Introduccion

Problema:

Supongamos que eres el administrador de un centro de estudios que ofrece clases de tutoría para estudiantes de todas las edades. Quieres crear un programa para gestionar la lista de estudiantes inscritos en tu centro. Deseas clasificar a los estudiantes según si son menores o mayores de edad, y también deseas poder eliminar estudiantes de esta lista cuando se retiren del centro.

Solución:

Desarrollar un programa en C++ que permita al administrador del centro de estudios gestionar la lista de estudiantes. El programa solicitará al usuario que ingrese el nombre, la edad y el coeficiente intelectual de cada estudiante. Luego, clasificará a los estudiantes en dos grupos: menores de edad y mayores de edad. Además, proporcionará la funcionalidad para eliminar estudiantes de la lista.

Desarrollo

Aquí hay una descripción general de lo que hace el código:

Definición de las clases Persona, Adulto y Menor:

Persona es una clase base que almacena información común a todas las personas, como el nombre, la edad y el coeficiente intelectual.

Adulto y Menor son clases derivadas de Persona, especializadas para adultos y menores respectivamente. Cada una de estas clases tiene su propio método mostrar() para imprimir la información específica de cada tipo de persona.

Funciones para la gestión de la lista de personas:

agregarPersona(): Solicita al usuario que ingrese información sobre una persona y la agrega a la lista correspondiente de adultos o menores.

mostrarLista(): Muestra la lista de personas, invocando el método mostrar() de cada objeto en la lista.

eliminarPersona(): Solicita al usuario el nombre de la persona que desea eliminar de la lista y la elimina si se encuentra.

Función main():

Inicializa dos listas, una para adultos y otra para menores.

Muestra un menú de opciones para que el usuario pueda agregar, mostrar o eliminar personas de la lista, y salir del programa.

Dentro del bucle do-while, según la opción seleccionada por el usuario, llama a las funciones correspondientes para llevar a cabo la acción solicitada.

Liberación de memoria:

Al final del main(), el programa libera la memoria de los objetos creados dinámicamente en las listas de personas.

En resumen, el código utiliza herencia de clases para modelar diferentes tipos de personas, proporciona funciones para agregar, mostrar y eliminar personas de

la lista, y gestiona la memoria correctamente para evitar fugas de memoria. Esto permite una gestión eficiente de la lista de personas, clasificándolas según su edad y permitiendo al usuario realizar operaciones sobre ellas según sus necesidades.

Concluciones

La problemática planteada, que consiste en la gestión de una lista de personas clasificadas por edad y la posibilidad de eliminar personas de la lista, se aborda de manera efectiva con la solución propuesta. Aquí hay algunas conclusiones sobre esta problemática y su solución:

Organización y clasificación: La solución permite organizar a las personas en dos categorías distintas: adultos y menores de edad. Esto facilita la gestión y el análisis de la lista, ya que se pueden aplicar acciones específicas según la categoría a la que pertenezcan las personas.

Interfaz de usuario clara: La implementación de un menú de opciones en el programa proporciona una interfaz clara y fácil de usar para el usuario. Esto facilita la interacción con el programa y permite realizar operaciones específicas de manera intuitiva.

Flexibilidad: La solución es flexible y fácilmente adaptable a diferentes necesidades. Por ejemplo, se podrían agregar más categorías de personas o funcionalidades adicionales según los requisitos específicos del sistema.

Gestión de memoria segura: Se presta especial atención a la gestión de la memoria dinámica, asegurándose de liberar correctamente la memoria asignada a los objetos cuando ya no son necesarios. Esto evita fugas de memoria y garantiza un uso eficiente de los recursos del sistema.

Abstracción y reutilización de código: La implementación utiliza conceptos de programación orientada a objetos, como la herencia y el polimorfismo, lo que permite una estructura modular y la reutilización de código. Esto facilita el mantenimiento del programa y su escalabilidad a medida que crecen los requisitos.

En conclusión, la solución propuesta proporciona una forma efectiva y eficiente de gestionar una lista de personas clasificadas por edad, permitiendo al usuario realizar operaciones de forma fácil y segura.