

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

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

**BÁO CÁO BÀI TẬP LỚN**

LẬP TRÌNH NÂNG CAO

CHƯƠNG TRÌNH QUẢN LÝ SÁCH THƯ VIỆN TẠI UTC2

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | **Trần Thị Dung** |
| **Sinh viên: Đặng Ngọc Chung**  **Lê Âu Hải**  **Nguyễn Thành Đạt**  **Mã học phần:**  **CPM215.3** |  |

MỤC LỤC

[Phần I: Sơ Lược Về Đề Tài 4](#_Toc42632521)

[1.1 Lí do chọn đề tài 4](#_Toc42632522)

[1.2 Mục đích đề tài 4](#_Toc42632523)

[Phần II: Làm Việc Với Tệp 5](#_Toc42632524)

[2.1 Làm việc với tệp 5](#_Toc42632525)

[a. File văn bản – text files 5](#_Toc42632526)

[b. File nhị phân – Binary files 6](#_Toc42632527)

[Các thao tác với file 6](#_Toc42632528)

[Thao tác mở file 7](#_Toc42632531)

[Thao tác đóng file 8](#_Toc42632532)

[Đọc/Ghi file văn bản trong C 8](#_Toc42632533)

[Đọc/Ghi file nhị phân trong C 10](#_Toc42632534)

[Một số ví dụ về đọc ghi file trong C 12](#_Toc42632535)

[2.2 Danh sách cấu trúc liên kết đơn: 13](#_Toc42632536)

[a. Danh sách liên kết là gì? 13](#_Toc42632537)

[b. Cài đặt danh sách liên kết đơn 14](#_Toc42632538)

[b.1. Khai báo linked list 14](#_Toc42632539)

[b.2. Tạo mới 1 Node 15](#_Toc42632540)

[b.3. Thêm Node vào danh sách liên kết 16](#_Toc42632541)

[b.4. Xóa Node khỏi danh sách liên kết 18](#_Toc42632542)

[b.5. Lấy giá trị ở vị trí bất kỳ 20](#_Toc42632543)

[b.6. Tìm kiếm trong danh sách liên kết 21](#_Toc42632544)

[\*) Duyệt danh sách liên kết 21](#_Toc42632545)

[2.3 Các thuật toán sắp xếp 22](#_Toc42632546)

[a. Sắp xếp dãy số giảm dần 22](#_Toc42632547)

[b. Sắp xếp dãy số tăng dần 23](#_Toc42632548)

[Ví dụ minh họa 24](#_Toc42632549)

[Code sắp xếp nổi bọt trong C/C++ 25](#_Toc42632550)

[26](#_Toc42632551)

[2.4 Các thuật toán tìm kiếm 27](#_Toc42632552)

[Minh họa code cho thuật toán tìm kiếm nhị phân 28](#_Toc42632553)

[Phần III: Demo code Đề tài quản lý sách thư viện 33](#_Toc42632554)

[3.1 Các hàm trong chương trình 33](#_Toc42632555)

Phần I: Sơ Lược Về Đề Tài

1.1 Lí do chọn đề tài

Chúng ta đang sống trong những năm đầu của thế kỉ XXI, thế kỉ phát triển của nghành công nghệ thông tin (CNTT). Vì vậy việc tiếp cận và cập nhật công nghệ này đã, đang và sẽ mang lại cho chúng ta rất nhiều lợi ích và thuận tiện trong công việc: tiết kiệm thời gian cũng như giảm các chi phí trong công tác quản lí.

Trong những năm gần đây CNTT đóng vai trò rất quan trọng: song song với việc phát triển ứng dụng tin học vào trong cuộc sống, tin học được ứng dụng rất rộng rãi trong các lĩnh vực khoa học kĩ thuật, các ngành quản lí và các hoạt động xã hội thông qua mạng Internet. Đóng vai trò quan trọng như trên nên tin học đã được đưa vào quản lí nhân sự, điểm, bán hàng, nhập xuất vật tư. Vấn đề mượn trả sách cần được quản lí hơn vì việc nhập các thông tin bằng phương pháp viết tay sẽ tốn thời gian và việc quản lí sách sẽ khó khăn hơn.

Vì vậy em xây dựng phần mềm quản lí sách thư viện.

Phần mềm các chức năng chính sau:

+ Quản lý người mượn, trả sách

+ Nhập dữ liệu liên quan đến sách cần mượn, trả

+ Tìm kiếm, sắp xếp

+ Quản lí mượn, trả sách

+ Quản trị

+ Trợ giúp

Trong quá trình làm còn nhiều sai sót, em mong nhận được những ý kiến đánh giá của cô.

1.2 Mục đích đề tài

* Tiết kiệm tối đa nguồn lực và thời gian
* Giao diện đơn giản dễ cài đặt và sử dụng
* Thông tin về mã sách được quản lí tập trung giúp dễ dàng tìm kiếm và phân loại
* Quản lý thông tin, thời gian mượn sách, trả sách
* Dễ dàng trao đổi và hỗ trợ xử lí công việc giữa các nhân viên và các ca làm
* Dễ dàng tùy chỉnh phần mêm khi quy định công việc thay đổi
* Phân quyền sử dụng hệ thống để đảm bảo tính bảo mật: Quyền người sử dụng và quản lý.
* Phần mềm quản lý thư viện hỗ trợ đắc lực cho cán bộ quản lý nguồn tài nguyên hiện có trong thư viện nhanh chóng và chính xác. Đồng thời giúp ban lãnh đạo đưa ra những quyết định kịp thời trong việc bổ sung nguồn tài nguyên mới cũng như mất mát, hư hỏng trong thư viện.

Phần II: Làm Việc Với Tệp

2.1 Làm việc với tệp

Trước khi bạn làm việc với file, bạn nên biết về 2 kiểu file khác nhau sau đây:

1. File văn bản – text files
2. File nhị phân – binary file

a. 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 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.

b. File 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.

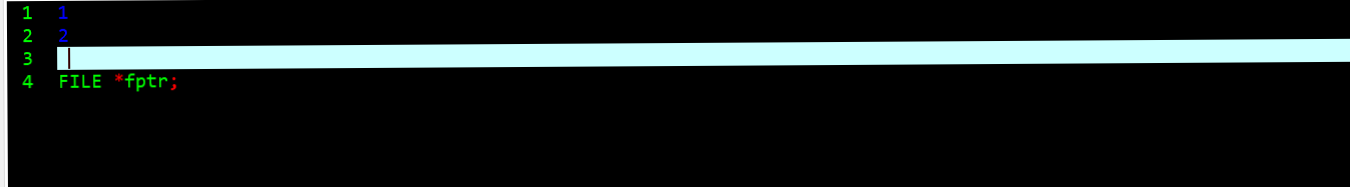
Các thao tác với file

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

## Thao tác với file trên ngôn ngữ C

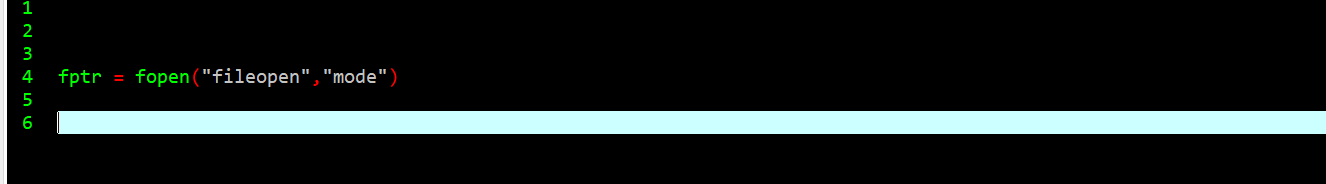
Khi làm việc với file, bạn cần khai báo 1 con trỏ kiểu FILE. Việc khai báo này là cần thiết để có sự kết nối giữa chương trình của bạn và tập tin mà bạn cần thao tác.



### Thao tác mở file

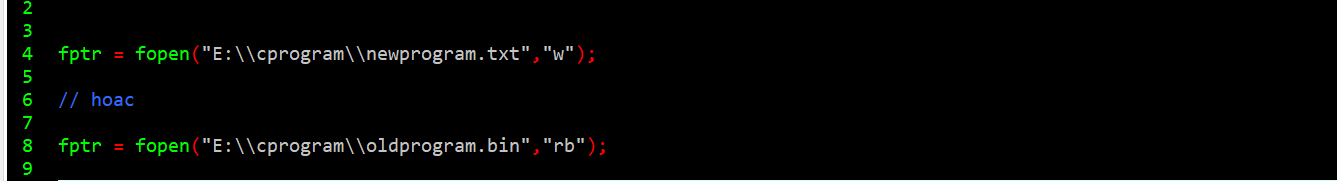
Để đọc ghi file trong C cũng như trong mọi ngôn ngữ lập trình, việc đầu tiên bạn cần làm là mở file mà bạn muốn làm việc. Trong ngôn ngữ lập trình C, chúng ta có thể mở file bằng cách sử dụng hàm fopen() trong thư viện stdio.h như sau:0

fptr = fopen("fileopen","mode")



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

Ví dụ:

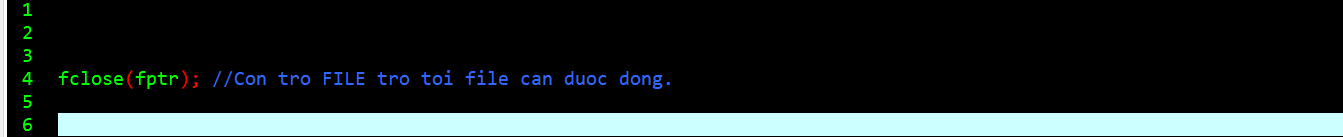


* Giả sử tập tin newprogram.txt chưa có trong thư mục E:\cprogram. Ví dụ đầu tiên với mode = "w" sẽ cho phép chương trình tự động tạo ra file newprogram.txt nếu nó chưa có. Và sau đó mở file này lên nhưng chương trình chỉ có thể ghi dữ liệu vào mà không thể đọc.
* Mode là w chỉ cho phép chương trình 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.

### 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().

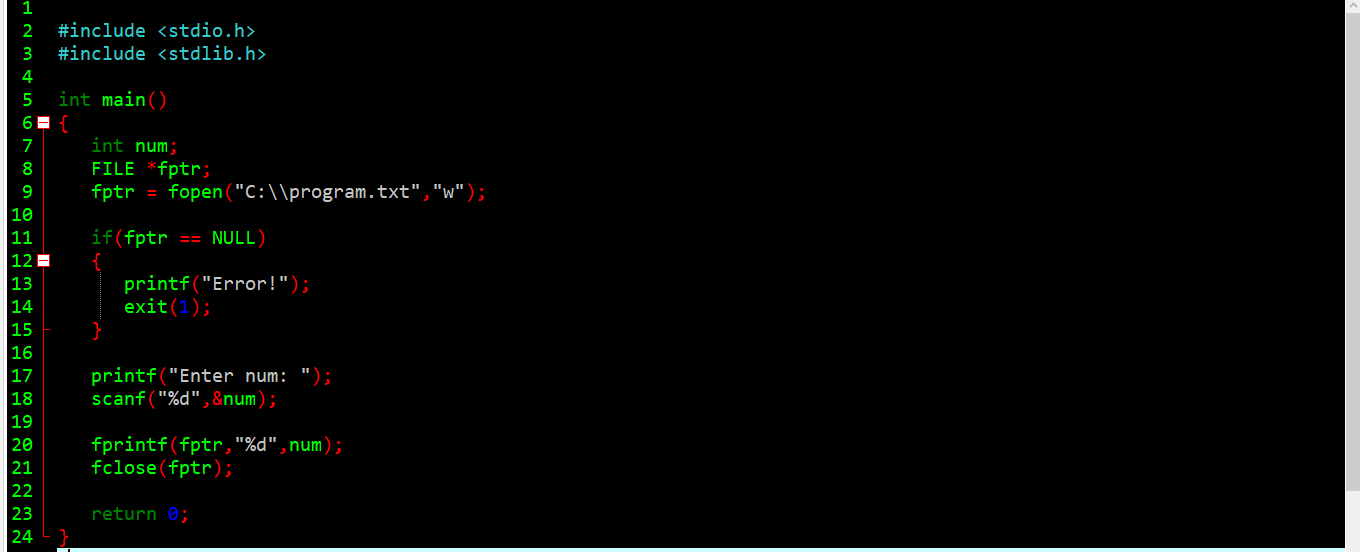


## Đọc/Ghi file văn bản trong C

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().

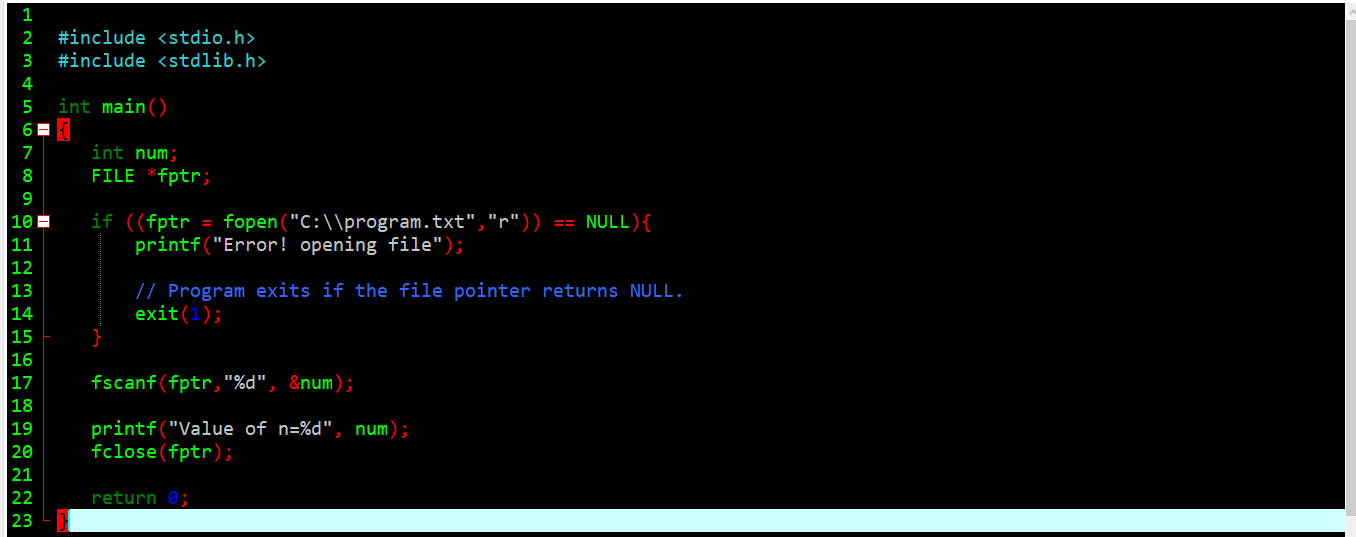
#### **VD1. Ghi file sử dụng fprintf()**



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()**



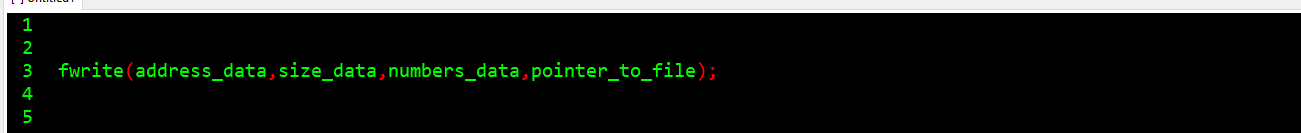
Chương trình này sẽ đọc giá trị số được lưu trong file program.txt mà chương trình ở VD1 vừa tạo ra và in lên màn hình.

## Đọc/Ghi file nhị phân trong C

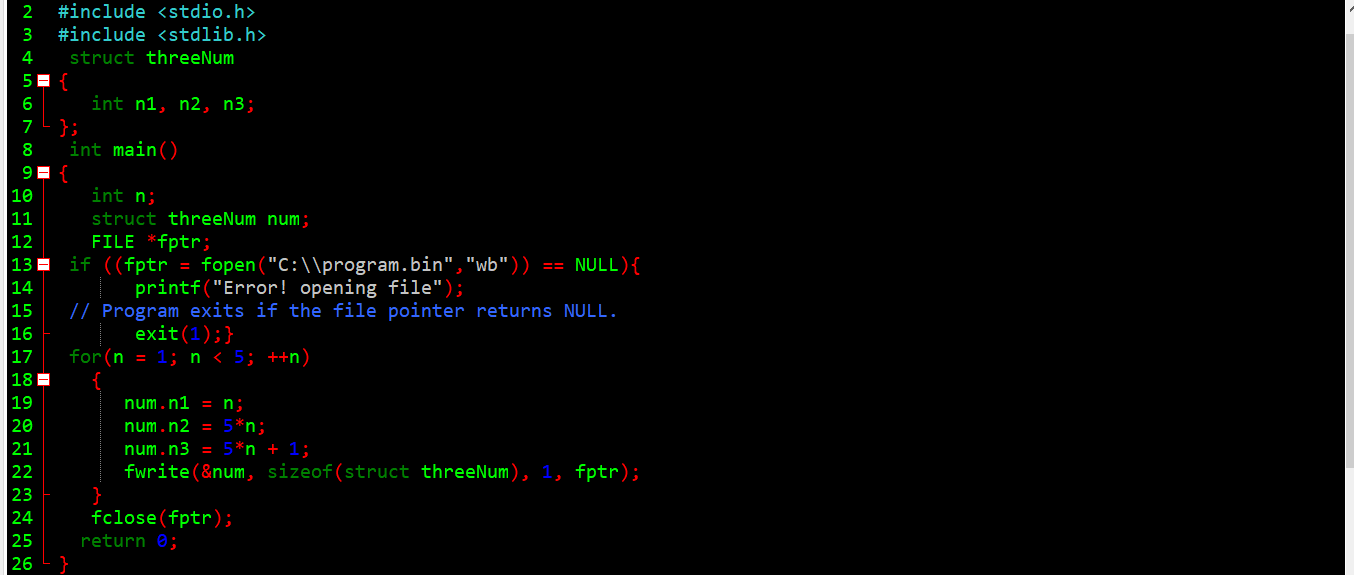
Các hàm fread() và fwrite() trong C được sử dụng để đọc và ghi file trong C ở dạng nhị phân.

#### **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.



#### **VD3. Ghi file nhị phân sử dụng fwrite()**



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 n1, n2, n3; Và nó được sử dụng trong hàm main có tên biến là num.

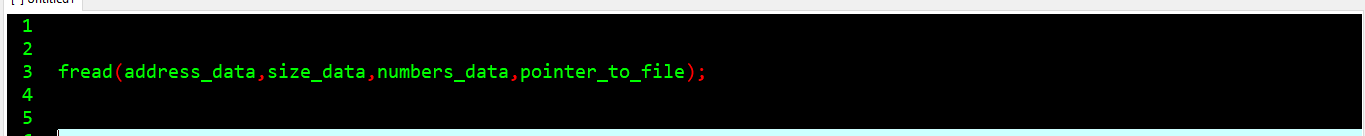
Trong vòng lặp, các số được ghi vào file sử dụng hàm fwrite(). Các tham số gồm:

* Tham số đầu tiên là địa chỉ của biến num
* Tham số thứ 2 là kích thước của biến num
* 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

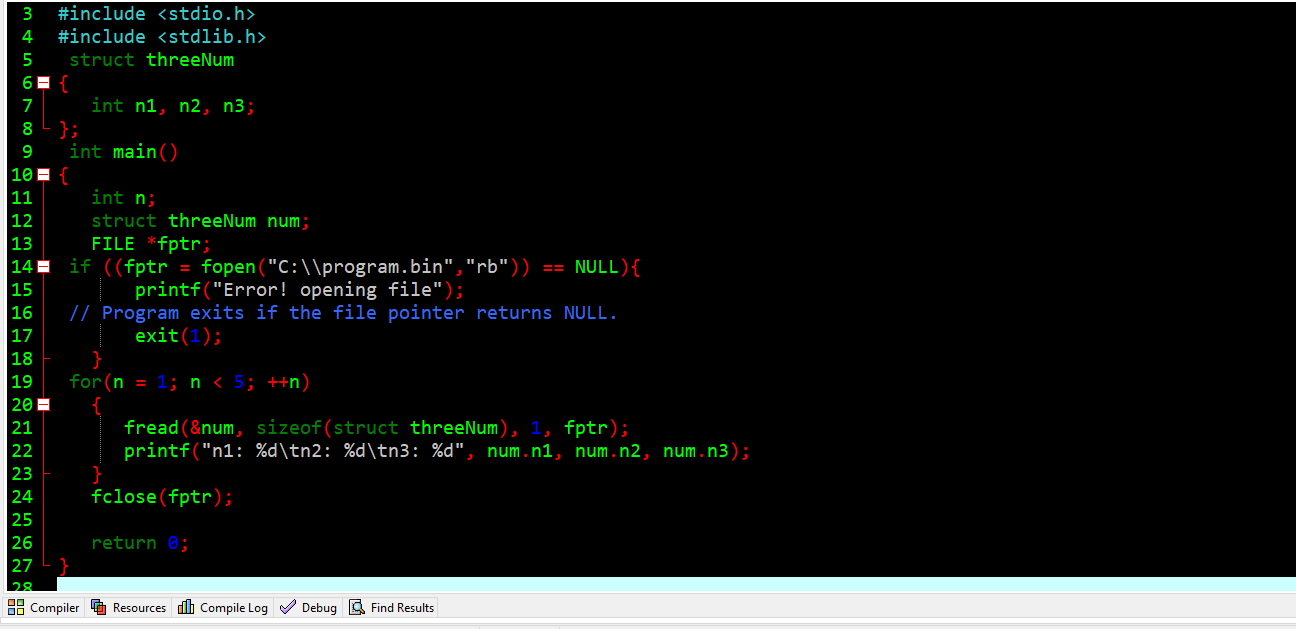
Cuối cùng, chúng ta đóng file sử dụng fclose().

#### **Đọc file nhị phân**

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



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



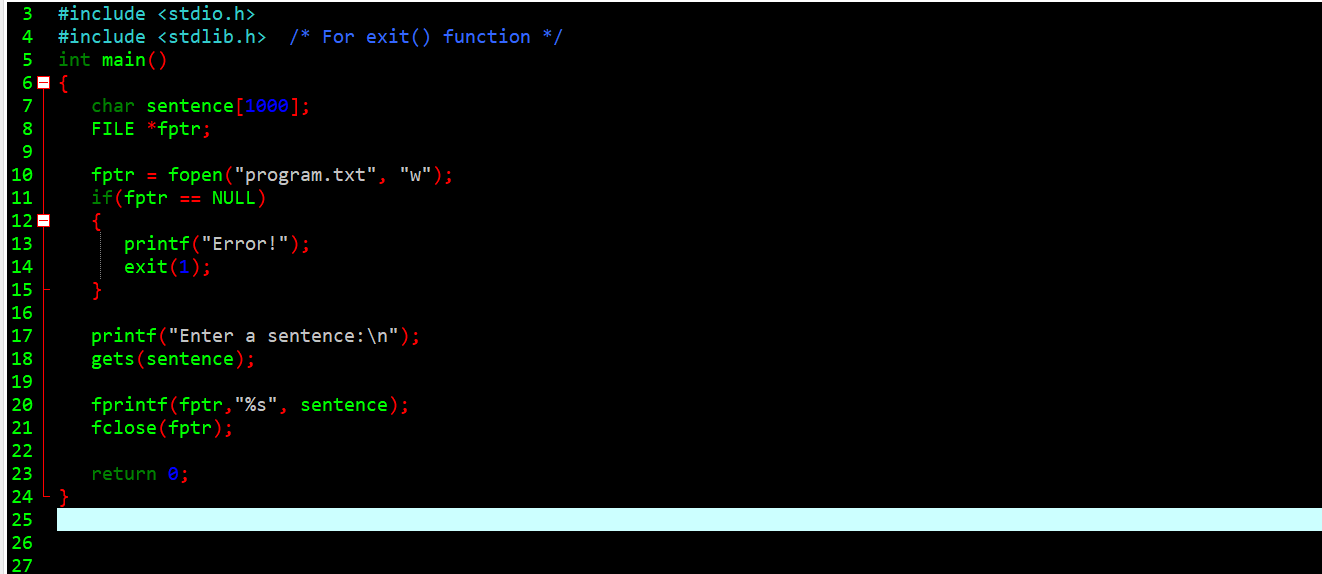
Trong ví dụ này, bạn đọc file program.bin và lặp qua từng dòng. Bạn sẽ nhận được các giá trị tương ứng khi bạn ghi vào trong VD3.

## 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

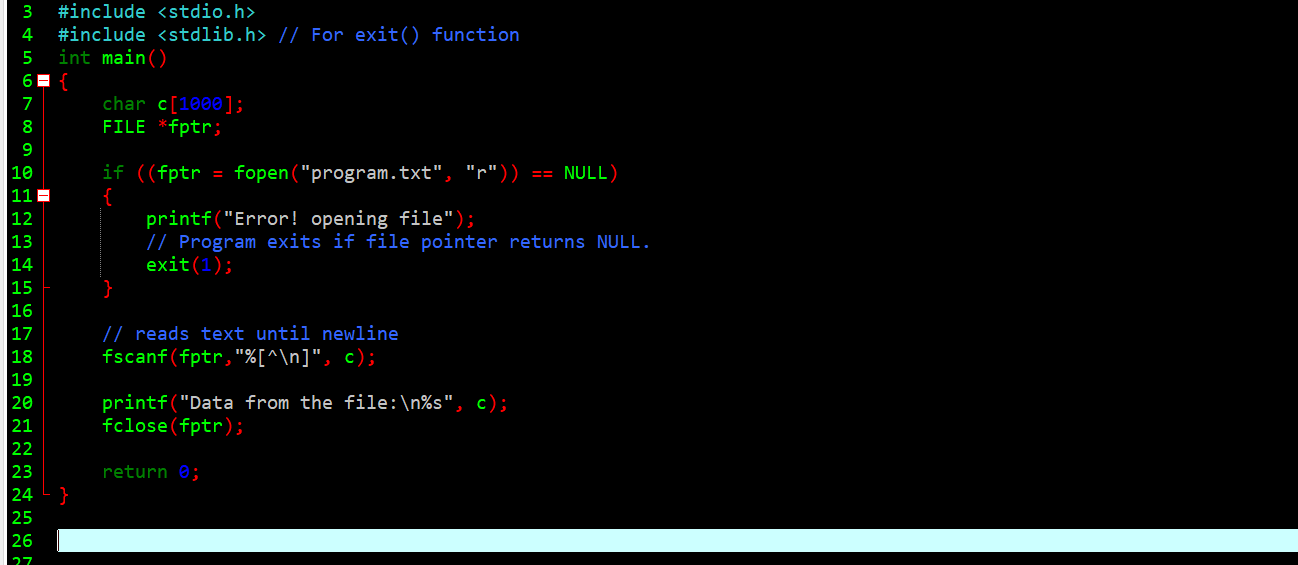
#### **Ghi vào file một câu văn bản**



Chạy thử:

|  |  |
| --- | --- |
| 0  1  2  3 | Enter sentence:  I am awesome and so are files. |

#### **Đọc dữ liệu văn bản từ file**



2.2 Danh sách cấu trúc liên kết đơn:

## a. 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, nghĩa là đó là phần tử cuối cùng của linked list.

Hình ảnh mô tả cho một Node trong danh sách liên kết đơn:



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



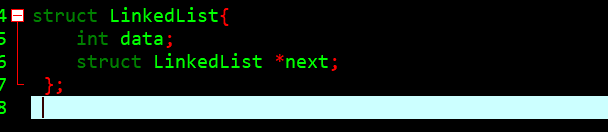
Danh sách các kiểu danh sách liên kết:

* 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.

b. Cài đặt danh sách liên kết đơn

b.1. 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.

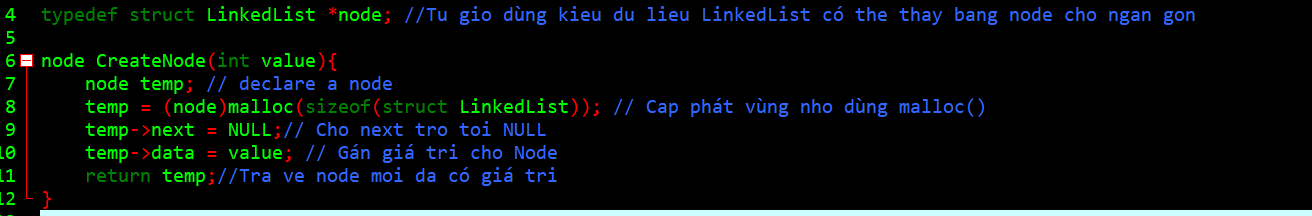


Khai báo trên sẽ được sử dụng cho mọi Node trong linked list. Trường data sẽ lưu giữa giá trị và next sẽ là con trỏ để trỏ đến thằng kế tiếp của nó.

**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.

### b.2. Tạo mới 1 Node

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



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

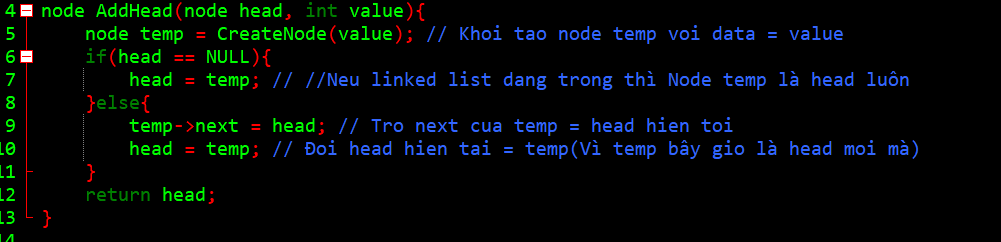
**Lưu ý:** Không giống với mảng, cần khai báo arr[size]. Trong linked list, vì mỗi Node sẽ có con trỏ liên kết đến Node tiếp theo. Do đó, với danh sách liên kết đơn, bạn chỉ cần lưu giữ Node đầu tiên(HEAD). Có head rồi bạn có thể đi tới bất cứ Node nào.

### b.3. 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:

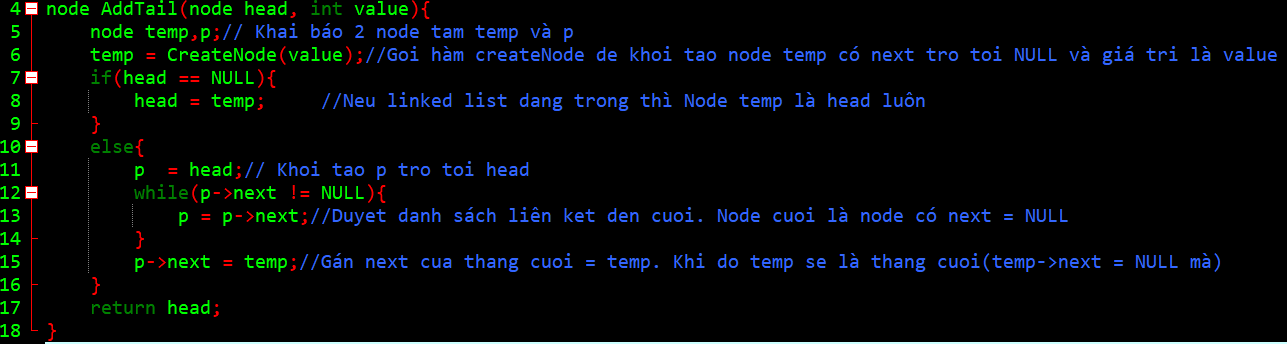
* Nếu head đang trỏ tới NULL, nghĩa là linked list đang trống, Node mới thêm vào sẽ làm head luôn
* Ngược lại, ta phải thay thế thằng head cũ bằng head mới. Việc này phải làm theo thứ tự như sau:
  + Cho next của temp trỏ tới head hiện hành
  + Đặt temp làm head mới



#### **Thêm vào cuối**

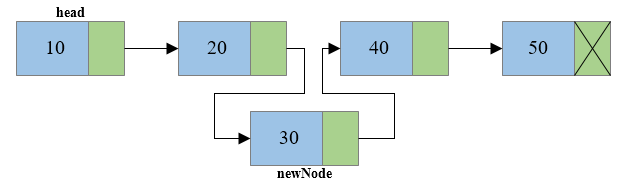
Chúng ta sẽ cần Node đầu tiên, và giá trị muốn thêm. Khi đó, ta sẽ:

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).



Tổng quan hơn, chúng ta sẽ sẽ viết hàm thêm một Node vào vị trí bất kỳ nhé.

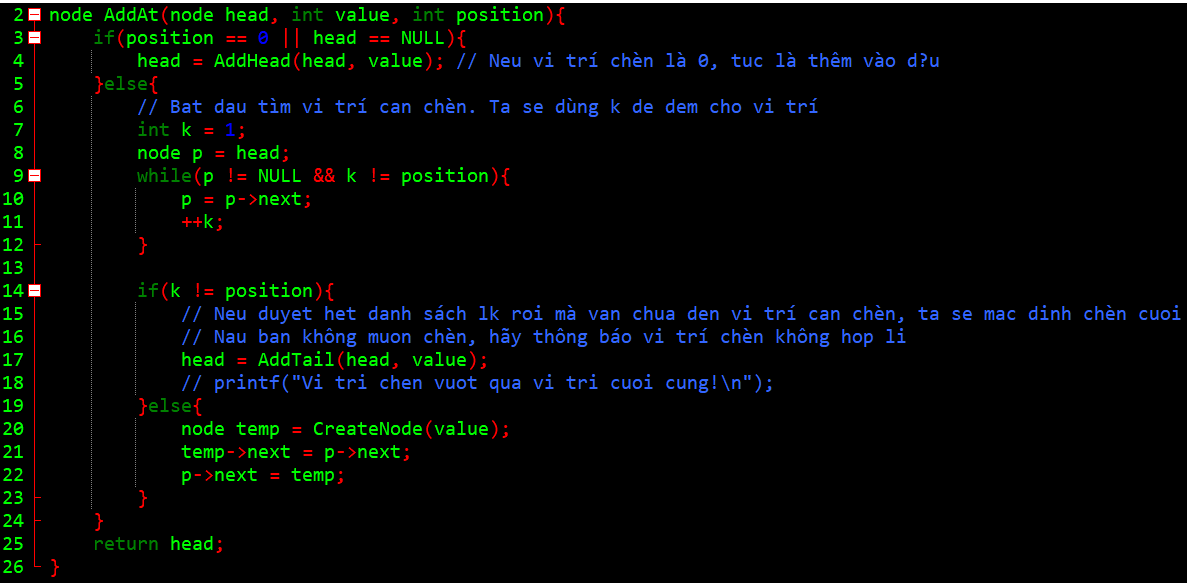
#### **Thêm vào vị trí bất kỳ**



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 nhé các bạn

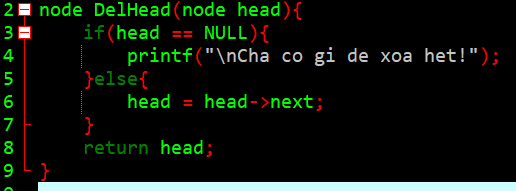


Lưu ý: Bạn phải làm theo thứ tự trên, nếu bạn cho p->next = temp trước. Khi đó, bạn sẽ không thể lấy lại phần sau của danh sách liên kết nữa(Vì next chỉ được được lưu trong p->next mà thay đổi p->next rồi thì còn đâu giá trị cũ).

### b.4. Xóa Node khỏi danh sách liên kết

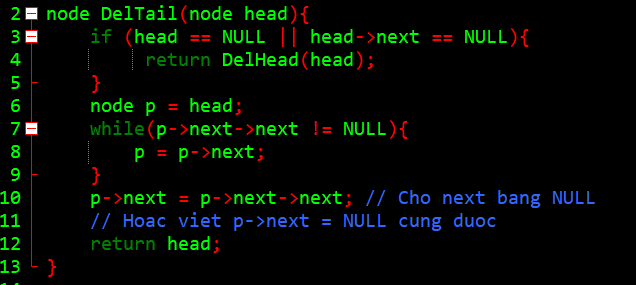
#### **Xóa đầu**

Xóa đầu đơn giản lắm, bây giờ chỉ cần cho thằng kế tiếp của head làm head là được thôi. Mà thằng kế tiếp của head chính là head->next.



#### **Xóa cuối**

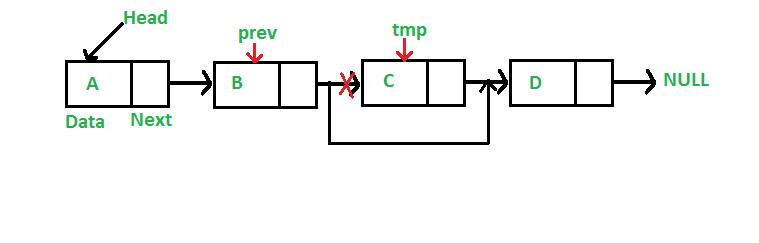
Xóa cuối mới nhọc nè, nhọc ở chỗ phải duyệt đến thằng cuối – 1, cho next của cuối – 1 đó bằng NULL.



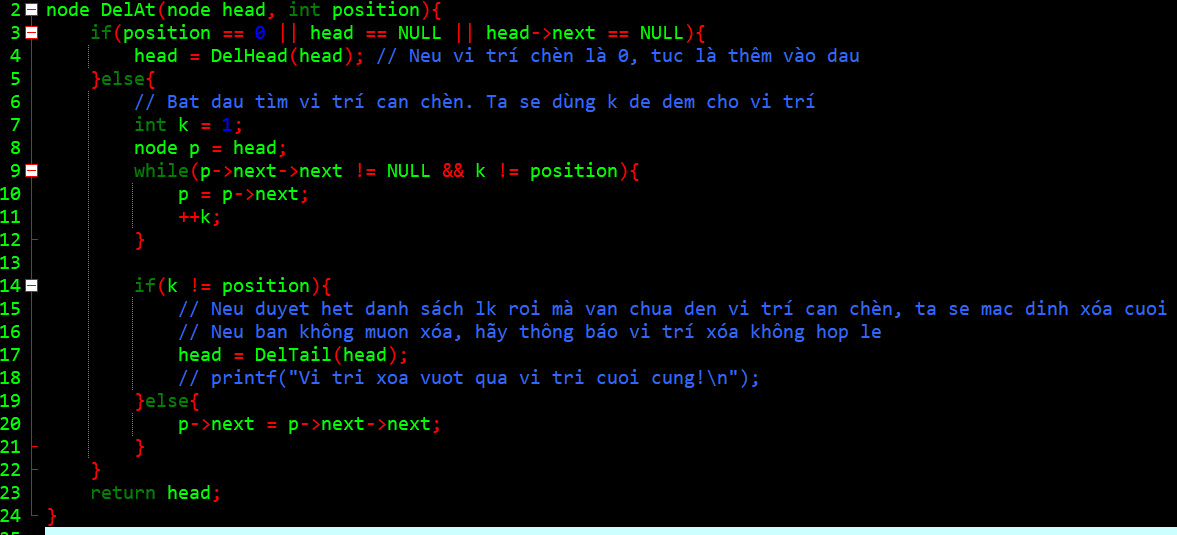
Thằng Node cuối – 1 là thằng có p->next->next = NULL. Bạn cho next của nó bằng NULL là xong.

#### **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:

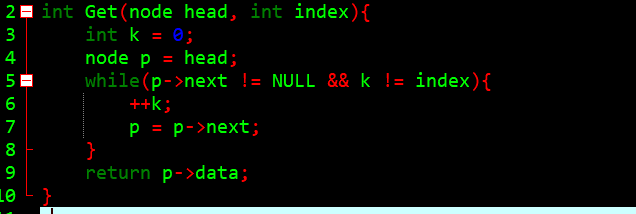


Lưu ý: Chỉ số xóa bắt đầu từ 0 nhé các bạn. Việc tìm vị trí càn xóa chỉ duyệt tới Node gần cuối thôi(cuối – 1). Sau đây là code xóa Node ở vị trí bất kỳ



### b.5. Lấy giá trị ở vị trí bất kỳ

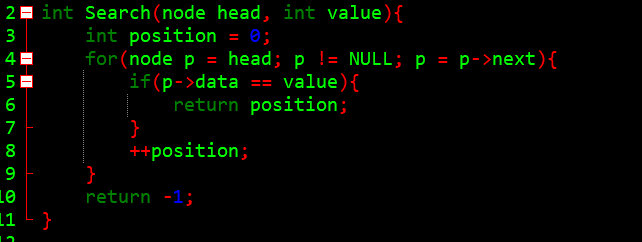
Chúng ta sẽ viết một hàm để truy xuất giá trị ở chỉ số bất kỳ nhé. Trong trường hợp chỉ số vượt quá chiều dài của linked list – 1, hàm này trả về vị trí cuối cùng. Do hạn chế là chúng ta không thể raise error khi chỉ số không hợp lệ. Tôi mặc định chỉ số bạn truyền vào phải là số nguyên không âm. Nếu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



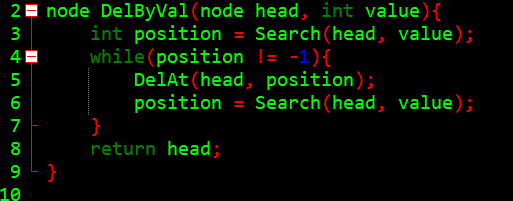
Lý do dùng p->next != NULL là vì chúng ta chỉ muốn đi qua các phần tử có value.

### b.6. 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.

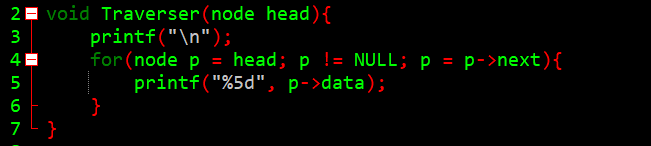


Chúng ta có thể sử dụng hàm này để xóa tất cả các Node trong danh sách liên kết có giá trị chỉ định như sau:



### \*) 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.

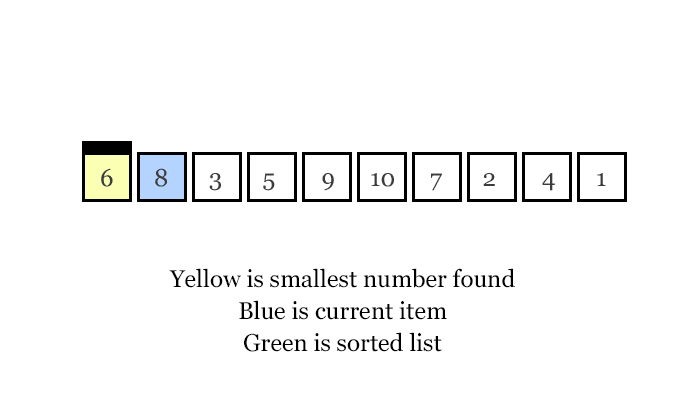


2.3 Các thuật toán sắp xếp

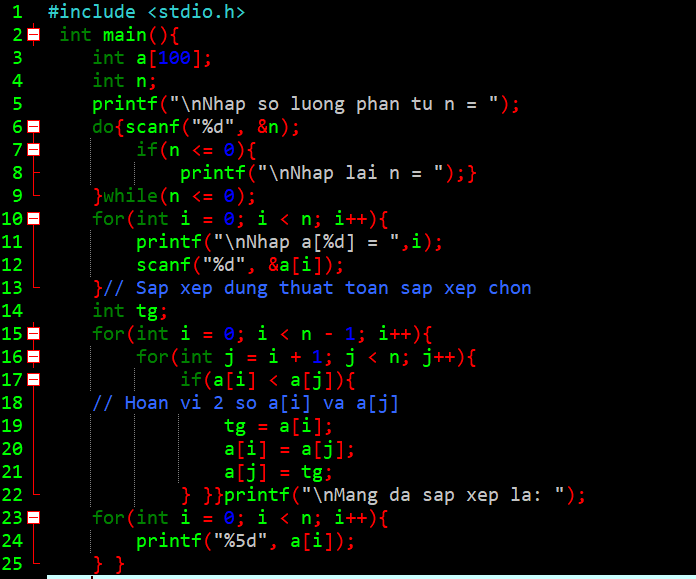
### a. Sắp xếp dãy số giảm dần

Trong code mà mình cung cấp dưới đây thì mình sẽ dùng thuật toán sắp xếp chọn – một thuật toán sắp xếp dễ hiểu và dễ cài đặt nhất.

Xem hình dưới đây để hiểu ý tưởng sắp xếp:

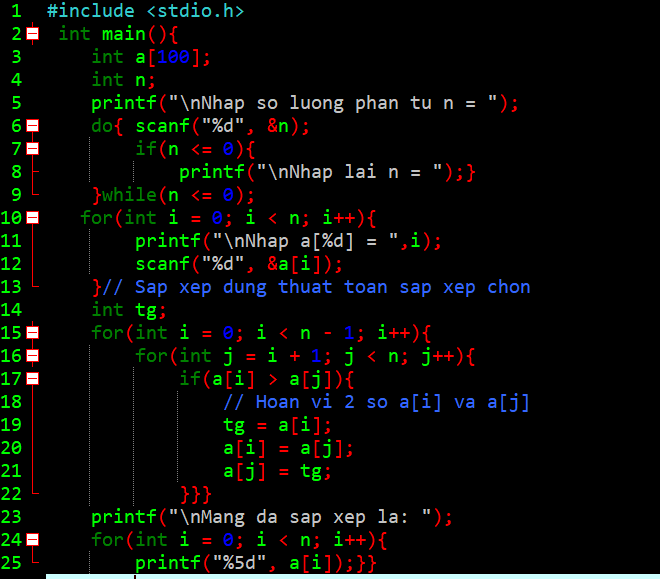


Code sắp xếp mảng/ dãy số giảm dần với C/C++:



### b. Sắp xếp dãy số tăng dần

Việc sắp xếp dãy số tăng dần chỉ khác sắp xếp giảm dần duy nhât ở bước kiểm tra điều kiện để hoán vị.



c. Thuật toán sắp xếp nổi bọt.

Thuật toán sắp xếp nổi bọt thực hiện sắp xếp dãy số bằng cách lặp lại công việc đổi chỗ 2 số liên tiếp nhau nếu chúng đứng sai thứ tự(số sau bé hơn số trước với trường hợp sắp xếp tăng dần) cho đến khi dãy số được sắp xếp.

## Ví dụ minh họa

Giả sử chúng ta cần sắp xếp dãy số [5 1 4 2 8] này tăng dần.  
**Lần lặp đầu tiên:**  
( **5** **1** 4 2 8 ) –> ( **1** **5** 4 2 8 ), Ở đây, thuật toán sẽ so sánh hai phần tử đầu tiên, và đổi chỗ cho nhau do 5 > 1.  
( 1 **5** **4** 2 8 ) –>  ( 1 **4** **5** 2 8 ), Đổi chỗ do 5 > 4  
( 1 4 **5** **2** 8 ) –>  ( 1 4 **2** **5** 8 ), Đổi chỗ do 5 > 2  
( 1 4 2 **5** **8** ) –> ( 1 4 2 **5** **8** ), Ở đây, hai phần tử đang xét đã đúng thứ tự (8 > 5), vậy ta không cần đổi chỗ.

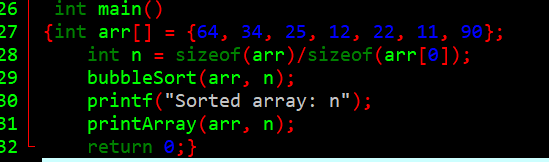
**Lần lặp thứ 2:**  
( **1** **4** 2 5 8 ) –> ( **1** **4** 2 5 8 )  
( 1 **4** **2** 5 8 ) –> ( 1 **2** **4** 5 8 ), Đổi chỗ do 4 > 2  
( 1 2 **4** **5** 8 ) –> ( 1 2 **4** **5** 8 )  
( 1 2 4 **5** **8** ) –>  ( 1 2 4 **5** **8** )  
Bây giờ, dãy số đã được sắp xếp, Nhưng thuật toán của chúng ta không nhận ra điều đó ngay được. Thuật toán sẽ cần thêm một lần lặp nữa để kết luận dãy đã sắp xếp khi và khi khi nó đi từ đầu tới cuối mà không có bất kỳ lần đổi chỗ nào được thực hiện.

**Lần lặp thứ 3:**  
( **1** **2** 4 5 8 ) –> ( **1** **2** 4 5 8 )  
( 1 **2** **4** 5 8 ) –> ( 1 **2** **4** 5 8 )  
( 1 2 **4** **5** 8 ) –> ( 1 2 **4** **5** 8 )  
( 1 2 4 **5** **8** ) –> ( 1 2 4 **5** **8** )

## Code sắp xếp nổi bọt trong C/C++

## 

Hàm chính như sau:



Ở đây, trong hàm bubbleSort tôi sử dụng thêm một biến haveSwap để kiểm tra tại lần lặp hiện hành có xảy ra việc đổi chỗ hai số không. Nếu không, ta có thể kết luận mảng đã sắp xếp mà không cần phải thêm một lần lặp nữa.

Kiểm tra kết quả:



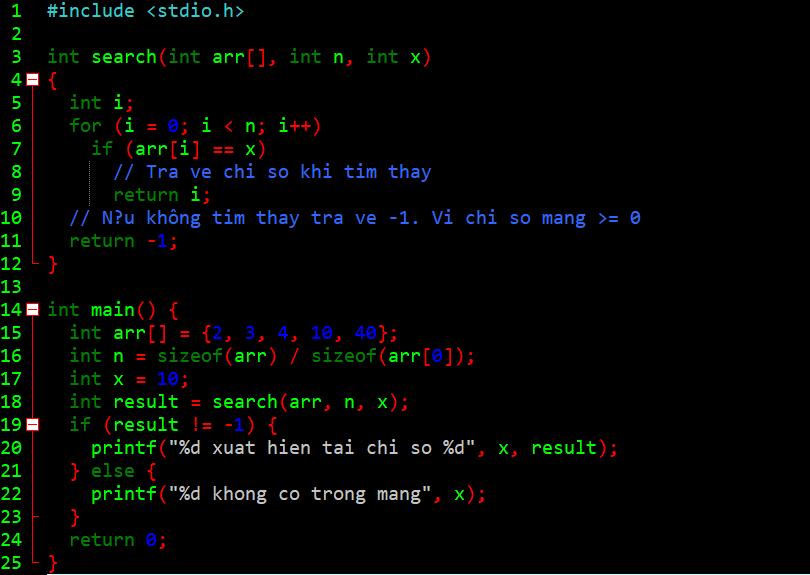
2.4 Các thuật toán tìm kiếm

a.Thuật toán tìm kiếm nhị phân

Cho một mảng đã sắp xếp arr[] có n phần tử, viết một hàm tìm kiếm trả về chỉ số của phần tử có giá trị x trong arr[].

Giải thuật đơn giản nhất cho bài toán này là sử dụng linear search(tìm kiếm tuyến tính). Tức là bạn sẽ phải đi qua từng phần tử của mảng để đối chiếu với x cần tìm. Thuật toán này trong trường hợp xấu nhất cho độ phức tạp là O(n). Mình cũng sẽ minh họa code của thuật toán này dưới đây.

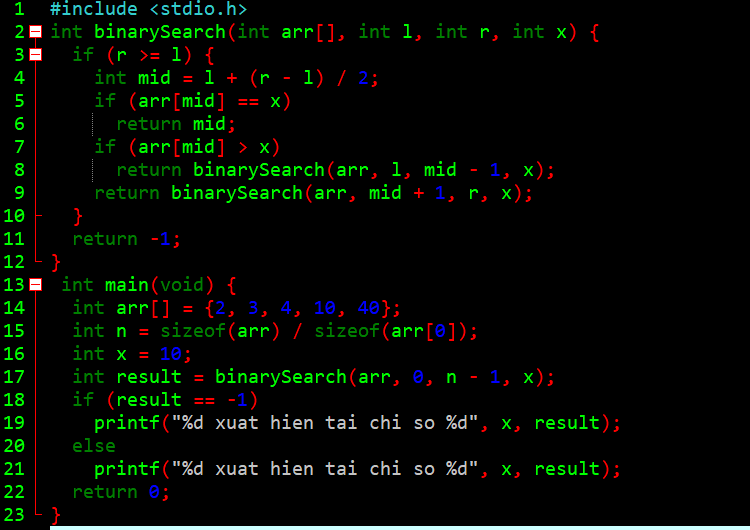
Đây là code C/C++ sử dụng linear search.



## Minh họa code cho thuật toán tìm kiếm nhị phân

Trong phần này, mình sẽ minh họa code sử dụng giải thuật đệ quy dùng Java và C/C++. Ngoài ra, tôi sẽ áp dụng thêm giải thuật khử đệ quy với C/C++.

Code minh họa C/C++ sử dụng đệ quy



**b.Thuật toán tìm kiếm tuyến tính**.

Giải thuật chính của tìm kiếm tuyến tính chính là: 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 thõa mãn thì cho kết quả là không tìm thấy.

**Giải thuật:**

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

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

**Ví dụ**: Cho dãy A gồm các phần tử: 11 4 3 9 8 0 2 45. Dùng giải thuật tìm kiếm tuyến tính để tìm xem có phần tử 8 nằm ở trong mảng hay không.

Bước 1: gán i=0.

Bước 2: so sánh a[0]= 11 != 8. Tăng i lên một đơn vị.

Bước 3: i=1 < n-1 (n = 8). Quay lại bước 2.

So sánh a[1] = 4 !=8. Tăng i lên một đơn vị .

Lặp lại Bước 3: i=2 < n-1 (n = 8). Quay lại bước 2.

So sánh a[2] = 3 !=8. Tăng i lên một đơn vị .

Lặp lại Bước 3: i=3 < n-1 (n = 8). Quay lại bước 2.

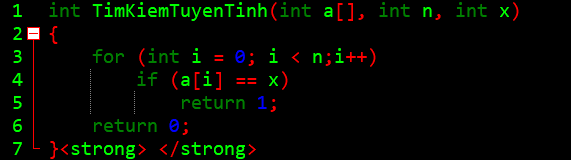
So sánh a[3] = 9 !=8. Tăng i lên một đơn vị .

Lặp lại Bước 3: i=4 < n-1 (n = 8). Quay lại bước 2.

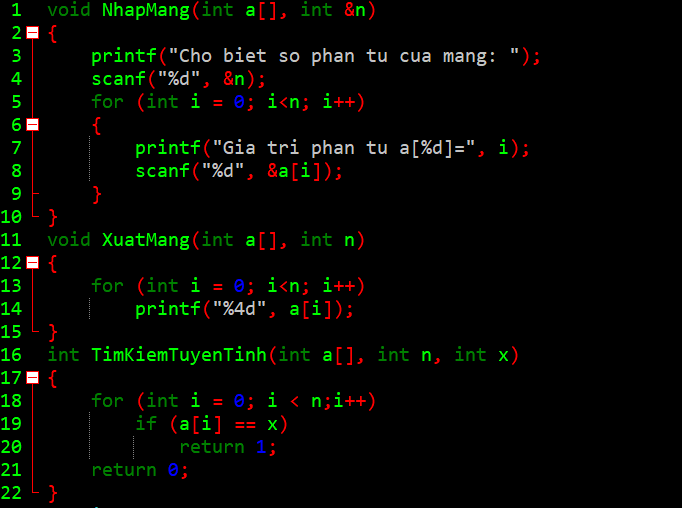
So sánh a[4] = 8 = 8. Nên kết thúc và tìm được x ở vị trí số a[4].

Ngoài việc tìm kiếm trong mảng số chúng ta còn có thể áp dụng giải thuật tìm kiếm cho mảng, danh sách với cấu trúc dưới dạng struct.

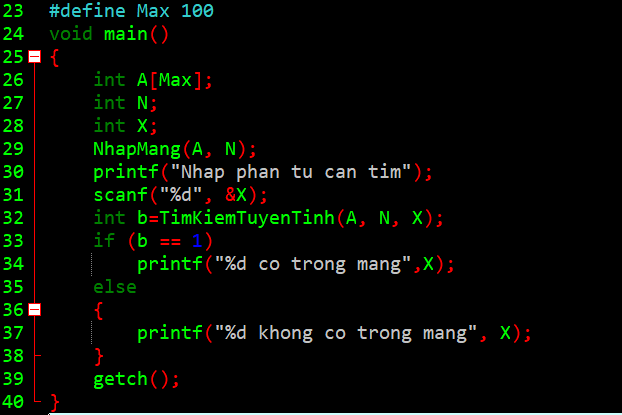
**Code tổng quát giải thuật:**



**Code ví dụ 1.**



Hàm main chính là:



Phần III: Demo code Đề tài quản lý sách thư viện

3.1 Các hàm trong chương trình

+ void nhapN(book\_st ds[], int n): là hàm dùng để nhập thông tin cuốn sách mượn của thư viện bao gồm có nhập tên sách, thể loại sách, giá tiền của sách.

+ void xuatN(book\_st ds[], int n): là hàm dùng để xuất ra danh sách mượn của thư viện gồm thông tin của cuốn sách thứ i nào đó, số thứ tự sách, tên sách, thể loại sách và giá của sách.

+ void hienthithongtinsach(struct book\_st bs): tạo ra danh sách các thông tin của sách theo cột.

+ void sapxeptheoten(struct book\_st \*ds, int n): sắp xếp các sách theo thứ tự chữ cái từ Z-A.

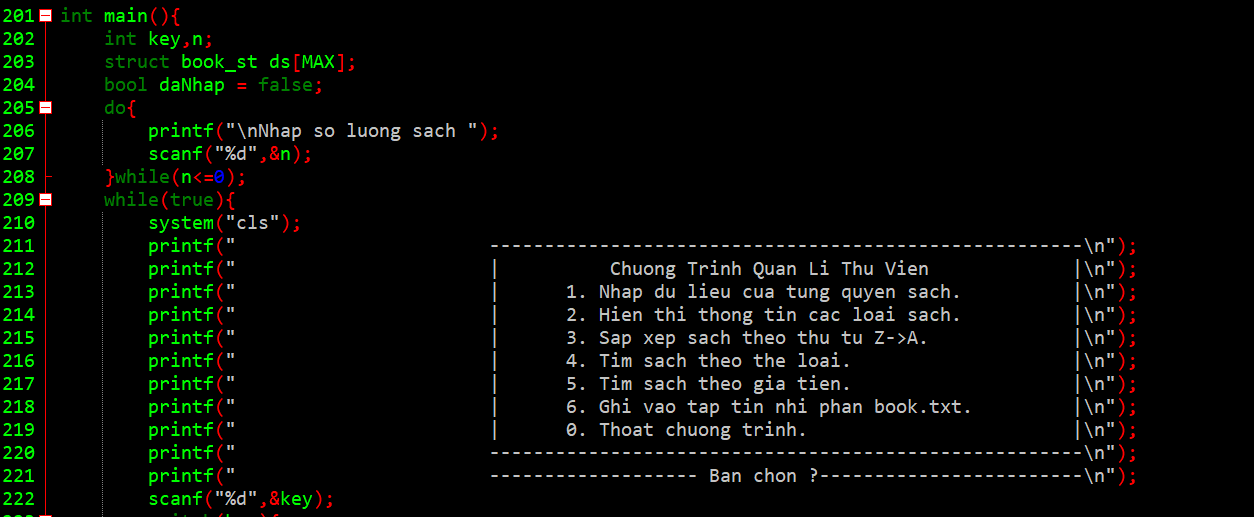
+ void timTheotheloai(struct book\_st \*ds, int n): tìm kiếm sách theo thể loại của sách.

+ void timtheogiatien(struct book\_st \*ds ,int n): ngoài tìm theo thể loại thì ta cũng sẽ tìm theo giá tiền.

+ void ghiFile(struct book\_st\* ds, int n): đây làm hàm dùng để đọc ghi file.

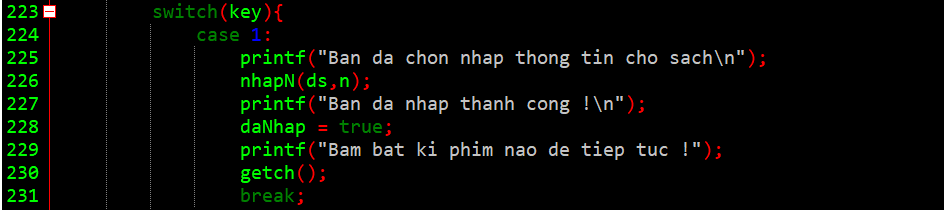
+ void docFile(struct book\_st\* ds, int\* n): là hàm để đọc file bằng book.txt.

+ Hàm main:

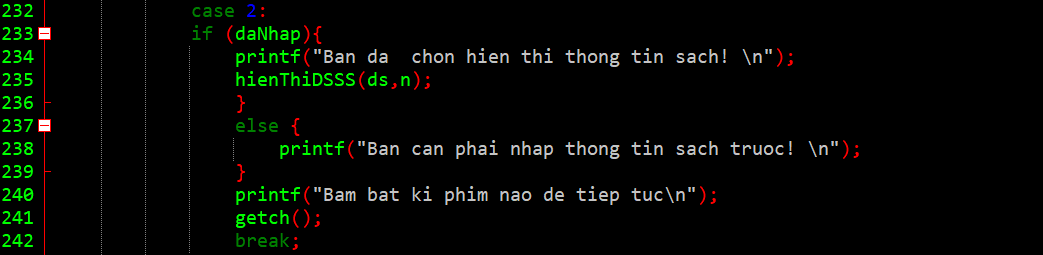


Tạo ra khung menu cho chương trình gồm có: Nhập dữ liệu, hiển thị thông tin các loại sách, sắp xếp tên theo thứ tự Z-A, tìm sách theo thể loại, tìm theo giá tiền, ghi vào tập tin nhị phân book.txt và phàn thoát chương trình.

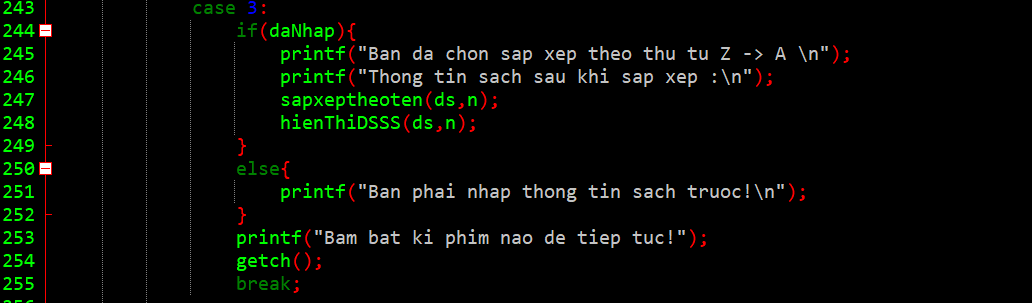
+ Case 1 dùng để nhập thông tin cho sách



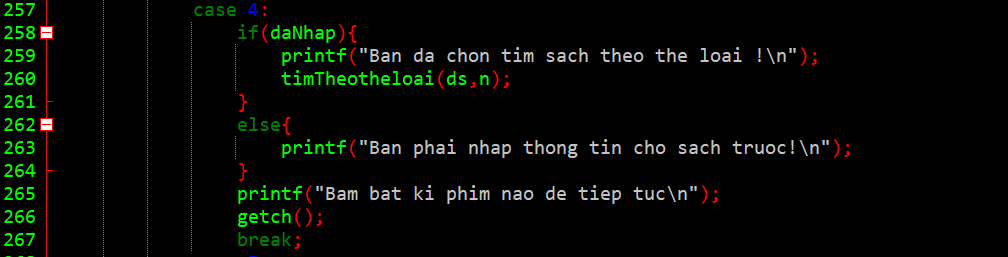
+Case 2 dùng để tạo ra mục để hiển thị thông tin cho sách



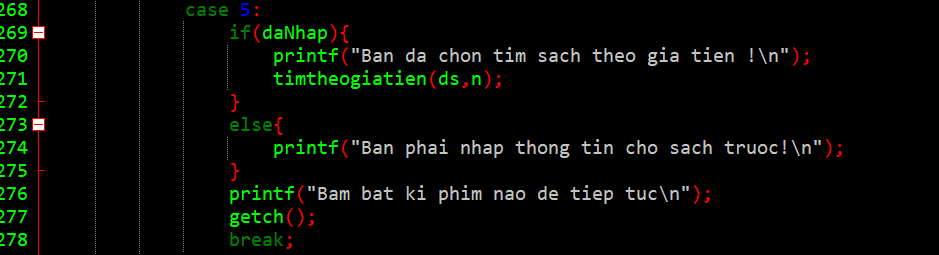
+Case 3 tạo ra mục sắp xếp sách theo thứ tự Z-A



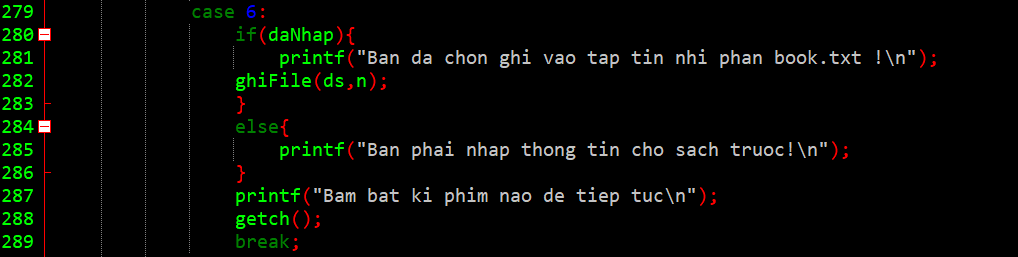
+Case 4 tạo ra mục tìm sách theo thể loại



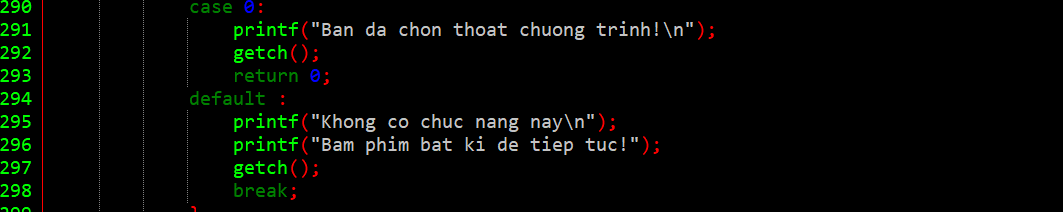
+ Case 5 tạo ra mục để chọn tìm sách theo giá tiền



+ Case 6 tạo ra mục ghi vạo tệp nhị phân book.txt



+ Case 0 tạo ra mục để bạn thoát khỏi chương trình



3.2 Hướng dẫn sử dụng chương trình

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

Ví dụ:

Nhap quyen sach 1:

Ten: Tham tu lung danh

The loai: Truyen tranh

Gia tien: 50000

* Thông tin được lưu các giá trị nhập từ người sử dụng vào mảng bookList trong hàm main().

Khi chọn 2 từ Menu, hiển thị thông tin sách sau đó hiển thị thông tin chi tiết của từng quyển sách theo cột.

Khi chọn 3 từ Menu, sắp xếp các quyển sách theo thể loại từ Z-> A sau đó hiển thị thông tin chi tiết của từng quyển sách như dưới đây:

STT||Ten ||The loai ||Gia tien

001||Tham tu lung danh||Truyen tranh ||50000

002||Than Dieu Dai Hiep ||Truyen kiem hiep ||30000

003||Anh Hung Xa Dieu ||Truyen kiem hiep ||35000

Khi chọn 4 từ Menu, thực hiện các yêu cầu dưới đây:

 Hiển thị “Nhap the loai: ”

 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 3. Trong trường hợp không tìm thấy quyển sách nào, hiển thị thông báo “Khong co quyen sach

thuoc the loai nay”.

Khi chọn 5 từ Menu, thực hiện các yêu cầu dưới đây:

 Hiển thị “Nhap gia tien: ”

 Tìm và hiển thị thông tin chi tiết của tất cả các quyển sách thuộc giá tiền này theo định dạng như trên câu hỏi 3. Trong trường hợp không tìm thấy quyển sách nào, hiển thị thông báo “Khong co quyen sach thuoc gia tien nay”.

Khi chọn 6 từ Menu, ghi tất cả thông tin của từng quyển sách vào file nhị phân “book.txt”.

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.txt”.

- Lưu trữ tất cả thông tin của từng quyển sách vào trong file này.

- Đóng file.

Khi chọn 0 từ Menu, chương trình kết thúc.

**Tài Liệu Tham Khảo**

<https://sinhvientot.net/giai-thuat-tim-kiem-tuyen-tinh/>

<https://nguyenvanhieu.vn/thuat-toan-tim-kiem-nhi-phan/>

<https://nguyenvanhieu.vn/thuat-toan-sap-xep-chon/>

<https://nguyenvanhieu.vn/thuat-toan-sap-xep-noi-bot/>

<https://nguyenvanhieu.vn/sap-xep-day-so-tang-dan/>

<https://nguyenvanhieu.vn/danh-sach-lien-ket-don/>

<https://nguyenvanhieu.vn/doc-ghi-file-trong-c/#cac-kieu-file>

<https://vietjack.com/bai-tap-c/sap-xep-struct-trong-c.jsp>