

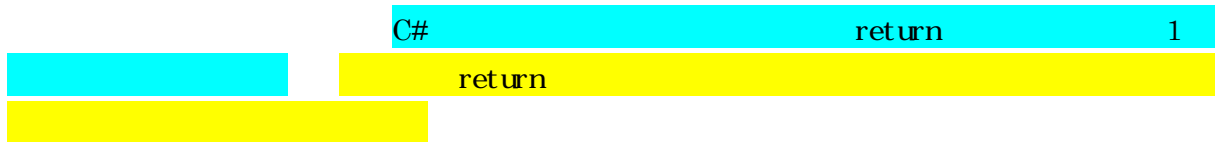
1

1

18x0123		z1021234	100	S
---------	--	----------	-----	---

C#

structure



C#

```

1)
2)
3)
    
```

```

1)
struct Student
{
    public String id;
    public String name;
    public String class;
    public int point;
    public char eval;
}

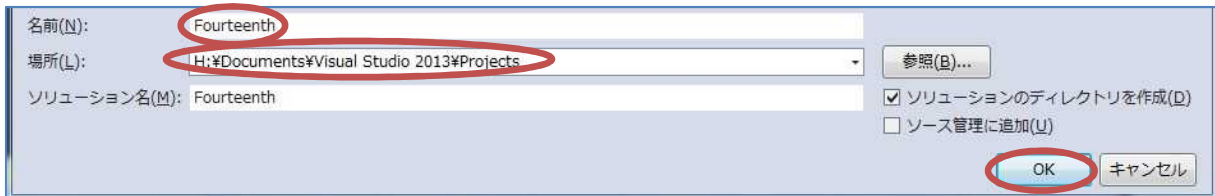
2)
Student st;

3)
st.id = "18x0123";
st.name = " ";
st.class = "z1021234";
st.point = 100;
st.eval = 'S';
    
```

1)

Visual Studio 2013 [ ] [ ] [ ]

Visual C# Windows  
 Fourteenth H:\Documents\Visual Studio 2013\Projects  
 3\Projects OK 1



2)

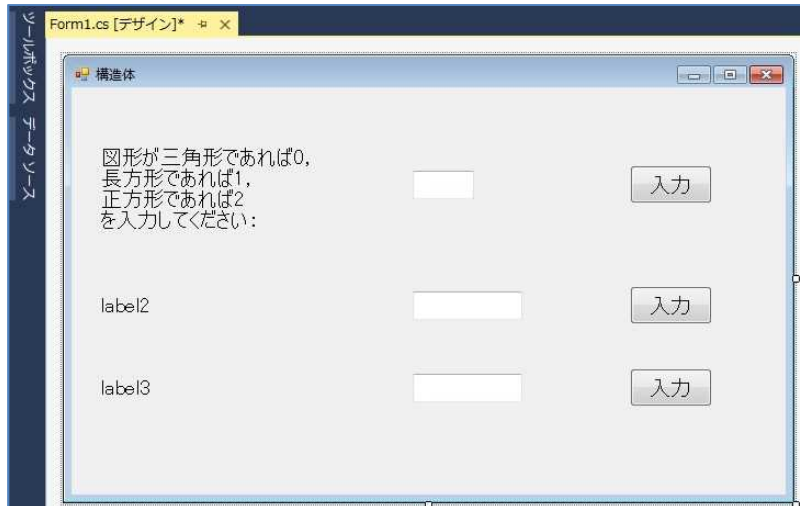
14

Form1 3 3 3

Form1 Text  
 label1 Text

0  
 1  
 2

button1 Text  
 button2 Text  
 button3 Text

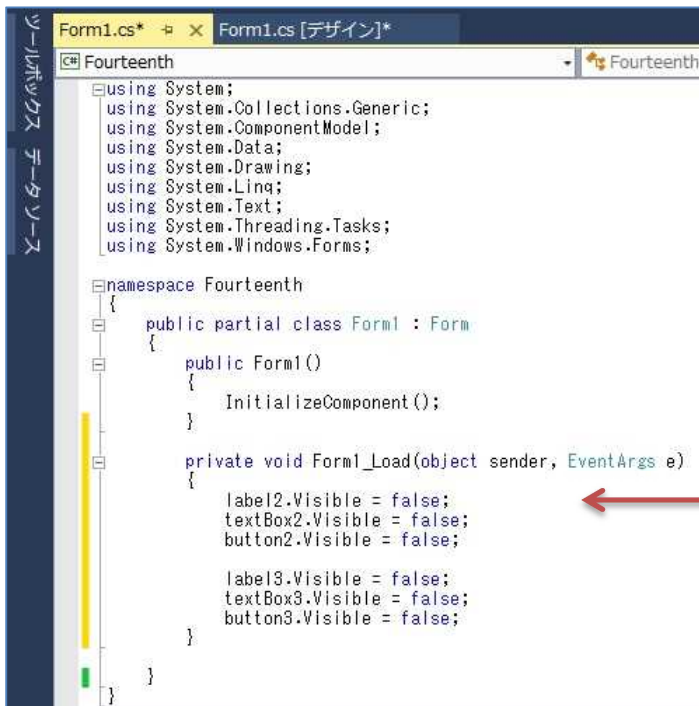


3)

Form1  
 Form1.cs Form1\_Load Form1  
 label2 textBox2 button2 label3 textBox3  
 button3 Visible false

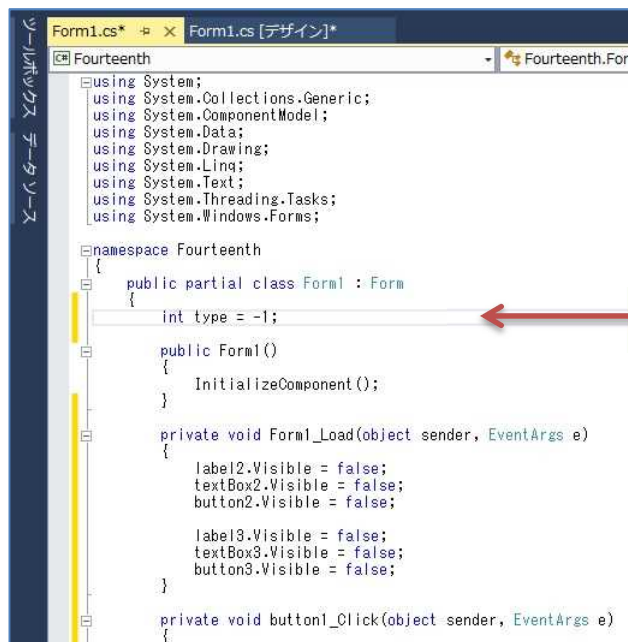
```
private void Form1_Load(object sender, EventArgs e)
{
    label2.Visible = false;
    textBox2.Visible = false;
    button2.Visible = false;

    label3.Visible = false;
    textBox3.Visible = false;
    button3.Visible = false;
}
```



Form1  
button1  
Form1.cs

```
int type = -1;
```



button1\_Click

```

private void button1_Click(object sender, EventArgs e)
{
    try
    {
        type = Int32.Parse(textBox1.Text);
        String s1 = "", s2 = "";

        if (type < 0 || type > 2)
        {
            MessageBox.Show("Invalid Input\n 0 1 2", "Invalid Input");
        }
        else if (type == 0)
        {
            s1 = "Enter X ( )\n";
            s2 = "Enter Y ( )\n";
        }
        else if (type == 1)
        {
            s1 = "Enter X ( )\n";
            s2 = "Enter Y ( )\n";
        }
        else
        {
            s1 = "Enter ( )\n";
        }

        label2.Text = s1;
        label3.Text = s2;

        if (type >= 0 && type <= 2)
        {
            label2.Visible = true;
            textBox2.Visible = true;
            button2.Visible = true;
        }
    }
    catch (FormatException fex)
    {
        MessageBox.Show(fex.Message, "Invalid Input");
    }
    finally
    {
        textBox1.Text = "";
    }
}

```

```

private void button1_Click(object sender, EventArgs e)
{
    try
    {
        type = Int32.Parse(textBox1.Text);
        String s1 = "", s2 = "";

        if (type < 0 || type > 2)
        {
            MessageBox.Show("入力する数値は\n 0, 1, 2のいずれかです。", "範囲外");
        }
        else if (type == 0)
        {
            s1 = "三角形の底辺の長さ(実数値)\nを入力してください。";
            s2 = "三角形の高さ(実数値)\nを入力してください。";
        }
        else if (type == 1)
        {
            s1 = "長方形の一边Xの長さ(実数値)\nを入力してください。";
            s2 = "長方形の一边Yの長さ(実数値)\nを入力してください。";
        }
        else
        {
            s1 = "正方形の一边の長さ(実数値)\nを入力してください。";
        }

        label2.Text = s1;
        label3.Text = s2;

        if (type >= 0 && type <= 2)
        {
            label2.Visible = true;
            textBox2.Visible = true;
            button2.Visible = true;
        }
    }
    catch (FormatException fex)
    {
        MessageBox.Show(fex.Message, "エラー");
    }
    finally
    {
        textBox1.Text = "";
    }
}

```



Form1

button2

button3

Form1.cs

```

struct Trirect // triangle/rectangle ( / )
{
    public String name; // name ( )
    public double s; // side ( )
    public double h; // height ( )
    public double a; // area ( )
}

private Trirect figure;

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Fourteenth
{
    public partial class Form1 : Form
    {
        int type = -1;

        struct Trirect // triangle/rectangle (三角形/四角形)
        {
            public String name; // name (名称)
            public double s; // side (辺)
            public double h; // height (高さ)
            public double a; // area (面積)
        }

        private Trirect figure;

        public Form1()
    }
}

```



getArea

```

Trirect getArea(Trirect fig, int t)
{
    if (t == 0)
    {
        fig.name = " ";
        fig.a = fig.s * fig.h / 2;
    }
    else
    {
        fig.a = fig.s * fig.h;

        if (t == 1)
        {
            fig.name = " ";
        }
        else
            fig.name = " ";
    }

    return fig;
}

```

The screenshot shows the Visual Studio IDE with the following code in Form1.cs:

```

}
}
private void button2_Click(object sender, EventArgs e)
{
}
private void button3_Click(object sender, EventArgs e)
{
}
Trirect getArea(Trirect fig, int t)
{
    if (t == 0)
    {
        fig.name = "三角形";
        fig.a = fig.s * fig.h / 2;
    }
    else
    {
        fig.a = fig.s * fig.h;

        if (t == 1)
        {
            fig.name = "長方形";
        }
        else
            fig.name = "正方形";
    }

    return fig;
}
}

```

A red arrow points to the line `Trirect getArea(Trirect fig, int t)`.

button2\_Click

```

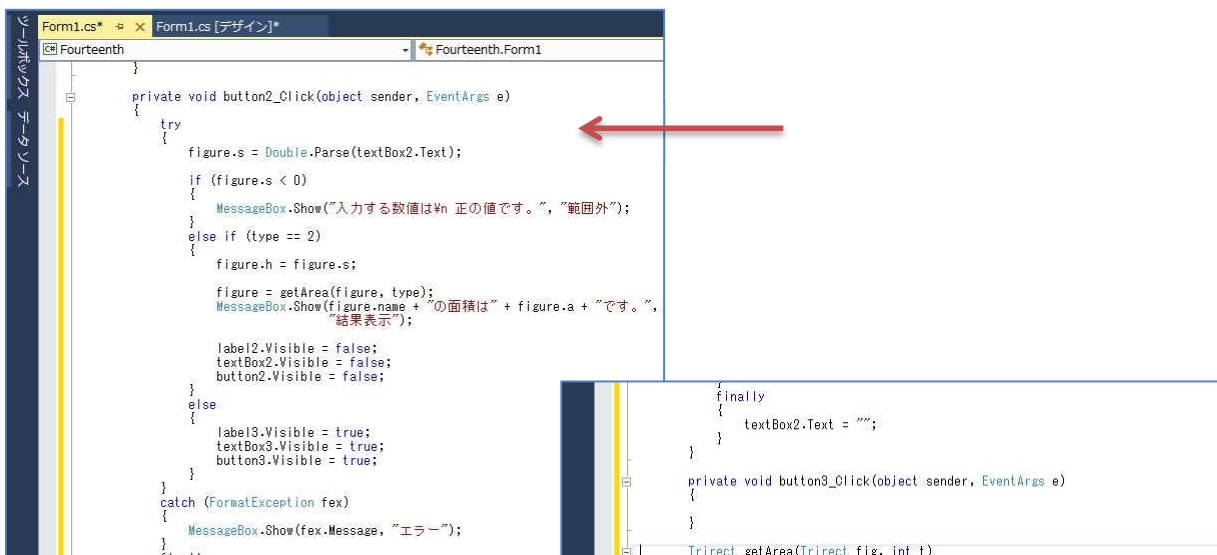
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        figure.s = Double.Parse(textBox2.Text);

        if (figure.s < 0)
        {
            MessageBox.Show("          \n          ", "          ");
        }
        else if (type == 2)
        {
            figure.h = figure.s;

            figure = getArea(figure, type);
            MessageBox.Show(figure.name + "          " + figure.a + "          ",
                "          ");

            label2.Visible = false;
            textBox2.Visible = false;
            button2.Visible = false;
        }
        else
        {
            label3.Visible = true;
            textBox3.Visible = true;
            button3.Visible = true;
        }
    }
    catch (FormatException fex)
    {
        MessageBox.Show(fex.Message, "          ");
    }
    finally
    {
        textBox2.Text = "";
    }
}

```



button3\_Click

```

private void button3_Click(object sender, EventArgs e)
{
    try
    {
        figure.h = Double.Parse(textBox3.Text);

        if (figure.h < 0)
        {
            MessageBox.Show("          \n          ", "          ");
        }
        else
        {
            figure = getArea(figure, type);
            MessageBox.Show(figure.name + "          " + figure.a + "          ",
                "          ");

            label2.Visible = false;
            textBox2.Visible = false;
            button2.Visible = false;

            label3.Visible = false;
            textBox3.Visible = false;
            button3.Visible = false;
        }
    }
    catch (FormatException fex)
    {
        MessageBox.Show(fex.Message, "          ");
    }
    finally
    {
        textBox3.Text = "";
    }
}

```

The screenshot shows the Visual Studio IDE with the C# code for the button3\_Click event handler. The code is identical to the one shown in the previous block. A red arrow points to the try block, highlighting the parsing and validation logic. The IDE window title is 'Form1.cs [デザイン]\*' and the file name is 'Fourteenth.Form1'. The code is as follows:

```

private void button3_Click(object sender, EventArgs e)
{
    try
    {
        figure.h = Double.Parse(textBox3.Text);

        if (figure.h < 0)
        {
            MessageBox.Show("入力する数値は正の値です。", "範囲外");
        }
        else
        {
            figure = getArea(figure, type);
            MessageBox.Show(figure.name + "の面積は" + figure.a + "です。",
                "結果表示");

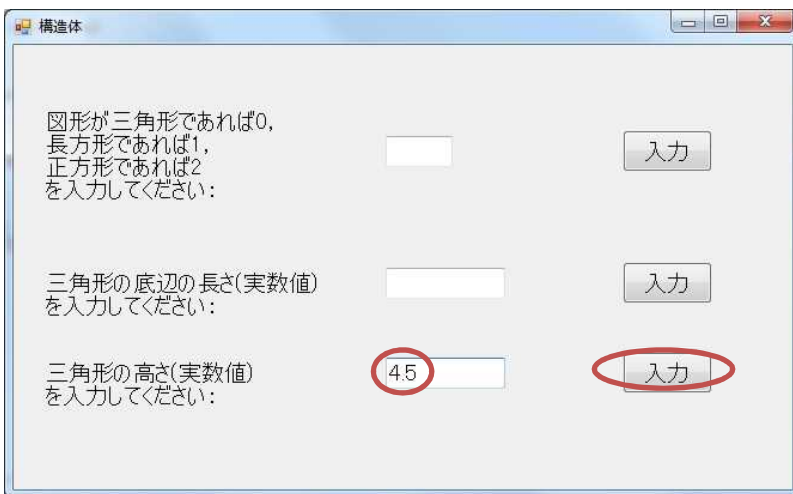
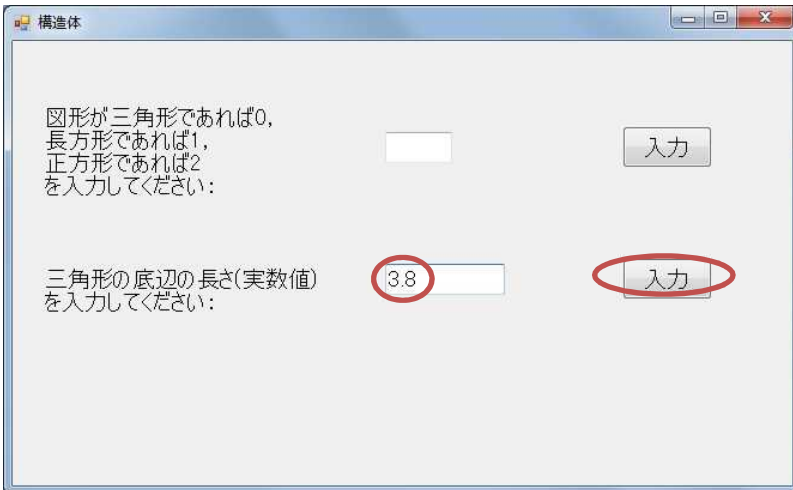
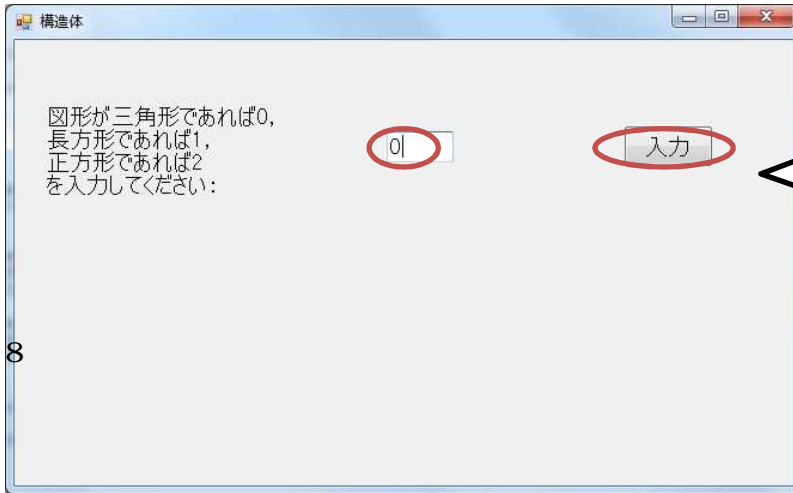
            label2.Visible = false;
            textBox2.Visible = false;
            button2.Visible = false;

            label3.Visible = false;
            textBox3.Visible = false;
            button3.Visible = false;
        }
    }
    catch (FormatException fex)
    {
        MessageBox.Show(fex.Message, "エラー");
    }
    finally
    {
        textBox3.Text = "";
    }
}

```



4)



Fourteenth



- 1) Form1.Designer.cs
- 2) Form1.cs
- 3) Questions\_14th.txt