NOW WE'RE TALKING

El ejercicio está orientado a la creación de una clase Fixed que representa números en formato de punto fijo en C++, y tiene como objetivo enseñar la sobrecarga de operadores y el uso de funciones estáticas, temas clave en la programación orientada a objetos en C++.

Resumen del Ejercicio

1. **Objetivo Principal**: Desarrollar una clase Fixed que simule el comportamiento de números en punto fijo, permitiendo:
    a. Sobrecargar los operadores de comparación (>, <, >=, <=, ==, !=).
    b. Sobrecargar los operadores aritméticos (+, -, *, /).
    c. Sobrecargar los operadores de incremento y decremento (pre y post incrementos y decrementos).
    d. Implementar funciones estáticas para encontrar el valor mínimo y máximo entre dos números Fixed, tanto si son modificables como constantes.
2. **Requerimientos**:
    a. Crear una clase Fixed con los operadores mencionados.
    b. Implementar métodos estáticos min y max que puedan trabajar con objetos modificables y constantes.
    c. Probar la clase utilizando el código de ejemplo proporcionado.
3. **Archivos a Entregar**:
    a. Makefile: para compilar el proyecto.
    b. main.cpp: el archivo que contiene la función main para probar el código.
    c. Fixed.{h, hpp}: los archivos de cabecera donde se declara la clase.
    d. Fixed.cpp: el archivo donde se implementan las funciones de la clase.
4. **Comportamiento Esperado**:
    a. Se imprimen los resultados de las operaciones en la consola, demostrando el funcionamiento de los operadores sobrecargados.

Objetivo de Aprendizaje

El ejercicio tiene como propósito enseñar y practicar los siguientes conceptos de C++:

- **Sobrecarga de Operadores**: permitir que los operadores estándar (comparación, aritméticos, y de incremento/decremento) funcionen con objetos de la clase Fixed.
- **Funciones Estáticas**: cómo trabajar con funciones estáticas que no dependen de instancias de la clase.
- **Clases en C++**: creación de clases con miembros estáticos y no estáticos, y la implementación de funcionalidades que manipulan objetos.

- **C++ 98**: utilizar C++ 98, lo que implica no utilizar características más modernas de C++ como nullptr, auto, y otras herramientas introducidas en versiones posteriores.

Materias de C++ a Desarrollar

- **Clases y Objetos**: Se desarrollan conceptos fundamentales de las clases en C++, como la declaración de clases, funciones miembro, y constructores/destructores.
- **Sobrecarga de Operadores**: Es una de las partes clave del ejercicio, aprenderás cómo redefinir el comportamiento de los operadores estándar para que funcionen con objetos de la clase Fixed.
- **Funciones Estáticas**: Se aprenderá a declarar e implementar funciones que no necesitan una instancia de la clase para ser llamadas.
- **Manejo de Tipos de Datos**: Implementación y manipulación de números en punto fijo, lo cual es fundamental para representar valores decimales en sistemas con recursos limitados.

Mejor Forma de Afrontar el Ejercicio

1. **Establecer la Clase Fixed**: Comienza creando la estructura básica de la clase con sus atributos y métodos. Un buen punto de partida es crear los miembros necesarios para almacenar los valores en punto fijo y su escala.
2. **Sobrecargar Operadores**: Define las sobrecargas de los operadores aritméticos y de comparación, asegurándote de manejar correctamente el tipo de datos en punto fijo y la conversión entre tipos (por ejemplo, de float a Fixed).
3. **Operadores de Incremento/Decremento**: Implementa las versiones pre y post de los operadores de incremento y decremento. Asegúrate de que el incremento y decremento sea muy pequeño, como se describe en el enunciado.
4. **Funciones Estáticas**: Implementa las funciones estáticas min y max, que comparan dos objetos Fixed y devuelven el más pequeño o el más grande, respectivamente.
5. **Pruebas**: Prueba el comportamiento de los operadores y las funciones estáticas utilizando el código de ejemplo proporcionado en la descripción del ejercicio. Asegúrate de que la salida sea correcta, especialmente con respecto a los valores de los números en punto fijo.
6. **Documentación y Makefile**: Es importante incluir un Makefile funcional para compilar el proyecto, así como comentarios en tu código que expliquen las decisiones y la lógica.

Al seguir estos pasos, abordarás de manera efectiva el ejercicio y alcanzarás los objetivos de aprendizaje de manera exitosa.

Fixed.hpp - Fixed.cpp - main.cpp

Explicación del Código

El programa define una clase Fixed que implementa números en formato de punto fijo y sobrecarga varios operadores para trabajar con ellos. Además, incluye un programa principal (main.cpp) que prueba las funcionalidades de la clase.

Detalle de las Funcionalidades

1. Clase Fixed

1. **Atributos**:
    a. _value: almacena el valor del número en punto fijo como un entero.
    b. _fractional_bits: constante estática que define cuántos bits están reservados para la parte fraccional (8 en este caso).
2. **Constructores**:
    a. **Por defecto** (Fixed()) inicializa _value en 0.
    b. **Por copia** (Fixed(const Fixed &src)) permite copiar otro objeto Fixed.
    c. **Con entero** (Fixed(const int val_int)) convierte un entero en punto fijo desplazando el valor entero 8 bits a la izquierda.
    d. **Con flotante** (Fixed(const float val_float)) convierte un flotante en punto fijo multiplicando el flotante por 2^8 (para moverlo 8 bits hacia la izquierda) y redondeando el resultado.
3. **Métodos de Conversión**:
    a. toFloat(): convierte _value a flotante dividiéndolo por 2^8.
    b. toInt(): convierte _value a entero desplazándolo 8 bits a la derecha.
4. **Métodos Get/Set**:
    a. getRawBits(): retorna el valor bruto almacenado en _value.
    b. setRawBits(): establece el valor de _value directamente.
5. **Sobrecarga de Operadores**:
    a. **Aritméticos** (+, -, *, /): realizan las operaciones correspondientes sobre los valores _value.
        i. La multiplicación y la división ajustan el resultado para manejar la representación de punto fijo.
    b. **Comparación** (>, <, >=, <=, ==, !=): comparan los valores almacenados en _value.
    c. **Incremento y Decremento**:
        i. Pre (++a) y post (a++) incrementan o decrementan _value en 1 (el incremento más pequeño representable en el formato).
6. **Funciones Estáticas**:
    a. min(): retorna el menor de dos objetos Fixed (referencia constante o modificable).
    b. max(): retorna el mayor de dos objetos Fixed (referencia constante o modificable).

7. **Sobrecarga del Operador <<**:
    a. Permite imprimir un objeto Fixed como un flotante usando toFloat().

2. Implementación del main

Este programa principal prueba las capacidades de la clase Fixed. A continuación, explicamos el comportamiento de cada línea:

Fixed a;

- Declara un objeto Fixed usando el constructor por defecto. Imprime: Default constructor called.

Fixed const b(Fixed(5.05f) * Fixed(2));

- Crea un objeto Fixed temporal con el valor 5.05 usando el constructor de flotantes.
- Crea otro objeto Fixed temporal con el valor 2 usando el constructor de enteros.
- Multiplica ambos objetos (*), ajustando el resultado al formato de punto fijo.
- Asigna el resultado al objeto constante b.

- std::cout << a << std::endl;
  Imprime el valor de a (inicialmente 0). Llama a toFloat() para convertir _value a flotante.
- std::cout << ++a << std::endl;
  Incrementa a en el menor valor representable (preincremento) y lo imprime.
- std::cout << a << std::endl;
  Imprime nuevamente el valor actualizado de a.
- std::cout << a++ << std::endl;
  Imprime el valor actual de a y luego lo incrementa (postincremento).
- std::cout << a << std::endl;
  Imprime el valor incrementado de a.
- std::cout << b << std::endl;
  Imprime el valor de b como flotante.
- std::cout << Fixed::max(a, b) << std::endl;
  Llama a la función estática max() para obtener el mayor entre a y b. Lo imprime como flotante.

Cómo Funciona el Código

1. La clase Fixed usa un entero _value para almacenar números en punto fijo, donde los últimos 8 bits representan la parte fraccional.
2. Los operadores están sobrecargados para realizar cálculos y comparaciones directamente entre objetos Fixed.
3. Las conversiones entre punto fijo, flotante e integer están encapsuladas en los métodos de conversión toFloat() y toInt().
4. Las funciones estáticas min() y max() comparan dos objetos Fixed y retornan una referencia al menor o mayor.

Salida del Programa

La salida del programa (sin los mensajes de constructor/destructor) será algo similar a:

```
0
0.00390625
0.00390625
0.00390625
0.0078125
10.1016
10.1016
```

Esta secuencia muestra:

- La inicialización de a y sus incrementos.
- La creación de b como resultado de la multiplicación de 5.05 y 2.
- El valor máximo entre a y b.