2018 年 11 月 8 日（木）実施

プログラムの制御構造

1960 年代後半にダイクストラが提唱した**構造化プログラミング**という考え方では，手続き型のプログラムを記述する際には，**順次**，**選択**，**反復**という標準的な**制御構造**のみを用い，先ずプログラムの概略構造を設計し，その大まかな単位を段階的に詳細化して処理を記述していく。

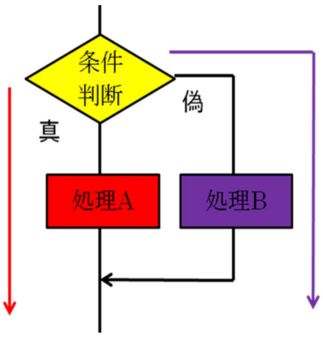
C#言語で Windows フォームアプリケーションを作成する場合にも，個々のイベントハンドラに記述される処理にはこの考え方が適用出来る。

順次構造

順次構造とは，プログラム中の文を処理していく順に記述したものである。これまで扱ったプログラムは，全て順次構造によって記述されたものであり，最も基本的な制御構造と言える。

プログラムの処理の流れを図示する手法の一つに**流れ図**がある。この流れ図で順次構造を表すと右図の様になる。(色矢印は処理の流れを補足)

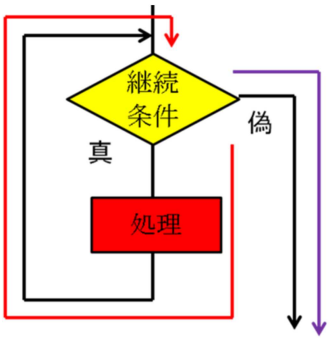
選択構造



選択構造とは，条件や式の値によってプログラムの処理の流れを分ける構造である。選択構造の基本は 2 分岐と呼ばれる構造で，この構造を流れ図で表すと右図の様になる。

また，式の値によって，幾つもの異なる処理が必要なときには，**多分岐**という選択構造も利用可能である。

反復構造

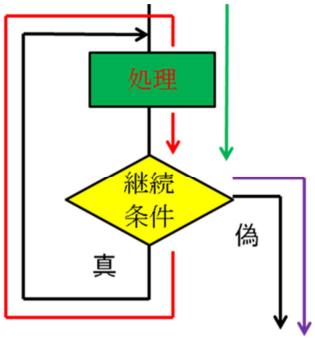


反復構造とは，継続条件が満たされている間，定められた範囲内の文に記述された処理を繰り返して実行する構造である。(C#言語以外のプログラム言語では，終了条件が満たされない間，文を繰り返して実行する構造を持つものもある)

なお，反復構造には，右の流れ図で表される 2 種類の場合がある。

1）0 回以上の繰り返し（右上図）

先ず継続条件が判定され，真であれば定められた範囲内の文に記述された処理を実行する。始めから継続条件が満たされない場合には，文は全く実行されないため，0 回以上の繰り返しと呼ばれる。

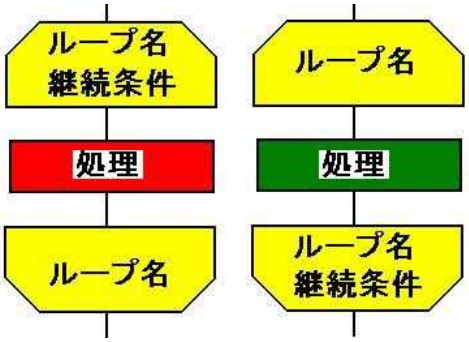


2）1 回以上の繰り返し（右下図）

先ず定められた範囲内の文に記述された処理が実行され，その後に継続条件が判定される。始めから継続条件が満たされない場合でも，最初の 1 回は定められた範囲内の文に記述された処理が実行されるため，1 回以上の繰り返しと呼ばれる。

＊ 反復構造に対する流れ図に表れる閉線図形をループと呼ぶことから，反復構造のプログラム構造をループと称することがある。

複雑な処理は，順次構造，選択構造，反復構造の組み合わせで実現される。プログラムの構造は，最も大掴みにした概略構造で見ると順次構造となる。選択構造や反復構造を中間の概略構造と看做した場合，その詳細構造として，それぞれの構造の流れ図で『処理』と書かれた箇所に，選択構造や反復構造を埋め込んだ構造も可能である。そこで，反復構造の開始位置と終了位置とを『ループ端』によって明示し，構造を見易くした右のような流れ図も利用される。



2 分岐のプログラム

<u>if 文</u>

C#言語で 2 分岐のプログラムを実現するための文として，if 文が用意されている。if 文の構文は次の様になる。

```
if ( 条件式 ) { 真文 } [ else { 偽文 } ]
```

ここで，[]内は省略可能であり，省略時は条件式が偽のときは何もしない場合を表す。また，真文または偽文は複数の文で構成することが出来，単独の文の場合は{}を省略可能である。

<u>条件式</u>

条件式では，条件が満たされる（真となる）場合には，値が true となり，条件が満たされない（偽となる）場合には，値が false となる。true 及び false は論理型（型名は bool）のリテラルである。（第 5 回教材の p.1 参照）

条件式で用いられる**関係演算子**及び**等値演算子**を次に挙げる。

関係演算子	書式	意味
<	x < y	x が y より小さければ true，それ以外は false
>	x > y	x が y より大きければ true，それ以外は false
<=	x <= y	x が y より小さいか，両者が等しければ true，それ以外は false
>=	x >= y	x が y より大きいか，両者が等しければ true，それ以外は false

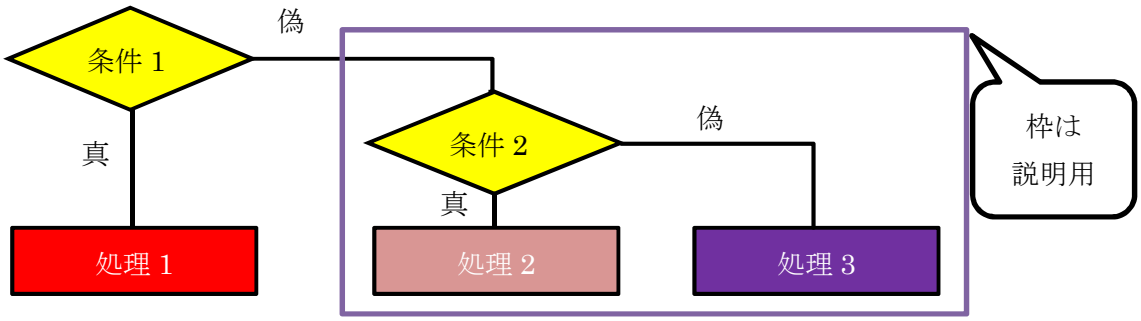
等値演算子	書式	意味
==	x == y	x と y とが等しければ true，それ以外は false
!=	x != y	x と y とが等しくなければ true，それ以外は false

また，複数の条件式を組み合わせるために用いられる**論理演算子（論理否定演算子，条件 AND 演算子**及び**条件 OR 演算子**）を次に挙げる。なお，論理否定演算子は第 6 回教材の p.1 に登場した単項演算子の一つであり，条件 AND 演算子及び条件 OR 演算子は加法演算子よりも優先順位が低い。

論理演算子	書式	意味
!	!式	式が true であれば false，式が false であれば true
&&	式 1 && 式 2	式 1 及び式 2 が共に true であれば true，それ以外は false
||	式 1 || 式 2	式 1 または式 2 のどちらか一方が true であれば true（両者が true の場合も含む），それ以外は false

入れ子の if 文

if 文では，真文または偽文として if 文を用いることが可能である。これを"**入れ子（ネスト）の if 文**"と呼ぶ。if 文は 2 分岐であるが，入れ子の if 文を用いることによって，3 以上の分岐(**多分岐**)を表現することが可能となる。次の図は，偽文として if 文を用いた入れ子の if 文の流れ図である。



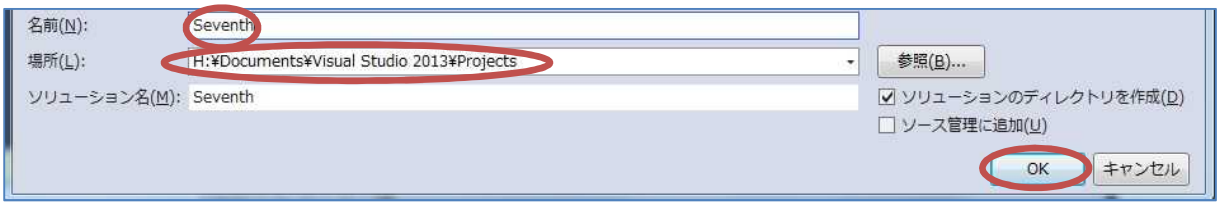
本日の課題

複数の条件式の組み合わせや入れ子の if 文を利用して，範囲外の点数を入力した際の処理，合否判定によるボタンの表示の可否，成績評価の表示を場合分けするプログラムを作成する。

手順

1）プロジェクトの作成

Visual Studio 2013 を起動したら，［ファイル］→［新規作成］→［プロジェクト］と辿って，プロジェクトを作成する。『新しいプロジェクト』ダイアログボックスでは，プログラミング言語を『Visual C#』，プロジェクトテンプレートとしては，『Windows フォームアプリケーション』を選択し，『名前』を「Seventh」に書き換え，『場所』が「H:¥Documents¥Visual Studio 2013¥Projects」となっていることを確認してから『OK』を押す（詳細は第 1 回の教材を参照）。

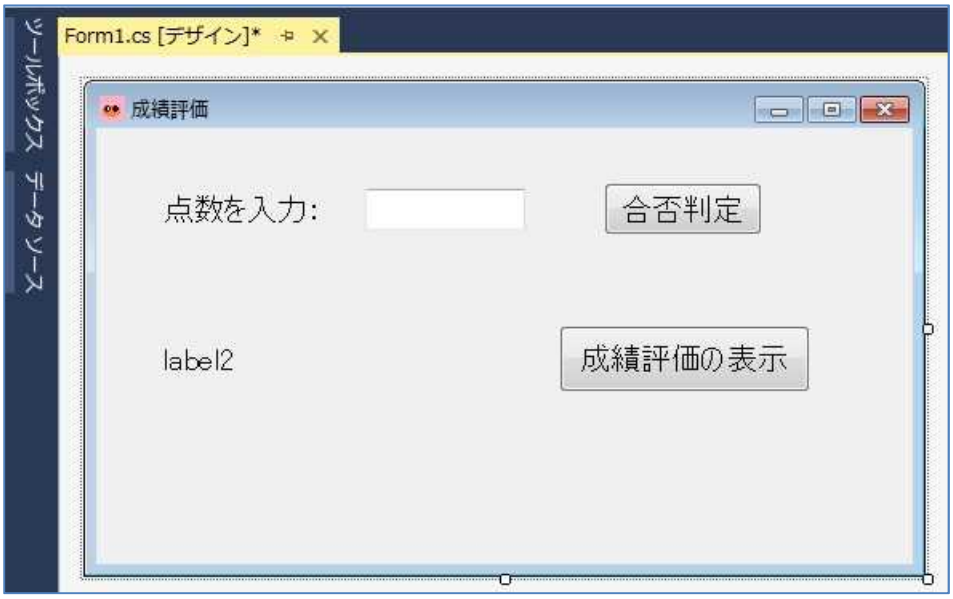


2）コントロールの配置及びフォームの作成

今後，フォーム上に配置するコントロールのプロパティのフォントサイズは全て 14 ポイントに変更するものとする。

Form1 上にラベルを 2 つ，テキストボックスを 1 つ，ボタンを 2 つ貼り付ける。それぞれのプロパティは次の様に設定する。

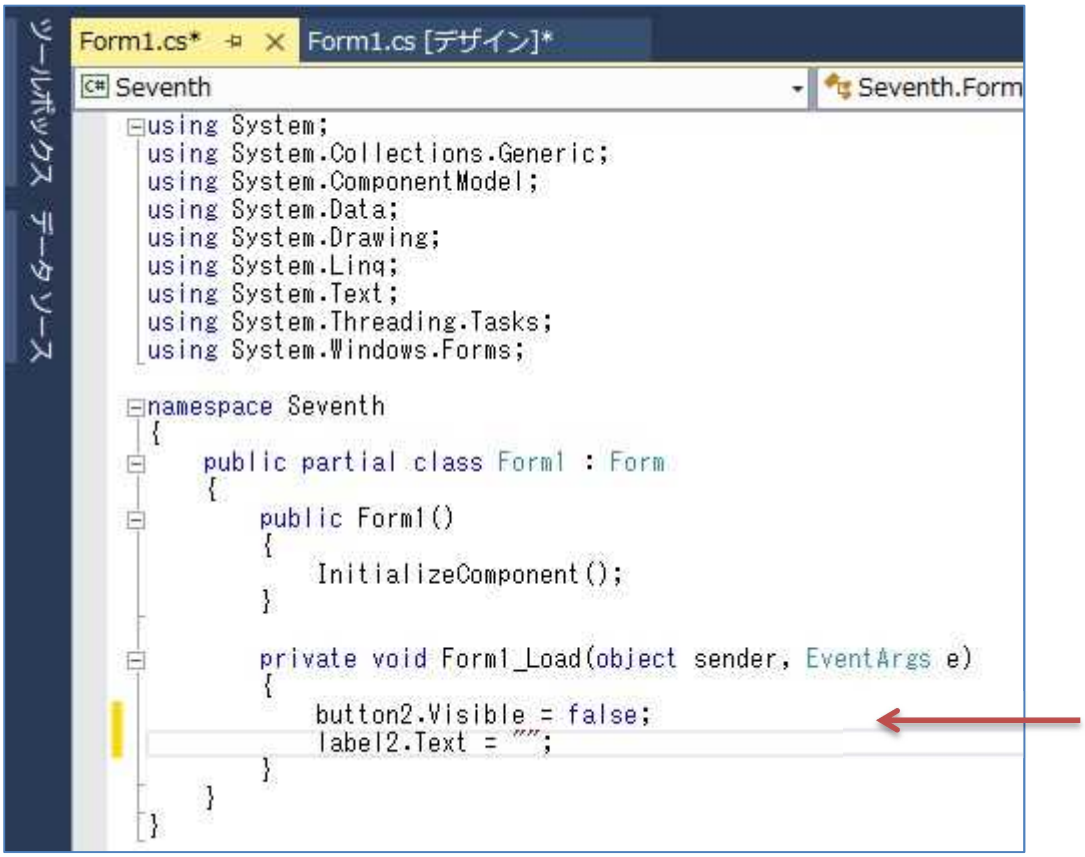
【Form1】Text　「**成績評価**」
　　Icon　自作のアイコン
【label1】Text　「**点数を入力：**」
【button1】Text　「**合否判定**」
【button2】Text　「**成績評価の表示**」



3）コーディング

Form1 のフォームデザイナー上でコントロールが貼られていない箇所をダブルクリックして Form1.cs のプログラムのソースコードを表示する。Form1_Load メソッドのブロック内に Form1 が読み込まれた際の処理として，『button2』の『Visible』プロパティに「false」を設定して非表示にする処理及び『label2』の『Text』プロパティに空の文字列（""）を設定する処理（赤枠の部分）を記述する。

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



フォームデザイナー上で『button1』をダブルクリックして，Form1.cs のプログラムのソースコードを表示する。先ず，クラス全体に適用可能な変数の宣言をクラスの冒頭に記述する。point はテキストボックスから入力された点数を格納しておく変数，s は『label2』の『Text』プロパティに設定すべき文字列を格納するための変数である。

```
int point;
string s;
```

次に，button1_Click メソッドのブロック内にボタンがクリックされた際の処理（赤枠の部分）を記述していく。

```
private void button1_Click(object sender, EventArgs e)
{
    point = Int32.Parse(textBox1.Text);

    if (point < 0 || point > 100)
    {
        s = "点数が範囲外です。¥n"
          + "0 点以上 100 点以下の点数を¥n 入力してください。";
    }
    else if (point < 60)
    {
        s = "不合格です。";
    }
    else
    {
        s = "合格です。¥n"
          + "成績評価の表示ボタンを¥n 押してください。";

        button2.Visible = true;
    }

    label2.Text = s;
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    point = Int32.Parse(textBox1.Text);

    if (point < 0 || point > 100)
    {
        s = "点数が範囲外です。¥n"
          + "0点以上100点以下の点数を¥n入力してください。";
    }
    else if (point < 60)
    {
        s = "不合格です。";
    }
    else
    {
        s = "合格です。¥n"
          + "成績評価の表示ボタンを¥n押してください。";

        button2.Visible = true;
    }

    label2.Text = s;
}
}
}
100 %
```

point < 0 || point > 100 は 2 つの条件式を条件 OR 演算子で組み合わせている。else if (point < 60) は else 節に if 文を入れ子にしたもので，単独の文なので{}を省略している（C#言語で書か

れたプログラムにはこの表記が多い)。最初の条件式の判定の結果，この else 節に到達するのは point の値が 0 以上 100 以下の場合である。最後の else 節は if (point < 60)の一部であり，point の値が 60 未満か 60 以上かで場合分けされている。なお，合格点の場合には，『button2』の『Visible』プロパティに「true」を設定してボタンを表示する。

更に，フォームデザイナー上で『button2』をダブルクリックして，コードにイベントハンドラを作成する。button2_Click メソッドのブロック内にボタンがクリックされた際の処理（赤枠の部分）を記述していく。

```
private void button2_Click(object sender, EventArgs e)
{
    string grade;

    if (point < 70)
    {
        grade = "可です。";
    }
    else if (point < 80)
    {
        grade = "良です。";
    }
    else if (point < 90)
    {
        grade = "優です。";
    }
    else
    {
        grade = "秀です。";
    }

    MessageBox.Show("評価は" + grade, "成績評価");
    button2.Visible = false;
}
```

ここでは，成績評価を表示する際に点数の範囲で場合分けされた文字列を格納するための変数 grade をローカルに宣言している。それに対して，point に関しては，クラスの先頭で宣言したので，button1_Click メソッド内で代入された値が保たれている。

MessageBox.Show("評価は" + grade, "成績評価")は指定した文字列を表示するメッセージボックスを元の Windows フォームとは別に表示する。MessageBox.Show("評価は" + grade)とした場合には，単純な文字列表示を行うメッセージボックスとなる。ここでの用法は，カンマ以降に記述した文字列をメッセージボックスのタイトルとして表示するものである。

なお，メッセージボックスが閉じられると，『button2』の『Visible』プロパティに「false」を設定してボタンを非表示として，次の入力に備えている。

（コードエディタの図は次のページ）

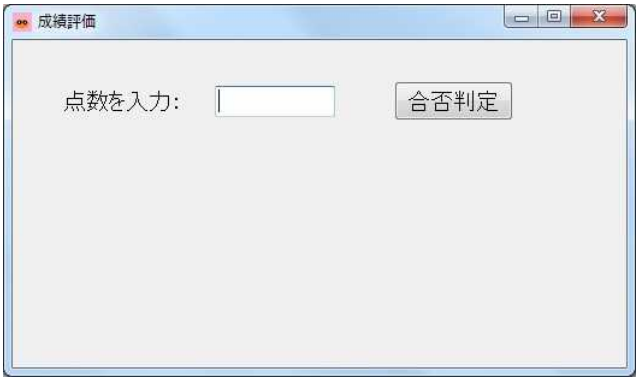
Form1.cs*　Form1.cs [デザイン]*

Seventh　Seventh.Form1

```
                s = "不合格です。";
            }
            else
            {
                s = "合格です。¥n"
                  + "成績評価の表示ボタンを¥n押してください。";

                button2.Visible = true;
            }

            label2.Text = s;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string grade;

            if (point < 70)
            {
                grade = "可です。";
            }
            else if (point < 80)
            {
                grade = "良です。";
            }
            else if (point < 90)
            {
                grade = "優です。";
            }
            else
            {
                grade = "秀です。";
            }

            MessageBox.Show("評価は" + grade, "成績評価");
            button2.Visible = false;
        }
    }
}
```

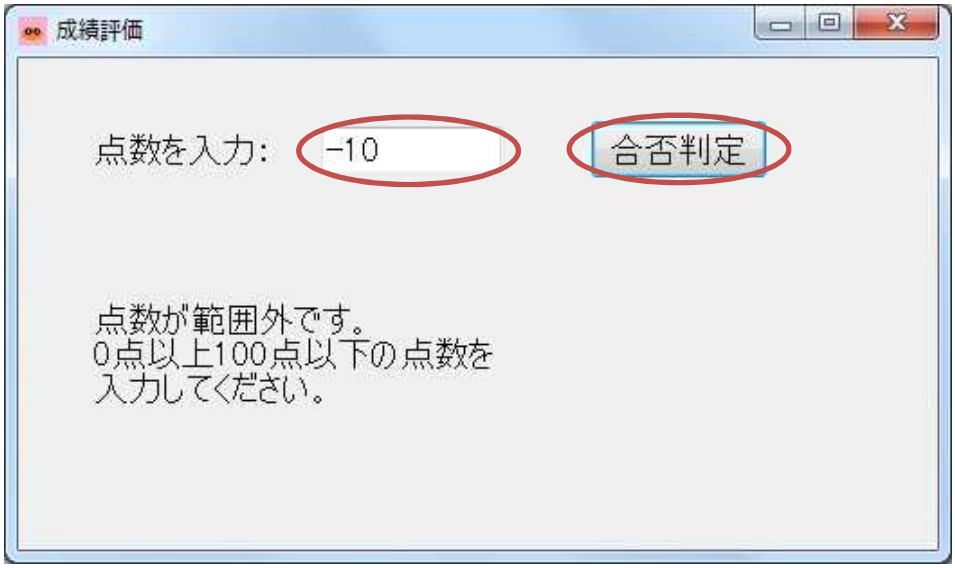
4）プログラムの実行・最終確認

『すべてを保存』ボタンを押してから，『開始』ボタンを押して，プログラムを実行する。エラーが出ている場合には，修正してから保存，開始と進む。

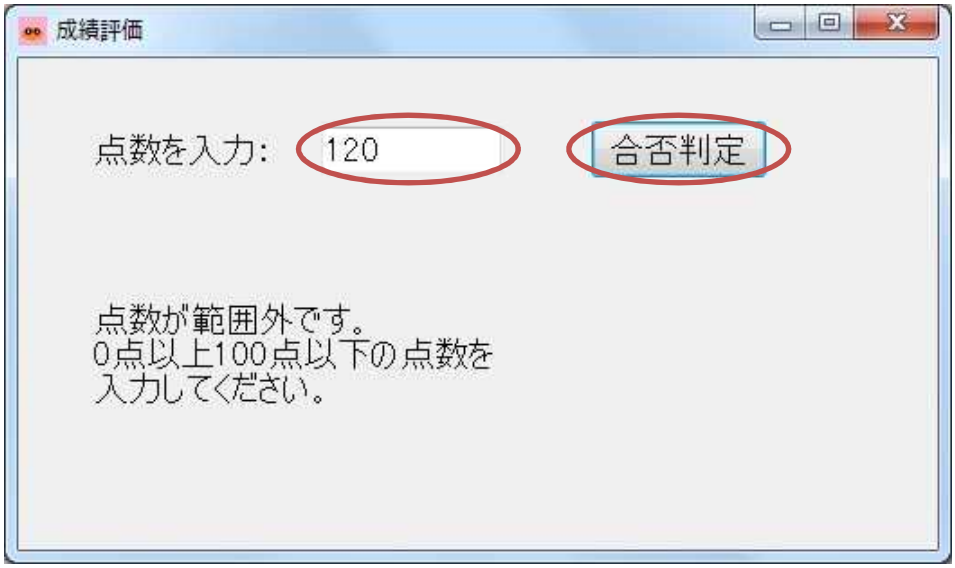


今回のプログラムでは場合分けがテーマなので，それぞれの分岐に行く道筋が正しく機能しているかどうか，一通り確かめる。

1）テキストボックスに負の整数を入力して，「合否判定」と表示されている button1 をクリックして，label2 を表示し，その内容を確かめる。

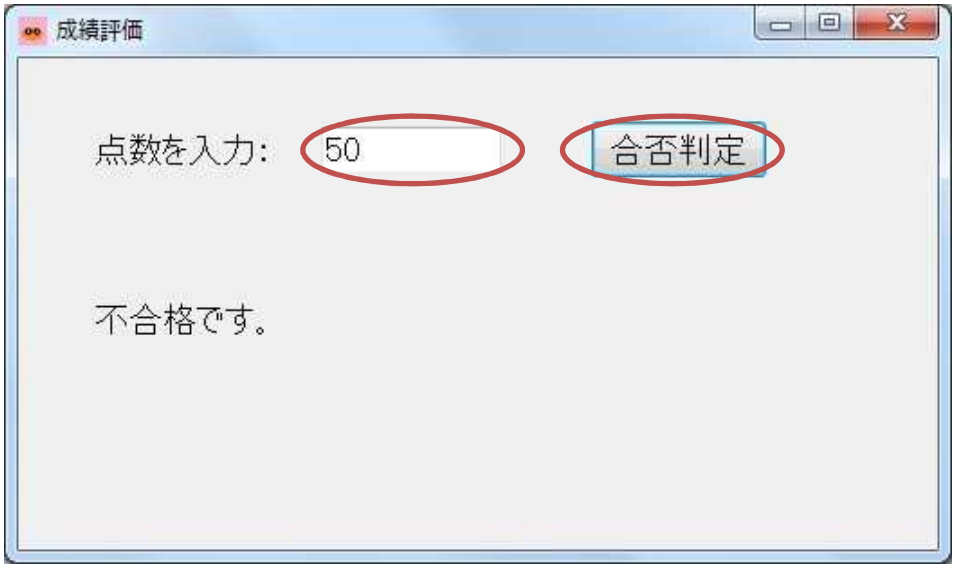


2）テキストボックスに 100 を超える整数を入力して，「合否判定」と表示されている button1 をクリックして，label2 を表示し，その内容を確かめる。

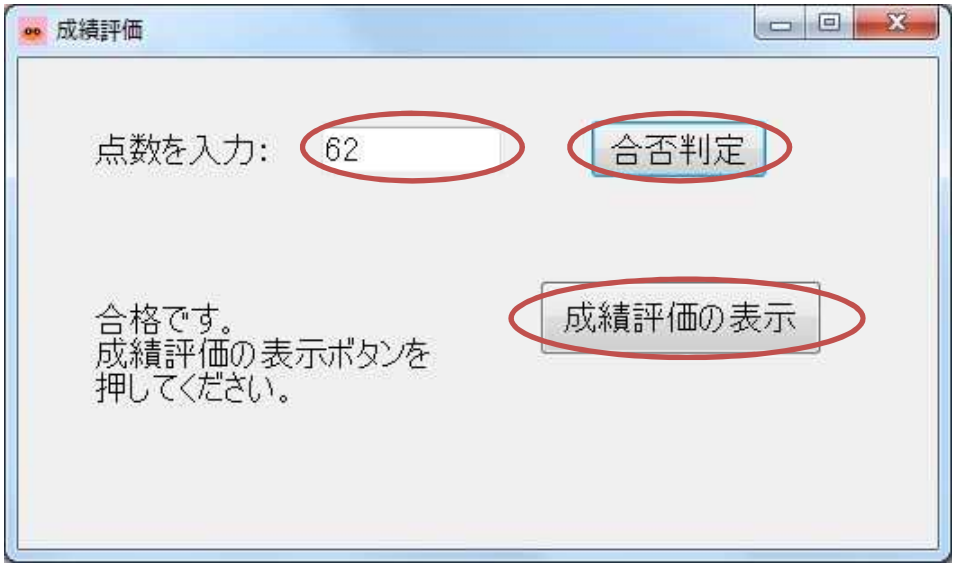


3）テキストボックスに 0 以上 60 未満の整数を入力して，「合否判定」と表示されている button1 をクリックして，label2 を表示し，その内容を確かめる。

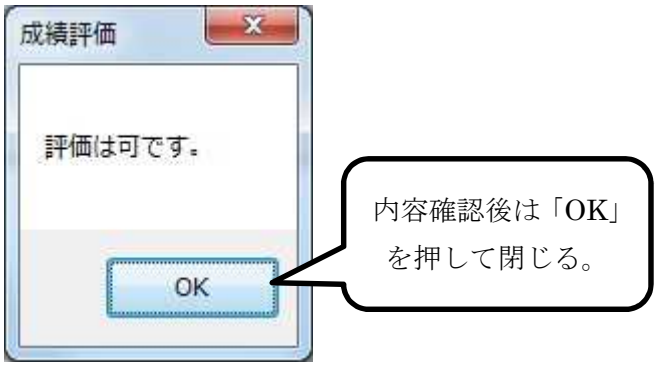
（図は次のページ）



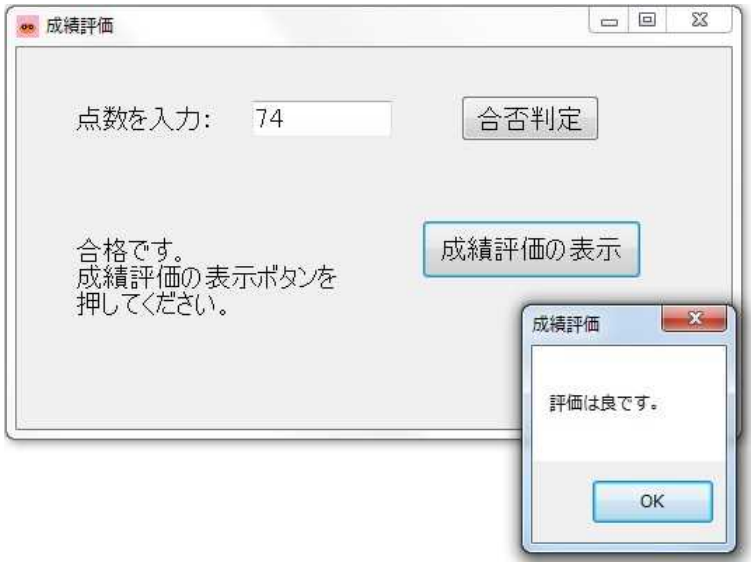
4）テキストボックスに 60 以上 69 以下の整数を入力して,「合否判定」と表示されている button1 をクリックして，label2 を表示し，その内容を確かめる。



また,「成績評価の表示」と表示されている button2 をクリックして，メッセージボックスを表示し，その内容を確かめる。



5）同様にして，他の分岐に到達するかどうかも確かめる。



確認を終えたら，プログラムを終了する。

【ファイルが保存されている場所】　H:¥Documents¥Visual Studio 2013¥Projects¥Seventh¥Seventh

提出物：
1）フォームのデザインファイル Form1.Designer.cs をメールに添付して提出する。
2）フォームを含むソースファイル Form1.cs をメールに添付して提出する。
3）質問を記述したファイル Questions_7th.txt に解答を書き込んで保存し，メールに添付して提出する。