This program simulates a basic banking system using file handling and C++ standard libraries. It supports account creation, deposit, withdrawal, listing, deletion, viewing, and transferring money between accounts. Each account is stored in a separate file, and a persistent index.txt tracks all account numbers.

### Module Descriptions

#### **1. Account Structure** : Stores basic account information: number and name.

**2. Filename Generator** : Generates filenames for account (account\_<accNumber>.txt) and balance (balance\_<accNumber>.bin) files dynamically. Two separate files are created for handling account balance and other info for more security. The balance file is a binary file, making it non readable through text editors for better security.

**3. Account Management Helper Function**:

bool accountExists(int accNumber);

void loadAccountNumbers();

void addAccount(int accNumber);

void removeAccount(int accNumber);

Handle existence checks, loading, updating, and maintaining a global vector<int> accountNumbers from index.txt for persistent tracking of all accounts.

**4. Account Operations**

**Account Creation**

User is prompted for:Account number,Name,Initial balance  
The program checks if the account already exists. If it exists, the function terminates. If not,  
it creates a new text file account\_<accNumber>.txt and a binary file balance\_<accNumber>.bin and writes the initial balance using write(). Finally it updates accountNumbers vector and appends to index.txt.

### **View Account**

### The program asks for an user input - the account number. After that, the program opens both: The .txt file - reads account number and name. The .bin file - reads float balance using read(). And displays the retrieved data.

### **Deposit**

### User is required to enter an account number.Balance is read from balance\_<accNumber>.bin. New deposit amount is input.New balance is calculated and written back into the binary file using write().

### **Withdraw**

### Similar to deposit, this function also reads the current balance and asks the user for withdrawal amount. If balance is sufficient, deducts it and finally overwrites the binary file with the updated balance.

### **List All Accounts**

The function Iterates over the accountNumbers vector.For each account, it opens .txt to get name and account number. Then it opens the .bin binary file to get balance. Displays a formatted table using setw() and fixed. The index.txt file and loaded accountNumbers vector ensure only valid accounts are shown.

### **Delete Account**

### User is required to enter account number. After verifying the existence of the account,it deletes the two files (.bin and .txt) using remove().After this, the number is also removed from the vector to update the list of all accounts after deletion. Finally,it rewrites index.txt without the deleted number by calling the removeAccount() function.

### **Transfer Money**

### User is required to input source account number and destination account number and the amount to transfer. Program checks if both accounts exist and reads both balances from their .bin files

### Then the program checks if sender has sufficient funds, deducts amount from sender, adds to recipient and finally writes updated balances back to respective .bin files.

This modular banking system adopts a secure and efficient design by separating account information and financial data into distinct files. Specifically, account balances are stored in binary files to enhance data security, while general account details such as account number and name are saved in plain text files. The system maintains an index.txt file to keep track of all active account numbers. This index is loaded into a vector<int> at runtime, allowing for efficient listing, retrieval, and deletion of accounts. Overall, the system is structured to be both user-friendly and secure, offering reliable data handling and straightforward extensibility.