**MỤC LỤC**

[**LỜI NÓI ĐẦU** 3](#_Toc26820156)

[**1.** **Tổng quan về thuật toán và sắp xếp** 4](#_Toc26820157)

[**1.1.** **Khái niệm về thuật toán và sắp xếp.** 4](#_Toc26820158)

[**1.2.** **Các tiêu chuẩn đánh giá một thuật toán sắp xếp** 4](#_Toc26820159)

[**2.** **Cơ sở lý thuyết của các thuật toán** 5](#_Toc26820160)

[**2.1.** **Thuật toán sắp xếp nổi bọt (Bubble Sort)** 5](#_Toc26820161)

[2.1.1. Ý tưởng 5](#_Toc26820162)

[2.1.2. Thực thi 5](#_Toc26820163)

[2.1.3. Nhận xét 6](#_Toc26820164)

[**2.2.** **Thuật toán sắp xếp đổi chỗ trực tiếp (Interchange Sort)** 7](#_Toc26820165)

[2.2.1. Ý tưởng 7](#_Toc26820166)

[2.2.2. Thực thi 7](#_Toc26820167)

[2.2.3. Nhận xét 8](#_Toc26820168)

[**2.3.** **Thuật toán sắp xếp chọn (Selection Sort)** 8](#_Toc26820169)

[2.3.1. Ý tưởng 8](#_Toc26820170)

[2.3.2. Thực thi 9](#_Toc26820171)

[2.3.3. Nhận xét 10](#_Toc26820172)

[**2.4.** **Thuật toán sắp xếp chèn (Insertion Sort)** 10](#_Toc26820173)

[2.4.1. Ý tưởng 10](#_Toc26820174)

[2.4.2. Thực thi 10](#_Toc26820175)

[2.4.3. Nhận xét 11](#_Toc26820176)

[**2.5.** **Thuật toán sắp xếp nhanh (Quick Sort)** 11](#_Toc26820177)

[2.5.1. Ý tưởng 11](#_Toc26820178)

[2.5.2. Thực thi 11](#_Toc26820179)

[2.5.3. Nhận xét 12](#_Toc26820180)

[**3.** **Chương trình demo** 12](#_Toc26820181)

[**3.1.** **Chức năng chính** 12](#_Toc26820182)

[**3.2.** **Thiết kế chương trình** 13](#_Toc26820183)

[3.2.1. Cấu trúc chương trình 13](#_Toc26820184)

[3.2.2. Chương trình mô phỏng 14](#_Toc26820185)

[3.2.3. Quá trình mô phỏng thuật toán 16](#_Toc26820186)

[**4.** **Kết luận** 18](#_Toc26820187)

[**4.1.** **Kết quả đạt được** 18](#_Toc26820188)

[**4.2.** **Hạn chế** 18](#_Toc26820189)

[**4.3.** **Bản phân công công việc** 18](#_Toc26820190)

[**5.** **Tài liệu tham khảo** 19](#_Toc26820191)

**LỜI NÓI ĐẦU**

Trong thế giới thực của chúng ta, khái niệm sắp xếp thật là đa dạng và phong phú. Nếu cho rằng, sắp xếp là tiến trình sắp đặt lại các thành phần của một tập hợp các thành phần của một tập hợp gồm nhiều phần tử để đưa chúng về một thứ tự logic nào đó thì đa phần các việc trong cuộc sống của chúng ta đều là sắp xếp. Ví dụ như các cuốn sách trong một thư viện được bố trí một cách hợp lí, các món ăn trên bàn ăn được bày trí một cách đẹp mắt….. Những sự bố trí này là biểu hiện của tính được sắp xếp theo một thứ tự logic nào đó mà người chủ động sắp xếp mong muốn.

Từ những ngày đầu tiên của kỉ nguyên tính toán bằng máy tính điện tử, người ta đã nhận ra rằng trong tổng số các chỉ thị mà máy tính phải thực thi thì đã mất gần ba phần dành cho việc sắp xếp. Đến ngày hôm nay thì tỉ lệ này đã ít hơn, nhưng lý do không nằm ở chỗ nhu cầu về sắp xếp giảm đi mà chính là các giải thuật sắp xếp đã được cải tiến khá hiệu quả. Chúng em thực hiện đề tài này có thể mô phỏng lại quá trình mà các thuật toán hoạt động, giúp cho việc học tập trở nên dễ dàng hơn, hiểu rõ được quá trình thực hiện của các thuật toán sắp xếp để đưa ra các ưu khuyết điểm của từng thuật toán nhằm tối ưu hóa thời gian thực hiện của một chương trình áp dụng sắp xếp logic.

Trong bản cáo Nhóm thực hiện trình bày dưới đây sẽ đề cập đến một số thuật toán sắp xếp cơ sở như *Thuật toán sắp xếp nổi bọt (Bubble Sort),* *Thuật toán sắp xếp đổi chỗ trực tiếp (Interchange Sort), Thuật toán sắp xếp chọn (Selection Sort), Thuật toán sắp xếp chèn (Insertion Sort) và Thuật toán sắp xếp nhanh (Quick Sort).*

Mặc dù Nhóm đã rất cố gắng để làm đề tài này nhưng do kinh nghiệm còn thiếu và kiến thức nắm chưa sâu nên không thể tránh khỏi những thiếu sót. Nhóm rất mong nhận được ý kiến đóng góp của các Thầy/Cô để lần sau có thể làm tốt hơn.

# **Tổng quan về thuật toán và sắp xếp**

* 1. **Khái niệm về thuật toán và sắp xếp.**

Thuật toán, còn gọi là giải thuật, là một tập hữu hạn các chỉ thị hay phương cách được định nghĩa rõ ràng cho việc hoàn tất một số sự việc từ một trạng thái ban đầu cho trước; khi các chỉ thị này được áp dụng triệt để thì sẽ dẫn đến kết quả sau cùng như đã dự đoán trước. Nói cách khác, thuật toán là một bộ các quy tắc hay quy trình cụ thể nhằm giải quyết một số vấn đề trong một số bước hữu hạn, hoặc nhằm cung cấp một kết quả từ một tập hợp các dữ kiện đưa vào.

Sắp xếp là một quá trình biến đổi một danh sách các đổi tượng thành một danh sách thỏa mãn một trật tự logic nào đó. Sắp xếp đóng vài trò quan trọng trong tìm kiếm dữ liệu. Trong thiết kế thuật toán, ta cũng thường xuyên cần đến sắp xếp, nhiều thuật toán được dựa trên ý tưởng xử lý các đối tượng theo một thứ tự xác định. Các thuật toán sắp xếp được chia thành 2 loại: sắp xếp trong và sắp xếp ngoài. Sắp xếp trong được thực hiện khi mà các đối tượng cần sắp xếp được lưu trong bộ nhớ trọng của máy tính dưới dạng mảng, khi các đối tượng cần được sắp xếp quá lớn cần lưu ở bộ nhớ ngoài dưới dạng file, ta cần sử dụng phương pháp sắp xếp ngoài. Báo cáo này trình bài 5 thuật toán sắp xếp trong.

* 1. **Các tiêu chuẩn đánh giá một thuật toán sắp xếp**

Thông thường để đánh giá một thuật toán, người ta thường dựa vào hai tiêu chí để đánh giá, đó là : *Độ phức tạp không gian* và *Độ phức tạp thời gian*.

***Độ phức tạp không gian*** gắn liền với cấu trúc dữ liệu cụ thể được dùng để thực hiện thuật toán.

***Độ phức tạp thời gian*** của một thuật toán có thể biểu diễn qua số phép toán thực hiện thuật toán đó khi các giá trị đầu vào có kích thước xác định.

# **Cơ sở lý thuyết của các thuật toán**

* 1. **Thuật toán sắp xếp nổi bọt (Bubble Sort)**
     1. Ý tưởng

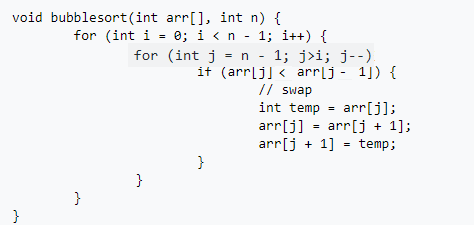
**Sắp xếp nổi bọt** (Bubble Sort ) là một [thuật toán sắp xếp](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_s%E1%BA%AFp_x%E1%BA%BFp) đơn giản, với thao tác cơ bản là so sánh hai phần tử **kề** nhau, nếu chúng chưa đứng đúng thứ tự thì đổi chỗ (*swap*). Có thể tiến hành từ trên xuống (bên trái sang) hoặc từ dưới lên (bên phải sang). **Sắp xếp nổi bọt** còn có tên là *sắp xếp bằng so sánh trực tiếp*. Nó sử dụng phép so sánh các phần tử nên là một giải thuật [sắp xếp kiểu so sánh](https://vi.wikipedia.org/w/index.php?title=S%E1%BA%AFp_x%E1%BA%BFp_ki%E1%BB%83u_so_s%C3%A1nh&action=edit&redlink=1).

**Sắp xếp từ trên xuống:** Giả sử dãy cần sắp xếp có *n* phần tử. Khi tiến hành từ trên xuống, ta so sánh hai phần tử đầu, nếu phần tử đứng trước lớn hơn phần tử đứng sau thì đổi chỗ chúng cho nhau. Tiếp tục làm như vậy với cặp phần tử thứ hai và thứ ba và tiếp tục cho đến cuối tập hợp dữ liệu, nghĩa là so sánh (và đổi chỗ nếu cần) phần tử thứ *n*-1 với phần tử thứ *n*. Sau bước này phần tử cuối cùng chính là phần tử lớn nhất của dãy. Sau đó, quay lại so sánh (và đổi chố nếu cần) hai phần tử đầu cho đến khi gặp phần tử thứ *n*-2... Nếu trong một lần duyệt, không phải đổi chỗ bất cứ cặp phần tử nào thì danh sách đã được sắp xếp xong.

**Sắp xếp từ dưới lên :** Sắp xếp từ dưới lên so sánh (và đổi chỗ nếu cần) bắt đầu từ việc so sánh cặp phần tử thứ *n*-1 và *n*. Tiếp theo là so sánh cặp phần tử thứ *n*-2 và *n*-1,... cho đến khi so sánh và đổi chỗ cặp phần tử thứ nhất và thứ hai. Sau bước này phần tử nhỏ nhất đã được **nổi** lên vị trí trên cùng (nó giống như hình ảnh của các "bọt" khí nhẹ hơn được nổi lên trên). Tiếp theo tiến hành với các phần tử từ thứ 2 đến thứ *n*.

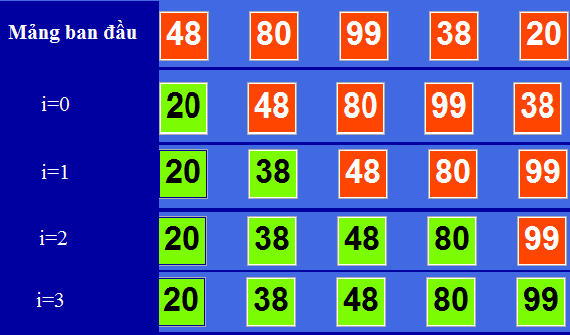
* + 1. Thực thi

Nội dung của phương pháp sắp xếp nổi bọt được trình bày bằng đoạn mã viết bằng ngôn ngữ C++ sau:



**Hàm BubbleSort**

Chỉ số i của dòng lặp for( ) đầu tiên chỉ cần duyệt qua n-1 phần tử và sau khi kết thúc thì phần tử cuối hiển nhiên đã được sắp xếp. Ở vòng lặp for() thứ hai, bất cứ khi nào phát hiện phần tử thứ j bé hơn phần tử thứ j-1 thì sẽ tiến hành đổi chỗ hai giá trị. Quá trình thực hiện được mô tả như sau:



**Các bước sắp xếp mảng có 5 phần tử**

* + 1. Nhận xét

Thuật toán sắp xếp nổi bọt là một thuật toán sắp xếp đơn giản, tuy nhiên giải thuật này không thích hợp với các tập dữ liệu lớn. Giải thuật này là giải thuật chậm nhất trong các thuật toán sắp xếp cơ bản. Nguyên nhân chính là bởi vì thuật toán này lạm dụng các thao tác hoán đổi giá trị.

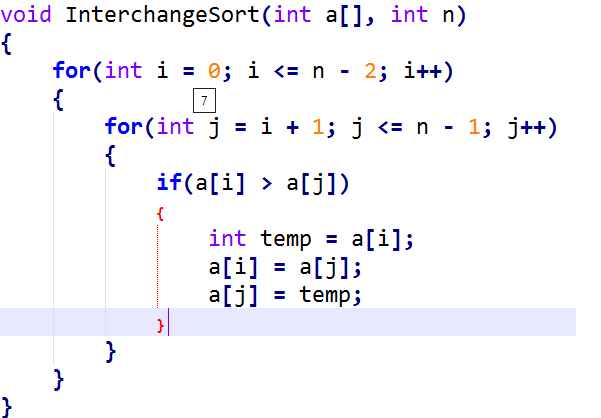
Độ phức tập thuật toán trong trường hợp tốt nhất là O(n) và trong trường hợp xấu nhất là O(n2).

* 1. **Thuật toán sắp xếp đổi chỗ trực tiếp (Interchange Sort)**
     1. Ý tưởng

Phương pháp này liên tục so sánh và đổi chỗ các cặp phần tử đứng kề nhau cho đến khi toàn bộ dãy được sắp xếp. Nói cách khác, ta không quan tâm đến từng phần tử cụ thể và đưa nó về đúng vị trí mà sẽ thực hiện việc đổi chỗ hay giá trị nếu ai > ai+1. Như vậy phần tử có giá trị lớn hơn sẽ được đẩy về cuối mảng và phần tử nhỏ hơn sẽ được đẩy về đầu mảng.

* + 1. Thực thi

Hàm Interchange Sort viết bằng ngôn ngữ C++:



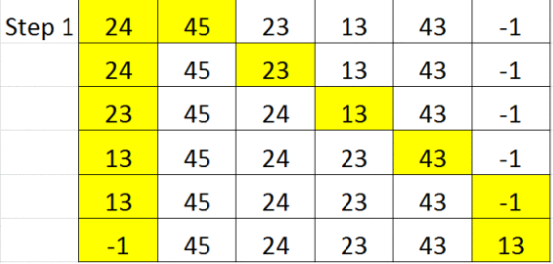
**Hàm I-Sort**

Thuật toán Interchange Sort sẽ duyệt qua tất cả các cặp giá trị trong mảng và hoán vị 2 giá trị trong 1 cặp nếu cặp giá trị đó là nghịch thế.

Ví dụ: Ta có mảng ban đầu



Vòng lặp đầu tiên cho ta bảng mô tả các bước thực hiện thuật toán sau đây :



Sau khi kết thúc vòng lặp đầu tiên, giá trị a[0] đã được sắp xếp đúng thứ tự, cứ như v cho đến khi thuật toán chay n-1 lần vòng lặp for với biến i, toàn bộ mảng đã được sắp xếp đúng vị trí.

* + 1. Nhận xét

Số phép so sánh của thuật toán này không đổi, không phụ thuộc vào tình trạng ban đầu của dãy. Trong trường hợp tốt nhất, tức dãy đã được sắp xếp hoàn toàn thì không cần phải hoán vị, nên số phép hoán vị là: 0.  
Trường hợp xấu nhất, tức dãy có thứ tự ngược với yêu cầu. Mỗi lần so sánh ta lại phải hoán vị nên số phép hoán vị là: n(n - 1)/2.

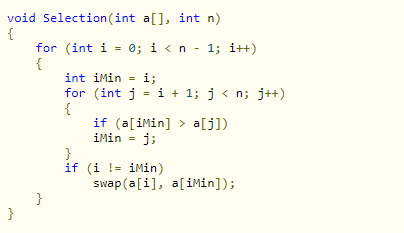
Độ phức tạp của thuật toán là O(n2).

* 1. **Thuật toán sắp xếp chọn (Selection Sort)**
     1. Ý tưởng

**Sắp xếp chọn (Selection Sort)** là một [thuật toán sắp xếp](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_s%E1%BA%AFp_x%E1%BA%BFp) đơn giản, dựa trên việc so sánh tại chỗ. Chọn phần tử nhỏ nhất trong n phần tử ban đầu, đưa phần tử này về vị trí đúng là đầu tiên của dãy hiện hành. Sau đó không quan tâm đến nó nữa, xem dãy hiện hành chỉ 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 hiện hành chỉ còn một phần tử. Dãy ban đầu có n phần tử, vậy tóm tắt ý tưởng thuật toán là thực hiện n-1 lượt việc đưa phần tử nhỏ nhất trong dãy hiện hành về vị trí đúng ở đầu dãy.

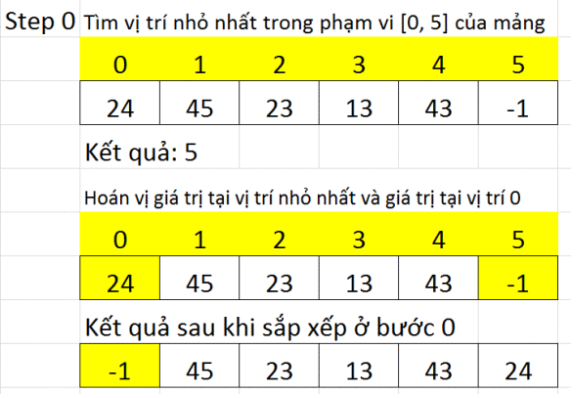
* + 1. Thực thi

Thuật toán Selection Sort được viết bằng ngôn ngữ C++ như sau:



Hàm SelectionSort

Thuật toán này cũng sử dụng n-1 lần duyệt ở vòng for() đầu tiên cho mảng có n phần tử. Biến iMin có vai trò lưu lại vị trí của phần tử có giá trị nhỏ nhất ở thời điểm hiện tại, khi giá trị của phần tử a[j] nhỏ hơn giá trị của phần tử a[iMin], giá trị của iMin sẽ được cập nhật lại. Sau khi thoát khỏi vòng lặp for() ‘B thứ hai với biến chạy là j, giá trị đầu tiên coi như đã được sắp xếp đúng vị trí. Quá trình thực hiện lần duyệt đầu tiên được mô tả trong bản sau :



**Sắp xếp mảng có 5 phần tử bằng S-Sort**

Sau mỗi lần duyệt, phạm vi tìm kiếm của mảng sẽ được giảm xuống 1 đơn vị, cứ như v cho đến khi i=n-1, thuật toán hoàn thành, toàn bộ mảng được sắp xếp.

* + 1. Nhận xét

Đây là một thuật toán đơn giản dễ hiểu, tuy nhiên mỗi lần duyệt của thuật toán này lại hoàn toàn độc lập với nhau, nghĩa là chương trình kiểm tra toàn bộ mảng chưa được sắp và không lưu trữ thông tin cho lần duyệt tiếp theo, hơn nữa, dù mảng đầu vào đã được sắp xếp có thứ tự thì vẫn được xử lí như là chưa được sắp xếp, điều này có nghĩa độ phức tạp thuật toán luôn là O(.

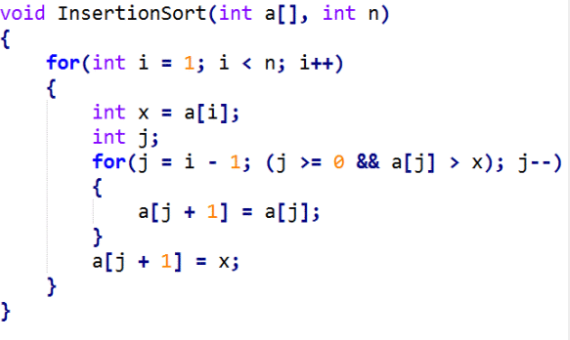
* 1. **Thuật toán sắp xếp chèn (Insertion Sort)**
     1. Ý tưởng

**Insertion Sort** là thuật toán dựa trên ý tưởng xếp một quân bài mới vào dãy các là bài đã được sắp xếp trước đó sao cho thứ tự của dãy mới không thay đổi so với sự sắp xếp ban đầu. Nghĩa là thuật toán sẽ tìm và chèn phần tử ai­ vào vị trí thích hợp trên một đoạn [x;y] đã được sắp xếp.

* + 1. Thực thi

So sánh ai  với ai-1. Giả sử ai < ai-1, nghĩa là ai không thể đứng bên phải ai-1 . Để tạo khoảng trống, ai-1 sẽ được chuyển dời sang phải, đặt vào vị trí I và xem như ai đã tiến sang trái một vị trí. Tiến trình được lặp lại liên tục cho đến khi so sánh ai với a­i-k (i-k 0) và ai-k < ai . Lúc này vị trí để chèn phần tử là i-k+1.

Chương trình xác định được mảng bên trái đã có thứ tự bằng cách xem phần mảng này chỉ chứ phần tử a0. HàmInsertion Sort được viết như sau:



Vòng lặp for() với biến i sẽ lấy từng phần tử a[i] ở mảng bên phải và chèn vào vị trí phù hợp ở nữa mảng bên trái đã được sắp xếp, việc gán i=1 chính là để cho thấy nửa mảng bên trái ban đầu chỉ có phần tử a0.

* + 1. Nhận xét

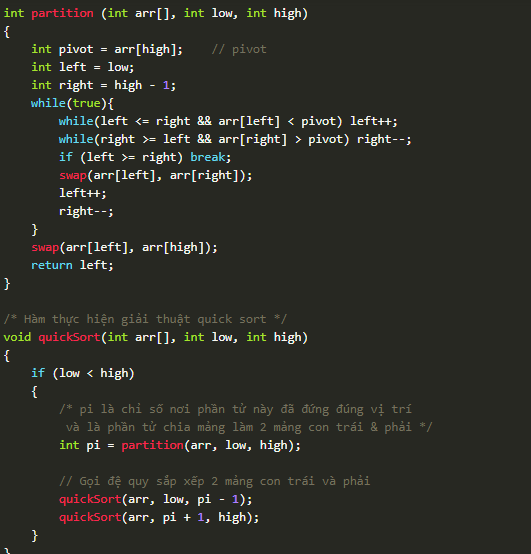
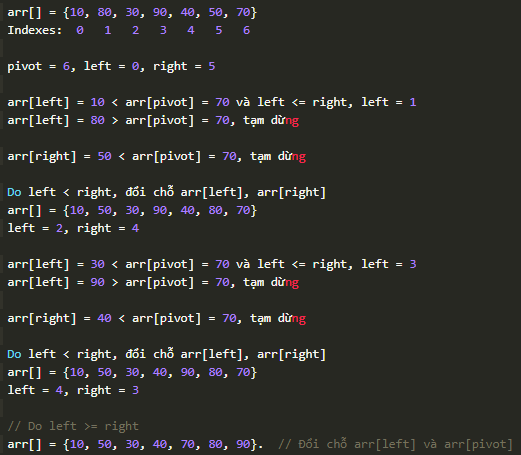
Phương pháp sắp xếp chèn hiệu quả nhất khi làm việc với mảng mà các phần tử đúng sai vị trí tương đối ít. Nếu dãy nhập vào tình cờ có thứ tự thì kỹ thuật này sẽ có độ phức tạp tuyến tính, tốt hơn nhiều so với phương pháp sắp xếp chọn. Tuy nhiên phương pháp này sẽ tỏ ra yếu đuối khi đứng trước lượng đầu vào lớn và các giá trị phân bố không đều. Khi đó cần phải chuyển dời một dãy dài các phần tử để tạo khoảng trống.

* 1. **Thuật toán sắp xếp nhanh (Quick Sort)**
     1. Ý tưởng

**Sắp xếp nhanh** **(*Quicksort*),** còn được gọi là sắp xếp kiểu phân chia (*part sort*) là một [thuật toán sắp xếp](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_s%E1%BA%AFp_x%E1%BA%BFp) phát triển bởi [C.A.R. Hoare](https://vi.wikipedia.org/wiki/Charles_Antony_Richard_Hoare)c sắp thành hai danh sách con. Khác với [sắp xếp trộn](https://vi.wikipedia.org/wiki/S%E1%BA%AFp_x%E1%BA%BFp_tr%E1%BB%99n), *Quicksort* chia danh sách cần sắp xếp {\displaystyle a[1..n]}a[1….n] thành hai danh sách con có kích thước tương đối bằng nhau nhờ chỉ số đứng giữa danh sách, sắp xếp nhanh chia nó thành hai danh sách bằng cách so sánh từng phần tử của danh sách với một phần tử được chọn được gọi là phần tử chốt. Những phần tử nhỏ hơn hoặc bằng phần tử chốt được đưa về phía trước và nằm trong danh sách con thứ nhất, các phần tử lớn hơn chốt được đưa về phía sau và thuộc danh sách đứng sau. Cứ tiếp tục chia như vậy tới khi các danh sách con đều có độ dài bằng 1.

* + 1. Thực thi

Mấu chốt chính của thuật toán quick sort là việc phân đoạn dãy số (Xem hàm partition() ). Mục tiêu của công việc này là: Cho một mảng và một phần tử x là pivot. Đặt x vào đúng vị trí của mảng đã sắp xếp. Di chuyển tất cả các phần tử của mảng mà nhỏ hơn x sang bên trái vị trí của x, và di chuyển tất cả các phần tử của mảng mà lớn hơn x sang bên phải vị trí của x. Khi đó ta sẽ có 2 mảng con: mảng bên trai của x và mảng bên phải của x. Tiếp tục công việc với mỗi mảng con(chọn pivot, phân đoạn) cho tới khi mảng được sắp xếp. Thuật toán phân đoạn, hàm QuickSort và quá trình phân đoạn được viết bằng đoạn mã giả c++ như sau:

** **

Sau khi tiến hành phân đoạn, lấy phần tử cuối làm phần tử chốt rồi chia mảng theo phần tử chốt, để dụng hàm QuickSort một các đệ quy lần lượt với mảng con bên trái rồi mảng con bên phải cho đến mảng đã được sắp xếp.

* + 1. Nhận xét

Độ phức tạp của thuật toán QuickSort phụ thuộc vào cách chọn phần tử pivot. Trường hợp tốt nhất của thuật toán là pivot nằm ở vị trí medium và trường hợp xấu nhất của thuật toán nằm ở vị trí min, max.

QuickSort là thuật toán sắp xếp chạy nhanh nhất ở trường hợp dữ liệu vào trung bình và sử dụng tốt cho danh sách liên kết đơn, tuy nhiên ở trường hợp xấu nhất thuật toán tốn rất nhiều thời gian vì sử dụng hàm đệ quy và thường không ổn định với dữ liệu đầu vào lớn.

# **Chương trình demo**

* 1. **Chức năng chính**

Chương trình mô phỏng các thuật toán sắp xếp có các chức năng chính:

- Nhập số phần tử của mảng cần được mô phỏng sắp xếp

- Tạo ngẫu nhiên các đối tượng là số phần tử cần được mô phỏng sắp xếp

- Chọn thuật toán cần mô phỏng

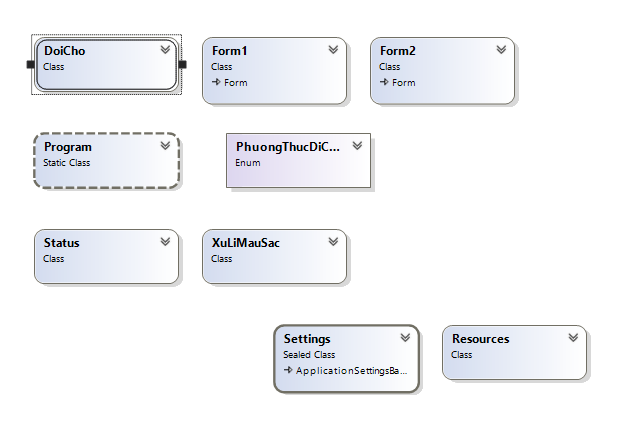
- Chọn tốc độ mô phỏng thuật toán

- Hiển thị thời giạn chương trình mô phỏng thuật toán

- Hiển thị thông tin sinh viên thực hiện

Cấu trúc tĩnh của chương trình được mô tả bằng Class-Diagram trên. Trong đó các *Class Program và Status* là Class hệ thống của chương trình.

* 1. **Thiết kế chương trình**
     1. Cấu trúc chương trình



*ClassDiagram của chương trình*

Cấu trúc chương trình được thiết kế như hình trên bao gồm các nhiệm vụ:

- Class *DoiCho* chứa Constructor khởi tạo giá trị và cập nhập lại giá trị cho đối tượng

- Static Class Program chứa hàm main của chương trình

- Enum PhuongThucDiChuyen khai báo một Property định nghĩa cách di chuyển của các đối tượng trong chương trình

- Class XuLiMauSac đóng vai trò cập nhật giao diện màu sắc cho các đối tượng trong quá trình mô phỏng

- Class Form1 xử lí toàn bộ các thông tin mà người dùng thao tác trên giao diện chính của chương trình

Toàn bộ chương trình mô phỏng được viết bằng ngôn ngữ lập trình C# trên phần mềm lập trình Microsoft visual studio.

* + 1. Chương trình mô phỏng

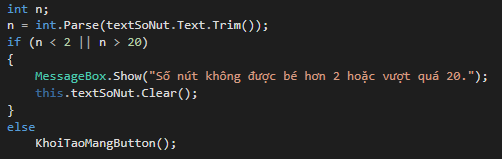
Giao diện chính của phần mềm mô phỏng như hình trên bao gồm:

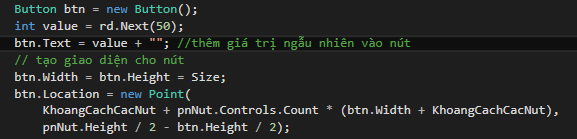


Dữ liệu đầu vào là một số nguyên bé hơn 20 thể hiện số lượng phần tử của mảng cần được mô phỏng. Nhấn vào ô *Vẽ ô,* các đối tượng cần được mô phỏng sẽ được tạo ra, ví dụ khi người dùng nhập vào giá trị ban đầu là 5, giao diện chương trình sẽ được cập nhật và 5 đối tượng được mô phỏng sẽ được tạo ra, như hình bên dưới:



Button vẽ ô sẽ lấy dữ liệu đầu vào sau đó khởi tạo các đối tượng sắp xếp, và khởi tạo các giá trị sắp xếp mang một cách ngẫu nhiên, đoạn code sau đây thực hiện các công việc trên:

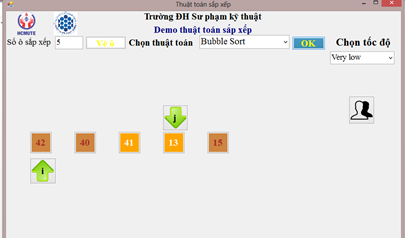




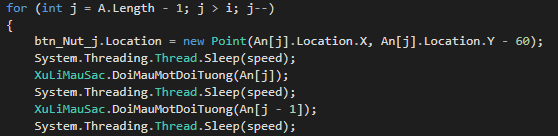
Giá trị nhập vào nếu vượt quá số lượng mà chương trình có thể mô phỏng, một thông báo sẽ được hiện ra để cảnh báo người dùng thay đổi giá trị ban đầu. Khi người dùng chọn thuật toán cần mô phỏng, tốc độ mô phỏng và nhấn *OK,* quá trình mô phỏng sẽ bắt đầu, ComboBox chọn thuật toán được thiết kế đánh thứ tự từ 0->4 từ trên xuống. Khi người dụng chọn một thuật toán, chương trình sẽ nhận số thứ tự tương ứng với thuật toán để gọi chương trình thực hiện. Tương tự như ComboBox chọn thuật toán, ComboBox chọn tốc độ cũng được đánh số từ 0->2 và gọi chương trình thực hiện.

* + 1. Quá trình mô phỏng thuật toán

Khi người dùng đã nhập tất cả các yêu cầu của mình vào giao diện của chương trình mô phỏng, chương trình sẽ bắt đầu tiến hành thực hiện từng bước sắp xếp của một thuật toán sắp xếp mà người dùng đã chọn, giao diện đầu tiên của chương trình sau khi bắt đầu quá trình mô phỏng như sau:

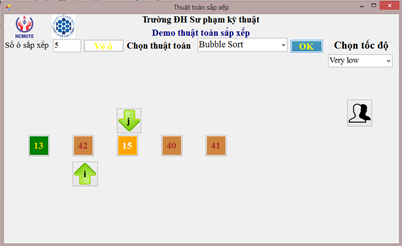


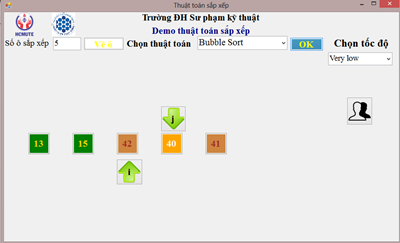
Mũi tên chỉ mục i có vai trò hiển thị số lần lặp so sánh các phần tử trong mảng, mũi tên chỉ mục j hiển thị phần tử đang được chọn để so sánh với các phần tử khác trong mảng, hai phần tử đang được so sanh với nhau sẽ được cập nhật một màu sắc khác so với các phần tử còn lại để phân biệt, các công việc trên được xử lí bởi đoạn code sau đây:



Sau khi so sánh hai đối tượng tương ứng trong thuật toán, nếu hai giá trị đó tương ứng không đúng thứ tự thì sẽ được hoán đổi vị trí cho nhau như sau:



Cứ như vậy cho đến N-1 lần so sánh, phần tử đầu tiên sẽ được sắp xếp đúng vị trí. Sau đó, khi vị trí của đối tượng mô phỏng đã ở đúng vị trí, màu sắc của đối tượng sẽ được cập nhật lại, thể hiện rằng đối tượng đã được thao tác xong và chương trình sẽ tiếp tục xử lí các đối tượng còn lại, cứ như vậy tiếp tục N-1 lần lặp tiếp theo, toàn bộ các đối tượng sẽ được sắp xếp hoàn toàn.



Khi toàn bộ các đối tượng đã được thao tác, toàn bộ giao diện chương trình một lần nữa được cập nhật lại, các mũi tên chỉ mục sẽ biến mất đồng thời các đối tượng sắp xếp cũng sẽ cập nhật trạng thái màu sắc đồng nhất thể hiện quá trình thao tác đã kết thúc.

# **Kết luận**

* 1. **Kết quả đạt được**

Qua đồ án mô học này có thể giúp sinh viên hiểu hơn về 5 thuật toán sắp xếp cơ bản, quá trình thực hiện từng bước của mỗi thuật toán, về lập trình hướng đối tượng, cách tạo ra một đối tượng cũng như các triển khai một đối tượng. Các tính chất cụ thể của mỗi thuật toán cũng được thể hiện qua chương trình làm cho sinh viên dễ dàng nắm bắt các khái niệm, tính chất cũng như cách dùng.

Chương trình mô phỏng thuật toán này giúp người sử dụng có thể xem được từng bước thực thi chương trình và nhờ vậy có thể hiểu chi tiết được thuật toán cũng như có thể đánh giá được mức độ hiệu quả của thuật toán trong từng trường hợp cụ thể. Song song đó, nhóm sinh viên thực hiện chương trình cũng thu được nhiều kiến thức cũng như kinh nghiệm làm việc trong quá trình thực hiện *Chương trình mổ phỏng thuật toán sắp xếp*, sự hiệu quả phần mềm lập trình và hiểu thêm được về ngôn ngữ lập trình C#.

* 1. **Hạn chế**

Do hạn chế về thời gian, tài nguyên và kiến thức nên bản mô phỏng này hạn chế dữ liệu đầu vào chỉ là các số nguyên và số phần tử không vượt quá 20.

Bản mô này chỉ cho phép người dùng khởi tạo các giá trị của mảng một cách ngẫu nhiên mà không cho phép người dùng nhập giá trị của các phần tử từ bàn phím.

* 1. **Bản phân công công việc**



# **Tài liệu tham khảo**

- Trần Đan Thư-Nguyễn Thanh Phương-Đinh Bá Tiến-Trần Minh Triết-Đặng Bình Phương . *Kỹ Thuật Lập Trình(2014).* Nhà XB Khoa học và Kỹ thuật.

- Alfred V.Aho - John E.Hopcroft – Jeffrey D.Ullman. *Data Structures and Algorithms.* Addison-Wesley Publishing Company.

- Vi.www.wikipedia.com/wiki/Thuật\_toán\_sắp\_xếp.

- https://sharecode.vn/source-code/mo-phong-thuat-toan-sap-xep.