ARREGLOS

- Variables comunes sólo almacenan un valor a la vez (variables escalares).
- Cuando identificamos valores que podemos agrupar usamos arreglos (Muchos valores para un sólo nombre).
- Arreglos son variables que contienen varios valores de un mismo tipo y se definen con un mismo nombre.
- Como son variables hay que declararlas.

Ejemplo: notas: 90, 70, 89, 73, 92

voltajes: 1, 1.05, 1, .95, .98, .99, 1

Nota: Existen 2 tipos: - De una dimensión y

- De dos dimensiones.

Arreglos de una dimensión:

- Lista de valores: - Del mismo tipo y

- Con un nombre común.

- Nombre: es el nombre del arreglo.

Ejemplo: En el anterior ejemplo de los voltajes

- Podemos usar volt como su nombre.
- Como hay números reales se debe declarar como float.

float volt [7] = { 1, 1.05, 1, .95, .98, .99, 1 };

Arreglos de dos dimensiones:

- Conocidos como matrices.

- Se componen de filas y columnas.

Ejemplo:

	0	1	2
0	8	16	9
1	3	15	27

→ Columnas

Filas

- 2 Filas → [8 16 9] y [3 15 27]

- 3 Columnas → [8 / 3] , [16 / 15] y [9 / 27]

Para guardar valores en este arreglo hay que especificar la fila y la columna.

Ejemplo: En el anterior ejemplo de los valores

- Podemos usar valor como su nombre.
- Como sólo hay números enteros se debe declarar como int.

int valor [2] [3];

tipo de valor — fila
nombre — columna

Para localizar cada elemento → identificar su fila y columna.

Ejemplo: cout << valor [2] [1];

cout << valor [2] [1]; → Error

cout << valor [1] [2]; → Imprime ?

Inicialización de arreglos de dos dimensiones:

1. Inicialización de elementos independientes:

```
int valor [2] [3];
valor [0] [0] = 8;
valor [0] [1] = 16;
valor [0] [2] = 9;
valor [1] [0] = 3;
valor [1] [1] = 15;
valor [1] [2] = 27;
```

2. Inicialización de elementos al ser declarados:

int valor [2] [3] = { { **8, 16, 9** } , { **3, 15, 27** } };

Otra forma permitida:

int valor [2] [3] = { **8, 16, 9 , 3, 15, 27** };

fila 0 fila 1

Trabajan igual porque:

int valor [2] [3]	0,0	0,1	0,2	1,0	1,1	1,2
	8	16	9	3	15	27

Ejemplo #2:

int valor [3] [3] = { { **8, 16** } , { **9 , 3** }, { **15, 27** } };

int valor [3] [3]	0,0	0,1	0,2	1,0	1,1	1,2	2,0	2,1	2,2
	8	16	0	9	3	0	15	27	0

Ejemplo #3:

int valor [3] [3] = { **8, 16, 9 , 3, 15, 27** };

int valor [3] [3]	0,0	0,1	0,2	1,0	1,1	1,2	2,0	2,1	2,2
	8	16	9	3	15	27	0	0	0

3. Inicialización usando “Loops”:

```
# define F 2
# define C 3
:
:
{
   :
   :
   int valor [F] [C], i, j;
   for (i = 0; i < F; i++)
   {
      for (j = 0; j < C; j++)
      {
         cout << “Entre un valor :\n”;
         cin >> valor [i] [j];
      }
   }
   :
   :
}
```

El **Prototipo** de la función que maneja el arreglo es:

```
int find_max(int [N]);
```

El **llamado** de la función “find_max” sería:

```
find_max (valor);
```

La función (“**header line**”) que recibe el arreglo como parámetros:

```
int find_max (int  arg_val [N]);
```