APRENDERAPROGRAMAR.COM

ARRAYS (ARREGLOS O FORMACIONES) UNIDIMENSIONALES EN C. DECLARACIÓN. LÍMITES. (CU00512F)

Sección: Cursos
Categoría: Curso básico de programación en lenguaje C desde cero

Fecha revisión: 2031

Resumen: Entrega nº12 del curso básico "Programación C desde cero".

Autor: Mario Rodríguez Rancel

VARIABLES CON INDICE O LOCALIZADOR. ARRAYS (ARREGLOS) EN C

El concepto de array en el lenguaje C coincide con el que se expone en el curso Bases de la programación nivel I de aprenderaprogramar.com cuando se habla de pseudocódigo. Veremos ahora cómo declarar arrays estáticos de una dimensión. La gestión de arrays multidimensionales la veremos más adelante. Los arrays dinámicos no están permitidos en C, aunque hay formas para conseguir almacenar información simulando un array dinámico.

ARRAYS (ARREGLOS) UNIDIMENSIONALES

La sintaxis a emplear es la siguiente:

```
tipoDeElementosDelArray  nombreDelArray  [numeroElementos];
```

Ejemplo: int vectorEnteros [4];

Esto declara que se crea un vector de enteros que contendrá 4 valores de tipo int. Fíjate que el número en la declaración es 4, pero el elemento vectorEnteros[4] no existirá. ¿Por qué? Porque en C, al igual que en otros lenguajes de programación, la numeración de elementos empieza en cero y no en uno. De esta manera al indicar un 4, los índices de elementos en el array serán 0, 1, 2, 3. Es decir, si indicamos 4 el array tendrá 4 elementos (índices 0 a 3). Si indicamos 10 el array tendrá 10 elementos (índices 0 a 9) y así sucesivamente para cualquier número declarado.

Ejemplos de declaración de arrays serían:

```
int vectorVez [9];
char vectorAmigo [1000];
double decimalNum [24];
int vectorInt [23];
int vectorLong[8];
int personasPorHabitacion[300];
```

Crea un proyecto y escribe el siguiente código:

```
#include <stdio.h>
#include <stdlib.h>
// Ejemplo aprenderaprogramar.com
int main() {
int numeroDeCoches [4];
numeroDeCoches[0] = 32;
printf ("El numero de coches en la hora cero fue %d \n", numeroDeCoches[0]);
printf ("El numero de coches en la hora uno fue %d \n", numeroDeCoches[1]);
printf ("El numero de coches en la hora dos fue %d \n", numeroDeCoches[2]);
printf ("El numero de coches en la hora tres fue %d \n", numeroDeCoches[3]);
return 0;
}
```

El resultado de ejecución puede ser similar a este:

```
El numero de coches en la hora cero fue 32
El numero de coches en la hora uno fue 8
El numero de coches en la hora dos fue 100
El numero de coches en la hora tres fue 100
```

¿Por qué ocurre esto? Realmente no siempre ocurrirá esto, podríamos decir que el resultado puede cambiar dependiendo del compilador. Lo que es cierto que es numeroDeCoches[0] vale 32 porque así lo hemos declarado. Sin embargo, es posible que nos aparezcan valores aparentemente aleatorios para el resto de elementos del array porque no los hemos inicializado. El compilador al no tener definido un valor específico para la inicialización les ha asignado unos valores aparentemente aleatorios. Para evitar esta circunstancia tendremos que inicializar todos los elementos de un array. De momento lo haremos manualmente como indicamos a continuación, más adelante veremos cómo hacerlo usando un bucle que nos permitirá inicializar los elementos de un array de decenas o cientos de elementos a un valor por defecto. En el caso de enteros lo más normal es inicializar los elementos que no tienen un valor definido a cero.

En el programa anterior añadiríamos la siguiente línea:

numeroDeCoches[1]=0; numeroDeCoches[2]=0; numeroDeCoches[3]=0;

Con esto quedan inicializados todos los elementos del array y al ejecutar el programa obtenemos un resultado “seguro”.

Es posible que en un momento dado necesitemos borrar el contenido de los elementos de un array, digamos que resetear o borrar su contenido. Para ello en algunos lenguajes existen instrucciones específicas, pero en C tendremos que volver a asignar los valores por defecto a los elementos del array realizando lo que podríamos denominar una “reinicialización”. Más adelante veremos cómo se puede realizar este proceso de forma cómoda.

¿Cómo elegir los nombres de los arrays? Los nombres de variables deben ser lo más descriptivos posibles para hacer el programa fácil de leer y de entender. Piensa que es válido tanto declarar int vectorInt; como int VI;. Sin embargo es más correcto usar vectorInt que VI porque resulta más descriptivo de la función y cometido de la variable vectorInt que dos letras cuyo significado no

entenderá una persona que lea el programa (y quizás tú mismo no entenderás cuando hayan pasado unos días después de haber escrito el programa).

Nota: C no realiza una comprobación de índices de arrays. Por ejemplo si hemos declarado int numeroCoches[4] e incluimos en nuestro código el uso de numeroCoches[5] el comportamiento es imprevisible. Es responsabilidad del programador el controlar y hacer uso exclusivamente de los índices válidos para cada array.

EJERCICIO RESUELTO

Crea el código de dos programas que cumplan las siguientes premisas:

a) ***Programa 1***: Declara un array de enteros denominado *numeroDeCoches* que contenga 24 variables. Declara una variable tipo *int* que se llame *R*. Establece el valor de *R* en 2 y el valor de *numeroDeCoches* para un localizador de valor *R* en 57. Procede a mostrar en pantalla un mensaje que indique cuál es la hora *R* y el número de coches para la hora *R*. Finalmente, modifica únicamente la asignación de valor a R de modo que en vez de 2 sea 21 y ejecuta de nuevo el programa.

b) ***Programa 2****:* Sobre el programa anterior realiza los siguientes cambios. Declara dos variables *A* y *B* de tipo *int*. Establece *A* con valor 8, *B* con valor 4 y *R* con valor *A* dividido entre *B*. Ejecuta el programa.

SOLUCIÓN

El programa 1 será el siguiente. Si lo ejecutamos obtendremos "La hora R es 2. El número de coches en la hora 2 fue de 57 coches". Si cambiamos R = 2 por R = 21 obtendremos "La hora R es 21. El número de coches en la hora 21 fue de 57 coches". Es importante prestar atención a que los índices del array comienzan en cero, de modo que la primera hora es la hora cero y la última hora la hora 23, existiendo un total de 24 horas. Las horas van de 0 a 23.

```
#include <stdio.h>
#include <stdlib.h>
// Ejemplo aprenderaprogramar.com
int main() {
int numeroDeCoches[24];
int R;
R = 2;
numeroDeCoches[R] = 57;
printf("La hora R es %d\n", R);
printf("El numero de coches en la hora %d fue de %d coches\n", R, numeroDeCoches[R]);
return 0;
}
```

El programa 2 será el siguiente. Recuerda que para que el array *numeroDeCoches* tenga 24 elementos siendo el primero el de localizador cero, tenemos que establecerlo como *numeroDeCoches[24].* En estos programas estamos usando *printf*, %d, \n, etc. sin haber dado una explicación al respecto. No te

preocupes por esto pues lo explicaremos más adelante. Ahora es suficiente con que comprendas la lógica general de estos programas.

```
#include <stdio.h>
#include <stdlib.h>
// Ejemplo aprenderaprogramar.com
int main() {
int numeroDeCoches[24];
int A = 8;
int B = 4;
int R = A / B;
numeroDeCoches[R] = 57;
printf("La hora R es %d\n", R);
printf("El numero de coches en la hora %d fue de %d coches\n", R, numeroDeCoches[R]);
return 0;
}
```

Nota: hay una cosa "formalmente" incorrecta en estos programas. Hemos declarado un array con 24 elementos pero dichos elementos no los hemos inicializado. De hecho, hemos dejado la mayor parte sin inicializar. Idealmente es conveniente inicializar siempre los elementos de los arrays a un valor por defecto que consideremos adecuado. En este caso resultaría un tanto costoso escribir todos los términos a inicializar uno por uno. Más adelante veremos cómo hacer este tipo de inicialización de forma cómoda.

EJERCICIO

Crea el código de un programa que cumpla las siguientes premisas. Declara un array de enteros denominado *numeroDeHijos* que contenga 10 elementos. Declara una variable tipo *int* que se llame *T*. Establece el valor de *T* en 8 y el valor de *numeroDeHijos* para un localizador de valor *T* en 3. Procede a mostrar en pantalla un mensaje que indique cuál es el valor *T* y el número de hijos para el valor T. Finalmente, modifica únicamente la asignación de valor a T de modo que en vez de 3 sea 5 y ejecuta de nuevo el programa. Responde a estas preguntas:

a) ¿Qué significado podríamos atribuir a los índices del array? (Supón que se trata de un programa donde debes atribuirle un significado).

b) ¿Cuál será el primer índice del array?

c) ¿Cuál será el último índice del array?

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00513F

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=82&Itemid=210