TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI PHÂN HIỆU THÀNH PHỐ HỒ CHÍ MINH



**BÁO CÁO BÀI TẬP LỚN**

**MÔN**: LẬP TRÌNH NÂNG CAO

**CHỦ ĐỀ**:GAME CARO 9 Ô

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | **Trần Thị Dung** |
| **Sinh viên: ◘Đỗ Hữu Khang**  **◘Bùi Văn Tân** | **MSSV: ◙ 6051071055**  **◙ 6051071104** |
|  |
|  |  |

|  |  |
| --- | --- |
| **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 15 tháng 6 năm 2020  Nhóm sinh viên thực hiện:  Đỗ Hữu Khang  Bùi Văn Tân |

**MỤC LỤC**

**LỜI CẢM ƠN**................................................................................................................

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**..........................................................

**MỤC LỤC**......................................................................................................................

**HÌNH ẢNH MINH HỌA**

**PHẦN 1:TỔNG QUAN**.................................................................................................1

I. Lý do chọn đề tài..........................................................................................................1

II. Đối tượng nghiên cứu..................................................................................................1

**PHẦN 2:CƠ SỞ LÝ THUYẾT**......................................................................................1

**I. Giới thiệu**

2.1.1.Giới thiệu về ngôn ngữ lập trình C.........................................................................2

2.1.2.Đặc điểm của ngôn ngữ lập trình C.........................................................................2

2.1.3.Cấu trúc của ngôn ngữ lập trình C...........................................................................3

**II. Làm việc với tệp**.........................................................................................................5

2.2.1.File\_văn\_bảnText\_file.........................................................................................................................................................................................................................................5

2.2.2.File\_nhị\_phân\_Binary\_file..................................................................................................................................................................................................................................5

**III. Danh sách liên kết đơn**............................................................................................8

2.3.1.Danh sách liên kết đơn là gì....................................................................................9

2.3.2.Cài\_đặt\_danh\_sách\_liên\_kết\_đơn.....................................................................................................................................................................................................................12

**IV. Các thuật toán sắp xếp và thuật toán tìm kiếm**................................................................................................................................18

2.4.1.Các\_thuật\_toán\_sắp\_xếp...................................................................................................................................................................................................................................18

2.4.2.Các thuật toán tìm kiếm.........................................................................................19

**PHẦN 3: MÔ TẢ THIẾT KẾ GAME**........................................................................30

3.1.Ý tưởng.....................................................................................................................30

3.2. Giải thích code........................................................................................................ 32

Hình

Hình 1, 2 -----------------------------------------------------------------trang 10

Hình 3---------------------------------------------------------------------trang 13

Hình 4---------------------------------------------------------------------trang 15

Hình 5---------------------------------------------------------------------trang 20

Hình 6---------------------------------------------------------------------trang 22

Hình 7, 8, 9---------------------------------------------------------------trang 27

Hình 10-------------------------------------------------------------------trang 32

Hình 11, 12---------------------------------------------------------------trang 36

Hình 13, 14---------------------------------------------------------------trang 37

Hình 15, 16---------------------------------------------------------------trang 38

**TÀI LIỆU THAM KHẢO:**

<https://voer.edu.vn/c/tap-tin-file/16cbe2bb/20bc3d33?fbclid=IwAR3Wp9K3tSisrVVOkMFiQyRe_M6S_tw4XxJmG2n2ZuYuj-kVIIw5KV6EvY0>

<https://cpp.daynhauhoc.com/10/0-file-va-cac-thao-tac-co-ban-voi-file-trong-c/?fbclid=IwAR0UVnNSJLjmAdXZWMFRHit3dQ2ljiZS4zzoH60rg-rQIywmU4ZZ92APuAo>

<https://viettuts.vn/c-file/ham-fscanf-va-fprintf-doc-va-ghi-file-trong-c?fbclid=IwAR18L_QVWBKUhDiLIV_Hs9T0oz14JiC23I42AZWsYEnsCUMdCIH8nlBVego>

<https://www.youtube.com/watch?v=mSOU_7krE8k&t=795s&fbclid=IwAR1cFfEfoeUcKwxYjcHkTbhPiV6PmtouRfW-9_VwgHYrCJeERCCP3mWCebw>

<https://www.it-swarm-vi.tech/vi/c++/lam-cach-nao-de-thay-doi-mau-van-ban-va-mau-bang-dieu-khien-trong-ma-khoi/1052247021/?fbclid=IwAR2GmsoanwJvTfPquRgcyEyV_Ni_moOkL9GkWyyY3nCf4Eol6J8l5RSSAbU>

**PHẦN 1:TỔNG QUAN**

**1.1.Lý do chọn đề tài:**

Xã hội phát triển, công việc và học tập ngày càng gây nhiều áp lực trong cuộc sống chúng ta. Do đó nhu cầu giải trí là một nhu cầu thiết yếu tỏng cuộc sống của con người. Giải trí là một lĩnh vực thật sự rộng lớn bao gồm cả thơ ca, nhạc họa, và nhiều hình thức khác nữa được con người sáng tạo ra hàng ngày hàng giờ, và theo dòng phát triển của công nghệ máy tính và công nghệ phần mềm. Hiện nay hình thức giải trí bằng trò chơi điện tử là xu thế và là một trong một ngành đem lại lợi nhuận cao nhất trên thị trường. Xong thiết kế game là một việc rất khó khăn và đòi hỏi tính sáng tạo. Chỉ dùng những dòng mã mà lại tạo ra những trò chơi hấp dẫn và đem lại sự thư giãn cho mọi người.

**1.2.Đối tượng nghiên cứu:**

Lập trình game caro 9 ô bằng ngôn ngữ C.

**1.1.Lý do chọn đề tài:**

Xã hội phát triển, công việc và học tập ngày càng gây nhiều áp lực trong cuộc sống chúng ta. Do đó nhu cầu giải trí là một nhu cầu thiết yếu tỏng cuộc sống của con người. Giải trí là một lĩnh vực thật sự rộng lớn bao gồm cả thơ ca, nhạc họa, và nhiều hình thức khác nữa được con người sáng tạo ra hàng ngày hàng giờ, và theo dòng phát triển của công nghệ máy tính và công nghệ phần mềm. Hiện nay hình thức giải trí bằng trò chơi điện tử là xu thế và là một trong một ngành đem lại lợi nhuận cao nhất trên thị trường. Xong thiết kế game là một việc rất khó khăn và đòi hỏi tính sáng tạo. Chỉ dùng những dòng mã mà lại tạo ra những trò chơi hấp dẫn và đem lại sự thư giãn cho mọi người.

**1.2.Đối tượng nghiên cứu:**

Lập trình game caro 9 ô bằng ngôn ngữ C.

**PHẦN 2:CƠ SỞ LÝ THUYẾT**

**I. 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 .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.

Tinh 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.

**2.1.3.Cấu trúc của ngôn ngữ lập trình c**

a. Biến

Một **biến trong C** là tên của vị trí bộ nhớ. Nó được sử dụng để lưu trữ dữ liệu. Giá trị của nó có thể được thay đổi và nó có thể được sử dụng lại nhiều lần. Mỗi biến trong C có một loại dữ liệu cụ thể, xác định kích thước của bộ nhớ của biến; phạm vi các giá trị có thể được lưu trữ trong bộ nhớ đó. Biến là một cách để thể hiện vị trí bộ nhớ thông qua một cái tên để nó có thể được xác định một cách dễ dàng. Tên của một biến có thể bao gồm các chữ cái, chữ số và ký tự gạch dưới. Nó phải bắt đầu bằng một chữ cái thư hoặc một gạch dưới. Biến trong C có **phân biệt chữ hoa và chữ thường**. Biến được sử dụng để lưu trữ dữ liệu tạm thời trong chương trình và giá trị của biến có thể được thay đối trong khi chạy chương trình.

b. Hàm

Một chương trình được viết trong ngôn ngữ C gọi là một dãy các hàm, trong đó có một hàm chính ( hàm main()). Hàm chia các bài toán lớn thành các công việc nhỏ hơn giúp thực hiện những công việc lặp lại nào đó một cách nhanh chóng mà không phải viết lại đoạn chương trình. Thứ tự các hàm trong chương trình C  là bất kỳ, song chương trình bao giờ cũng đi thực hiện từ hàm main(). Hàm có thể xem là một đơn vị độc lập của chương trình. Hàm trong C có vai trò ngang nhau, vì vậy không có phép xây dựng một hàm bên trong các hàm khác.

Hàm **là một tập hợp các đoạn mã** và nó sẽ thực thi các đoạn mã đó khi gọi hàm ra, nó sẽ được thực thi lại nhiều lần hoặc thực thi trong một trường hợp nhất định. Có nghĩa là khi bạn tạo hàm thì các đoạn mã bên trong đó sẽ không thực thi cho đến khi được gọi ra bên ngoài.

Hàm là một đoạn chương trình bao gồm một hoặc nhiều xử lý nhằm giải quyết một công việc nào đó và được xây dựng với mục đích **tái sử dụng.** Như vậy trong một chương trình nếu có những xử lý giống nhau chúng ta nên xây dựng hàm. Hàm có thể được xây dựng bởi lập trình viên hoặc hàm có sẵn trong ngôn ngữ lập trình. Ví dụ trong lập trình C, chúng ta có thể liệt kê các hàm có sẵn như printf, scanf, pow, sqrt,…

c. Toán tử ba ngôi

Trong [toán học](https://vi.wikipedia.org/wiki/To%C3%A1n_h%E1%BB%8Dc), một phép toán ba ngôi là một [phép toán](https://vi.wikipedia.org/wiki/Ph%C3%A9p_to%C3%A1n) đa ngôi với số ngôi là 3. Một phép toán ba ngôi trên tập hợp A sẽ lấy bất kỳ ba phần tử nào của A và kết hợp chúng thành một phần tử đơn nhất của A. Một ví dụ của một phép toán ba ngôi là tích trong một [đống](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BB%91ng_(to%C3%A1n_h%E1%BB%8Dc)&action=edit&redlink=1). Trong [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh), một toán tử ba ngôi là một [toán tử](https://vi.wikipedia.org/wiki/To%C3%A1n_t%E1%BB%AD_(l%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh)) với ba đối số. Các đối số và kết quả có thể có các loại khác nhau. Nhiều ngôn ngữ lập trình sử dụng [cú pháp ngôn ngữ C](https://vi.wikipedia.org/wiki/C%C3%BA_ph%C3%A1p_ng%C3%B4n_ng%E1%BB%AF_C) cho phép một toán tử ba ngôi, dùng định nghĩa [biểu thức có điều kiện](https://vi.wikipedia.org/w/index.php?title=Bi%E1%BB%83u_th%E1%BB%A9c_c%C3%B3_%C4%91i%E1%BB%81u_ki%E1%BB%87n&action=edit&redlink=1). Vì toán tử này thường là toán tử ba ngôi duy nhất trong ngôn ngữ, đôi khi nó chỉ được gọi là "toán tử ba ngôi". Trong một số ngôn ngữ, toán tử này được gọi là "toán tử điều kiện".

d. Mảng hai chiều

Mảng hai chiều là một bảng các phần tử cùng kiểu, mảng cũng được đặt tên như mảng một chiều, các phần tử cũng được đánh chỉ số để quản lý. Tuy nhiên, mỗi phần tử của mảng hai chiều được xác định bằng bằng ha**i** chỉ số, chúng ta tạm gọi là chí số dòng và chỉ số cột.

Mảng hai chiều (còn gọi là ma trận) thực chất là mảng một chiều trong đó mỗi phần tử của mảng là một mảng một chiều, và được truy xuất bởi hai chỉ số dòng và cột. Mảng 2 chiều, 3 chiều hoặc n chiều được gọi là mảng đa chiều trong C.

**Mảng 2 chiều trong C** được biểu diễn dưới dạng hàng và cột, còn được gọi là ma trận. Nó còn được gọi là mảng các mảng hoặc danh sách các mảng

Mảng 2 chiều, 3 chiều hoặc n chiều được gọi là mảng đa chiều trong C.

Khai báo mảng hai chiều

|  |  |
| --- | --- |
|  | data\_type array\_name[size1][size2]; |

Ví dụ:

|  |  |
| --- | --- |
|  | int ma\_tran [4][3]; |

Ở đây, 4 là số **hàng** và 3 là số **cột**

e. switch case

**Lệnh switch case** là một cấu trúc điều khiển & rẽ nhánh hoàn toàn có thể được thay thế bằng [cấu trúc if else](https://nguyenvanhieu.vn/cau-lenh-if-else-trong-c/). Tuy nhiên, việc sử dụng switch case sẽ giúp code của chúng ta dễ viết và dễ đọc hơn; Một điều nữa là sử dụng switch case có vẻ như cho hiệu năng tốt hơn so với sử dụng if else.

**Mệnh đề switch trong C** cho phép một biến được kiểm tra xem giá trị của nó có bằng một giá trị trong một danh sách hay không. Mỗi giá trị được gọi là một trường hợp (case).

Bạn có thể khai báo bất kỳ số lượng lệnh case bên trong một switch. Đẳng sau từ khóa case là một giá trị được sử dụng để so sánh và một dấu hai chấm (:).

f. do…While

Cấu trúc do … while thường được sử dụng để kiểm tra các giá trị đầu vào (thường là các giá trị được nhập từ bàn phím). Chương trình sau kiểm tra dữ liệu nhập vào có thể là một tháng trong năm hay không: int main()

ví dụ:

Nhập vào dãy số nguyên cho tới khi người dùng nhập số 0 thì dừng. Trong ví dụ, công việc nhập số nguyên được lặp đi lặp lại nhiều lần nhưng không biết là lặp bao nhiêu lần. Vòng lặp chỉ dừng khi người dùng nhập số 0.

g. if else

Bản chất của câu lệnh if (nếu) là câu điều kiện, nó giúp cho ta kiểm tra một điều kiện do ta đặt ra sau đó thực thi nếu điều kiện đó đúng hoặc sai. Else (ngược lại) là câu lệnh thực thi khi câu lệnh if trên sai. else if sử dụng để thêm điều kiện khác nếu lệnh if ở trên sai.

Câu lệnh điều kiện if else được sử dụng để thực hiện các hành động khác nhau dựa theo các điều kiện khác nhau. Các điều kiện được sử dụng thường là các [toán tử so sánh](https://hocwebchuan.com/tutorial/php/php_operators.php#comparison). Khi muốn kết hợp nhiều điều kiện khác nhau, chúng ta thường dùng kết hợp [toán tử so sánh](https://hocwebchuan.com/tutorial/php/php_operators.php#comparison) và [toán tử logic](https://hocwebchuan.com/tutorial/php/php_operators.php#logic)

II. 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

**2.2.1. 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.

**2.2.2. 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.

Các thao tác với file

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

1. Tạo mới một file
2. Mở một file đã có
3. Đóng file đang mở
4. Đọc thông tin từ file/ Ghi thông tin ra file

Thao tác với file trên ngôn ngữ C

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

Thao tác mở file

Để đọc ghi file trong C cũng như trong mọi ngôn ngữ lập trình, việc đầu tiên bạn cần làm là mở file mà bạn muốn làm việc. Trong ngôn ngữ lập trình C, chúng ta có thể mở file bằng cách sử dụng hàm

Ví dụ:

**FILE**\* fopen(**const** **char** \***file**, **const** **char** \*mode);

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

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 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. 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:

int fclose(file \*fp)

Đọc/Ghi file văn bản trong C

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

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

VD1. Ghi file sử dụng fprintf()

Hàm fprintf() trong C được sử dụng để ghi các ký tự vào file. Nó gửi dữ liệu được định dạng tới một stream.

Cú pháp

|  |  |
| --- | --- |
|  | int fprintf(FILE \*stream, const char \*format [, argument, ...]) |

Ví dụ

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #include <stdio.h>    main() {     file \*fp;     // mo file     fp = fopen("file.txt", "w");     // ghi du lieu vao file     fprintf(fp, "Hello C, su dung fprintf()...\n");     // dong file     fclose(fp);  } |

#### VD2. Đọc file sử dụng fscanf()

Hàm fscanf () được sử dụng để đọc tập hợp các ký tự từ file. Nó đọc một từ trong file và trả về EOF ở vị trí kết thúc file.

Cú pháp

|  |  |
| --- | --- |
|  | int fscanf(file \*stream, const char \*format [, argument, ...]) |

Ví dụ 1

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #include <stdio.h>    main() {     FILE \*fp;     char buff[255];     fp = fopen("file.txt", "r");     while (fscanf(fp, "%s", buff) != EOF) {        printf("%s ", buff);     }     fclose(fp);  } |

#### 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.

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

|  |
| --- |
|  |

#### Đọc dữ liệu văn bản từ file

Sau khi đã có file rồi thì chúng ta tiến hành đọc file lên và xử lý.

Ví dụ

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | #include<stido.h>  void docfile(int a[],int &n)  {      file \*f;      f=fopen("bcdonline.text","rt");      fscanf(f,"%d",&n);      for(int i=1;i<=n;i++)          fscanf(f,"%d",&a[i]);      fclose(f);  } |

Giải thích

Cũng như nói ở trên có các phần xử lý đặt trưng khi tương tác với FILE.

Tham số “rt” giúp bạn đọc dữ liệu từ file

Đọc file với lệnh fscanf();

Sau đó đóng file fclose();

**III. 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.

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 | 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.

**2.3.1. 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 1

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



Hình 2

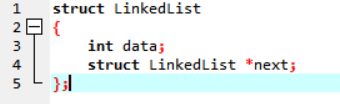
Danh sách các kiểu danh sách liên kết:

* 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.

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

a. 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.

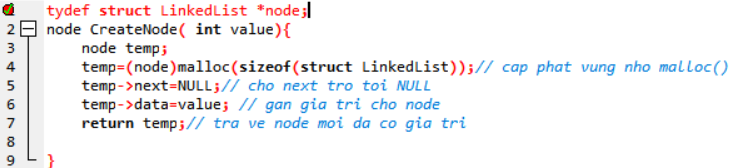


Hình 3

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, **next là kiểu LinkedList của chính nó**Bởi vì nó là con trỏ trỏ của chính bản thân nó, và nó trỏ tới một thằng Node kế tiếp cũng có kiểu LinkedList.

b. Tạo mới 1 Node

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



Mỗi một Node khi được khởi tạo, chúng ta cần cấp phát bộ nhớ cho nó, và mặc định cho con trỏ next trỏ tới NULL. Giá trị của Node sẽ được cung cấp khi thêm Node vào linked list.

* **typedef** được dùng để định nghĩa một kiểu dữ liệu trong C. VD: typeder long long LL;
* **malloc** là hàm cấp phát bộ nhớ của C. Với C++ chúng ta dùng new
* **sizeof** là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm malloc

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

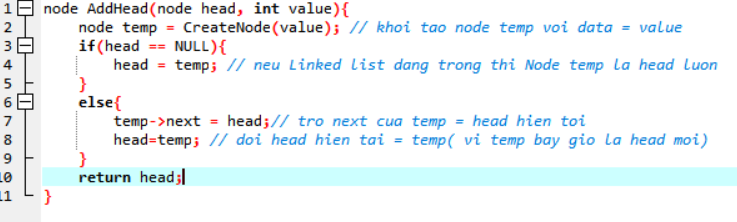
### c. Thêm Node vào danh sách liên kết

#### Thêm vào đầu

Việc thêm vào đầu chính là việc cập nhật lại thằng head. Ta gọi Node mới(temp), ta có. Nếu head đang trỏ tới NULL, nghĩa là linked list đang trống, Node mới thêm vào sẽ làm head luôn. Ngược lại, ta phải thay thế thằng head cũ bằng head mới. Việc này phải làm theo thứ tự như sau

Cho next của temp trỏ tới head hiện hành

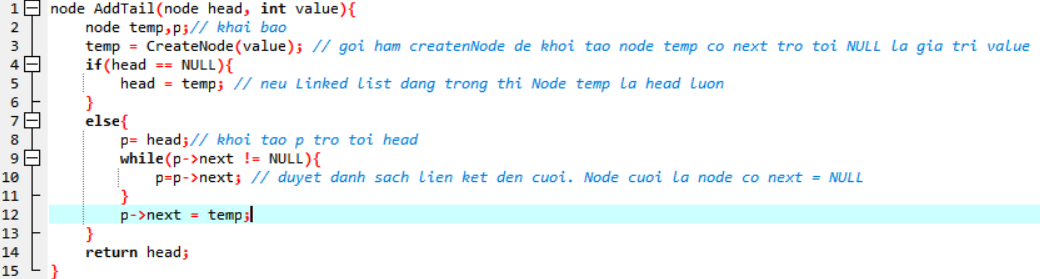
Đặt temp làm head mới



#### Thêm vào cuối

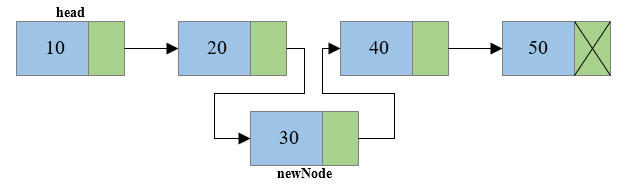
Chúng ta sẽ cần Node đầu tiên, và giá trị muốn thêm. Khi đó, ta sẽ:

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



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

#### Thêm vào vị trí bất kỳ



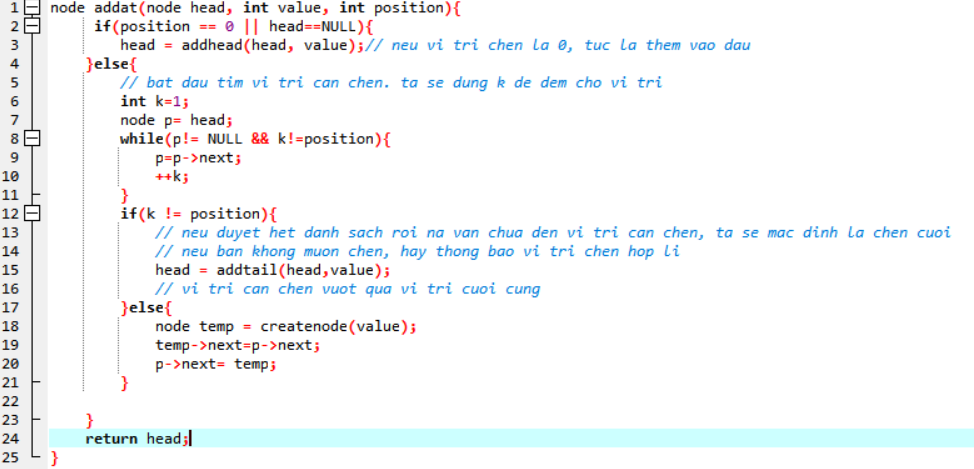
Hình 3

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

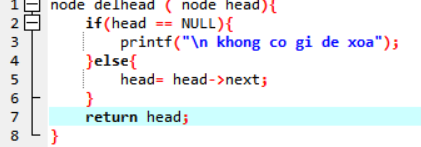


Lưu ý: Bạn phải làm theo thứ tự trên, nếu bạn cho p->next = temp trước. Khi đó, bạn sẽ không thể lấy lại phần sau của danh sách liên kết nữa(Vì next chỉ được được lưu trong p->next mà thay đổi p->next rồi thì còn đâu giá trị cũ).

### d. Xóa Node khỏi danh sách liên kết

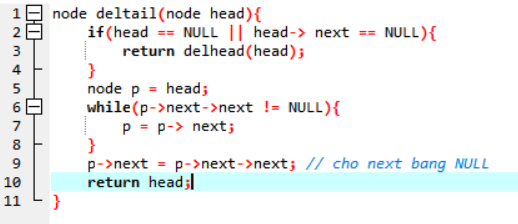
#### Xóa đầu

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



#### Xóa cuối

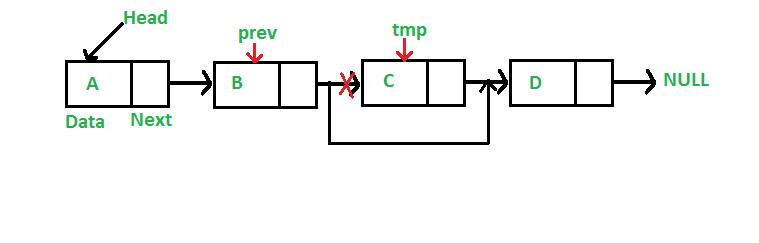
Xóa cuối mới nhọc nè, nhọc ở chỗ phải duyệt đến thằng cuối – 1, cho next của cuối – 1 đó bằng NULL.



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

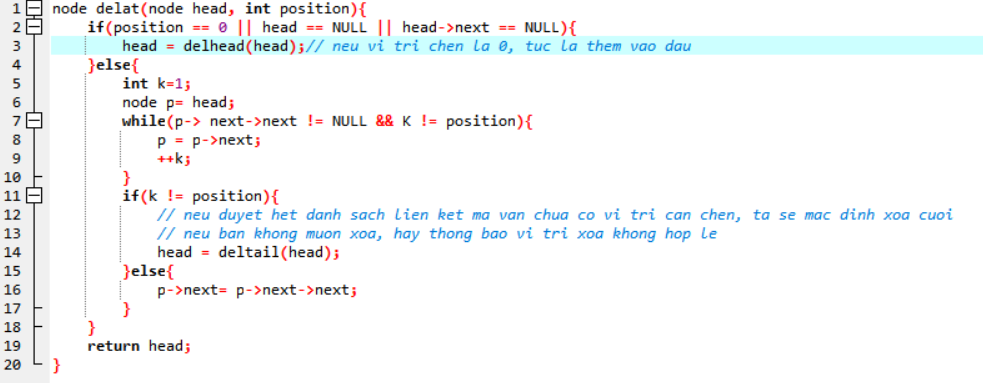
#### Xóa ở vị trí bất kỳ

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



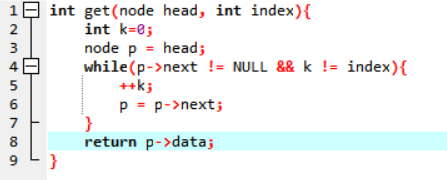
Hình 4

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



### e. Lấy giá trị ở vị trí bất kỳ

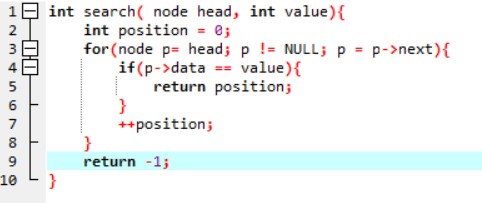
Chúng ta sẽ viết một hàm để truy xuất giá trị ở chỉ số bất kỳ nhé. Trong trường hợp chỉ số vượt quá chiều dài của linked list – 1, hàm này trả về vị trí cuối cùng. Do hạn chế là chúng ta không thể raise error khi chỉ số không hợp lệ. Tôi mặc định chỉ số bạn truyền vào phải là số nguyên không âm. Nếu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



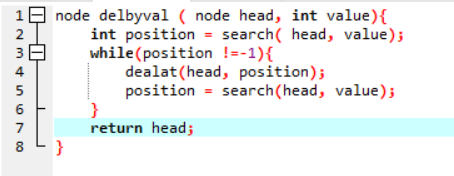
Lý do dùng p->next != NULL là vì chúng ta chỉ muốn đi qua các phần tử có value.

### Tìm kiếm trong danh sách liên kết

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

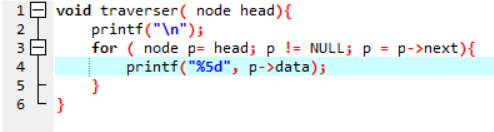


Chúng ta có thể sử dụng hàm này để xóa tất cả các Node trong danh sách liên kết có giá trị chỉ định như sau:



### Duyệt danh sách liên kết

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

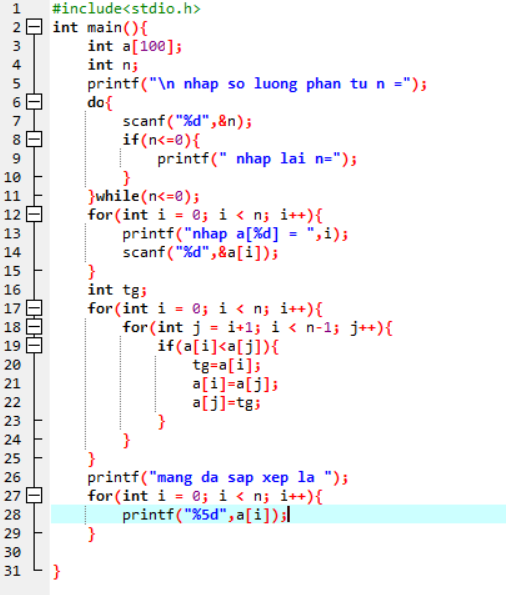


**IV. THUẬT TOÁN SẮP XẾP VÀ THUẬT TOÁN TÌM KIẾM**

**2.4.1. Các thuật toán sắp xếp**

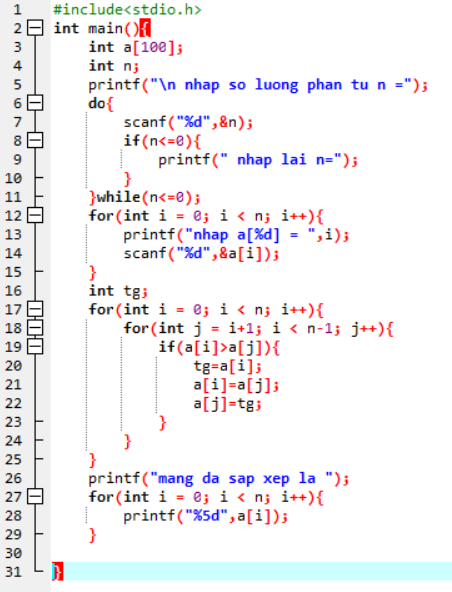
### a. Sắp xếp dãy số giảm dần

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



### b. Sắp xếp dãy số tăng dần

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

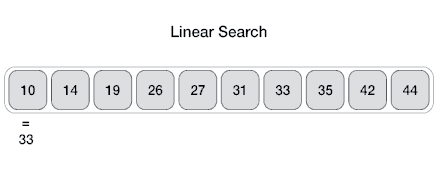


**2.4.2 Thuật toán tìm kiếm**

a. Thuật toán tìm kiếm tuyến tính

Tìm kiếm tuyến tính (Linear Search)

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.

Hình 5

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

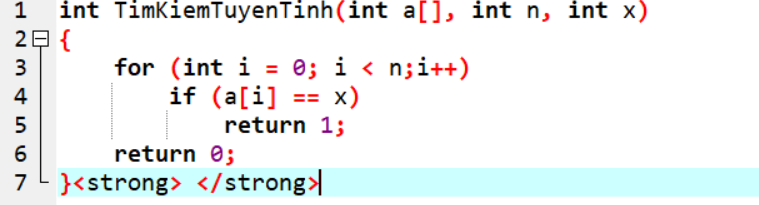
Hãy tìm sinh viên có tên Phat.

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.

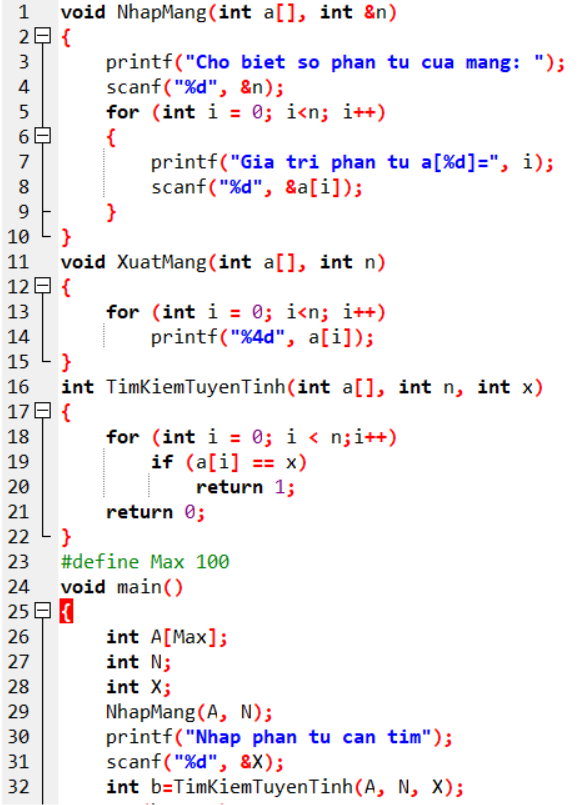
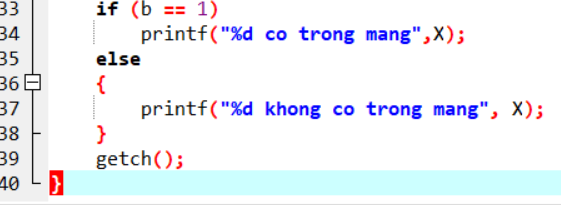
Với i=0 thì a[0].Ten == Tuan != Phat nên ta tăng i lên một đơn vị và i<n-1 nên

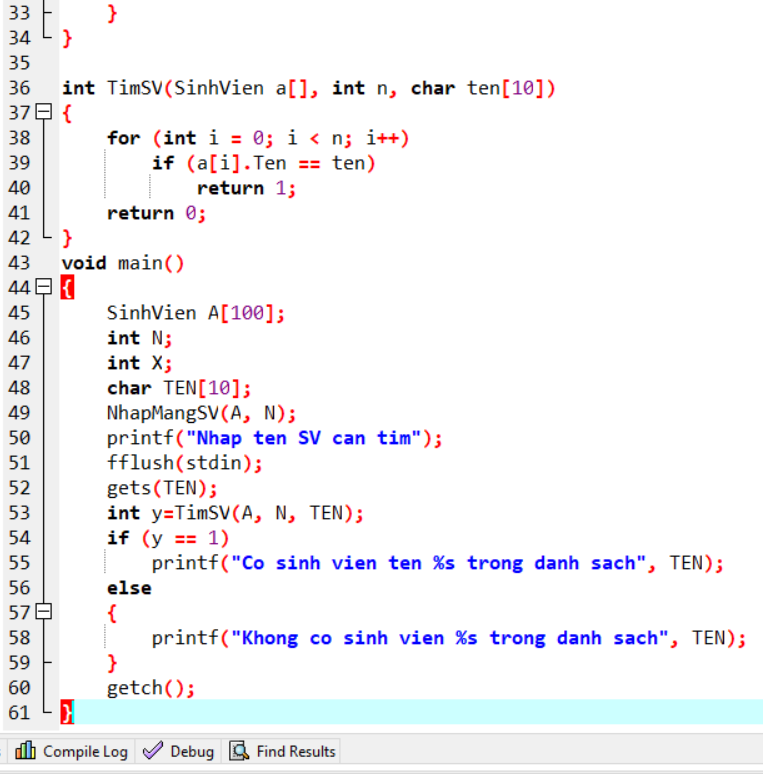
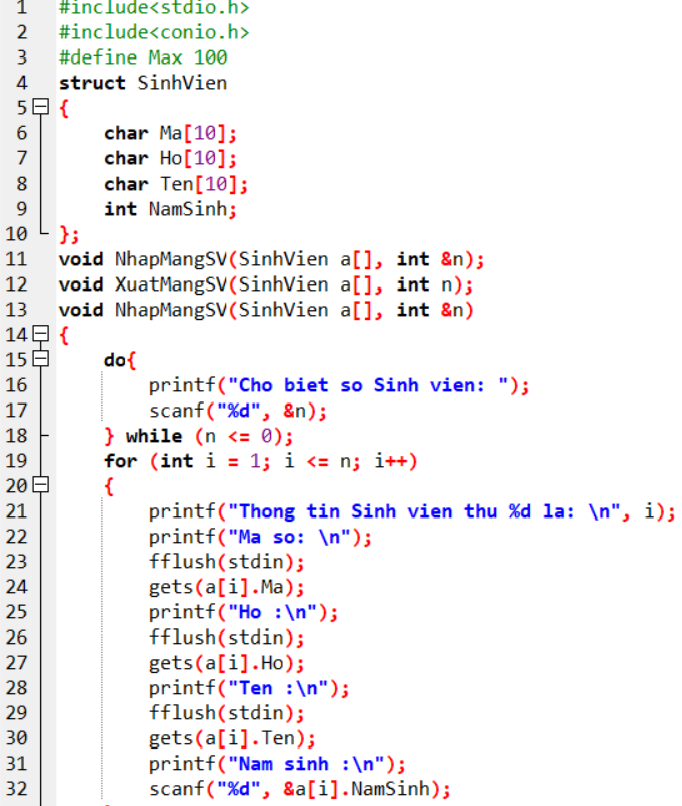
i=1 thì a[1].Ten == Huyen !Phat nên tăng i lên một đơn vị và i<n-1 nên.

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 tổng quát giải thuật:**

**Code ví dụ**





**2.4.1.2 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:

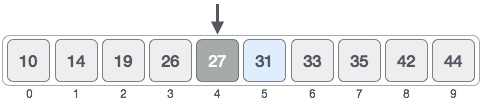


Hình 6

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.



Hình 7

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.

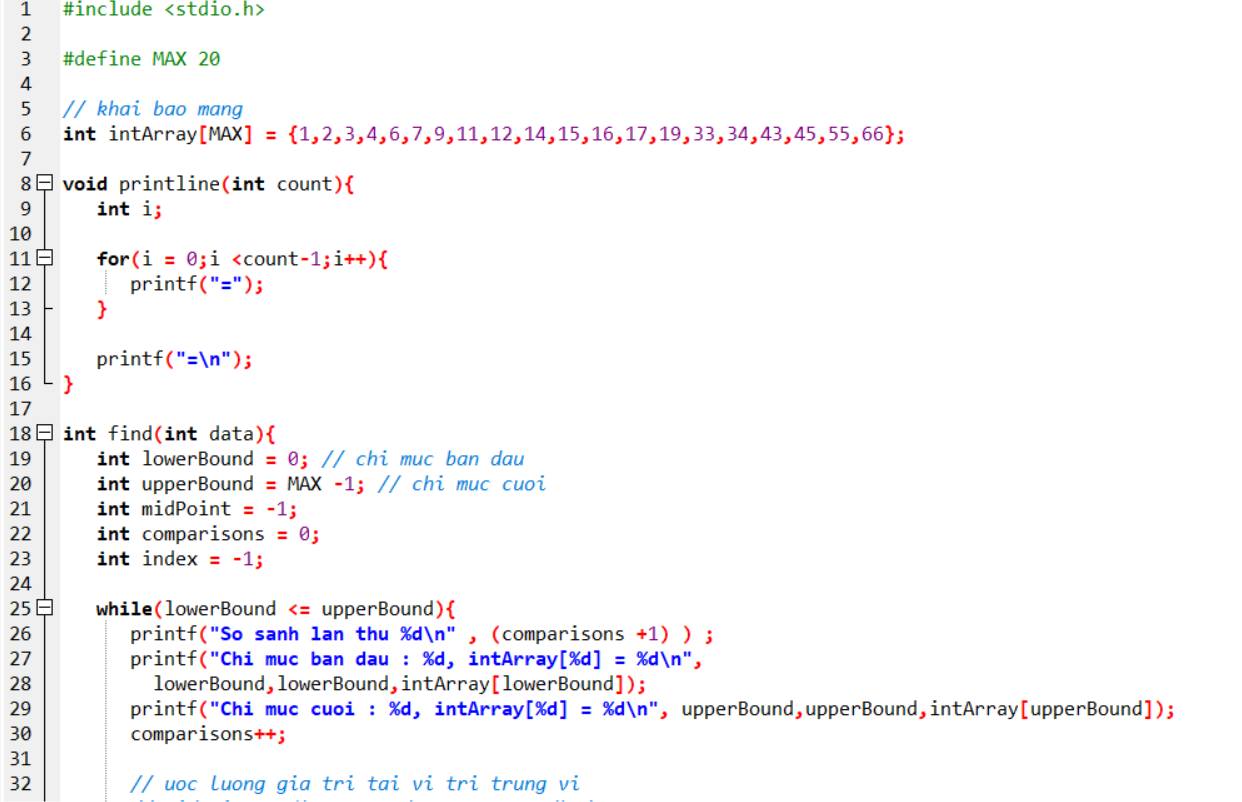


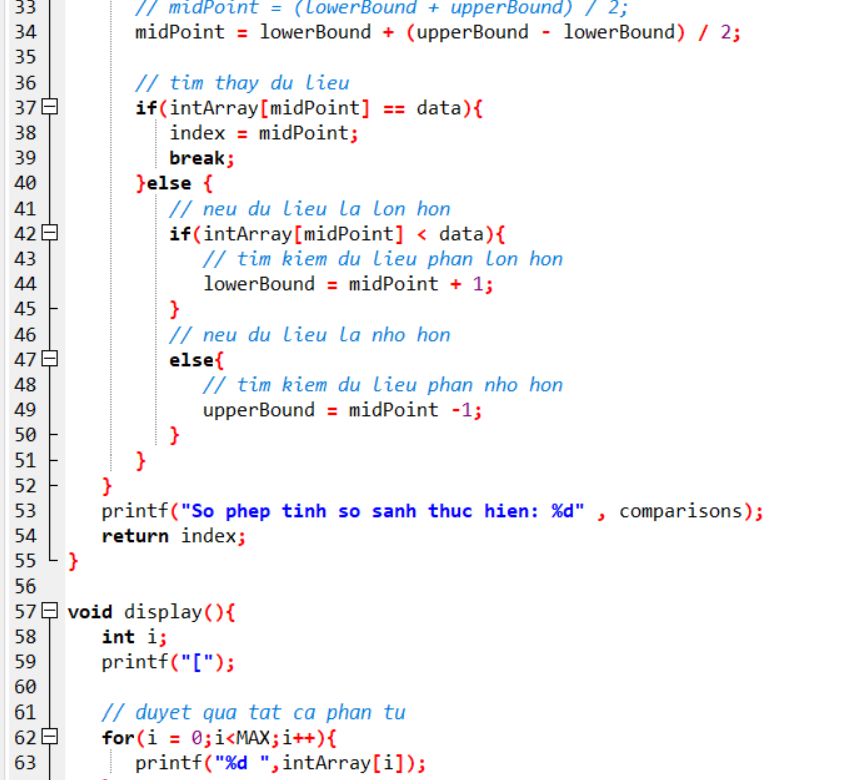
Hình 8

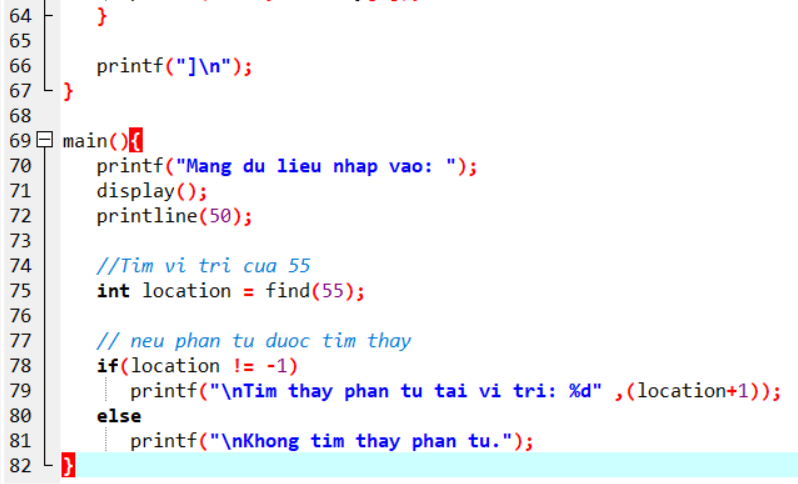
Chúng ta thay đổi giá trị ban-đầu thành chỉ-mục-giữa + 1 và lại tiếp tục tìm kiếm giá trị chỉ-mục-giữa.

ban-đầu = chỉ-mục-giữa + 1

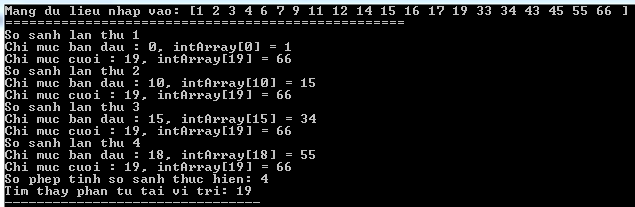
chỉ-mục-giữa = ban-đầu + (cuối + ban-đầu)/ 2







Kết quả



Hình 10

**CHƯƠNG 3: MÔ TẢ THIẾT KẾT GAME**

**3.1.Ý tưởng:**

Từ trò chơi cờ caro thông thường, nhóm của chúng em đã thuyết kế ra **game caro 9 ô** dành cho hai người chơi. **Game caro 9 ô** này có phần khác hơn nhiều so với cờ caro thông thường. Đó là độ khó được tăng lên rất nhiều lần, trong đó có cả kĩ năng tư duy giải quyết vấn đề. Chúng ta phải suy nghĩ thật là kỹ lưỡng cho từng bước đi.

**3.2.Giải thích code:**

***1-Khai báo thư viện :***

#include <conio.h>//dùng để gọi hàm getch()

#include <stdio.h>//các hàm cơ bản

#include <windows.h>//để gọi lệnh xóa màn hình và đổi màu chữ và nền

***2-Hàm vẽ bàn cờ :***

Sử dụng mảng 2 chiều để gán tên cho các ô trong bàn cờ

Sử dụng lệnh đổi màu để trang trí cho bàn cờ

Đặt 1 lệnh xóa màn hình ở đầu hàm vẽ bàn cờ để mỗi khi đánh xong bàn cờ lun được làm mới

char banco[3][3] = { '1','2','3','4','5','6','7','8','9' };//khai báo 1 mảng 2 chiều với các giá trị các phần tử tăng lần lượt từ 1 đến 9

int ve()

{

system("cls");//lệnh xóa màn hình

system("color 37");//thay đổi màu nền và chữ của console, số phía trước là màu nền printf("\n\t\t\t\t\t\t\t~~~~~~~~~~~~~~~~~~~~~~~~~~~CARO~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n\n\n\n");//in ra màn hình tiêu đề

for (int i = 0;i < 3;i++)//vòng lặp for này tương ứng với dòng

{

printf("\t\t\t\t\t\t\t\t\t");

printf("-------------------------"); //các kí tự để trang trí cho bàn cờ

printf("\n");

printf("\t\t\t\t\t\t\t\t\t");

for (int j = 0;j < 3;j++) //vòng lặp for này tương ứng với cột

{

printf("| %c ", banco[i][j]); //in các phần tử của mảng

}

printf("|\n"); // xuống dòng mỗi khi in xong phần tử của 1 dòng

}

printf("\t\t\t\t\t\t\t\t\t");

printf("-------------------------\n"); // trang trí cho bàn cờ

}

***3-Hàm kiểm tra thắng thua hòa :***

Kiểm tra bàn cờ có rơi vào 8 trường hợp 3 ô thẳng hàng hay không.

Nếu không rơi vào 8 trường hợp trên và các ô đã được thay bằng X và O thì sẽ rơi vào trường hợp hòa

int kiemtra()

{

if (banco[0][0] == banco[0][1] && banco[0][1] == banco[0][2])//kiểm tra điều kiện thắng

return 1;

else if (banco[1][0] == banco[1][1] && banco[1][1] == banco[1][2])// kiểm tra điều kiện thắng

return 1;

else if (banco[2][0] == banco[2][1] && banco[2][1] == banco[2][2])// kiểm tra điều kiện thắng

return 1;

else if (banco[0][0] == banco[1][0] && banco[1][0] == banco[2][0]) //kiểm tra điều kiện thắng

return 1;

else if (banco[0][1] == banco[1][1] && banco[1][1] == banco[2][1])// kiểm tra điều kiện thắng

return 1;

else if (banco[0][2] == banco[1][2] && banco[1][2] == banco[2][2])//kiểm tra điều kiện thắng

return 1;

else if (banco[0][0] == banco[1][1] && banco[1][1] == banco[2][2])// kiểm tra điều kiện thắng

return 1;

else if (banco[0][2] == banco[1][1] && banco[1][1] == banco[2][0])// kiểm tra điều kiện thắng

return 1;

else if (banco[0][0] != '1' && banco[0][1] != '2' && banco[0][2] != '3' &&

banco[1][0] != '4' && banco[1][1] != '5' && banco[1][2] != '6' && banco[2][0]

!= '7' && banco[2][1] != '8' && banco[2][2] != '9') //kiểm tra điều kiện hòa

return 0;

else

return -1;

***4- Hàm main***

Bao gồm

Menu

Sử dụng cấu trúc switch case để tạo 2 lựa chọn chơi và thoát.

Đưa toàn bộ thuật toán để chơi vào case 1 và để thoát vào case 2.

Kiểm tra đánh trùng vị trí

Thực hiện kiểm tra ô muốn điền có sở hữu giá trị là 1 số hay đã được gán X hay O vào hay chưa

Xử lí và báo người thắng hay game hòa:

Sử dụng giá trị hàm kiemtra trả về để thông báo kết quả của trò chơi.

int main()

{

system("color 37");//màu nền cho menu

int player = 1, i, c;

char nguoi, menu;

printf("\n\n\n\n\n\t\t\t\t\t\t\t\t\t1.PLAY\n\t\t\t\t\t\t\t\t\t2.EXIT\n");

printf("\t\t\t\t\t\t\t\t\tCHOSE A NUMBER: "); // in menu

scanf("%c", &menu);

switch (menu)

{

case '1': // chọn để chơi

{

do

{

ve();

player = (player % 2) ? 1 : 2;//lần lượt thay đổi giá trị của player giữa 1 với 2

printf("\n\t\t\t\t\t\t\t\t\tPlayer %d, enter a number: \a", player);

scanf("%d", &c);

nguoi = (player == 1) ? 'X' : 'O';//lần lượt thay đổi giá trị của nguoi giữa X và O

if (c == 1 && banco[0][0] == '1')//kiểm tra điều kiện để nhập X hay O vào ô

banco[0][0] = nguoi;

else if (c == 2 && banco[0][1] == '2')

banco[0][1] = nguoi;

else if (c == 3 && banco[0][2] == '3')

banco[0][2] = nguoi;

else if (c == 4 && banco[1][0] == '4')

banco[1][0] = nguoi;

else if (c == 5 && banco[1][1] == '5')

banco[1][1] = nguoi;

else if (c == 6 && banco[1][2] == '6')

banco[1][2] = nguoi;

else if (c == 7 && banco[2][0] == '7')

banco[2][0] = nguoi;

else if (c == 8 && banco[2][1] == '8')

banco[2][1] = nguoi;

else if (c == 9 && banco[2][2] == '9')

banco[2][2] = nguoi;

else

{

printf("\t\t\t\t\t\t\t\t\tInvalid move, press any key to try agian ");

player--;

getch();//giúp dòng thông báo trên hiển thị đến khi nhấn enter

}

i = kiemtra();

player++;

} while (i == -1);

ve();

if (i == 1)

printf("\a\n\t\t\t\t\t\t\t\t\t <->PLAYER %d WIN<-> ", --player);// thông báo người chiến thắng

else

printf("\a\t\t\t\t\t\t\t\t\t\t <<<\_TIE\_>>>");//thông báo hòa

break;

}

case '2'://chọn để thoát

{

exit(0);// thoát chương trình

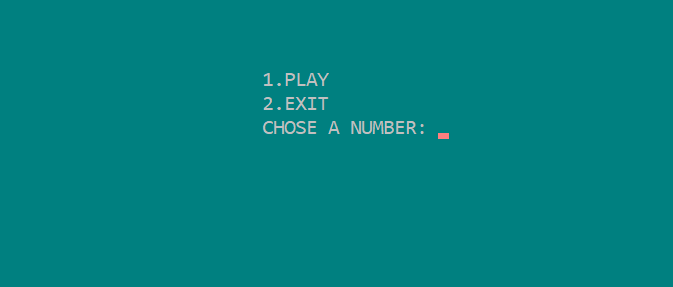
}

}

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

Hướng dẫn sử dụng:

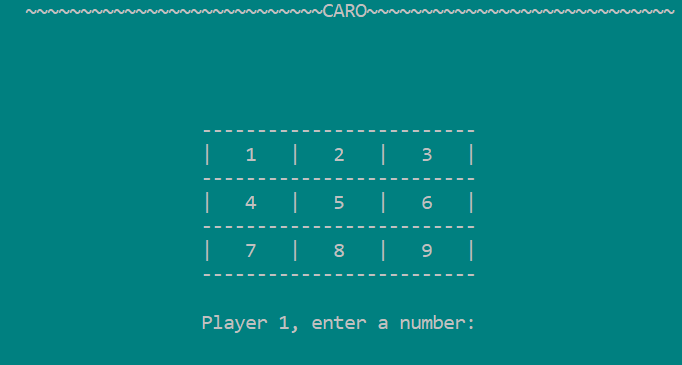
1-người dùng chọn mở vào ứng dụng ( caro9.exe ). Màn hình sẽ hiện thị như sau.



Hình 11

2-Người dùng có thể nhập 1 vào (chose a number: ) và bấm enter để bắt đầu chơi hoặc 2 và bấm enter để thoát khỏi chương trình.

Nếu nhập 1 chương trình sẽ hiển thị như sau và đã có thể bắt đầu chơi:



Hình 12

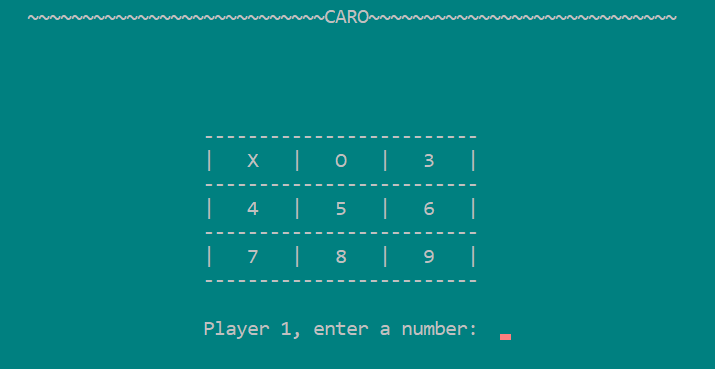
3- Game luôn bắt đầu là lượt của player 1(tương ứng với X), người chơi nhập số( 1->9 ) trên bàn phím để chọn ô

và bấm enter để điền (X) vào số đó:



Hình 13

Ví dụ ở đây là nhập 1 vào và bấm enter, ta thấy 1 được thay bằng X. Sau đó lượt chơi sẽ được chuyển cho player 2(tương ứng với O), làm như trên để nhập O vào số tương ứng:



Hình 14

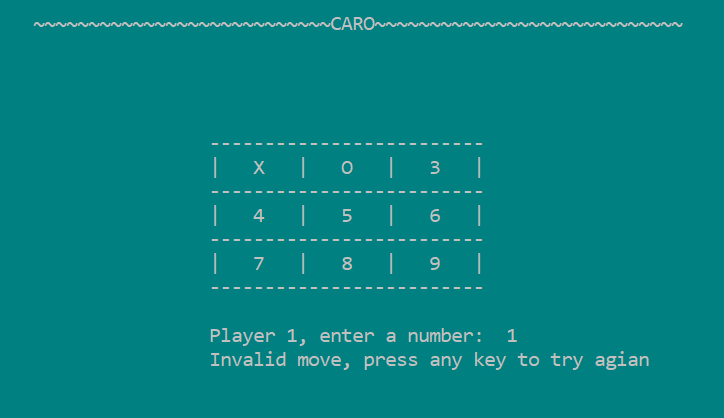
Ở đây ví dụ là nhập vào 2 rồi enter, ta thấy số 2 đã được thay bằng O. Lượt chơi lại được chuyển về cho player 2. Lượt chơi của 2 player thay đổi lần lượt, cho đến khi có người thắng là một trong 2 player có 1 hàng 3 kí tự X hay O hoặc các ô đã được điền bằng X và O mà vẫn không có ai thắng(trường hợp hòa).



Hình 15

Trên đây là trường hợp hòa

Chú ý: nếu nhập vào 1 ô đã được điền(ô đó được hiển thị lên màn hình không còn là một số) thì sẽ không hợp lệ màn hình sẽ báo như sau:



Hình 16

Nếu vậy, người chơi phải nhấn enter và nhâp lại số khác.