Proyecto Final de Programación: Sistema Bancario en C++

1. Nombre del Proyecto

Sistema Bancario en C++ con Gestión de Cuentas y Préstamos

Fecha de entrega: 22/06/2025

2. Descripción General

Este proyecto tiene como objetivo el desarrollo de un sistema bancario completo en C++, orientado a reforzar el uso de estructuras y conceptos avanzados del lenguaje como:
- Programación orientada a objetos (clases y herencia)
- Plantillas (templates) para implementar listas genéricas reutilizables
- Manejo de archivos binarios para la persistencia de datos
- Sobrecarga de operadores para hacer más intuitivas las operaciones con objetos del sistema.

El sistema permitirá la gestión de distintos tipos de cuentas bancarias (ahorro, nómina e inversión), así como el registro de préstamos personales para los clientes. El propósito pedagógico del proyecto es integrar múltiples temas del curso en una solución práctica, aplicable a problemas reales.

3. Objetivos

Objetivo General:
Desarrollar un sistema bancario en C++ que permita gestionar cuentas y préstamos de clientes, incorporando estructuras genéricas, persistencia de datos y técnicas de sobrecarga de operadores.

Objetivos Específicos:

- Implementar una lista genérica mediante plantillas (templates) para almacenar cuentas y préstamos.

- Definir clases específicas para distintos tipos de cuentas: ahorro, nómina e inversión.

- Registrar préstamos personales ligados a un cliente, incluyendo monto, tasa y plazo.

- Guardar y recuperar la información de manera persistente usando archivos binarios.

- Utilizar sobrecarga de operadores para facilitar operaciones entre objetos como comparación, impresión y cálculos financieros.

4. Tipos de Cuentas Bancarias

Cuenta de Ahorro:
- Acumula intereses sobre el saldo.
- Permite depósitos y retiros.
- Tiene restricciones en la cantidad de retiros mensuales.

- Debe existir un proceso masivo para cargar los beneficios en las cuentas.

Cuenta de Nómina:
- Asignada a empleados para recibir depósitos periódicos (salario).
- No genera intereses.
- Puede tener beneficios como retiros sin comisión.

Cuenta de Inversión:
- Diseñada para obtener rendimientos.
- Aplica una tasa de interés mayor pero restringe retiros anticipados.
- Se usa para simular productos como certificados de depósito.

- Debe existir un proceso masivo para cargar los beneficios en las cuentas.

5. Préstamos Personales

Cada préstamo se vincula a un número de cuenta. Almacena monto, tasa de interés, número de cuotas. Se puede calcular el monto total a pagar y la cuota mensual. También se almacenan en archivos binarios.

6. Funcionalidades del Sistema

- Crear cloentes y sus respectivas cuentas (de ahorro, nómina, inversión).

- Listar cuentas registradas.

- Realizar depósitos y retiros.

- Registrar préstamos personales.

- Consultar detalles de un préstamo.

- Generar procesos masivos (correr interes en prestamos y rendimientos)

- Guardar y recuperar información en archivos binarios (un solo archivo para el sistema).

- Buscar cuentas o préstamos por número.

7. Sobrecarga de Operadores

- << para mostrar detalles de cuentas o préstamos.

- == para buscar elementos por ID.

- + y - para depositar y retirar.

- * para calcular intereses o cuotas (en préstamos).

Ejemplo:
```
CuentaAhorro c1(101, "Luis", 5000);
c1 + 1000;
c1 - 500;
std::cout << c1;
```

8. Persistencia con Archivos Binarios

Cada clase que contiene datos persistentes implementa métodos guardarBinario() y cargarBinario() que utilizan ofstream y ifstream en modo binario para almacenar y recuperar información.

9. Manejo de errores

Maneje los errores mediante exepciones, complete una table de errores que tenga el Código del error, nombre del error y descripción.

10. Cálculo de Préstamos y Fórmulas Financieras

En el sistema bancario se incorporan cálculos financieros para gestionar adecuadamente los préstamos. A continuación se detallan las fórmulas empleadas y su propósito dentro del sistema:

10.1 Tasa de Amortización

La tasa de amortización permite calcular el valor de la cuota fija mensual que un cliente debe pagar por un préstamo. Se usa la fórmula del sistema francés de amortización, común en instituciones financieras:

Fórmula:

$$C = P * (i * (1 + i)^n) / ((1 + i)^n - 1)$$

Donde:

C: Cuota mensual a pagar
P: Monto del préstamo (principal)
i: Tasa de interés mensual (tasa anual / 12)
n: Número total de pagos (cuotas)

Esta fórmula se implementa en el sistema para ofrecer a los usuarios una estimación precisa del compromiso mensual del préstamo.

10.2 Pagos Extraordinarios

El sistema también permite calcular el impacto de pagos extraordinarios (adelantos) en el saldo pendiente del préstamo.

Cuando un cliente realiza un pago extraordinario, se puede aplicar de dos formas:
- Reduciendo el número de cuotas restantes (acortamiento del plazo).
- Reduciendo el monto de la cuota mensual.

El sistema ajusta el saldo pendiente del préstamo y recalcula la amortización usando el nuevo valor principal.

10.3 Tabla de Amortización

El sistema puede generar una tabla de amortización mensual, que incluye:
- Número de cuota
- Saldo inicial
- Interés del período
- Amortización a capital
- Saldo final

Esto permite al usuario visualizar el comportamiento del préstamo a lo largo del tiempo, y cómo se va reduciendo la deuda.

10.4 Interés Total Pagado

El interés total pagado por un préstamo se calcula como:

Interés Total = (Cuota mensual * número de pagos) - Monto del préstamo

Este valor ayuda a los usuarios a conocer cuánto habrán pagado en intereses al finalizar el préstamo.

11. Conclusiones

Este sistema bancario permite aplicar conceptos clave del lenguaje C++ en un entorno simulado pero realista. La implementación de plantillas para listas genéricas promueve la reutilización de código. El manejo de archivos binarios proporciona persistencia, mientras que la sobrecarga de operadores mejora la legibilidad del código y su usabilidad. Los distintos tipos de cuenta y la gestión de préstamos demuestran una correcta aplicación de herencia y abstracción en programación orientada a objetos.