I- CONCEPTOS BÁSICOS DE PROGRAMACIÓN

1. ¿Qué es el preprocesador?
2. ¿Qué es una directiva de preprocesador?
3. ¿Qué son los macros y para qué sirven?
4. Ejemplifica las siguientes directivas de preprocesador:

   Directiva **#define**:

   Directiva **#include:**

   Directiva **#include**:

   Directiva **#pragma:**

   Directiva **#warning:**

5. Fundamenta el uso práctico de las directivas pragma y warning
6. ¿Qué es una variable?
7. ¿Cuál es la diferencia entre una variable local y una global?
8. ¿Qué es un puntero?
9. ¿Qué es el heap y que es el stack?
10. Si declaramos una variable del tipo int de nombre “rojo” que es igual a 0 (int rojo=0) y posteriormente declaramos otro int de tipo puntero de nombre verde (int*verde):

    1) ¿Cuál de estas variables está almacenada en el heap?

    2) ¿Por qué?

11. Dado el siguiente problema crea el código en tu computadora y escribe los resultados:
    1) Usando el paso por valor, genera un código que a través de una función devuelva la suma de 2 valores enteros y escribe el resultado.
    2) Si agregas ampersand(&) a los parámetros de la función y le colocas constantes ¿Que sucede? ¿Por qué?
    3) Resuelve el problema usando el paso por referencia.

12. Sabiendo declarar variables y punteros, resuelva el siguiente ejercicio
    Crea una estructura de nombre *persona* y crea un arreglo char para una variable llamada apellido, y otro arreglo char para el apellido, más un entero para la edad.
    Una vez creada la estructura, genera una variable de tipo *persona* y genera el código para:
    1) Obtener el nombre
    2) Obtener el apellido
    3) obtener la concatenación de nombre y apellido
    4) obtener la edad

Una vez hecho esto, replicar el mismo ejercicio usando funciones dentro de las estructuras.Ejemplo:

```
struct Persona{
        char Nombre[30];
        char Apellido[30];
        int edad;
        int FuncionSumaDosNumeros(){
                return 5+15;
        }
};
```

A. Resuelva el mismo ejercicio usando punteros. Ej:
   Persona *UnaPersona;
   UnaPersona = new Persona;
B. Usa el puntero UnaPersona para asignar valores

13. ¿Qué es un enum?

14. ¿Cuáles son los beneficios de usar una estructura enum en vez de declarar variables?

15. Declara una estructura enum llamada “DiasSemana” con los siguientes datos: {lunes, martes, miércoles, jueves, viernes, sábado, domingo}
    1) Imprime todos los días de la semana, ¿Qué es lo que imprime? ¿Por qué?
    2) Si se cambia el valor del enumerado “jueves” a “jueves=12” ¿Volverá a imprimir el valor anterior o cambiará? ¿Qué pasa con el resto de los valores?
16. Utilizando try and catch, crea un programa en el que se le pide al usuario un número e imprimir si es par o impar. Los errores deberán ser procesados por el catch.
17. ¿Cuáles son los beneficios de encontrar errores de esta manera? Fundamente su respuesta.

II- CLASES

1. ¿Qué es un objeto?

2. Definición de instancia de un objeto.

3. ¿Cuáles son los 4 pilares de POO?

4. Crea una clase Persona con los campos privados *nombre*, *altura y peso*.

5. ¿Qué son los constructores y destructores?

6. ¿Cuándo se ejecutan un constructor y un destructor?

7. De las siguientes reglas de constructores CUAL NO es correcta:

   - Tienen que ser privados.
   - Tienen que llamarse igual.
   - No tienen tipo de dato, por ende "no retorna" ningún tipo de dato.
   - todas las clases tienen un constructor por default.
   - siempre y cuando no exista ambigüedad puede haber múltiples constructores.

8. ¿Cuáles son las 3 formas de acceder a los miembros de una clase?

9. Hay tres especificadores de acceso ¿Cuáles son y que las diferencia?

10. ¿Diferencias entre #ifndef, #define y #endif?

11. 3 reglas de funciones amigas "friend".

12. ¿Qué es una clase dentro de otra clase?

13. ¿Qué es un arreglo de clases?

14. ¿Qué es la sobrecarga?

III- HERENCIA Y POLIMORFISMO

1. ¿Qué es la herencia?

2. Dibuje una jerarquía de herencia para los estudiantes en una universidad, de manera similar a la jerarquía que se muestra en la figura 9.2. Use a Estudiante como la superclase de la jerarquía, y después extienda Estudiante con las clases *EstudianteNoGraduado* y *EstudianteGraduado*. Siga extendiendo la jerarquía con el mayor número de niveles que sea posible. Por ejemplo, EstudiantePrimerAnio, EstudianteSegundoAnio, EstudianteTercerAnio y EstudianteCuartoAnio podrían extender a *EstudianteNoGraduado*, y *EstudianteDoctorado* y *EstudianteMaestria* podrían ser subclases de *EstudianteGraduado*. Después de dibujar la jerarquía, hable sobre las relaciones que existen entre las clases. [Nota: no necesita escribir código para este ejercicio].

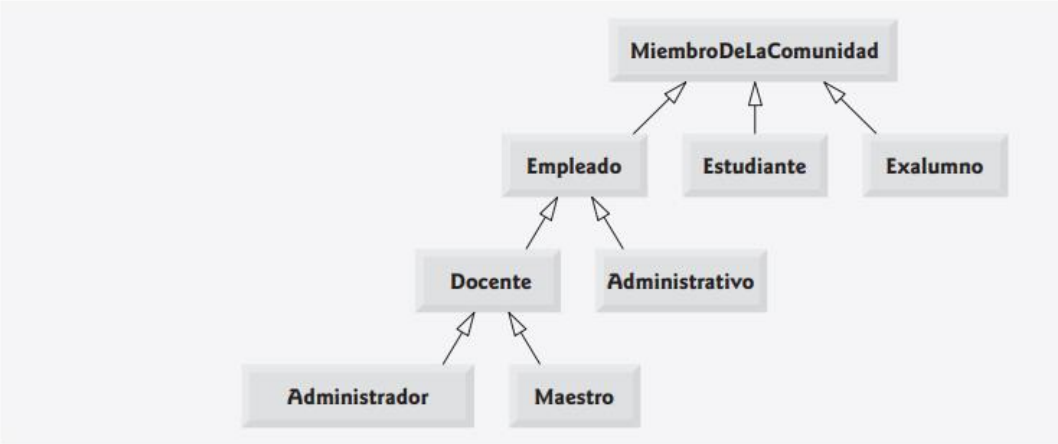

Figura 9.2 | Jerarquía de herencia para objetos `MiembroDeLaComunidad` universitaria.

3. ¿Qué es el efecto diamante?

4. ¿Qué es una clase virtual?

5. ¿Qué son los métodos abstractos? Describa las circunstancias en las que un método abstracto sería apropiado.

6. ¿Cómo es que el polimorfismo fomenta la extensibilidad?.

7. (Jerarquía de figuras) Implemente la jerarquía Figura que se muestra en la figura 9.3. Cada FiguraBidimensional debe contener el método obtenerArea para calcular el área de la figura bidimensional. Cada FiguraTridimensional debe tener los métodos

obtenerArea y obtenerVolumen para calcular el área superficial y el volumen, respectivamente, de la figura tridimensional. Cree un programa que utilice un arreglo de referencias Figura a objetos de cada clase concreta en la jerarquía. El programa deberá imprimir una descripción de texto del objeto al cual se refiere cada elemento del arreglo. Además, en el ciclo que procesa a todas las figuras en el arreglo, determine si cada figura es FiguraBidimensional o FiguraTridimensional. Si es FiguraBidimensional, muestre su área. Si es FiguraTridimensional, muestre su área y su volumen.

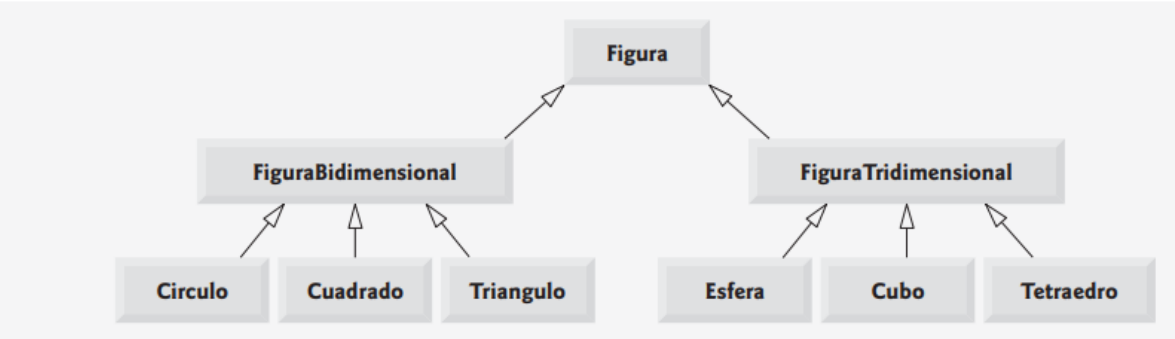

Figura 9.3 | Jerarquía de herencia para `Figuras`.

IV- PLANTILLAS

1. ¿Qué es una plantilla?

2. Sabiendo la definición de una plantilla resuelve el siguiente ejercicio

- Crea 3 funciones, una que imprima un resultado entero, otra que imprima un resultado flotante y otra que imprima un resultado en cadena de tipo std::string
- Una vez hecho eso ahora crea una función plantilla que imprima todos los resultados sin importar el tipo y observa su comportamiento (Usa template<typename> para mayor lectura)
- Crea una función plantilla en donde intercambies 2 valores de tipo plantilla, usa paso por referencia e imprime los resultados desde la función Main(), escribe los resultados
- Ahora crea 2 arreglos de char y asignales un valor, posteriormente crea 2 variables char de tipo puntero para asignar la dirección de memoria de los arreglos anteriores y coloca estos punteros en la función plantilla creada en el punto anterior ¿Cuál es el resultado?
- Crea 2 estructuras, una estructura de nombre y datos de tu preferencia y una estructura plantilla con un dato del typename de la plantilla, un número entero y una variable del mismo tipo de la estructura anterior. En el main crea un objeto del mismo tipo de la estructura plantilla y detalla en los diamantes el tipo de dato que quieres que sea el *typename*:

3. ¿Qué pasa con el dato de tipo plantilla?
4. ¿Qué pasa con el atributo de tipo estructura que está dentro del objeto?
5. ¿Por qué hay que especificar un argumento al objeto de tipo plantilla?
6. ¿Qué pasa si en vez de que la estructura plantilla tenga *typename* le colocas un argumento de tipo int?
7. ¿Qué pasa si en vez de que la estructura plantilla tenga *typename* le colocas un argumento de tipo std::string?
8. Crea un objeto del mismo tipo de la estructura plantilla, pero ahora asignando la primera estructura como argumento e imprime el valor de la plantilla. ¿Qué imprime?

V- PATRONES DE DISEÑO

1. ¿Qué es un patrón de diseño?
2. ¿Cuál es la diferencia entre un algoritmo a un patrón de diseño?
3. ¿En qué consiste un patrón de diseño?

En base al patrón de diseño *singleton*, responde:

1. ¿Qué es el singleton?
2. ¿Qué problemas resuelve el singleton?
3. ¿Cuál es la diferencia entre usar una variable estática a usar una clase singleton?
4. Crea una clase llamada "Personaje" que tenga 2 atributos de tipo entero llamados puntos y vida, después crea una clase llamada "Control". Crea 2 personajes y coloca 10 de vida y 0 puntos, de preferencia trata de usar constructores y destructores, getters y setters según corresponda y crea la clase "Control "de modo que se convierta en una clase singleton y resuelve:
    - Crea un método en la clase "Control" que otorgue puntos a los jugadores según el enemigo que hayan derrotado, envía los argumentos de forma manual y que ganen puntos según el tipo de enemigo a derrotar (Trata de solo usar una instancia o directamente usar métodos estáticos mientras se llama la instancia de la clase)
    - Haz que la clase "Control" tenga un contador de puntos totales y haz que los jugadores tengan acceso a la única instancia de "Control" desde su propia clase. Imprime el resultado
    - Crea un método en la clase control donde los jugadores pierdan vida y actualiza la vida del jugador atacado

En base al patrón de diseño *Builder*, responde:

1. ¿Qué es el Builder?
2. ¿Qué problemas resuelve el Builder?
3. En el constructor de la clase, sean cuales sean los valores que implementes, ¿sabes si hay alguna forma de saber en qué orden deben ir los atributos?
4. Crea una clase llamada Pizza, usando Builder genera:
    - Una pizza con tomate y queso
    - Una pizza con champiñones, carne y pepperoni
    - Una pizza especialidad

En base al patrón de diseño *Factory*, responde:

1. ¿Qué es el *Factory*?
2. ¿Qué problemas resuelve el *Factory*?