

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

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

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | **Trần Thị Dung** |
| **Sinh viên: ◘Lê Đức Việt**  **◘Bạch Long Vũ**  **◘Trần Tuấn Sơn** | **MSSV: ◙ 6051071141**  **BÁO CÁO BÀI TẬP LỚN**  **MÔN**: LẬP TRÌNH NÂNG CAO  **CHỦ ĐỀ** :GAME XE TRÁNH CHƯỚNG NGẠI VẬT  **◙ 6051071145**  **◙ 6051071101** |
|  |
| **Mã học phần:** | CPM215.3 |

LỜI CẢM ƠN

*Tập thể thành viên Nhóm xin gửi lời cảm ơn chân thành đối với 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 học “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 những ý kiến đóng góp 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, tập thể thành viên 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, ngày 16 tháng 6 năm 2020*  *Nhóm sinh viên thực hiện:*  *Lê Đức Việt*  *Bạch Long Vũ*  *Trần Tuấn Sơn* |

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.......................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................... ........................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................ ....................................................................................................................................................................................................................................................................

*TP. Hồ Chí Minh, ngày 16 tháng 6 năm 2020*

**Giảng viên hướng dẫn**

Trần Thị Dung

**MỤC LỤC**

**MỤC LỤC**..........................................................................................................-1-

**CHƯƠNG I:TỔNG QUAN**...............................................................................-3-

1.1. Lý do chọn đề tài..........................................................................................-3-

1.2. Đối tượng nghiên cứu...................................................................................-3-

**CHƯƠNG 2:CƠ SỞ LÝ THUYẾT**..................................................................-4-

**Phần1**:**Giới thiệu**................................................................................................-4-

2.1.1. Giới thiệu về ngôn ngữ lập trình C............................................................-4-

2.1.2. Đặc điểm của ngôn ngữ lập trình C...........................................................-5-

**Phần 2 : Làm việc với tệp**..................................................................................-5-

2.2.1. Làm việc với tệp........................................................................................-5-

a. File văn bản –Text file.....................................................................................-5-

b. File nhị phân – Binary File..............................................................................-6-

**Phần 3:Danh sách liên kết đơn**.......................................................................-13-

2.3.1 Lý thuyết về danh sách liên kết.. .............................................................-13-

2.3.2. Danh sách liên kết đơn là gì.....................................................................-13-

2.3.3 Cài đặt danh sách liên kết đơn..................................................................-14-

**PHẦN 4: Thuật Toán sắp xếp và thuật toán tìm kiếm**.................................-22-

2.4.1. Các thuật toán sắp xếp.............................................................................-22-

### a. Thuật toán sắp xếp nổi bọt ( bubble sort).......................................................-22-

## b.Thuật toán sắp xếp chèn (insertion sort).........................................................-24-

c. Sắp xếp nhanh (Quick Sort)...........................................................................-26-

2.4.2. Thuật toán tìm kiếm.................................................................................-28-

a. Thuật toán tìm kiếm tuyến tính......................................................................-28-

b.Thuật toán tìm kiếm nhị phân.........................................................................-35-

**CHƯƠNG III: XỬ LÝ CHUYỂN ĐỘNG MÀU SẮC ĐIỀU KHIỂN CƠ BẢN TRONG LẬP TRÌNH**............................................................................-40-

3.1. Định nghĩa các hàm....................................................................................-40-

3.1.1. Hàm putchar.............................................................................................-40-

3.1.2. Hàm getch................................................................................................-40-

3.1.3. Hàm kbhit................................................................................................-41-

* 1. Một số thư viện chuẩn trong C....................................................................-41-

3.3. Xóa màn hình..............................................................................................-42-

3.4. Tọa độ trong lập trình.................................................................................-42-

* 1. Di chuyển con trỏ đến tọa độ như ý muốn..................................................-42-
  2. . Xử lý chạm biên.........................................................................................-42-
  3. . Điều khiển..................................................................................................-43-
  4. Hiển thị màu................................................................................................-43-

**CHƯƠNG IV:MÔ TẢ THIẾT KẾ GAME**...................................................-45-

4.1 Ý tưởng........................................................................................................-45-

4.2 Thiết kế game...............................................................................................-45-

4.3 Hướng dẫn sử dụng chương trình................................................................-51-

**CHƯƠNG I:TỔNG QUAN**

**1.1.Lý do chọn đề tài:**

Xã hội ngày càng phát triển, nhu cầu giải trí của con người ngày càng tăng cao. Cùng với sự phát triển của công nghệ thông tin, điện tử, các thiết bị chơi game ngày càng phổ biến. Trong những năm gần đây ngành công nghệ game nổi lên và đem lại doanh thu khổng lồ. Có thể nói game không chỉ là một phương tiện giải trí cơ bản, nó còn là một lĩnh vực đáng quan tâm và theo đuổi. Thiết kế game là một công việc không dễ dàng và nó đòi hỏi rất nhiều thời gian và công sức, tuy nhiên qua đó ta có thể biến những dòng code khô khan thành những trải nghiệm thú vị cho người chơi.

**1.2.Đối tượng nghiên cứu:**

Lập trình game cơ bản bằng ngôn ngữ C.

Cách thiết kế nhân vật, khung cảnh, xử lý va chạm, xử lý chạm biên, điểm số ,..

**CHƯƠNG II: CƠ SỞ LÝ THUYẾT**

**PHẦN 1: Giới thiệu**

**2.1.1.Giới thiệu về ngôn ngữ lập trình C**

C là ngôn ngữ lập trình cấp cao, được sử dụng rất phổ biến để lập trình hệ thống cùng với Assembler và phát triển các ứng dụng.

Vào những năm cuối thập kỷ 60 đầu thập kỷ 70 của thế kỷ XX, Dennish Ritchie (làm việc tại phòng thí nghiệm Bell) đã phát triển ngôn ngữ lập trình C dựa trên ngôn ngữ BCPL (do Martin Richards đưa ra vào năm 1967) và ngôn ngữ B (do Ken Thompson phát triển từ ngôn ngữ BCPL vào năm 1970 khi viết hệ điều hành UNIX đầu tiên trên máy PDP-7) và được cài đặt lần đầu tiên trên hệ điều hành UNIX của máy DEC PDP-11.

Năm 1978, Dennish Ritchie và B.W Kernighan đã cho xuất bản quyển “Ngôn ngữ lập trình C” và được phổ biến rộng rãi đến nay.

Lúc ban đầu, C được thiết kế nhằm lập trình trong môi trường của hệ điều hành Unix nhằm mục đích hỗ trợ cho các câu lệnh lập trình phức tạp. Nhưng về sau, với những nhu cầu phát triển ngày một tăng của câu lệnh lập trình, C đã vượt qua khuôn khổ của phòng thí nghiệm Bell và nhanh chóng hội nhập vào thế giới lập trình, các công ty lập trình sử dụng ngôn ngữ lập trình C một cách rộng rãi. Sau đó, các công ty sản xuất phần mềm lần lượt đưa ra các phiên bản hỗ trợ cho việc lập trình bằng ngôn ngữ lập trình C và chuẩn ANSI C ra đời.

Ngôn ngữ lập trình C là một ngôn ngữ lập trình hệ thống rất mạnh và rất “mềm dẻo”, có một thư viện gồm rất nhiều các hàm (function) đã được tạo sẵn. Người lập trình có thể tận dụng các hàm này để giải quyết các bài toán mà không cần phải tạo mới. Hơn thế nữa, ngôn ngữ lập trình C hỗ trợ rất nhiều phép toán nên phù hợp cho việc giải quyết các bài toán kỹ thuật có nhiều công thức phức tạp. Ngoài ra, C cũng cho phép người lập trình tự định nghĩa thêm các kiểu dữ liệu trừu tượng mới. Tuy nhiên, điều mà người mới vừa học lập trình C thường gặp “rắc rối” là “hơi khó hiểu” do sự “mềm dẻo” của C. Dù vậy, C được phổ biến khá rộng rãi và đã trở thành một công cụ lập trình khá mạnh, được sử dụng như là một ngôn ngữ lập trình chủ yếu trong việc xây dựng những phần mềm hiện nay.

**2.1.2. Đặc điểm của ngôn ngữ lập trình C**

− Tính cô đọng (compact): C chỉ có 32 từ khóa chuẩn và 40 toán tử chuẩn, nhưng hầu hết đều được biểu diễn bằng những chuỗi ký tự ngắn gọn.

− Tính cấu trúc (structured): C có một tập hợp những chỉ thị của lập trình như cấu trúc lựa chọn, lặp… Từ đó các chương trình viết bằng C được tổ chức rõ ràng, dễ hiểu.

− Tính tương thích (compatible): C có bộ tiền xử lý và một thư viện chuẩn vô cùng phong phú nên khi chuyển từ máy tính này sang máy tính khác các chương trình viết bằng C vẫn hoàn toàn tương thích.

− Tính linh động (flexible): C là một ngôn ngữ rất uyển chuyển và cú pháp, chấp nhận nhiều cách thể hiện, có thể thu gọn kích thước của các mã lệnh làm chương trình chạy nhanh hơn.

− Biên dịch (compile): C cho phép biên dịch nhiều tập tin chương trình riêng rẽ thành các tập tin đối tượng (object) và liên kết (link) các đối tượng đó lại với nhau thành một chương trình có thể thực thi được (executable) thống nhất.

Ngôn ngữ lập trình C cũng là một công cụ để truy nhập vào bộ nhớ máy tính, truy cập các chức năng bên trong DOS và BIOS, lập trình điều khiển cho các linh kiện điện tử khác.

Phần II: Làm Việc Với Tệp

**2.2 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.

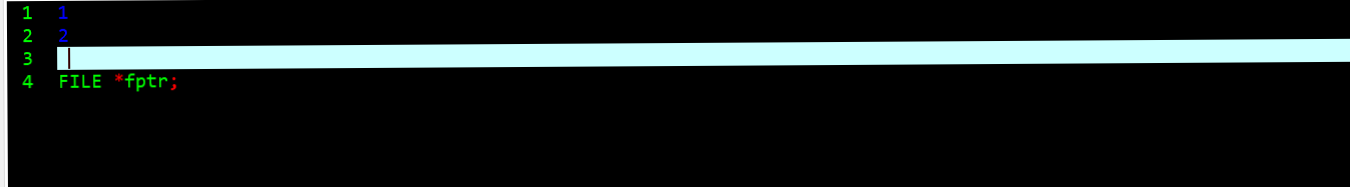
Các thao tác với file

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

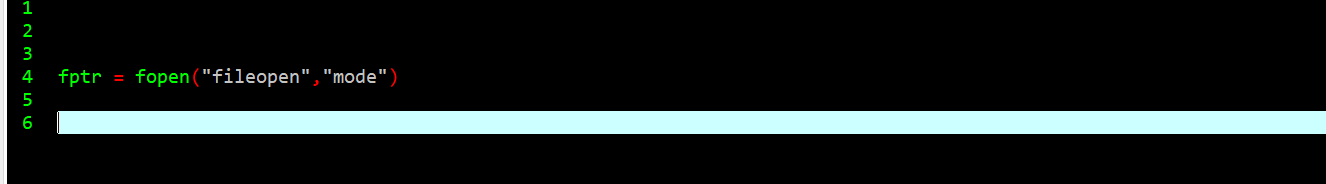
Khi làm việc với file, bạn cần khai báo 1 con trỏ kiểu FILE. Việc khai báo này là cần thiết để có sự kết nối giữa chương trình của bạn và tập tin mà bạn cần thao tác.

*Ví dụ 2.1:*

### Thao tác mở file

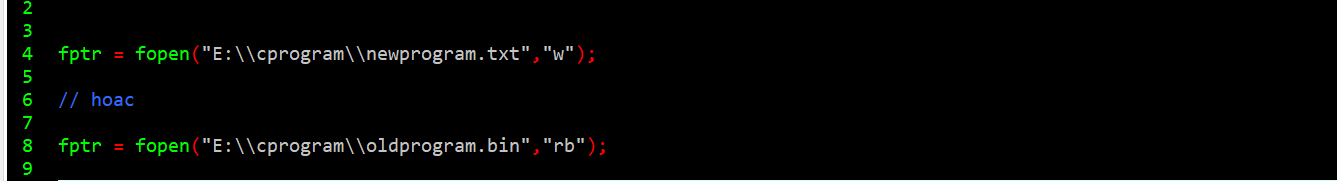
Để đọc ghi file trong C cũng như trong mọi ngôn ngữ lập trình, việc đầu tiên bạn cần làm là mở file mà bạn muốn làm việc. Trong ngôn ngữ lập trình C, chúng ta có thể mở file bằng cách sử dụng hàm fopen() trong thư viện stdio.h như sau:\

*Ví dụ 2.2 :*open("fileopen","mode")

 v

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

Ví dụ 2.3 :



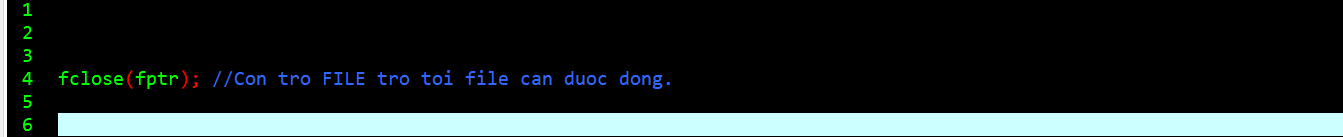
* Giả sử tập tin newprogram.txt chưa có trong thư mục E:\cprogram. Ví dụ đầu tiên với mode = "w" sẽ cho phép chương trình tự động tạo ra file newprogram.txt nếu nó chưa có. Và sau đó mở file này lên nhưng chương trình chỉ có thể ghi dữ liệu vào mà không thể đọc.
* Mode là w chỉ cho phép chương trình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

*Ví dụ 2.4:*

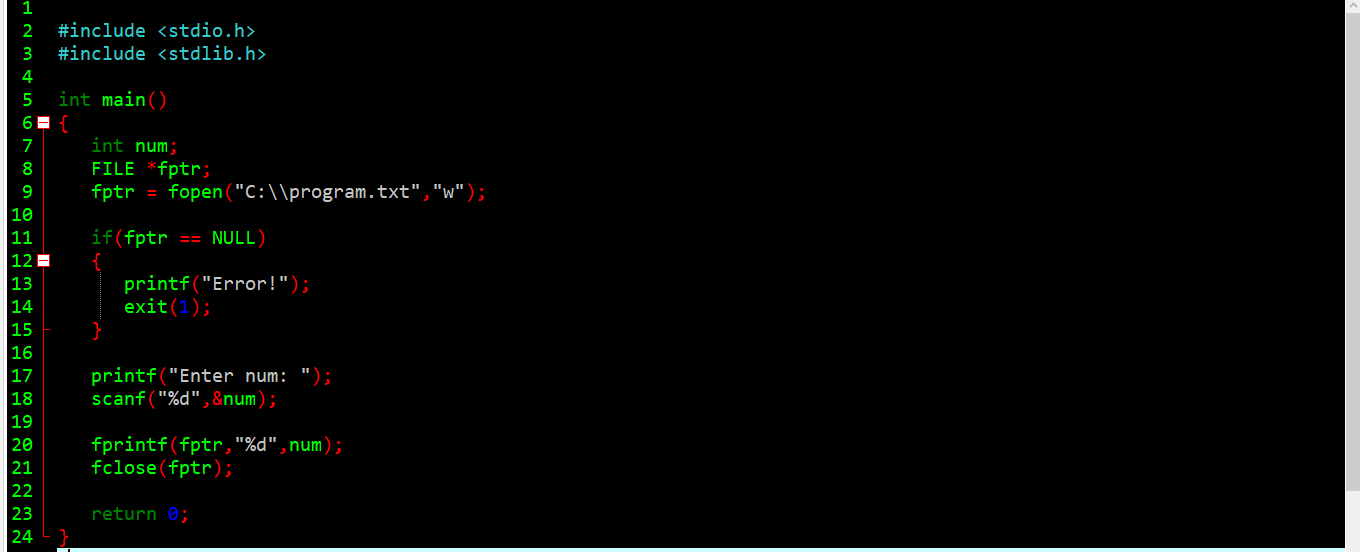


## \*.Đọc/Ghi file văn bản trong C

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.

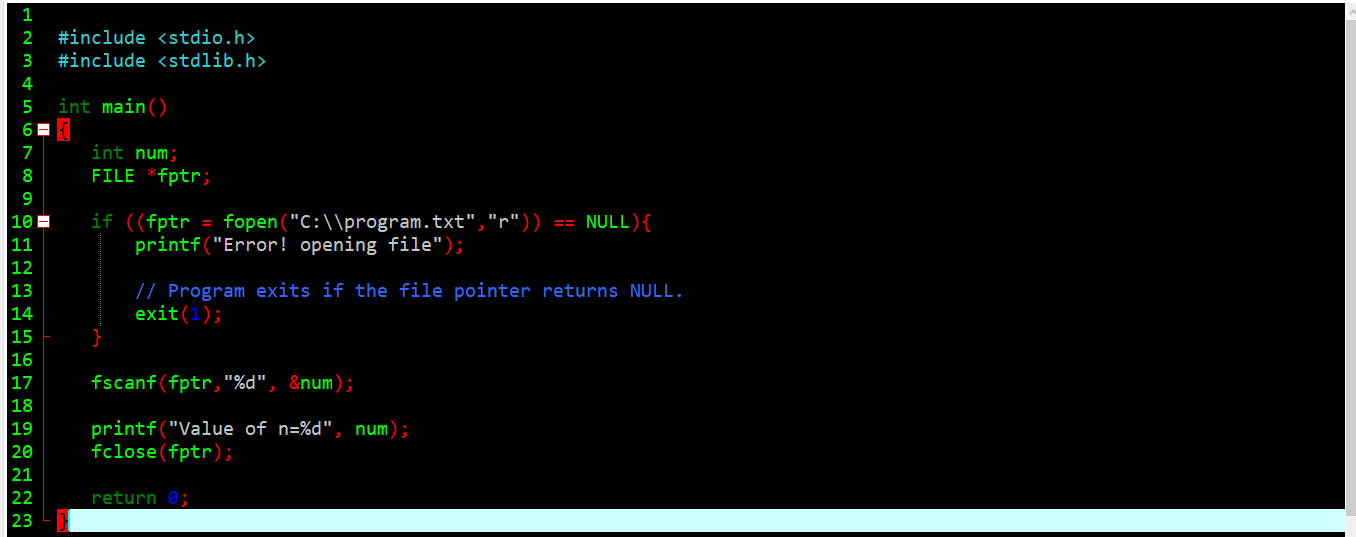
Để làm việc với file văn bản, chúng ta sẽ sử dụng fprintf() và fscanf().

#### Ví dụ 2.5: 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.

#### Ví dụ 2.6: 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.

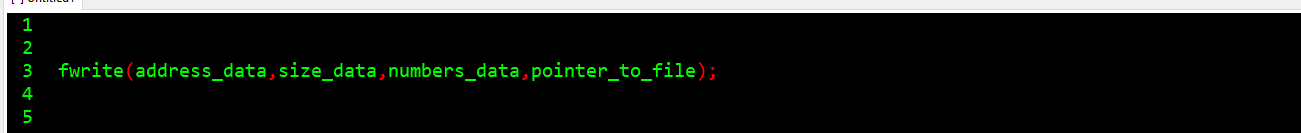
## Đọc/Ghi file nhị phân trong C

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

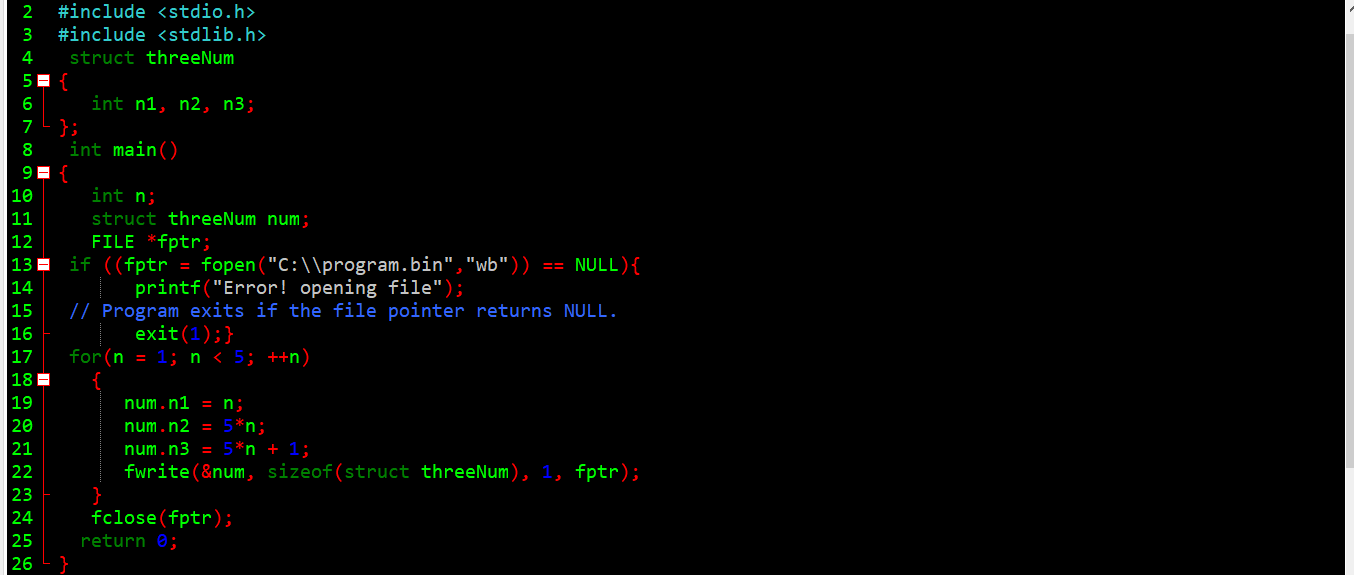
#### **Ghi file nhị phân**

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.

*Ví dụ 2.7:*



#### Ví dụ 2.8: Ghi file nhị phân sử dụng fwrite()



Trong Vd2.8 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:

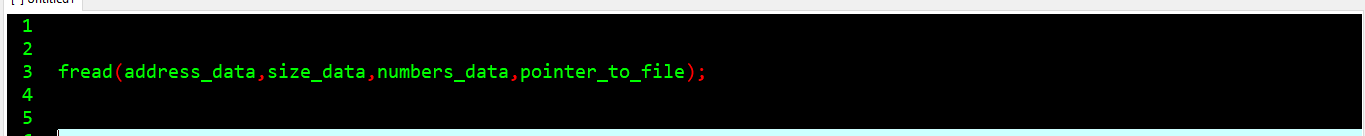
* Tham số đầu tiên là địa chỉ của biến num
* Tham số thứ 2 là kích thước của biến num
* Tham số thứ 3 là số lượng kiểu dữ liệu – ở đây là 1.
* Tham số thứ 4 là con trỏ FILE trỏ tới tệp tin program.bin

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

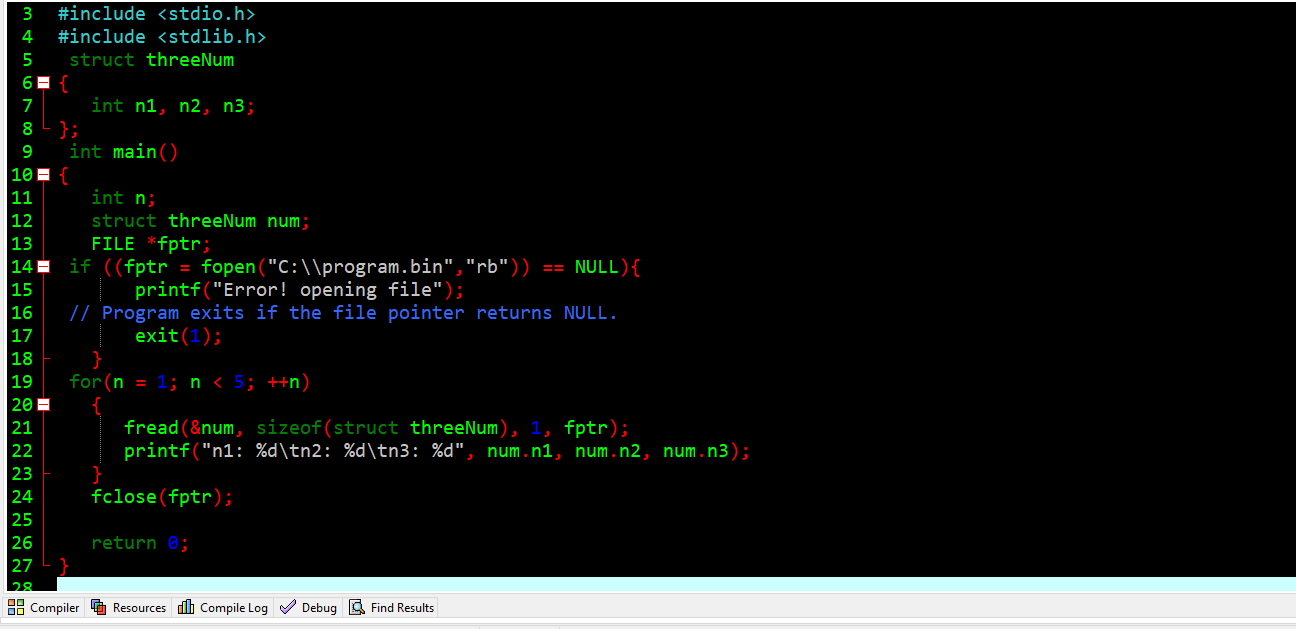
#### **Đọc file nhị phân**

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

*Ví dụ 2.9:*



Ví dụ 2.10: đọc file nhị phân sử dụng fread()



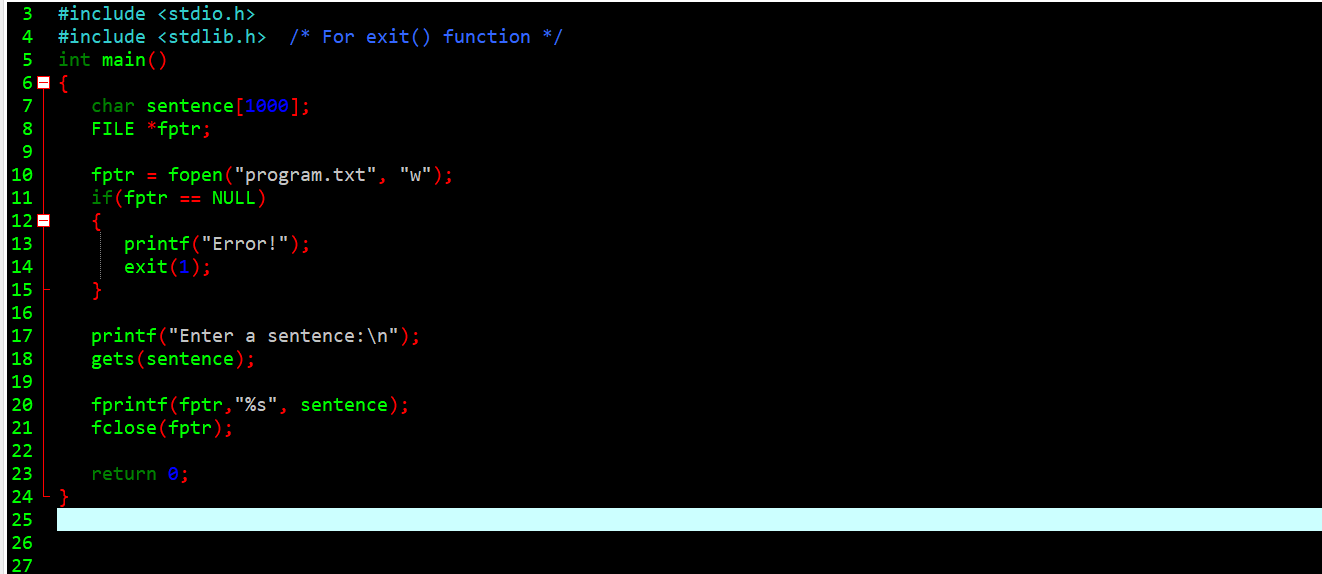
## Trong ví dụ này, bạn đọc file program.bin và lặp qua từng dòng. Bạn sẽ nhận được các giá trị tương ứng khi bạn ghi vào trong VD2.8.

## 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:

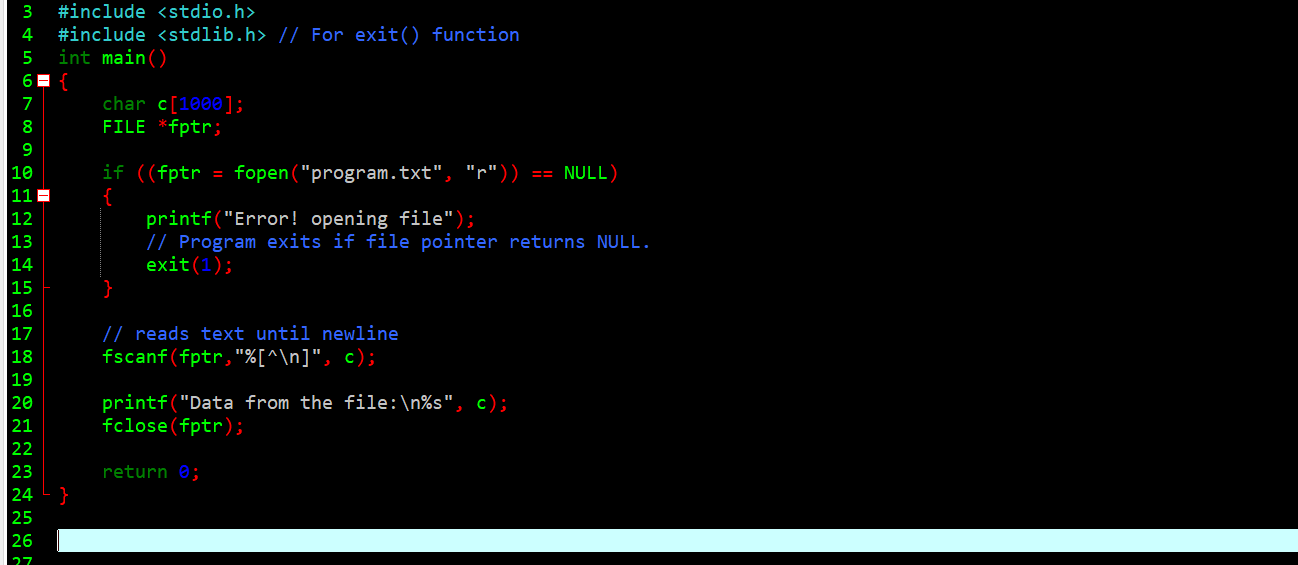
* Ghi văn bản vào file trong C
* Đọc dữ liệu văn bản từ file trong C

#### Ví dụ 2.11:Ghi vào file một câu văn bản



|  |
| --- |
|  |

#### Ví dụ 2.12: Đọc dữ liệu văn bản từ file



**PHẦN 3. DANH SÁCH LIÊN KẾT ĐƠN**

**Danh sách liên kết đơn(Single linked list)** là ví dụ tốt nhất và đơn giản nhất về cấu trúc dữ liệu động sử dụng con trỏ để cài đặt. Do đó, kiến thức con trỏ là rất quan trọng để hiểu cách danh sách liên kết hoạt động, vì vậy nếu bạn chưa có kiến thức về con trỏ thì bạn nên học về con trỏ trước. Bạn cũng cần hiểu một chút về cấp phát bộ nhớ động. Để đơn giản và dễ hiểu, phần nội dung cài đặt danh sách liên kết của bài viết này sẽ chỉ trình bày về danh sách liên kết đơn.

**2.3.1. Lý thuyết về danh sách liên kết**

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

|  |  |  |
| --- | --- | --- |
| **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 |

**Lưu ý:** Ở bảng phía trên, các phần in nghiêng thể hiện đó là ưu điểm so với đối thủ còn lại.

## 2.3.2. 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, 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:

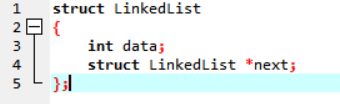
* Danh sách liên kết đơn(Single linked list): Chỉ có sự kết nối từ phần tử phía trước tới phần tử phía sau.
* Danh sách liên kết đôi(Double linked list): Có sự kết nối 2 chiều giữa phần tử phía trước với phần tử phía sau
* Danh sách liên kết vòng(Circular Linked List): Có thêm sự kết nối giữa 2 phần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

**2.3.3. Cài đặt danh sách liên kết đơn**

*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.

*Ví dụ 2.13:*



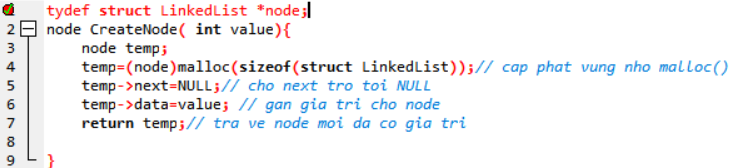
Khai báo trên sẽ được sử dụng cho mọi Node trong linked list. Trường data sẽ lưu giữa giá trị và next sẽ là con trỏ để trỏ đến thằng kế tiếp của nó.

**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.

### *2. Tạo mới 1 Node*

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

*Ví dụ 2.14:*



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.

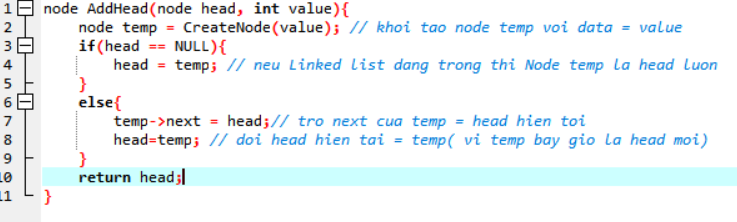
### *3. Thêm Node vào danh sách liên kết*

#### **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

*Ví dụ 2.15:*

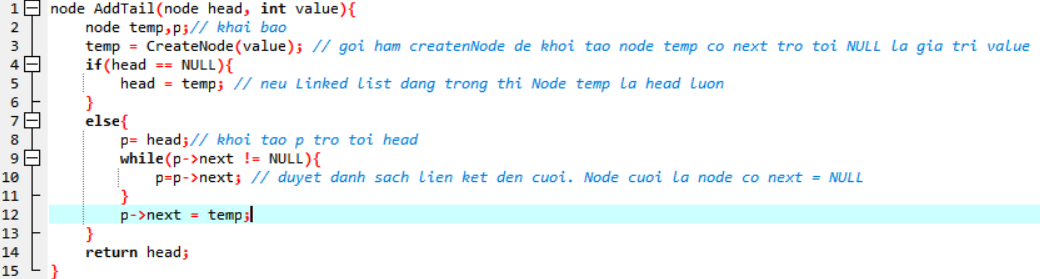


#### **Thêm vào cuối**

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

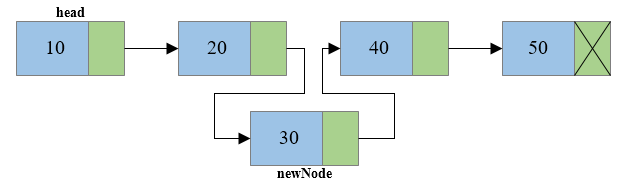
1. Tạo một Node mới với giá trị value
2. Nếu head = NULL, tức là danh sách liên kết đang trống. Khi đó Node mới(temp) sẽ là head luôn.
3. Ngược lại, ta sẽ duyệt tới Node cuối cùng(Node có next = NULL), và trỏ next của thằng cuối tới Node mới(temp).

*Ví dụ 2.16:*



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

#### **Thêm vào vị trí bất kỳ**

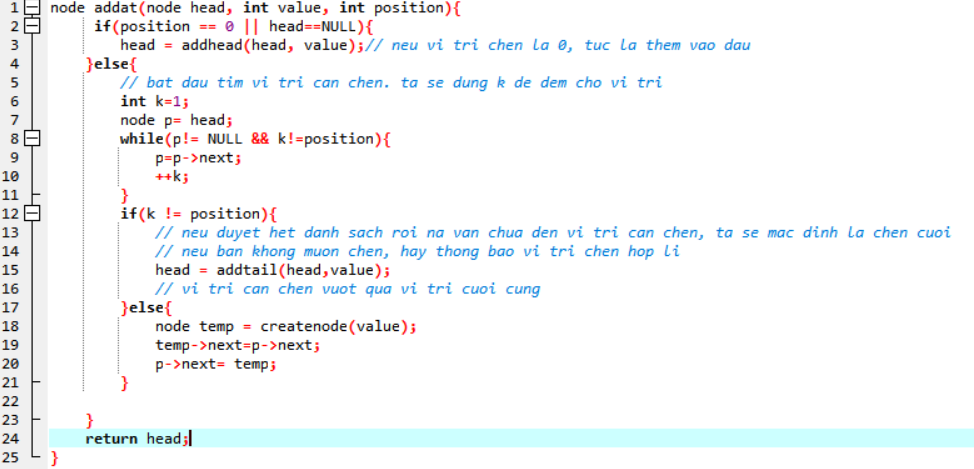


Để làm được việc này, ta phải duyệt từ đầu để tìm tới vị trí của Node cần chèn, giả sử là Node Q, khi đó ta cần làm theo thứ tự sau:

* 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

*Ví dụ 2.17:*

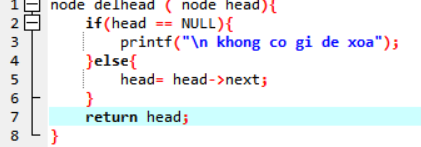


Lưu ý: Bạn phải làm theo thứ tự trên, nếu bạn cho p->next = temp trước. Khi đó, bạn sẽ không thể lấy lại phần sau của danh sách liên kết nữa(Vì next chỉ được được lưu trong p->next mà thay đổi p->next rồi thì còn đâu giá trị cũ).

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

#### **Xóa đầu**

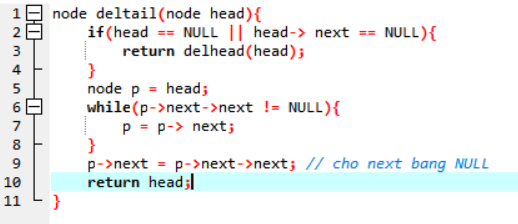
Xóa đầu đơn giản lắm, bây giờ chỉ cần cho thằng kế tiếp của head làm head là được thôi. Mà thằng kế tiếp của head chính là head->next.



#### **Xóa cuối**

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

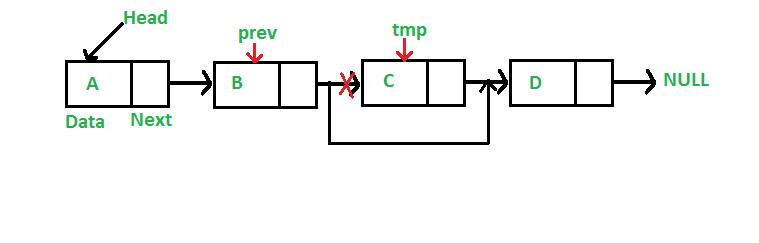
*Ví dụ 2.18:*



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

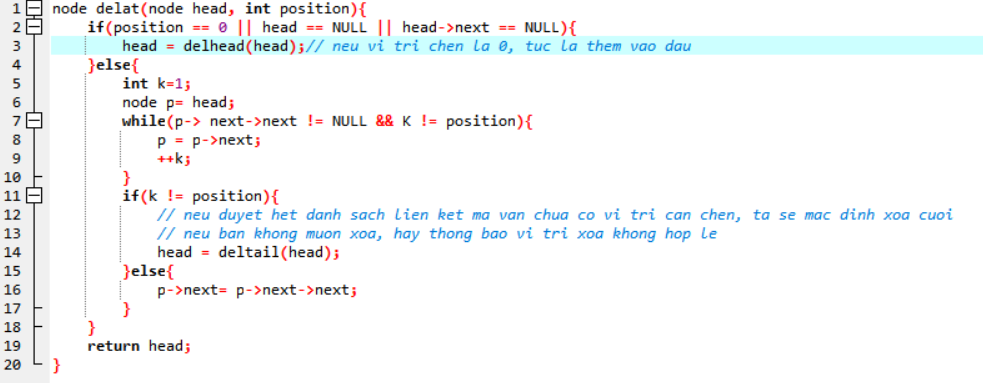
#### **Xóa ở vị trí bất kỳ**

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

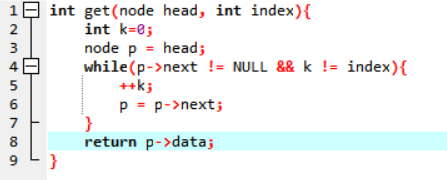
*Ví dụ 2.19:*



### *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.

*Ví dụ 2.20:*

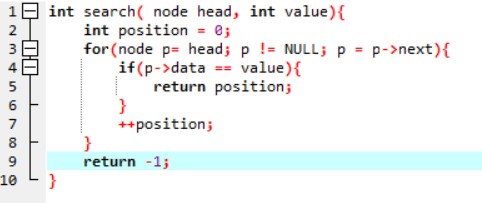


Lý do dùng p->next != NULL là vì chúng ta chỉ muốn đi qua các phần tử có value.

### Tìm kiếm trong danh sách liên kết

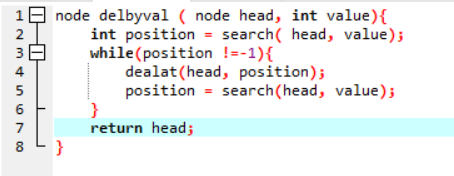
Hàm tìm kiếm này sẽ trả về chỉ số của Node đầu tiên có giá trị bằng với giá trị cần tìm. Nếu không tìm thấy, chúng ta trả về -1.

*Ví dụ 2.21:*



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:

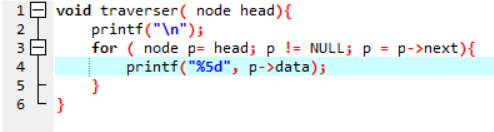
*Ví dụ 2.22:*



### *\*) 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.

*Ví dụ 2.23:*



**PHẦN 4: THUẬT TOÁN SẮP XẾP VÀ THUẬT TOÁN TÌM KIẾM**

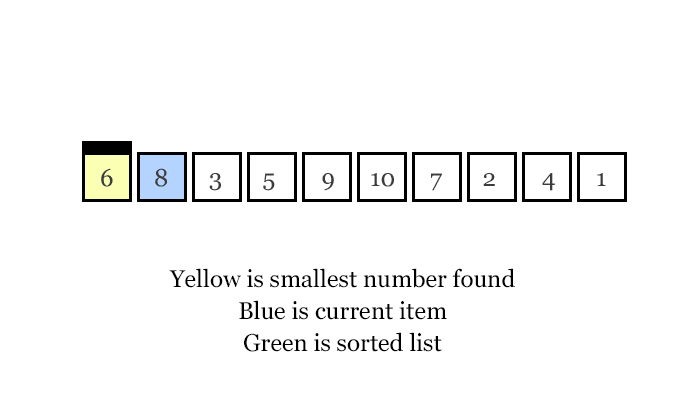
**2.4.1. Các thuật toán sắp xếp**

### a.Sắp xếp nổi bọt( bubble sort)

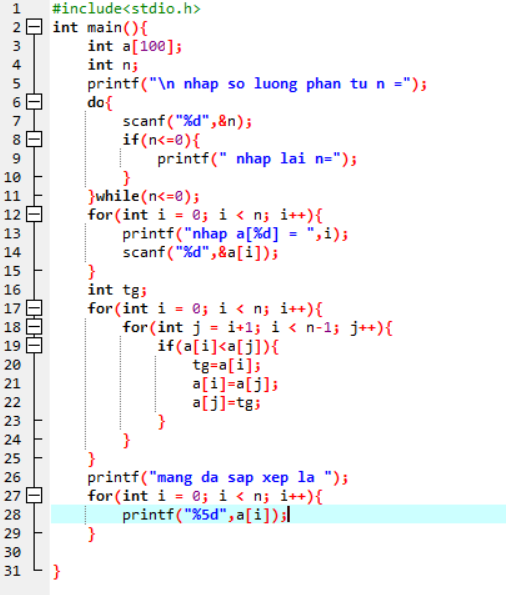
*1.sắp xếp giảm dần*

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:



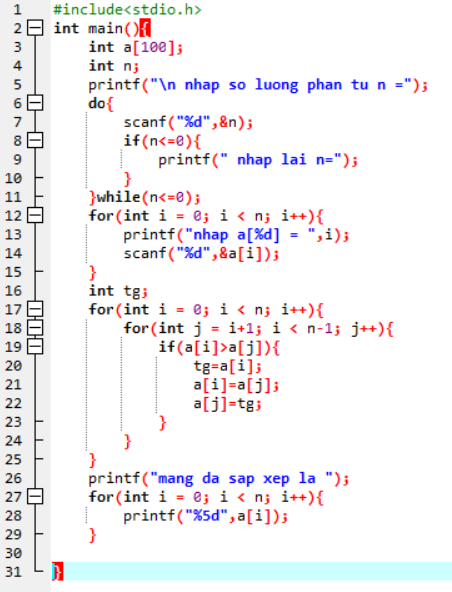
Code sắp xếp mảng/ dãy số giảm dần với C/C++ là *Ví dụ 2.24:*



### *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.

*Ví dụ 2.25:*



## b. Sắp xếp xen vào (insertion sort)

### Ý tưởng

Cái tên của thuật toán cũng giúp mình hình dung được phần nào ý tưởng của thuật toán. Giả sử đoạn đầu của mảng A[0..i-1] (với i >= 1) đã được sắp xếp, nghĩa là ta đã có A[0].key <= A[1].key <= ... <= A[i-1].key. Ta xen A[i] vào vị trí thích hợp trong đoạn đầu A[0..i-1] để nhận được A[0..i] được sắp xếp. Với i = 1 thì coi như đoạn đã được sắp xếp. Lặp lại quá trình đó với i = 2, ..., n-1 để có được mảng sắp xếp. Việc sắp xếp sẽ được tiến hành như sau: Cho chỉ số k chạy từ i, nếu A[k].key < A[k-1].key thì ta tráo đổi vị trí của A[k] và A[k-1] rồi giảm k đi 1.

### Ví dụ

Ta có 1 mảng số nguyên A[0..5] = (1, 3, 7, 2, 8, 4) và đoạn đầu A[0..2] = (1, 3, 7) đã được sắp xếp Đầu tiên ta sẽ chọn i = 3 ( vì giá trị từ 0 đến 2 đã được sắp xếp) và k = i = 3. Nhận thấy A[3] < A[2] nên ta tráo đổi A[3] và A[2], ta có:

| **A[0]** | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 2 | 7 | 8 | 4 |

| **A[0]** | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 2 | 7 | 8 | 4 |

Đến đây thì ta có k = 2, A[2] < A[1] nên ta lại tráo A[2] cho A[1]:

| **A[0]** | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 8 | 4 |

Lúc này k = 1 và A[1] >= A[0] nên ta dừng lại và ta đã có đoạn đầu A[0..3] được sắp xếp Lặp lại các bước trên với i = 4, 5 ta sẽ được mảng sắp xếp tăng dần

### Thuật toán

Hàm sắp xếp xen vào được viết như sau:

void InsertionSort(A[], int n) {

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

for (int k = i; k > 0; k--) {

if (A[k].key < A[k-].key) {

Swap(A[k], A[k-1]); //doi cho A[k] va A[k-1]

} else {

break;

}

}

# c. Sắp xếp nhanh (Quick Sort)

## 1. Giới thiệu



Sắp xếp nhanh (Quick Sort) còn có một tên gọi khác là sắp xếp phân chia (Part Sort) dựa trên ý tưởng thuật toán. Nó được phát minh lần đầu bởi C.A.Hoare vào năm 1960. Có lẽ đây là thuật toán được nghiên cứu và sử dụng rộng rãi nhất trong các thuật toán sắp xếp. Quick sort cũng là thuật toán đệ quy. Ngược với Mergesort gọi đệ quy rồi mới xử lý, Quick sort xử lý xong mới gọi đệ quy.

Ý tưởng của thuật toán này dựa trên phương pháp chia để trị, nghĩa là chia dãy cần sắp xếp thành 2 phần, sau đó thực hiện việc sắp xếp cho mỗi phần độc lập nhau. Để làm việc này thì ta cần phải làm các bước sau:

**B1**. Chọn ngẫu nhiên một phần tử nào đó của dãy làm phần tử khóa (pivot) Kĩ thuật chọn phần tử khóa rất quan trọng vì nếu không may bạn có thể bị rơi vào vòng lặp vô hạn đối với các trường hợp đặc biệt. Tốt nhất là chọn phần tử ở vị trí trung tâm của dãy. Khi đó sau https://viblo.asia/uploads/72878f97-ff33-46af-8e5b-dc3d04453c97.png lần phân chia ta sẽ đạt tới kích thước danh sách bằng 1. Tuy nhiên điều đó rất khó. Có các cách chọn phần tử khóa như sau:

* Chọn phần tử đứng đầu hoặc đứng cuối làm phần tử khóa.
* Chọn phần tử đứng giữa danh sách làm phần tử khóa.
* Chọn phần tử trung gian trong 3 phần tử đứng đầu, đứng giữa và đứng cuối làm phần tử khóa.
* Chọn phần tử ngẫu nhiên làm phần tử khóa. (Cách này có thể dẫn đến khả năng rơi vào các trường hợp đặc biệt)

**B2**. Xếp các phần tử nhỏ hơn phần tử chốt ở phía trước phần tử khóa

**B3**. Xếp các phần tử lớn hơn phần tử chốt ở phía sau phần tử khóa Để có được sự phân loại này thì ở 2 bước trên, các phần tử sẽ được so sánh với khóa và hoán đổi vị trí cho nhau hoặc cho khóa nếu nó lớn hơn khóa mà lại nằm trước khóa, hoặc nhỏ hơn mà lại nằm sau khóa. Áp dụng kĩ thuật như trên cho mỗi đoạn đó và tiếp tục làm vậy cho đến khi mỗi đoạn chỉ còn 2 phần tử. Khi đó toàn bộ dãy đã được sắp xếp

Quick sort là một thuật toán dễ cài đặt, hiệu quả trong hầu hết các trường hợp và tiêu tốn ít tài nguyên hơn so với các thuật toán khác. Độ phức tạp trung bình của giải thuật là O(NlogN).

## 2. Các bước thực hiện giải thuật

Giả sử ta có một dãy các phần tử như sau:

| **A[0]** | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** | **A[6]** | **A[7]** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 28 | 16 | 56 | 30 | 17 | 32 | 24 | 18 |

left = 0; right = 7; Ở ví dụ này ta lấy một phần tử làm khóa, ở đây mình chọn phần tử khóa là A[(0+7)/2] = A[3] = 30 Cho i chạy từ trái sang phải, bắt đầu từ vị trí 0 Cho j chạy từ phải sang trái, bắt đầu từ vị trí 7 Quá trình duyệt từ bên trái với biến duyệt i sẽ dừng lại ở 56 (i=2), vì đây không phải là phần tử nhỏ hơn khóa. Quá trình duyệt từ bên phải với biến duyệt j sẽ dừng lại ở 18 (j=7) vì đây là phần tử nhỏ hơn khóa. Tiến hành đổi chỗ 2 phần tử cho nhau.

| **A[0]** | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** | **A[6]** | **A[7]** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 28 | 16 | 18 | 30 | 17 | 32 | 24 | 56 |

Quá trình duyệt tiếp tục. Khóa ở đây vẫn là 30. Ta tiếp tục tăng biến duyệt i và giảm biến duyệt j, nhận thấy biến duyệt i chạy đến 30 (i=3), còn biến duyệt j dừng lại ở 24 (j=6). Tiến hành đổi chỗ 2 phần tử cho nhau.

| **A[0]** | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** | **A[6]** | **A[7]** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 28 | 16 | 18 | 24 | 17 | 32 | 30 | 56 |

Quá trình duyệt tiếp tục. Khóa ở đây vẫn là 30. Tiếp tục tăng biến duyệt i và giảm biến duyệt j, nhận thấy biến duyệt i chạy đến 32 (i=5), còn biến duyệt j dừng lại ở 30(j=6). Tiến hành đổi chỗ 2 phần tử cho nhau.

| **A[0]** | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** | **A[6]** | **A[7]** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 28 | 16 | 18 | 24 | 17 | 30 | 32 | 56 |

Quá trình duyệt tiếp tục. Khóa ở đây vẫn là 30. Tiếp tục tăng biến duyệt i và giảm biến duyệt j, nhận thấy biến duyệt i và j cùng chạy đến 30. Lúc này ta sẽ chia mảng trên ra làm 2 mảng con A[0..4] và A[5..7], vì ta thấy A[5..7] đã được sắp xếp tăng dần rồi nên ta sẽ xét A[0..4] A[0..4] = [28, 16, 18, 24, 17] Lặp lại những bước trên với phần tử được chọn là khóa là A[(0+4)/2] = A[2] = 18 left = 0; right = 4; Cho i chạy từ trái sang phải bắt đầu từ vị trí 0 Cho j chạy từ phải sang trái bắt đầu từ vị trí 4 Quá trình duyệt từ bên trái với biến duyệt i sẽ dừng lại ở 28(i=0) vì đây không phải là phần tử nhỏ hơn khóa. Quá trình duyệt từ bên phải với biến duyệt j sẽ dừng lại ở 17 (j=4) vì đây là phần tử không lớn hơn khóa. Tiến hành đổi chỗ 2 phần tử cho nhau.

| **A[0]** | **A[1]** | **A[2]** | **A[3]** | **A[4]** |
| --- | --- | --- | --- | --- |
| 17 | 16 | 18 | 24 | 28 |

Quá trình duyệt tiếp tục. Tăng biến duyệt i và giảm biến duyệt j. Lúc này i = j nên ta sẽ chia mảng trên ra làm 2 mảng con A[0..1] và A[2..4]. Ta thấy A[2..4] đã được sắp xếp tăng dần rồi, nên ta chỉ xét A[0..1], vì A[0] > A[1] nên ta đổi chỗ 2 phần tử cho nhau và được mảng sắp xếp tăng dần cuối cùng:

| **A[0]** | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** | **A[6]** | **A[7]** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 16 | 17 | 18 | 24 | 28 | 30 | 32 | 56 |

## 3. Cài đặt giải thuật

Như đã nói ở trên, Quick sort cũng là một thuật toán đệ quy, bao gồm việc chia mảng thành 2 mảng con thỏa mãn yêu cầu trên, sau đó thực hiện lời gọi đệ quy với 2 mảng con vừa tạo được.

## 4. Phân tích

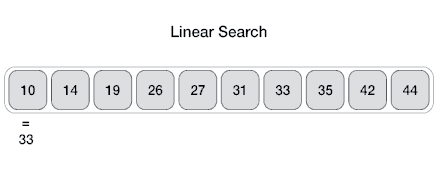
Nhược điểm của Quick Sort là không hiệu quả trên những dãy đã được sắp xếp sẵn. Khi đó ta phải mất N lần gọi đệ quy và mỗi lần chỉ loại được 1 phần tử. Thời gian thực hiện thuật toán trong trường hợp xấu nhất này là O(n2). Trong trường hợp tốt nhất, mỗi lần phân chia sẽ được 2 nửa dãy bằng nhau, khi đó thời gian thực hiện thuật toán là: T(N) = 2T(N/2) + N Khi đó T(N) xấp xỉ bằng NlogN. Như vậy Quick sort là thuật toán hiệu quả trong đa số các trường hợp.

**2.4.2 Thuật toán tìm kiếm**

***a.Thuật toán tìm kiếm tuyến tính***

Tìm kiếm tuyến tính (Linear Search) là gì ?

Linear Search là một giải thuật tìm kiếm rất cơ bản. Trong kiểu tìm kiếm này, một hoạt động tìm kiếm liên tiếp được diễn ra qua tất cả từng phần tử. Mỗi phần tử đều được kiểm tra và nếu tìm thấy bất kỳ kết nối nào thì phần tử cụ thể đó được trả về; nếu không tìm thấy thì quá trình tìm kiếm tiếp tục diễn ra cho tới khi tìm kiếm hết dữ liệu.



Giải thuật tìm kiếm tuyến tính

Giải thuật tìm kiếm tuyến tính ( Mảng A, Giá trị x)

Bước 1: Thiết lập i thành 1

Bước 2: Nếu i > n thì chuyển tới bước 7

Bước 3: Nếu A[i] = x thì chuyển tới bước 6

Bước 4: Thiết lập i thành i + 1

Bước 5: Tới bước 2

Bước 6: In phần tử x được tìm thấy tại chỉ mục i và tới bước 8

Bước 7: In phần tử không được tìm thấy

Bước 8: Thoát

**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.

Ví dụ: Cho danh sách sinh viên với với cấu trúc: mã sinh viên, họ, tên, năm sinh.

Ma SV                                     Ho                   Ten                  NamSinh

2001140442                            Nguyen           Tuan                1996

2001140443                            Le                    Huyen             1996

2001140444                            Nguyen           Phat                 1996

2001140445                            Tran                 An                   1996

2001140446                            Phan                Tran                 1996

Hãy tìm sinh viên có tên Phat.

Tương tự như dãy của mảng số nguyên ở ví dụ 1. Ta đi qua lần lượt các phần tử, ở đây chúng ta tạm đặt cấu trúc SinhVien và gọi là SV và mảng a[], khi đó ta xét a[i].Ten==Phat.

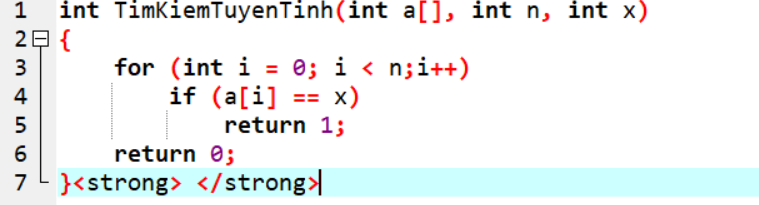
Với i=0 thì a[0].Ten == Tuan != Phat nên ta tăng i lên một đơn vị và i<n-1 nên

i=1 thì a[1].Ten == Huyen !Phat nên tăng i lên một đơn vị và i<n-1 nên.

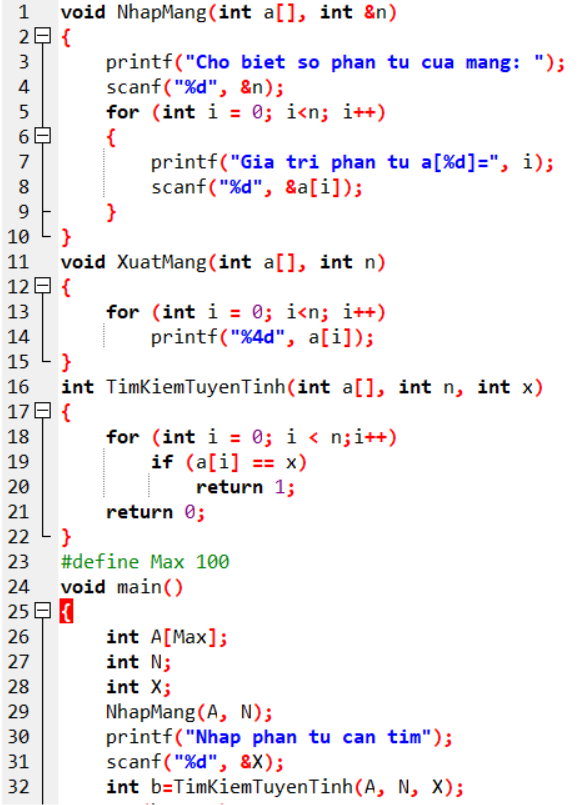
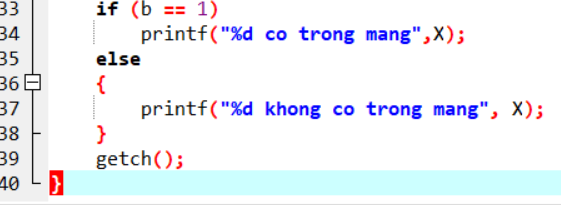
i=2 thì a[2].Ten == Phat == Phat nên dừng lại và cho kết quả là tìm thấy. Code về ví dụ này mình sẽ viết ở phần dưới.

**Code C/C++**

***Ví dụ 2.26:***

**Code tổng quát giải thuật:**

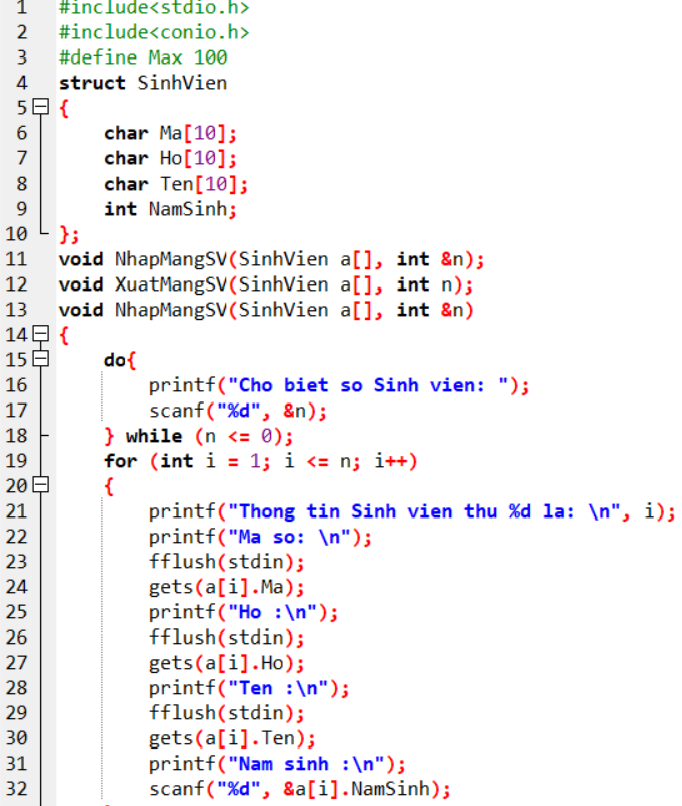
***ví dụ 2.27:***

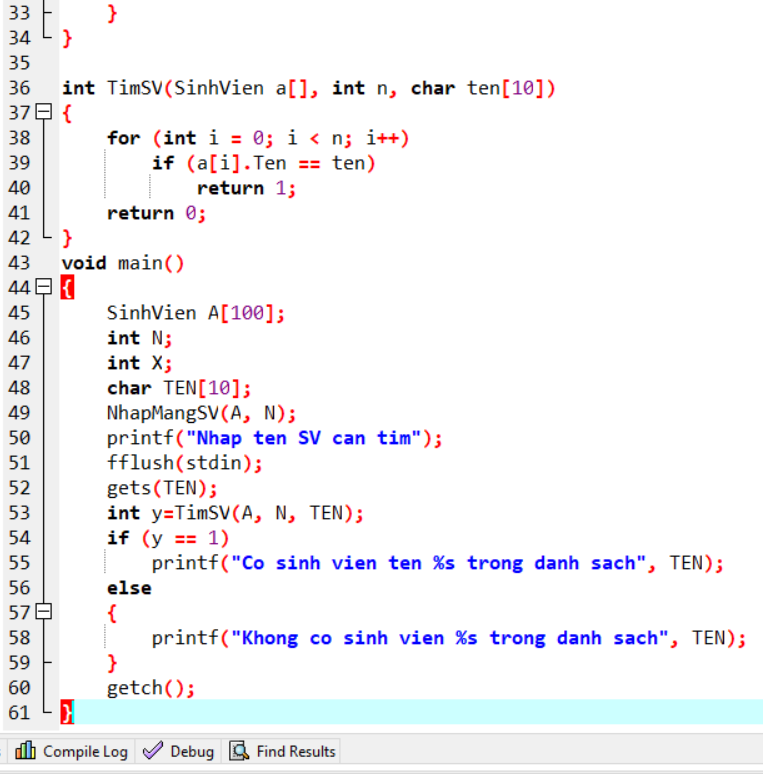


**Kết quả:**

Cho biet so phan tu cua mang: 8  
Gia tri phan tu a[0]=11  
Gia tri phan tu a[1]=4  
Gia tri phan tu a[2]=3  
Gia tri phan tu a[3]=9  
Gia tri phan tu a[4]=8  
Gia tri phan tu a[5]=0  
Gia tri phan tu a[6]=2  
Gia tri phan tu a[7]=45  
Nhap phan tu can tim 8  
8 co trong mang

##### **Code ví dụ về Struct: *Ví dụ 2.27*:**





**b.Thuật toán tìm kiếm nhị phân**

Giải thuật tìm kiếm nhị phân (Binary Search) là gì ?

Binany Search (Tìm kiếm nhị phân) là một giải thuật tìm kiếm nhanh với độ phức tạp thời gian chạy là Ο(log n). Giải thuật tìm kiếm nhị phân làm việc dựa trên nguyên tắc chia để trị (Divide and Conquer). Để giải thuật này có thể làm việc một cách chính xác thì tập dữ liệu nên ở trong dạng đã được sắp xếp.

Binary Search tìm kiếm một phần tử cụ thể bằng cách so sánh phần tử tại vị trí giữa nhất của tập dữ liệu. Nếu tìm thấy kết nối thì chỉ mục của phần tử được trả về. Nếu phần tử cần tìm là lớn hơn giá trị phần tử giữa thì phần tử cần tìm được tìm trong mảng con nằm ở bên phải phần tử giữa; nếu không thì sẽ tìm ở trong mảng con nằm ở bên trái phần tử giữa. Tiến trình sẽ tiếp tục như vậy trên mảng con cho tới khi tìm hết mọi phần tử trên mảng con này.

Cách Binary Search làm việc:

Để Binary Search làm việc thì mảng phải cần được sắp xếp. Để tiện cho việc theo dõi, mình sẽ cung cấp thêm các hình minh họa tương ứng với mỗi bước.

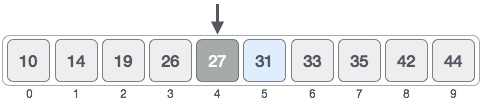
Giả sử chúng ta cần tìm vị trí của giá trị 31 trong một mảng bao gồm các giá trị như hình dưới đây bởi sử dụng Binary Search:



Đầu tiên, chúng ta chia mảng thành hai nửa theo phép toán sau:

chỉ-mục-giữa = ban-đầu + (cuối + ban-đầu)/ 2

Với ví dụ trên là 0 + (9 – 0)/ 2 = 4 (giá trị là 4.5). Do đó 4 là chỉ mục giữa của mảng.



Bây giờ chúng ta so sánh giá trị phần tử giữa với phần tử cần tìm. Giá trị phần tử giữa là 27 và phần tử cần tìm là 31, do đó là không kết nối. Bởi vì giá trị cần tìm là lớn hơn nên phần tử cần tìm sẽ nằm ở mảng con bên phải phần tử giữa.

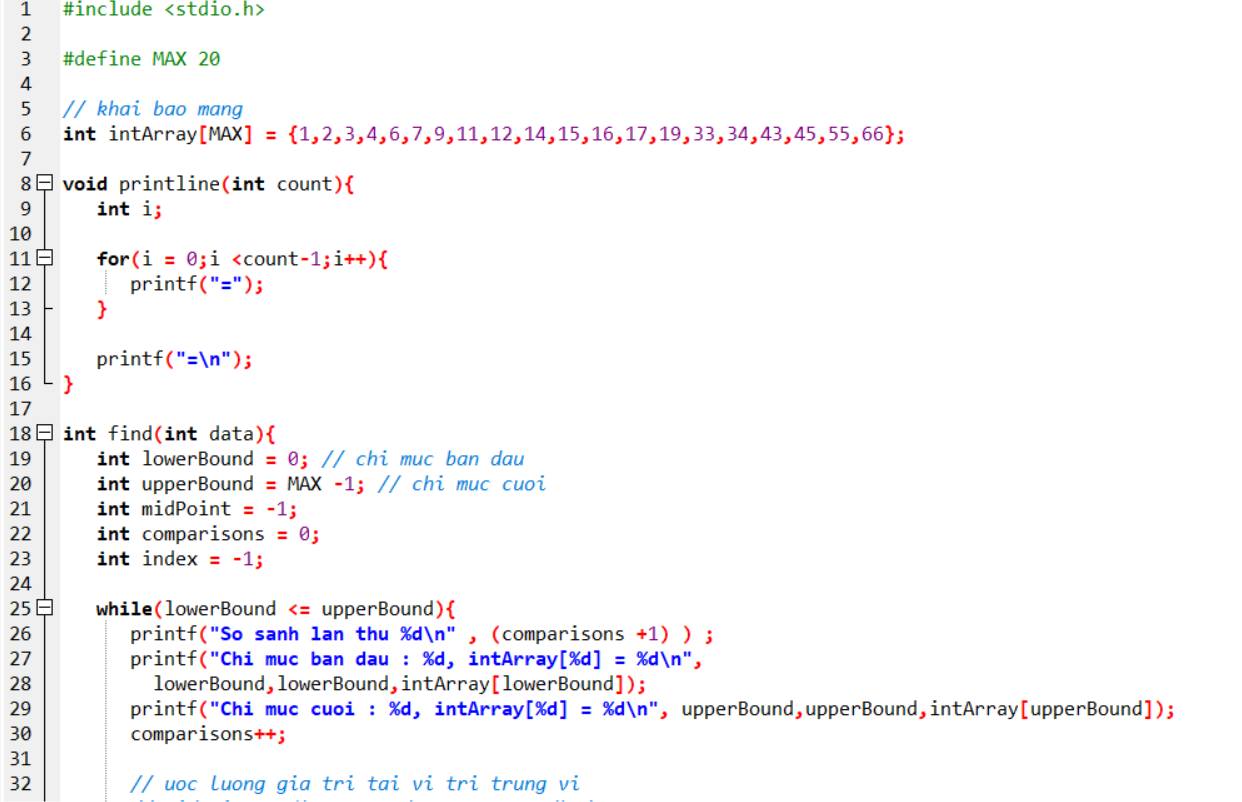


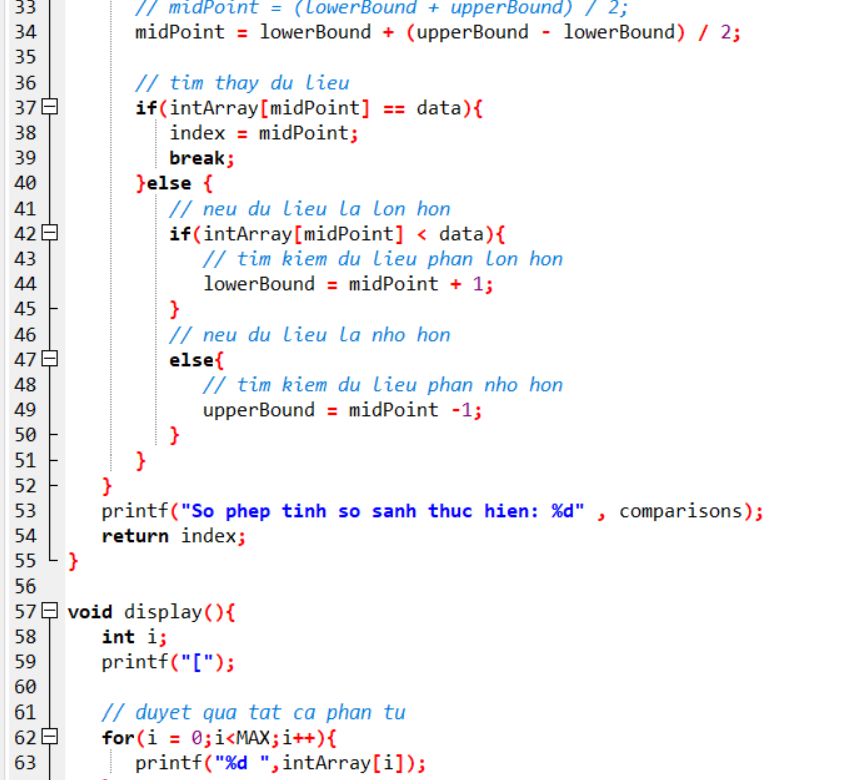
Chúng ta thay đổi giá trị ban-đầu thành chỉ-mục-giữa + 1 và lại tiếp tục tìm kiếm giá trị chỉ-mục-giữa.

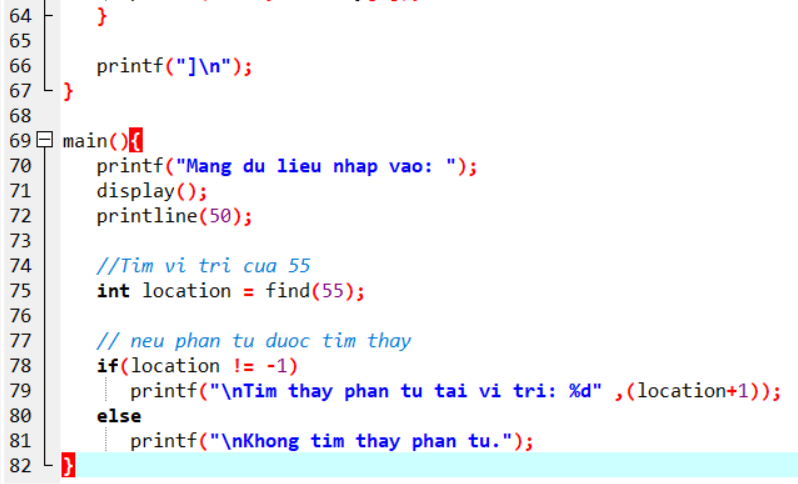
ban-đầu = chỉ-mục-giữa + 1

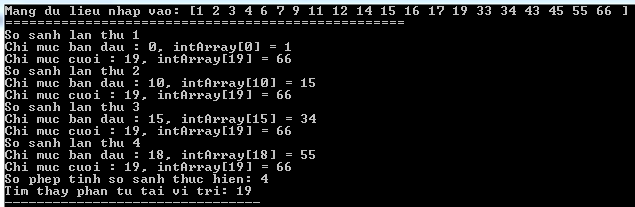
chỉ-mục-giữa = ban-đầu + (cuối + ban-đầu)/ 2

Chương trình minh họa Tìm kiếm nhị phân (Binary Search) trong C

*Ví* *dụ 2.28:*







**CHƯƠNG III :XỬ LÝ CHUYỂN ĐỘNG MÀU SẮC ĐIỀU KHIỂN CƠ BẢN TRONG LẬP TRÌNH**

**3.1 Định nghĩa các hàm**

**3.1.1.Hàm putchar:** Đưa một kí tự ra stdout.

+Dạng hàm:

int putchar(int ch);

+Đối: ch chứa mã của kí tự cần xuất

+Công dụng: Đưa kí tự ch lên stdout. Khi thành công hàm trả về kí tự được xuất chính là ch . Khi có lỗi hàm cho EOF.

+ Ví dụ:

Putchar(‘A’);

Sẽ in chữ A lên amnf hình tại vị trí hiện tại của con trỏ.

Putchar(7);

Sẽ gõ một tiếng chuông.

**3.1.2. Hàm getch:** Nhận một kí tự từ bộ đệm bàn phím, cho hiện lên màn hình

+Dạng hàm:

int getch(void)

+Công dụng:

Nếu có sẵn kí tự trong bộ đệm bàn phím, thì hàm nhận một kí tự trong đó.

Nếu bộ đệm rỗng, thì máy tạm dừng. Khi gõ một kí tự thì hàm nhận ngay được kí tự đó. Kí tự vừa gõ không được hiện lên màn hình.

Hàm trả về kí tự nhận được.

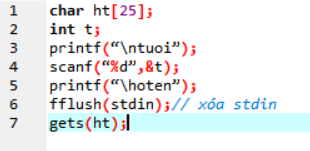
**Làm sạch stdin**

Có thể làm sạch stdin bằng cách dùng hàm fflush như sau:

fflush(stdin);

Dùng hàm này sẽ tránh được mọi hậu quả của các thao tác nhập số liệu trước đó. Ví dụ để dùng hàm gets sau scanf ta có thể làm như sau:

Ví dụ 3.1:



**3.1.3.Hàm kbhit:**

Kiểm tra bộ đệm bàn phím

+Dạng hàm:

int kbhit(void);

+Công dụng:Hàm có giá trị khác không nếu bộ đệm bàn phím khác rỗng, có giá trị 0 nếu trái lại.

+Chú ý: Nếu bấm một phím thì kí tự tương ứng có thể bị gửi vào stdin hay vào bộ đệm. Làm sao để phân biệt được?

Cách phân biệt như sau:

Nếu gõ phím khi máy dừng chờ trong các hàm scanf, gets, và getchar thì kí tự được gửi vào stdin.

Gõ phím trong các trường hợp khác thì ksi tự gửi vào bộ đệm.

Hàm kbhit là một hàm không chính quy.

**3.2. Một số thư viện chuẩn trong C:**

* + - **stdio.h**: Tập tin định nghĩa các hàm vào/ra chuẩn (standard input/output). Gồm các hàm in dữ liệu (printf()), nhập giá trị cho biến (scanf()), nhận ký tự từ bàn phím (getc()), in ký tự ra màn hình (putc()), nhận một dãy ký tự từ bàm phím (gets()), in chuỗi ký tự ra màn hình (puts()), xóa vùng đệm bàn phím (fflush()), fopen(), fclose(), fread(), fwrite(), getchar(), putchar(), getw(), putw()…
    - **conio.h** : Tập tin định nghĩa các hàm vào ra trong chế độ DOS (DOS console). Gồm các hàm clrscr(), getch(), getche(), getpass(), cgets(), cputs(), putch(), clreol(),…

**-<time.h>**: Để chuyển đổi giữa các định dạng khác nhau về thì giờ và ngày tháng.

-**<stdlib.h>**: Dùng để xúc tiến nhiều phép toán, bao gồm sự chuyển đổi, các số giả ngầu nhiên, cấp phát vùng nhớ, kiểm soát quá trình, môi trường, tín hiệu, tìm kiếm, và xếp thứ tự.

**3.3. Xóa màn hình**

- Để xóa màn hình at dùng hàm:

clrscr();

Hoặc system(“cls”);

**3.4. Tọa độ trong lập trình**

Ta có: Đi từ trái sang phải thì hoành độ tăng dần và ngược lại

Đi từ trên xuống dưới thì tung độ tăng dần và ngược lại.

**3.5. Di chuyển con trỏ đến tọa độ như ý muốn**

-Để di chuyển con trỏ( màn hình ) đến vị trí (x,y) ta dùng hàm

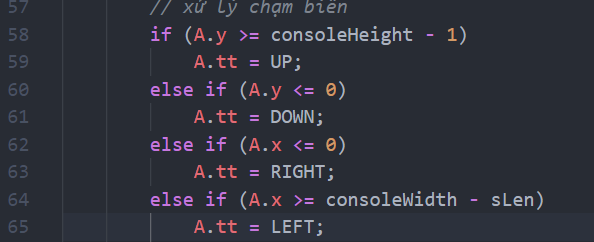
gotoxy(x,y);

Ở đây x là số hiệu cột nhận giá trị từ 1 đến 80, y là dòng có giá trị từ 1 đến 25.

**3.6. Xử lý chạm biên**

Khi đối tượng chạm biên trên, dưới, trái, phải đối tượng sẽ không thoát khỏi màn hình hiển thị.

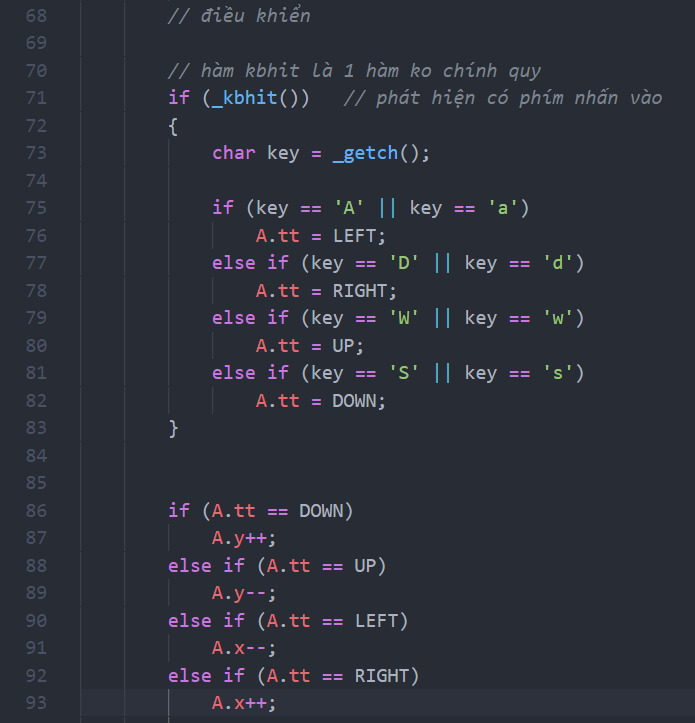
*Ví dụ 3.2:*



**3.7. Điều khiển**

Ý tưởng: điều khiển đối tượng duy chuyển bằng cách ta dùng hàm kbhit để điều khiển đối tượng. Nhấn qua trái A hoặc a và qua phải B hoặc b và W hoặc W để lên trên hoặc S hoặc s để xuống dưới.

*Ví dụ 3.3:*



**3.8. Hiển thị màu**

Để thay màu, chúng ta khai báo thư viện #include<stdio.h> và Sau đây là 15 màu cơ bản:

ColorCode\_DarkBlue 1

ColorCode\_DarkGreen 2

ColorCode\_DarkCyan 3

ColorCode\_DarkRed 4

ColorCode\_DarkPink 5

ColorCode\_DarkYellow 6

ColorCode\_DarkWhite 7

ColorCode\_Grey 8

ColorCode\_Blue 9

ColorCode\_Green 10

ColorCode\_Cyan 11

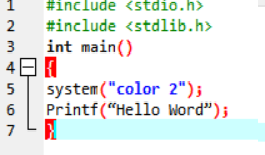
ColorCode\_Red 12

ColorCode\_Pink 13

ColorCode\_Yellow 14

ColorCode\_White 15

*Ví dụ 3.4:*



**Chương 4: MÔ TẢ THIẾT KẾ GAME**

**4.1. Ý tưởng**

Ý tưởng: Gia đình ông Join trong một chuyến đi du lịch đã vô tình lạc vào một con đường bí ẩn. Tại đây, gia đình ông Join gặp rất nhiều vật thể lạ đang tiến về hướng xe của gia đình mình, để bảo vệ gia đình thoát khỏi nguy hiểm ông Join phải điều khiển xe tránh những vật thể đó một cách an toàn.

Khi chơi game bạn sẽ hóa thân thành Join, thể hiện trình độ lái xe đẳng cấp, phản xạ thần tốc của mình để vượt qua các chướng ngại vật. Để xem bạn có thể ghi được bao nhiêu điểm nhé **.**

**4.2. Thiết kế game**

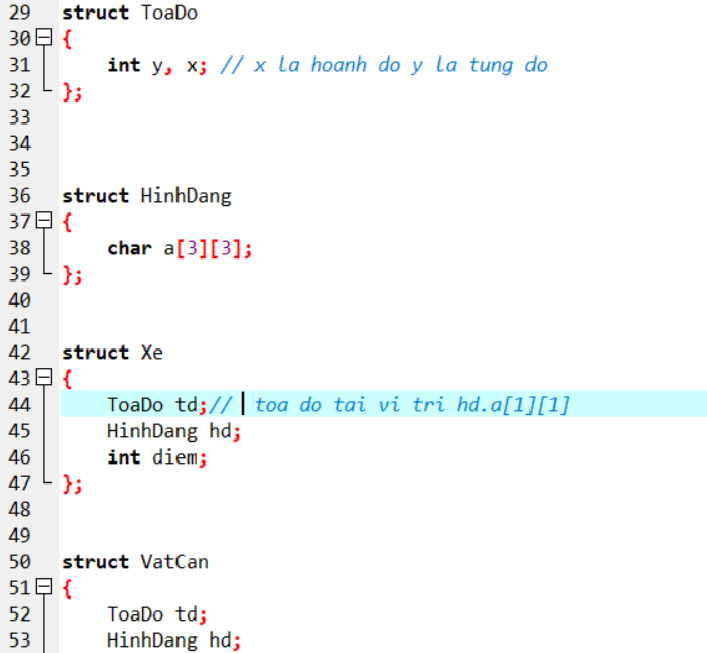
**Ta sử dụng struct để quản lý một cách dễ dàng hơn:**

**Đầu tiên tạo** struct ToaDo. Sau đó đến struct Xe ta sẽ thiết kế xe và thiết kế vật cản vì vậy tạo một Struct HinhDang để dễ thực hiện hơn, ở Struct HinhDang ta sẽ dùng mảng 2 chiều với 3x3(3 hàng và 3 cột) để tạo hình dạng cho xe và vật cản.

+struct Xe

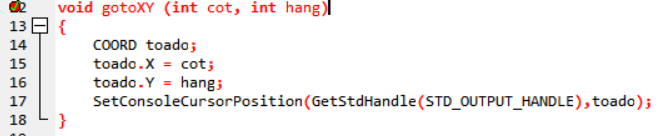
+struct VatCan

*Ví dụ 4.1:*



* Gồm có các hàm trong chương trình:

+void gotoXY (int cot, int hang): dùng để gọi tọa độ chính xác mà vị trí xe hiển thị trên màn hình.

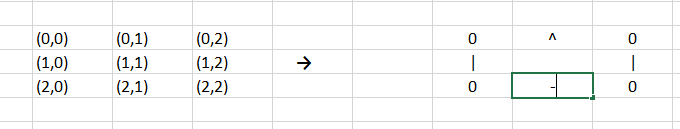


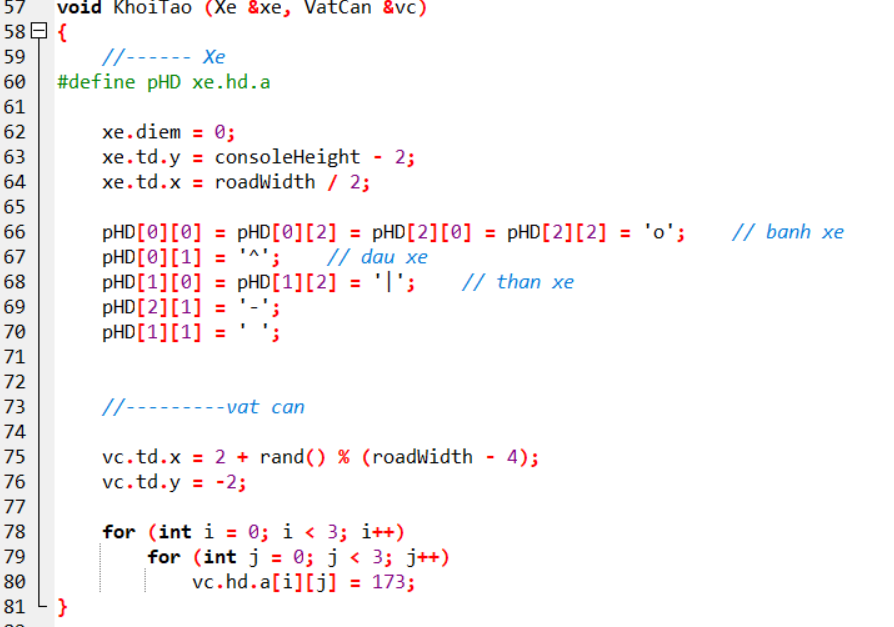
+void KhoiTao (Xe &xe, VatCan &vc)

Ta sử dụng mảng 2 chiều để tạo thân xe và tạo vật cản

Gốc tọa độ tại vị trí a.[1][1]

*Ví dụ 4.2: Minh họa tọa độ thân xe*





+void HienThi (Xe xe, VatCan vc): dùng để hiển thị xe,đường đi, vật cản ra màn hình.

Tạo ra 2 biên trái phải đề giới hạn vùng đi của xe

Hiển thị xe

Hiển thị vật cản

Hiển thị điểm

*for (int kDong = -1; kDong <= 1; kDong++)*

*for (int kCot = -1; kCot <= 1; kCot++)*

*{*

*int x = kCot + xe.td.x;*

*int y = kDong + xe.td.y;*

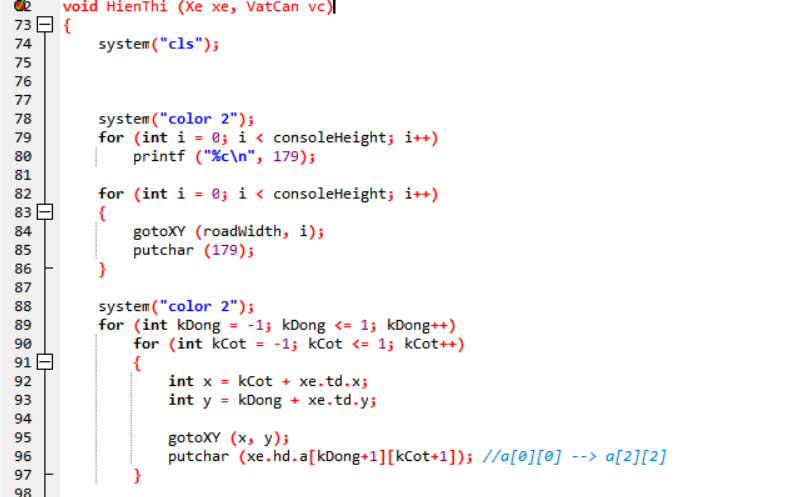
*gotoXY (x, y);*

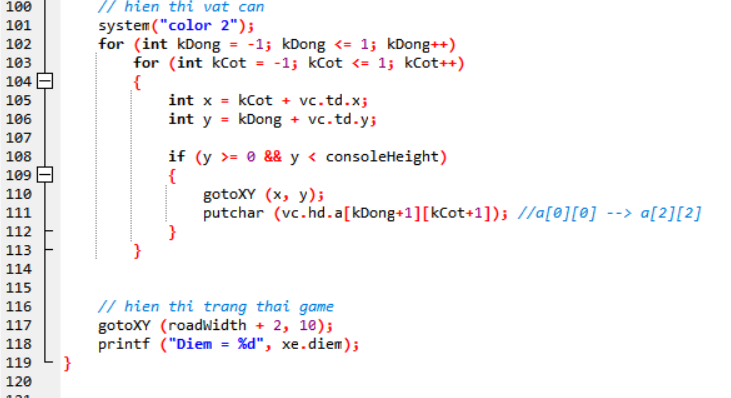
*putchar (xe.hd.a[kDong+1][kCot+1]); //a[0][0] --> a[2][2]*

*}*

Vòng lặp này dùng để tạo ra tọa độ tương đối chính xác để in ra

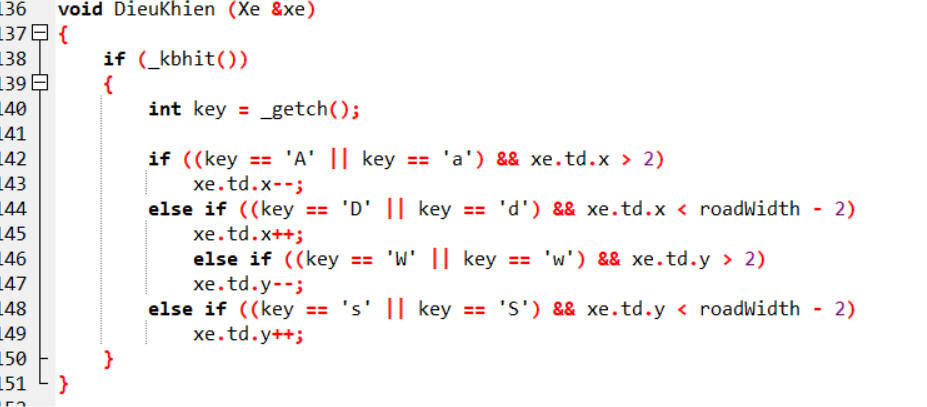
*Ví dụ 4.3:*



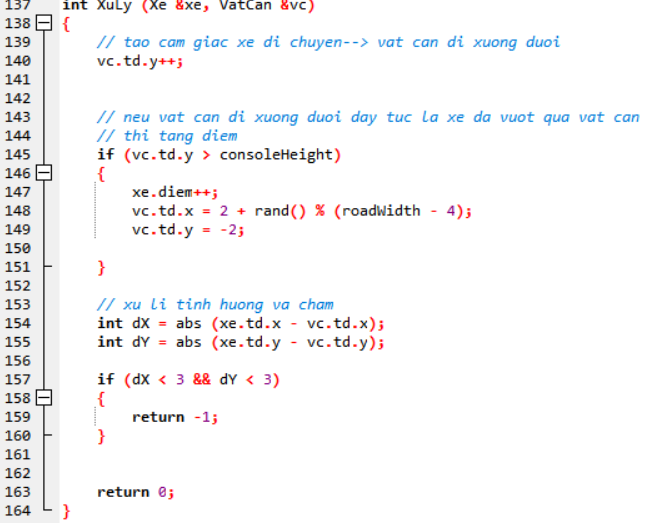


+void DieuKhien (Xe &xe): dùng để thiết lập các phím điều khiển xe. Sử dụng hàm kbhit để truyền từ bàn phím vào điều khiển xe theo lệnh của bàn phím và xử lí xe di chuyển nằm trong giới hạn 2 biên trái phải, trên dưới.

*Ví dụ 4.4:*



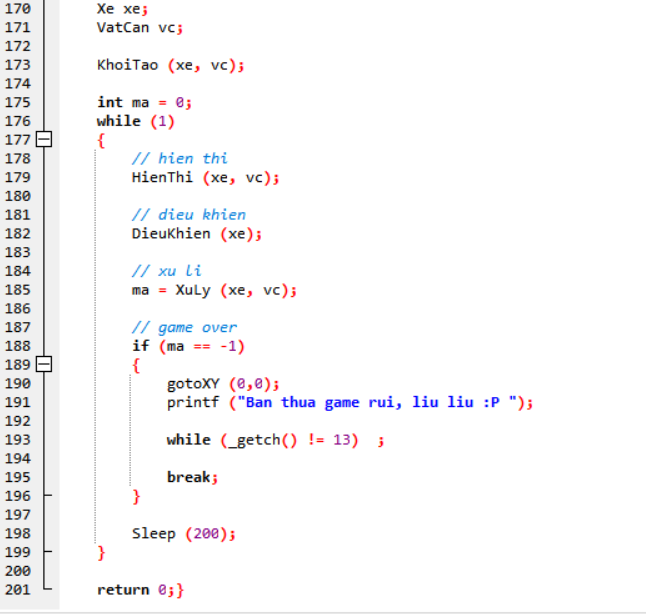
+int XuLy (Xe &xe, VatCan &vc): dùng để xử lí khi xe chạm biên, và khi chạm vật cản.



+Các cấu trúc:Hình dạng, Xe,Vật cản: dùng để thiết lập hình dạng xe và vật cản.

+Hàm main:

*Ví dụ 4.5:*



**4.3. Hướng dẫn sử dụng chương trình**

Khi vật cản tiến gần đến xe của bạn. Bạn hãy dùng phím “A” để di chuyển sang trái hoặc phím “D” duy chuyển sang phải hoặc phím “W” để di chuyển lên trên hoặc “S” để di chuyển xuống dưới để tránh các chướng ngại vật. Mỗi lần tránh được ,khi vật cản biến mất khỏi màn hình thì sẽ tăng 1 điểm dành cho bạn.

**Tài liệu tham khảo:**

**\*Sách:**

1. Phạm Văn Ất, **Kỹ thuật lập trình C cơ sở và nâng cao**,Nhà xuất bản Giao thông Vận tải, 1996.

2.Phạm Văn Ất, **Lập trình nâng cao**, Nhà xuất bản Giao thông Vận tải, 2015.

3.TS. Nguyễn Văn Long –THS.Hoàng Văn Thông, **Cấu trúc dữ liệu và giải thuật,** Nhà xuất bản Giao thông Vận tải, 2006.

## 4. Nhiều tác giả, Giáo trình kỹ thuật lập trình C căn bản và nâng cao , Nhà xuất bản Bách Khoa Hà Nội

**5.** Ngô Trung Việt**, Ngôn ngữ lập trình C và C++,** Nhà xuất bảnThống Kê

## 6. Dương Thăng Long - Trương Tiến Tùng,Kỹ thuật lập trình cơ sở với ngôn ngữ C/C++, Nhà xuất bản Khoa học và Kỹ thuật

**\*Website:**

**-** Kênh youtube: Nguyễn Không Tên – Video: Phần 1- Hiệu ứng động [Lập trình game cơ bản].

Link: <https://www.youtube.com/watch?v=C7yokRqdd4o&t=753s>

Đây là trang lấy ý tưởng cho game ở trên.

-Giải thuật tìm kiếm tuyến tính – Trang: Sinhvientot.net - Nguyễn Thanh Tuấn.

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

-Bài 50: Thuật toán và tìm kiếm nhị phân – Trang: nguyenvanhieu.vn - Lập trình không khó – Nguyễn Văn Hiếu.

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

-Bài 48: Thuật toán sắp xếp chọn(Selection sort) – Trang: nguyenvanhieu.vn - Lập trình không khó – Nguyễn Văn Hiếu.

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

-Sắp xếp struct trong C – Trang: viẹtack.com.

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