TP N°9

Revisión de Clases - Getters y Setters - Constructor y Destructor

Alumno: Eric Galera

Curso: 5to 5ta

A- Resolver las siguientes actividades:

1. ¿Qué es el encapsulamiento? ¿Cuáles son los niveles de acceso a miembros de la clase?

Encapsulamiento es uno de los principios fundamentales de la programación orientada a objetos (POO). Consiste en ocultar los detalles internos de una clase y solo exponer aquellos elementos que son necesarios para el uso externo. Esto se logra controlando el acceso a los datos y métodos de una clase, protegiendo así la integridad de los datos y mejorando la modularidad y mantenibilidad del código.

Niveles de acceso a miembros de la clase en C++:

Public: Los miembros declarados como públicos son accesibles desde cualquier parte del programa.

Protected: Los miembros declarados como protegidos solo son accesibles desde la clase misma, las clases derivadas y las funciones amigas.

Private: Los miembros declarados como privados solo son accesibles desde la clase misma y las funciones amigas.

2. ¿Qué son los getters y setters? ¿Cuál es su propósito principal? ¿Cuándo es conveniente usarlos?

Getters y setters son métodos usados para acceder y modificar los atributos privados de una clase.

Getter: Método para obtener el valor de un atributo privado.

Setter: Método para establecer o modificar el valor de un atributo privado.

Propósito principal: Su propósito principal es encapsular y controlar el acceso a los atributos de una clase y proporcionar una forma controlada de acceder y modificar los datos, permitiendo validaciones y otras lógicas adicionales.

Cuándo es conveniente usarlos:

- Cuando necesitas controlar el acceso a los atributos privados.

- Cuando se requiere realizar validaciones o ajustes al establecer o obtener los valores de los atributos.

- Para mantener la encapsulación y la integridad de los datos.

3. ¿Cuál es la diferencia entre un constructor y un destructor? ¿Cuándo se ejecutan cada uno?

Constructor: Es un método especial que se llama automáticamente cuando se crea una instancia de una clase. Se usa para inicializar los objetos de la clase

se ejecuta al crear un objeto.

Destructor: Es un método especial que se llama automáticamente cuando un objeto de una clase es destruido. Se usa para liberar los recursos que el objeto pueda haber adquirido durante su vida.

Se ejecuta al destruir un objeto, ya sea al salir del ámbito de un objeto de pila, cuando se llama a delete para un objeto de la memoria dinámica o cuando el programa termina.

4. Realizar un cuadro comparativo entre las características de constructores y destructores

Característica	Constructor	Destructor
Propósito	Inicializar un objeto	Liberar recursos antes de destruir un objeto
Nombre	Igual al nombre de la clase	Igual al nombre de la clase precedido por Igual al nombre de la clase precedido por ´ ~ ´
Parámetros	Puede aceptar parámetros	No acepta parámetros
Sobrecarga	Puede ser sobrecargado	No puede ser sobrecargado
Invocación	Automáticamente al crear un objeto	Automáticamente al destruir un objeto
Herencia	Invoca al constructor de la clase base si no se especifica	Invoca al destructor de la clase base automáticamente

5. Investigar la diferencia entre definir una función miembro completamente dentro de la definición de la clase o simplemente incluir su declaración en la función y definirla después fuera de la clase. Elaborar un cuadro comparativo incluyendo ventajas, desventajas y cuando usar cada opción.

Aspecto	Definición dentro de la clase	Definición fuera de la clase
Ventajas	- Código más conciso y directo - Puede mejorar la legibilidad si es corto	- Mejora la claridad al separar la interfaz de la implementación - Puede reducir el tiempo de compilación si el archivo de cabecera es incluido en varios lugares

Desventajas	- Puede hacer que el archivo de cabecera sea más largo y menos legible - Aumenta el tiempo de compilación si hay muchos cambios en el archivo de cabecera	- Puede hacer que el código sea menos intuitivo al separar la declaración de la implementación
Uso recomendado	- Para funciones muy cortas (ej. getters y setters)	- Para funciones más largas o complejas - Para mantener archivos de cabecera limpios y legibles