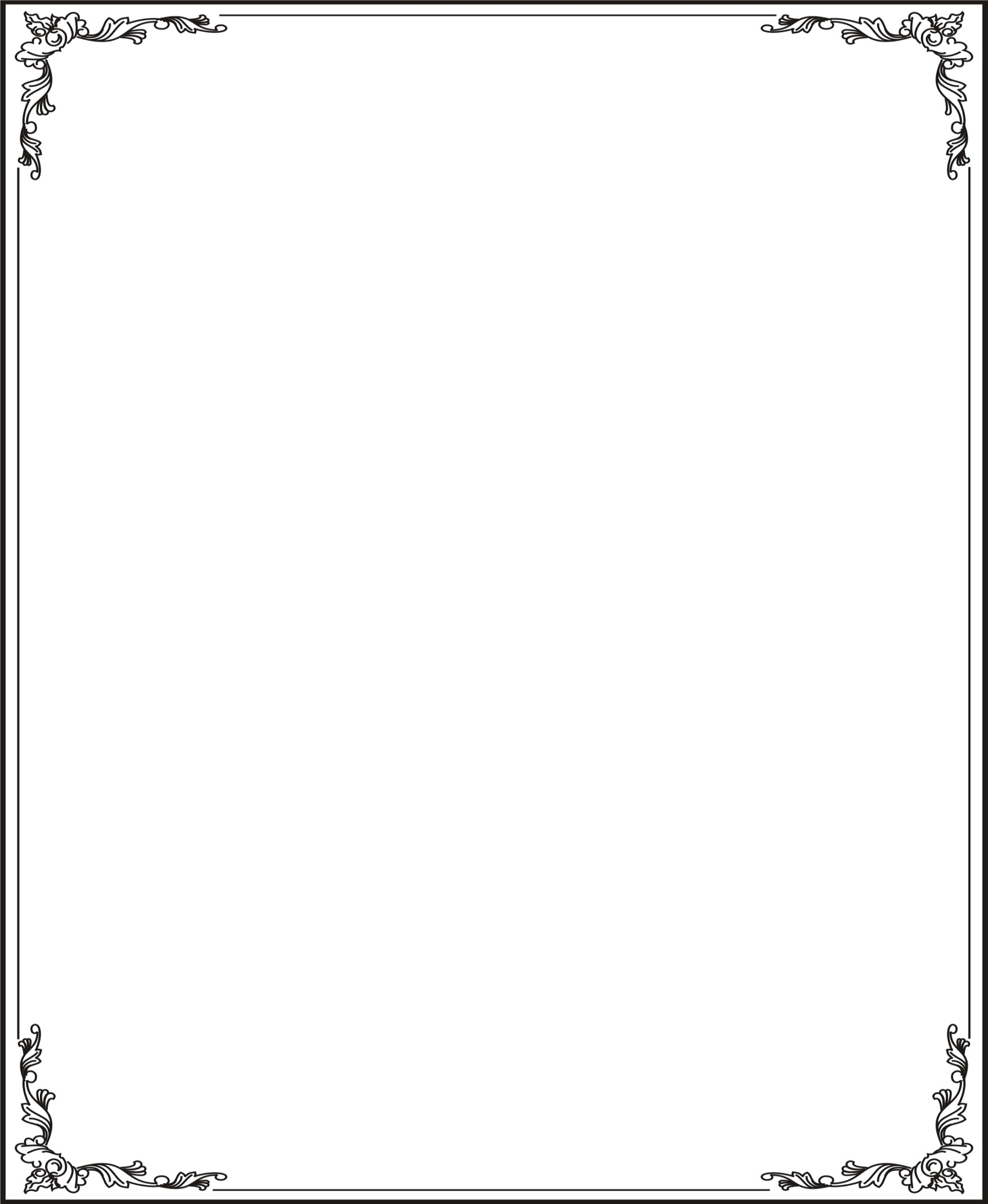
|  |
| --- |
| **TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**  **PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH** |





|  |
| --- |
| **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:** | **MSSV:** |
| Đặng Ngọc Chung | 6051071012 |
| Lê Âu hải | 6051071034 |
| Nguyễn Thành Đạt | 6051071028 |
| **Mã học phần:** | CPM215.3 |

MỤC LỤC

[Chương I: Sơ Lược Về Đề Tài 7](#_Toc42632521)

[1. Lí do chọn đề tài 7](#_Toc42632522)

[2. Mục đích đề tài 7](#_Toc42632523)

[Chương II: Cơ sở lý thuyết 8](#_Toc42632524)

[1. Làm việc với tệp 8](#_Toc42632525)

[1.1. File văn bản – Text files 8](#_Toc42632526)

[1.2. File nhị phân – Binary files 8](#_Toc42632527)

1.3. [Các thao tác với file 8](#_Toc42632528)

1.3.1. Thao tác với file trên ngôn ngữ C....................................................................6

1.3.2. [Thao tác mở file 9](#_Toc42632531)

1.3.3. [Thao tác đóng file 10](#_Toc42632532)

1.3.4. [Đọc/Ghi file văn bản trong C 10](#_Toc42632533)

1.3.5. [Đọc/Ghi file nhị phân trong C 11](#_Toc42632534)

1.3.6. [Một số ví dụ về đọc ghi file trong C 13](#_Toc42632535)

[2. Danh sách cấu trúc liên kết đơn: 15](#_Toc42632536)

[2.1. Danh sách liên kết là gì? 15](#_Toc42632537)

[2.2. Cài đặt danh sách liên kết đơn 17](#_Toc42632538)

[2.2.1. Khai báo linked list 17](#_Toc42632539)

[2.2.2. Tạo mới 1 Node 17](#_Toc42632540)

[2.2.3. Thêm Node vào danh sách liên kết 18](#_Toc42632541)

[2.2.4. Xóa Node khỏi danh sách liên kết 20](#_Toc42632542)

[2.2.5. Lấy giá trị ở vị trí bất kỳ 21](#_Toc42632543)

[2.2.6. Tìm kiếm trong danh sách liên kết 22](#_Toc42632544)

[2.2.7. Duyệt sách liên kết 23](#_Toc42632545)

[3. Các thuật toán sắp xếp 23](#_Toc42632546)

[3.1. Sắp xếp dãy số giảm dần 23](#_Toc42632547)

[3.2. Sắp xếp dãy số tăng dần 26](#_Toc42632548)

3.3. [Ví dụ minh họa 27](#_Toc42632549)

3.4. [Code sắp xếp nổi bọt trong C/C++ 27](#_Toc42632550)

4[. Các thuật toán tìm kiếm 29](#_Toc42632552)

4.1. [Minh họa code cho thuật toán tìm kiếm nhị phân 29](#_Toc42632553)

[Chương III: Demo code Đề tài quản lý sách thư viện 34](#_Toc42632554)

[1. Các hàm trong chương trình 34](#_Toc42632555)

2. Hướng dẫn sử dụng chương trình...............................................................................36

**Chương IV: Kết luận**.......................................................................................................38

MỤC LỤC HÌNH ẢNH

**Chương I: Sơ lược về đề tài**

**Chương II: Cơ sở lý thuyết**

Hình 1.3.1 Khai báo con trỏ FILE.......................................................................................9

Hình 1.3.2.1 Mở file bằng hàm fopen()...............................................................................9

Hình 1.3.2.2 Mode “w” and mode “rb”...............................................................................9

Hình 1.3.3 Đóng file bằng fclose()....................................................................................10

Hình 1.3.4.1 Ghi file sử dụng fprintf()……………………………………………………………………10

Hình 1.3.4.2 Đọc file sử dụng fscans()……………………………………………………………….…..11

Hình 1.3.5.1 Cấu trúc hàm ghi file nhị phân………………………………………………………….…11

Hình 1.3.5.2 Ghi file nhị phân sử dụng fwrite()………………………………………………………….12

Hình 1.3.5.3 Cấu trúc hàm đọc file nhị phân……………………………………………………….12

Hình 1.3.5.4 Đọc file nhị phân sử dụng fread()…………………………………………………….13

Hình 1.3.6.1 Ghi vào file một câu văn bản………………………………………………………………14

Hình 1.3.6.2 Đọc dữ liệu từ file……………………………………………………………………14

Hình 2.1.1 Mô tả một Node trong danh sách liên kết đơn…………………………………………..15

Hình 2.1.2 Danh sách liên kết đơn đầy đủ…………………………………………………………16

Hình 2.2.1 Khai báo Linked List………………………………………………………………………..17

Hình 2.2.2 Tạo mới một Node……………………………………………………………………17

Hình 2.2.3.1 Thêm Node vào đầu Linked List…………………………………………………….18

Hình 2.2.3.2 Thêm Node vào cuối Linked List…………………………………………………………18

Hình 2.2.3.3 Mô tả thuật toán newNode…………………………………………………………...……19

Hình 2.2.3.4 Thêm Node vào vị trí bất kì…………………………………………………………….…19

Hình 2.2.4.1 Xóa Node khỏi đầu danh sách………………………………………………………..20

Hình 2.2.4.2 Xóa Node khỏi cuối danh sách……………………………………………………….20

Hình 2.2.4.3 Mô tả thuật toán xóa Node ở vị trí bất kì………………………………………………21

Hình 2.2.4.4 Xóa Node ở vị trí bất kỳ………………………………………………………………21

Hinh 2.2.5 Lấy giá trị ở vị trí bất kỳ………………………………………………………………22

Hình 2.2.6.1 Tìm kiếm trong danh sách liên kết……………………………………………………22

Hình 2.2.6.2 Xóa tất cả các Node trong danh sách liên kết………………………………………….23

Hình 2.2.7 Duyệt danh sách liên kết…………………………………………………………….…23

Hình 3.1.1 Mô tả thuật toán sắp xếp…………………………………………………………….…24

Hình 3.1.2 Mảng giảm dần………………………………………………………………………..25

Hình 3.2 Mảng tăng dần…………………………………………………………………………..26

Hình 3.3.1 Sắp xếp nổi bọt……………………………………………………………………………...28

Hình 3.3.2 Hàm Main……………………………………………………………………………28

Hình 3.3.3 Kết quả………………………………………………………………………………………28

Hình 4.1.1 Tìm kiếm nhị phân…………………………………………………………………….……29

Hình 4.1.2 Tìm kiếm nhị phân với đệ quy…………………………………………………..……30

Hình 4.2.1 Code tổng quát giải thuật……………………………………………………………32

Hình 4.2.2 Code ví dụ………………………………………………………………………….32

Hình 4.2.3 Code main chính……………………………………………………………………33

**Chương III: Demo code đề tài quản lý sách thư viện**

Hình 1.1 Hàm main chính trong chương trình quản lý sách……………………………………………34

Hình 1.2 Case 1…………………………………………………………………………………….….35

Hình 1.3 Case 2……………………………………………………………………………………….35

Hình 1.4 Case 3…………………………………………………………………………………….…35

Hình 1.5 Case 4……………………………………………………………………………………..…36

Hình 1.6 Case 5……………………………………………………………………………………….36

Hình 1.7 Case 6……………………………………………………………………………………….36

Hình 1.8 Case 0……………………………………………………………………………………….36

**Chương IV: Kết luận**..........................................................................................................................38

**Chương I: Sơ Lược Về Đề Tài**

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 bao gồm 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.

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 II: Làm Việc Với Tệp

1. Làm việc với tệp:

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

* **File văn bản – text files**
* **File nhị phân – binary file**

1.1. 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.

1.2. 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.

1.3. Các thao tác với file:

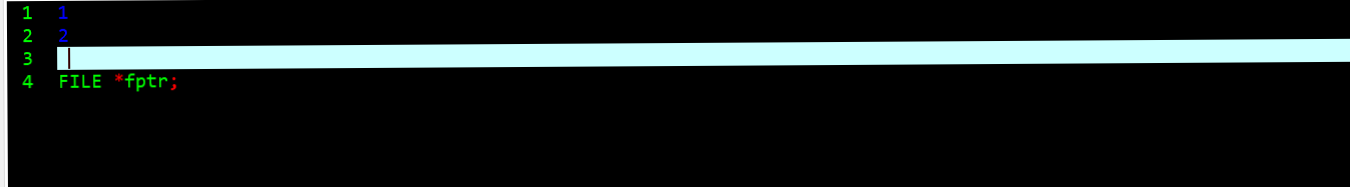
Trong ngôn ngữ lập trình C, có một số thao tác chính khi làm việc với file, bao gồm cả file văn bản và file nhị phân:

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

## 

## 1.3.1. 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 ta và tập tin mà ta cần thao tác.

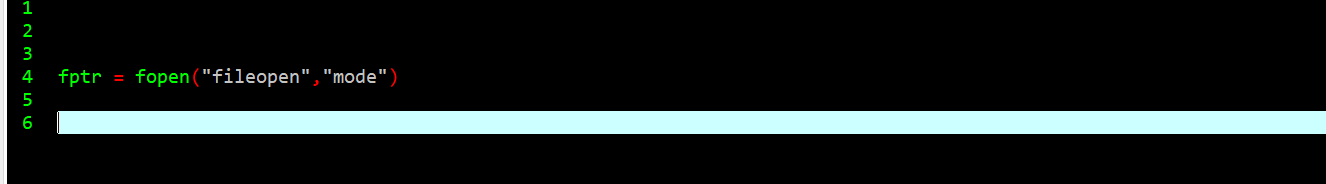


Hình 1.3.1 Khai báo con trỏ FILE

### 1.3.2. Thao tác mở file:

Để đọc ghi file trong C cũng như trong mọi ngôn ngữ lập trình, việc đầu tiên ta cần làm là mở file mà ta 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

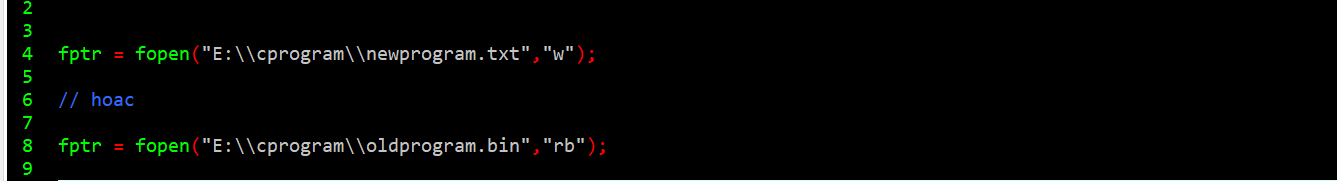
fptr = fopen("fileopen","mode")



Hình 1.3.2.1 Mở file bằng hàm fopen()

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

Ví dụ:



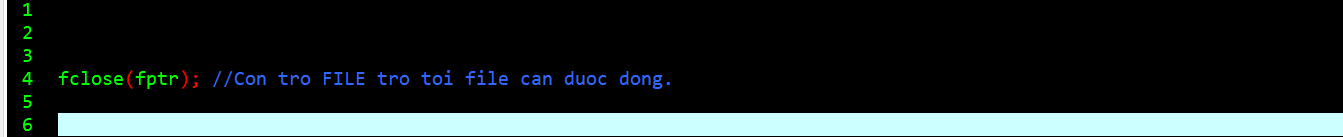
Hình 1.3.2.2 Mode “w” and mode “rb”

* Giả sử tập tin newprogram.txt chưa có trong thư mục E:\cprogram. Ví dụ đầu tiên với mode là 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.

### 1.3.3. 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().

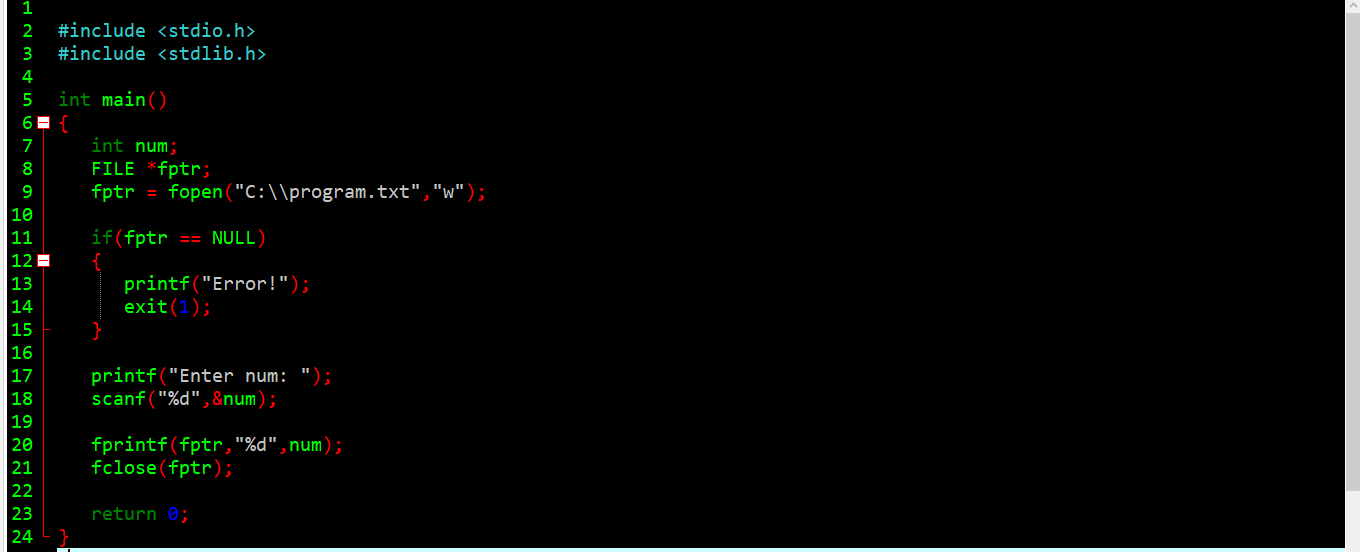


Hình 1.3.3 Đóng file bằng fclose()

## 1.3.4. Đọc/Ghi file văn bản trong C

Chúng ta sẽ học cách Đọc/Ghi file trong C với file văn bản trước.

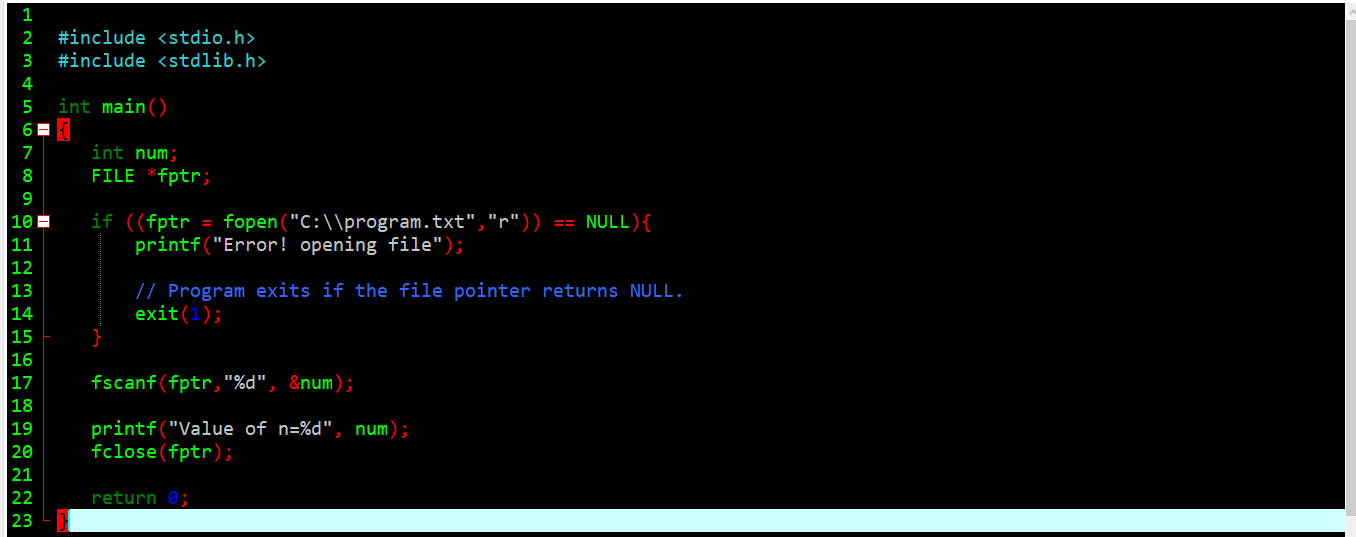
Để làm việc với file văn bản, chúng ta sẽ sử dụng fprintf() và fscanf().



Hình 1.3.4.1 Ghi file sử dụng fprintf()

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

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



Hình 1.3.4.2 Đọc file sử dụng fscans()

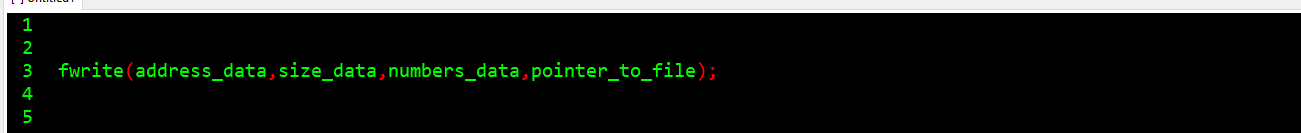
Chương trình này sẽ đọc giá trị số được lưu trong file program.txt mà chương trình vừa tạo ra và in lên màn hình.

## 1.3.5. Đọc/Ghi file nhị phân trong C

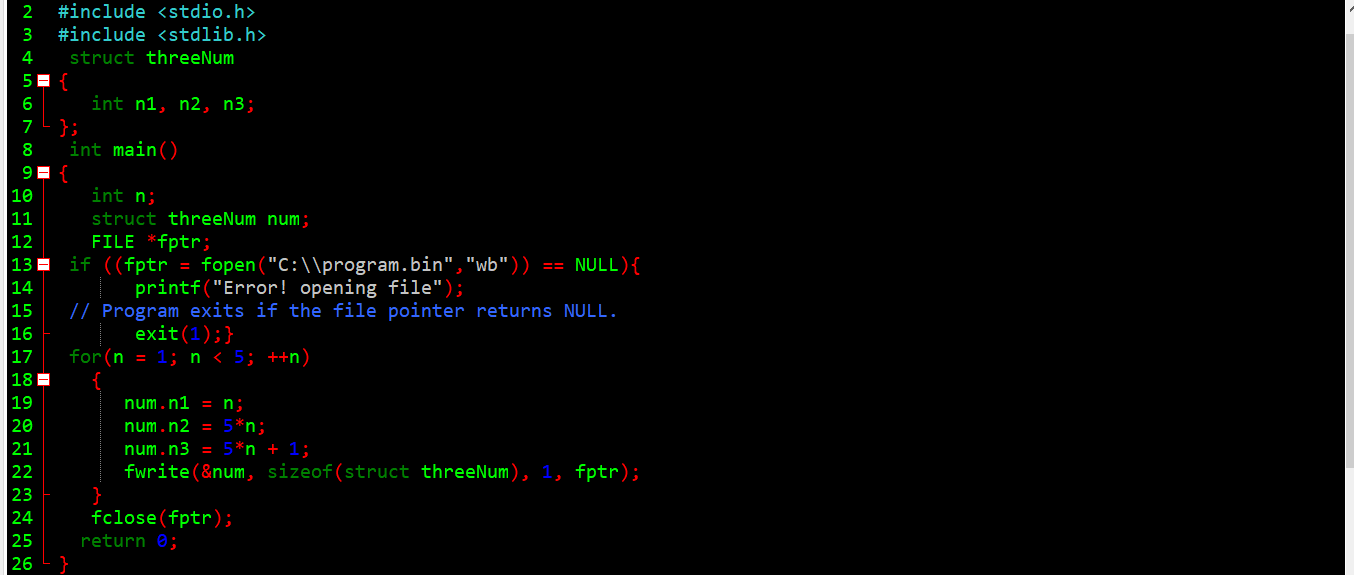
Các hàm fread() và fwrite() trong C được sử dụng để đọc và ghi file trong C ở dạng nhị phân.

#### **Ghi file nhị phân**

Để ghi file nhị phân, ta 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.



Hình 1.3.5.1 Cấu trúc hàm ghi file nhị phân



Hình 1.3.5.2 Ghi file nhị phân sử dụng fwrite()

Chương trình sẽ tạo ra một file program.bin trên ổ đĩa C. 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.

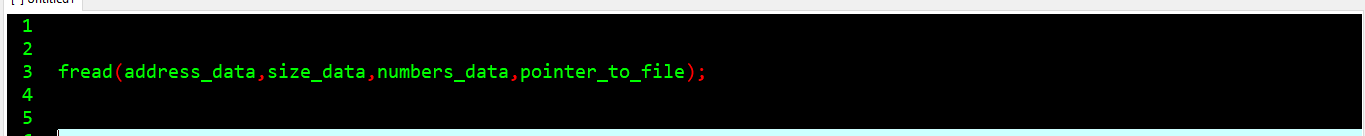
Trong vòng lặp, các số được ghi vào file sử dụng hàm fwrite(). Các tham số gồm:

* 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().

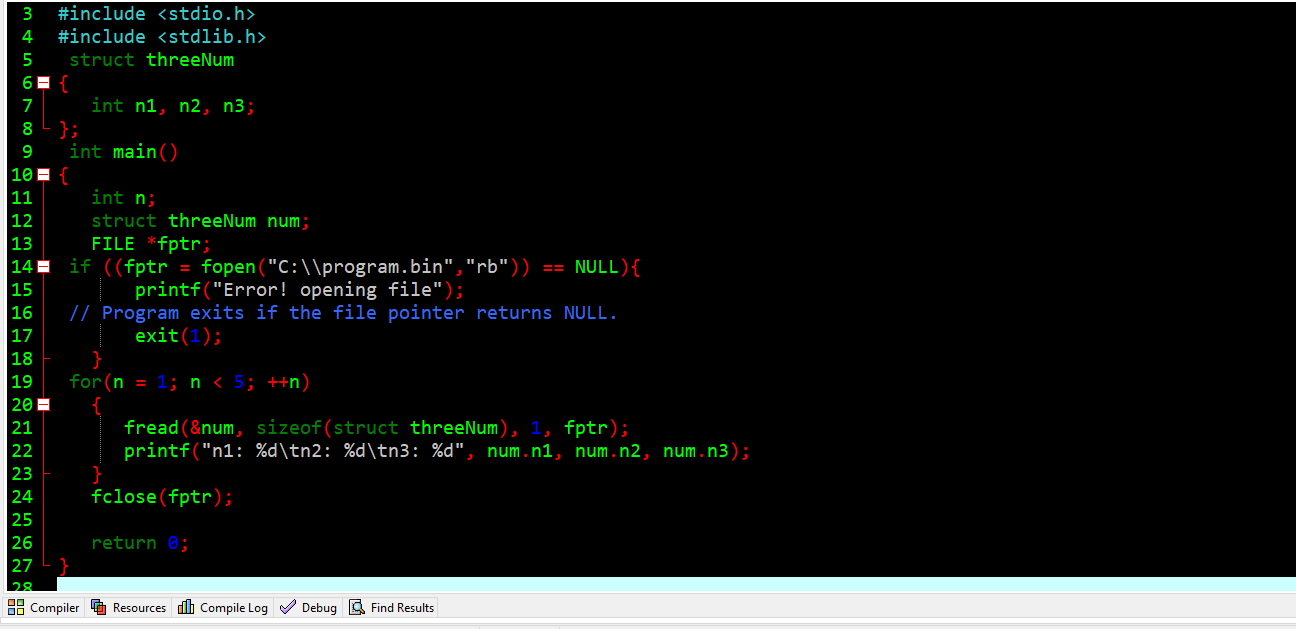
#### **Đọc file nhị phân**

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



Hình 1.3.5.3 Cấu trúc hàm đọc file nhị phân

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



Hình 1.3.5.4 Đọc file nhị phân sử dụng fread()

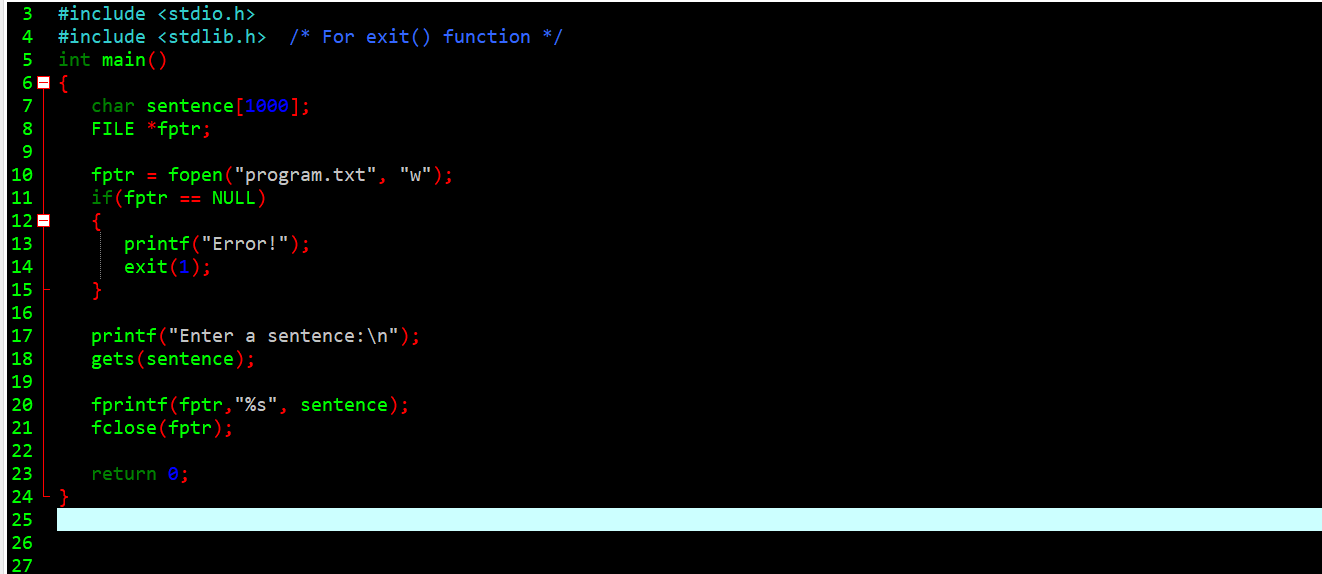
Trong ví dụ này, ta đọc file program.bin và lặp qua từng dòng. Ta sẽ nhận được các giá trị tương ứng khi bạn ghi vào trong ở ví dụ trên.

## 1.3.6. Một số ví dụ về đọc ghi file trong C

Trong phần này, em 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**

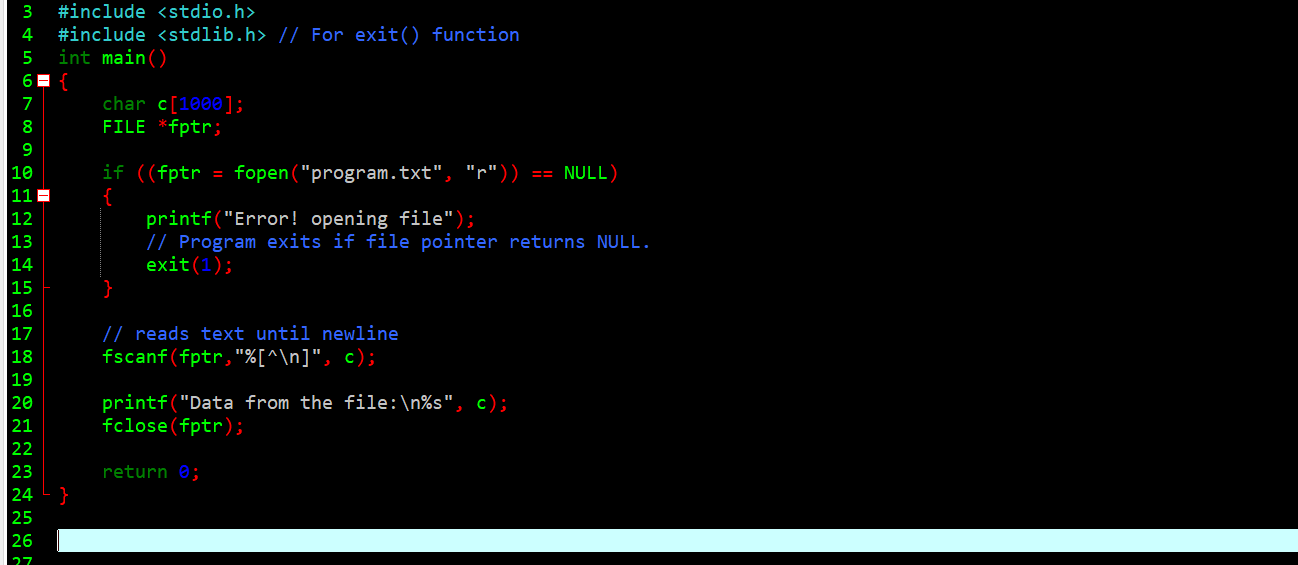


Hình 1.3.6.1 Ghi vào file một câu văn bản

Chạy thử:

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

#### **Đọc dữ liệu văn bản từ file**



Hình 1.3.6.2 Đọc dữ liệu từ file

2. Danh sách cấu trúc liên kết đơn:

## 2.1. 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:



Hình 2.1.1 Mô tả một Node trong danh sách liên kết đơn

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



Hình 2.1.2 Danh sách liên kết đơn đầy đủ

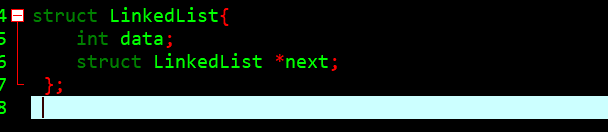
Danh sách các kiểu danh sách liên kết:

* 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.

2.2. Cài đặt danh sách liên kết đơn

2.2.1. Khai báo linked list

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



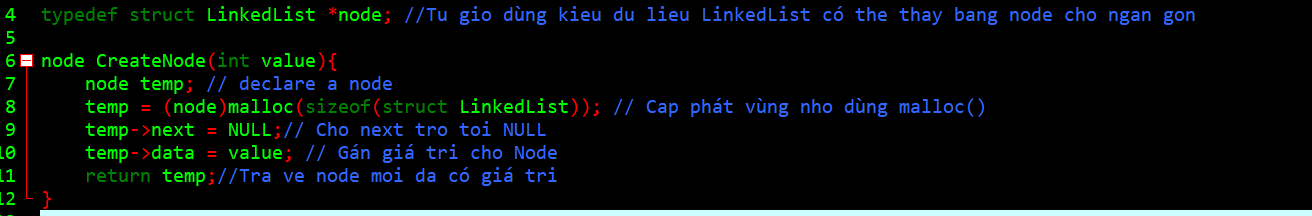
Hình 2.2.1 Khai báo Linked List

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 vị trí 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 vị trí Node kế tiếp cũng có kiểu LinkedList.

### 2.2.2. Tạo mới một Node

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



Hình 2.2.2 Tạo mới một Node

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

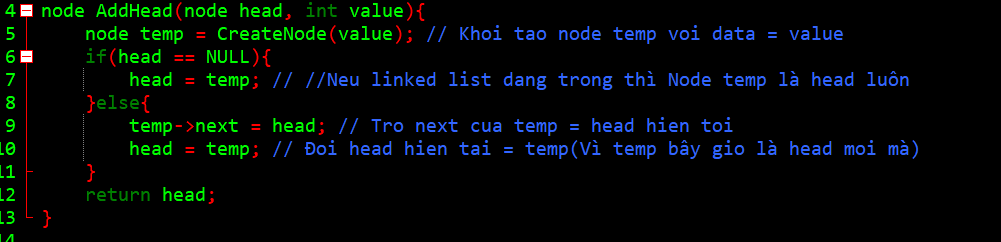
**Lưu ý:** Không giống với mảng, cần khai báo arr[size]. Trong linked list, vì mỗi Node sẽ có con trỏ liên kết đến Node tiếp theo. Do đó, với danh sách liên kết đơn, ta chỉ cần lưu giữ Node đầu tiên(HEAD). Có head rồi ta có thể đi tới bất cứ Node nào.

### 2.2.3. 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

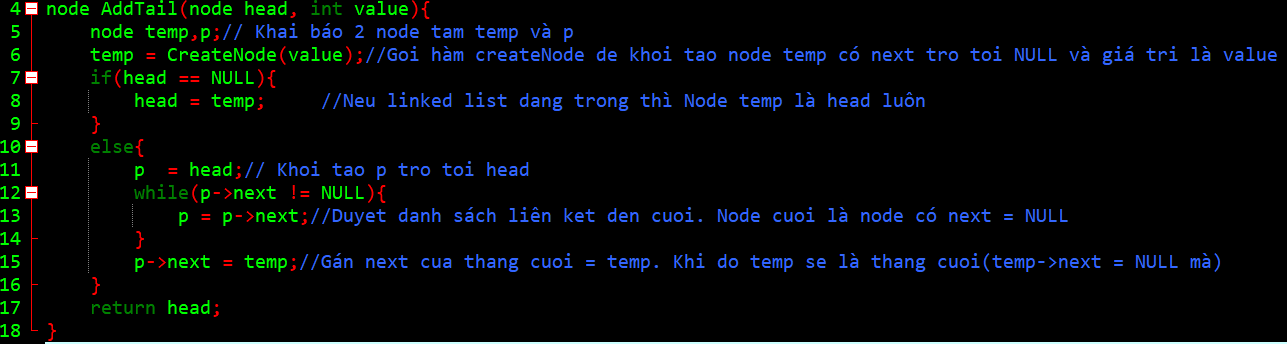


Hình 2.2.3.1 Thêm Node vào đầu Linked List

#### **Thêm vào cuối**

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

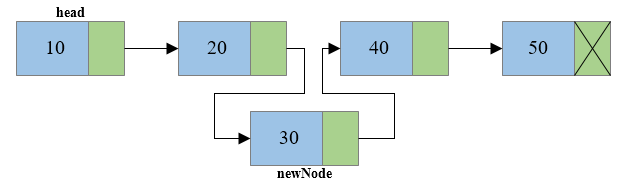
* + Tạo một Node mới với giá trị value
  + Nếu head = NULL, tức là danh sách liên kết đang trống. Khi đó Node mới(temp) sẽ là head luôn.
  + Ngược lại, ta sẽ duyệt tới Node cuối cùng(Node có next = NULL), và trỏ next của vị trí cuối tới Node mới(temp).



Hình 2.2.3.2 Thêm Node vào cuối Linked List

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

#### **Thêm vào vị trí bất kỳ**

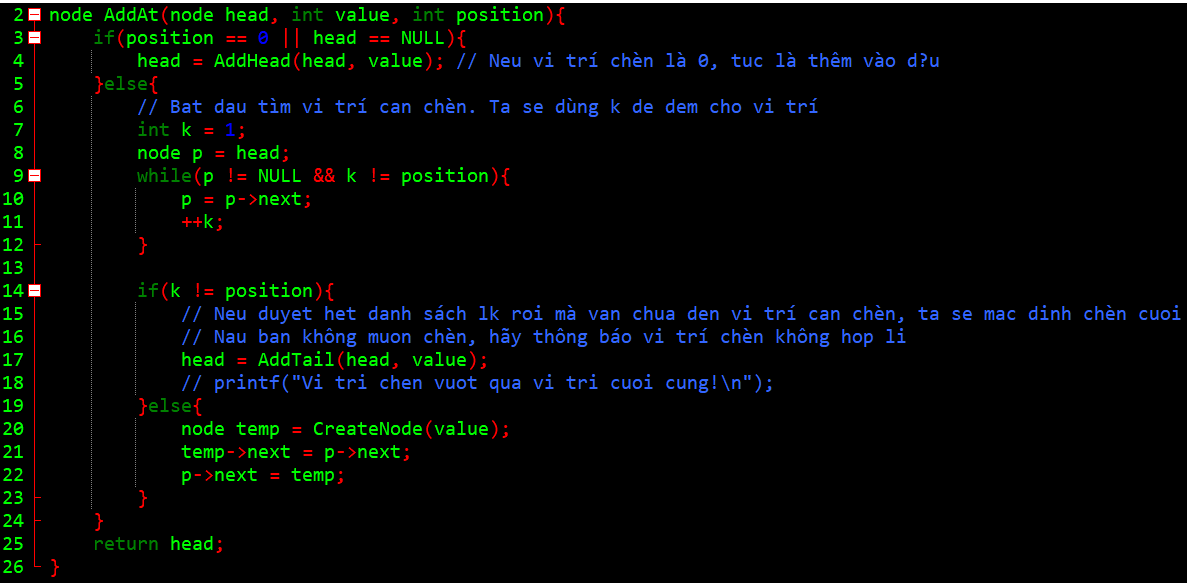


Hình 2.2.3.3 Mô tả thuật toán newNode

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



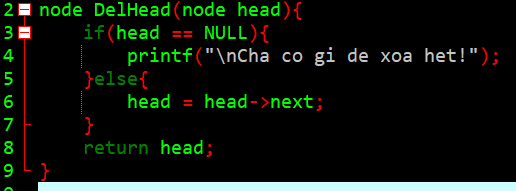
Hình 2.2.3.4 Thêm Node vào vị trí bất kì

Lưu ý: Phải làm theo thứ tự trên, nếu cho p->next = temp trước. Khi đó 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).

**2.2.4. Xóa Node khỏi danh sách liên kết**

#### **Xóa đầu**

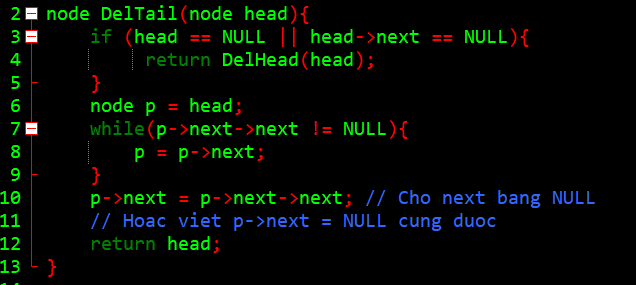
Xóa đầu đơn giản rất đơn giản, bây giờ chỉ cần cho vị trí kế tiếp của head làm head. Mà vị trí kế tiếp của head chính là head->next.



Hình 2.2.4.1 Xóa Node khỏi đầu danh sách

**Xóa cuối**

Xóa cuối sẽ khó hơn, khó ở chỗ phải duyệt đến vị trí cuối – 1, cho next của cuối – 1 đó bằng NULL.

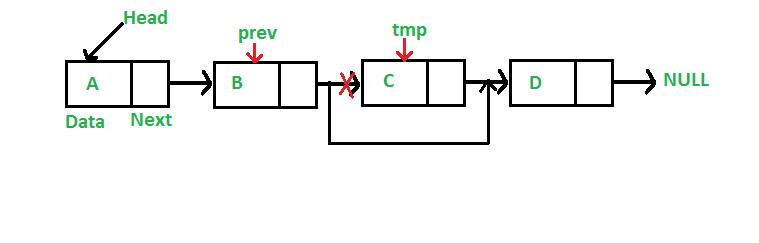


Hình 2.2.4.2 Xóa Node khỏi cuối danh sách

Vị trí Node cuối – 1 là vị trí có p->next->next = NULL. Ta cho next của nó bằng NULL.

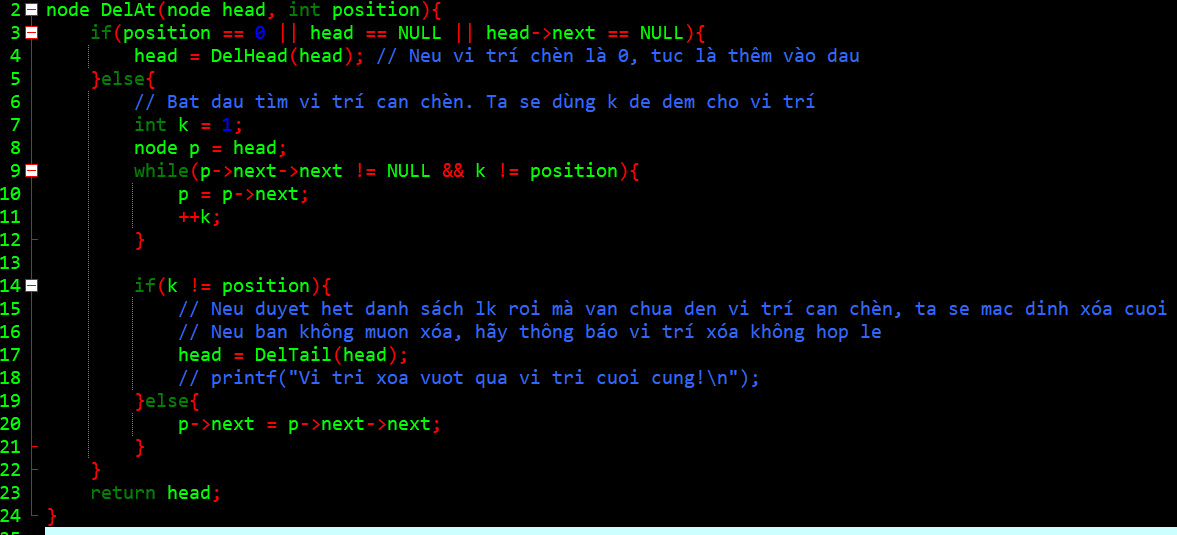
#### **Xóa ở vị trí bất kỳ**

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



Hình 2.2.4.3 Mô tả thuật toán xóa Node ở vị trí bất kì

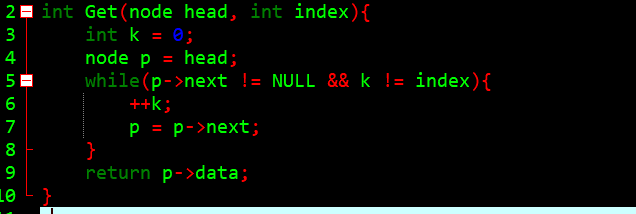
Lưu ý: Chỉ số xóa bắt đầu từ 0. Việc tìm vị trí càn xóa chỉ duyệt tới Node gần cuối thôi(cuối – 1). Sau đây là code xóa Node ở vị trí bất kỳ



Hình 2.2.4.4 Xóa Node ở vị trí bất kỳ

### 2.2.5. Lấy giá trị ở vị trí bất kỳ

Chúng ta sẽ viết một hàm để truy xuất giá trị ở chỉ số bất kỳ. 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. Em mặc định chỉ số ta truyền vào phải là số nguyên không âm. Nếu ta muốn kiểm tra chỉ số hợp lệ thì nên kiểm tra trước khi gọi hàm này.

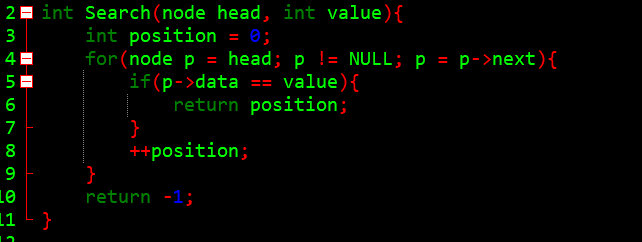


Hinh 2.2.5 Lấy giá trị ở vị trí bất kỳ

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

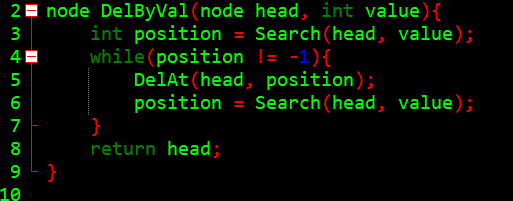
### 2.2.6. Tìm kiếm trong danh sách liên kết

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.



Hình 2.2.6.1 Tìm kiếm trong danh sách liên kết

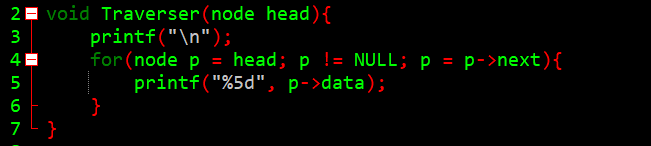
Chúng ta có thể sử dụng hàm này để xóa tất cả các Node trong danh sách liên kết có giá trị chỉ định như sau:



Hình 2.2.6.2 Xóa tất cả các Node trong danh sách liên kết

### 2.2.7. Duyệt danh sách liên kết

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



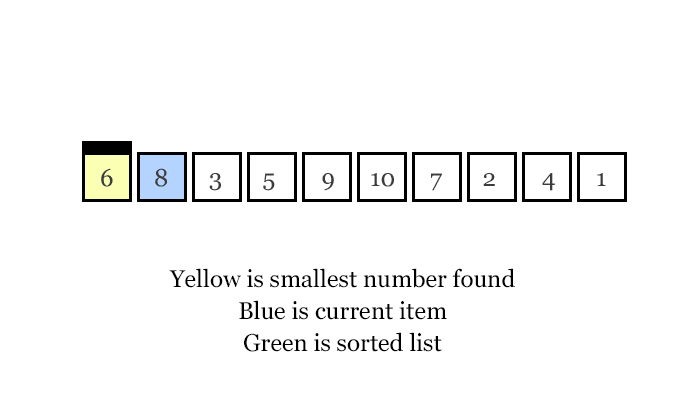
Hình 2.2.7 Duyệt danh sách liên kết

3 Các thuật toán sắp xếp

### 3.1. Sắp xếp dãy số giảm dần

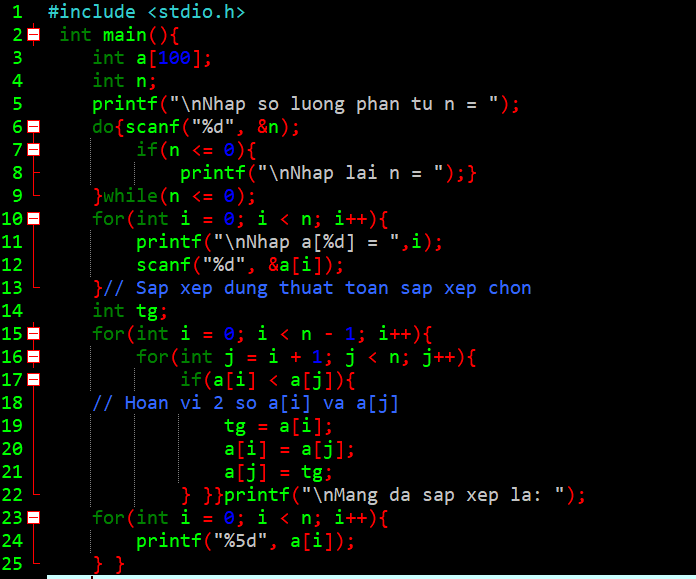
Trong code mà em cung cấp dưới đây thì sẽ dùng thuật toán sắp xếp chọn một thuật toán sắp xếp dễ hiểu và dễ cài đặt nhất.

Xem hình dưới đây để hiểu ý tưởng sắp xếp:



Hình 3.1.1 Mô tả thuật toán sắp xếp

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

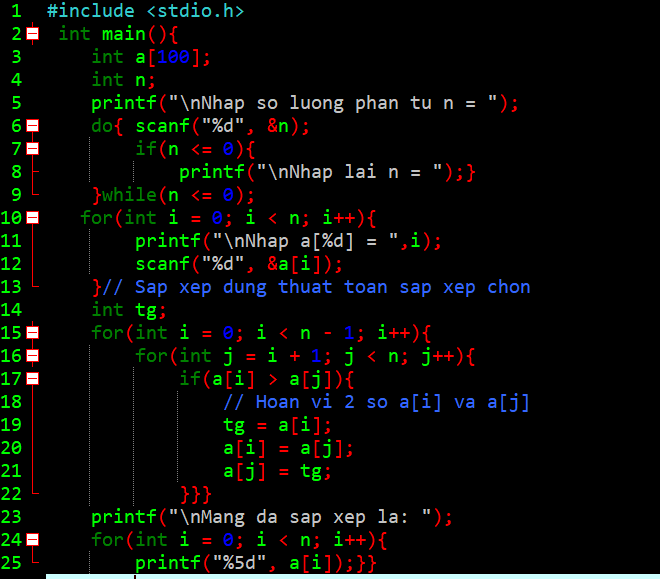


Hình 3.1.2 Mảng giảm dần

### 

### 3.2. 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.



Hình 3.2 Mảng tăng dần

3.3. 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.

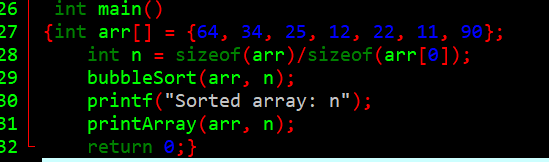
**Lần lặp thứ 2:**  
( **1** **4** 2 5 8 ) –> ( **1** **4** 2 5 8 )  
( 1 **4** **2** 5 8 ) –> ( 1 **2** **4** 5 8 ), Đổi chỗ do 4 > 2  
( 1 2 **4** **5** 8 ) –> ( 1 2 **4** **5** 8 )  
( 1 2 4 **5** **8** ) –>  ( 1 2 4 **5** **8** )  
Bây giờ, dãy số đã được sắp xếp, Nhưng thuật toán của chúng ta không 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++:

## Hình 3.3.1 Sắp xếp nổi bọt

Hàm chính như sau:



Hình 3.3.2 Hàm Main

Ở đây, trong hàm bubbleSort em 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:



Hình 3.3.3 Kết quả

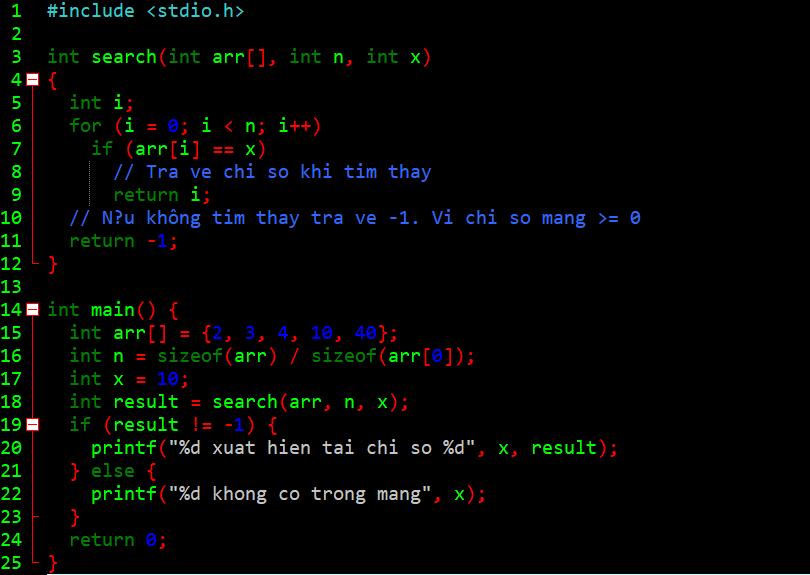
4 Các thuật toán tìm kiếm

4.1. 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 làbta 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). Em cũng sẽ minh họa code của thuật toán này dưới đây.

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

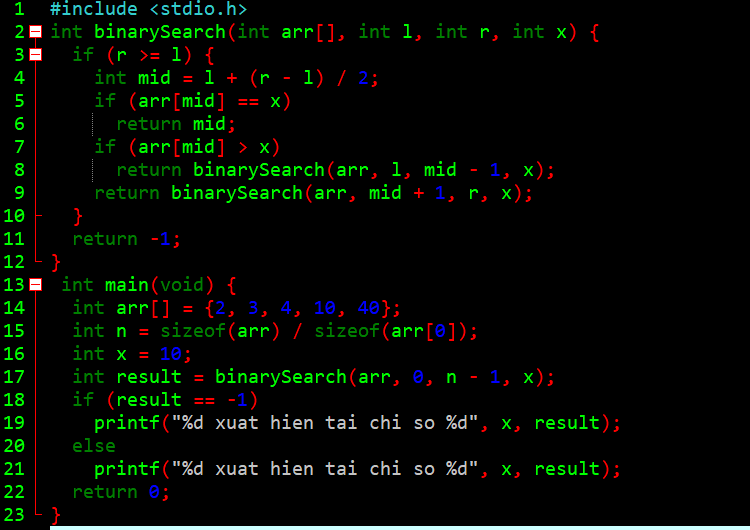


Hình 4.1.1 Tìm kiếm nhị phân

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

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

Code minh họa C/C++ sử dụng đệ quy



Hình 4.1.2 Tìm kiếm nhị phân với đệ quy

**4.2. 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 .

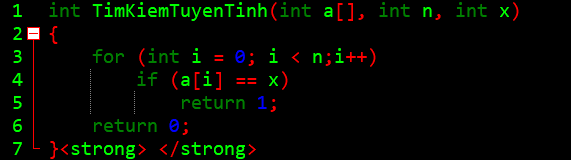
Lặp lại Bước 3: i=3 < n-1 (n = 8). Quay lại bước 2.

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

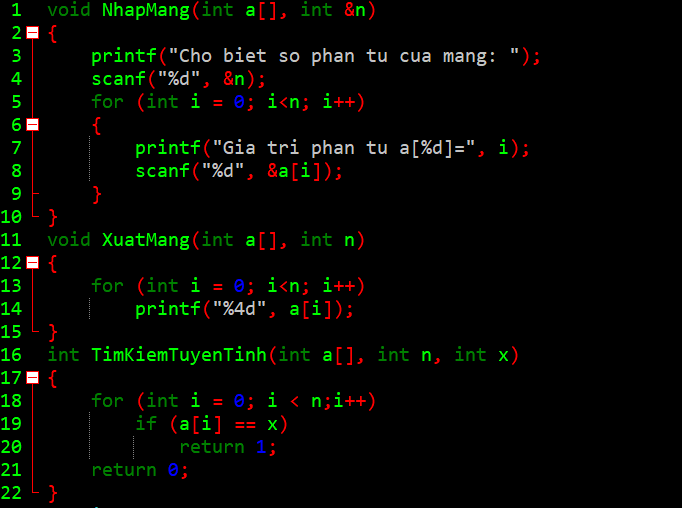
Lặp lại Bước 3: i=4 < n-1 (n = 8). Quay lại bước 2.

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.

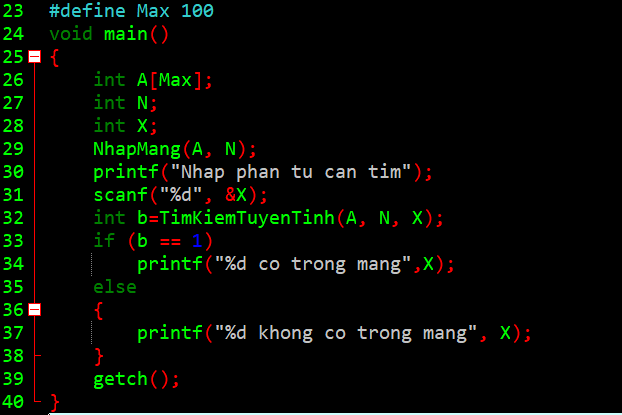


Hình 4.2.1 Code tổng quát giải thuật



Hình 4.2.2 Code ví dụ

Hàm main chính là:



Hình 4.2.3 Code main chính

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

1. Các hàm trong chương trình

+ void nhapN(book\_st book\_List[], 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 book\_List [], 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): hiển thị thông tin chi tiết của từng quyển sách.

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

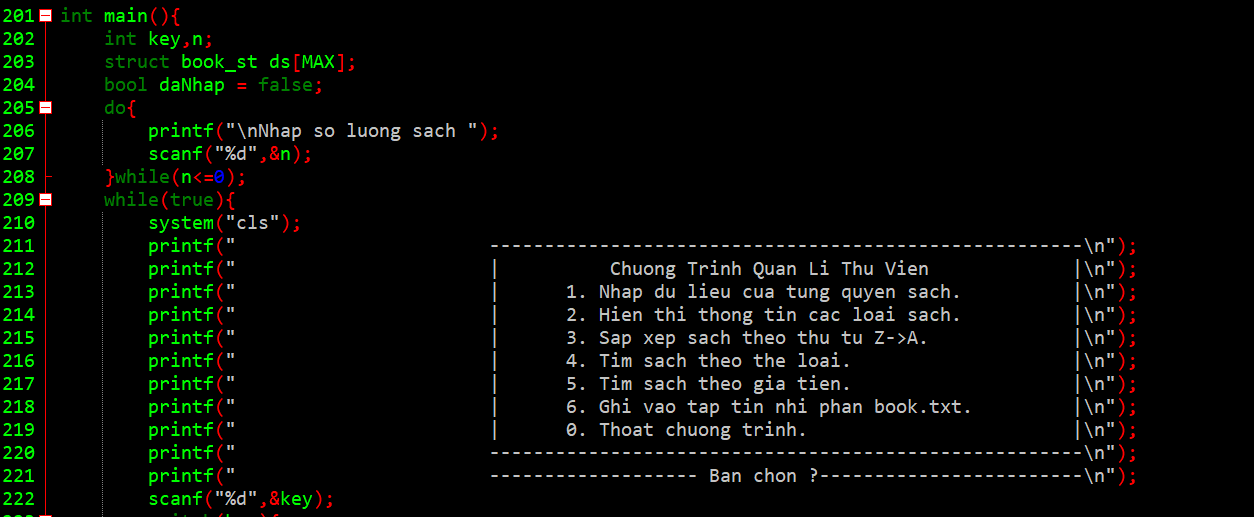
+ void timTheotheloai(struct book\_st \*ds, int n): tìm kiếm sách theo thể loại của sách.

+ 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.

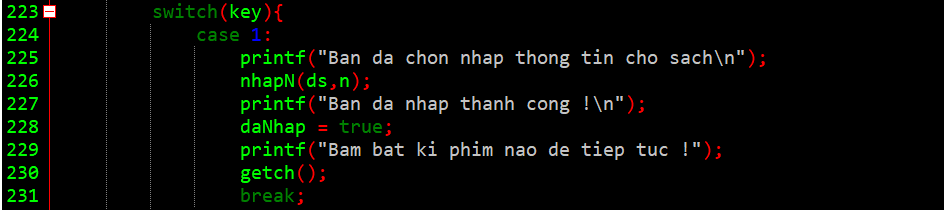
+ Hàm main:



Hình 1.1 Hàm main chính trong chương trình quản lý sách

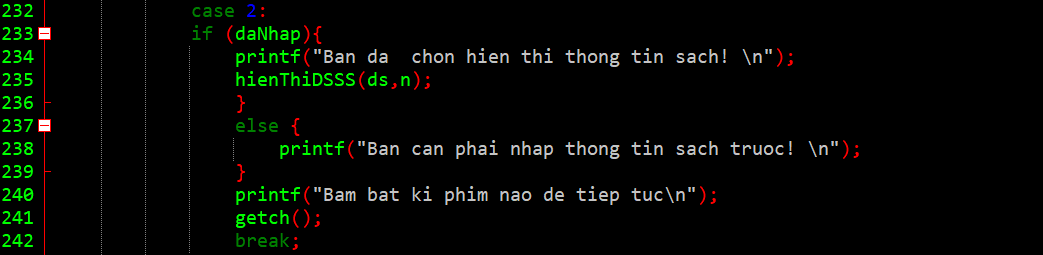
Tạo ra khung menu cho chương trình gồm có: Nhập dữ liệu, hiển thị thông tin các loại sách, sắp xếp tên theo thứ tự Z-A, tìm sách theo thể loại, tìm theo giá tiền, ghi vào tập tin nhị phân book.txt và phàn thoát chương trình.

+ Case 1 dùng để nhập thông tin cho sách



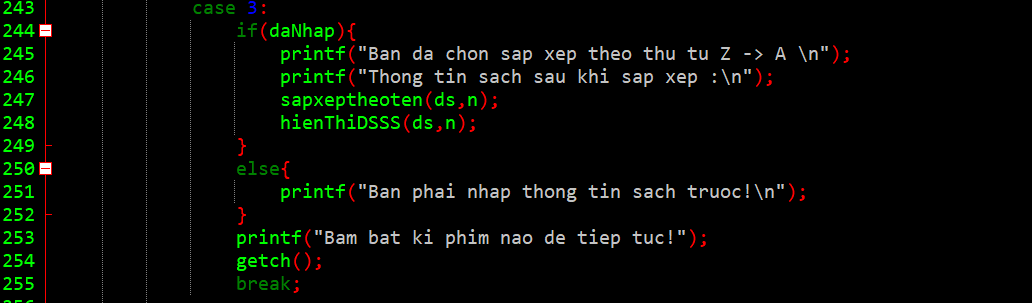
Hình 1.2 Case 1

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



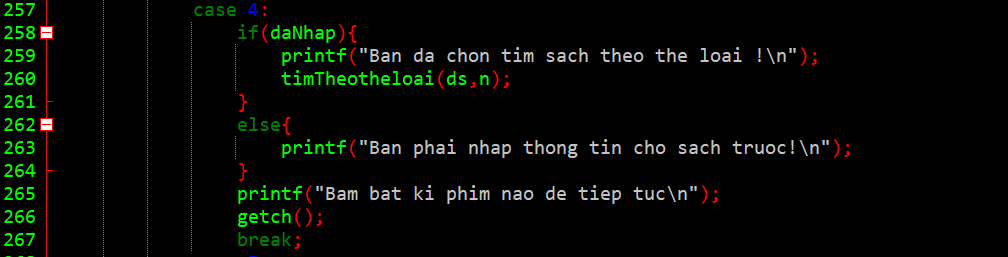
Hình 1.3 Case 2

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



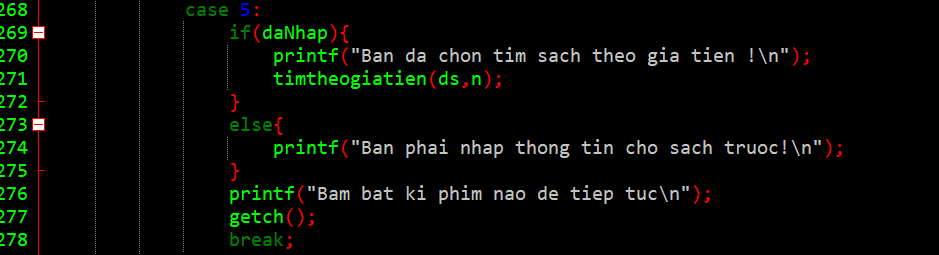
Hình 1.4 Case 3

+Case 4 tạo ra mục tìm sách theo thể loại



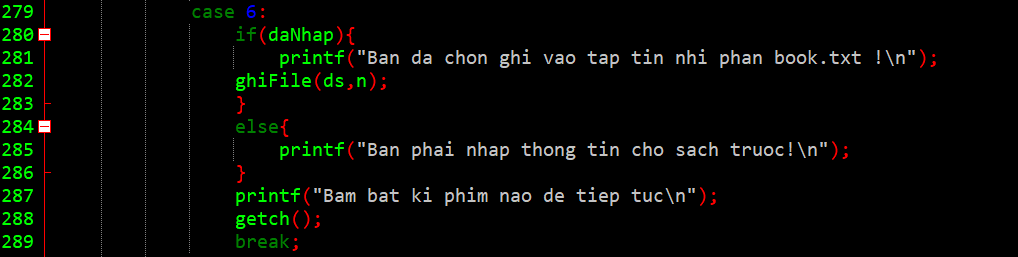
Hình 1.5 Case 4

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



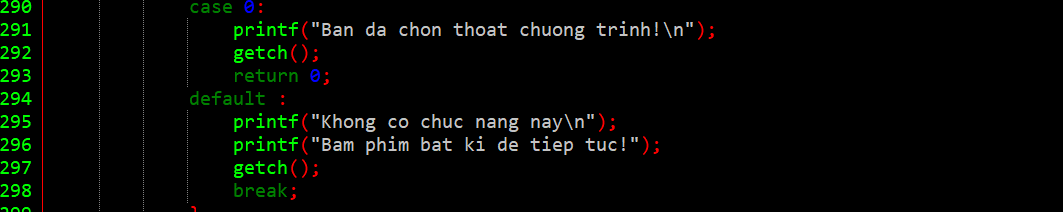
Hình 1.6 Case 5

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



Hình 1.7 Case 6

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



Hình 1.8 Case 0

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 chi tiết của từng quyển sách.

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 văn bản “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.

**Chương IV: Kết luận**

Chương trình đạt được một số mục tiêu nhất định tuy nhiên vẫn còn nhiều hạn chế như tính năng của chương trình.

Nâng cao khả năng tư duy và kĩ năng sử dụng ngôn ngữ lập trình C

Có kinh nghiệm để thực hiện các đề tài khác

Chương trình có thể dễ dàng phát triển thêm một số tính năng như: Quản lý người mượn, người trả sách; quản lý thời gian mượn và trả; tình trạng sách, chỉnh sửa thông tin cho sách...

**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>