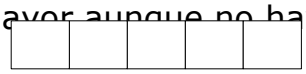
Guía Rápida de Arreglos

Los arreglos son estructuras básicas para resolver muchos de los problemas que podemos encontrar al programar. Un arreglo se puede decir que es una colección de variables del mismo tipo.

Normalmente se los representa con una cuadricula don de cada cuadro es una variable que se puede llenar con un valor (correspondiente al tipo del arreglo).

Los arreglos se dividen en por dimensiones, si los cuadrados forman una línea son de dimensión 1 y se los conoce como vectores, si forman una superficie (por ejemplo un rectángulo) son de dimensión 2 y se los conoce como matrices, si forman un cubo son de dimensión 3, y se pueden hacer de cualquier dimensión mayor aunque no haya representación gráfica.



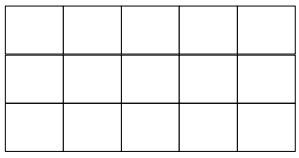
Los Arreglos tienen índices, tantos como dimensiones tenga el arreglo, los de dimensión 1 tienen 1 índice, los de dimensión 2 tienen 2 índices, los de dimensión 3 tienen 3 índices, etc.

Vector 1 dimensión

índice j

Matriz 2 dimensiones 3x5

índice i



índice j

Los índices sirven para ubicar un valor dentro del arreglo, y funcionan como las coordenadas de la batalla naval (en el caso de 2 dimensiones) solo que en vez de letras y número son números, siempre empiezan contando desde 0 y el último valor es el largo de la dimensión correspondiente menos uno (porque empezamos a contar desde cero y no desde uno).

a[1]=5;

Vector 1 dimensión a

0 1 2 3 4

índice j

La declaración de los arreglos se hace en forma idéntica que el de las variables la diferencia es que se le agregan las dimensiones y el largo de elementos por dimensión entre corchetes por ejemplo:

vector de 2 elementos enteros

```
int a[2];
```

matriz de 5x4 elementos float

```
float b[5][4];
```

arreglo de tres dimensiones de 3x4x5 char

```
char c[3][4][5];
```

Asignación de un Arreglo

Los arreglos pueden ser asignados directamente al declararse el arreglo, por ejemplo:

En el caso particular de los char se puede:

```
char nombre[15]="Juan";
```

Que asigna en las posiciones:
0 la 'J'
1 la 'u'
2 la 'a'
3 la 'n'
4 el 0

En forma general se puede asignar usando llaves y separando los elementos por comas:

```
int a[3]={5,3,8};
```

Después de declarar la única manera de asignar valores a un arreglo es asignando cada uno de sus elementos por separado:

```
int a[3];

a[0]=5;
a[1]=3;
a[2]=8;
```

lo que no se puede hacer es:

```
int a[3];
int b[3]={5,3,8};

a=b; //no se puede, se deben asignar elemento por elemento

a[0]=b[0]; //esta es la manera correcta
a[1]=b[1]; //se puede usar un for para que sea más fácil de escribir
a[2]=b[2];
```

Acceso a un elemento del Arreglo

Para acceder a un elemento del arreglo hay que ubicarlo mediante sus índices como si de una coordenada se tratara, hay que recordar que el primer elemento empieza con el valor 0 del índice. Por ejemplo:

primer elemento de a:

a[0]

primer elemento de b:

b[0][0]

primer elemento de c:

c[0][0][0]

El último elemento es la longitud de la dimensión menos uno:

último elemento de a:

a[1] - la longitud era 2.

último elemento de b:

b[4][3] - la longitud del primer índice era 5 y la longitud del segundo índice era 4.

último elemento de c

c[2][3][4] - la longitud del primer índice era 3, la longitud del segundo índice era 4 y la longitud del tercer índice era 5.

Cada elemento del arreglo funciona exactamente igual que una variable del tipo declarado en el arreglo, salvo que debe escribirse individualizando exactamente que elemento del arreglo nos estamos refiriendo en cada caso por medio de los índices.

es válido:

```
a[0]=5;
cout<<a[1];

b[2][1]=4.6;
```

```
cout<<b[3][0];
b[2][3]=b[0][1]+3.2*a[0];

c[1][2][3]='A';
cout<<c[3][0][1];
cin>>c[0][3][1];
```

No es válido:

```
a=7; //no hay forma de saber a cual de los elementos le queremos asignar 7
b[1]=5.0; //sabemos en que línea pero no cual elemento de la línea nos referimos
```

Se pueden usar otras variables como índices (cuidando de no salirnos de los límites)

```
int n,m,o;

n=1; m=2; o=3;
```

es válido:

```
a[0]=5;
cout<<a[n];

b[m][n]=4.6;
cout<<b[3][0];
b[m][o]=b[0][n]+3.2*a[0];

c[n][m][o]='A';
cout<<c[o][0][n];
cin>>c[0][o][n];
```

También pueden ser usados como índices elementos de otros arreglos siempre y cuando estos sean enteros.

```
b[a[1]][0]=4.5; //es válido, el primer índice de b ([a[1]]) es el valor que almacena la segunda posición del vector de enteros a.
```

El poder usar variables nos permite acceder en forma relativa a como vayamos necesitando según corra el programa. Por ejemplo, si tuviéramos la cuadrícula de la batalla naval definida en una matríz de 10x10 en cuyo interior estaría marcada en cada elemento si hay agua o un barco. Y durante el juego, un usuario ingresara unas coordenadas y querría saber si el resultado es agua o tocado. Entonce podríamos hacer algo como lo siguiente:

Nuestro usuario ingresa las coordenadas, supongamos B5.

primero tenemos que transformar nuestras coordenadas a las coordenadas de nuestra matríz:

La primera coordenada está en letras por lo que la A corresponde al 0, la B corresponde al 1 y así sucesivamente hasta la J que corresponde al 9.

```
int x,y;
char c1; //en c1 va la coordenada en letras ej B
int c2; // en c2 va la coordenada en números ej 5
char mapa[10][10]; //representación de la cuadrícula de la batalla naval

x=c1-'A'; //la resta de dos caracteres da la cantidad de elementos que hay entre ellos en este caso 'B'-'A' da 1, para 'A'-'A' da cero, para 'C'-'A' da 2.

y=c2-1; //es necesario restar 1 porque nuestra matriz comienza en 0 y la de la batalla naval en 1.

if(mapa[x][y]=='A') cout<<"Agua"; //suponemos que si hay una A en la matriz es agua

if(mapa[x][y]=='B') cout<<"Tocado"; //suponemos que si hay una B es un barco
```