BỘ GIÁO DỤC ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

THỰC TẬP CƠ SỞ

MÔ PHỎNG THUẬT TOÁN SẮP XẾP

(Selection Sort, Insertion Sort)

Giảng viên hướng dẫn	**Bùi Thị Hồng Minh**
Sinh viên thực hiện	**Trần Lê Quang Minh**
Mã số sinh viên	**62131114**

Nha Trang, ngày 30 tháng 12 năm 2022

NHẬN XÉT

(Của giảng viên hướng dẫn)

NHẬN XÉT

(Của giảng viên phản biện)

Mục lục

DANH MỤC HÌNH

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1. GIỚI THIỆU CHUNG

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). Ứng dụng về sắp xếp có ở khắp mọi nơi:

- Một danh sách lớp với các học sinh, sinh viên được sắp xếp theo thứ tự bảng chữ cái.
- Một danh bạ điện thoại.
- Danh sách lượt tìm kiếm nhiều nhất trên Google.

Do đó khi xây dựng một hệ thống quản lý thông tin trên máy cần có các thuật toán sắp xếp, thuật toán tìm kiếm. Hiện nay đã có rất nhiều thuật toán tìm kiếm cũng như thuật toán sắp xếp với mức độ hiệu quả của từng thuật toán phụ thuộc vào tính chất và cấu trúc của dữ liệu mà nó tác động. Và thường họ sắp xếp các phần tử ở đây là một dãy số, hầu hết các bài toán khác nhau có thể có nhiều cách giải quyết khác nhau. Để biết rõ hơn về hiệu quả của thuật toán sắp xếp ta đi vào các thuật toán sắp xếp điển hình như là: Sắp xếp nổi bọt (Bubble sort), sắp xếp chèn (Insertion sort), sắp xếp nhanh (Quicksort), sắp xếp trộn (Merge sort).

Đề tài thực hiện là “mô phỏng giải thuật sắp xếp” của thuật toán sắp xếp chèn (Insertion sort) và sắp xếp chọn (Selection sort).

2. MÔI TRƯỜNG CÀI ĐẶT

Sử dụng ngôn ngữ lập trình C# (Winform) để mô phỏng các thuật toán sắp xếp (Insertion sort, Selection sort).

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 ĐỊNH NGHĨA SẮP XẾP

Sắp xếp là một quá trình biến đổi một danh sách các đối tượng (dữ liệu) thành một danh sách thỏa mãn một thứ tự xác định nào đó. Sắp xếp đóng vai trò quan trọng trong tìm kiếm dữ liệu. Ví dụ, trong thực tế là các ứng dụng quản lý danh bạ điện thoại thì có sắp xếp theo tên, theo số. Quản lý lượt tìm kiếm trên Google thì hiển thị những thông tin có lượt tìm kiếm nhiều nhất,… nếu danh sách đã được sắp xếp theo thứ tự tăng dần (hoặc giảm dần). Trong thiết kế thuật toán, ta cũng thường xuyên cần đến việc sắp xếp, nhiều thuật toán được thiết kế dựa trên ý tưởng xử lý các đối tượng theo một thứ tự xác định.

Các thuật toán sắp xếp có thể được chia làm 2 loại:

- Sắp xếp ổn định: Một thuật toán sắp xếp được gọi là sắp xếp ổn định nếu sau khi tiến hành sắp xếp vị trí tương đối giữa các phần tử bằng nhau không bị thay đổi.
- Sắp xếp so sánh: Một thuật toán sắp xếp được gọi là sắp xếp so sánh nếu trong quá trình thực hiện thuật toán ta tiến hành so sánh các khoá và đổi chỗ các phần tử cho nhau. Đa số các thuật toán sắp xếp đều là sắp xếp so sánh, riêng sắp xếp đếm phân phối (Counting sort) không phải là sắp xếp so sánh.

Hoặc cũng có thể là:

- Sắp xếp trong: đòi hỏi phải đưa toàn tập dữ liệu vào bộ nhớ trong của máy tính dưới dạng mảng. Do đó sắp xếp trong còn được gọi là sắp xếp mảng.
- Sắp xếp ngoài: khi các đối tượng cần sắp xếp quá lớn không thể cùng lúc đưa vào bộ nhớ trong nhưng có thể đọc từng phần dưới dạng file, ra sẽ sử dụng phương pháp sắp xếp ngoài, hay còn gọi là sắp xếp file.

2.2 THUẬT TOÁN

2.2.1 Thuật toán sắp xếp chọn (Selection sort)

2.2.1.1 Khái niệm

Thuật toán sắp xếp chọn sẽ sắp xếp một mảng hoặc danh sách bằng cách lặp đi lặp lại việc tìm phần tử nhỏ nhất (đang nói đến chiều tăng dần) từ một danh sách chưa được sắp xếp và đặt phần tử đó lên đầu tiên.

2.2.1.2Phân tích bài toán

Thuật toán này duy trì 2 mảng con trong một mảng đã cho:

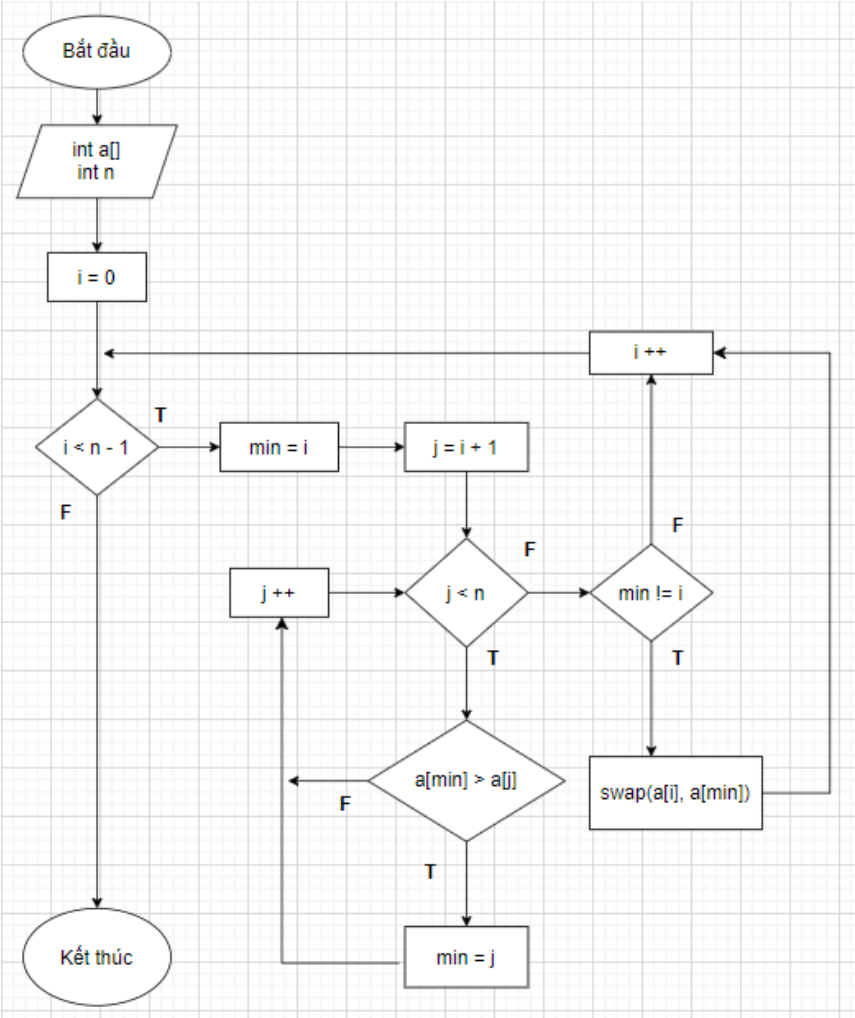
- Một mảng con đã được sắp xếp
- Một mảng con còn lại chưa được sắp xếp.

Và trong mỗi lần duyệt lại của thuật toán này, phần tử nhỏ nhất (đang nói đến chiều tăng dần) sẽ được lấy ra từ mảng chưa sắp xếp và sau đó cho vào mảng con đã được sắp xếp.

Rút kết lại, thuật toán này:

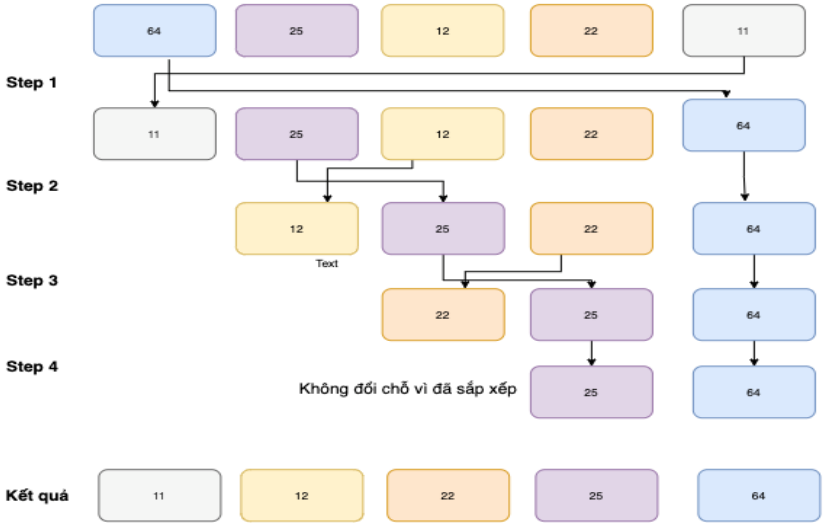
- Tìm phần tử nhỏ nhất đưa vào vị trí 1.
- Tìm phần tử nhỏ tiếp theo đưa vào vị trí 2.
- Tìm phần tử nhỏ tiếp theo đưa vào vị trí thứ 3.
- …

Dưới đây là lưu đồ giải thuật của thuật toán này:



Hình 2. 1 Lưu đồ giải thuật sắp xếp chọn

2.2.1.3Bài toán ví dụ



Hình 2. 2 Mô tả thuật toán sắp xếp chọn

2.2.2 Thuật toán sắp xếp chèn (Insertion sort)

2.2.2.1Khái niệm

Sắp xếp chèn là một giải thuật sắp xếp dựa trên so sánh. Ở đây, một danh sách con luôn luôn được duy trì dưới dạng đã qua sắp xếp. Sắp xếp chèn là chèn thêm một phần tử vào danh sách con đã sắp xếp trước đó. Phần tử được chèn vào vị trí thích hợp sao cho vẫn đảm bảo rằng danh sách con đó vẫn sắp theo thứ tự.

2.2.2.2Phân tích bài toán

Thuật toán sắp xếp chèn làm việc cũng giống như tên gọi - nó thực hiện việc quét một tập dữ liệu, với mỗi phân tử, thủ tục kiểm tra và chèn phần tử đó vào vị trí thích hợp trong danh sách đích (chứa các phần tử đứng trước nó đã được sắp xếp) được tiến hành.

Thuật toán được tiến hành bằng cách dịch chuyển phân tử hiện tại đang xét tuần tự qua những phân tử ở vùng dữ liệu phía trước đã được sắp xếp, phép hoán vị nó với phần tử liền kề được thực hiện một cách lặp lại cho tới khi tiến tới được vị trí thích hợp.

Sắp xếp chèn là một thuật toán sắp xếp bắt chước cách sắp xếp quân bài của những người chơi bài. Muốn sắp một bộ bài theo trật tự người chơi bài rút lần lượt từ quân thứ 2, so với các quân đứng trước nó để chèn vào vị trí thích hợp.

Thuật toán này duy trì 2 mảng con trong một mảng đã cho:

- Một mảng con đã được sắp xếp.
- Một mảng con còn lại chưa được sắp xếp.

Và trong mỗi lần duyệt lại của thuật toán này, phần tử nhỏ nhất (đang nói đến chiều tăng dần) sẽ được duyệt bởi mảng con và hoán vị cho đến khi về vị trí đầu danh sách.

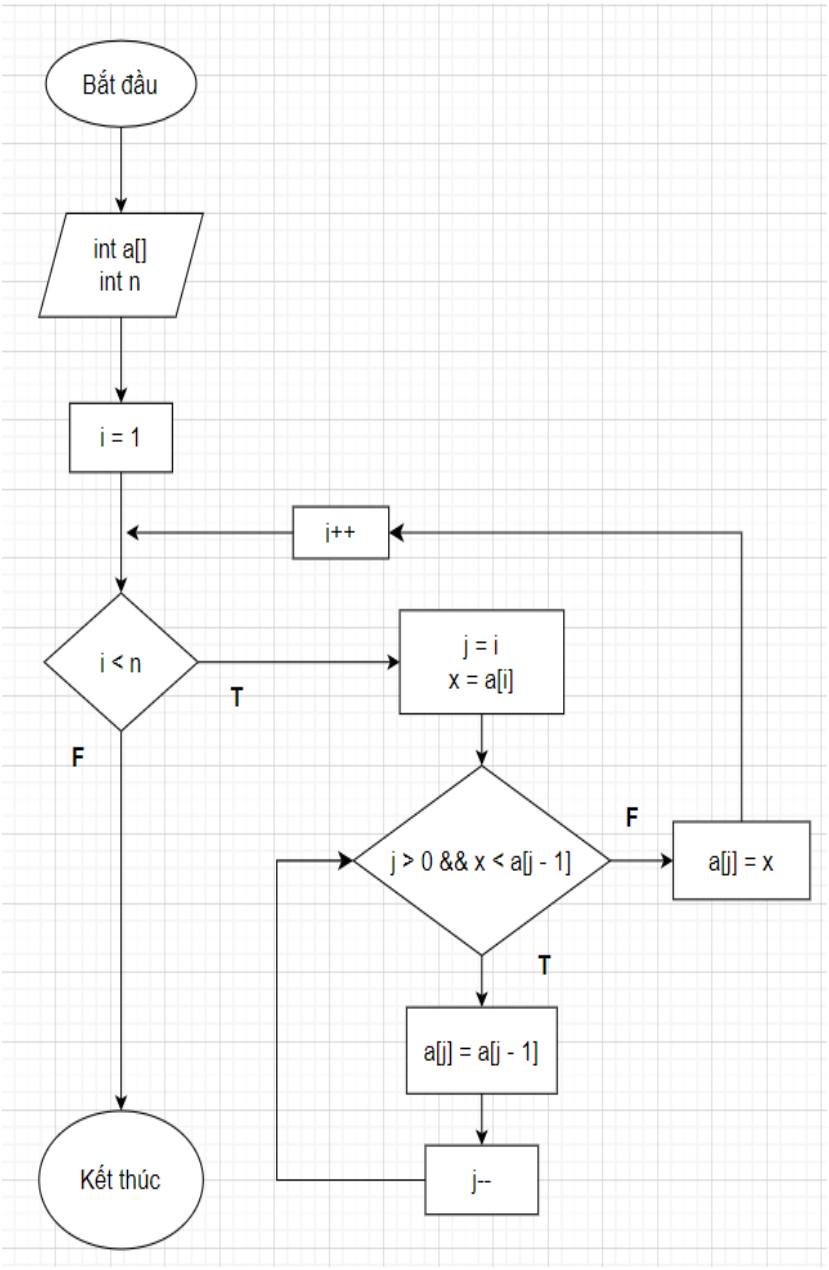
Rút kết lại, thuật toán này:

- Duyệt mảng từ phần tử thứ 2 (phần tử nhỏ nhất).
- So sánh giá trị và hoán đổi vị trí với các phần tử trước đó.
- Đưa về vị trí đầu danh sách (phần tử nhỏ nhất).
- Lặp lại với phần tử tiếp theo.

Xét bài toán: Sắp xếp một danh sách chứa các phần tử $k_1, k_2, k_3, \ldots, k_n$ theo thứ tự tăng dần. Các phần tử đưa vào danh sách được đặt đúng vị trí theo trật tự giá trị tăng dần.

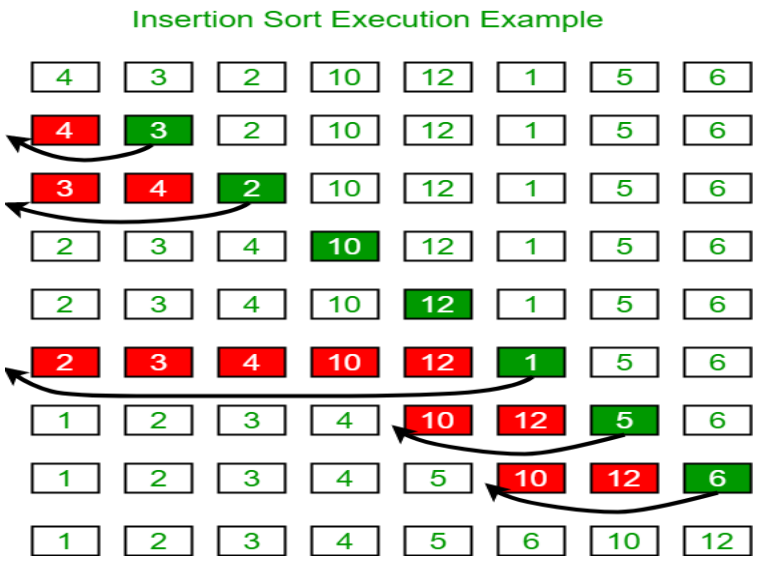
Chú ý là các $k_1, k_2, k_3, \ldots, k_i$ được đưa vào cho đến thời điểm thứ i $\Longrightarrow$ Ta tiến hành khởi tạo mảng với dãy con đã sắp xếp có k = 1 phần tử(phần tử đầu tiên, phần tử có chỉ số 0); Duyệt từng phần tử từ phần tử thứ 2, tại mỗi lần duyệt phần tử ở chỉ số i thì đặt phần tử đó vào một vị trí nào đó trong đoạn từ [0…i] sao cho dãy số từ [0…i] vẫn đảm bảo tính chất dãy số tăng dần. Sau mỗi lần duyệt, số phần tử đã được sắp xếp k trong mảng tăng thêm 1 phần tử; Lặp cho tới khi duyệt hết tất cả các phần tử của mảng.

Dưới đây là lưu đồ giải thuật của thuật toán này:



Hình 2. 3 Lưu đồ giải thuật sắp xếp chèn

2.2.2.3Bài toán ví dụ



Hình 2. 4 Mô tả thuật toán sắp xếp chèn

CHƯƠNG 3: CÀI ĐẶT CHƯƠNG TRÌNH

3.1 CÀI ĐẶT THUẬT TOÁN

3.1.1 Thuật toán sắp xếp chọn (Selection Sort)

3.1.2 Ý tưởng thuật toán

Sắp xếp chọn là một thuật toán sắp xếp đơn giản, dựa trên việc so sánh tại chỗ. Chọn phần tử nhỏ nhất trong n phần tử ban đầu, đưa phần tử này về vị trí đúng là đầu tiên của dãy hiện hành. Sau đó không quan tâm đến nó nữa, xem dãy hiện hành chỉ còn n-1 phần tử của dãy ban đầu, bắt đầu từ vị trí thứ 2.

3.1.3 Cài đặt thuật toán

```
class Selection_Sort
    {
        private int[] data;
        private static Random ngau_nhien = new Random();

        //tao mot mang gom 10 so tu nhien
        public Selection_Sort(int size)
        {
            data = new int[size];
            for (int i=0; i < size; i++)
            {
                data[i] = ngau_nhien.Next(20, 90);
            }
        }
        public void S_Sort()
        {
            //sap xep tang dan
            Console.WriteLine("Sap xep cac phan tu mang theo tung buoc");
            show_arr();
            int min;
            for(int i=0; i < data.Length-1; i++)
            {
                min = i;

                for(int index = i + 1; index < data.Length; index++)
                {
                    if (data[index] < data[min])
                        min = index;
                }
                swap(i, min);
                show_arr();
            }
        }
        public void swap(int a,int b)
        {
            int tmp = data[a];
            data[a] = data[b];
            data[b] = tmp;
        }
    }
```

Muốn sắp xếp thì trước tiên ta phải có mảng phần tử để sắp xếp nên đầu tiên ta phải tạo mảng phần tử.

Đầu tiên tạo mảng có tên data, dùng class Random để tạo giá trị ngẫu nhiên liên tiếp:

```
private int[] data;
private static Random ngau_nhien = new Random();
```

Tạo một hàm để tạo một mảng các số ngẫu nhiên có giá trị từ 20 đến 90 và một hàm hoán đổi vị trí cho nhau giữa các phần tử.

```
public Selection_Sort(int size)
        {
            data = new int[size];
            for (int i=0; i < size; i++)
            {
                data[i] = ngau_nhien.Next(20, 90);
            }
        }
public void swap(int a,int b)
        {
            int tmp = data[a];
            data[a] = data[b];
            data[b] = tmp;
        }
```

Khi đã có hàm thì ta có thể gọi nó được nhiều lần mà không cần phải viết lại.

```
public void S_Sort()
        {
            //sap xep tang dan
            Console.WriteLine("Sap xep cac phan tu mang theo tung buoc");
            show_arr();
            int min;
            for(int i=0; i < data.Length-1; i++)
            {
                min = i;

                for(int index = i + 1; index < data.Length; index++)
                {
                    if (data[index] < data[min])
                        min = index;
                }
                swap(i, min);
                show_arr();
            }
        }
```

Ta bắt đầu vòng lặp với i là vị trí của phần tử đầu tiên chạy đến vị trí kề cuối của mảng (data.Length -1) và đặt nó là phần tử nhỏ nhất (min), sau đó ta bắt đầu vòng lặp tiếp theo với j là vị trí của phần tử tiếp theo sau i chạy đến vị trí cuối cùng trong mảng (data.Length), trong vòng lặp j này ta bắt đầu duyệt từng phần tử với lệnh if, nếu giá trị của phần tử j (data[j]) bé hơn giá trị của phần tử nhỏ nhất (data[min]) thì ta thoát khỏi vòng lặp và thực hiện lệnh hoán đổi vị trí của 2 phần tử này và tiếp tục vòng lặp cho đến hết.

3.2 Thuật toán sắp xếp chèn (Insertion Sort)

3.2.1 Ý tưởng thuật toán

Sắp xếp chèn là thuật toán một thuật toán sắp xếp bắt chước cách sắp xếp quân bài của những người chơi bài. Muốn sắp một bộ bài theo trật tự người chơi rút lần lượt từ quân bài thứ hai so với quân bài bài đứng trước nó để chèn vào vị trí thích hợp.

3.2.2 Cài đặt thuật toán

```
class Insertion_Sort
    {
        private int[] data;
        private static Random ngau_nhien = new Random();

        public Insertion_Sort(int size)
        {
            data = new int[size];
            for (int i=0; i < size; i++)
            {
                data[i] = ngau_nhien.Next(20, 90);
            }
        }
        public void I_Sort()
        {
            Console.WriteLine("Sap xep cac phan tu theo tung buoc");
            show_arr();
            //sap xep tang dan
            for(int i=1; i < data.Length; i++)
            {
                int key = data[i];
                int j = i-1;

                while(j >= 0 && data[j]>key)
                {
                    data[j + 1] = data[j];
                    j = j - 1;
                }
                data[j + 1] = key;
            }
        }

    }
```

Đầu tiên tạo mảng có tên data, dùng class Random để tạo giá trị ngẫu nhiên liên tiếp:

```
private int[] data;
private static Random ngau_nhien = new Random();
```

Tạo một hàm để tạo một mảng các số ngẫu nhiên có giá trị từ 20 đến 90

```
public Insertion_Sort(int size)
        {
            data = new int[size];
            for (int i=0; i < size; i++)
            {
                data[i] = ngau_nhien.Next(20, 90);
            }
        }
```

Khi đã có hàm thì ta có thể gọi nó được nhiều lần mà không cần phải viết lại.

Ta bắt đầu thuật toán với vòng lặp i xuất phát từ đầu mảng chạy đến hết, đặt key là giá trị của phần tử trong mảng, tiếp đến ta dùng vòng lặp while với 2 điều kiện j phải dương và giá trị của j phải lớn hơn giá trị của key, nếu đúng điều kiện thì giá trị của j sẽ bằng giá trị tiếp theo (j+1) và vị trí của j sẽ trừ đi 1, sau đó thoát vòng lặp while và đặt giá trị của j+1 thành key. Có nghĩa là, ở đây key sẽ là giá trị nhỏ nhất, và khi key được so sánh với giá trị khác, nếu key nhỏ hơn thì key sẽ được di chuyển về vị trí đầu tiên của dãy số, sau đó đặt lại giá trị key mới.

3.3 XÂY DỰNG GIAO DIỆN VỚI WINFORM C#

3.3.1 Xây dựng giao diện mô phỏng thuật toán sắp xếp

Hình 3. 1 Giao diện mô phỏng thuật toán sắp xếp

Đầu tiên để có thể xây dựng một giao diện mô phỏng giải thuật sắp xếp thì ta phải xây dựng code:

```
TextBox[] Node;
        int[] a;
        int Spt = 0; //số phần tử
        Label[] Chi_so;
        int Toc_do;
        int i, Kthuoc_Code = 10;
        Boolean Da_tao_mang, KT_tam_dung = false; //Biến kiểm tra đã tạo mảng
và kiểm tra tạm dừng
        //Biến kiểm tra sắp xếp 1 lần hay từng bước
        Boolean Sap_xep_tung_buoc;

        int Khoang_cach;//khoảng cách 2 node liên tiếp
        int Canh_le;//canh lề node
        int Kich_thuoc; // kích thước node
        int Co_chu; //kích thước chữ
```

Đây là 1 số biến toàn cục sẽ được dùng trong quá trình xây dựng giao diện.

Trước hết để thực hiện sắp xếp thì ta phải có 1 dãy số để sắp xếp, với đề bài thì ta có thể đọc dãy số từ tập tin bên ngoài, cụ thể là file text:

```
private void btn_Docfile_Click(object sender, EventArgs e)
        {
            //Gọi hàm xóa mảng
            Xoa_mang();
            //Đọc file
            StreamReader Re = File.OpenText("TEST.txt");
            string input = null;
            int i = 0;
            int kt = 0;
            while ((kt < 1) && ((input = Re.ReadLine()) != null))
            {
                Spt = Convert.ToInt32(input);
                kt++;
            }
            //Gọi hàm tạo mảng
            Tao_mang();
            while (((input = Re.ReadLine()) != null) && (i < Spt))
            {
                Node[i].BackColor = Color.OrangeRed; // đặt lại màu cho mảng ngẫu nhiên
                Node[i].ForeColor = Color.White;
                a[i] = Convert.ToInt32(input);
                Node[i].Text = a[i].ToString();
                i++;
            }
            Re.Close();
        }
```

Đây là hàm đọc file trong giao diện sắp xếp, đầu tiên ta phải kiểm tra giao diện chúng ta được sạch sẽ nên chạy hàm xóa mảng đầu tiên sau đó mới tiến hành đọc file, vòng lặp while thực hiện nhiệm vụ đọc các phần tử trong file text và tiến hành đổi kiểu dữ liệu cho các phần tử. Tiếp đến sau khi đã đọc được hết các phần tử trong file, ta tiến hành tạo

mảng từ file nhờ hàm tạo mảng, vòng lặp while lúc này sẽ kiểm tra và tạo các phần tử cũng như đặt màu cho các phần tử đó.

Dưới đây là hàm tạo mảng và xóa mảng:

```
public void Tao_mang()
        {
            if ((Spt < 2) || (Spt > 30))
            {
                lbl_A.Visible = false;
                MessageBox.Show("2 <= Số Phần Tử <= 30");

                this.txt_sophantu.Clear();
                Da_tao_mang = false;
                return;
            }
```

Ở đây ta sẽ giới hạn số phần tử có thể tạo được là 30, nếu nhập hơn số thì sẽ thông báo cho người dùng rằng số phần tử chỉ từ 2 đến 30

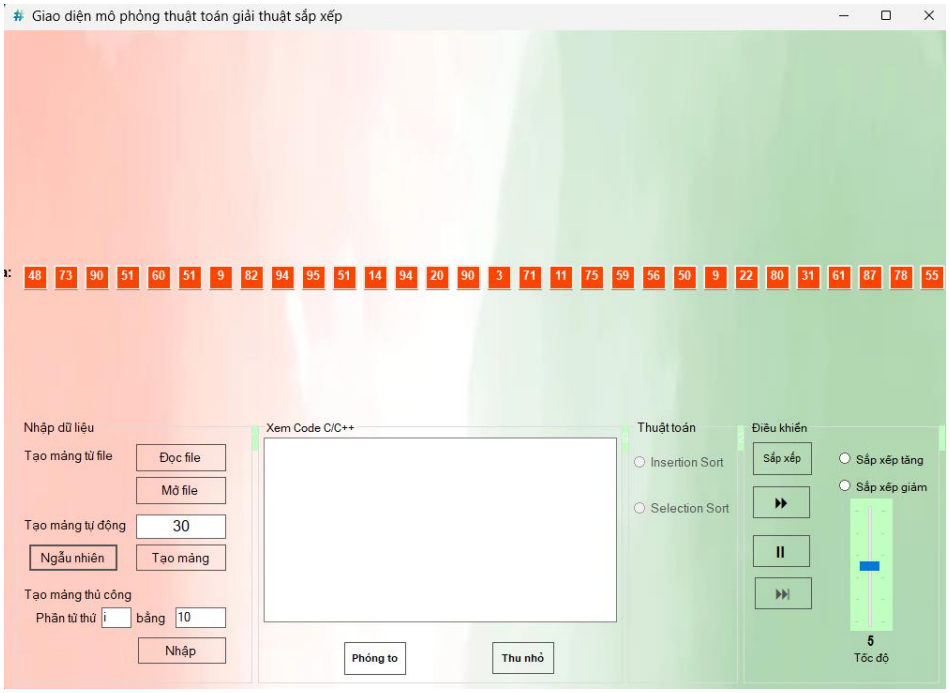
Với thiệt lập trên thì nếu số phần tử càng nhiều thì kích thước của các phần tử hiển thị sẽ nhỏ dần.

```
#region Thiết lập thuộc tính node ứng với số phần tử
            switch (Spt)
            {
                case 30:
                case 29:
                case 28:
                case 27:
                case 26:
                    Kich_thuoc = 27;
                    Co_chu = 10;
                    Khoang_cach = 6;
                    Canh_le = (1024 - Kich_thuoc * Spt - Khoang_cach * (Spt - 1)) / 2;
                    break;
                case 25:
                case 24:
                case 23:
                case 22:
                case 21:
                    Kich_thuoc = 30;
                    Co_chu = 13;
                    Khoang_cach = 10;
                    Canh_le = (1024 - Kich_thuoc * Spt - Khoang_cach * (Spt - 1)) / 2;
                    break;
                case 20:
                case 19:
                    Kich_thuoc = 40;
                    Co_chu = 18;
                    Khoang_cach = 5;
                    Canh_le = (1024 - Kich_thuoc * Spt - Khoang_cach * (Spt - 1)) / 2;
                    break;
```

```
                case 18:
                case 17:
                case 16:
                    Kich_thuoc = 40;
                    Co_chu = 18;
                    Khoang_cach = 10;
                    Canh_le = (1024 - Kich_thuoc * Spt - Khoang_cach * (Spt - 1)) / 2;
                    break;
                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;
            }
#endregion
```

Hình 3. 2 Giao diện khi có 10 phần tử

Hình 3. 3 Giao diện khi có 20 phần tử



Hình 3. 4 Giao diện khi có 30 phần tử

```
#region Tạo các mảng dữ liệu
            Chi_so = new Label[Spt];
            a = new int[Spt];
            Node = new TextBox[Spt];
```

```
//Dán nhãn mảng a
            lbl_A.Width = Kich_thuoc;
            lbl_A.Height = Kich_thuoc;
            lbl_A.Location = new Point(Canh_le - (Kich_thuoc), 250);
            lbl_A.Font = new System.Drawing.Font("Arial", Co_chu, FontStyle.Bold);
            lbl_A.Visible = true;
            #region Tạo node và chỉ số
            for (i = 0; i < Spt; i++)
            {
                //node
                a[i] = i;
                Node[i] = new TextBox();
                Node[i].Multiline = true;
                Node[i].Text = a[i].ToString();
                Node[i].TextAlign = HorizontalAlignment.Center;
                Node[i].Width = Kich_thuoc;
                Node[i].Height = Kich_thuoc;
                Node[i].Location = new Point(Canh_le + (Kich_thuoc + Khoang_cach)
* i, 250);
                Node[i].BackColor = Color.OrangeRed;
                Node[i].ForeColor = Color.White;
                Node[i].Font = new Font(this.Font, FontStyle.Bold);
                Node[i].Font = new System.Drawing.Font("Arial", Co_chu,
FontStyle.Bold);
                Node[i].ReadOnly = true;
                this.Controls.Add(Node[i]);
                //chỉ số
                Chi_so[i] = new Label();
                Chi_so[i].TextAlign = ContentAlignment.MiddleCenter;
                Chi_so[i].Text = i.ToString();
                Chi_so[i].Width = Kich_thuoc;
                Chi_so[i].Height = Kich_thuoc;
                Chi_so[i].ForeColor = Color.Azure;
                Chi_so[i].Location = new Point(Canh_le + (Kich_thuoc +
Khoang_cach) * i, 340 + 3 * Kich_thuoc);
                if (Spt <= 10)
                {
                    Chi_so[i].Font = new System.Drawing.Font("Arial", Co_chu - 10,
FontStyle.Regular);
                }
                else
                {
                    Chi_so[i].Font = new System.Drawing.Font("Arial", Co_chu,
FontStyle.Regular);
                }
                this.Controls.Add(Chi_so[i]);
                Da_tao_mang = true; //Xác nhận đã tạo mảng
                                    //Cho phép các nút điều khiển có tác dụng khi
đã tạo mảng
                btn_Sapxep.Enabled = true;
                btn_Ngaunhien.Enabled = true;
                btn_Nhap.Enabled = true;
            }
            #endregion
        }
```

Nội dung của đoạn code trên thể hiện cho chúng ta là tạo và thiết lập các thuộc tính node (textbox), cứ mỗi node sẽ thể hiện 1 phần tử cũng như màu và kích thước, cỡ chữ,… cứ mỗi lần chạy xong mảng thì hàm Da_tao_mang sẽ xác nhận cho chúng ta đã tạo xong mảng và cho phép sử dụng các nút bấm đã bị vô hiệu hóa trước đó.

```
public void Xoa_mang()
        {
            btn_Nhap.Enabled = false;
            btn_Ngaunhien.Enabled = false;
            btn_Sapxep.Enabled = false;
            if (Da_tao_mang == true)
            {
                for (i = 0; i < Spt; i++)
                {
                    this.Controls.Remove(Node[i]);
                    this.Controls.Remove(Chi_so[i]);
                }
            }
        }
```

Với hàm xóa mảng thì khi đã có sẵn mảng trước đó thì hàm này mới hoạt động, nguyên lý hoạt động của hàm là vòng lặp sẽ dùng con trỏ trỏ đến từng phần tử trong mảng và xóa phần tử đó đi.

Khi mô phỏng giao diện ta cần phải thực hiện các hoạt động của thuật toán, để các phần tử có thể di chuyển thì ta sẽ thiết lập những hàm di chuyển cho các phần tử:

```
public void Hoan_Vi_Node(Control t1, Control t2)
        {
            Application.DoEvents();

            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 - (Kich_thuoc + 5)) ||
(t2.Location.Y < p2.Y + (Kich_thuoc + 5)))
                    {
                        Application.DoEvents();
                        t1.Top -= 1;
                        t2.Top += 1;
                        Tre(Toc_do);

                    }
                    // 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;
                            Tre(Toc_do);
                        }
                    }

                     // 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;
                            Tre(Toc_do);
                        }

                    }
                    // t1 xuống, t2 lên
                    while ((t1.Location.Y < p2.Y) || (t2.Location.Y > p1.Y))
                    {
                        Application.DoEvents();
                        t1.Top += 1;
                        t2.Top -= 1;
                        Tre(Toc_do);
                    }
                    t1.Refresh();
                    t2.Refresh();
                }
            });

        }
```

Với hàm đầu tiên là hoán vị node, có nghĩa là hoán đổi vị trí của 2 node với nhau, ta sẽ dùng con trỏ để lưu vị trí ban đầu của node, sau đó so sánh giá trị của 2 node và tiến hành dịch chuyển vị trí của 2 node (lên, xuống, dịch sang trái, dịch sang phải).

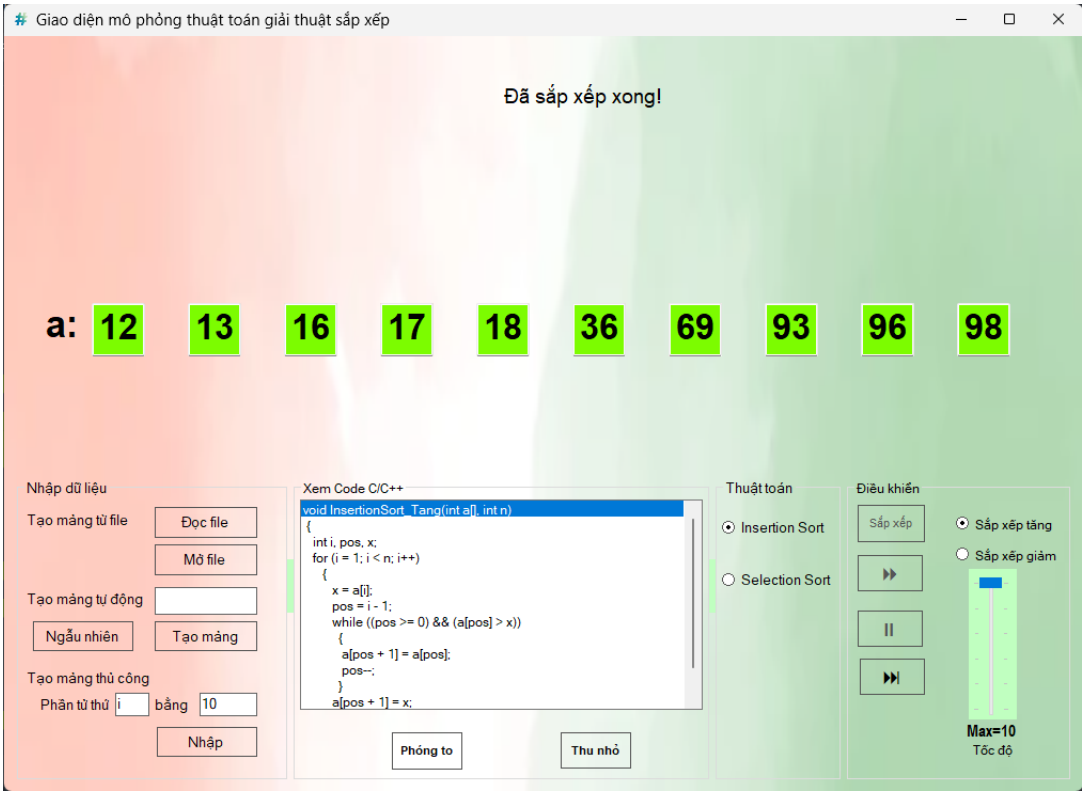
```
public void Node_qua_phai(Control t, int Step)
        {
            Application.DoEvents();

            this.Invoke((MethodInvoker)delegate
            {
                int Loop_Count = ((Kich_thuoc + Khoang_cach)) * Step; //Số lần
dịch chuyển
                {
                    while (Loop_Count > 0)
                    {
                        Application.DoEvents();
                        t.Left += 1;
                        Tre(Toc_do);
                        Loop_Count--;
                    }
                }
                t.Refresh();
            });
        }
```

```
// t dịch chuyển sang trai Step Node
        public void Node_qua_trai(Control t, int Step)
        {
            Application.DoEvents();
            this.Invoke((MethodInvoker)delegate
            {
                int Loop_Count = ((Kich_thuoc + Khoang_cach)) * Step; //Số lần dịch
chuyển
                while (Loop_Count > 0)
                {
                    Application.DoEvents();
                    t.Left -= 1;
                    Tre(Toc_do);
                    Loop_Count--;
                }
                t.Refresh();
            });
        }
        // t dịch chuyển lên với quãng đường S
        public void Node_di_len(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;
                    Tre(Toc_do);
                    loop_Count--;
                }
                t.Refresh();
            });
        }

// t dịch chuyển xuống với quãng đường S

public void Node_di_xuong(Control t, int S)
        {
            Application.DoEvents();
            this.Invoke((MethodInvoker)delegate
            {
                int loop_Count = S;
                // t lên
                while (loop_Count > 0)
                {
                    Application.DoEvents();
                    t.Top += 1;
                    Tre(Toc_do);
                    loop_Count--;
                }
                t.Refresh();
            });
        }
```

Các hàm tiếp theo thể hiện cho ta biết cách mà node sẽ dịch chuyển, với 2 hàm node qua trái và qua phải thì node sẽ di chuyển qua trái và qua phải tùy theo cách mà thuật toán hoạt động.

Giao diện sau khi hoàn thành bên dưới:

Hình 3. 5 Giao diện khi thực hiện sắp xếp



Hình 3. 6 Giao diện khi thực hiện xong

KẾT LUẬN

Qua báo cáo thực tập cơ sở lần này cá nhân em đã có những thay đổi mới trong việc cấu trúc lại kiến thức đã được học và đưa vào thực tế, thành qua đạt được là một chương trình mô phỏng khá trực quan, thông qua thực tập cơ sở lần này cho thấy quá trình học hỏi và tích lũy kinh nghiệm, kiến thức từ thầy/cô, anh/chị cũng như là bạn bè đã tạo ra một thành quả nhất định. Song, vẫn còn nhiều hạn chế trong việc xây dựng sản phẩm mà em chưa làm được, em sẽ cố gắng trau dồi, tích lũy thêm kiến thức, kinh nghiệm trong tương lai.

TÀI LIỆU THAM KHẢO

[1] Sắp xếp chọn – Wikipedia tiếng Việt

[2] Sắp xếp chèn – Wikipedia tiếng Việt

[3] Selection Sort Algorithm - GeeksforGeeks

[4] Insertion Sort - GeeksforGeeks

[5] C# programming with Visual Studio Code

[6] Khóa học lập trình C# nâng cao | How Kteam