

Bài báo cáo Lab6

Tên chương trình: game đoán số

Thành viên nhóm:

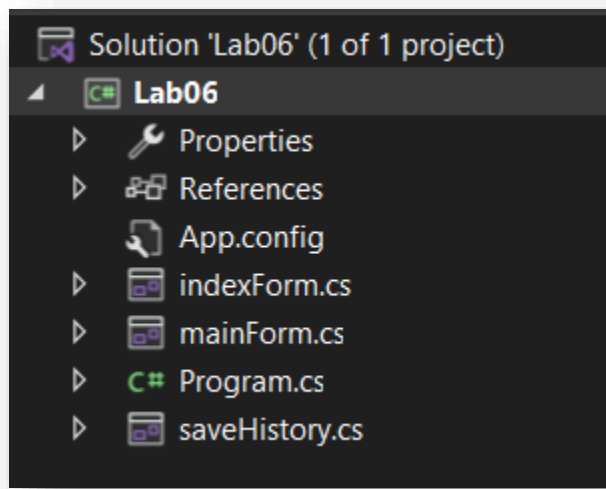
- Mai Nguyễn Nam Phương
- Hồ Diệp Huy

- Các yêu cầu của chương trình đã hoàn thành:

- ✓ Xây dựng ứng dụng trên mô hình Server – Clients
- ✓ Bên phía Clients, thiết kế giao diện cho phép điền tên người chơi, điền con số dự đoán và thông báo của server (lượt chơi mới bắt đầu, người thắng cuộc, phạm vi số cần đoán,...)
- ✓ Bên phía Server thiết kế giao diện hiển thị các thông tin số đang cần tìm, phạm vi số đang cần tìm, lịch sử trò chơi, số người đang tham gia, số lượt chơi.
- ✓ Giới hạn khoảng thời gian giữa 2 lần gửi kết quả dự đoán và hiển thị thời gian đếm ngược trên Form của client.
- ✓ Phát triển tính năng tự động chơi, cho phép client sinh ngẫu nhiên kết quả và tự động gửi (các kết quả tự sinh ra không trùng với kết quả đã thử trước đó và thuộc phạm vi con số cần tìm).
- ✓ Server tự quyết định số lần chơi để kết thúc trò chơi. Khi trò chơi kết thúc, phía clients sẽ lưu toàn bộ lịch sử thông báo của server thành file history.txt, server gửi toàn bộ lịch sử trò chơi lên website <https://ctxt.io> (hoặc các trang web khác có chức năng tương tự)
- ✓ Khi kết thúc trò chơi, thông báo người thắng chung cuộc (dựa trên các tiêu chí: có số lượt trả lời đúng nhiều nhất à có tổng số lần dự đoán sai ít nhất).
- ✓ Thêm tính năng tính điểm cho người chơi (ví dụ: trừ điểm mỗi lần đoán sai, cộng điểm mỗi lần đoán đúng)

- Giải thích về các đoạn code:

- Chương trình gồm 3 form bao gồm form chính để chơi trò chơi, form đăng nhập và form lưu lịch sử chơi



The image shows a Windows application window titled 'Dashboard'. The main content area has a yellow background and displays the title 'GUESS NUMBER' in large, green, italicized letters. Below the title, there are four input fields with labels: 'Username:', 'IP Address:', 'Port:', and 'ID Room:'. The 'IP Address' field contains the text 'localhost', and the 'Port' and 'ID Room' fields contain the number '65535'. At the bottom of the window, there are two large, yellow buttons with green text: 'New room' and 'Join room'.

Player

Range of number:


Answer:

Chat:

Your Answer:

Autoplay x1 Autoplay Clear

Ready



I/ indexForm

- Trong form đăng nhập thì người dùng có thể nhập tên của mình, sau đó là chọn IP và Port để tạo phòng hoặc tham gia phòng đã có sẵn (mặc định người tạo phòng sẽ là admin server không được tham gia vào trò chơi)
- Đầu tiên ta sẽ khởi tạo các biến liên quan tới user và rồi thiết lập kết nối TCP + Thread cho chương trình

```
namespace Lab06
{
    5 references
    public partial class indexForm : Form
    {
        private Thread thread = null;
        private TcpListener serverSocket;
        private MainForm serverForm = null;
        private int clientsCount = 0, currentRound, timeupCount, startRange, endRange, ansNumber, round = 0;
        private String correctPlayer, time = "";
        private Random rand;
        private bool ingame = false;
        private readonly object _lock = new object();
        private Dictionary<String, int> scoreBoard = new Dictionary<string, int>();
        private Dictionary<String, bool> readyPlayers = new Dictionary<string, bool>();
        private readonly Dictionary<String, TcpClient> clientsList = new Dictionary<string, TcpClient>();
    }
}
```

- Sau đó dựa trên kiểu tham gia là tạo phòng hay tham gia phòng thì ta sẽ tiếp cận được vào trò chơi

```
1 reference
private void btnClient_Click(object sender, EventArgs e)
{
    this.Hide();
    if (serverForm != null)
    {
        serverForm.Show();
        return;
    }

    serverForm = new MainForm(this, joinUsername.Text, joinIP.Text, joinPort.Text, time);
    serverForm.Show();

    if (thread == null)
    {
        serverForm = null;
    }
}

1 reference
private void btnServer_Click(object sender, EventArgs e)
{
    btnServer.Enabled = false;
    thread = new Thread(serverThread);
    thread.Start();
}
```

- Ta thiết lập các hàm xử lý cho trò chơi của ta cho mọi người trong phòng biết có thêm ai vào phòng, hàm random số vòng chơi khi trò chơi được bắt đầu, hoặc là

thông báo đã có người tìm được đúng số cần tìm (khi đó sẽ cộng 10 điểm cho người đúng, -1 mỗi lần đoán sai).. ngoài ra còn có thông báo người chiến thắng khi trò chơi kết thúc.

```
private void serverThread()
{
    int port;
    try
    {
        port = Int32.Parse(hostPort.Text);
        serverSocket = new TcpListener(IPAddress.Any, port);
        serverSocket.Start();
    }
    catch
    {
        btnServer.Invoke(new MethodInvoker(delegate ()
        {
            btnServer.Enabled = true;
        }));
        MessageBox.Show("Lỗi port!");
        return;
    }

    time = DateTime.Now.ToString("h:mm:ss tt");

    this.Invoke(new MethodInvoker(delegate ()
    {
        joinUsername.Text = "Server";
        joinIP.Text = "localhost";
        joinPort.Text = hostPort.Text;
        joinUsername.Enabled = joinIP.Enabled = joinPort.Enabled = hostPort.Enabled = false;
    }));
}
```

```

(new Thread(() => this.Invoke(new MethodInvoker(delegate ()
{
    btnClient.PerformClick();
}))))).Start();

MessageBox.Show("Tạo game mới thành công");

while (Thread.CurrentThread.IsAlive)
{
    TcpClient client = null;
    try
    {
        client = serverSocket.AcceptTcpClient();
    }
    catch (SocketException e)
    {
        if ((e.SocketErrorCode == SocketError.Interrupted))
        {
            break;
        }
    }

    NetworkStream stream = client.GetStream();
    byte[] buffer = new byte[1024];
    int bytesCount = stream.Read(buffer, 0, buffer.Length);
    String username = Encoding.UTF8.GetString(buffer, 0, bytesCount);

    if (thread != null && ingame)
    {
        buffer = Encoding.UTF8.GetBytes("!!!Ingame!!!!");
        stream.Write(buffer, 0, buffer.Length);
        continue;
    }
}

```

```

        if (username == " ")
        {
            username = $"Player {clientsCount}";
        }

        if (clientsList.ContainsKey(username))
        {
            buffer = Encoding.UTF8.GetBytes(" ");
            stream.Write(buffer, 0, buffer.Length);
            continue;
        }
        buffer = Encoding.UTF8.GetBytes(username);
        stream.Write(buffer, 0, buffer.Length);


        lock (_lock) clientsList.Add(username, client);
        if (username != "Server")
        {
            broadcast($"m>>> {username} vừa vào phòng chơi", username);
            scoreBoard.Add(username, 0);
        }

        clientsCount++;

        Thread handlingThread = new Thread(o => clientCheck((string)o));
        handlingThread.Start(username);
        broadcast($"t{clientsList.Count - 1}");
    }

    btnServer.Invoke(new MethodInvoker(delegate ()
    {
        btnServer.Enabled = true;
        btnServer.ResetText();
    }));
}

```

 `int Array.Length { get; }`
Gets the total number of elements in the array.

Returns:
The total number of elements in the array.

Exceptions:
`OverflowException`

```

// Set số round cho mỗi game và khoảng số cần đoán
2 references
private void roundStart()
{
    Thread.Sleep(2000);
    timeupCount = 0;

    if (round == 0)
    {
        round = rand.Next(3, 10);
        broadcast($"mTrò chơi có {round} round");
        currentRound = 1;
    }

    startRange = rand.Next(0, 50);
    endRange = startRange + rand.Next(1, 50);
    ansNumber = rand.Next(startRange, endRange + 1);
    broadcast($"m>>> Round {currentRound}: Đoán một số trong khoảng [{startRange}, {endRange}].\n@@@Nextround!@@@{rand.Next(5, 11)}\t{startRange}");
    currentRound++;
    correctPlayer = "";
}

```

```

private void timeUp()
{
    string message;
    if (correctPlayer == "") message = $"Không ai có đáp án chính xác";
    else message = $"m{correctPlayer} là người chơi đưa ra đáp án nhanh nhất, +10 điểm";
    broadcast($"m{message}\nmĐáp án là: {ansNumber}.\nm-----");
    if (currentRound > round) (new Thread(endGame)).Start();
    else if (ingame) (new Thread(roundStart)).Start();
}

```

```

private void endGame()
{
    if (ingame)
    {
        ingame = false;
        int highscore = int.MinValue;
        foreach (var i in scoreBoard)
        {
            if (i.Value > highscore)
            {
                highscore = i.Value;
            }
        }

        string message = $"mĐiểm cao nhất là: {highscore}\n";
        foreach (var i in scoreBoard)
        {
            if (i.Value == highscore)
            {
                message += $"mNgười chơi chiến thắng là: {i.Key}\n";
            }
        }

        foreach (var i in clientsList)
        {
            try
            {
                NetworkStream stream = i.Value.GetStream();
                byte[] buffer;
                if (i.Key == "Server")
                {
                    buffer = Encoding.UTF8.GetBytes($"m{message}\n");
                }
                else
                {

```

II/ mainForm

- Hàm đầu tiên sẽ kiểm tra có game tạo ở IP và Port người dùng nhập chưa, nếu người chưa thì người dùng chọn Server sẽ được làm host, còn Client thì sẽ không cho phép tham gia và xuất hiện thông báo chưa có phòng được tạo

```
1 reference
private void mainForm_Load(object sender, EventArgs e)
{
    answer.AutoSize = true;
    string username = joinUsername;
    IPAddress ip = null;
    int port = 0;
    if (username == "Username" || username == "") username = " ";
    try
    {
        ip = Dns.Resolve(joinIP).AddressList[0];
        port = Int32.Parse(joinPort);
    }
    catch
    {
        MessageBox.Show("Sai địa chỉ IP!");
        this.Close();
        return;
    }
    client = new TcpClient();
    try
    {
        client.Connect(ip, port);
    }
    catch
    {
        MessageBox.Show("Không có game nào đang diễn ra!");
        this.Close();
        return;
    }

    NetworkStream stream = client.GetStream();
    byte[] buffer = Encoding.UTF8.GetBytes(username);
    stream.Write(buffer, 0, buffer.Length);
}
```

- Hàm thông báo thời gian còn lại để đoán và thời gian cho phép đoán phải cách lần đoán gần nhất 3s

```
1 reference
private void timer_Tick(object sender, EventArgs e)
{
    timeLeft--;
    if (timeLeft > -1)
    {
        timerCnt.Text = timeLeft.ToString();
        if (timeLeft == 0)
        {
            btnSubmit.Enabled = btnAutoplayWholeGame.Enabled = btnAutoPlaySingleTurn.Enabled = answer.Enabled = label3.Enabled =
            send("@@@Timeup!@@@");
        }
        else if (isAuto && lastSubmitTime - timeLeft >= 3)
        {
            new Thread(() => autoSubmit()).Start();
        }
        else if (!isAuto && lastSubmitTime - timeLeft >= 3)
        {
            btnSubmit.Enabled = btnAutoPlaySingleTurn.Enabled = answer.Enabled = true;
            answer.Focus();
            answer.Select();
        }
    }
    else
    {
        timerCnt.Text = "X";
        timer.Stop();
    }
}
```

- Hàm submit dùng để cho người chơi tự submit đáp án của mình, dựa trên hàm đó ta có hàm auto submit để dùng cho chế độ Auto Play

```
4 references
private void autoSubmit()
{
    if (this.InvokeRequired)
        this.Invoke(new MethodInvoker(delegate ()
        {
            autoSubmit();
        }));
    else
    {
        btnSubmit.Enabled = btnAutoPlaySingleTurn.Enabled = answer.Enabled = false;
        if (isAuto)
        {
            btnAutoplayWholeGame.Enabled = false;
        }
        int val = rand.Next(0, valRange + 1);
        submit(ansList[val]);
    }
}
```

- Hàm nhận data dựa trên kết nối đã tạo ở indexForm, theo đó lấy được các giá trị như số lượng người chơi hay người chơi mới...

1 reference

```
private void ReceiveData(TcpClient client)
{
    NetworkStream stream = client.GetStream();
    byte[] receivedBytes = new byte[1024];
    int bytesCount;

    while (Thread.CurrentThread.IsAlive)
    {
        try
        {
            if ((bytesCount = stream.Read(receivedBytes, 0, receivedBytes.Length)) <= 0) break;
        }
        catch { break; }
        string respondFromServer = Encoding.UTF8.GetString(receivedBytes, 0, bytesCount);
        var dataList = respondFromServer.Split(new String[] { "\n" }, StringSplitOptions.RemoveEmptyEntries);
        foreach (String data in dataList)
        {
            if (data[0] == 'm')
            {
                conversation.Invoke(new MethodInvoker(delegate ()
                {
                    conversation.AppendText($"{data.Substring(1)}\n");
                    conversation.ScrollToCaret();
                }));
            }
            else if (data[0] == '\t')
            {
                this.Invoke(new MethodInvoker(delegate ()
                {
                    playerNum.Text = $"{data.Substring(1)} người chơi đã tham gia";
                }));
            }

            else if (data.StartsWith("@@@Nextround!@@@"))
            {
                if (!isIngame) isIngame = true;
            }
        }
    }
}
```

```
ansList = Enumerable.Range(startRange, valRange + 1).ToList();
range.Invoke(new MethodInvoker(delegate ()
{
    range.Text = $"[{startRange}, {endRange}]";
}));

if (isServer) ansNumber.Invoke(new MethodInvoker(delegate ()
{
    this.Invoke(new MethodInvoker(delegate ()
    {
        label4.Enabled = ansNumber.Enabled = true;
    }));
    ansNumber.Text = rand[3];
}));
else
{
    lastSubmitTime = 100;
    this.Invoke(new MethodInvoker(delegate ()
    {
        btnSubmit.Enabled = btnAutoplayWholeGame.Enabled = btnAutoplaySingleTurn.Enabled = answer.Enabled = true;
        answer.Focus();
        answer.Select();
    }));
}

timeLeft = int.Parse(rand[0]);
this.Invoke(new MethodInvoker(delegate ()
{
    timerCnt.Text = timeLeft.ToString();
    timer.Start();
}));
}
else if (!isServer && data == "@@@Newgame!@@@")
{
    this.Invoke(new MethodInvoker(delegate ()
    {
        btnSubmit.Enabled = btnAutoplayWholeGame.Enabled = btnAutoplaySingleTurn.Enabled = answer.Enabled = true;
        answer.Focus();
        answer.Select();
    }));
}
```

III/ saveHistory (Form này được mở khi Admin kết thúc chương trình)

- Thực hiện truy cập vào website <https://ctxt.io/> sau đó lưu file txt (lịch sử chơi mỗi khi kết thúc chương trình) vào website

```
1 reference
private void webBrowser1_DocumentCompleted(object sender, WebBrowserDocumentCompletedEventArgs e)
{
    if (webBrowser1.Url.ToString() == "https://ctxt.io/")
    {
        HTMLElement editable = getData("div", "className", "editable");
        editable.InnerHtml = "";
        string[] lines = text.Split('\n');
        foreach (string line in lines)
        {
            editable.InnerHtml += $"{line}<br>";
        }
        getData("select", "className", "select").SetAttribute("value", "1d");
        getData("input", "className", "button").InvokeMember("click");
    }
    else
    {
        string url = webBrowser1.Url.ToString();
        MessageBox.Show($"Lịch sử trò chơi được lưu tại:\n{url}");
    }
}
```

```
3 references
private HTMLElement getData(string tag, string att, string attVal)
{
    HTMLElementCollection elements = webBrowser1.Document.GetElementsByTagName(tag);
    foreach (HTMLElement element in elements)
    {
        if (element.GetAttribute(att).Equals(attVal))
        {
            return element;
        }
    }
    return null;
}
```

```
private string text;

1 reference
public saveHistory(string text)
{
    InitializeComponent();
    this.text = text;
    webBrowser1.ScriptErrorsSuppressed = true;
}

1 reference
private void saveHistory_Load(object sender, EventArgs e)
{
    webBrowser1.Navigate("https://ctxt.io/");
}
```

1 reference