**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN**

**ĐỀ TÀI : 701**

**XÂY DỰNG ỨNG DỤNG THỰC HIỆN**

**GIAO DỊCH TRÊN MÁY ATM**

Người hướng dẫn**: GVC.ThS: TRẦN HỒ THỦY TIÊN**

Sinh viên thực hiện:

**NGUYỄN TRẦN MỸ DUYÊN LỚP: 20T1 NHÓM: 20.10A**

**NGUYỄN PHƯỚC ĐẠI TOÀN LỚP: 20T1 NHÓM: 20.10A**

**Đà Nẵng, 08/2021**

**MỤC LỤC**

[**MỤC LỤC** 1](#_heading=h.gjdgxs)

[**DANH MỤC HÌNH VẼ** 2](#_heading=h.30j0zll)

[**LỜI MỞ ĐẦU** 3](#_heading=h.3znysh7)

[**I. TỔNG QUAN ĐỀ TÀI** 5](#_heading=h.2et92p0)

[**1. Khái niệm máy ATM và ứng dụng giao dịch ngân hàng** 5](#_heading=h.tyjcwt)

[**2. Các chức năng cơ bản** 5](#_heading=h.3dy6vkm)

[**3. Mô tả về đề tài** 5](#_heading=h.1t3h5sf)

[**4. Ứng dụng thực tế** 7](#_heading=h.4d34og8)

[**II. CƠ SỞ LÝ THUYẾT** 7](#_heading=h.2s8eyo1)

[**2.1. Ý tưởng** 7](#_heading=h.17dp8vu)

[**2.2. Cơ sở lý thuyết** 8](#_heading=h.3rdcrjn)

[**2.2.1. Các thư viện** 8](#_heading=h.26in1rg)

[**2.2.2. Kiểu cấu trúc** 10](#_heading=h.lnxbz9)

[**III. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN** 10](#_heading=h.35nkun2)

[**1. Phát biểu bài toán** 10](#_heading=h.1ksv4uv)

[**2. Cấu trúc dữ liệu** 11](#_heading=h.44sinio)

[**2.1. Danh sách đặc** 11](#_heading=h.2jxsxqh)

[**2.2. Dữ liệu về thông tin** 12](#_heading=h.z337ya)

[**2.3. Dữ liệu về giao dịch** 14](#_heading=h.3j2qqm3)

[**3. Các thuật toán** 14](#_heading=h.1y810tw)

[**3.1. Thuật toán hàm login\_account, login\_password** 14](#_heading=h.2xcytpi)

[**3.2. Thuật toán tìm kiếm nhị phân** 16](#_heading=h.4i7ojhp)

[**3.3. Thuật toán hàm formatMoney**](#_heading=h.1ci93xb) 17

[**3.3. Thuật toán hàm getMoney**](#_heading=h.1ci93xb) 20

[**IV. CHƯƠNG TRÌNH VÀ KẾT QUẢ** 19](#_heading=h.3whwml4)

[**1.Tổ chức chương trình** 19](#_heading=h.2bn6wsx)

[**2. Ngôn ngữ cài đặt** 22](#_heading=h.qsh70q)

[**3. Kết quả** 22](#_heading=h.3as4poj)

[**V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 26](#_heading=h.1pxezwc)

[**1. Kết luận** 26](#_heading=h.49x2ik5)

[**2.Hướng phát triển** 26](#_heading=h.2p2csry)

[**TÀI LIỆU THAM KHẢO** 27](#_heading=h.147n2zr)

[**PHỤ LỤC** 28](#_heading=h.3o7alnk)

**DANH MỤC HÌNH VẼ**

Hình 1. Mô tả khái quát ý tưởng tổ chức chương trình…………………………..7

Hình 2. Các thư viện được sử dụng………………………………………………8

Hình 3. Mô hình của danh sách đặc……………………………………………..11

Hình 4. Thông tin của mỗi khách hàng…………………………………………12

Hình 5. Dữ liệu về thông tin của khách hàng…………………………………...12

Hình 6. Dữ liệu thống kê giao dịch của khách hàng……………………………13

Hình 7. Cài đặt thuật toán hàm login\_account...………………………………..15

Hình 8. Cài đặt thuật toán hàm login\_password...………………………….…..15

Hình 9. Cài đặt thuật toán tìm kiếm nhị phân……………….………………….17

Hình 10. Cài đặt thuật toán hàm formatMoney……………………………..….18

Hình 11. Cài đặt thuật toán hàm getMoney………………………………….....29

Hình 12. Sơ đồ tổ chức chương trình…………………………………………...20

Hình 13. Màn hình khi nhân viên ngân hàng khởi động chương trình………....22

Hình 14. Màn hình khi khách hàng nhập tài khoản và mật khẩu……………….23

Hình 15.Thông báo khóa tài khoản khi nhập mật khẩu sai quá số lần quy định..24

Hình 16. Thông báo tài khoản đạt mức giao dịch tối đa trong ngày…………....24

Hình 17. Màn hình khi khách hàng thực hiện giao dịch………………………..25

**LỜI MỞ ĐẦU**

Trong những năm gần đây, sự phát triển không ngừng của Công nghệ thông tin nói chung và Internet nói riêng đã và đang mang lại những sự thay đổi đáng kể trong cuộc sống con người. Công nghệ tin học được ứng dụng trong nhiều lĩnh vực nghiên cứu khoa học, phát triển kinh tế, quân sự, nghệ thuật và các dịch vụ nhằm phục vụ lợi ích cho con người. Trong đó, đa phần các công việc về quản lý kinh tế, tài chính, ngân hàng đều có sự hỗ trợ đắc lực từ các phần mềm mang tính công nghệ, tối ưu và khoa học. Công việc quản lý dữ liệu và tạo môi trường giao dịch tiền tệ cũng nằm trong vấn đề này.

Với việc quản lý một lượng thông tin lớn về tài khoản, mật khẩu, tài sản, số lần giao dịch của người dùng là vô cùng quan trọng, yêu cầu tính logic và bảo mật cao. Việc này không chỉ đòi hỏi sự chính xác tuyệt đối về số liệu mà còn yêu cầu về tổ chức cơ sở dữ liệu sao cho hợp lý, tránh các trường hợp giả mạo, lừa đảo và đem lại trải nghiệm tốt, an toàn cho người sử dụng. Những yêu cầu kể trên đòi hỏi không chỉ dừng lại bằng việc quản lý bằng sổ sách, giấy tờ ghi chép mà cần phải có một hướng chuyên nghiệp, đơn giản, gọn nhẹ hơn.

Chính vì lý do này, nhóm chúng em đã chọn cho mình đề tài: “***Xây dựng ứng dụng thực hiện giao dịch trên máy ATM***” nhằm giải quyết vấn đề này và đưa ra các hướng phát triển trong tương lai.

Chúng em xin gửi lời cảm ơn chân thành đến cô Trần Hồ Thủy Tiên, người đã giúp đỡ và định hướng cho chúng em hoàn thành đề tài Đồ án cơ sở này. Trong quá trình hoàn thiện, cô đã giúp đỡ chúng em rất nhiều, cả về mặt kiến thức lẫn kỹ năng lập trình.

Mặc dù đã rất cố gắng hoàn thành đồ án nhưng chắc chắn chúng em sẽ không thể tránh khỏi những thiếu sót. Chúng em mong nhận được sự thông cảm, góp ý và chỉ bảo tận tình của quý thầy cô. Chúng em xin chân thành cảm ơn.

***Mục đích:*** Xây dựng một ứng dụng thực hiện giao dịch trên máy ATM, chạy bằng terminal với các chức năng cơ bản như sử dụng ứng dụng giao dịch thực sự: nạp tiền, rút tiền, xem tài khoản. Có giao diện thân thiện, dễ sử dụng, bảo đảm thông tin cho khách hàng.

***Mục tiêu:***

- Có giao diện thân thiện, bố cục rõ ràng.

- Quản lý cơ sở dữ liệu khách hàng logic, bảo mật.

- Thuật toán chính xác: xử lí cực kì chính xác số dư của mỗi khách hàng.

- Lập trình và triển khai chương trình rõ ràng

- Thống kê những giao dịch của khách hàng trong ngày chính xác, đầy đủ.

***Phạm vi và đối tượng nghiên cứu:***

Bao gồm các khách hàng đăng ký tài khoản để gửi tiền tại ngân hàng và nhân viên IT của ngân hàng để khởi động và kết thúc chương trình.

***Phương pháp nghiên cứu:***

Để thực hiện các công việc trên, chúng em đã sử dụng các phương pháp nghiên cứu:

- Phương pháp phân tích và tổng hợp lý thuyết.

- Phương pháp tổ chức và cấu trúc dữ liệu.

- Phương pháp nghiên cứu tài liệu.

- Phương pháp hỏi ý kiến của giáo viên hướng dẫn.

***Cấu trúc của đồ án môn học:***

Cấu trúc của đồ án gồm những chương sau:

**Chương 1: TỔNG QUAN ĐỀ TÀI**

**Chương 2: CƠ SỞ LÝ THUYẾT**

-Cơ sở lý luận cho việc xây dựng một hệ thống thông tin: Lý thuyết tổ chức cấu trúc dữ liệu và thuật toán.

**Chương 3: TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN**

**Chương 4: CHƯƠNG TRÌNH VÀ KẾT QUẢ**

**Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

**I. TỔNG QUAN ĐỀ TÀI**

**1. Khái niệm máy ATM và ứng dụng giao dịch ngân hàng**

ATM - Automated Teller Machine là thuật ngữ dùng để chỉ máy rút tiền tự động. Máy dùng để giao dịch và quản lý tài sản của khách hàng. Chức năng chính của máy ATM dùng để rút tiền. Ngày nay, ATM còn thực hiện được rất nhiều chức năng khác rất tiện lợi, nhằm giảm thiểu được số lượng công việc của nhân viên trong ngân hàng, giúp khách hàng linh hoạt hơn trong sự lựa chọn cũng như trải nghiệm sử dụng.

Ứng dụng giao dịch ATM là chương trình được lập trình dành cho nhân viên ngân hàng, có nhiệm vụ khởi động máy để chương trình được thực hiện. Khi ứng dụng được khởi động, khách hành có thể tương tác và quản lý tài sản, chi tiêu của mình thông qua ứng dụng này. Để sử dụng được ứng dụng ATM, bạn cần có thẻ ATM/tài khoản đăng nhập và mã PIN (mật khẩu đăng nhập) để xác nhận.

**2. Các chức năng cơ bản**

Ứng dụng giao dịch trên ATM có những chức năng sau:

- Nạp tiền: Khách hàng nhập vào màn hình máy số tiền mình muốn nạp vào, đồng thời đưa tiền cho nhân viên ngân hàng để xác nhận, sau khi đã xác nhận, tiền sẽ được cộng vào tài khoản ngân hàng của khách hàng.

- Rút tiền: Khách hàng nhập vào màn hình máy số tiền mình muốn rút và xác nhận. Nếu tài khoản ngân hàng của khách hàng lớn hơn số tiền muốn rút thì khách hàng sẽ nhận được số tiền yêu cầu và đồng thời số dư tài khoản ngân hàng của khách hàng bị trừ đi một lượng vừa rút. Còn nếu tài khoản ngân hàng không đủ số tiền muốn rút thì yêu cầu không hợp lệ.

- Truy vấn xem số dư tài khoản: màn hình sẽ hiển thị số dư hiện có của khách hàng

Bên cạnh đó, ứng dụng có một số chức năng khác như kiểm tra tài khoản mật khẩu, bảo mật mật khẩu,… sẽ được đề cập rõ hơn ở phần mô tả chi tiết.

**3. Mô tả về đề tài**

***Mô tả bao quát chung:***

- Mỗi ngày, ứng dụng giao dịch trên máy ATM sẽ được khởi động bởi nhân viên IT của ngân hàng. Tại thời điểm này, máy đọc danh sách các account number (số tài khoản), mã PIN (Personal Id Numbers), và số tiền hiện có (Account Balances) trong tài khoản của tất cả các khách hàng từ 1 file được lưu trữ.

- Sau khi máy bắt đầu, các khách hàng sử dụng ứng dụng giao dịch trên máy ATM để thực hiện các giao dịch gồm: *Nạp tiền, Rút tiền, Xem tài khoản.*

- Đối với mỗi khách hàng thực hiện giao dịch, ứng dụng ATM yêu cầu xác nhận số tài khoản và mã PIN, mỗi khách hàng chỉ được nhập mã PIN tối đa 3 lần, nếu sai sẽ bị khóa tài khoản và cần yêu cầu nhân viên IT mở khóa. Nếu đúng khách hàng có thể thực hiện giao dịch.

- Mỗi giao dịch thành công, ứng dụng ATM cập nhật lại số dư của mỗi khách hàng trong cơ sở dữ liệu. Đồng thời lưu lại danh sách giao dịch tại một file mới bao gồm tài khoản khách hàng nào đã giao dịch và biến động số dư như thế nào.

- Lưu ý ứng dụng ATM giới hạn số lần giao dịch của mỗi khách hàng trên một ngày và sẽ từ chối giao dịch nếu khách hàng giao dịch vượt quá số lần quy định (để đảm bảo tính an toàn cho nguồn tiền của khách hàng).

- Kết thúc một ngày, nhân viên IT của ngân hàng sẽ nhập vào mã bảo mật (được cấp quyền cho nhân viên) để kết thúc hoạt động của ứng dụng máy ATM và thực hiện ghi danh sách giao dịch vào file.

**a. Đối với người quản lý hệ thống**

Người quản lý hệ thống lúc này là nhân viên IT của ngân hàng. Chỉ nhân viên IT ngân hàng có mã để bắt đầu và kết thúc chương trình.

- Lúc bắt đầu, nhân viên IT nhập mã của mình để cập nhật dữ liệu và sau đó người dùng mới có thể sử dụng ATM

- Lúc kết thúc chương trình, dữ liệu lịch sử giao dịch sẽ được ghi vào file Giao dịch.

**b. Đối với người sử dụng ứng dụng**

Bước 1: Nhập tài khoản mật khẩu vào ứng dụng để xác thực.

Bước 2: Chọn các hình thức giao dịch trong menu.

Bước 3: Chọn thoát tài khoản khi không giao dịch nữa.

**Lưu ý:**

- Mật khẩu có 6 ký tự.

- Không nhập sai mật khẩu quá 3 lần, nếu sai quá giới hạn sẽ bị khóa tài khoản và phải yêu cầu nhân viên IT mở khóa.

- Mỗi ngày khách hàng bị hạn chế giao dịch không quá 5 lần.

**4. Ứng dụng thực tế**

Ứng dụng giao dịch được sử dụng rộng rãi trong khắp các hệ thống ngân hàng trên đất nước Việt Nam.

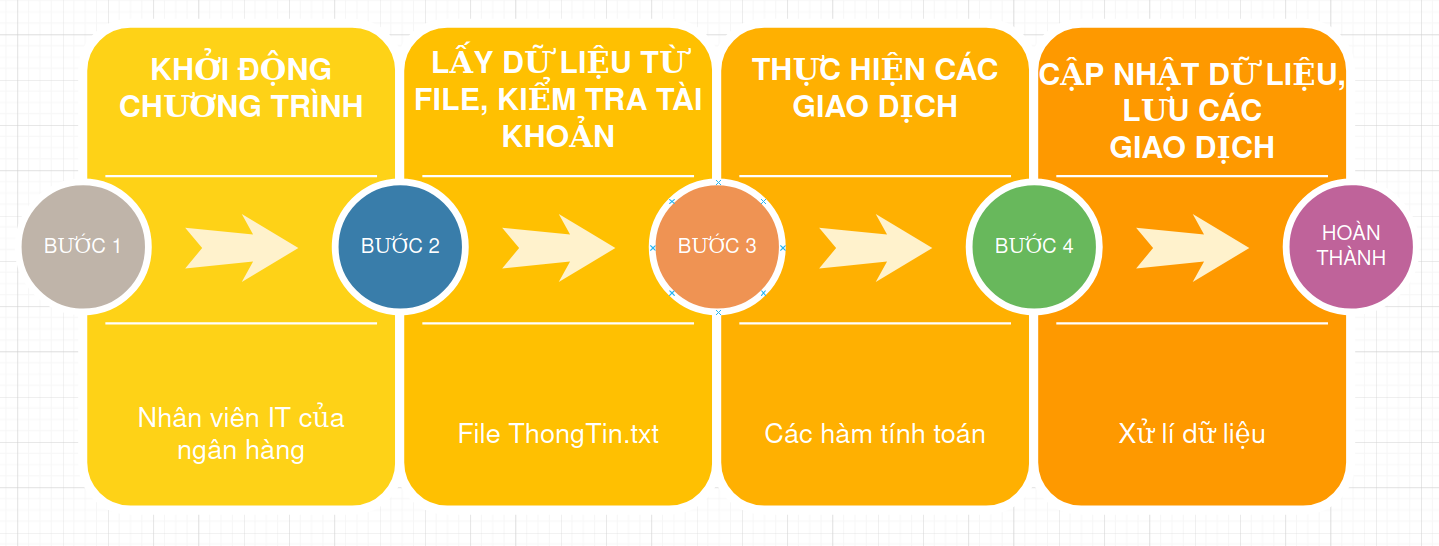
Chúng em đang xây dựng lại một ứng dụng nhỏ chạy bằng terminal với đầy đủ các chức năng cơ bản của một ứng dụng giao dịch của bất kỳ ngân hàng giao dịch và cây ATM nào.

Với việc sử dụng các thuật toán và lưu cơ sở dữ liệu, chúng em mong sẽ có thể áp dụng và định hướng trong tương lai sắp tới.

**II. CƠ SỞ LÝ THUYẾT**

**2.1. Ý tưởng**

Dựa vào mô tả đề tài xây dựng ứng dụng ATM, chúng em đã lên ý tưởng, khái quát hóa các bước cần thực hiện như sau:

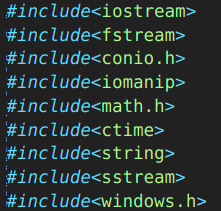


*Hình 1. Mô tả khái quát ý tưởng tổ chức chương trình*

**2.2. Cơ sở lý thuyết**

**2.2.1. Các thư viện**

Chúng em xây dựng bằng cách tiếp cận và ứng dụng các thư viện của ngôn ngữ lập trình.



*Hình 2. Các thư viện được sử dụng*

* ***Thư viện iostream:*** xây dựng chương trình biên dịch C++
* ***Thư viện fstream:*** sử dụng để đọc - ghi file

**Kỹ thuật thao tác với File:**

File được sử dụng để lưu trữ các dữ liệu cần thiết của chương trình, ngay cả khi chương trình không được khởi động thì nguồn dữ liệu của chúng ta không bị mất. Bên cạnh đó, sử dụng file dùng để lưu trữ dữ liệu đầu vào lớn, và sử dụng lại, cập nhật lại khi có yêu cầu chạy chương trình.

Các thao tác file với fstream: ***mở file -> đọc hoặc ghi dữ liệu -> đóng file***

* Phương thức để mở một file: ***f.open(filePath, mod)***;

Trong đó filePath có kiểu dữ liệu là const char\*, là đường dẫn đến file cần mở.

mod là chế độ mở. Có một số chế độ như sau:

*ios::in : Mở file để đọc*

*ios::out : Mở file có sẵn để ghi*

*ios::app : Mở và ghi dữ liệu vào cuối file, nếu file không tồn tại thì tạo file mới*

* Sau khi mở file, tiến hành đọc và ghi file bằng: ***operator >> và <<***
* Đóng file: ***f.close()***
* ***Thư viện conio.h:*** sử dụng hàm getch() để chuyển mật khẩu về \*\*\*\*\*\* nhằm mục đích bảo vệ thông tin của khách hàng khi nhập.
* ***Thư viện iomanip:*** sử dụng hàm setw() để thay đổi độ rộng khi lưu vào file.

-> Sử dụng để định dạng dữ liệu, giúp dữ liệu được lưu vào file dễ quản lý, quan sát.

* ***Thư viện math.h:*** đây là thư viện toán học của ngôn ngữ C/C++

-> Sử dụng để dùng hàm pow(), dùng trong hàm getMoney() (lấy dữ liệu số tiền từ file) để chuyển định dạng tiền từ string về kiểu long long dùng để tính toán.

* ***Thư viện ctime:*** sử dụng để lấy thời gian thực khi khách hàng thực hiện giao dịch tại ứng dụng.
* ***Thư viện string:*** dùng để định nghĩa biến password, money, withdraw, deposit thành kiểu string
* ***Thư viện sstream:*** sử dụng ostringstream (output string stream) là một class định nghĩa nguồn dữ liệu cho string. Đối tượng của class ostringstream này khi được tạo ra có thể truyền dữ liệu (các kiểu dữ liệu cơ bản, hoặc string) vào trong nó, để làm trung gian giữ dữ liệu đầu vào kiểu string.

-> Sử dụng để chuyển tiền dạng long long về string để định dạng tiền khi lưu vào file, khi xuất ra màn hình cho khách hàng, tạo sự trực quan.

* ***Thư viện window.h:*** đây là một header của Windows dành riêng cho ngôn ngữ lập trình C và C++. Trong đó chứa các khai báo cho tất cả các hàm (function) trong Windows API, tất cả các macro thường dùng bởi các lập trình viên Windows, và tất cả các kiểu dữ liệu (data type) sử dụng cho nhiều hàm và hệ thống con (subsystem).

-> Sử dụng để định dạng màu chữ.

**2.2.2. Kiểu cấu trúc**

Chúng em sử dụng kiểu cấu trúc (struct) để lưu các trường: số tài khoản (string), mật khẩu (string), số tiền (string), kiểm tra xem tài khoản có bị khóa (0/1), số giao dịch của khách hàng trong một ngày (int).

*struct UserAccount{*

*string account\_number;*

*string password\_number;*

*string money;*

*int lock;*

*int number\_of\_transactions;*

*};*

**III. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN**

**1. Phát biểu bài toán**

***Input:***

- File Thongtin.txt chứa dữ liệu của khách hàng bao gồm:

+ Số tài khoản

+ Mật khẩu

+ Số tiền hiện có trong tài khoản

+ Kiểm tra xem tài khoản có bị khóa không

+ Số lần giao dịch trong một ngày

- Tài khoản, mật khẩu mà khách hàng nhập vào

- Các sự lựa chọn giao dịch của khách hàng

***Output:***

- Kết quả giao dịch của khách hàng:

+ Thông báo nạp tiền / rút tiền có thành công hay không.

+ Số tiền hiện có của khách hàng.

+ Giao diện của ứng dụng ATM và menu chọn lựa.

…

- File Thongtin.txt chứa dữ liệu của khách hàng. Thông tin trong file thay đổi khi khách hàng giao dịch làm biến động số dư ban đầu.

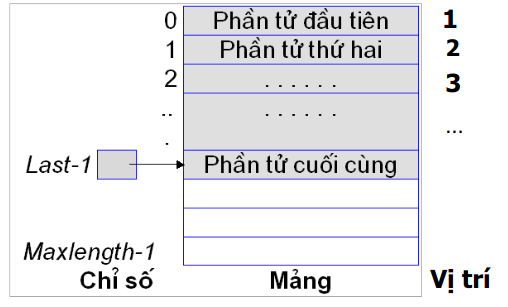
- File Giaodich.txt chứa dữ liệu các khách hàng đã giao dịch sau khi kết thúc chương trình.

**2. Cấu trúc dữ liệu**

Dữ liệu của khách hàng và xử lý dữ liệu này là mấu chốt quan trọng nhất trong đề tài của chúng em. Phải làm sao để tổ chức dữ liệu hợp lý, dễ truy xuất.

**2.1. Danh sách đặc**

* Danh sách là tập hợp hữu hạn các phần tử có cùng kiểu. Kiểu chung được gọi là kiểu phần tử (element type).
* Danh sách đặc là danh sách mà các phần tử được lưu trữ kế tiếp nhau trong bộ nhớ dưới hình thức một mảng.
* Mảng là một tập hợp tuần tự các phần tử có cùng kiểu dữ liệu và các phần tử được lưu trữ trong một dãy các ô nhớ liên tục trên bộ nhớ. Các phần tử của mảng được truy cập bằng cách sử dụng các chỉ số. Một mảng có kích thước n sẽ có chỉ số từ 0 tới n-1.
* Mô hình



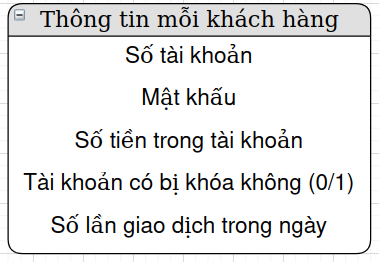
*Hình 3. Mô hình của danh sách đặc*

* Ta định nghĩa vị trí của một phần tử trong danh sách là “*chỉ số của mảng tại vị trí lưu trữ phần tử đó +1”.*

|  |  |
| --- | --- |
| Ưu điểm | Nhược điểm |
| - Truy xuất đến từng phần tử thông qua toán tử chỉ số => nhanh hơn và tiện lợi hơn.  - Ít tốn bộ nhớ. | - Phải biết trước số lượng phần tử.  - Việc thêm, xóa phần tử bản chất không phải thêm hoặc xóa thật sự mà là thay đổi vị trí các phần tử cho nhau.  - Các phần tử phải nằm liên tiếp nhau trong bộ nhớ máy tính => dễ phân mảnh bộ nhớ. |

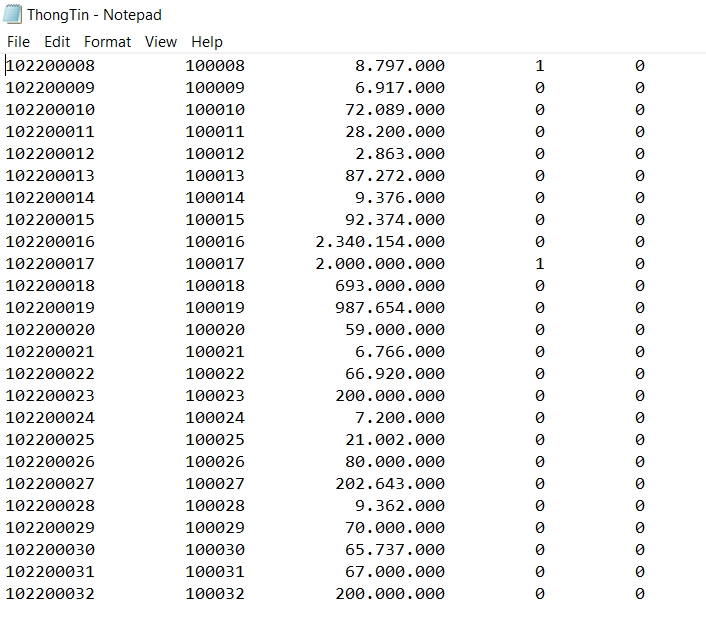
**2.2. Dữ liệu về thông tin**

Sau đây là dữ liệu thông tin của mỗi khách hàng cần lưu trữ:



*Hình 4. Thông tin của mỗi khách hàng*

Tất cả các dữ liệu về thông tin được chúng em lưu vào file Thongtin.txt được tổ chức như sau:



*Hình 5. Dữ liệu về thông tin của khách hàng*

**Trong đó:**

- Cột 1: Dùng để lưu số tài khoản của khách hàng, được khởi tạo theo mã số sinh viên của từng người (ví dụ). Được sắp xếp theo thứ tự tăng dần.

- Cột 2: Dùng để lưu mật khẩu (mã PIN) của khách hàng. Mật khẩu được ngân hàng được khởi tạo với giá trị ban đầu là 100xxx. Với xxx là 3 chữ số cuối của tài khoản ngân hàng. Khách hàng có thể đổi mật khẩu sau khi biết được tài khoản (Chức năng này sẽ được cập nhật sau.)

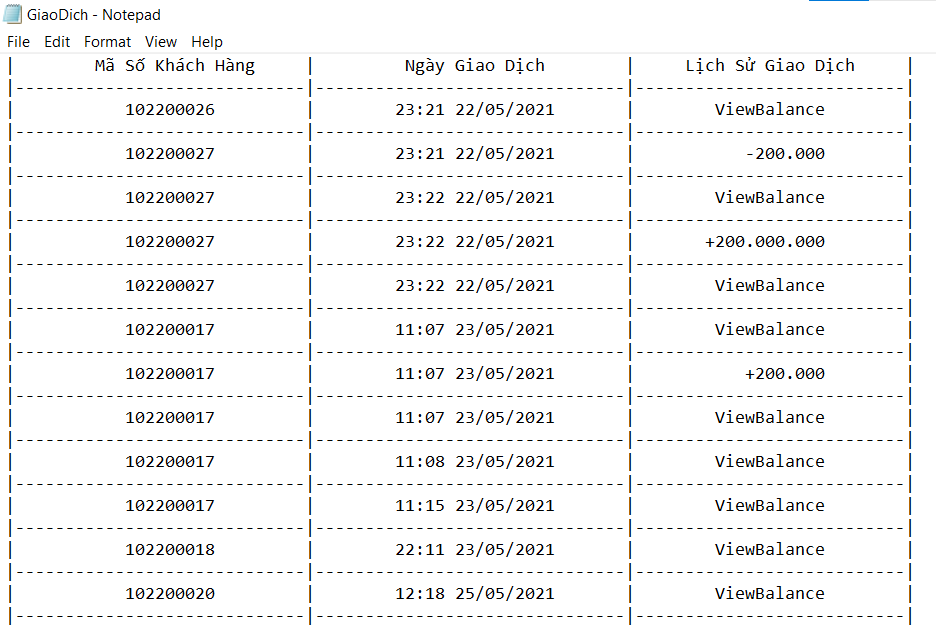
- Cột 3: Chứa số tiền của khách hàng. Số tiền được hiển thị theo dạng xxx.xxx.xxx để có thể dễ quản lý và giao dịch.

- Cột 4: Đây là cột để kiểm tra tài khoản ngân hàng của khách hàng có bị khóa hay chưa. Nếu chưa khóa có giá trị là 0. Nếu đã bị khóa có giá trị 1.

- Cột 5: Chứa số lần giao dịch của khách hàng trong một ngày. (Tối đa 5 lần).

**2.3. Dữ liệu về giao dịch**

Dữ liệu này được lưu vào file Giaodich.txt thể hiện các giao dịch của khách hàng trong ngày. Đây là hình ảnh về dữ liệu của file:



*Hình 6. Dữ liệu thống kê giao dịch của khách hàng*

**3. Các thuật toán**

**3.1. Thuật toán hàm login\_account, login\_password**

**a) Thuật toán hàm login\_account**

* **Mục đích của hàm login\_account:** tài khoản bắt buộc chứa 9 kí tự số, không nhận kí tự chữ.
* Tài khoản được mặc định có 9 ký tự số được lưu trong mảng char gồm 10 ký tự gồm 9 ký tự đầu là tài khoản người dùng, kí tự cuối chỉ nhận kí tự là ký tự delete hoặc enter.

**Thuật toán:**

Bước 1: i = 0 // Bắt đầu nhập ký tự đầu của tài khoản

Bước 2: Kiểm tra vị trí thứ i nhập ký tự vào:

Nếu i = 9:

* Nếu ký tự thứ 9 là enter thì đến bước 4
* Nếu ký tự thứ 9 là delete thì i = i - 2 đến bước 3 // xóa kí tự

Nếu i ≠ 9:

* Nếu ký tự nhập vào từ 0 đến 9 thì in ra kí tự.
* Nếu ký tự nhập vào là delete xảy ra hai trường hợp
* Nếu mảng kí tự đang rỗng thì i = i – 1;
* Nếu mảng ký tự không rỗng thì i = i - 2;
* Nếu nhập các ký tự khác thì i = i - 1;
* Đến bước 3

Bước 3: i = i + 1; // Nhập các kí tự tiếp theo

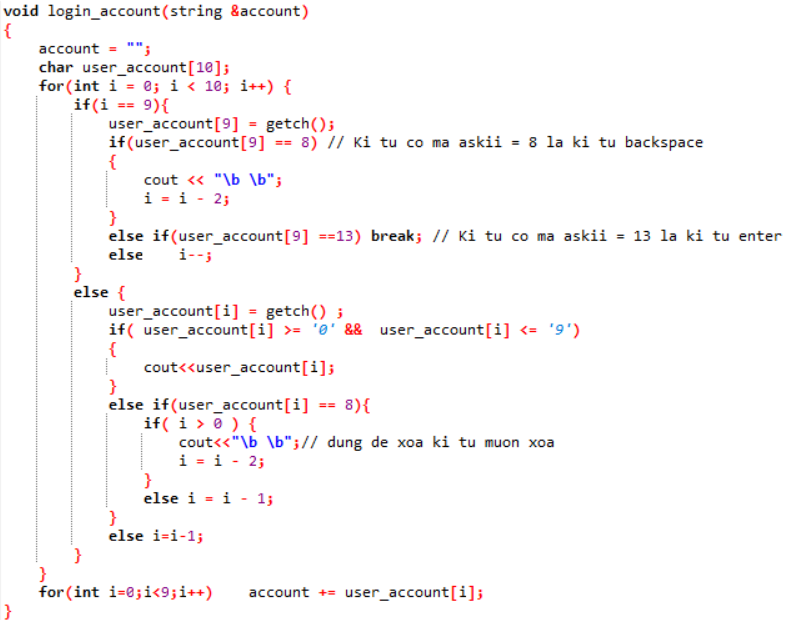
Quay lại bước 2

Bước 4: Thoát vòng lặp

**Đánh giá giải thuật:**

Bài toán sử dụng một vòng for với số lần lập hữu hạn với các điều kiện if - else lồng nhau với độ phức tạp O(1).

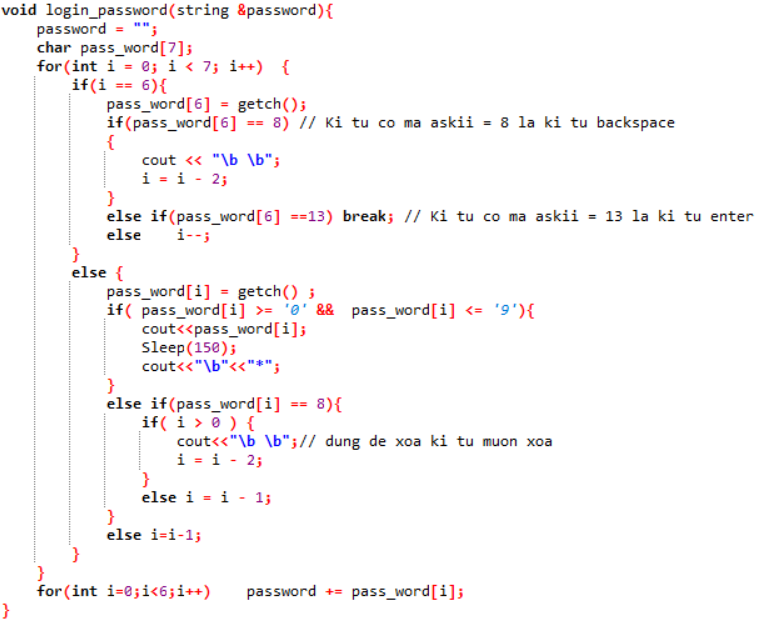
**Cài đặt chương trình:**



*Hình 7. Cài đặt thuật toán hàm login\_account*

**b) Thuật toán hàm login\_password**

Tương tự, như hàm login\_account



*Hình 8. Thuật toán login\_password*

**3.2. Thuật toán tìm kiếm nhị phân**

- Vì tài khoản người dùng đã được sắp xếp nên dùng thuật toán tìm kiếm nhị phân để giảm thiểu thời gian tìm kiếm

**- Giải thuật:** Mỗi bước tiến hành so sánh phần tử cần tìm là x so với phần tử nằm ở vị trí giữa của dãy tìm kiếm hiện hành, ta giả sử vị trí nằm giữa là mid, ta so sánh x với a[mid]

+ Khi x < a[mid]: có nghĩa là x chỉ có thể nằm bên nửa trái của mảng hiện hành

+ Khi x> a[mid]: có nghĩa là x chỉ có thể nằm bên nửa phải của mảng hiện hành

+ Khi x = a[mid] thì phần tử đã được tìm thấy

**Mô tả thuật toán:**

Bước 1:   left=1; right = n;

Bước 2:

+ mid = (left+right)/2;

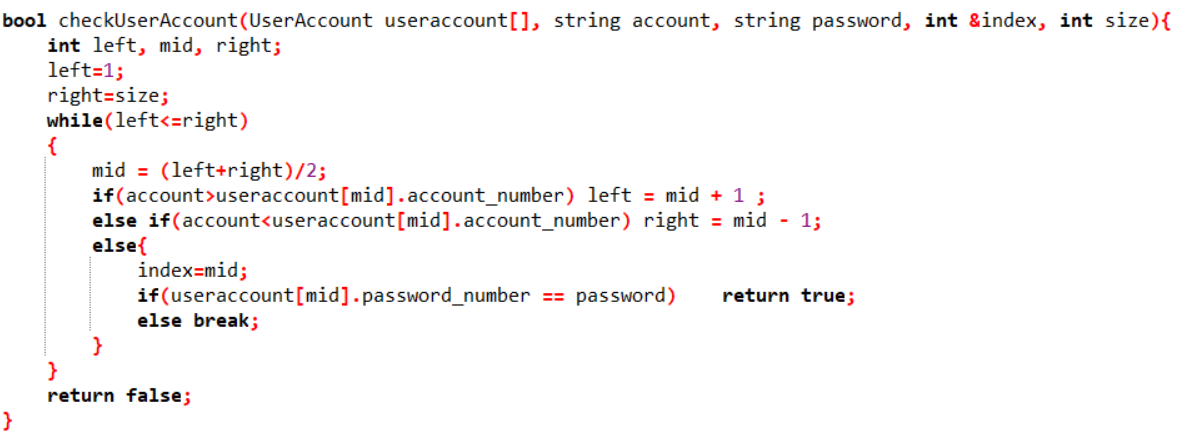
So sánh x với a[mid], có 3 khả năng:

* x < a[mid]: tìm kiếm x trong đoạn trước a[left] … a[mid-1]
  + right = mid – 1;
* x > a[mid]: tìm kiếm x trong đoạn sau a[mid+1] … a[right]
  + left = mid + 1;
* x = a[mid]: phần tử x đã được tìm thấy và dừng thuật toán.

Bước 3:

* Nếu left <= right: Quay lại bước 2
* Ngược lại: mảng đã được duyệt hết các phần tử có khả năng, không tìm thấy và trả về -1.

**Cài đặt: Được sử dụng trong hàm check\_UserAccount**



*Hình 9. Cài đặt thuật toán tìm kiếm nhị phân*

**Đánh giá giải thuật:**Ta có bảng phân tích sau:

|  |  |  |
| --- | --- | --- |
| Trường hợp | Số lần so sánh | Giải thích |
| Tốt nhất | 1 | Phần tử giữa mảng có giá trị x |
| Xấu nhất | log2n | Không có x trong mảng |
| Trung bình | (log2n)/2 | Giả sử xác suất các phần tử trong mảng nhận giá trị x là như nhau |

Vậy giải thuật tìm tuyến tính có độ phức tạp tính toán cấp n: T(n) = O(log2n)

**Nhận xét:**

* Giải thuật tìm kiếm nhị phân dựa vào quan hệ giá trị của các phần tử trong mảng để định hướng trong quá trình tìm kiếm, do vậy chỉ áp dụng được cho những dãy đã có thứ tự
* Giải thuật tìm kiếm nhị phân tiết kiệm thời gian hơn rất nhiều với độ phức tạp bằng O(log2n)
* Muốn áp dụng tìm kiếm nhị phân cần phải xét đến thời gian sắp xếp dãy số để thỏa mãn điều kiện có thứ tự. Thời gian này không nhỏ và khi dãy số biến động cần tiến hành sắp xếp lại. Tất cả các nhu cầu đó tạo ra khuyết điểm cho giải thuật tìm kiếm nhị phân. Do đó cần cân nhắc nhu cầu thực tế để áp dụng phù hợp.

**3.3. Thuật toán hàm formatMoney:**

* **Mục đích của hàm formatMoney:** dùng để chuyển tiền kiểu số nguyên thành kiểu string với chuỗi được định dạng x.xxx.xxx để dễ dàng quản lý cũng như trong lúc giao dịch hạn chế rủi ro khi giao dịch.

**Thuật toán:**

**//** Result là tiền đã được chuyển từ kiểu số nguyên về kiểu string

// length là độ dài chuỗi Result

Bước 1: i = length - 1; //Duyệt từ phần tử cuối về đầu

Bước 2:

* đếm ++; // ban đầu đếm = 0, khi đếm chia hết cho 3 thì thêm ‘.’ vào trước để định dạng.
* Nếu i = 0 thì s = Result[i] + s;

Ngược lại nếu đếm chia hết cho 3 thì s = Result[i] + s, sau đó s = ‘.’ + s;

Ngược lại thì s = Result[i] + s;

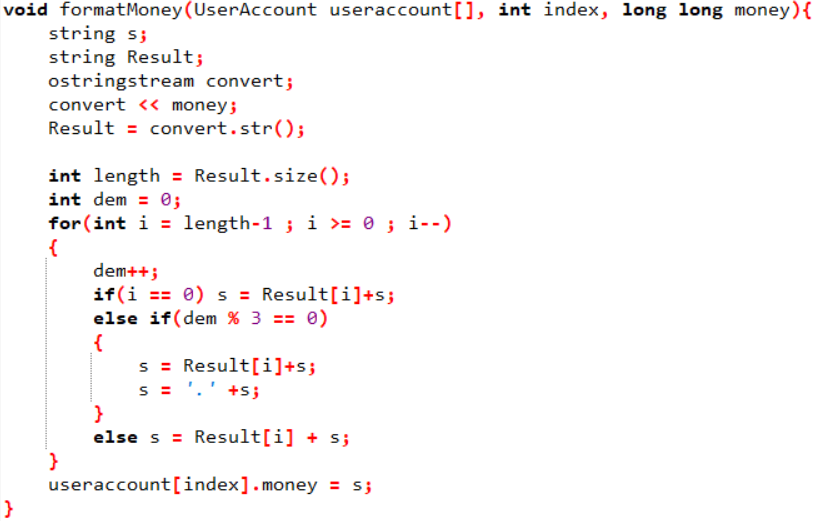
Bước 3: i--;

Nếu i ≥ 0 thì quay lại bước 2, ngược lại thì kết thúc vòng lặp

**Đánh giá giải thuật:**

Bài toán sử dụng một vòng for cùng với các điều kiện if - else lồng nhau với độ phức tạp O(1).

**Cài đặt chương trình:**

****

*Hình 10. Cài đặt thuật toán hàm formatMoney*

**3.4. Thuật toán hàm getMoney:**

* **Mục đích của hàm getMoney:** dùng để chuyển tiền được lưu định dạng chuỗi xxx.xxx.xxx thành kiểu số nguyên để giao dịch nạp tiền và rút tiền

**Thuật toán:**

Bước 1: i = 0 // bắt đầu duyệt từ phần tử đầu chuỗi

Bước 2: Kiểm tra

* Nếu money[i] ≠ ‘.’ thì tiền = tiền + (money[i] - ‘0’).10^(length-1-i);

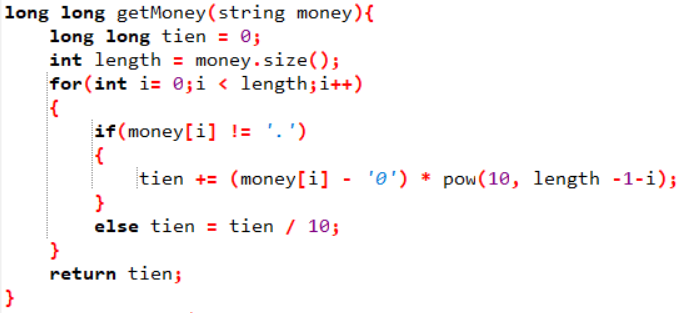
// length là độ dài chuỗi money

* Ngược lại thì tiền = tiền / 10;

Bước 3: i++;

Nếu i < length thì quay lại bước 2, ngược lại trả về tiền

**Cài đặt chương trình:**

****

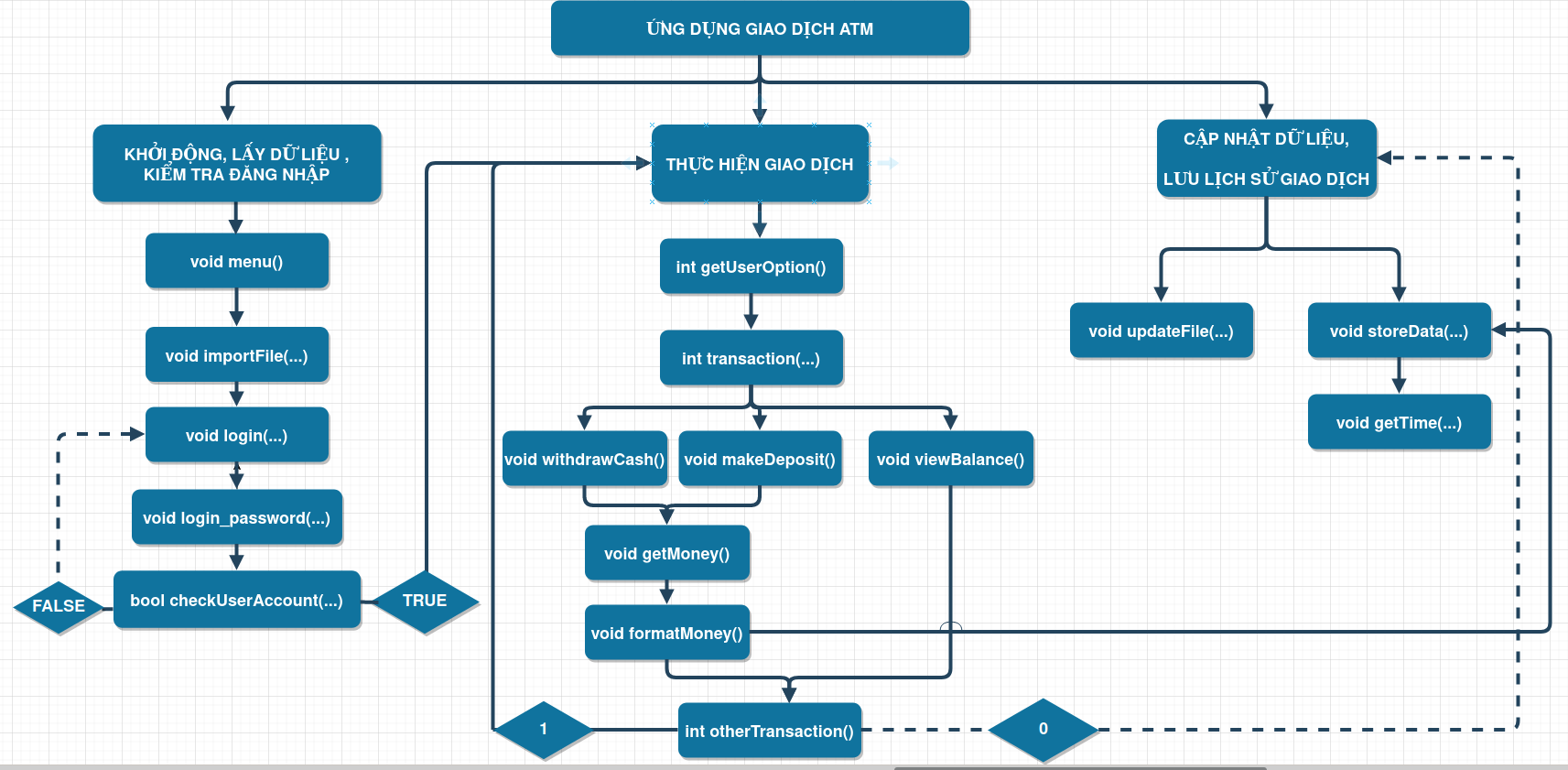
*Hình 11. Cài đặt thuật toán hàm getMoney*

**Đánh giá giải thuật:**

Bài toán sử dụng một vòng for cùng với các điều kiện if - else lồng nhau với độ phức tạp O(1).

**IV. CHƯƠNG TRÌNH VÀ KẾT QUẢ**

**1.Tổ chức chương trình**



*Hình 12. Sơ đồ tổ chức chương trình*

Chúng em chia chương trình thành 3 module lớn đó chính là:

* Khởi động, lấy dữ liệu từ file, kiểm tra tài khoản, mật khẩu và đăng nhập.
* Thực hiện các giao dịch.
* Cập nhật dữ liệu, thống kê.

Chi tiết về từng module như sau:

**Khởi động, lấy dữ liệu, kiểm tra đăng nhập:**

* *Tạo struct:*

Để lưu dữ liệu của UserAccount (Tài khoản của khách hàng), tài khoản để khởi động chương trình của nhân viên ngân hàng.

* *Hàm void menu():*

Tạo giao diện ban đầu của ứng dụng.

* *Hàm bool importFile(...):*

Lấy dữ liệu từ file ThongTin.txt chứa mảng các phần tử, mỗi phần tử là struct chứa dữ liệu khách hàng. Nếu file dữ liệu bị lỗi thì kết thúc chương trình.

* *Hàm void login(...):*

Hàm đăng nhập để xác nhận nhân viên IT ngân hàng khởi động chương trình, xác nhận tài khoản của khách hàng.

* *Hàm void login\_account (string & account)*:

Hàm nhận tài khoản nhập vào, chỉ nhận kí tự số, nếu là ký tự chữ thì sẽ không nhập được. Xóa ký tự nếu khách hàng nhập vào nhiều hơn 9 ký tự.

* *Hàm void login\_password (string & password)*:

Hàm nhận password nhập vào, kiểm tra điều kiện, xóa ký tự nếu nhiều hơn 6 ký tự, và chuyển các ký tự về dạng \*\*\*\*\*\* để đảm bảo bảo mật và thông tin của khách hàng.

* *Hàm bool checkUserAccount (...):*

Hàm kiểm tra tài khoản nhập vào có chính xác hay không.

**Thực hiện các giao dịch:**

* *Hàm int getUserOption():*

Hàm lấy lựa chọn của khách hàng khi đã in ra menu các giao dịch.

* *Hàm int transaction(...):*

Hàm giao dịch dùng để lấy thông tin khi khách hàng chọn loại giao dịch.

* *Hàm void withdrawCash(...):*

Hàm rút tiền.

* *Hàm void makeDeposit(...):*

Hàm nạp tiền vào tài khoản.

* *Hàm viewBalance(..):*

Hàm xem số tiền hiện có trong tài khoản.

* *Hàm long long getMoney(...):*

Chuyển tiền đang lưu ở dạng string về dạng long long để tính toán.

* *Hàm void formatMoney(...):*

Định dạng lại tiền từ dạng số nguyên về kiểu string, có dấu phân cách giữa các đơn vị.

* *Hàm int otherTransaction(..):*

Kiểm tra xem khách hàng có giao tiếp nữa hay không.

**Cập nhật dữ liệu, lưu lịch sử giao dịch:**

* *Hàm void updateFile(...):*

Cập nhật lại dữ liệu trong file ThongTin.txt chứa dữ liệu về tài khoản, mật khẩu, số lần giao dịch và số tiền của khách hàng.

* *Hàm void storeData(...):*

Để lưu dữ liệu các giao dịch đã được thực hiện trước khi kết thúc chương trình. Lưu vào file GiaoDich.txt để thống kê và quản lý.

* *Hàm getTime(...):*

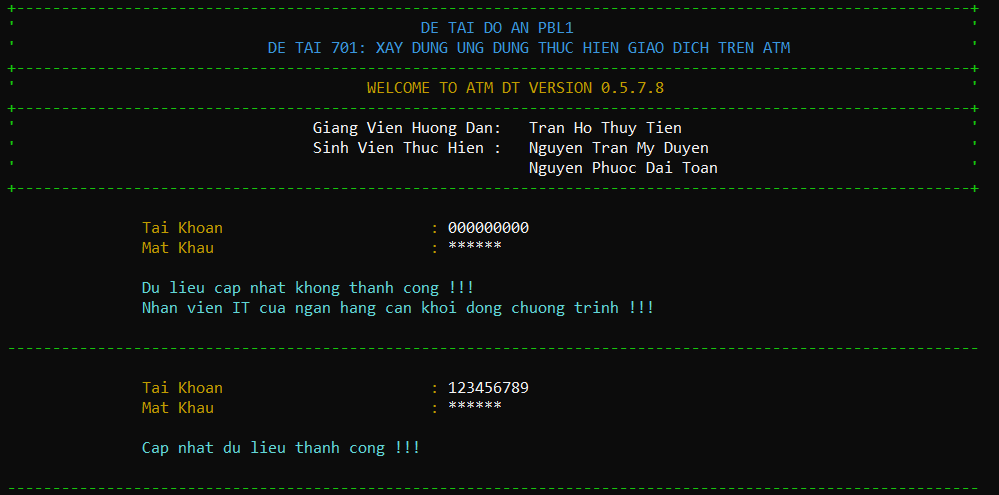
Để lấy thời gian thực, thống kê số lần giao dịch trong ngày của khách hàng, nhằm hạn chế số lần giao dịch.

**2. Ngôn ngữ cài đặt**

Chúng em sử dụng ngôn ngữ C++ để cài đặt chương trình.

**3. Kết quả**

***Màn hình khi nhân viên ngân hàng khởi động chương trình:***

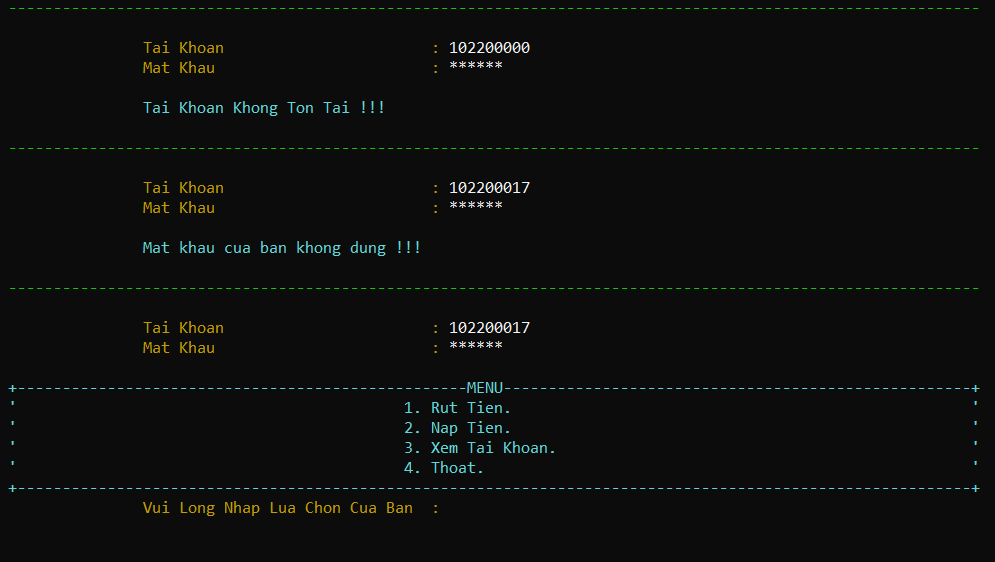


*Hình 13. Màn hình khi nhân viên ngân hàng khởi động chương trình*

Nhân viên IT của ngân hàng được cấp tài khoản ban đầu là: 123456789, mật khẩu 999999.

* Nếu không phải nhân viên ngân hàng mở chương trình, thì hệ thống yêu cầu nhập lại.
* Nếu hệ thống xác nhận được đúng tài khoản của ngân viên ngân hàng thì chương trình mới cho phép người dùng đăng nhập.

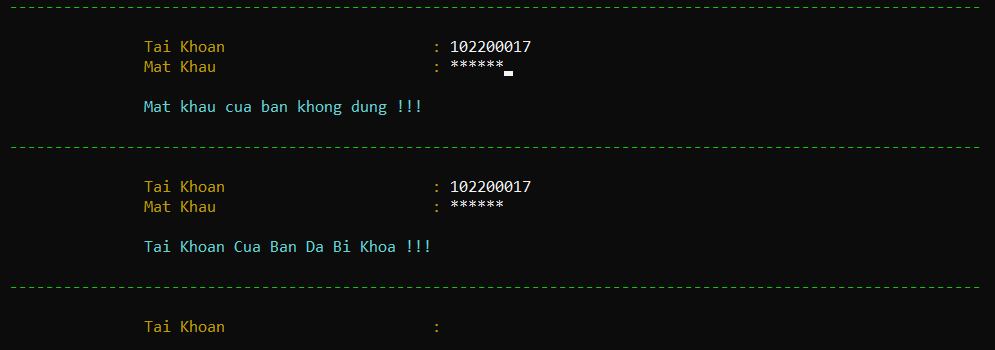
***Màn hình khi khách hàng đăng nhập:***

******

*Hình 14. Màn hình khi khách hàng nhập tài khoản và mật khẩu*

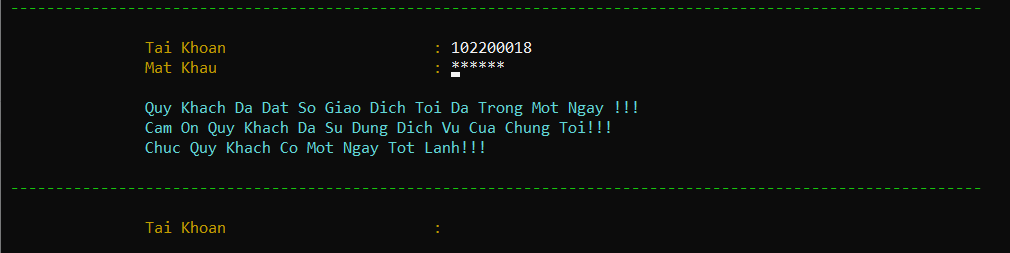
Khi khách hàng nhập tài khoản:

* Tài khoản ngân hàng được tạo bởi: Mã số sinh viên của mỗi người (giả sử) và mật khẩu cố định gồm 6 chữ số: 100xxx với xxx là 3 kí tự cuối trong mã số sinh viên.
* Nếu tài khoản chưa được đăng ký tại ngân hàng (không tồn tại) thì màn hình sẽ hiện thông báo “Tai Khoan Khong Ton Tai".
* Nếu tài khoản đúng nhưng mật khẩu sai, màn hình sẽ hiện thông báo “Mat khau cua ban khong dung !!!”
* Ở những trường hợp khác như nhập mật khẩu quá số lần quy định (3 lần), hay tài khoản đó đã đạt định mức giao dịch tối đa trong ngày, màn hình sẽ hiện những thông báo sau, và chuyển qua nhập tài khoản mới:



*Hình 15. Thông báo khóa tài khoản khi nhập mật khẩu sai quá số lần quy định*

* Tài khoản khách hàng bị khóa và cần phải tới địa điểm ngân hàng để yêu cầu mở lại tài khoản (set lock = 0).

**

*Hình 16. Thông báo tài khoản đạt mức giao dịch tối đa trong ngày*

* Khi khách hàng nhập đúng tài khoản, mật khẩu thỏa mãn tài khoản không bị quá và không vượt quá số lần giao dịch trong ngày, màn hình sẽ hiện ra menu để lựa chọn các thao tác giao dịch.

**

*Hình 17. Màn hình khi khách hàng thực hiện giao dịch*

* Sau khi thực hiện xong một giao dịch, chương trình luôn hỏi khách hàng có muốn giao dịch tiếp không và không quên cảm ơn quý khách đã sử dụng dịch vụ của ngân hàng.

**V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

**1. Kết luận**

* Sau thời gian thực hiện đồ án cơ sở PBL1, chúng em đã đạt được nhiều tiến bộ cả về mặt tìm hiểu, nghiên cứu lý thuyết, lẫn kỹ năng lập trình.
* Hiểu biết nhiều hơn về các kỹ thuật và kỹ xảo trong lập trình C++.
* Hiểu hơn về xử lý dữ liệu.
* Tạo được một ứng dụng ATM chạy bằng terminal, có giao diện thân thiện (định dạng số tiền, mật khẩu nhập vào..) và trực quan, hoạt động đúng theo yêu cầu và ý tưởng của đề tài.
* Xử lý được một số vấn đề cần thiết của người dùng khi sử dụng ứng dụng ATM.

**2.Hướng phát triển**

* Xây dựng một ứng dụng với các chức năng mở rộng hơn: đổi mật khẩu, liên kết với các nguồn tiền từ ngân hàng khác.
* Có thể ứng dụng để quản lý số tiền của sinh viên trong tương lai.
* Xử lý một số lượng dữ liệu, khách hàng nhiều hơn, truy cập tối ưu hơn.

**TÀI LIỆU THAM KHẢO**

[1] Steps to developing banking software

[*https://www.quora.com/What-are-the-first-steps-to-developing-banking-software*](https://www.quora.com/What-are-the-first-steps-to-developing-banking-software)

[2] Tham khảo ứng dụng điện thoại về banking

[*https://www.scnsoft.com/banking/mobile-app-development*](https://www.scnsoft.com/banking/mobile-app-development)

[3] Tham khảo cách tổ chức dữ liệu

[*https://0xboz.github.io/blog/build-simple-banking-system-cpp/*](https://0xboz.github.io/blog/build-simple-banking-system-cpp/)

[4] Danh sách đặc

[*https://vnseeder.blogspot.com/2014/04/danh-sach-ac.html*](https://vnseeder.blogspot.com/2014/04/danh-sach-ac.html)

[5] Kiểu cấu trúc

[*https://nguyenvanhieu.vn/struct-trong-c/*](https://nguyenvanhieu.vn/struct-trong-c/)

[6] Thư viện chuẩn C/C++

[*https://vi.wikipedia.org/wiki/Th%C6%B0\_vi%E1%BB%87n\_chu%E1%BA%A9n\_C%2B%2B*](https://vi.wikipedia.org/wiki/Th%C6%B0_vi%E1%BB%87n_chu%E1%BA%A9n_C%2B%2B)

[*https://vietjack.com/thu-vien-c/math-h-trong-c.jsp*](https://vietjack.com/thu-vien-c/math-h-trong-c.jsp)

**PHỤ LỤC**

1. **Hàm login(...)**

**int login** (**string** &account, **string** & password, **int** index)

{

set\_color(yellow);

cout<<"\n\t\t Tai Khoan : ";

set\_color(white);

login\_account(account)

set\_color(yellow);

cout<<"\t\t Mat Khau : ";

set\_color(white);

login\_password(password);

**if** (account == account\_staff && password == password\_staff) **return** 1;

set\_color (white);

**return** 0;

}

1. **Hàm login\_account(...)**

**void** **login\_account**(**string** &account){

account = "";

**char** user\_account[10];

**for**(int i = 0; i < 10; i++){

**if**(i == 9){

user\_account[9] = getch();

**if**(user\_account[9] == 8{

cout << "\b \b";

i = i - 2;

}

**else if** (user\_account[9] ==13) break;

**else** i--;

}

**else** {

user\_account[i] = getch() ;

**if**( user\_account[i] >= '0' && user\_account[i] <= '9')

cout<<user\_account[i];

**else if** (user\_account[i] == 8){

**if**( i > 0 ){

cout<<"\b \b";

i = i - 2;

}

**else** i = i - 1;

}

**else** i=i-1;

}

}

**for**(**int** i=0;i<9;i++) account += user\_account[i];

}

1. **Hàm login\_password(...)**

**void login\_password** (**string** & password)

{

password = "";

**char** pass\_word [7];

**for** (**int** i = 0; i < 7; i++){

**if** (i == 6){

pass\_word [6] = getch ();

**if** (pass\_word [6] == 8){

cout << "\b \b";

i = i - 2;

}

**else if** (pass\_word [6] ==13) **break;**

**else** i--;

}

**else** {

pass\_word[i] = getch ();

**if**(pass\_word[i] >= '0' && pass\_word[i] <= '9')

{

cout<<pass\_word[i];

Sleep (150);

cout<<"\b"<<"\*";

}

**else if**(pass\_word[i] == 8){

**if** (i > 0){

cout<<"\b \b";

i = i - 2;

}

**else** i = i - 1;

}

**else** i=i-1;

}

}

**for** (**int** i=0; i<6; i++) password += pass\_word[i];

}

1. **Hàm checkUserAccount ()**

**bool** **checkUserAccount** (UserAccount useraccount [], **string** account, **string** password, **int** &index, **int** size)

{

**int** left, mid, right;

left=1;

right=size;

**while**(left<=right)

{

mid = (left+right)/2;

**if**(account>useraccount[mid]. account\_number) left = mid + 1;

**else if**(account<useraccount[mid]. account\_number) right = mid - 1;

**else**{

index=mid;

**if**(useraccount[mid]. password\_number == password) **return** true;

**else** break;

}

}

**return** false;

}

1. **Hàm getUserOption()**

**int getUserOption** (){

**string** option;

cout<<endl;

set\_color(blue);

cout<<endl;

cout<<" +---------------------------MENU---------------------------+ "<<endl;

cout<<" ' 1. Rut Tien. ' "<<endl;

cout<<" ' 2. Nap Tien. ' "<<endl;

cout<<" ' 3. Xem Tai Khoan. ' "<<endl;

cout<<" ' 4. Thoat. ' "<<endl;

cout<<" +---------------------------------------------------------------+ "<<endl;

set\_color(yellow);

cout<<"\t\t Vui Long Nhap Lua Chon Cua Ban : ";

set\_color(white);

getline(cin,option);

**int** length = option.size();

**while** (length!=1 || option[0]<'1' || option[0]>'4'){

set\_color(blue);

cout<<"\n\t\tLua Chon Cua Ban Khong Hop Le !!!\n\n";

set\_color(yellow);

cout<<"\t\tVui Long Nhap Lai Lua Chon Cua Ban : ";

set\_color(white);

getline(cin,option);

length = option.size();

}

**return** option[0]-'0';}

1. **Hàm transaction(...)**

**int transaction** (UserAccount useraccount[], **int** index, **size** index, **int** option){

**switch** (option){

**case 1**:{

**string** withdraw;

**withdrawCash** (useraccount, index, size, withdraw);

**storeData** (useraccount, option, index, withdraw);

**return** 1;

}

**case 2**:{

**string** deposit;

**makeDeposit** (useraccount, index, size, deposit);

**storeData** (useraccount, option, index, deposit);

**return** 1;

}

**case 3**:{

**viewBalance**(useraccount[index].money);

**storeData** (useraccount, option, index, "0");

**return** 1;

}

**case 4**: {

set\_color(blue);

cout<<"\n\t\t Cam On Quy Khach Da Su Dung Dich Vu Cua

Chung Toi !!!";

cout<<"\n\t\t Chuc Quy Khach Co Mot Ngay Tot Lanh !!!";

cout<<endl;

set\_color(green);

cout<<"\n ------------------------------------------------"<<endl;

**return** 0;

}

}

}

1. **Hàm withdrawCash (...)**

**void withdrawCash** (UserAccount useraccount [], **int** index, **int** size, **string** & withdraw){

set\_color(blue);

cout<<"\n\t\tNhap So Tien Ban Muon Rut : ";

set\_color(white);

getline(cin,withdraw);

**long long** money = **getMoney**(useraccount[index].money);

**long long** money\_transactions = **getMoney**(withdraw);

**if** (money\_transactions > money) {

set\_color(blue);

cout<<"\n\t\tSo Tien Cua Ban Khong Du De Thuc Hien Giao Dich Nay

!!!";

}

**else** {

set\_color(blue);

cout<<"\n\t\tGiao Dich Thanh Cong !!!";

money = money - money\_transactions;

}

**formatMoney** (useraccount, index, money);

**updateFile** (useraccount, size);

}

1. **Hàm makeDeposit (...)**

**void makeDeposit** (UserAccount useraccount[], **int** index, **int** size, **string** & deposit){

set\_color(blue);

cout<<"\n\t\tNhap So Tien Ban Muon Nap : ";

set\_color(white );

getline(cin,deposit);

set\_color(blue);

cout<<"\n\t\tGiao Dich Thanh Cong !!!";

**long long** money = getMoney(useraccount[index].money);

**long long** money\_transactions = getMoney(deposit);

money = money + money\_transactions;

**formatMoney** (useraccount, index, money);

**updateFile**(useraccount,size);

}

1. **Hàm viewBalance (...)**

**void viewBalance** (**string** money){

set\_color(blue);

cout<<"\n\t\t So Tien Hien Tai Cua Ban : ";

set\_color(white);

cout<<money;

}

1. **Hàm otherTransaction ()**

**int otherTransaction** (){

**string** option;

set\_color(yellow);

cout<<"\n\n\t\t Ban Co Muon Giao Dich Tiep Khong?";

set\_color (white);

cout<<"\n\t\t 1. Co";

cout<<"\n\t\t 2. Khong";

set\_color(yellow);

cout<<"\n\t\t Vui Long Nhap Lua Chon Cua Ban : ";

set\_color(white);

getline(cin,option);

**int** length = option.size();

**while** (length!=1 || option[0]<'1' || option[0]>'2'){

set\_color(blue);

cout<<"\n\t\tLua Chon Cua Ban Khong Hop Le !!!";

set\_color(yellow);

cout<<"\n\t\tVui Long Nhap Lai Lua Chon Cua Ban : ";

set\_color(white);

getline(cin,option);

length = option.size();

}

**if**(option[0]=='1') return 1;

**else** {

set\_color(blue);

cout<<"\n\t\tCam On Quy Khach Da Su Dung Dich Vu Cua Chung Toi!!!";

cout<<"\n\t\tChuc Quy Khach Co Mot Ngay Tot Lanh!!!"<<endl;

set\_color(green);

cout<<"\n ------------------------------------------------------------------------------------------------------------"<<endl;

}

**return** 0;

}

1. **Hàm getMoney()**

**long long** **getMoney**(**string** money){

**long long** tien = 0;

**int length** = money.size();

**for**(**int** i= 0;i < length;i++){

**if**(money[i] != '.') tien += (money[i] - '0') \* pow(10, length -1-i);

**else** tien = tien / 10;

}

**return** tien;

}

**11. Hàm format Money(...)**

**void formatMoney**(UserAccount useraccount[], **int** index, **long long** money){

**string** s;

**string** Result;

ostringstream convert;

convert << money;

Result = convert.str();

**int** length = Result.size();

**int** dem = 0;

**for**(**int** i = length-1 ; i >= 0 ; i--) {

dem++;

**if**(i == 0) s = Result[i]+s;

**else if**(dem % 3 == 0){

s = Result[i]+s;

s = '.' +s;

}

**else** s = Result[i] + s;

}

useraccount[index].money = s;

}