1 Herencia y clases

a) ¿Qué es herencia múltiple?

Es cuando una clase hereda de más de una clase base.

Permite que la clase hija tenga atributos y métodos de varias clases.

Ejemplo:

```
class A { };

class B { };

class C : public A, public B { };
```

b) ¿Para qué sirve la palabra reservada virtual?

Permite declarar un método como polimórfico, de modo que al usar punteros o referencias a la clase base, se llame a la versión correcta en la clase derivada.

También se usa en destructores para que se ejecuten correctamente los destructores de clases derivadas.

c) ¿Qué es una clase abstracta?

Es una clase que no se puede instanciar y contiene al menos un método virtual puro (=0).

Obliga a las clases derivadas a implementar los métodos virtuales puros.

d) ¿En qué se diferencian agregación y herencia?

Herencia: relación "es un" (Alumno es una Persona).

Agregación: relación "tiene un" (Auto tiene un Motor).

e) Indique dos formas de obtener una dirección de memoria válida para un puntero.

Asignándole la dirección de un objeto existente:

```
int x = 5;
int* p = &x;
```

Reservando memoria dinámica:

```
int* p = new int;
int* arr = new int[10];
```

2 Contención, herencia y ocultación

a) ¿Qué significa que una clase A es contenedora de otra B?

Significa que A tiene objetos de tipo B como atributos → relación de composición o agregación.

b) Antes de codificar en C++, ¿cómo reconoce si 2 clases pueden relacionarse a través de herencia?

Si existe una relación "es un" entre las entidades (especialización).

Si la nueva clase puede comportarse como la clase base sin problemas de coherencia.

c) ¿Qué entiende por ocultación y cómo se implementa en C++?

Ocultación = encapsulamiento: proteger los datos internos de la clase.

Se implementa haciendo los atributos privados o protegidos y usando métodos públicos (getters/setters) para acceder.

d) ¿Cuándo es necesario sobrecargar el constructor de copia de una clase?

Cuando la clase maneja punteros o memoria dinámica.

Para evitar la copia superficial que puede provocar errores al liberar memoria (delete).

3 this, polimorfismo y encapsulamiento

a) ¿Cuál es el objetivo de this?

this es un puntero que apunta al objeto actual que está ejecutando un método.

Permite diferenciar entre atributos y parámetros con el mismo nombre o pasar el objeto actual como referencia.

b) ¿Qué es un método virtual puro?

Es un método declarado en la clase base con =0 y sin implementación, que obliga a las clases derivadas a implementarlo.

c) ¿Qué es encapsulamiento en POO?

Es proteger los datos internos de un objeto y controlar su acceso mediante métodos públicos.

Permite ocultar la implementación y mantener la integridad de los datos.

d) ¿Es necesario sobrecargar el operador = para una clase?

Solo si la clase usa punteros o recursos dinámicos.

Evita que la asignación haga una copia superficial, provocando errores al liberar memoria.

e) Complete el código para definir dinámicamente un arreglo de 20 enteros al azar menores que 500 y mostrarlos

```
int main() {
   srand(time(0)); // inicializa generador de números aleatorios
   int* p;
   p = new int[20]; // reservar memoria dinámica

   for (int i = 0; i < 20; i++) {
      p[i] = rand() % 500; // números menores que 500
   }

   for (int i = 0; i < 20; i++) {
```

```
        cout << *(p + i) << " ";
    }

    delete[] p; // liberar memoria
    return 0;
}
```