

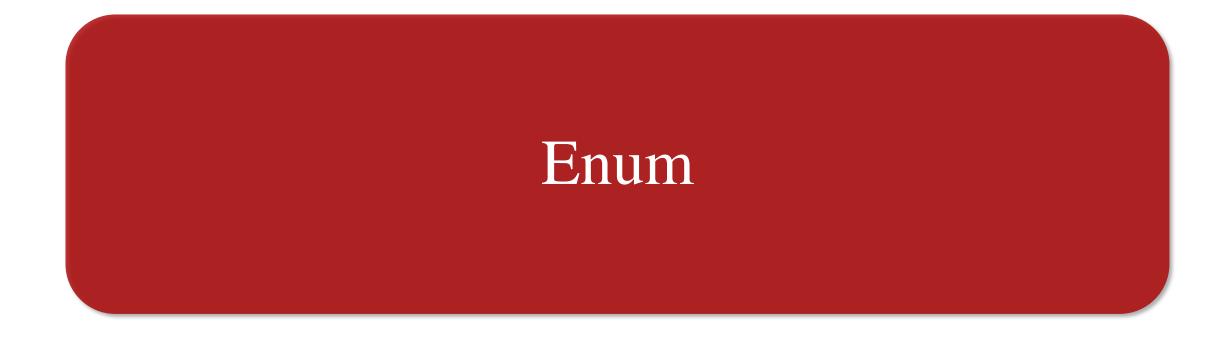
# Fundamentos de Programación 101 By Ernie

Ernesto José Canales Guillén

Círculos de estudio UCA

Ciclo Virtual 01/2021







### Enumeraciones

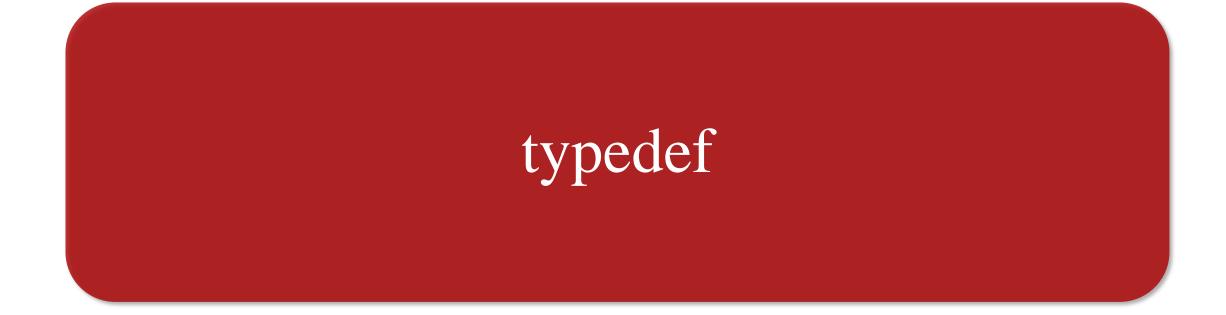
Una enumeración, enum, es un tipo definido por el usuario con constantes de nombre de tipo entero.

#### Usos típicos de enum



```
1 #include <iostream>
2 using namespace std;
4 enum week { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };
6 int main()
      week today;
      today = Wednesday;
      cout << "Day " << today+1; //Output Day 4</pre>
      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.
- 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;
}
```



# Header files

```
#include "header_name.h"
#include "header_name.hpp"
```



#### Archivos de cabecera

- Dado que un archivo de encabezado podría estar incluido en varios archivos, no puede contener definiciones que puedan producir varias definiciones del mismo nombre. Lo siguiente no está permitido o se considera una práctica muy mala:
  - Definiciones de tipo incorporadas en el espacio de nombres o el alcance global.
  - Definiciones de funciones no en línea.
  - Definiciones de variables no constantes
  - Definiciones agregadas
  - Namespaces sin nombre
  - Directivas
- El uso de la directiva using no necesariamente causará un error, pero puede causar un problema potencial porque trae el namespace al alcance en cada archivo .cpp que directa o indirectamente incluye ese encabezado.



```
1 //my_class.h
3 #pragma once //nclude guard
4 #ifndef MY_CLASS_H //nclude guard
5 #define MY_CLASS_H
6
7 //the entire header file code
8
9 #endif /* MY_CLASS_H */
```



# using namespaces



- Permiten agrupar entidades como clases, objetos y funciones bajo un nombre. De esta forma el ámbito global se puede dividir en "sub-ámbitos", cada uno con su propio nombre.
  - Los namespaces aparecen solo en el ámbito global.
  - Los namespaces se pueden anidar dentro de otro namespace.
  - Los namespaces no tienen especificadores de acceso. (Público o privado).
  - No es necesario poner punto y coma después de la llave de cierre de la definición de un namespaces.
  - Podemos dividir la definición de un namespaces.

```
1 namespace namespace_name {
2  // code declarations
3 }
```



```
• • •
  1 #include <iostream>
 2 using namespace std;
 5 namespace first_space {
      void func() {
         cout << "Inside first_space" << endl;</pre>
12 namespace second_space {
      void func() {
         cout << "Inside second_space" << endl;</pre>
18 int main () {
      first_space::func(); //Output Inside first_space
      second_space::func(); //Output Inside second_space
      return 0;
```







- 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.
- Microsoft Official Documentation C++