Lenguaje de Programación

C++

Estructuras

José Luis Alonzo Velázquez

Universidad de Guanajuato

Sesión 33

Estructuras

Una estructura es un grupo de variables las cuales pueden ser de diferentes tipos sostenidas o mantenidas juntas en una sola unidad. La unidad es la estructura.

Las estructuras de datos se emplean con el objetivo principal de organizar los datos contenidos dentro de la memoria de la PC. Así, nuestra primera experiencia con estructuras comienza desde el momento mismo en que usamos en nuestros programas variables de tipos primitivos (char, short, int, float, etc).

Sintaxis

En C/C++ se forma una estructura utilizando la palabra reservada struct, seguida por un campo etiqueta opcional, y luego una lista de miembros dentro de la estructura. La etiqueta opcional se utiliza para crear otras variables del tipo particular de la estructura:

```
struct [ <nombre tipo de estructura > ] {
    [<tipo> <nombre-variable[,nombre-variable,...]>];
    [<tipo> <nombre-variable[,nombre-variable,...]>];
    ...
    } [ <variables de estructura> ] ;
```

Un punto y coma finaliza la definición de una estructura puesto que ésta es realmente una sentencia C/C++.

Sintaxis: Caso uno, estructura anónima

De acuerdo con la sintaxis general de la orden struct es posible crear estructuras de datos anónimas. Solamente hay que tener en cuenta que en una declaración anónima se debe definir al menos una variable al final de la declaración. Por ejemplo, con el siguiente fragmento de código:

```
struct {
    int a;
    int b;
} p1;
```

se declara y define la variable estructurada p1, misma que se compone por los miembros a y b; ambos del tipo int.

Ahora bien, la sintaxis mostrada no es tan común ni conveniente, ya que con la misma solamente se esta creando una variable estructurada pero no un nuevo tipo. Es decir, si deseáramos tener otra variable que tuviera las mismas características que posee la variable p1, necesitaríamos escribir exactamente la misma instrucción, salvo que cambiando el nombre de la variable. Por ejemplo:

```
struct { int a, b; } p2;
```

Por supuesto, en una misma línea de instrucción podemos definir más de una variable. Ejemplo:

```
struct { int a, b; } p1, p2;
```

Entonces, para crear nuevos tipos con struct deberemos de modificar la sintaxis mostrada en los ejemplos anteriores.

Sintaxis: Caso dos, estructura con nombre

Observe que, la sintaxis para declarar estructuras con nombre es bastante parecida a la sintaxis para declarar estructuras anónimas; salvo que una declaración de estructura con nombre se debe especificar el nombre deseado para la misma. Además, en una declaración de estructura con nombre la o las variables definidas al final de la misma son opcionales.

```
struct pareja { int a, b; } p1;
```

Observación

En el fragmento de código anterior se declara la estructura identificada como pareja, misma que se compone de los miembros a y b, ambos de tipo int. En el mismo ejemplo, se define la variable p1; la cual es una variable estructurada de tipo pareja.

Uso del nombre

Una vez que una estructura con nombre ha sido creada, la misma puede ser usada para declarar cualquier número de variables. Por ejemplo, en el siguiente fragmento de código se crea la estructura tiempo compuesta por los miembros hora, minuto y segundo; todos del tipo int. En el mismo ejemplo, se declaran las variables t1 y t2.

```
/* declaración de estructura tiempo */
struct tiempo { int hora, minuto, segundo; };

/* declaración de variables de tipo tiempo */
struct tiempo t1, t2;
```

Nota

En C++ puede obviarse la palabra struct a la hora de declarar variables. Así, en C++ la línea de instrucción struct tiempo t1, t2; (del ejemplo anterior) puede escribirse como: tiempo t1, t2;

Acceso a los miembros de una estructura

En orden de poder leer o escribir uno de los miembros de una variable estructurada se debe usar el operador de acceso (.), o sea, el nombre de la variable seguida por un punto seguido por el nombre del miembro o componente deseado de la estructura. Por ejemplo, para acceder a los miembros de la variable t1 (mostrada arriba) podemos hacerlo de la siguiente manera:

Ejemplo

```
t1.hora = 12;
t1.minuto = 0;
t1.segundo = 0;

printf ("%i\n", t1.hora);
```

Estructuras anidadas

Los miembros de una estructura pueden ser ellos mismos otra estructura previamente identificada o bien una estructura anónima. Por ejemplo, en el siguiente fragmento de código se crean las estructuras pareja y pareja2. Obsérvese cómo dentro de los miembros de pareja2 se declara el miembro X, mismo que es una estructura del tipo pareja. Luego, las variables declaradas a raíz de la estructura pareja2 poseerán los miembros variables a y b heredados de pareja, y c.

```
struct pareja { int a, b ; };
struct pareja2 { struct pareja X; int c; } P3;
```

Acceso en este caso

Ahora bien, para acceder a los miembros de una estructura dentro de otra estructura se emplea el mismo mecanismo de acceso (el punto). Por ejemplo, para desplegar el miembro a de la variable P3 declarada en el ejemplo anterior, lo haremos más o menos así:

```
printf( "%i\n", P3.X.a );
```

Declaración de una estructura

```
#include <stdio.h>
using namespace std;

struct punto{
    int x;
    int y;
};
struct linea{
    punto p1;
    punto p2;
};

int main(){
    return 0;
}
```

Ejemplo de uso de las estructuras descritas

```
int main(){
    linea l1;
    l1.p1.x=1;
    l1.p1.y=1;
    l1.p2.x=2;
    l1.p2.y=2;
    printf("La linea l1 pasa por los puntos (%d,%d),
(%d,%d)\n",l1.p1.x,l1.p1.y,l1.p2.x,l1.p2.y);
    return 0;
}
```


Como Programar en C/C++, Deitel (Prentice Hall), 2da Edición.


Programming Principles and Practice Using C++, Bjarne Stroustrup.


http://www.codeblocks.org


http://www.wxwidgets.org


(O'Reilly) Practical C Programming (3rd Edition)