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

**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**BÁO CÁO ĐỒ ÁN LẬP TRÌNH TRỰC QUAN**



**ĐỒ ÁN PHẦN MỀM MÔ PHỎNG THUẬT TOÁN SẮP XẾP**

Sinh viên thực hiện: Nhóm PKP

Nguyễn Phi Phú 16520935

Mai Thắng Đăng Khoa 16520598

Nguyễn Tấn Phúc 16520954

Lớp: [IT008.I23.PMCL](https://courses.uit.edu.vn/course/view.php?id=3494)

Giảng viên hướng dẫn:

Huỳnh Tuấn Anh

Huỳnh Hồ Thị Mộng Trinh

# MỤC LỤC

[MỤC LỤC 2](#_Toc517086456)

[LỜI MỞ ĐẦU 4](#_Toc517086457)

[1. Lý do chọn đề tài 4](#_Toc517086458)

[2. Mục tiêu, nhiệm vụ 4](#_Toc517086459)

[3. Giới hạn và phạm vi 5](#_Toc517086460)

[Chương 1. Cơ sở lý thuyết 6](#_Toc517086461)

[1.1. Thuật toán 6](#_Toc517086462)

[1.1.1. Khái niệm thuật toán sắp xếp 6](#_Toc517086463)

[1.1.2. Các đặc trưng của thuật toán 6](#_Toc517086464)

[1.1.3. Độ phức tạp của thuật toán 6](#_Toc517086465)

[1.2. Lập trình trên Windows Forms 6](#_Toc517086466)

[1.2.1. Khái niệm 7](#_Toc517086467)

[1.2.2. Kiến trúc ứng dụng Windows Forms 7](#_Toc517086468)

[Chương 2. Kỹ thuật xử lý 8](#_Toc517086469)

[2.1. Phân tích 8](#_Toc517086470)

[2.1.1. Phân tích hiệu ứng 8](#_Toc517086471)

[2.1.2. Phân tích chức năng 8](#_Toc517086472)

[2.1.3. Biểu đồ phân cấp chức năng của hệ thống 9](#_Toc517086473)

[2.2. Kỹ thuật sử dụng 9](#_Toc517086474)

[2.2.1. Xử lý bất đồng bộ với Thread 9](#_Toc517086475)

[2.2.2. Xử lý việc thay đổi tốc độ sắp xếp 10](#_Toc517086476)

[2.2.3. Xử lý việc di chuyển các phần tử và các biến 11](#_Toc517086477)

[2.2.4. Xử lý việc tính thời gian thực hiện quá trình sắp xếp 11](#_Toc517086478)

[2.2.5. Xử lý việc sử dụng các phím tắt 11](#_Toc517086479)

[2.2.6. Xử lý việc chạy chế độ Debug 11](#_Toc517086480)

[Chương 3. Thiết kế 13](#_Toc517086481)

[1.1. Thiết kế giao diện và thành phần thiết kế giao diện 13](#_Toc517086482)

[3.3.1. Tổng thể về giao diện hệ thống 13](#_Toc517086483)

[3.3.2. Giao diện Trang Chủ (frmIntroduct) 13](#_Toc517086484)

[3.3.3. Giao diện Giới Thiệu Người Thực Hiện (frmExcecutors) 14](#_Toc517086485)

[3.3.4. Giao diện Ứng dụng Mô Phỏng (frmApplication) 16](#_Toc517086486)

[Chương 4. Cài đặt 17](#_Toc517086487)

[4.1. Nền tảng công nghệ 17](#_Toc517086488)

[4.2. Các hàm xử lý chính 17](#_Toc517086489)

[4.2.1. Node và các hàm có trong Node 17](#_Toc517086490)

[4.2.2. Hàm tạo mảng và xóa mảng 26](#_Toc517086491)

[4.2.3. Hàm chạy và tạm dừng việc sắp xếp 29](#_Toc517086492)

[4.2.4. Hàm chọn từng dòng Code trong khung listCode và chế độ Debug 31](#_Toc517086493)

[4.2.5. Các hàm sắp xếp mảng 32](#_Toc517086494)

[4.3. Cài đặt phần mềm 80](#_Toc517086495)

[Chương 5. Kiểm thử 84](#_Toc517086496)

[5.1. Unit Test 84](#_Toc517086497)

[5.2. Integration Test 84](#_Toc517086498)

[5.3. System Test 84](#_Toc517086499)

[5.3.1. Phương pháp kiểm tra 84](#_Toc517086500)

[Chương 6. Tổng kết 85](#_Toc517086501)

[6.1. Tổng kết 85](#_Toc517086502)

[6.2. Nhận xét và đánh giá 85](#_Toc517086503)

[6.3. Phân Rã Công Việc 85](#_Toc517086504)

[Chương 7. Tài liệu tham khảo 88](#_Toc517086505)

# LỜI MỞ ĐẦU

## Lý do chọn đề tài

Ngày nay, ứng dụng công nghệ thông tin vào việc dạy học đang

trờ thành trào lưu của nền giáo dục hiện đại. Trong nền giáo dục nói riêng thì với các chương trình đào tạo còn thiên về lý thuyết chưa gắn với thực tiễn do đó thông qua việc sử dụng các công cụ hỗ trợ làm cho bài giảng của giáo viên thêm phong phú và trực quan hơn, giúp các thầy giáo, cô giáo nâng cao chất lượng dạy học; trang bị cho học sinh kiến thức cần thiết về CNTT. Thì những phần mềm mô phỏng là một nhu cầu tất yếu của không những riêng cho nền giáo dục mà còn là cho cả đời sống, xã hội hiện nay.

Các phần mềm mô phỏng thuật toán được sử dụng như một công cụ có tính chất giúp đỡ trọng việc dạy các thuật toán đồ thị, các thuật toán sắp xếp,… khác nhau bằng máy tính. Nó có thể cung cấp các mô phỏng bằng đồ họa của một thuật toán và các thay đổi trong cấu trúc dữ liệu của nó trong suốt quá trình thực thi.

Đối với việc học cấu trúc dữ liệu và giải thuật của các thuật toán sắp xếp thì được xem là môn học mang tính trừu tượng dành cho học sinh và sinh viên, làm cho học sinh, sinh viên và cũng như giáo viên rất khó trong việc truyền đạt và lĩnh hội kiến thức vì học sinh, sinh viên không thể hình dùng được chương trình chạy như thế nào. Trong khi các phương pháp giảng giải thong thường chưa đủ để hỗ trợ trong việc tưởng tượng và hiểu được giải thuật thì các phần mềm mô phỏng các thuật toán sắp xếp là một giải pháp tuyệt vời cho giáo viên cũng như học sinh và sinh viên chúng ta.

Và đó cũng là đề tài mà được chọn để nghiên cứu và thực hiện trong khóa luận này.

## Mục tiêu, nhiệm vụ

* Nghiên cứu tổng quan về mô phỏng thuật toán.
* Hướng đến các kỹ thuật lập trình trên winform và ngôn ngữ lập trình C#.
* Áp dụng kết quả nghiên cứu và học tập tạo ra một ứng dụng mô phỏng thuật toán sắp xếp.
* Giúp đỡ người dùng có thể hiểu rõ hơn về cách thực hiện của các thuật toán sắp xếp.
* Xây dựng phần mềm với giao diện đơn giản, dễ dàng sử dụng và cài đặt chương trình
* Nắm vững được kiến thức cơ bản về lập trình Windows Forms
* Phát triển ứng dụng với có them nhiều chức năng hữu ích

## Giới hạn và phạm vi

* Xây dựng phần với chức năng mô phỏng thuật toán
* Phạm vi ứng dụng: Sử dụng offline bằng cái tải về
* Đối tượng sử dụng: Phần mềm có thể sử dụng bởi người quản trị, các thành viên, người dùng...

# Cơ sở lý thuyết

## Thuật toán

### Khái niệm thuật toán sắp xếp

* Thuật toán là một tập hữu hạn các chỉ thị hay phương cách được định nghĩa rõ rang cho việc hoàn tất một sự việc từ một trạng thái ban đầu cho trước; khi các chỉ thị này được áp dụng triệt để thì sẽ dẫn đến kết quả sau cùng như đã dự đoán.
* Sắp xếp là một quá trình biến đổi một danh sách các đối tượng thành một danh sách thỏa mãn một thứ tự xác định nào đó.

### Các đặc trưng của thuật toán

* Tính xác định: các phải rõ rang, thực hiện được ra một kết quả nào đó
* Tính hữu hạn: phải có số bước nhất định và điểm kết thúc
* Tính kết quả: với dữ liệu phù hợp, thuật toán phải cho kết quả đúng yêu cầu
* Tính phổ dụng: áp dụng được cho nhiều bài toán khác có cùng cấu trúc, với các dữ liệu khác nhau
* Tính hiệu quả: đơn giản, dễ hiểu, tối ưu hóa bộ nhớ và thời gian thực hiện
* Tính hình thức: từng bước trong thuật toán luôn thực hiện đúng như kịch bản (chương trình) mà không biết đến mục tiêu cuối cùng (không tự suy đoán)

### Độ phức tạp của thuật toán

* Để đánh giá hiệu quả của một thuật toán, có thể xét số các phép tính phải thực hiện khi thực hiện thuật toán này. Thông thường số các phép tính được thực hiện phụ thuộc vào cỡ của bài toán, tức là độ lớn của đầu vào. Vì thế độ phức tạp thuật toán là một hàm phụ thuộc đầu vào O(n) n là độ lớn đầu vào, tùy thuộc từng bài toán mà n có thể nhận những giá trị khác nhau.

## Lập trình trên Windows Forms

### Khái niệm

* Windows Forms là ứng dụng có giao diện độ họa chạy trên hệ điều hành Windows
* Windows Forms là ứng dụng chạy trên máy tính của người dùng
* Windows Forms là một phần của kiến trúc .NET

### Kiến trúc ứng dụng Windows Forms

* Windows Forms là cách cơ bản để cung cấp các thành phầ giao diện (GUI components) cho môi trường .NET Framework
* Windows Forms được xấy dựng trên thư viện Windows API
* Windows Forms cơ bản bao gồm:
* Một Form là khung dùng hiển thị thông tin người dùng
* Các Control được đặt trong form và được lập trình để đáp ứng sự kiện

# Kỹ thuật xử lý

## Phân tích

### Phân tích hiệu ứng

* Để mô phỏng hiệu ứng, đồ họa ta cần tạo ra 1 mảng để chứa các phần tử. Và cho các phần tử di chuyển một cách trực quan, đồng bộ trên form theo thuật toán cần sắp xếp. Chúng ta cũng có thể thay đổi tốc độ di chuyển của các phần tử dựa vào thanh tốc độ.
* 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.
* 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.
* Bên cạnh đó còn có hiệu ứng chọn listbox Code mẫu và listbox Ý tưởng thuật toán bằng các button tương ứng để cho form trở nên gọn gàng hơn.

### Phân tích chức năng

* Các chức năng chính:
* Khởi tạo mảng (2 cách):
* Tạo ngẫu nhiên: tạo một mảng với các phần tử có giá trị bất kì.
* 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.
* Bảng điều khiển:
* Bắt đầu: bắt đầu sắp xếp các phần tử dự vào thuật toán được chọn.
* Tạm dừng : cho tạm dừng quá trình sắp xếp.
* Chạy debug: chuyển quá trình sắp xếp vào chế độ Debug vào cho phép xem từng bước chạy của thuật toán.
* Hiển thị thời gian thực hiện: cho xem tổng thời gian thực hiện quá trình sắp xếp.
* Chế độ Reset: Khởi tạo lại tất cả như ban đầu
* Các chức năng phụ:
* Thay đổi tốc độ quá trình sắp xếp
* Cập nhật thông tin cho quá trình sắp xếp
* Cho phép chạy các loại thuật toán ở 2 hướng là tăng và giảm
* Canh lề giữa cho các Node ở vị trí giữa màn hình chạy mẫu tùy thuộc vào số lượng của các Node
* Có các phím tắt sử dụng phần mềm dễ hơn
* Khi việc mô phỏng đang chạy thì mội hoạt động như Create, Reset, Back, Exit, chọn các loại thuật toán khác…

### 2.1.3. Biểu đồ phân cấp chức năng của hệ thống



## Kỹ thuật sử dụng

### Xử lý bất đồng bộ với Thread

* 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.
* 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
* 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.

### 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

### 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.

### Xử lý việc tính thời gian thực hiện quá trình sắp xếp

* Thời gian của quá trình sắp xếp được tính bằng việc sử dụng Timer. Thời gian được tính từ lúc quá trình sắp xếp bắt đầu đến lúc quá trình kết thúc.
* Nếu chạy trong chế độ Debug thì khoảng thời gian này sẽ không được tính.

### Xử lý việc sử dụng các phím tắt

* Để thuận tiện cho việc sử dụng phần mềm ta có thể nhần các phím tắt thay thế cho việc nhấn các Button có sẵn.Để có thể bắt các phím được nhấn ta cần override lại hàm ProcessCmdKey của form, các control con cũng sẽ sử dụng lại hàm override này .Qua đó các phím ấn xuống sẽ tương ứng với các Button cần thực hiện

### Xử lý việc chạy chế độ Debug

* Để chạy chế độ Debug ta đặt một ManualResetEvent vào hàm chuyển qua từng dòng code. Mỗi khi thực hiện xong một lệnh thì quá trình sẽ tự động được dừng lại. Khi đó chương trình đợi người dùng nhấn vào nút Debug thì sẽ ra lệnh cho trương trình chạy lệnh tiếp theo. Cứ tiếp tục lặp lại các bước trên ta sẽ được chế độ chạy Debug

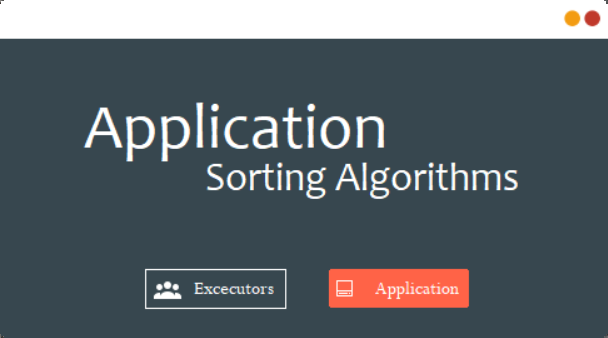
# Thiết kế

## Thiết kế giao diện và thành phần thiết kế giao diện

### Tổng thể về giao diện hệ thống

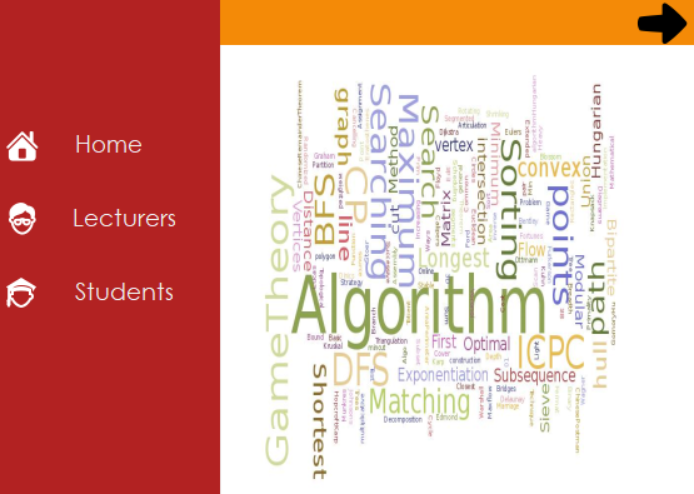


### Giao diện Trang Chủ (frmIntroduct)



* Mô tả thành phần giao diện:
* Sử dụng 4 Button:
* 2 Button hình tròn góc phải trên cùng có chức năng: Thoát và thu nhỏ ứng dụng
* Button Application có chức năng chuyển từ form frmIntroduct sang form frmApplication
* Button Excecutors có chức năng chuyển từ form frmIntroduct sang form frmExcecutors

### Giao diện Giới Thiệu Người Thực Hiện (frmExcecutors)



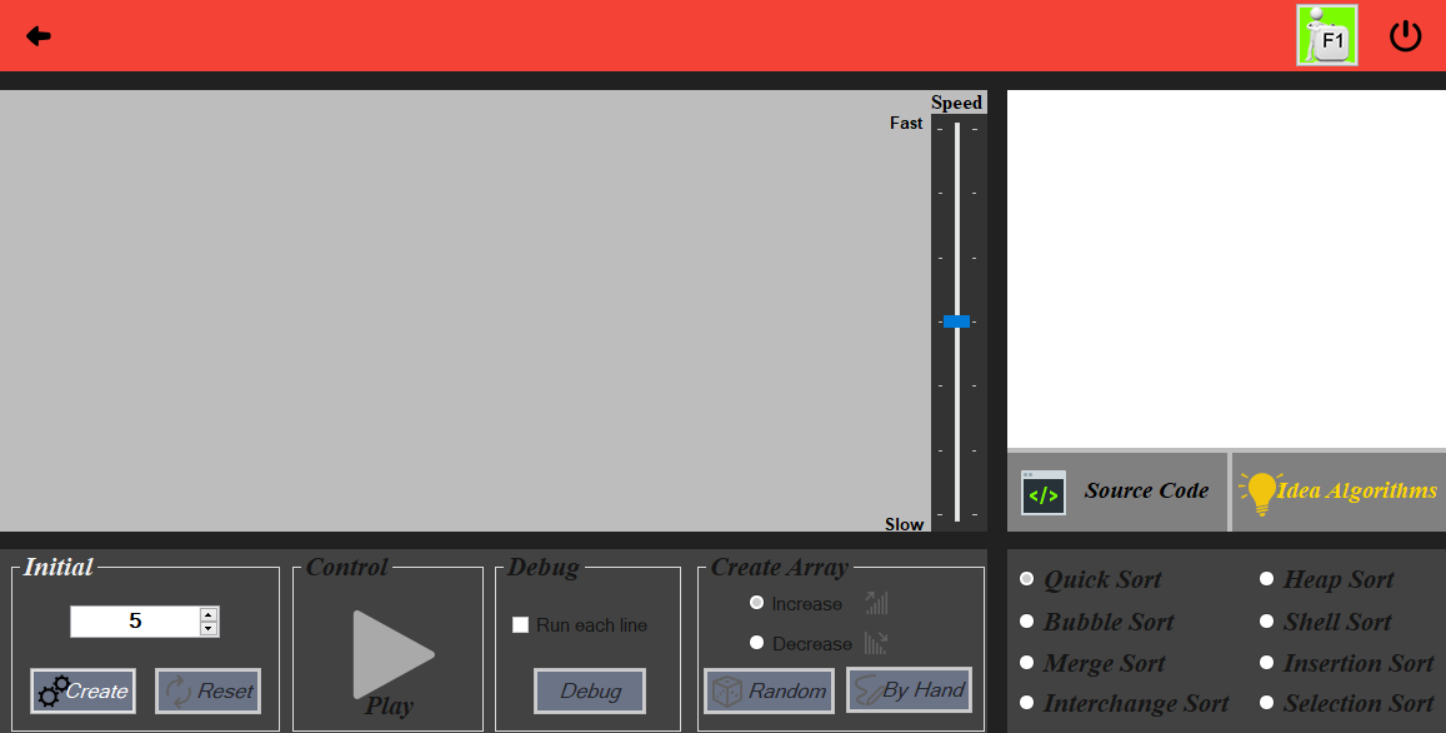
* Mô tả thành phần giao diện:
* Gồm có 4 Button, 1 PictureBox và 2 UserControl:
* Button hình mũi tên góc phải trên cùng có chức năng chuyển từ form frmExcecutors sang form frmIntroduct
* Button Home có chức năng hiển thị PictureBox khi có sự kiện Click
* Button Lecturers có chức năng hiển thị UserControl userLecturers khi có sự kiện Click



* Button Students có chức năng hiển thị UserControl userStudents khi có sự kiện Click



### Giao diện Ứng dụng Mô Phỏng (frmApplication)



* Mô tả thành phần giao diện:
* Gồm có 12 Button, 1 NumericUpDown, 1 CheckBox, 10 RadioButton, 2 ListBox, 1 TrackBar:
* Button Create có chức năng cho người dùng tạo số Node hiển thị lên màn hình chạy thử tương ứng với số ở NumericUpDown mà ta chọn
* Button Reset có chức năng cho người dùng đặt lại form frmApplication lại như ban đầu
* Button Random có chức năng cho người dùng nhập ngẫu nhiên số bất kì từ 0 tới 99 vào các Node mà ta đã tạo từ trước
* Button By Hand có chức năng cho người dùng mở form frmByHand và cho phép người dùng nhập từ bàn phím các số từ 0 tới 99
* RadioButton Increase và Decrease có chức năng cho người dùng chọn kiểu sắp xếp là tăng hoặc giảm
* Các RadioButton ở góc bên phải dưới cùng có chức năng cho người dùng chọn kiểu loại thuật toán muốn mô phỏng
* CheckBox Run ech line có chức năng cho người dùng mở tính năng chạy debug
* Button Debug có chức năng chạy từng lệnh
* Button Play có chức năng chạy mô phỏng các thao tác khởi tạo và chọn mà người dùng đã cài đặt

# Cài đặt

## Nền tảng công nghệ

* Phần mềm sử dụng: Visual Studio 2015.
* Ngôn ngữ: C# Window Form Application
* Các thư viện sử dụng: CodeC.cs; Idea.cs; Parameters.cs
* Các userControl được tạo: UserLectures.cs; UserStudents.cs

## Các hàm xử lý chính

### Node và các hàm có trong Node

* Để tạo ra và xử lý chạy trên màn hình của các Node ta dùng 1 mảng Label với các thuộc tính mà các Node chứa trong thư viện Parameters.cs

|  |
| --- |
| class Parameters  {  public int[] arrNode;  public Label[] arrLbl;  public Label[] node\_B, node\_C;  public int[] b, c;  public int disN; // Distance Node  public int nOe; // number of element  public int sizeN; // Size Node  public int firstN; // First Node  public int speed;  public bool increase;  public bool checkPause = false;  public Parameters(int[] iArr, Label[] lbArr, Label[] lbNb, Label[] lbNc, int[]  ib, int[] ic, bool bIncre)  {  arrNode = iArr;  arrLbl = lbArr;  node\_B = lbNb;  node\_C = lbNc; ;  b = ib;  c = ic;  increase = bIncre;  }  public Parameters() { }  } |

* Để xử lý việc nhập giá trị bằng tay cho Node ta cần các hàm chính sau:
* Hàm bntByHand\_Click: để mở form frmByHand và nhận dữ liệu nhập từ form đó

|  |
| --- |
| private void bntByHand\_Click(object sender, EventArgs e)  {  pnlChosesAlgorithms.Enabled = true;  grpControl.Enabled = true;  grpDebug.Enabled = true;  ShowCode();  frmByHand a = new frmByHand();  a.Message = numArray.Value.ToString();  a.ShowDialog();  for (int i = 0; i < thamSo.nOe; i++)  {  switch (i + 1)  {  case 1: thamSo.arrLbl[i].Text = PT1; thamSo.arrNode[i] = Int32.Parse(PT1); break;  case 2: thamSo.arrLbl[i].Text = PT2; thamSo.arrNode[i] = Int32.Parse(PT2); break;  case 3: thamSo.arrLbl[i].Text = PT3; thamSo.arrNode[i] = Int32.Parse(PT3); break;  case 4: thamSo.arrLbl[i].Text = PT4; thamSo.arrNode[i] = Int32.Parse(PT4); break;  case 5: thamSo.arrLbl[i].Text = PT5; thamSo.arrNode[i] = Int32.Parse(PT5); break;  case 6: thamSo.arrLbl[i].Text = PT6; thamSo.arrNode[i] = Int32.Parse(PT6); break;  case 7: thamSo.arrLbl[i].Text = PT7; thamSo.arrNode[i] = Int32.Parse(PT7); break;  case 8: thamSo.arrLbl[i].Text = PT8; thamSo.arrNode[i] = Int32.Parse(PT8); break;  case 9: thamSo.arrLbl[i].Text = PT9; thamSo.arrNode[i] = Int32.Parse(PT9); break;  case 10: thamSo.arrLbl[i].Text = PT10; thamSo.arrNode[i] = Int32.Parse(PT10); break;  }}  pnlChosesAlgorithms.Enabled = true;  grpControl.Enabled = true;  } |

* Hàm bntApply\_Click: để nhận giá trị từ các NumericUpDown rồi đưa giá trị lên các Node tương ứng

|  |
| --- |
| private void btApply\_Click(object sender, EventArgs e)  {  frmApplication.PT1 = pt1.Text;  frmApplication.PT2 = pt2.Text;  frmApplication.PT3 = pt3.Text;  frmApplication.PT4 = pt4.Text;  frmApplication.PT5 = pt5.Text;  frmApplication.PT6 = pt6.Text;  frmApplication.PT7 = pt7.Text;  frmApplication.PT8 = pt8.Text;  frmApplication.PT9 = pt9.Text;  frmApplication.PT10 = pt10.Text;  MessageBox.Show("OK", "", MessageBoxButtons.OK, MessageBoxIcon.Information);  this.Close();  } |

* Hàm bntBack\_Click: gán giá trị cho các Node là bằng 0

|  |
| --- |
| private void bntBack\_Click(object sender, EventArgs e)  {  string tempt = "0";  frmApplication.PT1 = tempt;  frmApplication.PT2 = tempt;  frmApplication.PT3 = tempt;  frmApplication.PT4 = tempt;  frmApplication.PT5 = tempt;  frmApplication.PT6 = tempt;  frmApplication.PT7 = tempt;  frmApplication.PT8 = tempt;  frmApplication.PT9 = tempt;  frmApplication.PT10 = tempt;  this.Dispose();  } |

* Để xử lý di chuyển các Node lên, xuống, trái, phải ta cần các hàm sau:
* Application.DoEvents() là để báo cho Proccess Winform không cần đợi nữa, cứ tiếp tục.
* this.Invoke((MethodInvoker)delegate : dòng bộ hóa tất cả các thread vs UI
* t.Refesh dùng để làm mơ
* Các hàm Node qua phải, trái, lên xuống là để thực hiện chỉ cho 1 Node.Nhìn chung khá là giống nhau chỉ khác hướng di chuyển. Thường dùng cho các thuật toán như InsersionSort, HeapSort,..
* Riêng hàm Swap thì thứ tự hơn, thực hiện cho 2 Node nên khi thỏa mãn 1 điều kiện nào nó là đổi chỗ cho nhau không cần xét gì thêm, nổi bật như thuật toán InterchangeSort
* Hàm Delay() : Là thay đổi tốc độ, nhanh or chậm được tính bằng millisecond. Các hàm di chuyển đều có Thread.Sleep(thamSo.speed); mặc định ban đầu thì đặt giá trị thamSo.speed = 60; thamSo.speed sẽ thay đổi theo giá trị mà TrackBar ngoài form thay đổi.

|  |
| --- |
| public void Delay(int milisecond)  {  //Nếu tốc độ max thì ko sleep  if (speedTrackBar.Value == speedTrackBar.Maximum)  {  Application.DoEvents();  return;  }  Application.DoEvents();  Thread.Sleep(milisecond);  } |

* Node qua phải : Dịch chuyển một hoặc nhiều Node đi qua phải

t.Left += 1 là di chuyển Node qua phải

|  |
| --- |
| public void Node\_Right(Control t, int Step)  {  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  int Loop\_Count = ((thamSo.sizeN + thamSo.disN)) \* Step; //Số lần dịch chuyển  {  while (Loop\_Count > 0)  {  Application.DoEvents();  t.Left += 1;  Delay(thamSo.speed);  Loop\_Count--;  t.BackColor = colorDefault;  }  }  t.Refresh();  });  } |

* Node qua trái : Dịch chuyển một hoặc nhiều Node đi qua trái

t.Left -= 1 là di chuyển Node qua trái

|  |
| --- |
| public void Node\_Left(Control t, int Step)  {  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  int Loop\_Count = ((thamSo.sizeN + thamSo.disN)) \* Step; //Số lần dịch chuyển  while (Loop\_Count > 0)  {  Application.DoEvents();  t.Left -= 1;  Delay(thamSo.speed);  Loop\_Count--;  t.BackColor = colorMove;  }  t.Refresh();  });  } |

* Node đi lên :

t.Top -= 1 là di chuyển Node lên trên

|  |
| --- |
| public void Node\_Up(Control t, int S)  {  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  int loop\_Count = S;  //t xuống  while (loop\_Count > 0)  {  Application.DoEvents();  t.Top -= 1;  Delay(thamSo.speed);  loop\_Count--;  t.BackColor = colorMove;  }  t.Refresh();  });  } |

* Node đi xuống

t.Top += 1 là di chuyển Node đi xuống

|  |
| --- |
| public void Node\_Down(Control t, int S)  {  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  int loop\_Count = S;  // t xuống  while (loop\_Count > 0)  {  Application.DoEvents();  t.Top += 1;  Delay(thamSo.speed);  loop\_Count--;  t.BackColor = colorMove;  }  t.Refresh();  });  } |

* Hoán vị
* Đặc trưng của hàm hoán vị là đổi chỗ 1 cách tuần tự,ví dụng Node1 qua trái và Node2 qua phải,Node 1 xuống và Node 2 lên,còn có hàm ý bao gồm nội dung các hàm trên nhưng được sắp xếp ban đầu và không thể thiếu 1 trong 4. Nếu thiếu thì các Node sẽ không đúng vị trí
* Các hàm chính dùng để swap các Node là Swap\_NodeAn, Swap\_Node và Swap\_Giatri
* Swap\_Node có chức năng thay đổi vị trí của 2 Node
* Swap\_NodeAn có chức năng bổ sung cho hàm Swap\_Node, hàm Swap\_Node chỉ thay đổi vị trí của các Node chứ không thay đổi giá trị nên cần có hàm Swap\_NodeAn để nó thay đổi giá trị của các Node
* Swap\_Giatri là hàm có kiểu int có chức năng swap các các mảng kiểu int

|  |
| --- |
| public void Swap\_NodeAn(int a, int b)  {  Label temp = thamSo.arrLbl[a];  thamSo.arrLbl[a] = thamSo.arrLbl[b];  thamSo.arrLbl[b] = temp;  }  public void Swap\_Node(Control t1, Control t2)  {  Application.DoEvents();  t1.BackColor = colorMove;  t2.BackColor = colorMove;  this.Invoke((MethodInvoker)delegate  {  Point p1 = t1.Location; //lưu vị trí ban đầu của t1  Point p2 = t2.Location; //lưu vị trí ban đầu của t2  if (p1 != p2)  {  // t1 lên, t2 xuống  while ((t1.Location.Y > p1.Y - (thamSo.sizeN + 5)) || (t2.Location.Y <  p2.Y + (thamSo.sizeN + 5)))  {  Application.DoEvents();  t1.Top -= 1;  t2.Top += 1;  Thread.Sleep(thamSo.speed);  }  // t1 dịch phải, t2 dịch trái  if (t1.Location.X < t2.Location.X)  {  while ((t1.Location.X < p2.X) || (t2.Location.X > p1.X))  {  Application.DoEvents();  t1.Left += 1;  t2.Left -= 1;  Thread.Sleep(thamSo.speed);  }  }  // t1 dịch trái, t2 dịch phải  else  {  while ((t1.Location.X > p2.X) || (t2.Location.X < p1.X))  {  Application.DoEvents();  t1.Left -= 1;  t2.Left += 1;  Thread.Sleep(thamSo.speed);  }  }  // t1 xuống, t2 lên  while ((t1.Location.Y < p2.Y) || (t2.Location.Y > p1.Y))  {  Application.DoEvents();  t1.Top += 1;  t2.Top -= 1;  Thread.Sleep(thamSo.speed);  }  t1.Refresh();  t2.Refresh();  }  });  }  public void Swap\_Giatri(ref int i, ref int j)  {  int Temp = i;  i = j;  j = Temp;  } |

* Hàm CompleteSwap: đây là hàm sau khi các Node swap xong sẽ có chức năng đổi màu lại các Node và đồng thời gọi lại hàm Complete

|  |
| --- |
| public void CompleteSwap()  {  thamSo.nOe = (int)numArray.Value;  for (int i = 0; i < thamSo.nOe; i++)  thamSo.arrLbl[i].BackColor = Color.Aqua;  Complete();  }  public void Complete()  {  ChonDongChoCodeListBox(0);  ExTime.Stop();  pnlExTime.Visible = false;  Init();  bntReset.Enabled = true;  bntCreate.Enabled = false;  bntExit.Enabled = true;  bntBack.Enabled = true;  Hidelbl();  MessageBox.Show("----------Sắp xếp hoàn tất----------\nThời gian thực thi là: " +  lblMinutes.Text + ":" + lblSeconds.Text, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);  } |

* Màu đặc trưng của các Node
* t1.BackColor = colorMove : đây là hàm thay đổi màu sắc khi Node đang di chuyển: để cho người dùng có thể dễ nhận biết hơn .
* Ngoài ra còn có các màu khác như colorDefault : màu mặc định, colorComplete: màu khi hoàn thành,… được gọi ở đầu
* 2 loại màu dưới cùng chỉ áp dụng cho thuật toán MergeSort

|  |
| --- |
| Color colorMove = Color.LightGreen;  Color colorComplete = Color.DarkViolet;  Color colorDefault = Color.Orange;  Color colorMergeU = Color.AliceBlue;  Color colorMergeD = Color.Aquamarine; |

### Hàm tạo mảng và xóa mảng

* Tạo mảng: bntCreate\_Click
* x.Enabled = true : Cho phép thực hiện các thao tác trên nó ( với x là bntCreate hoặc bntReset,…)
* x.Enabled = false : Không cho phép thực hiện các thao tác trên nó
* Khi numArray.Value < 2 thì báo rằng vi phạm lỗi ( 2->10 Node theo yêu cầu)
* Hàm switch case dùng để phân chia vị trí của các Node cho phù hợp với background
* Các thao tác khác được nói qua hình chụp

|  |
| --- |
| private void bntCreate\_Click(object sender, EventArgs e)  {  bntCreate.Enabled = false;  numArray.Enabled = false;  bntReset.Enabled = true;  thamSo.checkPause = false;  //DeleteArray(thamSo.arrLbl);  if (thamSo.nOe < 2 && thamSo.nOe > 10)  {  MessageBox.Show("Error", "Vui lòng nhập số phần tử tuwf 2->10 !",  MessageBoxButtons.OK, MessageBoxIcon.Error);  return;  }  else  {  switch(thamSo.nOe)  {  case 10:  case 9:  case 8:  case 7:  case 6:  thamSo.disN = 18;  thamSo.sizeN = 50;  thamSo.firstN = (1024 - thamSo.sizeN - thamSo.disN \* (thamSo.nOe - 1)) /  thamSo.nOe;  break;  case 5:  case 4:  case 3:  case 2:  thamSo.sizeN = 50;  thamSo.disN = 30;  thamSo.firstN = (1024 - thamSo.sizeN - thamSo.disN \* (thamSo.nOe - 1)) /  thamSo.nOe;  break;  }  thamSo.arrNode = new int[thamSo.nOe];  thamSo.arrLbl = new Label[thamSo.nOe];  for (int i = 0; i < thamSo.nOe; i++)  {  Label label = new Label();  label.AutoSize = false;  label.Size = new Size(40, 40);  label.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;  label.Font = new Font("Times New Roman", 14);  label.TextAlign = ContentAlignment.MiddleCenter;  label.Text = "0";  label.BackColor = colorDefault;  label.Location = new Point(thamSo.firstN + (thamSo.sizeN + thamSo.disN) \* i, 3 \*  thamSo.sizeN);  thamSo.arrLbl[i] = label;  pnlExcution.Controls.Add(thamSo.arrLbl[i]);  }  }  grpCreateArray.Enabled = true;  } |

* Xóa mảng:
* Gồm có 2 hàm chính là bntReset\_Click và DeleteArray
* Hàm bntReset\_Click có chức năng cài đặt lại tất cả những gì người dùng đã thiết lập trở lại mặc định ban đầu
* Hàm DeleteArray có chức năng xóa toàn bộ label trong mảng (Node) hiển có khỏi màng hình chạy mô phỏng và bổ trợ cho hàm bntReset\_Click

|  |
| --- |
| public void DeleteArray(Label[] a)  {  for (int i = 0; i < thamSo.nOe; i++)  {  this.Controls.Remove(a[i]);  }  }  private void bntReset\_Click(object sender, EventArgs e)  {  numArray.Enabled = true;  bntCreate.Enabled = true;  radSelection.Enabled = true;  radBubble.Enabled = true;  radShell.Enabled = true;  radQuick.Enabled = true;  radHeap.Enabled = true;  radInsertion.Enabled = true;  radInterchange.Enabled = true;  radMerge.Enabled = true;  Init();  Hidelbl();  for (int i = 0; i < thamSo.nOe; i++)  {  pnlExcution.Controls.Remove(thamSo.arrLbl[i]);  }  bntPlay.Show();  speedTrackBar.Value = 30;  }  public void Init()  {  grpCreateArray.Enabled = false;  pnlChosesAlgorithms.Enabled = false;  grpControl.Enabled = false;  grpDebug.Enabled = false;  thamSo.increase = true;  bntReset.Enabled = false;  listIdea.Hide();  } |

### Hàm chạy và tạm dừng việc sắp xếp

* Hàm Play\_Click có chức năng bắt đầu việc mô phỏng sắp xếp trên màn hình mô phỏng và bên cạnh đó người dùng sẽ không thể thực hiện thao tác lên một số Control
* Hàm Pause\_Click có chức năng tạm dừng việc mô phỏng sắp xếp trên màn hình mô phỏng
* Các hàm hỗ trợ cho hàm tạm dừng như Pause, Play\_or\_Stop
* Hàm Play\_or\_Stop có chức năng tạm dừng và kiểm tra xem cái checkpause = true hay ko nếu true thì nó chờ 1 sự kiện, nếu sự kiện đó mà ko thay đổi checkpause thì hàm vẫn sẽ chờ sự kiện tiếp, cho tới khi checkpause = false thì hàm sẽ thoát

|  |
| --- |
| private void bntPlay\_Click(object sender, EventArgs e)  {  bntPlay.Hide();  bntPause.Show();  pnlExTime.Visible = true;  ExTime.Start();  #region Enabled Control  bntBack.Enabled = false;  bntReset.Enabled = false;  bntExit.Enabled = false;  radSelection.Enabled = false;  radBubble.Enabled = false;  radShell.Enabled = false;  radQuick.Enabled = false;  radHeap.Enabled = false;  radInsertion.Enabled = false;  radInterchange.Enabled = false;  radMerge.Enabled = false;  #endregion  if (thamSo.checkPause == false)  {  if (radSelection.Checked == true && thamSo.increase == true)  SelectionSortincrease(thamSo.arrNode);  if (radSelection.Checked == true && thamSo.increase == false)  SelectionSortGiam(thamSo.arrNode);  if (radBubble.Checked == true && thamSo.increase == true)  BubbleSortincrease(thamSo.arrNode);  if (radBubble.Checked == true && thamSo.increase == false)  BubbleSortGiam(thamSo.arrNode);  if (radShell.Checked == true && thamSo.increase == true)  ShellSortincrease(thamSo.arrNode);  if (radShell.Checked == true && thamSo.increase == false)  ShellSortGiam(thamSo.arrNode);  if (radQuick.Checked == true && thamSo.increase == true)  QuickSortincrease(thamSo.arrNode, 0, thamSo.nOe - 1);  if (radQuick.Checked == true && thamSo.increase == false)  QuickSortGiam(thamSo.arrNode, 0, thamSo.nOe - 1);  if (radHeap.Checked == true && thamSo.increase == true)  HeapSortincrease(thamSo.nOe);  if (radHeap.Checked == true && thamSo.increase == false)  HeapSortGiam(thamSo.nOe);  if (radInsertion.Checked == true && thamSo.increase == true)  InsertionSortincrease(thamSo.arrNode);  if (radInsertion.Checked == true && thamSo.increase == false)  InsertionSortGiam(thamSo.arrNode);  if (radInterchange.Checked == true && thamSo.increase == true)  InterchangeSortincrease(thamSo.arrNode);  if (radInterchange.Checked == true && thamSo.increase == false)  InterchangeSortGiam(thamSo.arrNode);  if (radMerge.Checked == true && thamSo.increase == true)  MergeSortincrease();  if (radMerge.Checked == true && thamSo.increase == false)  MergeSortGiam();  CompleteSwap();  }  else  {  thamSo.checkPause = false;  }  }  private void bntPause\_Click(object sender, EventArgs e)  {  bntPlay.Show();  bntPause.Hide();  thamSo.checkPause = true;  Play\_or\_Stop();  }  public void Pause()  {  if (ckDebug.Checked == true)  {  thamSo.checkPause = true;  Play\_or\_Stop();  }  }  public void Play\_or\_Stop()  {  while (thamSo.checkPause == true)  {  Application.DoEvents();  }  } |

### Hàm chọn từng dòng Code trong khung listCode và chế độ Debug

* Hàm ChonDongChoCodeListBox: hàm nhận giá trị vị trí hàng được chọn trong khung Code C/C++ và chọn hàng đó. Hàm này giữ một ManualResetEvent giúp cho việc tạm dừng code và còn giúp cho việc Debug. Mỗi khi chạy chọn một lệnh nếu đang ở trong chế độ Debug thì cho dừng code lại đến khi nhấn nút Debug thì mới tiếp tục chạy.

|  |
| --- |
| public void ChonDongChoCodeListBox(int viTri)  {  Delay(thamSo.speed \* 5);  listCode.SelectedIndex = viTri;  // nếu đang trong chế độ Debug thì dừng sau mỗi câu lệnh chạy xong sẽ dừng lại  if (ckDebug.Checked)  {  thamSo.checkPause = true;  Play\_or\_Stop();  }  } |

### Các hàm sắp xếp mảng

* 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 Node khi chạy

|  |
| --- |
| private void InterchangeSortincrease(int[] arrNode)  {  ChonDongChoCodeListBox(2);  int i = 0, j = 0;  Mui\_ten\_xanh\_len\_1.Text = "i = " + i;  Mui\_ten\_xanh\_len\_1.Location = new Point(thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i, pnlExcution.Height - 50);  Mui\_ten\_xanh\_len\_1.Visible = true;  Refresh();  for (i = 0; i < thamSo.nOe - 1; i++)  {  ChonDongChoCodeListBox(3);  j = i + 1;  Mui\_ten\_xanh\_len\_2.Text = "j = " + j;  Mui\_ten\_xanh\_len\_2.Location = new Point(thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j, pnlExcution.Height - 50);  Mui\_ten\_xanh\_len\_2.Visible = true;  Refresh();  for (j = i + 1; j < thamSo.nOe; j++)  {  ChonDongChoCodeListBox(4);  lbl\_status\_02.Text = "SoSanh(a[" + i + "],a[" + j + "])";  lbl\_status\_02.Visible = true;  if (thamSo.arrNode[j] < thamSo.arrNode[i])  {  ChonDongChoCodeListBox(6);  lbl\_status\_02.Text = "HoanVi(a[" + i + "],a[" + j + "])";  Swap\_Giatri(ref thamSo.arrNode[j], ref thamSo.arrNode[i]);  Swap\_Node(thamSo.arrLbl[j], thamSo.arrLbl[i]);  Swap\_NodeAn(j, i);  thamSo.arrLbl[i].BackColor = colorComplete;  }  if (j + 1 < thamSo.nOe)  {  Mui\_ten\_xanh\_len\_2.Text = "j = " + (j + 1);  Mui\_ten\_xanh\_len\_2.Location = new Point(thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* (j + 1), pnlExcution.Height - 50);  Refresh();  }  thamSo.arrLbl[i].BackColor = colorDefault;  }  ChonDongChoCodeListBox(3);  lbl\_status\_02.Visible = false;  Mui\_ten\_xanh\_len\_1.Text = "i = " + (i + 1);  Mui\_ten\_xanh\_len\_1.Location = new Point(thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* (i + 1), pnlExcution.Height - 50);  Refresh();  }  ChonDongChoCodeListBox(0);  }  private void InterchangeSortGiam(int[] arrNode)  {  ChonDongChoCodeListBox(2);  int i = 0, j = 0;  Mui\_ten\_xanh\_len\_1.Text = "i = " + i;  Mui\_ten\_xanh\_len\_1.Location = new Point(thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i, pnlExcution.Height - 50);  Mui\_ten\_xanh\_len\_1.Visible = true;  for (i = 0; i < thamSo.nOe - 1; i++)  {  ChonDongChoCodeListBox(3);  j = i + 1;  Mui\_ten\_xanh\_len\_2.Text = "j = " + j;  Mui\_ten\_xanh\_len\_2.Location = new Point(thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j, pnlExcution.Height - 50);  Mui\_ten\_xanh\_len\_2.Visible = true;  for (j = i + 1; j < thamSo.nOe; j++)  {  ChonDongChoCodeListBox(4);  lbl\_status\_02.Text = "SoSanh(a[" + i + "],a[" + j + "])";  lbl\_status\_02.Visible = true;  if (thamSo.arrNode[j] > thamSo.arrNode[i])  {  ChonDongChoCodeListBox(6);  lbl\_status\_02.Text = "HoanVi(a[" + i + "],a[" + j + "])";  Swap\_Giatri(ref thamSo.arrNode[i], ref thamSo.arrNode[j]);  Swap\_Node(thamSo.arrLbl[j], thamSo.arrLbl[i]);  Swap\_NodeAn(j, i);  thamSo.arrLbl[i].BackColor = colorComplete;  }  if (j + 1 < thamSo.nOe)  {  Mui\_ten\_xanh\_len\_2.Text = "j = " + (j + 1);  Mui\_ten\_xanh\_len\_2.Location = new Point(thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* (j + 1), pnlExcution.Height - 50);  Refresh();  }  thamSo.arrLbl[i].BackColor = colorDefault;  }  ChonDongChoCodeListBox(3);  lbl\_status\_02.Visible = false;  Mui\_ten\_xanh\_len\_1.Text = "i = " + (i + 1);  Mui\_ten\_xanh\_len\_1.Location = new Point(thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* (i + 1), pnlExcution.Height - 50);  }  ChonDongChoCodeListBox(0);  } |

* Phần Code và Ý tưởng thuật toán được thể hiện vào khung listCode và listIdea

|  |
| --- |
| public void insertionsort(ListBox lst\_Code, Boolean tang)  {  lst\_Code.Items.Add("void InsertionSort(int a[], int n)");  lst\_Code.Items.Add(" {");  lst\_Code.Items.Add(" int i, pos, x;");  lst\_Code.Items.Add(" for (i = 1; i < n; i++)");  lst\_Code.Items.Add(" {");  lst\_Code.Items.Add(" x = a[i];");  lst\_Code.Items.Add(" pos = i - 1;");  if (tang)  lst\_Code.Items.Add(" while ((pos >= 0) && (a[pos] > x))");  else  lst\_Code.Items.Add(" while ((pos >= 0) && (a[pos] < x))");  lst\_Code.Items.Add(" {");  lst\_Code.Items.Add(" a[pos + 1] = a[pos];");  lst\_Code.Items.Add(" pos--;");  lst\_Code.Items.Add(" }");  lst\_Code.Items.Add(" a[pos + 1] = x;");  lst\_Code.Items.Add(" }");  lst\_Code.Items.Add(" }");  }  public void Selectionsort(ListBox lst\_Code)  {  lst\_Code.Items.Add("- Chọn phần tử nhỏ nhất(Lớn nhất) trong N phần tử trong dãy hiện  hành ban đầu");  lst\_Code.Items.Add("- Đưa phần tử này về vị trí đầu dãy hiện hành");  lst\_Code.Items.Add("- Xem dãy hiện hành chỉ còn lại N-1 phần tử của dãy hiện hành ban  đầu");  lst\_Code.Items.Add(" + Bắt đầu từ vị trí thứ 2");  lst\_Code.Items.Add(" + Lặp lại quá trình trên dãy cho hiện hành... đến khi dãy hiện  hành chỉ còn 1 phần tử");  } |

* Hàm SelectionSort

|  |
| --- |
| private void SelectionSortincrease(int[] arrNode)  {  int min;  this.Invoke((MethodInvoker)delegate  {  for (int i = 0; i < thamSo.nOe - 1; i++)  {  ChonDongChoCodeListBox(3);  ChonDongChoCodeListBox(5);  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Refresh();  min = i;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Text = "Min=" + min;  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* min) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[min].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Refresh();  for (int j = i + 1; j < thamSo.nOe; j++)  {  ChonDongChoCodeListBox(6);  Application.DoEvents();  ChonDongChoCodeListBox(7);  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "SoSanh( a[min] , a[" + j + "] )";  lbl\_status\_02.Refresh();  if (thamSo.arrNode[j] < thamSo.arrNode[min])  {  ChonDongChoCodeListBox(8);  min = j;  Mui\_ten\_xanh\_len\_1.Text = "Min=" + min;  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* min) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Refresh();  }  }  if (min != i)  {  ChonDongChoCodeListBox(10);  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "HoanVi( a[i] , a[min] )";  Swap\_Giatri(ref thamSo.arrNode[min], ref thamSo.arrNode[i]);  Swap\_Node(thamSo.arrLbl[min], thamSo.arrLbl[i]);  Swap\_NodeAn(min, i);  thamSo.arrLbl[i].BackColor = colorComplete;  thamSo.arrLbl[min].BackColor = colorDefault;  }  else  thamSo.arrLbl[i].BackColor = colorComplete;  lbl\_status\_02.Visible = false;  }  });  ChonDongChoCodeListBox(0);  }  private void SelectionSortGiam(int[] arrNode)  {  int max;  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  for (int i = 0; i < thamSo.nOe - 1; i++)  {  ChonDongChoCodeListBox(3);  ChonDongChoCodeListBox(5);  max = i;  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Refresh();  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Text = "Max=" + max;  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* max) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[max].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Refresh();  for (int j = i + 1; j < thamSo.nOe; j++)  {  ChonDongChoCodeListBox(6);  Application.DoEvents();  ChonDongChoCodeListBox(7);  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "SoSanh( a[max] , a[" + j + "] )";  if (thamSo.arrNode[j] > thamSo.arrNode[max])  {  max = j;  Mui\_ten\_xanh\_len\_1.Text = "Max=" + max;  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* max) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Refresh();  }  }  if (max != i)  {  ChonDongChoCodeListBox(10);  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "HoanVi( a[i] , a[max] )";  Swap\_Giatri(ref thamSo.arrNode[max], ref thamSo.arrNode[i]);  System.Threading.Thread.Sleep(5);  Swap\_Node(thamSo.arrLbl[max], thamSo.arrLbl[i]);  Swap\_NodeAn(max, i);  thamSo.arrLbl[i].BackColor = colorComplete;  thamSo.arrLbl[max].BackColor = colorDefault;  }  else  thamSo.arrLbl[i].BackColor = colorComplete;  lbl\_status\_02.Visible = false;  }  });  ChonDongChoCodeListBox(0);  } |

* Hàm BubbleSort

|  |
| --- |
| private void BubbleSortincrease(int[] arrNode)  {  int i, j;  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  for (i = 0; i < thamSo.nOe - 1; i++)  {  ChonDongChoCodeListBox(3);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Refresh();  Application.DoEvents();  for (j = thamSo.nOe - 1; j > i; j--)  {  Application.DoEvents();  ChonDongChoCodeListBox(4);  Mui\_ten\_xanh\_len\_1.Text = "j=" + j;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Refresh();  lbl\_status\_02.Visible = false;  lbl\_status\_02.Text = "So\_Sanh(a[" + j + "],a[" + (j - 1) + "])";  lbl\_status\_02.Refresh();  ChonDongChoCodeListBox(5);  lbl\_status\_02.Visible = true;  if (thamSo.arrNode[j] < thamSo.arrNode[j - 1])  {  //Hieu ung xem code  ChonDongChoCodeListBox(6);  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "Hoan\_vi(a[" + j + "],a[" + (j - 1) + "])";  lbl\_status\_02.Refresh();  Swap\_Giatri(ref thamSo.arrNode[j], ref thamSo.arrNode[j - 1]);  Swap\_Node(thamSo.arrLbl[j], thamSo.arrLbl[j - 1]);  Swap\_NodeAn(j, j - 1);  thamSo.arrLbl[j - 1].BackColor = colorComplete;  }  thamSo.arrLbl[j].BackColor = colorDefault;  }  //Cập nhật Status  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "a[" + i + "] Đã đúng vị trí";  lbl\_status\_02.Refresh();  }  lbl\_status\_02.Visible = false;  });  ChonDongChoCodeListBox(0);  }  private void BubbleSortGiam(int[] arrNode)  {  int i, j;  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  for (i = 0; i < thamSo.nOe - 1; i++)  {  ChonDongChoCodeListBox(3);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Refresh();  Application.DoEvents();  for (j = thamSo.nOe - 1; j > i; j--)  {  Application.DoEvents();  ChonDongChoCodeListBox(4);  Mui\_ten\_xanh\_len\_1.Text = "j=" + j;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Refresh();  lbl\_status\_02.Visible = false;  lbl\_status\_02.Text = "So\_Sanh(a[" + j + "],a[" + (j - 1) + "])";  lbl\_status\_02.Refresh();  ChonDongChoCodeListBox(5);  lbl\_status\_02.Visible = true;  if (thamSo.arrNode[j] > thamSo.arrNode[j - 1])  {  //Hieu ung xem code  ChonDongChoCodeListBox(6);  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "Hoan\_vi(a[" + j + "],a[" + (j - 1) + "])";  lbl\_status\_02.Refresh();  Swap\_Giatri(ref thamSo.arrNode[j], ref thamSo.arrNode[j - 1]);  Swap\_Node(thamSo.arrLbl[j], thamSo.arrLbl[j - 1]);  Swap\_NodeAn(j, j - 1);  thamSo.arrLbl[j - 1].BackColor = colorComplete;  }  thamSo.arrLbl[j].BackColor = colorDefault;  }  //Cập nhật Status  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "a[" + i + "] Đã đúng vị trí";  lbl\_status\_02.Refresh();  }  lbl\_status\_02.Visible = false;  });  ChonDongChoCodeListBox(0);  } |

* Hàm ShellSort

|  |
| --- |
| private void ShellSortincrease(int[] arrNode)  {  for (int i = thamSo.nOe / 2; i > 0; i /= 2)  {  ChonDongChoCodeListBox(2);  Mui\_ten\_xanh\_xuong\_1.Text = "i = " + i;  Mui\_ten\_xanh\_xuong\_1.Location = new Point(thamSo.firstN - 10 + (thamSo.sizeN +  thamSo.disN) \* 0, 40);  Mui\_ten\_xanh\_xuong\_1.SendToBack();  Mui\_ten\_xanh\_xuong\_1.Visible = true;  for (int j = i; j < thamSo.nOe; j++)  {  ChonDongChoCodeListBox(3);  Mui\_ten\_xanh\_len\_1.Text = "j = " + j;  Mui\_ten\_xanh\_len\_1.Location = new Point(thamSo.firstN - 10 + (thamSo.sizeN +  thamSo.disN) \* j, 40);  Mui\_ten\_xanh\_len\_1.Visible = true;  for (int k = j; k >= i && thamSo.arrNode[k] < thamSo.arrNode[k - i]; k -= i)  {  ChonDongChoCodeListBox(4);  Mui\_ten\_xanh\_xuong\_2.Text = "k = " + k;  Mui\_ten\_xanh\_xuong\_2.Location = new Point(thamSo.firstN - 10 +  (thamSo.sizeN + thamSo.disN) \* k, pnlExcution.Height - 50);  Mui\_ten\_xanh\_xuong\_2.SendToBack();  Mui\_ten\_xanh\_xuong\_2.Visible = true;  Refresh();  ChonDongChoCodeListBox(5);  lbl\_status\_02.Text = "HoanVi(a[" + k + "],a[" + (k - 1) + "])";  lbl\_status\_02.Visible = true;  Swap\_Giatri(ref thamSo.arrNode[k - i], ref thamSo.arrNode[k]);  Swap\_Node(thamSo.arrLbl[k], thamSo.arrLbl[k - i]);  Swap\_NodeAn(k, k - i);  thamSo.arrLbl[k - i].BackColor = colorDefault;  thamSo.arrLbl[k].BackColor = colorDefault;  }  lbl\_status\_02.Visible = false;  ChonDongChoCodeListBox(3);  }  }  ChonDongChoCodeListBox(0);  }  private void ShellSortGiam(int[] arrNode)  {  for (int i = thamSo.nOe / 2; i > 0; i /= 2)  {  ChonDongChoCodeListBox(2);  Mui\_ten\_xanh\_xuong\_1.Text = "i = " + i;  Mui\_ten\_xanh\_xuong\_1.Location = new Point(thamSo.firstN - 10 + (thamSo.sizeN +  thamSo.disN) \* 0, 40);  Mui\_ten\_xanh\_xuong\_1.SendToBack();  Mui\_ten\_xanh\_xuong\_1.Visible = true;  for (int j = i; j < thamSo.nOe; j++)  {  ChonDongChoCodeListBox(3);  Mui\_ten\_xanh\_len\_1.Text = "j = " + j;  Mui\_ten\_xanh\_len\_1.Location = new Point(thamSo.firstN - 10 + (thamSo.sizeN +  thamSo.disN) \* j, 40);  Mui\_ten\_xanh\_len\_1.Visible = true;  for (int k = j; k >= i && thamSo.arrNode[k] > thamSo.arrNode[k - i]; k -= i)  {  ChonDongChoCodeListBox(4);  Mui\_ten\_xanh\_xuong\_2.Text = "k = " + k;  Mui\_ten\_xanh\_xuong\_2.Location = new Point(thamSo.firstN - 10 +  (thamSo.sizeN + thamSo.disN) \* k, pnlExcution.Height - 50);  Mui\_ten\_xanh\_xuong\_2.SendToBack();  Mui\_ten\_xanh\_xuong\_2.Visible = true;  Refresh();  ChonDongChoCodeListBox(5);  lbl\_status\_02.Text = "HoanVi(a[" + k + "],a[" + (k - 1) + "])";  lbl\_status\_02.Visible = true;  Swap\_Giatri(ref thamSo.arrNode[k - i], ref thamSo.arrNode[k]);  System.Threading.Thread.Sleep(5);  Swap\_Node(thamSo.arrLbl[k], thamSo.arrLbl[k - i]);  Swap\_NodeAn(k, k - i);  thamSo.arrLbl[k - i].BackColor = colorDefault;  thamSo.arrLbl[k].BackColor = colorDefault;  }  lbl\_status\_02.Visible = false;  ChonDongChoCodeListBox(3);  }  }  ChonDongChoCodeListBox(0);  } |

* Hàm InsertSort

|  |
| --- |
| private void InsertionSortincrease(int[] arrNode)  {  int i, pos, x;  Label Node\_tam;  int Chi\_so\_tam;  int Dem\_node;  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  for (i = 1; i < thamSo.nOe; i++)  {  ChonDongChoCodeListBox(3);  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Refresh();  Application.DoEvents();  Dem\_node = 0;  ChonDongChoCodeListBox(5);  x = thamSo.arrNode[i];  Node\_tam = thamSo.arrLbl[i];  Chi\_so\_tam = i;  pos = i - 1;  //thiết lập mũi tên đánh dấu nút cần chèn  //Di chuyển Node cần chèn lên  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Node\_Up(Node\_tam, (thamSo.sizeN + 5));  });  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "So\_Sanh(a[" + pos + "],a[i)";  //Thiết lập bàn tay chỉ vi trí có phải vị trí cần chèn không  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* pos) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[pos].Location.Y + thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Text = "POS=" + pos;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Refresh();  //lbl\_status\_02.Visible = false  while ((pos >= 0) && (thamSo.arrNode[pos] > x))  {  Application.DoEvents();  ChonDongChoCodeListBox(7);  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* pos) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[pos].Location.Y + thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Text = "POS=" + pos;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Refresh();  //Thiết lập bàn tay chỉ vi trí có phải vị trí cần chèn không  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "So\_Sanh(a[" + pos + "],a[i])";  ChonDongChoCodeListBox(9);  //Dịch chuyển Node qua phải  thamSo.arrNode[pos + 1] = thamSo.arrNode[pos];  Dem\_node++;  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Node\_Right(thamSo.arrLbl[pos], 1);  });  Swap\_NodeAn(pos + 1, pos);  ChonDongChoCodeListBox(10);  pos--;  ChonDongChoCodeListBox(12);  thamSo.arrNode[pos + 1] = x;  }  //status hoán vị  if (Dem\_node > 0)  {  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "Hoan\_vi(a[pos],a[i])";  }  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Node\_Left(Node\_tam, Dem\_node);  thamSo.arrLbl[pos + 1].BackColor = colorDefault;  });  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Node\_Down(Node\_tam, (thamSo.sizeN + 5));  //thamSo.arrLbl[pos+1].BackColor = colorComplete;  thamSo.arrLbl[pos + 1].BackColor = colorDefault;  });  lbl\_status\_02.Visible = false;  Mui\_ten\_xanh\_len\_1.Visible = false; ;  thamSo.arrLbl[pos + 1] = Node\_tam;  }  });  ChonDongChoCodeListBox(0);  }  private void InsertionSortGiam(int[] arrNode)  {  int i, pos, x;  Label Node\_tam;  int Chi\_so\_tam;  int Dem\_node;  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  for (i = 1; i < thamSo.nOe; i++)  {  ChonDongChoCodeListBox(3);  Application.DoEvents();  //Thiết lập Node đầu tiên, là Node đã có thứ tự  //đềm số bước dịch chuyển 1 Node  Dem\_node = 0;  ChonDongChoCodeListBox(5);  x = thamSo.arrNode[i];  Node\_tam = thamSo.arrLbl[i];  Chi\_so\_tam = i;  pos = i - 1;  //thiết lập mũi tên đánh dấu nút cần chèn  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Refresh();  //Di chuyển Node cần chèn lên  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Node\_Up(Node\_tam, (thamSo.sizeN + 5));  });  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "So\_Sanh(a[" + pos + "],a[i])";  //Thiết lập bàn tay chỉ vi trí có phải vị trí cần chèn không  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* pos) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[pos].Location.Y + thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Text = "POS=" + pos;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Refresh();  //lbl\_status\_02.Visible = false;  while ((pos >= 0) && (thamSo.arrNode[pos] < x))  {  Application.DoEvents();  ChonDongChoCodeListBox(7);  //Thiết lập bàn tay chỉ vi trí có phải vị trí cần chèn không  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* pos) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[pos].Location.Y + thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Text = "POS=" + pos;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Refresh();  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "So\_Sanh(a[" + pos + "],a[i])";  ChonDongChoCodeListBox(9);  //Dịch chuyển Node qua phải  thamSo.arrNode[pos + 1] = thamSo.arrNode[pos];  Dem\_node++;  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Node\_Right(thamSo.arrLbl[pos], 1);  });  Swap\_NodeAn(pos + 1, pos);  thamSo.arrLbl[pos].BackColor = colorComplete;  ChonDongChoCodeListBox(10);  pos--;  ChonDongChoCodeListBox(12);  thamSo.arrNode[pos + 1] = x;  }  //status hoán vị  if (Dem\_node > 0)  {  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "Hoan\_vi(a[pos],a[i])";  }  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Node\_Left(Node\_tam, Dem\_node);  thamSo.arrLbl[pos + 1].BackColor = colorDefault;  });  Application.DoEvents();  Delay(thamSo.speed \* 30);  this.Invoke((MethodInvoker)delegate  {  Node\_Down(Node\_tam, (thamSo.sizeN + 5));  thamSo.arrLbl[pos + 1].BackColor = colorDefault;  });  //Ẩn status  lbl\_status\_02.Visible = false;  //Ẩn mũi tên POS sau khi đã tìm ra POS  Mui\_ten\_xanh\_len\_1.Visible = false; ;  //Thiết lập node nằm trong nhóm đã có thứ tự  thamSo.arrLbl[pos + 1] = Node\_tam;  //Tạm dừng sau 1 bước dịch chuyển Node  }  });  ChonDongChoCodeListBox(0);  } |

* Hàm QuickSort

|  |
| --- |
| private void QuickSortincrease(int[] arrNode, int left, int right)  {  if (left < right)  {  int pivot = thamSo.arrNode[(left + right) / 2];  int i = left, j = right;  int cs\_x = (left + right) / 2;  //đặt mũi tên chỉ left  ChonDongChoCodeListBox(2);  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* left) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[left].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Text = "L = " + left;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Refresh();  //đặt mũi tên chỉ Right  Mui\_ten\_xanh\_len\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* right) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[right].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_2.Text = "R = " + right;  Mui\_ten\_xanh\_len\_2.Visible = true;  Mui\_ten\_xanh\_len\_2.Refresh();  //doi mau node x  //Đặt màu nút x  ChonDongChoCodeListBox(3);  //Thiết lập vị trí của x  Mui\_ten\_do\_len.Visible = true;  Mui\_ten\_do\_len.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* cs\_x) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[cs\_x].Location.Y + 2 \* thamSo.sizeN + 40);  Mui\_ten\_do\_len.Text = "X";  Mui\_ten\_do\_len.Refresh();  //Thiết lập mũi tên chỉ i  ChonDongChoCodeListBox(4);  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Refresh();  //Thiết lập mũi tên chỉ j  Mui\_ten\_xanh\_xuong\_2.Visible = true;  Mui\_ten\_xanh\_xuong\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_2.Text = "j=" + j;  Mui\_ten\_xanh\_xuong\_2.Refresh();  ChonDongChoCodeListBox(5);  do  {  ChonDongChoCodeListBox(7);  lbl\_status\_02.Text = "SoSanh(a[" + i + "], a[x])";  lbl\_status\_02.Visible = true;  while (thamSo.arrNode[i] < pivot)  {  i++;  //Thiết lập mũi tên chỉ i  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Refresh();  //Hiệu ứng so sánh  // lbl\_status\_02.Text = "SoSanh(a[" + i + "], a[x])";  }  ChonDongChoCodeListBox(8);  lbl\_status\_02.Text = "SoSanh(a[" + j + "], a[x])";  while (thamSo.arrNode[j] > pivot)  {  j--;  //Thiết lập mũi tên chỉ j  Mui\_ten\_xanh\_xuong\_2.Visible = true;  Mui\_ten\_xanh\_xuong\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_2.Text = "j=" + j;  Mui\_ten\_xanh\_xuong\_2.Refresh();  }  ChonDongChoCodeListBox(9);  if (i <= j)  {  ChonDongChoCodeListBox(11);  lbl\_status\_02.Text = "HoanVi(a[" + i + "], a[" + j + "])";  Swap\_Giatri(ref thamSo.arrNode[i], ref thamSo.arrNode[j]);  if (i == cs\_x)  {  cs\_x = j;  }  else if (j == cs\_x)  {  cs\_x = i;  }  Swap\_Node(thamSo.arrLbl[j], thamSo.arrLbl[i]);  Swap\_NodeAn(j, i);  thamSo.arrLbl[j].BackColor = colorComplete;  thamSo.arrLbl[i].BackColor = colorDefault;  Mui\_ten\_do\_len.Visible = true;  Mui\_ten\_do\_len.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* cs\_x) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[cs\_x].Location.Y + 2 \* thamSo.sizeN + 40);  Mui\_ten\_do\_len.Text = "X = " + ((left + right) / 2);  Mui\_ten\_do\_len.Refresh();  ChonDongChoCodeListBox(12);  i++;  j--;  }  }  while (i <= j);  ChonDongChoCodeListBox(16);  if (left < j)  {  ChonDongChoCodeListBox(17);  QuickSortincrease(thamSo.arrNode, left, j);  }  ChonDongChoCodeListBox(18);  if (i < right)  {  ChonDongChoCodeListBox(19);  QuickSortincrease(thamSo.arrNode, i, right);  }  }  }  private void QuickSortGiam(int[] arrNode, int left, int right)  {  if (left < right)  {  int pivot = thamSo.arrNode[(left + right) / 2];  int i = left, j = right;  int cs\_x = (left + right) / 2;  //đặt mũi tên chỉ left  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* left) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[left].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Text = "L = " + left;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Refresh();  //đặt mũi tên chỉ Right  Mui\_ten\_xanh\_len\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* right) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[right].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_2.Text = "R = " + right;  Mui\_ten\_xanh\_len\_2.Visible = true;  Mui\_ten\_xanh\_len\_2.Refresh();  //doi mau node x  ChonDongChoCodeListBox(2);  cs\_x = (left + right) / 2;  //Thiết lập vị trí của x  Mui\_ten\_do\_len.Visible = true;  Mui\_ten\_do\_len.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* cs\_x) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[cs\_x].Location.Y + 2 \* thamSo.sizeN + 40);  Mui\_ten\_do\_len.Text = "X";  Mui\_ten\_do\_len.Refresh();  //Đặt màu nút x  i = left; j = right;  ChonDongChoCodeListBox(3);  //Thiết lập mũi tên chỉ i  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Refresh();  //Thiết lập mũi tên chỉ j  Mui\_ten\_xanh\_xuong\_2.Visible = true;  Mui\_ten\_xanh\_xuong\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_2.Text = "j=" + j;  Mui\_ten\_xanh\_xuong\_2.Refresh();  ChonDongChoCodeListBox(5);  do  {  //Hiệu ứng so sánh  lbl\_status\_02.Text = "So\_Sanh(a[" + i + "], a[x])";  ChonDongChoCodeListBox(7);  while (thamSo.arrNode[i] > pivot)  {  i++;  //Thiết lập mũi tên chỉ i  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Refresh();  //Hiệu ứng so sánh  lbl\_status\_02.Text = "So\_Sanh(a[" + i + "], a[x])";  }  //Hiệu ứng so sánh  lbl\_status\_02.Text = "So\_Sanh(a[" + j + "], a[x])";  ChonDongChoCodeListBox(8);  while (thamSo.arrNode[j] < pivot)  {  j--;  //Thiết lập mũi tên chỉ j  Mui\_ten\_xanh\_xuong\_2.Visible = true;  Mui\_ten\_xanh\_xuong\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_2.Text = "j=" + j;  Mui\_ten\_xanh\_xuong\_2.Refresh();  //Hiệu ứng so sánh  lbl\_status\_02.Text = "So\_Sanh(a[" + j + "], a[x])";  }  ChonDongChoCodeListBox(9);  if (i <= j)  {  //status hoán vị  lbl\_status\_02.Text = "Hoan\_Vi(a[" + i + "], a[" + j + "])";  Swap\_Giatri(ref thamSo.arrNode[i], ref thamSo.arrNode[j]);  ChonDongChoCodeListBox(11);  //Tìm vị trí mới của x  if (i == cs\_x)  {  cs\_x = j;  }  else if (j == cs\_x)  {  cs\_x = i;  }  Swap\_Node(thamSo.arrLbl[i], thamSo.arrLbl[j]);  Swap\_NodeAn(i, j);  thamSo.arrLbl[i].BackColor = colorComplete;  thamSo.arrLbl[j].BackColor = colorDefault;  //Thiết lập vị trí của  Mui\_ten\_do\_len.Visible = true;  Mui\_ten\_do\_len.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* cs\_x) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[cs\_x].Location.Y + 2 \* thamSo.sizeN + 40);  Mui\_ten\_do\_len.Text = "X = " + ((left + right) / 2);  Mui\_ten\_do\_len.Refresh();  ChonDongChoCodeListBox(12);  i++;  j--;  }  } while (i <= j);  //Đặt màu nút x  ChonDongChoCodeListBox(16);  if (left < j)  {  ChonDongChoCodeListBox(18);  QuickSortGiam(thamSo.arrNode, left, j);  }  if (i < right)  {  ChonDongChoCodeListBox(19);  QuickSortGiam(thamSo.arrNode, i, right);  }  }  } |

* Hàm HeapSort

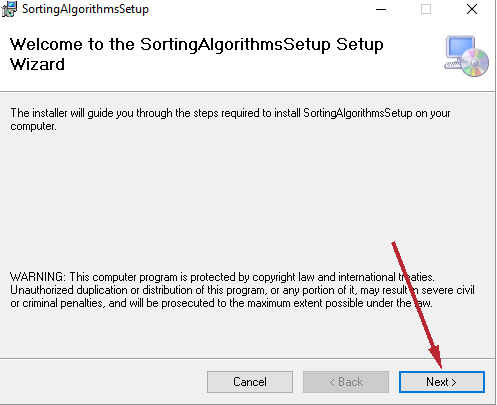
|  |
| --- |
| void Shift\_increase(int l, int r)  {  int i, j, index\_temp, x;  Label temp;  i = l;  j = 2 \* i + 1;  //Thiết lập mũi tên chỉ i  lbl\_status\_02.Text = "Đang shift heap";  lbl\_status\_02.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Refresh();  //thiết lập mũi tên chỉ j  Mui\_ten\_xanh\_xuong\_2.Visible = true;  Mui\_ten\_xanh\_xuong\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_2.Text = "j=" + j;  Mui\_ten\_xanh\_xuong\_2.Refresh();  x = thamSo.arrNode[i];  temp = thamSo.arrLbl[i];  index\_temp = i;  while (j <= r)  {  Application.DoEvents();  if (j < r)  if (thamSo.arrNode[j] < thamSo.arrNode[j + 1])  j++;  //thiết lập mũi tên chỉ j  Mui\_ten\_xanh\_xuong\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_2.Text = "j=" + j;  Mui\_ten\_xanh\_xuong\_2.Refresh();  if (thamSo.arrNode[j] <= x)  return;  else  {  thamSo.arrNode[i] = thamSo.arrNode[j];  thamSo.arrNode[j] = x;  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Swap\_Node(thamSo.arrLbl[j], temp);  });  Swap\_NodeAn(j, index\_temp);  if (ckDebug.Checked == true)  Pause();  else  Delay(5 \* thamSo.speed);  i = j;  j = 2 \* i + 1;  if (j <= r)  {  //Thiết lập mũi tên chỉ i  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Refresh();  //thiết lập mũi tên chỉ j  Mui\_ten\_xanh\_xuong\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_2.Text = "j=" + j;  Mui\_ten\_xanh\_xuong\_2.Refresh();  }  x = thamSo.arrNode[i];  temp = thamSo.arrLbl[i];  index\_temp = i;  }  }  }  void Shift\_giam(int l, int r)  {  int i, j, index\_temp, x;  Label temp;  i = l;  j = 2 \* i + 1;  lbl\_status\_02.Text = "Đang shift heap";  lbl\_status\_02.Visible = true;  //Thiết lập mũi tên chỉ i  Mui\_ten\_xanh\_xuong\_1.Visible = true;  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Refresh();  //thiết lập mũi tên chỉ j  Mui\_ten\_xanh\_xuong\_2.Visible = true;  Mui\_ten\_xanh\_xuong\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_2.Text = "j=" + j;  Mui\_ten\_xanh\_xuong\_2.Refresh();  x = thamSo.arrNode[i];  temp = thamSo.arrLbl[i];  index\_temp = i;  while (j <= r)  {  Application.DoEvents();  if (j < r)  if (thamSo.arrNode[j] > thamSo.arrNode[j + 1])  j++;  //thiết lập mũi tên chỉ j  Mui\_ten\_xanh\_xuong\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_2.Text = "j=" + j;  Mui\_ten\_xanh\_xuong\_2.Refresh();  if (thamSo.arrNode[j] >= x)  return;  else  {  thamSo.arrNode[i] = thamSo.arrNode[j];  thamSo.arrNode[j] = x;  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Swap\_Node(thamSo.arrLbl[j], temp);  });  Swap\_NodeAn(j, index\_temp);  if (ckDebug.Checked == true)  Pause();  else  Delay(5 \* thamSo.speed);  i = j;  j = 2 \* i + 1;  if (j <= r)  {  //Thiết lập mũi tên chỉ i  Mui\_ten\_xanh\_xuong\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* i) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[i].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_1.Text = "i=" + i;  Mui\_ten\_xanh\_xuong\_1.Refresh();  //thiết lập mũi tên chỉ j  Mui\_ten\_xanh\_xuong\_2.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* j) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[j].Location.Y - thamSo.sizeN - 70);  Mui\_ten\_xanh\_xuong\_2.Text = "j=" + j;  Mui\_ten\_xanh\_xuong\_2.Refresh();  }  x = thamSo.arrNode[i];  temp = thamSo.arrLbl[i];  index\_temp = i;  }  }  }  void CreateHeap\_increase(int n)  {  int l = n / 2 - 1;  while (l >= 0)  {  Shift\_increase(l, n - 1);  l--;  }  }  void CreateHeap\_giam(int n)  {  int l = n / 2 - 1;  while (l >= 0)  {  Shift\_giam(l, n - 1);  l--;  }  }  void HeapSortincrease(int n)  {  listCode.SelectedIndex = 2;  int r;  listCode.SelectedIndex = 3;  CreateHeap\_increase(n);  listCode.SelectedIndex = 4;  r = n - 1;  //Thiết lập mũi tên chỉ r  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* r) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[r].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Text = "r=" + r;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Refresh();  listCode.SelectedIndex = 5;  while (r > 0)  {  Application.DoEvents();  Swap\_Giatri(ref thamSo.arrNode[0], ref thamSo.arrNode[r]);  Mui\_ten\_xanh\_xuong\_1.Visible = false;  Mui\_ten\_xanh\_xuong\_2.Visible = false;  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "HoanVi( a[0] , a[" + r + "] )";  listCode.SelectedIndex = 7;  this.Invoke((MethodInvoker)delegate  {  Swap\_Node(thamSo.arrLbl[0], thamSo.arrLbl[r]);  });  lbl\_status\_02.Visible = false;  Swap\_NodeAn(0, r);  thamSo.arrLbl[r].BackColor = colorComplete;  thamSo.arrLbl[0].BackColor = colorDefault;  // Đặt lại màu cho phần tử đã được sắp xếp  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "a[" + r + "] đã đúng thứ tự!";  if (ckDebug.Checked == true)  Pause();  else  Delay(10 \* thamSo.speed);  lbl\_status\_02.Visible = false;  listCode.SelectedIndex = 8;  r--;  //Thiết lập mũi tên chỉ r  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* r) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[r].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Text = "r=" + r;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Refresh();  listCode.SelectedIndex = 9;  if (r > 0)  {  listCode.SelectedIndex = 10;  Shift\_increase(0, r);  }  }  }  void HeapSortGiam(int n)  {  listCode.SelectedIndex = 2;  int r;  listCode.SelectedIndex = 3;  CreateHeap\_giam(n);  listCode.SelectedIndex = 4;  r = n - 1;  //Thiết lập mũi tên chỉ r  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* r) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[r].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Text = "r=" + r;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Refresh();  listCode.SelectedIndex = 5;  while (r > 0)  {  Application.DoEvents();  Swap\_Giatri(ref thamSo.arrNode[0], ref thamSo.arrNode[r]);  Mui\_ten\_xanh\_xuong\_1.Visible = false;  Mui\_ten\_xanh\_xuong\_2.Visible = false;  //  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "HoanVi( a[0] , a[" + r + "] )";  listCode.SelectedIndex = 7;  this.Invoke((MethodInvoker)delegate  {  Swap\_Node(thamSo.arrLbl[0], thamSo.arrLbl[r]);  });  lbl\_status\_02.Visible = false;  Swap\_NodeAn(0, r);  thamSo.arrLbl[r].BackColor = colorComplete;  thamSo.arrLbl[0].BackColor = colorDefault;  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "a[" + r + "] đã đúng thứ tự!";  if (ckDebug.Checked == true)  Pause();  else  Delay(10 \* thamSo.speed);  lbl\_status\_02.Visible = false;  listCode.SelectedIndex = 8;  r--;  //Thiết lập mũi tên chỉ r  Mui\_ten\_xanh\_len\_1.Location = new Point((thamSo.firstN + (thamSo.sizeN +  thamSo.disN) \* r) + (thamSo.sizeN / 2) - 30, thamSo.arrLbl[r].Location.Y + 2 \* thamSo.sizeN + 5);  Mui\_ten\_xanh\_len\_1.Text = "r=" + r;  Mui\_ten\_xanh\_len\_1.Visible = true;  Mui\_ten\_xanh\_len\_1.Refresh();  listCode.SelectedIndex = 9;  if (r > 0)  {  listCode.SelectedIndex = 10;  Shift\_giam(0, r);  } }} |

* Hàm MergeSort

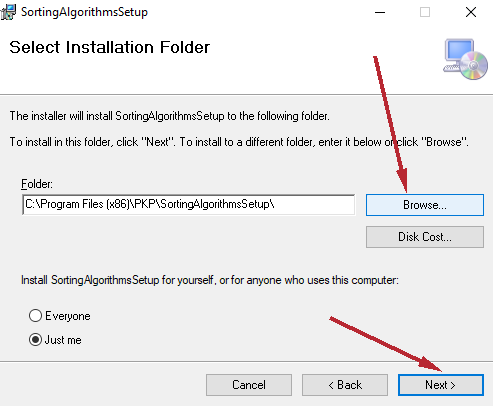
|  |
| --- |
| void Distribute(ref int nb, ref int nc, int k)  {  int i, pa, pb, pc;  pa = pb = pc = 0;  while (pa < thamSo.nOe)  {  Application.DoEvents();  for (i = 0; (pa < thamSo.nOe) && (i < k); i++, pa++, pb++)  {  thamSo.b[pb] = thamSo.arrNode[pa];  thamSo.node\_B[pb] = thamSo.arrLbl[pa];  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Node\_Up(thamSo.node\_B[pb], 2 \* (thamSo.sizeN));  Den\_tdo\_x\_node(thamSo.node\_B[pb], pb);  });  thamSo.arrLbl[pa].BackColor = colorQuickU; // màu trên  }  for (i = 0; (pa < thamSo.nOe) && (i < k); i++, pa++, pc++)  {  thamSo.c[pc] = thamSo.arrNode[pa];  thamSo.node\_C[pc] = thamSo.arrLbl[pa];  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  Node\_Down(thamSo.node\_C[pc], (thamSo.sizeN + 80));  Den\_tdo\_x\_node(thamSo.node\_C[pc], pc);  });  thamSo.arrLbl[pa].BackColor = colorQuickD; // màu dưới  }  }  nb = pb;  nc = pc;  }  //Hàm kết hợp b và c vào a  void Merge\_increase(int nb, int nc, int k)  {  int pa, pb, pc, ib, ic, kb, kc, lennb, lennc;  //lưu những giá trị để đếm Node dư  lennb = nb;  lennc = nc;  pa = pb = pc = 0; ib = ic = 0;  while ((nb > 0) && (nc > 0))  {  Application.DoEvents();  kb = min(k, nb);  kc = min(k, nc);  if (thamSo.b[pb + ib] <= thamSo.c[pc + ic])  {  thamSo.arrNode[pa] = thamSo.b[pb + ib];  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_B[pb + ib], pa);  });  thamSo.arrLbl[pa] = thamSo.node\_B[pb + ib];  pa++;  ib++;  if (ib == kb)  {  for (; ic < kc; ic++)  {  thamSo.arrNode[pa] = thamSo.c[pc + ic];  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_C[pc + ic], pa);  });  thamSo.arrLbl[pa] = thamSo.node\_C[pc + ic];  pa++;  }  pb += kb;  pc += kc;  ib = ic = 0;  nb -= kb;  nc -= kc;  }  }  else  {  thamSo.arrNode[pa] = thamSo.c[pc + ic];  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_C[pc + ic], pa);  });  thamSo.arrLbl[pa] = thamSo.node\_C[pc + ic];  pa++;  ic++;  if (ic == kc)  {  for (; ib < kb; ib++)  {  thamSo.arrNode[pa] = thamSo.b[pb + ib];  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_B[pb + ib], pa);  });  thamSo.arrLbl[pa] = thamSo.node\_B[pb + ib];  pa++;  }  pb += kb;  pc += kc;  ib = ic = 0;  nb -= kb;  nc -= kc;  }  }  }  //Di chuyển các Node dư thừa vào vị trí  for (; nb > 0; nb--)  {  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_B[lennb - nb], pa);  });  pa++;  }  for (; nc > 0; nc--)  {  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_C[lennc - nc], pa);  });  pa++;  }  }  void Merge\_giam(int nb, int nc, int k)  {  int pa, pb, pc, ib, ic, kb, kc, lennb, lennc;  //lưu những giá trị để đếm Node dư  lennb = nb;  lennc = nc;  pa = pb = pc = 0; ib = ic = 0;  while ((nb > 0) && (nc > 0))  {  Application.DoEvents();  kb = min(k, nb);  kc = min(k, nc);  if (thamSo.b[pb + ib] >= thamSo.c[pc + ic])  {  thamSo.arrNode[pa] = thamSo.b[pb + ib];  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_B[pb + ib], pa);  });  thamSo.arrLbl[pa] = thamSo.node\_B[pb + ib];  pa++;  ib++;  if (ib == kb)  {  for (; ic < kc; ic++)  {  thamSo.arrNode[pa] = thamSo.c[pc + ic];  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_C[pc + ic], pa);  });  thamSo.arrLbl[pa] = thamSo.node\_C[pc + ic];  pa++;  }  pb += kb;  pc += kc;  ib = ic = 0;  nb -= kb;  nc -= kc;  }  }  else  {  thamSo.arrNode[pa] = thamSo.c[pc + ic];  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_C[pc + ic], pa);  });  thamSo.arrLbl[pa] = thamSo.node\_C[pc + ic];  pa++;  ic++;  if (ic == kc)  {  for (; ib < kb; ib++)  {  thamSo.arrNode[pa] = thamSo.b[pb + ib];  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_B[pb + ib], pa);  });  thamSo.arrLbl[pa] = thamSo.node\_B[pb + ib];  pa++;  }  pb += kb;  pc += kc;  ib = ic = 0;  nb -= kb;  nc -= kc;  }  }  }  //Di chuyển các Node dư thừa vào vị trí  for (; nb > 0; nb--)  {  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_B[lennb - nb], pa);  });  pa++;  }  for (; nc > 0; nc--)  {  Application.DoEvents();  this.Invoke((MethodInvoker)delegate  {  toLocaN(thamSo.node\_C[lennc - nc], pa);  });  pa++;  }  }  void MergeSortincrease()  {  lblA.Visible = true;  lblC.Visible = true;  lblB.Visible = true;  thamSo.b = new int[thamSo.nOe];  thamSo.c = new int[thamSo.nOe];  thamSo.node\_B = new Label[thamSo.nOe];  thamSo.node\_C = new Label[thamSo.nOe];  //Dán nhãn mảng b  int k, nc = 0, nb = 0;  ChonDongChoCodeListBox(3);  for (k = 1; k < thamSo.nOe; k \*= 2)  {  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "k = " + k;  ChonDongChoCodeListBox(5);  Distribute(ref nb, ref nc, k);  Merge\_increase(nb, nc, k);  ChonDongChoCodeListBox(6);  }  }  void MergeSortGiam()  {  thamSo.b = new int[thamSo.nOe];  thamSo.c = new int[thamSo.nOe];  thamSo.node\_B = new Label[thamSo.nOe];  thamSo.node\_C = new Label[thamSo.nOe];  //Dán nhãn mảng b  int k, nc = 0, nb = 0;  ChonDongChoCodeListBox(3);  for (k = 1; k < thamSo.nOe; k \*= 2)  {  lbl\_status\_02.Visible = true;  lbl\_status\_02.Text = "k = " + k;  ChonDongChoCodeListBox(5);  Distribute(ref nb, ref nc, k);  Merge\_giam(nb, nc, k);  ChonDongChoCodeListBox(6);  }  } |

## Cài đặt phần mềm

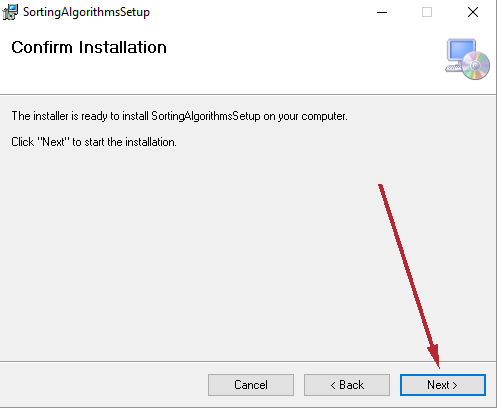
Nhấn Next để tiếp tục



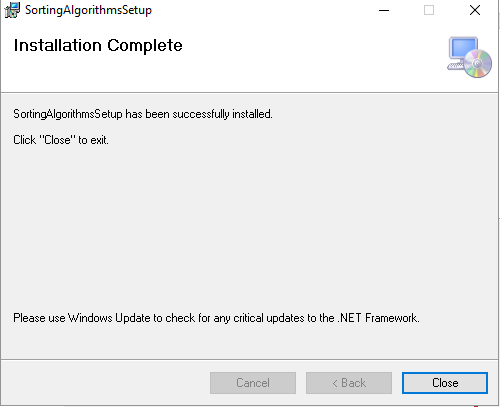
Chọn Browse để chọn vị trí lưu phần mềm rồi nhấn Next để tiếp tục



Nhấn next để Hoàn thành cài đặt



Cài đặt hoàn tât. Nhấn Close và vào phần mềm SortingAlgorithms để sử dụng



# Kiểm thử

## Unit Test

Để đảm bảo chất lượng của chương trình, từng lớp sẽ được kiểm tra một cách riêng rẽ bằng phương pháp hộp trắng. Người tiến hành sẽ là những người trực tiếp cài đặt các lớp và phương thức ấy.

## Integration Test

Kiểm tra sự liên kết và giao diện giữa các lớp trong phần mềm.

## System Test

### Phương pháp kiểm tra

Hệ thống sẽ được kiểm tra theo phương pháp hộp đen. Tức là dựa vào những đặc tả và yêu cầu của hệ thống đã thu thập được từ đầu để kiểm tra output chung của hệ thống mà không cần quan tâm đến việc thực hiện bên trong của chúng.

# Tổng kết

## Tổng kết

Phần mềm bước đầu đã được phát triển đi đúng theo dự tính và kế hoạch. Tuy gặp khá nhiều khó khăn trong vấn đề tiếp cận với công nghệ lập trình mới, nhóm vẫn đảm bảo được các tiến độ làm việc do đã đặt ra.

## Nhận xét và đánh giá

* **Thành quả**
* Hoàn tất tất cả các mục của phần mềm bao gồm: có hình ảnh mô phỏng lại từng thao tác xử lí đối với mỗi thuật toán sắp xếp với màu sắc hỗ trợ, có khung hiển thị code để đối chiếu, có chế độ Debug, có bộ đếm thời gian, có ý tưởng của thuật toán.
* Học được cách sử dụng Thread, Timer, cách quản lý các tiến trình, xử lý đồng bộ tiến trình. Cách tạo hiệu ứng chuyển động trên form. Cách quản lý sự kiện của bàn phím.
* Kỹ năng dùng git và github quản lý
* Kỹ năng làm việc nhóm và phân chia
* **Hạn chế**
* Số lượng phần tử nhỏ chỉ dùng trong giảng dạy không thể làm các thí nghiệm lớn.
* Giao diện chạy không được ổn định.
* Gặp nhiều khó khăn khi xử lí đa luồng Multithread trên C#.
* Chưa có mục so sánh giữa các thuật toán
* **Phát triển**
* Sẽ khắc phục các hạn chế trên để hoàn thiện sản phẩm tối yêu nhất

## Phân Rã Công Việc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Công việc** | **Duration (days)** | **Assign To** | **% Finished** |
| 1 | Tìm hiểu sơ bộ & đăng ký đồ án |  | Phúc, Khoa, Phú | 100% |
| 2 | Tìm hiểu công nghệ liên quan |  | Phúc, Khoa, Phú | 100% |
| **I. VIẾT BÁO CÁO** | | | | |
| 3 | **Chương 1 – Hiện trạng** |  |  |  |
| 4 | 1.1. Hiện trạng vấn đề |  | Khoa | 100% |
| 5 | 1.2. Hiện cơ sở vật chất và con người |  | Khoa | 100% |
| 6 | 1.3. Yêu cầu về phần mềm |  | Khoa | 100% |
| 7 | **Chương 2: Phân tích** |  |  |  |
| 8 | 2.1. Yêu cầu phần mềm |  | Khoa | 100% |
| 9 | 2.2. Mô hình hóa |  | Khoa | 100% |
| 10 | **Chương 3: Thiết kế** |  |  |  |
| 11 | 3.1. Thiết kế dữ liệu |  | Khoa | 100% |
| 12 | 3.2. Thiết kế dữ liệu bộ nhớ chính |  | Khoa | 100% |
| 13 | 3.3. Thiết kế giao diện và thành phần xử lí của giao diện |  | Khoa | 100% |
| 14 | **Chương 4: Cài đặt** |  |  |  |
| 15 | 4.1. Tổng quan về công nghệ sử dụng |  | Phúc, Khoa | 100% |
| 16 | 4.2. Cài đặt phần mềm |  | Phúc, Khoa | 100% |
| 17 | 4.3. Vấn đề khi cài đặt |  | Phúc, Khoa | 100% |
| 18 | **Chương 5: Kiểm thử** |  |  | 80%-90% |
| 29 | 5.1. Unit Test |  | Phúc, Khoa, Phú |  |
| 20 | 5.2. Integration Test |  | Phúc, Khoa, Phú |  |
| 21 | 5.3. System Test |  | Phúc, Khoa, Phú |  |
| 22 | **Chương 6: Kết luận** |  | Phú | 100% |
| 23 | **Tài liệu tham khảo** |  | Phú | 100% |
| **II. LẬP TRÌNH** | | | | |
|  | 1. Giao diện + Form | 10 | Khoa | 90%-95% |
|  | 2. Các xử lý userControl | 20 | Phúc, Phú | 80%-85% |
|  | 3. Coding các thuật toán | 15 | Phú, Phúc | 80%-85% |
|  | 4. Format code | 15 | Khoa | 85%-90% |
| **III. KIỂM THỬ** | | | | |
|  | 1. Unit Test |  |  |  |
|  | 2. Integration Test |  |  |  |
|  | 3. System Test |  |  |  |
|  | 4. Acceptance Test |  |  |  |
| **IV. NỘP BÁO CÁO TIẾN ĐỘ & SẢN PHẨM** | | | **Tiến độ sản phẩm** | |
|  | **Ngày 2 tháng 03 năm 2017** | | Bắt đầu nhận đồ án | |
|  | **Ngày 2 tháng 04 năm 2017** | | Cơ bản hoàn thành định hướng đồ án.  Thiết kế sơ bộ giao diện. | |
|  | **Ngày 2 tháng 05 năm 2017** | | Hoàn thành giao diện  Hoàn thành được các chức năng phím tắt.  Hoàn thành một nửa các loại thuật toán. | |
|  | **Ngày 2 tháng 06 năm 2017** | | Hoàn thành các thuật toán  Chạy sơ khai | |
|  | **Ngày 12 tháng 06 năm 2017** | | Thêm các chức để cải thiện chương trình | |
|  | **Ngày 17 tháng 06 năm 2017** | | Hoàn thành chương trình | |

# Tài liệu tham khảo

* Xử lý thread

<https://stackoverflow.com/questions/5181777/use-of-application-doevents>

<https://msdn.microsoft.com/en-us/library/zyzhdc6b(v=vs.110).aspx>

<https://stackoverflow.com/questions/14703698/invokedelegate>

* Video tham khảo

<https://www.youtube.com/watch?v=IeLBBgqII3c>

<https://www.youtube.com/watch?v=Sp3gmL3cwqM>

* Xử lý sự kiện phím

<https://stackoverflow.com/questions/50804731/winforms-with-onkeydown-and-additional-controls?noredirect=1&lq=1>

* Đặc biệt là kiến thức trong slide

<https://drive.google.com/drive/folders/1XmcwpPJCMPrAWeal5jFvkT7gYCiNZrR->