

CHƯƠNG 10: LÀM VIỆC VỚI MÀN HÌNH VÀ HỆ THỐNG

10.1. Lớp SystemInformation

Lớp *SystemInformation* là lớp cung cấp thông tin về môi trường hệ thống hiện tại, nằm trong không gian tên *System.Windows.Forms*. Các thuộc tính của lớp *SystemInformation* chứa thông tin về hệ thống như: tài khoản đăng nhập, tên máy tính, tình trạng kết nối mạng, tên miền, ...

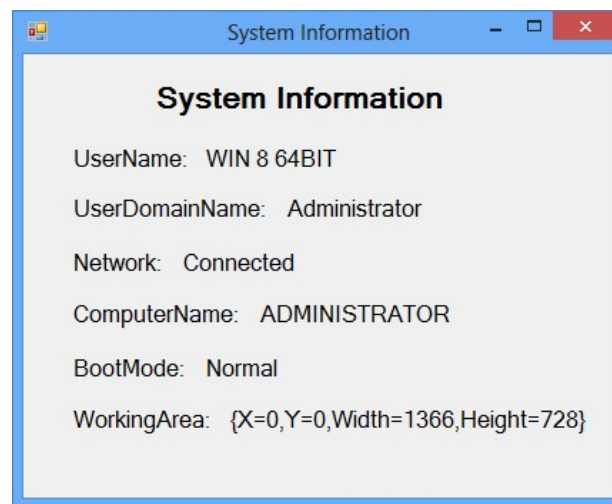
➤ Các thuộc tính thường dùng của lớp *SystemInformation*:

Bảng 10.1: Bảng mô tả các thuộc tính của lớp SystemInformation

Thuộc tính	Mô tả
<i>UserName</i>	Trả về tài khoản người dùng sử dụng trong tiến trình hiện tại.
<i>UserDomainName</i>	Trả về tên miền mà người dùng đã đăng nhập với tài khoản hợp lệ.
<i>Network</i>	Mang hai giá trị True hoặc False. <ul style="list-style-type: none">- Nếu là True: Máy đang trong tình trạng kết nối mạng.- Nếu là False: Máy trong tình trạng không kết nối mạng.
<i>ComputerName</i>	Trả về tên của máy tính đã đặt khi cài đặt hệ điều hành
<i>BootMode</i>	Trả về một trong ba giá trị: Normal, FailSafe (SafeMode), FailSafeWithNetwork (Safe Mode With Network). Các giá trị này là các chế độ khởi động của hệ điều hành Windows.
<i>PowerStatus</i>	Trả về giá trị cho biết tình trạng nguồn điện của máy tính trong tình trạng thế nào. <ul style="list-style-type: none">- <i>PowerStatus.BatteryChargeStatus</i>: Cho biết tình trạng nguồn điện của máy tính thế nào. Nếu là máy tính xách tay sẽ được kết quả: Trong tình trạng sạc pin, pin còn nhiều hay ít điện.- <i>PowerStatus.BatteryFullLifetime</i>: Trả về thời gian (tính bằng giây) của pin có thể sử dụng khi pin đầy. Nếu không xác định được sẽ trả về -1.

	<ul style="list-style-type: none"> - <code>PowerStatus.BatteryLifePercent</code>: Hiển thị % nguồn điện còn lại trong pin. - <code>PowerStatus.BatteryLifeRemaining</code>: Trả về thời gian sử dụng còn lại của pin (tính bằng giây). Trả về -1 nếu sử dụng nguồn điện trực tiếp. - <code>PowerStatus.PowerLineStatus</code>: Trả về giá trị Online nếu máy tính có cắm điện, ngược lại trả về Offline.
<i>WorkingArea</i>	Cho biết thông tin về kích cỡ (Width và Height) vùng làm việc trên màn hình.

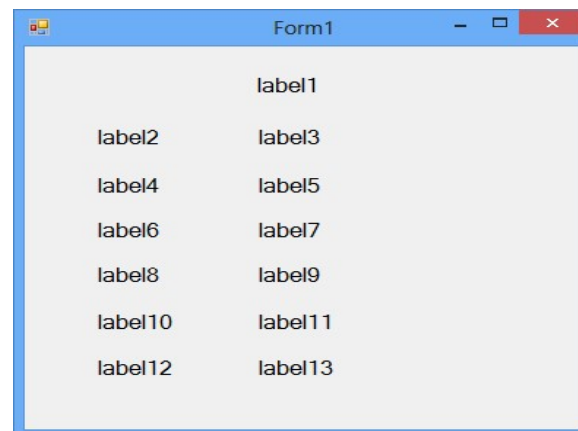
Ví dụ 10.1: Viết chương trình hiển thị thông tin máy tính có giao diện như hình 10.1.



Hình 10.1: Giao diện chương trình hiển thị thông tin máy tính

Hướng dẫn:

Bước 1: Thiết kế giao diện form ban đầu như hình 10.2.



Hình 10.2: Giao diện ban đầu form hiển thị thông tin máy tính

Bước 2: Thiết lập giá trị thuộc tính cho điều khiển trong cửa sổ Properties.

- Form1:
Thuộc tính *Text*: “System Information”
- label1:
Thuộc tính *Text*: “System Information”
Thuộc tính *Font Size*: 16
Thuộc tính *Font Style*: Bold
- label2:
Thuộc tính *Text*: “UserName:”
- label3:
Thuộc tính *Name*: username
- label4:
Thuộc tính *Text*: “UserDomainName:”
- label5:
Thuộc tính *Name*: userdomainname
- label6:
Thuộc tính *Text*: “NetWork:”
- label7:
Thuộc tính *Name*: network
- label8:
Thuộc tính *Text*: “ComputerName:”
- label9:
Thuộc tính *Name*: computername
- label10:
Thuộc tính *Text*: “BootMode:”
- label11:
Thuộc tính *Name*: bootmode
- label12:
Thuộc tính *Text*: “WorkingArea:”
- label13:
Thuộc tính *Name*: workingarea

Bước 3: Viết mã lệnh cho các điều khiển.

- Sự kiện *Load* của form:

```

private void Form1_Load(object sender, EventArgs e)
{
    username.Text = SystemInformation.UserName.ToString();
    userdomainname.Text =
    SystemInformation.UserDomainName.ToString();
    if (SystemInformation.Network == true)
        Network.Text = "Connected";
    else
        Network.Text = "Disconnected";
    computername.Text =
    SystemInformation.ComputerName.ToString();
    bootmode.Text = SystemInformation.BootMode.ToString();
    workingarea.Text =
    SystemInformation.WorkingArea.ToString();
}

```

10.2. Lớp Screen

Lớp *Screen* nằm trong không gian tên *System.Windows.Forms*, lớp *Screen* cung cấp các thông tin về độ phân giải hiện hành của màn hình, tên thiết bị và số lượng Bit trên một Pixel.

➤ Các thuộc tính thường dùng của lớp *Screen*:

Thuộc tính thường được sử dụng của lớp *Screen* là *PrimaryScreen*. Thông qua *PrimaryScreen*, lập trình viên có thể lấy được kích thước vùng hiển thị (Bounds), kích thước vùng làm việc, số bit trên một pixel, tên thiết bị.

Bảng 10.2: Bảng mô tả thuộc tính của lớp Screen

Thuộc tính	Mô tả
<i>PrimaryScreen.Bounds</i>	Lấy kích thước hiển thị trên màn hình
<i>PrimaryScreen.WorkingArea</i>	Lấy kích thước của màn hình làm việc
<i>PrimaryScreen.BitsPerPixel</i>	Lấy số lượng bit trên một Pixel

➤ Các phương thức thường dùng của lớp *Screen*:

Ngoài cách sử dụng thuộc tính *PrimaryScreen* để lấy kích thước màn hình làm việc, lập trình viên cũng có thể sử dụng phương thức *GetWorkingArea*.

Bảng 10.3: Bảng mô tả phương thức của lớp Screen

Thuộc tính	Mô tả
<i>GetWorkingArea</i>	Cung cấp các thông tin về kích thước của màn hình làm

	việc
--	------

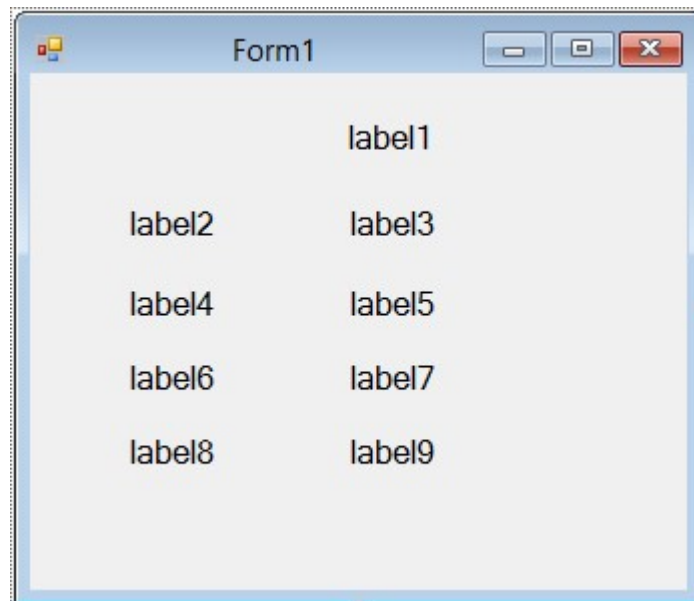
Ví dụ 10.2: Viết chương trình hiển thị các thông tin độ phân giải màn hình, tên thiết bị, số bit trên một pixel như hình 10.3.



Hình 10.3: Giao diện hiển thị thông tin của màn hình

Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu: Thêm các điều khiển Label từ cửa sổ Toolbox vào form như hình 10.4.



Hình 10.4: Giao diện ban đầu form hiển thị thông tin màn hình

Bước 2: Thiết lập giá trị thuộc tính cho điều khiển trong cửa sổ Properties.

- Form1:
Thuộc tính *Text*: "SCREEN"
- label1:
Thuộc tính *Text*: "System SCREEN"
Thuộc tính *Font Size*: 16
Thuộc tính *Font Style*: Bold
- label2:
Thuộc tính *Text*: "WorkingArea:"
- label3:
Thuộc tính *Name*: workingarea
- label4:
Thuộc tính *Text*: "Bits/Pixel:"
- label5:
Thuộc tính *Name*: bits
- label6:
Thuộc tính *Text*: "Bounds:"
- label7:
Thuộc tính *Name*: bounds
- label8:
Thuộc tính *Text*: "DeviceName:"
- label9:
Thuộc tính *Name*: devicename

Bước 3: Viết mã lệnh cho điều khiển

Sự kiện *Load* của form:

```
private void Form1_Load(object sender, EventArgs e)
{
    Rectangle rec = Screen.GetWorkingArea(this);
    workingarea.Text = "Height = " + rec.Height + " Width = " + rec.Width;
    bits.Text = Screen.PrimaryScreen.BitsPerPixel.ToString();
    bounds.Text = "Height = " + Screen.PrimaryScreen.Bounds.Height.ToString() + " Width = " + Screen.PrimaryScreen.Bounds.Width.ToString();
    devicename.Text = Screen.PrimaryScreen.DeviceName.ToString();
}
```

10.3. Lớp SendKeys

Lớp *SendKeys* nằm trong không gian tên *System.Windows.Forms*, lớp *SendKeys* cung cấp các phương thức cho phép làm việc với các phím trên bàn phím. Ba phương thức thường được sử dụng trong lớp *SendKeys* là phương thức: *Send*, *SendWait* và *Flush*.

- Các phương thức thường dùng của lớp *SendKeys*:

Bảng 10.4: Bảng mô tả các phương thức của lớp SendKeys

Phương thức	Mô tả
<i>Send(<Mã phím>)</i>	<p>Phương thức <i>Send</i> cho phép gửi phím đến ứng dụng hiện hành. Để gửi một phím nào đó đến ứng dụng lập trình viên chỉ việc gõ ký tự tương ứng trong phương thức <i>Send</i>.</p> <p>Ví dụ: Muốn gửi phím A đến ứng dụng, lập trình viên phải viết như sau:</p> <pre>SendKeys.Send("A");</pre> <p>Trường hợp muốn gửi tổ hợp phím đến ứng dụng, lập trình viên sẽ gửi một chuỗi ký tự tương ứng trong phương thức <i>Send</i>.</p> <p>Ví dụ: Muốn gửi tổ hợp phím A và B đến ứng dụng, lập trình viên phải viết như sau:</p> <pre>SendKeys.Send("AB");</pre> <p>Một số ký tự đặc biệt như +, %, ^, ~, (,), [,], {, } nếu muốn gửi đến ứng dụng phải đặt trong cặp dấu {}.</p> <p>Ví dụ: Muốn gửi ký tự + đến ứng dụng, lập trình viên phải viết như sau:</p> <pre>SendKeys.Send("{+}");</pre> <p>Các phím đặc biệt không thuộc nhóm ký tự như: <i>BackSpace</i>, <i>Enter</i>, <i>Caps Lock</i>, <i>Delete</i>, ... cũng có thể gửi đến ứng dụng nhưng phải gửi bằng mã tương ứng. Bảng mã được mô tả cụ thể trong bảng 10.4.</p> <p>Gửi các phím chức năng: <i>F1</i>, <i>F2</i>, <i>F3</i>, .. thì gửi bằng mã của phím tương ứng như mô tả trong bảng 10.5.</p>
<i>SendWait(<Mã phím>)</i>	Phương thức <i>SendWait</i> cũng thực hiện công việc gửi

	phím hoặc tổ hợp phím đến ứng dụng hiện hành như phương thức Send. Tuy nhiên, điểm khác biệt là phương thức SendWait sẽ chờ cho công việc được gọi khi gửi phím bằng SendWait đến ứng dụng thực hiện xong thì mới thực hiện các công việc sau đó.
<i>Flush()</i>	Phương thức Flush có chức năng bỏ qua những thực thi đang chờ để thực thi một công việc khác. Cụ thể nếu có một tiến trình đang chạy và muốn ứng dụng ngay lập tức thực thi một công việc khác mà không phải chờ tiến trình làm việc xong thì cần gọi phương thức Flush().

Bảng 10.5: Bảng mã các phím đặc biệt

Phím	Bảng mã
<i>BACKSPACE</i>	{BACKSPACE}, {BS}, {BKSP}
<i>BREAK</i>	{BREAK}
<i>CAPS LOCK</i>	{CAPSLOCK}
<i>DELETE</i>	{DELETE}, {DEL}
<i>END</i>	{END}
<i>ENTER</i>	{ENTER}
<i>ESC</i>	{ESC}
<i>HELP</i>	{HELP}
<i>HOME</i>	{HOME}
<i>INSERT</i>	{INSERT}
<i>NUM LOCK</i>	{NUMLOCK}
<i>PAGE DOWN</i>	{PGDN}
<i>PAGE UP</i>	{PGUP}
<i>DOWN ARROW</i>	{DOWN}
<i>UP ARROW</i>	{UP}
<i>LEFT ARROW</i>	{LEFT}
<i>RIGHT ARROW</i>	{RIGHT}
<i>PRINT SCREEN</i>	{PRTSC}
<i>SCROLL LOCK</i>	{SCROLLLOCK}

<i>TAB</i>	{TAB}
------------	-------

Bảng 10.6: Bảng mã các phím chức năng

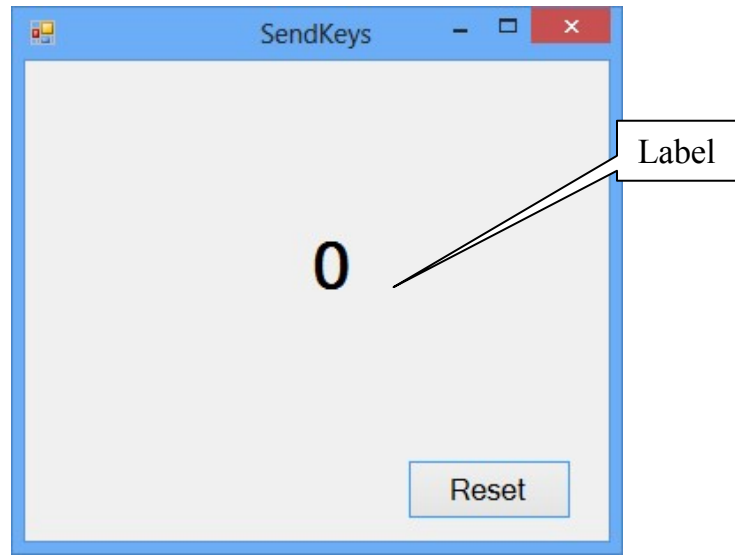
Phím	Bảng mã
<i>F1</i>	{F1}
<i>F2</i>	{F2}
<i>F3</i>	{F3}
<i>F4</i>	{F4}
<i>F5</i>	{F5}
<i>F6</i>	{F6}
<i>F7</i>	{F7}
<i>F8</i>	{F8}
<i>F9</i>	{F9}
<i>F10</i>	{F10}
<i>F11</i>	{F11}
<i>F12</i>	{F12}
+	{ADD}
-	{SUBTRACT}
*	{MULTIPLY}
/	{DIVIDE}

Trường hợp muốn gửi một tổ hợp phím mà trong đó có phím Shift, Ctrl hay Alt lập trình viên có thể sử dụng mã của các phím Shift, Ctrl và Alt tương ứng trong bảng 10.6.

Bảng 10.7: Bảng mã các phím chức năng

Phím	Bảng mã
<i>Ctrl</i>	^
<i>Shift</i>	+
<i>Alt</i>	%

Ví dụ 10.3: Viết chương trình tăng số có giao diện như hình 10.5.



Hình 10.5: Giao diện chương trình tăng số

Yêu cầu:

- Khi nhấn F5, form như hình 10.5 sẽ hiển thị. Mỗi một giây số hiển thị trên Label sẽ tăng 1 đơn vị.
- Khi nhấn Button “Reset” thì số hiển thị trong Label sẽ trở về 0.
- Khi số hiển thị trong Label đạt giá trị 60 thì chương trình sẽ tự động thực hiện việc nhấn Button “Reset” để thiết lập lại giá trị trong Label trở về 0.

Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu: Thêm các điều khiển Label, Timer, Button từ cửa sổ Toolbox vào form như hình 10.6.



Hình 10.6: Giao diện ban đầu chương trình tăng số

Bước 2: Thiết lập giá trị thuộc tính cho điều khiển.

- Form1:
Thuộc tính Text: "Sendkeys"
Thuộc tính AcceptButton: btnReset
- label1:
Thuộc tính Text: "---"
Thuộc tính Name: lblHienthi
Thuộc tính Font Size: 24
Thuộc tính Font Style: Bold
- button1:
Thuộc tính Text: "Reset"
Thuộc tính Name: btnReset
- timer1:
Thuộc tính Interval: 1000
Thuộc tính Enable: True

Bước 3: Viết mã lệnh cho các điều khiển

- Khai báo biến chương trình:

```
int time = 0;
```

- Sự kiện Click của btnReset:

```
private void btnReset_Click(object sender, EventArgs e)
{
    time = 0;
}
```

- Sự kiện Tick của Timer1:

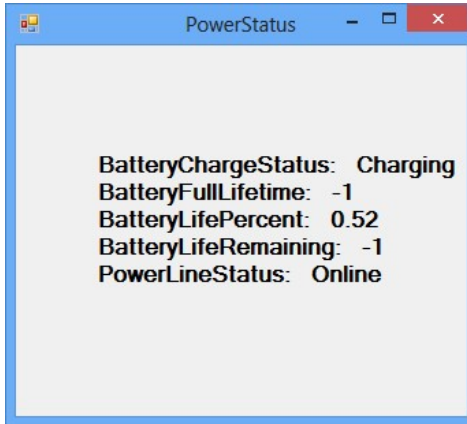
```
private void timer1_Tick(object sender, EventArgs e)
{
    lblHienThi.Text = time.ToString();
    time += 1;
    if (time == 60)
        SendKeys.Send("{Enter}");
}
```

10.4. Lớp PowerStatus

Lớp PowerStatus chứa các thuộc tính cung cấp thông tin về tình trạng nguồn điện đang sử dụng của máy tính. Điểm đặc biệt là lớp PowerStatus không có phương thức khởi tạo, do đó lập trình viên có thể tạo đối tượng lớp PowerStatus và sử dụng thông qua thuộc tính PowerStatus của lớp SystemInformation như sau:

```
PowerStatus pwts;  
pwts = SystemInformation.PowerStatus;  
string power = "";  
power = pwts.BatteryChargeStatus.ToString();  
power += " " + pwts.BatteryFullLifetime.ToString();  
power += " " + pwts.BatteryLifePercent.ToString();  
power += " " + pwts.PowerLineStatus.ToString();
```

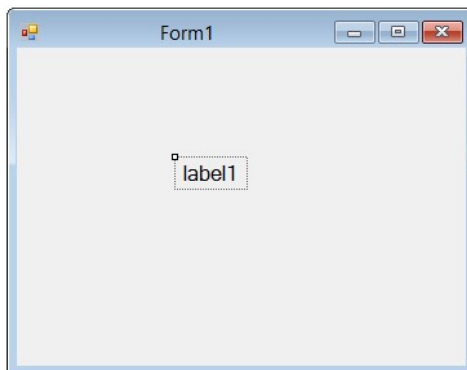
Ví dụ 10.4: Viết chương trình hiển thị tình trạng của pin có giao diện như hình 10.7.



Hình 10.7: Giao diện hiển thị thông tin tình trạng pin

Hướng dẫn:

Bước 1: Thiết kế form có giao diện ban đầu như hình 10.8: Thêm điều khiển Label từ Toolbox vào form.



Hình 10.8: Giao diện ban đầu form hiển thị thông tin tình trạng pin

Bước 2: Thiết lập giá trị thuộc tính trong cửa sổ Properties.

- Form1:

Thuộc tính *Text*: "PowerStatus"

label1:

Thuộc tính *Name*: lblHienthi

Bước 3: Viết mã lệnh cho chương trình

- Sự kiện *Load* của form:

```
private void Form1_Load(object sender, EventArgs e)
{
    PowerStatus pwts;
    pwts = SystemInformation.PowerStatus;
    string power = "BatteryChargeStatus: ";
    power += pwts.BatteryChargeStatus.ToString();
    power += "\nBatteryFullLifetime: " +
    pwts.BatteryFullLifetime.ToString();
    power += "\nBatteryLifePercent: " +
    pwts.BatteryLifePercent.ToString();
    power += "\nBatteryLifeRemaining: " +
    pwts.BatteryLifeRemaining.ToString();
    power += "\nPowerLineStatus: " +
    pwts.PowerLineStatus.ToString();
    lblHienThi.Text = power;
}
```

10.5. Lớp Application

Lớp *Application* chứa các thuộc tính và phương thức liên quan đến ứng dụng như: khởi động ứng dụng, đóng ứng dụng hoặc các thông tin về ứng dụng.

Lưu ý lớp *Application* là dạng lớp cô lập (Sealed) do đó không cho phép các lớp khác thừa kế.

- Các thuộc tính thường dùng của lớp *Application*:

Bảng 10.8: Bảng mô tả các thuộc tính của lớp *Application*

Thuộc tính	Mô tả
<i>CommonAppDataPath</i>	Trả về đường dẫn, đường dẫn này lưu trữ nhưng tập tin để chia sẻ cho người dùng khác sử dụng.
<i>CompanyName</i>	Trả về tên của công ty được khai báo trong thuộc

	<i>AssemblyCompany</i> của tập tin <i>AssemblyInfo.cs</i>
<i>CurrentCulture</i>	Cho phép thiết lập hoặc lấy các thông tin định dạng thời gian, ngày tháng, ngôn ngữ, ... hiện đang sử dụng
<i>CurrentInputLanguage</i>	Cho phép thiết lập hoặc lấy thông tin về ngôn ngữ đang sử dụng
<i>ExecutablePath</i>	Trả về đường dẫn của tập tin thực thi (tập tin có đuôi .exe). Ví dụ: C:\Users\PC_PHUC\Desktop\DuAnDauTien\DuAnDauTien\bin\Debug\DuAnDauTien.exe
<i>ProductName</i>	Trả về tên của ứng dụng được khai báo trong thuộc tính <i>AssemblyProduct</i> của tập tin <i>AssemblyInfo.cs</i>
<i>ProductVersion</i>	Trả về phiên bản của ứng dụng được khai báo trong thuộc tính <i>AssemblyVersion</i> của tập tin <i>AssemblyInfo.cs</i>
<i>StartupPath</i>	Trả về đường dẫn chứa tập tin thực thi, đường dẫn này không chứa tên của tập tin thực thi. Ví dụ: C:\Users\PC_PHUC\Desktop\DuAnDauTien\DuAnDauTien\bin\Debug
<i>UserAppDataPath</i>	Tương tự thuộc tính <i>CommonAppDataPath</i> , <i>UserAppDataPath</i> Trả về đường dẫn, đường dẫn này lưu trữ nhưng tập tin để chia sẻ cho người dùng khác sử dụng.

➤ Các phương thức thường dùng của lớp *Application*:

Bảng 10.9: Bảng mô tả các phương thức của lớp Application

Phương thức	Mô tả
<i>Exit()</i>	Giúp đóng ứng dụng.
<i>ExitThread()</i>	Giúp đóng tiến trình. Một ứng dụng có thể có nhiều tiến trình chạy song song, phương thức <i>ExitThread()</i> giúp đóng tiến trình hiện tại.
<i>DoEvents()</i>	Có chức năng như phương thức <i>Flush</i> của lớp <i>SendKeys</i> . có chức năng bỏ qua những thực thi đang chờ để thực thi một công việc khác. Cụ thể nếu có một

	tiến trình đang chạy và muốn ứng dụng ngay lập tức thực thi một công việc khác mà không phải chờ tiến trình làm việc xong thì cần gọi phương thức <code>DoEvents()</code> .
<i>Restart()</i>	Phương thức <code>Restart()</code> thực hiện hai công việc là đóng ứng dụng lại sau đó thì sẽ khởi động ứng dụng
<i>Run()</i>	Phương thức <code>Run()</code> giúp khởi động một đối tượng trong ứng dụng.

10.6. Lớp Clipboard

Lớp *Clipboard* cung cấp cho người một vùng nhớ tạm để lưu trữ dữ liệu, đồng thời cũng hỗ trợ các phương thức cho phép thực hiện công việc sao chép dữ liệu từ nơi này và dán dữ liệu đến một nơi khác thông qua vùng nhớ tạm mà *Clipboard* cung cấp.

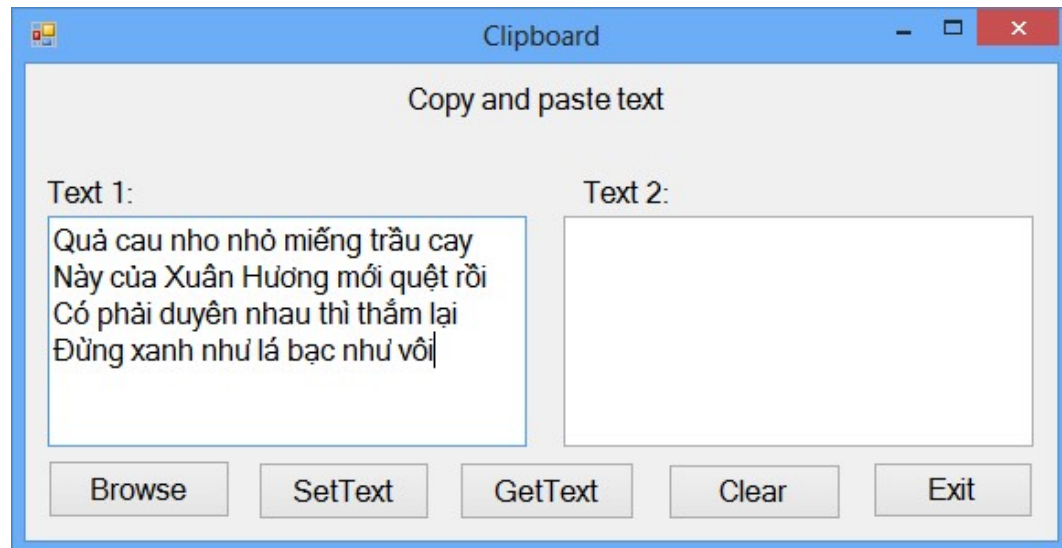
- Các phương thức thường dùng của lớp *Clipboard*:

Bảng 10.10: Bảng mô tả các phương thức của lớp Clipboard

Phương thức	Mô tả
<i>Clear()</i>	Cho phép xóa dữ liệu đang chứa trong <i>Clipboard</i>
<i>SetText(<Chuỗi>)</i>	Lưu <chuỗi> vào trong <i>Clipboard</i>
<i>GetText()</i>	Trả về chuỗi lưu trữ trong <i>Clipboard</i>
<i>ConstraintText()</i>	Trả về một trong hai giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: Trong <i>Clipboard</i> có nội dung - Nếu là False: Trong <i>Clipboard</i> không chứa một chuỗi nào cả.
<i>SetImage(<Image>)</i>	Lưu trữ hình ảnh vào <i>Clipboard</i>
<i>GetImage()</i>	Trả về hình đang lưu trữ trong <i>Clipboard</i>
<i>ConstraintsImage()</i>	Trả về một trong hai giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: Trong <i>Clipboard</i> có chứa hình ảnh - Nếu là False: Trong <i>Clipboard</i> không chứa hình ảnh.
<i>SetData(<Object>)</i>	Lưu trữ đối tượng <Object> vào <i>Clipboard</i>
<i>GetData()</i>	Trả về đối tượng đang lưu trữ trong <i>Clipboard</i>
<i>ContainsData()</i>	Trả về một trong hai giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: Trong <i>Clipboard</i> có chứa đối tượng

	- Nếu là False: Trong <i>Clipboard</i> không chứa đối tượng
--	---

Ví dụ 10.5: Viết chương trình thực hiện công việc sao chép văn bản từ hai TextBox. Chương trình gồm các điều khiển: TextBox, Label, Button, OpenFileDialog. Giao diện chương trình như hình 10.9.



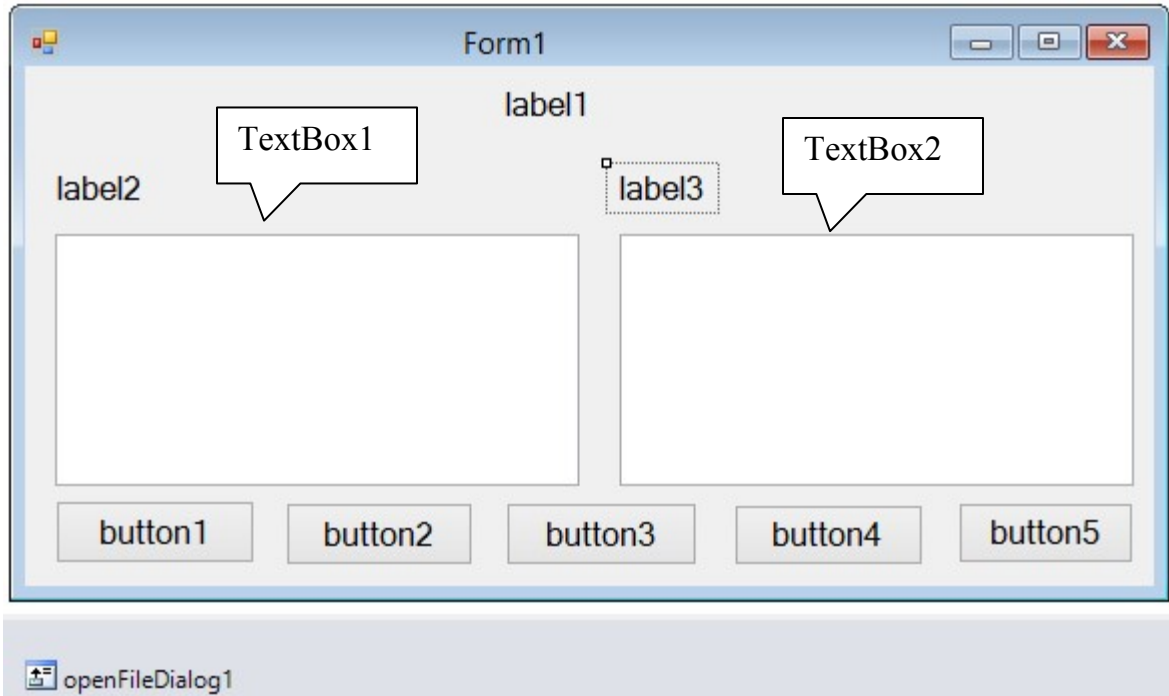
Hình 10.9: Giao diện form sao chép văn bản

Yêu cầu:

- Browse: Giúp chọn tập tin .txt trong hệ thống và hiển thị nội dung tập tin trên TextBox txt1.
- SetText: Sao chép nội dung văn bản trong TextBox txt1 vào Clipboard
- GetText: Chép nội dung văn bản trong Clipboard vào TextBox txt2
- Clear: Xóa dữ liệu lưu trong Clipboard
- Exit: Đóng chương trình.

Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu: Thêm các điều khiển Label, TextBox, Button, OpenFileDialog từ cửa sổ Toolbox vào form như hình 10.10.



Hình 10.10: Giao diện ban đầu form sao chép văn bản

Bước 2: Thiết lập giá trị thuộc tính cho điều khiển

- Form1:
Thuộc tính Text: “Clipboard”
Thuộc tính Size: 580, 299
- label1:
Thuộc tính Text: “Copy and paste Text”
- label2:
Thuộc tính Text: “Text 1:”
- label3:
Thuộc tính Text: “Text 2:”
- TextBox1:
Thuộc tính Multiline: True
Thuộc tính Name: txt1
Thuộc tính Size: 262, 126
- TextBox2:
Thuộc tính Multiline: True
Thuộc tính Name: txt2
Thuộc tính Size: 262, 126

- button1:
Thuộc tính Text: “Browse”
Thuộc tính Name: browse
- button2:
Thuộc tính Text: “SetText”
Thuộc tính Name: settext
- button3:
Thuộc tính Text: “GetText”
Thuộc tính Name: gettext
- button4:
Thuộc tính Text: “Clear”
Thuộc tính Name: clear
- button5:
Thuộc tính Text: “Exit”
Thuộc tính Name: exit
- openFileDialog1:
Thuộc tính Filter: “File Text|*.txt”

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* của Button SetText:

```
private void settext_Click(object sender, EventArgs e)
{
    Clipboard.SetText(txt1.Text);
}
```

- Sự kiện *Click* của Button Browse:

```
private void browse_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        string tentaptin = openFileDialog1.FileName;
        StreamReader rd = new StreamReader(tentaptin);
        txt1.Text = rd.ReadToEnd();
        rd.Close();
    }
}
```

- Sự kiện *Click* của Button GetText:

```
private void gettext_Click(object sender, EventArgs e)
{
    txt2.Text = Clipboard.GetText();
}
```

- Sự kiện *Click* của Button Clear:

```
private void clear_Click(object sender, EventArgs e)
{
    Clipboard.Clear();
}
```







- Sự kiện *Click* của Button Exit:















```
private void exit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```


10.7. Lớp Cursors

Lớp Cursors cung cấp tập các con trỏ có hình dạng khác nhau sử dụng trong ứng dụng Windows Forms như bảng 10.10. Thông thường, khi ứng dụng trong những tình trạng khác nhau thì lập trình viên sẽ thay đổi trạng thái của con trỏ cho phù hợp để người dùng dễ nhận biết.

Bảng 10.11: Bảng mô tả các thuộc tính thường dùng của lớp Clipboard

Thuộc tính lớp Cursors	Mô tả
<i>Cursors.AppStarting</i>	
<i>Cursors.Default</i>	
<i>Cursors.No</i>	
<i>Cursors.SizeAll</i>	
<i>Cursors.SizeNESW</i>	
<i>Cursors.SizeNS</i>	

<i>Cursors.SizeNWSE</i>	
<i>Cursors.SizeWE</i>	
<i>Cursors.WaitCursor</i>	
<i>Cursors.Hand</i>	
<i>Cursors.Help</i>	
<i>Cursors.NoMove2D</i>	
<i>Cursors.NoMoveHoriz</i>	
<i>Cursors.NoMoveVert</i>	
<i>Cursors.PanEast</i>	
<i>Cursors.PanNE</i>	
<i>Cursors.PanNorth</i>	
<i>Cursors.PanNW</i>	
<i>Cursors.PanSE</i>	
<i>Cursors.PanSouth</i>	

Ví dụ 10.6: Trên form sao chép văn bản hình 10. 9. Thiết lập thuộc tính Cursor sao cho khi người dùng rê chuột vào TextBox1 và TextBox2 thì con trỏ sẽ hiển thị ở dạng No .
Hướng dẫn:

- Viết mã lệnh cho sự kiện MouseMove của TextBox1 như sau:

```
private void txt1_MouseMove(object sender, MouseEventArgs e)
{
    Cursor.Current = Cursors.No;
}
```

- Viết mã lệnh cho sự kiện MouseMove của TextBox2 như sau:

```
private void txt2_MouseMove(object sender, MouseEventArgs e)
{
    Cursor.Current = Cursors.No;
}
```

CHƯƠNG 11: BÀI TẬP TỔNG HỢP

Câu 1:

Viết chương trình trắc nghiệm giúp học tiếng anh như sau:

- Khi khởi động chương trình sẽ hiển thị Form “Đăng nhập” như hình 11.1



Hình 11.1: Form đăng nhập

- Nhấn Button “Đăng ký” để đăng ký một tài khoản mới đăng nhập vào chương trình trắc nghiệm. Form “Đăng ký” như hình 11.2.



Hình 11.2: Form đăng ký

- o Nếu người dùng đăng ký tên đăng nhập sai thì sẽ hiện ErrorProvide “Tên đăng nhập đã có”.
- o Nhấn Button “Đăng ký” để hoàn tất việc đăng ký tài khoản, khi đó tên đăng ký sẽ được lưu trong tệp tin có tên TaiKhoan.txt để lưu trữ.
- Tại form “Đăng nhập” như hình 11.1, nếu người dùng đăng nhập với tài khoản admin và mật khẩu tài khoản là admin thì sẽ hiển thị form “Tạo câu hỏi trắc nghiệm” như hình 11.3.

Hình 11.3: Giao diện form tạo câu hỏi

- Người dùng có thể gõ nội dung câu hỏi và nội dung câu trả lời, đồng thời chọn RadioButton tương ứng với đáp án đúng.
- Nhấn Button “Thêm” để thêm câu hỏi vào tập tin có tên Cauhoi.txt.
- Tại form “Đăng nhập” như hình 11.1, nếu người dùng đăng nhập với tài khoản khác admin và tồn tại trong tập tin Taikhoan.txt thì sẽ hiển thị form “Chọn số câu hỏi trắc nghiệm” như hình 11.4. Nếu tài khoản đang nhập không có trong Takhoan.txt thì hiện thông báo “Bạn đăng nhập chưa đúng tài khoản”.

Hình 11.4: Giao diện form lựa chọn số câu hỏi trắc nghiệm và thời gian làm bài

- Khi người dùng đã chọn xong số câu hỏi trắc nghiệm và thời gian làm bài thì nhấn Enter sẽ hiển thị được form làm bài trắc nghiệm như hình 11.5



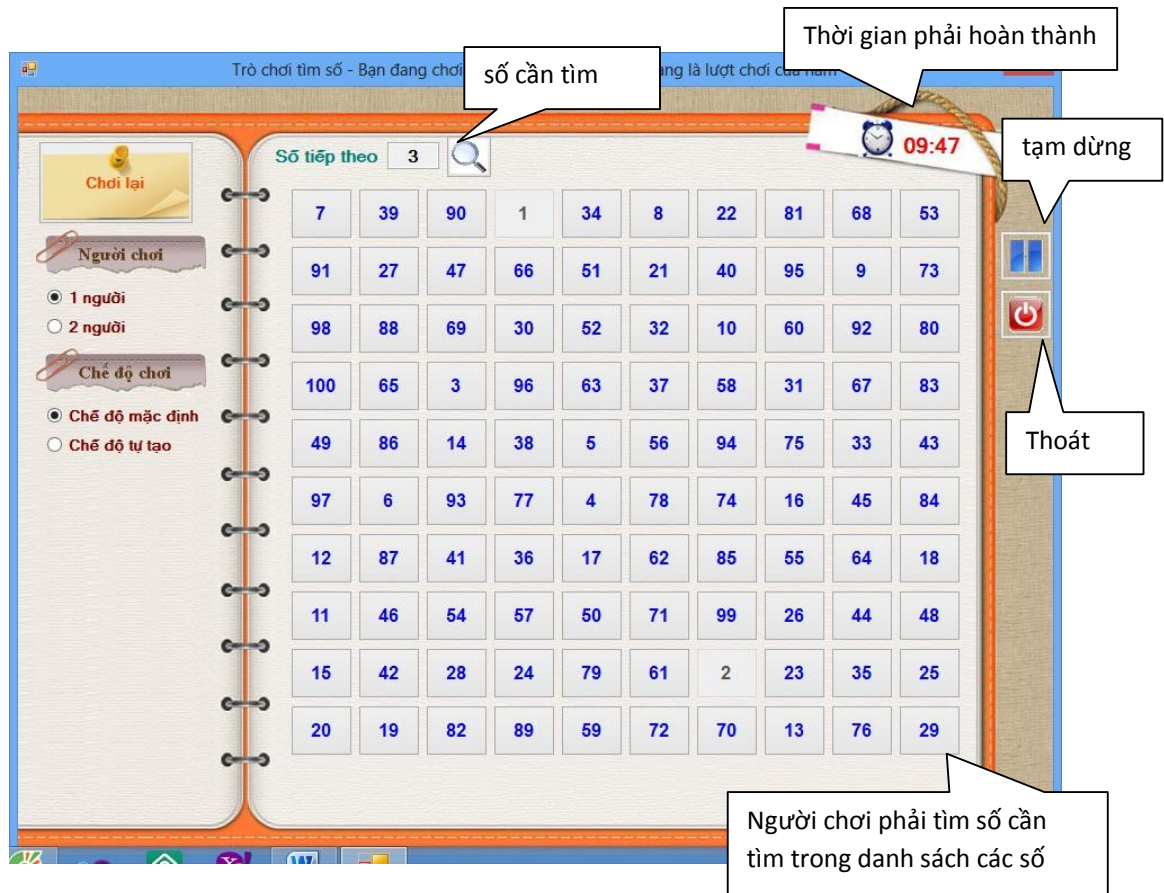
Hình 11.5: Giao diện form làm bài trắc nghiệm tiếng anh

- Bên tay trái giao diện 11.5 sẽ hiển thị các Button, số lượng Button tương ứng với số lượng câu hỏi trắc nghiệm đã chọn. Khi nhấn vào một Button bất kỳ thì sẽ hiển thị lên nội dung câu hỏi, câu hỏi này được lấy từ tập tin Cauhoi.txt.
- Trên giao diện làm bài có thanh ProgressBar sử dụng để mô tả thời gian làm bài mà người dùng chọn. Nếu hết thời gian sẽ tự động nộp bài.
- Người dùng có thể nộp bài trước khi thời gian kết thúc bằng cách nhấn Button “Nộp Bài”. Khi đó sẽ hiển thị giao diện như hình 11.6 để cho biết kết quả làm bài của người dùng.



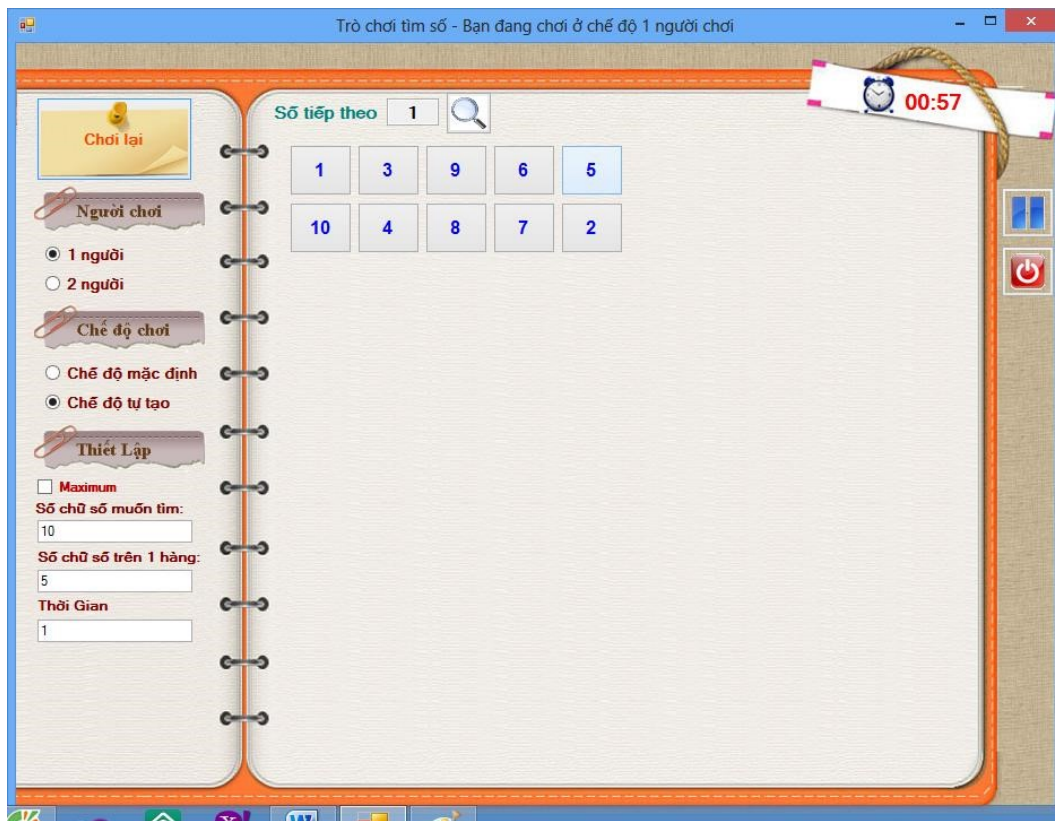
Hình 11.6: Kết quả làm bài của người dùng

Câu 2: Viết chương trình trò chơi tìm số như hình 11.7.



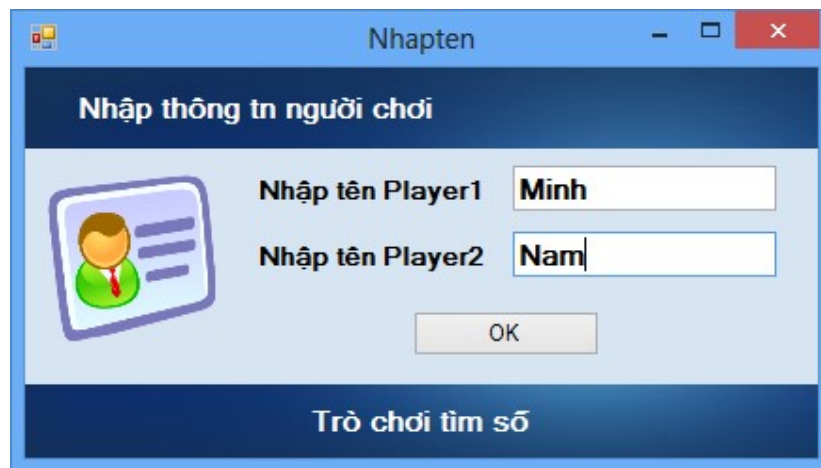
Hình 11.7: Giao diện trò chơi tìm số

- Trong trò chơi tìm số:
 - o Chọn chế độ chơi:
 - Chơi mặc định: khi đó chương trình sẽ hiển thị 100 số ở vị trí ngẫu nhiên bất kỳ. Khi đó người chơi sẽ phải tìm kiếm số cần tìm trong danh sách các số phát sinh ở các vị trí ngẫu nhiên đó. Khi tìm được người chơi sẽ nhấn vào số đó, số đó sẽ lập tức bị chuyển sang chế độ đã được chọn (nghĩa là không còn nhấn được nữa). Thời gian chơi ở chế độ mặc định là 10 phút, nếu hết 10 phút mà không tìm được tất cả 100 số thì người chơi thua cuộc.
 - Chế độ tự tạo: người dùng sẽ chọn số chữ số muốn tìm, số lượng số trên một hàng và thời gian phải hoàn thành việc tìm kiếm. Giao diện như hình 11.8.



Hình 11.8: Giao diện trò chơi chế độ tự chọn

- Chọn số người chơi: Nếu chọn hai người chơi thì sẽ hiển thị form như hình 11.9 để 2 người chơi nhập tên vào.



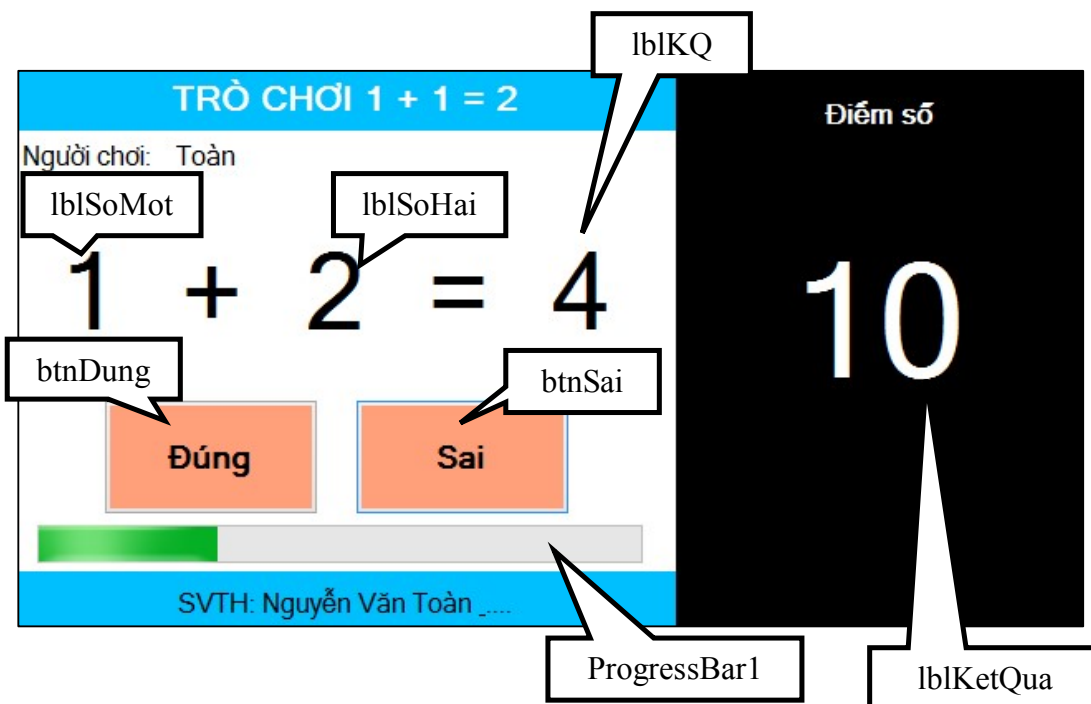
Hình 11.9: Nhập tên của hai người chơi

- Tuân tự người chơi thứ nhất sẽ tìm số trước, thời gian để tìm ra một số là 5 giây, nếu hết 5 giây mà người chơi thứ nhất chưa tìm ra sẽ chuyển lượt cho người chơi thứ hai.
- Với mỗi lần tìm được số thì điểm số của người chơi sẽ tăng lên 1 như hình 11.10



Hình 11.10: Chương trình tìm số hai người chơi

Câu 3: Viết chương trình trò chơi $1 + 1 = 2$ có giao diện như hình 11.11.



Hình 11.11: Giao diện trò chơi $1+1 = 2$

Yêu cầu:

- Chương trình sẽ phát sinh ngẫu nhiên các số nguyên trong Label: lblSoMot, lblSoHai, lblKQ.
- Người chơi sẽ xem kết quả phát sinh ở biểu thức và chọn biểu thức đúng hay sai.
 - Nếu đúng thì dùng chuột chọn nút “Đúng” trên giao diện chương trình hoặc nhấn nút “Left Arrow” trên bàn phím.
 - Nếu sai thì dùng chuột chọn nút “Sai” trên giao diện chương trình hoặc nhấn nút “Right Arrow” trên bàn phím.
- Mỗi lần chọn đúng thì sẽ được cộng một điểm vào lblKetQua và tiếp tục phát sinh ngẫu nhiên 1 biểu thức mới. Nếu chọn sai sẽ bắt đầu chơi lại trò chơi.
- Thời gian để tính toán và chọn đúng hay sai một biểu thức là 4 giây, được thể hiện ở thanh ProgressBar.