Code Structure

The program is written in C++ and is organized into several functions to enhance readability and maintainability. The main components of the code are as follows:

1. Data Structure

struct UserInfo: Defines a structure to store user information, including username, password, name, balance, and address.

2. File I/O Functions

map<string, UserInfo> readUserDatabase(const string& filename): Reads user information from a file into a map where the key is the username and the value is a UserInfo structure.

void writeUserDatabase(const string& filename, const map<string, UserInfo>& userDatabase): Writes user information from the map back to the file.

3. Authentication Functions

bool checkCredentials(const map<string, UserInfo>& userDatabase, const string& username, const string& password): Checks if the provided username and password match any user in the user database.

bool usernameExists(const map<string, UserInfo>& userDatabase, const string& username): Checks if a given username already exists in the user database.

4. Main Functionality

The main() function is the entry point of the program. It implements the main menu, login, and signup functionality. It interacts with the user and utilizes the functions mentioned above to perform various banking operations.

User Interface

The program provides a simple command-line interface with interactive menus for the user to navigate through different options. The main menu includes options for login, signup, and exiting the program. After logging in, users can perform operations like viewing account details, transferring money, depositing, withdrawing, and handling service payments.

How to Use

Compile the program using a C++ compiler.

Run the executable.

Follow the on-screen instructions to navigate through the main menu.

For login, enter a valid username and password. For signup, provide necessary information.

After successful login or signup, navigate through submenus to perform banking operations.

Exit the program when done.

Notes

User data is stored in a file named "user_data.txt".

Passwords are simple and stored in plaintext for educational purposes. In a real-world scenario, passwords should be securely hashed.

The program utilizes loops and goto statements for flow control. While this is generally discouraged, it simplifies the program's structure for educational purposes.

Conclusion

The ASTU Students Bank program provides a basic framework for a command-line banking system. Users can log in, sign up, and perform simple banking operations. The program can be extended and enhanced with additional features and security measures for practical use..

Algorithm

1. Include necessary libraries

- **iostream:** This library is used for standard input and output operations. It provides objects like **cin** (standard input) , **cout** (standard output) and **cerr**.
- **fstream:** This library is used for handling files.This library is used for handling files. It provides classes like **ifstream** (input file stream) and **ofstream** (output file stream) that enable reading from and writing to files.
- **string:** This library is used for string manipulation. It provides a versatile **string** class that simplifies working with strings, including functions for concatenation, comparison, and various other operations.
- **map**: This library provides the map container, which is an associative container that stores elements in key-value pairs. It allows efficient retrieval and insertion of elements based on their keys.
- **iomanip**; it is a header file in C++ that provides facilities for controlling the format of input and output. It stands for "input/output manipulators.

2. Define UserInfo struct

- Members information : username, password, name, balance, address

3. Define functions:

3.1 readUserDatabase(filename: string) -> map<string, UserInfo>

- Initialize an empty map userDatabase
- Open the file with the given filename
- If the file is open:
  - Read each line from the file
  - Extract username, password, name, balance, and address
  - Create a UserInfo object
  - Add the UserInfo object to the userDatabase map with the username as the key
- Close the file
- Return the userDatabase map

3.2 writeUserDatabase(filename: string, userDatabase: map<string, UserInfo>)

- Open the file with the given filename for writing
- If the file is open:
  - Loop through each entry in userDatabase
  - Write username, password, name, balance, and address to the file
- Close the file

3.3 checkCredentials(userDatabase: map<string, UserInfo>, username: string, password: string) -> bool

- Find the entry in userDatabase with the given username
- If the entry exists and the password matches, return true
- Otherwise, return false

3.4 usernameExists(userDatabase: map<string, UserInfo>, username: string) -> bool

- Check if the username exists in userDatabase
- Return true if found, false otherwise

4. Define main function:

4.1 Declare variables and constants:

- filename: string = "user_data.txt"
- userDatabase: map<string, UserInfo>
- user, password: string
- newBalance: double

4.2 Display welcome message and options:

- Display menu with options 1 (Login) and 2 (Sign Up)

4.3 Loop:

- Read user choice

4.3.1 If choice is 1 (Login):

- Loop for user login:
  - Read username and password
  - Check credentials using checkCredentials function
  - Display login success/failure
  - If successful, display user-specific options

4.3.1.1 Inside user-specific options loop:

- Read user's choice
- Implement corresponding actions (My account, Transfer, Deposit, Withdrawal, Service payment, Exit)
- Display results and options
- Handle sub-options and loops

4.3.2 If choice is 2 (Sign Up):

- Loop for user registration:
  - Read new user details
  - Check if username exists using usernameExists function
  - Register the new user
  - Display registration success/failure
  - If successful, provide option to go back to the main menu

4.3.3 Handle invalid choices and loops

5. End of the program