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

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**BÁO CÁO THỰC HÀNH**

**Môn: Lập trình trực quan**

**Đề tài: Phần mềm mô phỏng các thuật toán sắp xếp**



GVHD: Huỳnh Tuấn Anh

Nhóm: Quảng

Huỳnh Tấn Duy - 16520287

Huỳnh Phương Duy - 16520285

Nguyễn Văn Vinh - 16521444

TP.HCM, ngày 15 tháng 6 năm 2018

**LỜI MỞ ĐẦU**

Lời đầu tiên, em xin chân thành gửi lời cảm ơn đến thầy Huỳnh Tuấn Anh, người đã nhiệt tình giúp đỡ, hướng dẫn và định hướng cho em trong suốt quá trình thực hiện đồ án này.

Trong xã hội ngày nay, các ứng dụng tin học đang ngày càng phát triển và được ứng dụng trong rất nhiều lĩnh vực đời sống từ kinh tế đến khoa học xã hội...Và giáo dục cũng không là ngoại lệ, việc sử dụng các ứng dụng Công nghệ Thông tin là rất hợp lý và cần thiết, nhằm nâng cao chất lượng học tập của học sinh và sinh viên. Trên cơ sở đó, nhóm em tạo ra phần mềm này với hi vọng nó có thể giúp ích cho quá trình học tập của các bạn, đồng thời bọn em cũng có cơ hội ứng dụng các kiến thức học được ở môn Lập trình trực quan.

Cuối cùng, vì còn hạn chế về năng lực và kinh nghiệm làm phần mềm, sản phẩm của tụi em khó có thể tránh khỏi những thiếu sót. Kính mong nhận được những góp ý của thầy.

Mục lục

[Chương 1. Tổng quan 5](#_Toc517033510)

[1.1. Hiện trạng và yêu cầu thực tế 6](#_Toc517033511)

[1.2. Giới thiệu bài toán cần giải quyết 6](#_Toc517033512)

[1.3. Quy trình thực hiện các công việc chính 6](#_Toc517033513)

[1.4. Tổng quan các chức năng chính 6](#_Toc517033514)

[Chương 2. Phân tích yêu cầu phần mềm và mô hình hoá 7](#_Toc517033515)

[2.1. Yêu cầu phần mềm 8](#_Toc517033516)

[2.1.1. Yêu cầu chức năng 8](#_Toc517033517)

[2.1.2. Yêu cầu phi chức năng 8](#_Toc517033518)

[2.1.3. Bảng FURPS 8](#_Toc517033519)

[2.2. Mô hình hóa 9](#_Toc517033520)

[Chương 3. Kỹ thuật xử lí 10](#_Toc517033521)

[3.1. Phân tích 11](#_Toc517033522)

[3.1.1. Phân tích hiệu ứng. 11](#_Toc517033523)

[3.1.2. Phân tích chức năng. 11](#_Toc517033524)

[3.2. Kĩ thuật sử dụng. 11](#_Toc517033525)

[3.2.1 Xây dựng lớp ButtonNode kế thừa từ lớp Button. 11](#_Toc517033526)

[3.2.2 Xử lý bất đồng bộ với Thread bằng BackgroundWorker. 12](#_Toc517033527)

[3.2.3 Xử lý việc di chuyển các phần tử của dãy. 12](#_Toc517033528)

[Chương 4 : Thiết kế giao diện. 13](#_Toc517033529)

[4.1. Giao diện tổng thể 14](#_Toc517033530)

[4.2. Giao diện thực hiện các thuật toán Sort 15](#_Toc517033531)

[4.3. Giao diện Instrucion 16](#_Toc517033532)

[4.4. Giao diện About 17](#_Toc517033533)

[Chương 5 : Cài đặt và Kiểm thử 18](#_Toc517033534)

[5.1. Các công nghệ sử dụng 19](#_Toc517033535)

[5.1.1. Ngôn ngữ lập trình c# 19](#_Toc517033536)

[5.1.2. Bunifu framework 19](#_Toc517033537)

[5.2. Các hàm xử lý chính 19](#_Toc517033538)

[5.2.1. Button Node 19](#_Toc517033539)

[5.2.2. Hàm tạo mảng 22](#_Toc517033540)

[5.2.3. Tạo và quản lý thread 23](#_Toc517033541)

[5.2.4. Hàm huỷ tiến trình 25](#_Toc517033542)

[5.3. Kiểm thử 25](#_Toc517033543)

[5.3.1. Unit Test 25](#_Toc517033544)

[5.3.2. Intergration Test 25](#_Toc517033545)

[5.3.3. System Test 25](#_Toc517033546)

[Chương 6. Tổng kết 26](#_Toc517033547)

[6.1. Kết quả đạt được. 27](#_Toc517033548)

[6.2. Hạn chế. 27](#_Toc517033549)

[6.3. Hướng phát triển. 27](#_Toc517033550)

6.4. Bảng phân rã công việc…………………………………………………………………………….27

TÀI LIỆU THAM KHẢO…………………………………………………………………………………29

# 

# 

# Chương 1

# TỔNG QUAN

## Hiện trạng và yêu cầu thực tế

Các thuật toán sắp xếp như Quick Sort, Shell Sort, Interchange Sort,… có ứng dụng và ý nghĩa rất lớn trong quản lý và lưu trữ dữ liệu. Các thuật toán sắp sếp này được giới thiệu đầy đủ trong môn học Cấu trúc dữ liệu và giải thuật. Tuy nhiên, mỗi thuật toán lại có cơ chế, cách vận hành riêng. Một số thuật toán thâm chí còn có cơ chế còn rất phức tạp, trừu tượng, dẫn đến hiện tượng sinh viên khó nắm bắt, hiểu rõ tất cả các thuật toán. Do đó, một phần mềm có khả năng minh họa các thuật toán sắp xếp một cách rõ ràng, cụ thể, trực quan , sinh động là cực kỳ cần thiết cho sinh viên.

Ứng dụng công nghệ thông tin vào việc hỗ trợ học tập cho học sinh, sinh viên không còn là một việc xa lạ trong thời đại ngày nay. Việc này giúp người dùng là những học sinh, sinh viên có cái nhìn trực quan hơn đối với môn học của mình, giảm bớt tính đơn điệu, nhàm chán và khô khan của môn học. Vì vậy, từ những yêu cầu và khảo sát từ thực tế, nhóm chúng em đã quyết định xây dựng phần mềm Mô phỏng thuật toán – **Simulation Sort** nhằm giúp các bạn đang có nhu cầu tìm hiểu về các thuật toán sắp xếp dễ dàng nắm bắt hơn, nâng cao chất lượng học tập.

## Giới thiệu bài toán cần giải quyết

**Simulation Sort** là phần mềm giúp người dùng mô phỏng các thuật toán sắp xếp. Với việc di chuyển các con số một cách trực quan, sinh động, người sử dụng sẽ dễ dàng nắm bắt từng bước của mỗi thuật toán. Đồng thời, phần mềm còn cung cấp lí thuyết thuật toán và code minh họa để người dùng tham khảo. Về cơ bản, phần mềm sẽ mô phỏng chi tiết từng bước 10 thuật toán (các thuật toán ở môn Cấu trúc dữ liệu và giải thuật) với các tùy chọn cần thiết đối với mỗi thuật toán.

## Quy trình thực hiện các công việc chính

- Thiết kế giao diện phần mềm.

- Thiết kế hệ thống phần mềm.

- Cài đặt.

- Bảo trì, nâng cấp, sửa chữa.

## 1.4. Tổng quan các chức năng chính

- Cho phép người dùng nhập dãy các phần tử cần sắp xếp, hoặc tạo 1 dãy ngẫu nhiên theo số lượng phần tử mà người dùng yêu cầu.

- Cho phép người dùng lựa chọn thuật toán sắp xếp và hướng sắp xếp.

- Cung cấp lý thuyết thuật toán và code minh họa của thuật toán sắp xếp.

- Mô phỏng quá trình sắp xếp một cách trực quan.

- Ghi lại nhật ký sắp xếp cho người dùng xem lại.

# 

# Chương 2

# PHÂN TÍCH YÊU CẦU PHẦN MỀM VÀ MÔ HÌNH HO

## 2.1. Yêu cầu phần mềm

### 2.1.1. Yêu cầu chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Định danh** | **Độ ưu tiên** | **Mô tả yêu cầu** |
| 1 | Instruction | 1 | Người dùng đọc hướng dẫn sử dụng phần mềm |
| 2 | Choose Sort | 1 | Người dùng lựa chọn thuật toán cần sắp xếp |
| 3 | No.Element | 1 | Người dùng nhập số lượng phần tử cần sắp xếp |
| 4 | Generate | 1 | Người dùng chọn phương thức input bằng tay hoặc tự động ngẫu nhiên |
| 5 | Sort By | 1 | Người dùng chọn cách sắp xếp( tăng, giảm) |
| 6 | Start | 2 | Lấy thông tin từ input để xử lí và xuất kết quả |
| 7 | Simulation Algorithm | 3 | Hiển thị quá trình sắp xếp thuật toán theo yêu cầu người dùng |
| 8 | Save Process | 3 | Lưu lại quá trình sắp xếp |
| 9 | Step | 4 | Hiển thị quá trình sắp xếp theo từng bước |
| 10 | Idea | 4 | Hiển thị ý tưởng thuật toán |
| 11 | Code C/C++ | 4 | Hiển thị đoạn lệnh code C/C++ đã thực thi |
| 12 | Pause | 5 | Tạm dừng quá trình sắp xếp |
| 13 | Stop Process | 5 | Hủy quá trình sắp xếp |
| 14 | Show Dialog | 5 | Thông báo kết thúc thuật toán |

### 2.1.2. Yêu cầu phi chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Định danh** | **Độ ưu tiên** | **Mô tả yêu cầu** |
| 1 | Anti- reverse Engineering | 1 | Sử dụng các biện pháp bảo vệ mã nguồn. |

### 2.1.3. Bảng FURPS

|  |  |
| --- | --- |
| **Tiêu chí chất lượng** | **Mô tả** |
| Functionality | Hướng tới phục vụ người dùng với mục đích học tập |
| Usability | Giao diện thiết kế đơn giản, dễ hiểu, dễ sử dụng |
| Reliablity | Hệ thống được thiết kế với khả năng giảm thiểu bug trong quá trình sắp xếp thuật toán |
| Performance | Tối ưu hóa thuật toán, giảm thiểu thời gian sắp xếp  Việc sắp xếp sẽ nhanh hơn vì chỉ tập trung vào một giao diện đơn giản |
| Supportability | Truy vết người dùng để kịp thời sửa chữa khi có crash(đang hoàn thiện) |

## 2.2. Mô hình hóa

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Mô tả** | **Yêu cầu liên quan** |
| 1 | Nhập thông tin input | Người dùng nhập thông tin input theo các bước yêu cầu | Choose Sort, No.Element,  Generate,  Sort By, |
| 2 | Xử lí dữ liệu đầu vào | Hệ thống xử lí dữ liệu để sắp xếp thuật toán theo đúng yêu cầu người dùng | Start,  Pause  Stop Process  Save Process |
| 3 | Xem quá trình thực hiện thuật toán | Hệ thống hiển thị tứng bước sắp xếp của thuật toán đã chọn | Simulation Algorithm,  Step,Idea,,Code C/C++ |
| 4 | Kết thúc sắp xếp | Thông báo kết thúc thuật toán hoặc người dùng tự kết thúc | Pause, Stop Process, Show Dialog |

# 

# Chương 3

# KỸ THUẬT XỬ LÍ

## 3.1. Phân tích

### 3.1.1. Phân tích hiệu ứng.

Để mô phỏng các thuật toán sắp xếp, ta cần tạo ra một mảng để chứa các phần tử. Khi bắt đầu mô phỏng, các phần tử di chuyển hoán đổi vị trí với nhau một cách trực quan, đồng bộ theo thuật toán sắp xếp được chọn. Đồng thời, lý thuyết thuật toán và code minh họa cũng được hiển thị để người sử dụng có thể tham khảo.

Nhật ký sắp xếp được tạo ra, những lần hoán đổi vị trí giữa các phẩn tử đều được ghi lại. Khi kết thúc quá trình mô phỏng, nhật ký ghi lại đẩy đủ quá trình sắp xếp từ mảng ban đầu cho tới mảng kết quả.

### 3.1.2. Phân tích chức năng.

#### 3.1.2.1. Khởi tạo mảng

- Nhập số lượng phần tử của dãy cần sắp xếp.

- Khởi tạo mảng bằng một trong hai cách :

+ Tạo ngẫu nhiên : các phần tử được sinh ra với giá trị ngẫu nhiên.

+ Nhập bằng tay: các phần tử được khởi tạo với giá trị bằng 0, người dùng tiến hành nhập giá trị trực tiếp cho các phần tử.

#### 3.1.2.2. Bảng điều khiển

- Bắt đầu mô phỏng : tiến hành sắp xếp các phần tử của mảng, chạy tiểu trình minh họa và hiển thị các thông tin liên quan.

- Tạm dừng: tạm dừng quá trình hiển thị. Sau khi tạm dừng, người dùng có thể tiếp tục quá trình bằng cách click vào nút này 1 lần nữa.

- Hủy quá trình: hủy quá trình sắp xếp và xóa mảng.

#### 3.1.2.3. Các chức năng phụ :

- Cho phép chạy thuật toán sắp xếp theo hai hướng là tăng dần hoặc giảm dần.

- Cập nhật thông tin cho quá trình sắp xếp.

## 3.2. Kĩ thuật sử dụng.

### 3.2.1 Xây dựng lớp ButtonNode kế thừa từ lớp Button.

Để minh họa cho dãy phần tử cần sắp xếp, ta sử dụng một dãy các button, mỗi button đại diện cho một phần tử của mảng và được hiển thị theo đúng thứ tự trong dãy.

Một trong những chức năng quan trọng của phần mềm là cho phép người dùng nhập trực tiếp giá trị của phần tử ngay trên các “nút” được khởi tạo. Button không thể thực hiện được việc này, do đó ta xây dựng lớp ButtonNode kế thừa từ lớp Button, bao gồm các thuộc tính:

- Giá trị ( kiểu int) : lưu giá trị của phần tử.

- Vị trí hiện tại (kiểu int) : lưu vị trí của phần tử trong dãy.

- Nhập tay ( kiểu TextBox) : cho phép người dùng nhập giá trị trực tiếp trên mỗi “nút”. Nếu sự kiện nhập bằng tay diễn ra, thuộc tính “Giá trị” sẽ tự động cập nhật theo.

### 3.2.2 Xử lý bất đồng bộ với Thread bằng BackgroundWorker.

#### 3.2.2.1 Giới thiệu về BackgroundWorker.

BackgroundWorker là một component được C# cung cấp, giúp cho việc lập trình Thread trở nên dễ dàng do các thuộc tính và event mà nó hỗ trợ sẵn.

#### 3.2.2.1 Quy trình làm việc của BackgroundWorker.

- Bắt đầu thread : Khi bắt đầu mô phỏng, gọi hàm RunWorkerAsync() để thực thi phương thức OnDoWork(). Trong phương thức OnDoWork(), quá trình sắp xếp và hoán đổi các Button hiển thị bắt đầu.

- Cập nhật tiến độ công việc: Dùng phương thức ReportProgress() với tham số là phần trăm tiến độ hoàn thành công việc để kích hoạt sự kiện ProgressChanged nhằm cập nhật tiến độ thực hiện công việc.

- Hủy thread: để ngưng một thread đang hoạt động, ta có thể gọi phương thức CancelAsync(). Khi đó thuộc tính CancellationPending sẽ có giá trị True. Dựa vào thuộc tính này ta có thể kiểm tra và ngừng công việc lại.

- Sau khi phương thức OnDoWork() thực hiện xong, phương thức OnRunWorkerCompleted() sẽ tự động được gọi, hiển thị thông báo cho người dùng biết quá trình mô phỏng đã kết thúc.

### 3.2.3 Xử lý việc di chuyển các phần tử của dãy.

Dựa vào kích thước và vị trí hiện tại của mỗi nút trong dãy để xác định khoảng cách giữa 2 nút cần hoán đổi.

Gọi hàm ReportProgress() để cập nhật giao diện, các nút sẽ bắt đầu di chuyển :

- Nút thứ nhất (bên trái) đi lên, nút thứ hai (bên phải) đi xuống.

- Nút bên trái dịch sang phải đúng bằng khoảng cách hoán đổi đã xác định, ngược lại với nút bên phải.

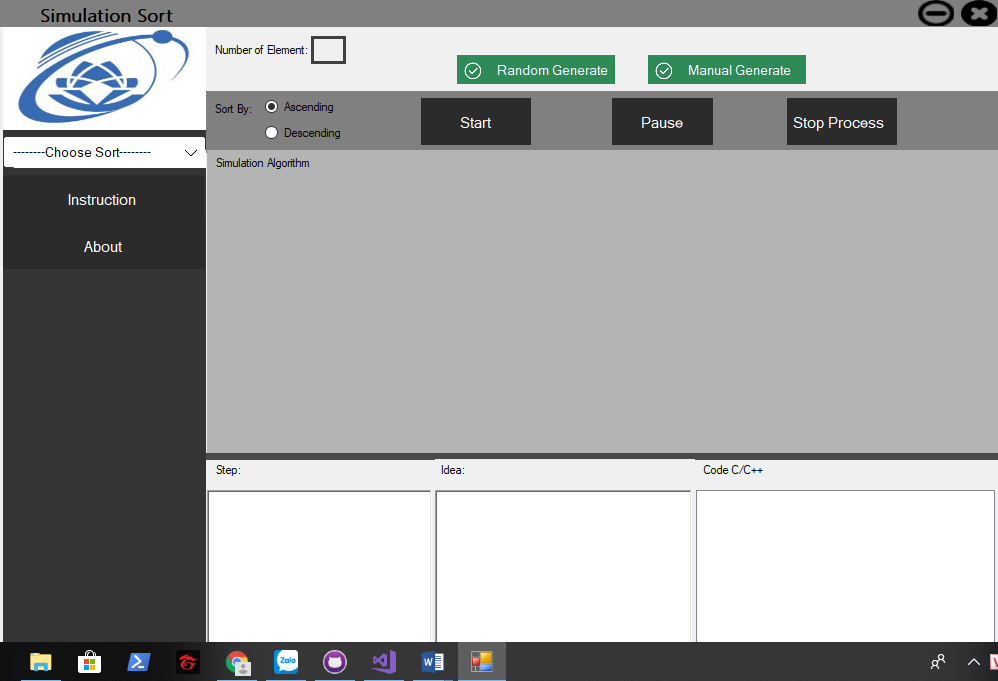
- Nút thứ nhất đi xuống, nút thứ hai đi lên. Kết thúc quá trình hoán đổi, gán lại giá trị và thứ tự cho các nút.

# 

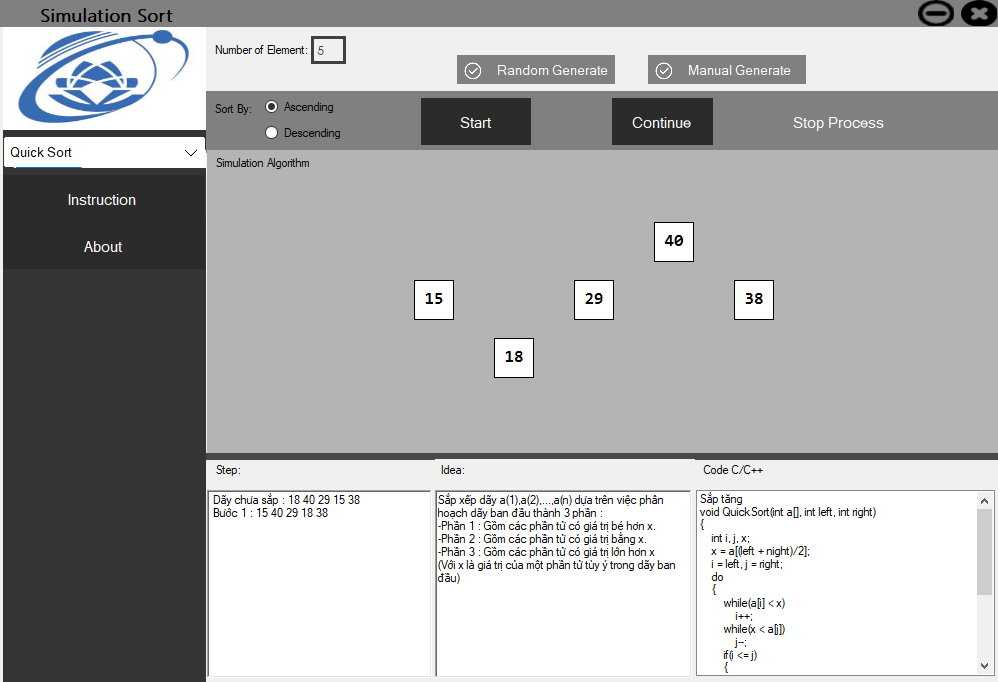
# Chương 4

# THIẾT KẾ GIAO DIỆN

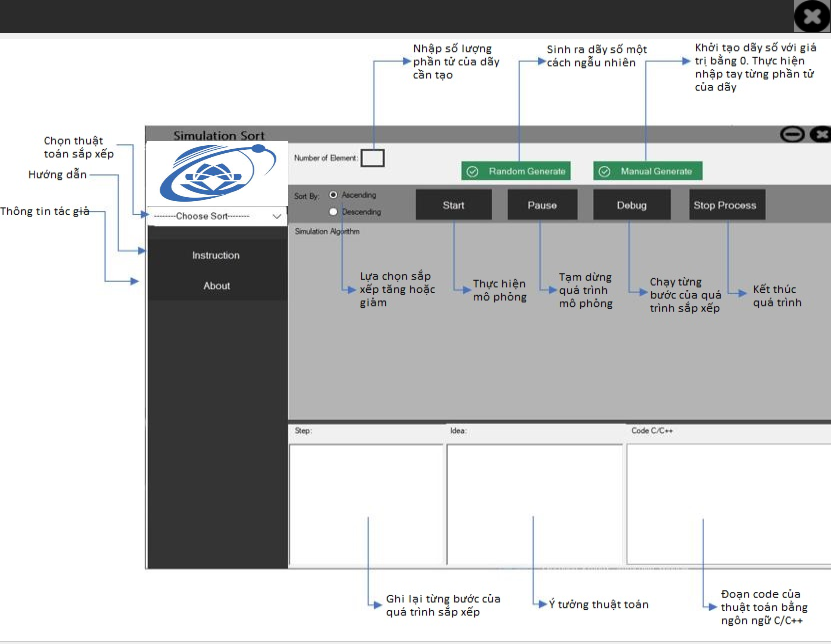
## 4.1. Giao diện tổng thể



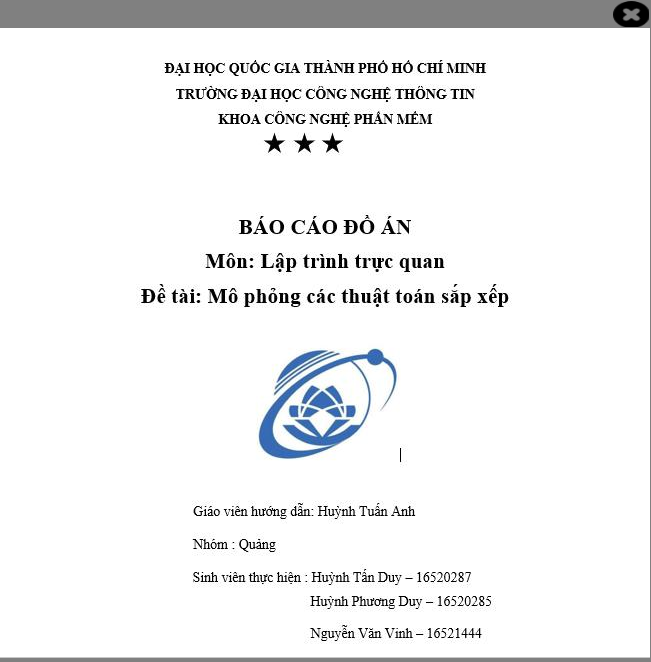
## 4.2. Giao diện thực hiện các thuật toán Sort



## 4.3. Giao diện Instrucion



## 4.4. Giao diện About



# 

# Chương 5

# CÀI ĐẶT VÀ KIỂM THỬ

## 

## 5.1. Các công nghệ sử dụng

### 5.1.1. Ngôn ngữ lập trình c#

C# là một ngôn ngữ cấp cao và thuần hướng đối tượng, hỗ trợ rất mạnh trong việc xây dựng các ứng dụng trên môi trường Windows. Đối với những lập trình viên đã từng sử dụng qua C, C++, hoặc Java, việc nắm bắt và sử dụng C# một cách hiệu quả là khá dễ dàng và ít tốn thời gian.

### 5.1.2. Bunifu framework

Bunifu framework giúp người lập trình dễ dàng thiết kế giao diện bắt mắt

Bunifu framework là một thư viện hỗ trợ việc thiết kế cho windows form đơn giản hơn. Với bunifu framework .NET, thiết kế winform chưa bao giờ dễ như thế.

[](https://1.bp.blogspot.com/-0UqslEEk08Y/WFUQukWcubI/AAAAAAAAAIA/mmymK2MAB60nE7z3mD9w8_Zj40MxwRPLwCLcB/s1600/Bunifu+UI++banneffr.png)

Với bunifu  bạn có thể làm nhiều thứ dễ dàng hơn như :

* Slide menu hoạt hình (đóng mở)
* Nền winform đẹp hơn với Gradient background - Tô màu cầu vồng cho form (trộn 3 màu)
* Hỗ trợ Flat UI - Thiết kế giao diện phẳng như Metro Design
* Nhiều chức năng khác như progressbar quay tròn,biểu đồ ,rất tiện lợi.

## 5.2. Các hàm xử lý chính

### 5.2.1. Button Node

- Để tạo một phần tử ta dùng một Button và thêm vào đó các thuộc tính mà phần tử cần: vị trí hiện tại, giá trị,… Ta đặt nên cho một phần tử là một Node . Ngoài ra mỗi Node còn chứa một TextBox bên trong nó có nhiệm vụ để cho người dùng có thể nhập giá trị cho Node mỗi khi nhấn vào (nhập tay).

|  |
| --- |
| class ButtonNode : Button  {  public int giaTri;  public int vitriHienTai;  public TextBox nhapTayTexbox;  public Node(int vitrihientai, int giatri)  {  // Node : property + event  this.FlatStyle = FlatStyle.Flat;  this.BackColor = ThamSo.MauNenNode;  this.TextAlign = ContentAlignment.MiddleCenter;  this.Size = new Size(ThamSo.KichCoNode, ThamSo.KichCoNode);  this.Padding = new Padding(0);  this.Font = new Font("Consolas", ThamSo.KichCoNode / 3.2f, FontStyle.Bold);  this.UseCompatibleTextRendering = true;    this.Text = giatri.ToString();  giaTri = giatri;  vitriHienTai = vitrihientai;  this.GotFocus += new EventHandler(Node\_GotFocus);  // NhapTay TextBox : property + event  nhapTayTexbox = new TextBox();  nhapTayTexbox.MaxLength = 2;  nhapTayTexbox.TextAlign = HorizontalAlignment.Center;  nhapTayTexbox.BorderStyle = BorderStyle.Fixed3D;  nhapTayTexbox.Visible = false;  nhapTayTexbox.Size = new Size(ThamSo.KichCoNode, ThamSo.KichCoNode);  nhapTayTexbox.Font = new Font("Consolas", ThamSo.KichCoNode/2, FontStyle.Bold);  this.Controls.Add(nhapTayTexbox);  nhapTayTexbox.KeyPress += new KeyPressEventHandler(nhapTayTexbox\_KeyPress);  nhapTayTexbox.KeyDown += new KeyEventHandler(nhapTayTexbox\_KeyDown);  nhapTayTexbox.TextChanged += new EventHandler(nhapTayTexbox\_TextChanged);  nhapTayTexbox.LostFocus += new EventHandler(nhapTayTexbox\_LostFocus);  }  } |

- Để xử lý việc nhập giá trị bằng tay cho Node ta cần một số hàm chính sau:

* Hàm Node\_GotFocus : mỗi khi Node được focus vào thì textbox nhập tay hiện lên cho người dùng nhập.

|  |
| --- |
| private void Node\_GotFocus(object sender, EventArgs e)  {  if (nhapTayTexbox.Enabled == true) // Nếu textbox bị tắt (khi node đang sắp xếp) thì texbox không đc bật lên để sửa  {  nhapTayTexbox.BackColor = this.BackColor;  nhapTayTexbox.Visible = true;  nhapTayTexbox.Text = this.Text;  nhapTayTexbox.SelectAll();  nhapTayTexbox.Focus();  }  } |

* Hàm nhapTayTexbox\_LostFocus : khi textobx nhập tay bị lost focus thì tắt textbox đi và lưu giá trị của textbox vào trong Node. Ngoài ra còn chạy một delegate để thông báo giá trị Node vừa được thay đổi.

|  |
| --- |
| public static Action NodeValueChangedHandler;  private void nhapTayTexbox\_LostFocus(object sender, EventArgs e)  {  nhapTayTexbox.Visible = false;  this.Text = nhapTayTexbox.Text;  this.giaTri = int.Parse(nhapTayTexbox.Text);  NodeValueChangedHandler();  } |

* Hàm nhapTayTexbox\_TextChanged : không cho người dùng nhập giá trị rỗng.

|  |
| --- |
| private void nhapTayTexbox\_TextChanged(object sender, EventArgs e)  {  if (nhapTayTexbox.Text.Count() == 0)  {  nhapTayTexbox.Text = "0";  nhapTayTexbox.SelectAll();  nhapTayTexbox.Focus();  }  } |

* Hàm nhapTayTexbox\_KeyDown : nếu nhấn Enter thì tắt textbox nhập tay và lưu giá trị vào Node.

|  |
| --- |
| private void nhapTayTexbox\_KeyDown(object sender, KeyEventArgs e)  {  if (e.KeyCode == Keys.Enter)  {  nhapTayTexbox.Visible = false;  this.Text = nhapTayTexbox.Text;  }  } |

* Hàm nhapTayTexbox\_KeyPress : cho cho phép nhập các phím là số khi nhập giá trị vào textbox.

|  |
| --- |
| private void nhapTayTexbox\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar != '\b') // nếu là nút BackSpace thì bỏ qua bước check này --> cho phép nút Backspace hoạt động  {  e.Handled = !char.IsNumber(e.KeyChar);  // Nếu : Handled == true thì event bị hủy  // Nhưng : [isNumber(True) + not] = false --> Handled = false --> cho phép nhập --> nếu là số thì cho phép nhập  if (nhapTayTexbox.Text.Count() == 0)  {  nhapTayTexbox.Text = "0";  }  }  } |

### 5.2.2. Hàm tạo mảng

- Hàm tạo mảng random

private void RandomGenerateButton(object sender, EventArgs e)

{

if (int.Parse(NumberOfElementTxt.Text) > 10) NumberOfElementTxt.Text = "10";

RandomGenerate(int.Parse(NumberOfElementTxt.Text));

}

private void RandomGenerate(int numberofelement)

{

deletebuttonnode();

Random rd = new Random();

M = new int[numberofelement];

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

{

int giaTri = rd.Next(0, 99);

ButtonNode temp = new ButtonNode(i, giaTri);

this.ViewPanel.Controls.Add(temp);

nodeArr.Add(temp);

temp.Location = new Point(ViewPanel.Location.X + i \* 80 - 20\*int.Parse(NumberOfElementTxt.Text), ViewPanel.Location.Y - 40 / 2);

M[i] = giaTri;

}

}

- Hàm tạo mảng bằng cách nhập tay

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

{

if (int.Parse(NumberOfElementTxt.Text) > 10) NumberOfElementTxt.Text = "10";

ManualGenerate(int.Parse(NumberOfElementTxt.Text));

}

private void ManualGenerate(int numberofelement)

{

deletebuttonnode();

M = new int[numberofelement];

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

{

int giaTri = 0;

ButtonNode temp = new ButtonNode(i, giaTri);

this.ViewPanel.Controls.Add(temp);

nodeArr.Add(temp);

temp.Location = new Point(ViewPanel.Location.X + i \* 80 - 20 \* int.Parse(NumberOfElementTxt.Text), ViewPanel.Location.Y - 40 / 2);

M[i] = giaTri;

}

nodeArr[0].Focus();

}

### 5.2.3. Tạo và quản lý thread

-Hàm Dowork

private void backgroundWorker1\_DoWork\_1(object sender, DoWorkEventArgs e) //được thực thi khi gọi hàm RunWorkerSync()

{

if (listSort.selectedValue == "Bubble Sort") BubbleSort(M);

else

if (listSort.selectedValue == "Interchange Sort") InterchangeSort(M);

else

if (listSort.selectedValue == "Insertion Sort") InsertionSort(M);

else

if (listSort.selectedValue == "Selection Sort") SelectionSort(M);

else

if (listSort.selectedValue == "Binary Insertion Sort") BinaryInsertionSort(M);

else

if (listSort.selectedValue == "Shaker Sort") ShakerSort(M);

else

if (listSort.selectedValue == "Heap Sort") HeapSort(M, M.Length);

else

if (listSort.selectedValue == "Quick Sort")

{

CodeSort.idea = ideaSort;

CodeSort.code = showCode;

CodeSort.QuickSort(AscRadioButton.Checked);

step = 0;

string str = " Dãy chưa sắp : ";

for (int u = 0; u < M.Length; u++) str += M[u].ToString() + " ";

saveQuaTrinh.Text += str;

Quick\_Sort(M, 0, M.Length - 1);

}

else

if (listSort.selectedValue == "Shell Sort") ShellSort(M);

else

if (listSort.selectedValue == "Merge Sort")

{

CodeSort.idea = ideaSort;

CodeSort.code = showCode;

CodeSort.MergeSort(AscRadioButton.Checked);

MergeSort1(M);

}

else

{

MessageBox.Show("Please Choose Sort !");

return;

}

}

-Hàm Progress Change

private void backgroundWorker1\_ProgressChanged\_1(object sender, ProgressChangedEventArgs e) //được thực thi khi gọi hàm ReportProgess()

{

//Cập nhật giao diện thời gian thực xong chuyển đến hàm dowork

Status st = e.UserState as Status;

if (st == null) return;//không làm gì cả

//dừng đã làm rồi

if (st.Type == LoaiDiChuyen.DUNG)//nếu dừng tiến trình thì thay đổi giá trị của 2 nút trong mảng

{

ButtonNode tam = nodeArr[st.Vt2];

nodeArr[st.Vt2] = nodeArr[st.Vt1];

nodeArr[st.Vt1] = tam;

return;

}

Button btn1 = nodeArr[st.Vt1];

Button btn2 = nodeArr[st.Vt2];

if (st.Type == LoaiDiChuyen.DI\_LEN\_DI\_XUONG)

{

btn1.Top = btn1.Top + 1;//Nút 1 đi lên

btn2.Top = btn2.Top - 1;//Nút 2 đi xuống

}

else if (st.Type == LoaiDiChuyen.QUA\_PHAI\_QUA\_TRAI)

{

btn1.Left = btn1.Left - 1;//nút 1 qa phải

btn2.Left = btn2.Left + 1;//Nút 2 di chuyển qua trái

}

else if (st.Type == LoaiDiChuyen.DI\_XUONG\_DI\_LEN)

{

btn1.Top = btn1.Top - 1;//Nút 1 đi xuống

btn2.Top = btn2.Top + 1;//Nút 2 đi lên

}

}

-Hàm Complete

private void backgroundWorker1\_RunWorkerCompleted\_1(object sender, RunWorkerCompletedEventArgs e) //hàm kết thúc tiểu trình, được thực thi sau khi hàm DoWork thực thi xong, hoặc người dùng chọn hủy tiến trình

{

MessageBox.Show("Mô phỏng kết thúc");

deletebuttonnode();

RandomGenerateBtn.Enabled = true;

ManualGenerateBtn.Enabled = true;

StartBtn.Enabled = true;

}

### 5.2.4. Hàm huỷ tiến trình

-Hàm này hủy quá trình đang sắp xếp bằng cách xóa mảng hiện có

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

{

if (backgroundWorker1.IsBusy) { backgroundWorker1.CancelAsync(); }

RandomGenerateBtn.Enabled = true;

ManualGenerateBtn.Enabled = true;

StartBtn.Enabled = true;

}

## 5.3. Kiểm thử

### 5.3.1. 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.

### 5.3.2. Intergration Test

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

### 5.3.3. System Test

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.

# Chương 6

# TỔNG KẾT

## 6.1. Kết quả đạt được.

- Hoàn thành tất các các chức năng của phần mềm đã đặt ra :

+ Khởi tạo dãy ( ngẫu nhiên hoặc nhập tay).

+ Cung cấp đủ 10 giải thuật sắp xếp cho người dùng lựa chọn.

+ Mô phỏng quá trình sắp xếp một cách trực quan.

+ Cung cấp thông tin về lí thuyết thuật toán và code minh họa.

+ Ghi lại nhật ký sắp xếp.

- Học được cách thức, quá trình thực hiện một dự án phần mềm.

- Hiểu và ứng dụng cách sử dụng và quản lý Thread vào phần mềm.

- Nắm được cách tạo hiệu ứng chuyển động trên form, cách quản lý các sự kiện bàn phím.

## 6.2. Hạn chế.

- Khung hiển thị còn nhỏ nên số lượng phần tử còn hạn chế.

- Gặp nhiều khó khăn khi xử lý Thread.

- Giao diện đồ họa chưa đẹp.

## 6.3. Hướng phát triển.

- Mở rộng khung hiển thị, tăng số lượng phần tử có thể mô phỏng được.

- Tạo thêm các hiệu ứng trực quan như đổi màu, âm thanh để làm rõ quá trình mô phỏng.

- Nâng cấp giao diện.

- Cài đặt thêm tính năng debug.

- Cài đặt thêm tính năng so sánh hai thuật toán sắp xếp cùng lúc.

- Tạo thêm 1 thanh tùy chỉnh tốc độ của quá trình mô phỏng

6.4. Bảng phân rã công việc

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Công việc | Người thực hiện | Mức độ hoàn thành |
| 1 | Tìm hiểu sơ bộ và đăng ký đồ án | Huỳnh Tấn Duy | 100% |

**I. Lập trình.**

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | Thiết kế giao diện | Huỳnh Phương Duy | 100% |
| 3 | Tạo lớp ButtonNode | Huỳnh Tấn Duy | 100% |
| 3 | Sự kiện tạo mảng | Huỳnh Tấn Duy | 100% |
| 4 | Sự kiện dừng, hủy tiến trình | Huỳnh Tấn Duy | 100% |
| 5 | Tạo thread để mô phỏng quá trình sắp xếp | Huỳnh Tấn Duy | 100% |
| 6 | Minh họa Bubble Sort | Huỳnh Tấn Duy | 100% |
| 7 | Minh họa Shell Sort, Merge Sort | Huỳnh Phương Duy | 100% |
| 8 | Minh họa Interchange Sort, Insertion Sort, Binary Insertion Sort, Selection Sort, Shaker Sort, Heap Sort, Quick Sort | Nguyễn Văn Vinh | 100% |
| 9 | Hiển thị code minh họa | Nguyễn Văn Vinh | 100% |
| 10 | Hiển thị nhật ký sắp xếp | Nguyễn Văn Vinh | 100% |
| 11 | Hiện thị lí thuyết thuật toán | Huỳnh Phương Duy | 100% |
| 12 | Form hướng dẫn sử dụng phần mềm | Huỳnh Phương Duy | 100% |
| 13 | Form thông tin phần mềm | Huỳnh Phương Duy | 100% |

**II. Viết báo cáo**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Chương 1 + 2 | Huỳnh Phương Duy | 100% |
| 2 | Chương 3 + 6 | Huỳnh Tấn Duy | 100% |
| 3 | Chương 4 + 5 | Nguyễn Văn Vinh | 100% |

**TÀI LIỆU THAM KHẢO**

1.Windows Form Programing in C# : <http://www.mediafire.com/file/vd1zyi0qngt/Windows+Form+Programming+in+CSharp.chm>

2. BackgroundWorker: <https://msdn.microsoft.com/en-us/library/system.componentmodel.backgroundworker(v=vs.110).aspx>

3. Các thuật toán sắp xếp : https://en.wikipedia.org/wiki/Sorting\_algorithm