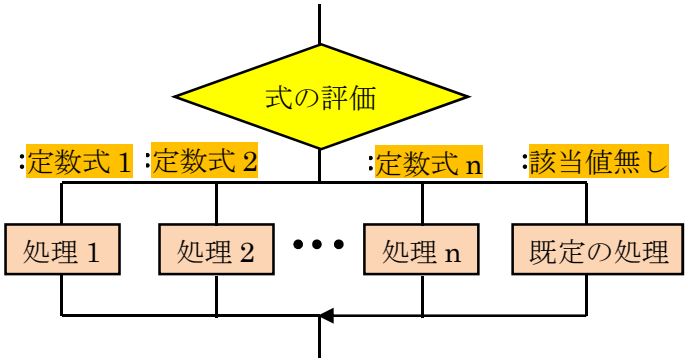
2019 年 6 月 6 日（木）実施

多分岐のプログラム

前回は多段階の２分岐を組み合わせて３種類以上の場合分けを実現したが，式の値の評価によって，一度に多種類の場合分けを行う**多分岐**の利用によって見通しのよいプログラムを作成出来る場合がある。（流れ図は右図）



switch 文

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

```
switch ( 制御式 ) {
    case 定数式 1: [[文 1-1 [文 1-2] ・・・ ] break;]
    case 定数式 2: [[文 2-1 [文 2-2] ・・・ ] break;]
    ・・・
    case 定数式 n: [[文 n-1 [文 n-2] ・・・ ] break;]
    [default: [既定文 1 [既定文 2] ・・・] break;]
}
```

ここで，[]内は省略可能である。switch 文の機能は，制御式を評価してその値が定数式（例えば，10，’a’の様なリテラルや 1+2+3 の様な値の定まった式）の何れかの値に一致したとき，その case ラベルに続く文を break 文に出会うまで実行する。break 文に到達すると switch 文から抜け出す。もし，一致した case ラベルに続く文及び break 文が省略されている場合には，その下の case ラベルに続く文を break 文に出会うまで実行する。

また，switch 文では，制御式を評価してその値が定数式の何れの値にも一致しないときは，default ラベルが書かれていれば，それに続く文を実行する。

Java 言語との違いは，default ラベルにも break 文を伴うこと，及び case ラベルに続く文がある時に break 文を省略して次の case ラベルに続く文も実行するという**フォールスルー**が禁じられていることである。

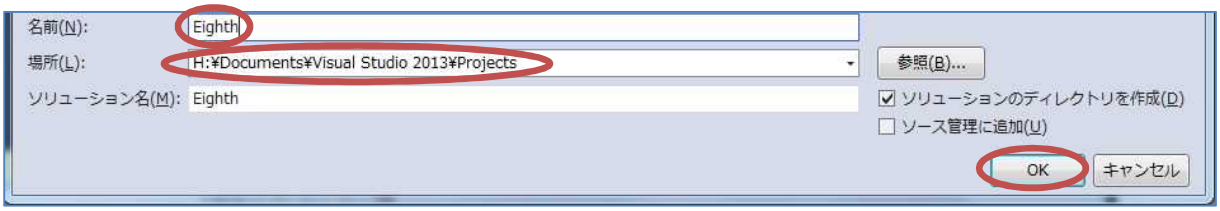
本日の課題

switch 文を利用して，成績評価の表示を場合分けするプログラム及びローマ字（母音）を平仮名に変換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「Eighth」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　「**多分岐**」
　　Icon　自作のアイコン
【button1】Text　「**成績評価**」
【button2】Text　「**ローマ字変換**」

［プロジェクト］→［Windows フォームの追加］を選択して Form2 を追加し（方法は第 6 回の教材を参照），ラベルを 1 つ，テキストボックスを 1 つ，ボタンを 1 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。

【Form2】Text　「**成績評価**」
　　Icon　自作のアイコン
【label1】Text　「**点数を入力：**」
【button1】Text　「**成績評価の表示**」

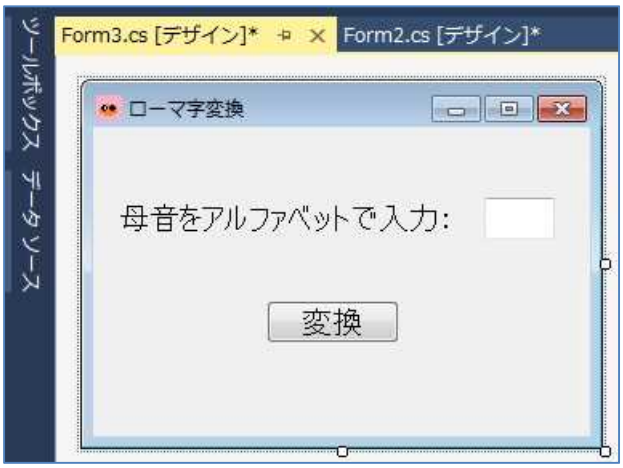
同様に，Form3 を追加し，ラベルを 1 つ，テキストボックスを 1 つ，ボタンを 1 つ貼り付ける。それぞれのプロパティは次の様に設定する。

【Form3】Text　「**ローマ字変換**」

　　Icon　自作のアイコン

【label1】Text　「**母音をアルファベットで入力**：」

【button1】Text　「**変換**」

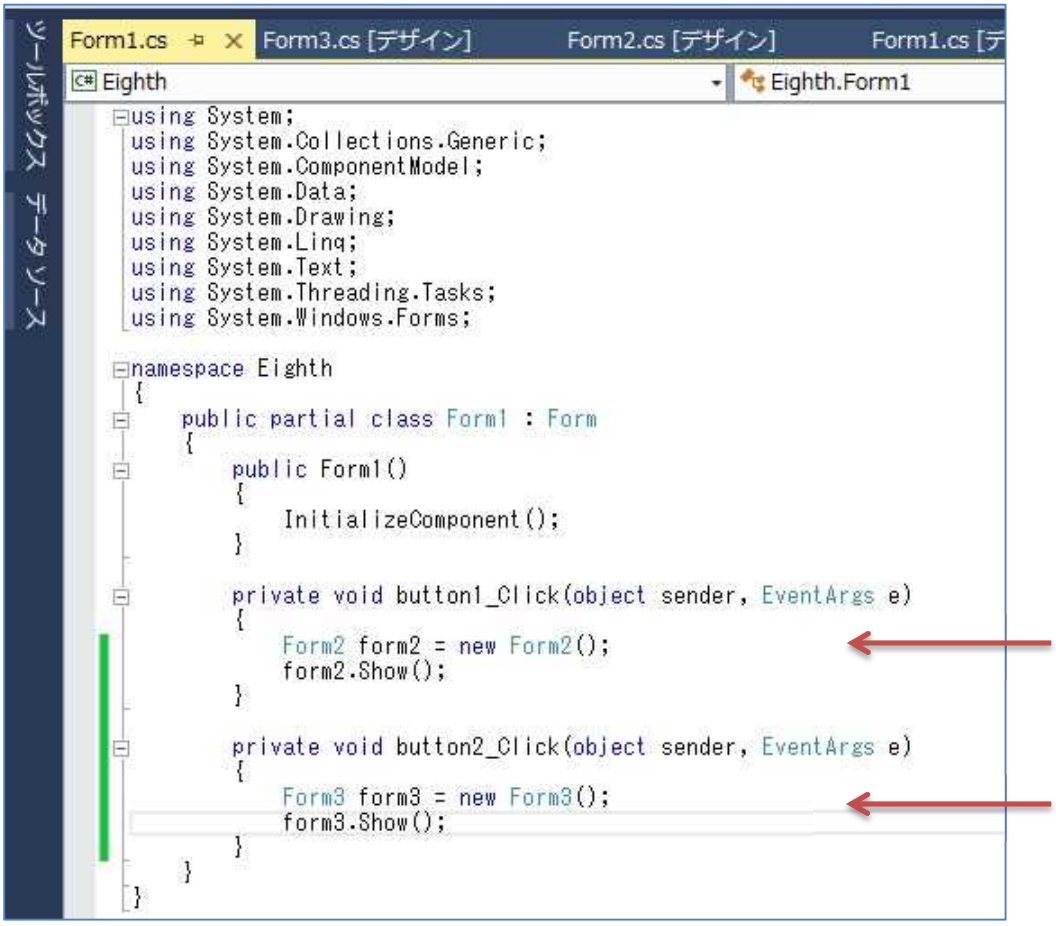


3) コーディング

Form1 の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。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 2 つのイベントハンドラを作成する。button1_Click メソッド及び button2_Click メソッドのブロック内にそれぞれの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)
        {
            Form2 form2 = new Form2();
            form2.Show();
        }
```

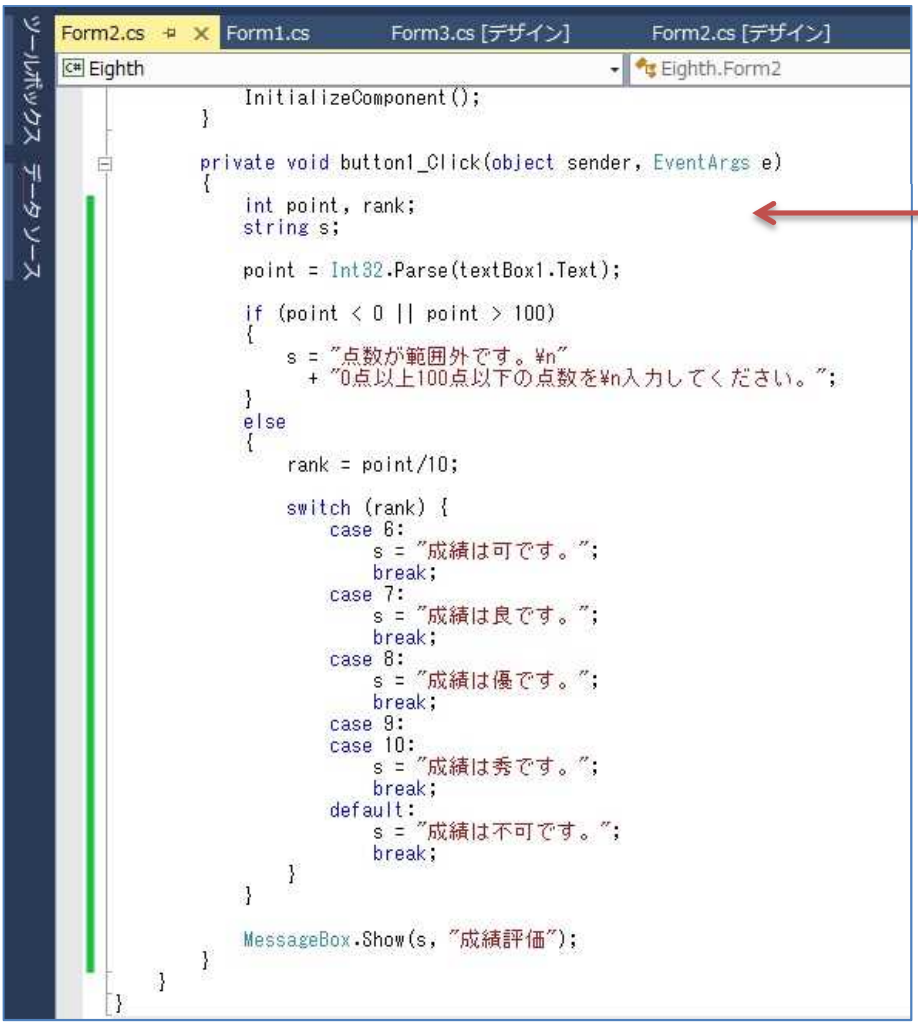
```
private void button2_Click(object sender, EventArgs e)
{
    Form3 form3 = new Form3();
    form3.Show();
}
```



Form2 のフォームデザイナー上で『button1』をダブルクリックして，Form2.cs のプログラムのソースコードを表示し，コードにイベントハンドラを作成する。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)
{
    int point, rank;
    string s;

    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
    {
```

```
        rank = point/10;

        switch (rank) {
            case 6:
                s = "成績は可です。";
                break;
            case 7:
                s = "成績は良です。";
                break;
            case 8:
                s = "成績は優です。";
                break;
            case 9:
            case 10:
                s = "成績は秀です。";
                break;
            default:
                s = "成績は不可です。";
                break;
        }
    }

    MessageBox.Show(s, "成績評価");
}
```

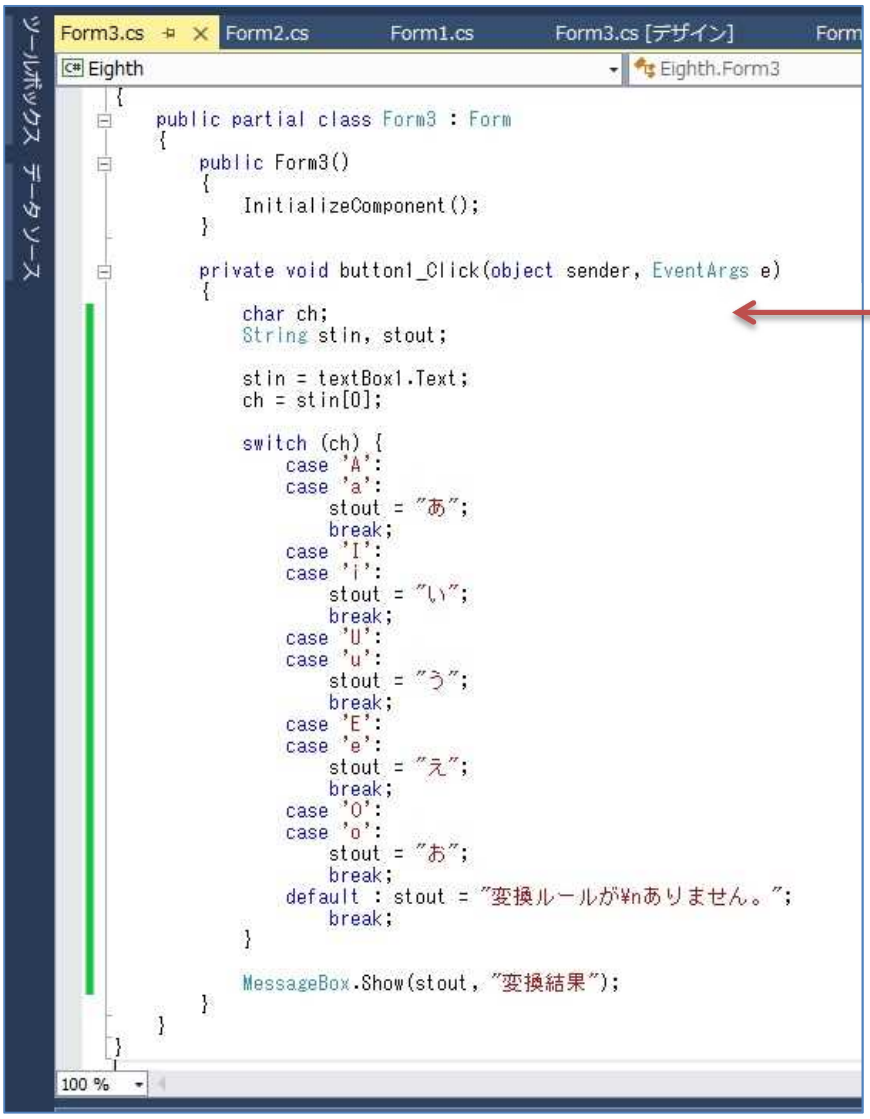


Form3 のフォームデザイナー上で『button1』をダブルクリックして，Form3.cs のプログラムのソースコードを表示し，コードにイベントハンドラを作成する。button1_Click メソッドのブロック内にボタンがクリックされた際の処理（赤枠の部分）を記述していく。

```
private void button1_Click(object sender, EventArgs e)
{
    char ch;
    String stin, stout;

    stin = textBox1.Text;
    ch = stin[0];

    switch (ch) {
        case 'A':
        case 'a':
            stout = "あ";
            break;
        case 'I':
        case 'i':
            stout = "い";
            break;
        case 'U':
        case 'u':
            stout = "う";
            break;
        case 'E':
        case 'e':
            stout = "え";
            break;
        case 'O':
        case 'o':
            stout = "お";
            break;
        default : stout = "変換ルールが¥n ありません。";
            break;
    }

    MessageBox.Show(stout, "変換結果");
}
```

stin[0]では，String クラスの**インデクサ**と呼ばれる，文字列の中のある位置の文字を指定する機能を利用して，先頭の文字を取り出している。

（コードエディタの図は次のページ）



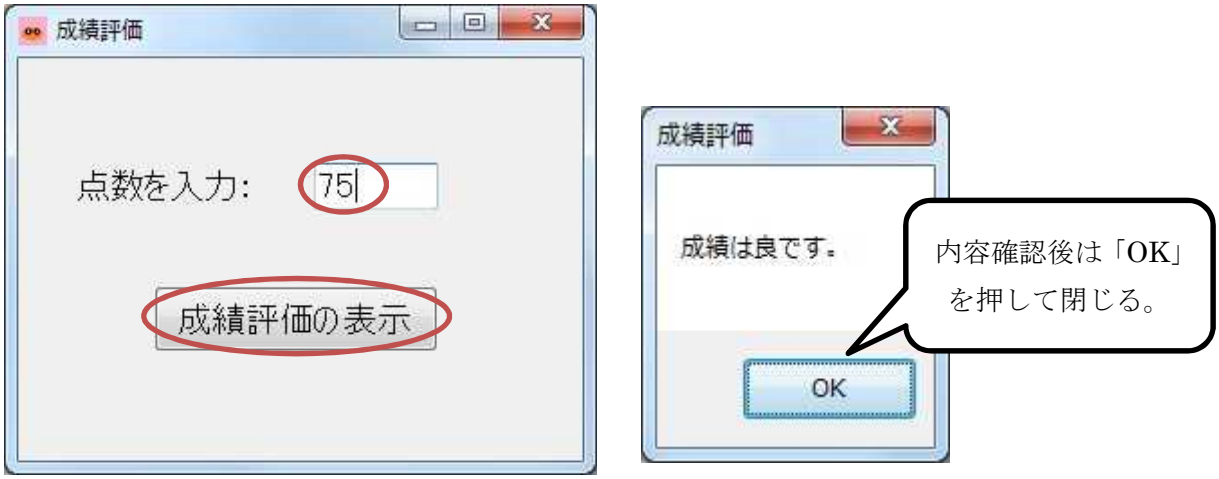

```
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            char ch;
            String stin, stout;

            stin = textBox1.Text;
            ch = stin[0];

            switch (ch) {
                case 'A':
                case 'a':
                    stout = "あ";
                    break;
                case 'I':
                case 'i':
                    stout = "い";
                    break;
                case 'U':
                case 'u':
                    stout = "う";
                    break;
                case 'E':
                case 'e':
                    stout = "え";
                    break;
                case 'O':
                case 'o':
                    stout = "お";
                    break;
                default : stout = "変換ルールが¥nありません。";
                    break;
            }

            MessageBox.Show(stout, "変換結果");
        }
    }
}
```

4) プログラムの実行・最終確認

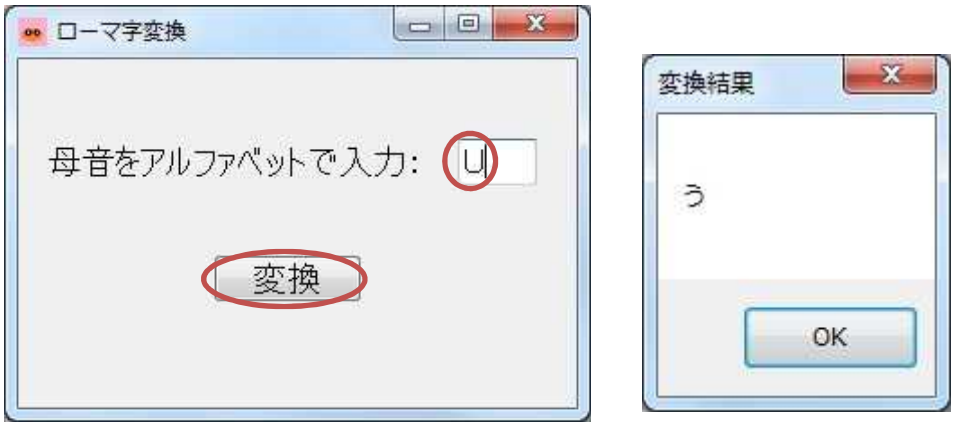
『すべてを保存』ボタンを押してから，『開始』ボタンを押して，プログラムを実行する。エラーが出ている場合には，修正してから保存，開始と進む。

先ず，成績評価の動作確認をする。

今回のプログラムでは前回の入れ子の if 文と同様に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。（図はその中の一例）



次に，ローマ字変換の動作確認をする。ここでも，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。（図はその中の一例）



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¥Eighth¥Eighth

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）フォームのデザインファイル Form2.Designer.cs をメールに添付して提出する。
4）フォームを含むソースファイル Form2.cs をメールに添付して提出する。
5）フォームのデザインファイル Form3.Designer.cs をメールに添付して提出する。
6）フォームを含むソースファイル Form3.cs をメールに添付して提出する。
7）質問を記述したファイル Questions_8th.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。