TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

****

HỌC PHẦN: THỰC TẬP CƠ SỞ

GIẢNG VIÊN : NGUYỄN HẢI TRIỀU

ĐỀ TÀI

LẬP TRÌNH GAME CARO 2 NGƯỜI CHƠI SỬ DỤNG WINFORM C#

Sinh viên : Nguyễn Lê Thành Tâm

Mssv : 61134311 Lớp : 61CNTT-1

MỤC LỤC

I. TỔNG QUAN ĐỀ TÀI

Mục tiêu đề tài – Ý nghĩa đề tài

a. Mục tiêu đề tài

Nhằm tìm hiểu thêm về kiến thức lập trình hướng đối tượng và WinForm để xây dựng một ứng dụng game trên hệ điều hành Window cho phép 2 người chơi có thể kết nối với nhau và thi đấu với nhau thông qua việc truyền mô thức Client – Server.

b. Ý nghĩa của đề tài

- Rèn luyện kĩ năng lập trình, khả năng tư duy và khả năng làm việc
- Tiếp cận lập trình phần mềm theo hướng chuyên nghiệp
- Tìm hiểu được cách thức quản lý, tổ chức, xây dựng ứng dụng game đơn giản

Phương pháp tiếp cận

Thông qua việc được tiếp cận với ngôn ngữ lập trình hướng đối tượng C# cùng một số hiểu biết về WinForm, chúng em muốn ứng dụng những kiến thức đã học đó để xây dựng một ứng dụng " *game Caro WinForm đơn giản* "nhằm tạo củng cố được kiến thức, ôn tập các tính chất của lập trình hướng đối tượng như kế thừa, trừu tượng, một số điều khiển trong Winform, vv.

Giới thiệu về ngôn ngữ C#

- C# là một ngôn ngữ lập trình hiện đã được phát triển bởi Microsoft và được phê duyệt bởi European Computer Manufactures Association (ECMA)
- C# được phát triển bởi Anders Hejlsberg và nhóm của ông trong việc phát triển .NET Framework. Microsoft phát triển C# dựa trên C++ và Java nên sẽ khắc phục được những khuyết điểm của hai ngôn ngữ lập trình này
- Sử dụng C# có thể tạo ra rất nhiều kiểu ứng dụng, ở đây ta quan tâm đến 3 kiểu chính đó là : Console, Window form và ứng dụng Web.

✓ Ứng dụng Console : Giao tiếp với người dùng bằng bàn phím, không có giao diện đồ họa (GUI)

✓ Ứng dụng Windows Form : Giao tiếp với người dùng bằng bàn phím và chuột, có giao diện đồ họa và xử lý sự kiện

✓ ứng dụng Web : Kết hợp ASP .NET, C# đóng vai trò xử lý bên dưới, có giao diện đồ họa và xử lý sự kiện.

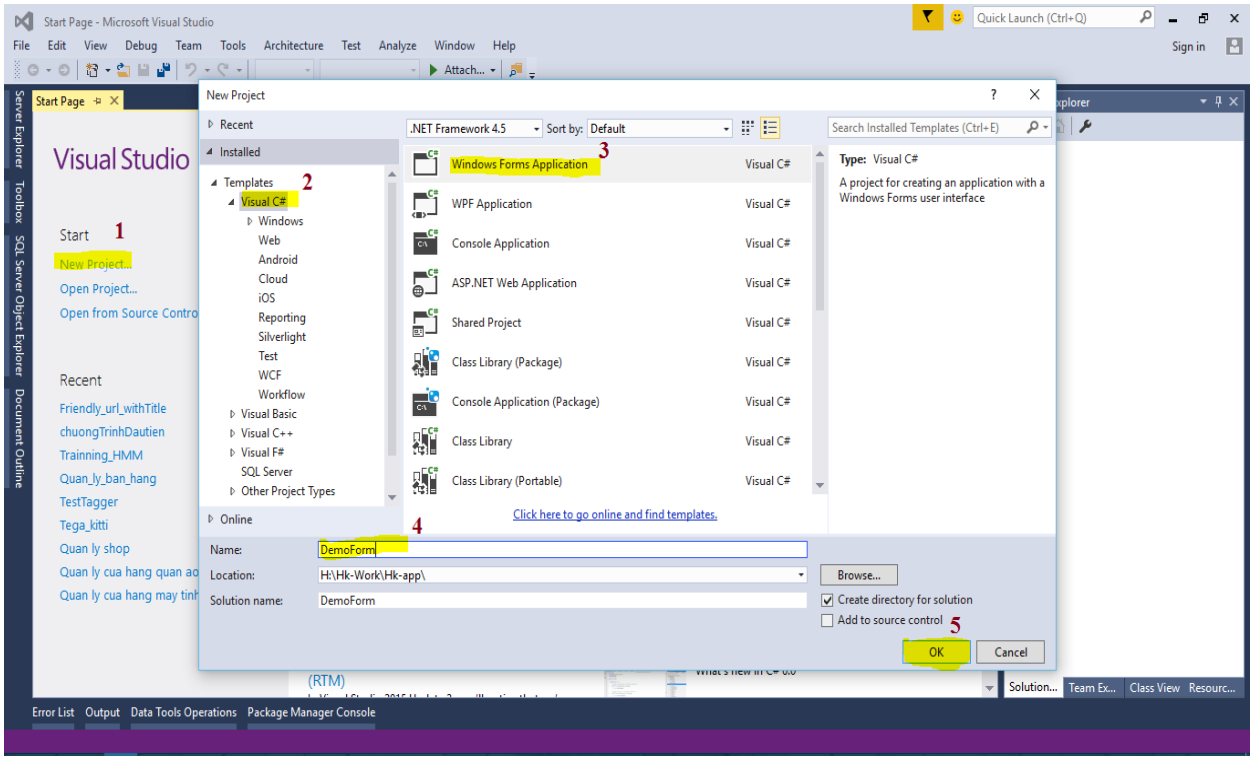
II. GIỚI THIỆU VỀ WINDOWS APPLICATION

Khái niệm

- Là ứng dụng được xây dựng trên giao diện là cá màn hình (Form)
- Người dùng tương tác với ứng dụng qua các thành phần giao diện như : màn hình (Form), thực đơn(Menu), thanh công cụ(Toolbar),..
- Cho phép tại một thời điểm có thể mở nhiều màn hình (Multi Document Interface)
- Namespace System.Windows.Form hỗ trợ ứng dụng này
- Có giao diện thân thiện với người dùng

1. Thao tác tạo

Từ thực đơn, chọn File → New → Project → Thiết lập các thông tin : Name , Location, Solution name.



2. Xây dựng màn hình chính cho ứng dụng

Tổng quan về ứng dụng MDI

- Ứng dụng MDI là ứng dụng tổ chức các màn hình con (Child form) trong cùng một màn hình cha(Parent form)
- Cho phép người dùng mở, tổ chức và làm việc trên nhiều tài liệu cùng lúc
- MDI parent form chứa tất cả các Child form tương tác với người dùng, quản lý cách trình bày và tổ chức cá Child form

3. **Xây dựng thực đơn**

- Sử dụng điều khiển MenuStrip và lần lượt thêm các chức năng (gọi là ToolStripMenuitem vào)
- Sử dụng sự kiện Click để xử lý trên menu thực đơn

4. **Các điều khiển trong Form**

- Thanh điều khiển ToolBox dùng để xây dựng giao diện người dùng, hỗ trợ kéo thả cho người dùng, dễ dàng sử dụng, xây dựng nhanh chóng
- Các điều khiển cơ sở đều kế thừa lớp control, nằm trong thanh điều khiển ToolBox, một số điều khiển trong Form như: Label, LinkLabel, Button, RadioButton, PictureBox, TextBox, CheckBox, MessageBox, OpenFileDialog, ListBox, vv

III. TỔNG QUAN VỀ GAME CARO

Nguồn gốc trò chơi

- Cờ caro là môn cờ logic lâu đời và cổ xưa trên Trái đất, được sáng tạo từ nhiều nền văn minh khác nhau một cách độc lập. Một số nhà khoa học đã tìm thấy bằng chứng chứng minh Caro đã được phát minh ở thời Hy Lạp cổ đại và ở châu Mỹ trước thời Colombo
- Luật chơi cờ Caro : Quân cờ được sử dụng là X và O, người người chơi đi luân phiên nhau, mỗi lần đi là một lượt và không trùng với lượt đi trước đó. Mỗi bên được xác định thắng khi tạo thành một hàng dọc, ngang hoặc chéo 5 quân cờ của mình

Phân tích bài toán

- Biểu diễn bài toán dưới dạng cây trò chơi

-Trò chơi có thể được biểu diễn như một cây gồm gốc, nút, lá và nhánh

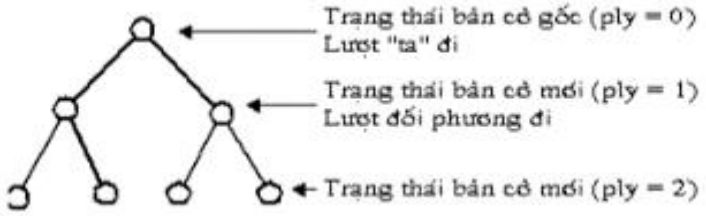
Gốc là trạng thái ban đầu của trò chơi.Với mỗi trò chơi cụ thể thì trạng thái (ở mỗi thời điểm) lại được đặc trưng bởi những thông số riêng

- Các nút (Node) của cây thể hiện tình trạng hiện tại của trò chơi, gồm nút cha (Parent Node) và nút con (Children Node)

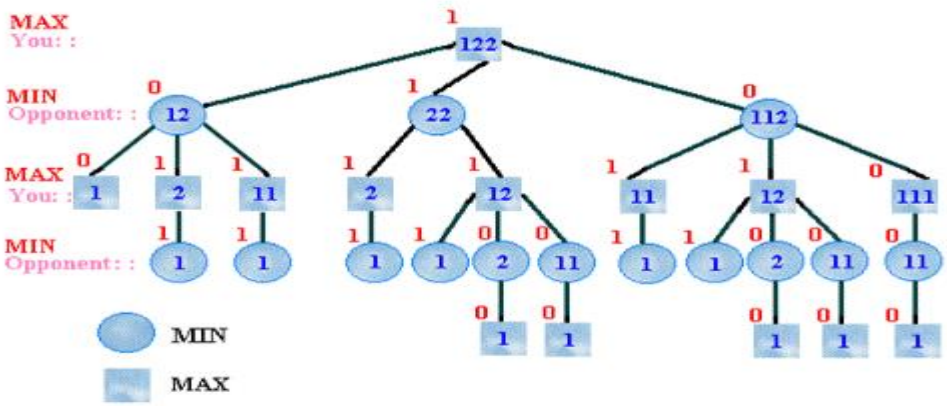
- Các nhánh nối giữa các nút thể hiện nước đi, tức là cho biết từ một tình huống chuyển sang tình huống khác thông qua chỉ một nước đi nào đó.

- Các lá (leave) hay còn gọi là nút lá (leave node), thể hiện thời điểm kết thúc khi mà kết quả của trò chơi đã rõ ràng

Dựa vào cây trò chơi này,người ta có thể tìm ra nước đi "tốt" để giành phần thắng cho mình (nếu có thể),



- Giải thuật min max được áp dụng cho việc tìm kiếm trường hợp "Tốt"



IV. THIẾT KẾ CÀI ĐẶT ỨNG DỤNG

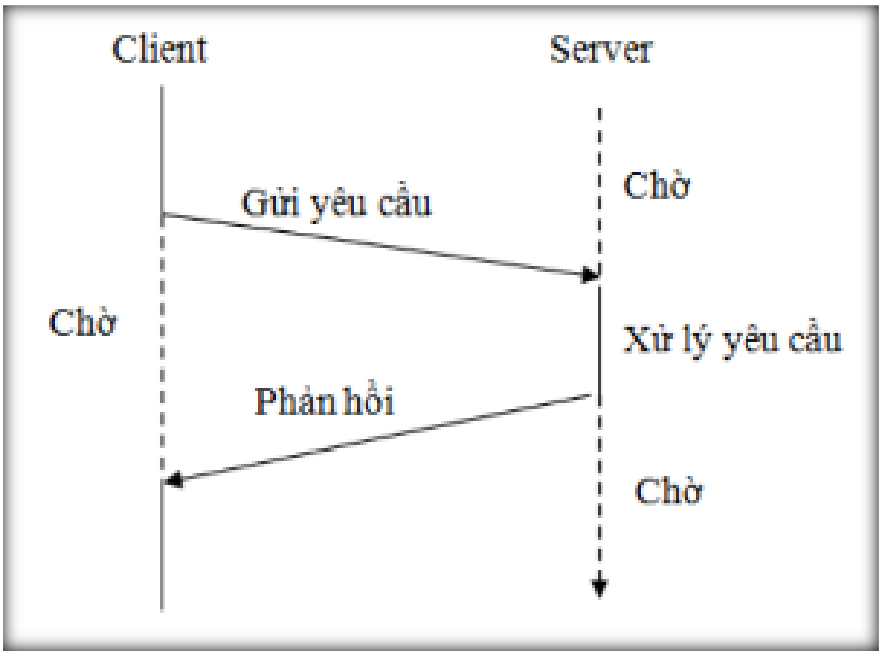
Mô hình Client-Server

- Trong việc xây dựng ứng dụng game này, em đã ứng dụng mô hình truyền tin Client-Server để tạo kết nối giữa hai người chơi.
- Mô hình Client-Server (mô hình khách chủ) là mô hình cơ bản và chủ đạo nhất trong các mô hình mạng. Xét cho cùng, tất cả các mô hình và công nghệ mạng phổ biến hiện nay như mô hình mạng ngang hàng(Peer to Peer), mô hình phân tán… đều có nguồn gốc từ mô hình Client-Server.
- Mô hình Client-Server tổng quan gồm có một chương trình đóng vai trò Server. Đối với mô hình này, một tiến trình Server sẽ giao tiếp và phục vụ nhiều tiến trình Client thông qua các phương tiện truyền thông trong môi trường mạng. Các tiến trình Client và Server có thể cùng chạy trên một hoặc khác máy tính.

➢ ***Nguyên lý hoạt động***

Tiến trình Server luôn thụ động chờ lắng nghe yêu cầu từ phía Client, do đó nó là tiến trình hoạt động trước để lắng nghe yêu cầu từ phía Client. Nguyên lý hoạt động của mô hình gồm 4 bước :

- Bước 1 : Client chủ động gửi yêu cầu đến Server.
- Bước 2 : Server nhận và xử lý yêu cầu từ Client
- Bước 3 : Server phản hồi kết quả cho Client
- Bước 4 : Client nhận và xử lý phản hồi từ Server



Nguyên lý hoạt động mô hình Client-Server

Kết nối Server-Client với TCP/IP

Khi được chạy, server cần được xác định rõ địa chỉ IP và sẽ "lắng nghe" trên một port cụ thể. Server sẽ nằm trong trạng thái này cho đến khi client gửi đến một yêu cầu kết nối. Sau khi được server chấp nhận, một connection sẽ hình thành cho phép server và client giao tiếp với nhau. Cụ thể hơn, các bước tiến hành trên server và client mà ta cần thực hiện sử dụng giao thức TCP/IP trong C# (có thể chạy server và client trên cùng một máy)

1. Tạo bàn cờ :

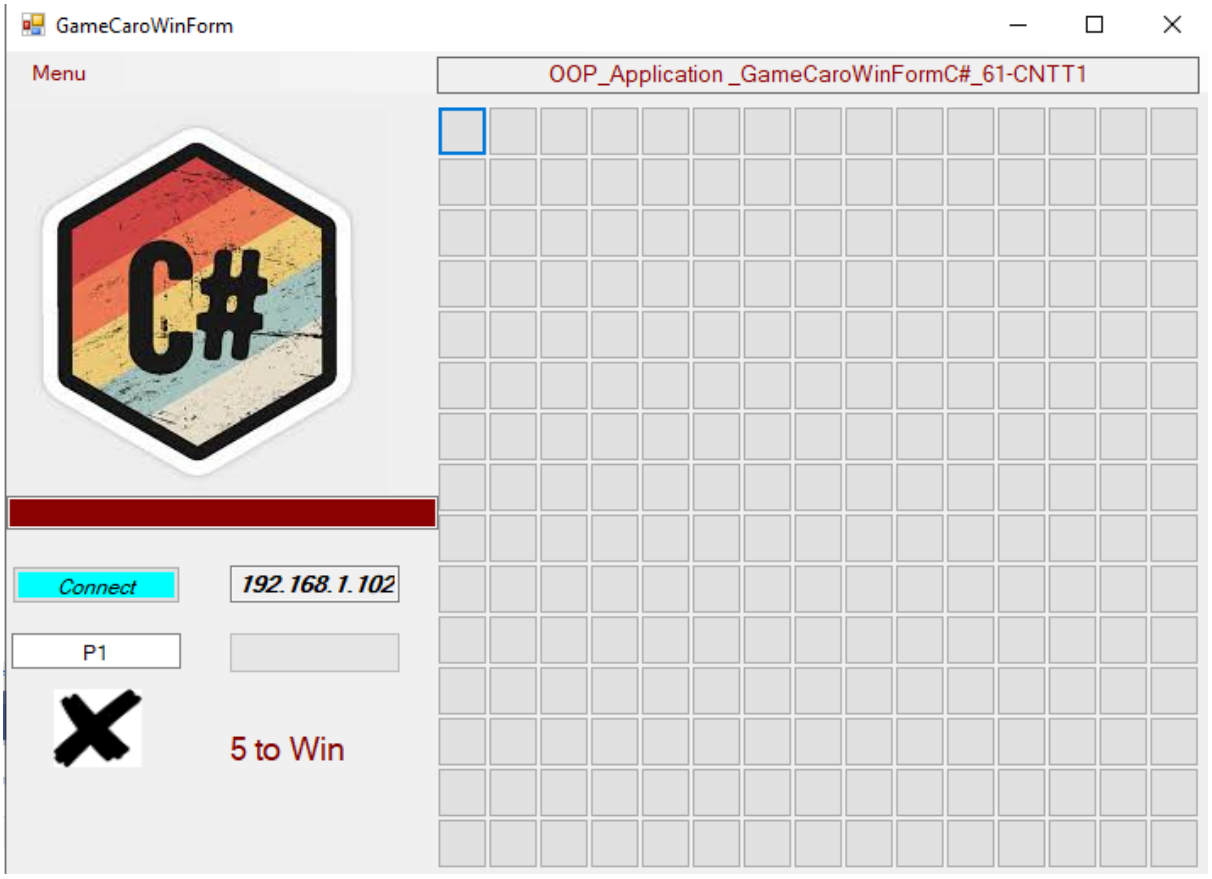
❖ Ý tưởng cho việc tạo bàn cờ trong ứng dụng :

- Là bàn cờ sẽ được cấu tạo bởi những button, mỗi ô cờ sẽ là một button để nhấn đặt quân cờ. Khi đó ta có thể sử dụng sự kiện nhấn , một số thuộc tính của button để sử dụng. Ta thêm button vào bàn cờ nhờ thuộc tính Add(), và xem nó như là những collection.

-Vị trí của button tiếp theo sẽ có 2 tọa độ x và y theo button ban đầu, thêm btn tiếp theo trên hàng ta có : btn.Location.X + btn.Width, btn.Location.Y, khi xuống hàng ta sẽ cập nhật lại các giá trị Height, Wight trở về 0.

- Dùng vòng lặp để có thể thực hiện việc vẽ bàn cờ là phương pháp hiệu quả nhất.

❖ Giao diện Game Caro đơn giản sau khi được thiết kế và tạo ra bàn cờ



❖ Code xử lý việc vẽ bàn cờ từ các button:

```
public void DrawBoard() // hàm vẽ bàn cờ
{
    Banco.Enabled = true;

    Banco.Controls.Clear();

    PlayTime = new Stack<InforPlayTime>();

    CurPlayer = 0;//  để xác định người chơi trước
    ChangePlayer(); // Gọi hàm chuyển người chơi , chuyển kí tự , tên người chơi

    Matrix = new List<List<Button>>(); //tao mot hang moi de luu

    Button old = new Button() { Width = 0, Location = new Point(0, 0) }; // tạo button mới

    for (int i = 0; i <= Const.boardh; i++)  //boar
    {
        Matrix.Add(new List<Button>());
        for (int j = 0; j <= Const.boardw; j++)
        {
            Button btn = new Button() // tạo đối tượng button
            {
                Width = Const.chessw,
                Height = Const.chessh,
                Location = new Point(old.Location.X + old.Width, old.Location.Y),
                BackgroundImageLayout = ImageLayout.Stretch,// sửa kích thước của ảnh cho bằng đúng kích thước button hiển thị
                Tag = i.ToString()  // xac dinh index
            };
            btn.Click += Btn_Click;
            Banco.Controls.Add(btn);
            Matrix[i].Add(btn);
            old = btn;
        }

            old.Location = new Point(0, old.Location.Y + Const.chessh); // xuống dòng và trả lại tọa độ 00 để trở lại đầu hàng
            old.Width = 0;
            old.Height = 0;

    }
```

2. Xử lý đổi người chơi

- Ý tưởng là ta sẽ nhấn vào button để đặt quân cờ thì nó sẽ đổi ký tự quân cờ thành ký tự của người chơi thứ hai và thực hiện lần lượt việc đổi tuần tự.

- Do lợi thế từ việc xây dựng các button từ code, nên trong vòng lặp for của hàm DrawBoard ta có thể gắn code event cho button, sau đó gọi hàm thực thi event

- Xây dựng hàm ChangePlayer để quản lý thông tin về người chơi, tên người chơi, ký tự quân cờ, khi bắt đầu thay đổi, tên và ký tự quân cờ sẽ được thay đổi tuần tự

- Khi một ô cờ đã được đặt thì ta không thực hiện việc đặt cờ nữa .

```
void Btn_Click(Object sender, EventArgs e)
{
    Button btn = sender as Button; // ép kiểu cho sender gửi even đi
                                   // đổi hình cho ô đó thành x o khi đc click vào
    if (btn.BackgroundImage != null)

        return; // Nếu button đã được đặt cờ thì không thực hiện đặt cờ được

    Mark(btn);

    PlayTime.Push(new InforPlayTime(ChessPoint(btn), curPlayer));

    CurPlayer = CurPlayer == 1 ? 0 : 1; // bắt đầu thực hiện việc đổi người chơi
    ChangePlayer();

    if (pMark != null)
    {
        pMark(this, new ButtonClickEvent(ChessPoint(btn)));
    }
    if (isEndGame(btn))
    {
        endGame();
    }
}
```

3. Xử lý thắng thua

-Để có thể quản lý các button trên bàn cờ cho việc xử lý thắng thua là vô cùng phức tạp và khó khăn khi ta dùng mảng 2 chiều, việc truy xuất từng phần tử và kiểm tra, do đó phương pháp được đề cập ở đây là sẽ tạo nên một cấu trúc List lồng List để quản lý

-Mỗi hàng sẽ là một List và tập hợp các hàng trong bàn cờ sẽ tạo thành một ma trận các danh sách button.

```
private List<List<Button>> matrix;
14 references
public List<List<Button>> Matrix
{
    get { return matrix; }
    set { matrix = value; }
}
```

- Từ đây, thông qua các thuộc tính của List để thao tác một cách dễ dàng như thuộc tính IndexOf mà ta có thể truy xuất được button đang xét.

```
public void DrawBoard() // hàm vẽ bàn cờ
{
    Banco.Enabled = true;

    Banco.Controls.Clear();

    PlayTime = new Stack<InforPlayTime>();

    CurPlayer = 0;//  để xác định người chơi trước
    ChangePlayer(); // Gọi hàm chuyển người chơi , chuyển kí tự , tên người chơi

    Matrix = new List<List<Button>>(); //tao mot hang moi de luu

    Button old = new Button() { Width = 0, Location = new Point(0, 0) }; // tạo button mới

    for (int i = 0; i <= Const.boardh; i++)  //boar
    {
        Matrix.Add(new List<Button>());
        for (int j = 0; j <= Const.boardw; j++)
        {
            Button btn = new Button() // tạo đối tượng button
            {
                Width = Const.chessw,
                Height = Const.chessh,
                Location = new Point(old.Location.X + old.Width, old.Location.Y),
                BackgroundImageLayout = ImageLayout.Stretch,// sửa kích thước của ảnh cho bằng đúng kích thước button hiển thị
                Tag = i.ToString()  // xac dinh index
            };
            btn.Click += Btn_Click;
            Banco.Controls.Add(btn);
            Matrix[i].Add(btn);
            old = btn;
        }

            old.Location = new Point(0, old.Location.Y + Const.chessh); // xuống dòng và trả lại tọa độ 00 để trở lại đầu hàng
            old.Width = 0;
            old.Height = 0;

    }
```

❖ ***Xử lý thắng thua trên hàng***

- Trước tiên, ta xây dựng một hàm ChessPoint có kiểu dữ trả về là Point để có thể nhận được button đặt cờ kết thúc, ta sẽ lấy được tọa độ x, y của button đó và tiến hành kiểm tra trên hàng, cột, chéo chính, chéo phụ.

```
private Point ChessPoint(Button btn)
{

    int ver = Convert.ToInt32(btn.Tag); // lấy tọa độ y điểm đặt cờ  kết thúc
    int hor = Matrix[ver].IndexOf(btn); // lay toa do x điểm đặt cờ kết thúc
    Point p = new Point(hor, ver);

    return p;
}
```

- Ý tưởng là sẽ kiểm tra từ button đặt cờ kết thúc game, ta duyệt bên trái và bên phải của button đó, đếm xem có bao nhiêu button có kí tự quân cờ giống với nó, nếu tổng bằng 5 thì trả về đúng

```
private bool isEndGameHor(Button btn)
{

    Point p = ChessPoint(btn);
    int top = 0;
    for (int i = p.Y; i >= 0; i--) // đếm số điểm giống bên trái
    {
        if (Matrix[i][p.X].BackgroundImage == btn.BackgroundImage)  // nếu 2 kí tự cờ của chúng giống nhau
        {
            top++;
        }
        else
        {
            break;
        }

    }
    int bottom = 0;
    for (int i = p.Y + 1; i < Const.boardh; i++) // đếm số điểm giống bên trái
    {

        if (Matrix[i][p.X].BackgroundImage == btn.BackgroundImage)
        {
            bottom++;
        }
        else
        {
            break;
        }
    }
    return top + bottom == 5;
}
1 reference
```

❖ *Xử lý thắng thua trên cột*

```
private bool isEndGameVer(Button btn)
{
    Point p = ChessPoint(btn);
    int left = 0;
    for (int i = p.X; i >= 0; i--) // đếm số điểm giống bên trái
    {

        if (Matrix[p.Y][i].BackgroundImage == btn.BackgroundImage)
        {
            left++;
        }
        else
        {
            break;
        }
    }
    int right = 0;
    for (int i = p.X + 1; i < Const.boardw; i++) // đếm số điểm giống bên trái
    {

        if (Matrix[p.Y][i].BackgroundImage == btn.BackgroundImage)
        {
            right++;
        }
        else
        {
            break;
        }
    }
    return left + right == 5;

}
```

❖ *Xử lý thắng thua trên đường chéo chính*

```
private bool isEndGameMainDia(Button btn)
{

    Point p = ChessPoint(btn);
    int top = 0;
    for (int i = 0; i <= p.X ; i++)
    {

        if (p.X - i < 0 || p.Y - i < 0)
            break;
        if (Matrix[p.Y-i][p.X-i].BackgroundImage == btn.BackgroundImage)
        {
            top++;
        }
        else
        {
            break;
        }
    }
    int bottom = 0;

    for (int i =  1; i <= Const.boardw - p.X; i++)
    {

        if (p.Y + i >= Const.boardh || p.X + i >= Const.boardw)
            break;
        if (Matrix[p.X+i][p.Y+i].BackgroundImage == btn.BackgroundImage)
        {
            bottom++;
        }
        else
        {
            break;
        }
    }
    return top + bottom == 5;
}
```

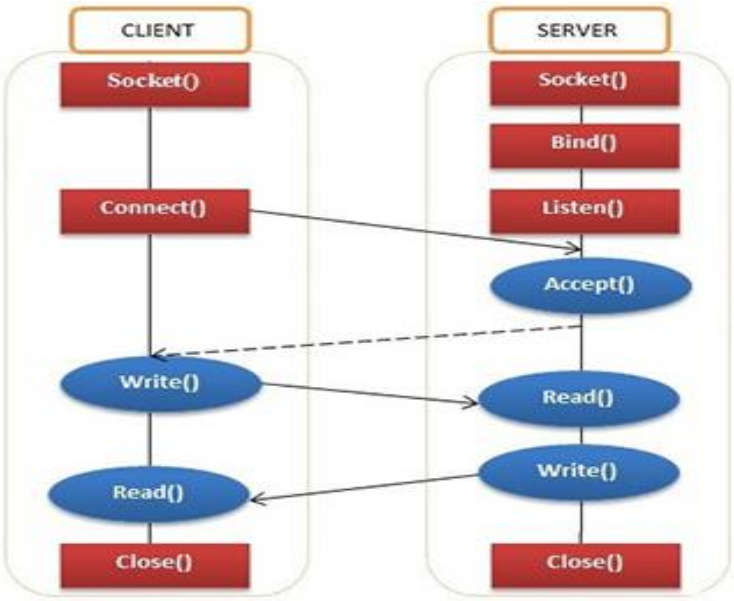
❖ ***Xử lý thắng thua trên đường chéo phụ***

```
private bool isEndGameSubDia(Button btn)
{
    Point p = ChessPoint(btn);
    int top = 0;
    for (int i = 0; i <= p.X; i++)
    {

        if (p.X + i > Const.boardw || p.Y - i < 0)
            break;
        if (Matrix[p.Y - i][p.X + i].BackgroundImage == btn.BackgroundImage)
        {
            top++;
        }
        else
        {
            break;
        }
    }
    int bottom = 0;

    for (int i = 1; i <= Const.boardw - p.X; i++)
    {

        if (p.Y + i >= Const.boardh || p.X - i < 0)
            break;
        if (Matrix[p.Y + i][p.X - i].BackgroundImage == btn.BackgroundImage)
        {
            bottom++;
        }
        else
        {
            break;
        }
    }
    return top + bottom == 5;
}
```

4. Tạo kết nối mạng LAN cho 2 người chơi

- Khái niệm socket : ***Socket là điểm cuối end-point trong liên kết truyền thông hai chiều (two-way communication) biểu diễn kết nối giữa Client – Server***. Các lớp Socket được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể định danh ứng dụng mà dữ liệu sẽ được gửi tới.
- Socket hoạt động thông qua các tầng TCP hoặc TCP Layer định danh ứng dụng, từ đó truyền dữ liệu thông qua sự ràng buộc với một cổng port

- **Socket** là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên internet. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều, hay còn gọi là two-way communication để kết nối 2 process trò chuyện với nhau. Điểm cuối (endpoint) của liên kết này được gọi là socket.
- Một chức năng khác của socket là giúp các tầng **TCP** hoặc **TCP Layer** định danh ứng dụng mà dữ liệu sẽ được gửi tới thông qua sự ràng buộc với một cổng port (thể hiện là một con số cụ thể), từ đó tiến hành kết nối giữa client và server



  - Ở đây ứng dụng game Caro có thể được xem vừa là Server vừa là Client, đầu tiên sẽ thử kết nối nếu có " phòng " rồi thì sẽ kết nối và trở thành Client, nếu thử kết nối thất bại, nó sẽ trở thành Server và tiến hành " nghe " client kết nối.
- Việc sử dụng được giao thức truyền tin trên, các namespace hỗ trợ tính năng bao gồm:

```
using System.Net;
using System.Net.NetworkInformation;
using System.Net.Sockets;
```

- Việc truyền được quân cờ của 2 người chơi phải được thực hiện thông qua việc thực hiện quá trình nén một đối tượng thành một mảng byte để lưu trữ, sau khi kết nối thành công, dữ liệu nhận được sẽ được giải nén trở lại thành một đối tượng (object)

```
//  nén đối tượng thành mảng byte[]
1 reference
public byte[] SerializeData (Object o)
{
    MemoryStream ms = new MemoryStream();
    BinaryFormatter bf1 = new BinaryFormatter();
    bf1.Serialize(ms, o);
    return ms.ToArray();
}
//giải nén mảng byte thành một đối tượng để nhận và hiểu được thông tin
1 reference
public object DeserializeData(byte[] theByteArray)
{

    MemoryStream ms = new MemoryStream(theByteArray);
    BinaryFormatter bf1 = new BinaryFormatter();
    ms.Position = 0;
    return bf1.Deserialize(ms);
}
```

- Để giúp người chơi có thể truy cập được địa chỉ IPv4 và dùng địa chỉ IP đó để kết nối với Server ta có thể dùng hàm GetLocalIPv4 sau:

```
//  lấy ra giá trị IP4 của card mạng đang sử dụng
2 references
public string GetLocalIPv4 (NetworkInterfaceType _type)
{
    string output = "";
    foreach(NetworkInterface item in NetworkInterface.GetAllNetworkInterfaces())
    {
        if (item.NetworkInterfaceType == _type && item.OperationalStatus == OperationalStatus.Up)
        {
            foreach (UnicastIPAddressInformation ip in item.GetIPProperties().UnicastAddresses)
            {
                if(ip.Address.AddressFamily == AddressFamily.InterNetwork)
                {
                    output = ip.Address.ToString();
                }
            }
        }
    }
    return output;
}
```

- Chương trình Server hỗ trợ việc tạo kết nối với Client

```
Socket server;
1 reference
public void CreateServer()
{
    IPEndPoint iep = new IPEndPoint(IPAddress.Parse(IP), PORT);
    server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    server.Bind(iep);
    server.Listen(20); //  đợi kết nối với client trong 20s

    Thread acceptClient = new Thread(() => // tạo 1 luồng khác thực hiện việc chấp nhập kết nối
    {
        client = server.Accept();//  chấp nhận kết nối
    });
    acceptClient.IsBackground = true;
    acceptClient.Start();
}
```

- Chương trình Client hỗ trợ việc truy cập đến kết nối với Server đã được tạo thông qua địa chỉ Ipv4 và cổng PORT

```
public class SocketManager
{
    #region Client
    Socket client;
    1 reference
    public bool ConnectServer()  // hàm kết nối cliet với server
    {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse(IP), PORT);
        client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        try
        {
            client.Connect(iep);
            return true;
        }
        catch
        {
            return false;
        }
    }
    #endregion
```

- Hàm thực hiện việc gửi và nhận dữ liệu giữa Client và Server thông qua Socket

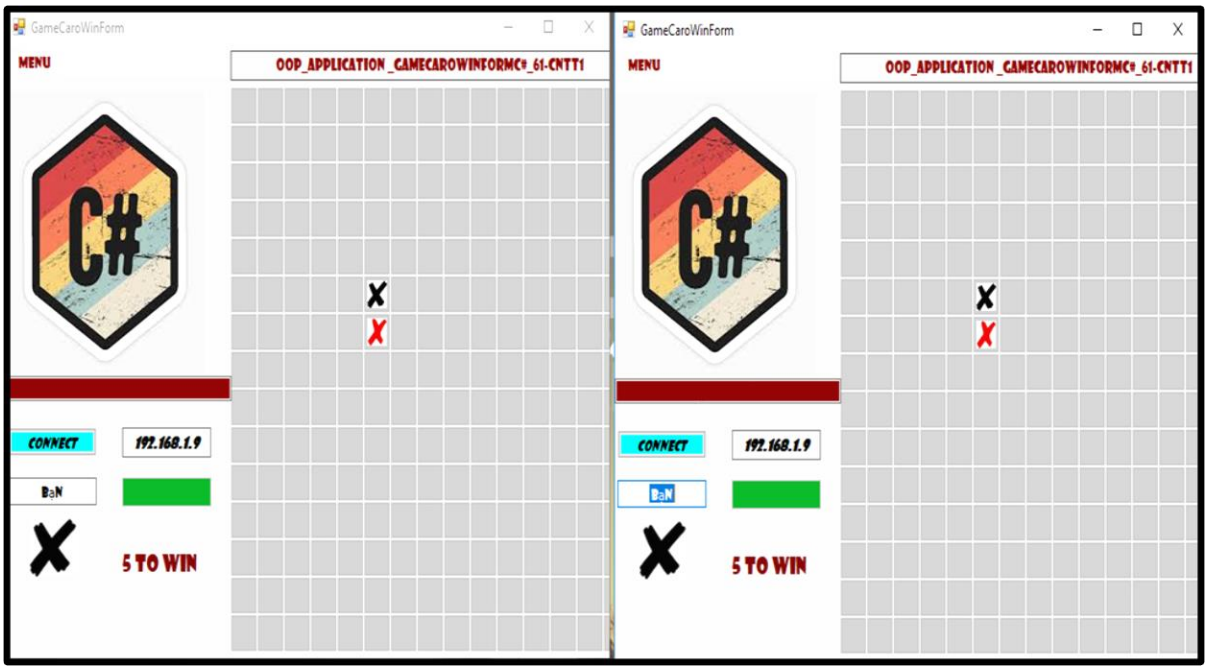
```
#region Both
public string IP = "127.0.0.1";
public int PORT = 9999;
public const int BUFFER = 1024;
public bool isServer = true;
6 references
public bool Send(object data) // gửi dữ liệu từ server và client
{
    byte[] sendData = SerializeData(data);

        return SendData(client, sendData);

}
1 reference
public object Receive() // hàm nhận thông tin
{
    byte[] receiveData = new byte[BUFFER];

    bool isOk = ReceiveData(client, receiveData);

    return DeserializeData(receiveData);
}
1 reference
private bool SendData(Socket target, byte[] data)
{
    return target.Send(data) == 1 ? true : false;
}
1 reference
private bool ReceiveData(Socket target, byte[] data)// không dùng từ khóa ref do mảng byte đc dùng tham chiếu
{
    return target.Receive(data) == 1 ? true : false;
}
```

- Để tối ưu chương trình ta có thể sử dụng lập trình bất động bộ để thực hiện hai phương thức này một cách đồng thời bằng cách cho mỗi phương thức được thực thi trên một luồng khác nhau. Bằng cách đó, các phương thức được thực thi một cách đồng thời, có nghĩa là nhiều tác vụ được thực thi cùng một lúc.
- Để làm giảm bớt dung lượng tài nguyên cho hàm Main, đồng thời có thể giảm thời gian phản hồi từ chương trình, chúng ta phải tạo một luồng xử lý mới để xử lý cho việc kiểm tra việc đã tạo được kết nối giữa Client và Server hay chưa, luồng xử lý này sẽ thực hiện độc lập với hàm Main của game.

  Trong ngôn ngữ C# để thực sử dụng **lập trình bất đồng bộ** ta có thể sử dụng đối tượng **Thread** trong namespace System.Threading hoặc đối tượng Tasks trong System.Threading.Tasks

```
void Listen()
{
    Thread listenThread = new Thread(() =>
    {
        try
        {
            SocketData data = (SocketData)socket.Receive();

            ProcessData(data);

        }
        catch (Exception e) { }

    });
    listenThread.IsBackground = true;
    listenThread.Start();

}
1 reference
private void btnConnect_Click(object sender, EventArgs e)
{
    socket.IP = textIP.Text;

    if (!socket.ConnectServer())
    {
        socket.isServer = true;
        panelBanco.Enabled = true;
        socket.CreateServer();
    }
    else
    {
        socket.isServer = false;
        panelBanco.Enabled = false;
        Listen();
    }
}
```

- Qua quá trình truyền và nhận dữ liệu của chương trình cho ta truyền được quân cờ cho 2 người chơi
- Ta mở 2 file .exe để chạy trên một máy , chúng sẽ truy cập địa chỉ Ipv4 của máy và tạo Server để Client kết nối



5. Xử lý tác vụ Undo

- ✓ Việc xây dựng tác vụ Undo dựa trên nguyên tắc, sau khi ta lưu trữ thông tin các lượt chơi vào stack InfoPayTime, ta sử dụng thao tác Pop để lấy ra được lượt đặt cờ cuối cùng.
- ✓ Tiếp theo, sử dụng thuộc tính của Point để lấy được tọa độ button được đặt cờ cuối cùng đó, từ đó ta thu hồi quân cờ đã được đặt, cập nhật lại BackgroundImage cho nó bằng Null và button chưa được đặt cờ.
- ✓ Sau khi Undo, ta phải cập nhật lại lượt chơi cũng như hình ảnh quân cờ sẽ đi tiếp theo trên bàn cờ bằng việc gọi hàm ChangePlayer
- ✓ Trong Fom.cs, ta xây dựng một menu các lựa chọn chứa tác vụ Undo.
- ✓ Tính năng Undo này phải được hiểu ở cả 2 người chơi tránh trường hợp chỉ thu hồi nước cờ ở 1 bên người chơi bên còn lại thì không, để khắc phục lỗi này, cần phải có sự trao đổi với nhau giữa 2 người chơi bằng việc sau khi kích hoạt tác vụ Undo từ màn hình một người chơi, một socket sẽ chịu trách nhiệm giúp chúng ta gửi đến cho người chơi còn lại một thông điệp Undo ngay, từ đó 2 quân cờ ở cả 2 người chơi đều được thu hồi ngay lập tức
- ✓ Đây là đoạn code khắc phục lỗi trên

```
private void undoToolStripMenuItem_Click(object sender, EventArgs e)
{
    Undo();
    socket.Send(new SocketData((int)SocketCommand.UNDO, "", new Point()));// thêm dòng lệnh này để sửa lỗi undo 2 bên
}
```

✓ **Các hàm được xây dựng trong class ChessBoardManager như sau :**

```
1 reference
public bool Undo()
{
    if (PlayTime.Count <= 0)
        return false;

    bool isUndo1 = UndoAStep();
    bool isUndo2 = UndoAStep();

    InforPlayTime oldPoint = PlayTime.Peek();
    CurPlayer = oldPoint.CurPlayer == 1 ? 0 : 1;

    return isUndo1 && isUndo2;// thực hiện việc undo 2 lần cho 2 mà hình người chơi
}

2 references
private bool UndoAStep()
{
    if (PlayTime.Count <= 0)
        return false;

    InforPlayTime oldPoint = PlayTime.Pop(); //xác định được điểm đặt quân cờ cuối cùng từ stack chứa lượt chơi

    Button btn = Matrix[oldPoint.Point.Y][oldPoint.Point.X]; // từ đó lấy được button cần undo, thu hồi quân cờ đã được đặt

    btn.BackgroundImage = null;

    if (PlayTime.Count <= 0)
    {
        CurPlayer = 0;
    }
    else
    {
        oldPoint = PlayTime.Peek();
    }

    ChangePlayer();

    return true;
}
```

6. Xử lý New Game

➢ Mỗi khi người chơi sử dụng tác vụ newgame trong menu hoặc sử dụng hotkey thì lập tức hàm new game sẽ được gọi thực hiện, thanh đếm giờ Processbar và thời gian chờ sẽ dừng đếm trả về 0, chức năng Undo sẽ được mở. Đồng thời thực hiện việc vẽ lại bàn cờ mới cho 2 người chơi.

```
void newGame()
{
    proDemgio.Value = 0;
    timer1.Stop();
    undoToolStripMenuItem.Enabled = true;
    Banco.DrawBoard();
}
1 reference
private void newGameToolStripMenuItem_Click(object sender, EventArgs e)
{
    newGame();
    socket.Send(new SocketData((int)SocketCommand.NEW_GAME, "", new Point()));
    panelBanco.Enabled = true;
}
```

- Trong Form.cs ta có các khai báo và thực hiện những thao tác sau

```
public partial class GameCaro : Form
{
    #region Properties

    ChessBoardManager Banco;
    SocketManager socket;

    #endregion

    1 reference
    public GameCaro()
    {
        InitializeComponent();

        Control.CheckForIllegalCrossThreadCalls = false;

        Banco = new ChessBoardManager(panelBanco, textName, picBox);

        Banco.EGame += Banco_EGame1;
        Banco.PMark += Banco_PMark;

        proDemgio.Step = Const.stepcooldown;
        proDemgio.Maximum = Const.cooldown;
        proDemgio.Value = 0;
        timer1.Interval = Const.intervalcooldown;
        socket = new SocketManager();
        newGame();
    }
```

V. KẾT QUẢ

- Xây dựng được ứng dụng game Caro 2 người chơi đơn giản thông qua Windows Form
- Phân tích thiết kế xây dựng các event, cấu trúc dữ liệu cho các lớp trong ứng dụng một cách rõ ràng cụ thể
- Về cá nhân, em đã có cơ hội được ôn tập, rèn luyện củng cố nhiều kĩ thuật về lập trình WinForm, lập trình Hướng đối tượng như :
    - ✓ Cách sử dụng các control cơ bản
    - ✓ Hàm ủy quyền Delegate
    - ✓ Collection, Stack, List
    - ✓ Lập trình mạng Socket, Mô hình Client – Server
    - ✓ Multi - thread, Event, Kế thừa , đóng gói thuộc tính ,vv
- Thực hiện được các tính năng như
    - ✓ Tạo kết nối 2 người chơi thông qua mô hình Client-Server
    - ✓ Chức năng Undo, New game, Quit game
    - ✓ Hotkey, Progressbar
    - ✓ Truy cập địa chỉ IPv4 của người dùng thông qua hàm hỗ trợ.
- Hướng phát triển của đề tài :
    - ✓ Tiếp tục xây dựng thêm một số tính năng phức tạp như AI hỗ trợ chơi với máy, chơi online thông qua mạng Internet, nhằm đáp ứng được yêu cầu của người dùng
    - ✓ Khắc phục khuyết điểm việc xây dựng ô bàn cơ với nhiều button gây hao tốn bộ nhớ

TÀI LIỆU THAM KHẢO

[1] Lập trình game Caro WinForm C# - HowKteam

https://www.howkteam.vn/course/lap-trinh-game-caro-voi-c-winform/demo-game-caro-lap-trinh-c-winform-111

[2] Lập trình mạng : Xây dựng ứng dụng Client – Server hướng kết nối (TCP Socket)

https://sinhvientot.net/lap-trinh-mang-xay-dung-ung-dung-client-server-huong-ket-noi-tcp-socket/

[3] Slide bài giảng Lập trình WinForm – Cô Phan Thị Kim Ngoan, Trường Đại học Nha Trang

https://elearning.ntu.edu.vn/pluginfile.php/356517/mod_resource/content/1/Chu%20De%207.pdf