5/24/2025

CP LAB PROJECT

SIR MOHSIN



Daud 017 Daud Mustafa 01-134251-017

**BASIC BANKING SYSTEM:**

# **Introduction to Our Project:**

This project presents a basic banking system developed in C++ that allows users to create accounts, deposit and withdraw money, and view account details. It demonstrates core concepts in programming such as file handling, arrays, and modular programming. Account data is persistently stored in a text file to ensure that user data is retained between program runs. The purpose of this project is to develop a foundational understanding of simple banking logic, data handling, and user interaction in C++.

# **System Design**

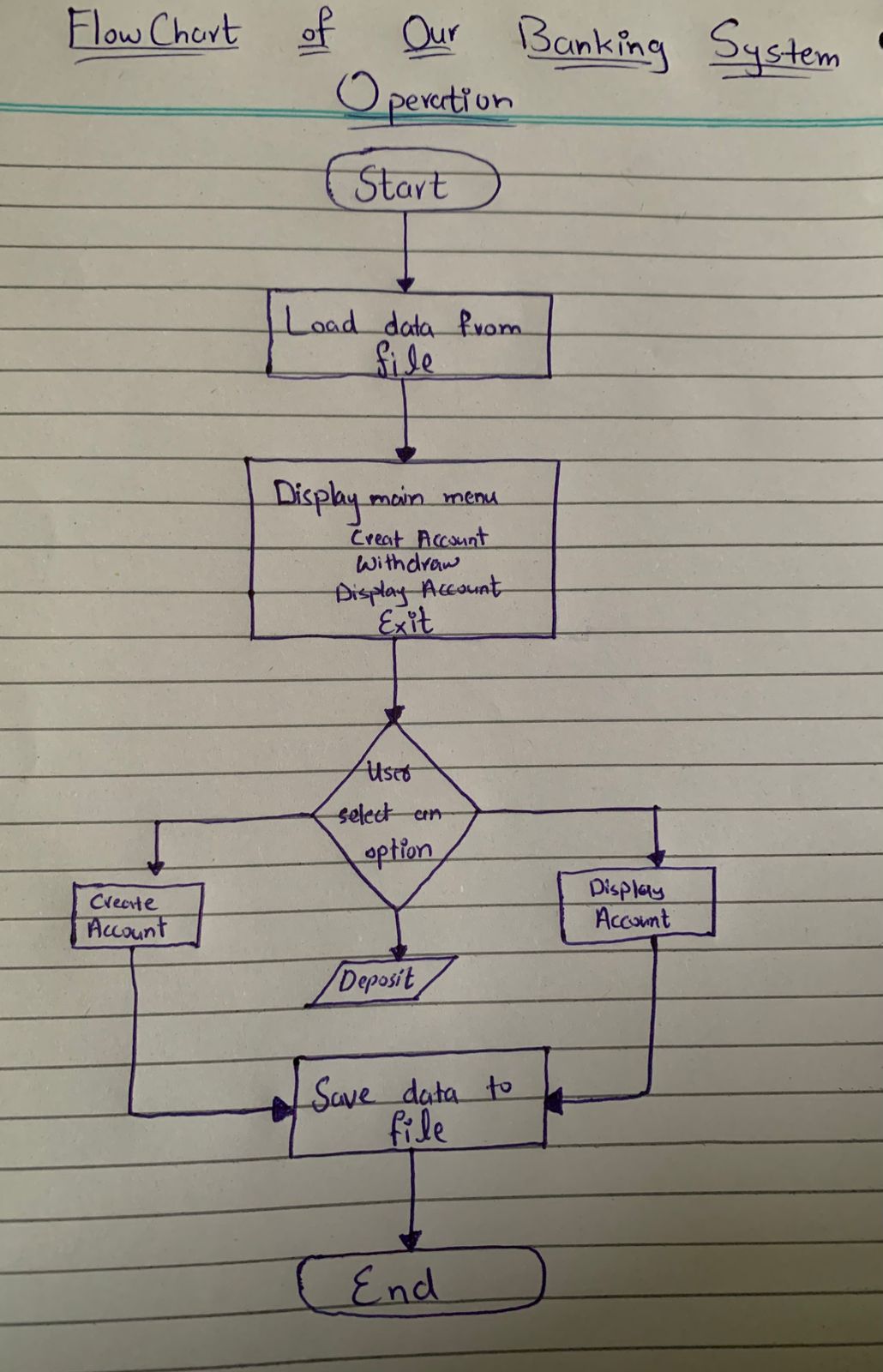
## Functional requirements:

* Create a bank account
* Deposit money into an account
* Withdraw money from an account
* Display account details
* Save and retrieve data using text file

## **Data Structure Used:**

* Arrays (Name[ ], AccountNumber[ ], Balance[ ]) to store account information.
* File streams (ofstream, ifstream) to handle persistent data.
* System (“cls & pause”)

## **Flow Chart:**



# **Code Snippets with Explanations**

## Snippet 1:

if (total >= number\_of\_acc) {

cout << "Maximum Accounts are made\n";

return;

}

***This snippet ensures that no more accounts than the predefined maximum (number\_of\_acc) can be created. It checks if the total number of accounts already created has reached the limit, and if so, it prevents any new account from being added to avoid exceeding the array capacity and causing******errors***

## Snippet 2:

for (char ch : Name[total]) {

if (isdigit(ch)) {

cout << "Invalid name. Name cannot contain numbers." << endl;

return;

}

}

***When creating a new account, this part validates the user's name by iterating over each character in the name string. It uses the isdigit() function to check if any character is a digit. If a digit is found, the program rejects the input by displaying an error message and stopping the account creation process, ensuring the name contains only alphabetic characters.***

## Snippet 3:

int findAccount(int accNum) {

for (int i = 0; i < total; i++) {

if (ACCnumber[i] == accNum)

return i;

}

return -1;

}

***This function searches through the existing accounts to locate an account with a matching account number. It loops over the stored account numbers and returns the index of the matching account. If no match is found, it returns -1, which is then used by other functions to handle errors like “Account not found.”***

## Snippet 4:

Amount[index] += deposit\_money;

cout << "New amount is now at: " << Amount[index] << endl;

***After confirming the account exists, this snippet adds the deposit amount to the existing balance stored in the Amount array at the account's index. It then outputs the updated balance to the user, reflecting the successful deposit.***

## Snippet 5:

if (withdraw\_money > Amount[index]) {

cout << "Not enough balance." << endl;

return;

}

Amount[index] -= withdraw\_money;

cout << "Withdraw done. New balance: " << Amount[index] << endl;

***Before allowing a withdrawal, this code checks if the requested withdrawal amount is greater than the current balance. If the balance is insufficient, it rejects the withdrawal and informs the user. If there is enough balance, it deducts the withdrawal amount from the stored balance and shows the new balance after the transaction.***

# **Implementation Summary:**

The program uses arrays to hold data for up to 10 accounts. It loads existing data from a file when the program starts and allows users to perform actions using a simple menu. All data is saved to the file after any modification. This makes the system efficient for learning purposes and demonstrates the use of file handling in C++.

# **Evaluation:**

Testing was done by running the program and trying various operations:

* Account creation (tested till 10 accounts)
* Deposit and withdrawal (both valid and invalid inputs)
* File data persistence (restarting the program retains data)

The program works as expected. One limitation is the maximum account number being fixed to 10 and lack of authentication for secure transactions.

# **Conclusion:**

This project provided hands-on experience with the fundamentals of C++ programming, including arrays, functions, and file handling. It simulates basic banking operations and helps develop logical thinking for menu-driven programs. Future improvements could include dynamic data structures, user authentication, and graphical interface.

# **Screen Recording:**

# 

# **Appendix:**

#include<iostream>

#include<fstream>

#include<conio.h>

#include<cctype> // for isdigit

#include<stdlib.h> // for system()

using namespace std;

const int number\_of\_acc = 10; // This is for 10 accounts maximum can be made

string Name[number\_of\_acc]; // Account user name

int ACCnumber[number\_of\_acc]; // Account user accout number

float Amount[number\_of\_acc]; // Amount of balance in Accoutint total = 0;

// Save all accounts to file

void saveToFile() {

ofstream file("bankdata.txt");

for (int i = 0; i < total; i++) {

file << Name[i] << " " << ACCnumber[i] << " " << Amount[i] << endl;

}

file.close();

}

// Load accounts from file at start

void loadFromFile() {

ifstream file("bankdata.txt");

while (file >> Name[total] >> ACCnumber[total] >> Amount[total]) {

total++;

}

file.close();

}

// Create Account

void CreateAccount() {

if (total >= number\_of\_acc) {

cout << "Maximum Accounts are made\n";

return;

}

cout << "Enter Name: ";

cin >> Name[total];

for (char ch : Name[total]) {

if (isdigit(ch)) {

cout << "Invalid name. Name cannot contain numbers." << endl;

return;

}

}

cout << "Enter Account Number: ";

cin >> ACCnumber[total];

if (!cin) {

cout << "Invalid input. Please Enter Digits only.\n";

cin.clear();

cin.ignore(1000, '\n');

return;

}

for (int i = 0; i < total; i++) {

if (ACCnumber[i] == ACCnumber[total]) {

cout << "Account number already exists.\n";

return;

}

}

cout << "Enter starting balance: ";

cin >> Amount[total];

if (!cin) {

cout << "Invalid balance input.\n";

cin.clear();

cin.ignore(1000, '\n');

return;

}

total++;

saveToFile(); // Save after creating account

cout << "Account is created successfully" << endl;

}

int findAccount(int accNum) {

for (int i = 0; i < total; i++) {

if (ACCnumber[i] == accNum)

return i;

}

return -1;

}

// Deposit Money

void Deposit() {

float deposit\_money = 0;

int ACCnum;

cout << "Enter the Account Number: ";

cin >> ACCnum;

int index = findAccount(ACCnum);

if (index == -1) {

cout << "Account not found." << endl;

return;

}

cout << "Enter the Amount of Money you wanna Deposit: ";

cin >> deposit\_money;

if (deposit\_money < 0) {

cout << "Invalid amount.\n";

return;

}

Amount[index] += deposit\_money;

saveToFile(); // Save after deposit

cout << "New amount is now at: " << Amount[index] << endl;

}

// Withdraw Money

void withdraw() {

int ACCnum;

float withdraw\_money;

cout << "Enter Account number: ";

cin >> ACCnum;

int index = findAccount(ACCnum);

if (index == -1) {

cout << "Account not found." << endl;

return;

}

cout << "Enter amount to withdraw: ";

cin >> withdraw\_money;

if (withdraw\_money > Amount[index]) {

cout << "Not enough balance." << endl;

return;

}

Amount[index] -= withdraw\_money;

saveToFile(); // Save after withdraw

cout << "Withdraw done. New balance: " << Amount[index] << endl;

}

// Display Account Info

void display() {

int ACCnum;

cout << "Enter Accout Number: ";

cin >> ACCnum;

int index = findAccount(ACCnum);

if (index == -1) {

cout << "Account not found." << endl;

return;

}

cout << "Name: " << Name[index] << endl;

cout << "Account Number: " << ACCnumber[index] << endl;

cout << "Balance: " << Amount[index] << endl;

}

int main() {

loadFromFile(); // Load data at start

int choice;

do {

cout << "\n--- BANK MENU ---\n";

cout << "1. Create Account\n";

cout << "2. Deposit\n";

cout << "3. Withdraw\n";

cout << "4. Display Account\n";

cout << "5. Exit\n";

cout << "\nEnter your choice: ";

cin >> choice;

cout << endl;

if (choice == 1) CreateAccount();

else if (choice == 2) Deposit();

else if (choice == 3) withdraw();

else if (choice == 4) display();

else if (choice == 5) cout << "Goodbye!\n";

else cout << "Invalid choice.\n";

if (choice != 5) {

system("pause");

system("cls");

}

} while (choice != 5);

\_getch();

return 0;

}