BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

---🙢🕮🙠---



**BÁO CÁO THỰC TẬP CƠ SỞ**

**ĐỀ TÀI: MÔ PHỎNG CÁC THUẬT TOÁN SẮP XẾP**

**Giáo viên hướng dẫn: Bùi Chí Thành**

**Sinh viên thực hiện: Trần Duy Phát**

**MSSV: 63135054**

**Khánh Hoà – 2023**

# 

# Mục lục

[Mục lục 2](#_Toc155522433)

[Danh mục hình vẽ 5](#_Toc155522434)

[Danh mục bảng biểu 6](#_Toc155522435)

[Chương 1. Giới thiệu đề tài 7](#_Toc155522436)

[1.1 Lý do chọn đề tài 7](#_Toc155522437)

[1.2 Mục tiêu và nhiệm vụ 7](#_Toc155522438)

[Chương 2. Cơ sở lý thuyết 8](#_Toc155522439)

[2.1 Ngôn ngữ C# 8](#_Toc155522440)

[2.1.1 Khái niệm 8](#_Toc155522441)

[2.1.2 Ứng dụng của ngôn ngữ lập trình C# 8](#_Toc155522442)

[2.2 Thuật toán 8](#_Toc155522443)

[2.2.1 Khái niệm thuật toán 8](#_Toc155522444)

[2.2.2 Đặc điểm thuật toán 8](#_Toc155522445)

[2.2.3 Tổng hợp các thuật toán sắp xếp hiện nay 9](#_Toc155522446)

[Chương 3. Chương trình ứng dụng thuật toán sắp xếp 10](#_Toc155522447)

[3.1 Sơ lược mô phỏng minh họa trên C# 10](#_Toc155522448)

[3.1.1 Các chức năng 10](#_Toc155522449)

[3.1.2 Các kỹ thuật xử lý 10](#_Toc155522450)

[3.1.2.1 Phân tích hiệu ứng 11](#_Toc155522451)

[3.1.2.2 Phân tích chức năng 11](#_Toc155522452)

[3.1.3 Các kỹ thuật sử dụng 12](#_Toc155522453)

[3.1.3.1 Xử lý bất đồng bộ với Thread và Task 12](#_Toc155522454)

[3.1.3.2 Xử lý việc thay đổi tốc độ sắp xếp 12](#_Toc155522455)

[3.1.3.3 Xử lý việc di chuyển các phần tử và các biến 13](#_Toc155522456)

[3.1.3.4 Xử lý việc tính thời gian thực hiện quá trình sắp xếp 13](#_Toc155522457)

[3.1.3.5 Xử lý debug 13](#_Toc155522458)

[3.2 Các thuật toán sắp xếp đơn giản 14](#_Toc155522459)

[3.2.1 Thuật toán Interchange Sort (Đổi chỗ trực tiếp) 14](#_Toc155522460)

[3.2.1.1 Nghịch thế 14](#_Toc155522461)

[3.2.1.2 Cài đặt mã giả 14](#_Toc155522462)

[3.2.1.3 Đánh giá độ phức tạp 15](#_Toc155522463)

[3.2.2 Thuật toán Selection Sort (Sắp xếp chọn trực tiếp) 15](#_Toc155522464)

[3.2.2.1 Ý tưởng giải thuật 15](#_Toc155522465)

[3.2.2.2 Cài đặt mã giả 15](#_Toc155522466)

[3.2.2.3 Đánh giá độ phức tạp 16](#_Toc155522467)

[3.2.3 Thuật toán Bubble Sort (Sắp xếp nổi bọt) 16](#_Toc155522468)

[3.2.3.1 Ý tưởng thuật toán 16](#_Toc155522469)

[3.2.3.2 Cài đặt mã giả 16](#_Toc155522470)

[3.2.3.3 Đánh giá độ phức tạp 17](#_Toc155522471)

[3.2.4 Thuật toán Insertion Sort (Sắp xếp chèn trực tiếp) 17](#_Toc155522472)

[3.2.4.1 Ý tưởng thuật toán 17](#_Toc155522473)

[3.2.4.2 Cài đặt mã giả 17](#_Toc155522474)

[3.2.4.3 Đánh giá độ phức tạp 18](#_Toc155522475)

[3.3 Các thuật toán sắp xếp sử dụng đệ qui 18](#_Toc155522476)

[3.3.1 Thuật toán Quick Sort (Sắp xếp nhanh) 18](#_Toc155522477)

[3.3.1.1 Ý tưởng thuật toán 18](#_Toc155522478)

[3.3.1.2 Cài đặt mã giả 18](#_Toc155522479)

[3.3.1.3 Đánh giá độ phức tạp 19](#_Toc155522480)

[3.3.2 Thuật toán Merge Sort (Sắp xếp trộn) 19](#_Toc155522481)

[3.3.2.1 Ý tưởng thuật toán 19](#_Toc155522482)

[3.3.2.2 Cài đặt mã giả 19](#_Toc155522483)

[3.3.2.3 Đánh giá độ phức tạp 20](#_Toc155522484)

[3.4 Giao diện phần mền 20](#_Toc155522485)

[3.4.1 Khởi tạo dãy số 20](#_Toc155522486)

[3.4.2 Bảng điều khiển 21](#_Toc155522487)

[3.4.3 Bảng lựa chọn thuật toán 21](#_Toc155522488)

[3.4.4 Hiện ý tưởng thuật toán, dãy đang sắp xếp và hiện code C++ 21](#_Toc155522489)

[3.4.5 Thuật toán Interchange Sort 22](#_Toc155522490)

[3.4.6 Thuật toán Selection Sort 24](#_Toc155522491)

[3.4.7 Thuật toán Bubble Sort 25](#_Toc155522492)

[3.4.8 Thuật toán Insertion Sort 26](#_Toc155522493)

[3.4.9 Thuật toán Quick Sort 27](#_Toc155522494)

[3.4.10 Thuật toán Merge Sort 29](#_Toc155522495)

[Chương 4. Kết luận 31](#_Toc155522496)

[4.1 Kết quả đạt được 31](#_Toc155522497)

[4.2 Hạn chế 31](#_Toc155522498)

[4.3 Hướng phát triển 31](#_Toc155522499)

[Tài liệu tham khảo 32](#_Toc155522500)

[Phụ lục 33](#_Toc155522501)

# Danh mục hình vẽ

[Hình 1 Bảng khởi tạo dãy số. 20](#_Toc155522409)

[Hình 2 Khởi tạo dãy số bằng nhập tay. 21](#_Toc155522410)

[Hình 3 Bảng điều khiển. 21](#_Toc155522411)

[Hình 4 Bảng Bảng lựa chọn thuật toán. 21](#_Toc155522412)

[Hình 5 Ý tưởng thuật toán, dãy đang sắp xếp. 22](#_Toc155522413)

[Hình 6 Hiện code C++. 22](#_Toc155522414)

[Hình 7 Interchange Sort trước khi thực hiện sắp xếp. 23](#_Toc155522415)

[Hình 8 Interchange Sort sau khi thực hiện sắp xếp. 24](#_Toc155522416)

[Hình 9 Selection Sort trước khi thực hiện sắp xếp. 24](#_Toc155522417)

[Hình 10 Selection Sort sau khi thực hiện sắp xếp. 25](#_Toc155522418)

[Hình 11 Bubble Sort trước khi thực hiện sắp xếp. 25](#_Toc155522419)

[Hình 12 Bubble Sort sau khi thực hiện sắp xếp. 26](#_Toc155522420)

[Hình 13 Insertion Sort trước khi thực hiện sắp xếp. 27](#_Toc155522421)

[Hình 14 Insertion Sort sau khi thực hiện sắp xếp. 27](#_Toc155522422)

[Hình 15 Quick Sort trước khi thực hiện sắp xếp. 28](#_Toc155522423)

[Hình 16 Quick Sort sau khi thực hiện sắp xếp. 28](#_Toc155522424)

[Hình 17 Merge Sort trước khi thực hiện sắp xếp. 29](#_Toc155522425)

[Hình 18 Merge Sort sau khi thực hiện sắp xếp. 30](#_Toc155522426)

# Danh mục bảng biểu

[Bảng 3. 1 Bảng đánh giá độ phức tạp của Interchange Sort. 15](#_Toc155522427)

[Bảng 3. 2 Bảng đánh giá độ phức tạp của Selection Sort. 16](#_Toc155522428)

[Bảng 3. 3 Bảng đánh giá độ phức tạp Bubble Sort. 17](#_Toc155522429)

[Bảng 3. 4 Bảng đánh giá độ phức tạp Insertion Sort. 18](#_Toc155522430)

[Bảng 3. 5 Bảng đánh giá độ phức tạp Quick Sort. 19](#_Toc155522431)

[Bảng 3. 6 Bảng đánh giá độ phức tạp Merge Sort. 20](#_Toc155522432)

1. Giới thiệu đề tài
   1. Lý do chọn đề tài

Như các bạn đã biết, cấu trúc dữ liệu và thuật toán là một trong những môn học cơ bản của sinh viên công nghệ thông tin. Cấu trúc dữ liệu và thuật toán được coi là hai thành phần quan trọng nhất của lập trình (Chương trình = Cấu trúc dữ liệu + Thuật toán). Trong khoa học máy tính, cấu trúc dữ liệu là một cách lưu trữ dữ liệu trong máy tính để có thể sử dụng một cách hiệu quả. Thuật toán (còn gọi là thuật toán trong tiếng Anh Algorithms) là một tập hợp hữu hạn các hướng dẫn được thiết kế để máy tính thực hiện theo một thứ tự cụ thể nhằm đạt được kết quả mong muốn.

Để giúp học sinh hình dung và hiểu rõ hơn về cấu trúc dữ liệu, thuật toán, giáo viên sử dụng mô phỏng thuật toán như một công cụ hỗ trợ để mô phỏng động và đồ họa những thay đổi về thuật toán, cấu trúc dữ liệu trong quá trình thực thi, điều này thường xuyên xảy ra.

Hiện nay, một số lượng lớn các hệ thống mô phỏng thuật toán đang được phát triển, hầu hết chúng phổ biến và phức tạp hơn các hệ thống được sử dụng trong thực tế.

Các thuật toán sắp xếp rất đa dạng và có phạm vi rộng. Vì vậy, nhiệm vụ “Mô phỏng thuật toán sắp xếp” được chọn để xây dựng báo cáo thực hành cơ bản này.

* 1. Mục tiêu và nhiệm vụ

• Nghiên cứu tổng quát về mô phỏng thuật toán.

• Ôn tập và tìm hiểu một số thuật toán sắp xếp

• Xây dựng chương trình mô phỏng để hình dung rõ hơn quá trình phân loại.

• Áp dụng nghiên cứu của bạn để tạo một bản trình diễn mô phỏng thuật toán sắp xếp trên màn hình bảng điều khiển và trong C#.

1. Cơ sở lý thuyết
   1. Ngôn ngữ C#
      1. Khái niệm

Ngôn ngữ C# (phát âm là "C Sharp") là ngôn ngữ lập trình hướng đối tượng được phát triển bởi Microsoft, làm điểm khởi đầu cho bản thiết kế .NET của họ.

* + 1. Ứng dụng của ngôn ngữ lập trình C#

• *Ứng dụng Windows*: Với sự trợ giúp của .Net framework, “C#” được sử dụng để phát triển các ứng dụng dựa trên Windows cho máy tính để bàn. Nhiều ứng dụng Windows phổ biến như công cụ Microsoft Office, Skype, Photoshop và Visual Studio được phát triển bằng ngôn ngữ này.

• *Thành phần và Điều khiển*: Thành phần và điều khiển là các thư viện có thể được sử dụng để tạo ra thứ gì đó dễ dàng phân phối và chia sẻ. Thư viện GPS là một ví dụ tuyệt vời về thư viện có thể được tạo bởi một lập trình viên và dễ dàng phân phối cho các lập trình viên khác để sử dụng trong ứng dụng của họ. Nó cũng được sử dụng để xây dựng các thành phần máy chủ và hơn thế nữa.

• *Ứng dụng web*: Với sự trợ giúp của .NET framework, C# có khả năng tạo nhiều ứng dụng web bằng ASP.NET. Đây là một ngôn ngữ phổ biến khác mà bất kỳ ai cũng có thể học ngay lập tức khi muốn chạy các ứng dụng web đúng cách trên máy chủ web. Các ứng dụng Windows chạy cả trên máy chủ và trong trình duyệt của máy khách, tùy thuộc vào cách viết mã. Nếu C# được sử dụng làm mã hóa trong phần phụ trợ, mã C# sẽ chạy trên máy chủ và HTML Frontend chạy trong trình duyệt của máy khách.

* 1. Thuật toán
     1. Khái niệm thuật toán

Thuật toán là một chuỗi hữu hạn các thao tác được sắp xếp theo một thứ tự nhất định sao cho sau khi thực hiện chuỗi thao tác này, từ đầu vào của bài toán ta nhận được đầu ra cần tìm.

* + 1. Đặc điểm thuật toán

Các thuật toán có một số thuộc tính chung, đó là:

• *Đầu vào*: Một thuật toán có các giá trị đầu vào từ một bộ xác định.

• *Đầu ra*: Từ mỗi bộ giá trị đầu vào, thuật toán sẽ tạo ra các giá trị đầu ra. Các giá trị đầu ra là giải pháp cho vấn đề.

• *Tính tất định*: Các bước của thuật toán phải được xác định chính xác.

• *Tính chính xác*: Một thuật toán phải đưa ra các giá trị đầu ra chính xác cho từng bộ giá trị đầu vào.

• *Tính hữu hạn*: Một thuật toán phải tạo ra các giá trị đầu ra sau một số bước hữu hạn (có thể rất lớn) cho mỗi bộ đầu vào.

• *Tính tổng quát*: Thuật toán cần áp dụng được cho tất cả các bộ dữ liệu đầu vào của bài toán chứ không chỉ một bộ giá trị đầu vào cụ thể.

* + 1. Tổng hợp các thuật toán sắp xếp hiện nay

Trong khoa học máy tính và toán học, thuật toán sắp xếp là thuật toán sắp xếp các phần tử của danh sách (hoặc mảng) theo thứ tự tăng dần (hoặc giảm dần). Chúng ta thường xét trường hợp các phần tử cần sắp xếp là các số.

Sắp xếp là quá trình xử lý danh sách các mục (hoặc thông tin) để sắp xếp chúng theo một thứ tự đáp ứng một tiêu chuẩn nhất định dựa trên nội dung thông tin được lưu trữ trong mỗi mục.

Hiệu quả của mỗi thuật toán phụ thuộc vào thuộc tính của cấu trúc dữ liệu cụ thể mà nó ảnh hưởng.

Có nhiều thuật toán sắp xếp: Selection Sort, Bubble Sort, Insertion Sort, Interchange Sort, Quick Sort, Merge Sort,… Nhưng trong báo cáo này, tôi muốn trình bày các giải pháp sắp xếp cổ điển. thuật toán để có cái nhìn khoa học hơn về thuật toán sắp xếp.

1. Chương trình ứng dụng thuật toán sắp xếp
   1. Sơ lược mô phỏng minh họa trên C#
      1. Các chức năng

Trong mô phỏng minh họa bằng C#, phần mềm của tôi có các chức năng sau:

+ Bao gồm 6 thuật toán sắp xếp điển hình trong cấu trúc dữ liệu và thuật toán.

+ Tạo ngẫu nhiên một dãy phần tử mới theo từng thuật toán sắp xếp phù hợp.

+ Xóa bảng: Khi xóa bảng, tất cả các mục trên màn hình chính sẽ không còn xuất hiện.

+ Hủy quá trình: Khi hủy quá trình sắp xếp, các mục trên màn hình chính sẽ trở về trạng thái chưa sắp xếp ban đầu.

+ Giới hạn số phần tử có thể thêm vào dãy số (tối đa 15 phần tử).

+ Hỗ trợ chức năng tạm dừng hoặc tiếp tục thực thi ứng dụng.

+ Điều chỉnh tốc độ thực hiện các thuật toán sắp xếp trong quá trình thực hiện.

+ Có cửa sổ hiển thị mã C++ của các thuật toán sắp xếp tương ứng để so sánh với ảnh mô phỏng.

+ Hiển thị ý tưởng thuật toán và hiển thị dãy số thay đổi khi sắp xếp.

+ Khởi tạo dãy số cần sắp xếp bằng cách click chuột vào phần khởi tạo rồi chọn chế độ.

+ Tạo ngẫu nhiên với số phần tử ngẫu nhiên.

+ Nhập thủ công: Nhập từng phần tử với mã số mục tương ứng khi gõ vào ô số mục.

+ Lựa chọn chế độ debug cho phép thực thi từng dòng lệnh theo mã C++ của form chính. + Bảng chưa sắp xếp: Hiển thị bảng trước khi sắp xếp.

+ Hiển thị quá trình sắp xếp và kết quả sau khi sắp xếp dãy số.

+ Hiển thị thời gian sắp xếp

* + 1. Các kỹ thuật xử lý
       1. Phân tích hiệu ứng

Để mô phỏng một phần tử mảng, chúng ta cần tạo một user control tên là Node để lưu giá trị của phần tử mảng và hiển thị trực tiếp trên GUI, người dùng có thể dễ dàng thay đổi tốc độ của quá trình.

Khi các phần tử di chuyển, phần tử đại diện cho thuật toán cũng được chọn lọc từng dòng tương ứng với cách sắp xếp các phần tử.

Việc hiển thị quá trình sắp xếp cũng giúp người dùng dễ dàng theo dõi và hiểu thuật toán hơn.

Ngoài ra, các biến thuật toán cũng được thể hiện trực quan trên biểu mẫu để khớp với vị trí của phần tử và có các giá trị tương ứng với giá trị của các biến thuật toán.

* + - 1. Phân tích chức năng

- **Khởi tạo mảng (3 cách):**

+ Tạo ngẫu nhiên: Tạo một mảng có giá trị và số lượng phần tử bất kỳ.

+ Nhập thủ công: Cho phép người dùng nhập giá trị cho từng trường riêng lẻ. Người dùng có thể thay đổi giá trị của từng phần tử khi cần thiết.

+ Nhập dãy: Yêu cầu người dùng nhập dãy số liên tiếp và tạo mảng phần tử dựa trên dãy số vừa nhập. Người dùng có thể lưu các đơn hàng đã nhập để nhập sau nếu muốn.

**Bảng điều khiển:**

+ Bắt đầu: Bắt đầu sắp xếp các phần tử dựa trên thuật toán đã chọn.

+ Tạm dừng: Tạm dừng quá trình sắp xếp.

+ Thời gian thực hiện: Hiển thị tổng thời gian thực hiện để hoàn thành sắp xếp.

**- Các chức năng phụ:**

+ Thay đổi tốc độ của quá trình sắp xếp.

+ Cập nhật thông tin về quá trình sắp xếp.

+ Cho phép thuật toán chạy theo hai hướng: tăng và giảm.

* + 1. Các kỹ thuật sử dụng
       1. Xử lý bất đồng bộ với Thread và Task

Để mô phỏng di chuyển các phần tử trong quá trình sắp xếp, chúng ta sử dụng kỹ thuật lập trình bất đồng bộ. Đối với việc này, có thể sử dụng Thread hoặc Task, hoặc kết hợp cả hai. Để đảm bảo sự mượt mà của di chuyển, chúng ta sử dụng hàm Thread.Sleep(<thời gian>) để tạm dừng tiến trình trong khoảng thời gian nhất định, giúp người dùng nhận thức được thay đổi vị trí của các phần tử.

Để chọn từng dòng code trong khung Code C/C++, chúng ta cần tạo một tiến trình chạy song song với tiến trình chính để tránh việc tiến trình chính bị chặn (block) khi thực hiện.

Đối với việc có hai chuyển động (hai hàm di chuyển phần tử) cùng chạy, chúng ta tạo hai tiến trình chạy song song với tiến trình chính.

Kết hợp sử dụng Thread và Task để thực hiện quá trình sắp xếp, chúng ta tạo một Thread để chọn từng dòng code và bên trong Thread đó, tạo hai Task con để di chuyển các phần tử. Điều này giúp thông báo việc hoàn tất di chuyển và cũng cho phép hủy bỏ quá trình bất cứ khi nào cần.

Trong xử lý đa tiến trình, có vấn đề quan trọng là không cho phép một tiến trình thay đổi hoặc chỉnh sửa các control trong tiến trình khác. Để giải quyết vấn đề này, chúng ta có thể sử dụng ManualResetEvent để tạm dừng và tiếp tục các tiến trình khi cần thiết.

* + - 1. Xử lý việc thay đổi tốc độ sắp xếp

Vì ở mỗi vòng lặp, khi điều chỉnh tọa độ của phần tử, chúng tôi liên tục sử dụng giá trị tốc độ hiện tại từ tham số. Khi giá trị tốc độ hiện tại thay đổi, điều này tác động trực tiếp lên tốc độ di chuyển của các phần tử trong mỗi vòng lặp tiếp theo. Điều này gây ra sự biến đổi trong tốc độ sắp xếp tổng thể của thuật toán.

* + - 1. Xử lý việc di chuyển các phần tử và các biến

Để xác định tọa độ vị trí và chuyển động của các phần tử và biến trong một giao diện đồ họa, chúng ta cần bắt đầu bằng việc xác định tọa độ ban đầu dựa trên kích thước (chiều dài và chiều rộng) của bảng hoặc panel. Từ tọa độ ban đầu này, chúng ta sử dụng thông tin về vị trí của các phần tử trong panel để tính toán tọa độ nơi mỗi phần tử sẽ được đặt trên bảng. Chúng ta cũng sử dụng phương pháp tương tự để xác định tọa độ đích khi cần di chuyển phần tử.

Đối với các biến có trong thuật toán sắp xếp, quy trình trên được thực hiện tương tự như với các phần tử. Điều này giúp định vị và theo dõi vị trí của các biến trong quá trình thực hiện thuật toán.

Để tăng cường trải nghiệm người dùng, các phần tử cũng có thể được tô màu để làm cho chúng dễ dàng phân biệt và nhận biết. Điều này giúp tạo ra một giao diện trực quan và thân thiện hơn, giúp người dùng theo dõi và hiểu rõ hơn về quá trình sắp xếp hoặc thực hiện thuật toán.

* + - 1. Xử lý việc tính thời gian thực hiện quá trình sắp xếp

Để đo lường thời gian thực hiện của quá trình sắp xếp, chúng ta có thể tăng giá trị thời gian dựa trên thời điểm thực hiện, đo lường thời gian bắt đầu và kết thúc quá trình sắp xếp. Khi quá trình sắp xếp hoàn thành, chênh lệch giữa thời điểm kết thúc và thời điểm bắt đầu sẽ là thời gian cần thiết để thuật toán thực hiện sắp xếp.

* + - 1. Xử lý debug

Để xử lý gỡ lỗi bằng cách thực hiện từng dòng code một cách thủ công, bạn có thể sử dụng ManualResetEvent để đặt một khoảng thời gian chờ vô hạn cho đến khi nó được kích hoạt (signaled).

* 1. Các thuật toán sắp xếp đơn giản
     1. Thuật toán Interchange Sort (Đổi chỗ trực tiếp)
        1. Nghịch thế

Một cặp giá trị (a, b) được gọi là nghịch thế khi a và b không thỏa điều kiện sắp thứ tự

**Ví dụ:** Cho mảng a gồm 6 phần tử: **14 29 -1 10 5 23**

**Yêu cầu:** Liệt kê các **cặp giá trị nghịch thế** trong mảng sau, biết rằng yêu cầu là sắp xếp mảng **tăng dần**

**Kết quả:**

**Duyệt lần 1: (14, -1) – (14, 10) – (14, 5)**

**Duyệt lần 2: (29, -1) – (29, 10) – (29, 5) – (29, 23)**

**Duyệt lần 3: Không có cặp nghịch thế nào**

**Duyệt lần 4 : (10, 5)**

**Duyệt lần 5 và 6: Không có cặp nghịch thế nào**

**Ý tưởng giải thuật**

Giải thuật sắp xếp bằng phương pháp Interchange Sort (Đổi chổ trực tiếp) được thực hiện bằng cách duyệt qua tất cả các cặp giá trị trong mảng và nếu gặp **cặp nghịch thế** sẽ thực hiện việc **hoán vị** (đổi chỗ) cho nhau.

* + - 1. Cài đặt mã giả
* **Bước 1**: i = 0; //bắt đầu từ đầu dãy
* **Bước 2**: j = i + 1; //tìm các nghịch thế với a[i]
* **Bước 3**:

Trong khi j < N thực hiện

Nếu a[j] < a[i]//Xét cặp a[i], a[j]

Swap(a[i], a[j]);

j = j + 1;

* **Bước 4**: i = i + 1;

Nếu i < N – 1: Lặp lại Bước 2.

Ngược lại: Dừng

* + - 1. Đánh giá độ phức tạp

|  |  |
| --- | --- |
| Trường hợp | Độ phức tạp |
| Tốt nhất | O(n) |
| Xấu nhất | O(n2) |

* + - * 1. Bảng đánh giá độ phức tạp của Interchange Sort.
    1. Thuật toán Selection Sort (Sắp xếp chọn trực tiếp)
       1. Ý tưởng giải thuật

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 1 phần tử.

Do 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. Cài đặt mã giả
* **Bước 1:** i = 0 ;
* **Bước 2**: Tìm phần tử a[min] nhỏ nhất trong dãy hiện hành từ a[i] đến a[n]
* **Bước 3**: Đổi chỗ a[i] và a[min]
* **Bước 4**:

Nếu i < n – 1 thì

i = i + 1;

Lặp lại bước 2;

Ngược lại: Dừng

* + - 1. Đánh giá độ phức tạp

|  |  |
| --- | --- |
| Trường hợp | Độ phức tạp |
| Tốt nhất | O(n2) |
| Xấu nhất | O(n2) |

* + - * 1. Bảng đánh giá độ phức tạp của Selection Sort.
    1. Thuật toán Bubble Sort (Sắp xếp nổi bọt)
       1. Ý tưởng thuật toán

Ý tưởng chính của thuật toán là xuất phát từ cuối (hoặc đầu) dãy, đổi chổ các cặp phần tử kế cận để đưa phần tử nhỏ (lớn) hơn trong cặp phần tử đó về vị trí đúng đầu (cuối) dãy hiện hành, sau đó sẽ không xét đến nó ở vị trí tiếp theo, do vậy ở lần xử lý thứ i sẽ có vị trí đầu dãy là i. Lặp lại xử lý trên cho đến khi không còn cặp phần tử nào để xét.

* + - 1. Cài đặt mã giả
* **Bước 1**: i=0; //lần xử lý đầu tiên
* **Bước 2**: j = n – 1; //duyệt từ cuối dãy ngược về vị trí i

Trong khi (j > i) thực hiện:

Nếu a[j] < a[j - 1]

HoanVi(a[j], a[j - 1]);

j = j – 1;

* **Bước 3**: i=i+1; // lần xử lý kế tiếp

Nếu i = n: hết dãy .Dừng

Ngược lại : Lặp lại Bước 2

* + - 1. Đánh giá độ phức tạp

|  |  |
| --- | --- |
| Trường hợp | Độ phức tạp |
| Tốt nhất | O(n) |
| Xấu nhất | O(n2) |

* + - * 1. Bảng đánh giá độ phức tạp Bubble Sort.
    1. Thuật toán Insertion Sort (Sắp xếp chèn trực tiếp)
       1. Ý tưởng thuật toán

Giả sử có một mảng **a0, a1, …., an-1** trong đó i phần tử đầu tiên **a0, a1 …,ai-1** đã có thứ tự.

Tìm cách chèn phần tử ai vào **vị trí thích hợp** của đoạn đã được sắp để có mảng mới a0, a1, …, ai  trở nên có thứ tự. Vị trí này chính là vị trí giữa hai phần tử ak-1 và ak thỏa ak-1 < ai < ak ( 1 )

* + - 1. Cài đặt mã giả

**Bước 1**: i = 1; // giả sử có đoạn a[1] đã được sắp

**Bước 2**: x = a[i]; Lưu giá trị a[i] và tìm vị trí pos thích hợp trong đoạn a[1] đến a[i- 1] để chèn a[i] vào

**Bước 3**: Dời chỗ các phần tử từ a[pos] đến a[i - 1] sang phải 1 vị trí để dành chổ cho a[i]

**Bước 4**: a[pos] = x; //có đoạn a[1]..a[i] đã được sắp

**Bước 5**: i = i + 1;

Nếu i < n; Lặp lại Bước 2

Ngược lại: dừng lại

* + - 1. Đánh giá độ phức tạp

|  |  |
| --- | --- |
| Trường hợp | Độ phức tạp |
| Tốt nhất | O(n) |
| Xấu nhất | O(n2) |

* + - * 1. Bảng đánh giá độ phức tạp Insertion Sort.
  1. Các thuật toán sắp xếp sử dụng đệ qui
     1. Thuật toán Quick Sort (Sắp xếp nhanh)
        1. Ý tưởng thuật toán

Giải thuật Quick sort sắp xếp các dãy a[0], a[1],…., a[n] dựa trên việc phân hoạch dãy ban đầu thành 3 đoạn:

* x **Đoạn 1**: gồm các phần tử có giá trị bé hơn x
* **Đoạn 2**: gồm các phần tử có giá trị bằng x
* **Đoạn 3**: gồm các phần tử có giá trị lớn hơn

x: giá trị của phần tử chính giữa dãy ban đầu.

Do vậy, đoạn thứ 2 đã có sắp xếp.

Để sắp xếp đoạn 1 và đoạn 3, ta lần lượt tiến hành việc phân hoạch từng dãy con theo cùng phương pháp phân hoạch ban đầu vừa trình bày.

* + - 1. Cài đặt mã giả
* **Bước 1**: Nếu left >= right // dãy có ít hơn 2 phần tử

=> Kết thúc

* **Bước 2**:

x = a(right – left) /2

Phân hoạch dãy ban đầu aleft…. aright thành các đoạn: aleft….aj, aj+1….ai-1, ai….. aright

* **Bước 3**: Sắp xếp đoạn 1: aleft….aj
* **Bước 4:** Sắp xếp đoạn 3: ai…..aright
  + - 1. Đánh giá độ phức tạp

|  |  |
| --- | --- |
| Trường hợp | Độ phức tạp |
| Tốt nhất | O(nlogn) |
| Xấu nhất | O(n2) |

* + - * 1. Bảng đánh giá độ phức tạp Quick Sort.
    1. Thuật toán Merge Sort (Sắp xếp trộn)
       1. Ý tưởng thuật toán

Giống như Quick sort, Merge sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành 2 nửa. Tiếp tục lặp lại việc này ở các nửa mảng đã chia. Sau cùng gộp các nửa đó thành mảng đã sắp xếp. Hàm merge() được sử dụng để gộp hai nửa mảng. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất sẽ gộp hai nửa mảng thành 1 mảng sắp xếp, các nửa mảng là arr[l…m] và arr[m+1…r] sau khi gộp sẽ thành một mảng duy nhất đã sắp xếp.

* + - 1. Cài đặt mã giả
* **Bước 1**: // chuẩn bị:

k = 1; //k là chiều dài của dãy con trong bước hiện hành

* **Bước 2** :

Tách dãy Array1, Array2, ., Arraynthành 2 dãy b và c theo nguyên tắc luân phiên từng nhóm k phần tử:

b = Array1, ., Arrayk, Array2k+1, ., Array3k, .

c = Arrayk+1, ., Array2k, Array3k+1, ., Array4k, .

* **Bước 3** :

Trộn từng cặp dãy con gồm k phần tử của 2 dãy temp\_1, temp\_2 vào a.

* **Bước 4 :**

k = k\*2;

Nếu k < n thì trở lại bước 2.

Ngược lại: Dừng

* + - 1. Đánh giá độ phức tạp

|  |  |
| --- | --- |
| Trường hợp | Độ phức tạp |
| Tốt nhất | O(nlogn) |
| Xấu nhất | O(n2) |

* + - * 1. Bảng đánh giá độ phức tạp Merge Sort.
  1. Giao diện phần mền
     1. Khởi tạo dãy số

Người dùng có thể khởi tạo input đầu vào bằng ba cách: Ngẫu nhiên, nhập tay , nhập số phần tử trong textbox và bấm vẽ nút

A screenshot of a phone

Description automatically generated

1. Bảng khởi tạo dãy số.

A screenshot of a computer

Description automatically generated

1. Khởi tạo dãy số bằng nhập tay.
   * 1. Bảng điều khiển

Người dùng có thể bắt đầu, dừng lại, tiếp tục, hủy quá trình sắp xếp tại đây. Và có thể điều chỉnh tốc độ của quá trình. Và cuối cùng hiện thời gian sắp xếp.

A screenshot of a computer

Description automatically generated

1. Bảng điều khiển.
   * 1. Bảng lựa chọn thuật toán

Người dùng có thể lựa chọn thuật toán để sắp xếp

A screenshot of a computer

Description automatically generated

1. Bảng Bảng lựa chọn thuật toán.
   * 1. Hiện ý tưởng thuật toán, dãy đang sắp xếp và hiện code C++

Nơi thể hiện ý tưởng của các thuật toán, và dãy đang sắp xếp sẽ hiện ở đó, hiện code C++ tương ứng với thuật toán đã chọn và sẽ thể hiện dòng code đang thực hiện.

A grey background with white text

Description automatically generated

1. Ý tưởng thuật toán, dãy đang sắp xếp.

A screenshot of a computer program

Description automatically generated

1. Hiện code C++.
   * 1. Thuật toán Interchange Sort

Người dùng chọn thuật toán Interchange Sort, và có thể chọn kiểu sắp xếp tăng dần hoặc giảm dần, và trong quá trình thực hiện người dùng có thể điều chỉnh tốc độ sắp xếp qua thanh điều khiển và có thể chọn debug chương trình sẽ chạy từng bước của thuật toán để người dùng có thể dễ dàng nắm bắt quá trình. A screenshot of a computer

Description automatically generated

1. Interchange Sort trước khi thực hiện sắp xếp.

Đây là mảng sau khi đã sắp xếp xong chương trình, nó sẽ xuất hiện một thông báo mảng đã sắp xếp xong, và hiển thị kết quả sắp xếp thuật toán Interchange Sort. A screenshot of a computer

Description automatically generated

1. Interchange Sort sau khi thực hiện sắp xếp.
   * 1. Thuật toán Selection Sort

Người dùng chọn thuật toán Selection Sort, và có thể chọn kiểu sắp xếp tăng dần hoặc giảm dần, và trong quá trình thực hiện người dùng có thể điều chỉnh tốc độ sắp xếp qua thanh điều khiển và có thể chọn debug chương trình sẽ chạy từng bước của thuật toán để người dùng có thể dễ dàng nắm bắt quá trình. A screenshot of a computer

Description automatically generated

1. Selection Sort trước khi thực hiện sắp xếp.

Đây là mảng sau khi đã sắp xếp xong chương trình, nó sẽ xuất hiện một thông báo mảng đã sắp xếp xong, và hiển thị kết quả sắp xếp thuật toán Selection Sort. A screenshot of a computer

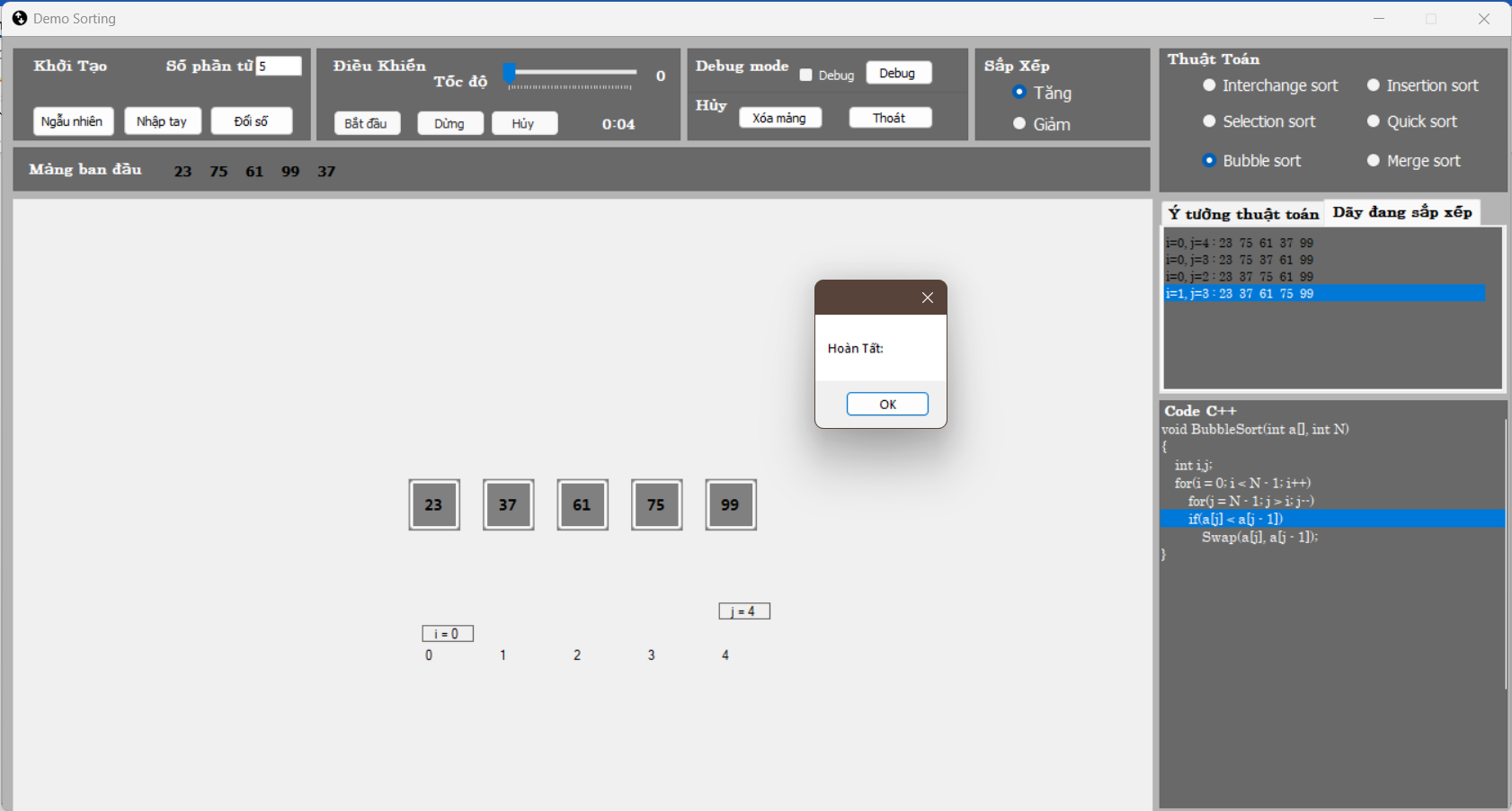
Description automatically generated

1. Selection Sort sau khi thực hiện sắp xếp.
   * 1. Thuật toán Bubble Sort

Người dùng chọn thuật toán Bubble Sort, và có thể chọn kiểu sắp xếp tăng dần hoặc giảm dần, và trong quá trình thực hiện người dùng có thể điều chỉnh tốc độ sắp xếp qua thanh điều khiển và có thể chọn debug chương trình sẽ chạy từng bước của thuật toán để người dùng có thể dễ dàng nắm bắt quá trình.A screenshot of a computer

Description automatically generated

1. Bubble Sort trước khi thực hiện sắp xếp.

Đây là mảng sau khi đã sắp xếp xong chương trình, nó sẽ xuất hiện một thông báo mảng đã sắp xếp xong, và hiển thị kết quả sắp xếp thuật toán Bubble Sort. 

1. Bubble Sort sau khi thực hiện sắp xếp.
   * 1. Thuật toán Insertion Sort

Người dùng chọn thuật toán Insertion Sort, và có thể chọn kiểu sắp xếp tăng dần hoặc giảm dần, và trong quá trình thực hiện người dùng có thể điều chỉnh tốc độ sắp xếp qua thanh điều khiển và có thể chọn debug chương trình sẽ chạy từng bước của thuật toán để người dùng có thể dễ dàng nắm bắt quá trình.A screenshot of a computer

Description automatically generated

1. Insertion Sort trước khi thực hiện sắp xếp.

Đây là mảng sau khi đã sắp xếp xong chương trình, nó sẽ xuất hiện một thông báo mảng đã sắp xếp xong, và hiển thị kết quả sắp xếp thuật toán Insertion Sort. A screenshot of a computer

Description automatically generated

1. Insertion Sort sau khi thực hiện sắp xếp.
   * 1. Thuật toán Quick Sort

Người dùng chọn thuật toán Quick Sort, và có thể chọn kiểu sắp xếp tăng dần hoặc giảm dần, và trong quá trình thực hiện người dùng có thể điều chỉnh tốc độ sắp xếp qua thanh điều khiển và có thể chọn debug chương trình sẽ chạy từng bước của thuật toán để người dùng có thể dễ dàng nắm bắt quá trình.A screenshot of a computer

Description automatically generated

1. Quick Sort trước khi thực hiện sắp xếp.

Đây là mảng sau khi đã sắp xếp xong chương trình, nó sẽ xuất hiện một thông báo mảng đã sắp xếp xong, và hiển thị kết quả sắp xếp thuật toán Quick Sort.

A screenshot of a computer

Description automatically generated

1. Quick Sort sau khi thực hiện sắp xếp.
   * 1. Thuật toán Merge Sort

Người dùng chọn thuật toán MergeSort, và có thể chọn kiểu sắp xếp tăng dần hoặc giảm dần, và trong quá trình thực hiện người dùng có thể điều chỉnh tốc độ sắp xếp qua thanh điều khiển và có thể chọn debug chương trình sẽ chạy từng bước của thuật toán để người dùng có thể dễ dàng nắm bắt quá trình.A screenshot of a computer

Description automatically generated

1. Merge Sort trước khi thực hiện sắp xếp.

Đây là mảng sau khi đã sắp xếp xong chương trình, nó sẽ xuất hiện một thông báo mảng đã sắp xếp xong, và hiển thị kết quả sắp xếp thuật toán MergeSort.

A screenshot of a computer

Description automatically generated

1. Merge Sort sau khi thực hiện sắp xếp.
2. Kết luận
   1. Kết quả đạt được

* Hoàn thành chương trình mô phỏng thuật toán sắp xếp như mục tiêu ban đầu có thể hiện node giữa màn hình sửa node trực tiếp trên màn hình chờ, mô phỏng thành công các thuật toán sắp xếp cùng khung ý tưởng và các tham số hiện thị trên màng hình chờ.
* Bước đầu nắm bắt được những kỹ thuật lập trình của C#, winform như timer, multithread, backgoundworker, đa ngôn ngữ sử dụng resources, tự tạo 1 user controls.
  1. Hạn chế
* Giao diện chương trình chưa được đẹp mắt và thân thiện với người dùng.
* Xử lý code còn rườm rà.
* Số lượng phần tử hiển thị hạn chế.
* Gặp một số lỗi khó khắc phục trong quá trình xử lý đa luồng multithread.
* Cung cấp ít thuật toán.
  1. Hướng phát triển
* Cải thiện giao diện chương trình thân thiện hơn với người sử dụng.
* Thêm nhiều chức năng như so sánh độ phức tạp của thuật toán (có animation) từ hai thuật toán trở lên.
* Tối ưu hóa hiệu suất chương trình.
* Đồng bộ tốc độ xử lý của bộ đếm thời gian và thread.
* Thêm nhiều tùy chọn cho người dùng trong phần cài đặt.
* Thêm thuật toán cho người dùng lựa chọn.
* Thêm nhiều chức năng mới cho phần mềm.

# Tài liệu tham khảo

[1] Nguyễn Đức Thuần, *Slỉdes Bài giảng Cấu trúc dữ liệu và giải thuật,* đại học Nha Trang.

[2] Nguyễn Đình Hoàng Sơn, *Slỉdes Bài giảng Cấu trúc dữ liệu và giải thuật,* đại học Nha Trang.

[3] Phạm Thị Kim Ngoan (2023), *Slides Bài giảng Lập trình hướng đối tượng,* đại học Nha Trang.

[4] HowKTeam.

[5] Library .NET Framework API Reference.

Các trang stackoverflow.com và các forum khác.

# Phụ lục

*Hàm xử lý chọn thuật toán.*

        #region  Chọn thuật toán

        void ChonThuatToan()

        {

            if (interchangerdbtn.Checked == true)

            {

                ThuatToanSapXep = InterchangeSort;

            }

            if (bubblerdbtn.Checked == true)

            {

                ThuatToanSapXep = ThuatToanSapXep = BubbleSort;

            }

            if (selectionrdbtn.Checked == true)

            {

                ThuatToanSapXep = SelectionSort;

            }

            if (insertionrdbtn.Checked == true)

            {

                ThuatToanSapXep = InsertionSort;

            }

            if (quickrdbtn.Checked == true)

            {

                ThuatToanSapXep = QuickSort;

            }

            if (mergerdbtn.Checked == true)

            {

                ThuatToanSapXep = MergeSort;

            }

        }

        #endregion

*Hàm khởi tạo Node.*

        #region Khởi tạo Node

        /// <summary>

        /// Khi sự kiện text đã thay đổi kiểm tra xem slnodetbx có là rỗng hay không nếu rổng rán lại số lượng node=0 vì trường hợp người dùng

        /// đã nhập 1 giá trị ví dụ số 12 mà muốn đổi thành số 20, nếu không có điều kiện sẽ bị lỗi không thể chuyển ra kiểu int được

        /// </summary>

        /// <param name="sender"></param>

        /// <param name="e"></param>

        private void textBox1\_TextChanged\_1(object sender, EventArgs e)

        {

            if (soluongNodetbx.Text == "")

            {

                SoLuongNode = 0;

            }

            else

            {

                SoLuongNode = Int32.Parse(soluongNodetbx.Text);

            }

        }

        /// <summary>

        /// Bắt sự kiện keyPress chỉ cho phép nhập số vào tbx

        /// Char.IsDigit(e.KeyChar) kiểm tra xem phím vừa nhập vào textbox có phải là ký tự số hay không, hàm này trả về kiểu bool

        /// Char.IsContro(e.KeyChar) –> kiểm tra xem phím vừa nhập vào textbox có phải là các ký tự điều khiển(nếu k có nó sẽ bắt nút enter,delete... như là không phải là số)

        /// </summary>

        /// <param name="sender"></param>

        /// <param name="e"></param>

        private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

        {

            if (!Char.IsDigit(e.KeyChar))

            {

                e.Handled = true;// dừng tất cả các tiến trình có liên quan đến thằng e

            }

        }

        private void taongaunhienbtn\_Click(object sender, EventArgs e)

        {

            SoLuongNode = rank.Next(2, ThamSo.SoluongNodeLonNhat);

            VeNut();

            soluongNodetbx.Text = SoLuongNode.ToString();

        }

        private void HienThiPanel\_Paint(object sender, PaintEventArgs e)

        {

        }

        private void venutbtn\_Click(object sender, EventArgs e)

        {

            if (SoLuongNode < 2 || SoLuongNode > ThamSo.SoluongNodeLonNhat)

            {

                MessageBox.Show(strSoLuongNode + ThamSo.SoluongNodeLonNhat);

                soluongNodetbx.Focus();

                return;

            }

            VeNut();

            Mangchuasapxep();

        }

        /// <summary>

        /// Khởi tạo 2 list 1 list int lưu giá trị dãy 1 list btn hiện thị trên GUI

        /// </summary>

        void VeNut()

        {

            DanhSachThamSo = new List<int>(SoLuongNode);

            DanhSachNode = new List<Node>(SoLuongNode);

            danhSachLabel = new List<Label>(SoLuongNode);

            sapxepPanel.Controls.Clear(); // Xóa những btn cũ trên panel ở phiên làm việc

            int temp = -SoLuongNode / 2;

            int temp2 = ThamSo.KhoangCachCacNode / 2;

            int temp3 = temp + 1;

            Point newPoint;

            for (int i = 0; i < SoLuongNode; i++)

            {

                Label tam = new Label() { Text = i.ToString() };

                tam.Size = new Size(ThamSo.KichCoNode/2, ThamSo.KichCoNode/2);

                tam.ForeColor = Color.Black;

                //tam.Enabled = false;

                Node btn = new Node(i);

                int Value = rank.Next(2, 100);

                btn.Value = Value;

                btn.Text = Value.ToString();

                btn.BackColor = ThamSo.mauNen;

                if (SoLuongNode % 2 != 0)

                {

                    newPoint = new Point(sapxepPanel.Width/2-btn.Width/2,

                                         sapxepPanel.Height/2-btn.Height/2);

                    btn.Location = new Point(newPoint.X+temp\* btn.Width + temp\* ThamSo.KhoangCachCacNode, newPoint.Y);

                    temp++;

                    tam.Location = new Point(btn.Location.X+btn.Width/2 - tam.Width / 2, btn.Location.Y + 150);

                }

                else

                {

                    if (i == SoLuongNode / 2)

                    {

                        temp2 = -temp2;

                        temp3 = 0;

                    }

                    newPoint = new Point(sapxepPanel.Width / 2 - btn.Width/2+ ThamSo.KichCoNode / 2, sapxepPanel.Height / 2 - btn.Height / 2);

                    btn.Location = new Point(newPoint.X - temp2 + temp3 \* ThamSo.KhoangCachCacNode + temp \* btn.Width, newPoint.Y);// Node Xuất hiện giữa panel

                    tam.Location = new Point(btn.Location.X + btn.Width / 2-tam.Width/2, btn.Location.Y + 150);

                    temp++;

                    temp3++;

                }

                sapxepPanel.Controls.Add(tam);

                sapxepPanel.Controls.Add(btn);

                DanhSachNode.Add(btn);

                DanhSachThamSo.Add(Value);

                danhSachLabel.Add(tam);

                btn.Capnhat += Btn\_Capnhat;

            }

            if (isRunning == false)

            {

                MangChuaSapXep = new List<int>(SoLuongNode);

                for (int i = 0; i < SoLuongNode; i++)

                {

                    MangChuaSapXep.Add(DanhSachThamSo[i]);

                }

            }

            Mangchuasapxep();

            batdaubtn.Enabled = true;

        }

        private void Mangchuasapxep()

        {

            chuasapxepPanel.Controls.Clear();

            lbMangChuaSapXep.Location = new Point(chuasapxepPanel.Location.X,10);

            chuasapxepPanel.Controls.Add(lbMangChuaSapXep);

            string \_string = "";

            foreach (var item in DanhSachThamSo)

            {

                \_string += item + "    ";

            }

            Label lbtext = new Label() { Text = \_string };

            lbtext.ForeColor = Color.Black;

            lbtext.Size = new Size(600, 20);

            lbtext.Location = new Point(lbMangChuaSapXep.Location.X+lbMangChuaSapXep.Width+10, 12);

            chuasapxepPanel.Controls.Add(lbtext);

        }

        #endregion

*Hàm hiện mảng đang sắp xếp.*

        #region  Hiển thị mảng đang sắp xếp

        private void Hienthimangdangsapxep(int k, int j, string thamso1, string thamso2)

        {

            string temp = "\n" + thamso1 + "=" + k + ", "+thamso2+"="+j+" : ";

            for (int i = 0; i < SoLuongNode; i++)

            {

                int t = DanhSachThamSo[i];

                temp += t.ToString() + "  ";

            }

            daydangxepListbox.Items.Add(temp);

            daydangxepListbox.ForeColor = Color.Black;

            daydangxepListbox.SelectedIndex = daydangxepListbox.Items.Count-1;

        }

        #endregion

*Hàm thêm ý tưởng vào yTuongTextBox.*

        #region Thêm ý tưởng vào yTuongTextBox

        private void addYTuong()

        {

            this.yTuongTextBox.Controls.Clear();

            this.yTuongTextBox.Text = HienThiThuatToan.yTuong;

        }

        #endregion

*Hàm hiển thị code c ++ và ý tưởng thuật toán*

        #region Hiển thị code c ++ và ý tưởng thuật toán

        private void hienThiThuatToanNCodeC()

        {

            addYTuong();

            //Gán yTuongTextBox, codeListBox và isSapXepTang

            HienThiThuatToan.yTuongThuatToan = yTuongTextBox;

            HienThiThuatToan.codeListBox = codeListBox;

            bool tang = tangrdbtn.Checked;

            //chọn sort

            if (selectionrdbtn.Checked)

            {

                HienThiThuatToan.SelectionSort(tang);

            }

            else if (insertionrdbtn.Checked)

            {

                HienThiThuatToan.InsertionSort(tang);

            }

            else if (quickrdbtn.Checked)

            {

                HienThiThuatToan.QuickSort(tang);

            }

            else if (interchangerdbtn.Checked)

            {

                HienThiThuatToan.InterchangeSort(tang);

            }

            else if (bubblerdbtn.Checked)

            {

                HienThiThuatToan.BubbleSort(tang);

            }

            else if (mergerdbtn.Checked)

            {

                HienThiThuatToan.MergeSort(tang);

            }

        }

        #endregion

*Hàm đếm thời gian thực hiện.*

        #region Đếm thời gian thực hiện

        private void timer1\_Tick\_1(object sender, EventArgs e)

        {

            DemThoiGian();

        }

        private void DemThoiGian()

        {

            Giay++;

            if (Giay > 59)

            {

                Giay = 0;

                Phut++;

            }

            if (Giay<10)

            {

                label11.Text = Phut + ":0" + Giay;

            }

            else

            {

                label11.Text = Phut + ":" + Giay;

            }

        }

        private void Reset\_CountTime()

        {

            Phut = Giay = 0;

            label11.Text = "00:00";

        }

        #endregion

*Hàm nút bắt đầu.*

        #region  Nút bắt đầu

        void batdaubtn\_Click(object sender, EventArgs e)

        {

            if (sapxepThread != null)

            {

                sapxepThread.Abort();

            }

            daydangxepListbox.Visible = true;

            huybnt.Enabled = true;

            isRunning = true;

            DieuChinhControls(isRunning);

            Reset\_CountTime();

            timer1.Start();

            ChonThuatToan();

            sapxepThread = new Thread(new ThreadStart(ThuatToanSapXep));

            sapxepThread.Start();

            //backgroundWorker1.RunWorkerAsync(); // goi ham do work

        }

        #endregion

*Hàm xử lý nút tạm dừng, tiếp tục, hủy.*

        #region Nút tạm dừng, tiếp tục, hủy

        // Tạm dừng

        public static bool CodeListBoxIsPause = false;

        private void dungbtn\_Click(object sender, EventArgs e)

        {

            if (Node.IsPause)

            {

                Node.pauseStatus.Set();     // hàm để tiếp

                Node.IsPause = false;

                dungbtn.Text = "Dừng";

                timer1.Start();

            }

            else

            {

                Node.pauseStatus.Reset();    // hàm để dừng

                Node.IsPause = true;

                dungbtn.Text = "Tiếp";

                timer1.Stop();

            }

        }

        // Hủy

        private void huybnt\_Click(object sender, EventArgs e)

        {

            sapxepThread.Abort();

            VeNut();

            isRunning = false;

            for (int i = 0; i < SoLuongNode; i++)

            {

                DanhSachNode[i].Text = MangChuaSapXep[i].ToString();

                DanhSachThamSo[i] = MangChuaSapXep[i];

            }

            DieuChinhControls(isRunning);

            Mangchuasapxep();

            Reset\_CountTime();

            timer1.Stop();

            huybnt.Enabled = false;

        }

        #endregion

*Hàm điều chỉnh tốc độ di chuyển của các node*

        #region  Điều chỉnh tốc độ

        private void trackBar1\_ValueChanged(object sender, EventArgs e)

        {

            ThamSo.ThoiGianDoi = trackBar1.Value;

            label1.Text = trackBar1.Value.ToString();

        }

        #endregion

*Hàm thông báo hoàn tất sắp xếp*

        #region  Thông báo hoàn tất sắp xếp

        private void hoanTatSapXep()

        {

            foreach (Node item in DanhSachNode)

            {

                item.BackColor = ThamSo.mauNodeHTSX;

            }

            isRunning = false;

            DieuChinhControls(isRunning);

            MessageBox.Show("Hoàn Tất: " + hoanTat);

            timer1.Stop();

            Reset\_CountTime();

            foreach (Label label in bienArr.Values)

            {

                label.Visible = false;

            }

            huybnt.Enabled = false;

        }

        #endregion

Hàm thoát và xóa mảng

        private void thoatbtn\_Click(object sender, EventArgs e)

        {

            if (sapxepThread!=null)

            {

                sapxepThread.Abort();

            }

            this.Close();

        }

        private void xoamangbtn\_Click(object sender, EventArgs e)

        {

            sapxepPanel.Controls.Clear();

            DanhSachNode.Clear();

            DanhSachThamSo.Clear();

            chuasapxepPanel.Controls.Clear();

            batdaubtn.Enabled = false;

        }

*Hàm nhập tay mảng sắp xếp*

        #region  Nhập tay mảng sắp xếp

        private void nhaptaybtn\_Click(object sender, EventArgs e)

        {

            frmInput input = new frmInput();

            input.ShowDialog();

            if (input.isNhap == true)

            {

                string temp = "";

                SoLuongNode = input.DayInput.Count();

                soluongNodetbx.Text = SoLuongNode.ToString();

                VeNut();

                for (int i = 0; i < SoLuongNode; i++)

                {

                    DanhSachNode[i].Text = input.DayInput[i].ToString();

                    DanhSachThamSo[i] = input.DayInput[i];

                }

                Mangchuasapxep();

            }

        }

        #endregion

*Hàm xử lý debug*

        #region Phần Debug

        public void XuLyDebug()

        {

            if (cboxdebug.Checked==true)

            {

                ThamSo.ThoiGianDoi = 1;

                dungbtn.Enabled = false;

                Node.pauseStatus.Set();

                Node.IsPause = false;

                isDebug = true;

                timer1.Stop();

                Reset\_CountTime();

            }

            else

            {

                HienThiThuatToan.tamdunglistbox.Set();

                Node.pauseStatus.Set();

                Node.IsPause = false;

                isDebug = false;

            }

        }

        private void cboxdebug\_CheckedChanged(object sender, EventArgs e)

        {

            XuLyDebug();

        }

        private void btndebug\_Click(object sender, EventArgs e)

        {

            if (isDebug == true)

            {

               // batdaubtn.Enabled = true;

                //  Nếu đang ở chế độ thường thì chuyển thành Debug

                if (!cboxdebug.Checked)

                    cboxdebug.Checked = true;

                // Chạy code

                //đặt sign kích hoạt manualresetevent => tạm dừng đợi trong khoảng thời gian vô tận

                HienThiThuatToan.tamdunglistbox.Set();

                CodeListBoxIsPause = false;

            }

        }

        #endregion

Hàm xử lý các thuật toán sắp xếp

*InterchangeSort*

        #region InterchangeSort

        private void InterchangeSort()

        {

            int i = 0, j = 0;

            bool flag;

            HienThiThuatToan.ChayCodeC(2);

            for (i = 0; i < SoLuongNode - 1; i++)

            {

                bienArr["i"].Location = new Point(danhSachLabel[i].Location.X, danhSachLabel[i].Location.Y - 20);

                bienArr["i"].Text = "i = " + i;

                sapxepPanel.Controls.Add(bienArr["i"]);

                bienArr["i"].Visible = true;

                HienThiThuatToan.ChayCodeC(3);

                for (j = i + 1; j < SoLuongNode; j++)

                {

                    DanhSachNode[j].BackColor = ThamSo.Nodeketiep;

                    bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 40);

                    bienArr["j"].Text = "j = " + j;

                    sapxepPanel.Controls.Add(bienArr["j"]);

                    bienArr["j"].Visible = true;

                    HienThiThuatToan.ChayCodeC(4);

                    flag = false;

                    if (tangrdbtn.Checked == true)

                    {

                        HienThiThuatToan.ChayCodeC(5);

                        if (DanhSachThamSo[j] < DanhSachThamSo[i]) flag = true;

                    }

                    if (giamrdbtn.Checked == true)

                    {

                        HienThiThuatToan.ChayCodeC(5);

                        if (DanhSachThamSo[j] > DanhSachThamSo[i]) flag = true;

                    }

                    if (flag)

                    {

                        HienThiThuatToan.ChayCodeC(6);

                        Thread.Sleep(ThamSo.ThoiGianDoi);

                        CapNhatThamSo(i, j);

                        DichuyenCacNode(j, i);

                        Hienthimangdangsapxep(i, j, "i", "j");

                    }

                    else

                    {

                        DanhSachNode[j].BackColor = ThamSo.mauNen;

                    }

                }

            }

            DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;

            DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;

            hoanTatSapXep()

        }

        #endregion

*SelectionSort*

        #region SelectionSort

        private void SelectionSort()

        {

            int min, i = 0, j = 0;

            bool flag = false;

            HienThiThuatToan.ChayCodeC(2);

            for (i = 0; i < SoLuongNode - 1; i++)

            {

                HienThiThuatToan.ChayCodeC(3);

                HienThiThuatToan.ChayCodeC(5);

                bienArr["i"].Location = new Point(danhSachLabel[i].Location.X, danhSachLabel[i].Location.Y - 20);

                bienArr["i"].Text = "i = " + i;

                sapxepPanel.Controls.Add(bienArr["i"]);

                bienArr["i"].Visible = true;

                for (j = i + 1; j < SoLuongNode; j++)

                {

                    DanhSachNode[j].BackColor = ThamSo.Nodeketiep;

                    HienThiThuatToan.ChayCodeC(6);

                    flag = false;

                    min = i;

                    bienArr["min"].Location = new Point(danhSachLabel[min].Location.X, danhSachLabel[min].Location.Y - ThamSo.KichCoNode - 180);

                    bienArr["min"].Text = "min = " + min;

                    bienArr["min"].SendToBack();

                    sapxepPanel.Controls.Add(bienArr["min"]);

                    bienArr["min"].Visible = true;

                    if (tangrdbtn.Checked == true)

                    {

                        HienThiThuatToan.ChayCodeC(7);

                        bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 40);

                        bienArr["j"].Text = "j = " + j;

                        sapxepPanel.Controls.Add(bienArr["j"]);

                        bienArr["j"].Visible = true;

                        if (DanhSachThamSo[j] < DanhSachThamSo[min])

                        {

                            min = j;

                            flag = true;

                            bienArr["min"].Location = new Point(danhSachLabel[min].Location.X, danhSachLabel[min].Location.Y - ThamSo.KichCoNode - 180);

                            bienArr["min"].Text = "min = " + min;

                            bienArr["min"].SendToBack();

                        }

                    }

                    if (giamrdbtn.Checked == true)

                    {

                        HienThiThuatToan.ChayCodeC(7);

                        bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 40);

                        bienArr["j"].Text = "j = " + j;

                        sapxepPanel.Controls.Add(bienArr["j"]);

                        if (DanhSachThamSo[j] > DanhSachThamSo[min])

                        {

                            min = j;

                            flag = true;

                            bienArr["min"].Location = new Point(danhSachLabel[min].Location.X, danhSachLabel[min].Location.Y - 180);

                            bienArr["min"].Text = "min = " + min;

                            bienArr["min"].SendToBack();

                        }

                    }

                    if (flag == true)

                    {

                        HienThiThuatToan.ChayCodeC(8);

                        HienThiThuatToan.ChayCodeC(9);

                        Thread.Sleep(ThamSo.ThoiGianDoi);

                        DanhSachNode[i].BackColor = DanhSachNode[min].BackColor = ThamSo.mauNodeDangSX;

                        CapNhatThamSo(min, i);

                        DichuyenCacNode(min, i);

                        Hienthimangdangsapxep(i, j, "i", "j");

                    }

                    else

                    {

                        DanhSachNode[j].BackColor = ThamSo.mauNen;

                    }

                }

            }

            DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;

            DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;

            hoanTatSapXep();

        }

        #endregion

*BubbleSort*

        #region BubbleSort

        private void BubbleSort()

        {

            bool flag = false;

            HienThiThuatToan.ChayCodeC(2);

            HienThiThuatToan.ChayCodeC(3);

            int i = 0, j = 0;

            bienArr["i"].Location = new Point(danhSachLabel[i].Location.X, danhSachLabel[i].Location.Y - 20);

            bienArr["i"].Text = "i = " + i;

            sapxepPanel.Controls.Add(bienArr["i"]);

            bienArr["i"].Visible = true;

            for (i = 0; i < SoLuongNode - 1; i++)

            {

                HienThiThuatToan.ChayCodeC(4);

                j = SoLuongNode - 1;

                bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 40);

                bienArr["j"].Text = "j = " + j;

                sapxepPanel.Controls.Add(bienArr["j"]);

                bienArr["j"].Visible = true;

                for (j = SoLuongNode - 1; j > i; j--)

                {

                    flag = false;

                    if (tangrdbtn.Checked == true)

                    {

                        HienThiThuatToan.ChayCodeC(5);

                        if (DanhSachThamSo[j] < DanhSachThamSo[j - 1]) flag = true;

                    }

                    if (giamrdbtn.Checked == true)

                    {

                        HienThiThuatToan.ChayCodeC(5);

                        if (DanhSachThamSo[j] > DanhSachThamSo[j - 1]) flag = true;

                    }

                    if (flag == true)

                    {

                        HienThiThuatToan.ChayCodeC(6);

                        Thread.Sleep(ThamSo.ThoiGianDoi);

                        CapNhatThamSo(j, j - 1);

                        DichuyenCacNode(j, j - 1);

                        Hienthimangdangsapxep(i, j, "i", "j");

                    }

                }

            }

            DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;

            DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;

            hoanTatSapXep();

        }

        #endregion

*InsertionSort*

        #region InsertionSort

        private void InsertionSort()

        {

            int pos, key;

            Node Nodetam;

            DanhSachNode[0].BackColor = ThamSo.mauNodeDangSX;

            for (int i = 1; i < SoLuongNode; i++)

            {

                HienThiThuatToan.ChayCodeC(5);

                DanhSachNode[i].BackColor = ThamSo.mauNodeDangSX;

                key = DanhSachThamSo[i];

                HienThiThuatToan.ChayCodeC(7);

                Nodetam = DanhSachNode[i];

                pos = i - 1;

                bienArr["i"].Location = new Point(danhSachLabel[i].Location.X, danhSachLabel[i].Location.Y - 40);

                bienArr["i"].Text = "i = " + i;

                sapxepPanel.Controls.Add(bienArr["i"]);

                bienArr["i"].Visible = true;

                DanhSachNode[i].ChuyenLen();

                DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;

                HienThiThuatToan.ChayCodeC(8);

                bienArr["pos"].Location = new Point(danhSachLabel[pos].Location.X, danhSachLabel[pos].Location.Y - 240);

                bienArr["pos"].Text = "pos = " + pos;

                sapxepPanel.Controls.Add(bienArr["pos"]);

                bienArr["pos"].Visible = true;

                if (tangrdbtn.Checked == true)

                {

                    while ((pos >= 0) && (DanhSachThamSo[pos] > key))

                    {

                        DanhSachNode[pos].ChuyenNgang(pos + 1);

                        DanhSachNode[pos].vitriHienTai = pos + 1;

                        DanhSachNode[pos + 1] = DanhSachNode[pos];

                        DanhSachThamSo[pos + 1] = DanhSachThamSo[pos];

                        Hienthimangdangsapxep(pos, pos + 1, "pos", "i");

                        HienThiThuatToan.ChayCodeC(10);

                        pos--;

                    }

                }

                else

                {

                    while ((pos >= 0) && (DanhSachThamSo[pos] < key))

                    {

                        DanhSachNode[pos].ChuyenNgang(pos + 1);

                        DanhSachNode[pos].vitriHienTai = pos + 1;

                        DanhSachNode[pos + 1] = DanhSachNode[pos];

                        DanhSachThamSo[pos + 1] = DanhSachThamSo[pos];

                        Hienthimangdangsapxep(pos, pos + 1, "pos", "i");

                        HienThiThuatToan.ChayCodeC(10);

                        pos--;

                    }

                }

                Nodetam.ChuyenNgang(pos + 1);

                Nodetam.ChuyenXuong();

                DanhSachThamSo[pos + 1] = key;

                DanhSachNode[pos + 1] = Nodetam;

                Nodetam.vitriHienTai = pos + 1;

                DanhSachNode[pos + 1].BackColor = ThamSo.mauNodeHTSX;

            }

            DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;

            DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;

            hoanTatSapXep();

        }

        #endregion

*QuickSort*

        #region QuickSort

        public void QuickSort()

        {

            ThucHienQuickSort(0, DanhSachThamSo.Count - 1);

            DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;

            DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;

            hoanTatSapXep();

        }

        private void ThucHienQuickSort(int left, int right)

        {

            HienThiThuatToan.ChayCodeC(1);

            int i, j, x, vt\_x;

            bienArr["left"].Text = "left = " + left;

            bienArr["left"].Location = new Point(danhSachLabel[left].Location.X, danhSachLabel[left].Location.Y - 260);

            sapxepPanel.Controls.Add(bienArr["left"]);

            bienArr["left"].Visible = true;

            bienArr["left"].SendToBack();

            bienArr["right"].Text = "right = " + right;

            bienArr["right"].Location = new Point(danhSachLabel[right].Location.X, danhSachLabel[right].Location.Y - 240);

            sapxepPanel.Controls.Add(bienArr["right"]);

            bienArr["right"].Visible = true;

            bienArr["right"].SendToBack();

            x = DanhSachThamSo[(left + right) / 2];

            HienThiThuatToan.ChayCodeC(3);

            vt\_x = (left + right) / 2;

            i = left; j = right;

            do

            {

                int z\_vt\_x = vt\_x;

                if (tangrdbtn.Checked == true)

                {

                    HienThiThuatToan.ChayCodeC(7);

                    bienArr["vt\_x"].Location = new Point(danhSachLabel[vt\_x].Location.X, danhSachLabel[vt\_x].Location.Y - 40);

                    bienArr["vt\_x"].Text = "x = a[" + vt\_x + "]";

                    sapxepPanel.Controls.Add(bienArr["vt\_x"]);

                    bienArr["vt\_x"].Visible = true;

                    bienArr["vt\_x"].SendToBack();

                    bienArr["i"].Text = "i = " + i;

                    bienArr["i"].Location = new Point(danhSachLabel[i].Location.X, danhSachLabel[i].Location.Y - 20);

                    sapxepPanel.Controls.Add(bienArr["i"]);

                    bienArr["i"].Visible = true;

                    bienArr["i"].SendToBack();

                    bienArr["j"].Text = "j = " + j;

                    bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 60);

                    sapxepPanel.Controls.Add(bienArr["j"]);

                    bienArr["j"].Visible = true;

                    bienArr["j"].SendToBack();

                    while (DanhSachThamSo[i] < x)

                    {

                        HienThiThuatToan.ChayCodeC(8);

                        int f\_i = i;

                        i++;

                        bienArr["i"].Text = "i = " + i;

                        bienArr["i"].Location = new Point(danhSachLabel[i].Location.X, danhSachLabel[i].Location.Y - 20);

                        sapxepPanel.Controls.Add(bienArr["i"]);

                        bienArr["i"].Visible = true;

                        bienArr["i"].SendToBack();

                    }

                    HienThiThuatToan.ChayCodeC(9);

                    while (DanhSachThamSo[j] > x)

                    {

                        HienThiThuatToan.ChayCodeC(10);

                        int f\_j = j;

                        j--;

                        bienArr["j"].Text = "j = " + j;

                        bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 60);

                        sapxepPanel.Controls.Add(bienArr["j"]);

                        bienArr["j"].Visible = true;

                        bienArr["j"].SendToBack();

                    }

                }

                else

                {

                    HienThiThuatToan.ChayCodeC(7);

                    bienArr["vt\_x"].Location = new Point(danhSachLabel[vt\_x].Location.X, danhSachLabel[vt\_x].Location.Y - 40);

                    bienArr["vt\_x"].Text = "x = a[" + vt\_x + "]";

                    sapxepPanel.Controls.Add(bienArr["vt\_x"]);

                    bienArr["vt\_x"].Visible = true;

                    bienArr["vt\_x"].SendToBack();

                    bienArr["i"].Text = "i = " + i;

                    bienArr["i"].Location = new Point(danhSachLabel[i].Location.X, danhSachLabel[i].Location.Y - 20);

                    sapxepPanel.Controls.Add(bienArr["i"]);

                    bienArr["i"].Visible = true;

                    bienArr["i"].SendToBack();

                    bienArr["j"].Text = "j = " + j;

                    bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 60);

                    sapxepPanel.Controls.Add(bienArr["j"]);

                    bienArr["j"].Visible = true;

                    bienArr["j"].SendToBack();

                    while (DanhSachThamSo[i] > x)

                    {

                        HienThiThuatToan.ChayCodeC(8);

                        int f\_i = i;

                        i++;

                        bienArr["i"].Text = "i = " + i;

                        bienArr["i"].Location = new Point(danhSachLabel[i].Location.X, danhSachLabel[i].Location.Y - 20);

                        sapxepPanel.Controls.Add(bienArr["i"]);

                        bienArr["i"].Visible = true;

                        bienArr["i"].SendToBack();

                    }

                    HienThiThuatToan.ChayCodeC(9);

                    while (DanhSachThamSo[j] < x)

                    {

                        HienThiThuatToan.ChayCodeC(10);

                        int f\_j = j;

                        j--;

                        bienArr["j"].Text = "j = " + j;

                        bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 60);

                        sapxepPanel.Controls.Add(bienArr["j"]);

                        bienArr["j"].Visible = true;

                        bienArr["j"].SendToBack();

                    }

                }

                HienThiThuatToan.ChayCodeC(11);

                if (i <= j)

                {

                    int f\_vt\_x = vt\_x;

                    if (i == vt\_x)

                    {

                        vt\_x = j;

                    }

                    else if (j == vt\_x)

                    {

                        vt\_x = i;

                    }

                    HienThiThuatToan.ChayCodeC(13);

                    CapNhatThamSo(i, j);

                    DichuyenCacNode(j, i);

                    Hienthimangdangsapxep(i, j, "i", "j");

                    bienArr["vt\_x"].Location = new Point(danhSachLabel[vt\_x].Location.X, danhSachLabel[vt\_x].Location.Y - 40);

                    bienArr["vt\_x"].Text = "x = a[" + vt\_x + "]";

                    sapxepPanel.Controls.Add(bienArr["vt\_x"]);

                    bienArr["vt\_x"].Visible = true;

                    bienArr["vt\_x"].SendToBack();

                    i++; j--;

                }

                HienThiThuatToan.ChayCodeC(16);

            } while (i <= j);

            HienThiThuatToan.ChayCodeC(17);

            if (left < j)

            {

                HienThiThuatToan.ChayCodeC(18);

                ThucHienQuickSort(left, j);

            }

            HienThiThuatToan.ChayCodeC(19);

            if (i < right)

            {

                HienThiThuatToan.ChayCodeC(20);

                ThucHienQuickSort(i, right);

            }

        }

        #endregion

*MergeSoft*

        #region MergeSoft

        List<Node> b = new List<Node>();

        List<Node> c = new List<Node>();

        int nb, nc;

        int Min(int a, int b)

        {

            if (a > b) return b;

            else return a;

        }

        void Distribute(List<Node> a, int N, ref int nb, ref int nc, int k)

        {

            int i, pa, pb, pc;

            pa = pb = pc = 0;

            HienThiThuatToan.ChayCodeC(10);

            while (pa < N)

            {

                HienThiThuatToan.ChayCodeC(12);

                for (i = 0; (pa < N) && (i < k); i++, pa++, pb++)

                {

                    HienThiThuatToan.ChayCodeC(13);

                    a[pa].BackColor = ThamSo.mauNodeDangSX;

                    a[pa].ChuyenLen();

                    a[pa].ChuyenNgang(pb);

                    a[pa].vitriHienTai = pb;

                    b[pb] = a[pa];

                    HienThiThuatToan.ChayCodeC(12);

                }

                HienThiThuatToan.ChayCodeC(12);

                HienThiThuatToan.ChayCodeC(14);

                for (i = 0; (pa < N) && (i < k); i++, pa++, pc++)

                {

                    HienThiThuatToan.ChayCodeC(15);

                    a[pa].BackColor = ThamSo.mauNodeDangSX;

                    a[pa].ChuyenXuong();

                    a[pa].ChuyenNgang(pc);

                    a[pa].vitriHienTai = pc;

                    c[pc] = a[pa];

                    HienThiThuatToan.ChayCodeC(14);

                }

                HienThiThuatToan.ChayCodeC(14);

                HienThiThuatToan.ChayCodeC(10);

            }

            HienThiThuatToan.ChayCodeC(10);

            nb = pb; nc = pc;

        }

        void Merge(List<Node> a, int nb, int nc, int k)

        {

            int p, pb, pc, ib, ic, kb, kc;

            p = pb = pc = 0; ib = ic = 0;

            HienThiThuatToan.ChayCodeC(23);

            while ((nb > 0) && (nc > 0))

            {

                kb = Min(k, nb);

                kc = Min(k, nc);

                HienThiThuatToan.ChayCodeC(26);

                bool thucHien = false; // dùng để xét tăng/giảm , nếu bằng true thì code sẽ chạy

                if (tangrdbtn.Checked == true)

                {

                    if (c[pc + ic].Value >= b[pb + ib].Value)

                        thucHien = true;

                }

                else

                {

                    if (c[pc + ic].Value <= b[pb + ib].Value)

                        thucHien = true;

                }

                if (thucHien)

                {

                    HienThiThuatToan.ChayCodeC(28);

                    b[pb + ib].BackColor = ThamSo.mauNen;

                    b[pb + ib].ChuyenXuong();

                    b[pb + ib].ChuyenNgang(p);

                    b[pb + ib].vitriHienTai = p;

                    a[p] = b[pb + ib];

                    p = p + 1;

                    ib++;

                    HienThiThuatToan.ChayCodeC(29);

                    if (ib == kb)

                    {

                        HienThiThuatToan.ChayCodeC(31);

                        for (; ic < kc; ic++)

                        {

                            HienThiThuatToan.ChayCodeC(32);

                            c[pc + ic].BackColor = ThamSo.mauNen;

                            c[pc + ic].ChuyenLen();

                            c[pc + ic].ChuyenNgang(p);

                            c[pc + ic].vitriHienTai = p;

                            a[p] = c[pc + ic];

                            p = p + 1;

                            HienThiThuatToan.ChayCodeC(32);

                        }

                        pb += kb; pc += kc; ib = ic = 0;

                        nb -= kb; nc -= kc;

                    }

                }

                else

                {

                    HienThiThuatToan.ChayCodeC(39);

                    c[pc + ic].BackColor = ThamSo.mauNen;

                    c[pc + ic].ChuyenLen();

                    c[pc + ic].ChuyenNgang(p);

                    c[pc + ic].vitriHienTai = p;

                    a[p] = c[pc + ic];

                    p = p + 1;

                    ic++;

                    HienThiThuatToan.ChayCodeC(40);

                    if (ic == kc)

                    {

                        HienThiThuatToan.ChayCodeC(42);

                        for (; ib < kb; ib++)

                        {

                            HienThiThuatToan.ChayCodeC(43);

                            b[pb + ib].BackColor = ThamSo.mauNen;

                            b[pb + ib].ChuyenXuong();

                            b[pb + ib].ChuyenNgang(p);

                            b[pb + ib].vitriHienTai = p;

                            a[p] = b[pb + ib];

                            p = p + 1;

                            HienThiThuatToan.ChayCodeC(43);

                        }

                        HienThiThuatToan.ChayCodeC(42);

                        pb += kb; pc += kc; ib = ic = 0;

                        nb -= kb; nc -= kc;

                    }

                }

                HienThiThuatToan.ChayCodeC(23);

            }

            HienThiThuatToan.ChayCodeC(23);

            if (a.Count % 2 == 1 && (k != (a.Count - 1)))

            {

                if (nb > nc)

                {

                    HienThiThuatToan.ChayCodeC(43);

                    b[pb].BackColor = ThamSo.mauNen;

                    b[pb].ChuyenXuong();

                    b[pb].ChuyenNgang(a.Count - 1);

                    b[pb].vitriHienTai = a.Count - 1;

                }

            }

            if (a.Count % 2 == 0 && Math.Abs(nb - nc) == 2)

            {

                HienThiThuatToan.ChayCodeC(43);

                b[pb].BackColor = ThamSo.mauNen;

                b[pb].ChuyenXuong();

                b[pb].ChuyenNgang(a.Count - 2);

                b[pb].vitriHienTai = a.Count - 2;

                HienThiThuatToan.ChayCodeC(43);

                b[pb + 1].BackColor = ThamSo.mauNen;

                b[pb + 1].ChuyenXuong();

                b[pb + 1].ChuyenNgang(a.Count - 1);

                b[pb + 1].vitriHienTai = a.Count - 1;

            }

        }

        void ThucHienMergeSort(List<Node> a, int N)

        {

            for (int i = 0; i < SoLuongNode; i++)

            {

                b.Add(new Node(i));

                c.Add(new Node(i));

            }

            int k;

            HienThiThuatToan.ChayCodeC(53);

            for (k = 1; k < N; k \*= 2)

            {

                bienArr["k"].Location = new Point(danhSachLabel[k].Location.X, danhSachLabel[k].Location.Y - 260);

                bienArr["k"].Text = "k = " + k;

                sapxepPanel.Controls.Add(bienArr["k"]);

                bienArr["k"].Visible = true;

                HienThiThuatToan.ChayCodeC(55);

                Distribute(a, N, ref nb, ref nc, k);

                HienThiThuatToan.ChayCodeC(56);

                Merge(a, nb, nc, k);

                HienThiThuatToan.ChayCodeC(53);

            }

            HienThiThuatToan.ChayCodeC(53);

        }

        private void label4\_Click(object sender, EventArgs e)

        {

        }

        private void label11\_Click(object sender, EventArgs e)

        {

        }

        void MergeSort()

        {

            bienArr["a:"].Location = new Point(0, 200 + DanhSachNode[0].Height / 2);

            bienArr["a:"].Text = "a:";

            sapxepPanel.Controls.Add(bienArr["a:"]);

            bienArr["c:"].Location = new Point(0, 250 + DanhSachNode[0].Height / 2);

            bienArr["c:"].Text = "c:";

            sapxepPanel.Controls.Add(bienArr["c:"]);

            bienArr["b:"].Location = new Point(0, 150 + DanhSachNode[0].Height / 2);

            bienArr["b:"].Text = "b:";

            sapxepPanel.Controls.Add(bienArr["b:"]);

            bienArr["a:"].Visible = bienArr["b:"].Visible = bienArr["c:"].Visible = true;

            HienThiThuatToan.ChayCodeC(51);

            ThucHienMergeSort(DanhSachNode, DanhSachNode.Count);

            foreach (Node node in DanhSachNode)

            {

                node.BackColor = ThamSo.mauNodeHTSX;

            }

            DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;

            DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;

            hoanTatSapXep();

        }

        #endregion

*Hàm thêm ý tưởng vào và hiện code c và chạy code theo từng bước*

public class HienThiThuatToan

    {

        //Một số biến toàn cục

        public  static ListBox codeListBox;

        public static TextBox yTuongThuatToan;

        public static string yTuong;

        public static ManualResetEvent tamdunglistbox = new ManualResetEvent(true);

        public static string[] ChuyenText(string doc)

        {

            string[] text;

            text = doc.Split('\n');//Cac string den xuong dong

            return text;

        }

        #region Hàm add Ý Tưởng

        private static void addYTuong()

        {

            yTuongThuatToan.Clear();

            yTuongThuatToan.Text = yTuong;

        }

        #endregion

        public static void ChayCodeC(int line)

        {

            Thread.Sleep(ThamSo.ThoiGianDoi \* 50);

            // chờ sign nếu có thì đợi đến vô tận

            tamdunglistbox.WaitOne(Timeout.Infinite);

            codeListBox.SelectedIndex = line;

            if (Mainform.isDebug==true)

            {

                tamdunglistbox.Reset();

            }

        }

        #region  Các thuật toán

        #region Interchange Sort

        public static void InterchangeSort(bool tang = true)

        {

            yTuong =

@"Xuất phát từ đầu dãy, tìm tất các các nghịch thế chứa phần tử này, triệt tiêu chúng bằng cách đổi chỗ 2 phần tử trong cặp nghịch thế. Lặp lại xử lý trên với phần tử kế trong dãy.";

            //Thêm yTuong vào yTuongThuatToan

            addYTuong();

            string[] code = ChuyenText(

@"void InterchangeSort( int a[], int N)

{

    int i, j;

    for(i = 0; i < N - 1; i++)

        for(j = i + 1; j < N; j++)

            if( a[j] < a[i] )

                Swap( a[i], a[j]);

}

");

            //thêm code vào codeListBox

            codeListBox.Items.Clear();

            foreach (string item in code)

            {

                codeListBox.Items.Add(item);

            }

            //nếu sắp giảm thì sửa lại

            if (!tang)

            {

                codeListBox.Items[5] = "            if( a[j] > a[i] )";

            }

        }

        #endregion

        #region Insertion Sort

        public static void InsertionSort(bool tang = true)

        {

            yTuong =

@"Giả sử có một dãy a0, a1,... ,an-1 trong đó i phần tử đầu tiên a0, a1,... ,ai-1 đã có thứ tự.

Tìm cách chèn phần tử ai vào vị trí thích hợp của đoạn đã được sắp để có dãy mới a0, a1,... ,ai trở nên có thứ tự. Vị trí này chính là vị trí giữa hai phần tử ak-1 và ak thỏa ak-1 < ai < ak(1≤k≤i).";

            // Thêm yTuong vào yTuongThuatToan

            addYTuong();

            string[] code = ChuyenText(

@"void InsertionSort(int a[], int N)

{

    int pos, i;

    int x;

    for(i = 1; i < N; i++)

    {

        x = a[i]; pos = i - 1;

        while((pos >= 0) && (x < a[pos]))

        {

            a[pos + 1] = a[pos];

            pos--;

        }

        a[pos + 1] = x;

    }

}");

            // Thêm code vào codeListBox

            codeListBox.Items.Clear();

            foreach (string item in code)

            {

                codeListBox.Items.Add(item);

            }

            if (!tang)

            {

                codeListBox.Items[7] = "        while((pos >= 0) && (x > a[pos]))";

            }

        }

        #endregion

        #region Selection Sort

        public static void SelectionSort(bool tang = true)

        {

            //vietnamese

            yTuong =

@"Chọn phần tử nhỏ nhất trong N phần tử trong dãy hiện hành ban đầu.

Đưa phần tử này về vị trí đầu dãy hiện hành.

Xem dãy hiện hành chỉ còn N-1 phần tử của dãy hiện hành 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 1 phần tử.";

            //Thêm yTuong vào yTuongThuatToan

            addYTuong();

            string[] code = ChuyenText(

@"void SelecttionSort(int arr[], int N)

{

    int min, i, j;

    for (i=0; i < N-1; i++)

        {

                min = i;

                for (j=i+1; j <N; j++)

                if (a[j] < a[min])

                    min=j;

                    Swap(a[min], a[i]);

        }

}");

            //thêm code vào codeListBox

            codeListBox.Items.Clear();

            foreach (string item in code)

            {

                codeListBox.Items.Add(item);

            }

            //Nếu sắp xếp giảm dần thì sửa lại

            if (!tang)

            {

                codeListBox.Items[7] = "             if (a[j] > a[min])";

            }

        }

        #endregion

        #region Quick Sort

        public static void QuickSort(bool tang = true)

        {

            yTuong =

@"Giải thuật QuickSort sắp xếp dãy a1, a2 ..., aN dựa trên việc phân hoạch dãy ban đầu thành 3 phần :

Phần 1:Gồm các phần tử  có giá trị bé hơn x

Phần 2: Gồm các phần tử  có giá trị bằng  x

Phần 3: Gồm các phần tử  có giá trị lớn hơn x

Với x là giá trị của một phần tử  tùy ý trong dãy ban đầu.";

            //Thêm yTuong vào yTuongThuatToan

            addYTuong();

            string[] code = ChuyenText(

@"void QuickSort(int a[], int left, int right)

{

    int i, j, x;

    x = a[(left + night)/2];

    i = left, j = right;

    do

    {

        while(a[i] < x)

            i++;

        while(x < a[j])

            j--;

        if(i <= j)

        {

            Swap(a[i], a[j]);

            i++, j--;

        }

    }while(i <= j);

    if(left < j)

        QuickSort(a, left, j);

    if(i < right)

        QuickSort(a, i, right);

}");

            //thêm code vào codeListBox

            codeListBox.Items.Clear();

            foreach (string item in code)

            {

                codeListBox.Items.Add(item);

            }

            //nếu sắp giảm thì sửa lại

            if (!tang)

            {

                codeListBox.Items[7] = "        while(a[i] > x)";

                codeListBox.Items[9] = "        while(x > a[j])";

            }

        }

        #endregion

        #region Bubble Sort

        public static void BubbleSort(bool tang = true)

        {

            yTuong =

@"Xuất phát từ cuối dãy, đổi chỗ các cặp phần tử kế cận để đưa phần tử nhỏ hơn trong cặp phần tử đó về vị trí đúng đầu dãy hiện hành, sau đó sẽ không xét đến nó ở bước tiếp theo, do vậy ở lần xử lý thứ i sẽ có vị trí đầu dãy là i.

Lặp lại xử lý trên cho đến khi không còn cặp phần tử nào để xét.";

            //Thêm yTuong vào yTuongThuatToan

            addYTuong();

            string[] code = ChuyenText(

@"void BubbleSort(int a[], int N)

{

    int i,j;

    for(i = 0; i < N - 1; i++)

        for(j = N - 1; j > i; j--)

        if(a[j] < a[j - 1])

            Swap(a[j], a[j - 1]);

}

");

            //thêm code vào codeListBox

            codeListBox.Items.Clear();

            foreach (string item in code)

            {

                codeListBox.Items.Add(item);

            }

            //nếu sắp giảm thì sửa lại

            if (!tang)

            {

                codeListBox.Items[5] = "       if(a[j] > a[j - 1])";

            }

        }

        #endregion

        #region Merge Sort

        public static void MergeSort(bool tang = true)

        {

            yTuong =

@"Sắp xếp dãy a(1),a(2),...,a(n) dựa trên nhận xét sau :

-Mỗi dãy a(1),a(2),...,a(n) bất kỳ là một tập hợp các dãy con liên tiếp mà mỗi dãy con đều đã có thứ tự.

-Dãy đã có thứ tự coi như có 1 dãy con.

Hướng tiếp cận : tìm cách làm giảm số dãy con không giảm của dãy ban đầu.";

            //Thêm yTuong vào yTuongThuatToan

            addYTuong();

            string[] code = ChuyenText(

@"int b[MAX], c[MAX], nb, nc;

int Min(int a, int b)

{

    if(a > b) return b;

    else return a;

}

void Distribute(int a[], int N, int &nb, int &nc, int k)

{

    int i, pa, pb, pc;

    pa = pb = pc = 0 ;

    while(pa < N)

    {

        for(i = 0; (pa < N) && (i < k); i++, pa++, pb++)

            b[pb] = a[pa];

        for(i = 0; (pa < N) && (i < k); i++, pa++, pc++)

            c[pc] = a[pa];

    }

    nb = pb; nc = pc;

}

void Merge(int a[], int nb, int nc, int k)

{

    int p, pb, pc, ib, ic, kb, kc;

    p = pb = pc = 0; ib = ic = 0;

    while((nb > 0) && (nc > 0))

    {

        kb = Min(k, nb); kc = Min(k, nc);

        if(c[pc + ic] < b[pb + ib] == false)

        {

            a[p++] = b[pb + ib]; ib++;

            if(ib == kb)

            {

                for(;ic < kc; ic++)

                    a[p++] = c[pc + ic];

                pb += kb; pc += kc; ib = ic = 0;

                nb -= kb; nc -= kc;

            }

        }

        else

        {

            a[p++] = c[pc + ic]; ic++;

            if(ic == kc)

            {

                for(;ib < kb; ib++)

                    a[p++] = b[pb + ib];

                pb += kb; pc += kc; ib = ib = 0;

                nb -= kb; nc -= kc;

            }

        }

    }

}

void MergeSort(int a[], int N)

{

        int k;

        for(k = 1; k < N; k\*= 2)

        {

            Distribute(a, N, nb, nc, k);

            Merge(a, nb, nc, k);

        }

}

");

            //thêm code vào codeListBox

            codeListBox.Items.Clear();

            foreach (string item in code)

            {

                codeListBox.Items.Add(item);

            }

            //nếu sắp giảm thì sửa lại

            if (!tang)

            {

                codeListBox.Items[26] = "        if(c[pc + ic] > b[pb + ib] == false)";

            }

        }

        #endregion

        #endregion

    }

*Hàm xử node để di chuyển*

        public Node(int vitrihientai)

        {

            InitializeComponent();

            DatGiaTriMatDinh();

            this.vitriHienTai = vitrihientai;

        }

        private void DatGiaTriMatDinh()

        {

            this.Width = this.Height = btn1.Width = btn1.Height = tbx1.Width = tbx1.Height = ThamSo.KichCoNode;

            tbx1.Location = this.Location;// neu 1 trong 3 thay doi vi tri thi tat ca deu thay doi theo

            btn1.Location = this.Location;

            tbx1.TextAlign = HorizontalAlignment.Center;

            this.Size = new Size(ThamSo.KichCoNode, ThamSo.KichCoNode);

            this.Anchor = (AnchorStyles.Top | AnchorStyles.Left);

            btn1.BackColor = Color.Red;

            tbx1.BackColor = Color.White;

            tbx1.MaxLength = 2;// chi nhap toi da la 99

            tbx1.AcceptsReturn = true;

            btn1.Font =tbx1.Font= new Font("Consolas", ThamSo.KichCoNode /4, FontStyle.Bold);

            //this.LostFocus += Node\_LostFocus;

            tbx1.LostFocus += Tbx1\_LostFocus;

        }

        private void Tbx1\_LostFocus(object sender, EventArgs e)

        {

            if (tbx1.Text=="")

            {

                tbx1.Text = "0";

            }

            else

            {

                this.Text = tbx1.Text;

                this.Value = Int32.Parse(this.Text);

            }

            btn1.Visible = true;

            btn1.BringToFront();

            btn1.Focus();

            if (Capnhat != null)

            {

                Capnhat(sender, e);

            }

        }

        private int value;

       //public static ManualResetEvent pauseStatus = new ManualResetEvent(true);

        public static bool IsPause = false;

        public event EventHandler Capnhat;

        public override string Text

        {

            get

            {

                return btn1.Text;

            }

            set

            {

                btn1.Text = value.ToString();

            }

        }

        public int Value

        {

            get

            {

                return value;

            }

            set

            {

                this.value = value;

            }

        }

        public int vitriHienTai;

        private void btn1\_MouseDown(object sender, MouseEventArgs e)

        {

            if (e.Clicks == 2)

            {

                btn1.Visible = false;

                tbx1.Text = btn1.Text;

                tbx1.SelectionStart = tbx1.Text.Length;// dat con tro chuot "cusor" qua ben phai neu so da co

                tbx1.BringToFront();

                tbx1.Focus();

            }

        }

        private void tbx1\_KeyPress(object sender, KeyPressEventArgs e)

        {

            if (!Char.IsDigit(e.KeyChar))

            {

                e.Handled = true;

            }

        }

        private void tbx1\_Enter(object sender, EventArgs e)

        {

            if (tbx1.Text!=null)

            {

                btn1.Text = tbx1.Text;

            }

        }

        private void tbx1\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                btn1.Visible = true;

                btn1.BringToFront();

                btn1.Focus();

            }

        }

        private void tbx1\_TextChanged(object sender, EventArgs e)

        {

            //if (tbx1.Text != "")

            //{

            //    //  value = Int32.Parse(tbx1.Text);

        }

        /// <summary>

        /// Các hàm di chuyển node;

        /// </summary>

        /// <param name="Node"></param>

        public static System.Threading.ManualResetEvent pauseStatus = new System.Threading.ManualResetEvent(true);

        public void ChuyenLen()

        {

            int y\_ViTriMoi = this.Location.Y - ThamSo.DoCaoDiChuyen;

            while (this.Location.Y > y\_ViTriMoi)

            {

                pauseStatus.WaitOne(Timeout.Infinite);

                this.Location = new Point(this.Location.X, this.Location.Y - 1);

                Thread.Sleep(ThamSo.ThoiGianDoi);

            }

        }

        public void ChuyenXuong()

        {

            int y\_ViTriMoi = this.Location.Y+ ThamSo.DoCaoDiChuyen;

            while (this.Location.Y < y\_ViTriMoi)

            {

                pauseStatus.WaitOne(Timeout.Infinite);

                this.Location = new Point(this.Location.X, this.Location.Y + 1);

                Thread.Sleep(ThamSo.ThoiGianDoi);

            }

        }

        public void ChuyenNgang(int vitriMoi)

        {

            int x\_vitririMoi;

            if (vitriMoi > this.vitriHienTai)

            {

                x\_vitririMoi = this.Location.X + ((vitriMoi - vitriHienTai) \* (ThamSo.KhoangCachCacNode+ThamSo.KichCoNode));

                while (this.Location.X < x\_vitririMoi)

                {

                    pauseStatus.WaitOne(Timeout.Infinite);

                    Thread.Sleep(ThamSo.ThoiGianDoi);

                    this.Location = new Point(this.Location.X + 2, this.Location.Y);

                }

            }

            else

            {

                x\_vitririMoi = this.Location.X - ((vitriHienTai - vitriMoi) \* (ThamSo.KhoangCachCacNode+ThamSo.KichCoNode));

                while (this.Location.X > x\_vitririMoi)

                {

                    pauseStatus.WaitOne(Timeout.Infinite);

                    Thread.Sleep(ThamSo.ThoiGianDoi);

                    this.Location = new Point(this.Location.X - 2, this.Location.Y);

                }

            }

        }

        private void TbxBtn\_BackColorChanged(object sender, EventArgs e)

        {

            btn1.BackColor = this.BackColor;

        }

        private void TbxBtn\_Load(object sender, EventArgs e)

        {

        }

        private void btn1\_Click(object sender, EventArgs e)

        {

        }

*Hàm xử lý nhập mảng bằng tay*

        public frmInput()

        {

            InitializeComponent();

            textBox2.Text = ThamSo.SoluongNodeLonNhat.ToString();

        }

        public List<int> DayInput;

        public bool isNhap;

        private void frmInput\_Load(object sender, EventArgs e)

        {

            DayInput = new List<int>();

            isNhap = false;

        }

        private void btnhuy\_Click(object sender, EventArgs e)

        {

            this.Close();

        }

        #region  Nhập mảng sắp xếp

        private void btnnhap\_Click(object sender, EventArgs e)

        {

            // xoa day cũ nếu đã tạo ở phiên trước

            DayInput.Clear();

            foreach (string item in tbxdayso.Text.Split(' '))

            {

                if (item == " ")

                {

                }

                else

                {

                    int temp = 10;

                    bool result = int.TryParse(item, out temp);

                    if (result == true && (temp < 0 || temp > 99))

                    {

                        MessageBox.Show("Phần tử có giá trị nhỏ hơn 0 hoặc lớn hơn 99");

                        return;

                    }

                    if (result == true)

                    {

                        DayInput.Add(temp);

                    }

                }

            }

            if (DayInput.Count > ThamSo.SoluongNodeLonNhat)

            {

                MessageBox.Show("Dãy vừa nhập vượt quá " + ThamSo.SoluongNodeLonNhat);

                return;

            }

            if (DayInput == null)

            {

                MessageBox.Show("Dãy nhập rỗng");

                return;

            }

            isNhap = true;

            this.Close();

        }

        private void tbxdayso\_KeyPress(object sender, KeyPressEventArgs e)

        {

            // chặn người dùng nhập chữ nhưng cho phép nhập backspace để xóa

            if (e.KeyChar != '\b' && !char.IsWhiteSpace(e.KeyChar))

            {

                e.Handled = !char.IsNumber(e.KeyChar);

            }

        }

        #endregion