



# Fundamentos de Programación 101

By Ernie

Ernesto José Canales Guillén

Círculos de estudio UCA

Ciclo Virtual 01/2021



# Struct



# 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

*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;
```

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

```
colecciones_CD cd1, cd2, cd3;

struct colecciones_CD cd1, cd2, cd3;
```



# 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;*

↑  
operador punto

```
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;
};
```

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



# Enum



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;
};
```





```
1 #include <iostream>
2 using namespace std;
3
4 enum week { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };
5
6 int main()
7 {
8     week today;
9
10    today = Wednesday;
11
12    cout << "Day " << today+1; //Output Day 4
13
14    return 0;
15 }
```



typedef



# Sinónimo de un tipo de datos: typedef

- Un typedef permite a un programador crear un sinónimo de un tipo de dato definido por el usuario o integral ya existente.
- La desventaja de typedef es que introduce nombres de tipos adicionales y pueden hacer los tipos que existen más abstractos.

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;
}
```



- L. J. Aguilar, Programación en C++. Algoritmos, estructuras de datos y objetos, Aravaca (Madrid): McGRAW-HILL, 2006.
- D. Malik, C++ Programming: From Problem Analysis to Program Design, Boston, MA: Cengage Learning, 2003.