ARREGLOS.

Los arreglos son un conjunto de localizaciones de memorias contiguas y que almacenan valores del mismo tipo de dato. Los arreglos pueden ser unidimensionales (vectores) bidimensionales (matrices) o multidimensionales.

Para declarar un arreglo se emplea la siguiente sintaxis:

```
<tipo> <nombre>[<número de elementos>][[<número de elementos>]...];
```

Ejemplo:

```
int vector[5];
int matriz[5][6];
int multidimensional[6][4][2];
```

Los valores para el número de elementos deben ser constantes, y se pueden usar tantas dimensiones como queramos, limitado sólo por la memoria disponible.

Los índices que referencian a cada elemento del arreglo comienzan desde el valor cero, por ejemplo: en la declaración **`int vector[5]`** los índices van del **`0`** al **`4`**.

En general C++ no verifica el ámbito de los subíndices. Si declaramos un array de 10 elementos, no obtendremos errores al acceder al elemento 11. Sin embargo, si asignamos valores a elementos fuera del ámbito declarado, estaremos accediendo a zonas de memoria que pueden pertenecer a otras variables o incluso al código ejecutable de nuestro programa, con consecuencias generalmente desastrosas.

Los arrays pueden ser inicializados en la declaración.

Ejemplos:

```
01 float R[10]     = {2, 32, 4.6, 2, 1, 0.5, 3, 8, 0, 12};
02 float S[]       = {2, 32, 4.6, 2, 1, 0.5, 3, 8, 0, 12};
03 int   N[]       = {1, 2, 3, 6};
04 int   M[][3]    = {213, 32, 32, 32, 43, 32, 3, 43, 21};
05 char  Mensaje[] = "Error de lectura";
```

En estos casos no es obligatorio especificar el tamaño para la primera dimensión, como ocurre en los ejemplos de las líneas 2, 3, 4 y 5. En estos casos la dimensión que queda indefinida se calcula a partir del número de elementos en la lista de valores iniciales.

En el caso 2, el número de elementos es 10, ya que hay diez valores en la lista.

En el caso 3, será 4.

En el caso 4, será 3, ya que hay 9 valores, y la segunda dimensión es 3: 9/3=3.

Y en el caso 5, el número de elementos es 17, 16 caracteres más el cero de fin de cadena.

Ejemplo: En ambos casos, el resultado en pantalla es el mismo.

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int main(){
05     int M[][3] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
06     printf("%d %d %d\n",M[0][0], M[0][1], M[0][2]);
07     printf("%d %d %d\n",M[0][3], M[0][4], M[0][5]);
08     printf("%d %d %d\n",M[0][6], M[0][7], M[0][8]);
09     printf("\n");
10     printf("%d %d %d\n",M[0][0], M[0][1], M[0][2]);
11     printf("%d %d %d\n",M[1][0], M[1][1], M[1][2]);
12     printf("%d %d %d\n",M[2][0], M[2][1], M[2][2]);
13 }
```

El siguiente ejemplo muestra cómo se puede ingresar valores por teclado y almacenarlos en un vector:

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int main(){
05   int vector[5];
06   int punt;
07   system("CLS");
08   for(punt = 0; punt < 5; punt++){
09       printf("ingrese un numero para vector[%d]: ", punt);
10       scanf("%d", &vector[punt]);
11   }
12   printf("\n");
13   for(punt = 0; punt < 5; punt++){
14       printf("ingrese un numero para vector[%d]: %d\n", punt, vector[punt]);
15   }
16 }
```