Arreglos

Vectores y Matrices en C

Arreglos Unidimensionales

- Los arreglos son una colección de variables del mismo tipo que se referencian utilizando un nombre común. Un arreglo consta de posiciones de memoria contigua. La dirección más baja corresponde al primer elemento y la más alta al último. Un arreglo puede tener una o varias dimensiones. Para acceder a un elemento en particular de un arreglo se usa un índice.

Arreglos Unidimensionales

- El formato para declarar un arreglo unidimensional es:

```
<tipo> nombre_arreglo[tamaño]
```

- Por ejemplo, para declarar un arreglo de enteros llamado *lista_numeros* con diez elementos se hace de la siguiente forma:

```
int lista_numeros[10];
```

Arreglos Unidimensionales

- En C, todos los arreglos usan cero como índice para el primer elemento. Por tanto, el ejemplo anterior declara un arreglo de enteros con diez elementos desde **lista_numeros[0]** hasta **lista_numeros[9]**.

- La forma como pueden ser accesados los elementos de un arreglo, es de la siguiente forma:

```
lista_numeros[2] = 15; /* Asigna 15 al 3er elemento del
   arreglo listanum*/
num = lista_numeros[2]; /* Asigna el contenido del 3er
   elemento a la variable num */
```

Arreglos Multidimensionales (matrices)

- C permite arreglos con más de una dimensión , el formato general es:

```
<tipo> nombre_arr [tam_d1][ tam_d2 ] ... [ tam_dN];
```

- Por ejemplo un arreglo de enteros bidimensionales se escribirá como:

```
int tabla_de_nums[5][5];
```

Observar que para declarar cada dimensión lleva sus propios paréntesis cuadrados.

Arreglos Multidimensionales (matrices)

- Para acceder los elementos se procede de forma similar al ejemplo del arreglo unidimensional, esto es,

```
tabladenums[2][3] = 15; /* Asigna 15 al elemento de la 3ª fila y
    la 4ª columna*/
num = tabladenums[4][3];
```

Inicialización de Arreglos Unidimensionales y multidimensionales

- En C se permite la inicialización de arreglos, debiendo seguir el siguiente formato:

```
<tipo> nombre_arr[ tam1 ][ tam2 ] ... [ tamN] =
    {<lista-valores>};
```

Por ejemplo:

```
int numeros[10] = {1,2,3,4,5,6,7,8,9,10};
int num[3][4]={0,1,2,3,4,5,6,7,8,9,10,11};
int num[3][4]={{0,1,2,3},{4,5,6,7},{8,9,10,11}};
```

Las últimas dos inicializaciones son similares

Strings (Cadenas)

- A diferencia de otros lenguajes de programación que emplean un tipo denominado *string* para manipular un conjunto de símbolos, en C, se debe simular mediante un arreglo de caracteres, en donde la terminación de la cadena se debe indicar con nulo.

- Un nulo se especifica como '\0'. Por lo anterior, cuando se declare un arreglo de caracteres se debe considerar un carácter adicional a la cadena más larga que se vaya a guardar.

Strings

- Por ejemplo, si se quiere declarar un arreglo *string* que guarde el *string* “Hola” (4 caracteres), se hará como:

```
char cadena[5];
```

Strings

- Se pueden hacer también inicializaciones de arreglos de caracteres en donde automáticamente C asigna el carácter nulo al final del string, de la siguiente forma:

```
char nombre[tam]="cadena";
```

Strings

Por ejemplo, el siguiente fragmento inicializa un string con “hola”:

```
char string[5]="hola";
```

El código anterior es equivalente a:

```
char cadena[5]={'h','o','l','a','\0'};
```

Strings

- Para asignar la entrada estándar a una cadena se puede usar la función scanf con la opción `%s` (observar que no se requiere usar el operador `&`), de igual forma para mostrarlo en la salida estándar.

- Por ejemplo:

```
main() {
 char nombre[15], apellidos[30];
    printf("Introduce tu nombre: ");
    scanf("%s",nombre);
    printf("Introduce tus apellidos: ");
    scanf("%s",apellidos);
    printf("Usted es %s %s\n",nombre,apellidos);
 }
```

Strings

- El lenguaje C no tiene un tipo de dato string, por lo que no se pueden manipular strings directamente como se hace con enteros o flotantes, por lo que lo siguiente no es válido:

```
main() {
 char nombre[40], apellidos[40], completo[80];
   nombre="Jose Manuel";                    /* Ilegal */
   apellidos="Gonzalez Tapia";              /* Ilegal */
   completo="Gral."+nombre+apellidos; /* Ilegal */
}
```

Ejercicios propuestos

1. Inicializar un arreglo de 10 posiciones con el valor 5.
2. Llenar un arreglo de 5 posiciones con valores dados por el usuario.
3. Sumar el contenido de dos arreglos y el resultado dejarlo en un 3er arreglo.

Inicializar un arreglo de 10 posiciones con el valor 5.

```
//programa que inicializa un vector con 5
#include <stdio.h>
#include <stdlib.h>
void main(){
   int vector[10],i,j;
   for(i=0 ; i<10 ; i++)
      vector[i]=5;
   for(j=0 ; j<10 ; j++)
      printf("vector[%d]=%d\n",j,vector[j]);
}
```

Llenar un arreglo de 5 posiciones por entrada del usuario.

```
//programa que llena un arreglo con valores de usuario
#include <stdio.h>
#include <stdlib.h>
void main(){
    int arreglo[5],i,j,numero;
    for(i=1 ; i<=5 ; i++){
         printf("\ningrese el valor %d para el arreglo:",i);
        scanf("%d",&numero);
        arreglo[i] = numero;
    }
    for(j=1 ; j<=5 ; j++)
         printf("arreglo[%d]=%d\n",j,arreglo[j]);
}
```

Sumar el contenido de dos arreglos y el resultado dejarlo en un 3er arreglo.

```
//sumar el contenido de dos arreglos y dejarlos en un 3ero
#include <stdio.h>
#include <stdlib.h>
void main(){
    int arr1[3]={2,4,6};
    int arr2[3]={1,3,5};
    int arr3[3];
    int i,j,numero;
    for(i=0 ; i<3 ; i++){
        arr3[i] = arr1[i] + arr2[i];
    }
    for(j=0 ; j<3 ; j++)
        printf("arr3[%d]=%d\n",j,arr3[j]);
}
```