TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO
NGÀNH CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

ỨNG DỤNG STACK VÀO TÍNH NĂNG UNDO TRONG GAME PUZZLE

Nhóm sinh viên thực hiện:

Trần Nguyên Tài **17110217**

Nguyễn Tiến Thành **17110225**

Giảng viên hướng dẫn: Ths. Trần Công Tú

TP. Hồ Chí Minh, tháng 12 năm 2018

Mục lục

Danh mục các bảng

Danh mục các hình

Lời mở đầu

1. Lý do chọn đề tài.

Cấu trúc dữ liệu và giải thuật được xem là môn học cơ sở, nền tảng của ngành Công nghệ thông tin. Môn học cung cấp cho chúng ta hiểu biết về các giải thuật tác động lên dữ liệu như thế nào, cũng như tổ chức dữ liệu để giải quyết các bài toán sao cho hiệu quả nhất, tối ưu hóa nhất.

Chúng em nghiên cứu và thực hiện đồ án **"Ứng dung Stack và tính năng undo trong game Puzzle"** này như một cách để cũng cố, mở rộng và ứng dụng kiến thức về **"Ngăn xếp – Stack"** nói riêng, cũng như những kiến thức đã được học từ trước đó nói chung. Thông qua quá trình thực hiện đồ án, chúng em đã nắm bắt được những kỹ thuật quan trọng của việc xây dựng cấu trúc dữ liệu và cách xây dựng thuộc toán sao cho tối ưu nhất.

2. Mục đích nghiên cứu.

Hiểu rõ vai trò, ứng dụng của ngăn xếp trong việc vận dụng để undo hành động di chuyển của người chơi..

Hiểu rõ bản chất của thuật toán, quy trình hoạt động của ngăn xếp trong việc thêm, xuất Stack.

3. Phương pháp nghiên cứu.

Nghiên cứu về cấu trúc dữ liệu trừu tượng của ngăn xếp.

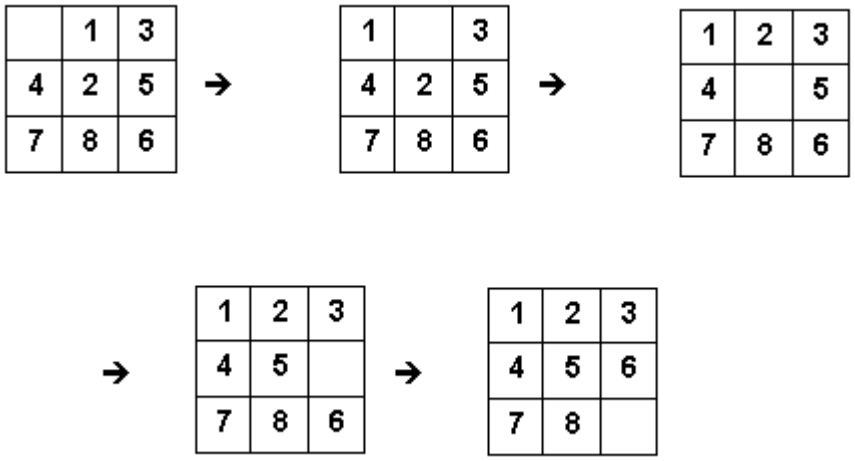
Nghiên cứu các cơ chế hoạt động của ngăn xếp cũng như việc vận dụng giải bài toán dựa trên các thao tác cơ bản của ngăn xếp.

Sử dụng ngôn ngữ lập trình C# để phục vụ cho việc lập trình.

1. Mô tả về đề tài.

1.1 *Giới thiệu game Puzzle.*[1]

Puzzle là một câu đố phát minh và phổ biến bởi Noyes Palmer Chapman trong những năm 1870. Bài toán gồm một bảng n × n với các ô số được đánh từ 1→n^2 -1và một ô trống. Ở trạng thái bắt đầu, các ô được sắp đặt ngẫu nhiên, và nhiệm vụ của người giải là tìm cách đưa chúng về đúng thứ tự (lớn dần từ trái qua phải và từ trên xuống, ô cuối dùng là ô trống) bằng phép trượt các khối theo chiều ngang hoặc chiều dọc vào hình vuông trống như minh họa dưới với n=3:



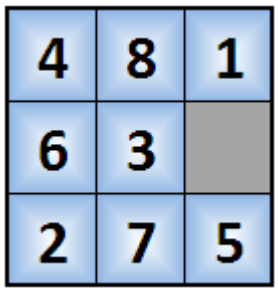
Hình 1:Các bước chuyển từ trạng thái ban đầu về trạng thái mục tiêu của puzzle 3x3

1.2 *Phân tích bài toán.* [2]

1.2.1 Puzzle 3 x 3.

Một trạng thái thì chỉ có tối đa 4 cách đi để chuyển sang trạng thái khác (trái, phải, lên, xuống). Người ta cũng nhận ra được rằng để có thể chuyển từ 1 trạng thái bất kì về trạng thái đích như trên thì trạng thái đầu đó phải theo một quy luật mà tôi trình bày sau đây.

Cho trạng thái đầu tiên như hình dưới, duyệt qua từng ô theo thứ tự từ trái qua và từ trên xuống, ở mỗi ô số duyệt đến, bạn hãy đếm xem có bao nhiêu ô số có giá trị bé hơn nó.



Hình 2:Trạng thái bắt đầu của puzzle 3x3.

Đầu tiên là ô số 4, ta thấy có 3 ô số {1;3;2} nằm phía sau và bé hơn nó nên n1 = 3.

– Tiếp đến là ô số 8 có 6 ô nhỏ hơn là {1;6;3;2;7;5}, vậy n2 = 6.

– Ô số 1 là bé nhất nên n3=0.

– Tương tự ô số 6, n4=3.

– Ô số 3, n5=1.

– Ô số 2, n6=0.

– Ô số 7, n7=1.

– Ô cuối luôn bằng 0 nên có thể bỏ qua, n8 = 0.

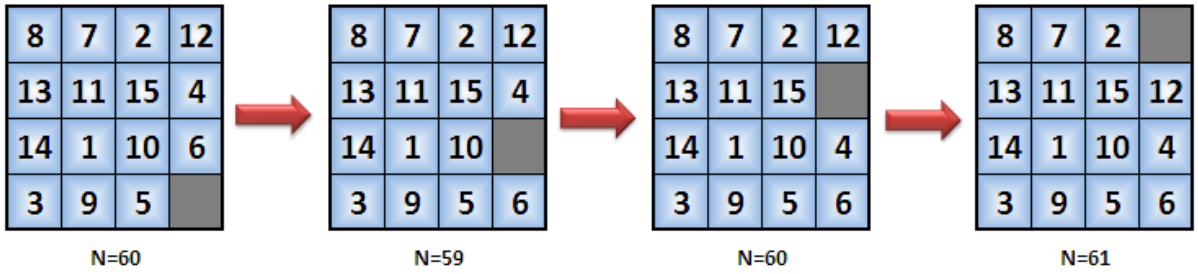
Tính tổng các số từ n1→ n8 ta có:

N = 3 + 6 + 0 + 3 + 1 + 0 + 1 + 0 = 14.

Với số N này ta chỉ cần biết 1 thông tin là nó có chia hết cho 2 hay không (lẻ hay chẵn). Nếu nó là số chẵn thì chắc chắn có thể chuyển về trạng thái đích từ trạng thái hiện tại này. Bởi vì khi di chuyển ô trống với hướng đi bất kì thì giá trị N mod 2 cũng không thay đổi. Tức là từ trạng thái đích bạn có thể xáo trộn bằng cách di chuyển ô trống nhiều lần thì giá trị N vẫn là số chẵn.

1.2.2 Puzzle 4 x 4.

Gặp phải vấn đề khi kiểm tra theo cách tương tự như với bài toán puzzle 3x3. Khi di chuyển ô trống trong bảng giữa các dòng và tính toán giá trị N thì bạn nhận ra rằng N sẽ thay đổi giá trị chẵn, lẽ.



Hình 3: Các trạng thái của puzzle 4x4 và N tương ứng.

Ví dụ như các bảng phía trên ta tính được N tương ứng khi di chuyển ô trống giữa các dòng lần lượt là 60, 59, 60, 61. Qua vài lần thử ta có thể nhận ra quy luật là N là lẻ khi ô trống nằm ở các dòng lẻ tính từ trên xuống (dòng đầu tiên là 1), và N là chẵn khi ô trống nằm ở các dòng chẵn tính từ trên xuống.

Sự khác biệt này so với puzzle 3x3 là do độ dài cạnh (n) khác nhau. Tức là với n lẻ thì giá trị N mod 2 sẽ ko thay đổi, với n chẵn thì nó sẽ thay đổi tương ứng với vị trí dòng của ô trống trên bảng.

1.2.3 Kết luận kết quả phân tích.

Ta suy ra phương pháp tổng quát để áp dụng cho mọi n để xác định được một puzzle $n \times n$ có thể chuyển về trạng thái đích với số lần di chuyển bất kì hay không

Với n lẻ:

Chỉ cần N mod 2 = 0.

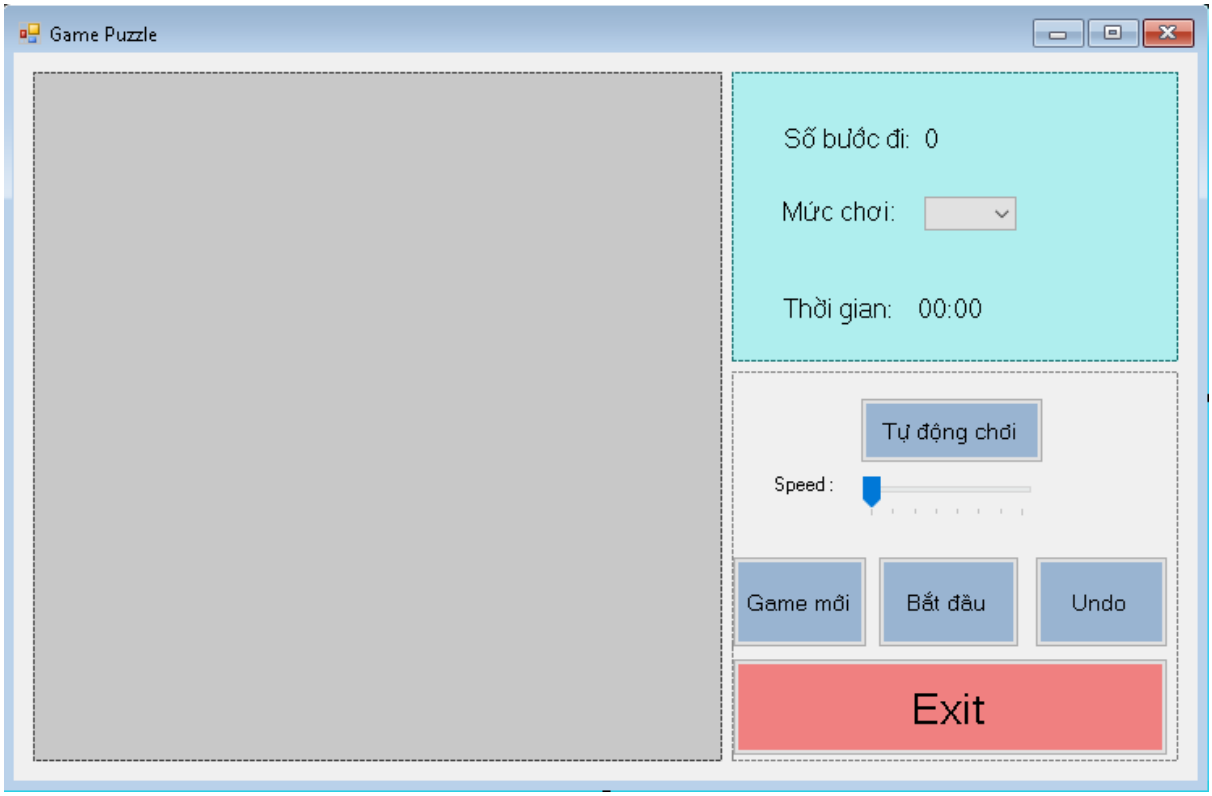
Với n chẵn:

N mod 2 = 0 và ô trống nằm trên dòng **chẵn** tính từ trên xuống (dòng đầu tiên là 1)

N mod 2 = 1 và ô trống nằm trên dòng **lẻ** tính từ trên xuống (dòng đầu tiên là 1).

2 Mô tả quá trình thực hiện.

2.1 *Thiết kế giao diện*



Hình 4: Giao diện chương trình

2.2 *Các chức năng trong game.*

- Game mới để tạo mới lại game.
- Bắt đầu để chơi game, tạm dừng, tiếp tục.
- Undo để quay lại bước đi trước đó.
- Tự động chơi để máy chơi tự động.
- Exit để thoát chương trình.

2.3 *Các class sử dụng trọng chương trình.*

TT	Tên Class	Mục đích
1	Form1.cs [Design]	Design giao diện game.
2	Form1.cs	Các hàm xử lý trong game.
3	ArrayStack.cs	Cài đặt Stack.
4	MyRandom.cs	Tạo số ngẫu nhiên dùng trong khởi tạo trạng thái bắt đầu.

Bảng 1:Các lớp được sử dụng trong chương trình.

2.4 *Các phương thức Form1.cs.*

TT	Phương thức	Mục đích
1	void initButton(int n)	Tạo ma trận Button[,] chứa các ô số.
2	public void swap(Button a, Button b)	Hoán đổi text giữa 2 Button.
3	public void go_left(int n)	Hoán đổi text giữa ô trống và ô bên phải nó.
4	public void go_right(int n)	Hoán đổi text giữa ô trống và ô bên trái nó.
5	public void go_top(int n)	Hoán đổi text giữa ô trống và ô bên dưới nó.
6	public void go_bottom(int n)	Hoán đổi text giữa ô trống và ô bên trên nó.
7	public void init(int n)	Khởi tạo trạng thái bắt đầu của game.
8	public bool ktWin()	Kiểm tra thắng.
9	private void bt_Click(object sender, EventArgs e)	Tạo event khi nhấn vào ma trận Button[,].
10	private void btnPause_Click(object sender, EventArgs e)	Tạo event khi nhấn vào Button Bắt đầu(tạm dựng, chơi tiếp).

11	private void btnnewGame_Click(object sender, EventArgs e)	Tạo event khi nhấn vào Button New game.
12	private void btnExit_Click(object sender, EventArgs e)	Tạo event khi nhấn vào Button Exit.
13	private void Undo(int x)	Undo hành động.
14	private void btnUndo_Click(object sender, EventArgs e)	Tạo event khi nhấn vào Button Undo.
15	private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)	Tạo event khi thay đổi combobox1.
16	private void timer_Tick(object sender, EventArgs e)	Tính thời gian chơi game.
17	private void trackBar1_Scroll(object sender, EventArgs e)	Tinh chỉnh timer1.Interval
18	private void btntudongChoi_Click(object sender, EventArgs e)	Tạo event khi nhấn vào Button Tự động chơi
19	private void timer1_Tick(object sender, EventArgs e)	Tự động chơi game

Bảng 2: Các phương thức trong Form.cs

2.5 *Cài đặt kiểu dữ liệu Stack.*

Ngăn xếp (Stack) là một dạng danh sách được cài đặt nhằm sử dụng cho các ứng dụng cần xử lý theo thứ tự đảo ngược. Trong cấu trúc dữ liệu ngăn xếp, tất cả các thao tác thêm, xóa một phần tử đều phải thực hiện ở một đầu danh sách, đầu này gọi là đỉnh (top) của ngăn xếp. Đặc điểm này làm cho ngăn xếp trở thành cấu trúc dữ liệu dạng LIFO. LIFO là viết tắt của Last-In-First-Out hay *"vào sau ra trước"*.

```
class ArrayStack
{
    private int[] data;     //mảng chứa các phần tử của stack
    private int max_count;      //số phần tử tối đa của stack
    private int top;            //chỉ số của phần tử ở đỉnh stack
                                //hàm khởi tạo
    public ArrayStack(int MaxCount)
    {
        max_count = MaxCount;
        data = new int[max_count];
        top = -1;
    }
    //
    public int spt()
    {
        return top;
    }
    //lấy phần tử ra khỏi stack
    public int Pop()
    {
        if (top == -1) throw new Exception("Stack rỗng");
        return data[top--];
    }
    //thêm phần tử vào stack
    public void Push(int Value)
    {
        if (top == max_count - 1) throw new Exception("Stack đầy");
        data[++top] = Value;
    }
    //kiểm tra stack có rỗng không
    public bool IsEmpty()
    {
        return (top == -1);
    }
```

```
    public void initArrayStack()
    {
        top = -1;
    }
}
```

2.6 *Cài đặt chương trình 8-Puzzle.*

Trạng thái các ô số của game Puzzle n x n sẽ được lưu trong mảng 2 chiều các Button[,].

```
void initButton(int n)
{
   int dem = 1;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            arr[i, j] = new Button();
            arr[i, j].SetBounds(panel1.Width / n * j, panel1.Height / n * i,
                                panel1.Width / n, panel1.Height / n);
            arr[i, j].Click += new EventHandler(this.bt_Click);
            arr[i, j].BackColor = Color.LightSteelBlue;
            arr[i, j].Cursor = Cursors.Hand;
            arr[i, j].ForeColor = Color.Black;
            arr[i, j].Text = dem++.ToString();
            arr[i,j].Font= new Font("Consolas", arr[i,j].Size.Height/3, FontStyle.Regular);
            panel1.Controls.Add(arr[i, j]);
        }
    }
    arr[n-1,n-1].Text = " ";
}
```

Khởi tạo trạng thái ban đầu ngẫu nhiên, các số từ 1→ n^2 -1 sẽ được xếp ngẫu nhiên.

Ý tưởng khởi tạo:

Như trong phần phân tích, có ½ trạng thái ban đầu không thể đưa về trạng thái đích vì vậy để chắc chắn trạng thái ban đầu luôn hợp lệ thì ta sẽ khởi tạo bằng cách hoán vị vị trí ô trống với 1 ô cạnh nó (hoán vị text của Button có text = “0” và 1 Button cạnh

nó trong mảng 2 chiều các Button[,]) thông qua các thao tác lên, xuống, trái, phải. Bằng cách này ta có thể chắc chắc trạng thái ban đầu của game luôn là hợp lệ.

```
public void swap(Button a, Button b)
{
    string temp = a.Text;
    a.Text = b.Text;
    b.Text = temp;
}
```

```
public void go_left(int n, int push)
{
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
        {
            if (arr[i, j].Text == " " && j + 1 < n)
            {
                swap(arr[i, j], arr[i, j + 1]);
                if (push == 1)
                    arrayStackinit.Push(4);
                return;
            }
        }
}
```

Tương tự với các hàm

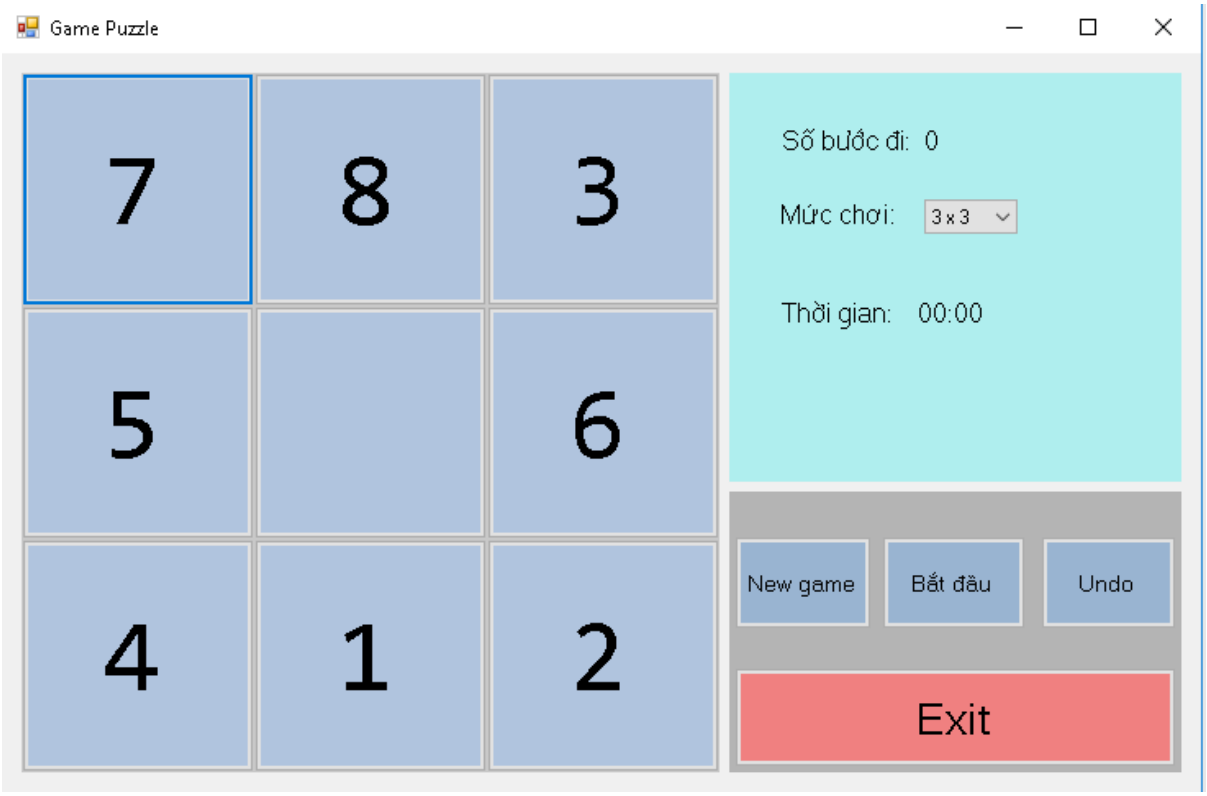
```
public void go_right(int n)
```

```
public void go_top(int n)
```

```
public void go_bottom(int n)
```

Khởi tạo trạng thái bắt đầu bằng cách cho Button[,] thực hiện 200 lần hành động hoán vị vị trí ô trống. Ứng với mỗi số ngẫu nhiên được tạo ra từ 1→4 sẽ là 1 hành động hoán vị (1 = go_top, 2 = go_right, 3 = go_bottom, 4 = go_left).

```
public void init(int n)
{
    arrayStackinit.initArrayStack();
    for (int i = 0; i < 200; i++)
    {
        int rd = MyRandom.Next(1, 5);
        switch (rd)
        {
            case 1: go_top(n, 1); break;
            case 2: go_right(n, 1);break;
            case 3: go_bottom(n, 1); break;
            case 4: go_left(n, 1); break;
        }
    }
}
```



Hình 5: Giao diện chương trình khi khởi tạo trạng thái bắt đầu của puzzle 3x3.

Hình 6: Giao diện Chương trình khi khởi tạo trạng thái bắt đầu của puzzle 4x4.

Tương tự với giao diện chương trình khi khởi tạo trạng thái bắt đầu của puzzle 5x5.

Tại một trạng thái thì chỉ có tối đa 4 cách đi để chuyển sang trạng thái khác (trái, phải, lên, xuống). Để di chuyển các ô trống ta thao tác trên chuột bằng cách click vào các button muốn di chuyển, ma trận Button [,] sẽ duyệt để tìm vị trí Button vừa click và tìm ô trống phía trên, dưới, trái, phải để thực hiện hành động di chuyển hợp lý, đồng thời trong thao tác di chuyển này ta cũng lưu các hành động vào stack để dùng cho tính năng undo. Giá trị được thêm vào Stack là các số ứng với go_top = 1, go_right = 2, go_bottom = 3, go_left = 4. Cuối các hành động di chuyển ta kiểm tra xem game đã thắng hay chưa.

Dưới đây là code để di chuyển khi click vào Button.

```
private void bt_Click(object sender, EventArgs e)
{
    if (TT_game == true)
    {
        Button b = sender as Button;
        for (int i = 0; i < comboBox1.SelectedIndex + 3; i++)
            for (int j = 0; j < comboBox1.SelectedIndex + 3; j++)
            {
                if (arr[i, j] == b && arr[i, j].Text != " " && i - 1 >= 0 && arr[i - 1, j].Text == " ")
                {
                    swap(b, arr[i - 1, j]);
                    arrayStack.Push(1);
                    lbsobuocdi.Text = (arrayStack.spt() + 1).ToString();
                }
                else if (arr[i, j] == b && arr[i, j].Text != " " && i + 1 < comboBox1.SelectedIndex + 3
                        && arr[i + 1, j].Text == " ")//bot
                {
                    swap(b, arr[i + 1, j]);
                    arrayStack.Push(3);
                    lbsobuocdi.Text = (arrayStack.spt() + 1).ToString();
                }
                else if (arr[i, j] == b && arr[i, j].Text != " " && j - 1 >= 0 && arr[i, j - 1].Text == " ")
                {
                    swap(b, arr[i, j - 1]);
                    arrayStack.Push(4);
                    lbsobuocdi.Text = (arrayStack.spt() + 1).ToString();
                }
                else if (arr[i, j] == b && arr[i, j].Text != " " && j + 1 < comboBox1.SelectedIndex + 3
                        && arr[i, j + 1].Text == " ")//right
                {
                    swap(b, arr[i, j + 1]);
                    arrayStack.Push(2);
                    lbsobuocdi.Text = (arrayStack.spt() + 1).ToString();
                }
                if(ktWin()==true)
                {
                    timer.Stop();
                    MessageBox.Show("Bạn đã chiến thắng với thời gian là: "+labelTime.Text,
                        "Chúc mừng", MessageBoxButtons.OK);
                    init(comboBox1.SelectedIndex + 3);
                }
            }
    }
```

Hàm ktWin() trả về true nó đang ở trạng thái win.

```
public bool ktWin()
{
    int dem = 1;
    for (int i = 0; i < comboBox1.SelectedIndex + 3; i++)
        for (int j = 0; j < comboBox1.SelectedIndex + 3; j++)
        {
            if (i != comboBox1.SelectedIndex + 3 - 1 || j != comboBox1.SelectedIndex + 3 - 1)
                if (arr[i, j].Text != dem.ToString())
                    return false;
            dem++;
        }
    return true;
}
```

.

Hàm undo di chuyển của người chơi.

```
private void btnUndo_Click(object sender, EventArgs e)
{
    if (arrayStack.IsEmpty() == false)
        Undo(arrayStack.Pop());
}

private void Undo(int x)
{
    if (x == 1)
    {
        go_bottom(comboBox1.SelectedIndex + 3, 0);
        lbsobuocdi.Text = (arrayStack.spt() + 1).ToString();
    }
    if (x == 2)
    {
        go_left(comboBox1.SelectedIndex + 3, 0);
        lbsobuocdi.Text = (arrayStack.spt() + 1).ToString();
    }
    if (x == 3)
    {
        go_top(comboBox1.SelectedIndex + 3, 0);
        lbsobuocdi.Text = (arrayStack.spt() + 1).ToString();
    }
    if (x == 4)
    {
        go_right(comboBox1.SelectedIndex + 3, 0);
        lbsobuocdi.Text = (arrayStack.spt() + 1).ToString();
    }
}
```

Hàm tính thời gian với Timer có interval = 1000.

```
private void timer_Tick(object sender, EventArgs e)
{
    labelTime.Text = (time / 60).ToString() + ":" + (time % 60).ToString()+"s";
    time++;
}
```

Chức năng tự động chơi của chương trình tương tự với chức năng undo, hàm sẽ undo các bước di chuyển của người dùng, sau đó undo các bước di chuyển trong phần khởi tạo để đưa về trạng thái win ban đầu trước khi khởi tạo.

```
private void btntudongChoi_Click(object sender, EventArgs e)
{
    if (TT_game == true)
    {
        timer1.Start();
        comboBox1.Enabled = false;
    }
}
```

```
private void timer1_Tick(object sender, EventArgs e)
{
    if (TT_game == true)
    {
        if (arrayStack.IsEmpty() == false)
            Undo(arrayStack.Pop());
        else if (arrayStackinit.IsEmpty() == false)
            Undo(arrayStackinit.Pop());
        if (ktWin() == true)
        {
            timer1.Stop();
            timer.Stop();
            MessageBox.Show("Bạn đã chiến thắng với thời gian là: " + labelTime.Text,
                "Chúc mừng", MessageBoxButtons.OK);
            btnnewGame_Click(sender, e);
        }
    }
}
```

3 Phân công công việc.

Công việc	Nguyên Tài	Tiến Thành
Tìm tài liệu, tìm hiểu hoạt động của game	X	X
Lập trình di chuyển các số		X
Tạo ma trận Button[,]	X	
Kiểm tra thắng, Lập trình undo bằng Stack.		X
Thiết kế giao diện	X	
Test, Kiểm tra	X	X
File Word	X	
File Powerpoint		X

Bảng 3: Bảng phân công kế hoạch.

4 Kết luận.

4.1 *Mức độ hoàn thành.*

Đồ án được hoàn thành 90% so với mục tiêu đặt ra.

4.2 *Ưu điểm và khuyết điểm.*

Ưu điểm: Giao diện đơn giản,dể sử dụng.

Chơi được các game puzzle 3x3, 4x4, 5x5. .

Nhược điểm: Chưa thể thao tác bằng bàn phím.

Chưa lưu được dữ liệu người chơi.

4.3 *Hướng phát triển.*

- Thêm tính năng tự động giải.
- Thêm các lựa chọn game với n>5.
- Thêm tính năng giới hạn thời gian chơi.
- Thêm điều khiển bằng bàn phím.

Tài liệu tham khảo

[1] http://coursera.cs.princeton.edu/algs4/assignments/8puzzle.html

[2] https://yinyangit.wordpress.com/2010/12/11/algorithm-tim-hi%E1%BB%83u-v%E1%BB%81-bai-toan-n-puzzle-updated/