



# Clases y Objetos

Por:

Yolanda Martínez Treviño

Ma. Guadalupe Roque Díaz de León

# Constructor

- Un **constructor** es una función miembro **especial** de la clase que se utiliza para **dar valor inicial** a los **atributos** de un objeto al momento de crearlo.



- **C++** llama al **constructor** cada vez que se crea un objeto.

# Constructor

---

- El constructor debe tener:
  - el mismo nombre de la clase y debe ser público
- El constructor se usa para inicializar las variables de instancia de la aplicación.
- El constructor **No debe tener** :
  - tipo de valor de retorno.



# Constructor



- Un constructor se conoce como **constructor default** si no tiene parámetros.
- Si el programador no define un constructor, C++ provee un **constructor default**, que no tiene parámetros.
- Una clase **puede tener más de un constructor**, para proporcionar varias formas de inicializar los objetos de la clase.

# Ejemplo - Constructor default

---

La **declaración** del constructor:

```
Rectangulo( );
```

La **definición** del constructor **default**:

```
Rectangulo :: Rectangulo( )
```

```
{
```

```
    largo = 0;
```

```
    ancho = 0;
```

```
}
```



# Constructor: **sobrecarga, firma**

---

- **Firma** - signature- es el nombre de la función y sus parámetros, **sin incluir el valor de retorno**.  
Rectangulo(double, double);  
Rectangulo( );
- **Sobrecarga -overload-** tener varias funciones con el mismo nombre dentro de una Clase, programa, etc. siempre y cuando su **firma** sea diferente.
  - **C++** decide a cuál función llamar dependiendo de la **cantidad y tipo de parámetros** que tenga la llamada.

# Ej -Constructor con Parámetros

**Declaración** de un constructor con parámetros:

**Rectangulo(double, double);**

**Definición** del constructor:

```
Rectangulo :: Rectangulo(double _largo, double _ancho)
{
    largo = _largo;
    ancho = _ancho;
}
```



# Llamada al Constructor

---

- El constructor se llama automáticamente en la aplicación, cuando se declara una variable - instancia de la clase, se llama al constructor.
- Ej.
- 

Rectangulo `rect:`



Llama al constructor default de la clase.



# Llamada al Constructor

- Si la clase tiene constructores con parámetros, se deben enviar los parámetros al momento de crear el objeto.
- Ej.

Rectangulo **rect1**{10,20}

Llama al constructor  
que tiene 2 parámetros  
de tipo double.



# Constructor

---

- Si la clase solo tiene constructores con parámetros, **la clase no tendrá constructor default**. Al declarar un objeto en la aplicación, se debe enviar los parámetros que requiere el constructor.



# Ejercicio

---

- Modifica la clase **Reloj** para que tenga dos constructores:
  - Un constructor **default** y
  - Un constructor con **parámetros**.
- 
- Ambos constructores deben garantizar que los atributos tengan valores válidos, si no son correctos, se deberá inicializar la hr y/o min con cero.



# ? Operador condicional C++

Sintaxis:

condición ? valor Si True : valor Si False

variable = condición ? valor Si True : valor Si False ;

Ejemplos:

```
cout << (calif >= 70 ? "Felicidades\n " : " Acude a Asesoría\n");
```

```
mes = (x > 0 && x < 13 ? x : -1);
```

# Ejercicio:

---

Cambia uno de los métodos **modificadores**  
**set** para que utilice el operador condicional  
**?** para validar

# Prototipo de una función:

---

En C++ es necesario que una función esté declarada y definida antes de utilizarla.

Para declarar una función se utiliza su prototipo que es el encabezado seguido de un ;

# Algunos conceptos útiles

---

## ■ Prototipo de una función

- Es cualquiera de los métodos declarados dentro de una clase.
- Al declarar los parámetros de la función no es necesario declarar el nombre de los parámetros, sino solo el tipo.

Ejemplos:

- Rectangulo(double, double);
- void setLargo(double);
- double getLargo( );
- string str( );

# Ejercicio:

---

Modifica la clase **Reloj** de modo que los métodos de la clase **Reloj** tenga un prototipo en el que **no** se especifique el nombre de los parámetros.



# Destructor

- El nombre del **destructor** es el mismo nombre de la clase precedido por el caracter ~

Por ejemplo:

**~Rectangulo();**



- El destructor no debe tener parámetros ni valor de retorno.
- Una clase solo puede tener **un destructor**.
- Cada clase tiene un destructor, si no se incluye, C++ provee un destructor por default.
- Ej. Rectangulo:: **~Rectangulo(){**
- **cout << "Se destruyo un objeto Rectangulo \n;**
- **}**

# Interface e Implementación de una clase - `#include "Clase.h"`



Header File

- La **interface** es la declaración de la clase. Se guarda en un archivo con extensión **.h**



C++ File

- La **implementación** es la definición, codificación de las funciones miembro de la clase. Se guarda en un archivo con extensión **.cpp**
  - Importante - se debe Añadir en el archivo de la implementación el `#include "Clase.h"`

<https://www.learncpp.com/cpp-tutorial/class-code-and-header-files/>



Circulo.cpp





Circulo.hpp

# Ocultamiento de Información (information hiding)

- La filosofía de este concepto es que la codificación de la clase no es asunto de la aplicación que usa la clase (App cliente).
- La aplicación cliente necesita conocer solamente la **interface** de la clase.

```
class Rectangulo
{ public:
    Rectangulo();
    Rectangulo(double _largo, double _ancho);
    ~Rectangulo();
    double calcArea( );
    void setLargo(double _largo);
    void setAncho(double _ancho);
    double getLargo();
    double getAncho();
private:
    double largo, ancho;
};
```

-  Si la implementación de la clase cambia, la aplicación cliente puede seguir trabajando sin cambios , siempre y cuando la interface no tenga modificaciones.

# Ejercicio:



Coloca la **interface** de la clase Reloj en el archivo **Reloj.h** y la **implementación** en el archivo **Reloj.cpp**

Importante -

Añade los include :

- En el archivo **Reloj.cpp** se debe incluir
  - `#include "Reloj.h"`