición, y así sucesivamente . Por ejemplo:

```
int mem[6] = {1,0,-3,24,15,1};
```

Esto significa que `mem[0]` tendrá el valor de 1, y que `mem[4]` tendrá el valor de 15. Para la inicialización de arreglos de caracteres existen dos maneras. Si el arreglo no tiene una cadena terminada en nulo, simplemente especifique cada carácter usando una lista separada por comas como se muestra a continuación:

```
char letras[3] = {'A','B','C'}
```

Si el arreglo va a contener una cadena, puede inicializar el arreglo usando una cadena encerrada entre comillas, tal como se muestra.

```
char nombre[4] = "ABC";
```

Los arreglos de múltiples dimensiones son inicializados de la misma manera que los de una dimensión.



Ejemplo 8. Operaciones con matrices

Suponga que debe construir un programa para registrar los tiempos de los chequeos de 4 velocistas que compiten para un cupo en la clasificación nacional. Cada velocista corre por un carril, y se deben realizar 3 chequeos por cada velocista. Usted debe registrar el nombre del velocista que corre por cada canal, registrar el tiempo de cada prueba y su tiempo promedio. Tome en cuenta que el nombre de tiene una longitud de máximo 40 caracteres.

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.     char competidores [4][40]; /*Matriz de nombres de competidores*/
6.     float carreras[4][3]; /*Matriz de tiempos en cada vuelta de los competidores*/
7.     float prom_competidores[4]; /*Vector de tiempos promedio de los competidores*/
8.     float acumulador=0;
9.     cout << "Información del tiempo de cada competidor en el chequeo" << endl;
10.    /*Llenando el vector de nombres de participantes*/
11.    for (int i=0;i<4;i++)
12.    {
13.        cout<<"Ingrese el nombre del competidor del carril "<<i+1<<endl;
14.        gets(competidores [i]);
15.    }
16.    /*Llenando la matriz de tiempos en cada chequeo*/
17.    for (int j=0;j<4;j++)
18.    {
19.        acumulador=0;
20.        cout<<"Indique el tiempo del Competidor "<<competidores[j]<<endl;
21.        for(int k=0;k<3;k++)
22.        {
23.            cout<<"En la prueba "<<k+1<<endl;
24.            cin>>carreras[j][k];
25.            acumulador=acumulador+carreras[j][k];
26.        }
27.        /*Guardando el promedio de las dos carreras en el vector*/
28.        prom_competidores[j]=acumulador/3;
29.    }
30.    /*Imprimiendo los resultados*/
31.    for (int j=0;j<4;j++)
32.    {
33.        cout<<"El tiempo del Competidor "<<competidores[j];
34.        for(int k=0;k<3;k++)
35.        {
```



```
36.     cout<<"\t"<<carreras[j][k];
37.
38.     }
39.     cout<<"\tTiempo promedio= "<<prom_competidores[j]<<endl;
40.     }
41.     return 0;
42. }
```



Ejercicios propuestos

1. Sea Par un arreglo de enteros de tamaño 10, realice un programa que cargue de forma automática números de pares en cada una de sus posiciones.
2. Sea Impar un arreglo de enteros de tamaño 10, realice un programa que cargue de forma automática números impares en cada una de sus posiciones.
3. Sea Par_Impar un arreglo de enteros de tamaño 10, realice un programa cargue de forma automática números pares en las posiciones que sean pares e impares donde las posiciones sean impares.
4. Sea un vector A de 5 elementos realice las siguientes operaciones.
 - Los elementos del vector deben ser números negativos pares.
 - Determine cual es el mayor elemento del vector (may_A).
 - Determine cual es el menor elemento del vector (men_B).
5. Desarrolle un programa que realice la siguiente operación. Se tienen dos vectores de 5 elementos numéricos cada uno, realizar la resta del Vector A menos el Vector B y almacenar el resultado en un Vector C ($C = A - B$), determinar el promedio de la suma de los elementos del Vector C (prom_C).
6. Continuando el ejercicio anterior, multiplique los valores que se encuentren en las posiciones impares del vector A (mult_imp_ar_A), realice lo mismo para el vector C (mult_imp_ar_B), determine cual de los 2 valores es mayor.
7. Llenar un Vector A de 6 elementos numéricos y determinar suma de todos sus elementos, promedio de los elementos y cuantos elementos están por encima del promedio. Realice la misma operación para el Vector B = 4A. Imprima el promedio de A y de B, indique cual es el mayor de los promedios.
8. Tiene 3 Vectores de 4 elementos numéricos cada uno, realice la suma del Vector A, B y C y guarde el resultado en un Vector D. Sobre el Vector resultante D sume en la posición 3 del Vector el valor numérico 10. En la primera posición del vector A sume el valor numérico 7, realice la misma operación para el vector B. Determine cual es el mayor elemento en la posición 0 entre los vectores A y B.
9. Dado dos Vectores A y B de 7 elementos numéricos cada uno realice la siguiente operación. Llene el Vector A con números pares comprendidos en el intervalo 100 a 120. El Vector B debe llenarse a partir del último elemento del Vector A (forma decreciente). El resultado es que el primer elemento del vector B debe ser el último elemento del vector A, y el último elemento del



vector B debe ser el primer elemento del vector A. Reste ambos Vectores A-B y almacénelo en el vector R.

10. Construya un algoritmo que realice la siguiente operación. Sean 2 vectores numéricos A y B de 9 elementos numéricos cada uno. El vector A se debe llenar solo con valores pares y el vector B solo con vectores impares. En un vector C guarde el resultado de ejecutar $3A + 2B$. Imprima el vector C en forma descendente. Determine el promedio del vector A y el promedio del Vector B.
11. Construya un algoritmo que realice las siguientes operaciones. Tiene dos vectores numéricos de 100 elementos cada uno. Asegúrese que para cada vector no existan dos elementos consecutivos repetidos. Para el vector A determine el promedio de valores de las posiciones pares del vector; para el vector B determine el promedio de valores de las posiciones impares del vector. Sume los dos vectores guárdelos en un vector C. Imprima C. Imprima el promedio pedido para el vector A y para el vector B.
12. Construya un algoritmo que realice las siguientes operaciones. Tiene 4 vectores A,B,C y D de 10 elementos numéricos cada uno, llene el vector A con números pares, el vector B con números impares, el vector C con la suma de $A + B$, el vector D con la resta de $A - B$. Determinar la suma de cada vector y utilizarlo para llenar el vector R que 4 elementos numéricos, es decir la suma del vector A corresponde a la posición 0 del Vector R, la suma del vector B corresponde a la posición 1 del vector R, y así respectivamente con C y D. Imprima el vector R.
13. Realice un algoritmo para Calcular el valor de PR y IM de :

$$PR = \frac{2}{4} + \frac{4}{8} + \frac{6}{12} + \frac{8}{16} + \frac{10}{20} + \dots +$$

$$IM = \frac{1}{2} + \frac{3}{6} + \frac{5}{10} + \frac{7}{14} + \frac{9}{18} + \dots +$$

14. Utilice estructura de datos tipo Vector para realizar los cálculos respectivos. Dado que el tamaño del vector debe conocerse en tiempo de compilación, inicialice el tamaño del vector en 100. La cantidad de elementos debe ser controlada con un ciclo Haga Mientras, preguntándole al operador si desea generar un nuevo elemento. Imprimir los vectores PR e IM.

Ejemplo para PR

PR[0] = 2/4
PR[1] = 2/4 + 4/8
PR[2] = 2/4 + 4/8 + 6 /12



Arreglos multidimensionales o matrices

15. Sean A , B y C arreglos de orden 2×2 de tipo de datos entero, realice las siguientes operaciones:
- $C = A - B$
 - $C = B - A$
 - $C = 2A + 3B$
16. Sea *nombres* un arreglo de caracteres de orden 10×50 , realice una programa que solicite al usuario la carga de los nombres del personal de la una empresa, la final indique cuantos caracteres contiene cada nombre y cuantos caracteres hay en total en el arreglo *nombres*.
17. Sean A , B y C matrices cuadradas de orden 3×3 de tipo de dato float, realice las siguientes operaciones:
- $C = A - B$
 - $C = B - A$
 - $C = 2A + 3B$
 - $C = 3B$
 - $C = 4A - 3B$
18. Sea una matriz M de orden 5×5 realice las siguientes operaciones:
- Llene la matriz con valores positivos impares.
 - Determine cual es el mayor elemento de la matriz.
 - Determine cual es el menor elemento de la matriz
19. Sean W , X , Y y Z matrices de orden 6×6 , realice la siguiente operación.

$$W = \frac{1}{3}X + \frac{4}{5}Y + 7Z$$

20. Realice la siguiente operación de matrices. Multiplicación de matrices.

$$A \times B = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \cdot \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} a_{00} \cdot b_{00} + a_{01} \cdot b_{10} & a_{00} \cdot b_{01} + a_{01} \cdot b_{11} \\ a_{10} \cdot b_{00} + a_{11} \cdot b_{10} & a_{10} \cdot b_{01} + a_{11} \cdot b_{11} \end{bmatrix}$$

21. Un sensor submarino instalado frente a nuestras costas toma lecturas de la temperatura del agua (grados centígrados) y el nivel ruido (decibelios) todos los meses y los almacena en una matriz de orden 12×2 . Construya un programa que determine mensualmente los valores promedios, mínimos y máximos del sensor.



22. El equipo Leones del Caracas realiza en la temporada regular 50 juegos, almacene en una matriz de orden 50x2 los resultados de cada uno de los juegos. La columna 0 contiene la cantidad de carreras realizadas por Los Leones y la columna 1 contiene las carreras realizadas por su oponente en ese juego. Determine el promedio de carreras anotadas y recibidas durante toda la campaña regular.
23. Se necesita implementar un control de minutos en un centro comunicaciones. Usted dispone de una matriz de orden 300x5. Una llamada local tiene un precio de Bs. 100 por minuto, una llamada a celular Bs. 350 y una llamada internacional Bs. 500 . Al finalizar cada llamada almacene lo siguiente: En la columna 0 se almacenan la cantidad de minutos de la llamada, en la columna 1 se almacena el tipo de llamada (local=1, celular=2, internacional=3), en la columna 2 se coloca la tarifa del tipo de llamada, en la columna 3 se coloca costo de la llamada (minutos*tarifa), en la columna 4 se coloca el monto del impuesto (costo_llamada*iva), en la columna 5 se almacena el costo total de la llamada. Al final del día se necesita generar un reporte con los totales , promedios, mínimos y máximos de cada columna.
24. Construya un programa que permita almacenar los datos obtenidos del sorteo ganadores del Kino Táchira durante los 53 semanas del año 2009. La estructura se debe representar de la siguiente manera:

Sorteo/ Posición	Pos1	Pos2	Pos3	Pos4	Pos5	Pos6	Pos7	Pos8	Pos9	Pos10	Pos11	Pos12	Pos13	Pos14	Pos15
1	1	5	9	10	12	13	15	16	17	19	20	21	23	24	25
2	3	4	5	6	7	12	15	17	18	19	20	22	23	24	25
.															
.															
53	1	2	3	4	5	9	10	11	13	15	18	20	22	23	24
Dato															

Determinar el número que más veces salió en cada posición.

Bibliografía

- Schildt Herbert, *C++ Para programadores*. McGraw-Hill. 1996.
- [http://es.wikipedia.org/wiki/Matriz_\(matem%C3%A1tica\)](http://es.wikipedia.org/wiki/Matriz_(matem%C3%A1tica))