Arreglos unidimensionales en C

- **Arreglo es un grupo consecutivo de localidades de memoria que tienen el mismo nombre y el mismo tipo.**

Los arreglos se clasifican de acuerdo con el número de dimensiones que tienen. Así se tienen los:

Unidimensionales (vectores)
Bidimensionales (tablas o matrices)
Multidimensionales (tres o más dimensiones)

Sintaxis:

```
<tipo> <identificador>[<núm_elemen>][[<núm_elemen>]...[ ];
```

Ejemplo:

```
int Vector[10];
```

- Creará un arreglo con 10 enteros a los que accederemos como Vector[0] a Vector[9].
- Como subíndice podremos usar cualquier expresión entera. Empiezan con 0.
- En general C++ no verifica el ámbito de los subíndices. Cuidado!
- Los arreglos pueden ser inicializados en la declaración. Por ejemplo:

int n[5] = {0}; *// se asigna 0 a todos los elementos de arreglo*

```
float R[10] = {2, 32, 4.6, 2, 1, 0.5, 3, 8, 0, 12};
float S[] = {2, 32, 4.6, 2, 1, 0.5, 3, 8, 0, 12};
int N[] = {1, 2, 3, 6};
int M[][3] = { 213, 32, 32, 32, 43, 32, 3, 43, 21};
char Mensaje[] = "Error de lectura";
char Saludo[] = {'H', 'o', 'l', 'a'};
int n[ 10 ]={0};
```

// en las líneas 2, 3, 4, 5 y 6. En estos casos la dimensión que queda indefinida se calcula a partir del número de elementos en la lista de valores iniciales.

VENTAJAS de su uso frente a variables aisladas:

- Podemos referirnos a todos los elementos a la vez (al declararlos, al pasarlos a una función, etc).
- Permite resolver problemas de forma sencilla y compacta.

- Utilización de arreglos en las funciones nos permiten hacer cambios en el programa que llama la función sin utilizar el paso de argumentos por referencia.

Ejemplos sencillos de uso de arreglos:

Arreglos1.cpp

Pruebas con el código:

1. cambiar la asignación inicial del vector. Ahora le asignan valor cero a todos sus elementos de alguna otra forma distinta de la actual.
2. averiguar que hace aquí el operador “setw”

Ejercicio: cambiar el código de Arreglos2 para que la inicialización del arreglo sea explicita sin ciclo así: los primeros tres elementos sean 1, 2, 3 y los demás sean ceros.

Ejercicio: Dados los números reales: $a_1,\ldots,a_{15}$**, calcular**

$$\tilde{a}=\frac{1}{15}\sum_{i=1}^{15}a_i$$

$$s=\sqrt{\frac{\sum_{i=1}^{15}(a_i-\tilde{a})^2}{14}}$$

Ejercicio: Hacer un programa a base de algoritmo de ordenamiento (ampliarlo) que al final nos imprima la lista ordenada. El vector de la entrada pueden declararlo directamente en el programa o obtenerlo online utilizando “for” (introduciendo elementos del vector que piensan ordenar uno por uno). Los tipos de las variables están a continuación:

Ordenamiento de los elementos:

```
1. for (i=1; i<TAM; i++)
2.     { for (j=0 ; j<TAM - 1; j++)
3.         { if (lista[ j ] > lista[ j+1])
4.             { temp = lista[ j ];
5.               lista [ j ] = lista[j+1];
6.               lista[ j+1] = temp;
                 }
          }
        }
```

Tabla de variables

Nombre	Tipo	Uso
lista	Cualquiera	Lista a ordenar
TAM	Constante entera	Tamaño de la lista
I	Entero	Contador
J	Entero	Contador
temp	El mismo que los elementos de la lista	Para realizar los intercambios

¿Cómo se ALMACENAN los arrays en MEMORIA?

- Los elementos se almacenan **CONTIGUOS en memoria.**

	`char c[5]`					`int a[5]`				
Elemento	0	1	2	3	4	0	1	2	3	4
Dirección	1000	1001	1002	1003	1004	1000	1002	1004	1006	1008

Suponemos que ambos arrays comienzan en la posición 1000, que un entero ocupa 2 bytes y un char ocupa 1 byte (cada número de posición indica 1 byte).

```
int b[10];
```

- Cada elemento del array `b (b[0], ...,b[9])` es un entero ⇒ puede ser usado en cualquier contexto donde es usado un entero.

- El nombre de un array es un **"puntero constante"** que tiene la dirección del primer elemento del array. Su valor es constante, no se puede cambiar
- Por tanto, `b` es un puntero que contiene la dirección del primer elemento del array `b`, o sea, de `b[0]`.

NOTA: Un **puntero** es un tipo de dato que sirve para almacenar direcciones de memoria.

Ejercicio para la casa: Suponga que se desea desarrollar un programa para:

1. Leer una lista de calificaciones de un examen
2.Encontrar su media
3.Imprimir una lista de las calificaciones mayores que la media
4.Ordenar la lista de las calificaciones en orden descendente (usando el algoritmo de ordenamiento).