

## Lời cảm ơn!

Toàn bộ thành viên nhóm xin gửi lời cảm ơn chân thành đến cô Trần Thị Dung. Cảm ơn cô đã tạo điều kiện và đã nhiệt tình hướng dẫn và giải đáp các thắc mắc để nhóm có thể hoàn thành báo cáo bài tập lớn môn “Lập trình nâng cao”.

Trong quá trình thực hành và làm báo cáo, kiến thức của các thành viên trong nhóm còn hạn chế, do vậy khó tránh khỏi sự thiếu sót, cả nhóm rất mong nhận được những nhận xét quý báu của Cô và toàn thể các bạn học cùng lớp để kiến thức của nhóm trong lĩnh vực này được hoàn thiện hơn.

Sau cùng, toàn bộ thành viên trong nhóm xin kính chúc cô Trần Thị Dung thật dồi dào sức khỏe để tiếp tục thực hiện sứ mệnh cao đẹp là truyền đạt kiến thức cho thế hệ chúng em và mai sau.

Trân trọng!

Thành phố Hồ Chí Minh

Nhóm thực hiện

Nguyễn Hoàng Hiệp

Nguyễn Hữu Đại

Trần Viết Học

## Mục lục

Lời cảm ơn!.....	1
CHƯƠNG 1 CƠ SỞ LÝ THUYẾT.....	5
1.1 Các thuật toán tìm kiếm.....	5
1.1.1 Tìm kiếm tuyến tính:.....	5
1.1.2 Tìm kiếm nhị phân:.....	6
1.2 Các thuật toán sắp xếp.....	7
1.2.1 Sắp xếp nổi bọt trong C.....	7
1.2.2 Sắp xếp chọn trong C.....	7
1.3 Danh sách liên kết đơn.....	9
1.3.1 Bản chất của danh sách liên kết đơn.....	9
1.3.2 Danh sách liên kết là gì?.....	10
Danh sách các kiểu danh sách liên kết:.....	11
1.3.3 Cài đặt danh sách liên kết đơn.....	11
1.3.4 Thêm Node vào danh sách liên kết.....	12
1.4 Làm việc với tệp (thao tác trên tệp tin).....	18
1.4.1 Các hàm dùng chung cho cả hai loại:.....	19
1.4.2 Các hàm chỉ dùng cho tệp nhị phân.....	21
1.4.3 Các hàm chỉ dùng cho tệp văn bản.....	22
1.4.4 Truy cập tệp tin ngẫu nhiên.....	23
CHƯƠNG 2 CHƯƠNG TRÌNH ỨNG DỤNG.....	25
2.1 Lý do chọn đề tài.....	25
2.2 Hướng dẫn sử dụng và chức năng của chương trình.....	26
2.3 Ý tưởng xây dựng chương trình.....	31
2.3.1 Dữ liệu.....	31
2.3.2 Tạo ra Menu dưới đây:.....	31
2.3.3 Nhập dữ liệu của từng quyển sách.....	31
2.3.4 Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng quyển sách theo thể loại ( $Z \rightarrow A$ ).....	32
2.3.5 Tìm quyển sách theo thể loại.....	32
2.3.6 Ghi vào tệp tin nhị phân book.dat.....	32
2.3.7 Exit.....	33
2.4 Thiết kế chương trình.....	34
CHƯƠNG 3 TỔNG KẾT.....	50
3.1 Kết quả đạt được.....	50

3.2 Nhược điểm.....	50
3.3 Hướng phát triển.....	50
TÀI LIỆU THAM KHẢO.....	50

## Danh mục hình ảnh

Hình 1 .....	5
Hình 2 .....	6
Hình 3 .....	7
Hình 4 .....	8
Hình 5 .....	11
Hình 6 .....	18
Hình 7 .....	21
Hình 8 .....	26
Hình 9 .....	27
Hình 10 .....	28
Hình 11 .....	28
Hình 12 .....	29
Hình 13 .....	29
Hình 14 .....	30
Hình 15 .....	30
Hình 16 .....	45

## CHƯƠNG 1 CƠ SỞ LÝ THUYẾT

### 1.1 Các thuật toán tìm kiếm

#### 1.1.1 Tìm kiếm tuyến tính:

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.



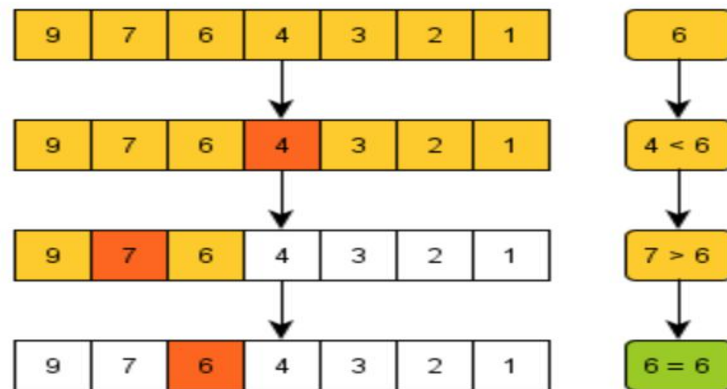
Hình 1

Ví dụ:

```
void timkiem(int a[],int n){  
    int x, t=0;  
    printf("\nNhap so can tim vao: ");    scanf( "%d",&x);  
    for( int i=0; i<n; i++){  
        if ( x== a[i]){  
            printf ("%d co trong day so\n",x) ;  
            t=1; break;}  
    }    if( t==0)  
        printf("%d khong co trong day so\n", x);  
}
```

### 1.1.2 Tìm kiếm nhị phân:

Thuật toán tìm kiếm nhị phân hoạt động trên các mảng đã được sắp xếp. Thuật toán bắt đầu bằng việc so sánh một phần tử đứng chính giữa mảng với giá trị cần tìm. Nếu bằng nhau, vị trí của nó trong mảng sẽ được trả về. Nếu giá trị cần tìm nhỏ hơn phần tử này, quá trình tìm kiếm tiếp tục ở nửa nhỏ hơn của mảng. Nếu giá trị cần tìm lớn hơn phần tử ở giữa, quá trình tìm kiếm tiếp tục ở nửa lớn hơn của mảng. Bằng cách này, ở mỗi phép lặp thuật toán có thể loại bỏ nửa mảng mà giá trị cần tìm chắc chắn không xuất hiện.



Hình 2

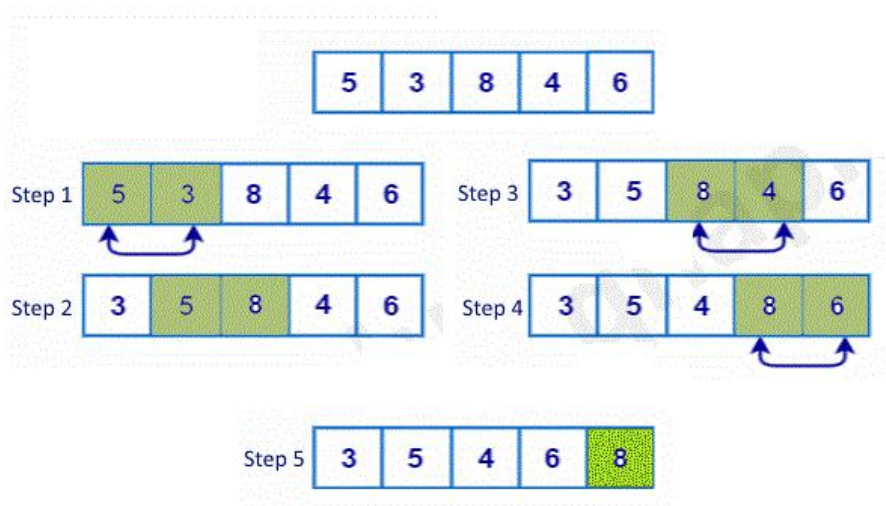
Ví dụ:

```
int binarySearch(int arr[], int l, int r, int x) {  
    if (r >= l) {  
        int mid = l + (r - l) / 2;  
        if (arr[mid] == x)  
            return mid;  
        if (arr[mid] > x)  
            return binarySearch(arr, mid + 1, r, x);  
        return binarySearch(arr, l, mid - 1, x);  
    }  
    return -1;  
}
```

## 1.2 Các thuật toán sắp xếp

### 1.2.1 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 trao đổi thứ tự nếu chúng không theo thứ tự.



Hình 3

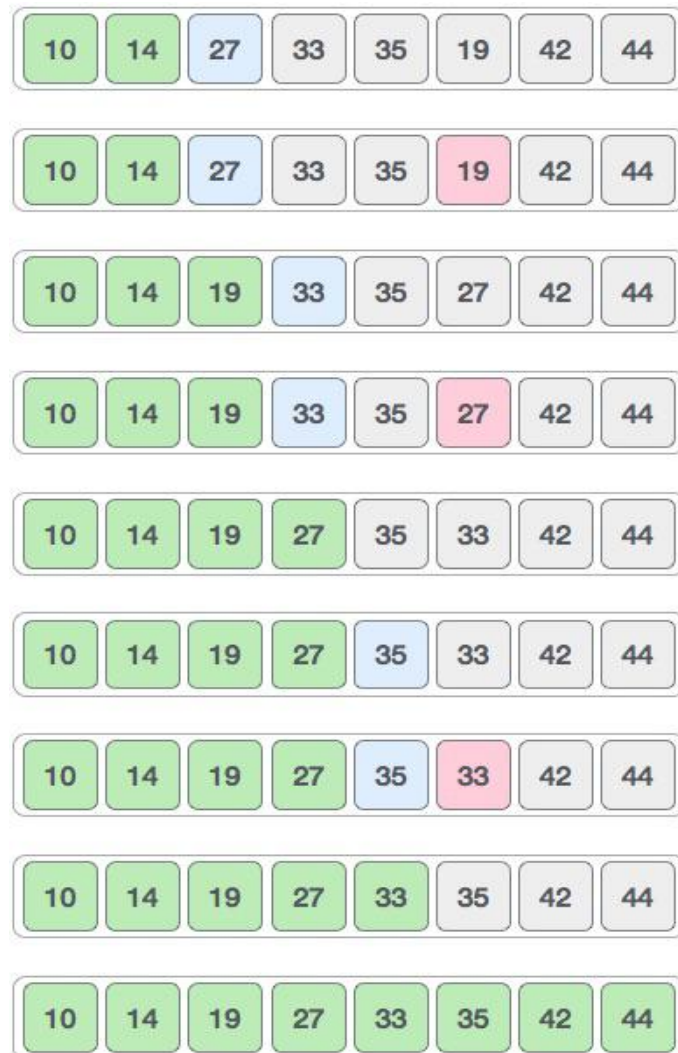
Ví dụ:

```
void sapxepnoibot(int a[],int n){  
    for(int i=0; i<n ; i++){  
        for( int j= 0; j<n; j++) {  
            if( a[i]<a[j]) {  
                int x = a[i];  
                a[i]= a[j];  
                a[j]= x;    }    }  
    }  
}
```

### 1.2.2 Sắp xếp chọn trong C

Chọn phần tử nhỏ nhất trong n phần tử ban đầu, đưa phần tử này về vị trí đúng là đầu tiên của dãy hiện hành. Sau đó không quan tâm đến nó nữa, xem dãy hiện hành chỉ còn n-1 phần tử của dãy ban đầu, bắt đầu từ vị trí thứ 2. Lặp lại quá trình trên

cho dãy hiện hành đến khi dãy hiện hành chỉ còn một phần tử. Dãy ban đầu có  $n$  phần tử, vậy tóm tắt ý tưởng thuật toán là thực hiện  $n-1$  lượt việc đưa phần tử nhỏ nhất trong dãy hiện hành về vị trí đúng ở đầu dãy.



Hình 4

Ví dụ:

```
void SelectionSort_Improve(int A[], int n)
{
    for (int i = 0; i < n - 1; i++){
```



```

    int max = i;
    for (int j = i + 1; j < n; j++){
        if (A[max] > A[j]){
            max = j;
        }
    }
    if (max != i){ // Sử dụng phép OR loại trừ bit
        A[max] ^= A[i];
        A[i] ^= A[max];
        A[max] ^= A[i];
    }
}
}

```

### 1.3 Danh sách liên kết đơn

#### 1.3.1 Bản chất của danh sách liên kết đơn

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:

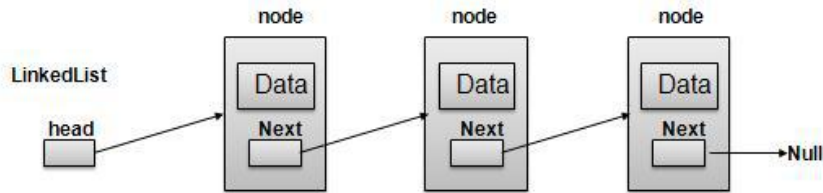
Nội dung	Mảng	Danh sách liên kết
Kích thước	Kích thước cố định Cần chỉ rõ kích thước trong khi khai báo	Kích thước thay đổi trong quá trình thêm/ xóa phần tử Kích thước tối đa phụ thuộc vào bộ nhớ
Cấp phát bộ nhớ	Tĩnh: Bộ nhớ được cấp phát trong quá trình biên dịch	Động: Bộ nhớ được cấp phát trong quá 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

Bảng 1. So sánh giữa mảng và danh sách liên kết.

### 1.3.2 Danh sách liên kết 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, có nghĩa đó là phần tử cuối cùng của linked list.

Hình ảnh mô phỏng một danh sách liên kết đơn đầy đủ:



Hình 5

Danh sách các kiểu danh sách liên kết:

- Danh sách liên kết đơn(Single linked list): Chỉ có sự kết nối từ phần tử phía trước tới phần tử phía sau.
- Danh sách liên kết đôi(Double linked list): Có sự kết nối 2 chiều giữa phần tử phía trước với phần tử phía sau
- Danh sách liên kết vòng(Circular Linked List): Có thêm sự kết nối giữa 2 phần tử đầu tiên và phần tử cuối cùng để tạo thành vòng khép kín.

### 1.3.3 Cài đặt danh sách liên kết đơn

Khai báo linked list

Ví dụ:

```
typedef struct LinkedList{

    int data;

    struct LinkedList *next;

}node;
```

Tạo mới một node

```
node CreateNode(int value){
```

```

node temp; // khai bao node

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

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

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

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

}

```

#### 1.3.4 Thêm Node vào danh sách liên kết

\*Thêm đầu:

- Thêm vào đầu danh sách có nghĩa là: chúng ta sẽ đưa một số bất kì vào trong danh sách.

Sẽ có 2 trường hợp xảy ra:

- + Nếu danh sách = NULL ( danh sách đang trống) thì số ta chèn vào đầu danh sách nó sẽ nằm ở vị trí đầu và cũng nằm ở vị trí cuối cùng.
- + Ngược lại thì head cũ sẽ được thay bởi head mới.
- Thêm vào đầu chính là việc cập nhật lại head. Ta gọi Node mới (temp), ta có:
  - Nếu head đang trở tới NULL, thì linked list đang trống, Node mới thêm vào sẽ làm head.
  - Ngược lại, ta phải thay thế head cũ bằng head mới. Việc này phải làm theo thứ tự như sau:
    - Cho next của temp trở tới head hiện hành.
    - Đặt temp làm head mới.

Ví dụ:

```

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
}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)
}
return head;
}

```

\*Thêm cuối

Tạo một node mới với giá trị value nếu Head = NULL thì tức là danh sách trống và head là đầu và cũng là cuối.

Ví dụ:

```

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.
    }
    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 nút cuối = temp. Khi đó temp sẽ là cuối(temp->next = NULL mà)
```

```
}
```

```
return head;
```

```
}
```

### 1.3.5 Xóa node khỏi danh sách liên kết

-Xóa đầu:

- Cập nhật lại biến con trỏ Head, sau đó giải phóng của Node cần xóa.

Ví dụ:

```
node DelHead(node head){
```

```
if( head == NULL){
```

```
    printf("\nKhong co gi de xoa");
```

```
} else {
```

```
    node* p = head;// Khai báo 1 con trỏ để lưu lại địa chỉ node đầu tiên
```

```
    head = head -> next;// Cập nhật lại giá trị head
```

```
    delete p; // Giải phóng vùng node cần xóa
```

```
}
```

\*Xóa cuối: duyệt đến node cuối – 1, cho next của cuối – 1 đó bằng NULL.

Ví dụ:

```
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 b?ng NULL
    // Ho?c vi?t p->next = NULL c?ng đư?c
    return head;
}

```

### 1.3.6 Lấy giá trị ở vị trí bất kì

- Hàm duyệt đến vị cần tìm và trả về giá trị của node đó,
- Hàm duyệt từ head đến vị trí cuối cùng (không có giá trị - NULL).

Ví dụ:

```

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;
}

```

### 1.3.7 Tìm kiếm trong danh sách liên kết

- Hàm này trả về chỉ số của node có giá bằng với giá trị cần tìm. Nếu không tìm thấy thì trả về -1.

Ví dụ:

```

int search(node head, int v){
    int pst = 0;
    for(node p = head; p != NULL; p = p-> next){

```

```

        if(p->data == v)
            return pst;
        ++pst;
    }
    return -1;
}

```

-Có thể sử dụng này để xóa tất cả các node trong danh sách có giá trị chỉ định.

Ví dụ:

```

node DelByVal(node head, int v){
    int pst = Search(head, v);
    while(pst != -1){
        DelAt(head, pst);
        pst = Search(head, v);
    }
    return head;
}

```

### 1.3.8 Duyệt danh sách liên kết

Khởi tạo từ node head, duyệt từng node theo con trỏ next đến khi node NULL.

Ví dụ:

```

void In(node head){
    printf("\n");
    for(node p = head; p != NULL; p = p->next){
        printf("%5d", p->data);
    }
}

```



```
}  
}
```

### 1.3.9 Một số hàm hỗ trợ khác

-Khởi tạo hàm node head

Ví dụ:

```
node InitHead(){  
    node head;  
    head = NULL;  
    return head;  
}
```

-Đếm số phần tử danh sách liên kết đơn:

Duyệt và trả về giá trị đếm được.

Ví dụ:

```
int Length(node head){  
    int length = 0;  
    for(node p = head; p != NULL; p = p->next){  
        ++length;  
    }  
    return length;  
}
```

-Nhập giá trị cho danh sách

Ví dụ:

```
node Input(){  
    node head = InitHead();  
    int n, value;  
    do{
```

```

printf("\nNhap so luong phan tu n = ");
scanf("%d", &n);
}while(n <= 0);

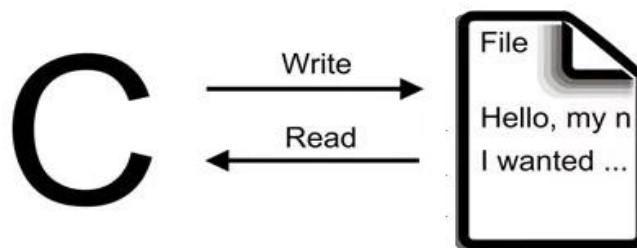
for(int i = 0; i < n; ++i){
    printf("\nNhap gia tri: ");
    scanf("%d", &value);
    head = AddTail(head, value);
}
return head;
}

```

#### 1.4 Làm việc với tệp (thao tác trên tệp tin)

Một tệp tin biểu diễn một chuỗi các bytes, không kể đó là tệp tin văn bản hay tệp tin nhị phân. Ngôn ngữ lập trình C cung cấp các hàm truy cập mức độ cao cũng như thấp (mức hệ điều hành) để thao tác với tệp tin trên thiết bị lưu trữ.

### Đọc ghi file trong C



Hình 6

Có hai kiểu tệp tin:

- Tệp tin nhị phân: Trong quá trình nhập xuất không bị biến đổi. Dữ liệu ghi trên tệp theo các bit nhị phân như trong bộ nhớ.

- Tập tin văn bản: Chỉ khác tập nhị phân khi xử lý kí tự chuyển dòng (mã 10) và ký tự mã 26. Đối với các ký tự khác, hai kiểu đều đọc ghi như nhau. Đối với các ký tự khác cả hai đều đọc ghi như nhau.

#### 1.4.1 Các hàm dùng chung cho cả hai loại:

Hàm fopen:

+ Dạng hàm: FILE \*fopen(const char \*tên\_tệp, const char \*kiểu);

Dùng để mở tệp.

+ Các đối:

Đối thứ nhất là xâu kí tự, đường dẫn đến tệp, đối thứ hai là kiểu truy nhập (có thể có các kiểu truy nhập sau).

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.

Bảng 2. Bảng giá trị truy cập

Hàm fclose:

- + Dạng hàm: int fclose(FILE \*fp);
- + Công dụng hàm dùng để đóng tệp.
- + Đối: fp là con trỏ tương ứng với tệp cần đóng.

Hàm fcloseall

- +Dạng hàm: int fcloseall(void);
- + Công dụng dùng đóng tất cả tệp đang mở.

Hàm fflush:

- + Dạng hàm: int fflush(FILE \*fp);

+ Công dụng: hàm dùng làm sạch vùng đệm của tệp do con trỏ fp trở tới.

+ Đối: fp là con trỏ tệp.

Hàm fflush:

+ Dạng hàm: int fflush(void);

+ Công dụng hàm làm sạch vùng đệm của các tệp đang mở.

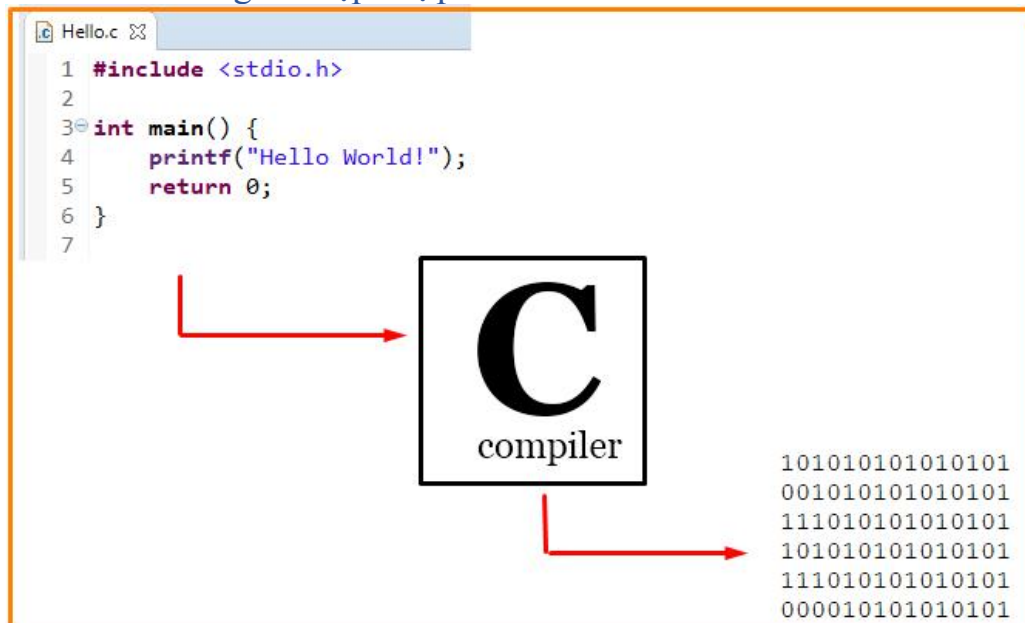
Hàm feof:

+ Dạng hàm: int feof(FILE \*fp);

+ Công dụng: Hàm dùng để kiểm tra con trỏ tệp đang mở có trở tới cuối tệp hay không. Hàm trả về giá trị khác 0 nếu gặp cuối tệp khi đọc, trái lại hàm trả về giá trị 0.

+ Đối: fp là con trỏ tệp.

#### 1.4.2 Các hàm chỉ dùng cho tệp nhị phân



Hình 7

Hàm fwrite:

+ Dạng hàm:

size\_t fwrite (void \*bufer, size\_t num\_bytes, size\_t count, FILE \*fp);

+ Công dụng: Dùng để ghi một số mẫu tin lên tệp.

+ Đối: bufer là con trỏ dùng để trỏ đến thông tin sẽ được ghi lên tệp, num\_bytes là số byte được đọc, count xác định số thành phần ( mỗi thành phần là num\_bytes byte) được đọc và fp là con trỏ tệp.

Hàm fread:

+ Dạng hàm: size\_t fread(void \* bufer, size\_t num\_bytes, size\_t count, FILE\* fp);

+ Công dụng: Dùng để đọc một số mẫu tin từ tệp.

+ Đối: bufer là con trỏ dùng để trỏ đến vùng bộ nhớ mà nhận dữ liệu đọc từ tệp, num\_bytes là số byte được đọc, count xác định số thành phần (mỗi thành phần là num\_bytes byte) được đọc và fp là con trỏ tệp.

### 1.4.3 Các hàm chỉ dùng cho tệp văn bản

Hàm fprintf:

+ Dạng hàm: int fprintf(FILE \*fp, const char \*dk, bt);

+ Công dụng: Giá trị các biểu thức bt được ghi trên tệp fp theo khuôn dạng xác định trong chuỗi dk. Hàm làm việc giống printf.

+ Đối: fp là con trỏ tệp, dk chứa địa chỉ của chuỗi điều khiển, bt là danh sách các biểu thức mà giá trị của chúng cần ghi lên tệp.

Hàm fscanf:

+ Dạng hàm: int fscanf(FILE \*fp, const char \*dk,...);

+ Công dụng: Đọc dữ liệu từ tệp fp, biến dạng theo khuôn dạng trong dk và lưu kết quả và các đối. Hàm làm việc giống scanf, trả về giá trị bằng số trường đã đọc được.

+ Đối: fp là con trỏ tệp, dk chứa địa chỉ của chuỗi điều khiển, ... là danh sách địa chỉ các đối chứa kết quả đọc được từ tệp.

Hàm fputs:

+ Dạng hàm: `int fputs(const char *s, FILE *fp);`

+ Công dụng: Chuỗi s ghi lên tệp fp (dấu '\0' không ghi lên tệp). Khi thành công, hàm trả về ký tự cuối cùng được ghi lên tệp. Khi có lỗi hàm cho EOF.

+ Đối: s là con trỏ tới địa chỉ đầu của chuỗi ký tự kết thúc bằng dấu '\0'. fp là con trỏ tệp.

Hàm fgets:

+ Dạng hàm: `char *fgets(char *s, int n, FILE *fp);`

+ Công dụng: Đọc 1 dãy ký tự từ tệp fp chứa vào vùng nhớ s. Việc đọc kết thúc khi:

- Hoặc đã đọc  $n - 1$  ký tự.
- Hoặc gặp dấu xuống dòng (Cặp mã 13 10). Khi đó mã được đưa vào xâu kết quả.
- Hoặc kết thúc tệp.

Xâu kết quả sẽ được bổ sung thêm dấu kết thúc chuỗi '\0'. Khi thành công, hàm trả về địa chỉ vùng nhận kết quả. Khi có lỗi hoặc gặp cuối tệp, hàm cho giá trị NULL.

+ Đối: s là con trỏ (kiểu char) trỏ tới một vùng nhớ đủ lớn để chứa chuỗi ký tự đọc từ tệp. n là số nguyên xác định độ dài cực đại của dãy cần đọc. fp là con trỏ tệp.

#### 1.4.4 Truy cập tệp tin ngẫu nhiên

Hàm rewind:

+ Dạng hàm: `void rewind(FILE *fp);`

+ Công dụng: Chuyển con trỏ chỉ vị của tệp fp về đầu tệp. khi đó việc nhập xuất trên tệp fp được thực hiện từ đầu tệp.

+ Đối: fp là con trỏ tệp.

Hàm fseek:

+ Dạng hàm: `int fseek(FILE *fp, long sb, int xp);`

+ Công dụng: Hàm di chuyển con trỏ chỉ vị của tệp fp từ vị trí xác định bởi xp qua một số byte bằng giá trị tuyệt đối của sb. Chiều di chuyển là về cuối tệp nếu sb dương, trái lại sẽ di chuyển về phai đầu tệp.

+ Đối:

fp là con trỏ tệp.

sb là số byte cần di chuyển.

xp cho biết vị trí xuất phát mà việc dịch chuyển được bắt đầu từ đây.

xp có thể nhận các giá trị sau:

xp = SEEK\_SET hay 0: Xuất phát từ đầu tệp.

xp = SEEK\_CUR hay 1: Xuất phát từ vị trí hiện tại của con trỏ chỉ vị.

xp = SEEK-END hay 2: Xuất phát từ cuối tệp.

CHÚ Ý: Không nên dùng fseek trên kiểu văn bản, vì sự chuyển đổi ký tự sẽ làm cho việc định vị thiếu chính xác.

Hàm ftell:

+ Dạng hàm: long ftell(FILE \*fp);

+ Công dụng: Khi thành công hàm cho biết vị trí hiện tại của con trỏ chỉ vị (byte thứ mấy trên tệp fp). Số thứ tự của byte được tính từ 0. Khi có lỗi hàm trả về -1L.

+ Đối: fp là con trỏ tệp.



## CHƯƠNG 2 CHƯƠNG TRÌNH ỨNG DỤNG

### 2.1 Lý do chọn đề tài

Trong xã hội ngày càng phát triển hiện nay, khoa học công nghệ là thứ không thể thiếu đối với mỗi quốc gia, doanh nghiệp, trường học hay mỗi cá nhân, đặc biệt là công nghệ thông tin. Với sự phát triển nhanh một cách không ngừng nghỉ như vậy của công nghệ thông tin đã giúp giải quyết các công việc học tập, nguyên cứu, quản lý thông tin,... một cách dễ dàng và tiện lợi. Thấy được tiềm năng đó các quốc gia, doanh nghiệp, trường học, các cá nhân, ... đã ứng dụng nó vào thực tiễn cuộc sống để giải quyết công việc, học tập, giải trí. Đối với công việc quản lý sách, người quản lý thường xuyên gặp những vấn đề như:

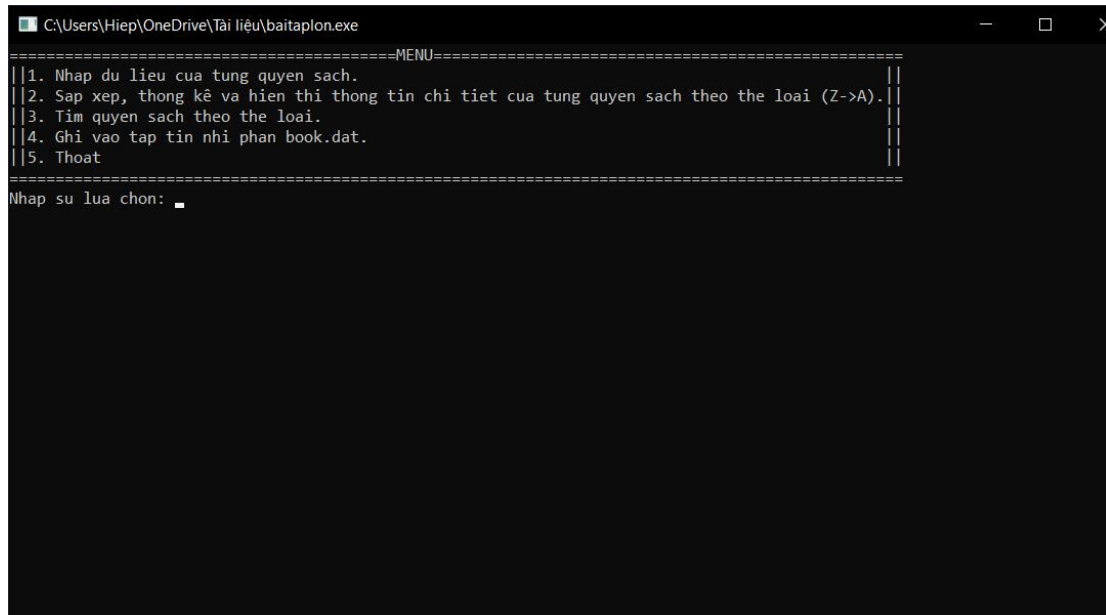
-Số lượng đầu sách quá nhiều khiến cho người quản lý không thể quản lý một cách triệt để các đầu mục.

-Thường xuyên xảy ra tình trạng mất mát sách khó kiểm soát -Người sử dụng, mượn sách, đọc sách không biết vị trí của từng loại sách, gây khó khăn trong việc sắp xếp và tìm kiếm sách. Nhận thấy được vấn đề đó nhóm em đã tìm hiểu và thực hiện đề tài: “Quản Lý Sách”. Nhằm giúp đỡ người sử dụng tiện lợi trong việc quản lý sách. Chương trình giúp mọi người sử dụng một cách dễ dàng.

## 2.2 Hướng dẫn sử dụng và chức năng của chương trình

**\*\*Chức năng chương trình: Nhập và quản lý sách.**

**\*\*Khi mở chạy, chương trình sẽ hiển thị:**



```
C:\Users\Hiep\OneDrive\Tài liệu\baitaplon.exe
=====MENU=====
||1. Nhập dữ liệu của từng quyển sách.                ||
||2. Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng quyển sách theo thể loại (Z->A).||
||3. Tìm quyển sách theo thể loại.                    ||
||4. Ghi vào tập tin nhị phân book.dat.               ||
||5. Thoát                                             ||
=====
Nhập sự lựa chọn: _
```

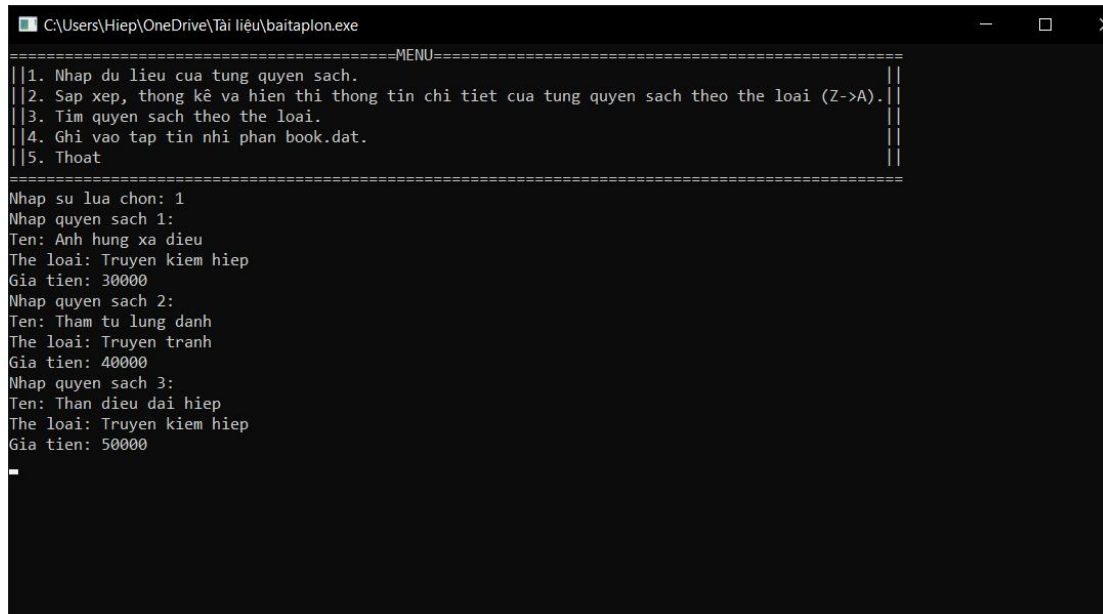
Hình 8

**\*\*Để sử dụng chương trình ta cần thực hiện theo những bước sau đây:**

**Bước 1. Nhập dữ liệu sách vào (bắt buộc thực hiện đầu tiên):**

-Để chương trình có dữ liệu thông tin của sách thì ta phải nhập thông tin sách vào. Để nhập dữ liệu thông tin sách ta ấn **phím 1**. Chương trình cho phép nhập vào ba quyển sách với nội dung gồm: tên sách, thể loại, giá tiền.

-Màn hình hiển thị của chương trình được nhập với ba quyển sách:



```
C:\Users\Hiiep\OneDrive\Tài liệu\baitaplon.exe
=====MENU=====
||1. Nhập dữ liệu của từng quyển sách.                ||
||2. Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng quyển sách theo thể loại (Z->A).||
||3. Tìm quyển sách theo thể loại.                    ||
||4. Ghi vào tập tin nhị phân book.dat.              ||
||5. Thoát                                             ||
=====
Nhập số lựa chọn: 1
Nhập quyển sách 1:
Tên: Anh hùng xạ điêu
Thể loại: Truyen kiem hiep
Giá tiền: 30000
Nhập quyển sách 2:
Tên: Tham tu lung danh
Thể loại: Truyen tranh
Giá tiền: 40000
Nhập quyển sách 3:
Tên: Than diêu dai hiep
Thể loại: Truyen kiem hiep
Giá tiền: 50000
-
```

Hình 9

Bước 2. Sử dụng chương trình:

-Hiển thị sắp xếp và thống kê:

+Nếu muốn sắp xếp, thống kê và hiển thị thông tin của từng quyển sách đã nhập ở **bước 1** theo thể loại từ (Z→A) ta ấn **phím 2**.

+Chương trình sẽ hiển thị như hình ảnh dưới đây:

```
=====MENU=====
||1. Nhập dữ liệu của từng quyển sách.
||2. Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng quyển sách theo thể loại (Z->A).
||3. Tìm quyển sách theo thể loại.
||4. Ghi vào tập tin nhí phán book.dat.
||5. Thoát
=====

Nhập sự lựa chọn: 2
STT|Tên
001|Tham tu lung danh
002|Anh hung xa dieu
003|Than dieu dai hiep
Truyen tranh co 1 quyên sách
Truyen kiem hiep co 2 quyên sách
||The loại
||Truyen tranh
||40000
||Truyen kiem hiep
||30000
||Truyen kiem hiep
||50000
||Giá tiền
```

Hình 10

-Tìm kiếm sách theo thể loại: Nếu muốn tìm quyển sách theo thể loại ta ấn **phím 3**. Sau khi ấn phím 3 thì chương trình sẽ yêu cầu nhập thể loại cần tìm vào:

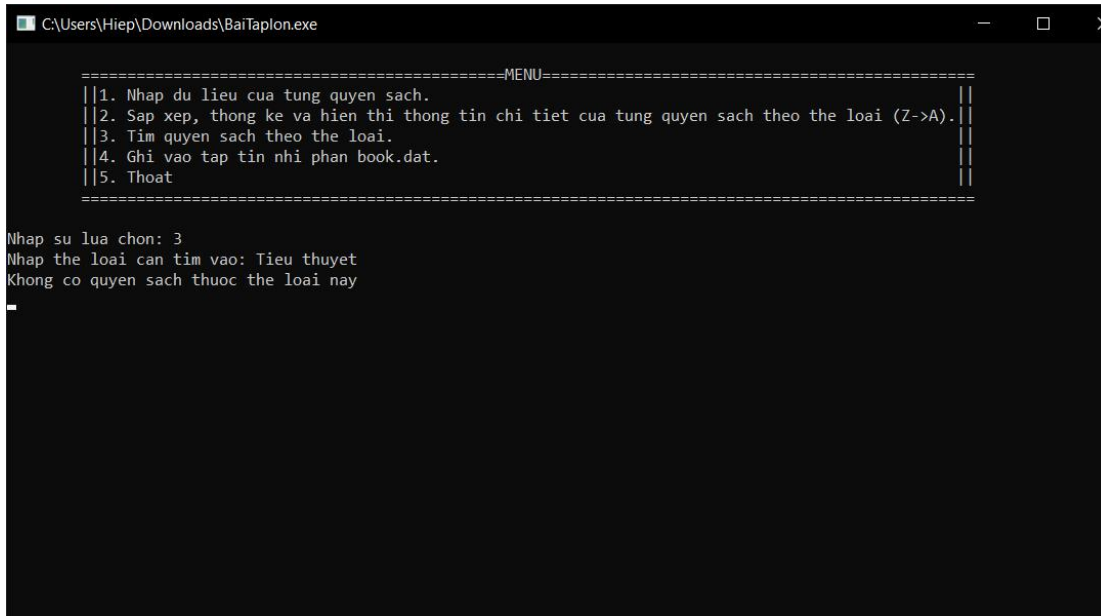
+Nếu thể loại cần tìm có trong dữ liệu nhập ở **bước 1** thì chương trình sẽ hiển thị:

```
=====MENU=====
||1. Nhập dữ liệu của từng quyển sách.
||2. Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng quyển sách theo thể loại (Z->A).
||3. Tìm quyển sách theo thể loại.
||4. Ghi vào tập tin nhí phán book.dat.
||5. Thoát
=====

Nhập sự lựa chọn: 3
Nhập thể loại cần tìm vào: Truyen kiem hiep
STT|Tên
001|Tham tu lung danh
002|Anh hung xa dieu
003|Than dieu dai hiep
Truyen tranh co 1 quyên sách
Truyen kiem hiep co 2 quyên sách
||The loại
||Truyen tranh
||40000
||Truyen kiem hiep
||30000
||Truyen kiem hiep
||50000
||Giá tiền
```

Hình 11

+Nếu thể loại cần tìm không có trong dữ liệu nhập ở **bước 1** thì chương trình sẽ hiển thị:

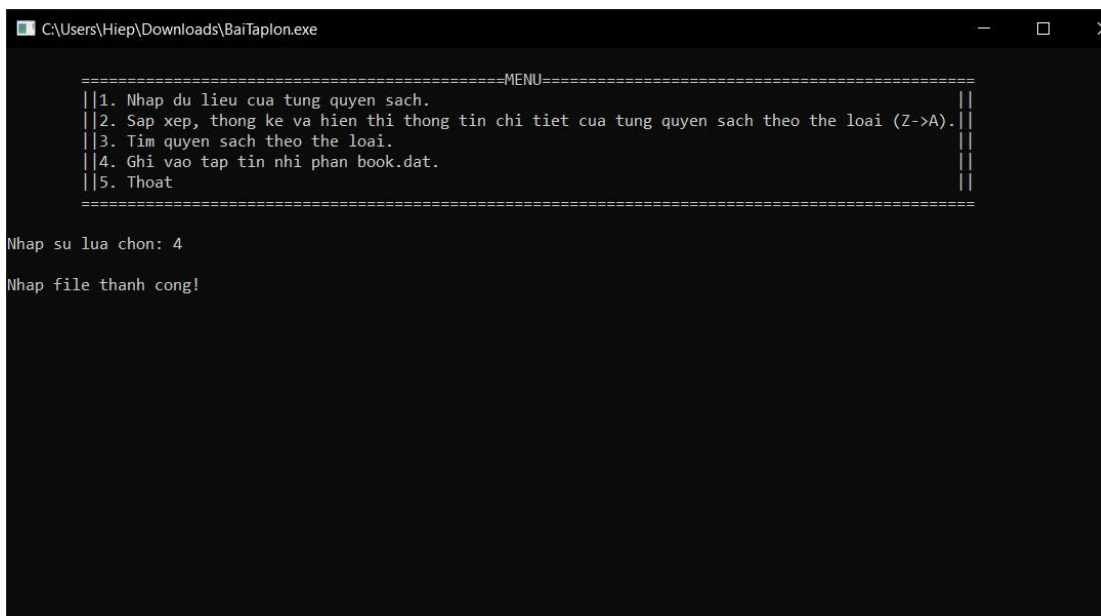


```
=====MENU=====
||1. Nhập dữ liệu của từng quyển sách.                ||
||2. Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng quyển sách theo thể loại (Z->A).||
||3. Tìm quyển sách theo thể loại.                    ||
||4. Ghi vào tập tin nhị phân book.dat.                ||
||5. Thoát                                             ||
=====
Nhập số lựa chọn: 3
Nhập thể loại cần tìm vào: Tieu thuyet
Không có quyển sách thuộc thể loại này
```

Hình 12

-Ghi tập tin nhị phân: Nếu muốn ghi vào tập tin nhị phân book.dat thì ta ấn **phím 4**. Sau khi ấn phím 4 chương trình sẽ hiển thị:

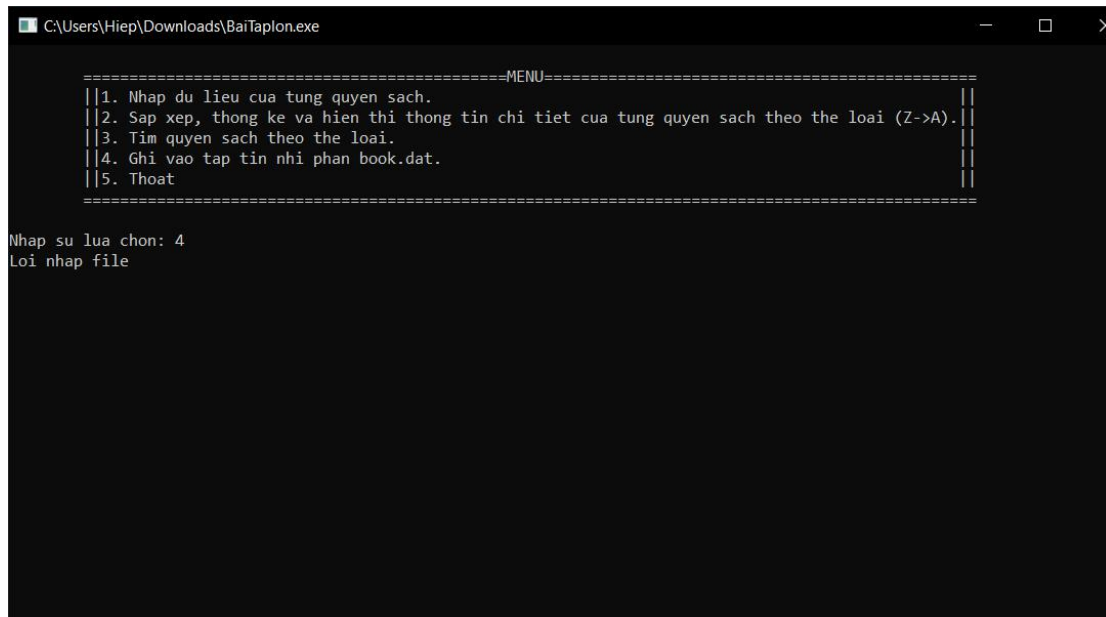
+Nếu file được ghi thành công:



```
=====MENU=====
||1. Nhập dữ liệu của từng quyển sách.                ||
||2. Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng quyển sách theo thể loại (Z->A).||
||3. Tìm quyển sách theo thể loại.                    ||
||4. Ghi vào tập tin nhị phân book.dat.                ||
||5. Thoát                                             ||
=====
Nhập số lựa chọn: 4
Nhập file thành công!
```

Hình 13

+Nếu file không được ghi:



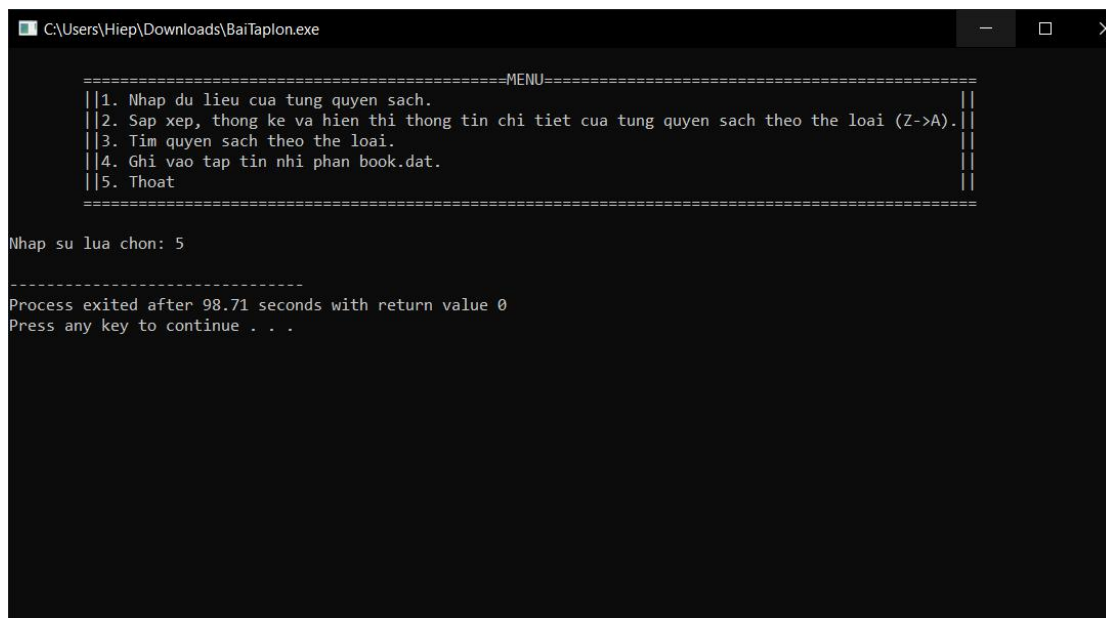
```
C:\Users\Hiep\Downloads\BaiTaplon.exe

=====MENU=====
||1. Nhập dữ liệu của từng quyển sách.                ||
||2. Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng quyển sách theo thể loại (Z->A).||
||3. Tìm quyển sách theo thể loại.                    ||
||4. Ghi vào tập tin nhị phân book.dat.                ||
||5. Thoát                                             ||
=====

Nhập số lựa chọn: 4
Lời nhắc file
```

Hình 14

+Thoát chương trình: Nếu muốn thoát khỏi chương trình thì ta ấn **phím 5**.



```
C:\Users\Hiep\Downloads\BaiTaplon.exe

=====MENU=====
||1. Nhập dữ liệu của từng quyển sách.                ||
||2. Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng quyển sách theo thể loại (Z->A).||
||3. Tìm quyển sách theo thể loại.                    ||
||4. Ghi vào tập tin nhị phân book.dat.                ||
||5. Thoát                                             ||
=====

Nhập số lựa chọn: 5

-----
Process exited after 98.71 seconds with return value 0
Press any key to continue . . .
```

Hình 15

## 2.3 Ý tưởng xây dựng chương trình

### 2.3.1 Dữ liệu

Định nghĩa 1 cấu trúc:

- struct book\_st: char ten[30], char theloai [30], int giatien
- Trong hàm main() khai báo một mảng bao gồm 3 quyền sách với cấu trúc ở trên có tên là "bookList".

### 2.3.2 Tạo ra Menu dưới đây:

1. Nhập du lieu cua tung quyen sach.
2. Sap xep, thong kê va hien thi thong tin chi tiet cua tung quyen sach theo the loai ( $Z \rightarrow A$ ).
3. Tim quyen sach theo the loai
4. Ghi vao tap tin nhi phan book.dat.
5. Thoat.

### 2.3.3 Nhập dữ liệu của từng quyền sách

Khi chọn 1 từ Menu, thực hiện nhập dữ liệu của từng quyền sách theo yêu cầu dưới đây:

Ví dụ: (chữ in đậm là dữ liệu được nhập vào từ người chạy chương trình)

Nhap quyen sach 1:

Ten: **Tham tu lung danh**

The loai: **Truyen tranh**

Gia tien: **50000**

Sử dụng hàm để thực hiện chức năng này:

- Sử dụng tham biến là con trỏ cấu trúc book\_st.
- Lưu các giá trị nhập từ người sử dụng vào mảng bookList trong hàm main().

#### 2.3.4 Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng quyển sách theo thể loại ( $Z \rightarrow A$ ).

Khi chọn 2 từ Menu, sắp xếp các quyển sách theo thể loại từ  $Z \rightarrow A$  sau đó hiển thị thông tin chi tiết của từng

quyển sách như dưới đây:

STT	Tên	The loại	Gia
001	Tham tu lung danh	Truyen tranh	50000
002	Than Dieu Dai Hiep	Truyenkiem hiep	30000
003	Anh Hung Xa Dieu	Truyenkiem hiep	35000

Hiển thị thống kê số quyển sách theo thể loại như dưới đây:

**Truyen tranh co 1 quyen sach**

**Truyenkiem hiep co 2 quyen sach**

Sử dụng hàm để thực hiện chức năng này:

- Sử dụng tham biến là mảng cấu trúc book\_st.

#### 2.3.5 Tìm quyển sách theo thể loại

Khi chọn 3 từ Menu, thực hiện các yêu cầu dưới đây:

Hiển thị “**Nhap the loại:** ”

Tìm và hiển thị thông tin chi tiết của tất cả các quyển sách thuộc thể loại này theo định dạng như trên câu hỏi 4. Trong trường hợp không tìm thấy quyển sách nào, hiển thị thông báo “**Khong co quyen sach thuoc the loại nay**”.

Sử dụng hàm để thực hiện chức năng này:

- Sử dụng tham biến là mảng cấu trúc book\_st.

#### 2.3.6 Ghi vào tập tin nhị phân book.dat.

Khi chọn 4 từ Menu, ghi tất cả thông tin của từng quyển sách vào file nhị phân “book.dat”.



Sử dụng hàm để thực hiện chức năng này theo yêu cầu dưới đây:

- Sử dụng tham biến là con trỏ cấu trúc book\_st.
- Tạo file nhị phân “book.dat”.
- Lưu trữ tất cả thông tin của từng quyển sách vào trong file này.
- Đóng file.

### 2.3.7 Exit

Khi chọn 5 từ Menu, chương trình kết thúc.

## 2.4 Thiết kế chương trình

-Chương trình được thiết kế bằng phần mềm Dev-C++

-Thư viện trong chương trình: Để xây dựng chương trình ta cần khai báo các thư viện sau:

**#include<stdio.h>** (Thư viện định nghĩa kiểu biến, thực hiện input và output)

**#include<stdlib.h>** (Thư viện chứa hàm để cấp phát bộ nhớ cho con trỏ)

**#include<string.h>** (Thư viện chứa hàm thực hiện thao tác giữa hai chuỗi kí tự )

**#include<conio.h>** (Thư viện chứa hàm getch() để xóa màn hình)

-Các hàm chính: Chương trình thư viện sách được xây dựng bằng các hàm chính:

```
void Nhap(Sach *bookList);  
void Sapxep(Sach bookList[]);  
void Timkiem(Sach bookList[]);  
void Ghivaotaptin(Sach *bookList);  
void Menu(Sach bookList[]);  
int main();
```

-Để tạo thư viện lưu trữ dữ liệu sách ta dùng cấu trúc **struct**. Dữ liệu sách được tạo trong chương trình là **Sach** và được khai báo bằng đoạn code dưới đây:

```
typedef struct book_st{  
    char ten[30];  
    char theloai[30];  
    int giatien;  
}Sach;
```

-Nhập dữ liệu cho sách:

+Để nhập dữ liệu cho sách ta xây dựng hàm **void Nhap(Sach \*bookList)**. Để lưu trữ thông tin của ba quyển sách ta dùng con trỏ **\*bookList** với kiểu dữ liệu là **Sach**. Để có bộ nhớ lưu trữ ba quyển sách, ta phải cấp phát bộ nhớ cho con trỏ **\*bookList** bằng hàm **malloc** trong hàm **int main()**:

```
int main(){  
    Sach *bookList= (Sach*) malloc ( 3* sizeof(Sach)) ;
```

+Xây dựng hàm **void Nhap(Sach \*bookList)**: Dùng câu lệnh vòng lặp **for()** để thực hiện nhập sách. Với ba quyển sách, ta thực hiện ba vòng lặp để nhập sách. Dùng hàm **gest()** để nhập chuỗi kí tự (tên sách và thể loại sách) và dùng hàm **scanf()** để nhập số (giá tiền), dùng hàm **fflush(stdin)** để xóa bộ nhớ đệm trước khi đọc. Ta được hàm nhập dưới đây:

```
void Nhap(Sach *bookList ){  
    int i;  
    for( i=0; i<3; i++){  
        printf("Nhap quyen sach %d:\n", i+1) ;  
        printf("Ten: ");  
        fflush(stdin);  
        gets((bookList+i)->ten);  
        printf("The loai: ");  
        fflush(stdin);  
        gets((bookList+i)->theloai);  
        printf("Gia tien: ");  
        scanf("%d",&(bookList+i)->giatien);
```

```

    }
}

```

-Sắp xếp, thống kê và hiển thị chi tiết thông tin của từng quyển sách theo thể loại ( $Z \rightarrow A$ ):

+Sắp xếp: Để sắp xếp sách theo thể loại ta dùng phương pháp sắp xếp nổi bọt. Để thực hiện phương pháp này ta dùng hai vòng lặp **for()**, một vòng lặp **for (int i= 0; i<2 ; i++)** và một vòng lặp **for (int j=i; j<3; j++)**. Để so sánh hai chuỗi kí tự (so sánh hai thể loại) với nhau ta dùng hàm **strcmp()**. Hàm sắp xếp sau khi xây dựng xong:

```

for ( int i= 0; i<2 ; i++) {
    for ( int j=i; j<3; j++){
        if( strcmp((bookList[i]).theloai,bookList[j].theloai)==-1) {
            Sach t=bookList[i];
            bookList[i]=bookList[j];
            bookList[j]=t;
        }
    }
}

```

+Thống kê:

.Sau khi sắp xếp theo thể loại từ ( $Z \rightarrow A$ ) nên ta có bốn trường hợp xảy ra khi thống kê thể loại sách:

- ..Trường hợp 1: Có ba quyển sách cùng thể loại.
- ..Trường hợp 2: Hai quyển sách đầu cùng thể loại.
- ..Trường hợp 3: Ba quyển sách với ba thể loại khác nhau.

..Trường hợp 4: Hai quyển cuối cùng thể loại.

.Để kiểm tra các trường hợp xảy ra ta dùng hai dòng lệnh **for()**. Dùng hàm **strcmp()** để so sánh hai chuỗi kí tự (thể loại sách) với nhau. Dùng biến **t** và câu lệnh **for(i=0; i<2; i++)** để đếm thể loại trùng nhau từ đầu đến cuối:

```
int t=0;
    for(i=0; i<2; i++){
        if(strcmp(bookList[0].theloai,bookList[i+1].theloai)==0)
            t=t+1;
    }
```

.Dùng biến **x** và câu lệnh **for(j=2;j>0;j--)** để đếm thể loại trùng nhau từ cuối đến đầu:

```
int x=0;
    for(j=2;j>0;j--){
        if(strcmp(bookList[2].theloai,bookList[j-1].theloai)==0)
            x=x+1;
    }
```

Nếu **t=2** thì có ba quyển sách cùng thể loại.

```
if(t==2){
    printf("%s co 3 quyen sach\n",bookList[0].theloai);
}
```

.Nếu t=1 thì hai quyển sách đầu cùng thể loại.

```
if(t==1){  
    printf("%s co 2 quyen sach\n%s co 1 quyen  
sach\n",bookList[0].theloai,bookList[2].theloai);  
}
```

.Nếu t=0 có 2 trường hợp xảy ra:

..Trường hợp 1: x=0 ba quyển sách có ba thể loại khác nhau.

```
if(t==0){  
    if(x==0){  
        printf("%s co 1 quyen sach\n%s co 1 quyen sach\n%s co 1  
quyen sach\n",bookList[0].theloai,bookList[1].theloai,bookList[2].theloai);  
    }  
}
```

..Trường hợp 2: x=1 hai quyển sách cuối cùng thể loại.

```
if(t==0){  
    if(x==1){  
        printf("%s co 1 quyen sach\n%s co 2 quyen  
sach\n",bookList[0].theloai,bookList[2].theloai);  
    }  
}
```

+Ta được hàm sắp xếp, thống kê chi tiết thông tin của từng quyển sách theo thể loại ( $Z \rightarrow A$ ) void Sapxep(Sach bookList[]):

```
void Sapxep(Sach bookList[]){
```

```
//Sap xep theo the loai tu Z->A
```

```
    for ( int i= 0; i<2 ; i++) {
```

```
        for ( int j=i; j<3; j++){
```

```
            if( strcmp((bookList[i]).theloai,bookList[j].theloai)==-1) {
```

```
                Sach t=bookList[i];
```

```
                bookList[i]=bookList[j];
```

```
                bookList[j]=t;
```

```
            }
```

```
        }
```

```
    }
```

```
    printf("STT||Ten           ||The loai           ||Gia tien           \n");
```

```
    for ( i=0; i<3; i++){
```

```
        printf("%0.3d||%-20s||%-20s||%d\n",i+1,(bookList[i]).ten,(bookList[i]).theloai,(bookList[i]).giatien);
```

```
    }
```

```
//Thong ke the loai sach
```

```
    int t=0,x=0;
```

```
    for(i=0; i<2; i++){
```

```
        if(strcmp(bookList[0].theloai,bookList[i+1].theloai)==0){
```

```
            t=t+1;
```

```
        }
```

```
    }
```

```

for(j=2;j>0;j--){
    if(strcmp(bookList[2].theloai,bookList[j-1].theloai)==0)
        x=x+1;
}
if(t==2){
    printf("%s co 3 quyen sach\n",bookList[0].theloai);
}
if(t==1){
    printf("%s co 2 quyen sach\n%s co 1 quyen
sach\n",bookList[0].theloai,bookList[2].theloai);
}
if(t==0){
    if(x==0){
        printf("%s co 1 quyen sach\n%s co 1 quyen sach\n%s co 1
quyen sach\n",bookList[0].theloai,bookList[1].theloai,bookList[2].theloai);
    }
    if(x==1){
        printf("%s co 1 quyen sach\n%s co 2 quyen
sach\n",bookList[0].theloai,bookList[2].theloai);
    }
}
}

```

-Tìm kiếm theo thể loại:

+Đầu tiên ta cần nhập thể loại cần tìm kiếm vào. Để nhập thể loại vào ta dùng mảng **char TentheLoai[30]** để lưu trữ chuỗi kí tự. Dùng hàm **fflush(stdin)** để xóa bộ nhớ đệm, và dùng hàm **gets()** để lưu chuỗi kí tự. Khai báo mảng **Sach book[3]**



có kiểu dữ liệu là **Sach** để lưu trữ những quyển sách có cùng thể loại đang tìm. Dùng lần lượt hai biến **t=0** và **x=0** để đếm số sách cùng thể loại cần tìm và tăng vị trí phần tử trong mảng. Dùng câu lệnh **for()** và hàm **strcmp()** để tạo vòng lặp và so sánh hai chuỗi kí tự (thể loại sách) với nhau.

```
char Tentheloai[30];  
printf("Nhap the loai can tim vao: ");  
fflush(stdin);  
gets(Tentheloai);  
Sach book[3];  
int t=0,x=0;  
int i;  
for( i = 0; i < 3; i++){  
    if(strcmp(Tentheloai,bookList[i].theloai)==0){  
        t=t+1;  
        book[x]=bookList[i];  
        x=x+1;  
    }  
}
```

+Nếu  $t = 0$  thì không có quyển sách thuộc thể loại đang tìm.

```
if(t==0) printf("Khong co quyen sach thuoc the loai nay\n");
```

+Nếu  $t \neq 0$  thì in ra thể loại cần tìm và số quyển sách.

```

else{
    printf("STT||Ten          ||The loai          ||Gia tien          \n");
    for( i=0; i<t; i++){
        printf("%0.3d||%-20s||%-
20s||%d\n",i+1,(book[i]).ten,(book[i]).theloai,(book[i]).giatien);
    }
}

```

+Ta được hàm tìm kiếm sách **void Timkiem(Sach bookList[])**:

```

void Timkiem(Sach bookList[]){
    char Tentheloai[30];
    printf("Nhap the loai can tim vao: ");
    fflush(stdin);
    gets(Tentheloai);
    Sach book[3];
    int t=0,x=0;
    int i;
    for( i = 0; i < 3; i++){
        if(strcmp(Tentheloai,bookList[i].theloai)==0){
            t=t+1;
            book[x]=bookList[i];
            x=x+1;
        }
    }
    if (t==0) printf("Khong co quyen sach thuoc the loai nay\n");
}

```

```

else{
    printf("STT||Ten          ||The loai          ||Gia tien          \n");
    for( i=0; i<t; i++){
        printf("%0.3d||%-20s||%-
20s||%d\n",i+1,(book[i]).ten,(book[i]).theloai,(book[i]).giatien);
    }
}
}
}

```

-Ghi tập tin nhị phân **book.dat**:

+Để làm việc với tệp, đầu tiên ta phải mở tệp. Dùng con trỏ **\*o** và câu lệnh **fopen()** để mở tệp.

```
FILE *o = fopen("\\BaitapLon\\book.dat","wb");
```

+Nhập file:

.Nếu **o == NULL** thì file bị lỗi.

```
if(o == NULL) printf("Loi nhap file");
```

.Ngược lại thì nhập file thành công. Dùng hàm **fwrite()** để lưu file nhị phân.

```
else {fwrite(bookList, sizeof(Sach),1,o);
```

```
printf("\nNhap file thanh cong!");
```

+Đóng file: Sau khi làm việc với file xong thì ta đóng file bằng hàm **fclose()**.

**fclose(o);**

+Ta được hàm ghi file nhị phân **void GhiVaotapin(Sach \*bookList):**

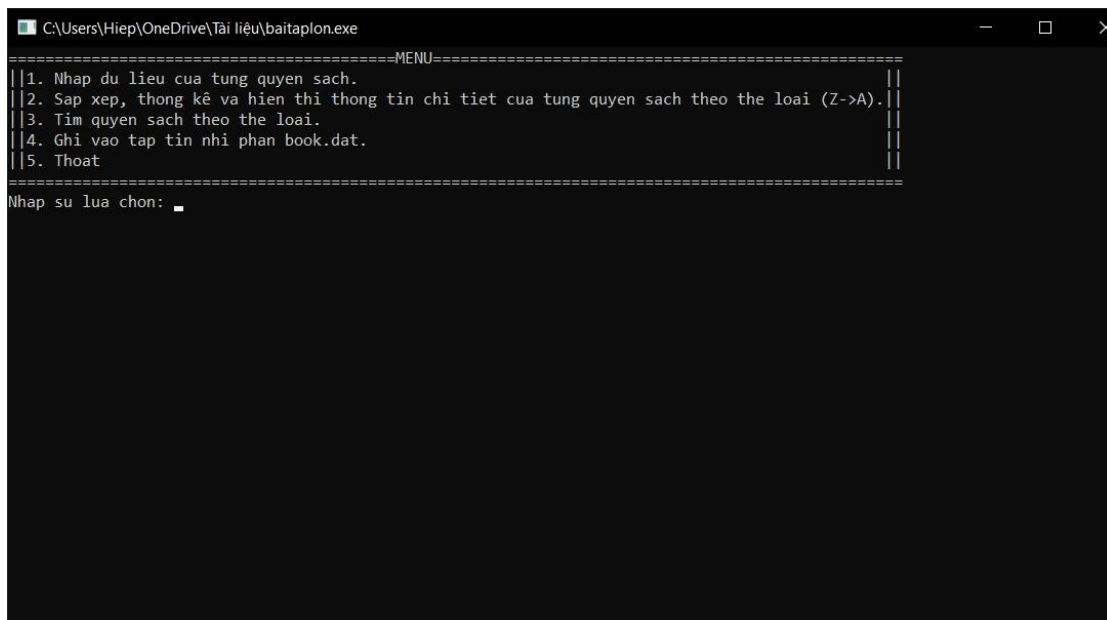
```
void GhiVaotapin(Sach *bookList){  
    FILE *o = fopen("\\btlon\\book.dat","wb+");  
    if(o == NULL) printf("Loi nhap file");  
    else {fwrite(bookList, sizeof(Sach),1,o);  
  
        printf("\nNhap file thanh cong!");  
    } fclose(o);  
}
```

-Thiết kế bảng Menu:

+Ý tưởng: Thiết kế Menu bằng hàm **void Menu(Sach bookList[])**. Để thiết kế Menu ta dùng cấu trúc rẽ nhánh **switch....case** để tạo các sự lựa chọn trong Menu. Đồng thời ta kết hợp sử dụng vòng lặp **do....while** để tạo ra sự lặp lại trong Menu.

+Đi vào cụ thể:

.Menu sẽ có dạng:



Hình 16

.Đầu tiên ta cần in bảng Menu ra để người sử dụng lựa chọn

```

printf("\n\t=====MENU=====\\n");

printf("\t|1. Nhập dữ liệu của từng quyển sách.\\n");

printf("\t|2. Sắp xếp, thống kê và hiển thị thông tin chi tiết của từng
quyển sách theo thể loại (Z->A).\\n");

printf("\t|3. Tìm quyển sách theo thể loại.\\n");

printf("\t|4. Ghi vào tập tin nhị phân book.dat.\\n");

printf("\t|5. Thoát\\n");

printf("\t=====\\n");

```

.Tạo biến **x** để nhập sự lựa chọn của người sử dụng và biến **x** này được đặt trong câu lệnh **do** để thực hiện nhiều lần. Câu lệnh **switch... case** được đặt trong câu lệnh **do** để cho người sử dụng chương trình vô số lần. **Case 1** ta đưa hàm **Nhap(bookList)** vào để nhập thông tin sách. **Case 2** ta đưa hàm **Sapxep(bookList)** vào để sắp xếp, thống kê hiển thị sách. **Case 3** ta đưa hàm **Timkiem(bookList)** vào để tìm kiếm sách theo thể loại. **Case 4** ta đưa hàm **Ghivaotaptin(bookList)** vào để in tập tin nhị phân. **Case 5** ta đưa hàm **exit(0)** vào để thoát khỏi chương trình. Câu lệnh **getch()** để dừng màn hình và hàm **system("cls")** để xóa màn hình.

```
do {  
  
    printf(" \nNhap su lua chon: ");  
    scanf("%d",&x);  
    switch(x) {  
        case 1:{  
            Nhap(bookList);  
            getch();  
            return Menu(bookList);  
        }  
        case 2:{  
            Sapxep(bookList);  
            getch();  
            return Menu(bookList);  
        }  
        case 3:{  
            Timkiem(bookList) ;  
            getch();  
            return Menu(bookList);  
        }  
    }  
}
```

```

    }
    case 4:{
        Ghivaotaptin(bookList);
        getch();
        return Menu(bookList);
    }
    case 5:{
        exit(0);
        break;
    }
}
} while( x<=5);
}

```

+Ta được hàm **void Menu(Sach bookList[])**:

```

void Menu(Sach bookList[]){
    system("cls");

    printf("\n\t=====
=MENU=====\\n");

    printf("\t|1. Nhap du lieu cua tung quyen sach.
||\\n");

    printf("\t|2. Sap xep, thong ke va hien thi thong tin chi tiet cua tung
quyen sach theo the loai (Z->A).||\\n");

    printf("\t|3. Tim quyen sach theo the loai.
||\\n");

```

```

        printf("\t||4. Ghi vao tap tin nhi phan book.dat.
||\n");

        printf("\t||5. Thoat                                     ||\n");

        printf("\t=====
=====\\n");

        int x;

        do {

            printf(" \nNhap su lua chon: ");
            scanf("%d",&x);
            switch(x) {
                case 1:{
                    Nhap(bookList);
                    getch();
                    return Menu(bookList);
                }
                case 2:{
                    Sapxep(bookList);
                    getch();
                    return Menu(bookList);
                }
                case 3:{
                    Timkiem(bookList) ;
                    getch();
                    return Menu(bookList);
                }
            }
        }
    }

```



```

        case 4:{
            Ghivaotaptin(bookList);
            getch();
            return Menu(bookList);
        }
        case 5:{
            exit(0);
            break;
        }
    }
    } while( x<=5);
}

```

-Xây dựng hàm chính cho chương trình: Hàm int main(): Ta chỉ cần gọi hàm Menu(bookList) vào trong hàm chính để nó thực hiện nhiệm vụ. Câu lệnh free() để giải phóng bộ nhớ cho con trỏ \*bookList. Ta được hàm int main():

```

int main(){
    Sach *bookList= (Sach*) malloc ( 3* sizeof(Sach)) ;
    Menu(bookList);
    getch();
    free(bookList);
}

```

## CHƯƠNG 3 TỔNG KẾT

### 3.1 Kết quả đạt được

- Chúng em đã hoàn thành được bài tập nhóm, chương trình quản lí sách đã chạy được. đã có thể hiểu được thêm về các thuật toán trong C và cách ứng dụng của nó trong từng trường hợp và đã có những cải thiện thêm về tư duy lập trình, ngoài ra chúng em được lần đầu tiên tiếp xúc với bài tập lớn ở bậc đại học và cách trình bày ứng dụng word làm sao cho bài mình làm ra có tính logic, trình bày dễ nhìn, dễ hiểu và cách làm powerpoint có thể chưa chuyên nghiệp nhưng đã cải thiện các kĩ năng chỉnh sửa và đưa nội dung hình ảnh phù hợp. Ngoài những kết quả đạt được ở trên điều quan trọng hơn là chúng em đã hiểu cách làm việc nhóm hiệu quả ban đầu có nhiều làm bài tập có nhiều ý kiến trái chiều nhưng sau khi họp lại chúng em đã đưa ra được ý kiến chung. Những kết quả đạt được sau khi làm bài bên trên đã được tất cả các thành viên trong nhóm chúng em nhận xét và đưa ra ý kiến của mình.

### 3.2 Nhược điểm

Chương trình còn chưa thân thiện với người dùng. Giao diện đơn giản, chỉ có hai màu trắng và đen. Chỉ chạy được những chức năng cơ bản, yêu cầu người dùng thực hiện chức năng đầu tiên để thực hiện những chức năng tiếp theo.

Nhóm chưa đạt được hiệu quả cao, kiến thức của các thành viên trong nhóm còn chưa được tốt. Thời gian làm việc phân chia chưa hợp lý. Còn nhiều thiếu sót, lỗi xảy ra liên tục trong suốt quá trình thực hiện.

### 3.3 Hướng phát triển

Thêm chức năng vào chương trình: Đọc file để lưu lại dữ liệu cho lần sử dụng tiếp theo, xóa sao chép dữ liệu, sắp xếp, thực hiện nhiều chức năng cùng 1 lúc.

Chỉnh sửa lại giao diện: thân thiện, dễ nhìn, dễ sử dụng hơn.

## TÀI LIỆU THAM KHẢO

Giáo trình Lập trình nâng cao – Trường Đại học Giao thông Vận Tải

Một số liên kết:

[https://vietjack.com/lap\\_trinh\\_c/](https://vietjack.com/lap_trinh_c/)

<https://nguyenvanhieu.vn/khoa-hoc-lap-trinh-c/>

<http://www.cplusplus.com/>

<https://en.wikipedia.org/>

<https://vi.wikipedia.org/>

<https://www.google.com.vn/>