**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN HỌC**

**LẬP TRÌNH NÂNG CAO**

**ĐỀ TÀI: XÂY DỰNG GAME TIC TAC TOE**

Sinh viên thực hiện:

PHAN VĂN TUYỂN – MSSV: 6051071139

TRẦN TẤN NHỰT – MSSV: 6051071087

MẠNH QUỐC TOÀN – MSSV: 6051071124

Lớp : CÔNG NGHỆ THÔNG TIN K60

Khoá :K60

Tp. Hồ Chí Minh, năm 2020

MỤC LỤC

[**LỜI CẢM ƠN** 3](#_Toc43274067)

[**PHẦN MỘT:TỔNG QUAN ĐỀ TÀI** 4](#_Toc43274068)

[**I.Giới thiệu** 4](#_Toc43274069)

[**II.. Lý do chọn đề tài** 5](#_Toc43274070)

[**PHẦN HAI: NỘI DUNG LÝ THUYẾT** 6](#_Toc43274071)

[**I. Các thuật toán sắp xếp** 6](#_Toc43274072)

[**1. Sắp xếp nổi bọt( Bubble sort)** 6](#_Toc43274073)

[1.1. Ý tưởng 6](#_Toc43274074)

[1.2. Xây dựng thuật toán 6](#_Toc43274075)

[1.3. Ví dụ minh hoạ 7](#_Toc43274076)

[**2. Sắp xếp chọn trực tiếp( Selection sort)** 7](#_Toc43274077)

[2.1. Ý tưởng 7](#_Toc43274078)

[2.2. Xây dựng thuật toán 7](#_Toc43274079)

[2.3. Ví dụ minh họa 8](#_Toc43274080)

[**3. Sắp xếp chèn( Insertion sort)** 9](#_Toc43274081)

[3.1. Ý tưởng 9](#_Toc43274082)

[3.2. Xây dựng thuật toán 9](#_Toc43274083)

[3.3. Ví dụ minh hoạ 10](#_Toc43274084)

[**4. Sắp xếp đổi chỗ trực tiếp( Interchange sort)** 10](#_Toc43274085)

[4.1. Ý tưởng 10](#_Toc43274086)

[4.2. Xây dựng thuật toán 10](#_Toc43274087)

[4.3. Ví dụ minh hoạ 11](#_Toc43274088)

[**II. Các thuật toán tìm kiếm** 12](#_Toc43274089)

[**1. Tìm kiếm tuần tự** 12](#_Toc43274090)

[1.1. Ý tưởng 12](#_Toc43274091)

[1.2. Xây dựng thuật toán 12](#_Toc43274092)

[1.3. Ví dụ minh họa 13](#_Toc43274093)

[**2. Tìm kiếm nhị phân** 14](#_Toc43274094)

[2.1. Ý tưởng 14](#_Toc43274095)

[2.2. Xây dựng thuật toán 14](#_Toc43274096)

[2.3. Ví dụ minh họa 16](#_Toc43274097)

[**III. Đọc ghi file trong C** 17](#_Toc43274098)

[**1. Tại sao ta phải cần đến tệp tin?** 17](#_Toc43274099)

[**2. Các kiểu file.** 17](#_Toc43274100)

[a. File văn bản ( Text Files) 17](#_Toc43274101)

[b. File nhị phân( Binary Files) 17](#_Toc43274102)

[**3. Các thao tác với tệp** 18](#_Toc43274103)

[a) Thao tác mở file 18](#_Toc43274104)

[b) Thao tác đóng file 19](#_Toc43274105)

[d) Đọc/Ghi file nhị phân trong C 21](#_Toc43274106)

[**IV. Danh sách liên kết đơn** 22](#_Toc43274107)

[**1.** **Khái niệm danh sách đơn là gì?** 22](#_Toc43274108)

[**2.** **Cài đặt danh sách liên kết đơn** 22](#_Toc43274109)

[a) Khai báo danh sách 22](#_Toc43274110)

[b) Tạo một Node 23](#_Toc43274111)

[c) Thêm Node và danh sách lien kết 24](#_Toc43274112)

[d) Xóa Node khỏi danh sách 26](#_Toc43274113)

[e) Tìm phần tử có giá trị x trong danh sách 28](#_Toc43274114)

[**PHẦN BA: NỘI DUNG THỰC HÀNH** 29](#_Toc43274115)

[**I.. Giới thiệu đề tài** 29](#_Toc43274116)

[**II.. Ý tưởng xây dựng** 29](#_Toc43274117)

[**III.. Viết chương trình** 29](#_Toc43274118)

[**PHẦN BỐN: TỔNG KẾT** 35](#_Toc43274119)

[**I.. Kết quả đạt được** 35](#_Toc43274120)

[**II.. Hạn chế** 35](#_Toc43274121)

[**III.. Hướng phát triển đề tài** 35](#_Toc43274122)

[TÀI LIỆU THAM KHẢO 36](#_Toc43274123)

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!*

# **PHẦN MỘT:TỔNG QUAN ĐỀ TÀI**

## **I.Giới thiệu**

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.

## **II.. Lý do chọn đề tài**

Mục đích khi chúng tôi tạo ra game Tic Tac Toe chính là để giải quyết vấn đề giải trí . Sau những ngày làm việc mệt mỏi tựa lưng vào gối và mở game Tic Tac Toe thưởng thức sẽ làm giảm rất nhiều stress sinh ra . Đặc biệt Tic Tac Toe là một game đơn giản , không cần nhiều chất xám ở người chơi , một người chưa từng chơi cũng sẽ dễ dàng để hiểu được quy luật của trò chơi .

PHẦN HAI: NỘI DUNG LÝ THUYẾT

I. Các thuật toán sắp xếp

**1. Sắp xếp nổi bọt( Bubble sort)**

**1.1. Ý tưởng**

-Xuất phát từ cuối( đầu) dãy, đổi chỗ các cặp phần tử kế cận để đưa phần tử nhỏ( lớn) lên trong cặp phần tử đó về đầu( cuối) dãy, sau đó không xét đến nó nữa ở bước tiếp theo.

-Lặp lại cho đến khi không còn cặp phần tử nào để xét.

**1.2. Xây dựng thuật toán**

**-Bước 1:** Đầu tiên cho i=0 , j=n-1.

**- Bước 2:** Xét j>i thì ta làm:

+ Nếu a[j] < a[j-1] :

Temp = a[j-1]

a[j-1]=a[j]

a[j]=temp

+j=j-1

**- Bước 3:** Nếu i> n-1 thì ta sẽ dừng, ngược lại thì ta lặp lại bước 2.

* Code minh họa:

**void** **BubbleSort** (**int** a[],**int** n) {

**for**(**int** i=**0**;i<n-**1**;i++) {

**for**(**int** j=n-**1**; j>i ; j--) {

**if**(a[j]<a[j-**1**]){

**int** temp=a[j-**1**];

a[j-**1**]=a[j];

a[j]=temp;

}

}

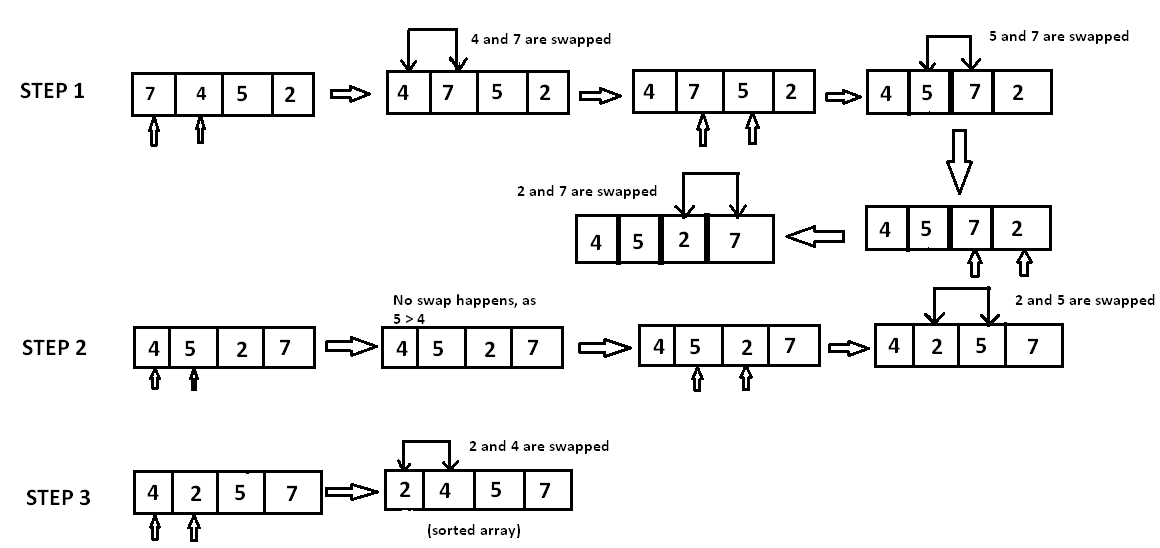
}

}

**1.3. Ví dụ minh hoạ:**

Sắp xếp dãy số 7, 4, 5, 2 sau đây tăng dần theo cách sắp xếp nổi bọt

\*Lưu ý giải thuật ví dụ này ta xét từ đầu dãy đến cuối dãy



**2. Sắp xếp chọn trực tiếp( Selection sort)**

**2.1. Ý tưởng**

- Chọn phần tử nhỏ nhất( lớn nhất) trong N phần tử ban đầu, đưa phần tử này về vị trí đầu dãy

- Xem dãy hiện hành còn N-1 phần tử của dãy ban đầu, bắt đầu từ vị trí thứ 2, lặp lại quá trình trên cho dãy hiện hành đến khi dãy chỉ còn phần tử.

**2.2. Xây dựng thuật toán**

**-**Giả sử sắp xếp dãy tăng dần có N phần tử:

**+Bước 1:** Đầu tiên ta cho i=0.

**+Bước 2:** Tìm phần tử nhỏ nhất a[MIN] trong dãy từ a[i] đến a[N-1].

**+Bước 3:** Xét nếu MIN khác i, ta hoán vị a[MIN] và a[i].

**+Bước 4:** Nếu i<N-1 thì i=i+1, thì lặp lại các bước 2.

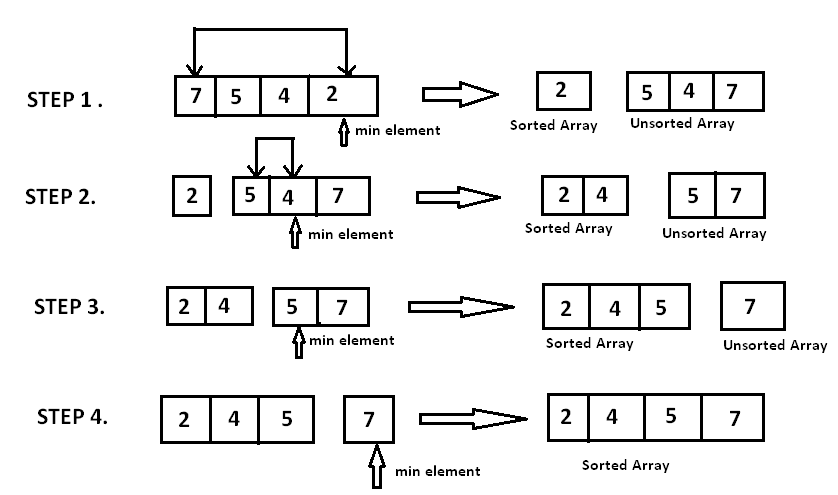
Ngược lại thì dừng.

- Code minh họa:

|  |
| --- |
| **void** **SelectionSort**(**int** a[],**int** n){  **int** min;  **for** (**int** i=**0**; i<n; i++){  min = i; //Gia su so ban dau trong day la min  **for**(**int** j=i+**1**;j<n;j++){ //Tim so nho nhat trong day hien hanh  **if**(a[j]<a[min]){  min = j;  }  }  **if**(min != i){ //Neu so nao nho hon so dau tien ta hoan vi cho nhau  **int** temp= a[i];  a[i]=a[min];  a[min]=temp;  }  }  } |

**2.3. Ví dụ minh họa**

Sắp xếp dãy số 7,5,4,2 tăng dần sử dụng thuật toán sắp xếp chọn trực tiếp



**3. Sắp xếp chèn( Insertion sort)**

**3.1. Ý tưởng**

-Giả sử ta xét dãy có n phần tử: a0, a1,….,a(n-1)

**-** Xét dãy gồm 1 phần tử là a0 dãy có thứ tự

**-** Thêm a1 vào dãy có thứ tự a0 sao cho dãy mới a0,a1 là dãy có thứ tự, nếu a1< a0 thì ta đổi chỗ cho a1 với a0 cho nhau.

**-** Thêm a2 vào dãy a0, a1 sao cho dãy mới a0, a1, a2 là dãy có thứ tự.

**-**Tiếp tục như thế đến n-1 bước ta sẽ có dãy thứ tự a0, a1, a2 … a(n-1).

**3.2. Xây dựng thuật toán**

**-** Giả sử sắp xếp dãy a tăng dần gồm N phần tử:

+Bước 1: Cho i=1 ( Giả sử có đoạn a[0] được sắp xếp

+Bước 2: Chọn x= a[i] ( chọn vị trí pos thích hợp trong đoạn a[i] đến a[i-1] để chèn a[i] vào).  
 +Bước 3: Dời chỗ các phần tử từ vị trí a[pos] đến a[i-1] sang phải 1 vị trí để vị trí đó cho a[i].

+Bước 4: a[pos]=x.

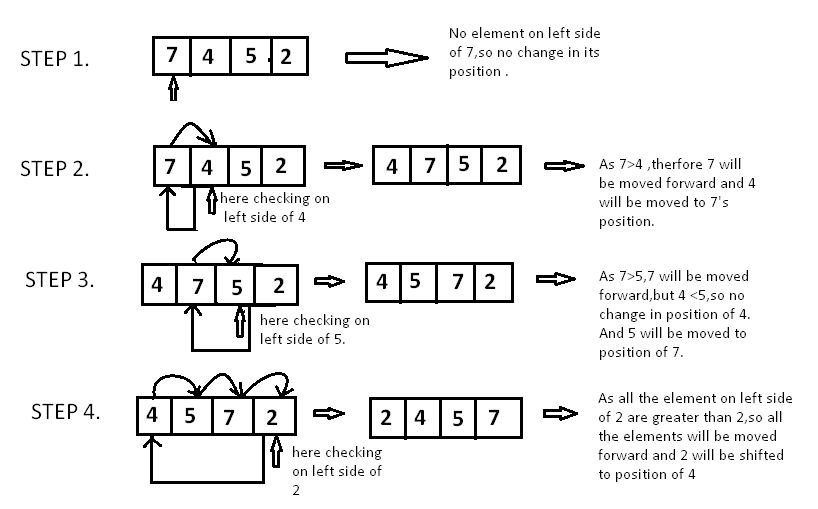
+Bước 5: Tăng i lên một đơn vị. Nếu i<N thì ta lặp lại bước 2, ngược lại thì dừng.

-Code minh họa:

|  |
| --- |
| **void** **InsertionSort**(**int** a[],**int** n){  **int** pos =**0**; // luu vi tri can chen  **int** x; // luu gia tri a[i] tranh bi ghi de khi doi cho cac phan tu  **for**( **int** i=**1**; i<n ; i++){ //xem doan a[0] da sap  x=a[i];  **for** ( pos = i; (pos>**0**) && (a[pos-**1**]>x);pos--){  a[pos]=a[pos-**1**];  }  a[pos]=x; // chon x vao day  }  } |

**3.3. Ví dụ minh hoạ**

Sắp xếp tăng dần của dãy số 7, 4, 5, 2 theo phương pháp sắp xếp chèn



**4. Sắp xếp đổi chỗ trực tiếp( Interchange sort)**

**4.1. Ý tưởng**

**-** Xuất phát từ vị trí đầu dãy, tìm tất cả nghịch thế chứa phần tử này, triệt tiêu chúng bằng cách đổi chỗ phần tử này với phần tử tương ứng trong cặp nghịch thế.

- Lặp lại các xử lý trên với các phần tử tiếp theo trong dãy.

**4.2. Xây dựng thuật toán**

-Giả sử sắp xếp dãy số nguyên tăng dần có n phần tử

**+Bước 1:** Ta bắt đầu từ vị trí đầu tiên của dãy i=0

**+Bước 2:** Ta xét các phần tử sau nó j=i+1

+ Nếu a[j]< a[i] thì ta đổi chỗ hai phần tử này với nhau, sau đó biến j tăng thêm 1 đơn vị.Cho đến khi j<n thì kết thúc.

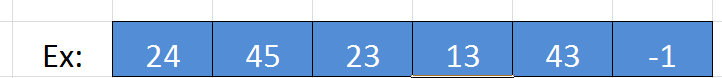
**+Bước 3:** Biến i tăng lên 1 và nếu i<n thì ta lặp lại các bước 2 ngược lại thì dừng.

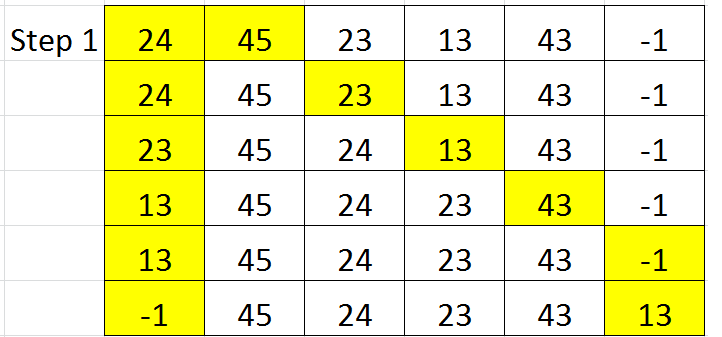
- Code minh họa:

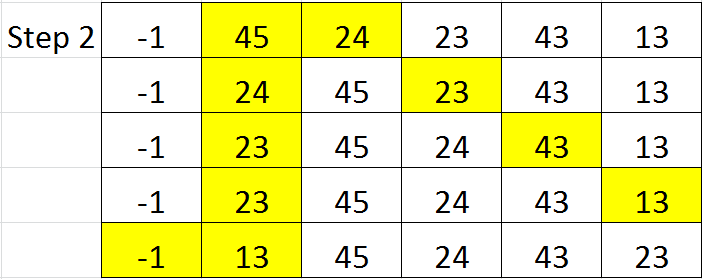
|  |
| --- |
| **void** **InterchangeSort**(**int** a[], **int** n)  {  **for** (**int** i = **0**; i < n-**1**; i++)  **for** (**int** j = i+**1**; j < n; j++)  **if** (a[i] > a[j])  {  **int** temp = a[i];  a[i] = a[j];  a[j] = temp;  }  } |

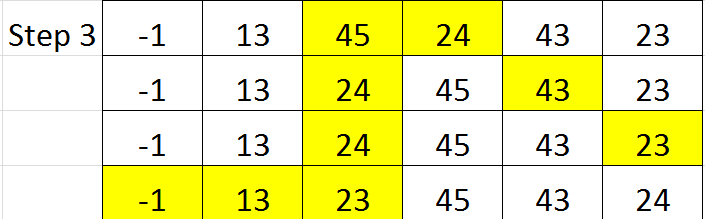
**4.3. Ví dụ minh hoạ**

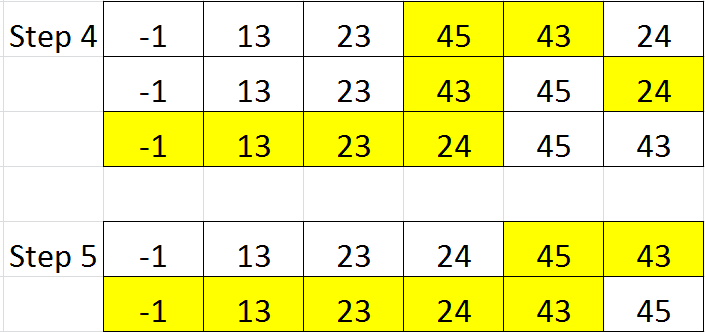
**-**Ví dụ sau là ví dụ sắp xếp dãy tăng dần theo phương pháp đổi chỗ trực tiếp.











II. Các thuật toán tìm kiếm

**1. Tìm kiếm tuần tự**

**1.1. Ý tưởng:**

- Ta lần lượt so sánh phần tử cần tìm X với các phần tử trong dãy, cho đến khi tìm thấy hoặc duyệt hết các phần tử trong dãy mà không thấy.

#### 1.2. Xây dựng thuật toán:

Giả sử tìm kiếm x trong dãy có N phần tử:

**-Bước 1:** i=0 =>Bắt đầu với phần tử đầu tiên trong dãy.

**-Bước 2:** So sánh a[i] với x:

+Nếu a[i]=x =>tìm thấy và dừng thuật toán.

+Nếu a[i] khác x thì ta sang bước tiếp theo.

**-Bước 3:** i=i+1 => Xét phần tử kế tiếp trong dãy

+Nếu i>=N thì ta dừng và thông báo không có phần tử x trong dãy.

+Ngược lại ta quay lại bước 2.

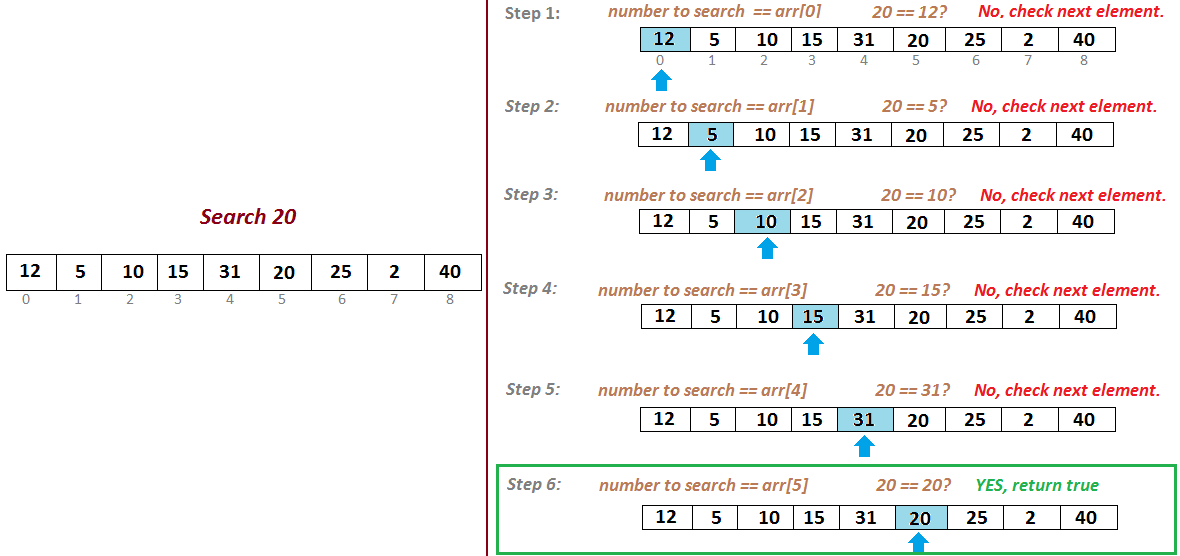
-Code minh họa:

|  |
| --- |
| **int** **LinearSearch**(**int** a[], **int** n,**int** x){  **for**(**int** i=**0**; i<n; i++){  **if**(a[i]==x){  **return** i;  }  }  **return** -**1**;  } |

|  |
| --- |
| **int** **main**(){  **int** kiemTra = LinearSearch(a, n,k);  **if**( kiemTra == -**1**){  printf("**\n** Khong tim thay k trong day!!");  }**else**  printf("**\n** Tim thay k o vi tri thu %d", kiemTra);  } |

+Gọi hàm trong hàm main:

#### 1.3. Ví dụ minh họa

****

### **2. Tìm kiếm nhị phân**

#### 2.1. Ý tưởng

- Việc tìm kiếm nhị phân được thực hiện trên dãy đã được sắp xếp.

- Chia dãy cần cần tìm phần tử thành hai phần, xác định phần nào chứa phần tử cần tìm, việc tìm kiếm sẽ thực hiện trên một nửa dãy.

- Để tìm kiếm phần tử X có trong dãy hay không ta bắt đầu so sánh X với phần tử ở giữa dãy. Nếu X nhỏ hơn thì X chỉ có thể nằm trong nửa dãy đầu tiên, ngược lại nếu X lớn hơn thì nằm trong nửa dãy còn lại

-Ta cứ lặp lại quá trình này cho đến khi tìm thấy X hoặc phạm vi tìm kiếm không còn nữa.

#### 2.2. Xây dựng thuật toán

-Giả sử ta tìm kiếm phần tử k trong dãy a có N phần tử đã được sắp xếp

+ Bước 1: Đặt right=N-1 và left=0

+Bước 2: mid= (left + right)/2. ( Lấy mốc so sánh)

-So sánh k với a[mid], có 3 trường hợp xảy ra :

+ Nếu a[mid]= k. => Tìm thấy => Dừng.

+ Nếu a[mid] > k: thì tìm trong đoạn từ a[left] đến a[right]

right= mid -1.

+ Còn lại a[mid] < k: thì tìm trong đoạn từ a[left] đến a[right]

left= mid +1.

+Bước 3: Nếu left <= right: còn phần tử chưa xét => tìm tiếp ,lặp lại bước 2.

Ngược lại: Dừng. Đã xét hết các phần tử mà không thấy giá trị k => Không tìm thấy.

* Code minh họa:

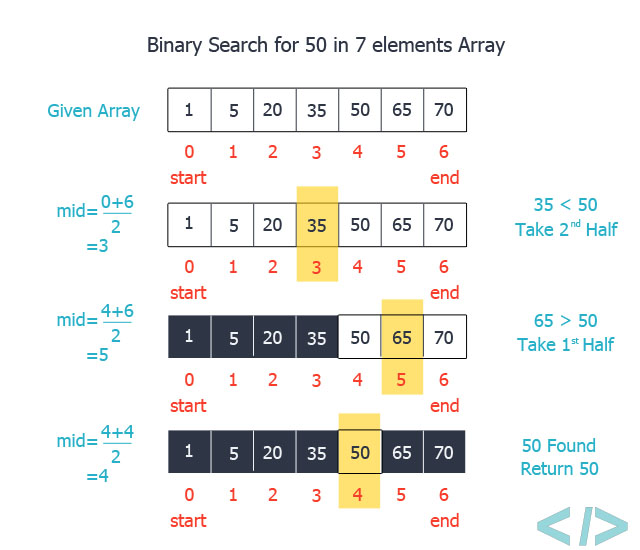
|  |
| --- |
| **int** **BinarySearch**(**int** a[], **int** n,**int** k){  **int** left=**0**, right= n-**1**,mid;    **do**{  mid= (left + right)/**2**;  **if**( k==a[mid]){  **return** mid; // Tim thay k tai mid  }  **else** **if**(k < a[mid]){  right= mid - **1**;  } **else**{  left= mid + **1**;  }    } **while**( left <= right);    **return** -**1**;} //Tim het day ma khong ton tai k |

-Gọi hàm trong hàm main:

|  |
| --- |
| **int** **main**(){  **int** kiemTra = BinarySearch(a, n,k);  **if**(kiemTra == -**1**){  printf("**\n** Khong tim thay k trong day!!");  }**else**  printf("**\n** Tim thay k o vi tri thu %d",kiemTra);  } |

**Lưu ý:** Ở hai thuật toán tìm kiếm tuần tự và tìm kiếm nhị phân ta sử dụng hàm trả về giá trị để khi ta tìm kiếm phần tử nào đó có trong dãy. Nếu phần tử cần tìm trùng với một phần tử trong dãy ta sẽ trả về giá trị tại phần tử đó và thoát chương trình hàm. Và không xét tiếp các phần tử còn lại.

#### 2.3. Ví dụ minh họa

****

## **III. Đọc ghi file trong C**

### **1. Tại sao ta phải cần đến tệp tin?**

* Dữ liệu được lưu ở biến của chương trình, và nó sẽ biến mất khi chương trình kết thúc. Sử dụng file để lưu trữ dữ liệu cần thiết để đảm bảo dữ liệu của chúng ta không bị mất ngay cả khi chương trình của chúng ta ngừng chạy.
* Nếu chương trình của bạn có đầu vào(input) là lớn, bạn sẽ rất vất vả nếu phải nhập mỗi khi chạy. Thay vào đó, hãy lưu vào file và chương trình của bạn sẽ tự đọc mỗi lần khởi chạy.
* Dễ dàng sao chép, di chuyển dữ liệu giữa các thiết bị với nhau.

### **2. Các kiểu file.**

Trước khi làm việc với file, ta cần biết về 2 kiểu file thường gặp đó là : **file văn bản( text files), file nhị phân( binary files).**

#### 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.

### **3. Các thao tác với tệp**

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.

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.

**FILE \*fprt;**

#### a) Thao tác mở file

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:

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

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

Ví dụ:

**fptr = fopen("E:\\cprogram\\newprogram.txt","w");**

**fptr = fopen("E:\\cprogram\\oldprogram.bin","rb");**

**Lưu ý:** địa chỉ sẽ có dạng **\** , mà **\** là 1 kí tự đặc biệt trong C , nên ta sẽ thêm vào 1 dấu **\** nữa ở mỗi dấu **\** trong địa chỉ**.**

**- Các tham số của “ mode “:**

w : Chế độ ghi ,nó sẽ xóa toàn bộ dữ liệu có trong file và sau đó tiến hành ghi , nếu file không tồn tại sẽ tự động tạo 1 file mới ( sử dụng trong công việc ghi file).

w+ : chế độ ghi , nó sẽ không xóa dữ liệu có trong file , nhưng dữ liệu cũ trong file sẽ bị overwrite , tức là bị ghi đè lên(sử dụng trong công việc ghi file)

r : Chỉ đọc(sử dụng trong công việc đọc file).

r+ : Đọc và ghi (sử dụng trong đọc và ghi file).

a hoặc a+ : Thêm vào nhưng không thay đổi thông tin trong file , tức là dữ liệu sẽ được đưa vào cuối file ( sử dụng trong ghi file).

#### b) 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().**

**fclose(fptr); //** ở đây fptr làcon trỏ trỏ đến file cần đóng.

**c) Đọc/Ghi file văn bản trong C**

Để đọc và ghi vào tệp văn bản, chúng ta sử dụng các hàm **fprintf ()** và **fscanf ().**

-Ví dụ 1: Ghi file sử dụng fprintf()

#include <stdio.h>

#include <stdlib.h>

**int** **main**()

{

**int** num;

**FILE** \*fptr;

fptr = fopen("C:**\\**program.txt","w");

**if**(fptr == NULL)

{

printf("Error!");

exit(**1**);

}

printf("Enter num: ");

scanf("%d",&num);

fprintf(fptr,"%d",num);

fclose(fptr);

**return** **0**;

}

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: **Đọc file sử dụng fscanf()**

#include <stdio.h>

#include <stdlib.h>

**int** **main**()

{

**int** num;

**FILE** \*fptr;

**if** ((fptr = fopen("C:**\\**program.txt","r")) == NULL){

printf("Error! opening file");

// Program exits if the file pointer returns NULL.

exit(**1**);

}

fscanf(fptr,"%d", &num);

printf("Value of n=%d", num);

fclose(fptr);

**return** **0**;

}

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.

**d)** **Đọc/Ghi file nhị phân trong C**

Khác với ghi file văn bản ghi file nhị nhân ta sử dụng hàm **fread()** và **fwrite().**

- 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.

**fwrite ( address\_data, size\_data, numbers\_data, pointer\_to\_file);**

|  |
| --- |
| **if** ((fptr = fopen("C:**\\**program.bin","wb")) == NULL){  printf("Error! opening file");    // Program exits if the file pointer returns NULL.  exit(**1**);  }    **for**(n = **1**; n < **5**; ++n)  {  num.n1 = n;  num.n2 = **5**\*n;  num.n3 = **5**\*n + **1**;  fwrite(&num, **sizeof**(**struct** threeNum), **1**, fptr);  }  fclose(fptr); |

Ví dụ:

- Để đọc file nhị phân ta sử dụng hàm fread(). Hàm fread() cũng có 4 tham số tương tự như hàm fwrite() phía trên.

**fread(address\_data, size\_data, numbers\_data, pointer\_to\_file);**

Ví dụ:

|  |
| --- |
| **if** ((fptr = fopen("C:**\\**program.bin","rb")) == NULL){  printf("Error! opening file");    // Program exits if the file pointer returns NULL.  exit(**1**);  }    **for**(n = **1**; n < **5**; ++n)  {  fread(&num, **sizeof**(**struct** threeNum), **1**, fptr);  printf("n1: %d**\t**n2: %d**\t**n3: %d", num.n1, num.n2, num.n3);  }  fclose(fptr); |

## **IV. Danh sách liên kết đơn**

1. **Khái niệm danh sách đơn là gì?**

Danh sách liên kết đơn : mỗi phần tử liên kết với phần tử đứng sau nótrong danh sách. Mỗi phần tử gọi là một Node, mỗi Node có tối thiểu hai thông tin: Giá trị ( Data) và con trỏ pNext trỏ tới phần tử kế tiếp.



1. **Cài đặt danh sách liên kết đơn**
2. **Khai báo danh sách :**

Cấu trúc của một danh sách liên kết được khai báo như sau:

|  |
| --- |
| **typedef** Kieu\_Du\_Lieu ElementType;  **struct** NodeType{  ElementType data;  **struct** NodeType \*Link;  } NodeType \*List; |

Ví dụ:

|  |
| --- |
| **struct** LinkedList{  **int** data;  **struct** LinkedList \*next;  }; |

#### b) Tạo một Node

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

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

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

* Ví dụ:

|  |
| --- |
| **typedef** **struct** LinkedList \*node; //Từ giờ dùng kiểu dữ liệu LinkedList có thể thay bằng node cho ngắn gọn    node **CreateNode**(**int** value){  node temp; // declare a node  temp = (node)malloc(**sizeof**(**struct** LinkedList)); // Cấp phát vùng nhớ dùng malloc()  temp->next = NULL;// Cho next trỏ tới NULL  temp->data = value; // Gán giá trị cho Node  **return** temp;//Trả về node mới đã có giá trị |

#### c) Thêm Node và danh sách lien 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:

|  |
| --- |
| node **AddHead**(node head, **int** value){  node temp = CreateNode(value); // Khởi tạo node temp với data = value  **if**(head == NULL){  head = temp; // //Nếu linked list đang trống thì Node temp là head luôn  }**else**{  temp->next = head; // Trỏ next của temp = head hiện tại  head = temp; // Đổi head hiện tại = temp(Vì temp bây giờ là head mới mà)  }  **return** head;  } |

**- Thêm vào cuối:**

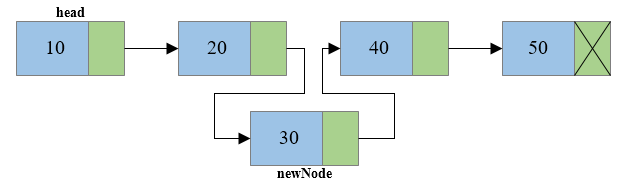
Chúng ta sẽ cần Node đầu tiên, và giá trị muốn thêm. Khi đó, ta sẽ:

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:

|  |
| --- |
| node **AddTail**(node head, **int** value){  node temp,p;// Khai báo 2 node tạm temp và p  temp = CreateNode(value);//Gọi hàm createNode để khởi tạo node temp có next trỏ tới NULL và giá trị là value  **if**(head == NULL){  head = temp; //Nếu linked list đang trống thì Node temp là head luôn  }  **else**{  p = head;// Khởi tạo p trỏ tới head  **while**(p->next != NULL){  p = p->next;//Duyệt danh sách liên kết đến cuối. Node cuối là node có next = NULL  }  p->next = temp;//Gán next của thằng cuối = temp. Khi đó temp sẽ là thằng cuối(temp->next = NULL mà)  }  **return** head;  } |

**- 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.

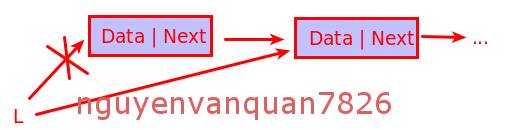
Lưu ý: Chỉ số chèn bắt đầu từ chỉ số 0.

|  |
| --- |
| node **AddAt**(node head, **int** value, **int** position){  **if**(position == **0** || head == NULL){  head = AddHead(head, value); // Nếu vị trí chèn là 0, tức là thêm vào đầu  }**else**{  // Bắt đầu tìm vị trí cần chèn. Ta sẽ dùng k để đếm cho vị trí  **int** k = **1**;  node p = head;  **while**(p != NULL && k != position){  p = p->next;  ++k;  }    **if**(k != position){  // Nếu duyệt hết danh sách lk rồi mà vẫn chưa đến vị trí cần chèn, ta sẽ mặc định chèn cuối  // Nếu bạn không muốn chèn, hãy thông báo vị trí chèn không hợp lệ  head = AddTail(head, value);  // printf("Vi tri chen vuot qua vi tri cuoi cung!\n");  }**else**{  node temp = CreateNode(value);  temp->next = p->next;  p->next = temp;  }  }  **return** head;  } |

**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 không còn giá trị cũ).

#### d) Xóa Node khỏi danh sách

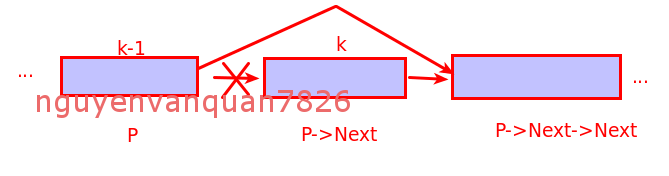
**-Xóa vị trị đầu tiên:**



Xóa 1 phần tử ở đầu danh sách không chỉ đơn giản là cập nhập lại biến con trỏ Head, mà ta phải giải phóng được vùng nhớ của Node cần xóa.

|  |
| --- |
| node **DelHead**(node head){  **if**(head == NULL){  printf("**\n**Khong co gi de xoa het!");  }**else**{  head = head->next;  }  **return** head;  } |

**- Xóa phần tử ở vị trí k:**

****

Dùng P duyệt đến vị trí k-1 và tiến hành cho P->Next trỏ đến phần tư kế tiếp k mà bỏ qua k. Lưu ý trong hình mình quên không lưu lại giá trị cần xóa tuy nhiên các bạn cần lưu lại như khi xóa ở vị trí đầu tiên.

|  |
| --- |
| **void** **Del\_k** (List &L, item &x, **int** k) //Xoa Node k trong danh sach  {  Node \*P=L;  **int** i=**1**;  **if** (k<**1** || k>len(L))  printf("Vi tri xoa khong hop le !"); //kiem tra dieu kien  **else**  {  **if** (k==**1**) Del\_frist(L,x); //xoa vi tri dau tien  **else** //xoa vi tri k != 1  {  **while** (P != NULL && i != k-**1**) //duyet den vi tri k-1  {  P=P->next;  i++;  }  P->next = P->next->next; //cho P tro sang Node ke tiep vi tri k  }  }  } |

#### e) Tìm phần tử có giá trị x trong danh sách

Ta duyệt danh sách cho đến khi tìm thấy hoặc kết thúc và trả về vị trí nếu tìm thấy, ngược lại trả về 0.

|  |
| --- |
| **int** **Search** (List L, item x) //tim x trong danh sach  {  Node \*P=L;  **int** i=**1**;  **while** (P != NULL && P->Data != x) //duyet danh sach den khi tim thay hoac ket thuc danh sach  {  P = P->next;  i++;  }  **if** (P != NULL) **return** i; //tra ve vi tri tim thay  **else** **return** **0**;} //khong tim thay |

**PHẦN BA: NỘI DUNG THỰC HÀNH**

## **I.. Giới thiệu đề tài**

Game **Tic-tac-toe** là một trò chơi phổ biến dùng viết trên bàn cờ giấy có chín ô, 3x3. Hai người chơi, người dùng ký hiệu **O**, người kia dùng ký hiệu **X**, lần lượt điền ký hiệu của mình vào các ô. Người thắng là người thể tạo được đầu tiên một dãy thẳng hàng ký hiệu của mình, ngang dọc hay chéo đều được.

Người Việt thường chơi trò tương tự, gọi là [**Cờ ca-rô**](https://vi.wikipedia.org/wiki/C%E1%BB%9D_ca-r%C3%B4), bàn cờ không giới hạn trong 9 ô, có thể vẽ thêm ô, để kéo rộng ra cho đến khi người nào đạt được một dãy 5 thì thắng cuộc.

**II.. Ý tưởng xây dựng**

-Xây dựng game “ Tic Tac Toe “ với chế độ hai người chơi.

- Vì giao diện xây dựng theo ma trận 3x3 nên bước đầu tiên ta sẽ xây dựng giao diện. Có hai cách thiết kế giao diện là sử dụng mảng 1 chiều hoặc 2 chiều. Trong bài này chúng ta sẽ sử dụng mảng 1 chiều.

- Bước tiếp theo để bắt đầu trò chơi chũng ta cho người chơi 1 chọn kí hiệu X, người chơi 2 chọn ký hiệu O.

- Tiếp đó 2 người chơi lần lượt chọn đánh kí hiệu của mình vào ô mà mình chọn. Mỗi người chơi lần lượt điền vào cho đến khi hết ô trống để đánh.

- Sau đó chúng ta sẽ kiểm tra xem ai thắng ai thua hoặc cả hai đều hòa. Điều kiện để thắng khi một trong hai người đều đánh dấu được thẳng hàng theo hàng ngang hoặc hàng dọc hoặc đường chéo thì người chơi đó thắng. Ngược lại nếu cả người chơi đều không đánh được như trên sẽ hòa.

**III.. Viết chương trình**

1.. Xây dựng giao diện game

- Bước 1: Ta tạo một mảng toàn cục kí tự char , với 10 kí tự:

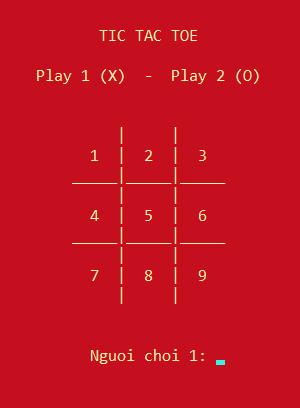
**char** background [**10**]= {'o','1','2','3','4','5','6','7','8','9'};

-Bước 2: Tiến hành làm giao diện

+ Xây dựng hàm có tên giaoDien vì hàm này không có giá trị trả về nên chúng ta dùng kiểu void, như sau:

|  |
| --- |
| **void** **giaoDien**(){  system("cls");  system("color 4E");  printf(" TIC TAC TOE");  printf(" Play 1 (X) - Play 2 (O)**\n\n\n**");  printf(" | | **\n**");  printf(" %c | %c | %c **\n**", background[**1**], background[**2**], background[**3**]);  printf("\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_**\n**");  printf(" | | **\n**");  printf(" %c | %c | %c **\n**", background[**4**],background[**5**],background[**6**]);  printf("\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_**\n**");  printf(" | | **\n**");  printf(" %c | %c | %c **\n**", background[**7**], background[**8**], background[**9**]);  printf(" | | **\n**");  } |

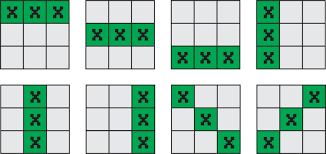
+ Kết quả khi chạy chương trình:



- Bước 3: Ta xây dựng hàm xetDieuKienThangThua():

Xây dựng hàm: làm nhiệm vụ xét thắng thua hay hòa giữa 2 người chơi. Ở đây ta quy ước nếu hàm trả về giá là 1( tức là người chơi 1 hoặc 2 thắng). Nếu hàm trả về 0 (thì người chơi 1 và 2 hòa nhau) và nếu trả về -1 là ta sẽ gọi lại hàm giaoDien và tiếp tục trò chơi.

Khi người chơi tạo được 1 đường thẳng của kí hiệu mình chọn thì người chơi đó sẽ thắng. Ta xem hình ảnh bên dưới sẽ hiểu rõ :



Vì có nhiều trường hợp thắng nên chúng ta sẽ sử dụng câu lệnh điều if…else để xét các trường hợp thắng. Đoạn chương trình sau khi hoàn tất:

|  |
| --- |
| **int** **xetDieuKienThangThua**(){  **if** (background[**1**]== background [**2**] && background [**2**]==background[**3**])  **return** **1**;    **else** **if** (background [**4**]==background [**5**]&& background [**5**]==background [**6**])  **return** **1**;    **else** **if**(background [**7**]==background [**8**]&& background [**8**]==background [**9**])  **return** **1**;    **else** **if**(background [**1**]==background [**4**] && background [**4**]==background[**7**])  **return** **1**;    **else** **if** (background[**2**]==background[**5**]&&background [**5**]==background[**8**])  **return** **1**;    **else** **if**(background[**3**]==background[**6**]&&background [**6**]==background[**9**])  **return** **1**;    **else** **if**(background[**1**]==background[**5**]&&background[**5**]==background[**9**])  **return** **1**;    **else** **if**(background[**3**]==background[**5**]&&background[**5**]==background[**7**])  **return** **1**;    **else** **if**(background[**1**]!='1'&&background [**2**]!='2'&& background[**3**]!='3'&&background[**4**]!='4'  &&background[**5**]!='5'&&background[**6**]!='6'&&background[**7**]!='7'&&background[**8**]!='8'&&background[**9**]!=**9**)  **return** **0**;    **else** {  **return** -**1**;  }  } |

- Bước 4: Chúng ta bắt đầu xây dựng hàm playGame() đẻ làm nhiệm vụ chạy game trong lúc 2 người chơi đang chơi. Cụ thể như sau:

+ Ban đầu ta khai báo các biến như sau: Play ( dùng để chỉ người chơi thứ nhất hoặc hai), biến i để gán giá trị cho hàm cần xét, biến o\_chon ( dùng để chọn ô cần nhập), và biến thay đổi kí tự X\_O.

+ Tiếp theo ta cho vòng lặp do-while với điều kiện lặp là -1 để duy trì và chạy game trong vòng lặp ta thực hiện gọi lại hàm giaoDien() và sử dụng toán tử 3 ngôi để thay đổi người chơi 1 và 2 và các kí tự X và O.

+ Tiếp là cho người chơi nhập vào từ bàn phím các số từ 1->9 nếu người chơi nhập số nào thì ở ô số đó đổi thành kí tự X hoặc O và biến nguời chơi cộng thêm 1 đơn vị để người chơi kia nhập và nếu nguời chơi 1 hoặc 2 nhập trùng thì thông báo cho nguời chơi biết và biến người chơi giảm 1 đơn vị để cho người chơi đó nhập lại.

+ Cuối cùng ta gán hàm xetDieuKienThangThua() bằng biến i để xét điều kiện. Nếu giá trị biến đó bằng -1 thì tiếp tục chạy game bỏ vào vòng lặp do-while ở trên. Nếu bằng 1 thì thông báo nguời chơi 1 hoặc 2 thắng và bằng 0 thì cả 2 hòa nhau.

Ta xem đoạn code chương trình:

|  |
| --- |
| **void** **playGame**(){  **int** Play=**1** ,i ,o\_chon;  **char** X\_O;  **do**{  giaoDien();    Play=(Play % **2**) ? **1** : **2**;  printf(" Nguoi choi %d: ",Play);  scanf("%d",&o\_chon);    X\_O=(Play==**1**) ? 'X':'O';    **if**(o\_chon == **1** && background[**1**] =='1')  background[**1**] = X\_O;    **else** **if**(o\_chon == **2** && background[**2**] == '2')  background[**2**] = X\_O;    **else** **if**(o\_chon ==**3** && background[**3**] == '3')  background[**3**] = X\_O;    **else** **if**(o\_chon == **4** && background[**4**] == '4')  background[**4**] = X\_O;    **else** **if**(o\_chon== **5** && background[**5**] =='5')  background [**5**] = X\_O;    **else** **if**(o\_chon == **6** && background[**6**] == '6')  background [**6**] = X\_O;    **else** **if**(o\_chon == **7** && background[**7**] == '7')  background [**7**] = X\_O;    **else** **if**(o\_chon == **8** && background[**8**] == '8')  background [**8**] = X\_O;    **else** **if**(o\_chon == **9** && background[**9**] == '9')  background [**9**] = X\_O;    **else** {  printf(" O nay da duoc nhap hoac khong co**\n**");  Play--;  getch();  }  i=xetDieuKienThangThua();  play++;  }**while**(i == (-**1**));  giaoDien();  **if**(i==**1**){  printf(" Play %d win!!!",--Play);  }  **else**{  printf(" Game hoa!!!");  }  getch();  } |

-Bước 5:

Hoàn thiện chương trình. Sau khi tạo xong các hàm trên ta bắt đầu gọi lại trong hàm main và chạy chương trình.

Chương trình trong hàm main như hình bên dưới:

|  |
| --- |
| #include <stdio.h>  #include <conio.h>  #include<stdlib.h>  #include<string.h>  #include<windows.h>  **char** background [**10**]= {'o','1','2','3','4','5','6','7','8','9'};  **void** **giaoDien**();  **void** **playGame**();  **int** **xetDieuKienThangThua**();  **int** **main**(){  **int** lua\_chon;  system("color 4E");  system("cls");  printf(" GAME TIC TAC TOE \n");  printf(" 1.BAT DAU **\n**");  printf(" 2.THOAT **\n**");  printf(" Nhap lua chon: ");  scanf("%d",&lua\_chon);  **switch** (lua\_chon){  **case** **1**:  system("cls");  playGame();  **break**;  **case** **2**:  exit(**0**);  }  **return** **0**;  } |

Lưu ý: Ở chương trình có sử dụng các lệnh:

* + System(“cls”): dùng để xóa màn sau khi gặp lệnh này.
  + System(“color value”): Dùng để thay đổi màu nền cũng như màu chữ cho chương trình.

Khi sử dụng hai lệnh này ta phải khai báo thư viện **windows.h** và **stdlib.h**

# **PHẦN BỐN: TỔNG KẾT**

## **I.. Kết quả đạt được**

Sau quá trình nguyên cứu và thực hiện đề tài đã đạt được một số kết quả như sau:

* Hoàn thành được một phần cơ bản của game.
* Hoàn thành được một số phần nội dung lý thuyết có hình ảnh ví dụ và code minh họa.

## **II.. Hạn chế**

Do lần đầu nghiên cứu và xây dựng một đề tài nên không thể tránh khỏi những sai xót:

* Nội dung lý thuyết đưa ra chưa đầy đủ còn thiếu một số phần.
* Chế độ cũng như phần chạy game vẫn còn một số lỗi, chưa hoàn thiện.

## **III.. Hướng phát triển đề tài**

Nếu có thời gian và điều kiện sau này em muốn phát triển thêm cho đề tài để hoàn thiện một cách tốt nhất như:

* Xây dựng thêm các chế độ người đấu với máy cho game.
* Hoàn thiện những lỗi khi chạy game và hoàn thiện game.
* Cải thiện giao diện người dùng đẹp hơn.

# TÀI LIỆU THAM KHẢO

[1]. <https://javabypatel.blogspot.com/2017/09/linear-search-algorithm-in-java.html>

[2]. <https://nguyenvanhieu.vn/doc-ghi-file-trong-c/>

[3]. <https://www.hackerearth.com/fr/practice/algorithms/sorting>

[4]. <https://www.programiz.com/c-programming/c-file-input-output>

[5]. <https://cachhoc.net/2014/12/21/lap-trinh-c-bai-13-danh-sach-lien-ket-don-cai-bang-con-tro/>

[6]. Kỹ Thuật Lập trình C Cơ sở và Nâng cao - Gs.Phạm Văn Ất.