BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC TẬP CƠ SỞ

THUẬT TOÁN SẮP XẾP SHAKER SORT VÀ SHELL SORT

Giảng viên hướng dẫn: Bùi Thị Hồng Minh

Sinh viên thực hiện: Huỳnh Công Lợi

Mã số sinh viên: 62133105

KHÁNH HÒA – 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC TẬP CƠ SỞ

THUẬT TOÁN SẮP XẾP SHAKER SORT VÀ SHELL SORT

Giảng viên hướng dẫn: Bùi Thị Hồng Minh

Sinh viên thực hiện: Huỳnh Công Lợi

Mã số sinh viên: 62133105

Khánh Hòa, tháng 01/2022

LỜI CẢM ƠN

Để có thể hoàn thành đợt thực tập lần này, em xin chân thành cảm ơn đến quý thầy cô khoa Công nghệ Thông tin đã tạo điều kiện hỗ trợ và giúp đỡ em trong quá trình học tập và nghiên cứu đề tài này.

Qua đây, em xin chân thành cảm ơn cô Bùi Thị Hồng Minh, người đã trực tiếp quan tâm và hướng dẫn chúng em hoàn thành tốt đợt thực tập trong thời gian qua.

Do kiến thức còn hạn chế và thời gian thực hiện còn ngắn nên bài báo cáo của em còn nhiều thiếu sót, kính mong sự góp ý của quý thầy cô.

Em xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC HÌNH

TÓM TẮT

Sắp xếp là khái niệm cơ bản trong tin học nói chung và trong chuyên ngành lập trình nói riêng. Sắp xếp là quá trình bố trí lại các phần tử trong một tập hợp theo một trình tự nào đó nhằm mục đích giúp quản lý và tìm kiếm các phần tử dễ dàng và nhanh chóng hơn. Điển hình như việc sắp xếp họ và tên của học sinh theo bảng chữ cái trong học tập, sắp xếp điểm số từ thấp tới cao trong các cuộc thi, ... Như vậy có rất nhiều mục đích cần phải sắp xếp các phần tử theo một trình tự.

Trong khoa học máy tính và trong toán học, thuật toán sắp xếp là một thuật toán sắp xếp các phần tử của một danh sách (*hoặc một mảng*) theo thứ tự (*tăng hoặc giảm*). Và để dễ dàng cho việc nghiên cứu và học tập thì người ta thường gán các phần tử được sắp xếp là các chữ số.

Các thuật toán **Shaker Sort** và **Shell Sort** được cài đặt trong lần thực tập lần này giúp hiểu rõ hơn về những thuật toán nâng cao được phát triển từ những thuật toán sắp xếp đơn giản, dễ hiểu thường được dạy trong khoa học máy tính như thuật toán Bubble Sort, Insertion Sort ,... Quy trình thực hiện được trải qua các bước từ thiết kế giao diện, cài đặt thuật toán, hiển thị kết quả đầu ra trên màn hình hiển thị đều được thực hiện trên môi trường winform(C#) thông qua ứng dụng Visual Studio có kết hợp với thư viện Windows.Froms. Kết quả của việc cài đặt thuật toán đáp ứng được các yếu cầu đặt ra của đợt thực tập cơ sở lần này.

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

Cùng với sự phát triển của công nghệ hiện nay thì các ứng dụng của công nghệ thông tin ngày càng nhiều và có mặt hầu hết trong các lĩnh vực cuộc sống đi cùng với nó thì các thuật toán được sử dụng trong các ứng dụng này đòi hỏi cần phải ngày càng được điều chỉnh và cải tiến để phù hợp với từng lĩnh vực. Với đề tài: "**Xây dựng chương trình mô phỏng thuật toán sắp xếp dãy số bất kỳ bằng thuật toán shaker sort và shell sort**", ý nghĩa của các thuật toán sắp xếp trong khoa học máy tính và trong toán học là không hề nhỏ, đặc biệt đối với chuyên ngành lập trình nói riêng thì nó có ý nghĩa vô cùng quan trọng.

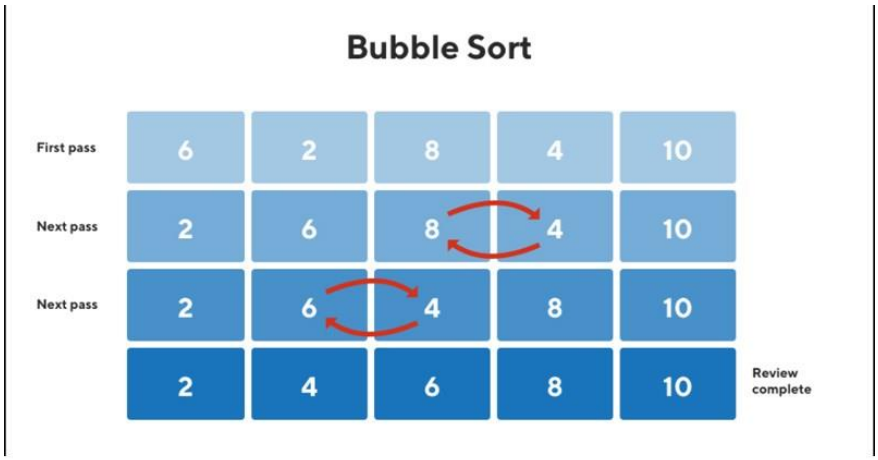
1.1 Thuật toán sắp xếp cơ bản

Các thuật toán sắp xếp như Bubble Sort, Insertion Sort, Selection Sort,… Là các phương pháp sắp xếp đơn giản, dễ hiểu thường được dạy trong khoa học máy tính.

a) Thuật toán Bubble Sort (Sắp xếp nổi bọt)

Sắp xếp nổi bọt (bubble sort) là một thuật toán sắp xếp như cách sắp xếp các em học sinh trong một hàng theo thứ tự từ thấp đến cao.

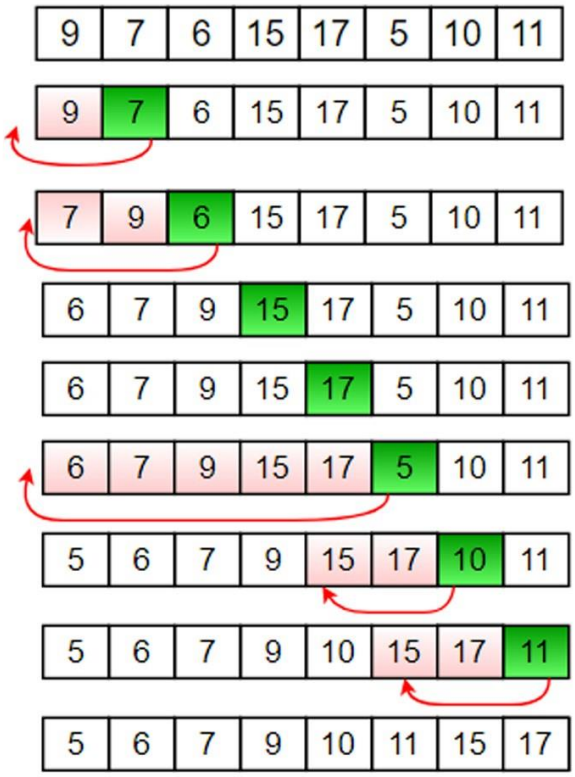
Giải thuật bắt đầu từ đầu của tập dữ liệu. Nó so sánh hai phần tử đầu, nếu phần tử đứng trước lớn hơn phần tử đứng sau thì đổi chỗ chúng cho nhau. Tiếp tục làm như vậy với cặp phần tử tiếp theo cho đến cuối tập hợp dữ liệu. Sau đó nó quay lại với hai phần tử đầu cho đến khi không còn cần phải đổi chỗ nữa.



Hình 1.1 Mô phỏng thuật toán Bubble Sort

b) Thuật toán Insertion Sort (Sắp xếp Chèn)

Sắp xếp chèn (insertion sort) là một thuật toán sắp xếp bắt chước cách sắp xếp quân bài của những người chơi bài.



Hình 1.2 Mô phỏng thuật toán Insertion Sort

Ý tưởng của thuật toán này như sau: ta có mảng ban đầu gồm phần tử A[0] xem như đã sắp xếp, ta sẽ duyệt từ phần tử 1 đến n – 1, tìm cách chèn những phần tử đó vào vị trí thích hợp trong mảng ban đầu đã được sắp xếp.

1.2 Thuật toán Shaker Sort

Shaker Sort cũng dựa trên nguyên tắc đổi chỗ trực tiếp nhưng tìm các khắc phục nhược điểm của **Bubble Sort**.

Sau khi đưa phần tử nhỏ nhất về đầu mảng sẽ đưa phần tử lớn nhất về cuối dãy. Do đưa các phần tử về đúng vị trí ở cả hai đầu nên **Shaker Sort** sẽ giúp cải thiện thời gian sắp xếp dãy số do giảm được độ lớn của mảng đang xét ở lần so sánh kế tiếp.

Giải thuật này khá hiệu quả với các tập dữ liệu có kích cỡ trung bình khi mà độ phức tạp trường hợp xấu nhất và trường hợp trung bình là **O(n^2)**, với n là số phần tử.

Ý tưởng

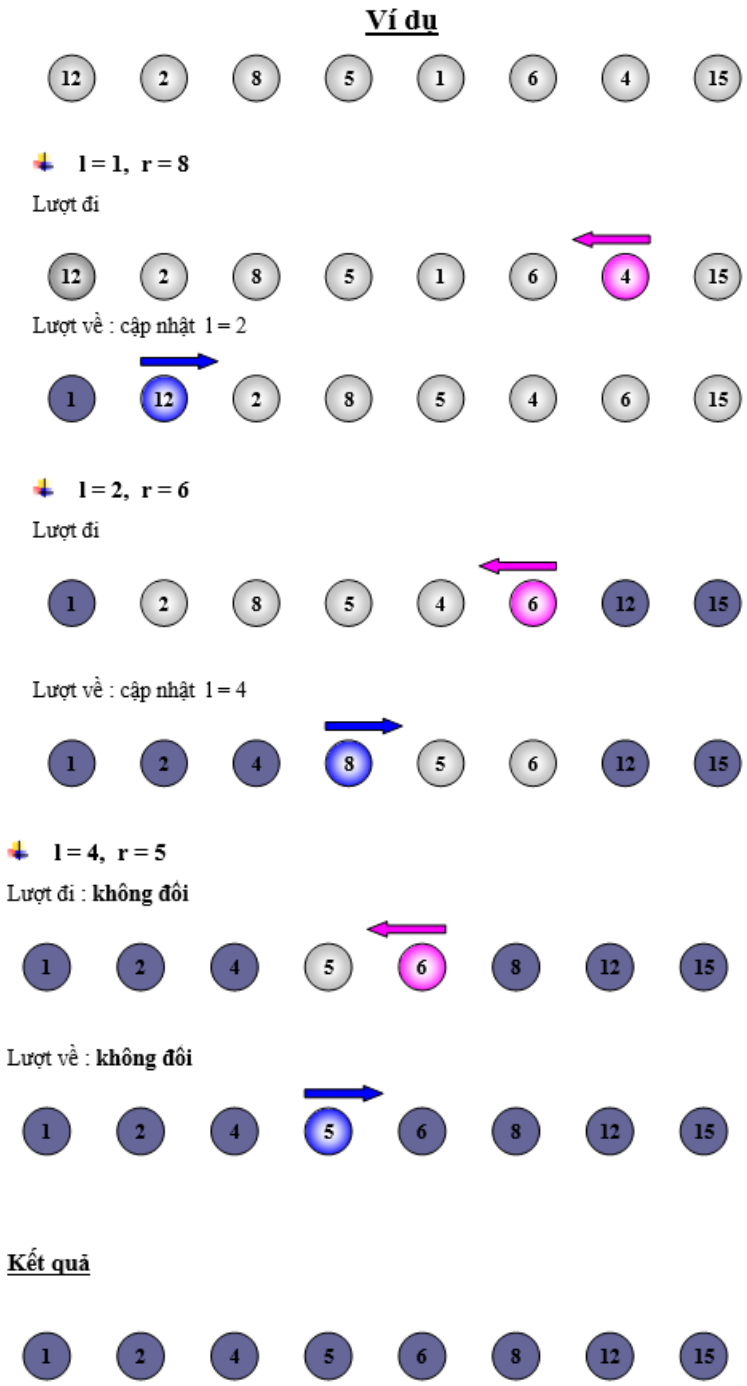
Trong mỗi lần sắp xếp, duyệt mảng theo 2 lượt từ 2 phía khác nhau.

+ Lượt đi: đẩy phần tử nhỏ về đầu mảng

+ Lượt về: đẩy phần tử lớn về cuối mảng

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

Cách **shaker Sort** hoạt động:



Hình 1.3 Mô phỏng quá trình sắp xếp của Shaker Sort

Code mẫu C++:

```
void ShakerSort(int a[], int n)
        {
            int Left = 0;
            int Right = n = 1;
            int k = 0;
            while (Left < Right)
            {
                for (int i = Left; i < Right; i++)
                {
                    if (a[i] > a[i + 1])
                    {
                        swap(a[i], a[i + 1]);
                        k = i;
                    }
                }
                Right = k;
                for (i = Right; i > Left; i--)
                {
                    if (a[i] < a[i - 1])
                    {
                        swap(a[i], a[i - 1]);
                        k = i;
                    }
                }
                Left = k;
            }
```

1.3 Thuật toán Shell Sort

Shell sort là một giải thuật sắp xếp mang lại hiệu quả cao dựa trên giải thuật sắp sếp chèn (**Insertion sort)**.

Giải thuật này tránh được các trường hợp phải tráo đổi vị trí của hai phần tử xa nhau trong giải thuật sắp sếp chọn. (nếu như phần tử nhỏ hơn ở vị trí bên phải khá xa so với phần tử lớn hơn ở vị trí bên trái).

Giải thuật này khá hiệu quả với các tập dữ liệu có kích cỡ trung bình khi mà độ phức tạp trường hợp xấu nhất và trường hợp trung bình là **O(n)**, với n là số phần tử.

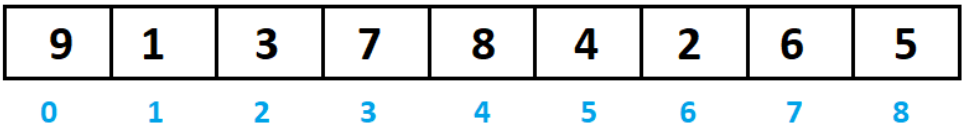
Ý tưởng

Đầu tiên, giải thuật này sử dụng giải thuật sắp xếp chọn trên các phần tử có khoảng cách xa nhau, sau đó sắp xếp các phần tử có khoảng cách hẹp hơn. Khoảng cách này còn được gọi là **khoảng (interval).**

interval sẽ nhận giá trị lần lượt là n/2, n/4, n/8 cho đến khi interval = 1.

Cách **shell Sort** hoạt động:

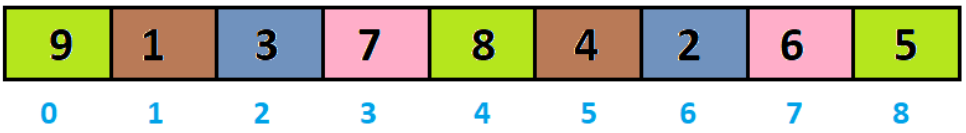
Ví dụ sắp xết dãy a = [9, 1, 3, 7, 8, 4, 2, 6, 5] thành dãy tăng dần.



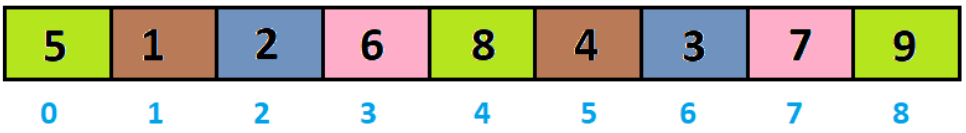
Hình 1.4 Mô phỏng quá trình sắp xếp của Shell Sort

Với interval = 9/2 = 4, ta sẽ chia dãy thành các dãy con với các số cách nhau một khoảng là interval: [9, 8, 5], [1, 4], [3, 2] và [7, 6].

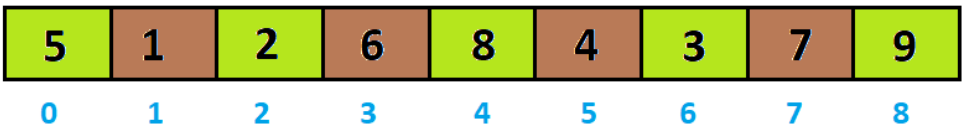
Sắp xếp những dãy con này theo cách **sắp xếp chèn (Insertion Sort).**



Sau khi sắp xếp các dãy con dãy sẽ thành.

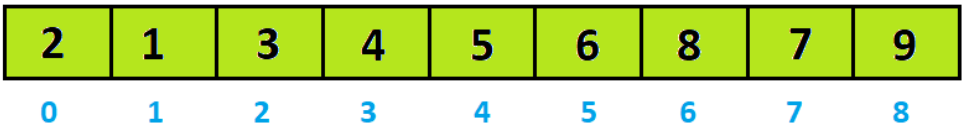


Với interval = 9/4 = 2, ta sẽ chia dãy thành các dãy con với các số cách nhau một khoảng là interval: [5, 2, 8, 3, 9], [1, 6, 4, 7].

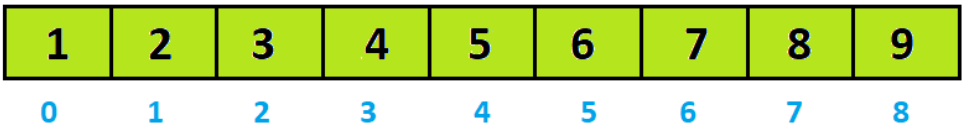


Sau khi sắp xếp các dãy con dãy sẽ thành.

Với interval = 9/8 = 1, lúc này interval = 1 ta áp dụng sắp xếp chèn với cả dãy a:



Dãy sau khi sắp xếp là:



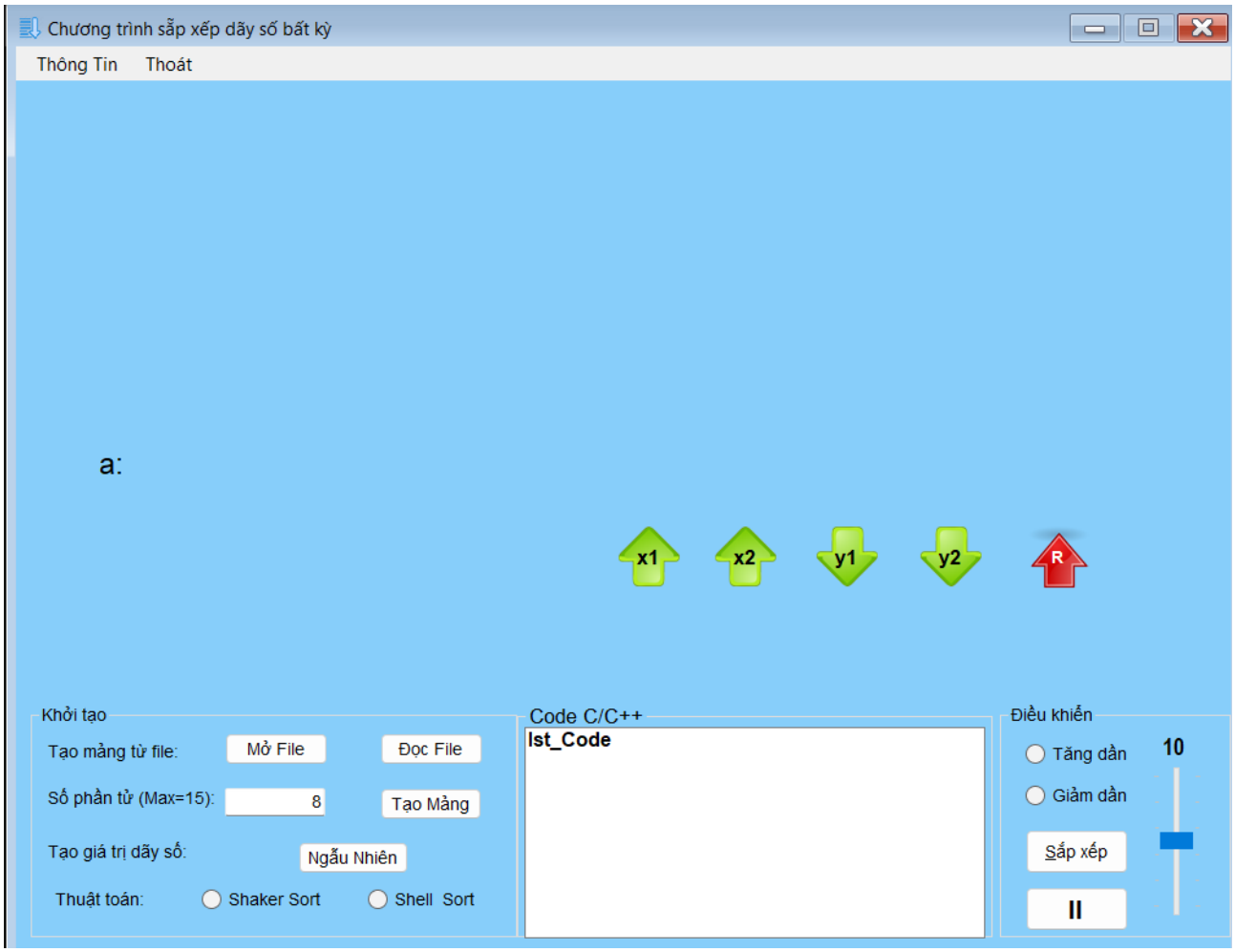
Code mẫu C++:

```
void ShellSort(int a[], int n, int h[], int k){
           int step, i, pos, x, len;
           for (step = 0; step < k; step++){
               len = h[step];
               for (i = len; i < n; i++){
                   x = a[i];
                   pos = i - len;
                   while ((pos >= 0) && (x < a[pos])){
                       a[pos + len] = a[pos];
                       pos = pos - len;
                   }
                   a[pos + len] = x;}
                   }
                  }
```

CHƯƠNG 2: TRIỂN KHAI

2.1 Thiết kế giao diện

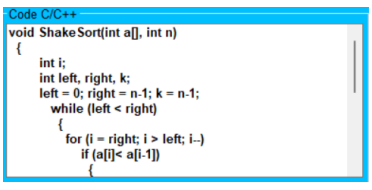
Các đối tượng được thiết lập và cài đặt trong giao diện Winform của Visual Studio.



Hình 2.1 Giao diện làm việc chính

Về từng phần trong giao diện gồm có:

Phần khởi tạo mảng dữ liệu và danh sách các thuật toán sắp xếp.

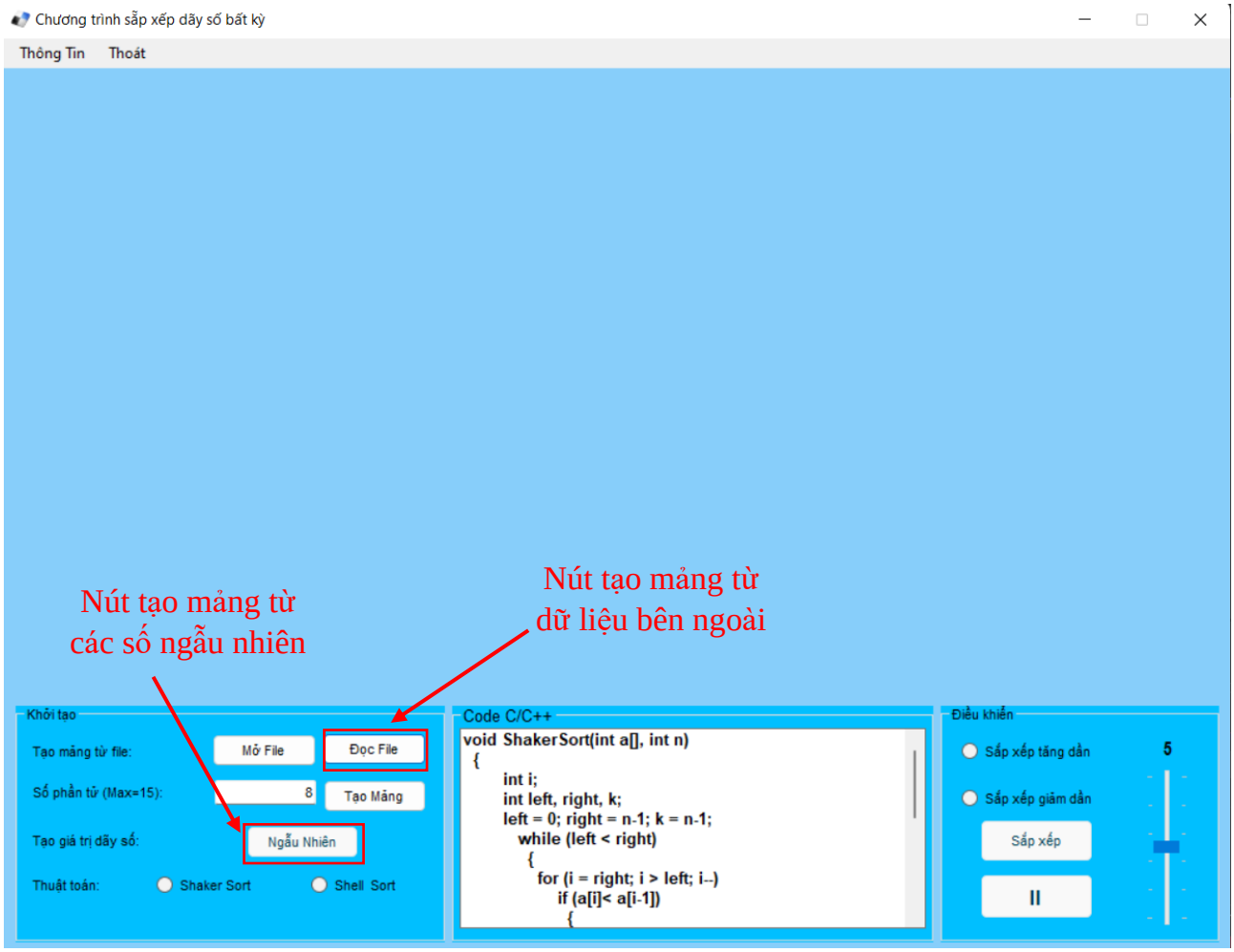


Phần hiện thị code minh họa các thuật toán sắp xếp dưới dạng C/C++.

Phần điều khiển các phương thức sắp xếp và tốc độ sắp xếp.

2.2 Khởi tạo code

2.2.1 Các phương thức nhập dữ liệu



Hình 2.2 Giao diện khởi tạo dữ liệu

Về phương pháp nhập dữ liệu ta có hai phương pháp đó là nhập từ file và tạo ngẫu nhiên. Ở phần tạo dữ liệu từ file ta có thể mở file để tùy chỉnh mảng trước khi đọc mảng dữ liệu từ file lên giao diện.

**Mở file*

Sử dụng lớp đối tượng Process mở file bằng file name và đường dẫn đến file dữ liệu.

```
Process p = new Process();
ProcessStartInfo ps = new ProcessStartInfo();
ps.FileName = "NotePad.exe";
ps.Arguments = "D:\\TTCS\\SOURCE\\demo_sort\\bin\\Debug\\TEST.txt";
```

**Đọc file*

Sử dụng lớp đối tượng StreamReader chỉ đường dẫn đến file chứa dữ liệu. Tiến hành đọc dòng đầu tiên để lấy số lượng phần tử của mảng dữ liệu.

```
StreamReader Re =
File.OpenText("D:\\TTCS\\SOURCE\\demo_sort\\bin\\Debug\\TEST.txt");
            string input = null;
            int i = 0;
            int kt = 0;
            while ((kt < 1) && ((input = Re.ReadLine()) != null))
            {
                Spt = Convert.ToInt32(input);
                kt++;
            }
```

**Tạo ngẫu nhiên*

Với phần tạo ngẫu nhiên này trước tiên ta phải tạo 1 mạng các số sau đó mới bắt đầu lấy ngẫu nhiên các số vào.

Về phần tạo mảng dữ liệu, bắt đầu với bước kiểm tra số phần tử đã được nhập vào có ít hơn 2 hay nhiều hơn 15 phần tử hay không.

```
          if ((Spt < 2) || (Spt > 15))
         {
               lbl_A.Visible = false;
               MessageBox.Show("2 <= Số Phần Tử <= 15");
              this.txt_sophantu.Clear();
                Da_tao_mang = false;
                return;}
```

Tiếp theo là bước thiết lập thuộc tính node ứng với số phần tử đã được nhập vào ứng với các trường hợp để danh sách các node hiện trên giao diện sẽ nằm ở giữa giao diện.

```
                switch (Spt){
                case 15:
                case 14:
                case 13:
                case 12:
                case 11:
                 Kich_thuoc = 40;
                 Co_chu = 18;
                 Khoang_cach = 18;
                 Canh_le = (1024 - Kich_thuoc * Spt - Khoang_cach * (Spt - 1)) / 2;
                    break;
                case 10:
                case 9:
                case 8:
                case 7:
                case 6:
                case 5:
                case 4:
                case 3:
                case 2:
                 Kich_thuoc = 50;
                 Co_chu = 25;
                 Khoang_cach = 40;
                 Canh_le = (1024 - Kich_thuoc * Spt - Khoang_cach * (Spt - 1)) / 2;
                    break;}
```

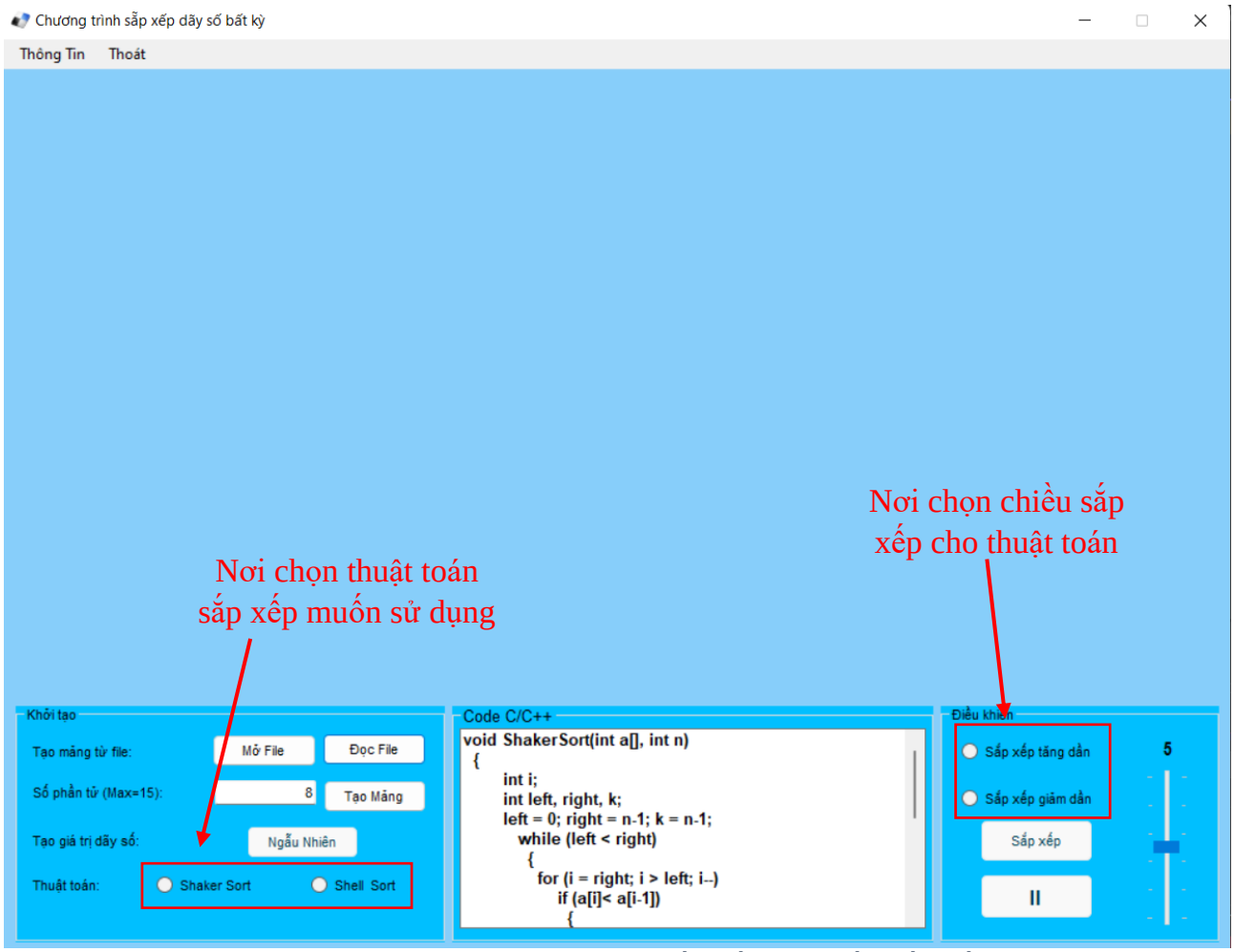
Sau đó tiến hành tạo mảng dữ liệu với các mảng sẽ là các textbox và chỉ số của mảng là label.

```
Chi_so = new Label[Spt];
a = new int[Spt];
Node = new TextBox[Spt];
```

Sau khi tạo xong mảng ta sẽ gọi lớp đối tượng Random để lấy các số ngẫu nhiên chèn vào trong mảng.

```
Random r = new Random();
```

2.2.2 Các hàm sắp xếp



Hình 2.3 Giao diện thuật toán sắp xếp và chiều sắp xếp

SHAKER SORT

Đầu tiên với thuật shaker sort ta sẽ tiến hành sắp xếp trên 2 chiều thuận và ngược tùy vào tùy chọn của người sử dụng.

Với thuật toán **shaker sort chiều sắp xếp tăng dần** ta sẽ xét theo các trường hợp sau:

**Node a[i] < node a[i – 1]*

```
                        if (a[i] < a[i - 1])
                    {
                        lst_Code.SelectedIndex = 10;
                        Tre(40 * Toc_do);
                        Hoan_vi(ref a[i], ref a[i - 1]);
                        this.Invoke((MethodInvoker)delegate
                        {
                            Hoan_Vi_Node(Node[i], Node[i - 1]);
                        });
                        Tam_dung();
                        Hoan_Tri_Node(i, i - 1);
                        lst_Code.SelectedIndex = 11;

                        k = i;
                        //Thiết lập vị trí của k
                        Mui_ten_do_len.Visible = true;
                        Mui_ten_do_len.Location   =   new   Point((Canh_le   +
(Kich_thuoc + Khoang_cach) * k) + (Kich_thuoc / 2) - 30, Node[k].Location.Y + 2 *
Kich_thuoc + 65);
                        Mui_ten_do_len.Text = "K=" + k;
                        Mui_ten_do_len.Refresh();
                        Tre(50 * Toc_do);
                    }}
```

**Node a[i] > node a[i + 1]*

```
                         if (a[i] > a[i + 1])
                    {
                        lst_Code.SelectedIndex = 17;
                        Tre(40 * Toc_do);
                        Hoan_vi(ref a[i], ref a[i + 1]);
                        this.Invoke((MethodInvoker)delegate
                        {
                            Hoan_Vi_Node(Node[i], Node[i + 1]);
                        });
                        Tam_dung();
                        Hoan_Tri_Node(i, i + 1);
                        lst_Code.SelectedIndex = 18;

                        k = i;
                        //Thiết lập vị trí của k
                        Mui_ten_do_len.Visible = true;
                        Mui_ten_do_len.Location   =   new   Point((Canh_le   +
(Kich_thuoc + Khoang_cach) * k) + (Kich_thuoc / 2) - 30, Node[k].Location.Y + 2 *
Kich_thuoc + 65);
                        Mui_ten_do_len.Text = "K=" + k;
                        Mui_ten_do_len.Refresh();
                        Tre(50 * Toc_do);
                    }
```

Với hai trường hợp này ta sẽ tiến hành hoán vị hai node để đưa node có giá trị bé hơn về bên trái và đưa node có giá trị lớn hơn về bên phải nhằm đưa số bé nhất lên đầu và số lớn nhất về cuối. Nếu ngược lại với trường hợp trên thì sẽ giữ nguyên vị trí của node.

**Node đã được sắp xếp (node đã có thứ tự)*

Trường hợp này được thực hiện khi đã thỏa mãn trường hợp trên, sau khi so sánh giá trị của 2 node sẽ tiến hành hoán vị nếu 1 trong 2 node hoán vị đã được sắp xếp trước đó thì

nó sẽ giữ nguyên vị trí và đặt màu cho node đã được sắp xếp. Nếu ngược với trường hợp trên thì sẽ tiến hành hoán vị 2 node.

```
//Thiết lập Node đã có thứ tự
for (i = Spt - 1; i > k; i--)
{
    Dat_mau_node(Node[i], Color.LawnGreen, Color.Black);
    Tre(50 * Toc_do);
}
//
right = k;
```

Với thuật toán **shaker sort chiều sắp xếp giảm dần** thì cũng xét theo các trường hợp như shaker sort tăng dần chỉ khác 2 trường hợp:

**Node a[i] > node a[i - 1]* và *Node a[i] < node a[i + 1]*

Với hai trường hợp này sẽ hoán vị hai node để đưa node có giá trị lớn hơn về bên trái và đưa node có giá trị bé hơn về bên phải nhằm đưa số lớn nhất lên đầu và số bé nhất về cuối. Nếu ngược lại với trường hợp trên thì sẽ giữ nguyên vị trí của node.

SHELL SORT

Tiếp theo với thuật toán shell sort ta cũng sẽ tiến hành sắp xếp trên 2 chiều thuận và ngược tùy vào tùy chọn của người sử dụng.

Với thuật toán **shell sort chiều sắp xếp tăng dần** thì trước tiên ta sẽ tạo mảng con h để chứa các node có kích cỡ bằng ½ mảng chính ứng với các node có khoảng cách h = 2*h. Thực hiện cho đến khi khoảng cách các node h = 1.

```
int k = Convert.ToInt32(Math.Log10(Spt) / Math.Log10(2));
h = new int[k];
h[k - 1] = 1;
for (i = k - 2; i >= 0; i--)
{
    Application.DoEvents();
    h[i] = (2 * h[i + 1]);
}
```

Sau đó sẽ tiến hành chia các mảng con với từng khoảng cách h để sắp xếp, số lần chia này sẽ dừng khi bằng ½ kích cỡ của mảng chính. Khi chia xong ta bắt đầu sắp xếp theo trường hợp sau:

**x < a[pos] với x =a[i] và a[pos] = a[i-len]*

Với trường hợp này sẽ hoán vị các node trong mảng để sắp xếp các node theo giá trị tăng dần sau đó giảm khoảng cách các node h để tạo mảng mới. Nếu ngược lại với trường hợp trên thì sẽ giữ nguyên vị trí của node và giảm khoảng cách các node h để tạo mảng mới.

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

              Application.DoEvents();
              a[pos + len] = a[pos];
              this.Invoke((MethodInvoker)delegate
              {
               if (len == 1)
               {
                  Node_qua_phai(Node[pos], len);
               }
               else
               {
                  Node_di_xuong(Node[pos], Kich_thuoc + 5);
                  Node_qua_phai(Node[pos], len);
                  Node_di_len(Node[pos], Kich_thuoc + 5);
               }
                       });

                       Node[pos + len] = Node[pos];
                       pos = pos - len;

               }
                  Node[pos + len] = x;
```

**Node đã được sắp xếp (node đã có thứ tự)*

Trường hợp này được thực hiện khi thỏa mãn trường hợp, sau khi so sánh giá trị của 2 node sẽ tiến hành hoán vị nếu 1 trong 2 node hoán vị đã được sắp xếp trước đó thì nó sẽ giữ nguyên vị trí và đặt màu cho node đã được sắp xếp. Nếu ngược với trường hợp trên thì sẽ tiến hành hoán vị 2 node.

```
           while (pos >= 0)
                   {
                       Dat_mau_node(Node[pos], Color.Red, Color.White);
                       pos = pos - len;
                   }
```

Với thuật toán **shell sort chiều sắp xếp giảm dần** các bước cũng giống shell sort tăng dần chỉ khác trường hợp.

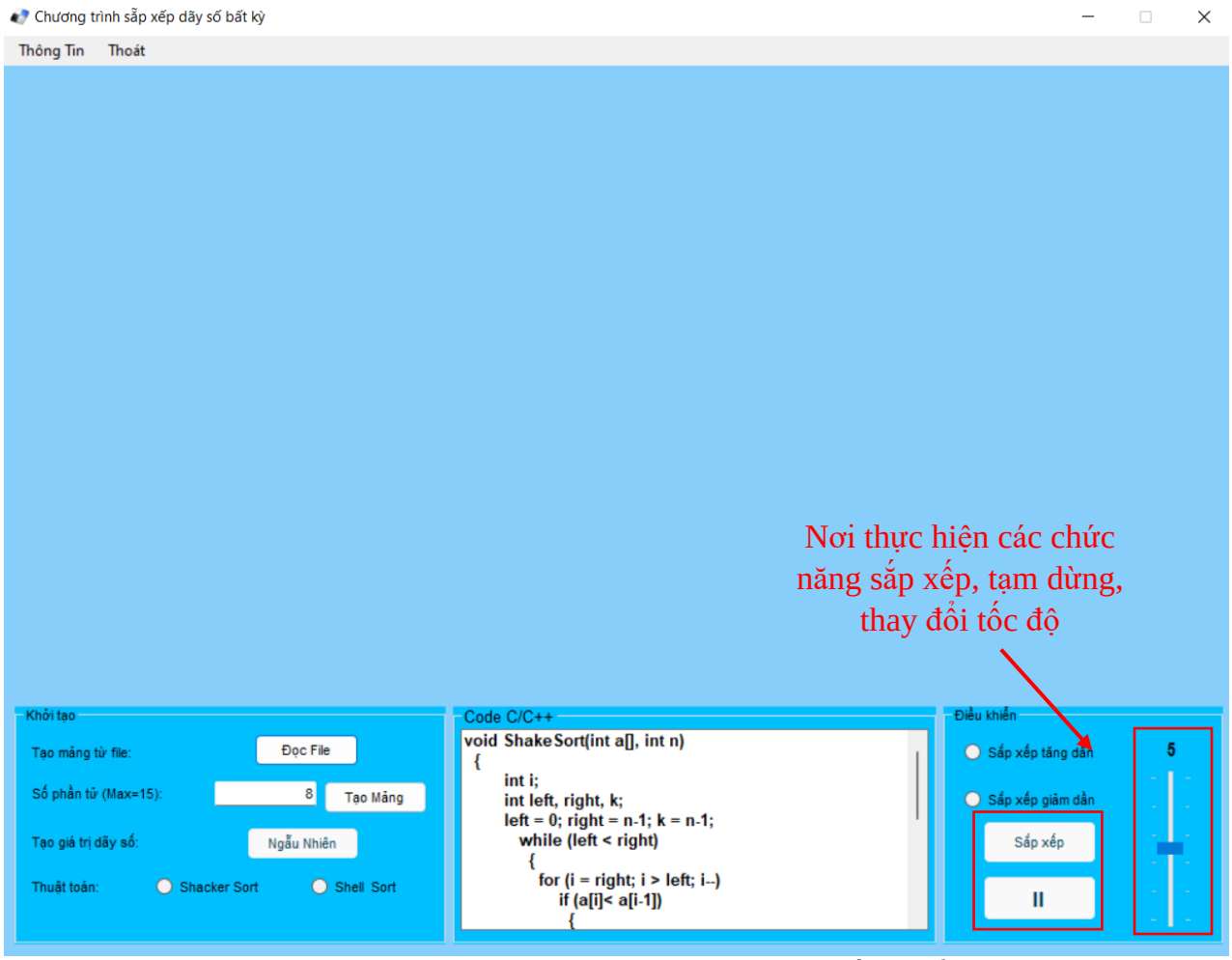
**x > a[pos] với x =a[i] và a[pos] = a[i-len]*

Với trường hợp này sẽ tiến hành hoán vị hai node trong mảng để sắp xếp các node theo giá trị tăng dần sau đó giảm khoảng cách các node h để tạo mảng mới. Nếu ngược lại với trường hợp trên thì sẽ giữ nguyên vị trí của node và giảm khoảng cách các node h để tạo mảng mới.

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

                       Application.DoEvents();
                       a[pos + len] = a[pos];
                       this.Invoke((MethodInvoker)delegate
                       {
```

```
                    if (len == 1)
                    {
                        Node_qua_phai(Node[pos], len);
                    }
                    else
                    {
                        Node_di_xuong(Node[pos], Kich_thuoc + 5);
                        Node_qua_phai(Node[pos], len);
                        Node_di_len(Node[pos], Kich_thuoc + 5);
                    }
                });

                Node[pos + len] = Node[pos];
                pos = pos - len;

            }
```

2.2.3 Các hàm điều khiển



Hình 2.4 Giao diện thực hiện chức năng điều khiển

Đầu tiên với nút sắp xếp sẽ gọi hàm sắp xếp cùng với chiều sắp xếp mà người sử dụng đã lựa chọn.

SHACKE SORT

```
if (rad_shackersort.Checked == true && rad_tang.Checked == true )
{
    ShackeSort_tang();
}
if (rad_shackersort.Checked == true && rad_giam.Checked == true)
{
    ShackeSort_giam();
}
```

SHELL SORT

```
if (rad_shellsort.Checked == true && rad_tang.Checked == true)
{
    ShellSort_tang();
    for (i = 0; i < Spt; i++)
    {
        lbl_status_01.Text += a[i].ToString() + " ";
    }
}
if (rad_shellsort.Checked == true && rad_giam.Checked == true)
{
    ShellSort_giam();
    for (i = 0; i < Spt; i++)
    {
        lbl_status_01.Text += a[i].ToString() + " ";
    }}
```

Tiếp đến là thanh tùy chỉnh tốc độ sắp xếp với tốc độ dưới 10 sẽ luôn có độ trễ nhất định tùy theo từng mức độ chỉ riêng mức 10 là max sẽ chạy liên tục không có độ trễ.

```
Toc_do = (Trb_Toc_do.Maximum - Trb_Toc_do.Value);
lbl_Toc_do.Text = Trb_Toc_do.Value.ToString();
if (Trb_Toc_do.Value == Trb_Toc_do.Maximum){
        lbl_Toc_do.Text = "Max=10";
    }
else if (Trb_Toc_do.Value == Trb_Toc_do.Minimum)
    {
        lbl_Toc_do.Text = "Min=0";
    }
```

Cuối cùng là nút dừng trên phần điều khiển ta sẽ sử dụng biến kiểm tra tạm dừng để thực hiện dừng quá trình sắp xếp.

```
if (btn_Dung.Text == ";")
    {
        btn_Dung.Text = "4";
        KT_tam_dung = true;
        Play_or_Stop();
    }
    else
    {
        btn_Dung.Text = ";";
        KT_tam_dung = false;}
```

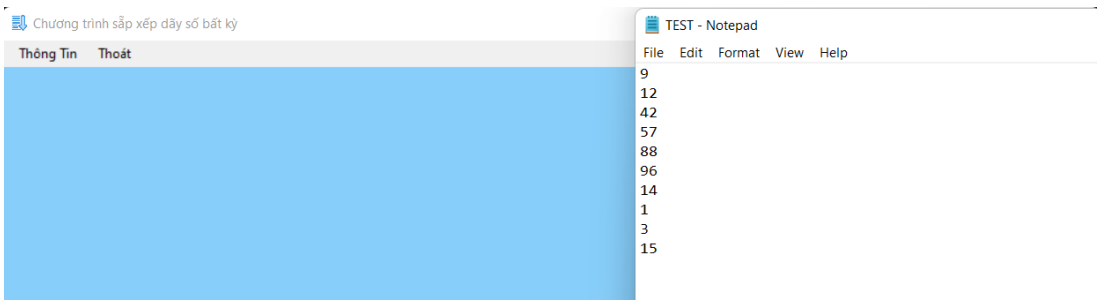
CHƯƠNG 3: KẾT QUẢ ĐẠT ĐƯỢC

Sau khi hoàn thành thiết kế cài đặt giao diện và xử lí code, bắt đầu chạy chương trình như hình (Hình 3.1).

Hình 3.1 Giao diện bắt đầu

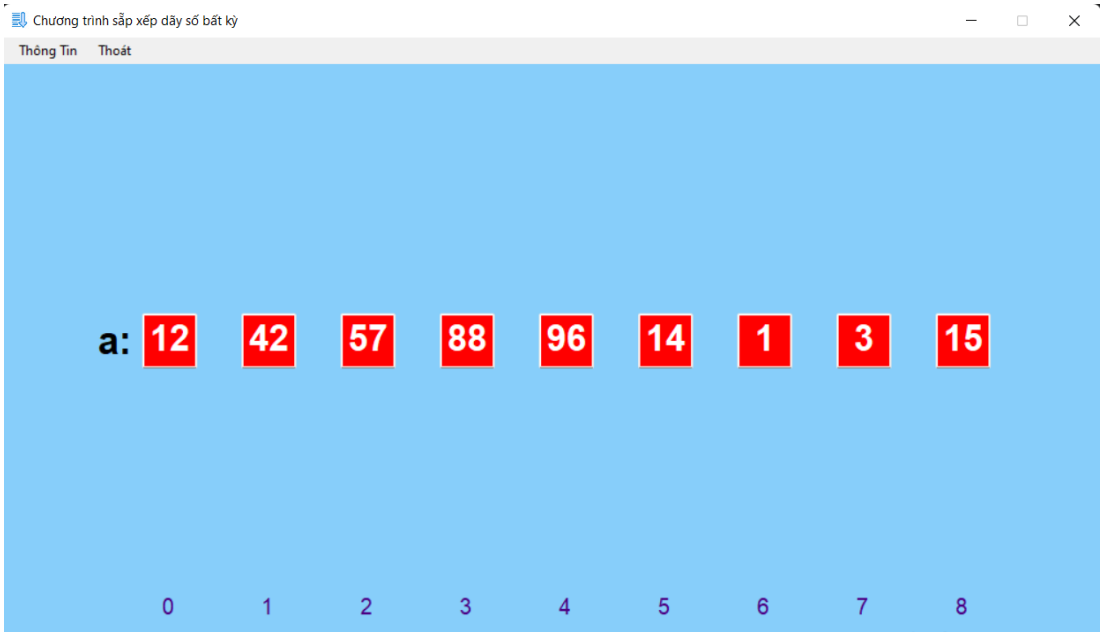
Như đã nói trên ta có 2 cách để tạo mảng dữ liệu đó là lấy dữ liệu từ bên ngoài hoặc tạo trực tiếp trên giao diện.

Với cách lấy dữ liệu từ bên ngoài ta tiếp tục click chuột vào nút mở file sẽ hiện lên file lưu dữ liệu như hình (Hình 3.2).



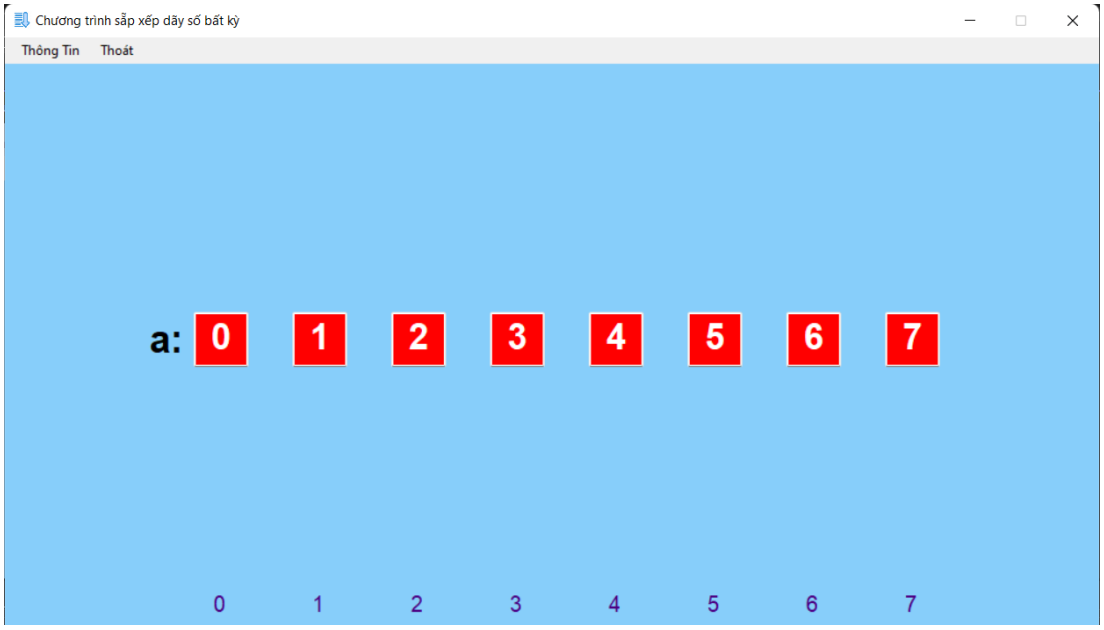
Hình 3.2 Giao diện khi mở file

Sau khi đã xác định dữ liệu mình muốn sử dụng thì tiếp tục click chuột vào nút đọc file để đọc dữ liệu lên giao diện ta sẽ được kết quả như hình (Hình 3.3).



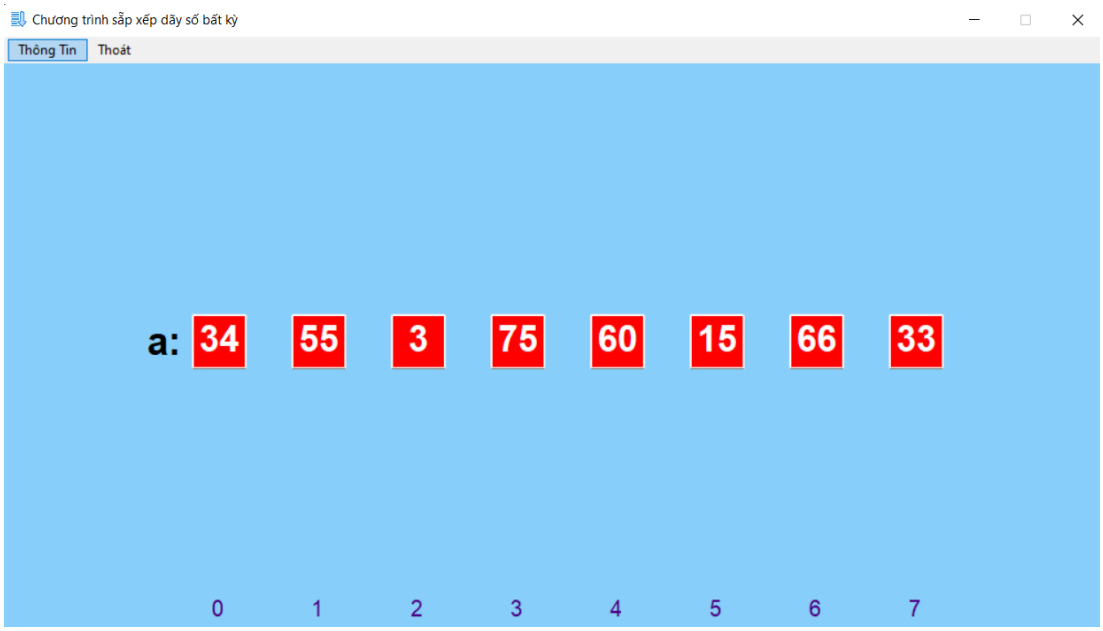
Hình 3.3 Giao diện sau khi đọc file

Với cách tạo dữ liệu trực tiếp trên giao diện thì đầu tiên ta điền số phần tử của mảng mà ta mong muốn vào sau đó click vào nút tạo mảng sẽ được mảng như hình (Hình 3.4).



Hình 3.4 Giao diện sau khi tạo mảng

Tiếp đến ta click vào nút ngẫu nhiên để lấy các số ngẫu nhiên chèn vào mảng mà ta đã tạo thì sẽ được kết quả như hình (Hình 3.5).

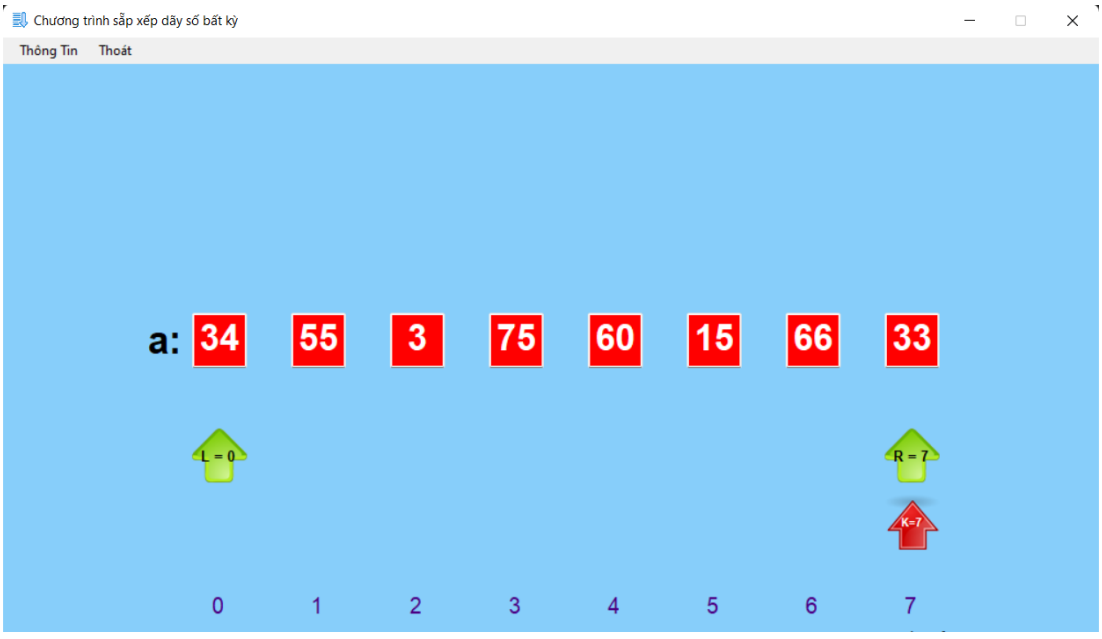


Hình 3.5 Giao diện sau khi trộn số ngẫu nhiên

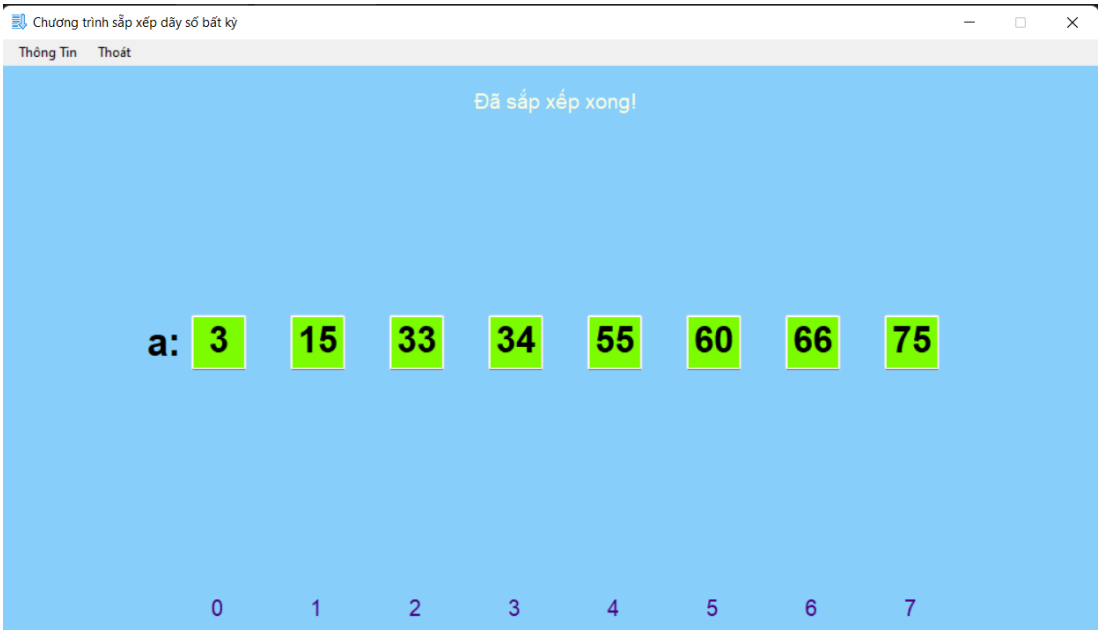
Sau khi đã tạo xong mảng dữ liệu ta tiếp tục chọn thuật toán và chiều sắp xếp mà ta muốn sử dụng để sắp xếp mảng vừa mới tạo.

3.1 Shaker Sort

Đầu tiên với shaker sort tăng dần sau khi bắt đầu sắp xếp thì các mũi tên sẽ bắt đầu từ 2 bên của mảng (Hình 3.1.1) và sau khi sắp xếp xong mảng sẽ đổi màu từ đỏ sang xanh như hình (Hình 3.1.2).

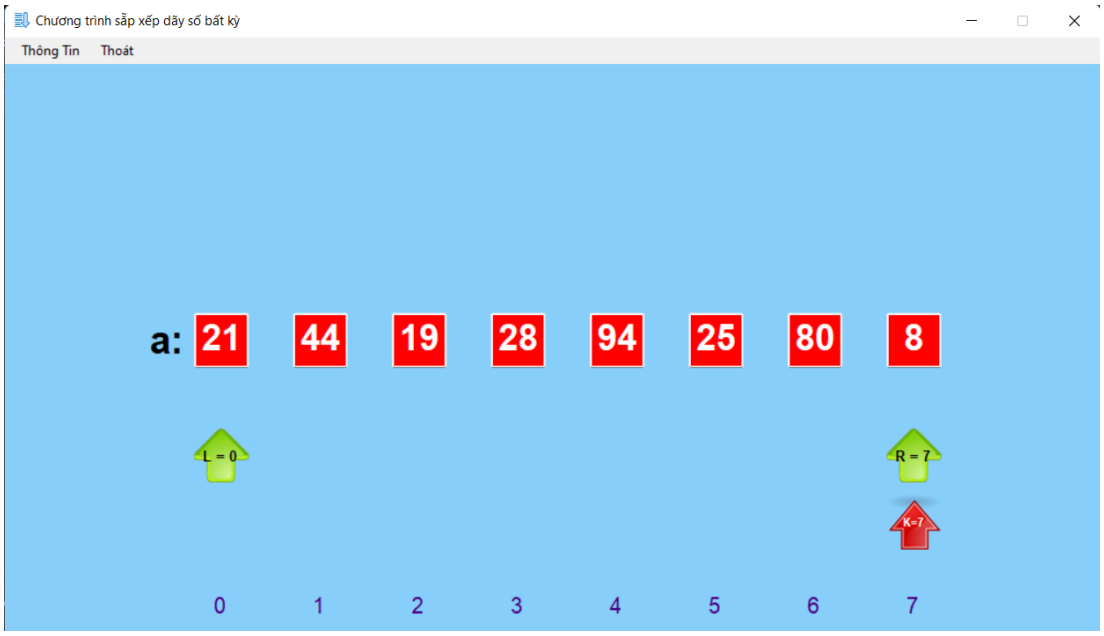


Hình 3.1.1 Giao diện bắt đầu sắp xếp của Shaker Sort tăng dần

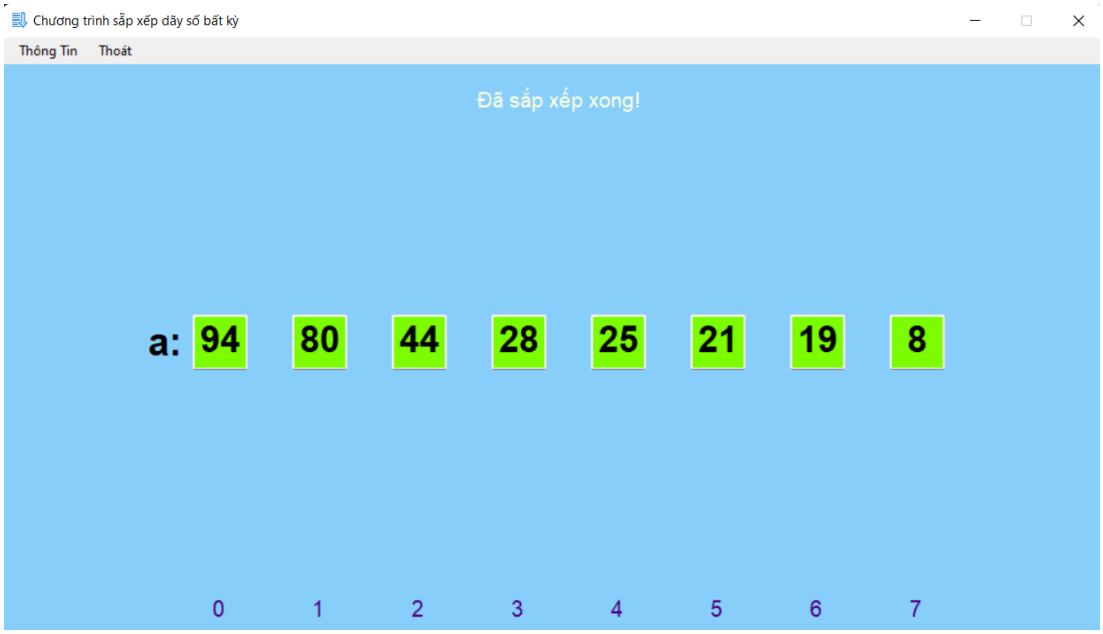


Hình 3.1.2 Giao diện sau khi sắp xếp của Shaker Sort tăng dần

Với shaker sort giảm dần cũng tương tự ta cũng sẽ được kết quả trước khi sắp xếp (Hình 3.1.3) và sau khi sắp xếp như hình (Hình 3.1.4).



Hình 3.1.3 Giao diện bắt đầu sắp xếp của Shaker Sort giảm dần

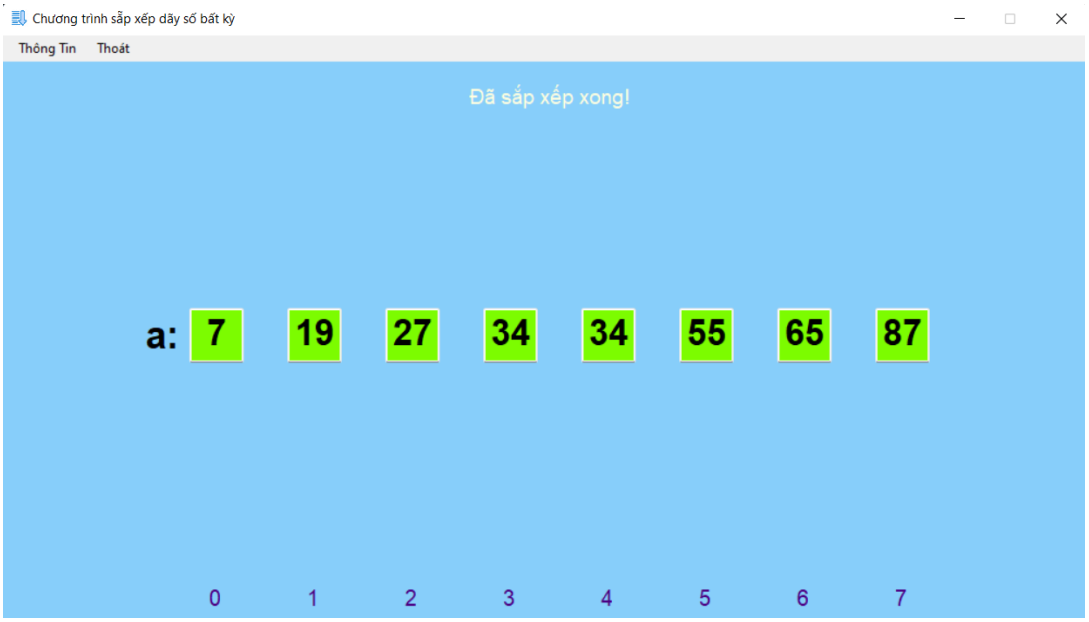


Hình 3.1.4 Giao diện sau khi sắp xếp của Shaker Sort giảm dần

3.2 Shell Sort

Tiếp theo với shell sort tăng dần sau khi bắt đầu sắp xếp thì mũi tên sẽ bắt đầu từ bên trong của mảng con cùng với thông số của mảng con (Hình 3.2.1) và sau khi sắp xếp xong mảng cũng sẽ đổi màu từ đỏ sang xanh như hình (Hình 3.2.2).

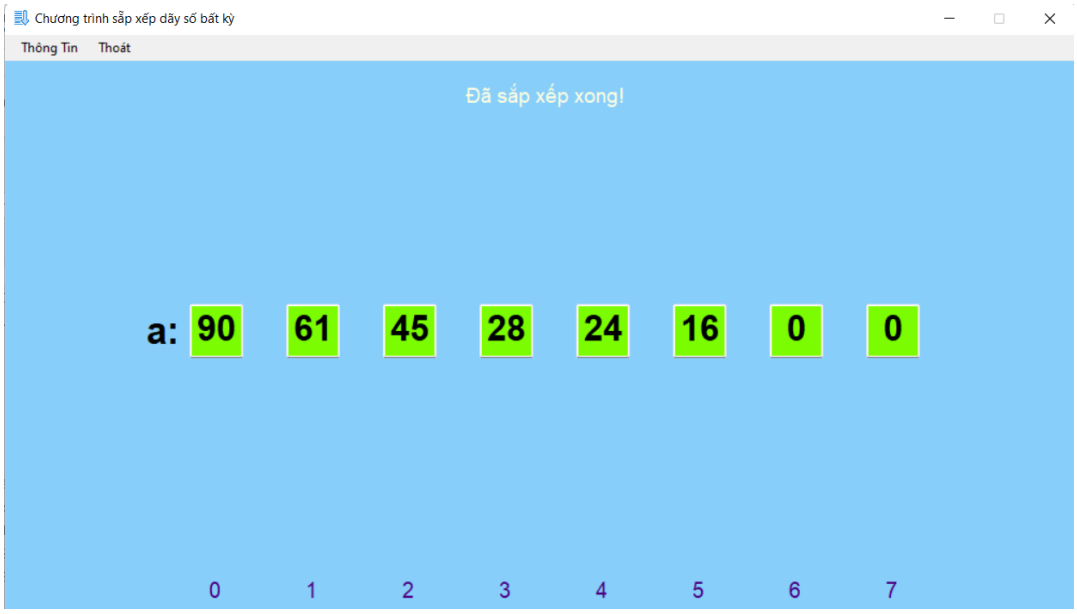
Hình 3.2.1 Giao diện bắt đầu sắp xếp của Shell Sort tăng dần



Hình 3.2.2 Giao diện sau khi sắp xếp của Shell Sort tăng dần

Với shell sort giảm dần cũng tương tự ta cũng sẽ được kết quả trước khi sắp xếp (Hình 3.2.3) và sau khi sắp xếp như hình (Hình 3.2.4).

Hình 3.2.3 Giao diện bắt đầu sắp xếp của Shell Sort giảm dần



Hình 3.2.4 Giao diện sau khi sắp xếp của Shell Sort giảm dần

Trong quá trình sắp xếp ta có thể tùy chỉnh thông qua phần điều khiển như tạm dừng , thay đổi tốc độ sắp xếp. Tiếp đó nếu muốn xóa mảng thì ta chỉ cần click vào nút đọc file, tạo mảng hoặc ngẫu nhiên để tạo mảng dữ liệu mới sau đó chọn thuật toán và chiều sắp xếp mà mình mong muốn (Hình 3.1).

Cuối cùng khi muốn thoát chương trình chỉ cần click nút thoát trên thanh công cụ hoặc click nút X trên giao diện.

CHƯƠNG 4. KẾT LUẬN

4.1 Hướng phát triển

Ở phần giao diện được mô phỏng vẫn còn rất thiếu tính năng so với thực tế. Vì thế trong thời gian tới nếu có cơ hội em sẽ thiết kế thêm các tính năng để giao diện hoàn thiện và thu hút hơn.

- Thiết kế thêm chức năng hiện mảng trước và sau khi sắp xếp.
- Thiết kế thêm các chức năng trong giao diện như sắp xếp chạy từng code bắt đầu từ giá trị chẵn(lẻ).
- Thêm các lựa chọn trong giao diện điều khiển như nút hủy quá trình sắp xếp và hiện thời gian thực hiện.

4.2 Kết luận

Xây dựng giao diện mô phỏng thuật toán với môi trường Visual Studio ngôn ngữ C# và giao diện winform đối với bản thân em có nhiều ưu điểm hơn so với Dev C và thư viện graphics. Ngôn ngữ C# được học ở môn Lập trình hướng đối tượng mà ở đó em đã bước đầu tiếp xúc với việc làm phần mềm ở giao diện Winform thông qua các bài thực hành nhỏ. Ngôn ngữ C# cũng dễ dàng để sử dụng do nó có ít từ khóa và mang đầy đủ các điểm mạnh từ những phần mềm nó kế thừa như C/C++. Thiết kế giao diện cũng đơn giản hơn khi chỉ cần kéo thả và cài đặt thuộc tính .

Qua việc mô phỏng các thuật toán ở đề tài thực tập lần này em đã từng bước tiếp cận được cách thức cũng như phương pháp hoạt động của các thuật toán sắp xếp mà em đã được học. Trong quá trình mô phỏng em nhận ra còn nhiều vấn đề phát sinh, nhiều khó khăn hơn mình suy nghĩ, hơn hết là tính kiên trì và sáng tạo vì vậy sản phẩm lần này còn nhiều thiếu sót, em sẽ cố gắng khắc phục và phát triển ở cơ hội tiếp theo.

TÀI LIỆU THAM KHẢO

[1] https://thuvienhuongdan.com/thuat-toan-sap-xep-bubble-sort-shaker-sort-929.html

[2] https://codelearn.io/learning/data-structure-and-algorithms/856660

[3] https://laptrinhtudau.com/bai-tap-thuat-toan-sap-xep-shaker-sort-trong-c/

[4] https://viettuts.vn/cau-truc-du-lieu-va-giai-thuat/giai-thuat-sap-xep-shell-sort

[5] https://cafedev.vn/thuat-toan-insertion-sort-gioi-thieu-chi-tiet-va-code-vi-du-tren-nhieu-ngon-ngu-lap-trinh/

[6] https://cafedev.vn/thuat-toan-bubble-sort-gioi-thieu-chi-tiet-va-code-vi-du-tren-nhieu-ngon-ngu-lap-trinh/