# Ayudantía 2

# Repaso y NULL

Benjamín Aceituno

Programación Avanzada S09

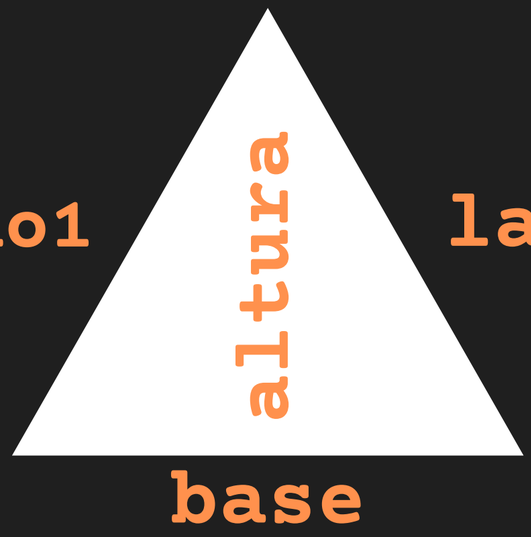
# Set y Get

**En la ayudantía anterior, vimos cómo funcionan los métodos set y get. Estos métodos se pueden usar sin necesidad de inicializar un constructor porque nos permiten asignar valores a los atributos del objeto después de que el objeto ha sido creado. Esto significa que, aunque no tengamos un constructor que establezca los valores iniciales, podemos establecerlos posteriormente usando los métodos set.**

```cpp
#include <iostream>
using namespace std;
class Triangulo{
private:

int lado1, lado2, base, altura;

public:

// SET

void setLado1(int l1){
    lado1 = l1;
}
void setLado2(int l2){
    lado2 = l2;
}
void setBase(int b){
    base = b;
}
void setAltura(int a){
    altura = a;
}
```

# Ejercicio

Usando métodos set y get, haga un programa que permita ingresar base, altura y dos lados de un triangulo, los muestre por pantalla y permita calcular su perímetro y área.

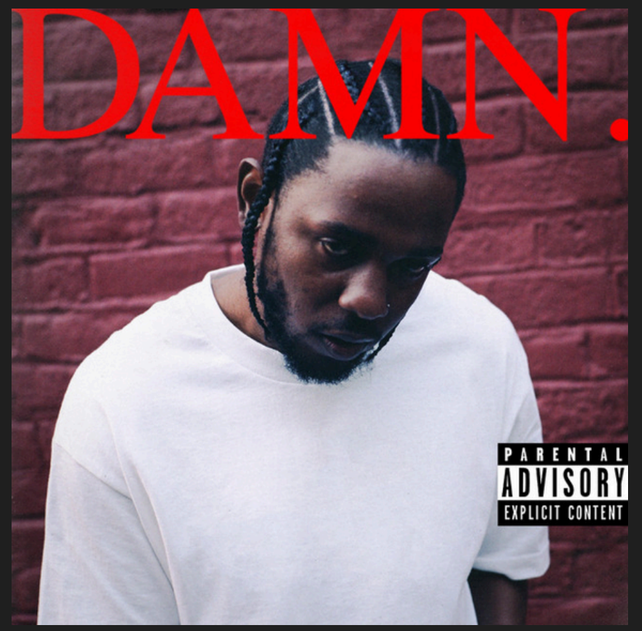



# Objeto dentro de otra Clase

**Puede ser útil tener un objeto dentro de otra clase, ejemplo, una clase Estudiante tiene un objeto derivado de clase “Asignaturas”. Veremos más ejemplos de esto a continuación**

**Un artista puede tener atributos como su nombre, edad y discos.**
**En este caso discos es una clase que tiene un objeto disco, que contiene el nombre del disco, titulo de las canciones, duración, etc.**

**La clase artista tiene como atributos nombre “Kendrick Lamar” edad 37 y disco “DAMN”**




**Y la clase discos tiene un objeto disco que tiene el titulo de las canciones “BLOOD, DNA, FEEL”. su duración 3:00, 4:00, etc.**

# Ejemplo

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

class Direccion {
public:
    string calle;
    int numero;

    Direccion(string c, int n) : calle(c), numero(n) {}
};

class Persona {
public:
    string nombre;
    Direccion direccion; // La clase Persona tiene un objeto de la clase Direccion, de cierto modo le paso el objeto a otra clase

    Persona(string n, Direccion d) : nombre(n), direccion(d) {} // Inicializando la clase persona con un objeto dirección de la clase persona

    void mostrarInformacion() {
        cout << "Nombre: " << nombre << endl;
        cout << "Calle: " << direccion.calle << endl;
        cout << "Numero: " << direccion.numero << endl;
    }
};

int main() {
    Direccion miDireccion("Plaza UDPorros", 420); // Crear un objeto Direccion

    Persona persona("Charles Piña", miDireccion); // Crear un objeto Persona con un objeto miDireccion dentro

    persona.mostrarInformacion(); // Mostrar información de la persona

    return 0;
}
```

# Ejercicio

Sorpresivamente, usted recibió el beneficio de la JUNA™, para lo cual se le pide crear un programa para gastarla. Este programa tiene una clase CarritodeCompras que tiene objetos de la clase Producto, el carrito tiene un arreglo de productos y la cantidad de productos que tiene dentro, por otro lado, los productos tienen nombre y precio. Debe programar los métodos `agregarProducto`, que agrega un producto al carrito, `calcularTotal`, que calcula el precio total de productos en el carrito, y `mostrarCarrito` que muestra los productos en el carrito.

# NULL

Un puntero nulo o NULL, es un puntero que no contiene información, no apunta a ningun acceso de memoria del programa. Podemos utilizarlo para rellenar arreglos con “nada”. Esto nos puede servir para por ejemplo, un asiento que no tiene ninguna persona asignada.

Por ejemplo, esto es lo que haría un progama en C++ de un cine para inicializar todos los asientos de una sala como asientos sin ocupar

```
butacas[i] = NULL;
```

# Ejercicio

La famosa empresa de cines “CinePepe” sufrió un hackeo y borraron su programa para asignar asientos a las personas en sus salas de cine, por lo cual, debe crear un programa que tenga una clase Persona y una clase Sala. La clase Persona tiene atributos como RUT y nombre, y la sala tiene un arreglo de 100 butacas para ingresar personas.

Programe los siguientes métodos: **Venta:** Asigna una persona a un asiento de la sala. **personaYaTieneAsiento:** Verifica si la persona ya tiene un asiento (No puede haber más de un asiento a rut de una persona) **buscarPersona:** Busca una persona por su rut y dice su butaca. **imprimirSala:** Muestra butaca, rut y nombre de todas las personas de la sala. **eliminarPersona:** Elimina a una persona y libera la butaca que tiene asignada

# Contacto

+569 86031881

benjamin.aceituno@mail.udp.cl

benja.mp4

https://github.com/benjamp4/prog.av-2024-2