**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN LẬP TRÌNH TRỰC QUAN**

**ĐỀ TÀI**

**MÔ PHỎNG CÁC THUẬT TOÁN SẮP XẾP**

**Giảng viên hướng dẫn: Huỳnh Tuấn Anh**

**Nhóm thực hiện : Sky**

**Sinh viên 1 :** Nguyễn Huỳnh Sáng - 16521022

**Sinh viên 2 :** Bùi Đang Quy - 16521009

**Sinh viên 3 :** Nguyễn Thị Tuyết Nhung - 16520899

*TP. Hồ Chí Minh, tháng 06 năm 2018*

# Mục lục

[1. GIỚI THIỆU 3](#_Toc517014897)

[1.1 Mô phỏng các thuật toán sắp xếp tiêu biểu trong lập trình 3](#_Toc517014898)

[1.2 Các chức năng 3](#_Toc517014899)

[2. KỸ THUẬT XỬ LÝ 4](#_Toc517014900)

[2.1 Phân tích 4](#_Toc517014901)

[2.1.1 Phân tích hiệu ứng 4](#_Toc517014902)

[2.1.2 Phân tích chức năng 5](#_Toc517014903)

[2.2 Kỹ thuật sử dụng 5](#_Toc517014904)

[2.2.1 Xử lý bất đồng bộ với Thread và Task 5](#_Toc517014905)

[2.2.2 Xử lý việc thay đổi tốc độ sắp xếp 6](#_Toc517014906)

[2.2.3 Xử lý việc di chuyển di chuyển các phần tử và các biến 7](#_Toc517014907)

[2.2.4 Xử lý việc tính thời gian thực hiện quá trình sắp xếp 7](#_Toc517014908)

[2.2.5 Xử lý việc lưu thông tin cài đặt của người dùng 7](#_Toc517014909)

[3.MINH HỌA SỬ DỤNG PHẦN MỀM 7](#_Toc517014910)

[4. CÀI ĐẶT 17](#_Toc517014911)

[4.1 Nền tảng công nghệ 17](#_Toc517014912)

[4.2 Các hàm xử lí chính 17](#_Toc517014913)

[4.2.1 Phần tử và các hàm có trong phần tử 17](#_Toc517014914)

[4.2.2 Các hàm tạo mảng 20](#_Toc517014915)

[4.2.3 Các hàm bắt đầu và hủy quá trình sắp xếp và các hàm hỗ trợ 23](#_Toc517014916)

[4.2.4 Các hàm di chuyển vị trí của Node và hoán vị hai Node 24](#_Toc517014917)

[4.2.5 Hàm tạm dừng quá trình sắp xếp 25](#_Toc517014918)

[4.2.6 Hàm chọn từng dòng code trong khung Code C/C++ 26](#_Toc517014919)

[4.2.7 Cập nhật trạng thái ý tưởng thuật toán, dãy chưa sắp xếp 26](#_Toc517014920)

[4.2.8 Các hàm sắp xếp mảng 27](#_Toc517014921)

[5. KẾT LUẬN 57](#_Toc517014922)

[5.1 Kết quả đạt được 57](#_Toc517014923)

[5.2 Hạn chế 57](#_Toc517014924)

5.3 Hướng phát triển 57

[5.4 Phân rã công việc 57](#_Toc517014925)

[6. TÀI LIỆU THAM KHẢO 59](#_Toc517014926)

# GIỚI THIỆU

## **Mô phỏng các thuật toán sắp xếp tiêu biểu trong lập trình**

Theo cuốn sách óc sáng suốt của Thu Giang – Nguyễn Duy Cần có đoạn:

“Đây là một anh thợ máy cần dùng nhiều cỡ ốc, và mỗi con ốc phải dùng riêng một cây vặn ốc cùng một cỡ mới được. Anh đổ lộn đống cây vặn ốc trong một cái hộp. Mỗi khi anh cần dùng một cỡ nào, anh phải soạn kiếm hồi lâu mới thấy. Như thế, anh đã làm mất thời giờ và công việc, làm tốn công một cách vô ích. Có người mách cho anh làm một tấm bảng cây đem những cây vặn ốc phân sắp theo cỡ, máng vô đó và biên số ni cho dễ thấy. Bấy giờ, anh làm việc rất mau lẹ, không sai chạy vì hễ dùng đến cây nào anh thấy liền nó trước mắt.

Anh phát thư hay phát báo cũng phải biết phân sắp thư từ báo chí theo con đường và theo thứ tự trước sau của số nhà, để một khi đi qua rồi, không cần phải trở lại nữa.

Người bán sách cũng phải biết sắp sách vỡ báo chí theo từng loại một, như báo xuất bản ở Bắc hay Nam, báo thuộc về chính trị, văn học hay khoa học… mỗi cái đều sắp theo loại của nó…

Bất kì là đứng vào địa vị, nghề nghiệp nào, hễ biết sắp đặt, biết phân loại là giúp cho công việc làm của mình được tiện lợi, đầy đủ.”

Khi người xưa đã xem trọng việc sắp xếp như vậy, mà với lượng thông tin khổng lồ như ngày nay, việc tổ chức sắp xếp và lưu trữ thông tin càng quan trọng biết bao nhiêu.

Hiểu được điều đó, nhóm đã tạo ra một sản phẩm mô phỏng các thuật toán sắp xếp hết sức công phu. Sản phẩm mô tả các bước hoạt động một cách cụ thể và dễ hiểu không mất quá nhiều thời gian để đọc code. Có hình ảnh, và màu sắc bắt mắt, làm thu hút người dùng làm cho việc học môn Lập trình trực quan trở nên thú vị hơn.

## **Các chức năng**

1. Bao gồm 10 thuật toán sắp xếp tiêu biểu nhất trong cấu trúc dữ liệu và giải thuật.
2. Tạo mới ngẫu nhiên một dãy số các phần tử theo từng thuật toán sắp xếp tương ứng.
3. Xóa mảng: Khi thực hiện xóa mảng tất cả các phần tử trên màn hình chính sẽ không còn xuất hiện nữa.
4. Hủy quá trình: Khi thực hiện hủy quá trình sắp xếp các phần tử trên màn hình chính sẽ trở về trạng thái ban đầu chưa sắp xếp.
5. Giới hạn số phần tử được phép thêm vào dãy số(tối đa là 15 phần tử).
6. Hỗ trợ chức năng tạm dừng hay tiếp tục chạy ứng dụng.
7. Điều chỉnh tốc độ thực thi các hoạt động của các thuật toán sắp xếp trong lúc chạy.
8. Có cửa sổ hiện thị code C++ của các thuật toán sắp xếp tương ứng nhằm đối chiếu với hình ảnh mô phỏng.
9. Hiển thị ý tưởng thuật toán và hiện trạng dãy số thay đổi khi sắp xếp.
10. Khởi tạo dãy số cần sắp xếp bằng cách click vào phần khởi tạo rồi chọn chế độ:
11. Tạo ngẫu nhiên với số phần tử ngẫu nhiên.
12. Nhập bằng tay: nhập từng phần tử với số phần tử tương ứng khi nhập vào ô số phần tử.
13. Lựa chọn chế độ debug cho phép chạy từng dòng lệnh theo mã code C++ trên Form chính.
14. Mảng chưa sắp xếp: Hiển thị mảng trước khi sắp xếp.
15. Hiển thị quá trình sắp xếp và kết quả sau khi sắp xếp xong của dãy số.
16. Hiển thị thời gian sắp xếp
17. Cho đổi tông màu hiển thị của phần mềm
18. Lưu các thông tin cài đặt của người dùng

# 2. 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.

### **2.1.2 Phân tích chức năng**

1. Khởi tạo mảng (3 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.
   3. Nhập một dãy : cho người dùng nhập một dãy số liên tiếp và tạo mảng các phần tử dựa vào dãy số vừa nhập. Người dùng có thể lưu lại dãy vừa nhập cho các lần nhập sau nếu muốn.
2. Bảng điều khiển:
   1. +Bắt đầu sắp xếp : bắt đầu sắp xếp các phần tử dựa vào thuật toán được chọn
   2. Tạm dừng : cho tạm dừng quá trình sắp xếp
   3. Thời gian thực hiện : cho xem tổng thời gian thực hiện quá trình sắp xếp
3. **Các chức năng phụ:** 
   1. Thay đổi tốc độ quá trình sắp xếp
   2. Đa ngôn ngữ hổ trợ Anh-Việt
   3. Cập nhật thông tin cho quá trình sắp xếp
   4. Cho phép chạy thuật toán ở hai hướng là tăng và giảm
   5. Thông qua mục cài đặt ta có thể thay đổi các thuộc tính của phần mềm như : font chữ của các khung hiển thị, màu, kích cỡ của các phần tử, khoản cách giữa các phần tử

## **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 sử dụng cả hai.

Để có thể nhìn thấy các phần tử di chuyển mượt mà khi sắp xếp ta cần dùng đến hàm Thread.Sleep(<thời gian>) để cho tiến trình dừng lại trong một khoản thời gian nhất định và giúp cho người dùng kịp nhận ra 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) xảy ra.

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 tiến trình chính.

Khi sử dụng Thread ta có thể hủy Thread bất kì khi nào mình muốn nhưng Thread lại không hỗ trợ các thư việc 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 đang thực hiện, việc hủy Task phải thông qua một CancellationTokenSource và cần phải thực hiện nhiều việc mới có thể 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 việc chọn từng dòng code, bên trong Thread đó ta tạo ra hai Task con đê thực hiện việc di chuyển các phần tử. Vì vậy ta vừa có thể báo việc hoàn tất di chuyển của các phần tử và vừa có thể hủy quá trình bất cứ khi nào.

Khi xử lý đa tiến trình ta cũng gặp phải một vấn đề khá 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 ta lỗi này

Việc tạm dừng các tiến trình ta sẽ sử dụng đên ManualResetEvent đây là một lớp có thể dùng để tạm dừng (block) các tiến trình đang chạy và tiếp tục (resume) các tiến trình khi cần.

### **2.2.2 Xử lý việc thay đổi 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 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 việc 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.

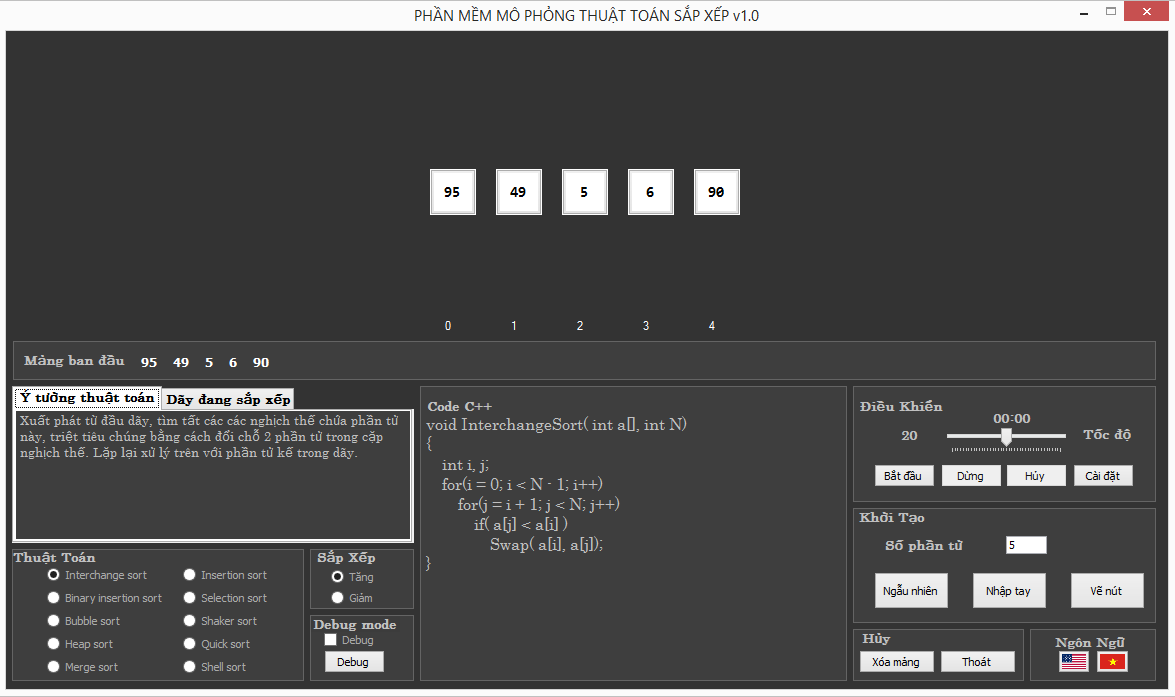
Ngoài ra các biến và các phần tử cũng được đổi màu để người sử dụng dễ dàng phân biệt.

### **2.2.5 Xử lý việc lưu thông tin cài đặt của người dùng**

Các thông tin mà người dùng chỉnh sửa trong phần cài đặt sẽ được lưu vào mục Cài Đặt của phần mềm bao gồm các thông tin như : màu phần tử đang xét, màu phần tử đã xét, màu nền của phần tử, số phần tử mặc định, kích cỡ phần tử, khoảng cách giữa các phần tử.

# 3.MINH HỌA SỬ DỤNG PHẦN MỀM

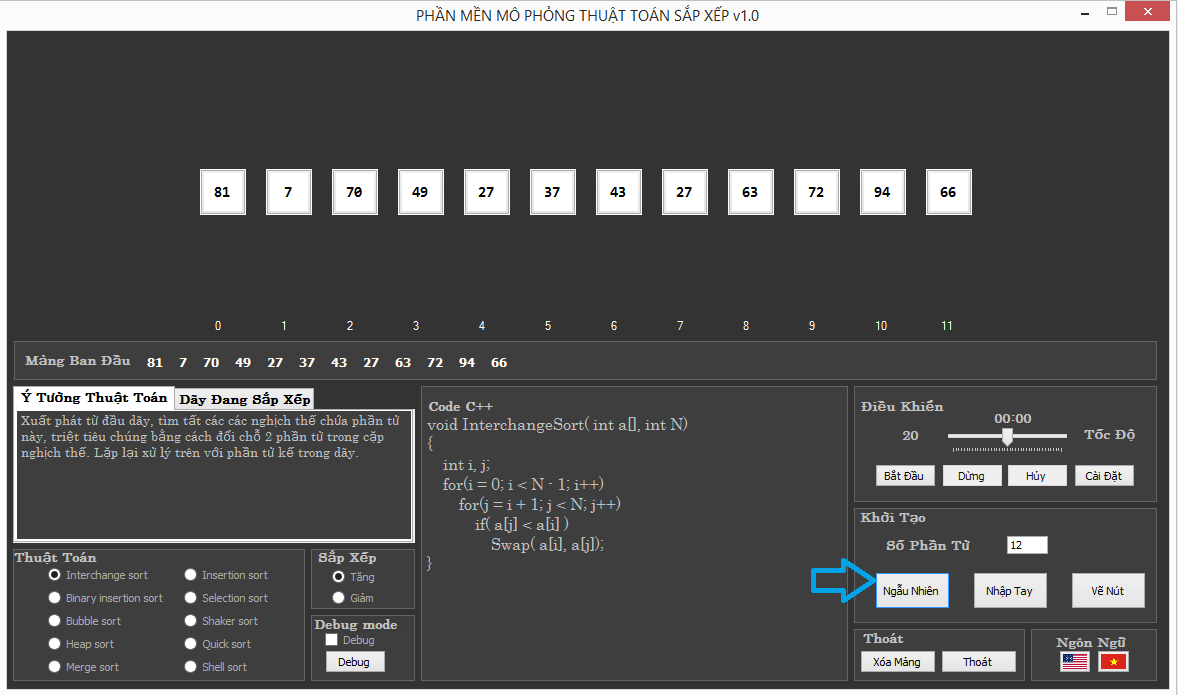
* Giao diện chương trình khi khởi động , chương trình sẽ tạo sẵn cho người dùng 5 Node để kiểm tra.



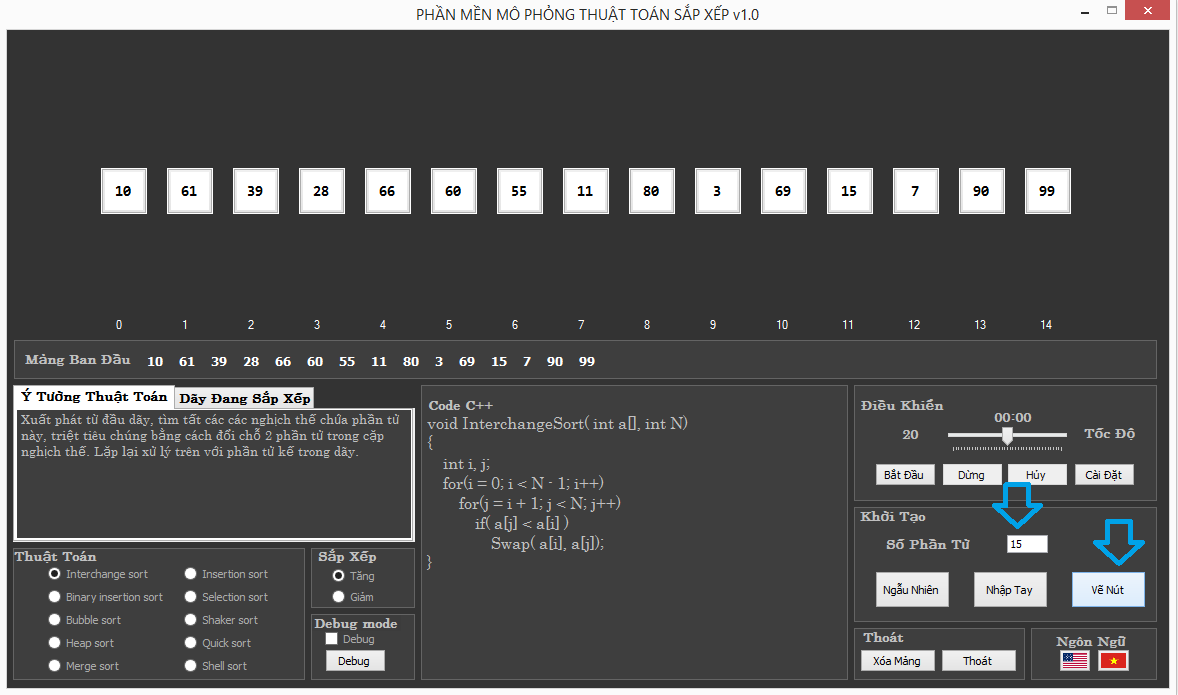
* Chương trình hỗ trợ hai ngôn ngữ là tiếng Việt và tiếng Anh người dùng có thể dễ dàng thay đổi ngôn ngữ bằng cách click chuột vào lá cờ tương ứng.



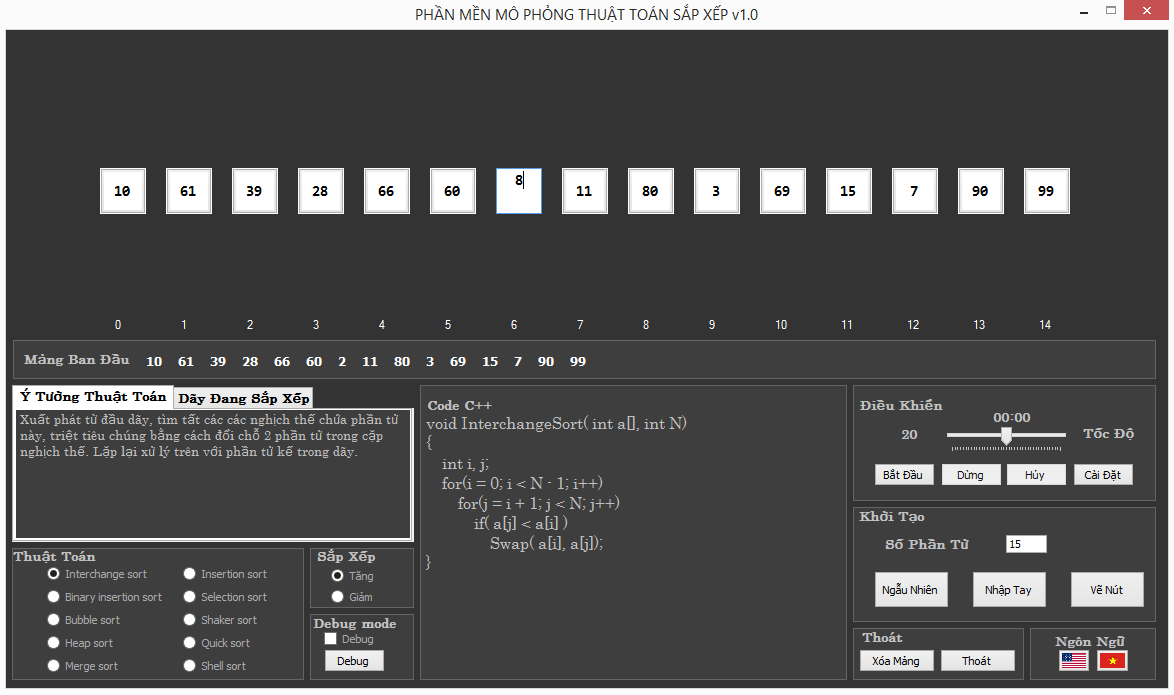
* Người dùng có thể khởi tạo input đầu vào bằng ba cách: Ngẫu nhiên, nhập tay , nhập số phần tử trong textbox và bấm vẽ nút:
  1. Khi bấm ngẫu nhiên chương trình sẽ tự động cấp phát một mãng ngẫu nhiên n phần tử trong khoản từ 2 đến số lượng node lớn nhấn do người dùng cài đặt, và miền giá trị của từng node vào khoảng từ 1 đến 99.



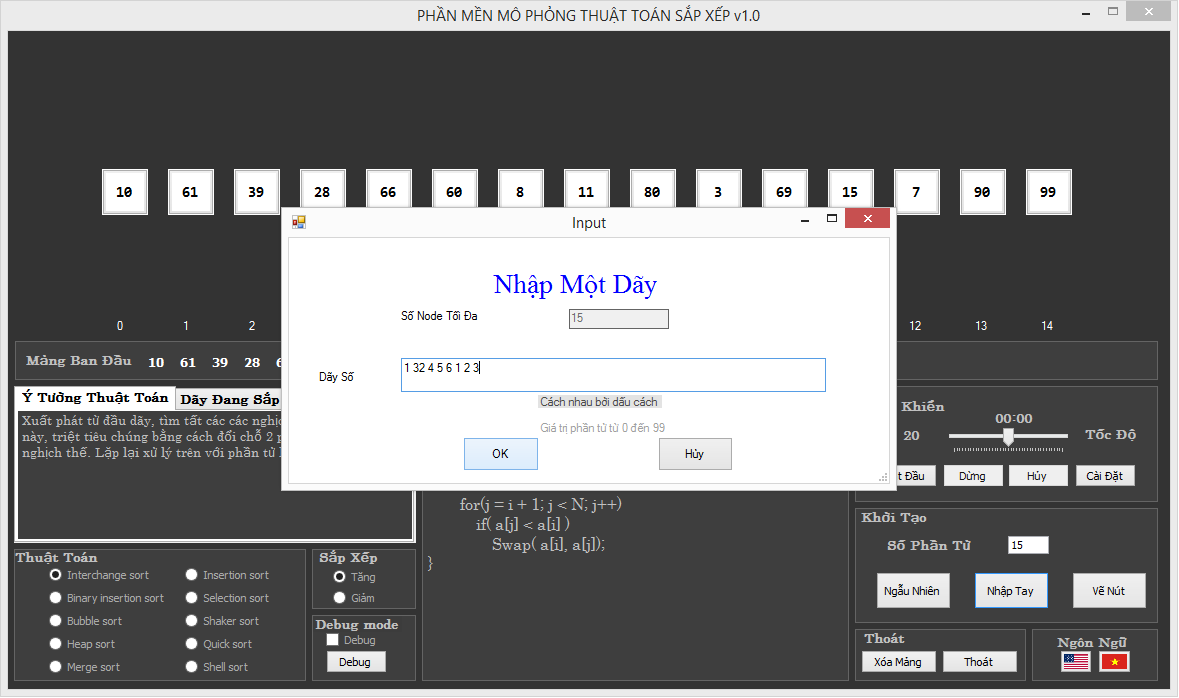
* 1. Chức năng nhập số phần tử và vẽ nút cho phép người dùng quy định số phần tử của mãng, chương trình sẽ hiện thị lên GUI số lượng Node và miền giá trị ngẫu nhiên từ 1 đến 99.



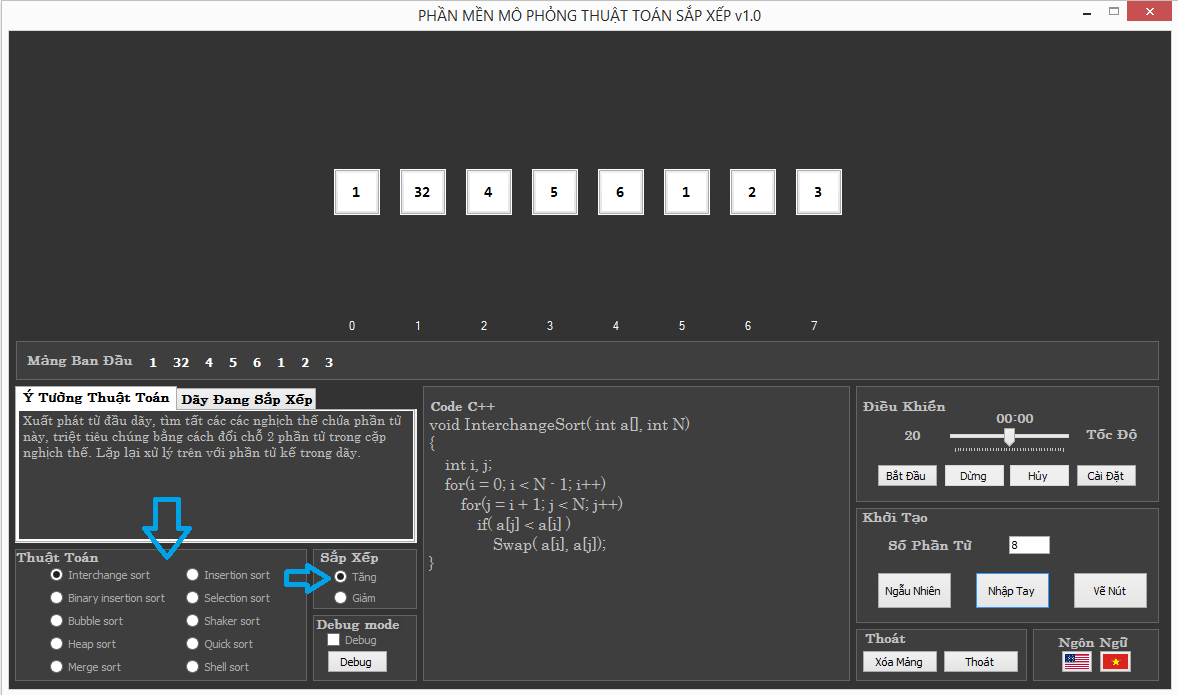
* 1. Người dùng cũng có thể thay đổi input đầu vào của mãng bằng cách double-click vào Node bất kỳ cần thay đổi giá trị sau đó nhấn Enter hoặc click lên panel chương trình sẽ tự động cập nhật lại giá trị của Node.



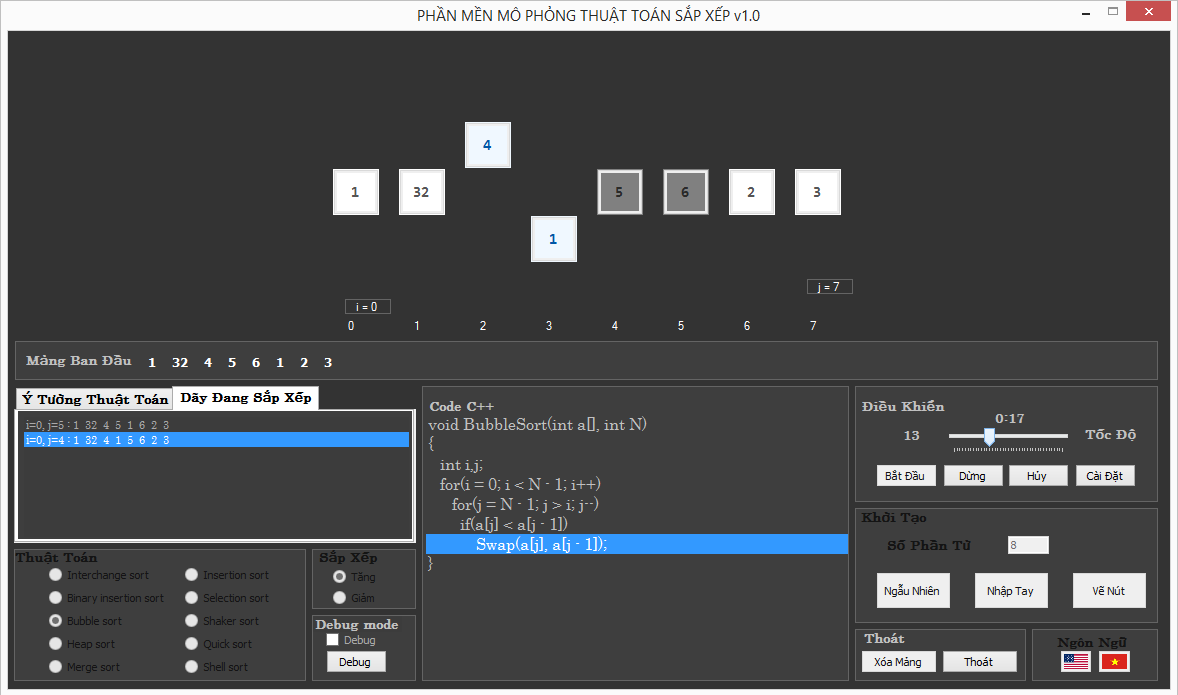
* 1. Ngoài ra người dùng có thể khởi tạo bằng chức năng nhập tay chương trình sẽ tạo một form con cho phép người dùng nhập giá trị mãng đầu vào.



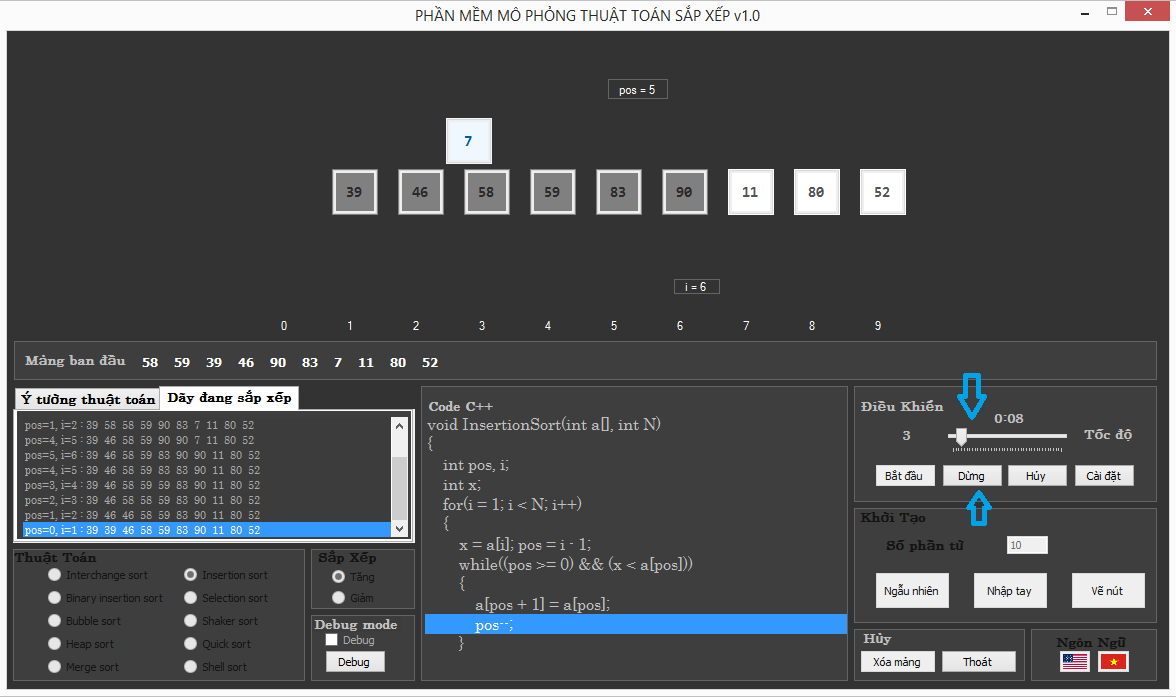
* 1. Chương trình có thể phỏng 10 thuật toán sắp xếp thông dụng người dùng có thể lựa chọn thuật toán cần dùng bằng cách check vào thuật toán tương ứng trong phần thuật toán , lựa chọn sắp xếp tăng giảm.



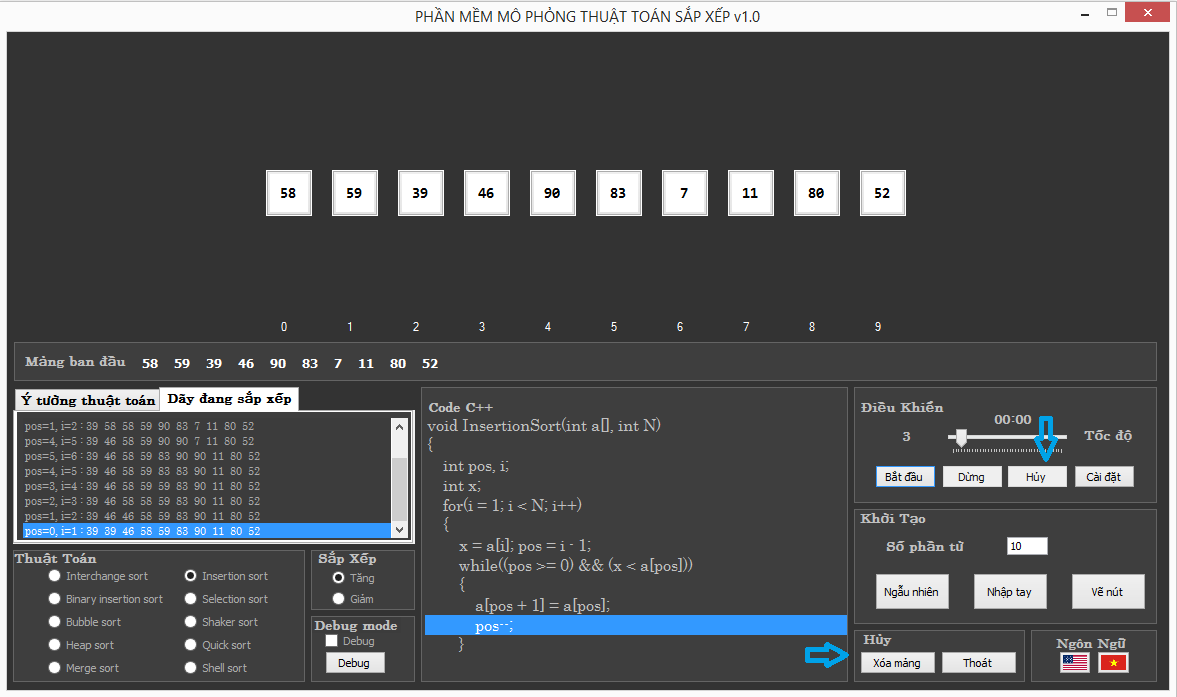
* 1. Sau khi đã có input đầu vào và chọn thuật toán tương ứng , người dùng có thể nhấn bắt đầu quá trình mô phỏng



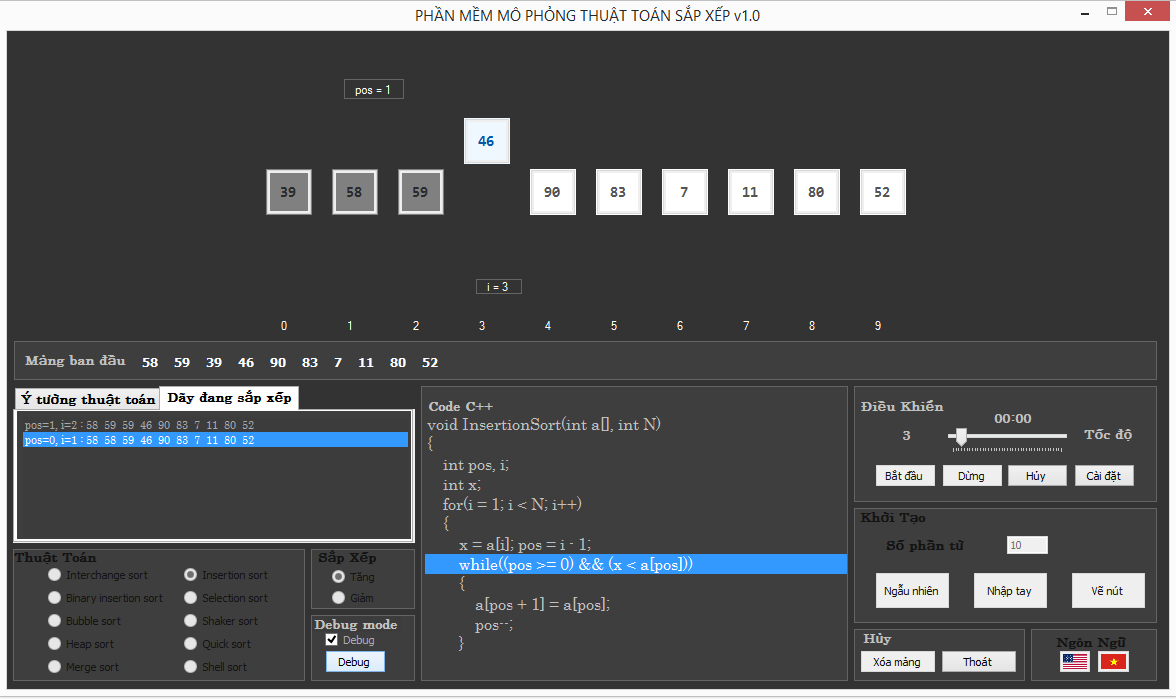
* 1. Trong quá trình thực hiện sắp xếp người dùng có thể điều chỉnh tốc độ của việc thực hiện chương trình trên thanh tốc độ, hoặc có thể bấm dừng để xem kỹ hơn phần thực hiện của thuật toán



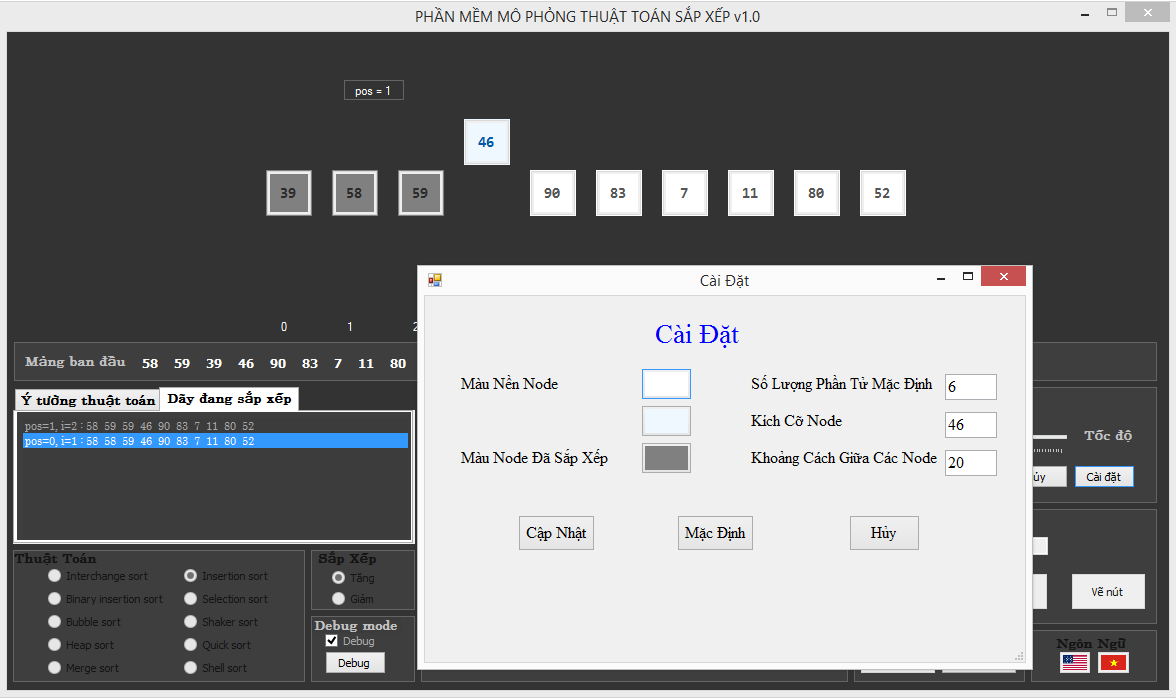
* 1. Chương trình còn cung cấp chức năng cho phép người dùng hủy quá trình sắp xếp chương trình sẽ tạo lại các Node với input ban đầu, hoặc người dùng có thể chọn chức năng xóa mãng để xóa hết input ban đầu và bắt đầu 1 phiên làm việc với dữ liệu đầu vào mới



* 1. Ở chế độ debug chương trình sẽ chạy từng bước của thuật toán để người dùng có thể dễ dàng nắm bắt



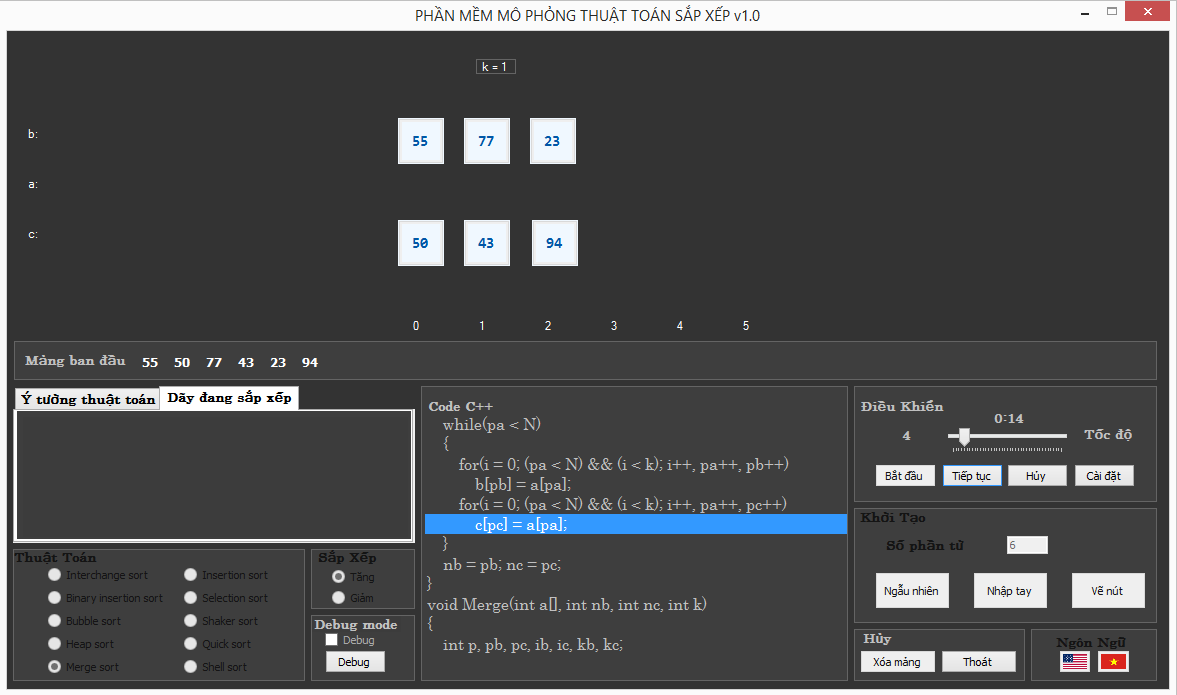
* 1. Phần Cài đặt của chương trình cho phép người dùng thay đổi màu nền Node, số lượng Node, Khoảng cách giữa các Node



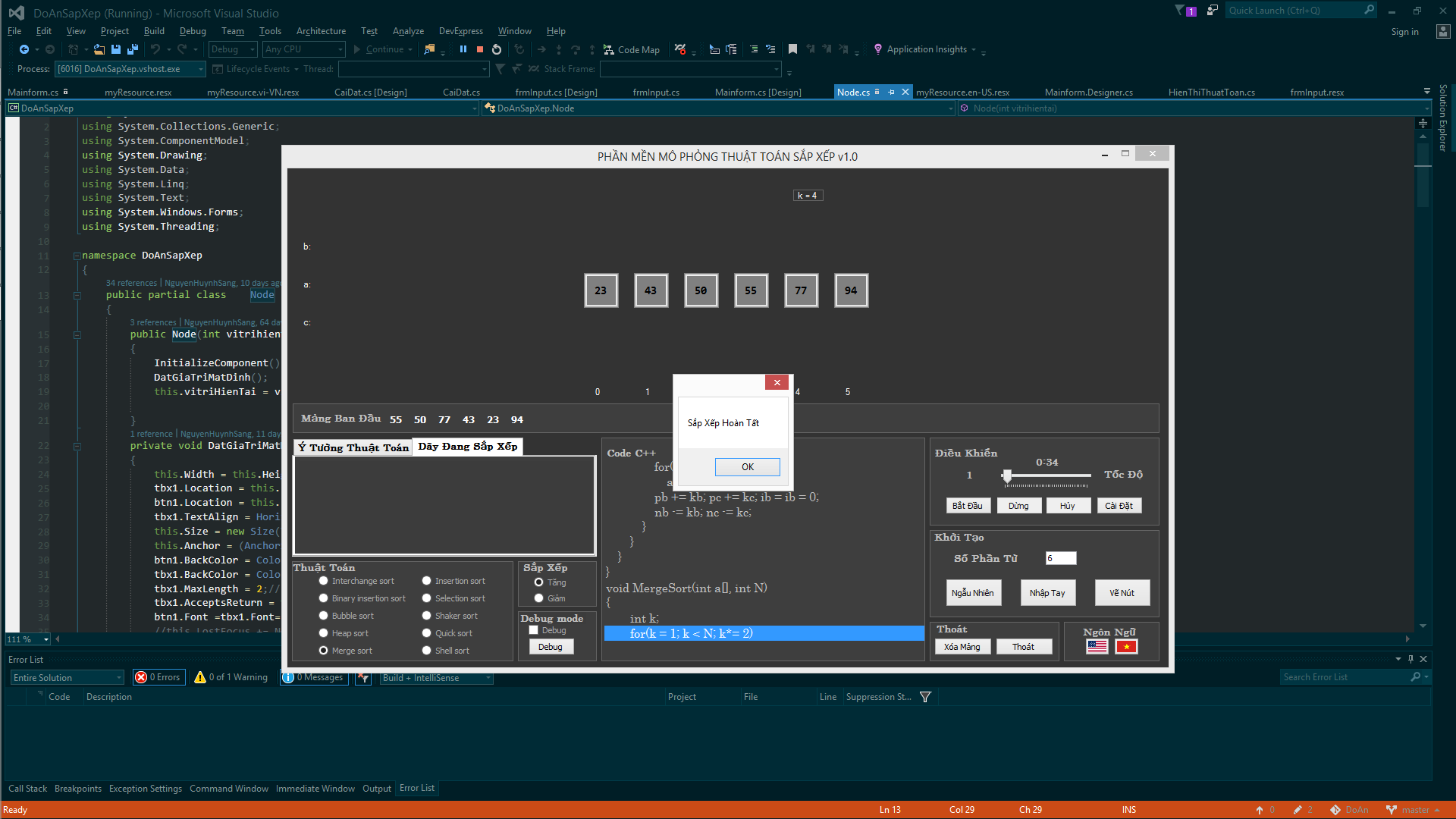
* Minh họa các bước thực hiện thuật toán sắp xếp

1. Nhập số lượng Node và Input đầu vào:.
2. Lựa chọn thuật toán sắp xếp
3. Chọn loại sắp xếp tăng hay giảm
4. Bắt đầu
5. Hoàn tất quá trình sắp xếp hiện thông báo cho người dùng

* Quá trình sắp xếp



* Hoàn tất sắp xếp



# CÀI ĐẶT

## **4.1 Nền tảng công nghệ**

Phần mềm sử dụng: Visual Studio 2017.

Ngôn ngữ: C# Window Form Application

Các thư viện sử dụng: Không có

## **4.2 Các hàm xử lí chính**

### **4.2.1 Phần tử và các hàm có trong phần tử**

1. Đối tượng thao tác chính trong phần mềm là Node. Node là một UserControl gồm một Button chứa các phần tử: value (giá trị), vị trí hiện tại,… và một TextBox để người dung có thể nhập giá trị cho Node mỗi khi double-clicked

|  |
| --- |
| class Node : UserControl  {  private int value;  public static bool IsPause = false;  public int vitriHienTai;  public event EventHandler Capnhat;  public Node(int vitrihientai)  {  this.Width = this.Height = btn1.Width = btn1.Height = tbx1.Width = tbx1.Height = ThamSo.KichCoNode;  tbx1.Location = this.Location;  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;  tbx1.AcceptsReturn = true;  btn1.Font = tbx1.Font = new Font("Consolas", ThamSo.KichCoNode / 4, FontStyle.Bold);  this.vitriHienTai = vitrihientai;  tbx1.LostFocus += Tbx1\_LostFocus;    }  } |

1. Để xử lý việc nhập giá trị bằng tay cho Node ta cần một số hàm chính sau:
   1. Hàm Tbx1\_LostFocus : Không cho người dùng nhập giá trị rỗng, khi TextBox nhập tay bị Lost-Focus 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);  }  } |

* Hàm btn1\_MouseDown : Bắt sự kiện 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;// dat con tro chuot "cusor" qua ben phai neu so da co  tbx1.BringToFront();  tbx1.Focus();    }  } |

* Hàm nhapTayTexbox\_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();  }  } |

* Hàm nhapTayTexbox\_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;  }  } |

- Để xử lý di chuyển các Node lên, xuống, qua trái, qua phải ta cần các hàm 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 ChuyenQua : chuyển Node đi qua trái hoặc qua phải tùy theo 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);  }  }  } |

### **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 một 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 mảng các Node

labelSTTArr: chứa dãy số thứ tự bên dưới mảng

bienArr: Chứa label các biến có trong thuật toan

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:

1. 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 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.Blue;

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 + 120);

}

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 + 120);

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;

}

1. Sự kiện venutbnt\_Click: Tạo ngẫu nhiên một mảng các Node và cặp nhật lại giá trị cho Mảng chưa sắp xếp.

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

{

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

{

if (isEnglish == true)

{

setLang("en-US");

}

else

setLang("vi-VN");

MessageBox.Show(strSoLuongNode + ThamSo.SoluongNodeLonNhat);

soluongNodetbx.Focus();

return;

}

VeNut();

Mangchuasapxep();

}

1. Hàm DieuChinhConTrol: Khi chương trình đang được thực thi thì không cho điều chỉnh các Button trên chương trình để tạo một phiên làm việc.

private void DieuChinhControls(bool isRunning)

{

if (isRunning == true)

{

daydangxepListbox.Items.Clear();

daydangxepListbox.BringToFront();

thuattoanpanel.Enabled = khoitaopanel.Enabled = ngonngupanel.Enabled = batdaubtn.Enabled = Loaisapxeppanel.Enabled = HienThiPanel.Enabled = false;

}

else

{

thuattoanpanel.Enabled = khoitaopanel.Enabled = ngonngupanel.Enabled = batdaubtn.Enabled = Loaisapxeppanel.Enabled = HienThiPanel.Enabled = true;

}

}

1. Hàm SetMauAllControl: Để set màu của các Controls

private void setMauAllControl()

{

this.BackColor = Color.White;

batdaubtn.BackColor = dungbtn.BackColor = huybnt.BackColor = taongaunhienbtn.BackColor

= venutbtn.BackColor = docfilebtn.BackColor = nhaptaybtn.BackColor = xoamangbtn.BackColor = thoatbtn.BackColor = Color.White;

DebugPanel.BackColor = thuattoanpanel.BackColor = Loaisapxeppanel.BackColor = panel10.BackColor = HienThiPanel.BackColor

= dieukhienpanel.BackColor = khoitaopanel.BackColor = huypanel.BackColor = ngonngupanel.BackColor = Color.White;

codeListBox.BackColor = chuasapxepPanel.BackColor =yTuongTextBox.BackColor= Color.White;

}

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

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

chuasapxepPanel.Controls.Add(lbtext);

}

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

|  |
| --- |
|  |

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

|  |
| --- |
| private void KhoiTaoMacDinh()  {  interchangerdbtn.Checked = true;  tangrdbtn.Checked = true;  isRunning = false;  soluongNodetbx.Text = "5";  SoLuongNode = 5;  VeNut();  DieuChinhControls(isRunning);  setMauAllControl();  } |

* + Sự kiện batdaubtn\_Click: Cho phép chương trình bắt đầu thực hiện một phiên làm việc.

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

{

daydangxepListbox.Visible = true;

huybnt.Enabled = true;

isRunning = true;

DieuChinhControls(isRunning);

Reset\_CountTime();

timer1.Start();

ChonThuatToan();

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

sapxepThread.Start();

}

* + Hàm ChonThuatToan : hàm thực hiện chọn thuật toán sắp xếp để bắt đầu một phiên làm việc.

private 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 (shakerrdbtn.Checked == true)

{

ThuatToanSapXep = ShakerSort;

}

if (shellrdbtn.Checked == true)

{

ThuatToanSapXep = ShellSort;

}

if (heaprdbtn.Checked == true)

{

ThuatToanSapXep = HeapSort;

}

if (quickrdbtn.Checked == true)

{

ThuatToanSapXep = QuickSort;

}

if (binaryinsertionrdbtn.Checked == true)

{

ThuatToanSapXep = BinaryInsertionSort;

}

if (mergerdbtn.Checked == true)

{

ThuatToanSapXep = MergeSort;

}

}

* Sự kiện huybnt\_Click: hủy bỏ quá trình đang sắp xếp và tạo lại một mảng Node mới, cập nhật giá trị mảng chưa sắp xếp, Reset\_CountTime.

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";

}

### **Các hàm di chuyển vị trí của Node và hoán vị hai Node**

* Hàm DiChuyenNodeDen : hàm này di chuyển một Node đến một vị trí 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();  }  // set lai vi tri moi  node.vitriHienTai = vitriMoi;  } |

* Hàm HoanVi2Node : hàm này giúp đổi chổ 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 xong 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;  } |

### **Hàm tạm dừng quá trình sắp xếp**

* Hàm TamDung : hàm này kiểm tra xem Node có đang di chuyển hay không nếu không thì tạm dừng Node lại còn có thì cho Node tiếp tục di chuyển. Hàm này điều khiển ManualResetEvent của cả Node và ManualResetEvent có trong hàm chọn từng dòng code.

|  |
| --- |
| public static ManualResetEvent codeListBoxPauseStatus = new ManualResetEvent(true);  public static bool CodeListBoxIsPause = false;  private void TamDung()  {  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();  }  } |

### **Hàm chọn từng dòng code trong khung Code C/C++**

* Hàm ChayCodeC : hàm nhận giá trị vị trí hàng được chọn trong khung Code C/C++ và chọn hàng đó.

|  |
| --- |
| public static void ChayCodeC(int line)  {  Thread.Sleep(ThamSo.ThoiGianDoi\*50);  codeListBox.SelectedIndex = line;  } |

### **Cập nhật trạng thái ý tưởng thuật toán**

- Khi bắt đầu quá trình sắp xếp, tùy thuộc vào phương pháp sắp xếp mà sẽ có những biến và số lượng biến khác nhau phù hợp cho phương pháp sắp xếp đó. Trong quá trình sắp xếp, những phần tử của dãy số sẽ đổi chỗ cho nhau tùy thuộc vào thuật toán mà người sử dụng chọn. Do đó ý tưởng thuật toán sẽ cập nhật liên tục trong quá trình sắp xếp.

* Hàm addYTuong : Dựa vào ngôn ngữ của giao diện để thêm vào yTuongThuatToan

|  |
| --- |
| private static void addYTuong()  {  yTuongThuatToan.Clear();  if (Mainform.isEnglish == true)  {  yTuongThuatToan.Text = yTuongEn;  }  else if (Mainform.isEnglish == false)  {  yTuongThuatToan.Text = yTuongVi;  }  } |

+ Hàm Mangchuasapxep :Cập nhật dãy số chưa sắp xếp tương ứng với dãy số ban đầu khi mới khởi tạo.

|  |
| --- |
| private void Mangchuasapxep()  {  chuasapxepPanel.Controls.Clear();  string \_string = "";  foreach (var item in DanhSachThamSo)  {  \_string += item + " ";  }  Label lbtext = new Label() { Text = \_string };  lbtext.Size = new Size(6000, 20);  lbtext.Location = new Point(30, 12);  chuasapxepPanel.Controls.Add(lbtext);  } |

### **Các hàm sắp xếp mảng**

* 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 đang sắp xếp.

private void hoanTatSapXep()

{

HienThiThuatToan.ChayCodeC(1);

foreach (Node item in DanhSachNode)

{

item.BackColor = ThamSo.mauNodeHTSX;

}

isRunning = false;

DieuChinhControls(isRunning);

if (DanhSachNode.Count !=0)

{

if (isEnglish == true)

{

setLang("en-US");

}

else

setLang("vi-VN");

MessageBox.Show(hoanTat, hoanTatMessageBoxName, MessageBoxButtons.OK, MessageBoxIcon.Information);

}

timer1.Stop();

sapxepThread.Abort();

foreach (Label label in bienArr.Values)

{

label.Visible = false;

}

if (!isRunning)

{

timer1.Stop();

}

huybnt.Enabled = false;

}

* Hàm InterchangeSort : sắp xếp mảng theo thuật toán InterchangeSort theo hai hướng tăng và giảm. Có thể hiện các biến trong thuật toán. Có điều chỉnh màu của các Node

|  |
| --- |
| 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++)  {  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");  }  }  }  DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;  DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;  hoanTatSapXep();  } |

* Phần code sẽ được cập nhật vào khung Code C/C++:

|  |
| --- |
| public static void InterchangeSort(bool tang = true)  {  //Vietnamese  yTuongVi =  @"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.";  //English  yTuongEn = @"• Find the smallest number in the set and interchange it with the number in the first position.  • Then find the second smallest number, and swap it with the number in the second position.  • So on.";  //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] )";  }  } |

* Hàm 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++)  {  HienThiThuatToan.ChayCodeC(6);  flag = false;  min = i;  bienArr["min"].Location = new Point(danhSachLabel[min].Location.X, danhSachLabel[min].Location.Y - 140);  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 - 140);  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 - 140);  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");  }  }  }  DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;  DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;  hoanTatSapXep();  } |

* Phần code sẽ được cập nhật vào khung Code C/C++:

public static void SelectionSort(bool tang = true)

{

//vietnamese

yTuongVi =

@"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.";

//English

yTuongEn = @"• find the smallest element

• put it in the first position

• find the next smallest element in the remaining elements

• put it in the second position

• …

• And so on, until we get to the end of the array";

//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])";

}

}

* Hàm 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 - 20);  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 - 200);  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();  } |

* Phần code sẽ được cập nhật vào khung Code C/C++:

public static void InsertionSort(bool tang = true)

{

//Vietnamese

yTuongVi =

@"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).";

//English

yTuongEn = @"• Start by considering the first two elements of the array data. If found out of order, swap them

• Consider the third element; insert it into the proper position among the first three elements.

• Consider the fourth element; insert it into the proper position among the first four elements.

• … …";

// 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]))";

}

}

* Hàm BinaryInsertionSort :

|  |
| --- |
| public void BinaryInsertionSort()  {  yTuongTextBox.Clear();  int left, right, m, i, pos;  int x;  Node nodeTam, nodeTam2;  for (i = 1; i < DanhSachNode.Count; i++)  {  pos = i - 1;  HienThiThuatToan.ChayCodeC(4);  DanhSachNode[0].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;  nodeTam = DanhSachNode[i];  x = DanhSachThamSo[i];  HienThiThuatToan.ChayCodeC(6);  DanhSachNode[i].ChuyenLen();  DanhSachNode[i].BackColor = ThamSo.mauNodeDangSX;  left = 0;  right = i - 1;  bienArr["left"].Location = new Point(danhSachLabel[left].Location.X, danhSachLabel[left].Location.Y - 140);  bienArr["left"].Text = "left = " + left;  sapxepPanel.Controls.Add(bienArr["left"]);  bienArr["pos"].Location = new Point(danhSachLabel[pos].Location.X, danhSachLabel[pos].Location.Y - 180);  bienArr["pos"].Text = "pos = " + pos;  sapxepPanel.Controls.Add(bienArr["pos"]);  bienArr["right"].Location = new Point(danhSachLabel[right].Location.X, danhSachLabel[right].Location.Y - 160);  bienArr["right"].Text = "right = " + right;  sapxepPanel.Controls.Add(bienArr["right"]);  HienThiThuatToan.ChayCodeC(7);  while (left <= right)  {  m = (left + right) / 2;  if (tangrdbtn.Checked == true)  {  if (x < DanhSachThamSo[m])  {  HienThiThuatToan.ChayCodeC(10);  bienArr["m"].Location = new Point(danhSachLabel[i].Location.X, danhSachLabel[i].Location.Y - 200);  bienArr["m"].Text = "m = " + i;  sapxepPanel.Controls.Add(bienArr["m"]);  bienArr["right"].Text = "right = " + right;  sapxepPanel.Controls.Add(bienArr["right"]);  right = m - 1;  bienArr["right"].Text = "right = " + right;  sapxepPanel.Controls.Add(bienArr["right"]);  }  else  {  HienThiThuatToan.ChayCodeC(11);  bienArr["m"].Text = "m = " + i;  sapxepPanel.Controls.Add(bienArr["m"]);  left = m + 1;  }  }  else  {  if (x > DanhSachThamSo[m])  {  HienThiThuatToan.ChayCodeC(10);  bienArr["m"].Text = "m = " + i;  sapxepPanel.Controls.Add(bienArr["m"]);  bienArr["right"].Text = "right = " + right;  sapxepPanel.Controls.Add(bienArr["right"]);  right = m - 1;  bienArr["right"].Text = "right = " + right;  sapxepPanel.Controls.Add(bienArr["right"]);  }  else  {  HienThiThuatToan.ChayCodeC(11);  bienArr["m"].Text = "m = " + i;  sapxepPanel.Controls.Add(bienArr["m"]);  left = m + 1;  }  }  HienThiThuatToan.ChayCodeC(7);  bienArr["left"].Text = "left = " + left;  sapxepPanel.Controls.Add(bienArr["left"]);  }  HienThiThuatToan.ChayCodeC(13);  bienArr["right"].Text = "right = " + right;  sapxepPanel.Controls.Add(bienArr["right"]);  pos = i - 1;  bienArr["pos"].Text = "pos = " + pos;  sapxepPanel.Controls.Add(bienArr["pos"]);  for (pos = i - 1; pos >= left; pos--)  {  bienArr["pos"].Text = "pos = " + pos;  sapxepPanel.Controls.Add(bienArr["pos"]);  bienArr["pos"].Visible = true;  HienThiThuatToan.ChayCodeC(14);  DanhSachNode[pos].ChuyenNgang(pos + 1);  DanhSachNode[pos].vitriHienTai = pos + 1;  nodeTam2 = DanhSachNode[pos + 1];  DanhSachNode[pos + 1] = DanhSachNode[pos];  DanhSachThamSo[pos + 1] = DanhSachThamSo[pos];  DanhSachNode[pos] = nodeTam2;  }  bienArr["pos"].Text = "pos = " + pos;  sapxepPanel.Controls.Add(bienArr["pos"]);  bienArr["pos"].Visible = true;  HienThiThuatToan.ChayCodeC(15);  nodeTam.ChuyenNgang(pos + 1);  nodeTam.ChuyenXuong();  DanhSachNode[pos + 1] = nodeTam;  nodeTam.vitriHienTai = pos + 1;  DanhSachNode[pos+1].BackColor = ThamSo.mauNodeHTSX;  DanhSachThamSo[left] = x;  //Cập nhật ý tưởng Text Box  Hienthimangdangsapxep(pos, i, "pos", "i");  }  DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;  DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;  hoanTatSapXep();  } |

* Phần code sẽ được cập nhật vào khung Code C/C++:

public static void BinaryInsertionSort(bool tang = true)

{

//Vietnamese

yTuongVi =

@"Cải tiến của giải thuật InsertionSort. Giải thuật chèn nhị phân cho phép giảm số lần so sánh, nhưng không làm giảm số lần dời chỗ.";

//English

yTuongEn = @"• Use binary search to reduce the number of comparisons in normal insertion sort.

• Binary Insertion Sort find use binary search to find the proper location to insert the selected item at each iteration.";

//Thêm yTuong vào yTuongThuatToan

addYTuong();

string[] code = ChuyenText(

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

{

int left, right, m, i , pos;

int x;

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

{

x = a[i]; left = 0; right = i - 1;

while(left <= right)

{

m = (left + right)/2;

if(x < a[m]) right = m - 1;

else left = m + 1;

}

for(pos = i - 1; pos >= left; pos--)

a[pos+1] = a[pos];

a[left] = x;

}

}");

//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[11] = " if(x > a[m]) right = m - 1;";

}

}

* Hàm BubbleSort :

|  |
| --- |
| private void BubbleSort()  {  int i, j;  bool flag = false;  HienThiThuatToan.ChayCodeC(2);  HienThiThuatToan.ChayCodeC(3);  for (i = 0; i < SoLuongNode - 1; i++)  {  HienThiThuatToan.ChayCodeC(4);  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--)  {  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 = Color.Green;  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();  } |

* Phần code sẽ được cập nhật vào khung Code C/C++:

public static void BubbleSort(bool tang = true)

{

//Vietnamese

yTuongVi =

@"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.";

//English

yTuongEn = @"• Exchange neighboring items until the largest item reaches the end of the array.

• Repeat the above step for the rest of the array.";

//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])";

}

}

* Hàm ShakerSort :

|  |
| --- |
| private void ShakerSort()  {  HienThiThuatToan.ChayCodeC(2);  int j, left, right, k;  HienThiThuatToan.ChayCodeC(3);  left = 0; right = DanhSachThamSo.Count - 1; k = DanhSachThamSo.Count - 1;  HienThiThuatToan.ChayCodeC(4);  bienArr["left"].Location = new Point(danhSachLabel[left].Location.X, danhSachLabel[left].Location.Y - 140);  bienArr["left"].Text = "left = " + left;  sapxepPanel.Controls.Add(bienArr["left"]);  bienArr["left"].Visible = true;  bienArr["right"].Location = new Point(danhSachLabel[right].Location.X, danhSachLabel[right].Location.Y - 160);  bienArr["right"].Text = "right = " + right;  sapxepPanel.Controls.Add(bienArr["right"]);  bienArr["right"].Visible = true;  bienArr["k"].Location = new Point(danhSachLabel[k].Location.X, danhSachLabel[k].Location.Y - 60);  bienArr["k"].Text = "k = " + k;  sapxepPanel.Controls.Add(bienArr["k"]);  bienArr["k"].Visible = true;  while (left < right)  {  HienThiThuatToan.ChayCodeC(6);  j = right;  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["j"].SendToBack();  for (j = right; j > left; j--)  {  bool thucHien = false;  if (tangrdbtn.Checked == true)  {  if (DanhSachThamSo[j] < DanhSachThamSo[j - 1])  {  thucHien = true;  }  }  else  {  if (DanhSachThamSo[j] > DanhSachThamSo[j - 1])  {  thucHien = true;  }  }  HienThiThuatToan.ChayCodeC(7);  if (thucHien)  {  HienThiThuatToan.ChayCodeC(9);  CapNhatThamSo(j, j - 1);  DichuyenCacNode(j, j - 1);  Hienthimangdangsapxep(j - 1, j, "k", "j"); // mang dang sap xep  HienThiThuatToan.ChayCodeC(10);  k = j;  bienArr["k"].Location = new Point(danhSachLabel[k].Location.X, danhSachLabel[k].Location.Y - 60);  bienArr["k"].Text = "k = " + k;  sapxepPanel.Controls.Add(bienArr["k"]);  }  HienThiThuatToan.ChayCodeC(6);  bienArr["k"].Location = new Point(danhSachLabel[k].Location.X, danhSachLabel[k].Location.Y - 60);  bienArr["k"].Text = "k = " + k;  sapxepPanel.Controls.Add(bienArr["k"]);  }  HienThiThuatToan.ChayCodeC(12);  left = k;  bienArr["left"].Location = new Point(danhSachLabel[left].Location.X, danhSachLabel[left].Location.Y - 140);  bienArr["left"].Text = "left = " + left;  sapxepPanel.Controls.Add(bienArr["left"]);  HienThiThuatToan.ChayCodeC(13);  j = left;  bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 40);  bienArr["j"].Text = "j = " + j;  sapxepPanel.Controls.Add(bienArr["j"]);  for (j = left; j < right; j++)  {  bool thucHien = false;  if (tangrdbtn.Checked == true)  {  if (DanhSachThamSo[j + 1] < DanhSachThamSo[j])  thucHien = true;  }  else  {  if (DanhSachThamSo[j + 1] > DanhSachThamSo[j])  thucHien = true;  }  HienThiThuatToan.ChayCodeC(14);  if (thucHien)  {  HienThiThuatToan.ChayCodeC(16);  CapNhatThamSo(j, j + 1);  DichuyenCacNode(j + 1, j);  Hienthimangdangsapxep(j, j + 1, "k", "j");  HienThiThuatToan.ChayCodeC(17);  k = j;  bienArr["k"].Location = new Point(danhSachLabel[k].Location.X, danhSachLabel[k].Location.Y - 60);  bienArr["k"].Text = "k = " + k;  sapxepPanel.Controls.Add(bienArr["k"]);  }  HienThiThuatToan.ChayCodeC(13);  bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 40);  bienArr["j"].Text = "j = " + j;  sapxepPanel.Controls.Add(bienArr["j"]);  }  HienThiThuatToan.ChayCodeC(19);  right = k;  bienArr["right"].Location = new Point(danhSachLabel[right].Location.X, danhSachLabel[right].Location.Y - 160);  bienArr["right"].Text = "right = " + right;  sapxepPanel.Controls.Add(bienArr["right"]);  HienThiThuatToan.ChayCodeC(4);  bienArr["left"].Location = new Point(danhSachLabel[left].Location.X, danhSachLabel[left].Location.Y - 140);  bienArr["left"].Text = "left = " + left;  sapxepPanel.Controls.Add(bienArr["left"]);  bienArr["right"].Location = new Point(danhSachLabel[right].Location.X, danhSachLabel[right].Location.Y - 160);  bienArr["right"].Text = "right = " + right;  sapxepPanel.Controls.Add(bienArr["right"]);  }  HienThiThuatToan.ChayCodeC(4);  DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;  DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;  hoanTatSapXep();  } |

* Phần code sẽ được cập nhật vào khung Code C/C++:

public static void ShakerSort(bool tang = true)

{

//Vietnamese

yTuongVi = @"Trong mỗi lần sắp xếp, duyệt mảng theo 2 lượt từ 2 phía khác nhau:

Lượt đi: đẩy phần tử nhỏ về đầu mảng.

Lượt về: đẩy phần tử lớn về cuối mảng.

Ghi nhận lại những đoạn đã sắp xếp nhằm tiết kiệm các phép so sánh thừa.";

//English

yTuongEn = @"A variant of the bubble sort method, in shaker sort, n elements are sorted in n/2 phases:

• Each phase of shaker sort consists of a left to right bubbling pass followed by a right to left bubbling pass.

• In a bubbling pass pairs of adjacent elements are compared and swapped if they are out of order.";

//thêm ytuong

addYTuong();

string[] code = ChuyenText(

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

{

int j, left, right, k;

left = 0; right = N - 1, k = N - 1;

while(left < right)

{

for(j = right; j > left; j--)

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

{

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

k = j;

}

left = k;

for(j = left; j < right; j++)

if(a[j + 1] < a[j])

{

Swap(a[j], a[j + 1]);

k = j;

}

right = 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[7] = " if(a[j] > a[j - 1])";

codeListBox.Items[14] = " if(a[j + 1] > a[j])";

}

}

* Hàm HeapSort :

|  |
| --- |
| private void HeapSort()  {  HienThiThuatToan.ChayCodeC(1);  HienThiThuatToan.ChayCodeC(2);  CreateHeap(DanhSachThamSo.Count);  HienThiThuatToan.ChayCodeC(3);  HienThiThuatToan.ChayCodeC(4);  int r = DanhSachThamSo.Count - 1;  HienThiThuatToan.ChayCodeC(5);  while (r > 0)  {  HienThiThuatToan.ChayCodeC(6);  HienThiThuatToan.ChayCodeC(7);  CapNhatThamSo(0, r);  DichuyenCacNode(r, 0);  HienThiThuatToan.ChayCodeC(8);  bienArr["right"].Location = new Point(danhSachLabel[r].Location.X, danhSachLabel[r].Location.Y - 160);  bienArr["right"].Text = "right = " + r;  sapxepPanel.Controls.Add(bienArr["right"]);  bienArr["right"].Visible = true;  bienArr["right"].SendToBack();  r--;  HienThiThuatToan.ChayCodeC(9);  if (r > 0)  {  HienThiThuatToan.ChayCodeC(10);  Shift(0, r);  }  HienThiThuatToan.ChayCodeC(11);  HienThiThuatToan.ChayCodeC(5);  }  HienThiThuatToan.ChayCodeC(12);  DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;  DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;  if (DanhSachNode.Count != 0)  {  hoanTatSapXep();  }  }  private void CreateHeap(int N)  {  HienThiThuatToan.ChayCodeC(14);  HienThiThuatToan.ChayCodeC(15);  HienThiThuatToan.ChayCodeC(16);  HienThiThuatToan.ChayCodeC(17);  int l = N / 2 - 1;  HienThiThuatToan.ChayCodeC(18);  while (l >= 0)  {  HienThiThuatToan.ChayCodeC(19);  HienThiThuatToan.ChayCodeC(20);  Shift(l, N - 1);  HienThiThuatToan.ChayCodeC(21);  bienArr["left"].Location = new Point(danhSachLabel[l].Location.X, danhSachLabel[l].Location.Y - 140);  bienArr["left"].Text = "left = " + l;  sapxepPanel.Controls.Add(bienArr["left"]);  bienArr["left"].Visible = true;  bienArr["left"].SendToBack();  l--;  HienThiThuatToan.ChayCodeC(22);  HienThiThuatToan.ChayCodeC(18);  }  HienThiThuatToan.ChayCodeC(23);  }  private void Shift(int l, int r)  {  HienThiThuatToan.ChayCodeC(25);  HienThiThuatToan.ChayCodeC(26);  HienThiThuatToan.ChayCodeC(27);  HienThiThuatToan.ChayCodeC(28);  int i = l;  int j = 2 \* i + 1;  HienThiThuatToan.ChayCodeC(29);  while (j <= r)  {  HienThiThuatToan.ChayCodeC(30);  HienThiThuatToan.ChayCodeC(31);  if (tangrdbtn.Checked == true)  {  if (j < r && DanhSachThamSo[j] < DanhSachThamSo[j + 1])  {  HienThiThuatToan.ChayCodeC(32);  j++;  }  }  else  {  if (j < r && DanhSachThamSo[j] > DanhSachThamSo[j + 1])  {  HienThiThuatToan.ChayCodeC(32);  j++;  }  }  HienThiThuatToan.ChayCodeC(33);  if (tangrdbtn.Checked == true)  {  if (DanhSachThamSo[i] < DanhSachThamSo[j])  {  HienThiThuatToan.ChayCodeC(34);  HienThiThuatToan.ChayCodeC(35);  DichuyenCacNode(j, i);  CapNhatThamSo(i, j);  Hienthimangdangsapxep(i, j, "i", "j");  HienThiThuatToan.ChayCodeC(36);  i = j;  HienThiThuatToan.ChayCodeC(37);  j = 2 \* i + 1;  HienThiThuatToan.ChayCodeC(38);  }  else  {  HienThiThuatToan.ChayCodeC(39);  return;  }  }  else  {  if (DanhSachThamSo[i] > DanhSachThamSo[j])  {  HienThiThuatToan.ChayCodeC(34);  HienThiThuatToan.ChayCodeC(35);  CapNhatThamSo(i, j);  DichuyenCacNode(i, j);  Hienthimangdangsapxep(i, j, "i", "j");  HienThiThuatToan.ChayCodeC(36);  i = j;  HienThiThuatToan.ChayCodeC(37);  j = 2 \* i + 1;  HienThiThuatToan.ChayCodeC(38);  }  else  {  HienThiThuatToan.ChayCodeC(39);  return;  }  }  HienThiThuatToan.ChayCodeC(40);  HienThiThuatToan.ChayCodeC(29);  }  HienThiThuatToan.ChayCodeC(41);  } |

* Phần code sẽ được cập nhật vào khung Code C/C++:

public static void HeapSort(bool tang = true)

{

//Vietnamese

yTuongVi =

@"Heap Sort tận dụng được các phép so sánh ở bước i-1, mà thuật toán sắp xếp chọn trực tiếp (Selectin Sort) không tận dụng được.

Để làm được điều này Heap sort thao tác dựa trên cây.";

//English

yTuongEn = @"• Transform the array of items into a heap.

• Invoke the “retrieve & delete” operation repeatedly, to extract the largest item remaining in the heap, until the heap is empty. Store each item retrieved from the heap into the array from back to front.";

//Thêm yTuong vào yTuongThuatToan

addYTuong();

string[] code = ChuyenText(

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

{

CreateHeap(a,N - 1);

int r;

r = N - 1;

while(r >= 0)

{

Swap(a[0], a[r]);

r--;

if(r > 0 )

Shift(a,0,r);

}

}

void CreateHeap(int a[], int N)

{

int l;

l = N/2 - 1;

while(l >= 0)

{

Shift(a,l,N - 1);

l--;

}

}

void Shift(int a[], int l, int r)

{

int i = l;

int j = 2\*i + 1;

while(j <= r)

{

if(j < r && a[j] < a[j+1])

j++;

if(a[i] < a[j])

{

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

i = j;

j = 2\*i + 1;

}

else return;

}

}");

//thêm code vào codeListBox

codeListBox.Items.Clear();

foreach (string item in code)

{

codeListBox.Items.Add(item);

}

//nếu sắp giảm thì sửa lại

if (!tang)

{

}

}

* Hàm 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 - 140);  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 - 160);  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 - 60);  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 - 40);  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 - 40);  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 - 60);  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 - 40);  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 - 40);  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 - 60);  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);  }  } |

* Phần code sẽ được cập nhật vào khung Code C/C++:

public static void QuickSort(bool tang = true)

{

//Vietnamese

yTuongVi =

@"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.";

//English

yTuongEn = @"If the number of elements in S is 0 or 1, then

return (base case).

• Pick any element v in S (called the pivot).

• Partition the elements in S except v into two

disjoint groups:

• S1 = {x ∈ S – {v} | x ≤ v}

• S2 = {x ∈ S – {v} | x ≥ v}

• Return {QuickSort(S1

) + v + QuickSort(S2

)}";

//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])";

}

}

* Hàm MergeSort :

|  |
| --- |
| 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 - 200);  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);  }  void MergeSort()  {  yTuongTextBox.Clear();  bienArr["a:"].Location = new Point(0, 110 + DanhSachNode[0].Height / 2);  bienArr["a:"].Text = "a:";  sapxepPanel.Controls.Add(bienArr["a:"]);  bienArr["c:"].Location = new Point(0, 160 + DanhSachNode[0].Height / 2);  bienArr["c:"].Text = "c:";  sapxepPanel.Controls.Add(bienArr["c:"]);  bienArr["b:"].Location = new Point(0, 60 + 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();  } |

* Phần code sẽ được cập nhật vào khung Code C/C++:

public static void MergeSort(bool tang = true)

{

//Vietnamese

yTuongVi =

@"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.";

//English

yTuongEn = @"• Divide the array into two halves.

• Conquer each half by sorting it recursively

• Combine the results obtained, by merging the two halves to obtain one fully sorted array";

//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)";

}

}

* Hàm ShellSort :

|  |
| --- |
| private void ShellSort()  {  HienThiThuatToan.ChayCodeC(1);  for (int gap = SoLuongNode / 2; gap > 0; gap /= 2)  {  HienThiThuatToan.ChayCodeC(3);  bienArr["gap"].Location = new Point(danhSachLabel[gap].Location.X, danhSachLabel[gap].Location.Y - 180);  bienArr["gap"].Text = "gap = " + gap;  sapxepPanel.Controls.Add(bienArr["gap"]);  bienArr["gap"].SendToBack();  bienArr["gap"].Visible = true;  for (int i = gap; i < SoLuongNode; 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(5);  if (tangrdbtn.Checked == true)  {  HienThiThuatToan.ChayCodeC(7);  for (int j = i; j >= gap && DanhSachThamSo[j] < DanhSachThamSo[j - gap]; j -= gap)  {  bienArr["j"].Location = new Point(danhSachLabel[j].Location.X, danhSachLabel[j].Location.Y - 40);  bienArr["j"].Text = "j = " + j;  bienArr["j"].SendToBack();  bienArr["j"].Visible = true;  sapxepPanel.Controls.Add(bienArr["j"]);  HienThiThuatToan.ChayCodeC(9);  Thread.Sleep(ThamSo.ThoiGianDoi);  DanhSachNode[j].BackColor = DanhSachNode[j - gap].BackColor = ThamSo.mauNodeDangSX;  CapNhatThamSo(j, j - gap);  DichuyenCacNode(j, j - gap);  Hienthimangdangsapxep(i, j, "i", "j");  }  }  if (giamrdbtn.Checked == true)  {  HienThiThuatToan.ChayCodeC(7);  for (int j = i; j >= gap && DanhSachThamSo[j] > DanhSachThamSo[j - gap]; j -= gap)  {  HienThiThuatToan.ChayCodeC(9);  Thread.Sleep(ThamSo.ThoiGianDoi);  DanhSachNode[j].BackColor = DanhSachNode[j - gap].BackColor = ThamSo.mauNodeDangSX;  CapNhatThamSo(j, j - gap);  DichuyenCacNode(j, j - gap);  Hienthimangdangsapxep(i, j, "i", "j");  }  }  }  }  DanhSachNode[0].BackColor = ThamSo.mauNodeHTSX;  DanhSachNode[DanhSachNode.Count - 1].BackColor = ThamSo.mauNodeHTSX;  hoanTatSapXep();  } |

* Phần code sẽ được cập nhật vào khung Code C/C++:

public static void ShellSort(bool tang = true)

{

//Vietnamese

yTuongVi =

@"Cải tiến của phương pháp chèn trực tiếp (Insertion Sort).

Phân hoạch dãy thành các dãy con.

Sắp xếp các dãy con theo phương pháp chèn trực tiếp.

Dùng phương pháp chèn trực tiếp sắp xếp lại cả dãy.";

//English

yTuongEn = @"• Arrange the data sequence in a two-dimensional array.

• Sort the columns of the array.";

//Thêm yTuong vào yTuongThuatToan

addYTuong();

string[] code = ChuyenText(

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

{

for (int gap = N / 2; gap > 0; gap /= 2)

{

for (int i = gap; i < N; i++)

{

for (int j = i; j >= gap && a[j] < a[j - gap]; j -= gap)

{

Swap(a[j], a[j - gap]);

}

}

}

}");

//thêm code vào codeListBox

codeListBox.Items.Clear();

foreach (string item in code)

{

codeListBox.Items.Add(item);

}

//nếu sắp giảm thì sửa lại

//if (!tang)

//{

//}

}

# **5.** **KẾT LUẬN**

## **5.1 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,

## **5.2 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

## **5.3 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

## **5.4 Phân rã công việc**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Công việc** | **Duration (days)** | **Assign To** | **% Finished** |
| 1 | Tìm hiểu sơ bộ & đăng ký đồ án |  | Sáng, Quy, Nhung | 100% |
| 2 | Tìm hiểu công nghệ liên quan |  | Sáng | 100% |
| 3 | Thiết kế giao diện ban đầu |  | Nhung | 100% |
| 4 | Tạo UserControls |  | Sáng | 100% |
| 5 | Xử lý bất đồng bộ, mô phỏng animation |  | Sáng | 100% |
| 6 | Xử lý đa ngôn ngữ dùng resource |  | Quy | 100% |
| 7 | Viết các hàm tạo Node, frm input |  | Sáng | 100% |
| 8 | Tạo Form cài đặt |  | Quy | 100% |
| 9 | Xử lý thuật toán insection sort, merge sort, binary Insection sort, heap sort |  | Sáng | 100% |
| 10 | Xử lý các thuật toán shellsoft,shaker sort, quick  Soft |  | Quy | 100% |
| 11 | Xử lý các thuật toán selection sort, bubble sort  Interchange sort |  | Nhung | 100% |
| 12 | Xử lý hiển thị tham số khi thực hiện sắp xếp |  | Quy | 100% |
| 13 | Xử lý hiện mảng đang sắp xếp |  | Nhung | 100 |
| 14 | Xử lý dừng quá trình sắp xếp |  | Quy | 100% |
| 15 | Xử lý hủy quá trình, debug |  | Sáng | 100 |
| 16 | Tạo file cài đặt, sửa lỗi |  | Sáng | 100% |
| **I. VIẾT BÁO CÁO** | | | | |
| 3 | **Chương 1 – Giới thiệu** |  |  |  |
| 4 | 1.1. Mô phỏng các thuật toán sắp xếp tiêu biểu trong lập trình |  | Nhung | 100% |
| 5 | 1.2. Các chức năng |  | Nhung | 100% |
| 6 | **Chương 2: Phân tích** |  |  |  |
| 7 | 2.1. Phân tích |  | Nhung | 100% |
| 8 | 2.1.1 Phân tích hiệu ứng |  | Nhung | 100% |
| 9 | 2.1.2 Phân tích chức năng |  | Nhung | 100% |
| 10 | 2.2. Kỹ Thuật sử dụng |  | Nhung, Quy | 100% |
| 11 | 2.2.1 Xử lý bất đồng bộ với Thread và Task |  | Quy | 100% |
| 12 | 2.2.2 Xử lý việc thay đổi tốc độ sắp xếp |  | Nhung | 100% |
| 13 | 2.2.3 Xử lý việc thay di chuyển các phần tử và các biến |  | Nhung | 100% |
| 14 | 2.2.4 Xử lý việc tính thời gian thực hiện quá trình sắp xếp |  | Nhung | 100% |
| 15 | 2.2.5 Xử lý việc lưu thông tin cài đặt của người dùng |  | Quy | 100% |
| 16 | **Chương 3: Minh họa việc sử dụng phần mềm** |  | Sáng | 100% |
| 17 | **Chương 4: Cài đặt** |  |  |  |
| 18 | 4.1. Nền tảng công nghệ |  | Quy | 100% |
| 19 | 4.2. Các hàm xử lí chính |  | Quy | 90% |
| 20 | 4.2.1 Phần tử và các hàm có trong phần tử |  | Quy | 90% |
| 21 | 4.2.2 Các hàm tạo mảng |  | Quy | 90% |
| 22 | 4.2.3 Các hàm bắt đầu, hủy quá trình sắp xếp và các hàm hổ trợ |  | Quy | 90% |
| 23 | 4.2.4 Các hàm di chuyển vị trí của Node và hoán vị 2 Node |  | Quy | 90% |
| 24 | 4.2.5 Hàm tạm dừng quá trình sắp xếp |  | Quy | 90% |
| 25 | 4.2.6 Hàm chọn từng dòng code trong khung Code C/C++ |  | Quy | 90% |
| 26 | 4.2.7 Cập nhật trạng thái ý tưởng thuật toán |  | Quy | 90% |
| 27 | 4.2.8 Các hàm sắp xếp mảng |  | Quy | 90% |
| 28 | **Chương 5: Kết luận** |  | Sáng | 100% |
| 29 | 5.1. Kết quả đạt được |  | Sáng | 100% |
| 30 | 5.2. Hạn chế |  | Sáng | 100% |
| 31 | 5.3 Phân rã công việc |  | Quy | 100% |
| 32 | **Chương 6: Tài liệu tham khảo** |  | Sáng | 100% |
| 33 | Edit báo cáo |  | Quy | 80% |
| **II. KIỂM THỬ** | | | | |
|  | 1. Acceptance Test |  |  |  |
| **III. NỘP BÁO CÁO TIẾN ĐỘ & SẢN PHẨM** | | | **Tiến độ sản phẩm** | |
|  | **Ngày 20 tháng 03 năm 2018** | | Cơ bản hoàn thành định hướng đồ án. Thiết kế sơ bộ giao diện. | |
|  | **Ngày 25 tháng 04 năm 2018** | | Hoàn thành sơ bộ các chức năng lớn: Di chuyển node, vị trí node, thuật toán, khởi tạo. | |
|  | **Ngày 20 tháng 05 năm 2018** | | Hoàn thành các chức năng chính sắp xếp, di chuyển node trên GUI, ngôn ngữ, ý tưởng,.. | |
|  | **Ngày 01 tháng 06 năm 2018** | | Hoàn thành các chức năng phụ còn lại: Điều khiển, huỷ, debug, hướng dẫn, cài đặt, viết báo cáo.. | |
|  | **Ngày 17 tháng 06 năm 2018** | | Hoàn thiện hết mức có thể về đồ án. | |

# 6. TÀI LIỆU THAM KHẢO

[www.Howkteam.vn](http://www.Howkteam.vn)

Slide bài giảng

Timer

<https://msdn.microsoft.com/en-us/library/system.timers.timer(v=vs.110).aspx>

ManualResetevent

<https://msdn.microsoft.com/en-us/library/system.threading.manualresetevent(v=vs.110).aspx>

tạo user controls

youtube, và bài tập

Thread: slide

Task:

https://viblo.asia/p/lap-trinh-bat-dong-bo-trong-c-DZrGNDoWkVB

<https://msdn.microsoft.com/en-us/library/mt472912(v=vs.110).aspx>

Các trang stackoverflow.com và các forum khác.