Sobrecarga del operador << en la clase complejo

1. ¿Qué es el operador <<?

En C++ el operador << se usa con streams de salida como std::cout para mostrar datos en pantalla. Funciona con tipos básicos (int, double, string), pero el compilador no sabe cómo imprimir objetos definidos por el usuario, como complejo. Por eso es necesario sobrecargar operator<<, para enseñar al compilador cómo mostrar un objeto complejo.

2. ¿Por qué no puede ser un método miembro?

Si intentáramos implementarlo como miembro de la clase, la llamada sería c.operator<<(std::cout), lo cual es poco intuitivo. Lo que realmente queremos es std::cout << c. Aquí el primer operando es std::cout (un objeto ostream), no un complejo. Por tanto, no puede resolverse con un método miembro de complejo. Necesitamos una función no miembro.

3. ¿Por qué usar una función amiga?

Las funciones no miembro no tienen acceso a los atributos privados de la clase. Para imprimir los valores real e imaginario, hay dos alternativas:
1. Usar los métodos getr() y geti().
2. Declarar operator<< como función amiga (friend), lo que le permite acceder directamente a los atributos privados.

En la práctica indicada, está permitido tener una única función amiga, y esta es la candidata ideal.

4. Declaración en la clase

Dentro de la clase complejo se escribe:

```
friend std::ostream operator<<(std::ostream os, const complejo c);
```

- friend → da acceso a atributos privados.
- std::ostream → se devuelve el flujo para permitir encadenar impresiones (cout << c1 << c2).
- const complejo c → se pasa por referencia constante, evitando copias y garantizando que no se modifica el objeto.

5. Implementación fuera de la clase

La definición se hace fuera de la clase como función libre:

```
std::ostream operator<<(std::ostream os, const complejo c) {
  os << c.real;
  if (c.imaginario >= 0) os << "+" << c.imaginario << "i";
```

```
    else os << c.imaginario << "i";
    return os;
}
```

Explicación paso a paso del punto 5

1. Cabecera de la función: Devuelve una referencia a std::ostream y recibe dos parámetros:
- os: el flujo de salida (ejemplo: std::cout).
- c: el objeto de tipo complejo a imprimir, pasado como referencia constante.
Esto permite escribir expresiones como std::cout << c1;

2. Primera salida (parte real): La instrucción os << c.real; envía la parte real del número complejo al flujo de salida.

3. Impresión de la parte imaginaria si es positiva o cero: Si c.imaginario >= 0, se imprime el signo '+' explícito, seguido del número y la letra i. Por ejemplo, si c.imaginario = 3, la salida será '+3i'.

4. Impresión de la parte imaginaria si es negativa: Si la parte imaginaria es negativa, el signo '-' aparece automáticamente. Por ejemplo, si c.imaginario = -4, la salida será '-4i'.

5. Retorno del flujo de salida: La instrucción return os; devuelve el flujo de salida modificado. Esto permite encadenar operaciones de impresión como std::cout << c1 << c2 << c3;.

6. Uso

Con esta sobrecarga, imprimir un complejo es tan natural como imprimir un entero:

```
complejo z1(1,3), z2(-2,0);
std::cout << "z1 = " << z1 << std::endl;
std::cout << "z2 = " << z2 << std::endl;
```

La salida sería:
z1 = 1+3i
z2 = -2+0i

7. Resumen didáctico

- El operador << es necesario para integrar la clase complejo con std::cout.
- No puede ser método miembro porque el primer operando es ostream.
- Se implementa como función no miembro.
- Se declara como friend en la clase para poder acceder a los atributos privados.
- Devuelve ostream& para permitir encadenar impresiones.
- Es la única función amiga permitida en la práctica.