**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH**

**🙢🙢🙢∙🙠🙠🙠**

**BÁO CÁO**

**GIỚI THIỆU CHƯƠNG TRÌNH QUẢN LÝ NHÂN VIÊN**

****

**PHÙNG ĐĂNG KHOA – HỒ THỊ KIỀU LINH – NGUYỄN TUẤN KIỆT**

**PHẦN MỞ ĐẦU**

**GIỚI THIỆU VỀ NGÔN NGỮ LẬP TRÌNH C**

1. **Ngôn ngữ lập trình C:**

C là một [ngôn ngữ lập trình](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh) tương đối nhỏ gọn vận hành gần với phần cứng. Hơn thế, C đôi khi được đánh giá như là "có khả năng di động", vì mã C có thể được dịch và chạy trong hầu hết các máy tính, hơn hẳn các ngôn ngữ hiện tại trong khi ngôn ngữ bậc thấp như [Assembler](https://vi.wikipedia.org/wiki/H%E1%BB%A3p_ng%E1%BB%AF) chỉ có thể chạy trong một số máy tính đặc biệt. Vì vậy C được xem là [ngôn ngữ bậc trung](https://vi.wikipedia.org/w/index.php?title=Ng%C3%B4n_ng%E1%BB%AF_b%E1%BA%ADc_trung&action=edit&redlink=1).

Ngôn ngữ lập trình C là một ngôn ngữ lập trình hệ thống rất mạnh và “mềm dẻo”, có thư viện gồm rất nhiều các hàm (function) được tạo sẵn. Lập trình viên có thể tận dụng các hàm này để giải quyết các bài toán mà nhanh chóng. Bên cạnh đó, ngôn ngữ C hỗ trợ rất nhiều phép toán phù hợp cho việc giải quyết các bài toán phức tạp.Tuy nhiên, người mới tiếp cận lập trình C thường gặp khó khăn bởi sự “mềm dẻo” của C. Dù vậy, C vẫn được sử dụng rộng rãi và trở thành một công cụ lập trình mạnh, được sử dụng như một ngôn ngữ lập trình chủ yếu trong việc xây dựng phần mềm hiện nay.

1. **Các thao tác làm việc với tệp trong C:**

***⮘Thao tác chuẩn:*** lập trình viên không cần biết quá trình thực hiện việc thao tác với file như thế nào. Đó là việc của hệ thống.

***⮘Thao tác mức hệ thống****:* thao tác file thông qua bộ đệm. Lập trình viên phải tự quản lý các bộ đệm đọc ghi file. Thao tác file này gần giống với cách thao tác file của hệ điều hành MS – DOS. Thông thường chỉ có những người lập trình hệ thống mới sử dụng thao tác file mức hệ thống.

🞟Thao tác files

* Khai báo biến tập tin.
* Mở tập tin bằng hàm fopen().
* Thực hiện các thao tác xử lý dữ liệu của tập tin bằng các hàm đọc/ghi dữ liệu.
* Đóng tập tin bằng hàm fclose().

Ở đây, ta thao tác với tập tin nhờ các hàm được định nghĩa trong thư viện stdio.h.

1. **Khai báo biến tập tin.**

* **Cú pháp thực hiện**

FILE <Danh sách các biến con trỏ, đại diện cho tập tin>

Thao tác file chuẩn  **FILE \* f;**

Thao tác mức hệ thống**int f;**

Dữ liệu trên file là một dãy các byte(8 bit) cógiá trị từ 0 đến 255. Số byte của dãy là kích thước thật của file(size on disk).

Các biến trong danh sách phải là các con trỏ và được phân cách bởi dấu phẩy(,).  Ví dụ: FILE \*f1,\*f2;

1. **Mở tập tin.**

* **Cú pháp**

FILE \*fopen(char \*Path, const char \*Mode)

Trong đó:

* Path: chuỗi chỉ đường dẫn đến tập tin trên đĩa.
* Mode: chuỗi xác định cách thức mà tập tin sẽ mở. Các giá trị có thể của Mode:

|  |  |
| --- | --- |
| Chế độ | Ý nghĩa |
| r | Mở tập tin văn bản để đọc. |
| w | Tạo ra tập tin văn bản mới để ghi. |
| a | Nối vào tập tin văn bản. |
| rb | Mở tập tin nhị phân để đọc. |
| wb | Tạo ra tập tin nhị phân để ghi. |
| ab | Nối vào tập tin nhị phân. |
| r+ | Mở tập tin văn bản để đọc, ghi. |
| w+ | Tạo ra tập tin văn bản để đọc, ghi. |
| a+ | Nối vào hay tạo mới tập tin văn bản để đọc/ghi |
| r+b | Mở ra tập tin nhị phân để đọc/ghi |
| w+b | Tạo ra tập tin nhị phân để đọc/ghi |
| a+b | Nối vào hay tạo mới tập tin nhị phân |

Mặc định là mở dạng text nếu không có xác định là b, nếu rõ ràng hơn thì thêm chỉ định t để xác định là kiểu text.

Hàm fopen trả về một con trỏ tập tin. Chương trình của ta không thể thay đổi giá trị của con trỏ này. Nếu có một lỗi xuất hiện trong khi mở tập tin thì hàm này trả về con trỏ NULL.

Ví dụ: Mở một tập tin tên TEST.txt để ghi.

FILE \*f;

f = fopen("TEST.txt", "w"); {

/\* Các câu lệnh để thao tác với tập tin\*/

/\* Đóng tập tin\*/

}

1. **Các thao tác xử lý số liệu.**

∙Cú pháp: int fclose(FILE \*f)

Trong đó f là con trỏ tập tin được mở bởi hàm fopen(). Giá trị trả về của hàm là 0 báo rằng việc đóng tập tin thành công. Hàm trả về EOF nếu có xuất hiện lỗi.

Ngoài ra, ta còn có thể sử dụng hàm fcloseall() để đóng tất cả các tập tin lại.

∙Cú pháp: int fcloseall()

Kết quả trả về của hàm là tổng số các tập tin được đóng lại. Nếu không thành công, kết quả trả về là EOF.

**1 Tệp đĩa và I / O khác**.

Nếu không có tệp, chương trình của bạn không thể làm gì. Nó sẽ không có đầu vào, không thểcung cấp đầu ra và không thể lưu kết quả của bất kỳ tính toán nào của nó. May mắn thay, ANSI C cung cấp một bộ sưu tập tuyệt vời các chức năng I / O của tệp. Bạn có thể ghi vào một tệp trong một cách không được định dạng (máy có thể đọc được) hoặc chương trình của bạn có thể ghi vào một tệp trong một được định dạng (mọi người có thể đọc được) theo cách. Chương này đề cập đến tất cả các loại I / O: đến và đi từ tệp, bảng điều khiển và các thiết bị khác. Bạn học tất cả (ít nhất là tôi hy vọng tất cả) mà bạn sẽ cần để phát triển I / O các phần của chương trình của bạn. Phần đầu tiên của chương này đề cập đến các tệp đĩa. Các phần thứ hai bao gồm I / O bảng điều khiển và I / O cổng trực tiếp, bao gồm I / O đến các cổng máy in và các cổng thông tin liên lạc. Phần lớn cổng I / O trực tiếp phụ thuộc vào phần cứng; các thảo luận áp dụng cho hệ thống dựa trên PC.

1. **Khái niệm cơ bản về tệp I / O**

Phần này không thảo luận về cách các tệp được sắp xếp, lưu trữ và quản lý trên ổ đĩa. Đó là một chủ đề cho một cuốn sách về hệ điều hành, không phải một cuốn sách về lập trình C. Phần này bao gồm cách sử dụng tệp. Phần quan trọng nhất của hầu hết các chương trình là khả năng lưu kết quả hoặc dữ liệu họ sử dụng (chẳng hạn như cơ sở dữ liệu) vào tệp đĩa. Tất cả các tệp đĩa chia sẻ mọi thứ trong com-mon. (Một lần nữa, cuộc thảo luận này chỉ giới hạn trong hệ điều hành DOS của PC.)

Tất cả các tệp đều có tên. Định dạng cho tên tệp đĩa là tám ký tự cho tên và ba ký tự cho phần mở rộng. Việc lựa chọn tên tệp vàtiện ích mở rộng thường được để lại cho người dùng, ngoại trừ các tệp mà các ứng dụng khác sử dụng. Nhiều ứng dụng đề xuất một tiện ích mở rộng (một số yêu cầu nó), nhưng ít chương trình đặt hạn chế về tên.

Tất cả các tệp đều có ngày và giờ tạo, được lưu trong tem thời gian. Các dấu thời gian được cập nhật mỗi khi bạn lưu các thay đổi vào tệp, do đó, nó sẽ đóng vai trò là một tệpdấu thời gian sửa đổi.

Tất cả các tệp đều có độ dài. Độ dài này nằm trong cấu trúc của hệ điều hành được lưu chomỗi tệp. Khi một tệp chỉ bao gồm văn bản, nó cũng có thể chứa một điểm đánh dấu EOF (cuối tệp), trong DOS là 0x1A (Ctrl-Z).

Tất cả các tệp cũng có các thuộc tính, như sau:

Bình thường: không có thuộc tính đặc biệt nào được đặt cho tệp.

Thư mục Tệp: là một thư mục. Thuộc tính thư mục có thể được sử dụng vớithuộc tính ẩn.

Ẩn: tên của tệp không được hiển thị khi bạn phát hành DIR com-bắt buộc không có tùy chọn / A: H.

Hệ thống Tệp: chỉ được sử dụng bởi hệ điều hành. Nói chung, chỉhai tệp thuộc hệ điều hành có hệ thốngthuộc tính.

Chỉ đọc Tập tin:chỉ có thể được đọc, không được ghi hoặc bị xóa.

Lưu trữ Tệp: chưa được sao lưu kể từ lần thay đổi cuối cùng.BACKUP và XCOPY có thể sử dụng và thay đổi thuộc tính lưu trữ.

Bạn có thể chỉ định thuộc tính chỉ đọc khi tạo tệp. Tất cả các phụ tùng khác-butes phải được đặt (hoặc xóa) bằng lệnh DOS ATTRIB. Bạn sử dụng hệ thống ()chức năng gọi lệnh ATTRIB từ các ứng dụng DOS.Khi một tệp được mở, chương trình thường phải chỉ định tên tệp và chế độ: đọc, ghi hoặc cả đọc và ghi. Nếu tệp đang được tạo, chương trình cũng có thể chỉ định liệu tệp sẽ chỉ được đọc sau khi nó được tạo.

Các hàm đang mở trả về một mã định danh là một trình xử lý tệp hoặc một con trỏ đến đã mở tệp. Bạn sử dụng mã định danh này khi bạn gọi các hàm đọc và ghi. Khi nàochương trình đã kết thúc với tệp, tệp sẽ được đóng lại hoặc nếu đó là tệp tạm thờitệp công việc đã bị xóa.

1. **Tệp văn bản và tệp nhị phân**

Nếu một tệp văn bản được hiển thị trên màn hình, nó có thể đọc được. Nó không nên chứa bất kỳ điều gì đặc biệt các ký tự khác với tab và dòng mới và thường được định dạng để cung cấp thông tin cho người dùng một cách có tổ chức. Tuy nhiên, chương trình, mặc dù các tệp hoàn toàn có thể đọc được (và thậm chí có thể có thể hiểu được). Một tệp nhị phân có thể chứa bất kỳ dữ liệu nào, bao gồm cả tệp đại diện bên trong-tation của số, ký tự điều khiển đặc biệt.

Sự cố phát sinh khi bạn sử dụng tệp văn bản, ngôn ngữ C và giao diện của C đểDOS. C chỉ định một ký tự (được gọi là ký tự dòng mới) để biểu thị sự kết thúc củamột dòng. DOS sử dụng hai ký tự (một ký tự dòng mới theo sau là ký tự xuống dòngký tự) để biểu thị sự kết thúc của một dòng. Mỗi là một ký tự điều khiển, và như vậy, phải được chỉ định bằng cách sử dụng chuỗi thoát ANSI C, bắt đầu bằng ký tự gạch chéo ngược. Để chỉ định một dòng mới, bạn sử dụng chuỗi ký tự \ n. Khi C gặpký tự dòng mới trong một chuỗi được xuất ra DOS (thành tệp hoặc trên màn hình),C chuyển đổi nó thành chuỗi hai ký tự mà DOS mong đợi (một dòng mới theo sau làdấu xuống dòng). Khi đầu vào được đọc, C làm ngược lại, chuyển đổi dòng mớivà cặp ký tự xuống dòng thành một ký tự dòng mới. Điều này tạo ramột vấn đề nhỏ. Khi chương trình đọc một số lượng cụ thể byte trong tệp chế độ văn bản, mỗi khi gặp phải cặp dòng mới và ký tự xuống dòng-ed, số lượng ký tự chỉ tăng lên 1 (chỉ một ký tự dòng mới làtính). Nếu chương trình của bạn đọc chuỗi từ một tệp, như thể hiện trong Hình 9.1, chuỗichỉ dài 29 ký tự, không phải 31 ký tự được lưu trữ trong tệp.

Vấn đề nảy sinh khi chương trình phải đi (tìm kiếm) đến một nơi cụ thể trong tập tin. Bạn không thể giả định có bao nhiêu ký tự (hoặc byte) giữa một điểm đã cho và điểm mong muốn. Nếu bạn đã viết một tệp chứa năm chuỗi khác nhau, mỗi chuỗi dài 50 ký tự (và chứa một số ký tự dòng mới không xác định), chẳng hạn như không nhận được đầu của chuỗi thứ tư, bằng cách tìm kiếm ký tự số 150.

Để tìm kiếm thành công trong tệp văn bản, bạn phải tạo chỉ mục cho mỗi chuỗi (hoặcghi lại). Chỉ mục này được gán giá trị bằng cách sử dụng một trong các hàm trả vềvị trí hiện tại trong tệp, chẳng hạn như ftell () hoặc fgetpos (). Các hàm này trả về giá trị chính xác vị trí của một đối tượng dữ liệu nhất định, có tính đến việc chuyển đổi ký tự dòng đến một dòng mới và cặp ký tự xuống dòng. Bạn phải lưu chỉ mục này để có thể truy cập các chuỗi một cách ngẫu nhiên. Al- duy nhất điều quan trọng để lưu một chỉ mục là đọc tệp văn bản một cách tuần tự, điều này có thể không có thể chấp nhận được dựa trên các cân nhắc về hiệu suất. Chương trình TEXTFILE.C, được hiển thị trong Liệt kê 9.1, cho thấy các hiệu ứng của văn bản.

1. **Cấu trúc liên kết đơn.**
   1. **Khái niệm**

Cấu trúc liên kết đơn (single linked list) là một cấu trúc dữ liệu tuyến tính. Tuy nhiên khác với mảng (array), các phần tử của cấu trúc liên kết đơn không được lưu trữ tại các vị trí ô nhớ liền kề nhau, mà các phần tử của cấu trúc liên kết đơn sẽ được liên kết với nhau bằng cách sử dụng các con trỏ.

* 1. **Ưu điểm**

- Linked List có kích thước động

- Dễ dàng chèn thêm phần tử vào mảng, và xóa phần tử khỏi mảng.

* 1. **Nhược điểm**

- Không thể thực hiện truy cập ngẫu nhiên (random access). Chúng ta phải truy cập đến các phần tử của cấu trúc liên kết đơn một cách tuần tự, bắt đầu từ nút đầu tiên. Vì vậy chúng ta sẽ không thể thực hiện tìm kiếm nhị phân với cấu trúc liên kết đơn một cách hiệu quả, đối với dạng cài đặt mặc định của nó.

- Mỗi phần tử của cấu trúc đều chứa một con trỏ (pointer) và cần phải cấp phát bộ nhớ cho các con trỏ này.

- Cấu trúc liên kết đơn không thân thiện với bộ nhớ cache. Bởi vì các phần tử trong mảng được bố trí nằm liền kề, liên tiếp nhau, nên chúng ta có thể dễ dàng truy cập đến các phần tử của mảng thông qua các vị trí tham chiếu được thể hiện bằng các chữ số, trong khi đó điều này là không thể đối với cấu trúc liên kết đơn.

* 1. **Mô tả bằng code**

-Một danh sách liên kết đơn sẽ được biểu diễn bởi một con trỏ (pointer) trỏ đến node đầu tiên của danh sách liên kết đơn đó. Node đầu tiên của nó được gọi là node head. Nếu Linked List là trống, giá trị của node head sẽ là NULL.

-Mỗi node ở bên trong một cấu trúc liên kết đơn sẽ bao gồm ít nhất hai thành phần sau:

- Data (dữ liệu, có thể là kiểu số, kiểu ký tự, …)

- Pointer (con trỏ) hoặc có thể hiểu là tham chiếu tới node tiếp theo trong cấu trúc liên kết đơn.

🞼Trong ngôn ngữ lập trình C, chúng ta có thể biểu diễn một node bằng cách sử dụng kiểu dữ liệu struct.

struct Node {

int data;

struct Node\* next;

};

1. **Các thuật toán dùng để sắp xếp trong C**.
2. **Sắp xếp nổi bọt trong C**

Sắp xếp nổi bọt (Bubble Sort) là một giải thuật sắp xếp đơn giản. Giải thuật sắp xếp này được tiến hành dựa trên việc so sánh cặp phần tử liền kề nhau và tráo đổi thứ tự nếu chúng không theo thứ tự.

void Hoanvi(int &a,int &b)

{

c=a;

a=b;

b=c;

}

void Sapxep(int a[], int n)

{

for (int i=0 ; i<n-2 ; i++)

{

for (int j= 0; j< n-1;j++)

{

if ( a[j] < a[j])

Hoanvi(a,b);

}

}

}

1. **Sắp xếp chọn trong C**

Giải thuật sắp xếp chọn (Selection Sort) là một giải thuật đơn giản. Giải thuật sắp xếp này là một giải thuật dựa trên việc so sánh **in-place**, trong đó danh sách được chia thành hai phần, phần được sắp xếp (sorted list) ở bên trái và phần chưa được sắp xếp (unsorted list) ở bên phải. Ban đầu, phần được sắp xếp là trống và phần chưa được sắp xếp là toàn bộ danh sách ban đầu..

voidSapxepchon(inta[], intn)

{

    inti, j, min\_idx;

    for(i = 0; i < n-1; i++)

    {

    min\_idx = i;

    for(j = i+1; j < n; j++)

        if(a[j] < a[min\_idx])

        min\_idx = j;

          swap(a[min\_idx], a[i]);

    }

}

1. **Sắp xếp chèn trong C**

Sắp xếp chèn (Insertion Sort) là một giải thuật sắp xếp dựa trên so sánh in-place. Ở đây, một danh sách con luôn luôn được duy trì dưới dạng đã qua sắp xếp. Sắp xếp chèn là chèn thêm một phần tử vào danh sách con đã qua sắp xếp. Phần tử được chèn vào vị trí thích hợp sao cho vẫn đảm bảo rằng danh sách con đó vẫn sắp theo thứ tự

voidSapxepchen(inta[], intn)

{

   inti, key, j;

   for(i = 1; i < n; i++)

   {

       key = a[i];

       j = i-1;

                while(j >= 0 && arr[j] > key)

       {

           a[j+1] = a[j];

           j = j-1;

       }

       a[j+1] = key;

   }

}

1. **Sắp xếp trộn trong C**

Sắp xếp trộn (Merge Sort) là một giải thuật sắp xếp dựa trên giải thuật **Chia để trị (Divide and Conquer)**. Với độ phức tạp thời gian trường hợp xấu nhất là Ο(n log n) thì đây là một trong các giải thuật đáng được quan tâm nhất.

Đầu tiên, giải thuật sắp xếp trộn chia mảng thành hai nửa và sau đó kết hợp chúng lại với nhau thành một mảng đã được sắp xếp.

voidmerge(inta[], intl, intm, intr)

{

    inti, j, k;

    intn1 = m - l + 1;

    intn2 =  r - m;

    intL[n1], R[n2];

    for(i = 0; i < n1; i++)

        L[i] = a[l + i];

    for(j = 0; j < n2; j++)

        R[j] = a[m + 1+ j];

    i = 0;

    j = 0;

    k = l;

    while(i < n1 && j < n2)

    {

        if(L[i] <= R[j])

        {

            a[k] = L[i];

            i++;

        }

        else

        {

            a[k] = R[j];

            j++;

        }

        k++;

    }

    while(i < n1)

    {

        a[k] = L[i];

        i++;

        k++;

    }

    while(j < n2)

    {

        a[k] = R[j];

        j++;

        k++;

    }

}

voidmergeSort(inta[], intl, intr)

{

    if(l < r)

    {

        intm = l+(r-l)/2;

        mergeSort(a, l, m);

        mergeSort(a, m+1, r);

        merge(a, l, m, r);

    }

}

1. **Sắp xếp nhanh trong C**

Giải thuật sắp xếp nhanh (Quick Sort) là một giải thuật hiệu quả cao và dựa trên việc chia mảng dữa liệu thành các mảng nhỏ hơn. Giải thuật sắp xếp nhanh chia mảng thành hai phần bằng cách so sánh từng phần tử của mảng với một phần tử được chọn gọi là **phần tử chốt (Pivot)**: một mảng bao gồm các phần tử nhỏ hơn hoặc bằng phần tử chốt và mảng còn lại bao gồm các phần tử lớn hơn hoặc bằng phần tử chốt

voidswap(int&a, int&b)

{

    intt = a;

    a = b;

    b = t;

}

voidquickSort(inta[], intlow, inthigh)

{

    if(low < high)

    {

            intindex = partition(arr, low, high);

         quickSort(a, low, index - 1);

         quickSort(a, index + 1, high);

    }

}

intpartition (inta[], intlow, inthigh)

{

    intpivot = a[high];

 intleft = low;

    intright = high - 1;

    while(true){

        while(left <= right && a[left] < pivot) left++;

        while(right >= left && a[right] > pivot) right-;

   if(left >= right) break;

        swap(a[left], a[right]);

        left++;

        right--;

    }

    swap(a[left], a[high]);

    returnleft;

}

* 1. **Shell Sort trong C**
  2. **Shell Sort trong C**

Shell Sort là một giải thuật sắp xếp mang lại hiệu quả cao dựa trên giải thuật **sắp xếp chèn (Insertion Sort)**. Giải thuật này tránh các trường hợp phải tráo đổi vị trí của hai phần tử xa nhau trong giải thuật sắp xếp chọn (nếu như phần tử nhỏ hơn ở vị trí bên phải khá xa so với phần tử lớn hơn bên trái).

Đầu tiên, giải thuật này sử dụng giải thuật sắp xếp chọn trên các phần tử có khoảng cách xa nhau, sau đó sắp xếp các phần tử có khoảng cách hẹp hơn. Khoảng cách này còn được gọi là **khoảng (interval)** – là số vị trí từ phần tử này tới phần tử khác. Khoảng này được tính dựa vào công thức Knuth như sau:

Bắt đầu hàm shellSort()

    A : mảng các phần tử

   /\* Tính toán giá trị Khoảng (interval)\*/

   whileinterval < A.length /3 thực hiện:

      interval = interval \* 3 + 1

   kết thúc while

   whileinterval > 0 thực hiện:

   forouter = interval; outer < A.length; outer ++ thực hiện:

   /\* chọn giá trị để chèn \*/

      valueToInsert = A[outer]

      inner = outer;

      /\*dịch chuyển phần tử sang phải\*/

         whileinner > interval -1 && A[inner - interval] >= valueToInsert

         do:

            A[inner] = A[inner - interval]

            inner = inner - interval

         kết thúc while

         /\* chèn giá trị vào vị trí trên \*/

      A[inner] = valueToInsert

      kết thúc for

      /\* Tính toán giá trị Khoảng (interval)\*/

   interval = (interval -1) /3;

   kết thúc while

Kết thúc hàm

**Thuật toán tìm kiếm trong C**

**Tìm kiếm tuần tự**

Là thuật toán tìm kiếm tự nhiên và cơ bản nhất mà ai cũng nghĩ ra và cài đặt bằng C.

#include <stdio.h>

int search(int arr[], int n, int x)

{

int i;

for (i = 0; i < n; i++)

if (arr[i] == x)

return i;

return -1;

}

int main(void)

{

int arr[] = { 2, 3, 4, 10, 40 };

int x = 10;

int n = sizeof(arr) / sizeof(arr[0]);

int result = search(arr, n, x);

(result == -1) ? printf("Element is not present in array")

: printf("Element is present at index %d",

result);

return 0;

}

1. **Tìm kiếm nhị phân**

Là một thuật toán tìm kiếm rất nhanh so với tìm kiếm tuần tự, đặc biệt là trong danh sách lớn.

#include <bits/stdc++.h>

usingnamespacestd;

//Hàm tìm kiếm nhi phân

intbinarySearch(intarr[], intleft, intright, intx) {

    intmiddle;

    while(left <= right) {

        // Lấy vị trí ở giữa left và right

        middle = (left + right) / 2;

        // Nếu phần từ ở giữa bằng x thì trả về

        // vị trí

        If (arr[middle] == x)

            returnmiddle;

        // Nếu x lớn hơn phần tử ở giữa thì

        // chắc chắn nó nằm bên phải.

        // Chỉ định left phần từ ở giữa + 1

        if(x > arr[middle])

            left = middle + 1;

    else

            //Ngược lại

            right = middle - 1;

    }

//Trả về -1 nếu không tìm thấy gía trị trong mảng.

    return-1;

--------------------------------------------------------------------------------------------------------

**PHẦN 2**

**CHƯƠNG TRÌNH QUẢN LÝ SINH VIÊN**

1. **Lý do chọn đề tài**

Trong thời đại ngày càng tiên tiến như hiện nay thì việc phát triển trong lĩnh vực khoa học và kỹ thuật đóng vai trò vô cùng quan trọng trong việc phát triển kinh tế xã hội. Trong đó, ngành công nghệ thông tin cũng không ngoại lệ, có thể nói rằng công nghệ thông tin đang trở nên vô cùng cần thiết và dần chiếm lĩnh mọi lĩnh vực trong đời sống mà phổ biến nhất là trong việc quản lý các hệ thống dữ liệu. Việc hệ số hóa quản lý giúp cho công việc này có thể đảm bảo dộ chính xác cao hơn, ít tiêu tốn thời gian hơn, đồng thời nâng cao mức độ hiệu quả của công việc. Đặc biệt là trong những công ty cũng như các tập đoàn trong công tác quản lý mà cụ thể chính là đối với việc quản lý số lượng nhân viên lên đến con số hàng trăm hay thậm chí là hàng ngàn thì việc quản lý bằng cách ứng dụng công nghệ vào thực tiễn là vô cùng cần thiết. Do đó, nhóm chúng tôi quyết định chọn đề tài này: xây dựng chương trình quản lý nhân viên dựa trên ngôn ngữ lập trình C.

1. **Thiết kế chương trình**

Chương trình được xây dựng bằng ngôn ngữ lập trình C. với việc sử dụng các hàm, các câu lệnh, kiến thức đã được đào tạo và thục hành cùng với sự tìm hiểu và nghiên cứu các nguồn tài liệu tham khảo khác nhau để có thể đạt được mục tiêu chính là xây dựng nên một chương trình hoàn chỉnh. Chương trình cho phép người dùng có thể thực hiện các thao tác như thêm, xem, sửa, tìm kiếm danh sách khi cần thiết.

Chương trình sử dung mảng (struct employee\_st ) để lưu trữ các phần tử nhập vào. Sau khi nhập vào danh sách nhân viên chương trình sẽ cho phép người dùng thực hiện các chức năng sau:

***Nhập danh sách nhân viên***

Cho phép người dùng nhâp vào các thông tin nhân viên bao gồm: họ tên nhân viên, tên tỉnh, năm sinh, lương …

***Xuất ra danh sách nhân viên***

In ra danh sách bao gồm tất cả nhân viên đã lưu trong mảng, bao gồm cả trường hợp mở file có sẵn.

***Sắp xếp danh sách theo tên tỉnh***

Sắp xếp lại danh sách được lưu trong bộ nhớ theo tên tỉnh (Z->A) phục vụ việc sắp xếp nhân viên.

***Thêm vào danh sách***

Chức năng này cho phép người dùng nhập vào vị trí mình muốn thêm vào sau đó xuất danh sách nhân viên đã thêm vào ra màn hình.

***Xóa nhân viên***

Xóa nhân viên ở vị trí bạn muốn và in danh sách đã xóa ra màn hình.

***Tìm nhân viên theo tên***

Chức năng này cho phép tìm kiếm nhân viên theo tên

Tìm thấy sẽ hiện ra danh sách, neeud không tìm thấ sẽ hiện thị thông báo ra màn hình.

***Ghi file vào tệp nhị phân***

Lưu danh sách đã nhập vào file và in thông báo lưu file ra màn hình.

***Thoát chương trình***

1. **Hướng dẫn sử dụng chương trình**

Người dùng chỉ cần chạy chương trình bằng cách nhấn nút Run hoặc F11 trên bàn phím. Sau đó nhấn các nút theo sự hướng dẫn của menu có trong chương trình.

Nhấn 1 để thêm danh sách.

Nhấn 2 để xem danh sách.

Nhấn 3 để sắp xếp danh sách nhân viên theo tỉnh.

Nhấn 4 để thêm nhân viên.

Nhấn 5 để xóa nhân viên.

Nhấn 6 để tìm kiếm nhân viên theo tên.

Nhấn 7 để ghi file vào tệp nhị phân.

Nhấn 8 để thoát khỏi chương trình.

Vây chỉ với những thao tác đơn giản, mỗi người có thể quản lý nhân viên một cách dễ dàng, nhanh chóng mà hiệu quả.

1. **Các kiến thức được ứng dụng vào trong chương trình.**

**Hàm**

Khái niệm: hàm là một đoạn chương trình con thực hiện một công việc nhất định.

Vd:hàm nhập giá trị một mảng

void Nhapmang( int a[],int &n)

{

printf("Nhap n=");

scanf("%d",&n);

for(int i=0; i<n;i++)

{

printf ("Nhap a[%d]=",i);

scanf("%d",&a[i]);

}

}

**Kiểu dữ liệu struct**

Khái niệm:Struct là một kiểu dữ liệu có cấu trúc, được kết hợp từ các kiểu dữ liệu nguyên thuỷ do người lập trình định nghĩa để thuận tiện trong việc quản lý dữ liệu và lập trình.

Vd:

structStudent {

    charname[20];

    intid;

    intage;

}

**Câu lệnh switch case**

Khái niệm: **Lệnh switch case** là một cấu trúc điều khiển & rẽ nhánh hoàn toàn có thể được thay thế bằng [cấu trúc if else](https://nguyenvanhieu.vn/cau-lenh-if-else-trong-c/).

Vd:

**Câu lệnh goto**

Khái niệm: Lệnh goto cho phép code của bạn nhảy đến thực hiện ở vị trí bất kỳ của chương trình mà không cần nhất định phải theo thứ tự từ trên xuống.

**5. Chương trình quản lí nhân viên:**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

struct employee\_st{

char name[30];

char province[30];

int year;

float luong;

float Giolamviec;

float Luongtrongmotgio;

float Giokhonglamviec;

float Luongbitru;

};

void Nhap(employee\_st &e){

printf("Nhap ten cua nhan vien : ");

fflush(stdin);

gets(e.name);

printf("Nhap tinh : ");

gets(e.province);

fflush(stdin);

printf("Nhap nam sinh : ");

scanf("%d",&e.year);

fflush(stdin);

printf("Nhap so gio lam viec: ");

scanf("%f",&e.Giolamviec);

printf("Nhap luong duoc tra trong mot gio: ");

scanf("%f",&e.Luongtrongmotgio);

printf("Nhap so gio khong lam viec: ");

scanf("%f",&e.Giokhonglamviec);

printf("Nhap luong bi tru trong mot gio: ");

scanf("%f",&e.Luongbitru);

e.luong=e.Giolamviec\*e.Luongtrongmotgio-e.Giokhonglamviec\*e.Luongbitru;

}

void NhapDS(employee\_st a[], int &n){

for (int i=0;i<n;i++){

printf("nhap nhan vien thu %d:\n",i);

Nhap(a[i]);

}

}

void Sapxeptinh(employee\_st a[], int n){

employee\_st l;

for(int i=0;i<n-1;i++){

for(int j=i+1;j<n;j++){

if(strcmp(a[i].province,a[j].province)<0){

l=a[i];

a[i]=a[j];

a[j]=l;

}

}

}

printf("\n name \tprovincne \tyear \tluong ");

for(int i=0;i<n;i++){

printf("\n%s \t%s \t%2d \t%.2f", a[i].name,a[i].province,a[i].year,a[i].luong);

}

}

void Xuat(employee\_st e){

printf("Name %3s\n",e.name);

printf("Province %10s\n",e.province );

printf("Year %15d\n",e.year);

printf("Luong %.2f\n",e.luong);

}

void XuatDS(employee\_st a[],int n){

for(int i=0;i<n;i++){

Xuat(a[i]);

}

}

void themnv(employee\_st a[], employee\_st x,int k,int n){

for(int i=n;i>k;i--){

a[i]=a[i-1];

}

a[k]=x;

n++;

printf("\nDanh sach nhan vien sau khi them \n");

XuatDS(a,n);

}

void Xoanv(employee\_st a[],int h,int n ){

for(int i=0;i<n;i++)

a[h]=a[h+1];

n=n-1;

printf("Danh sach nhan vien sau khi xoa mot nhan vien \n");

XuatDS(a,n);

}

void timtheoten(employee\_st a[],int n, char ten[]){

int dem=0;

for(int i=0;i<n;i++){

if(strcmp(ten,a[i].name)==0){

printf("\n name \tprovincne \tyear \tluong ");

printf("\n%s \t%s \t%2d \t%.2f", a[i].name,a[i].province,a[i].year,a[i].luong);

dem++;

}

}

if(dem==0)

printf("Khong co nhan vien %s trong danh sach!\n", ten);

}

void ghifile(employee\_st a[], int n, char fName[]){

FILE \*fOut;

fOut = fopen(fName,"w");

fprintf(fOut,"%-10s%-20s%-10s%-10s\n", "Ten", "Tinh", "Nam sinh", "Luong");

for(int i=0;i<n;i++){

fprintf(fOut,"%-10s%-20s%-10d%-10.2f\n",a[i].name,a[i].province,a[i].year,a[i].luong);

}

fclose(fOut);

}

void Menu(){

char fName[] = "program.txt";

int chon, key=1;

int k,h;

employee\_st x;

while(key){

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

printf("\*\*1.Nhap danh sach nhan vien. \*\* \n");

printf("\*\*2.Xuat danh sach nhan vien. \*\*\n");

printf("\*\*3.Sap xep danh sach nhan vien theo tinh (Z->A). \*\* \n");

printf("\*\*4. Them vao mot han vien \*\* \n");

printf("\*\*5. Xoa mot nhan vien \*\* \n");

printf("\*\*6. Tim kiem nhan vien theo ten \*\* \n");

printf("\*\* 7. Ghi file vao tep \*\* \n");

printf("\*\*8. Thoat . \*\* \n");

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

printf("Moi ban chon chuc nang: ");

scanf("%d", &chon);

switch(chon){

case 1:

struct employee\_st employeeList[10];

int n;

printf ("Nhap n: ");

scanf ("%d", &n);

NhapDS(employeeList,n);

break;

case 2:

printf("\nThong tin nhan vien da nhap vao la: \n");

XuatDS(employeeList,n);

break;

case 3:

Sapxeptinh(employeeList,n);

break;

case 4:

printf("\n \t vi tri them: ");

scanf("%d",&k);

Nhap(x);

themnv(employeeList,x,k,n);

break;

case 5:

printf("\nnhap vi tri van xoa: ");

scanf("%d",&h);

Xoanv(employeeList,h,n);

break;

case 6:

char ten[30];

fflush(stdin);

printf("\nNhap ten can tim la :");

gets(ten);

timtheoten(employeeList, n, ten);

break;

case 7:

ghifile(employeeList, n, fName);

printf("\nBan da xuat file thanh cong vao %s!", fName);

case 8:

key=0;

}

}

}

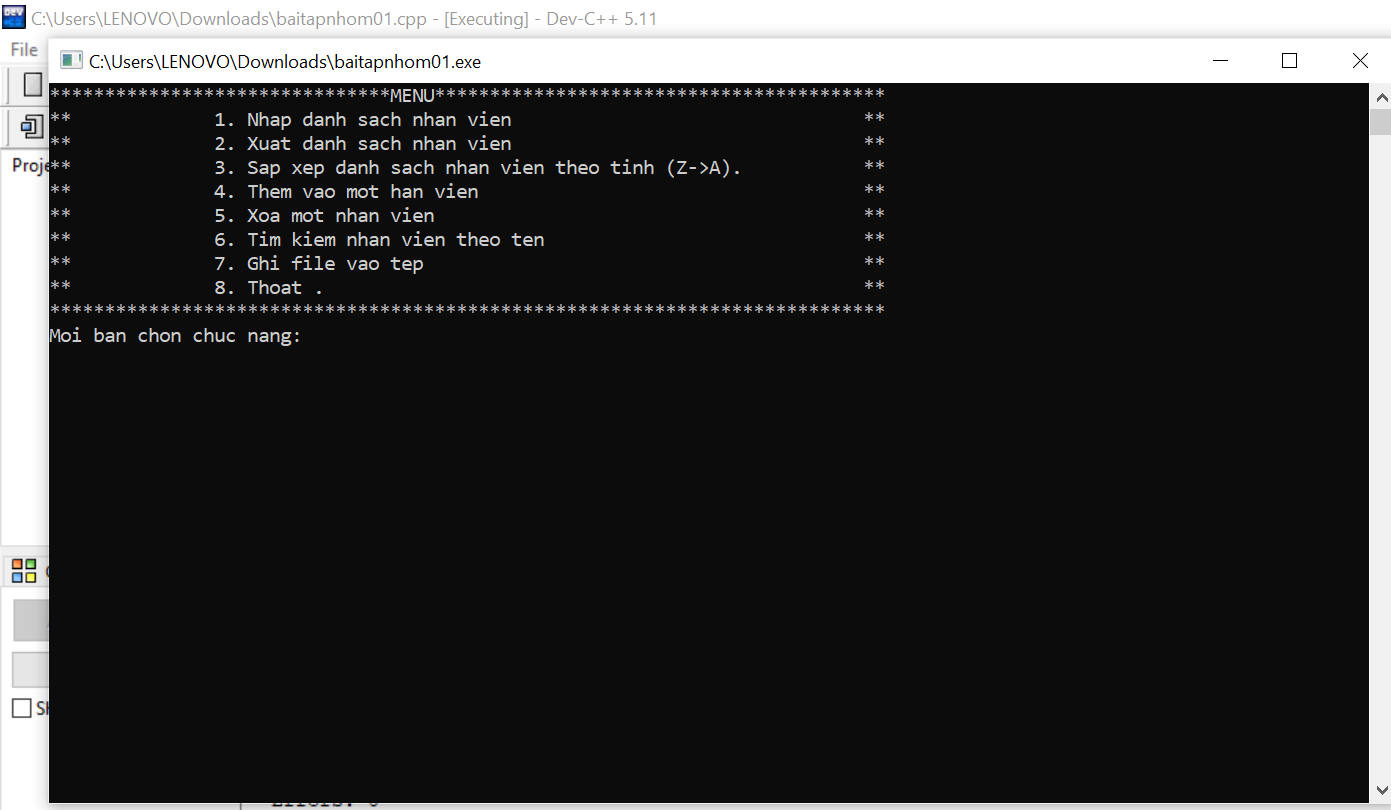
int main (){

Menu();

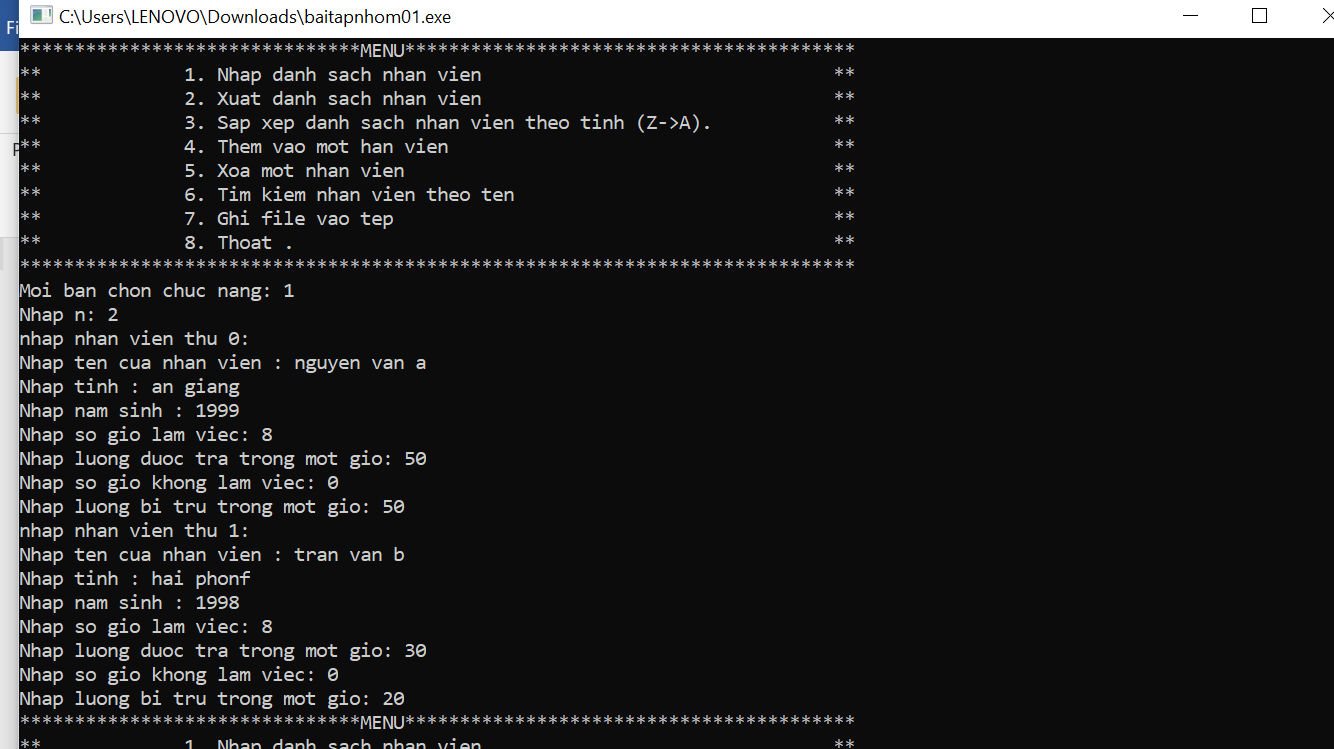
}

**6 Kết quả đạt được.**

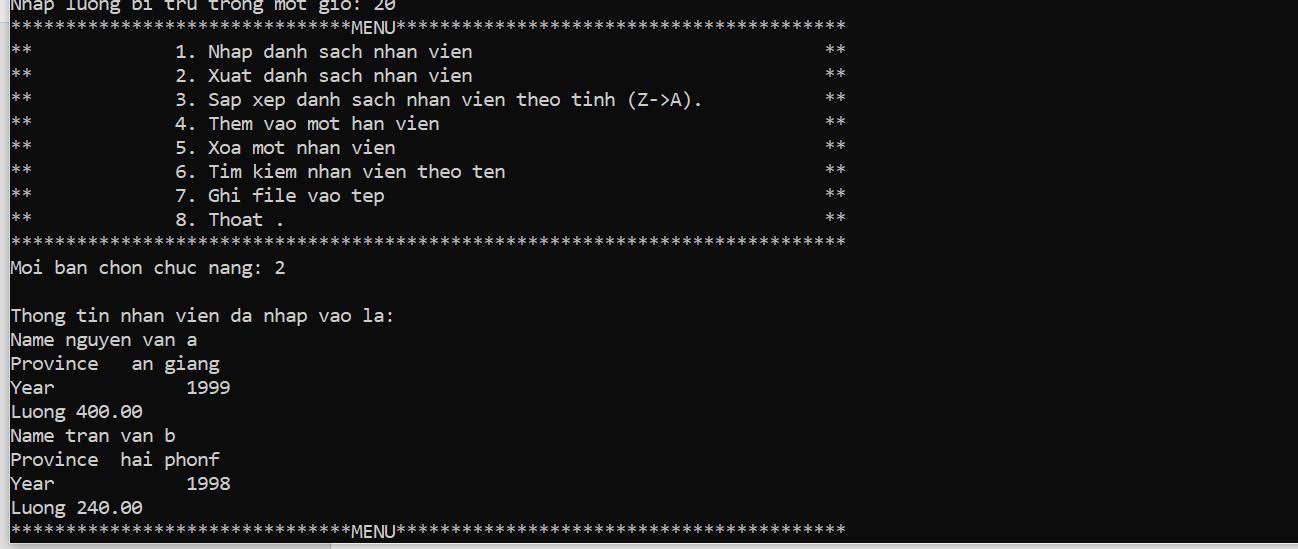
a Menu chính:



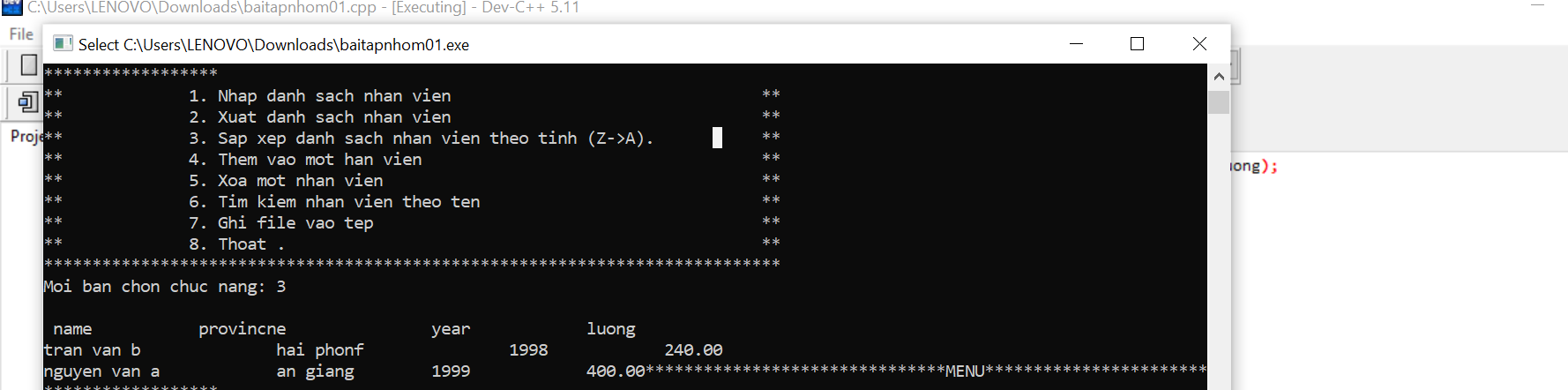
b Phần nhập danh sách:



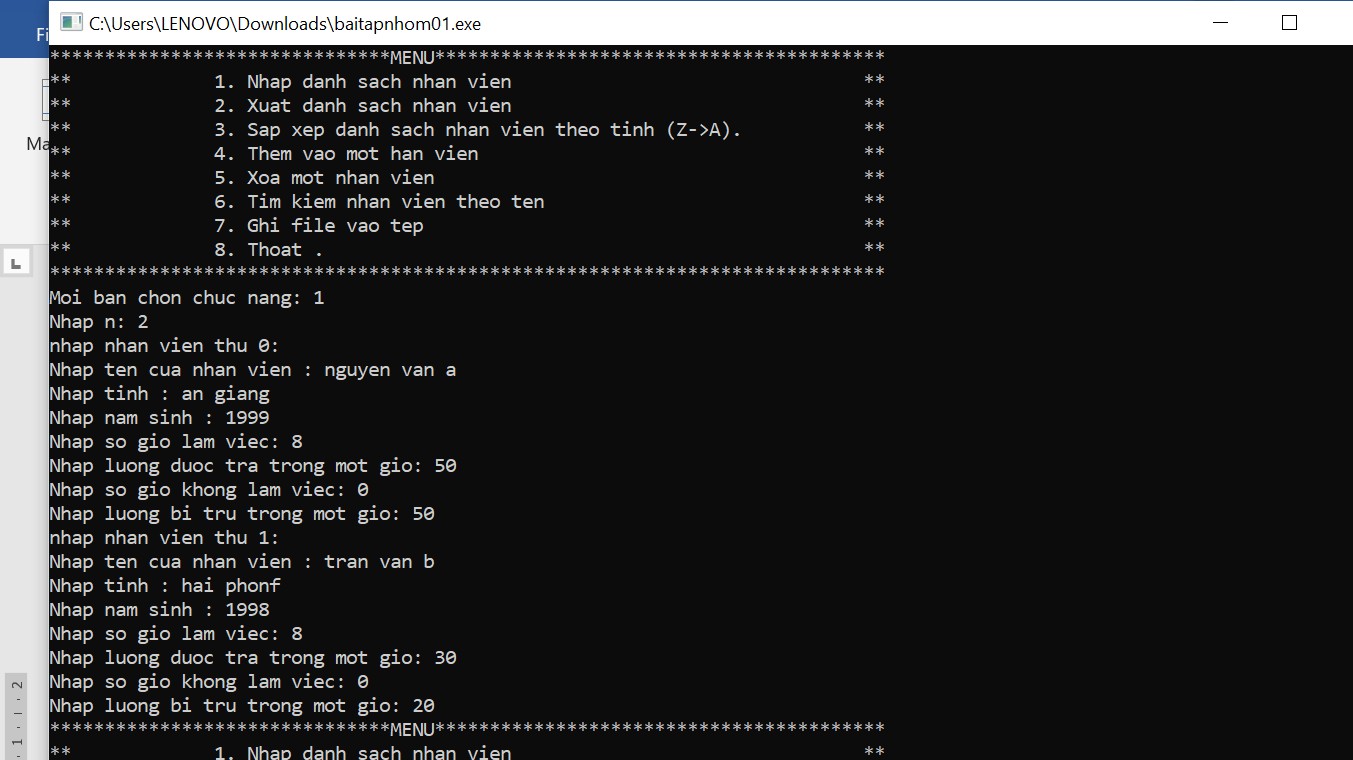
c Phần xuất danh sách:

****

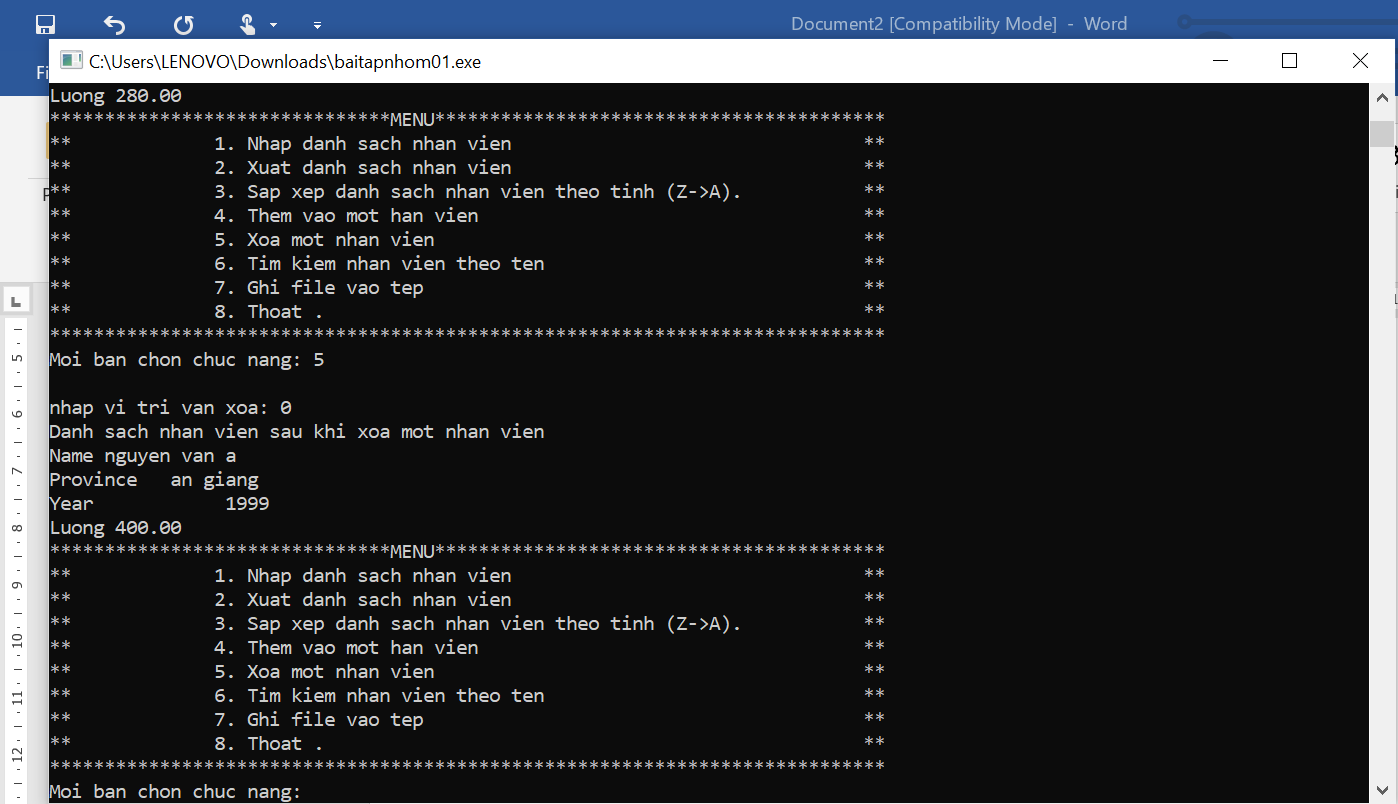
d Sắp xếp danh sách theo tên tỉnh:



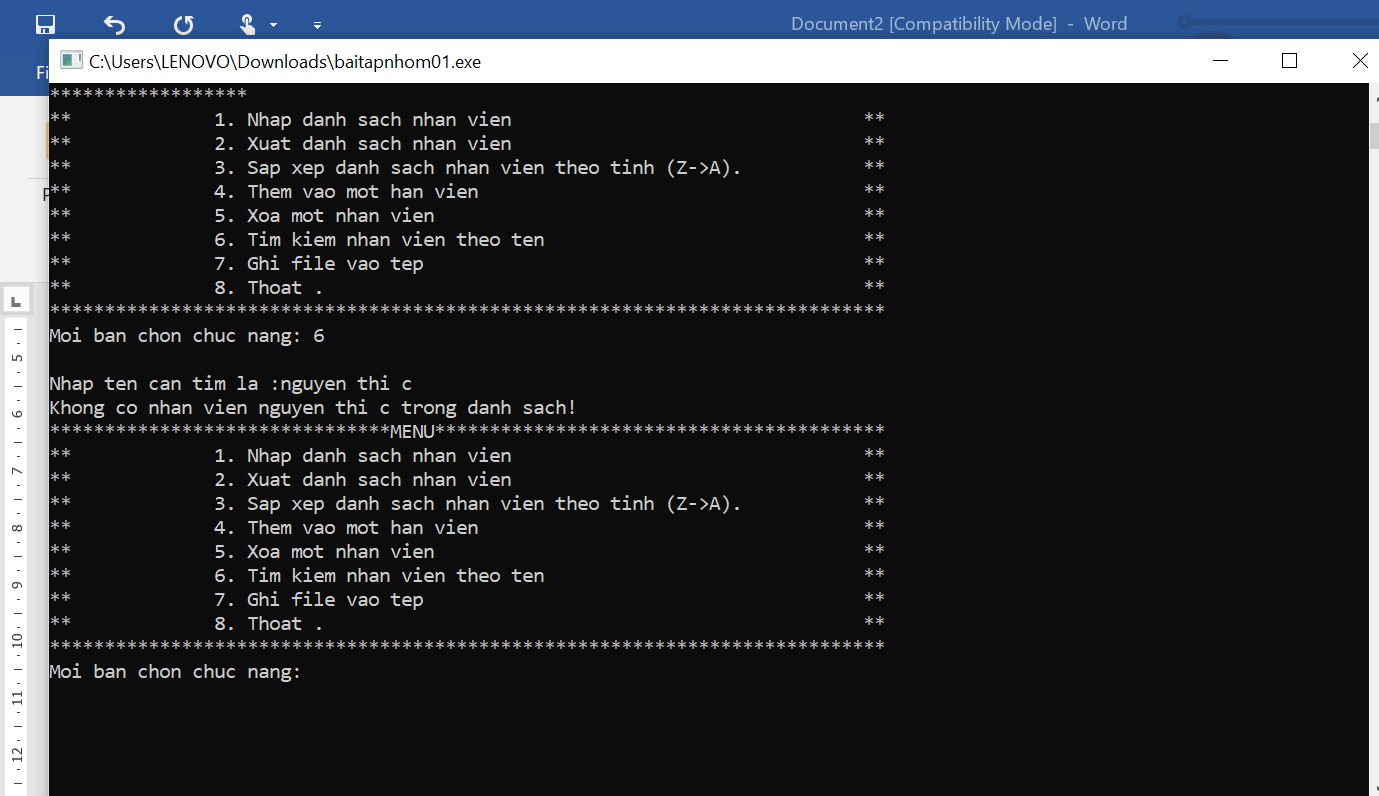
e Thêm nhân viên vào danh sách:



f Xóa nhân viên khỏi danh sách:



g Tìm kiếm nhân viên theo tên:

****

h Ghi file vào tệp nhị phân:

