Paradigmas de programación

Universidad de San Andrés

Programación Orientada a Objetos

Tutorial 4

REPASO

- Conceptos de OOP
- Miembros ***public*** y ***private***.
- Constructores y destructores.
- Sobrecarga de operadores.

Definición básica de una clase

```h
#pragma once

#include <string>
#include <cmath>
#include <iostream>

using namespace std;

class Auto {
private:
    string marca;
    string modelo;
    int anio;
    int kilometraje;
    float velocidad;
    float posX;
    float posY;
    float direccion;
public:
    Auto(string m, string mo, int a, int k, float v, float x,
float y, float d);
    void acelerar();
    void frenar();
    void girar(float grados);
    void mover(float x, float y);
    string getPosicion();
}
```

```cpp
#include "auto.h"

Auto::Auto(string m, string mo, int a, int k, float v, float x, float y, float
d){
    marca = m;
    modelo = mo;
    anio = a;
    kilometraje = k;
    velocidad = v;
    posX = x;
    posY = y;
    direccion = d;
};

void Auto::acelerar(){ velocidad += 10; };

void Auto::frenar(){ velocidad -= 10; };

void Auto::girar(float grados){ direccion += grados; };

void Auto::mover(float t){
    posX += t * velocidad * cos(direccion);
    posY += t * velocidad * sin(direccion);
};

string Auto::getPosicion(){ return to_string(posX) + ", " + to_string(posY); };
```

Definición básica de una clase

- Por defecto, todos los miembros y funciones son públicas.
- Los modificadores de acceso public y private controlan cómo se usa una clase.
- Es una categoría de objetos; un nuevo tipo de dato más complejo que los tipos básicos.

Uso de una clase

```
#include "auto.h"

int main() {
   Auto a("Ford", "Fiesta", 2019, 0, 0, 0, 0, 0);
   a.acelerar();
   a.girar(3.14159 / 2);
   a.mover(1);
   cout << a.getPosicion() << endl;
   return 0;
}
```

Ejecución:

```
0.000013, 10.000000
```

Constructores

```
class Persona {
private:
   string nombre;
   string apellido;
   int dni;
public:
   // Ejemplo de constructor, el parametro unDNI tiene un valor por defecto.
   Persona (string unNombre, string unApellido, int unDNI = -1)
     : nombre {unNombre}, apellido {unApellido}, dni {unDNI} {
     // En este espacio se pueden inicializar los miembros o verificar que los
     // parametros sean correctos.
     }
   void leerNombreCompleto() { cout << "Nombre completo: " << nombre << " " << apellido << endl; }
   void leerDNI() { cout << "DNI: " << dni << endl;}
};
```

- Pueden tener valores por defecto en los argumentos.
- Se recomienda que una clase se inicialice en un estado válido.
- Se pueden hacer overloads de constructores.

```
int main()
{
    // Instanciar un objeto de la clase Persona.
    // Si no se provee argumento de dni se usa él por defecto.
    Persona desconocido("Bjarne", "Stroustrup");

    desconocido.leerNombreCompleto();
    desconocido.leerDNI();
    return 0;
}
```

```
$ ./ejemplo
Nombre completo: Bjarne Stroustrup
DNI: -1
```

Archivo header (.h) vs Archivo fuente (.cpp)

	Header (.h)	Fuente (.cpp)
Propósito	Define interfaz	Implementa funciones de clases
Contiene	Prototipos de funciones, declaraciones de clases, definiciones de tipos, constantes	Implementaciones, código ejecutable
Acceso	Se comparte con otros archivos fuente	Es compilado a un archivo objeto (.o)
Ejemplo	void unaFuncion();	void unaFuncion() {...};

Estructura de un archivo .h

```
// libro.h
#ifndef LIBRO_H
#define LIBRO_H

class declaracion;

#endif
```

Si una clase está incluida en múltiples archivos, utilizamos ifndef para que se incluya por única vez.

¿Qué desventajas tiene la programación procedural?

- Repetición de código.
- Dificultad para manejar errores y encontrar.
- Dificultad para separar responsabilidades.
- Falta de abstracción.

Dan Ingalls sobre la programación orientada a objetos - Youtube

Homework 2
Consultas Homework 1