TP 15 – POLIMORFISMO

1. ¿Qué es una función virtual en C++?

Una función virtual en C++ es una función declarada en una clase base que puede ser sobrescrita por las clases derivadas. Su principal característica es que permite determinar en tiempo de ejecución qué implementación de la función se ejecutará, dependiendo del tipo del objeto al que apunta un puntero o referencia.

2. ¿Cuál es la diferencia entre una función virtual y una función normal en C++?

- Función normal: La función que se ejecuta depende del tipo estático del puntero o referencia (determinado en tiempo de compilación).

- Función virtual: La función que se ejecuta depende del tipo dinámico del objeto (determinado en tiempo de ejecución).

Diferencia entre función virtual y sobrecarga de métodos:

- Función virtual: Implica redefinición de una función en una clase derivada para implementar comportamiento polimórfico.

- Sobrecarga de métodos: Es tener varias funciones con el mismo nombre pero diferentes parámetros dentro de la misma clase o en clases derivadas.

3. ¿Cómo se declara una función virtual en una clase en C++?

Se usa la palabra clave virtual, por ejemplo:

```
#include <iostream>

using namespace std;


class Base {

public:

   virtual void mostrar() {

      cout << "Base" << endl;

   }

};

class Derivada : public Base {
```

```
public:
    void mostrar() override {
        cout << "Derivada" << endl;
    }
};
```

4. ¿Qué es el polimorfismo en C++?

El polimorfismo es la capacidad de un mismo código de tratar objetos de diferentes tipos de manera uniforme. En C++, permite que una referencia o puntero a una clase base invoque métodos específicos de una clase derivada en tiempo de ejecución (polimorfismo dinámico).

5. ¿Cuáles son las ventajas y desventajas del polimorfismo?

Ventajas:

1. Facilita la extensibilidad al permitir nuevas clases derivadas sin modificar el código existente.
2. Mejora la reutilización de código al utilizar un diseño orientado a objetos.

Desventajas:

1. Introduce un ligero costo de rendimiento debido a la resolución dinámica en tiempo de ejecución.
2. Puede incrementar la complejidad del diseño del programa.

6. ¿Se puede tener una función virtual en una clase base en C++? Justificar

Sí, se puede y es común. Una clase base define una función virtual para que las clases derivadas puedan sobrescribirla y permitir polimorfismo dinámico. Esto es fundamental en diseños orientados a objetos para trabajar con interfaces generales.

7. ¿Qué es una función virtual pura y cómo se declara?

Una función virtual pura es una función que no tiene implementación en la clase base y se define asignándole = 0.
Ejemplo:

```
class Base {

public:

   virtual void mostrar() = 0; // Función virtual pura

};
```

Una clase que tiene al menos una función virtual pura se convierte en una clase abstracta y no puede instanciarse.

8. ¿Cómo se logra el polimorfismo en C++ utilizando punteros y referencias a objetos?

Se logra definiendo funciones virtuales en la clase base y utilizando punteros o referencias a la clase base para apuntar a objetos de clases derivadas.
Ejemplo:

```
#include <iostream>

using namespace std;



class Base {

public:

   virtual void mostrar() {

      cout << "Base" << endl;

   }

};



class Derivada : public Base {

public:

   void mostrar() override {

      cout << "Derivada" << endl;

   }
```

```
};


int main() {
    Base* ptr = new Derivada();
    ptr->mostrar(); // Llama a Derivada::mostrar()
    delete ptr;
}
```

9. ¿Qué son las clases abstractas en C++?

Una clase abstracta es una clase que no puede instanciarse y está diseñada para ser utilizada como base. Tiene al menos una función virtual pura.

10. ¿Cuándo se debe utilizar el modificador override al definir una función virtual en una clase derivada?

Se debe usar override para indicar explícitamente que una función en una clase derivada está sobrescribiendo una función virtual de la clase base. Esto ayuda al compilador a detectar errores si la función no coincide exactamente en su firma.