**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN HỌC**

**LẬP TRÌNH NÂNG CAO**

**ĐỀ TÀI: XÂY DỰNG PHẦN MỀM QUẢN LÍ NHÂN VIÊN**

SINH VIÊN THỰC HIỆN

TRẦN MẶC KHẢI – MSSV: 6051071053

TRẦN ĐĂNG CHƯƠNG – MSSV 6051071013

PHẠM MINH HIẾU – MSSV:6051071041

LỚP: CÔNG NGHỆ THÔNG TIN K60

KHÓA: K60

**LỜI NÓI ĐẦU**

Trong thời đại công nghệ thông tin đang phát triển, các công ty cần nguồn nhân lực dồi dào để phát triển và tồn tại. Với nguồn nhân lực dồi dào như vậy thì mỗi công ty cần phải có một hệ thống quản lý nhân sự nhằm giám sát và quản lý nguồn lực mình. Vì vậy, chúng em xây dựng nên phầm mềm quản lý nhân viên. Nội dung báo cáo bao gồm những chương sau:

CHƯƠNG 1: TỔNG QUAN VỀ QUẢN LÝ NHÂN VIÊN

CHƯƠNG 2: CÁCH XÂY DỰNG CHƯƠNG TRÌNH

CHƯƠNG 3: NỘI DUNG LÝ THUYẾT

CHƯƠNG 4: CÁC TÀI LIỆU THAM KHẢO

Trong quá trình làm bài tập lớn và báo cáo sẽ không tránh khỏi những sai sót vì đây là lần đầu tiên nên mong cô có thể góp ý và chia sẻ cho nhóm chúng em để rút kinh nghiệm lần sau.

**MỤC LỤC**

**CHƯƠNG 1: TỔNG QUAN VỀ QUẢN LÝ NHÂN VIÊN**

1. Lí do chọn đề tài......................................................................1
2. Mục đích xây dựng đề tài........................................................2

**CHƯƠNG 2: CÁCH XÂY DỰNG CHƯƠNG TRÌNH**

1. Ý tưởng....................................................................................3
2. Chi tiết và giải thích code .......................................................4

**CHƯƠNG 3: NỘI DUNG LÝ THUYẾT**

1. Làm việc với tệp (Work with the file)......................................9
   1. Các kiểu file......................................................................9
   2. Các thao tác với file..........................................................9
   3. Thao tác với file trên ngôn ngữ C.....................................10
   4. Các kiểu file......................................................................13
   5. Đọc/Ghi flie nhị phân trong C..........................................14
   6. Một số ví dụ về đọc ghi file trong C.................................16
2. Danh sách liên kết đơn (single linked list)...............................19
   1. Cài đặt danh sách liên kết đơn..........................................20
3. Thuật toán.................................................................................23
   1. Sắp xếp..............................................................................23
   2. Tìm kiếm...........................................................................30
   3. Thêm/bớt phần tử..............................................................33

**BÁO CÁO BÀI TẬP LỚN LẬP TRÌNH C**

**CHƯƠNG 1**

**TỔNG QUAN VỀ QUẢN LÝ NHÂN VIÊN**

1. **Lý do chọn đề tài:**

Ngày nay Công nghệ thông tin đã trở thành một phần tất yếu của cuộc sống con người, có người đã nói rằng nước Mỹ giàu mạnh một phần là nhờ và Công nghệ thông tin. Nếu lúc trước Công nghệ thông tin là một điều viễn tưởng thì giờ đây nó đã trở thành một phần không thể thiếu trong tất cả các lĩnh vực từ y học, kinh doanh đến cả giáo dục Đất nước ta đang ngày một phát triển, đang cố gắng hòa nhập và rút ngắn khoảng cách với thế giới, việc nước ta trở thành thành viên của WTO đã được các nhà kinh doanh chú ý đến và đâu tư ngày càng nhiều vào Việt Nam. Vì vậy để không bị quá lạc hậu, để rút ngắn khoảng cách với các nước, để đáp ứng yêu cầu tất yếu của các nhà đầu tư vào Việt Nam và để các nhà kinh doanh trong nước có đủ sức cạnh tranh với nước ngoài thì bắt buộc phải đầu tư cho Công nghệ thông tin mà ở đây chính xác là các phần mềm tin học dùng cho các công ty, bệnh viện v.v. Việc áp dụng các phần mềm tin học vào các lĩnh vực giúp nâng cao tính hiệu quả và chính xác của công việc, ngoài ra còn tiết kiệm thời gian và giảm bớt mệt nhọc cho con người khiến hệ thống công việc hoạt động nhịp nhàng hơn. Nhu cầu thực tế của xã hội đòi hỏi con người phải luôn năng động và sáng tạo để tạo ra ngày càng nhiều sản phẩm cho xã hội. Các công ty luôn luôn phát triển, các hồ sơ tuyển vào sẽ nhiều lên vì vậy đòi hỏi phải cần rất nhiều những kho chứa hồ sơ để lưu trữ hồ sơ của nhân viên khi vào công ty. để giảm đi những bất cập trong công tác quản lý nhân sự và lương trong công ty, giải pháp hiệu quả nhất hiện nay là đầu tư công nghệ và thiết bị hiện đại, ứng dụng công nghệ thông tin, tự động hoá trong điều hành và quản lý nhân sự để nâng cao chất lượng phục vụ, đổi mới phương thức quản lý, tăng năng xuất hiệu quả. Đó là những nội dung cơ bản đề cập đến trong

đề tài này. Hôm nay em chọn đề tài tốt nghiệp là “PHẦN MỀM QUẢN LÝ NHÂN SỰ”.

1. **Mục đích xây dựng phần mềm:**

Các công ty muốn tồn tại và phát triển thì phải có nhân tố con người. Cùng với tốc độ phát triển mạnh mẽ của Công nghệ thông tin các lập trình viên đã phát minh ra nhiều phần mềm hữu ích nhằm phục vụ cho công việc của con người và phần mềm quản lý nhân sự cũng là một trong những vấn đề con người quan tâm nhiều nhất, nó giúp cho công tác nghiệp vụ của các công ty giảm thiểu tối đa những vất vả trong công việc giúp cho việc lưu trữ hồ sơ dễ dàng hơn, giảm thiểu diện tích các kho và thậm chí là không cần.

Trong thời buổi công nghê thông tin phát triển, do các doanh nghiệp đang phát triển cần quản lý tốt nhân viên của mình. Chương trình quản lý nhân sự sẽ giúp tiết kiệm thời gian và tài chính của các chủ doanh nghiệp.

**CHƯƠNG 2**

**CÁCH XÂY DỰNG CHƯƠNG TRÌNH**

**I. Ý tưởng:**

Đầu tiên sẽ định nghĩa 1 kiểu dữ liệu mới là NhanVien sau đó khai báo các thành phần của các nhân viên:

// Định nghĩa kiểu dữ liệu của nhân viên.

struct NhanVien

{

    char ma[10];

    char ho[10];

    char dem[10];

    char ten[10];

    char gioitinh[10];

    int tuoi;

    int songaylam;

    float luong=1;

};

Sau đó chúng ta sẽ đến với các hàm sử dụng trong bài:

//Các hàm sử dụng trong bài:

void nhapNhanVien(NhanVien &nhanvien);

*Hàm này cho phép ta nhập các thông số của nhân viên.*

void tinhLuongNhanVien(NhanVien &nhanvien);

*Hàm này sẽ tính lương của nhân viên.*

void hienThiThuTuNhanVien(NhanVien nhanvien);

Hàm này sẽ hiển thị một nhân viên đã nhập.

void hienThiDanhSachNhanVien(NhanVien \*danhsach, int soluong);

Hàm này sẽ hiển thị danh sách nhân viên bạn đã nhập.

void hienThiTenCot();

*Hàm này sẽ hiển thị thông số của các nhân viên.*

void sapXepTheoTen(NhanVien \*danhsach, int soluong);

*Hàm này sẽ sắp xếp các nhân viên đã nhập từ a-z*

void timTheoTen(NhanVien \*danhsach, int soluong);

*Hàm này sẽ tìm nhân viên theo tên.*

void docFile(FILE \*file, char str[MAX\_SIZE]);

*Hàm này sẽ đọc danh sách các nhân viên đã ghi vào file.*

void huongDanSuDung();

*Hàm này sẽ hiển thị Menu.*

**II. Chi tiết và giải thích code:**

Đầu tiên khi chạy chương trình sẽ có một menu để cho người dùng sử dụng:

void huongDanSuDung()

    {

        printf("============ MENU ============");

        printf("\n1. Them nhan vien vao danh sach.");

        printf("\n2. Hien thi danh sach nhan vien.");

        printf("\n3. Sap xep theo ten.");

        printf("\n4. Tim nhan vien theo ten.");

        printf("\n5. Ghi thong tin ra File");

        printf("\n0. Thoat chuong trinh.");

**1. Khi chọn phím 1:**

Thêm một dữ liệu nhân viên vào danh sách đã có:

// Nhập thông số các nhân viên.

void nhapNhanVien(NhanVien &nhanvien)

{

    printf("Nhap ma nhan vien: ");

    scanf("%s",&nhanvien.ma);

    printf("\nNhap ho: ");

    scanf("%s",&nhanvien.ho);

    printf("\nNhap ten dem: ");

    scanf("%s",&nhanvien.dem);

printf("\nNhap ten: ");

    fflush(stdin);

    scanf("%s",&nhanvien.ten);

    printf("\nNhap tuoi: ");

    scanf("%d",&nhanvien.tuoi);

    printf("\nNhap gioi tinh: ");

    scanf("%s",&nhanvien.gioitinh);

    printf("\nNhap so ngay lam: ");

    scanf("%d",&nhanvien.songaylam);

tinhLuongNhanVien(nhanvien);

}

Tiếp theo sẽ là hàm hiển thị một nhân viên đã nhập:

// Hiển thị nhân viên đã nhập.

void hienThiThuTuNhanVien(NhanVien nhanvien)

{

    printf("%12s%10s%12s%10s%8d%20s%15d%15.0f\n",

    nhanvien.ma,nhanvien.ho,nhanvien.dem,nhanvien.ten,nhanvien.tuoi,nhanvien.gioitinh,nhanvien.songaylam,tinhLuongNhanVien(nhanvien));

}

Hàm này sẽ hiển thị thông số của các nhân viên.

**Khi chọn phím 2:**

Để hiển thị danh sách tất cả các nhân viên đã nhập chương trình đã thực hiện một thao tác ở case số một của menu là:

case 1:

                nhapNhanVien(nhanvien);

                danhsach[soluong++] = nhanvien;

                break;

Sau khi nhập từng nhân viên biến Soluongsẽ tăng lên theo số lần bạn nhập sau đó để hiển thị danh sách nhân viên chúng ta sẽ sử dụng vòng lắp for với giới hạn là soluong.

// Hiển thị danh sác các nhân viên đã nhập.

void hienThiDanhSachNhanVien(NhanVien \*danhsach, int soluong)

{

    printf("\nDANH SACH NHAN VIEN\n");

    hienThiTenCot();

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

        hienThiThuTuNhanVien(danhsach[i]);

    }

    printf("-----------------------------------------------------"

    "-----------------------------------------------------------------\n");

}

**Khi chọn phím số 3:**

Chương trình sẽ sắp xếp các nhân viên đã nhập theo thứ tự từ a-z

// Sắp xếp từ a-z theo tên các nhân viên đã nhập.

void sapXepTheoTen(NhanVien \*danhsach, int soluong)

{

    for(int i=0; i<soluong-1; i++){

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

            if(strcmp(danhsach[i].ten, danhsach[i+1].ten)>0){

                NhanVien nhanvien = danhsach[j];

                danhsach[j] = danhsach[i];

                danhsach[i] = nhanvien;

            }

 }

    }

}

**Khi chọn phím số 4:**

Chương trình sẽ yêu cầu bạn nhập 1 tên nhân viên bạn muốn tìm

// Tìm kiếm tên nhân viên trong danh sách đã nhập.

void timTheoTen(NhanVien \*danhsach, int soluong)

{

    char ten[10];

    fflush(stdin);

    printf("Nhap ten: ");

    gets(ten);

    getchar;

    hienThiTenCot();

    int timnhanvien = 0;

    for(int i=0; i<soluong; i++){

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

            hienThiThuTuNhanVien(danhsach[i]);

            timnhanvien = 1;

        }

    }

    if(timnhanvien == 0){

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

    }

}

Vậy tổng kết lại chương trình quản lý nhân sự giúp chung ta quản lý tốt hơn về nhân viên của mình. Cho phép thêm nhân viên tìm kiểm và quản lý lương một cách dễ dàng fiups tiết kiệm thời gian tiền bạc của các doanh nghiệp.

**CHƯƠNG 3**

**NỘI DUNG LÝ THUYẾT**

1. ****Làm việc với tệp (Work with the file):****
   1. **Các kiểu file**

**\***Flie văn bản – text files****

File văn bản là file thường có đuôi là .txt. Những file này bạn có thể dễ dàng tạo ra bằng cách dùng các text editer thông dụng như Notepad, Notepad++, Sublime Text,…

Khi bạn mở các file này bằng các text editer nói trên, bạn sẽ thấy được văn bản ngay và có thể dễ dàng thao tác sửa, xóa, thêm nội dung của file này.

Kiểu file này thuận tiện cho chúng ta trong việc sử dụng hàng ngày, nhưng nó sẽ kém bảo mật và cần nhiều bộ nhớ để lưu trữ hơn.

****\*Flie nhị phân – Binary files****

File nhị phân thường có đuôi mở rộng là .bin

Thay vì lưu trữ dưới dạng văn bản thuần thúy, các file này được lưu dưới dạng nhị phân, chỉ bao gồm các số 0 và 1. Bạn cũng sẽ thấy các con số này nếu cố mở nó bằng 1 text editer kể trên.

Loại file này giúp lưu trữ được dữ liệu với kích thước lớn hơn, không thể đọc bằng các text editer thông thường và thông tin lưu trữ ở loại file được bảo mật hơn so với file văn bản.

**1.2 Các thao tác với file**

Trong ngôn ngữ lập trình C, có một số thao tác chính khi làm việc với file, bao gồm cả file văn bản và file nhị phân:

1. Tạo mới một file
2. Mở một file đã có
3. Đóng file đang mở
4. Đọc thông tin từ file/ Ghi thông tin ra file

**1.3 Thao tác với file trên ngôn ngữ C**

Khi làm việc với file, bạn cần khai báo 1 con trỏ kiểu FILE. Việc khai báo này là cần thiết để có sự kết nối giữa chương trình của bạn và tập tin mà bạn cần thao tác.

*\****Thao tác mở flie**

Để đọc ghi file trong C cũng như trong mọi ngôn ngữ lập trình, việc đầu tiên bạn cần làm là mở file mà bạn muốn làm việc. Trong ngôn ngữ lập trình C, chúng ta có thể mở file bằng cách sử dụng hàm fopen() trong thư viện stdio.h như sau:

fptr = fopen("fileopen","mode");

Trong đó mode là một tham số chúng ta cần chỉ định.

Ví dụ:

fptr = fopen("E:\\cprogram\\newprogram.txt","w");

// hoặc

fptr = fopen("E:\\cprogram\\oldprogram.bin","rb");

**\*Các tham số của “mode”**

* Giả sử tập tin newprogram.txt chưa có trong thư mục E:\cprogram. Ví dụ đầu tiên với mode = "w" sẽ cho phép chương trình tự động tạo ra file newprogram.txt nếu nó chưa có. Và sau đó mở file này lên nhưng chương trình chỉ có thể ghi dữ liệu vào mà không thể đọc.
* Mode là w chỉ cho phép chương trình ghi(nếu đã có dữ liệu thì ghi đè) nội dung của file.
* Với ví dụ thứ 2, mode là rb cho phép chương trình mở 1 file nhị phân đã có sẵn oldprogram.bin. Với trường hợp này, chương trình của bạn chỉ có thể đọc file và không thể ghi nội dung vào file.

Dưới đây là các giá trị có thể có của tham số mode  nói trên:

**Bảng 1.1**

| **Mode** | **Ý nghĩa** | **Nếu file không tồn tại** |
| --- | --- | --- |
| r | Mở file chỉ cho phép đọc | Nếu file không tồn tại, fopen() trả về NULL. |
| rb | Mở file chỉ cho phép đọc dưới dạng nhị phân. | Nếu file không tồn tại, fopen() trả về NULL. |
| w | Mở file chỉ cho phép ghi. | Nếu file đã tồn tại, nội dung sẽ bị ghi đè. Nếu file không tồn tại, nó sẽ được tạo tự động. |
| wb | Open for writing in binary mode. | Nếu file đã tồn tại, nội dung sẽ bị ghi đè. Nếu file không tồn tại, nó sẽ được tạo tự động. |
| a | Mở file ở chế độ ghi “append”. Tức là sẽ ghi vào cuối của nội dung đã có. | Nếu file không tồn tại, nó sẽ được tạo tự động. |
| ab | Mở file ở chế độ ghi nhị phân “append”. Tức là sẽ ghi vào cuối của nội dung đã có. | Nếu file không tồn tại, nó sẽ được tạo tự động. |
| r+ | Mở file cho phép cả đọc và ghi. | Nếu file không tồn tại, fopen() trả về NULL. |
| rb+ | Mở file cho phép cả đọc và ghi ở dạng nhị phân. | Nếu file không tồn tại, fopen() trả về NULL. |
| w+ | Mở file cho phép cả đọc và ghi. | Nếu file đã tồn tại, nội dung sẽ bị ghi đè. Nếu file không tồn tại, nó sẽ được tạo tự động. |
| wb+ | Mở file cho phép cả đọc và ghi ở dạng nhị phân. | Nếu file đã tồn tại, nội dung sẽ bị ghi đè. Nếu file không tồn tại, nó sẽ được tạo tự động. |
| a+ | Mở file cho phép đọc và ghi “append”. | Nếu file không tồn tại, nó sẽ được tạo tự động. |
| ab+ | Mở file cho phép đọc và ghi “append” ở dạng nhị phân. | Nếu file không tồn tại, nó sẽ được tạo tự động. |

**\*Thao tác đóng file**

Khi làm việc với tập tin hoàn tất, kể cả là file nhị phân hay file văn bản. Bạn cần đóng file sau khi làm việc với nó xong.Việc đóng file đang mở có thể được thực hiện bằng cách dùng hàm fclose().

fclose(fptr); //Con trỏ FILE trỏ tới file cần được đóng.

**1.4 Các kiểu file**

Chúng ta sẽ học cách đọc ghi file trong C với file văn bản trước. Với file nhị phân, bạn kéo xuống dưới để xem tiếp.

Để làm việc với file văn bản, chúng ta sẽ sử dụng fprintf() và fscanf().

VD1: Ghi file sử dụng fprintf()

#include <stdio.h>

#include <stdlib.h>

int main()

{

   int num;

   FILE \*fptr;

   fptr = fopen("C:\\program.txt","w");

   if(fptr == NULL)

   {

      printf("Error!");

      exit(1);

   }

   printf("Enter num: ");

   scanf("%d",&num);

   fprintf(fptr,"%d",num);

   fclose(fptr);

   return 0;

}

Chương trình nhận số num từ bàn phím và ghi vào file văn bản program.txt.

Sau khi bạn chạy chương trình này, bạn sẽ thấy file văn bản program.txt được tạo mới trong ổ C trên máy tính bạn. Khi mở file này lên, bạn sẽ thấy số mà bạn vừa nhập cho biến num kia.

VD2: Đọc file sử dụng fscanf()

#include <stdio.h>

#include <stdlib.h>

int main()

{

   int num;

   FILE \*fptr;

   if ((fptr = fopen("C:\\program.txt","r")) == NULL){

       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.

       exit(1);

   }

   fscanf(fptr,"%d", &num);

   printf("Value of n=%d", num);

   fclose(fptr);

   return 0;

}

Chương trình này sẽ đọc giá trị số được lưu trong file program.txt mà chương trình ở VD1 vừa tạo ra và in lên màn hình.

**1.5 Đọc/Ghi file nhị phân trong C**

Các hàm fread() và fwrite() trong C được sử dụng để đọc và ghi file trong C ở dạng nhị phân.

**\*Ghi file nhị phân**

Để ghi file nhị phân, bạn cần sử dụng hàm fwrite(). Hàm này cần 4 tham số: địa chỉ của biến lưu dữ liệu cần ghi, kích thước của biến lưu dữ liệu đó, số lượng kiểu dữ liệu của biến đó và con trỏ FILE trỏ tới file bạn muốn ghi.

fwrite(address\_data,size\_data,numbers\_data,pointer\_to\_file);

VD3: Ghi file nhị phân sử dụng fwrite()

#include <stdio.h>

#include <stdlib.h>

struct threeNum

{

   int n1, n2, n3;

};

int main()

{

   int n;

   struct threeNum num;

   FILE \*fptr;

   if ((fptr = fopen("C:\\program.bin","wb")) == NULL){

       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.

       exit(1);

   }

   for(n = 1; n < 5; ++n)

   {

      num.n1 = n;

      num.n2 = 5\*n;

      num.n3 = 5\*n + 1;

      fwrite(&num, sizeof(struct threeNum), 1, fptr);

   }

   fclose(fptr);

   return 0;

}

Trong VD3 này, chương trình sẽ tạo ra một file program.bin trên ổ đĩa C của bạn. Chương trình này đã khai báo 1 kiểu dữ liệu cấu trúc lưu 3 giá trị số n0, n1, n2; Và nó được sử dụng trong hàm main có tên biến là So.

Trong vòng lặp, các số được ghi vào file sử dụng hàm fwrite(). Các tham số gồm:

* Tham số đầu tiên là địa chỉ của biến So
* Tham số thứ 2 là kích thước của biến So
* Tham số thứ 3 là số lượng kiểu dữ liệu – ở đây là 1.
* Tham số thứ 4 là con trỏ FILE trỏ tới tệp tin program.bin

Cuối cùng, chúng ta đóng file sử dụng fclose().

**\* Đọc file nhị phân**

Hàm fread() cũng có 4 tham số tương tự như hàm fwrite() phía trên.

fread(address\_data,size\_data,numbers\_data,pointer\_to\_file);

**Ví dụ đọc file nhị phân sử dụng fread()**

#include <stdio.h>

#include <stdlib.h>

struct threeNum

{

   int n1, n2, n3;

};

int main()

{

   int n;

   struct threeNum num;

   FILE \*fptr;

   if ((fptr = fopen("C:\\program.bin","rb")) == NULL){

       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.

       exit(1);

   }

   for(n = 1; n < 5; ++n)

   {

      fread(&num, sizeof(struct threeNum), 1, fptr);

      printf("n1: %d\tn2: %d\tn3: %d", num.n1, num.n2, num.n3);

   }

   fclose(fptr);

   return 0;

Trong ví dụ này, bạn đọc file program.bin và lặp qua từng dòng. Bạn sẽ nhận được các giá trị tương ứng khi bạn ghi vào trong VD3.

**1.6 Một số ví dụ về đọc ghi file trong C**

Trong phần này, mình sẽ trình bày 2 ví dụ về đọc ghi file trong C, bao gồm các bài tập đọc ghi file sau:

* Ghi văn bản vào file trong C
* Đọc dữ liệu văn bản từ file trong C

**\*Ghi vào file một câu văn bản**

#include <stdio.h>

#include <stdlib.h>  /\* For exit() function \*/

int main()

{

   char sentence[1000];

   FILE \*fptr;

   fptr = fopen("program.txt", "w");

   if(fptr == NULL)

   {

      printf("Error!");

      exit(1);

   }

   printf("Enter a sentence:\n");

   gets(sentence);

   fprintf(fptr,"%s", sentence);

   fclose(fptr);

   return 0;

Chạy thử:

Nhập một câu:

Tôi là một lập trình viên.

**\* Đọc dữ liệu văn bản từ file**

#include <stdio.h>

#include <stdlib.h> // For exit() function

int main()

{

    char c[1000];

    FILE \*fptr;

    if ((fptr = fopen("program.txt", "r")) == NULL)

    {

        printf("Error! opening file");

        // Program exits if file pointer returns NULL.

        exit(1);

    }

    // reads text until newline

    fscanf(fptr,"%[^\n]", c);

    printf("Data from the file:\n%s", c);

    fclose(fptr);

    return 0;

}

**Chạy thử:**

Dữ liệu từ tệp:

Tôi là một lập trình viên.

**Danh sách liên kết đơn là gì?**

Danh sách liên kết đơn là một tập hợp các Node được phân bố động, được sắp xếp theo cách sao cho mỗi Node chứa “một giá trị”(Data) và “một con trỏ”(Next). Con trỏ sẽ trỏ đến phần tử kế tiếp của danh sách liên kết đó. Nếu con trỏ mà trỏ tới NULL, nghĩa là đó là phần tử cuối cùng của danh sách liên kết.

1. **Danh sách liên kết đơn (Single linked list):**

***\*Danh sách liên kết.***

Về bản chất, danh sách liên kết có chức năng như một mảng, có thể thêm và xóa các phần tử ở bất kỳ vị trí nào khi cần thiết. Một số sự khác nhau giữa danh sách liên kết và mảng.

**Bảng 2.1**

|  |  |  |
| --- | --- | --- |
| **Nội dung** | **Mảng** | **Danh sách liên kết** |
| Kích thước | * Kích thước cố định * Cần chỉ rõ kích thước trong khi khai báo | * Kích thước thay đổi trong quá trình thêm/ xóa phần tử * Kích thước tối đa phụ thuộc vào bộ nhớ |
| Cấp phát bộ nhớ | * Tĩnh: Bộ nhớ được cấp phát trong quá trình biên dịch | * Động: Bộ nhớ được cấp phát trong quá trình chạy |
| Thứ tự & sắp xếp | * Được lưu trữ trên một dãy ô nhớ liên tục | * Được lưu trữ trên các ô nhớ ngẫu nhiên |
| Truy cập | * Truy cập tới phần tử ngẫu nhiên trực tiếp bằng cách sử dụng chỉ số mảng: O(1) | * Truy cập tới phần tử ngẫu nhiên cần phải duyệt từ đầu/cuối đến phần tử đó: O(n) |
| Tìm kiếm | * Tìm kiếm tuyến tính hoặc tìm kiếm nhị phân | * Chỉ có thể tìm kiếm tuyến tính |

Hình ảnh mô tả cho 1 node trong danh sách liên kết đơn

[](https://nguyenvanhieu.vn/wp-content/uploads/2018/12/data-node-danh-sach-lien-ket.png)

Và đây là hình ảnh mô phỏng một danh sách liên đơn kết đầy đủ:

[](https://nguyenvanhieu.vn/wp-content/uploads/2018/12/danh-sach-lien-ket-la-gi.jpg)

Mô phỏng của danh sách liên kết đơn.

Danh sách liên kết đơn(Single linked list): Chỉ có sự kết nối từ phần tử phía trước tới phần tử phía sau.

## 2.1 Cài đặt danh sách liên kết đơn

### \*Khai báo linked list

Để đơn giản hóa, data của chúng ta sẽ là số nguyên(int). Bạn cũng có thể sử dụng các kiểu nguyên thủy khác(float, char,…) hay kiểu dữ liệu struct(SinhVien, CanBo,…) tự tạo.

struct LinkedList{

    int data;

    struct LinkedList \*next;

 };

Khai báo trên sẽ được sử dụng cho mọi Node trong danh sách liên kết. Trường data sẽ lưu giữa giá trị và next sẽ là con trỏ để trỏ đến thằng kế tiếp của nó.

**Tại sao next lại là kiểu LinkedList của chính nó?**Bởi vì nó là con trỏ trỏ của chính bản thân nó, và nó trỏ tới một thằng Node kế tiếp cũng có kiểu LinkedList.

### \*Tạo mới 1 Node

Hãy tạo một kiểu dữ liệu của struct LinkedList để code clear hơn:

typedef struct LinkedList \*node; //Từ giờ dùng kiểu dữ liệu LinkedList có thể thay bằng node cho ngắn gọn

node CreateNode(int value){

    node temp; // declare a node

    temp = (node)malloc(sizeof(struct LinkedList)); // Cấp phát vùng nhớ dùng malloc()

    temp->next = NULL;// Cho next trỏ tới NULL

    temp->data = value; // Gán giá trị cho Node

    return temp;//Trả về node mới đã có giá trị

}

Mỗi một Node khi được khởi tạo, chúng ta cần cấp phát bộ nhớ cho nó, và mặc định cho con trỏ next trỏ tới NULL. Giá trị của Node sẽ được cung cấp khi thêm Node vào linked list.

* **typedef** được dùng để định nghĩa một kiểu dữ liệu trong C. VD: typeder long long LL;
* **malloc** là hàm cấp phát bộ nhớ của C. Với C++ chúng ta dùng new
* **sizeof** là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm malloc

**Lưu ý:** Không giống với mảng, cần khai báo arr[size]. Trong linked list, vì mỗi Node sẽ có con trỏ liên kết đến Node tiếp theo. Do đó, với danh sách liên kết đơn, bạn chỉ cần lưu giữ Node đầu tiên(HEAD). Có head rồi bạn có thể đi tới bất cứ Node nào.

### \*Thêm Node vào danh sách liên kết

#### Thêm vào đầu

Việc thêm vào đầu chính là việc cập nhật lại thằng head. Ta gọi Node mới(temp), ta có:

* Nếu head đang trỏ tới NULL, nghĩa là linked list đang trống, Node mới thêm vào sẽ làm head luôn
* Ngược lại, ta phải thay thế thằng head cũ bằng head mới. Việc này phải làm theo thứ tự như sau:
  + Cho next của temp trỏ tới head hiện hành
  + Đặt temp làm head mới

node AddHead(node head, int value){

    node temp = CreateNode(value); // Khởi tạo node temp với data = value

    if(head == NULL){

        head = temp; // //Nếu linked list đang trống thì Node temp là head luôn

    }else{

        temp->next = head; // Trỏ next của temp = head hiện tại

        head = temp; // Đổi head hiện tại = temp(Vì temp bây giờ là head mới mà)

    }

    return head;

}

#### Thêm vào cuối

Chúng ta sẽ cần Node đầu tiên, và giá trị muốn thêm. Khi đó, ta sẽ:

1. Tạo một Node mới với giá trị value
2. Nếu head = NULL, tức là danh sách liên kết đang trống. Khi đó Node mới(temp) sẽ là head luôn.
3. Ngược lại, ta sẽ duyệt tới Node cuối cùng(Node có next = NULL), và trỏ next của thằng cuối tới Node mới(temp).

node AddTail(node head, int value){

    node temp,p;// Khai báo 2 node tạm temp và p

    temp = CreateNode(value);//Gọi hàm createNode để khởi tạo node temp có next trỏ tới NULL và giá trị là value

    if(head == NULL){

        head = temp;     //Nếu linked list đang trống thì Node temp là head luôn

    }

    else{

        p  = head;// Khởi tạo p trỏ tới head

        while(p->next != NULL){

            p = p->next;//Duyệt danh sách liên kết đến cuối. Node cuối là node có next = NULL

        }

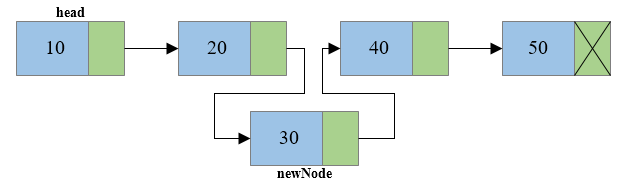
        p->next = temp;//Gán next của thằng cuối = temp. Khi đó temp sẽ là thằng cuối(temp->next = NULL mà)

    }

    return head;

}

Thêm vào vị trí bất kỳ

[](https://nguyenvanhieu.vn/wp-content/uploads/2018/12/them-node-danh-sach-lien-ket.png)Thêm Node vào giữa danh sách liên kết

Để làm được việc này, ta phải duyệt từ đầu để tìm tới vị trí của Node cần chèn, giả sử là Node Q, khi đó ta cần làm theo thứ tự sau:

* Cho next của Node mới trỏ tới Node mà Q đang trỏ tới
* Cho Node Q trỏ tới Node mới

Lưu ý: Chỉ số chèn bắt đầu từ chỉ số 0

### \* Xóa Node khỏi danh sách liên kết

#### Xóa đầu

Xóa đầu : chúng bây giờ chỉ cần cho node kế tiếp của head làm head Mà node kế tiếp của head chính là head->next.

node DelHead(node head){

    if(head == NULL){

        printf("\nCha co gi de xoa het!");

    }else{

        head = head->next;

    }

    return head;

}

#### Xóa cuối

Xóa cuối ở đây chúng ta phải duyệt đến node cuối – 1, cho next của cuối – 1 đó bằng NULL.

node DelTail(node head){

    if (head == NULL || head->next == NULL){

         return DelHead(head);

    }

    node p = head;

    while(p->next->next != NULL){

        p = p->next;

    }

    p->next = p->next->next; // Cho next bằng NULL

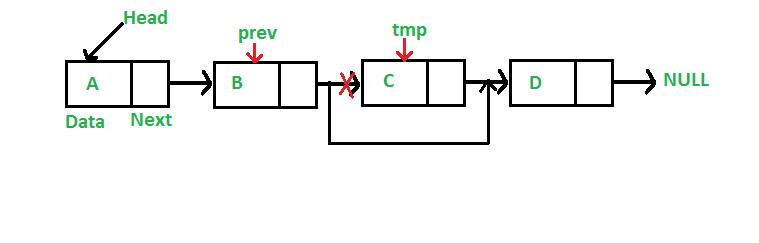
    // Hoặc viết p->next = NULL cũng được

    return head;

}

#### Xóa ở vị trí bất kỳ

Việc xóa ở vị trí bất kỳ cũng khá giống xóa ở cuối kia. Đơn giản là chúng ta bỏ qua một phần tử, như ảnh sau:

[](https://nguyenvanhieu.vn/wp-content/uploads/2018/12/xoa-node-danh-sach-lien-ket.png)

Xóa node trong danh sách liên kết

Lưu ý: Chỉ số xóa bắt đầu từ 0 . Việc tìm vị trí càn xóa chỉ duyệt tới Node gần cuối (cuối – 1).

node DelAt(node head, int position){

    if(position == 0 || head == NULL || head->next == NULL){

        head = DelHead(head); // Nếu vị trí chèn là 0, tức là thêm vào đầu

    }else{

        // Bắt đầu tìm vị trí cần chèn. Ta sẽ dùng k để đếm cho vị trí

        int k = 1;

        node p = head;

        while(p->next->next != NULL && k != position){

            p = p->next;

            ++k;

        }

        if(k != position){

            // Nếu duyệt hết danh sách lk rồi mà vẫn chưa đến vị trí cần chèn, ta sẽ mặc định xóa cuối

            // Nếu bạn không muốn xóa, hãy thông báo vị trí xóa không hợp lệ

            head = DelTail(head);

            // printf("Vi tri xoa vuot qua vi tri cuoi cung!\n");

        }else{

            p->next = p->next->next;

        }

    }

    return head;

}

### \*Lấy giá trị ở vị trí bất kỳ

Chúng ta sẽ viết một hàm để truy xuất giá trị ở chỉ số bất kỳ . Trong trường hợp chỉ số vượt quá chiều dài của linked list – 1, hàm này trả về vị trí cuối cùng. Do hạn chế là chúng ta không thể raise error khi chỉ số không hợp lệ. Tôi mặc định chỉ số bạn truyền vào phải là số nguyên không âm. Nếu bạn muốn kiểm tra chỉ số hợp lệ thì nên kiểm tra trước khi gọi hàm này.

int Get(node head, int index){

    int k = 0;

    node p = head;

    while(p->next != NULL && k != index){

        ++k;

        p = p->next;

    }

    return p->data;

}

Lý do dùng p->next != NULL là vì chúng ta chỉ muốn đi qua các phần tử có value.

**\*Tìm kiếm trong danh sách liên kết**

Hàm tìm kiếm này sẽ trả về chỉ số của Node đầu tiên có giá trị bằng với giá trị cần tìm. Nếu không tìm thấy, chúng ta trả về -1.

int Search(node head, int value){

    int position = 0;

    for(node p = head; p != NULL; p = p->next){

        if(p->data == value){

            return position;

        }

        ++position;

    }

    return -1;

}

Chúng ta có thể sử dụng hàm này để xóa tất cả các Node trong danh sách liên kết có giá trị chỉ định như sau:

node DelByVal(node head, int value){

    int position = Search(head, value);

    while(position != -1){

        DelAt(head, position);

        position = Search(head, value);

    }

    return head;

}

### \*Duyệt danh sách liên kết

Việc duyệt danh sách liên kết cực đơn giản. Khởi tạo từ Node head, bạn cứ thế đi theo con trỏ next cho tới trước khi Node đó NULL.

 void Traverser(node head){

    printf("\n");

    for(node p = head; p != NULL; p = p->next){

        printf("%5d", p->data);

    }

}

**3. Các thuật toán**

**3.1 Sắp xếp:**

**\*Sắp xếp tăng giảm**

**Ý tưởng:**

Đầu tiên, ta sẽ cho nhập dãy số gồm n phần tử

Sau đó sử dụng cách hoán vị để thay đổi thứ tự 2 số kế nhau, nếu như số ở trước bé hơn số ở sau thì 2 số sẽ đổi chỗ cho nhau. Cứ liên tục như vậy ta sẽ có được 1 dãy số giảm dần.

Code minh họa:

#include <stdio.h>

int main(){

    int a[100];

    int n;

    printf("\nNhap so luong phan tu n = ");

    scanf("%d", &n);

    // Sap xep dung thuat toan sap xep chon

    int tg;

    for(int i = 0; i < n - 1; i++){

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

            if(a[i] < a[j]){

                // Hoan vi 2 so a[i] va a[j]

                tg = a[i];

                a[i] = a[j];

                a[j] = tg;

            }

        }

    }

    printf("\nMang da sap xep la: ");

    for(int i = 0; i < n; i++){

        printf("%5d", a[i]);

    }

}

VD: cho dãy số gồm 5 phần tử: [1 5 4 2 3]

**Lần lặp đầu tiên:**

[**1 5** 4 2 3] [**5 1** 4 2 3] Thuật toán sẽ so sánh 2 giá trị đầu và vì 1 < 5 nên thuật toán sẽ đổi chỗ 2 giá trị

[5 **1 4** 2 3] [5 **4 1** 2 3] đổi chỗ 1 và 4 do 1 < 4

[5 4 **1 2** 3] [5 4 **2 1** 3] đổi chỗ 1 và 2 do 1 < 2

[5 4 2 **1 3**] [5 4 2 **3 1**] đổi chỗ 1 và 3 do 1 < 3

**Lần lặp thứ 2:**

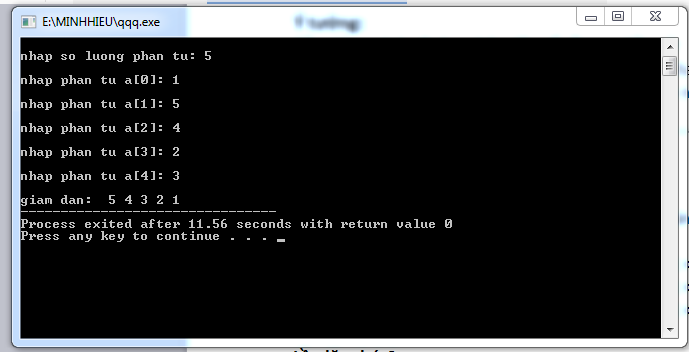
[**5 4** 2 3 1] [**5 4** 2 3 1] không đổi vì 2 giá trị 5 và 4 đã đúng thứ tự

[5 **4 2** 3 1] [5 **4 2** 3 1] không đổi

[5 4 **2 3** 1] [5 4 **3 2** 1] đổi chỗ 2 và 3 do 2 < 3

**Giảm dần: 5 4 3 2 1**

Hình ảnh minh họa:



Tương tự như sắp xếp giảm dần, sắp xếp tăng dần cũng như vậy, ta sẽ cho nhập dãy số gồm n phần tử, sau đó sử dụng swap để thay đổi thứ tự 2 số kế nhau, nếu như số ở trước lớn hơn số ở sau thì 2 số sẽ đổi chỗ cho nhau. Cứ liên tục như vậy ta sẽ có được 1 dãy số tăng dần.

Code minh họa:

#include <stdio.h>

int main(){

    int a[100];

    int n;

    printf("\nNhap so luong phan tu n = ");

    scanf("%d", &n);

    // Sap xep dung thuat toan sap xep chon

    int tg;

    for(int i = 0; i < n - 1; i++){

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

            if(a[i] > a[j]){

                // Hoan vi 2 so a[i] va a[j]

                tg = a[i];

                a[i] = a[j];

                a[j] = tg;

            }

        }

    }

    printf("\nMang da sap xep la: ");

    for(int i = 0; i < n; i++){

        printf("%5d", a[i]);

    }

}

VD: cho dãy số gồm 5 phần tử: [1 5 4 2 3]

**Lần lặp đầu tiên:**

[**1 5** 4 2 3] [**1 5** 4 2 3] Thuật toán sẽ so sánh 2 giá trị đầu và vì 1 < 5 nên thuật toán sẽ không đổi chỗ 2 giá trị

[1 **5 4** 2 3] [1 **4 5** 2 3] đổi chỗ 5 và 4 do 5 > 4

[1 4 **5 2** 3] [1 4 **2 5** 3] đổi chỗ 5 và 2 do 5 > 2

[1 4 2 **5 3**] [1 4 2 **3 5**] đổi chỗ 5 và 3 do 5 > 3

**Lần lặp thứ 2:**

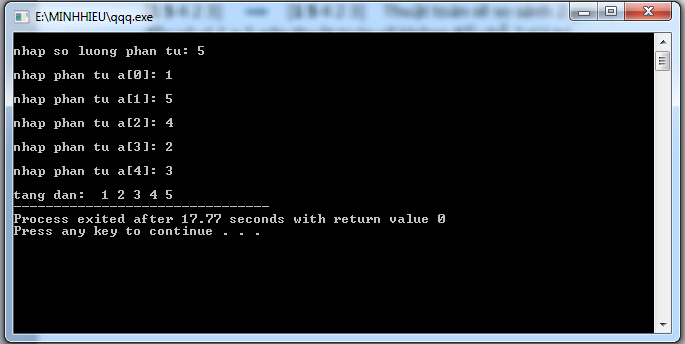
[**1 4** 2 3 5] [**1 4** 2 3 5] không đổi

[1 **4 2** 3 5] [1 **2 4** 3 5] đổi chỗ 4 và 2 do 4 > 2

[1 2 **4 3** 5] [1 2 **3 4** 5] đổi chỗ 4 và 3 do 4 >3

**Tăng dần: 1 2 3 4 5**

Hình ảnh minhhọa:



**3.2 Tìm kiếm (Search):**

**\*Tìm kiếm một phần tử bất kì trong một mảng**

Ý tưởng:

Đầu tiên ta sẽ cho nhập 1 dãy số gồm n phần tử, sau đó cho 1 biến x, chạy vòng lặp for và nếu như x = a[i] ta sẽ in ra “tìm thấy x” còn nếu không có thì in ra “không tìm thấy x”.

Code minh họa:

void tiemKiem(int a[], int &n, int &X)

{

    for(int i = 0; i<n; i++){

        if(a[i]==x);

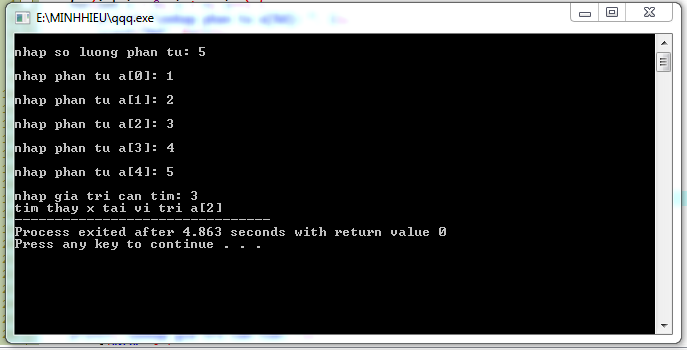
        printf("tim thay phan tu %d o vi tri a[%d]",x,i);

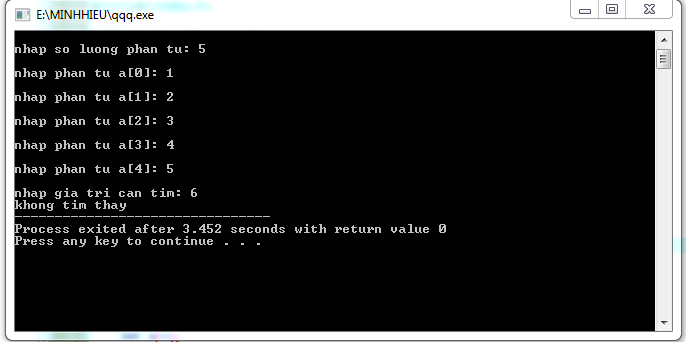
    }

    printf("khong co phan thu %d trong mang",x);

}

Chạy thử:





(khi x không được tìm thấy)

**\*Tìm kiếm số lớn nhất:**

Ý tưởng:

Đầu tiên ta đặt 1 biến ví dụ như biến max và cho bằng với a[0], sau đó cho chạy vào vòng lặp for và nếu như phần tử đứng trước a[i] bé hơn phần tử đứng sau a[i +1] thì lúc này biến max sẽ bằng a[i + 1].

Code minh họa:

int max(int a[], int n)

{

    int max = a[0];

    for (int i = 1; i < n; i++)

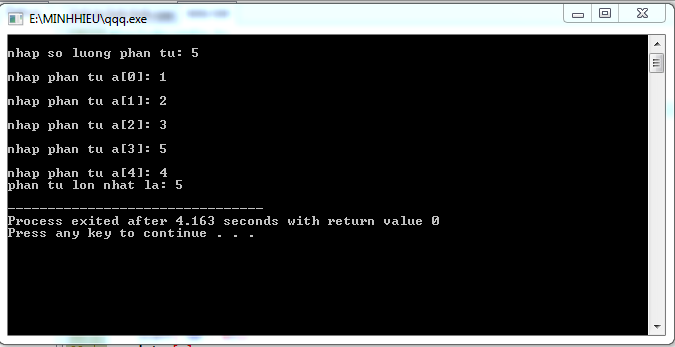
        if (max < a[i])

            max = a[i];

    return max;

}

Chạy thử:



**\*Tìm kiếm số bé nhất:**

Ý tưởng:

Đầu tiên ta đặt 1 biến ví dụ như biến min và cho bằng với a[0], sau đó cho chạy vào vòng lặp for và nếu như phần tử đứng trước a[i] lớn hơn phần tử đứng sau a[i +1] thì lúc này biến min sẽ bằng a[i + 1].

Code minh họa:

int min(int a[], int n)

{

    int min = a[0];

    for (int i = 1; i < n; i++)

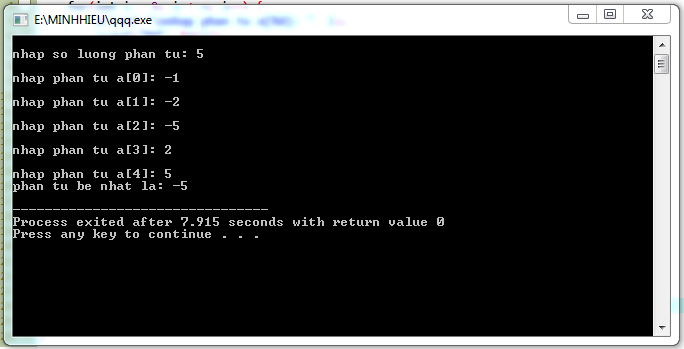
        if (min > a[i])

            min = a[i];

    return min;

}

Chạy thử:



**3.3 Thêm/bớt phần tử:**

**\*Thêm phần tử:**

Ý tưởng:

Đầu tiên ta sẽ cho nhập 1 dãy số gồm n phần tử, sau đó cho xuất dãy số đó ra, ta gọi vt sẽ là vị trí để thêm phân tử và xét nếu như vt < 0 hoặc vt > n + 1 thì sẽ return lại sau đó chạy vòng lặp for, ở đây t cũng sẽ xét ở 2 điểm 1 là vị trí trước chỗ cần thêm thì b[i] = a[i], vị trí thứ 2 là sau vị trí cần thêm cho đến hết lúc này b sẽ hơn a 1 phần tử nên ta sẽ cho b[i + 1] = a[i] và kết thúc là xuất dãy b ra.

Code minh họa:

void ThemPhanTu(int a[], int &n, int val, int pos){

    if(n >= MAX){

        return;

    }

    if(pos < 0){

        pos = 0;

    }

    else if(pos > n){

        pos = n;

    }

    for(int i = n; i > pos; i--){

        a[i] = a[i-1];

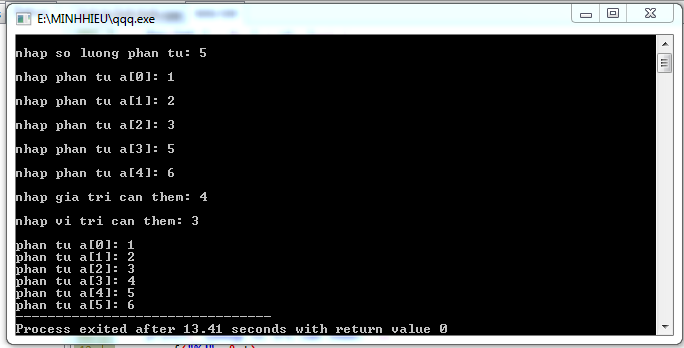
    }

    a[pos] = val;

    ++n;

}

Chạy thử:



**Tài liệu tham khảo**

1. Thuật toán sắp xếp chọn

**<https://www.youtube.com/watch?v=IMWCALHGxYk>**

1. Sắp xếp chèn

**<https://www.youtube.com/watch?v=rHZiWas7z_s>**

1. Sắp xếp nổi bọt

**<https://sinhvientot.net/thuat-toan-sap-xep-noi-bot-bubblesort/>**

1. Thuật toán tìm kiếm nhị phân

**<https://nguyenvanhieu.vn/thuat-toan-tim-kiem-nhi-phan/?fbclid=IwAR3_yWK2LNlkYS1_2rYvNvx_460WR-Nw8zjAOXJZfbhZ7_qHMQZRwRm8vvk>**

1. Đọc ghi file:

**<https://nguyenvanhieu.vn/doc-ghi-file-trong-c/?fbclid=IwAR1Ly-mZ603lfm_KTnhZ9ovwhXz538mZk1wX_qCB_pHeoHCw5pTyGqy01EA>**

**TỔNG KẾT**

1. Kết quả đạt được:

Hoàn thành những nội dung được yêu cầu từ phía giảng viên.

Chương trình quản lý nhân viên chạy thành công.

Cả nhóm biết và học được những lý thuyết và những thuật toán cơ bản trong C.

1. Định hướng tương lai:

Chương trình quản lý nhân viên sẽ ngày càng được hoàn thiện hơn và sẽ được tối ưu hóa hơn để có thêm nhiều chức năng mới trong tương lai.

**KẾT THÚC**

|  |
| --- |
|  |