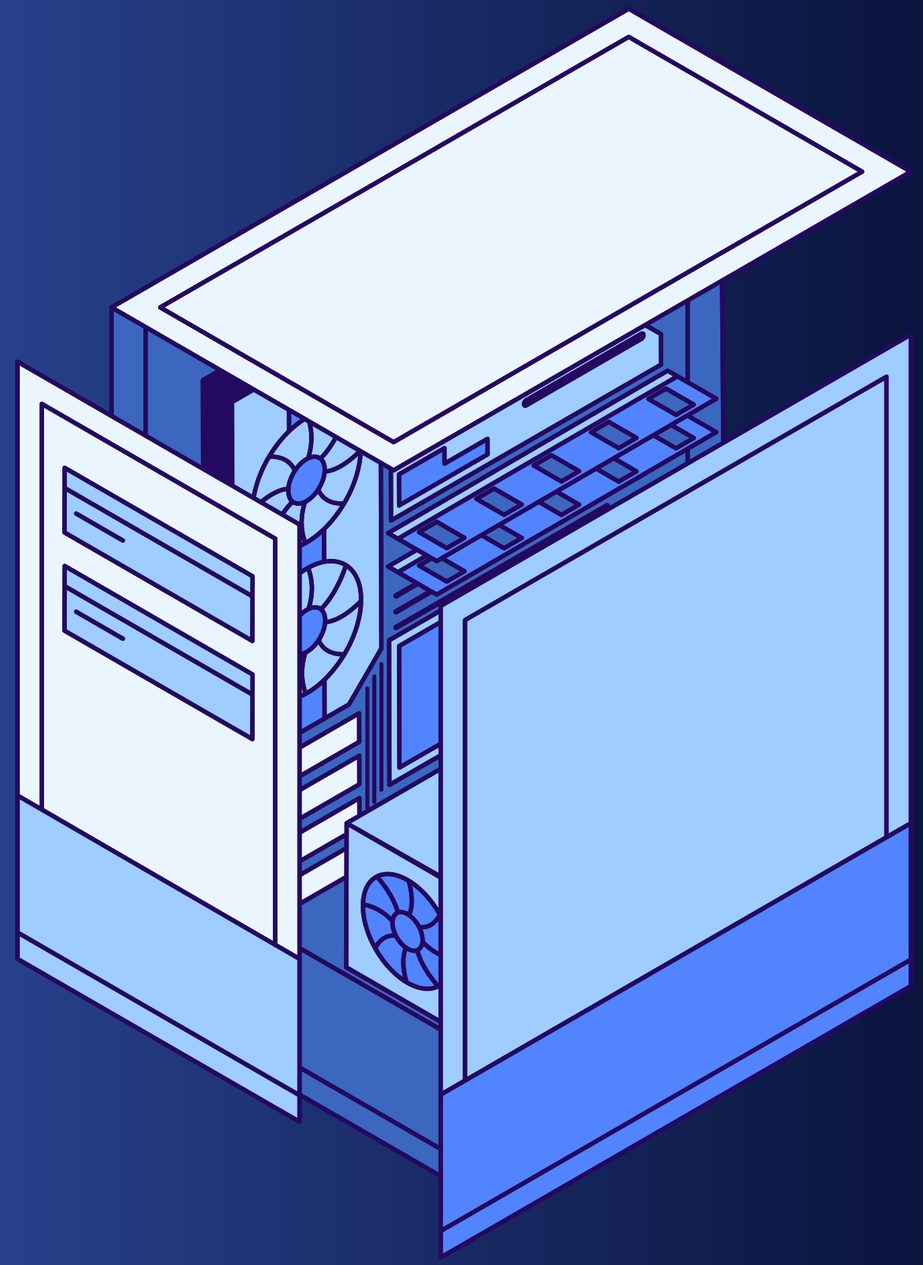
Instructor : DR . Prashant Trivedi

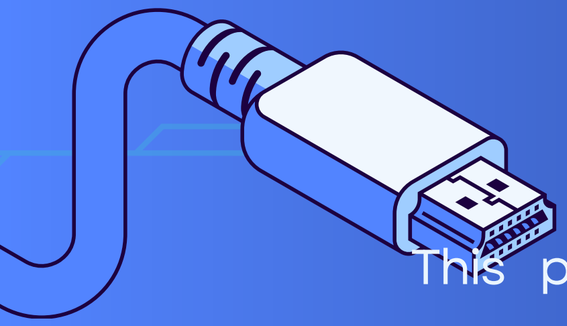
C PROGRAMMING PROJECT

NAME: RUPESH KUMAR

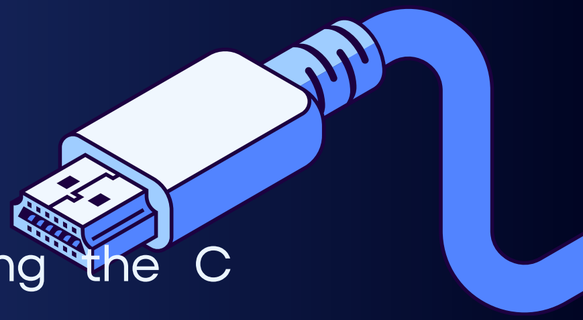
SAP ID: 590028036

BATCH: 59

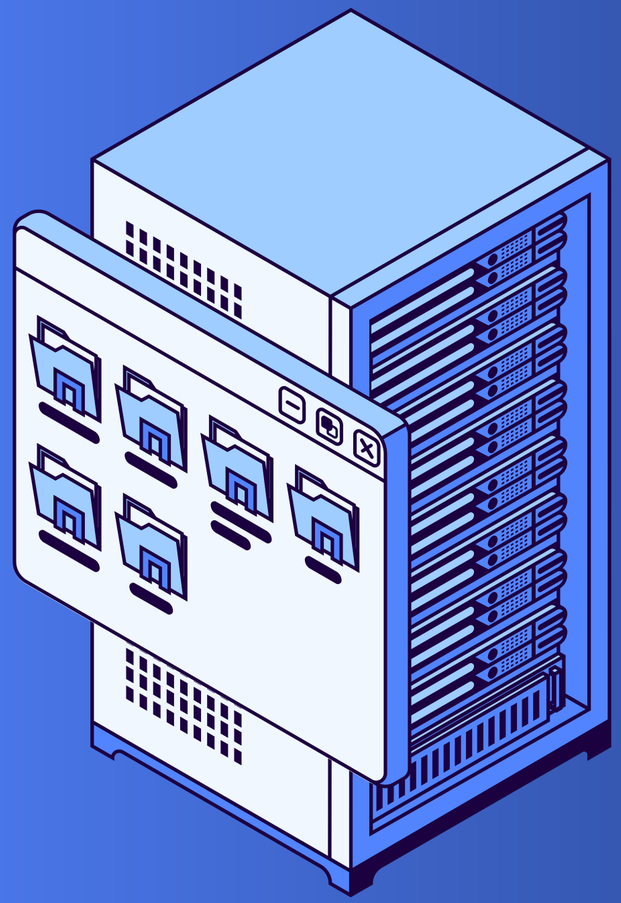




ABSTRACT



This project presents a simple ATM Simulator developed using the C programming language. The system performs essential banking operations such as balance inquiry, deposit, withdrawal, and secure PIN authentication. The program uses modular design through functions, conditional statements, loops, and input validation. The objective of this project is to understand basic C concepts and develop a console-based interactive banking system. The simulation helps users experience basic ATM functionalities in a simplified environment.



PROBLEM DEFINITION

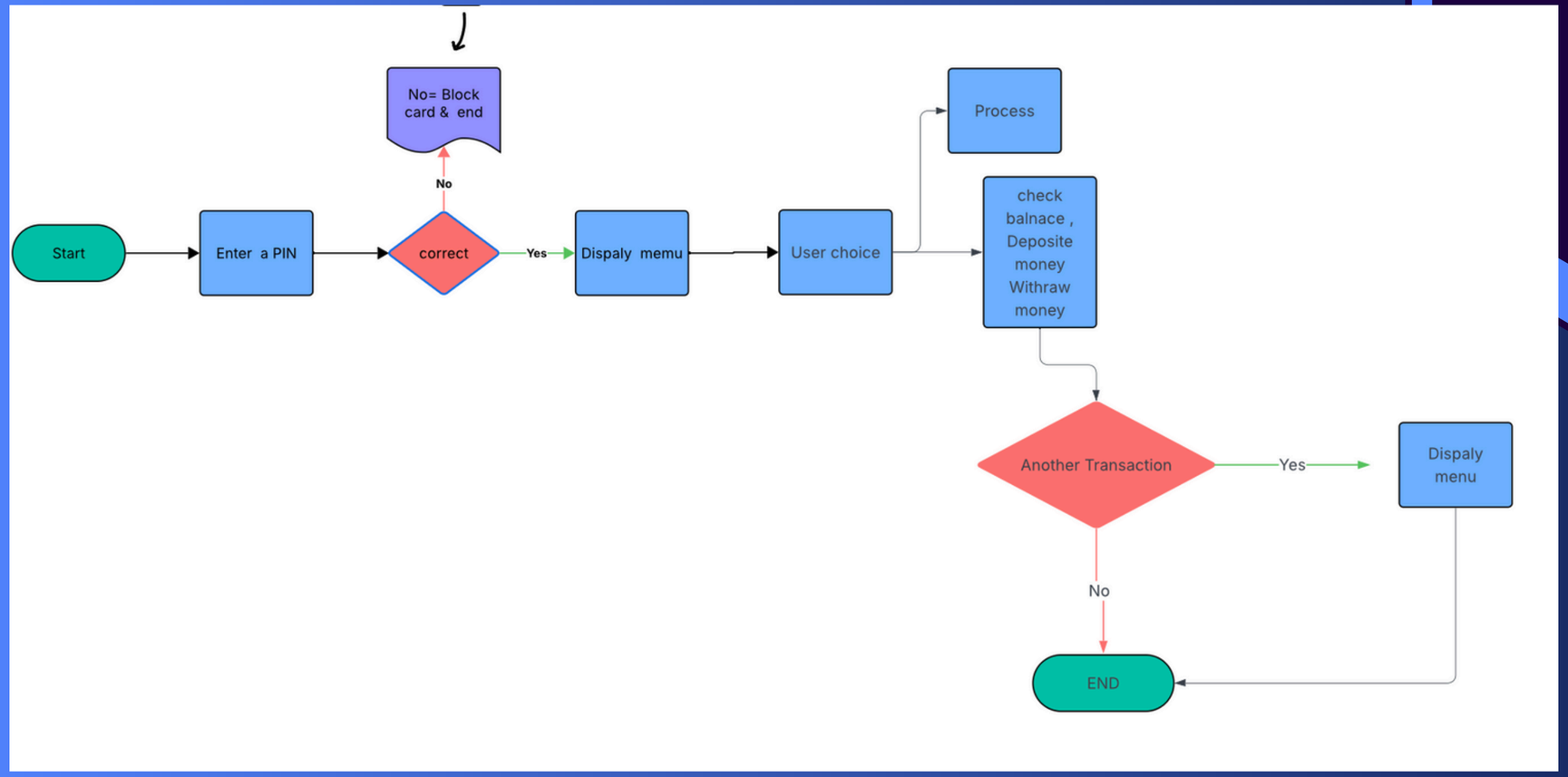
The goal of the ATM Simulator is to replicate basic banking operations in a computerized environment.

The system must:

- Allow secure login using a 4-digit PIN.
- Display a menu with ATM operations.
- Enable checking account balance.
- Allow depositing money.
- Allow withdrawing money with sufficient balance.
- Prevent invalid entries.
- Limit PIN attempts to three before blocking access.
- Provide a user-friendly and continuous transaction experience.

The system should be simple, accurate, and reliable for educational purposes.

SYSTEM DESIGN

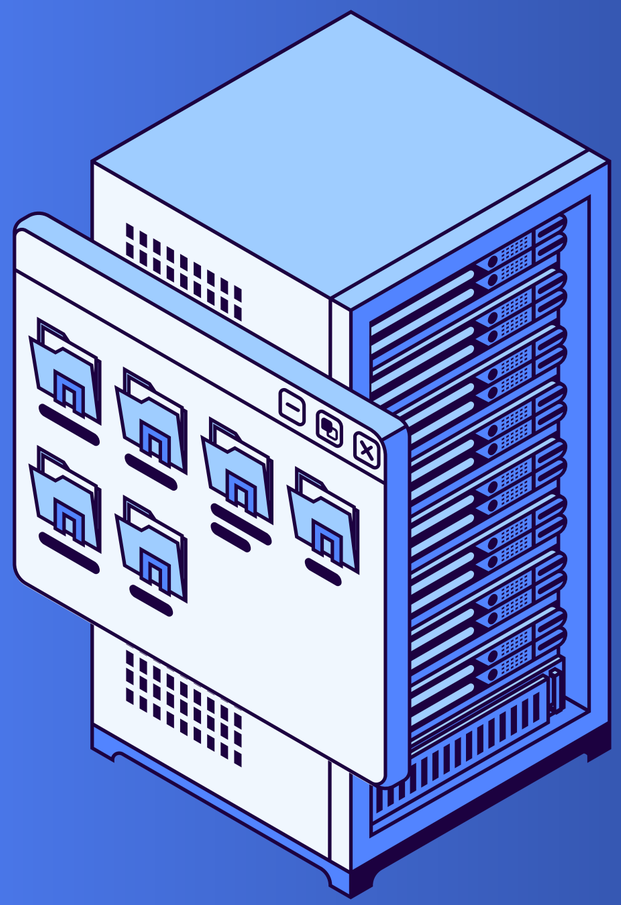


ALGORITHM

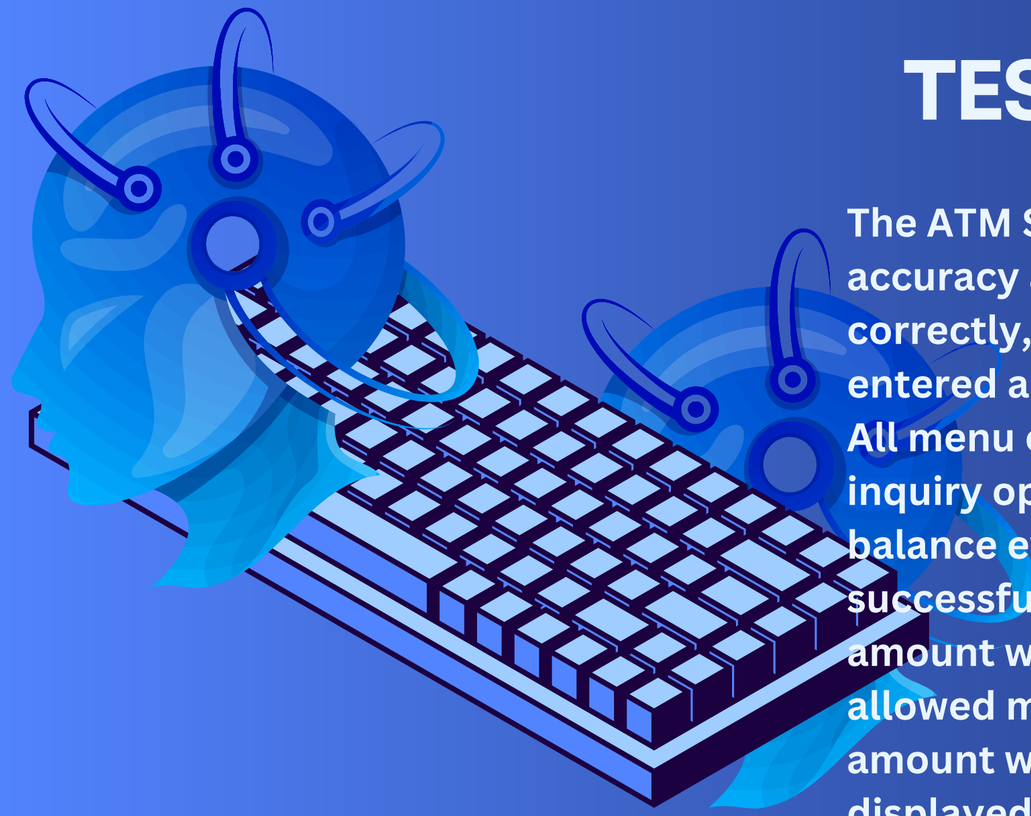
Algorithm for PIN Verification

1. Initialize correct PIN as 1234.
2. Ask user to enter PIN.
3. Compare with stored PIN.
4. If correct, allow access.
5. If incorrect, increment attempts.
6. If attempts reach 3, block access.
7. Algorithm for Menu
8. Display menu options (Check, Deposit, Withdraw, Exit).
9. Ask user to choose an option.
10. Call the corresponding function.
11. Ask if user wants another transaction.
12. Algorithm for Deposit
13. Ask for amount.
14. Validate amount > 0.
15. Add to balance.
16. Display updated balance.

IMPLEMENTATION DETAILS

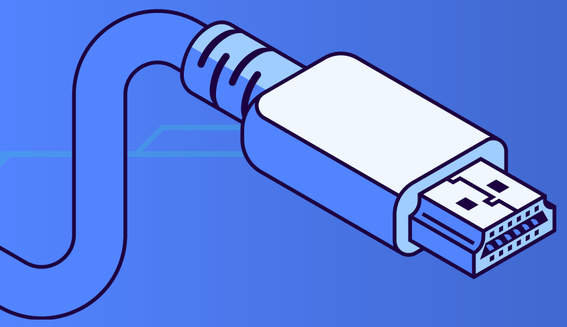


The ATM Simulator was implemented using the C programming language with a structured and modular approach. The program begins by defining several functions such as displayMenu(), checkBalance(), depositMoney(), and withdrawMoney(), which help keep the code organized and easy to understand. The main function handles overall program control, including variable initialization, PIN verification, and the transaction loop. The PIN verification system uses a loop that allows the user three attempts to enter the correct PIN; if the correct PIN is entered, access is granted, otherwise the card is blocked after three failed attempts. Once the user is authenticated, the system repeatedly displays a menu where the user can choose to check their balance, deposit money, withdraw money, or exit. Each menu option calls its respective function. The deposit function asks the user for an amount and adds it to the balance if the value is valid. The withdrawal function ensures that the user cannot withdraw more than the available balance and allows successful withdrawal for valid amounts. After each transaction, the user is prompted whether they want to continue using the ATM. This modular and function-based implementation ensures clarity, reusability, and smooth execution of all ATM operations.

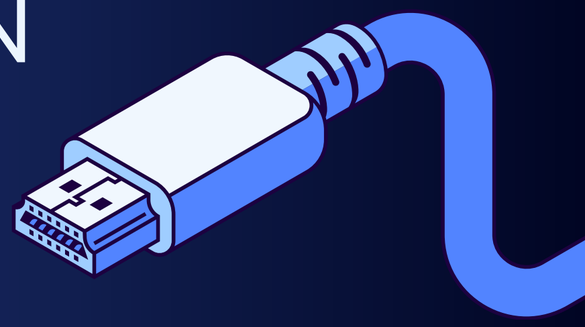


TESTING & RESULTS

The ATM Simulator program was tested thoroughly to ensure accuracy and reliability. The PIN verification system worked correctly, granting access when the correct PIN (1234) was entered and blocking the user after three incorrect attempts. All menu operations performed as expected. The balance inquiry option accurately displayed the current account balance every time it was selected. The deposit function successfully increased the balance when a valid positive amount was entered. Similarly, the withdrawal function allowed money to be withdrawn only when the requested amount was less than or equal to the available balance, and it displayed an “Insufficient funds” message for higher amounts. Invalid inputs, such as negative deposit or withdrawal values, were also handled properly. Overall, all major features of the ATM Simulator were tested using various scenarios, and the results confirmed that the system is functioning smoothly and reliably.



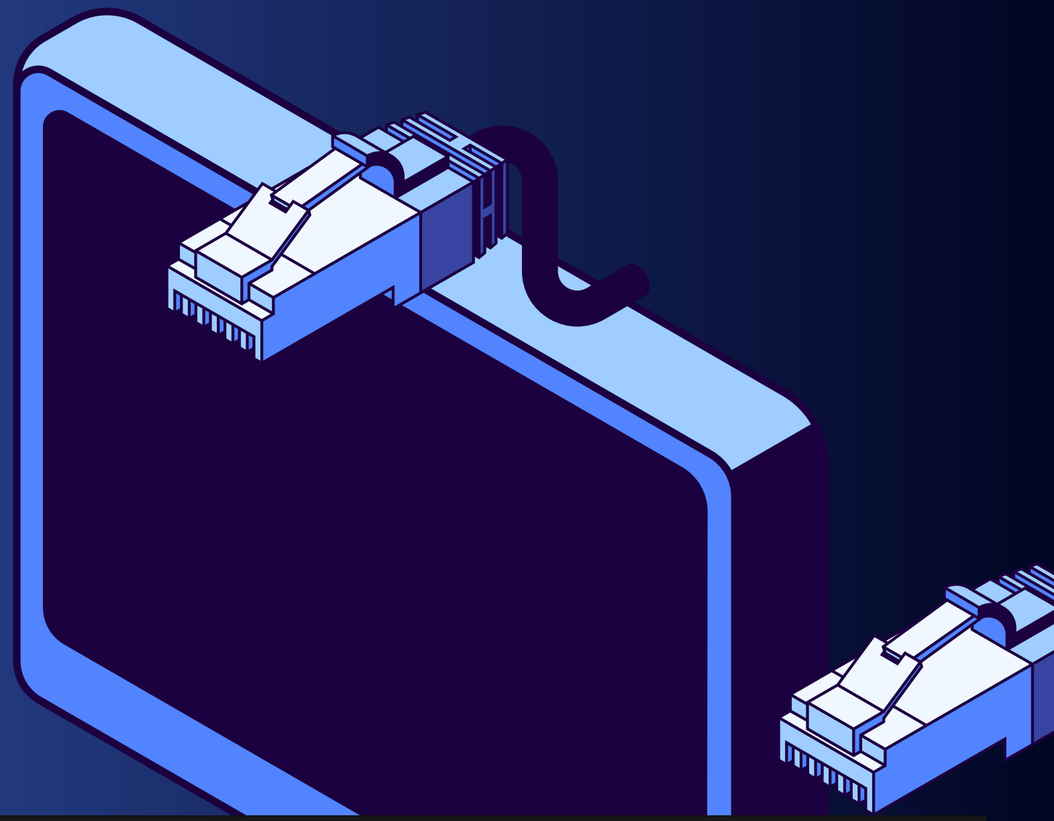
HEADER FILES AND FUNCTION



- #include <stdio.h>: Allows input/output functions like printf and scanf.
- #include <stdlib.h>: Used for general utilities, though not strictly needed here (could be useful for exit() if used).
- Function prototypes: Declare the functions that will be defined later. This allows main() to call these functions before they are defined in the file.

```c
#include <stdio.h>
#include <stdlib.h>

// FUNCTION PROTOTYPES
void displayMenu();
void checkBalance(float balance);
float depositMoney(float balance);
float withdrawMoney(float balance);
```

MAIN FUNCTION & VARIABLE DECLARATIONS



Explanation:

- choice: Stores the menu option selected by the user.
- balance: Keeps track of the user's account balance. Initialized to $5,000,000.
- continueTransaction: Controls the main transaction loop.
- pin & enteredPin: Used for PIN verification.
- pinAttempts: Counts failed PIN attempts.
- Prints a welcome message.

```c
int main() {
    // VARIABLE DECLARATIONS
    int choice;
    float balance = 5000000.00;
    int continueTransaction = 1;
    int pin = 1234;
    int enteredPin;
    int pinAttempts = 0;

    printf(" WELCOME TO SIMPLE ATM SIMULATOR \n\n");
```

. PIN VERIFICATION SYSTEM

- Allows the user 3 attempts to enter the correct PIN.
- If correct, prints success and continues.
- If incorrect after 3 attempts, the program ends, simulating a card block.

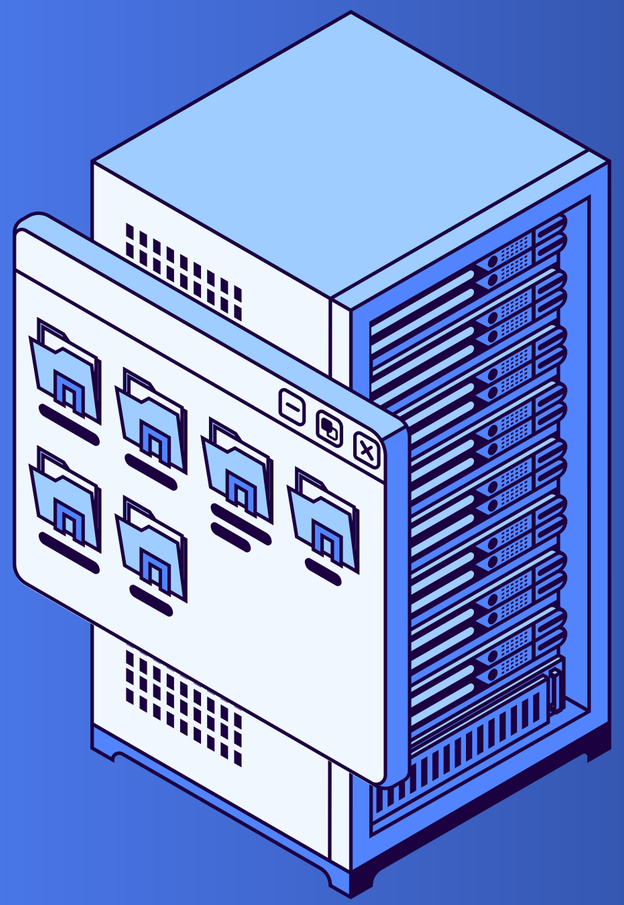
```
// PIN VERIFICATION SYSTEM
while (pinAttempts < 3) {
    printf("Please enter your 4-digit PIN: ");
    scanf("%d", &enteredPin);

    if (enteredPin == pin) {
        printf("\nPIN verified successfully!\n");
        break;
    } else {
        pinAttempts++;
        printf("Invalid PIN! Attempts remaining: %d\n\n",
    }
}
```

MAIN TRANSACTION LOOP

- LOOPS UNTIL THE USER DECIDES TO EXIT.
- CALLS DISPLAYMENU() TO SHOW OPTIONS.
- USES A SWITCH STATEMENT TO HANDLE EACH MENU OPTION:
  - 1: CHECK BALANCE
  - 2: DEPOSIT MONEY
  - 3: WITHDRAW MONEY
  - 4: EXIT
- AFTER EACH TRANSACTION, ASKS IF THE USER WANTS TO PERFORM ANOTHER.

```c
// MAIN TRANSACTION LOOP
while (continueTransaction) {
    displayMenu();
    printf("\nPlease select an option (1-4): ");
    scanf("%d", &choice);

    // MENU OPTION HANDLING
    switch (choice) {
        case 1:
            checkBalance(balance);
            break;

        case 2:
            balance = depositMoney(balance);
            break;

        case 3:
            balance = withdrawMoney(balance);
            break;

        case 4:
            printf("\nThank you for using our ATM service!\n");
            printf("Please take your card.\n");
            continueTransaction = 0;
            break;

        default:
            printf("\nInvalid option! Please try again.\n");
            break;
```

```c
main() {
while (continueTransaction) {
    switch (choice) {

    // CONTINUE TRANSACTION PROMPT
    if (continueTransaction) {
        printf("\nDo you want to perform another transaction?\n");
        printf("1. Yes\n");
        printf("2. No\n");
        printf("Enter your choice: ");
        scanf("%d", &continueTransaction);
        continueTransaction = (continueTransaction == 1) ? 1 : 0;
    }
}

return 0;
```



DISPLAY MENU

Explanation:

Simply prints the menu options.

Keeps main() cleaner and easier to read.

```c
// FUNCTION TO DISPLAY MAIN MENU
void displayMenu() {
    printf("\n ATM MAIN MENU \n");
    printf("1. Check Balance\n");
    printf("2. Deposit Money\n");
    printf("3. Withdraw Money\n");
    printf("4. Exit\n");
}

// FUNCTION TO CHECK ACCOUNT BALANCE
void checkBalance(float balance) {
    printf("\n BALANCE INQUIRY \n");
    printf("Your current balance is: $%.2f\n", balance);
}
```

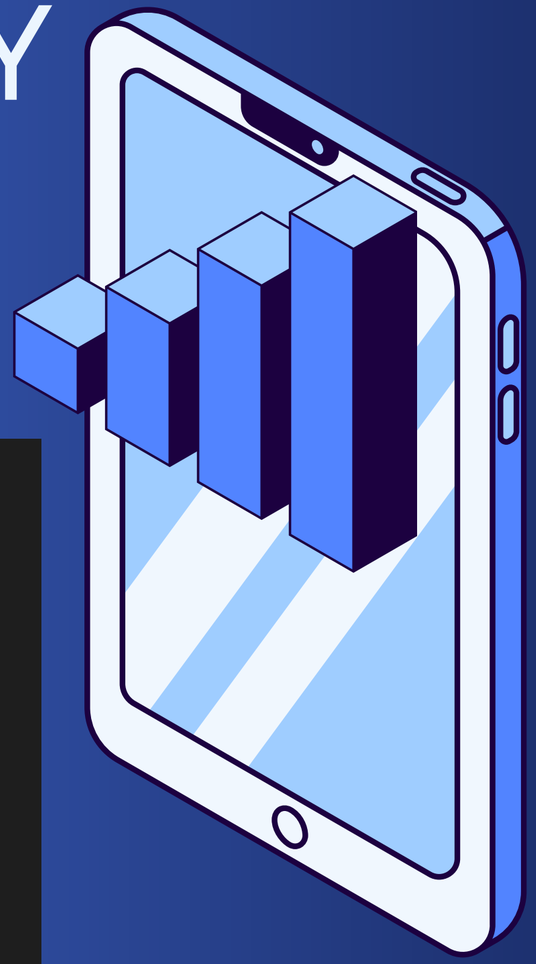
CHECK BALANCE FUNCTION

- Displays the current balance passed from main().
- %.2f ensures the balance is displayed with 2 decimal places.

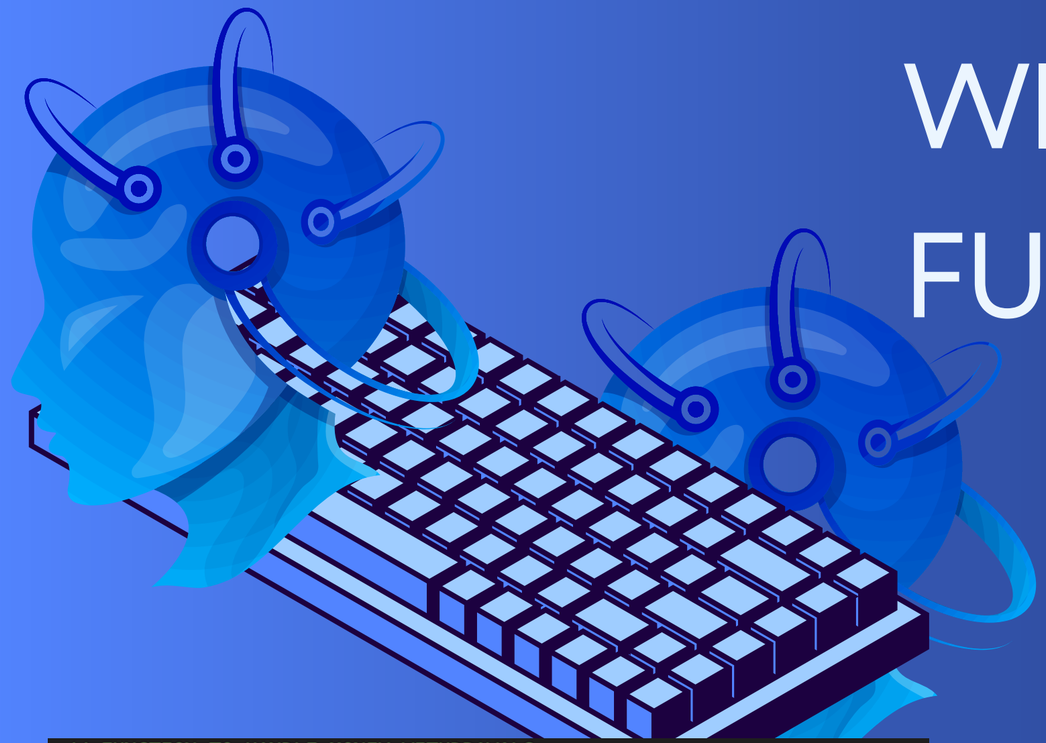
```c
// FUNCTION TO CHECK ACCOUNT BALANCE
void checkBalance(float balance) {
    printf("\n BALANCE INQUIRY \n");
    printf("Your current balance is: $%.2f\n", balance);
}

// FUNCTION TO HANDLE MONEY DEPOSITS
float depositMoney(float balance) {
    float amount;

    printf("\n DEPOSIT MONEY \n");
    printf("Enter amount to deposit: $");
    scanf("%f", &amount);

    if (amount > 0) {
        balance += amount;
        printf("Deposit successful!\n");
        printf("New balance: $%.2f\n", balance);
    } else {
        printf("Invalid amount! Please enter a positive value.\n");
    }
    return balance;
}
```

DEPOSIT MONEY FUNCTION

```c
// FUNCTION TO HANDLE MONEY DEPOSITS
float depositMoney(float balance) {
    float amount;

    printf("\n DEPOSIT MONEY \n");
    printf("Enter amount to deposit: $");
    scanf("%f", &amount);

    if (amount > 0) {
        balance += amount;
        printf("Deposit successful!\n");
        printf("New balance: $%.2f\n", balance);
    } else {
        printf("Invalid amount! Please enter a positive value.\n");
    }
    return balance;
}
```



- Prompts the user for the deposit amount.
- Adds the amount to balance if positive.
- Returns the updated balance to main().



WITHDRAW MONEY FUNCTION

- Prompts for withdrawal amount.
- Checks if the amount is valid and if there are sufficient funds.
- Deducts the amount from balance and prints the updated balance.
- Returns updated balance to main().

```c
// FUNCTION TO HANDLE MONEY WITHDRAWALS
float withdrawMoney(float balance) {
    float amount;

    printf("\n WITHDRAW MONEY \n");
    printf("Enter amount to withdraw: $");
    scanf("%f", &amount);

    if (amount <= 0) {
        printf("Invalid amount! Please enter a positive value.\n");
    } else if (amount > balance) {
        printf("Insufficient funds! Your balance is $%.2f\n", balance);
    } else {
        balance -= amount;
        printf("Withdrawal successful!\n");
        printf("Please take your cash: $%.2f\n", amount);
        printf("Remaining balance: $%.2f\n", balance);
    }
    return balance;
}
```

Loading...

Conclusion

The ATM Simulator project successfully demonstrates basic banking operations using C programming. It reinforces core programming concepts such as loops, conditional statements, modularization, and user input handling. The simulation offers a secure and interactive experience similar to a real ATM.

THANK **YOU.**