

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**  
**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**  
**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN**  
**LẬP TRÌNH NÂNG CAO**  
**ĐỀ TÀI:**

***CHƯƠNG TRÌNH QUẢN LÝ SINH VIÊN***

Giảng viên hướng dẫn: TRẦN THỊ DUNG

Sinh viên thực hiện: NGUYỄN TUẤN ĐẠT

VÕ THÀNH NHÂN

HOÀNG MINH TÀI

Lớp: CQ.60.CNTT

Tp. Hồ Chí Minh, năm 2020

## MỤC LỤC

<b>MỤC LỤC.....</b>	<b>1</b>
<b>MỤC LỤC HÌNH ẢNH.....</b>	<b>2</b>
<b>1. Tổng quan về đề tài quản lý sinh viên :.....</b>	<b>3</b>
<b>2. Cơ sở lý thuyết:.....</b>	<b>3</b>
<b>2.1. Làm việc với tệp .....</b>	<b>3</b>
<b>2.1.1. Các kiểu file .....</b>	<b>3</b>
<b>2.1.2. Các thao tác với file .....</b>	<b>4</b>
<b>2.1.3. Lý thuyết và code minh họa.....</b>	<b>4</b>
<b>2.2. Cấu trúc danh sách liên kết đơn .....</b>	<b>10</b>
<b>2.2.1. Danh sách liên kết đơn là gì? .....</b>	<b>10</b>
<b>2.2.2. Đặc điểm của danh sách liên kết đơn .....</b>	<b>10</b>
<b>2.2.3. Cài đặt danh sách liên kết đơn .....</b>	<b>11</b>
<b>2.2.4. Thêm vào phần tử .....</b>	<b>12</b>
<b>2.2.5. Xóa một phần tử .....</b>	<b>13</b>
<b>2.3. Các thuật toán sắp xếp .....</b>	<b>15</b>
<b>2.4. Các thuật toán tìm kiếm .....</b>	<b>17</b>
<b>3. Phân tích và giải thích về code.....</b>	<b>21</b>
<b>3.1. Sơ lược về chương trình.....</b>	<b>21</b>
<b>3.2. Nhập danh sách.....</b>	<b>21</b>
<b>3.3. In danh sách.....</b>	<b>22</b>
<b>3.4. Xếp loại sinh viên.....</b>	<b>22</b>
<b>3.5. Tìm kiếm sinh viên.....</b>	<b>23</b>
<b>3.6. Sắp xếp sinh viên theo điểm giảm dần.....</b>	<b>24</b>
<b>3.7. Sắp xếp tên theo thứ tự A-Z.....</b>	<b>26</b>
<b>3.8. Ghi file dạng văn bản.....</b>	<b>27</b>
<b>3.9. Đọc file dạng văn bản.....</b>	<b>27</b>
<b>3.10. Hướng dẫn sử dụng chương trình.....</b>	<b>28</b>
<b>KẾT LUẬN.....</b>	<b>29</b>

## MỤC LỤC HÌNH ẢNH

Hình 2.1: Tạo và ghi file văn bản thành công bằng <code>fprintf</code> .....	7
Hình 2.2: Đọc file văn bản thành công bằng <code>fscanf</code> .....	8
Hình 2.3: Đọc file nhị phân thành công bằng <code>fread</code> .....	9
Hình 2.4: Đọc file nhị phân thành công bằng <code>fread</code> .....	10
Hình 2.5: Thêm node vào đầu danh sách .....	12
Hình 2.6: Thêm node vào cuối danh sách .....	13
Hình 2.7: Xóa node ở đầu danh sách .....	14
Hình 2.8: Xóa node ở sau một phần tử bất kì .....	15
Hình 2.9: Kết quả sau khi áp dụng các thuật toán sắp xếp là như nhau ( bỏ qua chênh lệch thời gian) .....	17
Hình 2.10: Kết quả khi áp dụng thuật toán tìm kiếm tuyến tính.....	19
Hình 2.11: Kết quả khi áp dụng thuật toán tìm kiếm nhị phân.....	21
Hình 3.1: Nhập vào các yếu tố cần thiết (MSSV, họ tên, giới tính,...)....	22
Hình 3.2: In ra danh sách sinh viên vừa nhập.....	22
Hình 3.3: Tìm kiếm sinh viên theo tên.....	24
Hình 3.4: Sắp xếp sinh viên theo điểm trung bình giảm dần.....	26
Hình 3.5: Sắp xếp sinh viên theo tên .....	26

## **NỘI DUNG**

### **1. Tổng quan về đề tài quản lý sinh viên**

- Lý do chọn đề tài

Trong hệ thống giáo dục hiện nay ở tất cả trường học việc quản lý học sinh sinh viên trở nên vô cùng cấp thiết và quan trọng. Đặc biệt với các trường đại học với một số lượng sinh viên lớn, nhu cầu quản lý sinh viên được đặt ra, từ đó xuất hiện các chương trình quản lý sinh viên.

Nắm bắt được nhu cầu đó và nhằm nâng cao khả năng tư duy cũng như sử dụng ngôn ngữ lập trình C, các phần mềm CFree, DevC++....nhóm đã tiến hành nghiên cứu và phát triển đề tài “Chương trình quản lý sinh viên” để giúp việc quản lý điểm trong nhà trường trở nên hiệu quả hơn.

- Mục đích và đối tượng của đề tài

Chương trình được tạo ra để người dùng nhập thông tin, bổ sung thông tin và thống kê danh sách sinh viên thông qua điểm số, môn học,...

Đối tượng sử dụng chương trình có thể là các nhân viên văn phòng trường hay giảng viên sử dụng để quản lý sinh viên trong trường hay trong bộ môn.

### **2. Cơ sở lý thuyết**

#### **2.1. Làm việc với tệp**

##### **2.1.1. Các kiểu file**

- File 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 editor thông dụng như Notepad, Notepad++, Sublime Text,...

Khi bạn mở các file này bằng các text editor 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.

- File nhị phân – Binary files

File nhị phân thường có đuôi mở rộng là **.bin**

Thay vì lưu trữ dưới dạng văn bản thuần túy, 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 editor kể trên.

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

### **2.1.2. Các thao tác với file**

- Tạo mới một file
- Mở một file đã có
- Đóng file đang mở
- Đọc thông tin từ file/ Ghi thông tin ra file

### **2.1.3. Lý thuyết và code minh họa**

- 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.
- Muốn thao tác với 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()` có sẵn trong thư viện `stdio.h` theo cú pháp sau:

```
f = fopen("file location", "mode");
```

Trong đó, `mode` là tham số nhiệm vụ chúng ta muốn thực hiện với file.

**Bảng 2.1.** Các giá trị có thể có của tham số `mode`

Mode	Ý nghĩa	Nếu file không tồn tại
r	Mở file chỉ cho phép đọc	Nếu file không tồn tại, <code>fopen()</code> trả về NULL.
rb	Mở file chỉ cho phép đọc dưới dạng nhị phân.	Nếu file không tồn tại, <code>fopen()</code> 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, <code>fopen()</code> 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.

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()** theo cú pháp

**fclose(fptr) ;**

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

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

- Ghi file văn bản bằng **fprintf**

```
#include<stdio.h>
int main() {
    int n;
    FILE *f;
    f=fopen("C:/Users/hoang/OneDrive/Desktop/program.txt"."a");

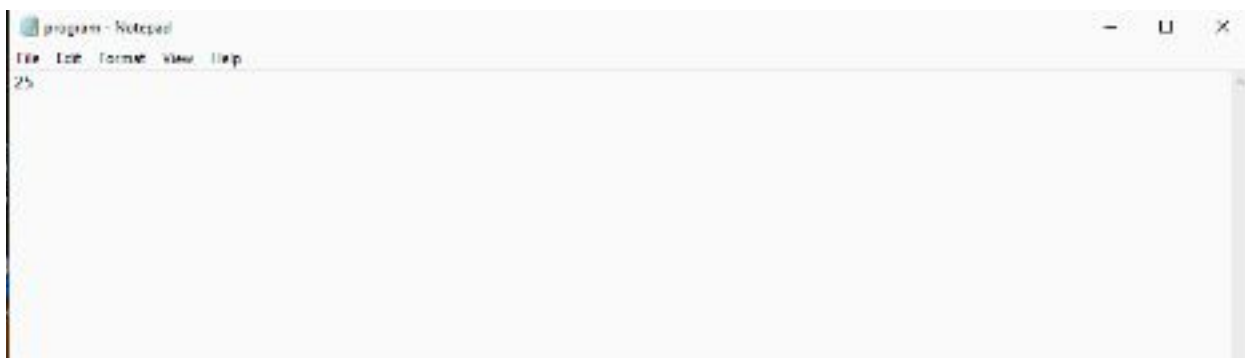
    printf("Nhap vao mot so bat ky ");
    scanf("%d",&n);

    fprintf(f,"%d",n);

    fclose(f);

    return 0;
}
```

Sau khi chạy chương trình, tại địa chỉ file đã trích dẫn, giá trị của n được nhập trong quá trình chạy chương trình sẽ được lưu. Cụ thể:



Hình 2.1. Tạo và ghi file văn bản thành công bằng fprintf

- Đọc file văn bản bằng fscanf

```
#include<stdio.h>
#include<stdlib.h>
int main() {
    int n;
    FILE *f;
    if
((f=fopen("C:/Users/hoang/OneDrive/Desktop/program.txt"."r"))==NULL) {

    printf("Error! không tìm thấy file ");

    exit(1);}
```



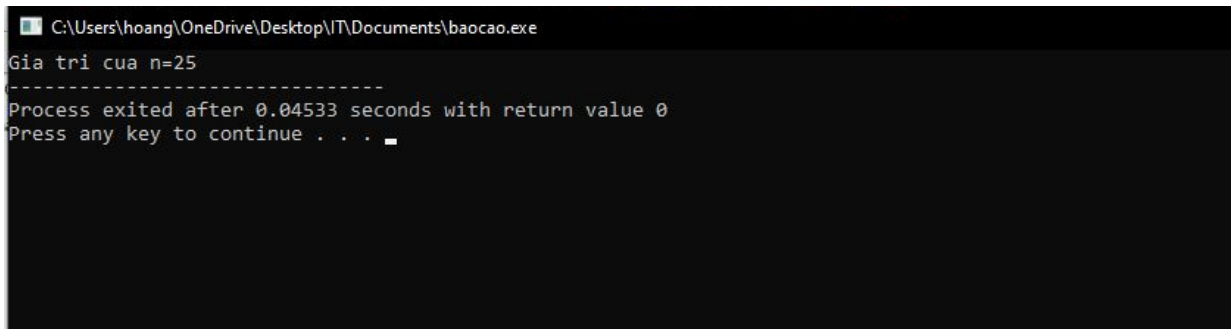
```

        fscanf(f,"%d",&n);
        printf("Gia tri cua n= %d",n);
        fclose(f);

        return 0;
}

```

Khi chạy chương trình này, thì nó sẽ đọc file program.txt đã được ghi ở trên để đưa ra giá trị của n trong màn hình console:



Hình 2.2. Đọc file văn bản thành công bằng `fscanf`

- Ghi file nhị phân bằng `fwrite`

```

#include<stdio.h>
#include<stdlib.h>
struct conso
{
    int t1,t2,t3;
}
int main(){
    int n;
    struct conso num;
    FILE *f;
    if
    ((f=fopen("C:/Users/hoang/OneDrive/Desktop/program.bin"."ab")) !=NULL
    ){

        for (n=1;n<5;n++){
            num.t1=n;
            num.t2=3*n;
            num.t3=3*n+3;
            fwrite(&num,sizeof(struct conso),1,f);
        }
    }
}

```

```

fclose(f);

return 0;
}

```

Chương trình sẽ tạo ra một file **program.bin** trên ổ đĩa C theo đường dẫn. Chương trình này đã khai báo 1 kiểu dữ liệu cấu trúc lưu 3 giá trị số **t1**, **t2**, **t3**; Và nó được sử dụng trong hàm main có tên biến là **num**.

Đọc file nhị phân bằng **fread**

```

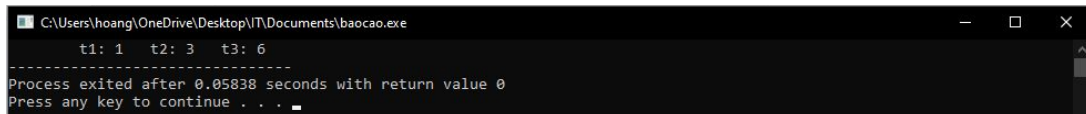
#include<stdio.h>
#include<stdlib.h>
struct conso
{
    int t1,t2,t3;
}
int main(){
    int n;
    struct conso num;
    FILE *f;
    if
((f=fopen("C:/Users/hoang/OneDrive/Desktop/program.bin"."ab"))==NULL
){
    printf("Error! không tìm thấy file ");

    exit(1);
}

for(n=1;n<2;n++){
    fread(&num,sizeof(struct conso),1,f);
    printf("\tt1: %d\tt2: %d\tt3: %d");
}
fclose(f);
return 0;
}

```

Chương trình giúp in ra các giá trị **t1**, **t2**, **t3** trong file **program.bin** được ghi ở chương trình trên. Cụ thể:



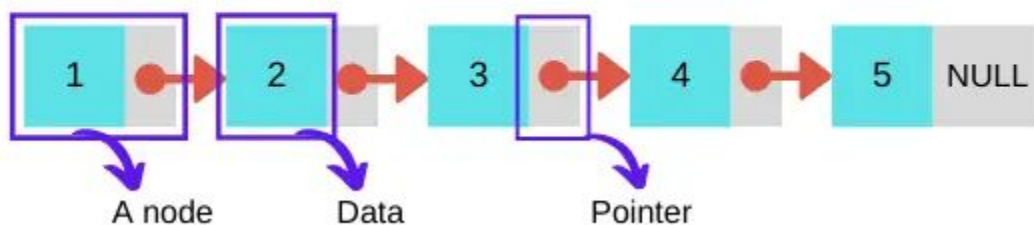
Hình 2.3. Đọc file nhị phân thành công bằng **fread**

## 2.2. Cấu trúc danh sách liên kết đơn

### 2.2.1. Danh sách liên kết đơn là gì?

Danh sách liên kết đơn (Single Linked List) là một cấu trúc dữ liệu động, nó là một danh sách mà mỗi phần tử đều liên kết với phần tử đứng sau nó trong danh sách. Mỗi phần tử (được gọi là một node hay nút) trong danh sách liên kết đơn là một cấu trúc có hai thành phần:

- Thành phần dữ liệu: lưu thông tin về bản thân phần tử đó.
- Thành phần liên kết: lưu địa chỉ phần tử đứng sau trong danh sách, nếu phần tử đó là phần tử cuối cùng thì thành phần này bằng NULL.



Hình 2.4. Đọc file nhị phân thành công bằng `fread`

### 2.2.2. Đặc điểm của danh sách liên kết đơn

Do danh sách liên kết đơn là một cấu trúc dữ liệu động, được tạo nên nhờ việc cấp phát động nên nó có một số đặc điểm sau đây:

- Được cấp phát bộ nhớ khi chạy chương trình
- Có thể thay đổi kích thước qua việc thêm, xóa phần tử
- Kích thước tối đa phụ thuộc vào bộ nhớ khả dụng của RAM
- Các phần tử được lưu trữ ngẫu nhiên (không liên tiếp) trong RAM

Và do tính liên kết của phần tử đầu và phần tử đứng sau nó trong danh sách liên kết đơn, nó có các đặc điểm sau:

- Chỉ cần nắm được phần tử đầu và cuối là có thể quản lý được danh sách
- Truy cập tới phần tử ngẫu nhiên phải duyệt từ đầu đến vị trí đó

- Chỉ có thể tìm kiếm tuyến tính một phần tử

### 2.2.3. Cài đặt danh sách liên kết đơn

- Tạo node

Một node gồm hai thành phần là thành phần dữ liệu và thành phần liên kết. Thành phần dữ liệu có thể là kiểu dữ liệu có sẵn hoặc bạn tự định nghĩa (struct), để đơn giản ta sẽ sử dụng kiểu int cho phần dữ liệu. Thành phần liên kết là địa chỉ đương nhiên sẽ là con trỏ, con trỏ này trỏ đến node tiếp theo, do đó, con trỏ này là con trỏ trỏ vào một node. Cụ thể:

```
struct node
{
    int data;
    node *next;
};
```

- Khởi tạo node

Để tạo một node mới, ta thực hiện cấp phát động cho node mới, khởi tạo giá trị ban đầu và trả về địa chỉ của node mới được cấp phát.

```
node khoitaonode(int n){
    node *t=new node;
    t->data=n;
    t->next=NULL;
    free(t);
}
```

- Tạo danh sách liên kết đơn

Quá trình này gồm 2 giai đoạn:

- Một là tạo ra một cấu trúc chứa hai phần tử đầu và cuối. Vì mỗi phần tử đều liên kết với phần tử kế nó vậy nên ta chỉ cần biết phần tử đầu và cuối là có thể quản lý được danh sách này

```
struct lkdon{
    node *head;
    node *tail;};
```

- Hai là gán cho chúng giá trị là NULL vì khi mới khởi tạo, danh sách chưa có phần tử nào.

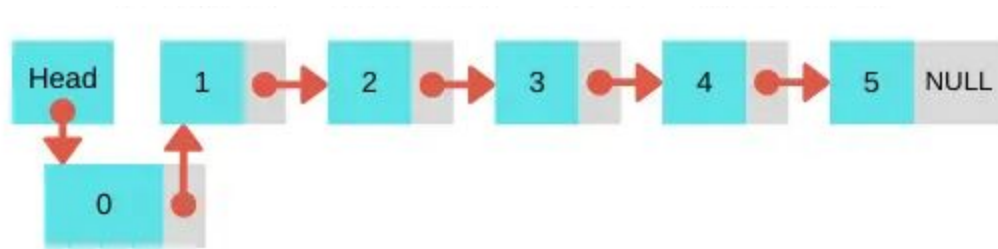
```
void khoitaods(lkdon &s){
    s.head=NULL;
    s.tail=NULL;
}
```

## 2.2.4. Thêm phần tử vào danh sách

- Thêm vào đầu

Để thêm node vào đầu danh sách, đầu tiên ta cần kiểm tra xem danh sách đó có rỗng hay không, nếu danh sách rỗng, ta chỉ cần gán head và tail của danh sách bằng node đó. Ngược lại nếu danh sách không rỗng, ta thực hiện trở thành phần liên kết vào head, sau đó gán lại head bằng node mới. Cụ thể:

```
void themdau(node *t,lkdon &s){
    if (s.head==NULL){
        s.head=t;
        s.tail=t;
    }
    else{
        t->next=s.head;
        s.head=t;
    }
}
```



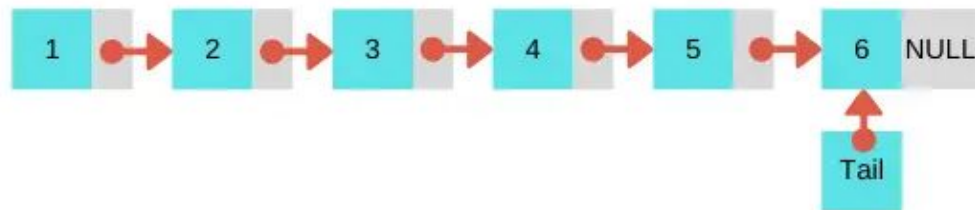
Hình 2.5. Thêm node vào đầu danh sách

Ta thêm node có data bằng 0 vào danh sách. Ta thực hiện trở next của node đó vào head của danh sách (chính là node đầu tiên của danh sách có data bằng 1), sau đó ta trở head vào node có data 0 vừa được thêm.

- Thêm vào cuối

d

```
void themcuoi(node *t, lkdon &s){
    if (s.head==NULL){
        s.head=t;
        s.tail=t;
    }
    else{
        t->next=s.tail;
        s.tail=t;
    }
}
```



Hình 2.6. Thêm node vào cuối danh sách

Ta thực hiện thêm node có data bằng 6 vào danh sách. Tail hiện tại là node có data 5, thực hiện gán `tail->next` bằng node mới để nối thêm nó vào đuôi danh sách, lúc này node mới trở thành phần tử cuối danh sách nên ta gán tail lại bằng node mới.

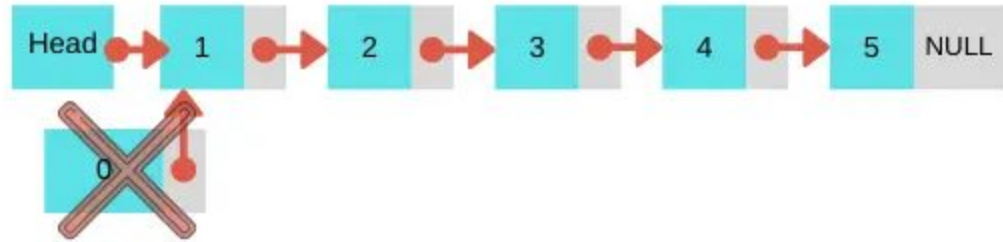
## 2.2.5. Xóa phần tử khỏi danh sách

- Xóa ở đầu

```
int xoadau(lkdon &s, int &x){
    if (s.head==NULL) return 0;
    else{
        node *t=s.head;
        x=t->data;
        s.head=t->next;
        delete t;
        if (s.head==NULL){
            s.tail=NULL;
            return 1;
        }
    }
}
```

Để xóa phần tử ở đầu danh sách, ta kiểm tra xem danh sách đó có rỗng hay không, nếu rỗng, ta không cần xóa, trả về kết quả là 0. Nếu

danh sách không rỗng, ta thực hiện lưu node head lại, sau đó gán head bằng next của node head, sau đó xóa node head đi. Tiếp theo ta cần kiểm tra xem danh sách vừa bị xóa đi node head có rỗng hay không, nếu rỗng ta gán lại tail bằng NULL luôn sau đó trả về kết quả 1.

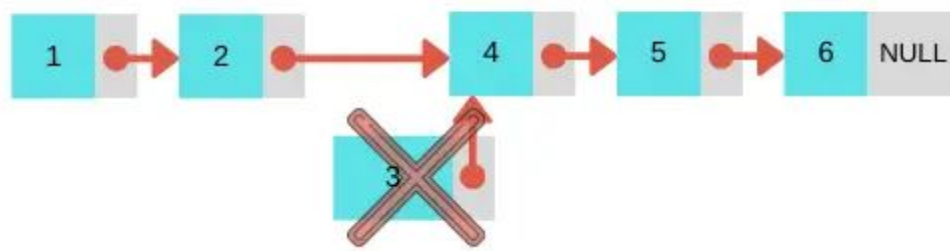


Hình 2.7. Xóa node ở đầu danh sách

- Xóa ở cuối

```
int xoacuoi(lkdon &s, node *q, int &x){
    node *p=q->next;
    if(q==NULL) return 0;
    else{
        if(s.tail==p){
            s.tail=q;
            q->next=p->next;
            x=p->data;
            delete p;
            return 1;
        }
    }
    return 0;
}
```

Để xóa một node p (ở cuối) sau node q, cần kiểm tra xem node q có NULL hay không, nếu node q NULL thì không tồn tại trong danh sách, do đó trả về 0, không xóa. Nếu node q khác NULL nhưng next của q là NULL, tức là p bằng NULL thì không xóa, trả về 0 (do sau q không có node nào cả, q là tail). Nếu node p tồn tại, ta thực hiện kiểm tra xem node p có phải là tail hay không, nếu node p là tail thì gán lại tail là q, tức là node trước đó để xóa node p đi.



Hình 2.8. Xóa node ở sau một phần tử bất kì

### 2.3. Các thuật toán sắp xếp

- Thuật toán sắp xếp chọn (Selection sort)
  - Ý tưởng của thuật toán

Thuật toán sắp xếp chọn sẽ sắp xếp một mảng bằng cách đi tìm phần tử có giá trị nhỏ nhất(giả sử với sắp xếp mảng tăng dần) trong đoạn đoạn chưa được sắp xếp và đổi chỗ phần tử nhỏ nhất đó với phần tử ở đầu đoạn chưa được sắp xếp(không phải đầu mảng). Thuật toán sẽ chia mảng làm 2 mảng con.

1. Một mảng con đã được sắp xếp.
2. Một mảng con chưa được sắp xếp.

Tại mỗi bước lặp của thuật toán, phần tử nhỏ nhất ở mảng con chưa được sắp xếp sẽ được di chuyển về đoạn đã sắp xếp.

Thuật toán cụ thể như sau:

**Bước 1:**  $i = 0$  ;

**Bước 2:** Tìm phần tử  $a[\text{min}]$  nhỏ nhất trong dãy hiện hành từ  $a[i]$  đến  $a[n]$ .

**Bước 3:** Dùng hàm doicho để đổi chỗ  $a[i]$  và  $a[\text{min}]$ .

**Bước 4:** Nếu  $i < n - 1$  thì  $i = i + 1$ ; Lặp lại bước 2; Ngược lại: Dừng.

```
#include<stdio.h>
#include<conio.h>
void nhap(int a[],int n){
    for (int i=0;i<n;i++){
        printf("nhap so thu %d: ",i+1);
        scanf("%d",&a[i]);
    }
}
```



```

void doicho(int a[],int i,int j){
    int sx=a[i];
    a[i]=a[j];
    a[j]=sx;
}
void xuat(int a[],int n){
    for (int i=0;i<n;i++){
        printf("%5d",a[i]);
    }
}
int main(){
    int a[10],n;
    printf("nhap so phan tu trong day ");
    scanf("%d",&n);
    nhap(a,n);
    for (int i=0;i<n-1;i++){
        for (int j=1;j<n;j++){
            if (a[i]>a[j])
                doicho(a,i,j);
        }
    }
    printf("day so tang dan la:\n");
    xuat(a,n);
}

```

- Thuật toán sắp xếp nổi bọt (Bubble Sort)

- Ý tưởng của thuật toán

Ý chính của thuật toán là xuất phát từ cuối (hoặc đầu) dãy, đổi chỗ các cặp phần tử kế cận để đưa phần tử nhỏ (lớn) hơn trong cặp phần tử đó về vị trí đứng đầu (cuối) dãy hiện hành, sau đó sẽ không xét đến nó ở vị trí tiếp theo, do vậy ở lần xử lý thứ  $i$  sẽ có vị trí đầu dãy là  $i$ . Lặp lại xử lý trên cho đến khi không còn cặp phần tử nào để xét.

Thuật toán cụ thể như sau:

**Bước 1:**  $i=0$ .

**Bước 2:** Lần lượt so sánh và đổi chỗ (nếu cần) từ phải sang trái đối với các phần tử từ  $a[n]$  đến  $a[i]$ . với biến gán  $j=n-i$ . và lặp lại khi  $j>i$ .

**Bước 3:**  $i=i+1$ .

**Bước 4:** Nếu  $i < n$ , quay lại Bước 2.

Ngược lại, dừng, dãy đã cho đã sắp xếp đúng vị trí.

```

#include<stdio.h>

#include<conio.h>
void nhap(int a[],int n){
    for (int i=0;i<n;i++){

```

```

        printf("nhap so thu %d: ",i+1);
        scanf("%d",&a[i]);
    }
}
void doichonoibot(int a,int j){
    int sx=a;
    a=j;
    j=sx;
}
void xuat(int a[],int n){
    for (int i=0;i<n;i++){
        printf("%5d",a[i]);
    }
}
int main(){
    int a[10],n;
    printf("nhap so phan tu trong day ");
    scanf("%d",&n);
    nhap(a,n);
    for (int i=0;i<n-2;i++){

        for (int j=n-1;j>i;j--){
            if (a[j]<a[j-1]){
                doichonoibot(a[j],a[j-1]);
            }
        }
    }
    printf("day so tang dan la:\n");
    xuat(a,n);
}

```

```

nhap so phan tu trong day 3
nhap so thu 1: 3
nhap so thu 2: 2
nhap so thu 3: 1
day so tang dan la:
 1    2    3
-----
Process exited after 4.789 seconds with return value 0
Press any key to continue . . .

```

Hình 2.9. Kết quả sau khi áp dụng các thuật toán sắp xếp là như nhau (bỏ qua chênh lệch thời gian)

## 2.4. Các thuật toán tìm kiếm

- Tìm kiếm tuyến tính
  - Ý tưởng của thuật toán: so sánh phần tử cần tìm với tất cả các phần tử có trong mảng hoặc danh sách cần tìm. Chạy từ phần tử đầu đến cuối và so sánh từng đôi một, nếu bằng thì thông báo có, ngược lại nếu đã đi hết dãy mà vẫn chưa có phần tử nào thỏa mãn thì cho kết quả là không tìm thấy.

- Thuật toán cụ thể:

Bước 1: Khởi tạo biến i và gán biến i bằng 0;

Bước 2: so sánh a[i] với giá trị cần tìm.

+ Nếu tìm được giá trị a[i] bằng giá trị cần tìm thì dừng lại và dừng. Ngược lại.

+ Nếu a[i] khác giá trị cần tìm thì sang bước 3.

Bước 3: Tăng i lên một đơn vị, nếu i bằng số phần tử trừ 1 của mảng thì dừng lại và cho kết quả là không tìm thấy. Ngược lại quay lại bước 2.

```
#include<stdio.h>
#include<conio.h>
void nhapmang(int a[], int &n)
{
    printf("Cho biet so phan tu cua mang: ");
    scanf("%d", &n);
    for (int i = 0; i<n; i++)
    {
        printf("Gia tri phan tu a[%d]=", i);
        scanf("%d", &a[i]);
    }
}
void xuatmang(int a[], int n)
{
    for (int i = 0; i<n; i++)
        printf("%5d", a[i]);
}
int timkiemtuyentinh(int a[], int n, int x)
{
    for (int i = 0; i < n;i++)
        if (a[i] == x)
            return 1;
    return 0;
}

int main()
{
    int a[50];
    int n;
    int x;
    nhapmang(a,n);
    printf("Nhap phan tu can tim");
    scanf("%d", &x);
    int b=timkiemtuyentinh(a,n,x);
    if (b == 1)
        printf("%d co trong day",x);
    else
```

```

    {
        printf("%d không có trong day", x);
    }
}

```

```

Cho biet so phan tu cua mang: 5
Gia tri phan tu a[0]=9
Gia tri phan tu a[1]=1
Gia tri phan tu a[2]=36
Gia tri phan tu a[3]=48
Gia tri phan tu a[4]=15
Nhap phan tu can tim12
12 không có trong day
-----
Process exited after 14.18 seconds with return value 0
Press any key to continue . . .

```

Hình 2.10. Kết quả khi áp dụng thuật toán tìm kiếm tuyến tính

- Tìm kiếm nhị phân
  - Ý tưởng của thuật toán: là tìm kiếm dựa trên việc chia đôi khoảng đang xét sau mỗi lần lặp, sau đó xét tiếp trong nửa khoảng có khả năng chứa giá trị cần tìm, cứ như vậy cho đến khi không chia đôi khoảng được nữa. Thuật toán tìm kiếm nhị phân **chỉ áp dụng được cho danh sách đã có thứ tự hay đã được sắp xếp**.
  - Thuật toán cụ thể : ví dụ cho một mảng A có n phần tử bắt đầu từ vị trí 0, mảng A được **sắp xếp tăng dần** . Để tìm phần tử có giá trị x trong mảng A chúng ta sẽ cài đặt thuật toán tìm kiếm nhị phân như sau:

**Bước 1:** Gán  $left = 0$ ,  $right = n - 1$ .

**Bước 2:** Gán  $mid = (left + right) / 2$  (lấy phần nguyên, đây là phần tử chính giữa của khoảng hiện tại)

- Nếu như  $A[mid] == x$ : Dừng lại và trả về giá trị của mid (chính là vị trí của x trong mảng A).
- Nếu như  $A[mid] > x$  (có thể x nằm trong nửa khoảng trước):
  - $right = mid - 1$  // giới hạn khoảng tìm kiếm lại là nửa khoảng trước
- Nếu như  $A[mid] < x$  (có thể x nằm trong nửa khoảng sau):

- $left = mid + 1$  // giới hạn khoảng tìm kiếm lại là nửa khoảng sau

**Bước 3:** Nếu  $left \leq right$ :

- Đúng thì quay lại bước 2 (còn chia đôi được).
- Sai thì dừng và trả về kết quả -1 (không tìm thấy x)

```
#include<stdio.h>
void nhapmang(int a[], int &n){
    printf("Cho biet so phan tu cua mang: ");
    scanf("%d", &n);
    for (int i = 0; i<n; i++)
    {
        printf("Gia tri phan tu a[%d]=", i);
        scanf("%d", &a[i]);
    }
}
void xuatmang(int a[], int n){
    for (int i = 0; i<n; i++)
        printf("%5d", a[i]);
}
int timkiemnhiphan(int A[], int n, int x){
    int left = 0;
    int right = n - 1;
    int mid;
    while (left <= right){
        mid = (left + right) / 2;
        if (A[mid] == x)
            return mid;
        if (A[mid] > x)
            right = mid - 1;
        else if (A[mid] < x)
            left = mid + 1; }
    return -1;}
int main(){
    int a[50];
    int n;
    int x;
    nhapmang(a,n);
    printf("Nhap phan tu can tim");
    scanf("%d", &x);
    int b=timkiemnhiphan(a,n,x);
    if (b == 1)
        printf("%d co trong day",x);
    else
    {
        printf("%d khong co trong day", x);}}}
```

```

Cho biet so phan tu cua mang: 4
Gia tri phan tu a[0]=12
Gia tri phan tu a[1]=23
Gia tri phan tu a[2]=34
Gia tri phan tu a[3]=45
Nhap phan tu can tim56
56 khong co trong day
-----
Process exited after 24.52 seconds with return value 0
Press any key to continue . . .

```

Hình 2.11. **Kết quả khi áp dụng thuật toán tìm kiếm nhị phân**

### 3. Phân tích và giải thích về code.

#### 3.1.Sơ lược về Chương trình:

**Chương trình có tên: Bai\_tap\_mini.**

- Chương trình sử dụng cấu trúc struct Sinhvien để lưu trữ các phần tử liên quan đến tên sinh viên như họ và tên, giới tính, mã số sinh viên, ngày tháng năm sinh, điểm trung bình.

- Chương trình sẽ cho phép xử lý các chức năng sau: nhập danh sách, hiển thị danh sách(đọc file), sắp xếp theo tên a-z và điểm trung bình giảm dần, tìm kiếm theo tên, ghi thông tin ra file, thoát chương trình.

#### 3.2.Nhập danh sách:

Cho phép người sử dụng nhập vào thông tin sinh viên gồm:

- Mã số SV được khai báo kiểu char có tối đa 10 kí tự.
- Họ tên sinh viên được khai báo kiểu char có tối đa 40 kí tự.
- Ngày, tháng và năm sinh được khai báo theo kiểu integer.
- Giới tính được khai báo theo kiểu char có tối đa 5 kí tự.
- Điểm TB học kỳ gần nhất(quy về hệ 10) được khai báo theo kiểu float.

```

struct Sinhvien{

    char hoten[40];

```

```

    char mssv[10];

    char gioitinh[5];

    int ngay,thang,nam;

    float dtb;

} ;

```

```

===== MENU =====
1. Them n Sinh vien vao danh sach va Hien thi danh sach sinh vien.
2. Sap xep theo ten.
3. Sap xep theo diem tb giam dan.
4. Tim sinh vien theo ten.
5. Ghi thong tin sinh vien ra file.
0. Thoat chuong trinh.
Nhap su lua chon cua ban: 1
Nhap so sv ban muon lam viec: 1
Nhap sinh vien 1
MSSV: 123456

Ho va ten: nguyen tuan dat

Gioi tinh: nam

Ngay, thang, nam: 16 1 2001

Nhap DTB: 10

```

Hình 3.1. Nhập vào các yếu tố cần thiết (MSSV, họ tên, giới tính,...)

### 3.3. In danh sách:

Chương trình in ra danh sách tất cả sinh viên được lưu trong mảng bao gồm cả việc mở file có sẵn.

Danh sách sinh viên được in theo cột như hình.

MSSV	Ho va ten	Gioi Tinh	Ngay	Thang	Nam	DTB	Xep loai
123456	nguyen tuan dat	nam	16	1	2001	10.0	G

Hình 3.2. In ra danh sách sinh viên vừa nhập

### 3.4. Xếp loại sinh viên

```

if((s->dtb)<=5){
    printf("TB");
}
else if((s->dtb)>5&&(s->dtb)<=8){
    printf("K");
}
else{
    printf("G");
}

```

```
}
```

Để xếp loại sinh viên ta sử dụng hàm if để đặt điều kiện cho học lực, dưới 5 sẽ là TB, từ 5->8 là Khá và từ 8 trở lên sẽ là Giỏi.

### 3.5. Tìm kiếm sinh viên:

```
void timkiem(Sinhvien *s,int n){
    char ten[30];
    fflush(stdin);
    printf("\n Nhập ten sinh vien can tim: ");
    gets(ten);
    for(int i=0;i<n;i++){
        if(strcmp(ten,(s+i)->hoten)==0){
            printf("%-10s %-20s %-5s %-5s %-5s %-5s %-8.1s\n",
                "MSSV", "Ho va ten", "Gioi Tinh", "Ngay", "Thang", "Nam", "DTB\n");
            printf("%-10s %-20s %-10s %-5d %-5d %-5d %-8.1f\n",
                (s+i)->mssv, (s+i)->hoten, (s+i)->gioitinh, (s+i)->ngay,
                (s+i)->thang, (s+i)->nam, (s+i)->dtb);
        }
        else{
            printf("Khong ton tai sinh vien trong danh sach! ahihi");
        }
    }
}
```

**Bước 1:** Nhập tên sinh viên cần tìm kiếm.

**Bước 2:** Đặt điều kiện `if(strcmp(ten,(s+i)->hoten)==0)` trong dòng lặp for. Dùng lệnh `strcmp` để so sánh hai chuỗi, ví dụ: ten là chuỗi 1, hoten là chuỗi 2. Nếu giá trị trả về <0, thì chuỗi 1 sẽ ngắn hơn chuỗi 2; nếu giá trị trả >0 thì chuỗi 2 sẽ ngắn hơn chuỗi 1 và nếu giá trị trả về =0 thì 2 chuỗi sẽ bằng nhau.

**Bước 3:** Sau đó xuất thông tin sinh viên ra màn hình.



```
C:\Users\admin\Documents\PROGRAM-C & C++\Baitapnho.exe

Nhap ten sinh vien can tim: nguyen tuan dat
MSSV      Ho va ten      Gioi Tinh Ngay  Thang Nam  D
1234      nguyen tuan dat  nam      16    1    2001  10.0
deo co

Ban co muon tiep tục chương trình không? (an 1 de tiếp tục): █
```

Hình 3.3. Tìm kiếm sinh viên theo tên

### 3.6. Sắp xếp điểm sinh viên giảm dần.

**Bước 1:** Sử dụng 2 dòng lặp for int i và int j. Cho i chạy từ 0, j bắt đầu chạy từ thứ i+1.

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

**Bước 2:** Đặt điều kiện cho điểm trung bình theo trình tự giảm dần:

```
if((s+i)->dtb<(s+j)->dtb) ;
```

Khai báo 1 biến bất kì trong bài đặt tên là temp theo kiểu struct.

```
Sinhvien temp;
```

**Bước 3:** Dùng hàm strcpy dùng để hoán đổi, ví dụ trong bài code gán giá trị của con trỏ (s+j) vào biến temp với từng thông tin của sinh viên như mssv, hoten, sex,...Sau đó gán giá trị của con trỏ (s+i) vào giá trị (s+i) với nhau. Cuối cùng gán giá trị của temp vào con trỏ (s+i) theo từng thông tin của sinh viên để thực hiện việc sắp xếp.

```
if((s+i)->dtb<(s+j)->dtb){//khai báo biến trung gian là temp có kiểu struct
```

```
Sinhvien temp;
```

```
strcpy(temp.mssv, (s+j)->mssv) ;
```

```
strcpy(temp.hoten, (s+j)->hoten) ;
```

```

        strcpy(temp.gioitinh, (s+j)->gioitinh);

        temp.ngay=(s+j)->ngay;

        temp.thang=(s+j)->thang;

        temp.nam=(s+j)->nam;

        temp.dtb=(s+j)->dtb ;

    strcpy((s+j)->mssv, (s+i)->mssv);

    strcpy((s+j)->hoten, (s+i)->hoten);

    strcpy((s+j)->gioitinh, (s+i)->gioitinh);

    (s+j)->ngay=(s+i)->ngay;

    (s+j)->thang=(s+i)->thang;

    (s+j)->nam=(s+i)->nam;

    (s+j)->dtb=(s+i)->dtb ;

    strcpy((s+i)->mssv, temp.mssv);

    strcpy((s+i)->hoten, temp.hoten);

    strcpy((s+i)->gioitinh, temp.gioitinh);

    (s+i)->ngay=temp.ngay;

    (s+i)->thang=temp.thang;

    (s+i)->nam=temp.nam;

    (s+i)->dtb=temp.dtb ;

    }

}

}

```

**Bước 4:** In kết quả ra màn hình.

```

C:\Users\admin\Documents\PROGRAM-C & C++\Baitapnho.exe
Ngay, thang, nam: 4 5 2001
Nhap DTB: 9
MSSV      Ho va ten      Gioi Tinh Ngay  Thang Nam  DTB      Xep loai
123      dat              nam       15   1   2001  10.0      G
-----
345      nhan              nam       23   4   2001  4.0       TB
-----
678      tai                nam       4    5   2001  9.0       G
-----

Ban co muon tiep tục chương trình không? (an 1 de tiep tục): 1
===== MENU =====
1. Them n Sinh vien vao danh sach va Hien thi danh sach sinh vien.
2. Sap xep theo ten.
3. Sap xep theo diem tb giam dan.
4. Tim sinh vien theo ten.
5. Ghi thông tin sinh vien ra file.
0. Thoat chương trình.
Nhap su lua chon cua ban: 3
MSSV      Ho va ten      Gioi Tinh Ngay  Thang Nam  DTB      Xep loai
123      dat              nam       15   1   2001  10.0      G
-----
678      tai                nam       4    5   2001  9.0       G
-----
345      nhan              nam       23   4   2001  4.0       TB
-----

```

Hình 3.4. Sắp xếp sinh viên theo điểm trung bình giảm dần

### 3.7. Sắp xếp tên theo thứ tự A-Z

**Bước 1 :** Sử dụng 2 dòng lặp for int i và int j. Cho i chạy từ 0, j bắt đầu chạy từ thứ i+1:

```

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

```

**Bước 2:** Đặt điều kiện cho tên: `if(strcmp((s+i)->hoten, (s+j)->hoten)==1)`

**Bước 3:** Tương tự như sắp xếp điểm và in ra màn hình.

```

Nhap su lua chon cua ban: 2
MSSV      Ho va ten      Gioi Tinh Ngay  Thang Nam  DTB      Xep loa
123      dat              nam       15   1   2001  10.0      G
-----
345      nhan              nam       23   4   2001  4.0       TB
-----
678      tai                nam       4    5   2001  9.0       G
-----

Ban co muon tiep tục chương trình không? (an 1 de tiep tục): _

```

Hình 3.5. Sắp xếp sinh viên theo tên

### 3.8. Ghi file dạng văn bản

**Bước 1:** Khai báo con trỏ kiểu file và dùng hàm fopen() để mở 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. 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.

```
FILE *fOut=fopen("sv.txt","a");
```

**Bước 2:** Gọi lại hàm nhập để nhập thông tin sinh viên vào file sv.txt. Số sinh viên muốn nhập là n sẽ được nhập vào file đã tạo (hay file đã có sẵn).

```
FILE *fOut=fopen("sv.txt","a");

nhap(s,&n);

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

    fprintf(fOut,"%-10s %-20s %-10s %-5d %-5d %-5d %-8.1f\n",
(s+i)->mssv,(s+i)->hoten,(s+i)->gioitinh,(s+i)->ngay,
(s+i)->thang,(s+i)->nam,(s+i)->dtb);

}

fclose(fOut);
```

**Bước 3:** Đóng file vừa nhập bằng lệnh fclose.

```
fclose(fOut);
```

### 3.9. Đọc file dạng văn bản

**Bước 1:** Mở file txt đã tạo từ trước, dùng hàm fopen() để mở file và trong ngoặc là đường dẫn tới file trong máy của bạn.

```
FILE *fOut ;

fOut= fopen("C:\\Users\\admin\\Desktop\\sv.txt", "r");
```

Bước 2: Sử dụng hàm `fscanf` để in ra dữ liệu văn bản và sử dụng hàm `feof()` để kết thúc việc in ra nội dung văn bản khi kết thúc nội dung, hàm `fscanf` sẽ trả về giá trị 0 để kết thúc.

Khi con trỏ không nhận được nội dung văn bản theo đường dẫn thì sẽ trả về giá trị Null là in ra màn hình “khong the mo file”.

```
FILE *fOut ;

fOut= fopen("C:\\Users\\admin\\Desktop\\sv", "r");

int i = 0;

if(fOut) {

    printf("%-10s %-20s %-5s %-5s %-5s %-5s %-8s\n",

    "MSSV", "Ho va ten", "Gioi Tinh", "Ngay ", "Thang", "Nam", "DTB");

    for(;;) {

        fscanf(fOut, "%-10s %-20[^\n] %-10s %-5d %-5d %-5d %-8.1f\n", (s+i)->mssv, (s+i)->hoten, (s+i)->gioitinh, &(s+i)->ngay,

        &(s+i)->thang, &(s+i)->nam, &(s+i)->dtb);

        if(feof(fOut)) {

            break:} }

}
```

**Bước 3:** Đóng file bằng lệnh `fclose`.

### 3.10. Hướng dẫn sử dụng chương trình

- Nhập dữ liệu nhấn phím 1 thông tin sẽ được ghi vào file.txt.
- Sắp xếp theo tên nhấn phím 2 .
- Sắp xếp theo ĐTB nhấn phím 3.
- Tìm sinh viên theo tên nhấn phím 4 sau đó nhập tên sinh viên muốn tìm.
- Nhấn phím 5 để in ra sách sinh viên sau khi xếp loại.
- Nhấn phím 0 để thoát ngay chương trình nếu như muốn tiếp tục thì ấn phím 1.

## KẾT LUẬN

- Kết quả đạt được
  - Hoàn thành được yêu cầu mà nhóm đề ra ban đầu là tạo được một chương trình với chức năng là quản lý sinh viên thông qua các yếu tố thiết yếu là MSSV, họ tên, điểm...
  - Hiểu rõ và áp dụng thành công các kiến thức về ngôn ngữ lập trình C vào trong chương trình như: cấu trúc (struct), con trỏ, vòng lặp for, câu lệnh if - else, đặc biệt là các thuật toán tìm kiếm và sắp xếp.
  - Tăng khả năng phân tích thuật toán, khả năng giao tiếp và kỹ năng làm việc nhóm.
- Nhược điểm
  - Các thành viên còn khá yếu về phần làm việc với file.
  - Mất nhiều thời gian trong việc lựa chọn chủ đề bài tập lớn.
  - Chương trình còn sơ khai và chưa được bắt mắt (không có đồ họa).

➤ Nguồn tài liệu tham khảo

- Giáo trình môn học Lập trình nâng cao - Phạm Văn Ất, Nguyễn Hiếu Cường
- Series video “Học lập trình C từ cơ bản đến nâng cao”- thân triệu trên Youtube
- 
- [programiz.com](https://programiz.com)
- [learn-c.org](https://learn-c.org)
- [freecodecamp.org/news/the-c-beginners-handbook](https://freecodecamp.org/news/the-c-beginners-handbook)
- [www.studytonight.com](https://www.studytonight.com)