Arrays

Un array (también conocido como *arreglo* o *vector*) es un modo de manejar una gran cantidad de datos del mismo tipo bajo un mismo nombre o identificador. Por ejemplo, mediante la sentencia:

```
double a[10];
```

Se reserva espacio para 10 variables de tipo *double*. Las 10 variables se llaman *a* y se accede a una u otra por medio de un subíndice, que es una expresión entera escrita a continuación del nombre entre corchetes [...]. La forma general de la declaración de un vector es la siguiente:

```
tipo nombre[numero_elementos];
```

Los elementos se numeran desde 0 hasta (numero_elementos-1). El tamaño de un vector puede definirse con cualquier tipo de expresión de constante entera, para definir los tamaños es útil recurrir a las constantes simbólicas.

Ejemplos de declaración:

```
const int TAMANIO_ARRAY = 10;
float vFlotantes[TAMANIO_ARRAY];
int vEnteros[100];
char vCaracteres[TAMANIO_ARRAY + 10];
```

Como ya se ha dicho, para acceder a un elemento del vector debe incluirse en la expresión el nombre del mismo seguido del subíndice entre corchetes. Los elementos del vector se utilizan en C como cualquier otra variable.

Ejemplos de uso:

```
vFlotantes[5] = 0.8;
vFlotantes[9] = vFlotantes[5] * 0.2;
vFlotantes[0] = 3 * vFlotantes[5] – (vFlotantes[9] / vFlotantes[8]);
vEnteros[0] = (int) vFlotantes[0];
vEnteros[1] = 100;
vCaracteres[0] = ‘a’;
```

Cadena de caracteres

Una cadena de caracteres (también conocido como *string*) no es más que un array de tipo de dato *char*. Con ciertas particularidades importantes, las cadenas suelen contener (texto, frases, nombres, etc.). El primer caracter de la cadena se almacena desde el primer elemento del array y continúa en las posiciones contiguas hasta el último carácter adicionándole uno más a la cadena, el *caracter terminador*.
Para separar la parte del array que contiene texto de la parte no utilizada del array, se utiliza un *caracter terminador* o *caracter de fin de texto* que es el caracter núlo (‘\0’) según el código ASCII. Éste se introduce al leer o inicializar las cadenas de caracteres.
En el siguiente código:

```
char ciudad[30] = "General Pacheco";
```

Se declara un array del tipo de dato char de 30 elementos, en memoria esto equivale a 30 bytes (30 elementos x 1 byte por elemento). A partir del caracter número 15 se añade un decimosexto que es el ‘\0’, el resto del espacio reservado hasta la posición ciudad[29] no es utilizado.

Luego:

```
char nombre[10] = "Mar";
```

equivale a:

```
char nombre[10];
nombre[0] = 'M';
nombre[1] = 'a';
nombre[2] = 'r';
nombre[3] = '\0';
```

Pasaje de arrays a funciones

Si deseamos pasar un array a una función, debemos hacerlo utilizando *punteros*. En el caso de los arrays se pasa la dirección de memoria del primer elemento.

El nombre de un array equivale a la dirección de memoria del primer elemento del array

Por ejemplo, si tenemos un array de enteros de 5 elementos llamado LineasColectivos con los datos: 60, 15, 152, 71, 203. *LineasColectivos* es la dirección de memoria al primer elemento. Utilizando el operador de indirección (*) accedemos al contenido.
Entonces:

```
LineasColectivos → 0x22fef0
&LineasColectivos[0] → 0x22fef0
*LineasColectivos → 60
LineasColectivos[0] → 60
```

Como vimos anteriormente al pasar una variable a una función por dirección, se recibe la misma dentro de la función declarando un puntero. De la misma forma se realiza el pasaje de arrays a funciones.

Ejemplo:

```
#include <iostream>
using namespace std;

void pasajeVecEnteros(int *);
void pasajeVecFlotante(float *);

int main(void){
   int v1[5];
   float v2[3];
   v1[0]= 10000;
   v2[0] = 0.0908;
   pasajeVecEnteros(v1);
   cout << endl;
   pasajeVecFlotante(v2);
   system("pause >nul");
   return 1;
}

void pasajeVecEnteros(int *VectorEntero){
   cout << "pasajeVecEnteros) Valor del primer elemento: " << VectorEntero[0];
```

```
}
void pasajeVecFlotante(float *VectorFloat){
   cout << "pasajeVecFlotante) Valor del primer elemento: " << VectorFloat[0];
}
```

Salida:

```
pasajeVecEnteros) Valor del primer elemento: 10000
pasajeVecFlotante) Valor del primer elemento: 0.0908
```

Errores comunes al pasar un array a una función:

```
int v[10];
funcion_ejemplo_1(&v); //Error
funcion_ejemplo_2(v[0]); //Envía el valor del primer elemento
```

Errores comunes al recibir un array en una función:

```
void funcion_ejemplo_3(int v){
        v[0] = 10; //Error porque v es una variable del tipo int y no un array de tipo int.
}

void funcion_ejemplo_4(int *v[10]){
        v[0] = 10; //Error
}
```

Ejemplo práctico

Una empresa dispone de 5 productos para la venta, cada producto tiene un código identificatorio (número entre 1000 y 9000) y precio. Se dispone de las ventas realizadas en un día, por cada venta se registra Código de producto, cantidad vendida y precio. Con la información proporcionada se solicita:

1. Por cada producto listar el importe total vendido en el día.
2. Los productos que no registraron ventas.

Ver archivo adjunto T07CF01.cpp

Análisis:

- vProd del tipo de dato int para ingresar los códigos de productos. (Noten que los códigos de producto pueden oscilar entre 1000 y 9000 por lo que no se podrá utilizar un for o codigoproducto – 1 para poder ubicar el valor en el array).
- vPrecio del tipo de dato float para ingresar los precios de los productos. (El siguiente array funciona en conjunto con el array vProd, de modo que lo que contenga vPrecio[3] pertenecerá al código de producto que figure en vProd[3], de esta manera se puede simular una tabla o matriz utilizando varios arrays en conjunto).
- vVentas del tipo de dato float para acumular las ventas. (Ídem aclaración vPrecio).
- Los productos vendidos tendrán la suma de los valores en su respectiva posición del array vVentas, por lo tanto será necesario una función ponerCero para inicializar dicho array con todos sus elementos en cero. Luego de procesar datos se puede llegar a la conclusión de que si alguna de las posiciones de este array permaneció en cero, la misma posición en el array vProd será el producto que no se vendió.
- Como los códigos de producto son valores entre 1000 y 9000, no se podrá restar 1 al código de producto para ubicar los datos en el array (salvo que declaremos un array de

9000 posiciones). Por lo tanto, será necesaria una función buscarProducto para que busque el código de producto y devuelva la posición o simplemente devuelva -1 si no puede encontrarlo (esto es útil a la hora de validar datos).

- Como vProd, vPrecio y vVentas se encuentran en "paralelo" todos tienen que tener el mismo tamaño, por lo tanto se utiliza una constante simbólica llamada PRODUCTOS.

Tabla de datos de prueba del programa T07CF01.cpp

Valor de los arrays		
vProductos	vPrecio	vVentas
1000	30	150
2000	40	240
5000	10,5	52,5
7000	90	0
8000	15	0

Registro de ventas		
Producto	Cantidad Vend	Importe
1000	5	150
2000	2	80
2000	4	160
5000	2	21
5000	3	31,5

Bibliografía:

- Zhang, Tony, Aprendiendo C en 24 horas, Pearson Educación, México, 2001, ISBN: 968-444-495-8.