BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO THỰC TẬP CƠ SỞ CÔNG NGHỆ THÔNG TIN**

**MÔ PHỎNG THUẬT TOÁN SẮP XẾP NỔI BỌT BẰNG C#**

**Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh**

**Sinh viên thực hiện: Phù Quốc khánh**

**Mã số sinh viên: 59131051**

KHÁNH HÒA - 2022

TRƯỜNG ĐẠI HỌC NHA TRANG

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN HỆ THỐNG THÔNG TIN**



**BÁO CÁO THỰC TẬP CƠ SỞ CÔNG NGHỆ THÔNG TIN**

**MÔ PHỎNG THUẬT TOÁN SẮP XẾP NỔI BỌT BẰNG C#**

Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh

Sinh viên thực hiện: Phù Quốc Khánh

Mã số sinh viên: 59131051

Khánh Hòa, tháng 01/2022

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**Khoa: Công nghệ Thông tin**

**PHIẾU THEO DÕI TIẾN ĐỘ VÀ ĐÁNH GIÁ BÁO CÁO THỰC TẬP CƠ SỞ**

**Tên đề tài:** Mô phỏng thuật toán sắp xếp nổi bọt bằng C#

**Giảng viên hướng dẫn:** ThS. Đoàn Vũ Thịnh

**Sinh viên được hướng dẫn:** Phù Quốc Khánh

**MSSV:** 59131051

**Khóa:** 59 **Ngành:** Công nghệ Thông tin

|  |  |  |  |
| --- | --- | --- | --- |
| **Lần** | **Ngày** | **Nội dung** | **Nhận xét của GVHD** |
| 1 | 6/12 | Nhận đề tài hướng dẫn và định hướng giải vấn đề. Sau đó, lên kế hoạch triển khai phần mềm | . |
| 2 | 13/12 | Sinh viên trình bày mô phỏng thuật toán dựa trên kiến thức đã học ở môn Cấu Trúc Dữ Liệu và Giải Thuật và các kiến thức thu được từ Internet để minh họa bài toán đa dạng nhất có thể |  |
| 3 | 20/12 | Sinh viên hoàn thiện thuật toán đã đề ra với dữ liệu được in trên màn hình. Và tiếp tục xử lý đồ họa cho thuật toán |  |
| 4 | 27/12 | Sinh viên demo chương trình và trình bày và bắt đầu tiến hành chỉnh sửa vài điểm sai hoặc chưa hoàn hảo |  |
| 5 | 4/12 | Sinh viên nộp bản thảo lần 1 |  |
| 6 | 5/12 | Sinh viên nộp bản thảo lần 2 khi đã chỉnh sửa một số điểm sai trong bài báo cáo lần 1 |  |

**Nhận xét chung (sau khi sinh viên hoàn thành ĐA/KL)**:

Khánh Hòa, ngày 05 tháng 01 năm 2022

Cán bộ hướng dẫn

*(Ký và ghi rõ họ tên)*

lời cam kết

Tôi xin cam đoan mọi kết quả của đề tài “Mô phỏng thuật toán sắp xếp nổi bọt bằng C#” là công trình nguyên cứu cá nhân tôi và chưa từng công bố trong bất cứ công trình khoa học nào khác cho tới thời điểm này.

Khánh Hòa, ngày 05 tháng 01 năm 2022

Tác giả đồ án

(ký và ghi rõ họ tên)

LỜI CẢM ƠN

Trong suốt thời gian thực hiện đề tài, tôi đã nhận được sự giúp đỡ của quý phòng ban trường Đại học Nha Trang, Khoa Công nghệ Thông tin nói chung và Bộ môn Kỹ thuật phần mềm nói riêng đã tạo điều kiện tốt nhất cho tôi được hoàn thành đề tài. Đặc biệt là sự hướng dẫn tận tình của ThS Đoàn Vũ Thịnh đã giúp tôi hoàn thành tốt đề tài. Qua đây, tôi xin gửi lời cảm ơn sâu sắc đến sự giúp đỡ này.

Xin chân thành cảm ơn các quý thầy cô trong bộ môn Kỹ thuật phần mềm trong thời gian qua đã truyền đạt và trang bị cho em kiến thức giúp em hoàn thành tốt đề tài tốt nghiệp.

Cuối cùng tôi xin gửi lời cảm ơn chân thành đến gia đình và tất cả bạn bè đã giúp đỡ, động viên tôi trong suốt quá trình học tập và thực hiện đề tài.

Tôi xin chân thành cảm ơn!

Khánh Hòa, ngày 05 tháng 01 năm 2022

Tác giả đồ án

(ký và ghi rõ họ tên)

Mục lục

[LỜI CAM KẾT i](#_Toc92290322)

[LỜI CẢM ƠN ii](#_Toc92290323)

[DANH MỤC HÌNH v](#_Toc92290325)

[GIỚI THIỆU vi](#_Toc92290326)

[CHƯƠNG 1 GIỚI THIỆU CHUNG 1](#_Toc92290327)

[1.1 Tổng quan Sắp xếp nổi bọt 1](#_Toc92290328)

[1.2 Giải thuật sắp xếp nổi bọt 1](#_Toc92290329)

[1.1.1 Sắp xếp từ trên xuống 1](#_Toc92290330)

[1.1.2 Sắp xếp từ dưới lên 1](#_Toc92290331)

[1.1.3 Thời gian tính 1](#_Toc92290332)

[CHƯƠNG 2 PHÂN TÍCH KỸ THUẬT XỬ LÝ 2](#_Toc92290333)

[2.1.1 Phân tích hiệu ứng 2](#_Toc92290334)

[2.1.2 Phân tích chức năng 2](#_Toc92290335)

[2.2 Kỹ thuật sử dụng 3](#_Toc92290336)

[2.2.1 Xử lý bất đồng bộ với Thread và Task 3](#_Toc92290337)

[2.2.2 Xử lý việc thay dổi tốc độ sắp xếp 4](#_Toc92290338)

[2.2.3 Xử lý việc di chuyển các phần tử và các biến 4](#_Toc92290339)

[2.2.4 Xử lý tính thời gian thực hiện quá trình sắp xếp 4](#_Toc92290340)

[CHƯƠNG 3 Thiết Kế Giao Diện 5](#_Toc92290341)

[CHƯƠNG 4 CÀI ĐẶT 10](#_Toc92290342)

[4.1 Nền tảng công nghệ 10](#_Toc92290343)

[4.2 Các hàm xử lý chính 10](#_Toc92290344)

[4.2.1 Phần tử và các hàm có trong phần tử 10](#_Toc92290345)

[4.2.2 Các hàm tạo mảng 15](#_Toc92290346)

[4.2.3 Tạo mảng bằng tay 18](#_Toc92290347)

[4.2.4 Các hàm bắt đầu, hủy quá trình sắp xếp và các hàm hỗ trợ 19](#_Toc92290348)

[4.2.5 Các hàm di chuyển vị trí của Node và hoán vị hai Node 21](#_Toc92290349)

[4.2.6 Sự kiện tạm dừng quá trình sắp xếp 22](#_Toc92290350)

[4.2.7 Hàm sắp xếp mảng nổi bọt 23](#_Toc92290351)

[CHƯƠNG 5 KẾT LUẬN VÀ ĐÁNH GIÁ BẢN THÂN 26](#_Toc92290352)

[Tài Liệu Kham Khảo 27](#_Toc92290353)

DANH MỤC HÌNH

[Hình 3.1: Giao diện chính của chương trình 5](#_Toc92295962)

[Hình 3.2: thực hiện bấm ngẫu nhiên một dạy số 5](#_Toc92295963)

[Hình 3.3: Thực hiện thay đổi số khi bấm double-click vào node 6](#_Toc92295964)

[Hình 3.4: thực hiện nhập khởi tạo nhập tay 6](#_Toc92295965)

[Hình 3.5: Chương trình cho phép chọn tăng dần hoặc giảm dần 7](#_Toc92295966)

[Hình 3.6: Chương trình bắt đầu thực hiện 7](#_Toc92295967)

[Hình 3.7: Chương trình có thể tăng giảm tốc độ di chuyển hoặc tạm dừng 8](#_Toc92295968)

[Hình 3.7: Chương trình có sự kiện xóa mảng 8](#_Toc92295969)

[Hình 3.8: Hiển thị thông báo hoàn tất việc sắp xếp 9](#_Toc92295970)

**giới thiệu**

Winform là thuật ngữ mô tả một ứng dụng được viết dùng .NET FrameWorrk và có giao diện người dùng Windows Forms. Mỗi Màn hình windows cung cấp một giao diện giúp người dùng giao tiếp với ứng dụng. Giao diện này được gọi là giao diện đồ họa (GUI) của ứng dụng.Là các ứng dụng windows chạy trên máy tính – mã lệnh thực thi ngay trên máy tính: Microsoft, Word, Excel, Access, Calculator, yahoo, Mail… là các ứng dụng Windows Forms. Và Sắp xếp nổi bọt là một thuật toán sắp xếp vô cùng gần gủi với lập trình viên chúng ta. Quy trình thực hiện được trải qua các bước từ thiết kế giao diện đến cài đặt thuật toán và đều thực hiện trên ngôn ngữ C#. Kết quả của việc cài đặt thuật toán đáp ứng được các yêu cầu của thực tập cơ sơ này.

Nội dung của bài báo cáo gồm 5 Chương:

Chương 1: Giới thiệu chung về sắp xếp nổi bọt

Chương 2: Phân tích kỹ thuật xử lý

Chương 3: Thiết kế giao diện

**Chương 4:** Cài đặt

**Chương 5:** Kết luận và đánh giá bản thân

CHƯƠNG 1 GIỚI THIỆU CHUNG

1.1 Tổng quan Sắp xếp nổi bọt

Sắp xếp nổi bọt (tiếng Anh: bubble sort) là một thuật toán sắp xếp 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.

1.2 Giải thuật sắp xếp nổi bọt

1.1.1 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....

1.1.2 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.

1.1.3 Thời gian tính

Với mỗi I = 1,2,...,n-1 ta cần i phép so sánh. Do đó số nhiều nhất các lần so sánh và đổi chỗ trong giải thuật là

(n - 1) + (n - 2) +…+ 2 + 1 =

Do đó độ phức tạp của giải thuật là O(n2)

CHƯƠNG 2 PHÂN TÍCH KỸ THUẬT XỬ LÝ

2.1 Phân tích

2.1.1 Phân tích hiệu ứng

Để mô phỏng một phần tử trên mảng ta cần tạo một user controls gọi là Node để lưu giá trị của phần tử mảng và hiển thị trực tiếp lên GUI, có thanh trackbar cho người dùng có thể dễ dàng thay đổi tốc độ của quá trình sắp xếp

Trong lúc các phần tử di chuyển thì phần code thể hiện của thuật toán cũng được chọn từng dòng tương ứng với việc sắp xếp các phần tử.

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

Ngoài ra các biến trong thuật toán còn được biễu diễn trên form một cách trực quan sao cho phù hợp với vị trí của phần tử và có giá trị tương ứng với giá trị của biến trong thuật toán.Phân tích hiệu ứng

2.1.2 Phân tích chức năng

1. **Khởi tạo mảng (2 cách) :**
   1. Tạo ngẫu nhiên : tạo một mảng với các phần tử có giá trị bất kì và số phần tử bất kì
   2. Nhập bằng tay : cho người dùng nhập giá trị của từng phần tử một. Người dùng có thể thay đổi giá trị của một phần tử bất kì theo ý muốn.
2. **Bảng điều khiển:**
   * Bắt đầu sắp xếp : bắt đầu sắp xếp các phần tử dựa vào thuật toán được chọn
   * Tạm dừng : cho tạm dừng quá trình sắp xếp
   * Thời gian thực hiện : cho xem tổng thời gian thực hiện quá trình sắp xếp
3. **Các chức năng phụ:** 
   * Thay đổi tốc độ quá trình sắp xếp
   * Cập nhật thông tin cho quá trình sắp xếp
   * Cho phép chạy thuật toán ở hai hướng là tăng và giảm

2.2 Kỹ thuật sử dụng

2.2.1 Xử lý bất đồng bộ với Thread và Task

Để có thể mô phỏng việc di chuyển của các phần tử khi sắp xếp ta cần sử dụng kỹ thuật lập trình bất đồng bộ. Để lập trình bất đồng bộ ta có thể sử dụng Thread hoặc Task hoặc có thể sử dụng cả hai

Để có thể nhìn thấy di chuyển mượt khi sắp xếp ta cần sử dụng hàm Thread.Sleep(<thời gian>) để cho tiến trình dừng lại trong một khoảng thời gian nhất định (milisecond) và giúp cho người dùng kịp nhìn thấy sự thay đổi vị trí của các phần tử

Để chọn từng dòng code trong khung Code C/C++ thì ta cần tạo một tiến trình chạy song song với tiến trình chính để khi tiến trình thực hiện sẽ không ngăn cản (block) tiến trình chính tiếp nhận các sự kiện (event)

Để có thể có hai chuyển động (hai hàm di chuyển phần tử) cùng chạy một lúc thì ta cần tạo ra hai tiến trình chạy song song với tiền trình chính

Khi sử dụng Thread ta có thể hủy Thread bất kỳ khi nào mình muốn nhưng Thread lại không hỗ trợ các thư viện giúp thông báo khi tiến trình đã hoàn tất (callback). Vì thế ta sử dụng Task, Task hỗ trợ đủ các hàm thông báo việc tiến trình kết thúc nhưng lại thiếu hàm hủy Task khi thực hiện, việc hủy Task thông qua một CancellationTokenSource và cần phải thực hiện nhiều việc mới có thể hủy một Task đang thực hiện,

Vì các nhược điểm và ưu điểm trên nên ta kết hợp việc sử dụng Thread và Task lại để thực hiện quá trình sắp xếp, Ta cho tạo một Thread để chạy chọn từng dòng code, bên trong Thread đó ta tạo ra hai Task con để thực hiện di chuyển các phần tử. Vì Vậy ta có thể báo việc hoàn tất di chuyển của các phần tử và vừa có thể hủy bất cứ khi nào

Khi xử lý đa hình ta cũng gặp phải một vấn đề quan trọng đó là việc các tiến trình không cho phép tiến trình khác thay đổi hay chỉnh sửa các Control được tạo ra bên trong tiến trình của mình. Ví dụ: khi ta tạo ra tiến trình con để thay đổi đoạn Text trên một Button của tiến trình chính thì Compiler sẽ báo lỗi. Khi đó ta cần thông báo cho Compiler biết là sẽ bỏ qua và không kiểm tra lỗi này

Việc tạm dừng các tiến trình ta sẽ sử dụng đến ManualResetEvent đây là một lớp có thể dùng để tạm dừng các tiến trình đang chạy và tiếp tục các tiến trình khi cần

2.2.2 Xử lý việc thay dổi tốc độ sắp xếp

Do ở mỗi vòng lặp khi thay đổi tọa độ của phần tử ta đều lấy giá trị tốc độ hiện tại được lưu trữ trong tham số nên khi thay đổi giá trị tốc độ hiện tại thì các vòng lặp trong tương lai sẽ sử dụng giá trị tốc độ mới. Điều này sẽ làm thay đổi tốc độ di chuyển các phần tử, dẫn đến thay đổi tốc độ sắp xếp

2.2.3 Xử lý việc di chuyển các phần tử và các biến

Để xác định được tọa độ đặt và tọa độ di chuyển các phần tử và các biến ta cần dựa vào chiều dài, rộng của panel xác định được tọa độ gốc. Từ tọa độ gốc ta dựa vào vị trí phần tử trong mảng để tính ra tọa độ mà phần tử sẽ được đặt trên panel và cũng dùng cách tương tự để xác định tọa độ đích khi cần di chuyển phần tử.

Đối với các biến có trong các thuật toán sắp xếp ta cũng làm tương tự như cách trên. Ngoài ra các biến và các phần tử cũng được đổi màu để người sử dụng dễ dàng phân biệt.

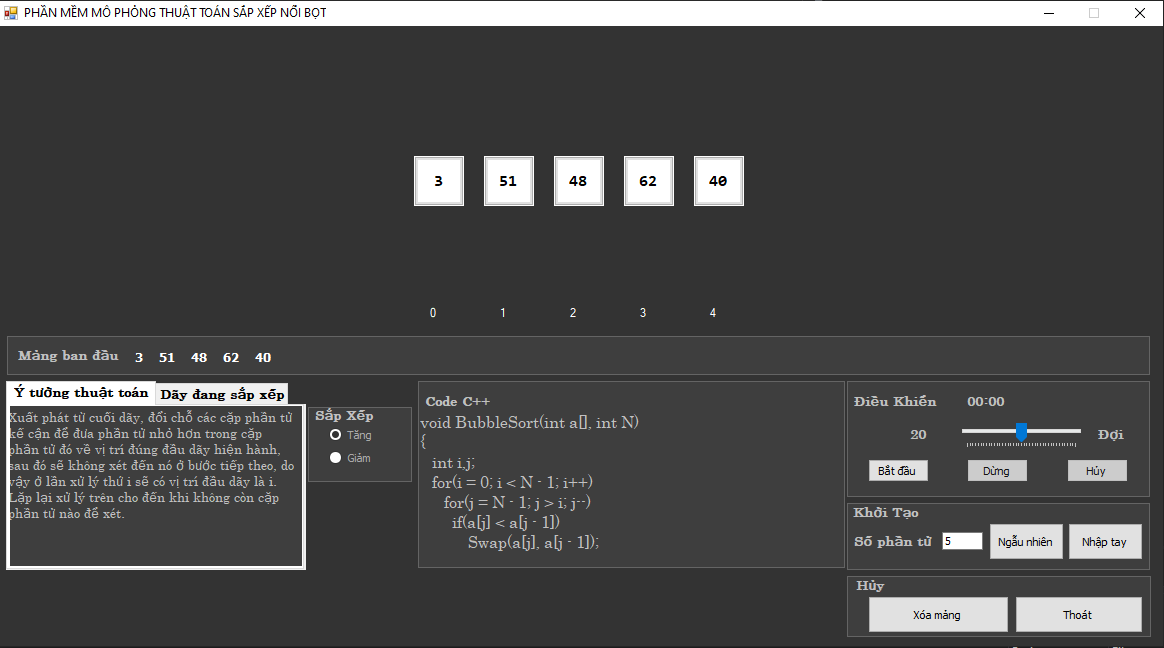
2.2.4 Xử lý tính thời gian thực hiện quá trình sắp xếp

Để xác định được tọa độ đặt và tọa độ di chuyển các phần tử và các biến ta cần dựa vào chiều dài, rộng của panel xác định được tọa độ gốc. Từ tọa độ gốc ta dựa vào vị trí phần tử trong mảng để tính ra tọa độ mà phần tử sẽ được đặt trên panel và cũng dùng cách tương tự để xác định tọa độ đích khi cần di chuyển phần tử.

Đối với các biến có trong các thuật toán sắp xếp ta cũng làm tương tự như cách trên. Ngoài ra các biến và các phần tử cũng được đổi màu để người sử dụng dễ dàng phân biệt.

CHƯƠNG 3 Thiết Kế Giao Diện

* Giao diện chương trình khi khởi động, chương trình sẽ tạo sẵn cho người dùng 5 Node để kiểm tra



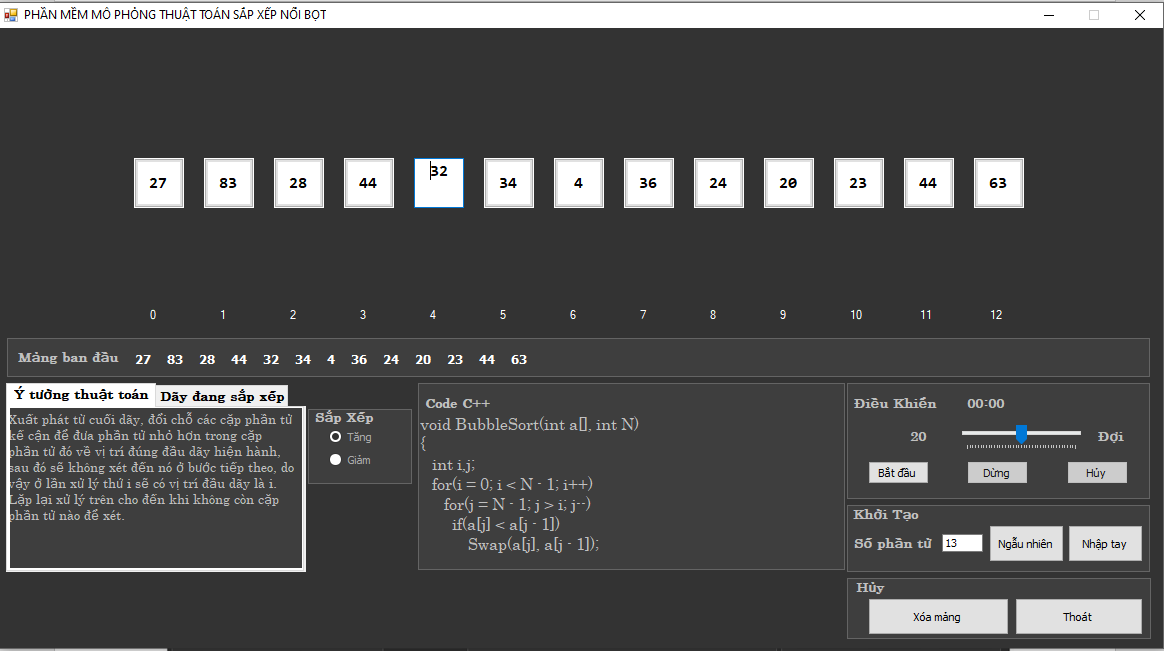
Hình 3.1: Giao diện chính của chương trình

* Người dùng có thể khởi tạo input đầu vào 2 cách: ngẫu nhiên, nhập tay
  + Khi bấm ngẫu nhiên chương trình sẽ tự động cấp phát một mảng ngẫu nhiên n phần tử khoảng [2,15] và miền giá trị của từng node là từ [1,99]



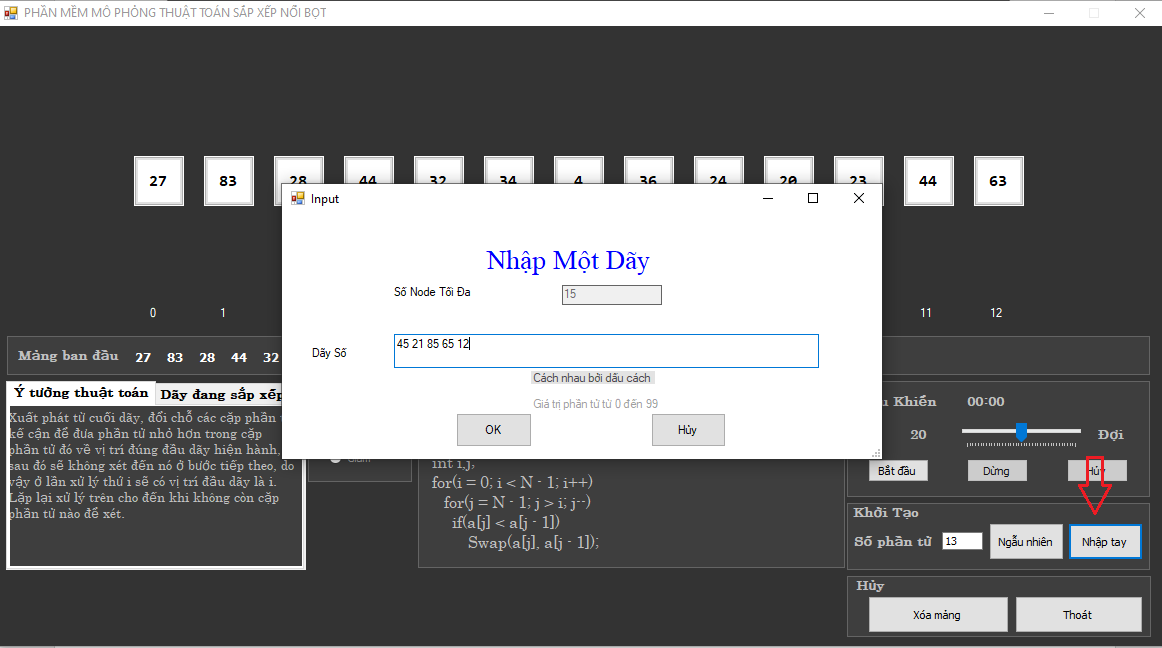
Hình 3.2:thực hiện bấm ngẫu nhiên một dạy số

* + Người dùng cũng có thể thay đổi input đầu vào của mảng bằng cách double-click vào bất kỳ Node nào cần thay đổi giá trị sau đó nhấn Enter hoặc Click vào panel chương trình sẽ tự động cập nhật lại giá trị của Node



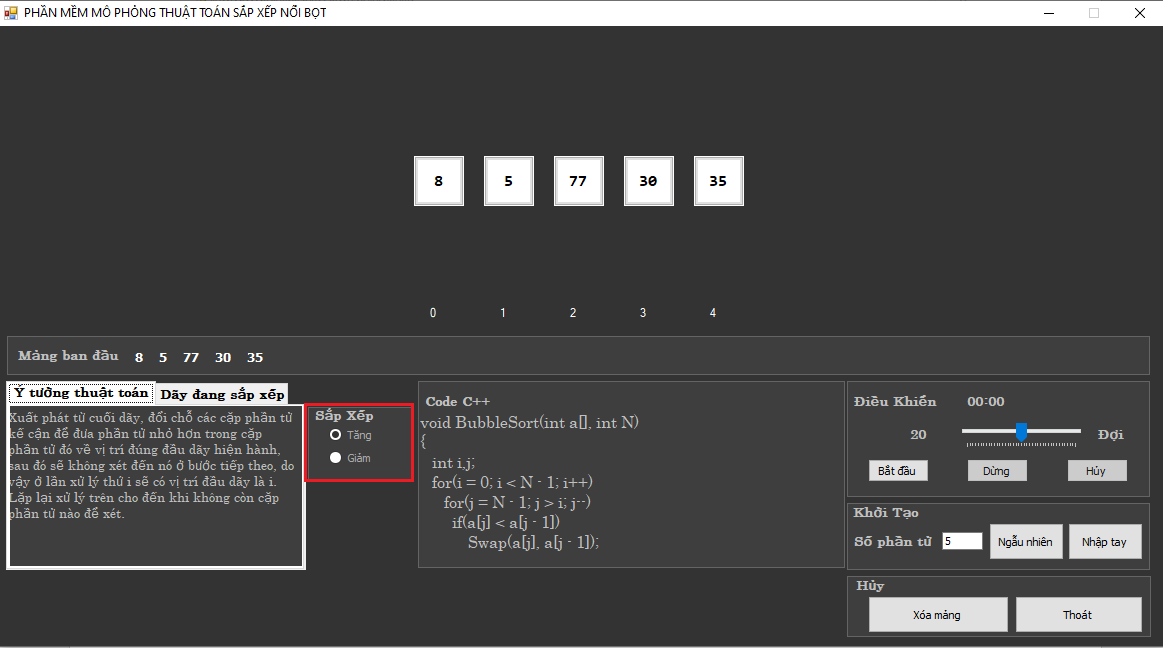
Hình 3.3: Thực hiện thay đổi số khi bấm double-click vào node

* + Ngoài ra người dùng có thể khởi tạo bằng chức năng nhập tay chương trình sẽ tạo một form con cho phép người dùng nhập giá trị mảng đầu vào



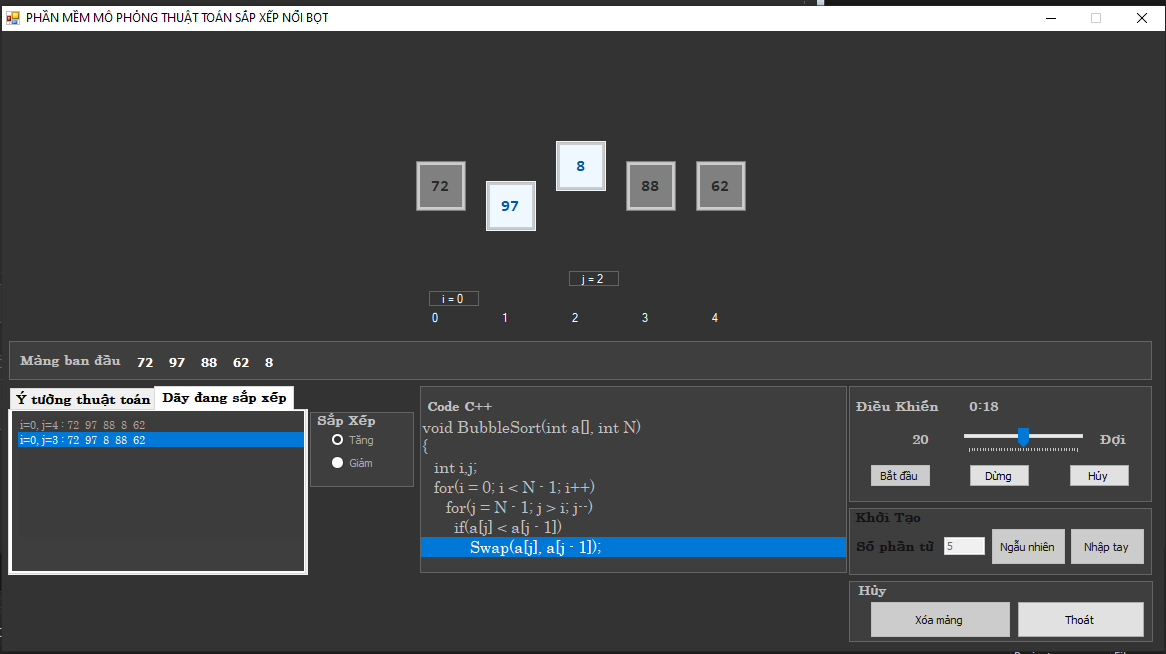
Hình 3.4: thực hiện nhập khởi tạo nhập tay

* Người dùng có thể chọn sắp xếp tăng dần hoặc giảm dần. Khi mở chương trình, chương trình sẽ auto Checked sắp xếp tăng dần



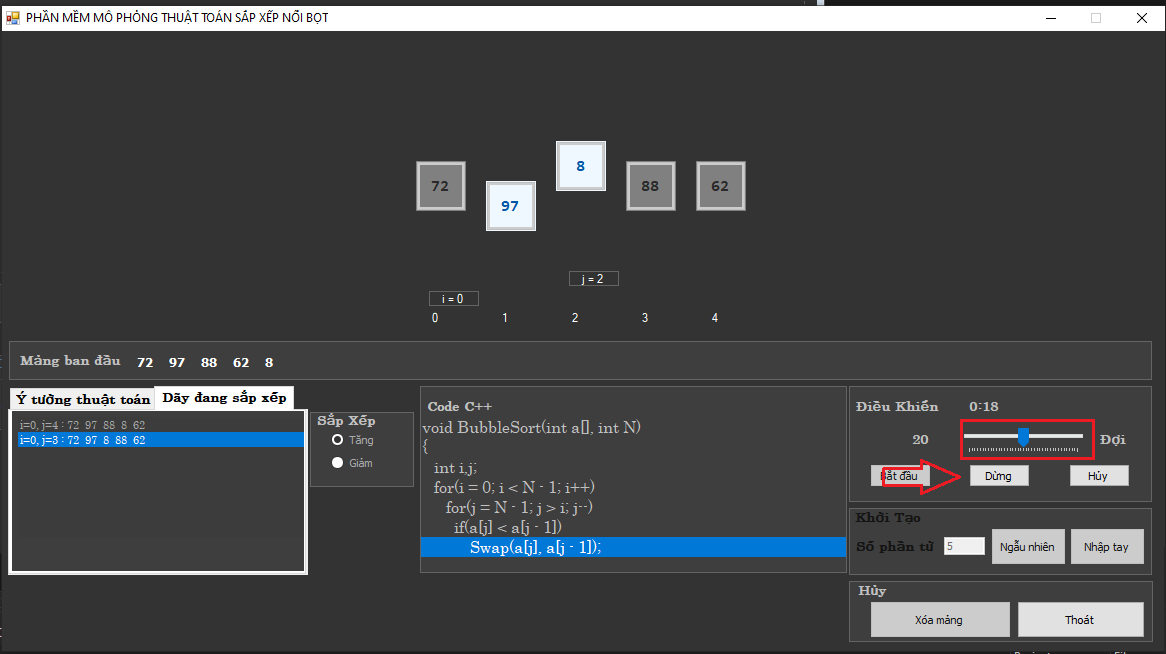
Hình 3.5: Chương trình cho phép chọn tăng dần hoặc giảm dần

* Sau khi input đầu vào và chọn kiểu sắp xếp, người dùng có thể nhấn bắt đầu quá trình mô phỏng



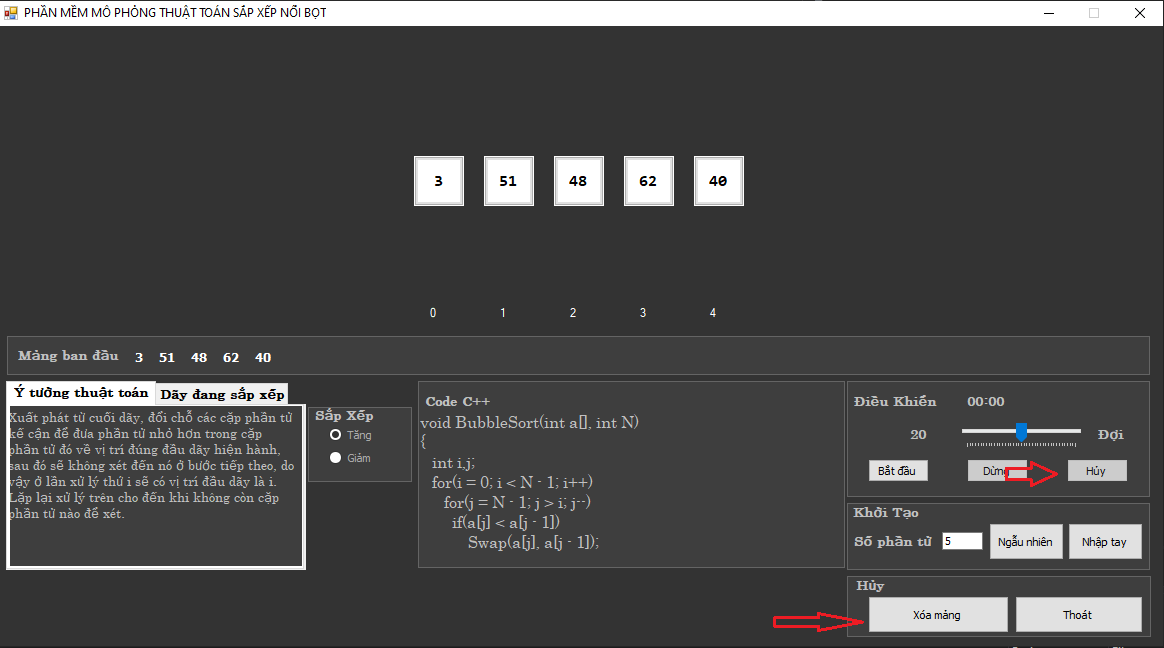
Hình 3.6: Chương trình bắt đầu thực hiện

* Khi chương trình đang hoạt động người dùng có thể bấm vào nút **Tạm Dừng** hoặc điều chỉnh thời gian Sleep trên thanh **Trackbar** để dễ dàng theo dõi quá trình sắp xếp hơn



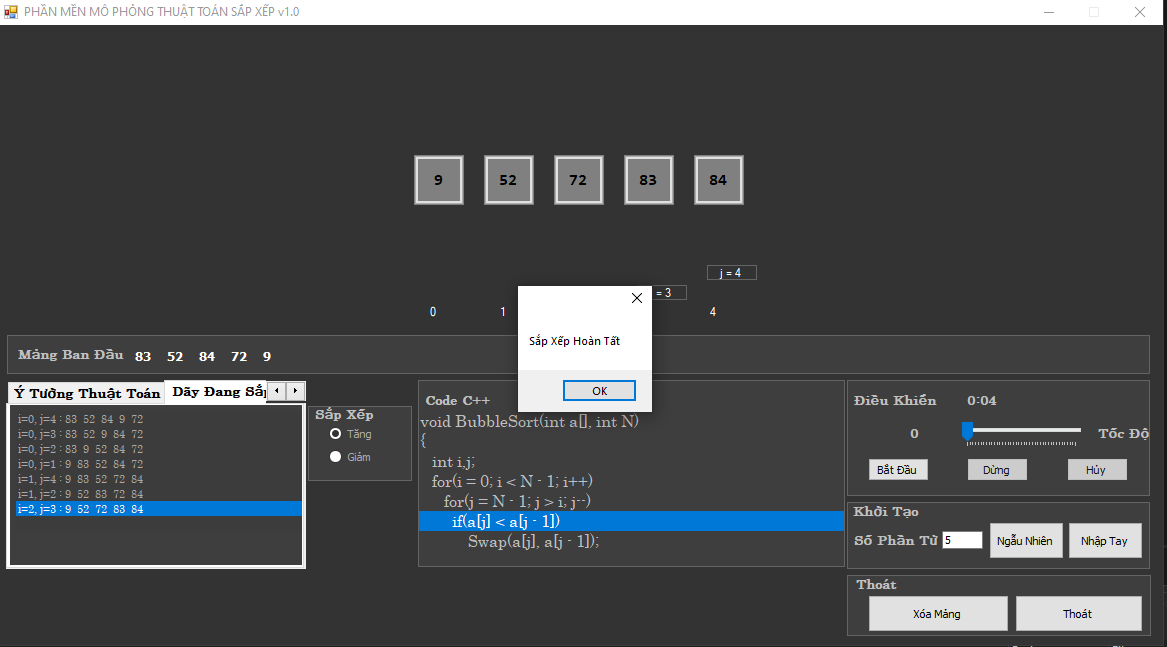
Hình 3.7: Chương trình có thể tăng giảm tốc độ di chuyển hoặc tạm dừng

* Chương trình còn cung cấp chức năng cho phép người dùng hủy quá trình sắp xếp chương trình sẽ tạo lại các Node với Input ban đầu, hoặc người dùng có thể chọn chức năng xóa mảng để xóa hết input ban đầu và bắt đầu



Hình 3.7: Chương trình có sự kiện xóa mảng

* Khi chương trình sắp xếp xong sẽ hiện một bảng thông báo rằng đã kết thúc quá trình sắp xếp



Hình 3.8: Hiển thị thông báo hoàn tất việc sắp xếp

CHƯƠNG 4 Cài đặt

4.1 Nền tảng công nghệ

* Phần mềm sử dụng Visual Studio 2019
* Ngôn Ngữ: C# Window Form Application
* Các thư viện sử dụng: không có

4.2 Các hàm xử lý chính

4.2.1 Phần tử và các hàm có trong phần tử

* Đối tượng thao tác chính trong phần mềm là Node. Node là một UserControl gồm một button chứa các phần tử: giá trị (values), vị trí hiện tại,… và một TextBox để người dùng nhập giá trị cho Node mỗi khi double-click

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);

tbx1.LostFocus += Tbx1\_LostFocus;

}

* Để xử lý việc nhập giá trị tay bằng Node ta cần một số sự kiện như sau:
* Sự kiện Tbx1\_LostFocus: không cho người dùng nhập giá trị rỗng, khi TextBox nhập tay bị LostFocus thì tắt TextBox đi và lưu giá trị của TextBox vào trong Node

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);

}

* Sự kiện btn\_MouseDown bắt sự kiện khi người dùng double-click cho TextBox để cập nhật giá trị cho node

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

{

if (e.Clicks == 2)

{

btn1.Visible = false;

tbx1.Text = btn1.Text;

tbx1.SelectionStart = tbx1.Text.Length;

tbx1.BringToFront();

tbx1.Focus();

}

}

* Sự kiện txt1\_KeyDown : nếu nhấn Enter thì tắt TextBox nhập tay và lưu giá trị vào Node.

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

{

if (e.KeyCode == Keys.Enter)

{

btn1.Visible = true;

btn1.BringToFront();

btn1.Focus();

}

}

* Sự kiện txt1\_KeyPress : chỉ cho phép nhập các phím là số khi nhập giá trị vào TextBox.

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

{

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

{

e.Handled = true;

}

}

* Sự kiện tbx1\_Enter : khi người dùng bấm nút Enter nếu giá trị đó khác rỗng thì sẽ lưu giá trị đó lên Node

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

{

if (tbx1.Text!=null)

{

btn1.Text = tbx1.Text;

}

}

* Để xử lý di chuyển các Node lên, xuống, trái, phải ta cần phải có những hàm như sau:
* Hàm ChuyenXuong : chuyển Node đi xuống

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);

}

}

* Hàm ChuyenLen : chuyển Node đi lên

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);

}

}

* Hàm ChuyenNgang: chuyển Node đi qua trái hoặc qua phải tùy thuộc vào vị trí hiện tại của Node và vị trí mới mà Node chuyển đến

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);

}

}

}

* + 1. Các hàm tạo mảng
* Để tạo một mảng các phần tử ta cần tạo ra một mảng các Node và lưu các Node lại bên trong danh sách liên kết đơn để thuận tiện cho việc điều khiển
* Một số các thuộc tính cần để tạo một mảng:
* isRunning: cho biết quá trình sắp xếp có đang chạy hay không
* DanhSachNode: chứa các Node
* labelSTTArr: chứ dãy số thứ tự ở dưới Node
* bienArr: chứa các biến có trong thuật toán
* MangChuaSapXep: Chứa dãy số ban đầu

private bool isRunning;

public List<Node> DanhSachNode;

private List<Label> labelSTTArr = new List<Label>();

private Dictionary<string, Label> bienArr;

public List<int> MangChuaSapXep;

* Các hàm tạo mảng:
* Hàm VeNut: vẽ một Node

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 làviệc

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.White;

//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;

}

* Hàm DieuChinhControl: ẩn các Button không cần thiết khi chương trình đang thực thi

private void DieuChinhControls(bool isRunning)

{

batdaubtn.Focus();

if (isRunning == true)

{

foreach (Node item in DanhSachNode)

{

item.Enabled = false;

}

batdaubtn.Enabled = false;

xoamangbtn.Enabled = false;

dungbtn.Enabled =huybnt.Enabled=true;

daydangxepListbox.Items.Clear();

daydangxepListbox.BringToFront();

tabctlytuong.SelectedIndex = 1;

khoitaopanel.Enabled = false;

}

else

{

foreach (Node item in DanhSachNode)

{

item.Enabled = true;

}

xoamangbtn.Enabled = true;

dungbtn.Enabled = huybnt.Enabled = false;

khoitaopanel.Enabled = batdaubtn.Enabled = true;

}

}

* Hàm Mangchuasapxep: Cập nhật giá trị của mảng được tạo để sắp xếp

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.White;

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

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

chuasapxepPanel.Controls.Add(lbtext);

}

4.2.3 Tạo mảng bằng tay

* **Form Nhập mảng gồm:**
* TextBox (nhập dữ liệu , hiển thị node tối đa)
* Button (nhập , hủy)
* **Các sự kiện chính trong Form tạo mảng**
* Sự kiện btnnhap\_Click là khi click vào button nhập, chương trình sẽ lấy dữ liệu từ TextBox cách nhau khoảng trắng

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

{

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();

}

* Sự kiện tbxDaySo\_KeyPress : chỉ cho người dùng nhập số nhưng vẫn cho người dùng nhập dấu khoảng trắng

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

{

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

{

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

}

}

4.2.4 Các hàm bắt đầu, hủy quá trình sắp xếp và các hàm hỗ trợ

* Hàm để bắt đầu quá trình sắp xếp
* Hàm KhoiTaoMacDinh: khởi tạo giá trị mặc định khi mở chương trình

private void KhoiTaoMacDinh()

{

bienArr = new Dictionary<string, Label>();

List<string> bienArrString = new List<string>() { "i", "j"};

foreach (string item in bienArrString)

{

bienArr.Add(item, new Label());

}

foreach (var item in bienArr)

{

this.sapxepPanel.Controls.Add(item.Value);

item.Value.TextAlign = ContentAlignment.MiddleCenter;

}

bienArr["i"].Size = bienArr["j"].Size = new Size(ThamSo.KichCoNode, 15);

bienArr["i"].ForeColor = bienArr["j"].ForeColor = Color.White;

bienArr["i"].BorderStyle = bienArr["j"].BorderStyle = BorderStyle.FixedSingle;

tangrdbtn.Checked = true;

isRunning = false;

soluongNodetbx.Text = "5";

SoLuongNode = 5;

dungbtn.Enabled = huybnt.Enabled = false;

VeNut();

DieuChinhControls(isRunning);

}

* Sự kiện batdaubtn\_Click: cho phép chương trình thực thi khi Click vào button

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();

ThuatToanSapXep = BubbleSort;

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

sapxepThread.Start();

}

* Sự kiện huybtn\_Click: hủy bỏ quá trình sắp xếp và tạo ra một mảng Node mới, cập nhật giá trị chưa sắp xếp

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;

}

* Hàm Reset\_CountTime: Reset lại bộ đếm giờ của chương trình

private void Reset\_CountTime()

{

Phut = Giay = 0;

label11.Text = "00:00";

}

4.2.5 Các hàm di chuyển vị trí của Node và hoán vị hai Node

* Hàm DiChuyenNodeDen: Di chuyển Node đến vị trí đã được chỉ định

public void DiChuyenNodeDen(object oNode, object oVitriMoi)

{

int vitriMoi = (int)oVitriMoi;

Node node = oNode as Node;

if (vitriMoi > node.vitriHienTai)

{

node.ChuyenLen();

node.ChuyenNgang(vitriMoi);

node.ChuyenXuong();

}

else if (vitriMoi < node.vitriHienTai)

{

node.ChuyenXuong();

node.ChuyenNgang(vitriMoi);

node.ChuyenLen();

}

node.vitriHienTai = vitriMoi;

}

* Hàm DiChuyenCacNode: hàm này giúp hoán đổi hai Node trong mảng. Hàm tạo ra hai Task với mỗi Task di chuyển một Node đến vị trí của Node còn lại. Khi thực hiện nó sẽ đổi màu hai Node và đổi vị trí của hai Node bên trong NodeArr

Task hoanVi1Task;

Task hoanVi2Task;

private void DichuyenCacNode(int vitriNodeA, int vitriNodeB)

{

DanhSachNode[vitriNodeA].BackColor = DanhSachNode[vitriNodeB].BackColor = ThamSo.mauNodeDangSX;

// Cách dùng task

hoanVi1Task = Task.Factory.StartNew(() => { DiChuyenNodeDen(DanhSachNode[vitriNodeA], vitriNodeB); });

hoanVi2Task = Task.Factory.StartNew(() => { DiChuyenNodeDen(DanhSachNode[vitriNodeB], vitriNodeA); });

// note: Task.Factory.StartNew = THÊM task vào cuối hàng đợi và CHẠY ngay khi có thể

Task.WaitAll(hoanVi1Task, hoanVi1Task);

// Thay đổi vị trí của node trong mảng nodeArray

if (DanhSachNode.Count != 0) //check xem nếu mảng còn tồn tại --> trong trường hợp mảng đã bị hủy

{

CapNhatDanhSachNode(vitriNodeA, vitriNodeB);

}

// Đổi màu 2 node sau khi sắp xếp

DanhSachNode[vitriNodeA].BackColor = DanhSachNode[vitriNodeB].BackColor = ThamSo.mauNodeHTSX;

}

4.2.6 Sự kiện tạm dừng quá trình sắp xếp

* Sự Kiện dungbtn\_Click : Sự kiện khi click vào sẽ các Node đang di chuyển sẽ đứng im. Sự kiện này điều khiển ManualResetEvent của cả Node và ManualResetEvent có trong hàm chọn từng dòng code

public static bool CodeListBoxIsPause = false;

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

{

if (Node.IsPause)

{

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

Node.IsPause = false;

dungbtn.Text = "Tạm dừng";

timer1.Start();

}

else

{

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

Node.IsPause = true;

dungbtn.Text = "Tiếp tục";

timer1.Stop();

}

}

* + 1. Hàm sắp xếp mảng nổi bọt
* Hàm BubbleSort

private void BubbleSort(){

bool flag = false;

HienThiThuatToan.ChayCodeC(2);

HienThiThuatToan.ChayCodeC(3);

int i = 0, j = 0;

DanhSachNode[i].BackColor = ThamSo.mauNodeHTSX;

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;

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

{

DanhSachNode[j].BackColor = ThamSo.Nodeketiep;

flag = false;

if (tangrdbtn.Checked == true)

{

HienThiThuatToan.ChayCodeC(5);

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

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 (giamrdbtn.Checked == true)

{

HienThiThuatToan.ChayCodeC(5);

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

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 (flag == true)

{

HienThiThuatToan.ChayCodeC(6);

Thread.Sleep(ThamSo.ThoiGianDoi);

DanhSachNode[j].BackColor = DanhSachNode[j - 1].BackColor = ThamSo.mauNodeDangSX;

CapNhatThamSo(j, j - 1);

DichuyenCacNode(j, j - 1);

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

}

else

{

DanhSachNode[j].BackColor = ThamSo.mauNen;

}

}

}

DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;

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

hoanTatSapXep();

}

* Hàm CapNhatThamSo: bắt đầu Hoán vị 2 giá trị sẽ cập nhật lại tham số trong DanhSachThamSo

private void CapNhatThamSo(int vt1, int vt2)

{

int temp = DanhSachThamSo[vt1];

DanhSachThamSo[vt1] = DanhSachThamSo[vt2];

DanhSachThamSo[vt2] = temp;

DanhSachNode[vt1].BackColor = DanhSachNode[vt2].BackColor = ThamSo.mauNodeDangSX;

}

* Hàm CapNhatDanhSachNode: khi Hoán vị 2 giá trị sẽ cập nhật lại Node trong DanhsachNode

private void CapNhatDanhSachNode(int vt1, int vt2)

{

Node temp = DanhSachNode[vt1];

DanhSachNode[vt1] = DanhSachNode[vt2];

DanhSachNode[vt2] = temp;

DanhSachNode[vt1].BackColor = DanhSachNode[vt2].BackColor = ThamSo.mauNodeDangSX;

}

* Hàm HoanTatSapXep: Xuất ra thông báo khi hoàn tất sắp xếp, Enable các button trên giao diện chương trình bị vô hiệu hóa khi chương trình sắp xếp

private void hoanTatSapXep(){

foreach (Node item in DanhSachNode)

{

item.BackColor = ThamSo.mauNodeHTSX;

}

isRunning = false;

DieuChinhControls(isRunning);

MessageBox.Show(hoanTat);

timer1.Stop();

Reset\_CountTime();

foreach (Label label in bienArr.Values)

{

label.Visible = false;

}

huybnt.Enabled = false;

}

CHƯƠNG 5 Kết Luận và đánh giá bản thân

1. **Kết quả đạt được**

* Hoàn thành quá trình mô phỏng thuật toán sắp xếp nổi bọt
* Được tìm hiểu thêm các kỹ thuật lập trình trong C# Winform như : timer, multithread, UserControl

1. **Nhược điểm**

* Giao diện chương trình chưa bắt mắt
* Số lượng phần tử trong chương trình còn hạn chế
* Xử lý code còn rườm rà

Tài Liệu Kham Khảo

* + - 1. <https://plctech.com.vn/winform-la-gi/>
      2. <https://www.howkteam.vn/course/lap-trinh-winform-co-ban/timer-trong-lap-trinh-c-winform-1305>
      3. <https://msdn.microsoft.com/enus/library/system.timers.timer(v=vs.110).aspx>
      4. <https://docs.microsoft.com/enus/dotnet/api/system.threading.manualresetevent?view=net-6.0>
      5. <https://docs.microsoft.com/enus/dotnet/api/system.threading.manualresetevent?view=net-6.0>
      6. <https://msdn.microsoft.com/en-us/library/mt472912(v=vs.110).aspx>
      7. https://viblo.asia/p/lap-trinh-bat-dong-bo-trong-c-DZrGNDoWkVB
      8. <https://github.com/thinhdoanvu/ThuctapCoSo2020>
      9. <https://github.com/NguyenHuynhSang/MoPhongThuatToanSapXep/tree/master/DoAnSapXep?fbclid=IwAR3RJ-SzZa34vW-6ojQfcZkB0q-P9hvCpWs_snLHcyXO4Gy8X-mVF8Gaz6k>
      10. https://www.youtube.com/watch?v=WBNa3fj04Bg&list=PL\_Ks8fYUz-ApWY-\_klKTrI0xQn8SVXQpH&index=38