1 Nội dung

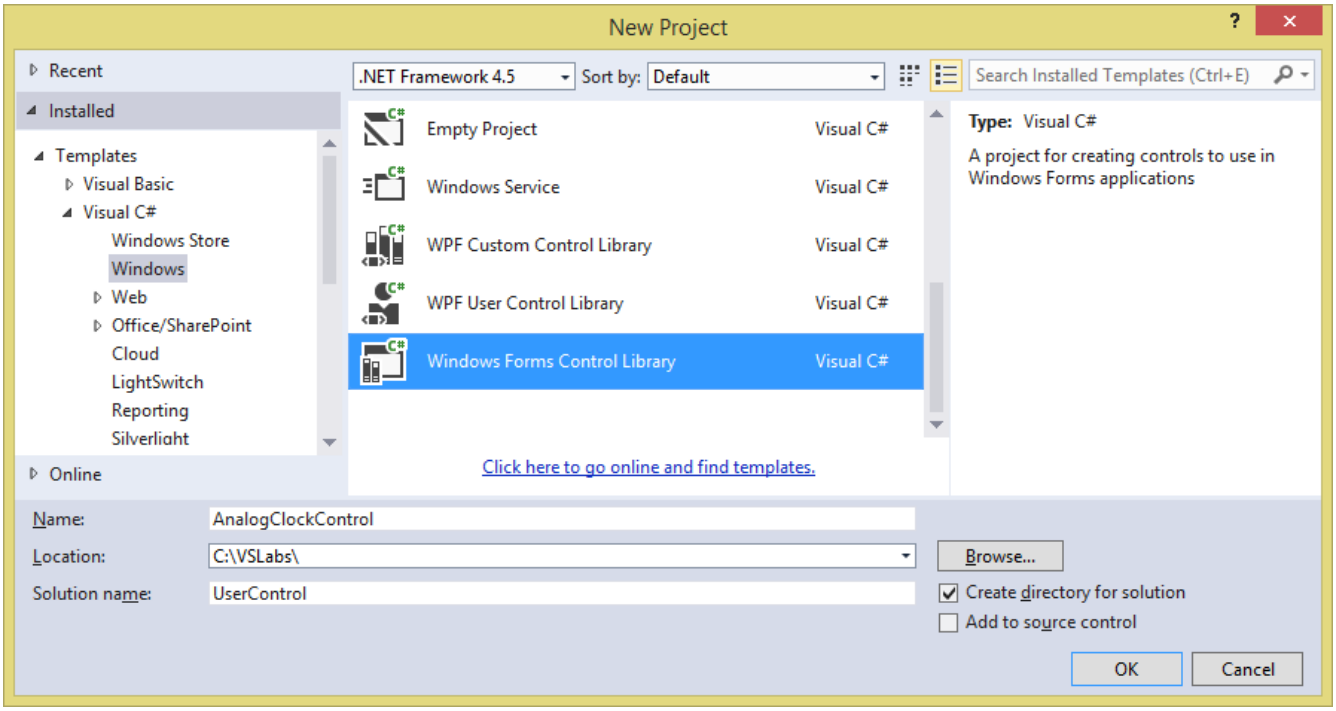
- Làm quen với User Control
- Làm quen với TreeView, ListView

2 Bài tập UserControl có hướng dẫn

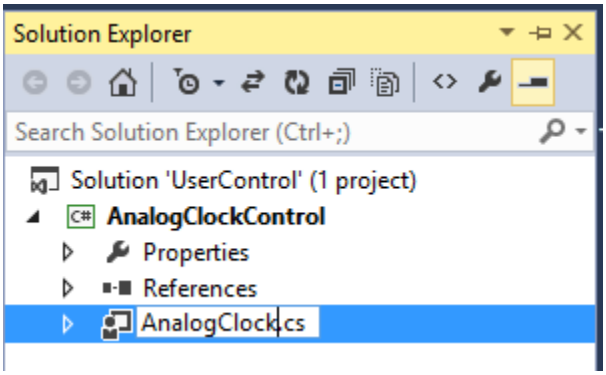
Bài này dựa vào bài viết ANALOG CLOCK CONTROL IN C# của tác giả Syed Mehroz Alam

2.1 Tạo Project & Class

Tạo project dạng Windows Form Control Library, đặt tên **AnalogClockControl**.



Sửa tên lại thành **AnalogClock**.



Viết code cho class AnalogClok

2.2 Phần Field & Property

```
public partial class AnalogClock: UserControl
{
    #region Field
    const float PI = (float)Math.PI;

    DateTime dateTime;

    float fRadius;
    float fCenterX;
    float fCenterY;
    float fCenterCircleRadius;

    float fHourLength;
    float fMinLength;
    float fSecLength;

    float fHourThickness;
    float fMinThickness;
    float fSecThickness;

    float fTicksThickness = 1;

    Color hrColor = Color.DarkMagenta;
    Color minColor = Color.Green;
    Color secColor = Color.Red;
    Color circleColor = Color.Red;
    Color ticksColor = Color.Black;
    #endregion
    #region Property
    public Color HourHandColor
    {
        get { return this.hrColor; }
        set { this.hrColor = value; }
    }
    public Color MinuteHandColor
    {
        get { return this.minColor; }
        set { this.minColor = value; }
    }
    public Color SecondHandColor
    {
        get { return this.secColor; }
        set
        {
            this.secColor = value;
            this.circleColor = value;
```

```
        }
    }
    public Color TicksColor
    {
        get { return this.ticksColor; }
        set { this.ticksColor = value; }
    }
    public bool Draw1MinuteTicks {get;set;}
    public bool Draw5MinuteTicks { get; set; }
    #endregion
```

2.3 Thêm control timer để lấy thời gian

```
    private void timer1_Tick(object sender, EventArgs e)
    {
        this.dateTime = DateTime.Now;
        this.Refresh();
    }
```

2.4 Viết các sự kiện Load(), Resize()

```
    private void AnalogClock_Load(object sender, EventArgs e)
    {
        dateTime = DateTime.Now;
        this.AnalogClock_Resize(sender, e);
    }

    private void AnalogClock_Resize(object sender, EventArgs e)
    {
        this.Width = this.Height;
        this.fRadius = this.Height / 2;
        this.fCenterX = this.ClientSize.Width / 2;
        this.fCenterY = this.ClientSize.Height / 2;
        this.fHourLength = (float)this.Height / 3 / 1.65F;
        this.fMinLength = (float)this.Height / 3 / 1.20F;
        this.fSecLength = (float)this.Height / 3 / 1.15F;
        this.fHourThickness = (float)this.Height / 100;
        this.fMinThickness = (float)this.Height / 150;
        this.fSecThickness = (float)this.Height / 200;
        this.fCenterCircleRadius = this.Height / 50;
        this.Refresh();
    }
```

2.5 Các method vẽ kim đồng hồ

```
    private void DrawPolygon(float fThickness, float fLength, Color color,
float fRadians, PaintEventArgs e)
    {
        PointF A = new PointF
        {
```

```
                X = (float)(fCenterX + fThickness * 2 * Math.Sin(fRadians + PI /
2)),
                Y = (float)(fCenterY - fThickness * 2 * Math.Cos(fRadians + PI /
2))
            };
            PointF B = new PointF
            {
                X = (float)(fCenterX + fThickness * 2 * Math.Sin(fRadians - PI /
2)),
                Y = (float)(fCenterY - fThickness * 2 * Math.Cos(fRadians - PI /
2))
            };
            PointF C = new PointF
            {
                X = (float)(fCenterX + fLength * Math.Sin(fRadians)),
                Y = (float)(fCenterY - fLength * Math.Cos(fRadians))
            };
            PointF D = new PointF{
                X = (float)(fCenterX - fThickness * 4 * Math.Sin(fRadians)),
                Y = (float)(fCenterY + fThickness * 4 * Math.Cos(fRadians))
            };
            PointF[] points = { A, D, B, C };
            e.Graphics.FillPolygon(new SolidBrush(color), points);
        }
```

2.6 Sự kiện Paint()

```
        private void AnalogClock_Paint(object sender, PaintEventArgs e)
        {
            float fRadHr = (dateTime.Hour % 12 + dateTime.Minute / 60F) * 30 * PI
/ 180;
            float fRadMin = (dateTime.Minute) * 6 * PI / 180;
            float fRadSec = (dateTime.Second) * 6 * PI / 180;
            DrawPolygon(this.fHourThickness, this.fHourLength, hrColor, fRadHr,
e);
            DrawPolygon(this.fMinThickness,  this.fMinLength, minColor, fRadMin,
e);
            DrawLine(this.fSecThickness, this.fSecLength, secColor, fRadSec, e);

            for(int i = 0;i < 60; i++)
            {
                if (this.Draw5MinuteTicks && i%5 == 0 )
                // Draw 5 minute ticks
                {
                    e.Graphics.DrawLine( new Pen( ticksColor, fTicksThickness ),
                     fCenterX + (float)( this.fRadius/1.50F*Math.Sin(i*6*PI/180)
),
                     fCenterY - (float)( this.fRadius/1.50F*Math.Cos(i*6*PI/180)
),
```

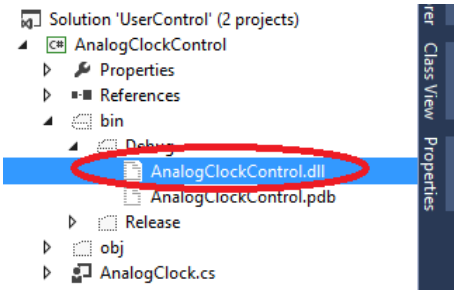
```
                    fCenterX + (float)( this.fRadius/1.65F*Math.Sin(i*6*PI/180)
),
                    fCenterY - (float)( this.fRadius/1.65F*Math.Cos(i*6*PI/180))
);
                }
                else if (this.Draw1MinuteTicks) // draw 1 minute ticks
                {
                    e.Graphics.DrawLine( new Pen( ticksColor, fTicksThickness ),
                       fCenterX + (float) (
this.fRadius/1.50F*Math.Sin(i*6*PI/180) ),
                       fCenterY - (float) (
this.fRadius/1.50F*Math.Cos(i*6*PI/180) ),
                       fCenterX +
(float)(this.fRadius/1.55F*Math.Sin(i*6*PI/180)),
                       fCenterY -
(float)(this.fRadius/1.55F*Math.Cos(i*6*PI/180)));
                }
             }
            //draw circle at center
            e.Graphics.FillEllipse(new SolidBrush(circleColor),
                fCenterX - fCenterCircleRadius / 2,
                fCenterY - fCenterCircleRadius / 2,
                fCenterCircleRadius, fCenterCircleRadius);
        }
```

2.7 Viết method start() và stop()

```
        public void Start()
        {
            timer1.Enabled = true;
            this.Refresh();
        }
        public void Stop()
        {
            timer1.Enabled = false;
        }
```

2.8 Biên dịch project

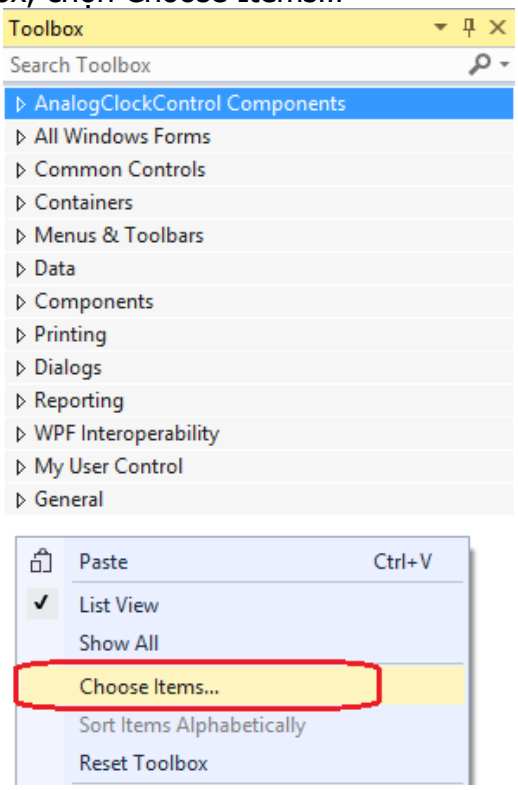
Kết quả của quá trình biên dịch, ta được file dll.



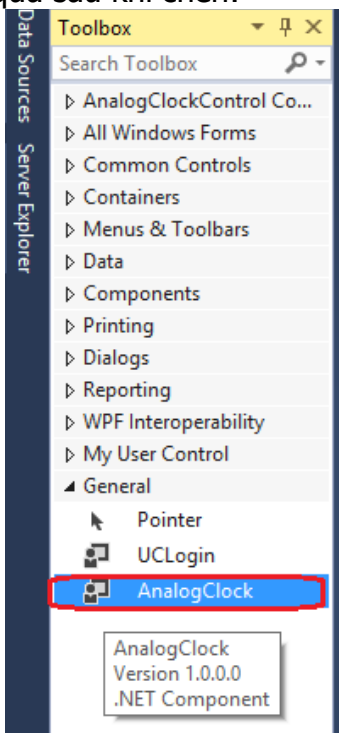
2.9 Sử dụng User Control

2.9.1 Chèn user control vào thanh Toolbox

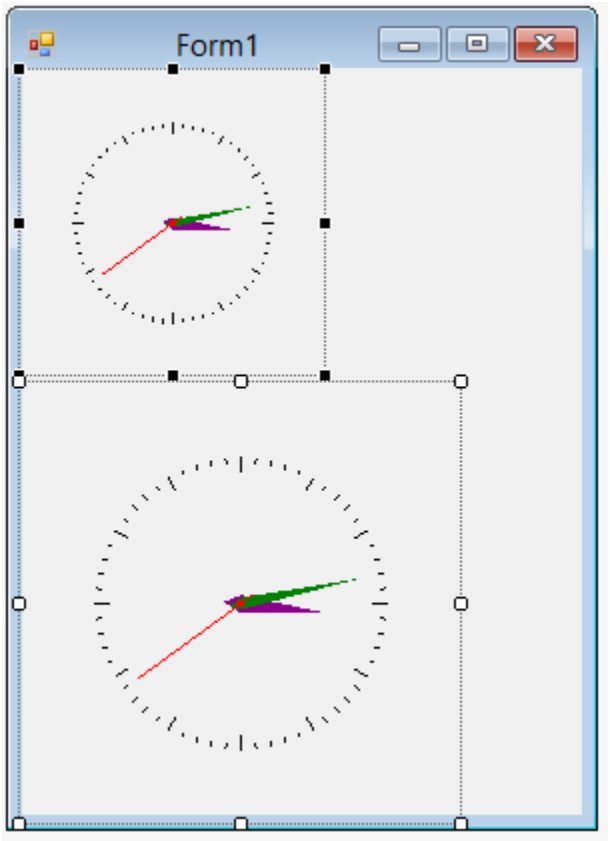
Chuột phải trên thanh Toolbox, chọn Choose Items…



Sau đó chọn file dll cần chèn. Kết quả sau khi chèn:

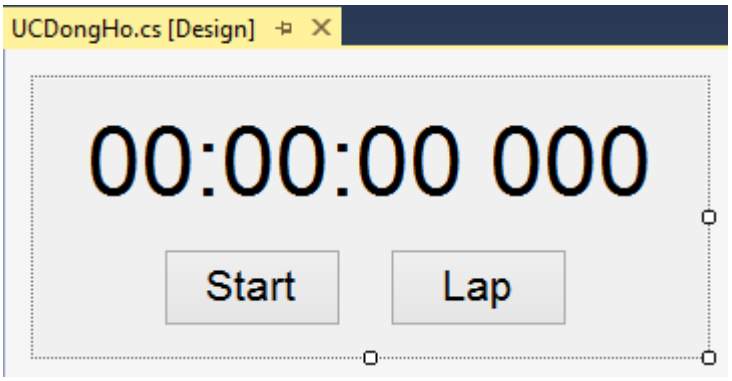


2.9.2 Kéo vào form và sử dụng



3 Bài tập UserControl Đồng hồ bấm giờ

3.1 Mô tả



- Bấm **Start** thì cho đồng hồ chạy, đồng thời đổi Text thành *Stop.*
- Bấm **Stop** thì dừng đồng hồ và đổi Text thành *Start*.
- Bấm **Lap** (ghi giờ) thực hiện ghi xuống file số lần cho trước.

3.2 Hướng dẫn code

Khai báo các biến cần thiết:

```
DateTime timestart;
int LapStep = 0;
const int LapCount = 5;
```

Xử lý cho nút Start/Stop:

```
private void btnStartSTop_Click(object sender, EventArgs e)
{
    if(timer1.Enabled)//true - đang chạy
    {
        timer1.Stop();
        btnStartSTop.Text = "Start";
    }
    else//đang đứng
    {
        timestart = DateTime.Now;
        timer1.Start();
        btnStartSTop.Text = "Stop";
    }
}
```

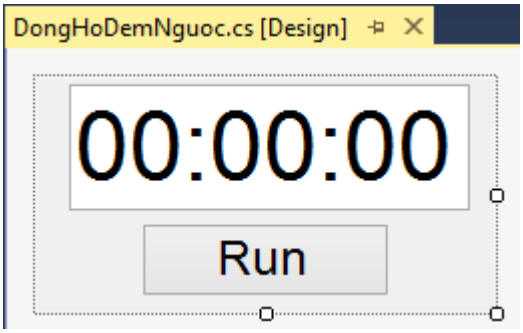
Xử lý cho sự kiện tick:

```
private void timer1_Tick(object sender, EventArgs e)
{
    TimeSpan timetick = DateTime.Now - timestart;
    lblDongHo.Text = string.Format("{0}:{1}:{2} {3}",
      timetick.Hours, timetick.Minutes, timetick.Seconds,
      timetick.Milliseconds);
}
```

Xử lý cho button **Lap**:

```
private void btnLap_Click(object sender, EventArgs e)
{
    lapstep++;
    StreamWriter sw = new StreamWriter("Data.txt", true);//ghi thêm
    sw.WriteLine(string.Format("Lap {0}: {1}",lapstep, lblDongHo.Text));
    if(lapstep == LapCount)
    {
        sw.WriteLine("FINISH");
        lapstep = 0;
        timer1.Stop();
        btnStartSTop.Text = "Start";
    }
    sw.Close();
}
```

4 Bài tập UserControl đồng hồ đếm ngược



Thiết kế và coding đồng hồ đếm ngược cho phép người dùng nhập giá trị vào (nhớ kiểm tra hợp lệ), khi bấm "**Run**" thì tiến hành đếm ngược.

5 Form có hình dáng bất kỳ

Nguồn tham khảo: http://quanghd.code5s.com/dotnet/tao_form_co_hinh_dang_bat_ky.html

Form và hầu hết các control (điều khiển) trong C# đều có thuộc tính ***Region***. Thuộc tính này quy định một vùng để Form hoặc control hiển thị. Để tạo ra form hay control có hình dạng bất kỳ, chúng ta phải tạo ra một vùng màn hình theo hình dạng mong muốn và truyền vùng màn hình này vào cho thuộc tính ***Region***.

Chúng ta có thể tự vẽ một region hoặc tạo nó từ một ảnh bitmap. Trong bài hướng dẫn này, tôi sẽ trình bày cách tạo một region từ một ảnh bitmap và "ép" nó vào form.


```
public static void CreateRegion(Control ct, Bitmap bm)
        {
            if (ct == null || bm == null)
            {
                return;
            }
            // Thiết lập kích thước của control bằng với
            // kích thước của bitmap
            ct.Width = bm.Width;
            ct.Height = bm.Height;
            // kiểm tra xem ct có phải là Form
            if (ct is System.Windows.Forms.Form)
```

```
        {
            Form fm = (Form)ct;
            // Tăng kích thước form để dành chỗ cho đường viền (nếu có)
            fm.Width += 15;
            fm.Height += 35;
            // thiết lập form trở thành không có border
            fm.FormBorderStyle = FormBorderStyle.None;
            // đưa bitmap trở thành background image
            fm.BackgroundImage = bm;
            // tính toán graphics path dựa trên bitmap
            GraphicsPath path = CalculateGraphicsPath(bm);
            // thay đổi thuộc tính Region của form
            fm.Region = new Region(path);
        }
        else if (ct is System.Windows.Forms.Button)
        {
            Button bt = (Button)ct;
            bt.Text = ""; // không hiển thị label của button
            bt.Cursor = Cursors.Hand;
            // thiết lập background image cho button
            bt.BackgroundImage = bm;
            GraphicsPath path = CalculateGraphicsPath(bm);
            bt.Region = new Region(path);
        }
    }

    private static GraphicsPath CalculateGraphicsPath(Bitmap bm)
    {
        GraphicsPath path = new GraphicsPath();
        // dùng màu của góc trên trái của bm là màu transparent
        Color transparentColor = bm.GetPixel(0, 0);
        // duyệt bm theo cột
        for (int row = 0; row < bm.Height; row++)
        {
            // duyệt bm theo hàng
            for (int col = 0; col < bm.Width; col++)
            {
                if (bm.GetPixel(col, row) != transparentColor)
                {
                    int i = col;
                    // đếm số pixel trong hàng khác màu với transparentColor
                    for (i = col + 1; i < bm.Width; i++)
                    {
                        if (bm.GetPixel(i, row) == transparentColor)
                            break;
                    }
                    path.AddRectangle(new Rectangle(col, row, i - col, 1));
                    col = i;
                }
            }
```

```
            }
            return path;
        }
```

Gắn các hàm vào trong FormInit để quét ảnh và gắn hình dạng form theo nó:

```
public StopWatch()
{
    InitializeComponent();
    //Giữa màn hình
    this.StartPosition = FormStartPosition.CenterScreen;
    //Gắn hình dạng form
    Bitmap bmForm = new Bitmap("myclock.png");
    CreateRegion(this, bmForm);
    //Gắn hình cho nút Stop
    Bitmap bmExit = new Bitmap("Stop_X.png");
    CreateRegion(btnExit, bmExit);
}
```

Mở rộng:

Thiết kế dạng UserControl cho phép bắt phím:

- phím **P**: Tạm dừng hay tiếp tục hiển thị đồng hồ
- phím **mũi tên lên** sẽ Reset lại đồng hồ về 00:00:00
- phím **mũi tên xuống** sẽ lưu lại "Lap" (Lưu lại thời gian mà người dùng vừa bấm phím mũi tên xuống, đồng thời đồng hồ vẫn chạy tiếp tục, nếu người dùng bấm tiếp sẽ tiếp tục lưu thêm 1 "Lap" cứ tiếp tục như vậy đồng hồ lưu tối đa 10 "Lap"

---Hết---