Universidad Centroamericana “José Simeón Cañas”
Facultad de Ingeniería y Arquitectura

Tutorías - Fundamentos de Programación Ciclo 01/2021
Tutor: Ernesto José Canales Guillén

Registros y Enumeraciones

Struct – Estructura – Registro

Una estructura es una colección de uno o más tipos de elementos denominados miembros, cada uno de los cuales puede ser un tipo de dato diferente.

Una estructura es un tipo de dato definido por el usuario, que se debe declarar antes de que se pueda utilizar. El formato de la declaración es:

```
struct <nombre de la estructura>
{
   <tipo de dato miembro1> <nombre miembro1>;
   <tipo de dato miembro2> <nombre miembro2>;
   ...
   <tipo de dato miembron> <nombre miembron>;
};
```

La declaración de la estructura `CD` es

```
struct coleccion_CD
{
   char titulo[30];
   char artista[25];
   int num_canciones;
   float precio;
   char fecha_compra[8];
}
```

Definición de variables de estructuras

Al igual que a los tipos de datos enumerados, a una estructura se accede utilizando una variable o variables que se deben definir después de la declaración de la estructura. Del mismo modo que sucede en otras situaciones, en C++ existen dos conceptos similares a considerar, declaración y definición. La diferencia técnica es la siguiente.

- Una declaración especifica simplemente el nombre y el formato de la estructura de datos, pero no reserva almacenamiento en memoria.
- Una definición de variable para una estructura dada crea un área en memoria en donde los datos se almacenan de acuerdo al formato estructurado declarado.

Las variables de estructuras se pueden definir de dos formas:

1. Listándolas inmediatamente después de la llave de cierre de la declaración de la estructura.

```
struct colecciones_CD
   {
      char título[30];
      char artista[25];
      int num_canciones;
      float precio;
      char fecha_compra[8];
   } cd1, cd2, cd3;
```

2. Listando el nombre de la estructura seguida por las variables correspondientes en cualquier lugar del programa antes de utilizarlas.

```
colecciones_CD cd1, cd2, cd3;
```

Obsérvese que esta definición difiere del sistema empleado en el lenguaje C, en donde es obligatorio el uso de la palabra reservada struct en la definición.

```
struct colecciones_CD cd1, cd2, cd3;
```

Inicialización de una declaración de estructuras

Se puede inicializar una estructura de dos formas. Se puede inicializar una estructura dentro de la sección de código de su programa, o bien se puede inicializar la estructura como parte de la declaración.

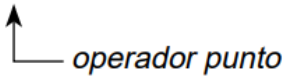
```
struct info_libro
{
   char título[60];
   char auto[30];
   char editorial[30];
   int anyo;
} libro1 = { "C++ a su alcance", "Luis Joyanes","McGraw-Hill", 1994 };
```

```
struct coleccion_CD
{
   char titulo[30];
   char artista[25];
   int num_canciones;
   float precio;
   char fecha_compra[8];
} cd1 =  {
            "El humo ciega tus ojos",
            "Col Porter",
            15,
            2545,
            "02/6/94"
          };
```

Acceso a estructuras - Almacenamiento de información en estructuras

1. Acceso a una estructura de datos mediante el operador punto

```
<nombre variable estructura> . <nombre miembro> = datos;
```




```
struct RegEstudiante
{
   int NumExpEstudiante;
   char curso;
};

int main()
{  RegEstudiante RegPersonal;
   RegPersonal.NumExpEstudiante = 2010;
}  RegPersonal.curso = 'Doctorado';
```

2. Acceso a una estructura de datos mediante el operador puntero

```
<puntero estructura> -> <nombre miembro> = datos;
```

```
struct estudiante
{
   char *Nombre;
   int Num_Estudiante;
   int Anyo_de_matricula;
   float Nota;
};

 Estudiante *Mortimer;

Mortimer -> Num_Estudiante = 3425;
Mortimer -> Nota = 7.5;
strcpy(Mortimer -> Nombre, "Pepe Mortimer");
```

Enumeraciones

Una enumeración, enum, es un tipo definido por el usuario con constantes de nombre de tipo entero.

***Usos típicos de* `enum`**

```
enum Interruptor
{
   ENCENDIDO;        // ON, "prendido"
   APAGADO;          // OFF, "apagado"
};

enum Boolean
{
   FALSE;
   TRUE;
};
```

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

enum week { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };

int main()
{
    week today;

    today = Wednesday;

    cout << "Day " << today+1; //Output Day 4

    return 0;
}
```

Sinónimo de un tipo de datos: typedef

Un typedef permite a un programador crear un sinómimo de un tipo de dato definido por el usuario o integral ya existente

Uso de `typedef` para declarar un nuevo nombre, longitud de tipo dato integral.

```
// ...
typedef double Longitud;
// ...
Longitud Distancia (const Punto& p, const Punto& p2)
{
   // ...
   Longitud longitud = sqrt(r-cua);
   return longitud;
}
```

La desventaja de typedef es que introduce nombres de tipos adicionales y pueden hacer los tipos que existen más abstractos.