1. ¿Qué son los paradigmas? Estructurado vs Orientado a Objetos

Un paradigma de programación es un enfoque que define cómo se estructuran y organizan los programas.
Indica cómo resolver problemas y organizar el código en base a una cosmovisión particular.
Cada paradigma tiene sus propias reglas, conceptos y técnicas, y algunos lenguajes pueden admitir múltiples paradigmas.

El paradigma Estructurado se basa en la división del problema en funciones o procedimientos. Usa estructuras de control como condicionales (if, switch) y bucles (for, while).
El foco está en la lógica del programa y la secuencia de instrucciones.
Tiene la ventaja de mejorar la legibilidad y mantenibilidad del código,
y de esa forma mejorar la calidad del software, así como los tiempos de producción de los equipos.

El paradigma Orientado a Objetos (POO) tiene el enfoque de modelar los objetos del mundo real, que tienen estado (atributos) y comportamiento (métodos), realizando una abstracción del mismo para plasmarlo en una clase, que luego se usará de base para otras clases o instancias.

Sus principales conceptos son:
Abstracción: ocultar detalles innecesarios y resaltar lo esencial.
Encapsulamiento: ocultar el estado interno y exponer solo lo necesario.
Herencia: reutilizar código mediante jerarquías de clases.
Polimorfismo: tratar objetos diferentes de manera uniforme.

Promueve la modularidad y reutilización del código, siendo ideal para proyectos grandes y sistemas complejos.

2. ¿Qué es la Abstracción, dar ejemplos?

La abstracción consiste en enfocarse en los aspectos esenciales de un objeto o sistema, ocultando los detalles innecesarios o internos de su implementación. Permite así el mostrar lo que un objeto hace, sin necesidad de revelar cómo lo hace.

Este principio ayuda a reducir la complejidad del sistema, ya que el usuario o programador no necesita entender el funcionamiento interno para poder utilizar una funcionalidad. Así mismo facilita el uso de componentes de software, al proporcionar interfaces simples e intuitivas. Promueve la reutilización, ya que una vez definido un comportamiento general, se puede aplicar en diferentes contextos sin preocuparse por su implementación interna.
Y mejora el mantenimiento del código, ya que los detalles internos pueden cambiar sin afectar a quienes usan ese componente, siempre que la interfaz pública se mantenga.

La abstracción está presente cada vez que interactuamos con un sistema sin conocer su funcionamiento interno: electrodomésticos, automóviles, o al llamar a una función en un programa.

3. ¿Qué es el Isomorfismo de Estructuras y el GAP Semántico?

El Isomorfismo de Estructuras es la idea de que la estructura de los datos en la computadora debe representar fielmente la estructura lógica del problema en la vida real. Por ejemplo, si tenés un sistema de empleados con sucursales, deberías tener clases como Empleado y Sucursal que representen esa relación del universo del problema, realidad, dentro del universo de la solución, que es el programa.

El GAP Semántico es la distancia conceptual entre el problema real y la solución computacional. Cuanto mayor el GAP, más difícil es traducir el problema al código particular que busca solucionarlo.

La POO reduce el GAP, ya que los objetos del código se parecen más a los del mundo real.

4. ¿Qué es un Puntero? Dar un ejemplo de creación de Objeto en memoria dinámica

Un puntero es una variable que almacena la dirección de memoria de otra variable.
En C++, es fundamental en la creación de objetos en memoria dinámica (con new y delete), lo cual es clave para el manejo de objetos y estructuras de forma óptima.

- Crear objeto dinámico:

```
class Empleado {
public:
    std::string nombre;
    Empleado(std::string n) : nombre(n) {}
    void saludar() {
        std::cout << "Hola, soy " << nombre << std::endl;
    }
};

int main() {

 // Creación en memoria dinámica
    Empleado* e1 = new Empleado("Juan");

    e1->saludar();

    delete e1;
    return 0;
```

```
}
```

- Se usa new para crear el objeto en el heap (memoria dinámica) y delete para liberarlo.