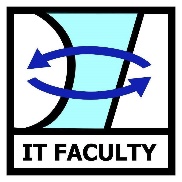
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN**

**ỨNG DỤNG GIAO DỊCH ATM**

Người hướng dẫn**: ThS. NGUYỄN THỊ LỆ QUYÊN**

Sinh viên thực hiện**:**

**Hồ Sỹ Thảo LỚP: 22T\_NHAT1 NHÓM: 09**

**Phạm Phan Thành LỚP: 22T\_NHAT1 NHÓM: 09**

**Đà Nẵng, 06/2023**

MỤC LỤC

[MỤC LỤC I](#_Toc137988382)

[DANH MỤC HÌNH VẼ II](#_Toc137988383)

[MỞ ĐẦU III](#_Toc137988384)

[1. TỔNG QUAN ĐỀ TÀI 1](#_Toc137988385)

[2. CƠ SỞ LÝ THUYẾT 2](#_Toc137988386)

[2.1. Ý tưởng 2](#_Toc137988387)

[2.2. Cơ sở lý thuyết 2](#_Toc137988388)

[3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 4](#_Toc137988389)

[3.1. Phát biểu bài toán 4](#_Toc137988390)

[3.2. Cấu trúc dữ liệu 4](#_Toc137988391)

[3.3. Thuật toán 6](#_Toc137988392)

[4. CHƯƠNG TRÌNH VÀ KẾT QUẢ 9](#_Toc137988393)

[4.1. Tổ chức chương trình 9](#_Toc137988394)

[4.2. Ngôn ngữ cài đặt 11](#_Toc137988395)

[4.3. Kết quả 11](#_Toc137988396)

[4.3.1. Giao diện chính của chương trình 11](#_Toc137988397)

[4.3.2. Kết quả thực thi của chương trình 12](#_Toc137988398)

[4.3.3. Nhận xét đánh giá 15](#_Toc137988399)

[5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 15](#_Toc137988400)

[5.1. Kết luận 15](#_Toc137988401)

[5.2. Hướng phát triển 15](#_Toc137988402)

[TÀI LIỆU THAM KHẢO 17](#_Toc137988403)

[PHỤ LỤC : 17](#_Toc137988404)

DANH MỤC HÌNH VẼ

[*Hình 1: MAATM.txt* 9](#_Toc137988405)

[*Hình 2: ADMIN.txt* 9](#_Toc137988406)

[*Hình 3: Giao diện chính* 11](#_Toc137988407)

[*Hình 4: Test Nhập* 11](#_Toc137988408)

[*Hình 5: MENU* 12](#_Toc137988409)

[*Hình 6: Xem tài khoản* 12](#_Toc137988410)

[*Hình 7: Nộp tiền hoặc rút tiền thành công* 13](#_Toc137988411)

[*Hình 8: Lỗi không tìm thấy ACC* 13](#_Toc137988412)

[*Hình 9: Lỗi mã PIN* 14](#_Toc137988413)

[*Hình 10: Lỗi không đủ tiền* 14](#_Toc137988414)

[*Hình 11: Giới hạn giao dịch* 15](#_Toc137988415)

MỞ ĐẦU

MỤC ĐÍCH:

* Đề tài này nhằm mục đích tạo ra một phần mềm ứng dụng thực hiện các giao dịch tại máy ATM.

MỤC TIÊU:

* Nắm được các phương pháp phân tích dữ liệu và áp dụng chúng vào việc xử lý dữ liệu giao dịch tại máy ATM.
* Hiểu được cách sử dụng các thuật toán để áp dụng xử lý các giao dịch.
* Xây dựng được một phần mềm ứng dụng đáp ứng yêu cầu về độ chính xác và hiệu quả trong các giao dịch tại máy ATM.
* Cải tiến thêm vài chức năng để hoàn thiện phần mềm.

PHẠM VI ĐỀ TÀI:

* Phạm vi của đề tài này là xây dựng một ứng dụng thực hiện các giao dịch trên máy ATM bao gồm gửi tiền, rút tiền và xem tài khoản. Ứng dụng cũng giới hạn số lần giao dịch của mỗi khách hàng trên một ngày và sẽ từ chối giao dịch nếu khách hàng giao dịch quá số lần quy định. Kết thúc một ngày, danh sách giao dịch được ghi vào một File.

ĐỐI TƯỢNG NGHIÊN CỨU:

* Đối tượng nghiên cứu là các khách hàng sử dụng máy ATM để thực hiện các giao dịch tài chính. Cụ thể hơn là, đối tượng nghiên cứu bao gồm các khách hàng có tài khoản ngân hàng được đăng kí với máy ATM và nhu cầu sử dụng các chức năng của máy ATM.

PHƯƠNG PHÁP NGHIÊN CỨU:

* Phân chia công việc, quản lý đồ án một cách hiệu quả để đạt đến kết quả tốt đồng thời trong quá trình thực hiện phải kiểm thử, tìm ra những biến số ảnh hưởng đến đồ án sau này. Xác định yêu cầu của bài toán để tìm ra phương pháp tối ưu nhất để giải quyết yêu cầu đề ra. Ngoài ra cần sử dụng các công cụ và ngôn ngữ lập trình C và các thư viện hỗ trợ đi kèm trong C như stdio.h, stdlib.h, float.h, v.v để phát triển chương trình.

CẤU TRÚC ĐỒ ÁN “ỨNG DỤNG GIAO DỊCH ATM”:

1. Giới thiệu về đề tài: phần này giới thiệu về đề tài nghiên cứu, mục đích và ý nghĩa của đề tài.
2. Tổng quan về ATM: Phần này trình bày một số thông tin cơ bản về máy ATM, quy trình và các tính năng cơ bản của máy ATM.
3. Cơ sở lý thuyết: Phần này mô tả chi tiết các chức năng cần có trong ứng dụng ATM, bao gồm gửi tiền, rút tiền và xem tài khoản và các ràng buộc về số lần giao dịch.
4. Tổ chức cấu trúc dữ liệu và thuật toán: Phần này trình bày về quá trình thiết kế và triển khai ứng dụng ATM, bao gồm cách lưu trữ dữ liệu, cách xác thực người dùng và các chức năng chính của ứng dụng.
5. Chương trình và kết quả: Phần này mô tả các phương pháp kiểm thử và đánh giá hiểu quả của ứng dụng ATM.
6. Kết luận và hướng phát triển: Phần này tổng kết các kết quả đạt được, những giới hạn và hướng phát triển của đề tài trong tương lai.

Ngoài ra còn có phần tài liệu tham khảo, ghi chú, phụ lục và mã nguồn.

# TỔNG QUAN ĐỀ TÀI

Xây dựng ứng dụng thực hiện giao dịch trên máy ATM, sử dụng mảng song song để lưu dữ liệu, đọc và ghi dữ liệu vào file, viết các hàm để xây dựng các chức năng, và giải quyết nhiều vấn đề với các biến, vòng lặp và ra quyết định. Thực hiện giao dịch trên máy ATM, bao gồm các chức năng Gửi tiền, Rút tiền, xem tài khoản và giới hạn số lần giao dịch của mỗi khách hàng trên một ngày.

Để xây dựng ứng dụng thực hiện giao dịch trên máy ATM như đã mô tả, ta cần thực hiện các bước sau:

1. Thiết kế cấu trúc dữ liệu: Sử dụng mảng song song để lưu trữ thông tin liên quan đến tài khoản của khách hàng, bao gồm số tài khoản, số dư, số lần giao dịch trong ngày, và các thông tin khác cần thiết.
2. Đọc và ghi dữ liệu vào file: Viết các hàm đọc và ghi dữ liệu từ/đến file để lưu trữ thông tin tài khoản một cách an toàn và có thể truy xuất khi cần thiết.
3. Xây dựng các hàm chức năng:
   * + 1. Gửi tiền: Viết hàm thực hiện việc gửi tiền vào tài khoản, cập nhật số dư và số lần giao dịch trong ngày cho tài khoản tương ứng.
       2. Rút tiền: Viết hàm thực hiện việc rút tiền từ tài khoản, kiểm tra số dư và giới hạn số lần giao dịch trong ngày trước khi cho phép rút tiền, sau đó cập nhật số dư và số lần giao dịch.
       3. Xem tài khoản: Viết hàm truy xuất thông tin tài khoản, hiển thị số dư và các thông tin khác mà khách hàng cần biết.
       4. Kiểm tra giới hạn giao dịch: Viết hàm kiểm tra số lần giao dịch của một tài khoản trong ngày, đảm bảo không vượt quá giới hạn cho phép.
4. Xử lý các vấn đề liên quan đến biến, vòng lặp và ra quyết định: Sử dụng các biến để lưu trữ thông tin tạm thời trong quá trình xử lý, vòng lặp để duyệt qua các phần tử trong mảng và ra quyết định dựa trên các điều kiện cần thiết.
5. Tích hợp các chức năng vào giao diện người dùng: Tạo giao diện người dùng đơn giản và thện, cho phép khách hàng thực hiện giao dịch một cách dễ dàng và nhanh chóng.

Việc xây dựng ứng dụng ATM với các chức năng Gửi tiền, Rút tiền, Xem tài khoản và giới hạn số lần giao dịch trong ngày sẽ giúp đảm bảo an toàn và tiện lợi cho khách hàng khi thực hiện giao dịch. Việc sử dụng mảng song song, đọc ghi dữ liệu vào file, và xử lý các vấn đề với biến, vòng lặp và ra quyết định giúp tối ưu hóa quá trình xử lý dữ liệu và cung cấp một ứng dụng hiệu quả cho người dùng.

# CƠ SỞ LÝ THUYẾT

## Ý tưởng

Để thực hiện đề tài, ta cần đọc danh sách các số tài khoản , PIN và số tiền hiện có trong tài khoản từ file vào trong máy ATM khi khởi động. Sau đó, ta cần phát triển các chức năng gửi tiền, rút tiền và xem tài khoản cho khách hàng. Mỗi giao dịch của khách hàng đều phải được xác nhận bằng số tài khoản và PIN. Nếu giao dịch thành công, máy ATM cần lưu lại danh sách các giao dịch bao gồm số tài khoản và số tiền và cập nhật số tiền hiện có trong tài khoản. Tuy nhiên, máy ATM giới hạn số lần giao dịch của mỗi khách hàng trên một ngày và sẽ từ chối giao dịch nếu khách hàng vượt quá số lần qui định. Khi kết thúc một ngày, máy ATM sẽ tắt / dừng và ghi danh sách giao dịch vào file.

## Cơ sở lý thuyết

Sử dụng các kiến thức về lập trình C, các kỹ thuật xử lý file, cơ chế bảo mật, và giải quyết các vấn đề với biến, vòng lặp, các thuật toán quan trọng và ra quyết định. Sử dụng những công thức tính toán trong các giao dịch.

Các định nghĩa:

**\*Ngôn ngữ lập trình C** là một ngôn ngữ lập trình mạnh mẽ và phổ biến được phát triển vào những năm 1970 bởi Dennis Ritchie tại Bell Labs. C được sử dụng rộng rãi trong việc phát triển phần mềm hệ thống, phần mềm nhúng và ứng dụng đa nền tảng.

**\*Kiểu dữ liệu cấu trúc (struct)** là một nhóm các thành phần dữ liệu, được gom lại với nhau và đặt trong một tên. Được sử dụng trong việc phát triển phần mềm hệ thống, phần mềm nhúng và ứng dụng đa nền tảng. Trong đó:

* + Mỗi thành phần dữ liệu gọi là trường dữ liệu hoặc thành viên.
  + Các trường dữ liệu có thể có kiểu dữ liệu khác nhau.
  + Lập trình viên tự định nghĩa kiểu dữ liệu cấu trúc gồm những thành phần dữ liệu.

**\*Cú pháp kiểu dữ liệu cấu trúc (struct) trong ngôn ngữ C**

struct TenCauTruc {

    // Khai báo thành phần dữ liệu

    kieu\_du\_lieu1 ten\_thanh\_phan\_1;

    kieu\_du\_lieu2 ten\_thanh\_phan\_2;

    // ...

    kieu\_du\_lieuN ten\_thanh\_phan\_N;

};

**\*Hàm** là một khối mã chứa một tập hợp các câu lệnh được đặt tên và có thể được gọi từ các phần khác trong chương trình. Hàm thường được sử dụng để thực hiện một tác vụ cụ thể, thực hiện một chuỗi các câu lệnh hoặc tính toán một kết quả và trả về giá trị cho người gọi.

**\*Cú pháp chung của một hàm trong ngôn ngữ C như sau:**

kieu\_tra\_ve ten\_ham(tham\_so1, tham\_so2, ..., tham\_soN) {

    // Các câu lệnh để thực hiện tác vụ của hàm

    // Trả về giá trị kết quả (nếu có)

    return ket\_qua;

}

**\*Quick Sort** là một thuật toán sắp xếp đệ quy dựa trên phân hoạch (partition) của mảng. Thuật toán này hoạt động bằng cách chọn một phần tử gọi là "phần tử chốt" (pivot) từ mảng và phân hoạch các phần tử còn lại thành hai phần, một phần nhỏ hơn pivot và một phần lớn hơn pivot. Sau đó, thuật toán được áp dụng đệ quy trên cả hai phần này để sắp xếp chúng.

\***Binary Search** là một thuật toán tìm kiếm trong một mảng đã được sắp xếp. Thuật toán này hoạt động bằng cách so sánh phần tử cần tìm với phần tử ở giữa mảng, sau đó thu hẹp phạm vi tìm kiếm dựa trên kết quả so sánh. Quá trình này được tiếp tục cho đến khi tìm thấy phần tử cần tìm hoặc không còn phần tử nào để xem xét.

\***Đinh nghĩa các thư viện trong C được sử dụng**:

* + **stdio.h**: Thư viện này chứa các hàm và định nghĩa liên quan đến nhập/xuất dữ liệu. Bao gồm các hàm như printf, scanf, fgets, fopen, fclose, và các hằng số như EOF (ký tự kết thúc tệp) và NULL (con trỏ rỗng).
  + **string.h**: Thư viện này chứa các hàm và định nghĩa liên quan đến chuỗi và xử lý chuỗi. Bao gồm các hàm như strcpy, strcat, strlen, strcmp, và các hằng số như NULL (con trỏ rỗng).
  + **math.h**: Thư viện này chứa các hàm toán học và hằng số. Bao gồm các hàm như sqrt, sin, cos, pow, và các hằng số như M\_PI (giá trị của pi).
  + **float.h**: Thư viện này chứa các định nghĩa và hằng số liên quan đến kiểu dữ liệu số thực. Bao gồm các hằng số như FLT\_MAX (giá trị lớn nhất mà một số thực có thể đại diện) và FLT\_EPSILON (độ sai số tối thiểu cho số thực).

# TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

## Phát biểu bài toán

Đầu vào (Input) của chương trình ATM bao gồm:

* Tệp chứa dữ liệu danh sách các tài khoản, mỗi tài khoản bao gồm số tài khoản, số PIN và số dư tài khoản.
* Tệp chứa mã bảo mật.
* Các thông tin nhập từ bàn phím khi khách hàng sử dụng máy ATM, bao gồm số tài khoản, số PIN, số tiền giao dịch.

Đầu ra (Output) của chương trình ATM bao gồm:

* Thông báo lỗi nếu có, ví dụ như nhập sai số tài khoản hoặc số PIN.
* Thông tin về số dư tài khoản sau khi giao dịch được thực hiện thành công.
* Tệp ghi lại các danh sách các giao dịch được thực hiện trong ngày, bao gồm số tài khoản, số tiền giao dịch.

## Cấu trúc dữ liệu

Chương trình sử dụng cấu trúc dữ liệu struct để lưu thông tin của mỗi tài khoản ngân hàng.

Các thuộc tính của struct THE\_ATM bao gồm:

* AccNumber: số tài khoản ngân hàng
* Pins: mật khẩu tài khoản
* Sodu: số dư trong tài khoản

Các giao dịch được xử lý thông qua các hàm GD1(), GD2(), và GD3().Mỗi giao dịch sẽ tạo ra một biên lai (struct BIEN\_LAI) để lưu thông tin về giao dịch.

Các thuộc tính của struct BIEN\_LAI bao gồm:

* MaThe: số tài khoản ngân hàng thực hiện giao dịch
* MaLoaiGD: mã loại giao dịch (1 - rút tiền, 2 - chuyển tiền, 3 - kiểm tra số dư)
* TenLoaiGD: tên loại giao dịch tương ứng với mã loại giao dịch
* SoTienGD: số tiền thực hiện giao dịch
* LePhi: lệ phí giao dịch
* SoDuConLai: số dư còn lại trong tài khoản sau khi thực hiện giao dịch.

Chương trình sử dụng các hàm để thực hiện các chức năng sau:

* findAcc: tìm kiếm vị trí của tài khoản dựa trên số tài khoản nhập vào.
* inputPassword: nhập mật khẩu của admin và lưu vào biến admin và passWord.
* sort: sắp xếp các tài khoản theo thứ tự tăng dần của số tài khoản.
* input: nhập thông tin các tài khoản từ tệp văn bản MAATM.txt và lưu vào mảng CARDS.
* withdrawMoney: thực hiện giao dịch rút tiền.
* transferMoney: thực hiện giao dịch chuyển tiền.
* checkBalance: thực hiện giao dịch kiểm tra số dư.
* withdrawMoneyActivity: gọi hàm withdrawMoney để thực hiện giao dịch rút tiền.
* transferMoneyActivity: gọi hàm transferMoney để thực hiện giao dịch chuyển tiền.
* checkBalanceActivity: gọi hàm checkBalance để thực hiện giao dịch kiểm tra số dư.
* menu: Vẽ giao diện menu
* atmScreen: thực hiện các thao tác trên màn hình ATM

Các hằng số định nghĩa:

* FEE\_AMOUNT: phí giao dịch
* TRANSACTION: giao dịch
* TRANSACTION\_LIMIT: giới hạn giao dịch
* MAX\_NUM\_ACCOUNTS: tổng số tài khoản đối đa
* MAX\_ACCNUMBER\_DIGITS: số tài khoản có độ dài tối đa
* MAX\_PIN\_DIGITS: số pin có độ dài tối đa
* NAME\_OF\_TRANSACTION: loại giao dịch

## Thuật toán

Chương trình sử dụng hai thuật toán quan trọng để viết nên chương trình là **Quick\_Sort** và **chặt nhị phân** (Binary Search Algorithm) :

* Đầu tiên chúng ta sử dụng “Quick\_Sort” để sắp xếp tài khoản ngân hàng theo một thứ tự tang dần.
* Sau đó , sử dụng thuật toán “Chặt nhịp phân” để tìm kiếm tài khoản ngân hàng của khách hàng khi sử dụng ATM bằng bằng cách chia mảng lưu dữ liệu tài khoản ngân hàng của ATM thành hai đoạn trái phải rồi sau đó tìm kiếm trên đoạn trái hoặc phải sao cho thỏa mãn chứa tài khoản của khách hàng
* Sau khi tìm kiếm ra thì truy cấp đến tài khoản ngân hàng đó để thực hiện các giao dịch như “Rút tiền” , “Gửi tiền” , “Xem tài khoản”.

**Quick\_Sort:**

* Hàm sort nhận vào hai chỉ số l và r, đại diện cho phần tử đầu và phần tử cuối của mảng cần sắp xếp, cùng với mảng CARDS chứa các phần tử cần sắp xếp.
* Một biến p được khai báo để lưu trữ giá trị của AccNumber tại vị trí trung tâm của mảng (giữa l và r). Giá trị này được sao chép vào biến p bằng hàm strcpy.
* Khởi tạo hai biến i và j, i được gán giá trị l và j được gán giá trị r.
* Một vòng lặp while được sử dụng để di chuyển các con trỏ i và j trong mảng. Vòng lặp này sẽ tiếp tục cho đến khi i >= j.
* Trong vòng lặp, hai vòng lặp while lồng nhau được sử dụng để tìm các phần tử phù hợp để hoán đổi. Vòng lặp đầu tiên kiểm tra các AccNumber từ vị trí i trong mảng CARDS cho đến khi tìm thấy một phần tử lớn hơn hoặc bằng giá trị p. Vòng lặp thứ hai kiểm tra các AccNumber từ vị trí j trong mảng CARDS cho đến khi tìm thấy một phần tử nhỏ hơn hoặc bằng giá trị p.
* Nếu i <= j sau khi hoàn thành vòng lặp while, tức là đã tìm thấy một cặp phần tử cần hoán đổi. Hai phần tử này được hoán đổi bằng cách sử dụng một biến tạm thời tam để trao đổi giá trị của CARDS[i] và CARDS[j]. Sau đó, i được tăng lên 1 và j được giảm đi 1 để di chuyển tiếp tục trong mảng.
* Sau khi hoàn thành vòng lặp while, mảng CARDS đã được chia thành hai phần: từ l đến j và từ i đến r. Để sắp xếp các phần này, ta sử dụng đệ quy bằng cách gọi hàm sort trên hai phần mảng này.
* Đệ quy tiếp tục cho đến khi mảng chỉ còn một phần tử hoặc không có phần tử nào cần sắp xếp. Trong trường hợp i < r, hàm sort được gọi trên phần mảng từ i đến r. Trong trường hợp i < r, tức là phần mảng từ i đến r vẫn chưa được sắp xếp, hàm sort được gọi đệ quy trên phần mảng này bằng cách truyền i làm chỉ số bắt đầu và r làm chỉ số kết thúc. Điều này sẽ tiếp tục quá trình sắp xếp các phần tử trong phạm vi từ i đến r.
* Tương tự, trong trường hợp j > l, tức là phần mảng từ l đến j vẫn chưa được sắp xếp, hàm sort được gọi đệ quy trên phần mảng này bằng cách truyền l làm chỉ số bắt đầu và j làm chỉ số kết thúc. Điều này sẽ tiếp tục quá trình sắp xếp các phần tử trong phạm vi từ l đến j.
* Khi cả hai đệ quy trên phần mảng từ i đến r và từ l đến j đã hoàn thành, tức là mảng CARDS từ vị trí l đến r đã được sắp xếp theo trường AccNumber.
* Độ phức tạp của hàm sort trong thuật toán Quick Sort trung bình là O(n log n), trong đó n là số lượng phần tử trong mảng, và độ phức tạp tối đa là O(n^2) trong trường hợp xấu nhất.

void sort(int l , int r , struct THE\_ATM CARDS[]) {

    char p[MAX\_ACCNUMBER\_DIGITS];

    strcpy(p , CARDS[(l + r) / 2].AccNumber);

    int i = l , j = r;

    while(i < j) {

        while(strcmp(CARDS[i].AccNumber , p) < 0) i++;

        while(strcmp(CARDS[j].AccNumber , p) > 0) j--;

        if(i <= j) {

            struct THE\_ATM tam = CARDS[i];

            CARDS[i] = CARDS[j];

            CARDS[j] = tam;

            i++;

            j--;

        }

    }

    if(i < r) sort(i , r , CARDS);

    if(j > l) sort(l , j , CARDS);

}

**Chặt nhị phân (Binary Search Algorithm):**

* Khởi tạo hai biến l và r để đánh dấu ranh giới trái và phải của mảng. Ban đầu, l được đặt bằng 1 (vị trí đầu tiên của mảng) và r được đặt bằng num\_of\_accounts (số lượng tài khoản).
* Tiếp theo, một vòng lặp while được sử dụng để tìm kiếm. Vòng lặp này sẽ tiếp tục cho đến khi l không còn nhỏ hơn hoặc bằng r.
* Trong vòng lặp, một biến i được khởi tạo làm trung tâm để tìm kiếm phần tử.
* Bước kiểm tra đầu tiên được thực hiện bằng cách sử dụng hàm strcmp để so sánh AccNumber tại vị trí i trong mảng CARDS với AccNumber cần tìm kiếm. Nếu hai chuỗi bằng nhau (trả về 0), tức là tìm thấy phần tử, thuật toán trả về giá trị i.
* Nếu AccNumber tại vị trí i trong mảng nhỏ hơn AccNumber cần tìm kiếm, ta cập nhật l = i + 1. Điều này đảm bảo rằng vùng tìm kiếm sẽ được thu hẹp xuống phía bên phải của i.
* Nếu AccNumber tại vị trí i trong mảng lớn hơn AccNumber cần tìm kiếm, ta cập nhật r = i - 1. Điều này đảm bảo rằng vùng tìm kiếm sẽ được thu hẹp xuống phía bên trái của i.
* Nếu không tìm thấy phần tử trong vòng lặp, tức là l > r, thuật toán sẽ trả về -1 để chỉ rằng phần tử không tồn tại trong mảng.
* Độ phức tạp của hàm chặt nhị phân trong thuật toán tìm kiếm là O(log n)

int findAcc(struct THE\_ATM CARDS[] , char AccNumber[] , int num\_of\_accounts) {

    int l = 1 , r = num\_of\_accounts;

    int i;

    while(l <= r) {

        int i = (l + r) / 2;

        if(strcmp(CARDS[i].AccNumber , AccNumber) == 0) return i;

        if(strcmp(CARDS[i].AccNumber , AccNumber) < 0) l = i + 1;

        else r = i - 1;

    }

    return -1;

}

# CHƯƠNG TRÌNH VÀ KẾT QUẢ

## Tổ chức chương trình

Để xây dựng ứng dụng thực hiện giao dịch trên máy ATM, có thể thực hiện các bước sau :

* Tạo Struct chứa các biến để lưu trữ thông tin tài khoản như account number, PIN, và balance.
* Đọc dữ liệu từ file chứa danh sách các tài khoản và lưu vào các mảng có kiểu dữ liệu Struct.
* Viết các hàm để thực hiện các chức năng như gửi tiền, rút tiền, xem tài khoản và kiểm tra số lần giao dịch của mỗi khách hàng.
* Sử dụng chặt nhị phân để xác nhận account number và PIN cho mỗi giao dịch của khách hàng.
* Thực hiện các giao dịch và cập nhật số tiền trong tài khoản.
* Lưu lại danh sách các giao dịch bao gồm account number và số tiền vào file.
* Thực hiện quá trình shutdown và ghi danh sách các giao dịch vào file.
* Ngoài ra cần có các file MAATM.txt (tệp chứa danh sách các tài khoản ATM, PIN và số tiền dư), Bienlai.txt (tệp chứa các giao dịch đã thực hiện thành công trong một ngày) và ADMIN.txt (tệp chứa lệnh kích hoạt và mã bảo mật).

A group of numbers on a white background

Description automatically generated with low confidence

*Hình 1: MAATM.txt*

A picture containing font, text, white, design

Description automatically generated

*Hình 2: ADMIN.txt*

**Sơ đồ hoạt động của chương trình**

Nhập số tài khoản , mã pin

Kiểm tra tài khoản , mã pin

Đúng

Chọn và thực hiện giao dịch “chuyển tiền” , “rút tiền” , “kiểm tra tài khoản”

Thông báo giao dịch thành công

Nhập kí tự bất kì để tiếp tục

Nhập mật khẩu

Sai

In ra biên lai

Nhập lại mã pin(tối đa 4 lần)

Sai

Đúng

Sai tài khoản

Sai mã pin

Đúng

Sai

Đúng

Sai

Kiểm tra

Kiểm tra kí tự có phải là “ADMIN”

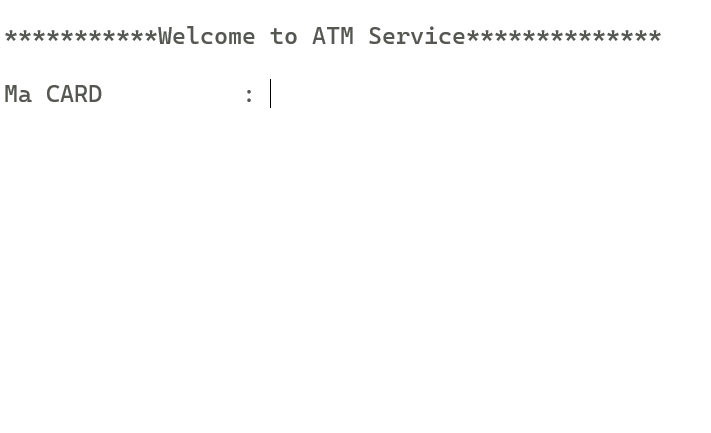
Kiểm tra

## Ngôn ngữ cài đặt

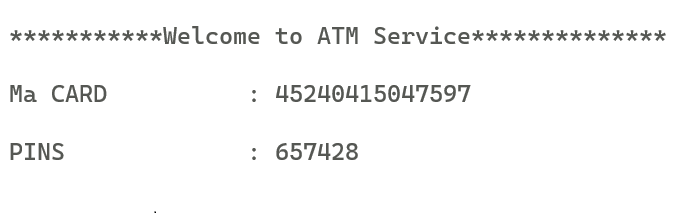
Ngôn ngữ lập trình C.

## Kết quả

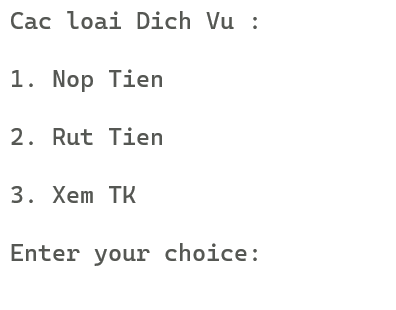
### Giao diện chính của chương trình



*Hình 3: Giao diện chính*



*Hình 4: Test Nhập*



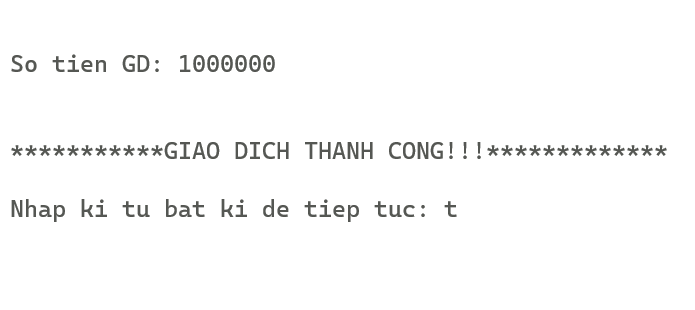
*Hình 5: MENU*

### Kết quả thực thi của chương trình

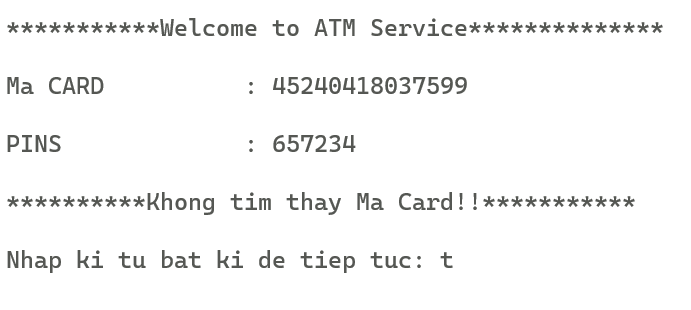
A picture containing text, screenshot, font, line

Description automatically generated

*Hình 6: Xem tài khoản*



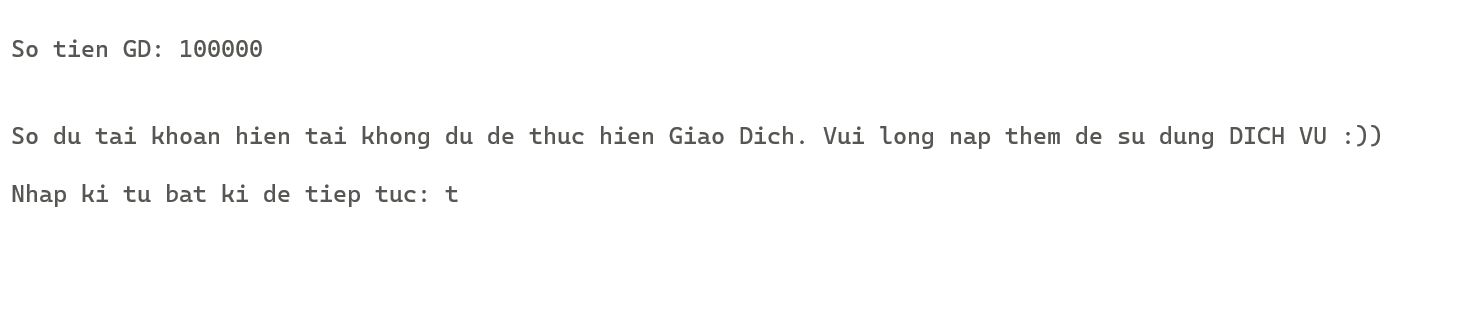
*Hình 7: Nộp tiền hoặc rút tiền thành công*



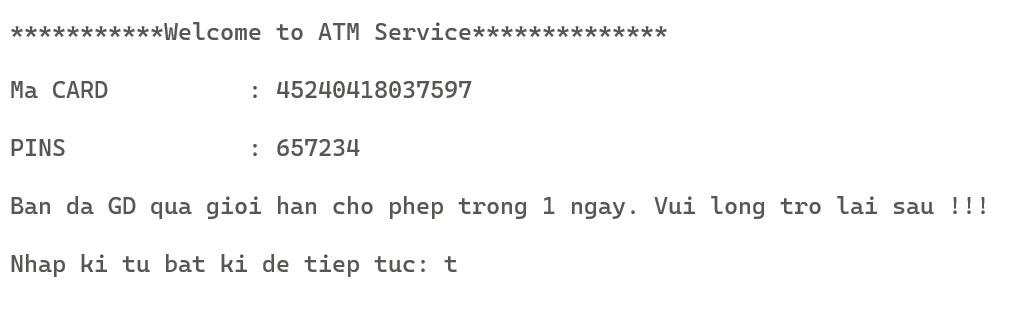
*Hình 8: Lỗi không tìm thấy ACC*



*Hình 9: Lỗi mã PIN*



*Hình 10: Lỗi không đủ tiền*



*Hình 11: Giới hạn giao dịch*

### Nhận xét đánh giá

Có thể thực hiện được hết 3 chức năng nộp tiền, rút tiền, xem tài khoản, kết quả giao dịch trong một ngày được in ra File Bienlai.txt. Tuy nhiên vẫn còn một vài mặt hạn chế về chức năng bảo mật và khả năng kiểm soát tiền. Đã sử dụng “Chặt nhị phân” thay vì duyệt thông thường đưa độ phức tạp từ O(n) -> O(log(n)).

Về phần giao diện vẫn còn đơn sơ.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Chương trình trên đã đáp ứng được mục tiêu, yêu cầu ban đầu đề ra, đáp ứng được các nhu cầu của khách hàng. Tuy nhiên vẫn còn vài mặt hạn chế.

## Hướng phát triển

* **Cải thiện tính bảo mật**: Tăng cường bảo mật ứng dụng để đảm bảo sự an toàn của khách hàng, như sử dụng mã PIN độ dài lớn hơn, thêm các phương thức xác thực hai bước, giám sát các giao dịch bất thường, vv.
* **Cải thiện tính tiện lợi**: Tối ưu hóa giao diện và trải nghiệm người dùng để giúp khách hàng có thể sử dụng ứng dụng dễ dàng và nhanh chóng, như tăng tốc độ giao dịch, thêm tính năng nạp tiền từ xa, vv.
* **Phát triển tính năng mới**: Thêm các tính năng mới như thanh toán di động, chuyển khoản ngay lập tức, thanh toán hóa đơn, quản lý tài khoản, vv.
* **Tích hợp trí tuệ nhân tạo**: Sử dụng trí tuệ nhân tạo và học máy để cải thiện tính năng bảo mật và tăng cường trải nghiệm người dùng, như phát hiện gian lận, phân tích hành vi khách hàng, vv.
* **Nâng cao tính tương tác**: Tăng cường tính tương tác của ứng dụng với khách hàng bằng cách thêm các tính năng chatbot hoặc trợ lý ảo để hỗ trợ khách hàng, giải đáp thắc mắc và cung cấp thông tin liên quan.
* **Tăng cường tính linh hoạt**: Cho phép khách hàng có thể sử dụng ứng dụng trên nhiều thiết bị khác nhau, cũng như tích hợp với các dịch vụ khác để tạo ra một hệ thống thanh toán toàn diện hơn.
* **Tối ưu hóa quản lý**: Cải thiện quản lý ứng dụng và dữ liệu để tăng cường hiệu quả và độ tin cậy của hệ thống, đồng thời cung cấp các báo cáo và phân tích dữ liệu để hỗ trợ quyết định kinh doanh.

Để cải thiện tính bảo mật, tiện lợi, tính năng mới, trí tuệ nhân tạo, tương tác, linh hoạt và quản lý của ứng dụng ATM, ta nên thực hiện các bước sau:

1. Đánh giá hiện trạng của ứng dụng ATM: Đánh giá mức độ hiệu quả và nhược điểm của ứng dụng hiện tại, cũng như các yêu cầu và mong muốn của khách hàng.
2. Xác định các ưu tiên và mục tiêu: Dựa trên đánh giá hiện trạng, xác định các ưu tiên và mục tiêu cải tiến cho từng lĩnh vực (bảo mật, tiện lợi, tính năng mới, trí tuệ nhân tạo, tương tác, linh hoạt và quản lý).
3. Lập kế hoạch và phân công: Lập kế hoạch chi tiết cho từng giai đoạn cải tiến, bao gồm nguồn lực, thời gian và trách nhiệm. Phân công nhiệm vụ cho các thành viên trong đội ngũ phát triển và quản lý.
4. Phát triển và triển khai các cải tiến: Hợp tác với đội ngũ kỹ thuật và phát triển để thực hiện các cải tiến đã lên kế hoạch. Đảm bảo hệ thống được kiểm tra và thử nghiệm kỹ lưỡng trước khi triển khai rộng rãi.
5. Thu thập và phân tích phản hồi: Sau khi triển khai các cải tiến, thu thập phản hồi từ khách hàng và phân tích hiệu quả của các cải tiến. Điều chỉnh và cải tiến liên tục dựa trên phản hồi khách hàng và dữ liệu thu được.
6. Đánh giá và theo dõi hiệu quả: Theo dõi chỉ số hiệu suất chính (KPIs) để đánh giá hiệu quả của các cải tiến và đưa ra quyết định về các cải tiến tiếp theo dựa trên dữ liệu và kết quả đánh giá.
7. Tái đánh giá và cập nhật kế hoạch: Định kỳ tái đánh giá hiện trạng ứng dụng ATM và cập nhật kế hoạch cải tiến dựa trên kết quả đánh giá và các yếu tố thay đổi trong thị trường và công nghệ.

Bằng cách thực hiện các bước trên, bạn sẽ giúp ứng dụng ATM của mình trở nên an toàn, tiện lợi và hấp dẫn hơn đối với khách hàng, từ đó tăng cường sự hài lòng của khách hàng và thúc đẩy doanh số giao dịch.

TÀI LIỆU THAM KHẢO

1. Tài liệu về “Quick Sort” : [QuickSort - GeeksforGeeks](https://www.geeksforgeeks.org/quick-sort/).
2. Tài liệu về “Chặt nhị phân” : [Tìm kiếm nhị phân – Wikipedia tiếng Việt](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_nh%E1%BB%8B_ph%C3%A2n).
3. Tài liệu tham khảo về “Lập trình C” : “Giáo trình kỹ thuật lập trình C căn bản và nâng cao – Phạm Văn Ất”

PHỤ LỤC :

* Source Code :

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <float.h>

#include <stdlib.h>

#define FEE\_AMOUNT 2.000

#define TRANSACTION 0

#define TRANSACTION\_LIMIT 2

#define MAX\_NUM\_ACCOUNTS 1000

#define MAX\_ACCNUMBER\_DIGITS 15

#define MAX\_PIN\_DIGITS 7

#define NAME\_OF\_TRANSACTION 50

int check;

struct THE\_ATM {

    char AccNumber[MAX\_ACCNUMBER\_DIGITS];

    char Pins[MAX\_PIN\_DIGITS];

    float Sodu;

};

struct BIEN\_LAI {

    char MaThe[MAX\_ACCNUMBER\_DIGITS];

    int MaLoaiGD;

    char TenLoaiGD[NAME\_OF\_TRANSACTION];

    float SoTienGD;

    float LePhi;

    float SoDuConLai;

};

int findAcc(struct THE\_ATM CARDS[] , char AccNumber[] , int num\_of\_accounts);

void inputPassword(char admin[MAX\_ACCNUMBER\_DIGITS] , char passWord[MAX\_ACCNUMBER\_DIGITS]);

void sort(int l , int r , struct THE\_ATM CARDS[]);

void input(int \*num\_of\_accounts , struct THE\_ATM CARDS[]);

void withdrawMoney(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[] , char AccNumber[] , char Pins[] , int SoLanGD[] , float SoTienGD , int \*total\_transactions , int i);

void transferMoney(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[]  , char AccNumber[] , char Pins[] , int SoLanGD[] , float SoTienGD , int \*total\_transactions , int i);

void checkBalance(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[]  , char AccNumber[] , char Pins[] , int SoLanGD[] , int \*total\_transactions , int i);

void withdrawMoneyActivity(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[] , char AccNumber[] , char Pins[] , int SoLanGD[] , float SoTienGD , int \*total\_transactions , int num\_of\_accounts , int i) ;

void transferMoneyActivity(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[] , char AccNumber[] , char Pins[] , int SoLanGD[] , float SoTienGD , int \*total\_transactions , int num\_of\_accounts , int i);

void checkBalanceActivity(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[] , char AccNumber[] , char Pins[] , int SoLanGD[] , int \*total\_transactions , int num\_of\_accounts , int i);

void print(struct BIEN\_LAI BL[] , int \*total\_transactions);

void checkAcc(struct THE\_ATM CARDS[] , char AccNumber[] , char Pins[]  , int SoLanGD[] , int i);

void menu(int \*selected\_option);

void atmScreen(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[] , int SoLanGD[] , float SoTienGD , int total\_transactions , int num\_of\_accounts , char admin[MAX\_ACCNUMBER\_DIGITS] , char passWord[MAX\_ACCNUMBER\_DIGITS]);

int main()

{

    int num\_of\_accounts, total\_transactions = 0;

    struct THE\_ATM CARDS[MAX\_NUM\_ACCOUNTS];

    char admin[MAX\_ACCNUMBER\_DIGITS] , passWord[MAX\_ACCNUMBER\_DIGITS];

    input(&num\_of\_accounts , CARDS);

    inputPassword(admin , passWord);

    struct BIEN\_LAI BL[num\_of\_accounts \* TRANSACTION\_LIMIT + 1];

    float SoTienGD;

    int SoLanGD[num\_of\_accounts + 1];

    // Khởi tạo số lần GD trong 1 ngày

    for(int i = 1; i <= num\_of\_accounts; i++)

        SoLanGD[i] = TRANSACTION;

    atmScreen(BL , CARDS , SoLanGD , SoTienGD , total\_transactions , num\_of\_accounts , admin , passWord);

    return 0;

}

void sort(int l , int r , struct THE\_ATM CARDS[]) {

    char p[MAX\_ACCNUMBER\_DIGITS];

    strcpy(p , CARDS[(l + r) / 2].AccNumber);

    int i = l , j = r;

    while(i < j) {

        while(strcmp(CARDS[i].AccNumber , p) < 0) i++;

        while(strcmp(CARDS[j].AccNumber , p) > 0) j--;

        if(i <= j) {

            struct THE\_ATM tam = CARDS[i];

            CARDS[i] = CARDS[j];

            CARDS[j] = tam;

            i++;

            j--;

        }

    }

    if(i < r) sort(i , r , CARDS);

    if(j > l) sort(l , j , CARDS);

}

void input(int \*num\_of\_accounts , struct THE\_ATM CARDS[]) {

    FILE \*fp = fopen("MAATM.txt", "r");

    int i = 1;

    while (fscanf(fp, "%s%s%f", CARDS[i].AccNumber, CARDS[i].Pins , &CARDS[i].Sodu) == 3) {

        i++;

    }

    \*num\_of\_accounts = i - 1;

    fclose(fp);

    sort(1 , i - 1, CARDS);

}

void inputPassword(char admin[MAX\_ACCNUMBER\_DIGITS] , char passWord[MAX\_ACCNUMBER\_DIGITS]) {

    FILE \*fp = fopen("ADMIN.txt", "r");

    fscanf(fp , "%s" , admin);

    fscanf(fp , "%s" , passWord);

    fclose(fp);

}

int findAcc(struct THE\_ATM CARDS[] , char AccNumber[] , int num\_of\_accounts) {

    int l = 1 , r = num\_of\_accounts;

    int i;

    while(l <= r) {

        int i = (l + r) / 2;

        if(strcmp(CARDS[i].AccNumber , AccNumber) == 0) return i;

        if(strcmp(CARDS[i].AccNumber , AccNumber) < 0) l = i + 1;

        else r = i - 1;

    }

    return -1;

}

void checkAcc(struct THE\_ATM CARDS[] , char AccNumber[] , char Pins[]  , int SoLanGD[] , int i) {

    check = -1;

    if(i == -1) {

        check = 0;

        return;

    }

    if(strcmp(CARDS[i].AccNumber , AccNumber) != 0) {

        check = 0;

        return;

    }

    if(strcmp(CARDS[i].Pins , Pins) != 0) {

        check = 2;

        return;

    }

    if(SoLanGD[i] >= TRANSACTION\_LIMIT) {

        check = 4;

        return;

    }

}

void print(struct BIEN\_LAI BL[] , int \*total\_transactions) {

    FILE \*fp;

    fp = fopen("Bienlai.txt", "w");

    int i;

    for(i = 0 ; i < \*total\_transactions ; ++i) {

        fprintf(fp , "# %d:\n" , i + 1);

        fprintf(fp , "MA CARD    :%s\n" , BL[i].MaThe);

        fprintf(fp , "Ma GD      :%d\n" , BL[i].MaLoaiGD);

        fprintf(fp , "Ten Loai GD:%s\n" , BL[i].TenLoaiGD);

        fprintf(fp , "So Tien GD :%0.0f\n" , BL[i].SoTienGD);

        fprintf(fp , "Le Phi     :%0.0f\n" , BL[i].LePhi);

        fprintf(fp , "So Du TK   :%0.0f\n" , BL[i].SoDuConLai);

        fprintf(fp , "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    }

    fclose(fp);

}

void withdrawMoney(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[]  , char AccNumber[] , char Pins[] , int SoLanGD[] , float SoTienGD , int \*total\_transactions , int i) {

    struct BIEN\_LAI RL;

    strcpy(RL.MaThe , AccNumber);

    RL.MaLoaiGD = 1;

    strcpy(RL.TenLoaiGD , "GUI TIEN");

    RL.SoTienGD = SoTienGD;

    RL.LePhi = TRANSACTION;

    RL.SoDuConLai = 0;

    CARDS[i].Sodu = CARDS[i].Sodu + SoTienGD;

    RL.SoDuConLai = RL.SoDuConLai + CARDS[i].Sodu;

    SoLanGD[i]++;

    BL[\*total\_transactions] = RL;

    \*total\_transactions+=1;

}

void transferMoney(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[]  , char AccNumber[] , char Pins[] , int SoLanGD[] , float SoTienGD , int \*total\_transactions, int i) {

    struct BIEN\_LAI RL;

    strcpy(RL.MaThe , AccNumber);

    RL.MaLoaiGD = 2;

    strcpy(RL.TenLoaiGD , "RUT TIEN");

    RL.SoTienGD = SoTienGD;

    RL.LePhi = FEE\_AMOUNT;

    RL.SoDuConLai = 0;

    CARDS[i].Sodu = CARDS[i].Sodu - SoTienGD - RL.LePhi;

    RL.SoDuConLai = RL.SoDuConLai + CARDS[i].Sodu;

    SoLanGD[i]++;

    BL[\*total\_transactions] = RL;

    \*total\_transactions+=1;

}

void checkBalance(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[]  , char AccNumber[] , char Pins[] , int SoLanGD[] , int \*total\_transactions , int i) {

    struct BIEN\_LAI RL;

    strcpy(RL.MaThe , AccNumber);

    RL.MaLoaiGD = 3;

    strcpy(RL.TenLoaiGD , "XEM TAI KHOAN");

    RL.SoTienGD = 0;

    RL.LePhi = FEE\_AMOUNT;

    RL.SoDuConLai = 0;

    CARDS[i].Sodu = CARDS[i].Sodu - RL.LePhi;

    RL.SoDuConLai = RL.SoDuConLai + CARDS[i].Sodu;

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

    printf("MA CARD             : %s\n\n", AccNumber);

    printf("So tien Du kha dung : %0.0f\n\n", RL.SoDuConLai);

    SoLanGD[i]++;

    BL[\*total\_transactions] = RL;

    \*total\_transactions+=1;

}

void withdrawMoneyActivity(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[] , char AccNumber[] , char Pins[] , int SoLanGD[] , float SoTienGD , int \*total\_transactions , int num\_of\_accounts , int i)  {

    if(strcmp(CARDS[i].AccNumber , AccNumber) == 0 && strcmp(CARDS[i].Pins , Pins) == 0) {

        printf("\n\nSo tien GD: ");

        scanf("%f" , &SoTienGD);

        printf("\n\n");

        while(SoTienGD <= 0) {

            printf("So tien khong hop le!\n");

            printf("Vui long nhap lai so tien GD: ", CARDS[i].Sodu);

            scanf("%f" , &SoTienGD);

        }

        withdrawMoney( BL ,  CARDS , AccNumber , Pins , SoLanGD ,  SoTienGD , total\_transactions , i);

        check = 1;

        return;

    }

}

void transferMoneyActivity(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[] , char AccNumber[] , char Pins[] , int SoLanGD[] , float SoTienGD , int \*total\_transactions , int num\_of\_accounts , int i) {

    if(strcmp(CARDS[i].AccNumber , AccNumber) == 0 && strcmp(CARDS[i].Pins , Pins) == 0) {

        printf("\n\nSo tien GD: ");

        scanf("%f" , &SoTienGD);

        printf("\n\n");

        while(SoTienGD <= 0) {

            printf("So tien khong hop le!\n");

            printf("Vui long nhap lai so tien GD: ", CARDS[i].Sodu);

            scanf("%f" , &SoTienGD);

        }

        if(SoTienGD > CARDS[i].Sodu - FEE\_AMOUNT) {

            check = 3;

            return;

        }

        transferMoney( BL ,  CARDS , AccNumber , Pins , SoLanGD ,  SoTienGD , total\_transactions , i);

        check = 1;

        return;

    }

}

void checkBalanceActivity(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[] , char AccNumber[] , char Pins[] , int SoLanGD[] , int \*total\_transactions , int num\_of\_accounts , int i) {

    if(strcmp(CARDS[i].AccNumber , AccNumber) == 0 && strcmp(CARDS[i].Pins , Pins) == 0) {

        checkBalance( BL ,  CARDS , AccNumber , Pins , SoLanGD  , total\_transactions , i);

        check = 1;

        return;

    }

}

void menu(int \*selected\_option) {

    int TempChoice;

    system("cls");

    printf("Cac loai Dich Vu : \n\n");

    printf("1. Nop Tien \n\n");

    printf("2. Rut Tien \n\n");

    printf("3. Xem TK \n\n");

    printf("Enter your choice: ");  scanf("%d" , &TempChoice);

    while(TempChoice < 1 || TempChoice > 3) {

        printf("Enter your choice again: ");

        scanf("%d", &TempChoice);

    }

    \*selected\_option = TempChoice;

    system("cls");

}

void atmScreen(struct BIEN\_LAI BL[] , struct THE\_ATM CARDS[] , int SoLanGD[] , float SoTienGD , int total\_transactions , int num\_of\_accounts , char admin[MAX\_ACCNUMBER\_DIGITS] , char passWord[MAX\_ACCNUMBER\_DIGITS]) {

    char AccNumber[15];

    char Pins[7];

    int selected\_option;

    int is\_active = 1;

    char TE[15];

    while(is\_active == 1) {

        selected\_option = 0;

        printf("\n\*\*\*\*\*\*\*\*\*\*\*Welcome to ATM Service\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

        printf("Ma CARD          : ");  scanf("%s" , &AccNumber); printf("\n");

        printf("PINS             : ");  scanf("%s" , &Pins);      printf("\n");

        int i = findAcc(CARDS , AccNumber , num\_of\_accounts);

        checkAcc(CARDS , AccNumber , Pins  , SoLanGD , i);

        if(check == 0) printf("\*\*\*\*\*\*\*\*\*\*Khong tim thay Ma Card!!\*\*\*\*\*\*\*\*\*\*\*\n\n");

        else if(check == 2) {

            printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ma Pin Sai!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

            int count = 0;

            while(count <= 3 && check == 2) {

                printf("Vui Long Nhap Lai Ma Pin       :  "); scanf("%s" , &Pins);    printf("\n");

                checkAcc(CARDS , AccNumber , Pins  , SoLanGD , i);

                if(check == 2) printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ma Pin Sai!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

                else {

                    menu(&selected\_option);

                    break;

                }

                count++;

            }

            if(check == 2) printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ban Da Vuot Qua So Lan Thu Ma Pin Cho Phep!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

        }

        else if(check == 4) printf("Ban da GD qua gioi han cho phep trong 1 ngay. Vui long tro lai sau !!!\n\n");

        else menu(&selected\_option);

        if(selected\_option == 1) withdrawMoneyActivity(BL , CARDS , AccNumber , Pins , SoLanGD , SoTienGD , &total\_transactions , num\_of\_accounts , i);

        else if(selected\_option == 2) transferMoneyActivity(BL , CARDS , AccNumber , Pins , SoLanGD , SoTienGD , &total\_transactions , num\_of\_accounts , i);

        else if(selected\_option == 3) checkBalanceActivity(BL , CARDS , AccNumber , Pins , SoLanGD , &total\_transactions , num\_of\_accounts , i);

        if(check == 1) printf("\*\*\*\*\*\*\*\*\*\*\*GIAO DICH THANH CONG!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

        else if(check == 3) printf("So du tai khoan hien tai khong du de thuc hien Giao Dich. Vui long nap them de su dung DICH VU :))\n\n");

        printf("Nhap ki tu bat ki de tiep tuc: ");

        scanf("%s" , &TE);

        system("cls");

        if(strcmp(TE , admin) == 0) {

            printf("Nhap MA BAO MAT ATM: ");

            scanf("%s" , &TE);

            printf("\n");

            if(strcmp(TE , passWord) == 0) is\_active = 0;

        }

    }

    print(BL , &total\_transactions);

}