TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN

ỨNG DỤNG GIAO DỊCH ATM

Người hướng dẫn**: PGS. TS. NGUYỄN THỊ LỆ QUYÊN**
Sinh viên thực hiện**:**
Hồ Sĩ Thảo **LỚP: 22T_NHAT1 NHÓM: 09**
Phạm Phan Thành **LỚP: 22T_NHAT1 NHÓM: 09**

Đà Nẵng, 04/2020

MỤC LỤC

DANH MỤC HÌNH VẼ

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.

1. 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.

2. CƠ SỞ LÝ THUYẾT

2.1. Ý tưởng

Để thực hiện đề tài, ta cần đọc danh sách các account number, 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 account number và PIN. Nếu giao dịch thành công, máy ATM cần lưu lại danh sách các giao dịch bao gồm số account number 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 shutdown và ghi danh sách giao dịch vào file.

2.2. Cơ sở lý thuyết

Cần 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.

3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

3.1. 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.

3.2. 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.
- SOLVEGD1: thực hiện giao dịch rút tiền.
- SOLVEGD2: thực hiện giao dịch chuyển tiền.
- SOLVEGD3: thực hiện giao dịch kiểm tra số dư.
- GD1: gọi hàm SOLVEGD1 để thực hiện giao dịch rút tiền.

- GD2: gọi hàm SOLVEGD2 để thực hiện giao dịch chuyển tiền.
- GD3: gọi hàm SOLVEGD3 để thực hiện giao dịch kiểm tra số dư.
- MENU: Vẽ giao diện menu
- ATM_SCREEN: 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

3.3. 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).

Quick_Sort:

```
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);
}
```

Hình 1: Quick_Sort

Chặt nhị phân (Binary Search Algorithm):

```
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;
}
```

Hình 2: Binary_Search_Algorithm

4. CHƯƠNG TRÌNH VÀ KẾT QUẢ

4.1. 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).

```
452404150475971  657428  0.0
452404180375971  657234  20000.000
452404197335971  656543  20000.000
452404187335971  653212  20000.000
```

Hình 3: MAATM.txt

```
ADMIN
121604
```

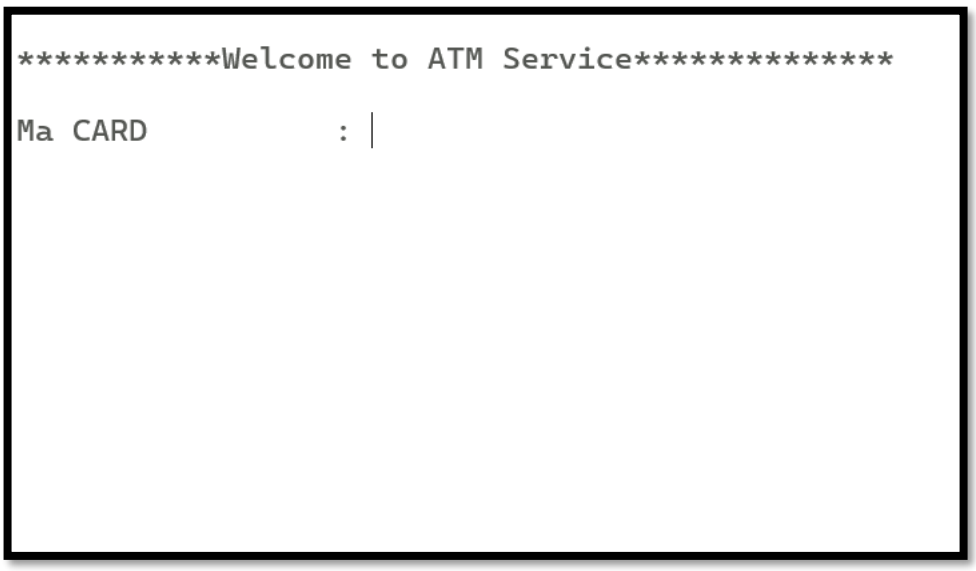
Hình 4: ADMIN.txt

4.2. Ngôn ngữ cài đặt

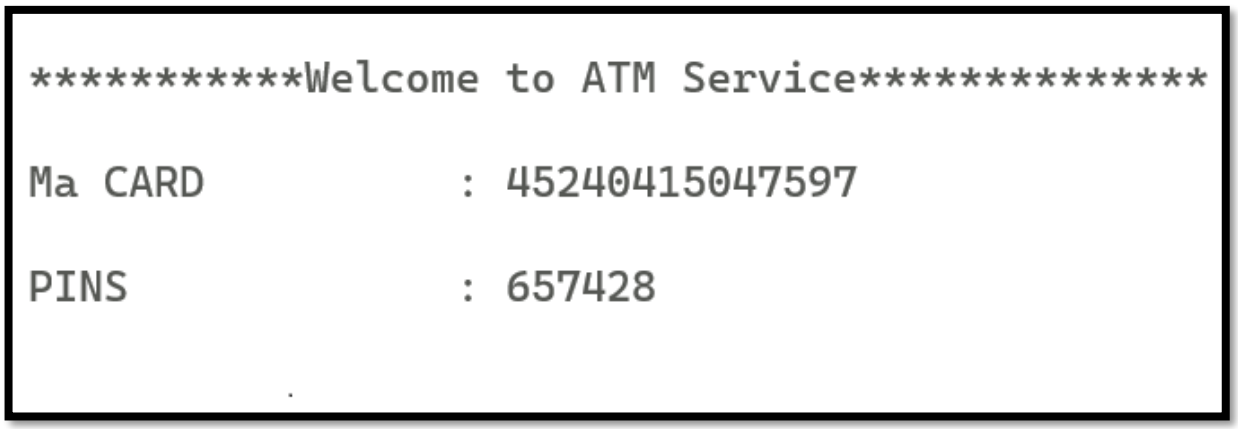
Ngôn ngữ lập trình C.

4.3. Kết quả

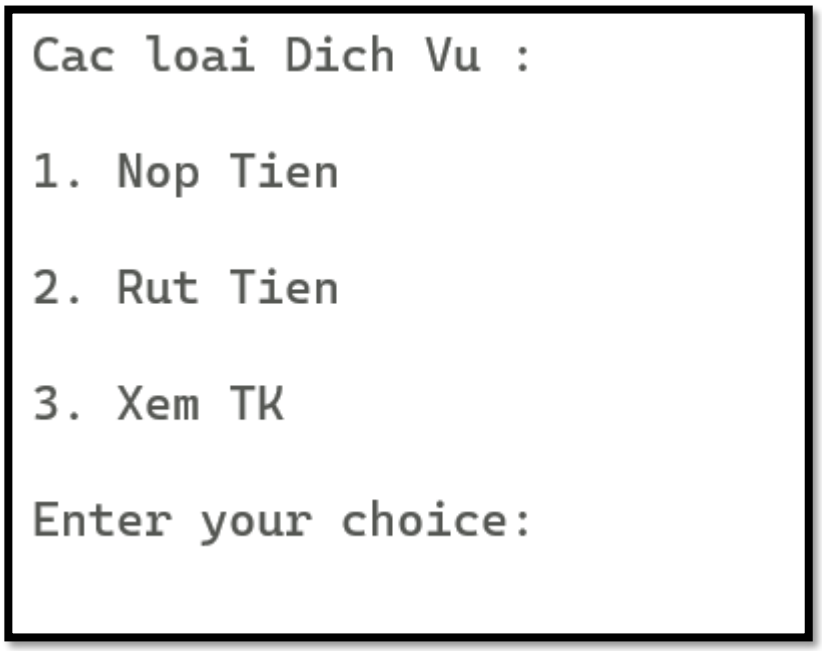
4.3.1. Giao diện chính của chương trình



Hình 5: Giao diện chính

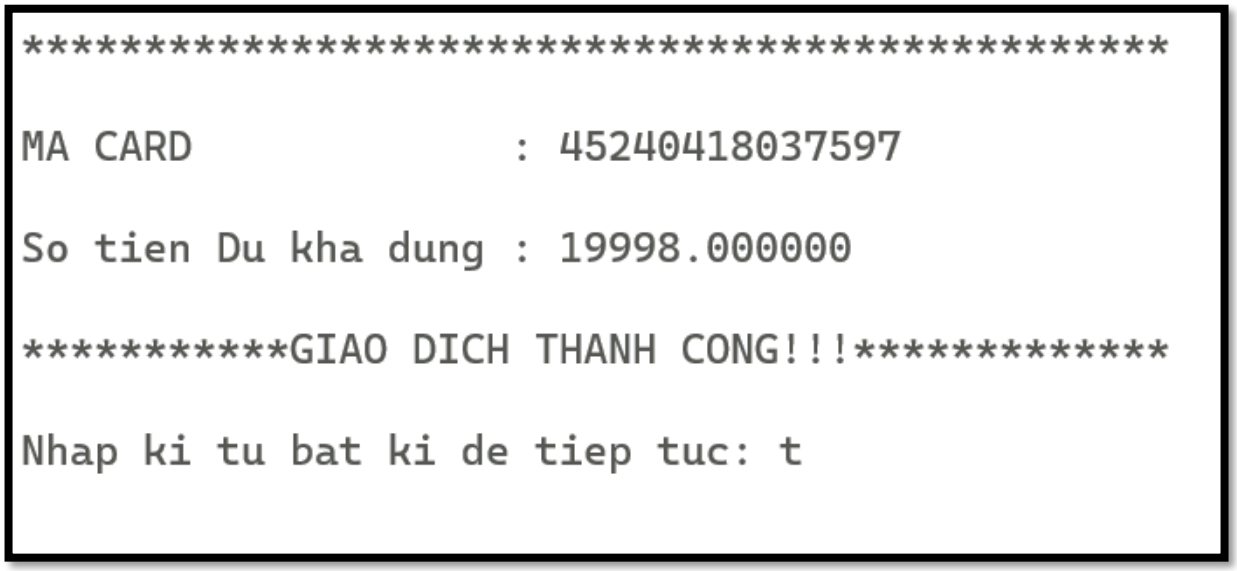


Hình 6: Test Nhập



Hình 7: MENU

4.3.2. Kết quả thực thi của chương trình



Hình 8: Xem tài khoản

Hình 9: Nộp tiền hoặc rút tiền thành công

```
**********Welcome to ATM Service**************

Ma CARD           : 45240418037599

PINS              : 657234

**********Khong tim thay Ma Card!!***********

Nhap ki tu bat ki de tiep tuc: t
```

Hình 10: Lỗi không tìm thấy ACC

```
**********Welcome to ATM Service**************
Ma CARD          : 45240418037597
PINS             : 656565
*****************Ma Pin Sai!!****************
Vui Long Nhap Lai Ma Pin       :  656565
*****************Ma Pin Sai!!****************
Vui Long Nhap Lai Ma Pin       :  656565
*****************Ma Pin Sai!!****************
Vui Long Nhap Lai Ma Pin       :  656565
*****************Ma Pin Sai!!****************
*****************Ban Da Vuot Qua So Lan Thu Ma Pin Cho Phep!!****************
Nhap ki tu bat ki de tiep tuc: t
```

Hình 11: Lỗi mã PIN

```
So tien GD: 100000

So du tai khoan hien tai khong du de thuc hien Giao Dich. Vui long nap them de su dung DICH VU :))
Nhap ki tu bat ki de tiep tuc: t
```

Hình 12: Lỗi không đủ tiền

```
**********Welcome to ATM Service**************
Ma CARD          : 45240418037597
PINS             : 657234
Ban da GD qua gioi han cho phep trong 1 ngay. Vui long tro lai sau !!!
Nhap ki tu bat ki de tiep tuc: t
```

Hình 13: Giới hạn giao dịch

4.3.3. 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 sửa 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.

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. 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.

5.2. 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.

TÀI LIỆU THAM KHẢO

Tài liệu về "Quick Sort" : QuickSort - GeeksforGeeks.

Tài liệu về "Chặt nhị phân" : Tìm kiếm nhị phân – Wikipedia tiếng Việt.

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

Hình 14: Source code

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <float.h>
#include <conio.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]);
```

```
37
38  void SORT(int l , int r , struct THE_ATM CARDS[]);
39
40  void INPUT(int *num_of_accounts , struct THE_ATM CARDS[]);
41
42  void SOLVEGD1(struct BIEN_LAI BL[] , struct THE_ATM CARDS[] , char AccNumber[] , char Pins[]
43      , int SoLanGD[] , float SoTienGD , int *total_transactions , int i);
44
45  void SOLVEGD2(struct BIEN_LAI BL[] , struct THE_ATM CARDS[]  , char AccNumber[] , char Pins[]
46      , int SoLanGD[] , float SoTienGD , int *total_transactions , int i);
47
48  void SOLVEGD3(struct BIEN_LAI BL[] , struct THE_ATM CARDS[]  , char AccNumber[] , char Pins[]
49      , int SoLanGD[] , int *total_transactions , int i);
50
51  void GD1(struct BIEN_LAI BL[] , struct THE_ATM CARDS[] , char AccNumber[] , char Pins[]
52      , int SoLanGD[] , float SoTienGD , int *total_transactions , int num_of_accounts , int i) ;
53
54  void GD2(struct BIEN_LAI BL[] , struct THE_ATM CARDS[] , char AccNumber[] , char Pins[]
55      , int SoLanGD[] , float SoTienGD , int *total_transactions , int num_of_accounts , int i);
56
57  void GD3(struct BIEN_LAI BL[] , struct THE_ATM CARDS[] , char AccNumber[] , char Pins[]
58      , int SoLanGD[] , int *total_transactions , int num_of_accounts , int i);
59
60  void PRINT(struct BIEN_LAI BL[] , int *total_transactions);
61
62  void CHECKACC(struct THE_ATM CARDS[] , char AccNumber[] , char Pins[]  , int SoLanGD[] , int i);
63
64  void MENU(int *selected_option);
65
66  void ATM_SCREEN(struct BIEN_LAI BL[] , struct THE_ATM CARDS[] , int SoLanGD[] , float SoTienGD
67      , int total_transactions , int num_of_accounts , char admin[max_accNumber_digits]
68      , char passWord[max_accNumber_digits]);
69
70  int main()
71  {
72      int num_of_accounts, total_transactions = 0;
73
74      struct THE_ATM CARDS[max_num_accounts];
75      char admin[max_accNumber_digits] , passWord[max_accNumber_digits];
76
77      INPUT(&num_of_accounts , CARDS);
78      INPUTPASSWORD(admin , passWord);
79
80
81      struct BIEN_LAI BL[num_of_accounts * transaction_limit + 1];
82
83      float SoTienGD;
84      int SoLanGD[num_of_accounts + 1];
85
86      // Khởi tạo số lần GD trong 1 ngày
87      for(int i = 1; i <= num_of_accounts; i++)
88          SoLanGD[i] = transaction;
89
90      ATM_SCREEN(BL , CARDS , SoLanGD , SoTienGD , total_transactions , num_of_accounts
91          , admin , passWord);
```

```
92
93      return 0;
94  }
95
96
97  void SORT(int l , int r , struct THE_ATM CARDS[]) {
98      char p[max_accNumber_digits];
99      strcpy(p , CARDS[(l + r) / 2].AccNumber);
100     int i = l , j = r;
101     while(i < j) {
102         while(strcmp(CARDS[i].AccNumber , p) < 0) i++;
103         while(strcmp(CARDS[j].AccNumber , p) > 0) j--;
104         if(i <= j) {
105             struct THE_ATM tam = CARDS[i];
106             CARDS[i] = CARDS[j];
107             CARDS[j] = tam;
108             i++;
109             j--;
110         }
111     }
112     if(i < r) SORT(i , r , CARDS);
113     if(j > l) SORT(l , j , CARDS);
114 }
```

```
115
116 void INPUT(int *num_of_accounts , struct THE_ATM CARDS[]) {
117
118     FILE *fp = fopen("MAATM.txt", "r");
119
120     int i = 1;
121     while (fscanf(fp, "%s%s%f", CARDS[i].AccNumber, CARDS[i].Pins , &CARDS[i].Sodu) == 3) {
122         i++;
123     }
124     *num_of_accounts = i - 1;
125
126     fclose(fp);
127
128     SORT(1 , i - 1, CARDS);
129 }
130
131
132 void INPUTPASSWORD(char admin[max_accNumber_digits] , char passWord[max_accNumber_digits]) {
133     FILE *fp = fopen("ADMIN.txt", "r");
134     fscanf(fp , "%s" , admin);
135     fscanf(fp , "%s" , passWord);
136     fclose(fp);
137 }
138
139 int FINDACC(struct THE_ATM CARDS[] , char AccNumber[] , int num_of_accounts) {
140     int l = 1 , r = num_of_accounts;
141     int i;
142
143     while(l <= r) {
144         int i = (l + r) / 2;
145
146         if(strcmp(CARDS[i].AccNumber , AccNumber) == 0) return i;
147
148         if(strcmp(CARDS[i].AccNumber , AccNumber) < 0) l = i + 1;
149
150         else r = i - 1;
150         else r = i - 1;
151     }
152
153     return -1;
154 }
155
156 void CHECKACC(struct THE_ATM CARDS[] , char AccNumber[] , char Pins[]  , int SoLanGD[] , int i) {
157     check = -1;
158
159     if(i == -1) {
160         check = 0;
161         return;
162     }
163
164     if(strcmp(CARDS[i].AccNumber , AccNumber) != 0) {
165         check = 0;
166         return;
167     }
168
169     if(strcmp(CARDS[i].Pins , Pins) != 0) {
170         check = 2;
171         return;
172     }
173
174     if(SoLanGD[i] >= transaction_limit) {
175         check = 4;
176         return;
177     }
178 }
```

```
179
180 void PRINT(struct BIEN_LAI BL[] , int *total_transactions) {
181
182     FILE *fp;
183     fp = fopen("Bienlai.txt", "w");
184
185     int i;
186
187     for(i = 0 ; i < *total_transactions ; ++i) {
188         fprintf(fp , "# %d:\n" , i + 1);
189         fprintf(fp , "MA CARD    :%s\n" , BL[i].MaThe);
190         fprintf(fp , "Ma GD      :%d\n" , BL[i].MaLoaiGD);
191         fprintf(fp , "Ten Loai GD:%s\n" , BL[i].TenLoaiGD);
192         fprintf(fp , "So Tien GD :%0.0f\n" , BL[i].SoTienGD);
193         fprintf(fp , "Le Phi     :%0.0f\n" , BL[i].LePhi);
194         fprintf(fp , "So Du TK   :%0.0f\n" , BL[i].SoDuConLai);
195         fprintf(fp , "*********************************************\n");
196     }
197
198     fclose(fp);
199 }
200
201
202 void SOLVEGD1(struct BIEN_LAI BL[] , struct THE_ATM CARDS[]  , char AccNumber[]
203     , char Pins[] , int SoLanGD[] , float SoTienGD , int *total_transactions , int i) {
204     struct BIEN_LAI RL;
205     strcpy(RL.MaThe , AccNumber);
206     RL.MaLoaiGD = 1;
207     strcpy(RL.TenLoaiGD , "GUI TIEN");
208     RL.SoTienGD = SoTienGD;
209     RL.LePhi = transaction;
210     RL.SoDuConLai = 0;
211
212     CARDS[i].Sodu = CARDS[i].Sodu + SoTienGD;
213     RL.SoDuConLai = RL.SoDuConLai + CARDS[i].Sodu;
214
215     SoLanGD[i]++;
216
217     BL[*total_transactions] = RL;
218     *total_transactions+=1;
219 }
220
221
222 void SOLVEGD2(struct BIEN_LAI BL[] , struct THE_ATM CARDS[]  , char AccNumber[]
223     , char Pins[] , int SoLanGD[] , float SoTienGD , int *total_transactions, int i) {
224     struct BIEN_LAI RL;
225     strcpy(RL.MaThe , AccNumber);
226     RL.MaLoaiGD = 2;
227     strcpy(RL.TenLoaiGD , "RUT TIEN");
228     RL.SoTienGD = SoTienGD;
229     RL.LePhi = fee_amount;
230     RL.SoDuConLai = 0;
231
232     CARDS[i].Sodu = CARDS[i].Sodu - SoTienGD - RL.LePhi;
233     RL.SoDuConLai = RL.SoDuConLai + CARDS[i].Sodu;
234
235     SoLanGD[i]++;
236
237     BL[*total_transactions] = RL;
238     *total_transactions+=1;
239 }
```

```
240
241 void SOLVEGD3(struct BIEN_LAI BL[] , struct THE_ATM CARDS[]  , char AccNumber[]
242     , char Pins[] , int SoLanGD[] , int *total_transactions , int i) {
243     struct BIEN_LAI RL;
244     strcpy(RL.MaThe , AccNumber);
245     RL.MaLoaiGD = 3;
246     strcpy(RL.TenLoaiGD , "XEM TAI KHOAN");
247     RL.SoTienGD = 0;
248     RL.LePhi = fee_amount;
249     RL.SoDuConLai = 0;
250
251     CARDS[i].Sodu = CARDS[i].Sodu - RL.LePhi;
252     RL.SoDuConLai = RL.SoDuConLai + CARDS[i].Sodu;
253
254     printf("*********************************************\n\n");
255     printf("MA CARD             : %s\n\n", AccNumber);
256     printf("So tien Du kha dung : %f\n\n", RL.SoDuConLai);
257
258     SoLanGD[i]++;
259
260     BL[*total_transactions] = RL;
261     *total_transactions+=1;
262 }
263
264
265 void GD1(struct BIEN_LAI BL[] , struct THE_ATM CARDS[] , char AccNumber[]
266     , char Pins[] , int SoLanGD[] , float SoTienGD , int *total_transactions
267     , int num_of_accounts , int i)  {
268     if(strcmp(CARDS[i].AccNumber , AccNumber) == 0 && strcmp(CARDS[i].Pins , Pins) == 0) {
269         printf("\n\nSo tien GD: ");
270         scanf("%f" , &SoTienGD);
271         printf("\n\n");
272
273         while(SoTienGD <= 0) {
274             printf("So tien khong hop le!\n");
275             printf("Vui long nhap lai so tien GD: ", CARDS[i].Sodu);
276             scanf("%f" , &SoTienGD);
277         }
278
279         SOLVEGD1( BL ,  CARDS , AccNumber , Pins , SoLanGD ,  SoTienGD , total_transactions , i);
280
281         check = 1;
282         return;
283     }
284 }
285
286
287 void GD2(struct BIEN_LAI BL[] , struct THE_ATM CARDS[] , char AccNumber[] , char Pins[]
288     , int SoLanGD[] , float SoTienGD , int *total_transactions , int num_of_accounts , int i) {
289     if(strcmp(CARDS[i].AccNumber , AccNumber) == 0 && strcmp(CARDS[i].Pins , Pins) == 0) {
290         printf("\n\nSo tien GD: ");
291         scanf("%f" , &SoTienGD);
292         printf("\n\n");
293
294         while(SoTienGD <= 0) {
295             printf("So tien khong hop le!\n");
296             printf("Vui long nhap lai so tien GD: ", CARDS[i].Sodu);
297             scanf("%f" , &SoTienGD);
298         }
299
300         if(SoTienGD > CARDS[i].Sodu - fee_amount) {
301             check = 3;
302             return;
303         }
304
```

```
305             SOLVEGD2( BL ,  CARDS , AccNumber , Pins , SoLanGD ,  SoTienGD , total_transactions , i);
306
307             check = 1;
308             return;
309         }
310     }
311
312
313     void GD3(struct BIEN_LAI BL[] , struct THE_ATM CARDS[] , char AccNumber[]
314         , char Pins[] , int SoLanGD[] , int *total_transactions , int num_of_accounts , int i) {
315         if(strcmp(CARDS[i].AccNumber , AccNumber) == 0 && strcmp(CARDS[i].Pins , Pins) == 0) {
316             SOLVEGD3( BL ,  CARDS , AccNumber , Pins , SoLanGD  , total_transactions , i);
317
318             check = 1;
319             return;
320         }
321     }
322
323     void MENU(int *selected_option) {
324         int TempChoice;
325
326         system("cls");
327         printf("Cac loai Dich Vu : \n\n");
328         printf("1. Nop Tien \n\n");
329         printf("2. Rut Tien \n\n");
330         printf("3. Xem TK \n\n");
331         printf("Enter your choice: ");  scanf("%d" , &TempChoice);
332
333         while(TempChoice < 1 || TempChoice > 3) {
334             printf("Enter your choice again: ");
335             scanf("%d", &TempChoice);
336         }
337
338         *selected_option = TempChoice;
339         system("cls");
340     }
341
342     void ATM_SCREEN(struct BIEN_LAI BL[] , struct THE_ATM CARDS[] , int SoLanGD[]
343         , float SoTienGD , int total_transactions , int num_of_accounts
344         , char admin[max_accNumber_digits] , char passWord[max_accNumber_digits]) {
345         char AccNumber[15];
346         char Pins[7];
347
348         int selected_option;
349         int is_active = 1;
350         char TE[15];
351
352         while(is_active == 1) {
353             selected_option = 0;
354
355             printf("\n**********Welcome to ATM Service**************\n\n");
356             printf("Ma CARD          : ");  scanf("%s" , &AccNumber); printf("\n");
357             printf("PINS             : ");  scanf("%s" , &Pins);      printf("\n");
358
359             int i = FINDACC(CARDS , AccNumber , num_of_accounts);
360             CHECKACC(CARDS , AccNumber , Pins  , SoLanGD , i);
361
362             if(check == 0) printf("**********Khong tim thay Ma Card!!**********\n\n");
363             else if(check == 2) {
364                 printf("*****************Ma Pin Sai!!*****************\n\n");
365                 int count = 1;
366                 while(count <= 3 && check == 2) {
367                     printf("Vui Long Nhap Lai Ma Pin       :  "); scanf("%s" , &Pins);    printf("\n");
368                     CHECKACC(CARDS , AccNumber , Pins  , SoLanGD , i);
369                     if(check == 2) printf("*****************Ma Pin Sai!!*****************\n\n");
370                     else {
371                         MENU(&selected_option);
```

```
372                     break;
373                 }
374                 count++;
375             }
376             if(check == 2) printf("*****************Ban Da Vuot Qua So Lan Thu Ma Pin Cho Phep!!***************\n\n");
377         }
378         else if(check == 4) printf("Ban da GD qua gioi han cho phep trong 1 ngay. Vui long tro lai sau !!!\n\n");
379         else MENU(&selected_option);
380 
381         if(selected_option == 1) GD1(BL , CARDS , AccNumber , Pins , SoLanGD , SoTienGD , &total_transactions , num_of_accounts , i);
382         else if(selected_option == 2) GD2(BL , CARDS , AccNumber , Pins , SoLanGD , SoTienGD , &total_transactions , num_of_accounts , i);
383         else if(selected_option == 3) GD3(BL , CARDS , AccNumber , Pins , SoLanGD , &total_transactions , num_of_accounts , i);
384 
385         if(check == 1) printf("***********GIAO DICH THANH CONG!!!*************\n\n");
386         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");
387 
388         printf("Nhap ki tu bat ki de tiep tuc: ");
389         scanf("%s" , &TE);
390         system("cls");
391 
392         if(strcmp(TE , admin) == 0) {
393             printf("Nhap MA BAO MAT ATM: ");
394             scanf("%s" , &TE);
395             printf("\n");
396 
397             while(strcmp(TE , passWord) != 0) {
398                 printf("ERROR!!!  Vui long nhap lai Ma Bao Mat: ");
399                 scanf("%s" , &TE);
400             }
401 
402             is_active = 0;
403         }
404     }
```

```
405 
406     PRINT(BL , &total_transactions);
407 
408 }
```