BANK ACCOUNT MANAGEMENT SYSTEM

C-Project by:
MANJUNATH E L

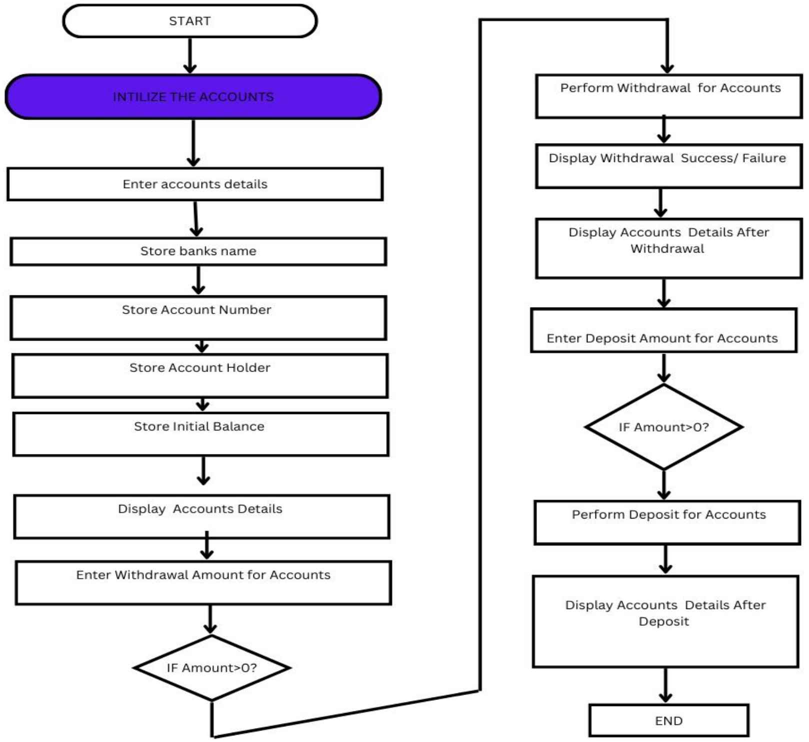
Introduction

The code you've provided is an implementation of a simple Bank account Management System in C. This system allows users to create and manage bank accounts, perform deposits, and withdrawals. It serves as a basic model to understand how data structures (in this case, a struct) can be used to represent and manage financial accounts.

The program is organized into several functions, each serving a specific purpose:

1. DisplayAccount: Displays the details of a bank account.
2. Withdraw: Allows users to make withdrawals from an account.
3. Deposit: Allows users to make deposits into an account.
4. Main: The main function that controls the flow of the program, including input, account creation, and transactions.

Flow chart


START
INTILIZE THE ACCOUNTS
Enter accounts details
Store banks name
Store Account Number
Store Account Holder
Store Initial Balance
Display Accounts Details
Enter Withdrawal Amount for Accounts
IF Amount>0?
Perform Withdrawal for Accounts
Display Withdrawal Success/ Failure
Display Accounts Details After Withdrawal
Enter Deposit Amount for Accounts
IF Amount>0?
Perform Deposit for Accounts
Display Accounts Details After Deposit
END

Project Description

Title: Bank Account Management System

This project is a Bank Account Management System developed in C. It allows users to create and manage two bank accounts, perform withdrawals, and make deposits. The system stores account details, including bank name, account number, account holder, and balance, using a structured approach. Users can input account information, view account details, and interactively perform transactions. Withdrawals and deposits are subject to validation checks, ensuring that operations are secure and accurate. The program displays success or failure messages for each transaction and updates account balances accordingly. It serves as a foundational system that can be expanded with additional features like transaction history and interest calculations.

Struct Definition

The code begins by defining a C struct called BankAccount. This struct is used to represent a bank account and consists of the following fields:

- Bank Name (char[50]): A character array to store the name of the bank.
- Account Number (int): An integer to store the account number.
- Account Holder (char[50]): A character array to store the name of the account holder.
- Balance (float): A floating-point number to store the account balance.

This struct is essential because it encapsulates all the information related to a bank account in a single data structure, making it easier to manage and manipulate account data.

Function Definitions

1. DisplayAccount

The DisplayAccount function is responsible for displaying the details of a bank account. It takes a BankAccount structure as a parameter and prints out the following information:

- Bank Name
- Account Number
- Account Holder
- Balance

The function uses the printf function to display this information in a user-friendly format. This function plays a crucial role in presenting account information to the user.

2. Withdraw

The withdraw function allows users to withdraw money from a bank account. It takes two parameters:

- acc (struct BankAccount*): A pointer to the BankAccount structure representing the account from which the withdrawal is to be made.
- amount (float): The amount to be withdrawn.

The function checks if the withdrawal amount is valid (greater than 0 and less than or equal to the account balance). If the withdrawal is valid, it deducts the amount from the account balance. If not, it informs the user that the withdrawal has failed due to insufficient funds.The function plays a critical role in managing the account's balance and ensuring that withdrawals are only allowed when funds are available.

Deposit

The deposit function allows users to deposit money into a bank account. It also takes two parameters:

- acc (struct BankAccount*): A pointer to the BankAccount structure representing the account into which the deposit is to be made.
- amount (float): The amount to be deposited.

The function checks if the deposit amount is valid (greater than 0) and adds the amount to the account's balance. If the amount is invalid, it informs the user of the failure.this function is crucial for increasing the account balance and facilitating deposits.

Main Function

The main function is the entry point of the program and controls its flow. Let's break down its functionality:

Program Initialization

- The program starts by displaying a welcome message to the user.

Account Creation

- Two BankAccount structures, account1 and account2, are declared to represent two bank accounts. The user is prompted to enter details for each account, including bank name, account number, account holder's name, and initial balance.
- This section demonstrates how the struct is used to store account information efficiently.

Displaying Account Details

- After account creation, the program displays the account details using the displayAccount function for both account1 and account2. This section illustrates how to access and display information stored in the struct.

Withdrawal Operations

- Users are prompted to enter withdrawal amounts for both accounts. The withdraw function is called to process these withdrawals.
- After each withdrawal, the program displays the updated account details. This demonstrates the impact of transactions on account balances.

Deposit Operations

- Similar to withdrawals, users are prompted to enter deposit amounts for both accounts. The deposit function is used to handle these deposits.
- After each deposit, the program displays the updated account details.

Functions

Displaying Account Details

The program includes a function displayAccount that takes a BankAccount structure as an argument and prints its details. This function is useful for displaying account information to the user.

```
void displayAccount(struct BankAccount acc)
{
printf("Bank Name: %s\n", acc.bankName);
printf("Account Number: %d\n", acc.accountNumber);
printf("Account Holder: %s\n", acc.accountHolder);
printf("Balance: $%.2f\n", acc.balance);
}
```

This function formats and prints the account's details in a user-friendly manner, including the bank name, account number, account holder's name, and balance.

Withdrawal and Deposit Functions

Two more functions, withdraw and deposit, are defined to perform withdrawal and deposit operations on a given bank account. These functions take a pointer to a BankAccount structure and the amount to withdraw or deposit as arguments.

Withdraw Function

```
void withdraw(struct BankAccount *acc, float amount) {

if (amount > 0 && amount <= acc->balance) {

acc->balance -= amount;

printf("Withdrawal of $%.2f successful.\n", amount);

} else {

printf("Withdrawal failed. Insufficient funds.\n");

}

}
```

The withdraw function checks if the withdrawal amount is valid and if the account has sufficient funds. If the conditions are met, it deducts the amount from the account's balance and provides a success message. Otherwise, it informs the user of the failure.

Deposit Function

```
void deposit(struct BankAccount *acc, float amount) {

if (amount > 0) {

acc->balance += amount;

printf("Deposit of $%.2f successful.\n", amount);

} else {

printf("Deposit failed. Invalid amount.\n");

}

}
```

The deposit function checks if the deposit amount is valid (greater than zero) and adds the amount to the account's balance. It also provides feedback to the user about the success or failure of the deposit.

User Input

The program prompts the user to input details for each account, including the bank name, account number, account holder's name, and initial balance. The scanf function is used to read these values from the user.

```
printf("Enter Account 1 details:\n");

printf("Bank Name: ");

scanf("%s", account1.bankName);

printf("Account Number: ");

scanf("%d", &account1.accountNumber);

printf("Account Holder: ");

scanf("%s", account1.accountHolder);

printf("Initial Balance: $");

scanf("%f", &account1.balance);
```

Displaying Account Details

After obtaining all the necessary information, the program prints the details of both accounts using the displayAccount function. This provides the user with an overview of the account information.

```
printf("\nAccount Details:\n");

printf("----------------\n");

displayAccount(account1);

printf("\n");

displayAccount(account2);
```

Performing Withdrawals

The program then proceeds to simulate withdrawals for both accounts. It prompts the user to enter the withdrawal amount for each account and uses the withdraw function to deduct the specified amount from the account's balance.

Displaying Updated Account Details

After each withdrawal, the program displays the updated account details to reflect the changes made by the withdraw function. This step provides transparency to the user about the account's new balance.

```
printf("\nAccount 1 Details After Withdrawal:\n");

printf("--------------------------------\n");

displayAccount(account1);


printf("\nAccount 2 Details After Withdrawal:\n");

printf("--------------------------------\n");

displayAccount(account2);
```

Performing Deposits: Similar to withdrawals, the program simulates deposits for both accounts. It prompts the user to enter the deposit amount for each account and uses the deposit function to add the specified amount to the account's balance.

Displaying Final Account Details

Finally, the program displays the updated account details after the deposits. This step provides the user with a complete overview of the account status.

```
printf("\nAccount 1 Details After Deposit:\n");
printf("----------------------------------\n");
displayAccount(account1);


printf("\nAccount 2 Details After Deposit:\n");
printf("----------------------------------\n");
displayAccount(account2);
```

Conclusion

In summary, the provided code implements a basic Bank Account Management System in C. It demonstrates the use of structures to represent bank accounts, functions to perform operations on accounts, and user interaction through the main function. This code serves as a foundation that can be further enhanced and extended to create a more comprehensive banking application with additional features such as transaction history, interest calculations, and account management

THANK YOU