**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN**

**DANH BẠ ĐIỆN THOẠI**

Người hướng dẫn**: TS. TRƯƠNG NGỌC CHÂU**

Sinh viên thực hiện**:**

**Nguyễn Thị Toàn LỚP: 22T\_DT1**

**Nguyễn Thị Trà My LỚP: 22T\_DT2**

**Huỳnh Phước Nguyên LỚP: 22T\_DT2**

**Đà Nẵng, 06/2023**

***LỜI CẢM ƠN***

Trong thời gian nghiên cứu và học tập, cùng với việc làm đồ án bộ môn này chúng em đã nhận được sự giúp đỡ quý báu của thầy/cô giáo đã tận tình giảng dạy chúng em trong năm học đầu tiên tại giảng đường Đại Học của chúng em , nhóm chúng em đã hoàn thành chương trình và bài báo cáo đồ án lập trình tính toán với đề tài A02: Quản lý danh bạ điện thoại

Cho phép chúng em được bày tỏ lời cảm ơn sâu sắc đến thầy giáo Trương Ngọc Châu trong Khoa Công Nghệ Thông Tin Trường Đại Học Bách Khoa Đại học Đà Nẵng đã hỗ trợ tài liệu, giúp đỡ trong quá trình lựa chọn đề tài và hướng dẫn các cách giải quyết vấn đề, tạo điều kiện tốt nhất chúng em hoàn thành chương trình và bài báo cáo này.

Tuy nhiên do thời gian có hạn, cũng như chưa có đầy đủ kinh nghiệm nên nội dung trong bài báo cáo này sẽ không tránh khỏi những thiếu sót trong quá trình tìm hiểu , đánh giá và trình bày về đề tài . Chúng em rất mong nhận được nhận xét, góp ý của thầy cô và các bạn để kiến thức của em trong lĩnh vực này được hoàn thiện hơn đồng thời có điều kiện bổ sung, nâng cao ý thức của mình.

Mọi sao chép không hợp lệ, vi phạm quy chế đào tạo, hay gian trá, nhóm em xin chịu hoàn toàn trách nhiệm.

Em xin chân thành cảm ơn!

***MỤC LỤC***

[MỤC LỤC](#_heading=h.gjdgxs)  3

[DANH MỤC HÌNH VẼ](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.30j0zll)  4

[MỞ ĐẦU](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.1fob9te) 5

[1. TỔNG QUAN ĐỀ TÀI](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.3znysh7) 6

[2. CƠ SỞ LÝ THUYẾT](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.2et92p0) 8

[2.1. Ý tưởng](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.tyjcwt) 8

[2.2. Cơ sở lý thuyết](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.3dy6vkm) 9

[3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 1](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.1t3h5sf)0

[3.1. Phát biểu bài toán 1](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.4d34og8)0

[3.2. Cấu trúc dữ liệu 1](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.2s8eyo1)1

[3.3. Thuật toán 1](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.17dp8vu)2

[4.](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.3rdcrjn) CHƯƠNG TRÌNH VÀ KẾT QUẢ 23

[4.1.](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.26in1rg) Tổ chức chương trình 23

[4.2.](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.lnxbz9) Ngôn ngữ cài đặt 24

[4.3.](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.35nkun2) Kết quả 24

[4.3.1.](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.1ksv4uv) Giao diện chính của chương trình 24

[4.3.2.](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.44sinio) Kết quả thực thi của chương trình 25

[5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.z337ya)  30

[5.1. Kết luận 30](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.3j2qqm3)

[5.2. Hướng phát triển 31](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.1y810tw)

[TÀI LIỆU THAM KHẢO 32](https://docs.google.com/document/d/1RNwh7-BK9UOnaNzWaaxkeIK3M2699Thj/edit#heading=h.4i7ojhp)

***DANH MỤC HÌNH VẼ***

Hình 1: Sơ đồ khối hàm tìm các liên hệ trùng số điện thoại trong 1 tỉnh ...............13

Hình 2: Sơ đồ khối hàm liệt kê các liên hệ trong tỉnh....…..…………….………..14

Hình 3: Sơ đồ khối hàm tìm liên hệ theo số điện thoại...…..…….……....…...…..15

Hình 4: Sơ đồ khối hàm tìm liên hệ theo địa chỉ............…..…….…...…………...16

Hình 5: Sơ đồ khối hàm xoá liên hệ theo số điện thoại, địa chỉ hoặc tên.........…..20

Hình 6: Sơ đồ khối hàm thêm liên hệ vào danh sách..............................................21

Hình 7: Giao diện chính của chương trình………………………………….…….24

Hình 8: Thực hiện đọc danh sách từ file.............…………………………..……...25

Hình 9: Tìm 2 liên hệ trùng và xoá nếu muốn........……………………..….……..26

Hình 10: Thực hiện liệt kê các liên hệ trong một tỉnh....……………...........……..27

Hình 11: Thực hiện tìm liên hệ thông qua số điện thoại..................................…...27

Hình 12: Thực hiện tìm liên hệ thông qua địa chỉ...................................................28

Hình 13: Thực hiện in danh sách ra file .................................................................28

Hình 14: Xóa các liên hệ bằng số điện thoại, địa chỉ hoặc tên................................29

Hình 15: Thêm liên hệ vào danh sách.....................................................................30

**MỞ ĐẦU**

1. **Mục đích thực hiện đề tài**

Việc xây dựng một chương trình giúp quản lý danh bạ điện thoại là yêu cầu cơ bản trong bộ môn “Cấu trúc dữ liệu” và “Kỹ thuật lập trình”. Trong đó, đề tài yêu cầu sinh viên phải nắm rõ kiến thức về các loại cấu trúc dữ liệu, cách cài đặt và quản lý thông tin trong struct, cũng như cách triển khai các chức năng chính của chương trình. Qua đó, sinh viên thực hành phương pháp giải quyết vấn đề, luyện tập và áp dụng kiến thức đã học vào trong một dự án thực tế.

**II. Mục tiêu đề tài**

Quản lý được một bộ dữ liệu bằng struct. Từ đây, có thể quản lý danh bạ điện thoại với các chức năng chính:

* Tìm các liên hệ có số điện thoại trùng nhau trong một tỉnh và yêu cầu xoá nếu người dùng muốn
* Liệt kê các liên hệ trong tỉnh
* Tìm liên hệ bằng số điện thoại
* Thêm liên hệ được nhập bàn phím
* Xoá liên hệ theo tỉnh, số điện thoại, địa chỉ, tên được nhập từ bàn phím
* Xuất danh sách liên hệ ra file

**III. Phạm vi và đối tượng nghiên cứu**

* Kiểu dữ liệu struct và ứng dụng
* Mảng cấp phát động
* Đọc ra và ghi vào file
* Ngôn ngữ C

**IV. Phương pháp nghiên cứu**

* Thông qua sự hướng dẫn của giảng viên phụ trách môn học
* Nghiên cứu những tài liệu liên quan: giáo trình, sách, mạng Internet,...

**V. Cấu trúc đồ án**

Phần 1: Tổng quan đề tài

Phần 2: Cơ sở lý thuyết

Phần 3: Chương trình và kết quả

Phần 4: Kết luận và hướng phát triển

1. **TỔNG QUAN ĐỀ TÀI**

**Đề A02: Danh bạ điện thoại**

***Danh bạ điện thoại cố định là một danh sách tuyến tính, mỗi phần tử của***

***nó ứng với một đơn vị thuê bao gồm 03 (ba) trường:***

- Tên đơn vị hoặc chủ đơn vị thuê bao.

- Địa chỉ

- Số điện thoại (không bao gồm mã vùng).

Một đơn vị Tổng đài muốn quản lý tất cả thuê bao của tất cả các tỉnh

(thành phố) nên muốn tập hợp dữ liệu từ 01 tập tin có nhiều trường có cấu

trúc như sau:

**(xxxx) Tên tỉnh 1**

Tên đơn vị 1 Địa chỉ 1 Số điện thoại 1

Tên đơn vị 2 Địa chỉ 2 Số điện thoại 1

….

**(xxxx) Tên tỉnh 2**

Tên đơn vị 1 Địa chỉ 1 Số điện thoại 1

Tên đơn vị 2 Địa chỉ 2 Số điện thoại 2

….

Chọn cấu trúc dữ liệu để lưu trữ Danh bạ trên và viết chương trình đọc dữ liệu

từ tập tin lưu trữ trên máy tính. Yêu cầu thực hiện các công việc sau:

*1. Tìm và thông báo nếu có 2 số điện thoại trùng nhau trong một tỉnh và yêu cầu*

*người dùng xóa 1 số nếu trùng.*

*2. Liệt kê danh bạ điện thoại từng tỉnh khi có yêu cầu.*

*3. Xây dựng chức năng tìm kiếm và hiển thị thông tin theo Số điện thoại hoặc tên*

*đơn vị.*

**Dữ liệu vào**: Tệp chứa danh sách danh bạ các tỉnh.

**Dữ liệu ra**: Hiển thị các thao tác theo menu chọn và lưu trữ sang tập tin nếu có

yêu cầu.

* 1. **Nhu cầu thiết yếu của đề tài:**

Việc quản lý danh bạ theo phương pháp truyền thống làm chúng ta mất rất nhiều thời gian và dễ bị trùng liên hệ. Việc chúng ta làm ra một chương trình giúp cho việc sắp xếp và tìm kiếm liên hệ trong danh bạ điện thoại giúp chúng ta tiết kiệm được rất nhiều thời gian so với phương pháp thủ công. Xuất phát từ lý do trên, nhóm em đã nghiên cứu đề tài: Danh bạ điện thoại giúp người dùng có thể tối ưu hóa danh bạ điện thoại của mình.

**1.2 Những ứng dụng của đề tài trong thực tiễn:**

* Giúp người dùng có thể tìm kiếm những số điện thoại bị trùng nhau trong một tỉnh và xoá một trong hai liên hệ trùng nếu người dùng muốn.
* Giúp người dùng có thể liệt kê các danh sách liên hệ trong một tỉnh mà họ mong muốn.
* Giúp người dùng tìm kiếm liên hệ theo số điện thoại hoặc địa chỉ.
* Giúp người dùng có thể xuất các danh sách liên hệ ra một file txt
* Giúp người dùng có thể xóa các liên hệ thông qua số điện thoại, địa chỉ hoặc tên.

**1.3 Giới thiệu tổng quan về đề tài:**

Đề tài “Danh bạ điện thoại” để giúp người dùng có thể quản lý danh bạ điện thoại của mình và giúp cho người lập trình vận dụng các kiến thức, kỹ năng cần thiết, đồng thời phát triển tư duy để thiết kế & lập trình ra những sản phẩm phần mềm thực tế. Nội dung của đề tài bao gồm:

* Nêu được cơ sở lý thuyết được áp dụng trong đề tài
* Đưa ra cách  tổ chức cấu trúc dữ liệu và thuật toán
* Viết chương trình và hiện thị ra kết quả

**2. CƠ SỞ LÝ THUYẾT**

**2.1 Ý tưởng**

Mỗi liên hệ được đưa vào một struct, danh sách liên hệ tạo thành một mảng với mỗi phần tử là một struct.

**2.2 Cơ sở lý thuyết**

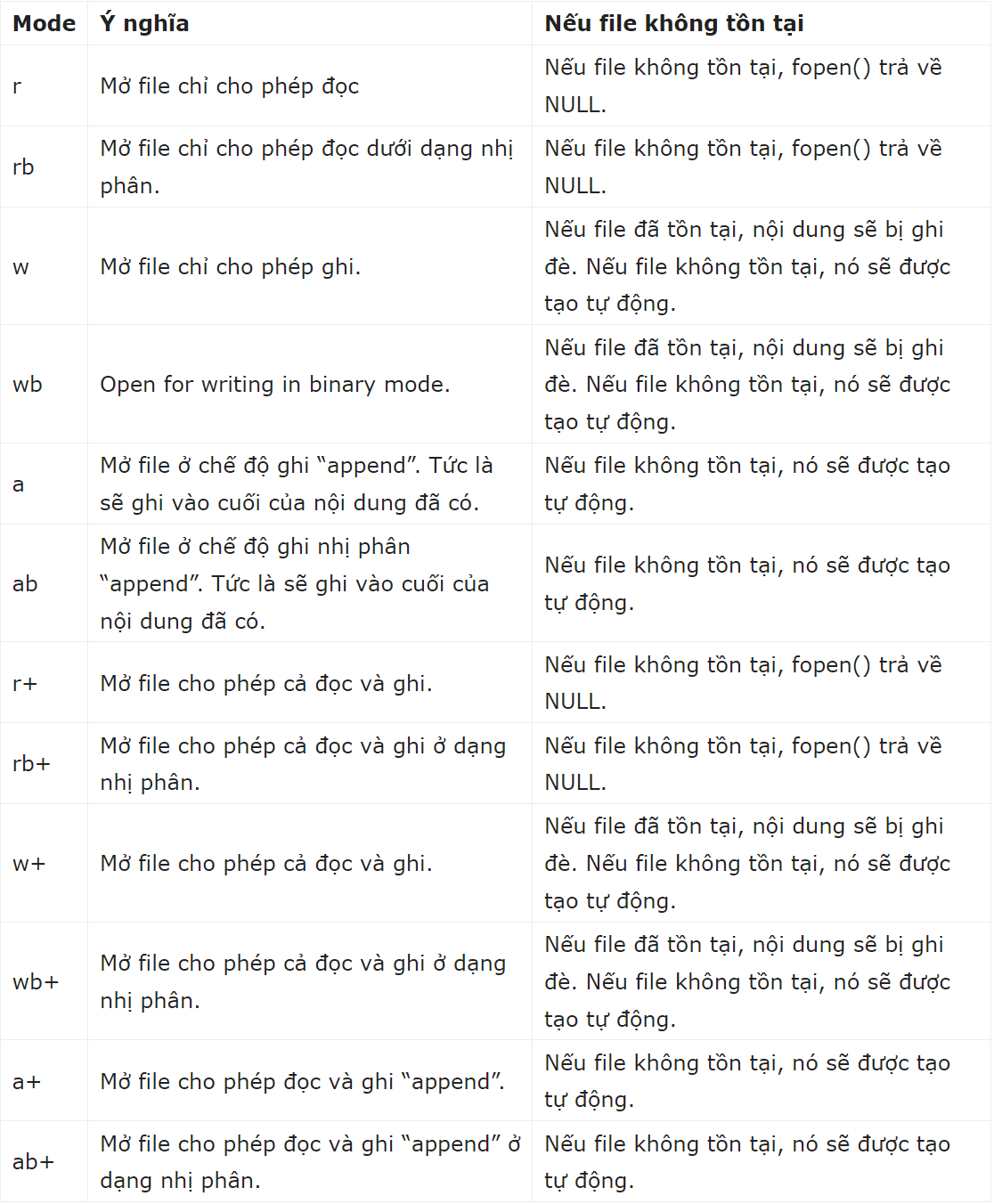
***2.2.1 Kiểu dữ liệu struct:***

Trong C, kiểu dữ liệu struct là một kiểu dữ liệu người dùng tự định nghĩa cho phép lưu trữ các loại phần tử khác nhau. Mỗi phần tử của một cấu trúc được gọi là một thành viên (member). Cấu trúc trong C có thể chứa các kiểu dữ liệu khác nhau như int, float, double, char và cả các kiểu dữ liệu khác như mảng và cấu trúc khác.Chúng ta có 2 toán tử dùng để truy xuất tới các biến thành viên của kiểu struct trong C.

* Sử dụng . => Toán tử truy xuất tới thành viên khi khai báo biến bình thường.
* Sử dụng -> => Toán tử truy xuất tới thành viên khi biến là con trỏ.

***2.2.2 Đọc và ghi vào file:***

* Thao tác mở file: FILE \*f=fopen(“ten file”,”mode”)
  + Lệnh fopen() được dùng trong thư viện stdio.h
  + Mode là tham số chúng ta cần chỉ định



* Thao tác đọc và ghi chuỗi vào file:
  + Muốn đọc file ta sử dụng mode “r”.Tuy nhiên nếu bạn muốn mở file để vừa đọc và vừa ghi đè nội dung mới vào file đó, lúc này mode cần dùng để mở file không phải là mode “r” mà sẽ là mode khác ví dụ như mode “r+”.
* Đọc vào 1 ký tự trong file ta dùng hàm fgetc
* Đọc vào 1 ký tự trong file ta dùng hàm fgets
  + Muốn ghi vào file ta sử dụng hàm**fputs(const char\*s, FILE \*fp)** ,hàm này giúp ta ghi chuỗi**s** đến một luồng ra tham chiếu bởi fp. Nó trả về một giá trị không âm nếu thành công và trả về ký tự EOF(là giá trị đánh dấu hết tập tin) nếu xảy ra một lỗi. Bạn có thể sử dụng **hàm int fprintf(FILE \*fp,const char \*format,...)** để ghi một chuỗi ra file.
* Thao tác đóng file: **fclose(FILE \*fp);**
  + Hàm **fclose()** được định nghĩa trong thư viện stdio.h
  + Hàm **fclose( )** trả về giá trị zero nếu thành công hoặc **EOF** nếu có lỗi trong quá trình đóng file. Hàm này thực tế xóa các dữ liệu trong bộ đệm đối với file, đóng file và giải phóng bộ nhớ được sử dụng với file

**3. THUẬT TOÁN**

**3.1.     Phát biểu bài toán**

Xây dựng chương trình Danh bạ điện thoại đơn giản để giúp người dùng quản lý danh bạ điện thoại của mình. Cài đặt chương trình thực hiện các chức năng sau:

* Tìm các liên hệ trùng nhau trong một tỉnh và xoá một trong hai liên hệ trùng nếu có yêu cầu
* Liệt kê các liên hệ trong tỉnh
* Tìm liên hệ theo số điện thoại hoặc địa chỉ
* Thêm liên hệ được nhập từ bàn phím
* Xóa các liên hệ có số điện thoại, tên hoặc địa chỉ được nhập từ bàn phím
* Xuất danh sách liên hệ ra file

**Input**:

* Đọc file: file gồm danh sách các liên hệ, với mỗi liên hệ cần lưu trữ các thông tin về tên, số điện thoại, địa chỉ và tỉnh của chủ liên hệ. Danh sách liên hệ trong file của một tỉnh có dạng như sau:

**Tên tỉnh**

Các dòng tiếp theo của tỉnh có dạng như sau:

**Địa chỉ|Tên|Số điện thoại**

Giữa các tỉnh cách nhau bởi một dòng trắng.

* Đối với các chức năng trên yêu cầu người dùng nhập vào một chuỗi để có thể thực hiện chức năng đó.
* Đối với chức năng chơi game, chương trình sẽ cho người dùng lựa chọn level, ứng với mỗi level chương trình chọn ngẫu nhiên từ vựng nào đó, xóa một cách ngẫu nhiên một ký tự trong từ và yêu cầu người chơi nhập vào ký tự bị thiếu.

**Output**:

* Đối với các chức năng liệt kê các liên hệ trong tỉnh, tìm liên hệ theo số điện thoại và địa chỉ, xóa các liên hệ có số điện thoại, tên hoặc địa chỉ yêu cầu người dùng nhập vào một chuỗi để có thể thực hiện chức năng đó.
* Đối với chức năng tìm các liên hệ trùng và xoá nếu muốn, chương trình sẽ cho người dùng danh sách các liên hệ trùng nhau trong một tỉnh và người dùng có thể chọn xoá liên hệ thứ mấy bị trùng hoặc không xoá liên hệ nào.
* Đối với chức năng thêm một liên hệ vào danh sách yêu cầu người dùng nhập vào tên, số điện thoại, tỉnh và địa chỉ của liên hệ cần thêm vào danh sách.
* Đối với chức năng xuất danh sách ra file txt người dùng cần tạo sẵn một file txt và sao chép đường dẫn đưa vào chuỗi ký tự hoặc chương trình

**3.2. Cấu trúc dữ liệu**

* **Cấu trúc dữ liệu Struct**

- Struct là một kiểu dữ liệu đặc biệt trong C , cho phép tự định nghĩa một kiểu dữ liệu mới bằng cách kết hợp các kiểu dữ liệu khác nhau. Cấu trúc struct có thể chứa nhiều thành phần khác nhau, mỗi thành phần có thể là một kiểu dữ liệu khác nhau.

- Sử dụng cấu trúc dữ liệu struct để cài đặt danh sách chứa nội dung các liên quan đến danh bạ.

- Dữ liệu đầu vào là danh sách danh bạ của người dùng.

- Dữ liệu ra là các kết quả mà người dùng mong muốn , tìm kiếm.

* Mảng cấp phát động:

- Là một khối dữ liệu liên tiếp trong bộ nhớ, được cấp phát theo kích thước tùy ý và có thể thay đổi kích thước trong quá trình thực thi.

- Có thể truy cập trực tiếp vào phần tử bằng số. Truy cập vào phần từ bằng chỉ số có thời gian truy cập 0(1).

- Chèn hoặc xoá phần tử tại vị trí bất kỳ trong mảng có thể đòi hỏi việc sao chép lại toàn bộ dữ liệu trong mảng, dẫn đến độ phức tạp là O(n).

- Có kích thước cố định khi được cấp phát, nhưng có thể thay đổi kích thước bằng cách cấp phát lại và sao chép dữ liệu sang mảng mới.

- Cần một khối bộ nhớ liên tiếp để lưu trữ dữ liệu.

**3.3 Thuật toán**

* **Hàm readConatctFromLocal(đọc dữ liệu từ file):**

**Ý tưởng:**

* Ta đọc từng tỉnh sau đó với mỗi tỉnh đọc lần lượt đọc các thông tin của các liên hệ trong tỉnh đó. Với định dạng mỗi tỉnh như sau:

**Tên tỉnh**

**Địa chỉ | Tên | Số điện thoại**

Và giữa các tỉnh có một dòng trống

* Trong hàm **readContactFromLocal**, chúng em đã sử dụng 2 hàm là **strch** và **strtok**:
  + Hàm char **\*strchr(const char \*str, int c)** tìm kiếm sự xuất hiện đầu tiên của ký tự c (một unsigned char) trong chuỗi được trỏ tới bởi tham số str.Hàm này trả về một con trỏ tới sự xuất hiện đầu tiên của ký tự c trong chuỗi str, hoặc trả về **NULL** nếu không tìm thấy ký tự đó.
  + Hàm **char \*strtok(char \*str, const char \*delim)** chia chuỗi str thành một dãy các **token**(các chuỗi nhỏ hơn được chia ra) được phân biệt riêng rẽ bởi dấu tách **delim** (ví dụ: dấu phẩy, ...).Hàm này trả về con trỏ tới token cuối cùng được tìm thấy trong chuỗi. Một con trỏ null được trả về nếu không thu được token nào

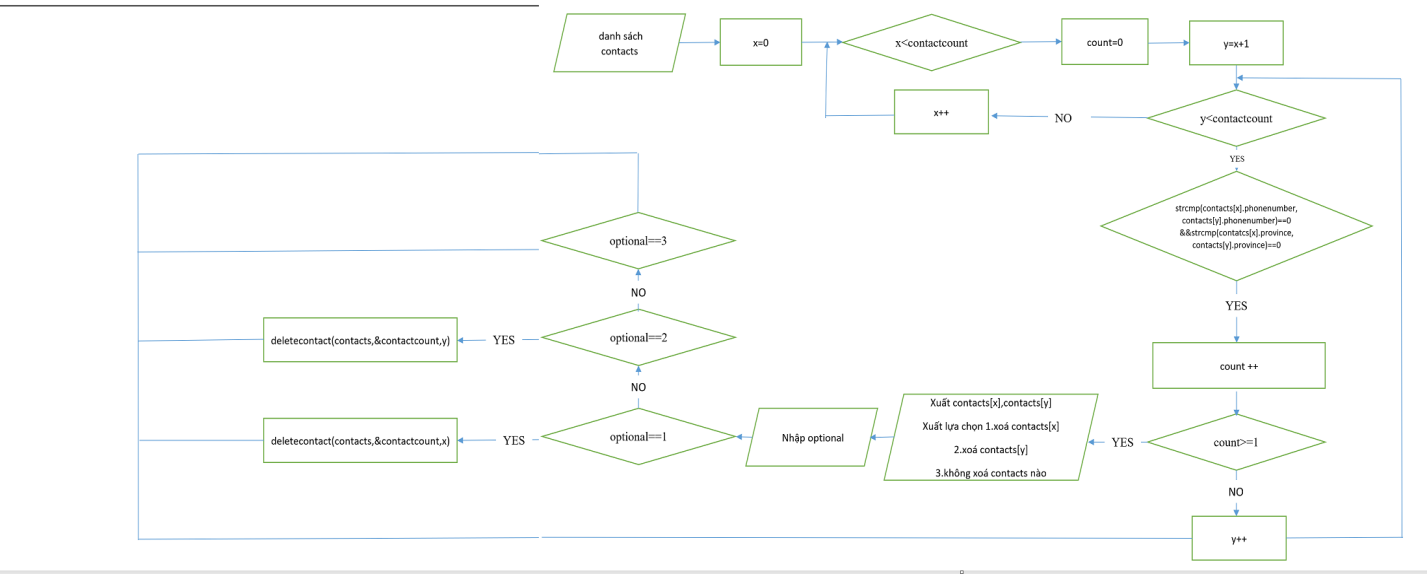
**Mô tả thuật toán bằng ngôn ngữ tự nhiên:**

Duyệt qua từng dòng của file*“phonenumber.txt”,* và thực hiện các bước sau:

1. Kiểm tra nếu file rỗng thì thông báo ra màn hình:”Error opening file”
2. Kiểm tra nếu độ dài của dòng bằng 0 thì continue.
3. Kiểm tra xem từng dòng có chứa kí tự “|” không. Nếu không thì copy dòng đó vào Provincename.
4. Nếu có xuất hiện ký tự “|” thì tách lần lượt các chuỗi và lưu lần lượt vào address, name, phonenumber.
5. Copy 4 biến Provincename, address, name, phonenumber vào contact (kiểu dữ liệu struct).
6. Tăng biến count để đếm số liên hệ.
7. Cấp phát bộ nhớ cho mảng các struct contact và lưu contact vào listContact[count-1].
8. Lặp lại các bước trên tới khi nào duyệt qua hết một từ, trả về kết quả cuối cùng.

* **Hàm tìm các số điện thoại trùng nhau ở cùng 1 tỉnh**

**Mô tả thuật toán bằng sơ đồ khối:**



Hình 1

**Mô tả thuật toán bằng giả lệnh (pseudocode) kết hợp ngôn ngữ tự nhiên:**

Hàm này tìm kiếm các số điện thoại trùng lặp trong danh sách liên hệ được đọc từ một đường dẫn tệp '**path**'.

Đầu tiên, nó đọc các liên hệ từ tệp cục bộ bằng chức năng '**readContactFromLocal**' và lưu trữ chúng trong một mảng các cấu trúc '**Contact**'.

Sau đó, nó khởi tạo một mảng cấu trúc '**Contact**' trống có tên là '**listDuplicateContact**' với kích thước tối đa là 1000.

**void findNumberPhoneDuplicate(char\* path) {// Tìm số điện thoại trùng lặp theo tỉnh**

**int contactCount = 0;**

**struct Contact\* contacts=readContactFromLocal(path,&contactCount);**

**struct Contact listDuplicateContact[1000];**

**//int contactCount = 0;**

**contacts=readContactFromLocal(path,&contactCount);**

Tiếp theo, nó lặp qua từng contact trong mảng **contacts** và lưu trữ liên hệ hiện tại trong một biến tạm thời có tên là '**temp**'.

Sau đó, nó khởi tạo một biến có tên '**count**' thành 0. Bước tiếp theo là so sánh số điện thoại trong cấu trúc '**temp**' với các số điện thoại trong tất cả các liên hệ khác trong mảng **contacts**.

**for (int x = 0; x < contactCount; ++x) {**

**struct Contact temp = contacts[x];**

**int count = 0;**

**for (int y = x+1; y < contactCount; y++) {**

**struct Contact current = contacts[y];**

**if (strcmp(temp.phonenumber, current.phonenumber) ==**

**0 && strcmp(temp.province, current.province) == 0 ) {**

**count++;**

**}**

Nếu tìm thấy kết quả trùng khớp, nó sẽ tăng biến '**count**'. Sau khi so sánh cấu trúc '**temp**' với tất cả các contact khác, nếu '**count**' lớn hơn 1, điều đó có nghĩa là có các số điện thoại trùng lặp trong danh sách.

Vòng lặp for bên trong ( vòng lặp thứ hai ) bắt đầu từ phần tử tiếp theo của mảng '**contacts**' sau phần tử hiện tại được lưu trữ trong biến '**temp**' . Biến '**current**' được sử dụng để lưu trữ phần tử hiện tại đang được xét .

=> So sánh số điện thoại và tỉnh/thành phố của biến 'temp' và biến 'current' bằng cách sử dụng hàm '**stcmp**'. Nếu phép so sánh trả về 0 (có nghĩa là hai số điện thoại này trùng nhau ở cùng 1 tỉnh ).

- Sau khi kết thúc vòng lặp

+ Nếu '**count**' =0 thì không có số điện thoại nào trùng nhau trong 1 tỉnh nên không cần xoá .

+ Nếu '**count**' >=1 thì đưa ra lựa chọn xoá số nào trong những số trùng nhau

**if (count >= 1) {**

**// in tên người dùng, số điện thoại, tỉnh trùng nhau.**

**printf("%s\t%s\t%s\n",temp.name,temp.phonenumber,temp.province);**

**printf("%s\t%s\t%s\n",current.name,current.phonenumber,current.province);**

**printf("bạn có muốn xóa 1 trong 2 liên hệ không?\n");**

**int optional;**

**printf("1.Xoá liên hệ trùng đầu tiên.\n");**

**printf("2.Xoá liên hệ trùng thứ 2.\n");**

**printf("3.Không xoá liên hệ trùng.\n");**

**scanf("%d",&optional);**

**switch (optional) {**

**case 1:**

**deletecontact(contacts,&contactCount,x);**

**break;**

**case 2:**

**deletecontact(contacts,&contactCount,y);**

**break;**

**case 3:**

**break;**

**default:**

**printf("Vui lòng nhập các số từ 1 đến 3:\n");**

**continue;**

**}**

**break;**

**}**

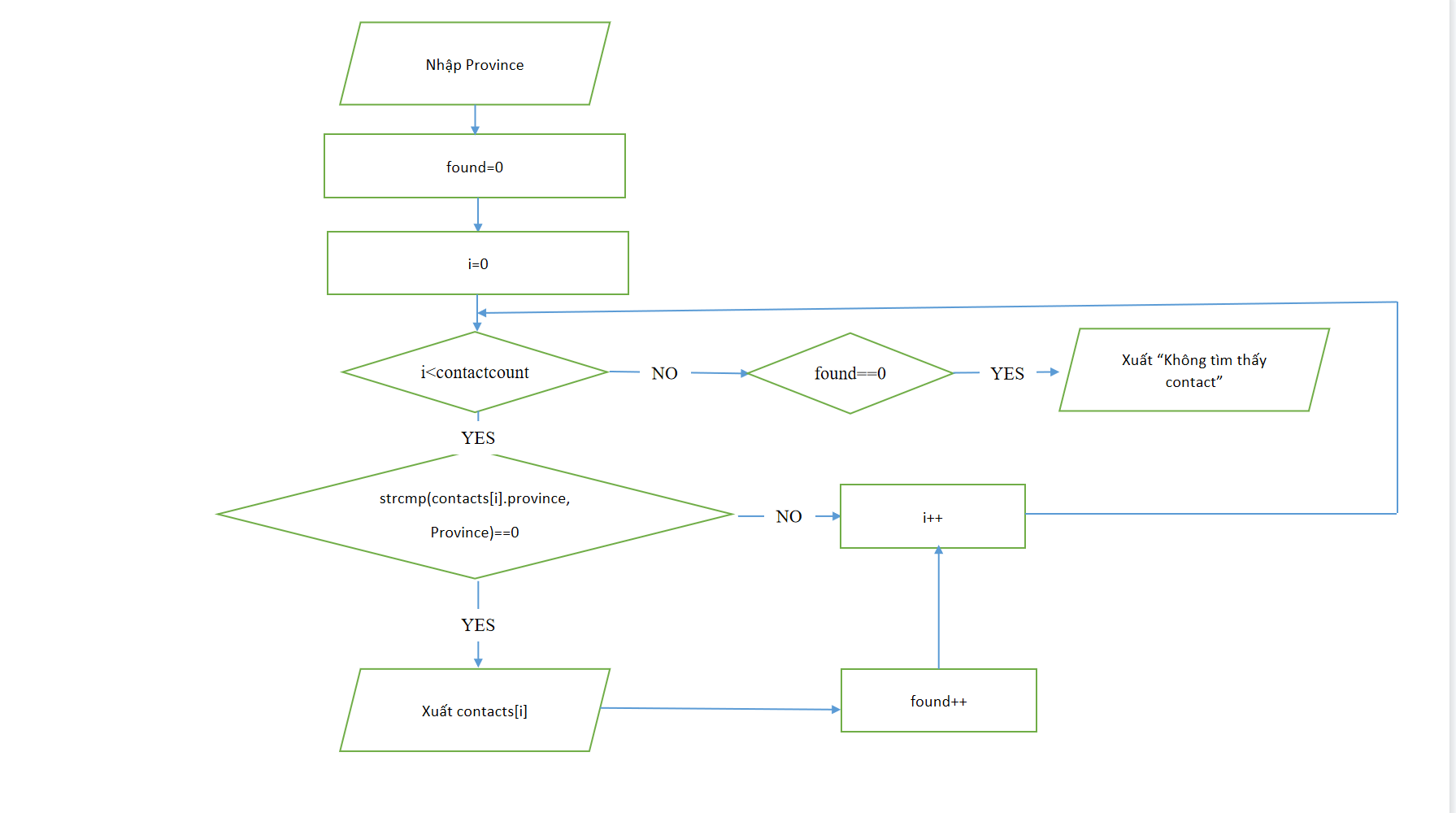
**}**

Sau đó xoá phần tử ra khỏi danh sách và cập nhật dữ liệu mới vào tệp tin local file

**printlistcontact(contacts,contactCount);**

* **Hàm liệt kê các liên hệ trong một tỉnh**

**Mô tả thuật toán bằng sơ đồ khối:**



Hình 2

**Mô tả thuật toán bằng giả lệnh (pseudocode) kết hợp ngôn ngữ tự nhiên:**

**algorithm**: getListContactByProvince

**input:** tên tỉnh cần liệt kê Province

**output:** bảng thông tin các liên hệ trong tỉnh

**begin**

x=0 -> n

begin

if strcmp(conatcts[x].province,Province)==0 found++;

List[found-1]=contacts[x];

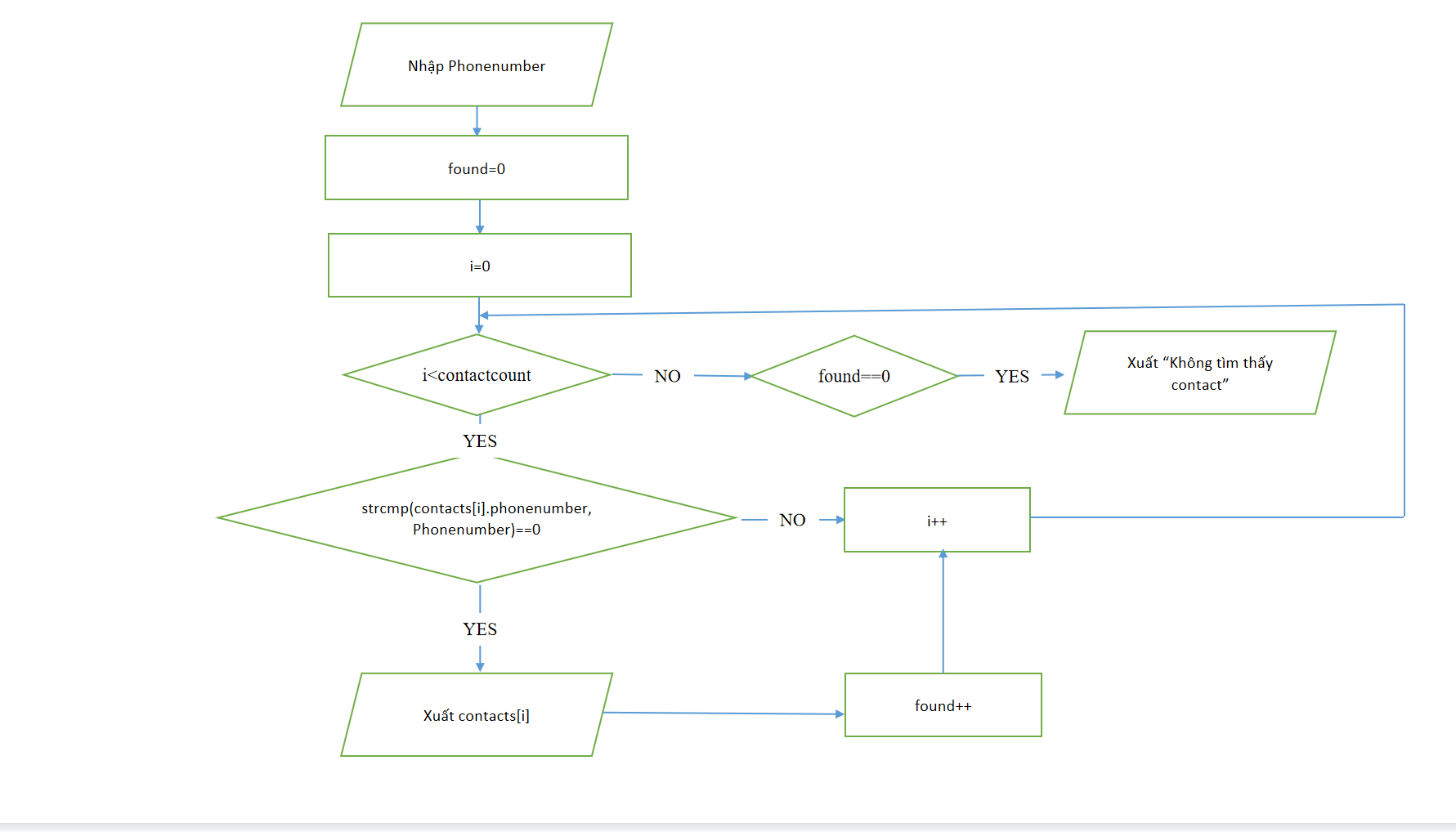
end

In danh sách các liên hệ

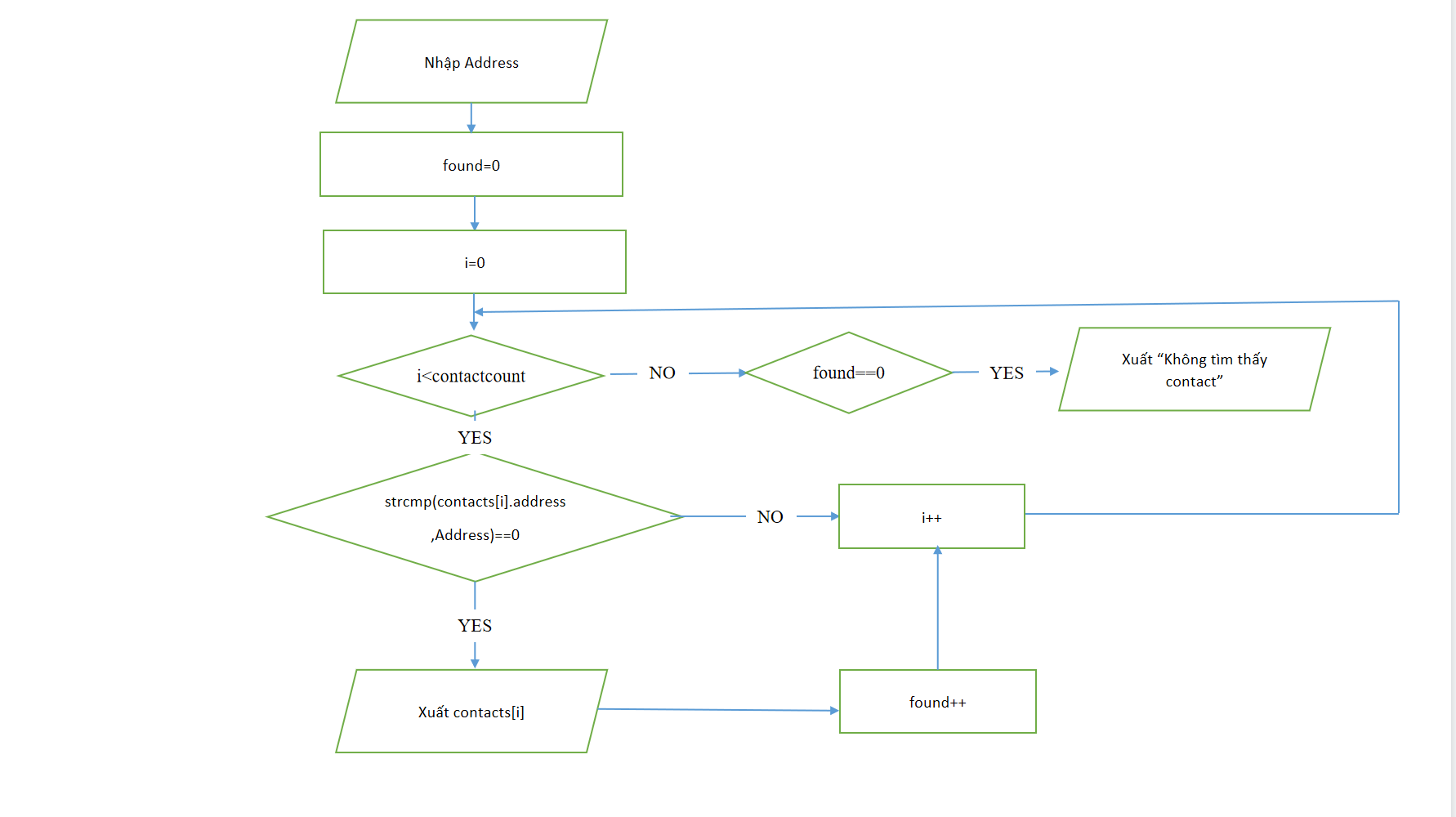
**end**

**\* Hàm tìm kiếm contact bằng địa chỉ ( tương tự bằng số điện thoại )**

**Mô tả thuật toán bằng sơ đồ khối:**



Hình 3



Hình 4:

**Mô tả thuật toán bằng giả lệnh (pseudocode) kết hợp ngôn ngữ tự nhiên:**

Hàm **findContactByAddress** nhận hai tham số: '**path**' (chuỗi biểu thị đường dẫn của tệp liên hệ) và '**Address**' (chuỗi biểu thị địa chỉ cần tìm kiếm trong tệp liên hệ). Bên trong hàm :

+ biến '**contactCount**' được khởi tạo bằng 0

+ con trỏ '**contacts**' được khai báo dưới dạng cấu trúc chứa thông tin liên hệ. Con trỏ này được chỉ định là kết quả của hàm '**readContactFromLocal**' đọc thông tin liên hệ từ tệp cục bộ được chỉ định bởi tham số '**path**' và cập nhật biến '**contactCount**' tương ứng.

**void findContactByAddress(char\* path, char\* Address){**

**int contactCount = 0;**

**struct Contact\* contacts;**

**contacts=readContactFromLocal(path,&contactCount);**

**printf("cac contact co dia chi %s la:\n",Address);**

**int found = 0;**

Sau đó, một thông báo được in trên bảng điều khiển cho biết rằng các liên hệ với địa chỉ đã cho đang được tìm kiếm. Một vòng lặp được bắt đầu lặp lại trên tất cả các liên hệ trong mảng '**contacts**'. Đối với mỗi contacts, trường địa chỉ được so sánh với tham số '**Address**' bằng cách sử dụng hàm '**stcmp**'.

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

**if (strcmp(contacts[i].address, Address) == 0) {**

**printf("Province: %s\n",contacts[i].province);**

**printf("Name: %s\n", contacts[i].name);**

**printf("Phone Number: %s\n", contacts[i].phonenumber);**

**printf("Address: %s\n", contacts[i].address);**

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

**found = 1;**

**}**

**}**

+ Nếu phép so sánh trả về 0 (có nghĩa là hai chuỗi bằng nhau), đoạn code sẽ in ra thông tin liên lạc , số điện thoại và tỉnh/thành phố của liên lạc đó.

+ Nếu phép so sánh trả về khác 0 ( có nghĩa hai chuỗi khác nhau ) không tìm thấy liên lạc nào có địa chỉ trùng khớp với Address thì sẽ in ra thông báo " **không tìm thấy contact có địa chỉ Address** ".

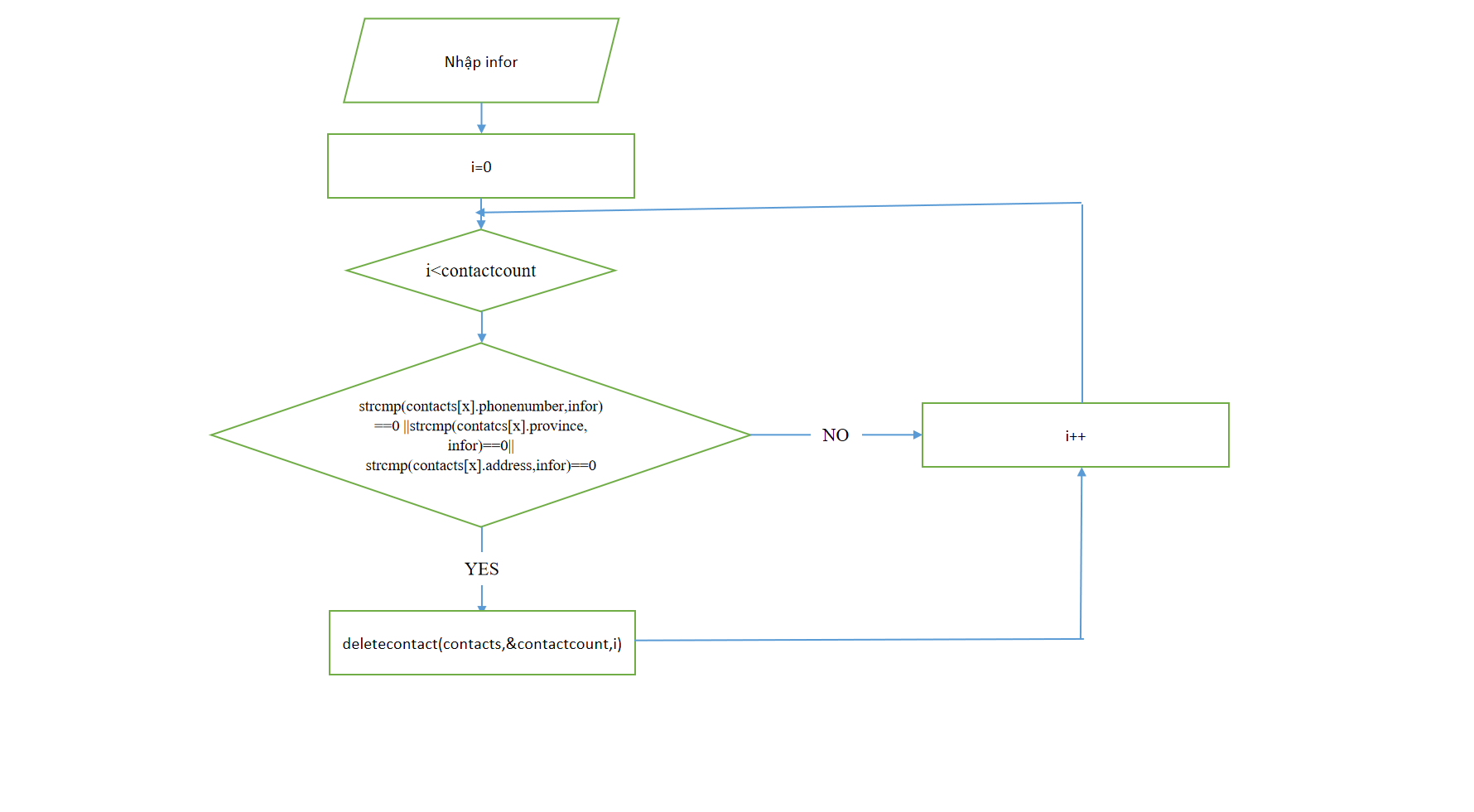
- Biến ' **Found** ' được sử dụng để kiểm tra xem có thấy liên lạc nào không .

+ Nếu tìm thấy, giá trị của ' **Found** ' sẽ được gán bằng 1.

+ Nếu không tìm thấy thì giá trị của ' **Found** ' sẽ vẫn là 0.

\* **Hàm xoá (xoá 1 contact bằng số điện thoại , tên người dùng hoặc địa chỉ )**

**Mô tả thuật toán bằng sơ đồ khối:**



Hình 5

**Mô tả thuật toán bằng giả lệnh (pseudocode) kết hợp ngôn ngữ tự nhiên:**

Đây là một hàm xoá dữ liệu liên lạc từ tệp tin local file được lưu trữ trong đường dẫn ( **path** ). Hàm này đọc toàn bộ danh sách liên lạc từ tệp tin local file thông qua hàm **readContactFromLocal ( )** và lưu vào mảng contacts với **numcontact** là số lượng liên lạc trong danh sách.

**void xoa(char\* path,char\* infor){**

**int numcontact = 0;**

**struct Contact\* contacts;**

**contacts=readContactFromLocal(path,&numcontact);**

Sau đó , hàm kiểm tra từng phần tử trong danh sách liên lạc để tìm kiếm thông tin trùng khớp với biến infor ( tên , địa chỉ , số điện thoại ) . Nếu tìm thấy , phần tử đó sẽ bị xoá khỏi danh sách .

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

**if(strcmp(contacts[i].address,infor)==0 || strcmp(contacts[i].name,infor)==0 || strcmp(contacts[i].phonenumber,infor)==0){**

**deletecontact(contacts,&numcontact,i);**

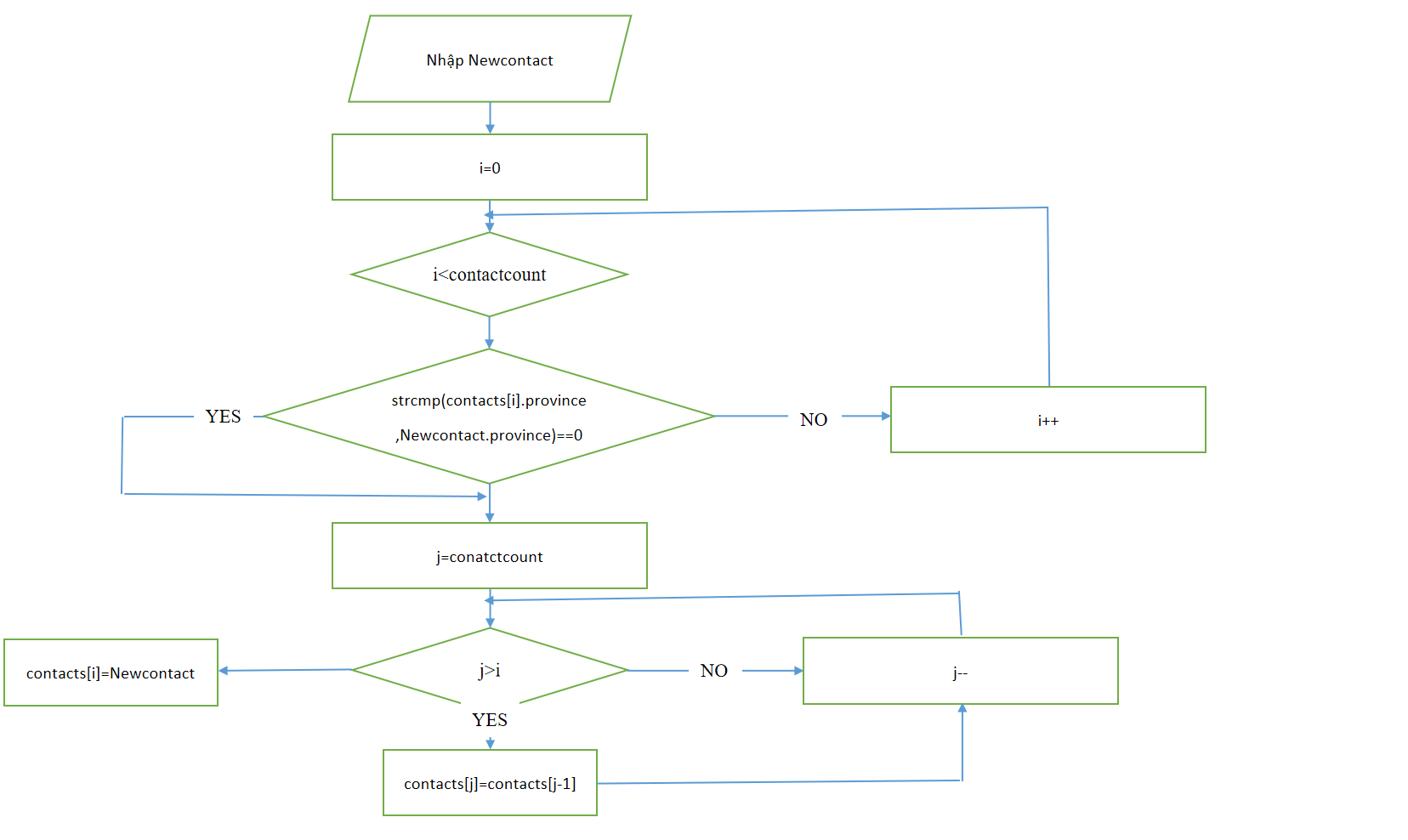
**}**

Sau đó xoá phần tử ra khỏi danh sách và cập nhật dữ liệu mới vào tệp tin local file

**printlistcontact(contacts,numcontact);**

**\* Hàm InsertContact ( thêm contact nhập từ bàn phím )**

**Mô tả thuật toán bằng sơ đồ khối:**

****

**Hình 6**

**Mô tả thuật toán bằng giả lệnh (pseudocode) kết hợp ngôn ngữ tự nhiên:**

*Các tham số đầu vào của hàm bao gồm :*

- '**NewContact**' : đây là biến kiểu '**Contact**' chứa thông tin liên lạc mới cần thêm vào .

- '**contactcount**' : đây là con trỏ tới biến kiểu '**int**' chứa số lượng liên lạc hiện tại trong danh sách .

- '**contacts**' : đây là con trỏ tới mảng kiểu '**Contact**' chứa danh sách các liên lạc hiện có .

Hàm bắt đầu với một vòng lặp for để duyệt qua danh sách các contacts :

+ Nếu contact mới có tỉnh/thành phố giống với tỉnh/thành phố của contact nào đó trong danh sách , thì vòng lặp sẽ dừng lại và biến i sẽ lưu chỉ số của contact đó .

+ Nếu không có contact nào trùng tỉnh/thành phố thì i sẽ bằng giá trị contactcount ( số lượng contact hiện tại ), tức là contact mới sẽ được thêm vào cuối danh sách.

**void InsertContact(struct Contact NewContact, int \*contactcount, struct Contact\* contacts) {**

**int i, j;**

**// Tìm vị trí chèn mới theo thứ tự tên**

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

**if (strcmp(NewContact.province, contacts[i].province) == 0)**

**break;**

**}**

Tiếp theo thực hiện 1 vòng lặp for để thêm 1 danh bạ mới ( **NewContact** ) vào vị trí i trong mảng danh bạ contacts và tăng đếm số lượng danh bạ ( **\*contactcount**) lên 1 . cụ thể :

/**/ Dịch chuyển các phần tử phía sau để tạo vị trí cho liên hệ mới**

**for (j = \*contactcount; j > i; j--) {**

**contacts[j] = contacts[j - 1];**

**}**

**// Chèn liên hệ mới vào vị trí tìm được**

**contacts[i] = NewContact;**

**(\*contactcount)++;**

**return;**

**}**

- Vòng lặp for sẽ di chuyển tất cả các phần tử của mảng contacts bắt đầu từ vị trí i đến cuối danh sách xuống 1 vị trí ( tức là **contact[j]** ) sẽ được gán bằng **contacts[j-1]** ) để tạo ra một chỗ trống ở vị trí i .

- Sau đó , phần tử mới **NewContact** sẽ được gán vào vị trí i trong mảng contacts.

- Cuối cùng , biến đếm ( **\*contactcount** ) sẽ được tăng lên 1 để chỉ số lượng danh bạ đã được thêm vào .

**4. CHƯƠNG TRÌNH VÀ KẾT QUẢ**

**4.1 Tổ chức chương trình :**

### **4.1.1 Thư viện**

Sử dụng thư viện: stdio.h, string.h, stdlib.h

### **4.1.2 Các hàm dùng trong chương trình**

hàm **readContactFromLocal** : đọc dữ liệu từ file path.

hàm **addContactToLocalFile** : thêm 1 contact vào file path.

hàm **getListContact** : lấy danh sách contact và in ra màn hình.

hàm **deletecontact** : xoá liên hệ thứ x.

hàm **printlistcontact** : in danh sách các liên hệ ra màn hình.

hàm **fineNumberPhoneDuplicate** : tìm các số điện thoại trùng nhau ở cùng một tỉnh

hàm **getListContactByProvince** : tìm kiếm và in ra các contact trong tỉnh.

hàm **findContactByPhoneNumber** : tìm kiếm và in ra các contact có số điện thoại ( được nhập từ bàn phím theo nhu cầu người dùng muốn ).

hàm **findContactByAddress** : tìm kiếm và in ra các contact có địa chỉ ( được nhập từ bàn phím theo nhu cầu người dùng muốn ).

hàm **InsertContact** : thêm contact nhập từ bàn phím.

hàm **writetofile** : in danh sách các liên hệ ra file out.

hàm **xoa** : xoá 1 contact bằng số điện thoại , tên người dùng hoặc địa chỉ.

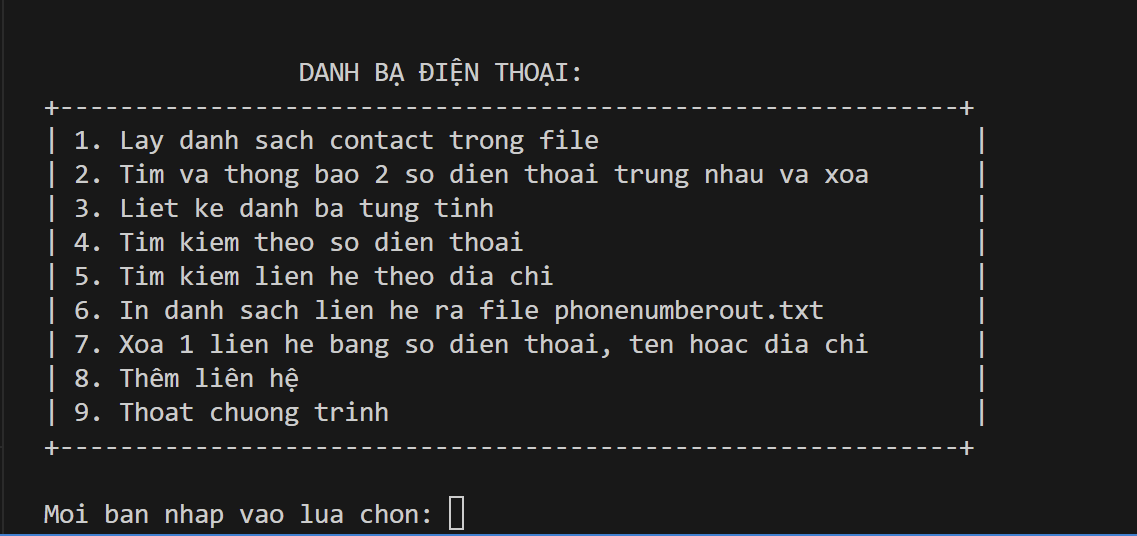
hàm **trim** : xoá các kí tự trắng , kí tự xuống dòng , kí tự chuyển tab đầu và cuối chuỗi.

**4.2 Ngôn ngữ cài đặt:**

Ngôn ngữ C

**4.3. Kết quả**

***4.3.1 Giao diện chính của chương trình***



Hình 7 . Giao diện chính của chương trình

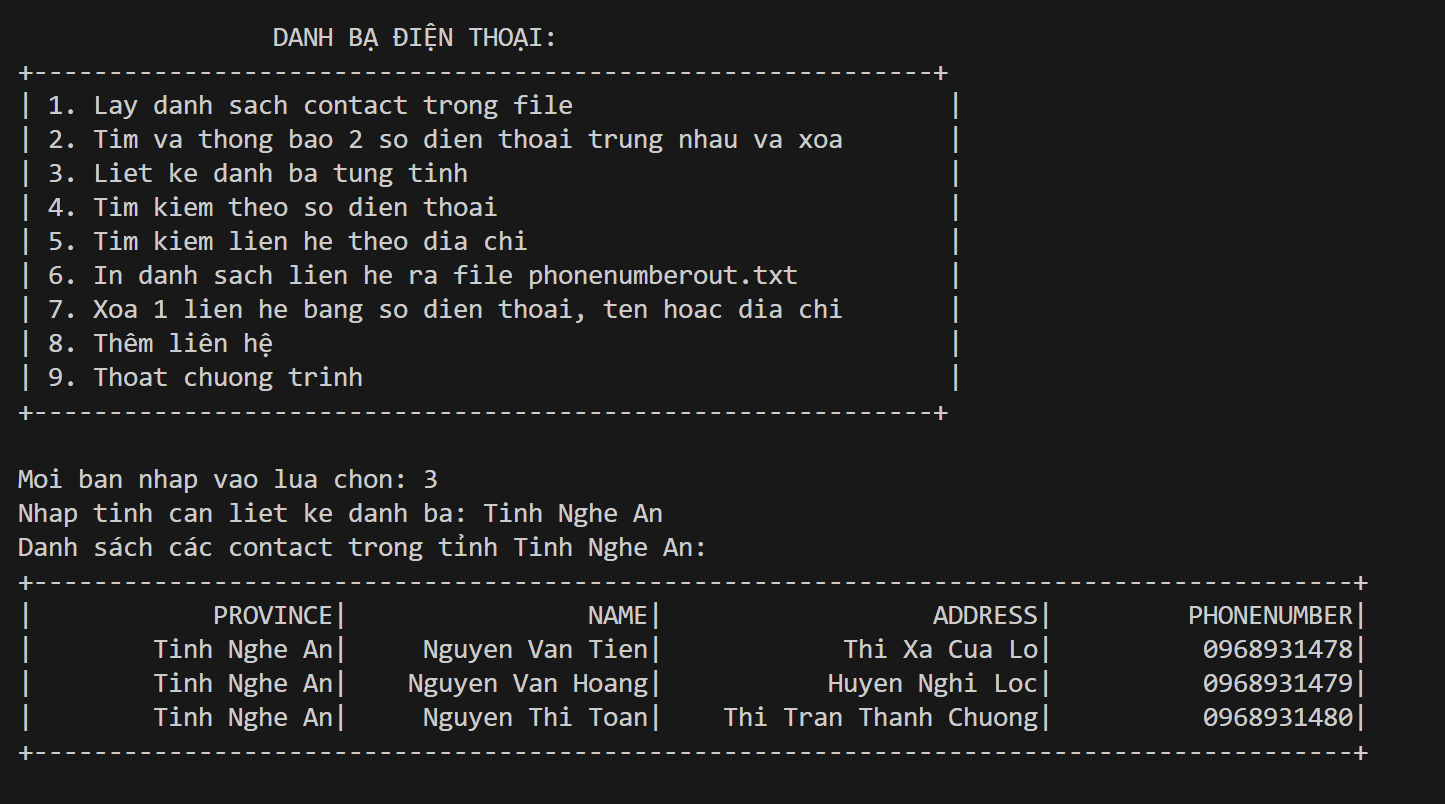
***4.3.2 Kết quả thực thi của chương trình***



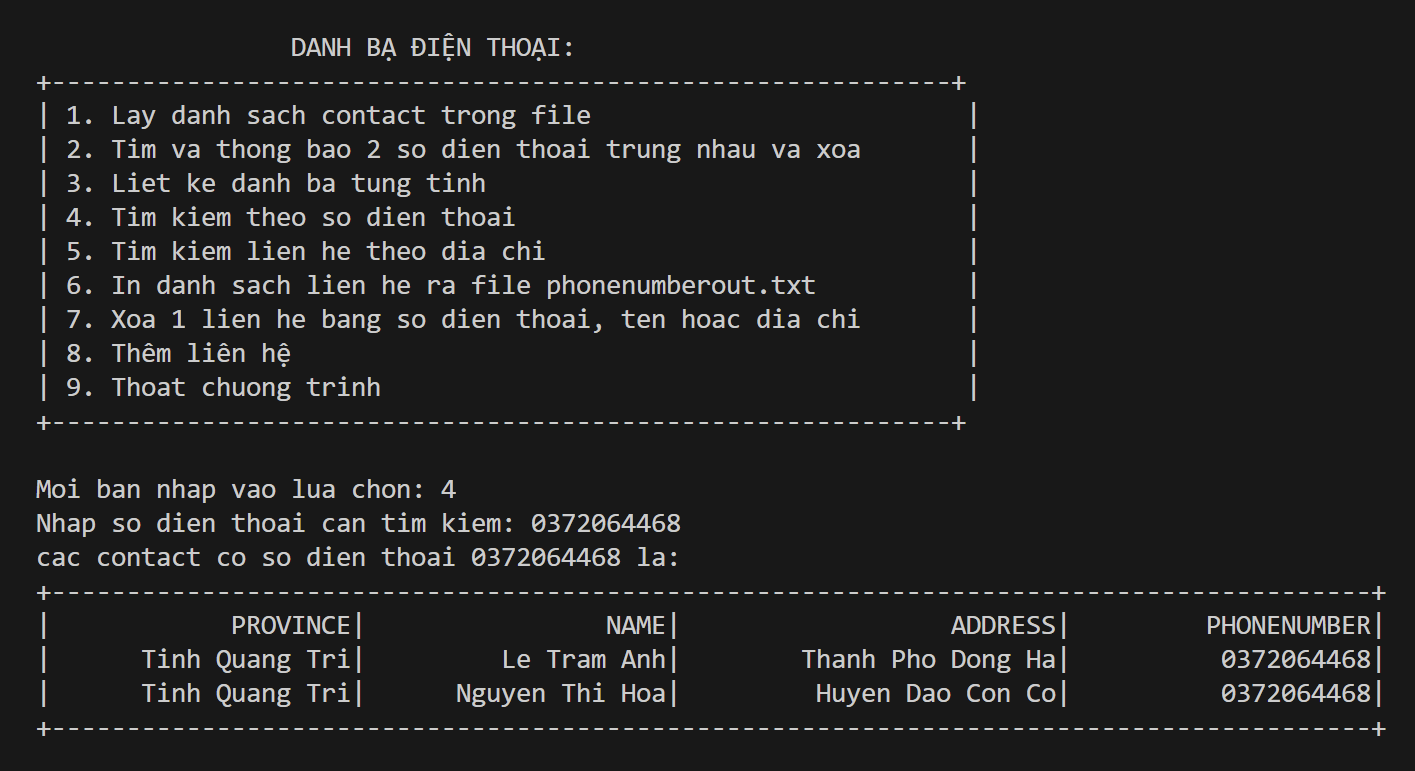
Hình 8: Thực hiện đọc danh sách từ file



Hình 9: Tìm 2 liên hệ trùng nhau trong 1 tỉnh và xoá nếu muốn



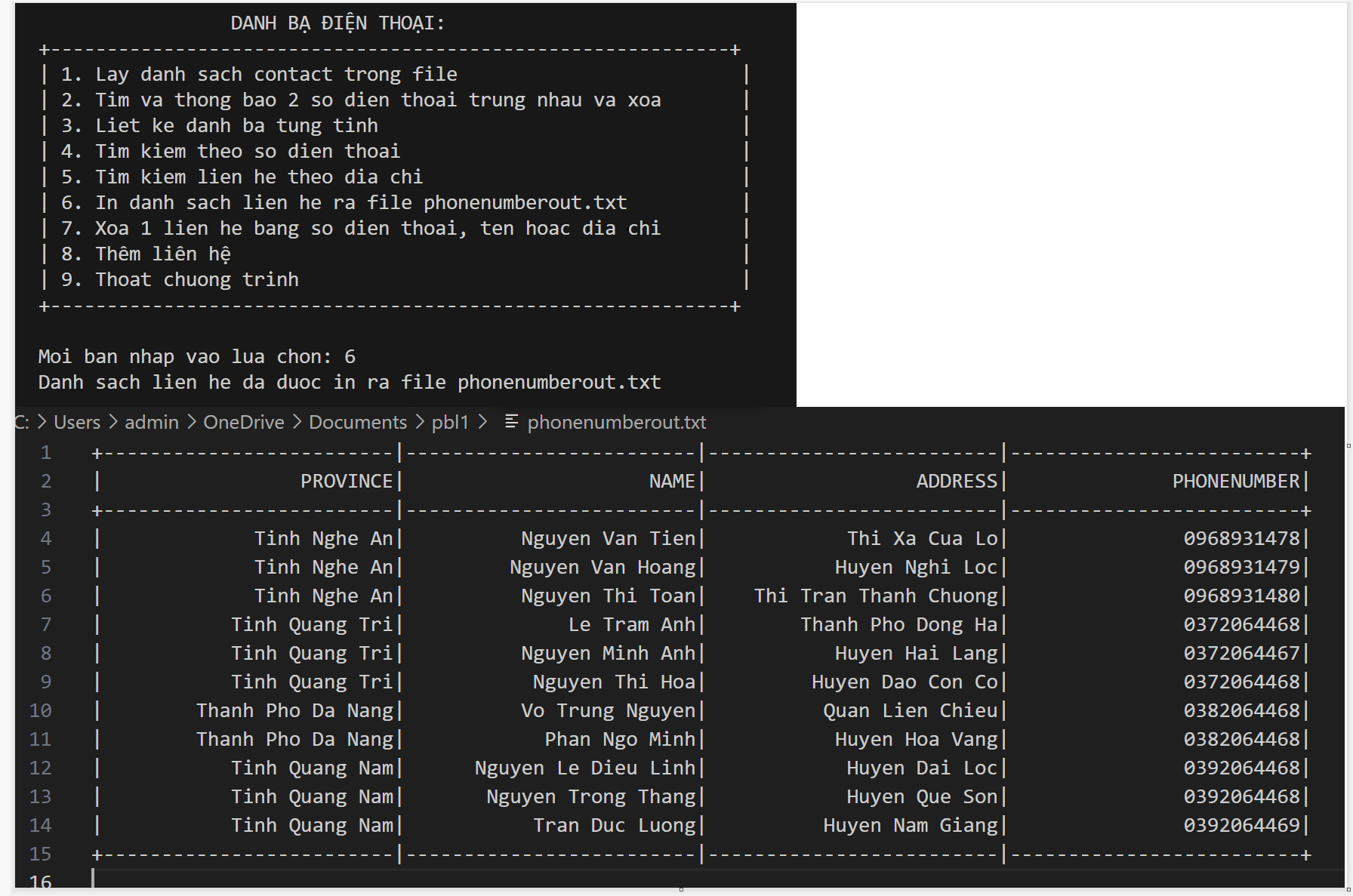
Hình 10: Thực hiện liệt kê các liên hệ trong một tỉnh



Hình 11. Thực hiện tìm các liên hệ theo số điện thoại



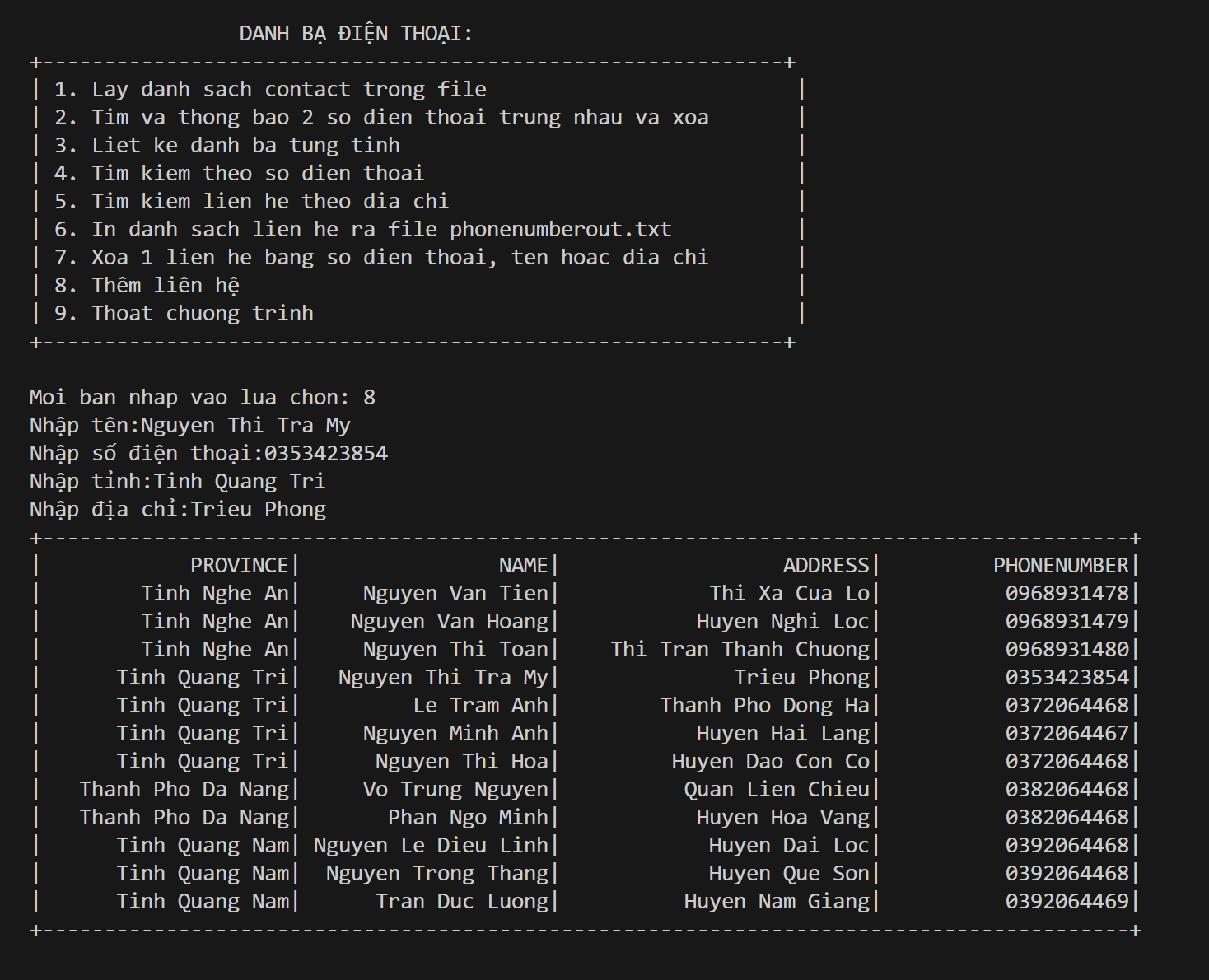
Hình 12: Thực hiện tìm các liên hệ theo địa chỉ



Hình 13:Giao diện chương trình và file phonenumberout.txt



Hình 14:Xoá liên hệ qua số điện thoại, tên hoặc địa chỉ



Hình 15:Thêm liên hệ vào danh sách

**5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

**5.1 Kết luận**

Qua một thời gian nghiên cứu kết hợp với học hỏi kinh nghiệm từ thầy giáo hướng dẫn cũng như các bạn có cùng đề tài, báo cáo đã hoàn thành và đạt được những kết quả sau:

* Về lí thuyết: Báo cáo đã trình bày được :
* Kiểu dữ liệu struct và ứng dụng.
* Đọc ra và ghi vào file.
* Về thực tiễn: Báo cáo đã trình bày chương trình quản lý danh bạ đơn giản trong bằng ngôn ngữ C.

**5.2. Hướng phát triển**

Bên cạnh những vấn đề đã đạt được như đã trình bày, đề tài cần được phát triển thêm các vấn đề sau:

* Chuyển sang ngôn ngữ lập trình bậc cao hơn ( như C++ hay java ) để sử dụng các cấu trúc dữ liệu trong thư viện để tối ưu hóa chương trình.
* Tạo nên một phần mềm để được ứng dụng rộng rãi trong cả nước.
* Tối ưu hóa thuật toán để mô hình có thể hoạt động trơn tru, linh hoạt hơn.

**TÀI LIỆU THAM KHẢO**

[1] Phân tích và thiết kế giải thuật (2016) - Chủ biên: Nguyễn Thanh Bình. Đồng tác giả: Phạm Minh Tuấn, Đặng Thiên Bình. XB: Nhà xuất bản Giáo dục Việt Nam.

[2] Bài 6: STRUCT AND CLASS, Hoàng Thị Điệp, khoa công nghệ thông tin- đại học Công nghệ

<http://uet.vnu.edu.vn/~diepht/s12_int2202/lect06_struct_class.pdf>

[3]Bài giảng cấu trúc dữ liệu, Phan Chí Tùng, Khoa CNTT, ĐHBK – ĐHĐN, lưu

hành nội bộ.

[4] C file input / output

<https://www.programiz.com/c-programming/c-file-input-output>

**CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

**struct** Contact {

**char** name[100];

**char** phonenumber[100];

**char** address[100];

**char** province[100];

};

**char**\* trim(**char** s[]);// xoá các kí tự trắng, kí tự xuống dòng, kí tự chuyển tab dầu và cuối chuỗi

**struct** Contact\* readContactFromLocal(**const** **char**\* pathFile, **int**\* count, **struct** Contact\* contacts);//đọc dữ liệu từ file path

**void** addContactToLocalFile(**const** **struct** Contact\* contact, **const** **char**\* pathFile);// thêm 1 contact vào file path

**void** getListContact(**const** **char**\* path); // lấy danh sách contact và in ra màn hình

**void** printlistcontact(**struct** Contact\* contacts,**int** contactcount);//in danh sách các liên hệ ra màn hình

**void** findNumberPhoneDuplicate(**char**\* path,**struct** Contact \*contacts,**int** contactcount); // tìm kiếm các số điện thoại trùng nhau trong tỉnh

**void** getListContactByProvince(**char**\* path, **char**\* province,**struct** Contact \*contacts,**int** contactcount); //tìm kiếm và in ra các contact trong tỉnh tinh

**void** findContactByPhoneNumber(**char**\* path, **char**\* phoneNumber,**struct** Contact \*contacts,**int** contactcount); //tim kiem và in ra các contact có số điện thoại sdt

**void** findContactByAddress(**char**\* path, **char**\* Adress,**struct** Contact \*contacts,**int** contactcount); //tim kiem và in ra các contact có địa chỉ Address

**void** writetofile(**char**\* fileout,**struct** Contact\* contacts,**int** count);// in danh sach cac lien he ra file out

**void** xoa(**char**\* path,**struct** Contact Newcontact,**struct** Contact\* contacts, **int** \*contactcount);//xoa 1 lien he bang so dien thoai, ten hoac dia chi

**void** InsertContact(**struct** Contact NewContact, **int** \*contactcount,**struct** Contact\* contacts);// them contact nhập từ bàn phím

**int** main() {

**struct** Contact\* contacts;

**char** path[] = "phonenumber.txt";

**int** contactcount;

contacts = readContactFromLocal(path, &contactcount,contacts);

**int** optional;

**do** {

printf("\n\n\t\t DANH BẠ ĐIỆN THOẠI:\n");

printf("+");**for**(**int** i=0;i<60;i++) printf("-");printf("+\n");

printf("| %-59s|\n","1. Lấy danh sách contact trong file");

printf("| %-59s|\n","2. Tìm các số điện thoại trùng nhau và xóa");

printf("| %-59s|\n","3. Liệt kê danh bạ từng tỉnh");

printf("| %-59s|\n","4. Tìm kiếm liên hệ theo số điện thoại");

printf("| %-59s|\n","5. Tìm kiếm liên hệ theo địa chỉ");

printf("| %-59s|\n","6. Tìm kiếm liên hệ theo tên");

printf("| %-59s|\n","7. Thêm liên hệ");

printf("| %-59s|\n","8. Xóa liên hệ");

printf("| %-59s|\n","9. In danh sách liên hệ ra file phonenumberout.txt");

printf("| %-59s|\n","10. Thoát chương trình");

printf("+");**for**(**int** i=0;i<60;i++) printf("-");printf("+\n\n");

printf("Mời bạn nhập lựa chọn: ");

scanf("%d", &optional);

**if** (optional == 10) **break**;

**struct** Contact Newcontact;

**char** tinh[100];

**char** sdt[11];

**char** Address[100];

**char** infor[100];

**switch** (optional) {

**case** 1:

getListContact(path);

**break**;

**case** 2:

findNumberPhoneDuplicate(path,contacts,contactcount);

**break**;

**case** 3:

printf("Nhập tỉnh cần liệt kê danh bạ: ");

getchar(); // Để xóa bộ đệm nhập trước đó

fgets(tinh, **sizeof**(tinh), stdin);

strcpy(tinh,trim(tinh));

getListContactByProvince(path, tinh,contacts,contactcount);

**break**;

**case** 4:

printf("Nhập số điện thoại cần tìm kiếm: ");

getchar(); // Để xóa bộ đệm nhập trước đó

fgets(sdt, **sizeof**(sdt), stdin);

strcpy(sdt,trim(sdt));

findContactByPhoneNumber(path, sdt,contacts,contactcount);

**break**;

**case** 5:

printf("Nhập đia chỉ cần tìm kiếm: ");

getchar(); // Để xóa bộ đệm nhập trước đó

fgets(Address, **sizeof**(Address), stdin);

strcpy(Address,trim(Address));

findContactByAddress(path, Address,contacts,contactcount);

**break**;

**case** 6:

printf("Nhập tên liên lạc cần tìm kiếm: ");

getchar(); // Để xóa bộ đệm nhập trước đó

fgets(infor, **sizeof**(infor), stdin);

strcpy(infor,trim(infor));

findContactByAddress(path, infor,contacts,contactcount);

**break**;

**case** 9:

writetofile("phonenumberout.txt", contacts, contactcount);

printf("Danh sách đã được in ra file phonenumberout.txt\n");

free(contacts);

**break**;

**case** 8:

**struct** Contact NewContact;

getchar();

printf ("Nhập tên:");

fgets(NewContact.name,**sizeof**(NewContact),stdin);

strcpy(NewContact.name,trim(NewContact.name));

printf ("Nhập số điện thoại:");

fgets(NewContact.phonenumber,**sizeof**(NewContact),stdin);

strcpy(NewContact.phonenumber,trim(NewContact.phonenumber));

printf ("Nhập tỉnh:");

fgets(NewContact.province,**sizeof**(NewContact),stdin);

strcpy(NewContact.province,trim(NewContact.province));

printf ("Nhập địa chỉ:");

fgets(NewContact.address,**sizeof**(NewContact),stdin);

strcpy(NewContact.address,trim(NewContact.address));

xoa(path,NewContact,contacts,&contactcount);

**break**;

**case** 7:

getchar();

printf ("Nhập tên:");

fgets(NewContact.name,**sizeof**(NewContact),stdin);

strcpy(NewContact.name,trim(NewContact.name));

printf ("Nhập số điện thoại:");

fgets(NewContact.phonenumber,**sizeof**(NewContact),stdin);

strcpy(NewContact.phonenumber,trim(NewContact.phonenumber));

printf ("Nhập tỉnh:");

fgets(NewContact.province,**sizeof**(NewContact),stdin);

strcpy(NewContact.province,trim(NewContact.province));

printf ("Nhập địa chỉ:");

fgets(NewContact.address,**sizeof**(NewContact),stdin);

strcpy(NewContact.address,trim(NewContact.address));

contacts = (**struct** Contact\*)realloc(contacts, (contactcount+1) \* **sizeof**(**struct** Contact));

InsertContact(NewContact,&contactcount,contacts);

printlistcontact(contacts,contactcount);

**break**;

**case** 10:

**break**;

**default**:

printf("Vui long nhap cac so tu 1 den 8\n");

**continue**;

}

} **while** (getch() =='y');

**return** 0;

}

**char**\* trim(**char** s[]){

**while**(s[0]==' '||s[0]=='\n'||s[0]=='\t')

**for**(**int** i=0;i<strlen(s);i++) s[i]=s[i+1];

**while**(s[strlen(s)-1]==' '||s[strlen(s)-1]=='\n'||s[strlen(s)-1]=='\t') s[strlen(s)-1]='\0';

**return** s;

}

**struct** Contact\* readContactFromLocal(**const** **char**\* pathFile, **int**\* count,**struct** Contact\* contacts) {

FILE\* myFile = fopen(pathFile, "r");

**char** line[100];

\*count = 0;

contacts=**NULL**;

**if** (myFile == **NULL**) {

printf("Error opening file.\n");

**return** **NULL**;

}

**char** provinceName[100] = "";

**while** (fgets(line, **sizeof**(line), myFile)) {

**if** (strlen(line) == 0) **continue**;

**if** (strchr(line, '|') == **NULL**) {

strcpy(provinceName, line);

**continue**;

}

**char**\* address = strtok(line, "|");

**char**\* name = strtok(**NULL**, "|");

**char**\* phoneNumber = strtok(**NULL**, "|");

**struct** Contact contact;

strcpy(contact.address, trim(address));

strcpy(contact.name, trim(name));

strcpy(contact.phonenumber, trim(phoneNumber));

strcpy(contact.province, trim(provinceName));

(\*count)++;

contacts = (**struct** Contact\*)realloc(contacts, (\*count) \* **sizeof**(**struct** Contact));

contacts[(\*count) - 1] = contact;

}

fclose(myFile);

**return** contacts;

}

**void** addContactToLocalFile(**const** **struct** Contact\* contact, **const** **char**\* pathFile) {

FILE\* file = fopen(pathFile, "a");

**if** (file == **NULL**) {

printf("Cannot open file: %s\n", pathFile);

**return**;

}

**char** content[256];

sprintf(content, "%s|%s|%s\n", contact->address, contact->name, contact->phonenumber);

fputs(content, file);

fclose(file);

}

**void** getListContact(**const** **char**\* path) {

**int** count;

**struct** Contact\* contacts = readContactFromLocal(path, &count,contacts);

**if** (contacts != **NULL**) {

printlistcontact(contacts,count);

}

}

**void** printlistcontact(**struct** Contact\* contacts, **int** contactcount){

printf( "+--------------------|--------------------|-------------------------|--------------------+\n");

printf("|%-20s|%-20s|%-25s|%-20s|\n","PROVINCE","NAME","ADDRESS","PHONENUMBER");

printf( "+--------------------|--------------------|-------------------------|--------------------+\n");

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

printf("|%-20s|%-20s|%-25s|%-20s|\n",contacts[i].province,contacts[i].name,contacts[i].address,contacts[i].phonenumber);

}

printf( "+--------------------|--------------------|-------------------------|--------------------+\n");

**return**;

}

**void** deletecontact(**struct** Contact\* contacts,**int** \*contactcount,**int** x){

**for**(**int** i=x;i<\*contactcount;i++) contacts[i]=contacts[i+1];

(\*contactcount)--;

}

// Tìm số điện thoại trùng lặp theo tỉnh

**void** findNumberPhoneDuplicate(**char**\* path, **struct** Contact\* contacts, **int** contactcount) {

**for** (**int** x = 0; x < contactcount-1; ++x) {

**int** count = 0;

**int** listDuplicateContact[1000];

listDuplicateContact[0] = x;

**for** (**int** y = x + 1; y < contactcount; y++) {

**struct** Contact current = contacts[y];

**if** (strcmp(contacts[x].phonenumber, current.phonenumber) == 0 &&

strcmp(contacts[x].province, current.province) == 0) {

++count;

listDuplicateContact[count] = y;

}

}

**if** (count >= 1) {

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

printf("%d|%-20s|%-20s|%-25s|%-20s|\n",

listDuplicateContact[i], contacts[listDuplicateContact[i]].name,

contacts[listDuplicateContact[i]].phonenumber,

contacts[listDuplicateContact[i]].address, contacts[listDuplicateContact[i]].province);

}

printf("\nNhập số thứ tự contact bạn muốn xóa hoặc nhập -1 để không xóa\n");

**int** optional;

scanf("%d", &optional);

deletecontact(contacts, &contactcount, optional);

}

}

printlistcontact(contacts, contactcount);

}

**void** getListContactByProvince(**char**\* path, **char**\* province,**struct** Contact \*contacts,**int** contactcount) {

**int** found = 0;

**struct** Contact\* list=**NULL**;

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

**if**(strcmp(contacts[i].province,province)==0){

list = (**struct** Contact\*)realloc(list, (found+1) \* **sizeof**(**struct** Contact));

list[found] = contacts[i];

found++;

}

}

**if**(!found) printf("Không có contact nào trong %s",province);

**else**{

printf("Danh sách các contact trong tỉnh %s:\n", province);

printlistcontact(list,found);

}

}

**void** findContactByName(**char**\* path, **char**\* infor,**struct** Contact \*contacts, **int** contactcount){

**struct** Contact\* list;

**int** found = 0;

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

**if** (strcmp(contacts[i].name, infor) == 0) {

list = (**struct** Contact\*)realloc(list, (found+1) \* **sizeof**(**struct** Contact));

list[found] = contacts[i];

found ++;

}

}

**if** (!found) {

printf("Không tìm thấy contact có tên %s\n", infor);

}

**else**{

printf("các contact có tên là %s la:\n", infor);

printlistcontact(list,found);

}

}

**void** findContactByPhoneNumber(**char**\* path, **char**\* phoneNumber,**struct** Contact \*contacts, **int** contactcount){

**struct** Contact\* list;

**int** found = 0;

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

**if** (strcmp(contacts[i].phonenumber, phoneNumber) == 0) {

list = (**struct** Contact\*)realloc(list, (found+1) \* **sizeof**(**struct** Contact));

list[found] = contacts[i];

found ++;

}

}

**if** (!found) {

printf("Không tìm thấy contact co so dien thoai %s\n", phoneNumber);

}

**else** {

printf("Các contact có số điện thoại %s là:\n",phoneNumber);

printlistcontact(list,found);

}

}

**void** findContactByAddress(**char**\* path, **char**\* Address,**struct** Contact \*contacts,**int** contactcount){

**struct** Contact\* list;

**int** found = 0;

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

**if** (strcmp(contacts[i].address, Address) == 0) {

list = (**struct** Contact\*)realloc(list, (found+1) \* **sizeof**(**struct** Contact));

list[found] = contacts[i];

found ++;

}

}

**if** (!found) {

printf("Không tìm thấy contact co địa chỉ %s\n", Address);

}

**else** {

printf("Các contact có địa chỉ %s là :\n",Address);

printlistcontact(list,found);

}

}

**void** writetofile(**char**\* fileout, **struct** Contact\* contacts, **int** count) {

FILE\* f = fopen(fileout, "w");

**if** (f == **NULL**) {

printf("Cannot open file: %s\n", fileout);

**return**;

}

fprintf(f, "+-------------------------|-------------------------|-------------------------|-------------------------+\n");

fprintf(f, "|%-25s|%-25s|%-25s|%-25s|\n", "PROVINCE", "NAME", "ADDRESS", "PHONENUMBER");

fprintf(f, "+-------------------------|-------------------------|-------------------------|-------------------------+\n");

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

fprintf(f, "|%-25s|%-25s|%-25s|%-25s|\n", contacts[i].province, contacts[i].name, contacts[i].address, contacts[i].phonenumber);

}

fprintf(f, "+-------------------------|-------------------------|-------------------------|-------------------------+\n");

fclose(f);

}

**void** xoa(**char**\* path,**struct** Contact Newcontact,**struct** Contact\* contacts, **int** \*contactcount){

**int** found=0;

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

**if**(strcmp(contacts[i].address,Newcontact.address)==0 && strcmp(contacts[i].name,Newcontact.name)==0 && strcmp(contacts[i].phonenumber,Newcontact.phonenumber)==0 && strcmp(contacts[i].province,Newcontact.province)==0){

deletecontact(contacts,contactcount,i);

found = 1;

}

}

**if**(found==0) printf("liên hệ không có trong danh sách\n");

**else** printf ("Đã xóa liên hệ thành công");

printlistcontact(contacts,\*contactcount);

}

**void** InsertContact(**struct** Contact NewContact, **int** \*contactcount, **struct** Contact\* contacts) {

**int** i, j;

// Tìm vị trí chèn mới theo thứ tự tên

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

**if** (strcmp(NewContact.province, contacts[i].province) == 0)

**break**;

}

// Dịch chuyển các phần tử phía sau để tạo vị trí cho liên hệ mới

**for** (j = \*contactcount; j > i; j--) {

contacts[j] = contacts[j - 1];

}

// Chèn liên hệ mới vào vị trí tìm được

contacts[i] = NewContact;

(\*contactcount)++;

printf ("Đã thêm liên hệ thành công");

**return**;

}