Harl 2.0

El ejercicio propuesto, llamado Harl 2.0, consiste en implementar una clase que representa a "Harl", un personaje que realiza comentarios clasificados en diferentes niveles: DEBUG, INFO, WARNING y ERROR. Cada nivel representa un tipo de mensaje con características específicas:

- DEBUG:

  Comentarios utilizados para diagnosticar problemas.
  Ejemplo: "¡Me encanta tener bacon extra para mi hamburguesa 7XL con doble de queso, triple de pepinillos y especial con ketchup! ¡De verdad que sí!"

- INFO:

  Mensajes que proporcionan información detallada sobre la ejecución del programa.
  Ejemplo: "No puedo creer que agregar bacon extra cueste más dinero. ¡No pusiste suficiente bacon en mi hamburguesa! Si lo hubieras hecho, ¡no pediría más!"

- WARNING:

  Mensajes que indican un posible problema, aunque manejable.
  Ejemplo: "Creo que merezco un poco más de bacon gratis. He estado viniendo durante años, mientras que tú comenzaste a trabajar aquí el mes pasado".

- ERROR:

  Mensajes críticos que requieren intervención manual.
  Ejemplo: "¡Esto es inaceptable! Quiero hablar con el gerente ahora."

Objetivo del Ejercicio

→ Crear una clase llamada Harl con las siguientes características:

- Funciones privadas:

  void debug(void): Imprime mensajes de nivel DEBUG.
  void info(void): Imprime mensajes de nivel INFO.
  void warning(void): Imprime mensajes de nivel WARNING.
  void error(void): Imprime mensajes de nivel ERROR.

- Función pública:

  void complain(std::string level): Dependiendo del nivel pasado como parámetro ("DEBUG", "INFO", "WARNING", "ERROR"), esta función debe llamar a la función privada correspondiente.

Requisitos Técnicos

- Prohibición: No se pueden usar estructuras de control como if/else if/else para implementar la lógica en complain.
- Requerimiento: Se deben usar punteros a funciones miembro para implementar la selección dinámica de las funciones.

Archivos a entregar:

- Makefile: Para compilar el programa.
- Archivos del proyecto: main.cpp, Harl.hpp, Harl.cpp.

Cómo abordar el ejercicio:

- Entiende la clase:

  Harl encapsula cuatro comportamientos distintos (niveles de mensajes) que se implementan como funciones privadas.

- Usa punteros a funciones miembro:

  Define un arreglo o mapa de punteros a las funciones privadas (debug, info, warning, error).
  En complain, selecciona y llama la función correspondiente según el nivel recibido como parámetro.

- Prueba exhaustiva:

  Crea un conjunto de pruebas en main.cpp para demostrar que Harl genera mensajes correctos en cada nivel.
  Utiliza los ejemplos proporcionados u otros propios.

Estructura del código:

Sigue buenas prácticas en la organización del código, separando la implementación (Harl.cpp) y la declaración (Harl.h/hpp).

Resumen de cómo afrontarlo

- Define las funciones miembro: Implementa las funciones privadas para cada nivel y asegúrate de que impriman los mensajes correctos.
- Implementa complain con punteros a funciones: Diseña una estructura eficiente para asociar los niveles de mensaje con las funciones correspondientes.
- Pruebas completas: Diseña casos de prueba para verificar que los mensajes sean los esperados en todos los niveles.
- Compila y depura: Usa el Makefile para compilar y verificar que todo funcione correctamente.

Harl.hpp

Este código es la definición de la clase Harl en un archivo de cabecera (Harl.hpp). Vamos a desglosar cada parte para entender qué hace y cómo está relacionado con el ejercicio.

1. Definición Básica

   #ifndef HARL_HPP ... #endif

   Esta es una guarda de inclusión. Se utiliza para evitar que el contenido del archivo se incluya más de una vez durante la compilación, lo que podría causar errores.

2. Clase Harl

   Sección Pública. La parte pública de la clase que contiene:

   - Constructor: Harl(); que define cómo se inicializa un objeto de la clase Harl.
   - Método complain: void complain(std::string severity); ´nico método público de la clase. Según el parámetro severity (el nivel de severidad: "DEBUG", "INFO", "WARNING", "ERROR"), este método decide cuál función privada (debug, info, warning, error) debe ejecutar. Esto se hace utilizando punteros a funciones miembro.

   Sección Privada

   - Contiene los elementos internos de la clase que no son accesibles desde fuera, pero que forman su funcionamiento.

     Estructura hadlers

     ```
     struct _handlers {
       std::string severity;
       void (Harl::*f)(void);
     };
     ```

     Define una estructura con dos miembros:

     - severity: Un std::string que almacena el nivel de severidad asociado (por ejemplo, "DEBUG").
     - f: Un puntero a una función miembro de Harl con la firma void (Harl::*)(void). Este puntero apunta a una de las funciones privadas de Harl (por ejemplo, debug, info, etc.).

     Esta estructura permite asociar cada nivel de severidad con su correspondiente función.

   Arreglo handlers:

```
struct _handlers handlers[4];
```

- Es un arreglo de cuatro elementos, donde cada elemento es una instancia de _handlers.
- Almacena las asociaciones entre los niveles de severidad ("DEBUG", "INFO", "WARNING", "ERROR") y sus respectivas funciones.

Funciones Privadas:

```
void debug(void);
void info(void);
void warning(void);
void error(void);
```

Estas funciones implementan las acciones correspondientes a cada nivel de severidad. No son accesibles directamente desde fuera de la clase. Se llaman a través del puntero a función almacenado en el arreglo handlers.

¿Qué Hace el Código?

- Define una clase Harl que encapsula un comportamiento (emitir mensajes según un nivel de severidad).
- Usa una estructura (_handlers) para relacionar:
- Cada nivel de severidad con un puntero a la función que debe ejecutarse.
- Facilita que el método complain seleccione dinámicamente qué función ejecutar según el nivel pasado como parámetro.

¿Cómo Funciona en la Práctica?

En el constructor de Harl, el arreglo handlers será configurado para asociar cada nivel de severidad ("DEBUG", "INFO", etc.) con su función correspondiente (debug, info, etc.).

Ejecución:

Cuando se llama a complain("INFO"), el método busca en el arreglo handlers el nivel "INFO". Encuentra el puntero a la función correspondiente (&Harl::info) y lo ejecuta mediante la sintaxis de punteros a funciones miembro.

Ejemplo simplificado:

```
for (int i = 0; i < 4; i++) {
  if (handlers[i].severity == severity) {
    (this->*handlers[i].f)(); // Llama a la función correspondiente
    break;
  }
}
```

En resumen

- Este código organiza y prepara la clase Harl para que pueda ejecutar funciones asociadas a niveles de severidad dinámicamente, sin usar una cascada de if/else.
- El diseño aprovecha punteros a funciones miembro y un arreglo de estructuras (handlers) para mantener el código limpio y eficiente.

main.cpp

El código presentado es un programa sencillo que utiliza la clase Harl definida en el ejercicio. Su objetivo es demostrar el funcionamiento de la clase al generar mensajes en función de los niveles de queja (DEBUG, INFO, WARNING, ERROR). Vamos a desglosarlo paso a paso para comprender qué hace y cómo lo hace.

Función principal main:

Harl harl;:

- Se crea una instancia de la clase Harl llamada harl. Esto permite usar las funciones miembro de la clase, incluyendo complain.

harl.complain("DEBUG"); (y similares):

- Se llama a la función pública complain con diferentes niveles de mensaje como argumento ("DEBUG", "INFO", "WARNING", "ERROR").
- La función complain evalúa el argumento pasado y, utilizando punteros a funciones (según lo requerido en el ejercicio), llama a la función privada correspondiente:
  "DEBUG" → llama a la función privada debug.
  "INFO" → llama a la función privada info.
  "WARNING" → llama a la función privada warning.
  "ERROR" → llama a la función privada error.

¿Cómo Funciona complain?

- En el archivo Harl.cpp (la implementación), complain utiliza un arreglo o mapa de punteros a las funciones privadas (debug, info, warning, error) para relacionar el nivel de queja recibido como argumento con la función que debe ejecutarse.
- En lugar de usar un if/else, accede directamente a la función correcta mediante el nivel pasado como parámetro ("DEBUG", "INFO", etc.).

En resumen:

Propósito del código: Mostrar cómo la clase Harl genera mensajes específicos según los niveles de queja. En cuanto a su funcionamiento:

- Se crea una instancia de Harl.
- Se llama a complain con diferentes niveles como argumento.
- Según el nivel, complain ejecuta una función privada de la clase para imprimir el mensaje correspondiente.

- Resultado esperado: Una serie de mensajes en la consola, uno por cada nivel de queja.