Práctica 6: Arrays Unidimensionales en C++.

En los temas anteriores se han estudiado los diferentes tipos de datos simples de C++, usados para representar valores simples como enteros, reales o caracteres. Sin embargo, en muchas situaciones se precisa procesar una colección de valores que están relacionados entre sí, como por ejemplo una lista de calificaciones, una tabla de temperaturas, etc. El procesamiento de tales conjuntos de datos utilizando tipos simples puede ser extremadamente complejo y tedioso, y por ello la mayoría de los lenguajes de programación incluyen mecanismos sintácticos para manipular agrupaciones de datos, son las llamadas "*estructuras de datos*". La estructura de datos más básica soportada por los lenguajes es el "***array***". Un array se puede visualizar como una colección de cajas que representan variables de un mismo tipo de datos, tal y como se ilustra en la siguiente figura. A este tipo de array se le suele denominar **array unidimensional o vector**.

Para hacer referencia a una caja o componente en particular, usaremos el nombre o identificador del array seguido por un número (índice) entre corchetes, que indicará su posición dentro del array. Realmente lo que va entre corchetes al referirse a la posición de un componente de un array, puede ser cualquier expresión que al evaluarse dé como resultado un valor numérico entero positivo.

Un rasgo importante de los arrays en C++ es que el tipo índice es siempre un subrango de naturales empezando por 0, por lo que a la hora de definir un array solo será necesario indicar su dimensión. Así definimos una variable de un tipo array como:

Donde "*tipo*" es el tipo base **tipo nombre_array [dimensión];** de los componentes del array y "*dimensión*" es un número natural mayor de cero que representa la longitud o dimensión del array. Ejemplos:

```
int  vector [10];                  // Array de 10 enteros
bool logicos [50];                 // Un array de 50 lógicos
```

Aunque lo más adecuado, tal como se ha comentado en el programa de ejemplo del apartado anterior, es usar **constantes globales** para definir la dimensión de los arrays.

```
const int MAXVECTOR = 10;
const int MAXLOGICOS = 50;

int  vector [MAXVECTOR];     // Array de 10 enteros
bool logicos [MAXLOGICOS];   // Array de 50 lógicos
```

Aunque como hemos visto se pueden definir variables de array al mismo tiempo que se define su tipo base y su dimensión (definición implícita de tipos array), lo correcto es definir separadamente los tipos de datos, asignándoles un nombre, y posteriormente declarar las variables como pertenecientes a los tipos anteriormente definidos. C++ incorpora un mecanismo para definir nuevos tipos de datos, la sentencia ***typedef***. Sintaxis:

```
typedef tipo_conocido nombre_nuevo_tipo;
```

Ejemplo: Declarar variables de tipo TContador:

```
typedef unsigned int Tcontador;
Tcontador i, j;
```

El objetivo fundamental de esta sentencia es mejorar la claridad del código fuente asociando nombres simbólicos a tipos de datos definidos por el programador. Según esto, en el caso de definición de tipos array la sintaxis concreta sería entonces:

```
typedef tipo_base nombre_nuevo_tipo [dimensión];
```

Ejemplos:

```
const unsigned int MAXVECTOR = 4;
const unsigned int MAXLISTA  = 7;
typedef int Tvector [MAXVECTOR];
typedef unsigned int Tlista [MAXLISTA];
```

Posteriormente se pueden definir variables de estos tipos:

```
Tvector     v;
Tlista      mi_lista;
```

Nota: Sólo se puede definir un único tipo en cada sentencia ***typedef***.

ENUNCIADO DE LA PRÁCTICA

Deseamos realizar un programa para la realización de operaciones con vectores de R^5 (5 elementos reales). El programa mostrará un menú con las siguientes opciones.

```
     MENU
     ====
Elaborado Por : Nombre Apellidos
E.T.S.I Informatica 1ºA Gestion
Fecha: 16 de Diciembre de 2.003

A. Suma Vectores.
B. Resta Vectores.
C. Producto Escalar de Vectores.
X. Salir del Programa
```

Descripción de Opciones:

A. `Suma Vectores`. Se leeran 2 vectores y se mostrará por pantalla el vector suma. Ejemplo:

```
  Introduzca su opcion: a

Introduzca los Valores del Vector 1
v[0] : 1
v[1] : 2
v[2] : 3
v[3] : 4
v[4] : 5

Introduzca los Valores del Vector 2
v[0] : 1
v[1] : 1
v[2] : 1
v[3] : 1
v[4] : 1

El Vector suma es: v = (2, 3, 4, 5, 6)

Presione una tecla para continuar . . .
```

B. `Resta Vectores`. Se leeran 2 vectores y se mostrará por pantalla el vector resta. Ejemplo:

```
  Introduzca su opcion: b

Introduzca los Valores del Vector 1
v[0] : 1
v[1] : 2
v[2] : 3
v[3] : 4
v[4] : 5

Introduzca los Valores del Vector 2
v[0] : 1
v[1] : 1
```

```
v[2] : 1
v[3] : 1
v[4] : 1

El Vector resta es: v = (0, 1, 2, 3, 4)


Presione una tecla para continuar . . .
```

C. Producto Escalar de Vectores. Se leeran 2 vectores y se mostrará por su producto escalar. Ejemplo:

```
  Introduzca su opcion: c

Introduzca los Valores del Vector 1
v[0] : 1
v[1] : 2
v[2] : 3
v[3] : 4
v[4] : 5

Introduzca los Valores del Vector 2
v[0] : 1
v[1] : 1
v[2] : 1
v[3] : 1
v[4] : 1

El Producto escalar es: 15

Presione una tecla para continuar . . .
```

X. Salir del Programa. Se solicita confirmación y sólo en caso de sea afirmativa se sale del programa. Ejemplo:

```
  Introduzca su opcion: x
¿Esta seguro de salir (S/N)?s


Presione una tecla para continuar . . .
```