

## Chương 7

# Tương tác với người dùng trong ứng dụng C#

### 7.0 Dẫn nhập

### 7.1 Tổng quát về tương tác giữa người dùng & chương trình

### 7.2 Đối tượng vẽ và cơ chế vẽ nội dung

### 7.3 Xuất chuỗi văn bản

### 7.4 Xuất ảnh bitmap

### 7.5 Xuất hình đồ họa toán học

### 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

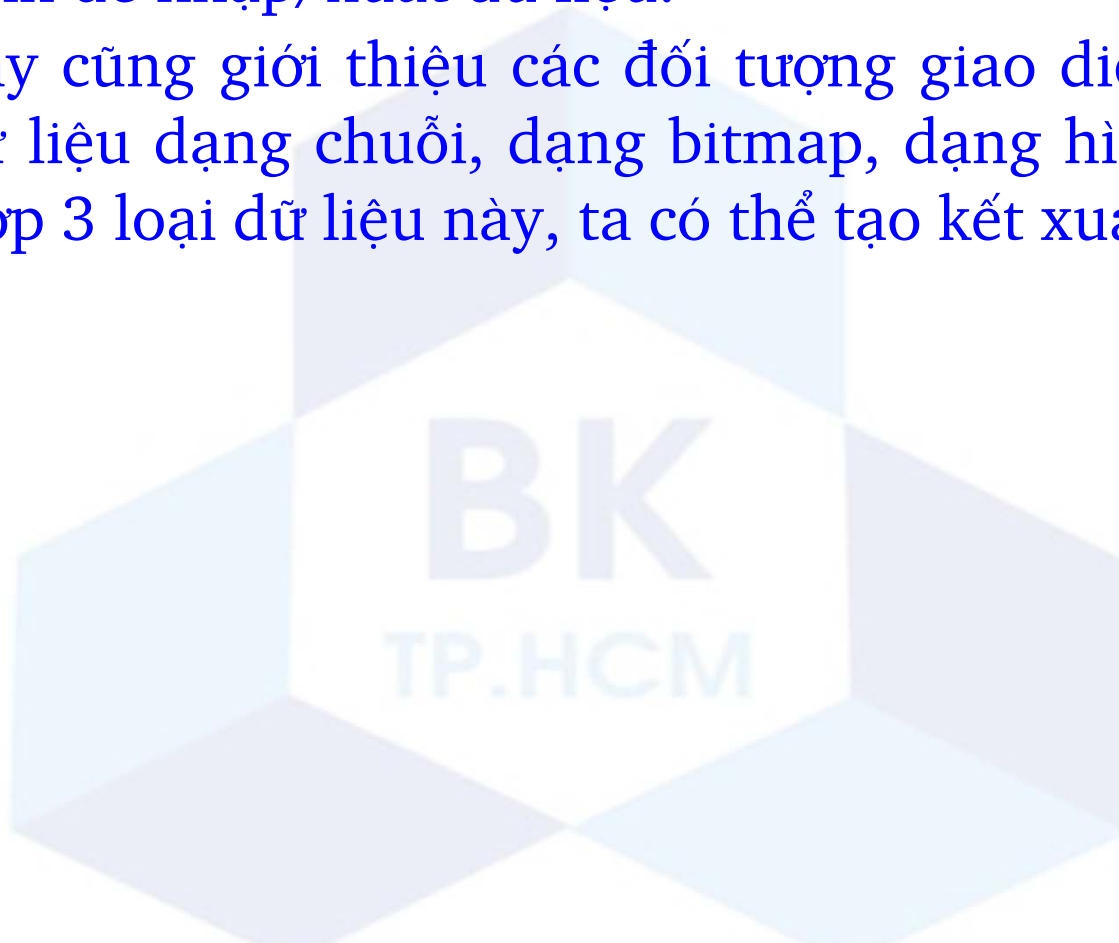
### 7.7 Xây dựng đối tượng giao diện có hình dạng tùy ý

### 7.8 Kết chương



## 7.0 Dẫn nhập

- ❑ Chương này giới thiệu cách thức tương tác giữa người dùng và chương trình để nhập/xuất dữ liệu.
- ❑ Chương này cũng giới thiệu các đối tượng giao diện cùng các tác vụ xuất dữ liệu dạng chuỗi, dạng bitmap, dạng hình đồ họa toán học. Kết hợp 3 loại dữ liệu này, ta có thể tạo kết xuất bất kỳ.



## 7.1 Tổng quát về tương tác người dùng/chương trình

- ❑ Trong lúc chương trình chạy, nó thường phải tương tác với người dùng. Sự tương tác gồm 2 hoạt động chính :
  - chờ nhận dữ liệu do người dùng cung cấp hay chờ nhận lệnh của người dùng để thực thi 1 chức năng nào đó.
  - hiển thị thông báo và/hoặc kết quả tính toán ra màn hình/máy in để người dùng biết và sử dụng.
- ❑ Sự tương tác giữa người dùng và máy tính được thực hiện thông qua các thiết bị nhập/xuất (thiết bị I/O - input/output) như bàn phím/chuột để nhập dữ liệu hay lệnh, màn hình/máy in để xuất kết quả hay thông báo...
- ❑ Hiện có hàng trăm hãng chế tạo thiết bị I/O, mỗi hãng chế tạo rất nhiều model của cùng 1 thiết bị (td. hãng HP chế rất nhiều model máy in phun mực, máy in laser,...). Mỗi model thiết bị của từng hãng có những tính chất vật lý riêng và khác với các model khác.



## 7.1 Tổng quát về tương tác người dùng/chương trình

- ❑ Để giúp người lập trình truy xuất các thiết bị I/O dễ dàng, độc lập với tính chất phần cứng của thiết bị, HĐH Windows và VC# đã che dấu mọi tính chất phần cứng của các thiết bị và cung cấp cho người lập trình 1 giao tiếp sử dụng duy nhất, độc lập với thiết bị : người dùng sẽ tương tác với chương trình thông qua các đối tượng giao diện :
  - người dùng ra lệnh bằng cách kích hoạt sự kiện xác định của 1 đối tượng giao diện. Thí dụ click chuột vào button "Bắt đầu giải" để ra lệnh chương trình giải dùm phương trình bậc 2 có 3 tham số a, b, c đã nhập.
  - nhập giá trị đúng/sai thông qua chọn/cấm chọn RadioButton hay checkbox.



## 7.1 Tổng quát về tương tác người dùng/chương trình

- nhập chọn lựa 1/n thông qua chọn RadioButton tương ứng trong GroupBox, hay chọn mục tương ứng trong Listbox, ComboBox.
  - nhập số nguyên, số thực, chuỗi thông qua TextBox...
  - xuất kết quả ra màn hình thông qua các đối tượng RadioButton, Checkbox, TextBox, ListBox, ComboBox, TreeView...
- Trong trường hợp cần xuất kết quả phức tạp bất kỳ, ta xem nó như là tập hợp nhiều chuỗi văn bản, nhiều phần tử ảnh bitmap, nhiều phần tử đồ họa toán học như hình chữ nhật, hình tròn,... Xuất kết quả phức tạp là quá trình lặp vẽ từng phần tử cấu thành kết quả phức tạp.



## 7.2 Đối tượng vẽ và cơ chế vẽ nội dung

- ❑ Các đối tượng Form, PictureBox, Printer cho phép vẽ nội dung bất kỳ lên chúng.
- ❑ Mỗi lần cần vẽ lại nội dung của đối tượng (lúc bắt đầu hiển thị, lúc thay đổi vị trí, kích thước của đối tượng), máy sẽ tạo sự kiện Paint, sự kiện này sẽ kích hoạt hàm xử lý tương ứng của đối tượng. Như vậy, nếu muốn vẽ thông tin chi tiết lên đối tượng, người lập trình phải định nghĩa hàm xử lý sự kiện Paint của đối tượng và hiện thực thuật giải để vẽ chi tiết thông tin lên đối tượng.
- ❑ Khi cần thiết, người lập trình có thể gọi tác vụ Refresh() của đối tượng để nhờ máy tạo dùm sự kiện Paint hầu vẽ lại đối tượng.



## 7.2 Đối tượng vẽ và cơ chế vẽ nội dung

- ❑ Template của hàm xử lý sự kiện Paint của đối tượng như sau :

```
private void Form1_Paint(object sender, PaintEventArgs e) {  
    //xác định đối tượng mục tiêu  
    Control control = (Control)sender;  
    //thay đổi kích thước, vị trí nếu cần  
    //xác định đối tượng graphics (đối tượng vẽ) của đối tượng  
    Graphics g = e.Graphics;  
    //gọi các tác vụ vẽ của đối tượng vẽ như DrawImage,  
    //DrawString, DrawLine,... để xuất các thông tin bitmap,  
    //chuỗi văn bản, hình đồ họa toán học.  
}
```



## 7.3 Xuất chuỗi văn bản

- ❑ Đối tượng vẽ (graphics) cung cấp khoảng 70 tác vụ vẽ khác nhau, mỗi tác vụ gồm nhiều biến thể (overloaded) để giúp ta điều khiển vẽ nội dung dễ dàng, tiện lợi. Ở đây chúng ta chỉ giới thiệu 1 số tác vụ phổ dụng.
- ❑ Tác vụ DrawString cho phép xuất chuỗi văn bản theo định dạng xác định. Nó có nhiều biến thể, biến thể khá mạnh và dùng phổ biến có đặc tả như sau :

```
public void DrawString (  
    string s,           //chuỗi cần xuất  
    Font font,          //các tính chất font chữ cần dùng để vẽ  
    Brush brush,        //màu vẽ chuỗi  
    float x,            //tọa độ x của điểm canh lề chuỗi  
    float y,            //tọa độ y của điểm canh lề chuỗi  
    StringFormat format); //thuộc tính điều khiển vẽ chuỗi
```





## 7.3 Xuất chuỗi văn bản

- ❑ Thí dụ ta có biến `now` miêu tả thông tin thời điểm hiện hành, ta có thể viết đoạn code sau để rút trích thông tin từ biến `now` và xuất thông tin giờ/phút/giây ra giữa form ứng dụng :

```
//tạo chuỗi miêu tả giờ/phút/giây hiện hành
```

```
String buf = "" + now.Hour + ":" + now.Minute + ":" + now.Second;
```

```
//tạo đối tượng font chữ cần dùng
```

```
Font myFont = new Font("Helvetica", 11);
```

```
//tạo biến miêu tả chế độ canh giữa khi xuất chuỗi
```

```
StringFormat format1 = new StringFormat(StringFormatFlags.NoClip);
```

```
format1.Alignment = StringAlignment.Center;
```

```
//xuất chuỗi miêu tả giờ/phút/giây
```

```
g.DrawString(buf, myFont, System.Drawing.Brushes.Blue,  
xo, rec.Height - 35, format1);
```



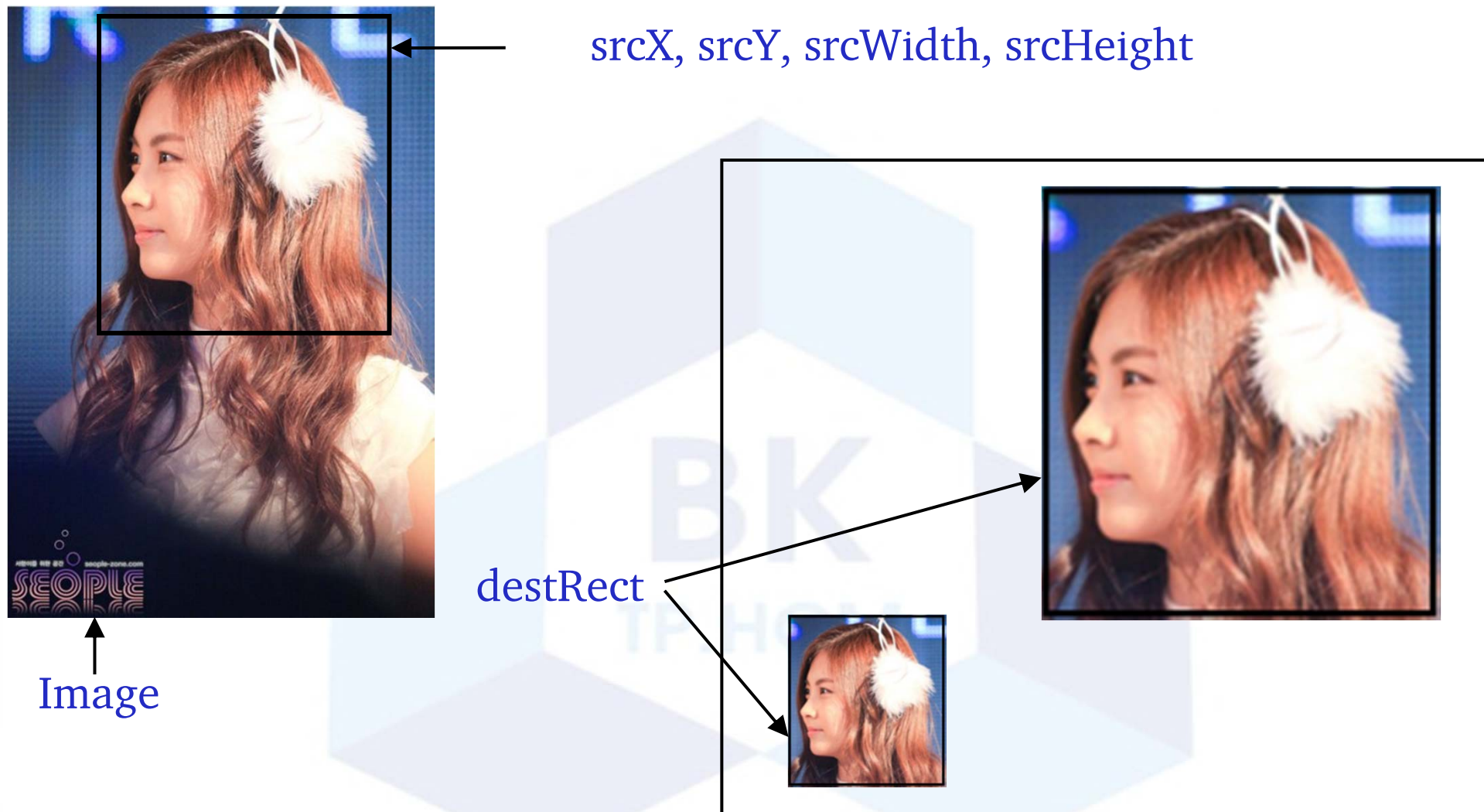
## 7.4 Xuất ảnh bitmap

- ❑ Tác vụ DrawImage cho phép vẽ bitmap từ nguồn có sẵn, thí dụ từ file bitmap. Nó có nhiều biến thể, biến thể khá mạnh và dùng phổ biến có đặc tả như sau :

```
public void DrawImage (  
    Image image,                                //đối tượng chứa ảnh bitmap gốc  
    Rectangle destRect,                        //vùng chữ nhật chứa kết quả  
                                              //trong đối tượng vẽ  
    int srcX,                                  //tọa độ x của vùng ảnh gốc  
    int srcY,                                  //tọa độ y của vùng ảnh gốc  
    int srcWidth,                             //độ rộng vùng ảnh gốc cần vẽ  
    int srcHeight,                            //độ cao vùng ảnh gốc cần vẽ  
    GraphicsUnit srcUnit,                     //đơn vị đo lường được dùng  
    ImageAttributes imageAttr)               //cách thức xử lý từng pixel  
                                              //ảnh gốc khi vẽ
```



## 7.4 Xuất ảnh bitmap



## 7.5 Xuất hình đồ họa - Tác vụ DrawLine

- ❑ Tác vụ DrawLine cho phép vẽ đoạn thẳng được xác định bởi 2 đỉnh. Nó có nhiều biến thể, biến thể khá mạnh và dùng phổ biến có đặc tả như sau :

```
public void DrawLine (
```

```
    Pen pen,          //miêu tả nét, màu đường vẽ  
    int x1,           //tọa độ x của điểm đầu  
    int y1,           //tọa độ y của điểm đầu  
    int x2,           //tọa độ x của điểm cuối  
    int y2);          //tọa độ y của điểm cuối
```

- ❑ Trước khi gọi DrawLine, phải tạo đối tượng Pen miêu tả nét, màu của đường vẽ :

```
//tạo pen với màu Blue, nét vẽ 2 pixel
```

```
Pen pen = new Pen(Color.FromArgb(0,0, 255), 2);
```



## 7.5 Xuất hình đồ họa - Tác vụ DrawLines

- ❑ Tác vụ DrawLines cho phép vẽ nhiều đoạn thẳng liên tiếp nhau được xác định bởi danh sách các đỉnh. Nó có nhiều biến thể, biến thể khá mạnh và dùng phổ biến có đặc tả như sau :

```
public void DrawLines (  
    Pen pen,           //miêu tả nét, màu đường vẽ  
    Point[] points);  //danh sách các đỉnh
```

- ❑ Trước khi gọi DrawLines, phải tạo đối tượng Pen miêu tả nét, màu của đường vẽ :

```
//tạo pen với màu Blue, nét vẽ 2 pixel  
Pen pen = new Pen(Color.FromArgb(0,0, 255), 2);
```



## 7.5 Xuất hình đồ họa - Tác vụ DrawRectangle

- ❑ Tác vụ DrawRectangle cho phép vẽ hình chữ nhật được xác định bởi 2 đỉnh chéo nhau. Nó có nhiều biến thể, biến thể khá mạnh và dùng phổ biến có đặc tả như sau :

```
public void DrawRectangle (  
    Pen pen,           //miêu tả nét, màu đường vẽ  
    int x1,            //tọa độ x của điểm đầu  
    int y1,            //tọa độ y của điểm đầu  
    int x2,            //tọa độ x của điểm cuối  
    int y2)            //tọa độ y của điểm cuối
```

- ❑ Lưu ý tác vụ DrawRectangle chỉ vẽ đường biên, muốn tô nền hình chữ nhật, ta cần gọi tác vụ FillRectangle (đặc tả giống như tác vụ DrawRectangle), chỉ khác là tham số đầu là đối tượng mẫu tô :

```
//tạo brush với màu đỏ, tô đặc
```

```
Brush brush = new SolidBrush(Color.FromArgb(255, 0, 0));
```





## 7.5 Xuất hình đồ họa - Tác vụ DrawEllipse

- ❑ Tác vụ DrawEllipse cho phép vẽ hình ellipse được xác định bởi hình chữ nhật bao quanh nó. Nó có nhiều biến thể, biến thể khá mạnh và dùng phổ biến có đặc tả như sau :

```
public void DrawEllipse (  
    Pen pen,           //miêu tả nét, màu đường vẽ  
    int x1,            //tọa độ x của điểm đầu  
    int y1,            //tọa độ y của điểm đầu  
    int x2,            //tọa độ x của điểm cuối  
    int y2)            //tọa độ y của điểm cuối
```

- ❑ Lưu ý tác vụ DrawEllipse chỉ vẽ đường biên, muốn tô nền hình ellipse, ta cần gọi tác vụ FillEllipse (đặc tả giống như tác vụ DrawEllipse), chỉ khác là tham số đầu là đối tượng mẫu tô :

```
//tạo brush với màu đỏ, tô đặc
```

```
Brush brush = new SolidBrush(Color.FromArgb(255, 0, 0));
```



## 7.5 Xuất hình đồ họa - Tác vụ DrawArc

- ❑ Tác vụ DrawArc cho phép vẽ 1 phần đường ellipse được xác định bởi hình chữ nhật bao quanh nó. Nó có nhiều biến thể, biến thể khá mạnh và dùng phổ biến có đặc tả như sau :

```
public void DrawArc (  
    Pen pen,           //miêu tả nét, màu đường vẽ  
    Rectangle rect,    //miêu tả hình chữ nhật ngoại tiếp  
    float startAngle,  //góc bắt đầu (theo chiều kim đồng hồ)  
    float sweepAngle ) //độ lớn phần đường ellipse cần vẽ
```

- ❑ Lưu ý tác vụ DrawArc chỉ vẽ đường biên, không có tác vụ FillArc để tô nền.





## 7.5 Xuất hình đồ họa - Tác vụ DrawPie

- ❑ Tác vụ DrawPie cho phép vẽ 1 phần bánh ellipse được xác định bởi hình chữ nhật bao quanh nó. Nó có nhiều biến thể, biến thể khá mạnh và dùng phổ biến có đặc tả như sau :

```
public void DrawPie (  
    Pen pen,           //miêu tả nét, màu đường vẽ  
    Rectangle rect,    //miêu tả hình chữ nhật ngoại tiếp  
    float startAngle,  //góc bắt đầu (theo chiều kim đồng hồ)  
    float sweepAngle ) //độ lớn phần bánh ellipse cần vẽ
```

- ❑ Lưu ý tác vụ DrawPie chỉ vẽ đường biên, muốn tô nền bánh ellipse, ta cần gọi tác vụ FillPie (đặc tả giống như tác vụ DrawPie), chỉ khác là tham số đầu là đối tượng mẫu tô :

```
//tạo brush với màu đỏ, tô đặc  
Brush brush = new SolidBrush(Color.FromArgb(255, 0, 0));
```



## 7.5 Xuất hình đồ họa - Tác vụ DrawPolygon

- ❑ Tác vụ DrawPolygon cho phép vẽ hình nhiều cạnh khép kín. Nó có nhiều biến thể, biến thể khá mạnh và dùng phổ biến có đặc tả như sau :

```
public void DrawPolygon (  
    Pen pen,           //miêu tả nét, màu đường vẽ  
    Point[] points)    //danh sách các đỉnh của polygon
```

- ❑ Lưu ý tác vụ DrawPolygon chỉ vẽ đường biên, muốn tô nền hình polygon, ta cần gọi tác vụ FillPolygon (đặc tả giống như tác vụ DrawPolygon), chỉ khác là tham số đầu là đối tượng mẫu tô :

```
//tạo brush với màu đỏ, tô đặc
```

```
Brush brush = new SolidBrush(Color.FromArgb(255, 0, 0));
```



## 7.5 Xuất hình đồ họa - Tác vụ DrawCurve

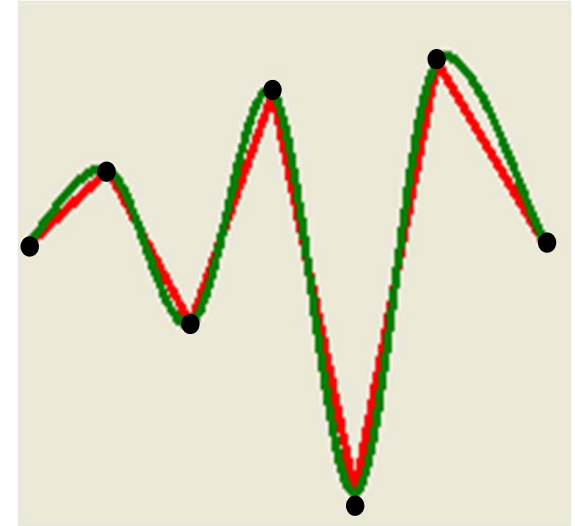
- ❑ Tác vụ DrawCurve cho phép vẽ cong tròn xuyên qua nhiều điểm theo phép tension xác định. Nó có nhiều biến thể, biến thể khá mạnh và dùng phổ biến có đặc tả như sau :

```
public void DrawCurve (  
    Pen pen,                //miêu tả nét, màu đường vẽ  
    Point[] points          //danh sách các đỉnh của polygon  
    int offset,             //vị trí điểm bắt đầu vẽ trong danh sách  
    int numberOfSegments,  //số đoạn cần vẽ  
    float tension);        //phép tension được dùng  
)
```



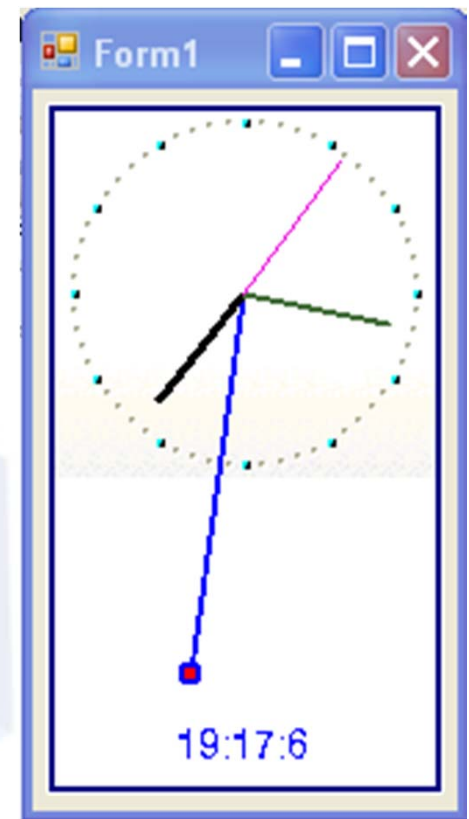
## 7.5 Xuất hình đồ họa - Tác vụ DrawCurve

```
private void Form1_Paint(object sender, PaintEventArgs e) {  
    //tạo 2 bút vẽ cho đường thẳng và cong  
    Pen redPen = new Pen(Color.Red, 3);  
    Pen greenPen = new Pen(Color.Green, 3);  
    //tạo các đỉnh  
    Point point1 = new Point(10, 100), point2 = new Point(40, 75);  
    Point point3 = new Point(70, 125), point4 = new Point(100, 50);  
    Point point5 = new Point(130, 180), point6 = new Point(160, 40);  
    Point point7 = new Point(200, 100);  
    Point[] curvePoints = { point1, point2, point3, point4, point5, point6, point7 };  
    //vẽ các đoạn thẳng.  
    e.Graphics.DrawLines(redPen, curvePoints);  
    //thiết lập offset, số đoạn cong, và tension.  
    int offset = 0, numSegments = 6;  
    float tension = 0.5F;  
    //vẽ đường cong trơn qua các đỉnh.  
    e.Graphics.DrawCurve(greenPen, curvePoints, offset, numSegments, tension);  
}
```



## 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

- ❑ Để củng cố kiến thức về các tác vụ xuất nội dung tổng hợp chứa chuỗi văn bản, ảnh bitmap và các hình đồ họa toán học, chúng ta hãy viết ứng dụng giả lập đồng hồ treo tường có 3 kim giờ/phút/giây và có quả lắc theo góc 20 độ.
- ❑ Phân tích thông tin cần xuất, ta thấy có các thành phần :
  - hình bitmap miêu tả khung đồng hồ, bản số đồng hồ.
  - 4 đoạn thẳng miêu tả 3 kim giờ/phút/giây và cần lắc. Vòng tròn nhỏ miêu tả quả lắc. Các hình toán học này thay đổi vị trí theo thời gian.
  - chuỗi hiển thị giờ/phút/giây.




## 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

- ❑ Dùng đối tượng Timer với thời gian đếm khoảng 40ms, mỗi lần đếm xong nó tạo sự kiện Paint để kích hoạt hàm vẽ lại Form ứng dụng. Như vậy mỗi giây ta vẽ lại khoảng 25 lần, tốc độ như thế này là vừa đủ để người dùng cảm thấy đồng hồ gần như thật.
- ❑ Qui trình điển hình để xây dựng ứng dụng đồng hồ quả lắc gồm các bước sau đây :
  1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.
  2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Windows, chọn icon "Windows Application" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (td. VCDongho), click button OK để tạo Project theo các thông số đã khai báo.



## 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

3. Form đầu tiên của ứng dụng đã hiển thị trong cửa sổ thiết kế, lúc này form hoàn toàn trống, chưa chứa đối tượng giao diện nào.
4. Nếu cửa sổ ToolBox chưa hiển thị, chọn menu View.Toolbox để hiển thị nó (thường nằm ở bên trái màn hình). Duyệt tìm phần tử Timer (trong nhóm Components hay nhóm All Window Forms), chọn nó, dôi chuột vào trong form (ở vị trí nào cũng được vì đối tượng này không được hiển thị) và vẽ nó với kích thước tùy ý. Hiệu chỉnh thuộc tính (Name) = myTimer.
5. Chọn đối tượng myTimer, cửa sổ thuộc tính của nó sẽ hiển thị, click icon  để hiển thị danh sách các sự kiện của đối tượng, ấn kép chuột vào comboBox bên phải sự kiện Tick để máy tạo tự động hàm xử lý cho sự kiện này.




## 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

6. Viết code cụ thể cho hàm như sau :

//hàm phục vụ Timer

```
private void myTimer_Tick(object sender, EventArgs e) {  
    myTimer.Stop(); //dừng đếm timer  
    this.Refresh(); //vẽ lại form theo giờ hiện hành  
}
```

7. Ấn phải chuột vào mục Form1.cs trong cửa sổ Solution Explorer rồi chọn option View Designer để hiển thị lại cửa sổ thiết kế Form. Chọn Form, cửa sổ thuộc tính của nó sẽ hiển thị, click icon  để hiển thị danh sách các sự kiện của Form, duyệt tìm sự kiện Paint, ấn kép chuột vào comboBox bên phải sự kiện Paint để máy tạo tự động hàm xử lý cho sự kiện này. Viết code cụ thể cho hàm như sau :





## 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

```
private void Form1_Paint(object sender, PaintEventArgs e) {  
    //tạo đối tượng image gốc  
    Image bgimg = Image.FromFile("c:\\bgclock.bmp");  
    //xác định đối tượng mục tiêu  
    Control control = (Control)sender;  
    //thay đổi kích thước form theo ảnh khung đồng hồ  
    control.Size = new Size(bgimg.Width + 10 + 8, bgimg.Height + 10 +  
35);  
    //xác định đối tượng graphics (đối tượng vẽ) của đối tượng  
    Graphics g = e.Graphics;  
    //vẽ bitmap miêu tả khung đồng hồ  
    g.DrawImage(bgimg, 5,5);  
    //định nghĩa các biến cần dùng  
    Rectangle rec = control.DisplayRectangle;  
    Pen hPen;
```



## 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

```
Brush hBrush;  
int xo,yo,rql,rh,rm, rs;  
int x, y;  
//thiết lập tâm đồng hồ  
xo = 76; yo = 74;  
//thiết lập bán kính cần lắc, kim giờ/phút/giây  
rql = 140; rh = 50; rm = 55; rs = 60;  
//tạo pen để vẽ cần lắc  
hPen = new Pen (Color.FromArgb(0,0, 255),2);  
//tạo brush để tô nền quả lắc  
hBrush = new SolidBrush(Color.FromArgb(255, 0, 0));  
//xác định giờ/phút/giây hiện hành  
DateTime now = DateTime.Now;
```



## 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

```
//tính góc của cần lắc (góc quay max. là 40 độ)
double goc = 80*now.Millisecond/1000;
if (goc < 40) goc = goc +70;
else goc = 150-goc;
//đổi góc cần lắc từ độ ra radian
goc = goc*3.1416/180;
//xác định tâm quả lắc (điểm còn lại của cần lắc)
x = xo+(int)(rql*Math.Cos(goc));
y = yo+(int)(rql*Math.Sin(goc));
//vẽ cần lắc
g.DrawLine(hPen, xo, yo, x, y);
//vẽ quả lắc
g.FillEllipse(hBrush, x-3, y-3, 5, 5);
g.DrawEllipse(hPen,x-4,y-4,7,7);
```



## 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

```
//tạo pen để vẽ kim giờ  
hPen = new Pen(Color.FromArgb(0,0,0),3);  
//tính góc của kim giờ  
goc = 90+360*(now.Hour+(double)now.Minute/60)/12;  
//đổi góc từ độ ra radian  
goc = goc*3.1416/180;  
//xác định tọa độ đỉnh thứ 2 của kim giờ  
x = xo - (int)(rh * Math.Cos(goc));  
y = yo - (int)(rh * Math.Sin(goc));  
//vẽ kim giờ  
g.DrawLine(hPen, xo, yo, x, y);
```



## 7.6 Thí dụ viết ứng dụng vẽ đồng hồ phức hợp

```
//tạo pen để vẽ kim phút  
hPen = new Pen(Color.FromArgb(65,110,55),2);  
//tính góc của kim phút  
goc = 90+360*now.Minute/60;  
//đổi góc từ độ ra radian  
goc = goc*3.1416/180;  
//xác định tọa độ đỉnh thứ 2 của kim phút  
x = xo - (int)(rm * Math.Cos(goc));  
y = yo - (int)(rm * Math.Sin(goc));  
//vẽ kim phút  
g.DrawLine(hPen, xo, yo, x, y);
```



## 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

```
//tạo pen để vẽ kim giây  
hPen = new Pen(Color.FromArgb(237,5,220),1);  
//tính góc của kim giây  
goc = 90+360*now.Minute/60;  
//đổi góc từ độ ra radian  
goc = goc*3.1416/180;  
//xác định tọa độ đỉnh thứ 2 của kim giây  
x = xo - (int)(rs * Math.Cos(goc));  
y = yo - (int)(rs * Math.Sin(goc));  
//vẽ kim giây  
g.DrawLine(hPen, xo, yo, x, y);
```



## 7.6 Thí dụ viết ứng dụng vẽ đối tượng phức hợp

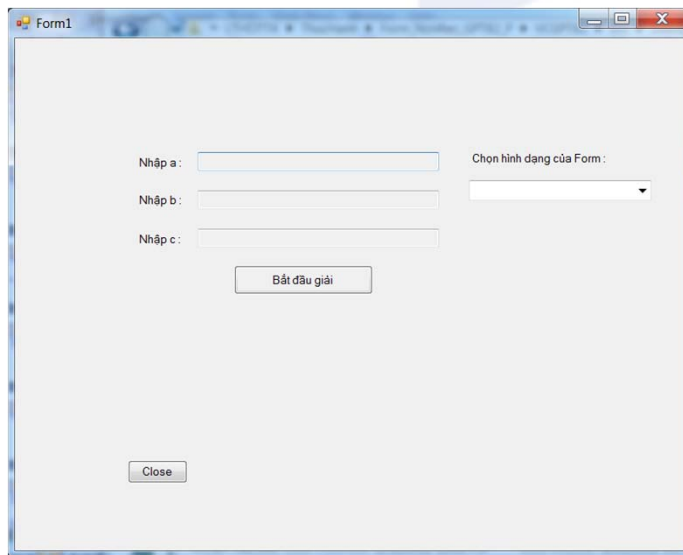
```
//tạo chuỗi miêu tả giờ/phút/giây hiện hành
String buf = "" + now.Hour + ":" + now.Minute + ":" + now.Second;
//tạo đối tượng font chữ cần dùng
Font myFont = new Font("Helvetica", 11);
//tạo biến miêu tả chế độ canh giữa khi xuất chuỗi
StringFormat format1 = new StringFormat(StringFormatFlags.NoClip);
format1.Alignment = StringAlignment.Center;
//xuất chuỗi miêu tả giờ/phút/giây
g.DrawString(buf, myFont, System.Drawing.Brushes.Blue,
    xo, rec.Height - 35, format1);
//cho phép timer chạy tiếp
myTimer.Start();
}
```

8. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng.  
Xem kết quả và đánh giá kết quả.



## 7.7 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Các đối tượng giao diện, dù nhỏ hay lớn (Button, TextBox, ListBox, TreeView,...), đều được Windows quản lý giống nhau : Windows xử lý chúng như là window.
- ❑ Mỗi window sẽ được hiển thị ở dạng mặc định là hình chữ nhật có đường viền xung quanh và titlebar ở phía trên. Tuy nhiên ta có thể miêu tả lại hình dạng cho window theo nhu cầu riêng của mình.





## 7.7 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Window chứa các thuộc tính sau đây có liên quan đến việc xác định chính xác hình dạng của nó :
  - BackgroundImage : miêu tả hình bitmap được dùng để hiển thị nền window và để xác định hình dạng của window.
  - FormBorderStyle : miêu tả chế độ hiển thị các đường biên và titlebar của window.
  - Region : miêu tả vùng hiển thị và làm việc của window, nó gồm từ 1 tới nhiều vùng rời rạc, mỗi vùng rời rạc được bao đóng bởi 1 đường viền khép kín.



## 7.7 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Đường viền khép kín của 1 vùng độc lập có thể được xác định bằng 1 trong 2 phương pháp :
  - Danh sách các đoạn thẳng hay cong liên tiếp và khép kín, mỗi đoạn thẳng hay cong có thể miêu tả bởi 1 hàm toán học như Line, Arc,....
  - Do hình bitmap nào đó xác định.
- ❑ Có 2 kỹ thuật xây dựng window có hình dạng bất kỳ :
  - Khai báo các thuộc tính liên quan 1 cách trực quan tại thời điểm thiết kế.
  - Lập trình động để thiết lập các giá trị phù hợp cho các thuộc tính liên quan đến window.



## 7.7 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Qui trình xây dựng Form giao diện có hình dạng bất kỳ bằng cách khai báo các thuộc tính liên quan 1 cách trực quan tại thời điểm thiết kế : Tạo form cần dùng, chọn nó để hiển thị cửa sổ thuộc tính, tìm và thiết lập giá trị cho các thuộc tính sau đây :
  - `BackgroundImage` : khai báo file bitmap được dùng để hiển thị nền của Form và để xác định hình dạng của Form. Lưu ý hình bitmap cần có tính chất : các vùng diện tích của bitmap phải có màu khác với màu nền của hình bitmap; kích thước hình bitmap nên phù hợp với nhu cầu sử dụng của form tương ứng.
  - `FormBorderStyle = None` để không hiển thị titlebar và đường viền mặc định.
  - `TransparenceKey` : miêu tả màu nền của hình bitmap cần lọc bỏ (theo định dạng RGB).



## 7.7 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Thí dụ hãy xây dựng ứng dụng giải phương trình bậc 2 có hình dạng như sau :



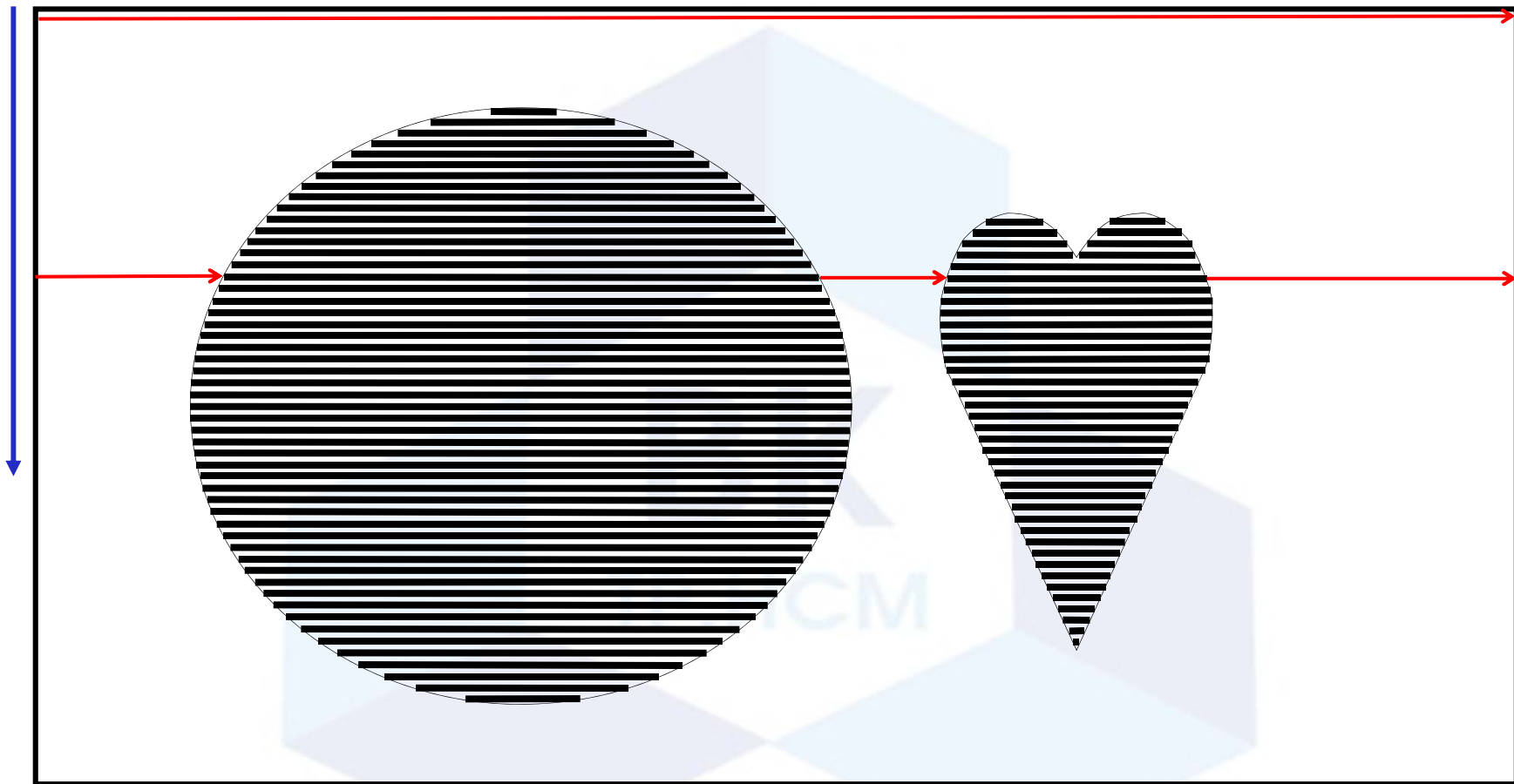
## 7.7 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Qui trình xây dựng đối tượng giao diện có hình dạng bất kỳ bằng cách viết code thiết lập động các thuộc tính liên quan : Tạo đối tượng cần dùng, viết đoạn code thiết lập 3 thuộc tính liên quan khi cần thiết :
  - `BackgroundImage` : miêu tả hình bitmap được dùng để hiển thị nền window (nếu muốn hiển thị hình nền).
  - `FormBorderStyle = None`.
  - `Region` : miêu tả vùng diện tích làm việc của đối tượng.
- ❑ Thường `Region` được xác định thông qua đối tượng `Path`, đối tượng này miêu tả đường viền của `Region`.
- ❑ Để tạo đối tượng `Path`, ta có thể dùng các hàm toán học miêu tả từng đoạn viền khép kín của `Region` hay dùng đường viền của hình bitmap bất kỳ.



## 7.7 Xây dựng đối tượng giao diện có hình dạng tùy ý

Qui trình tìm Path của hình đồ họa bitmap



## 7.7 Xây dựng đối tượng giao diện có hình dạng tùy ý

```
for (y = 0; y <= yMax - 1; y++) { //duyet từng hàng bitmap
    idx = isrow;    //xác định offset (trong buffer) chứa pixel cần xử lý
    for (x = 0; x <= xMax - 1; x++) { //duyet từng pixel trên hàng
        if (Equal(pbase, idx, Key)) { //nếu là pixel nền thì bỏ qua
            idx = idx + 4; continue; }
        //nhớ lại vị trí pixel tái nhất của vùng đang tìm được
        int x0 = x;
        //duyet tìm các pixel còn lại của vùng hiện hành
        while (x < xMax && (!Equal(pbase, idx, Key))) {
            x = x + 1; idx = idx + 4; }
        //add path của vùng tìm được vào đối tượng Path
        path.AddRectangle(new Rectangle(x0, y, x - x0, 1));
    }
    isrow = isrow + bitmapData.Stride; //đến pixel đầu hàng kế
}
```

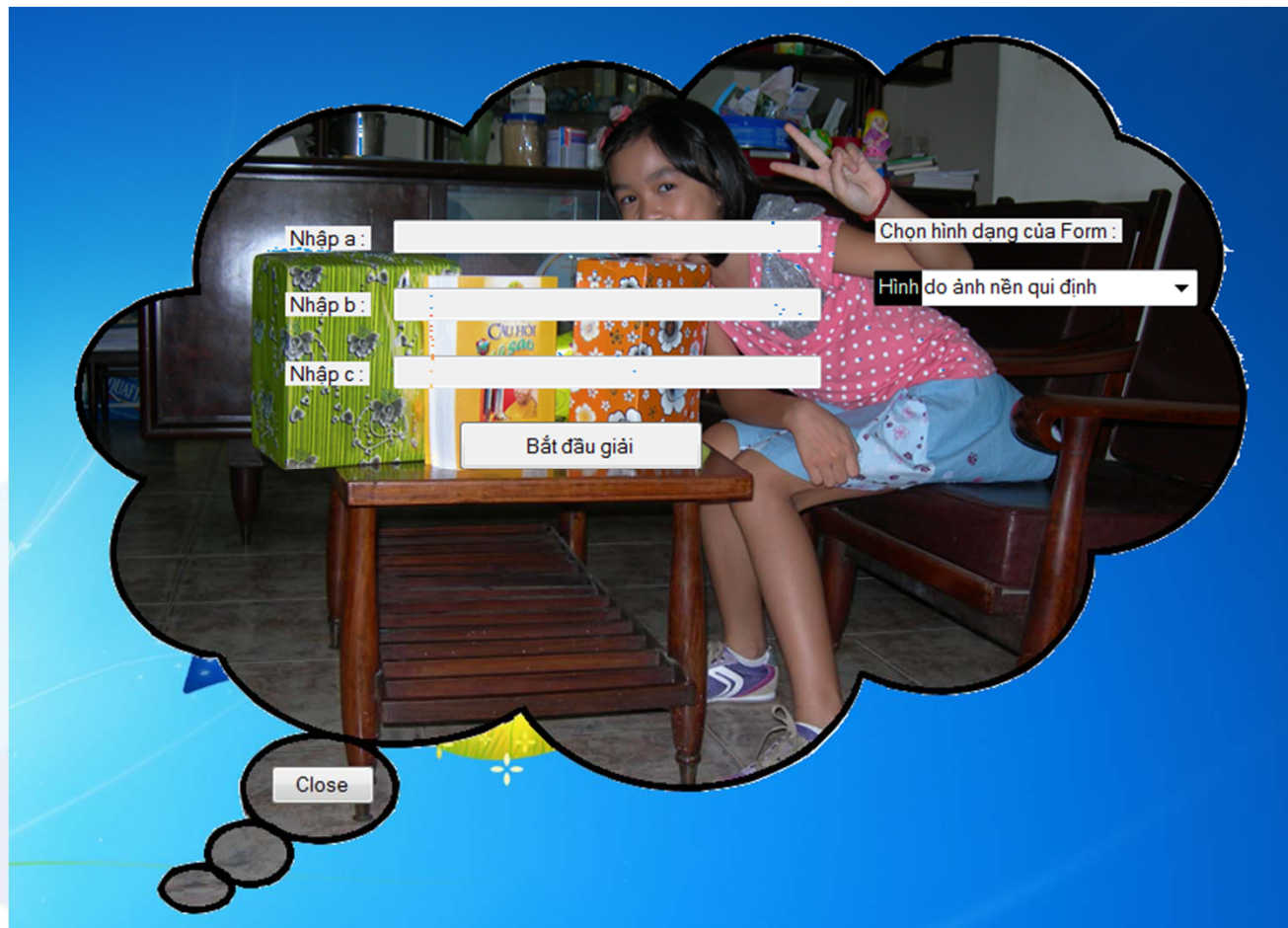
B
G
R
A





## 7.7 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Thí dụ hãy xây dựng ứng dụng giải phương trình bậc 2 có hình dạng như sau :





## 7.8 Kết chương

- ❑ Chương này đã giới thiệu cách thức tương tác giữa người dùng và chương trình để nhập/xuất dữ liệu.
- ❑ Chương này cũng đã giới thiệu các đối tượng giao diện cùng các tác vụ xuất dữ liệu dạng chuỗi, dạng bitmap, dạng hình đồ họa toán học. Kết hợp 3 loại dữ liệu này, ta có thể tạo kết xuất bất kỳ.

