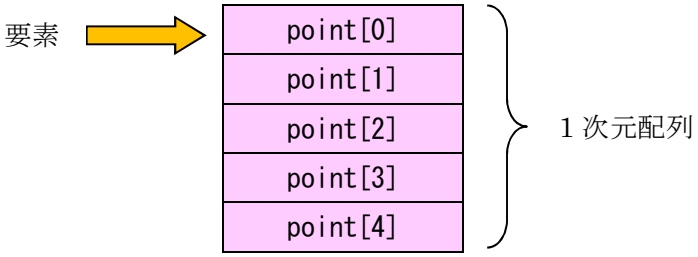
2019 年 6 月 13 日（木）実施

配列

同種のデータ型を有する複数のデータ（要素）を番号付けして，ひとまとまりの対象として扱うものを**配列**と呼ぶ。

1 次元配列

要素が 1 つの添え字（インデックス）で番号付けされるものを **1 次元配列**と呼ぶ。



1 次元配列の取り扱いに関して，次のような特徴がある。

1. プログラム中で用いる**配列変数**（配列の本体を参照する参照型の変数）は必ず**宣言**しておく。

   例）`int[] point;  /* int 型の配列変数 point を宣言 */`

2. 配列の本体を生成し，配列変数に代入して参照させる。

   例）`point = new int[5];  /* int 型の 5 個の要素を持つ配列を生成` ⇒ 要素数は配列が生成された時点で定まる `*/`

3. 配列変数を宣言する際に，配列の本体を生成することが出来る。

   例）`int[] point = new int[5];`

4. 配列の生成時に、各要素には初期値（値型の場合は 0，bool 型の場合は false，参照型の場合は null）が設定される。

5. 配列変数を宣言する際に，中括弧{ }の間に初期値を与えて初期化することが出来る。

   例）`int[] point = new int[] { 98, 76, 85, 67, 59 };` ⇒ この場合には，要素数は初期値の個数で決定される。右辺の `new int[]`を省略した書き方も許容される。

6. 配列の要素は 0 番から[要素数]－1 番までの添え字を用いて表される。

多次元配列

要素が 2 つ以上の添え字で番号付けされるものを**多次元配列**と呼ぶ。次の文は 1 つ目の添え字が 0，1，2，3 の範囲，2 つ目の添え字が 0，1 の範囲となる 2 次元配列の宣言の例である。

```
int[,] a = new int[4, 2];
```

また，次の文は，2 次元配列の初期化の例である。

```
int[,] a = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
```

同様に，3 次元配列の宣言及び初期化の例は次の様になる。

```
int[, ,] b = new int[2, 2, 3];
int[, ,] b = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } },
                           { { 7, 8, 9 }, { 10, 11, 12 } } }; /* 長いので折り返し */
```

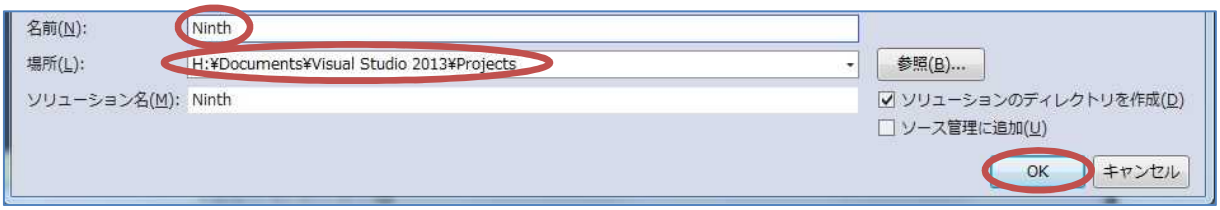
本日の課題

1 次元配列の取り扱いに関して，宣言，各要素への代入，各要素の参照，初期化を学ぶ。

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「Ninth」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）。

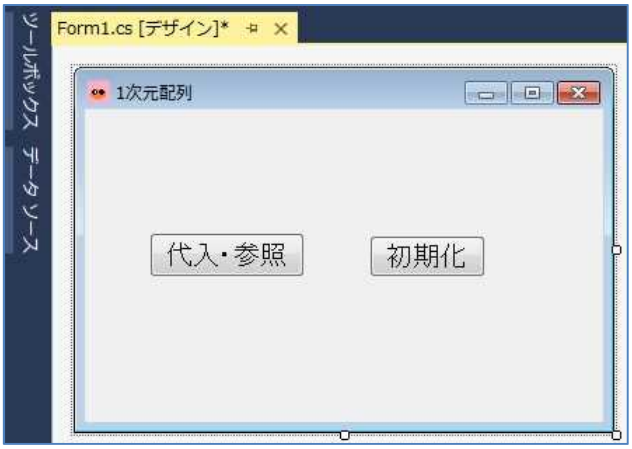


2) コントロールの配置及びフォームの作成

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　「1 次元配列」
　　Icon　自作のアイコン
【button1】Text　「代入・参照」
【button2】Text　「初期化」

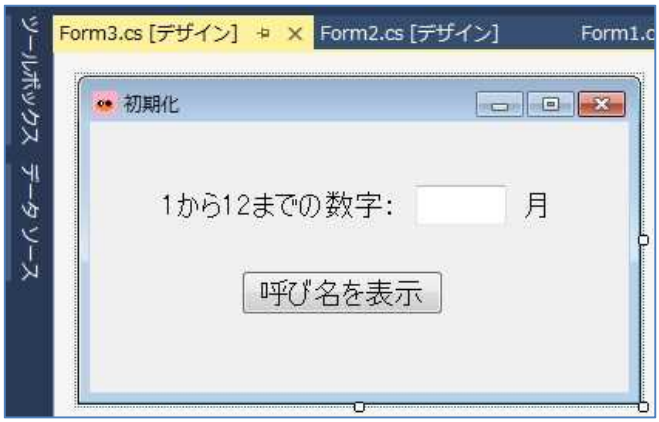


[プロジェクト] → [Windows フォームの追加]を選択して Form2 を追加し（方法は第 6 回の教材を参照），ラベルを 1 つ，テキストボックスを 1 つ，ボタンを 2 つ貼り付ける。それぞれのプロパティは次の様に設定する。

【Form2】Text　「**代入・参照**」
　　Icon　自作のアイコン
【button1】Text　「**入力**」
【button2】Text　「**結果表示**」

同様に，Form3 を追加し，ラベルを 2 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。

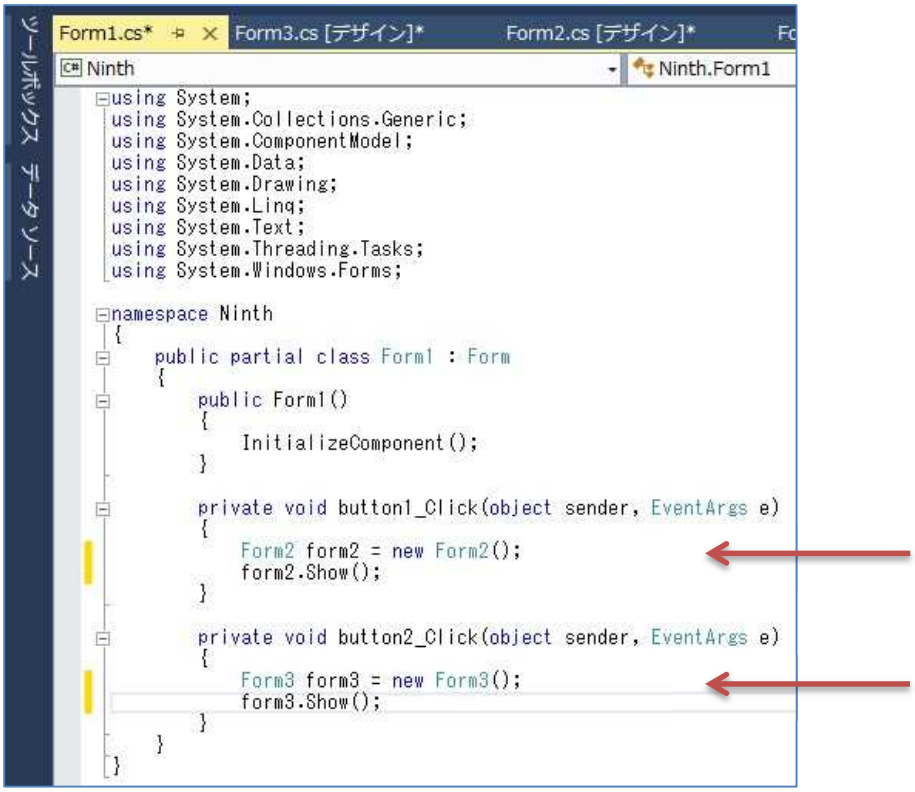
【Form3】Text　「**初期化**」
　　Icon　自作のアイコン
【label1】Text　「**1 から 12 までの数字：**」
【label2】Text　「**月**」
【button1】Text　「**呼び名を表示**」



3) コーディング

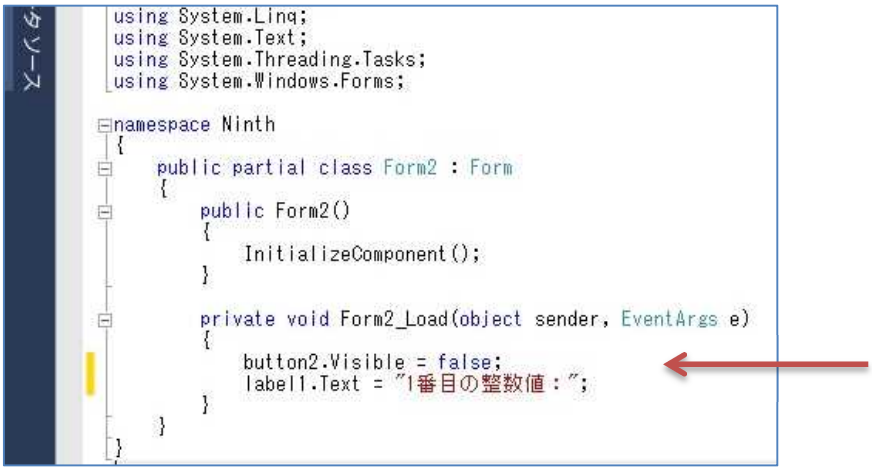
Form1 のフォームデザイナー上で『button1』をダブルクリックして，Form1.cs のプログラムのソースコードを表示する。更に，フォームデザイナー上で『button2』をダブルクリックして，コードに 2 つのイベントハンドラを作成する。button1_Click メソッド及び button2_Click メソッドのブロック内にそれぞれのボタンがクリックされた際の処理（赤枠の部分）を記述していく。

```
        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();
        }
```



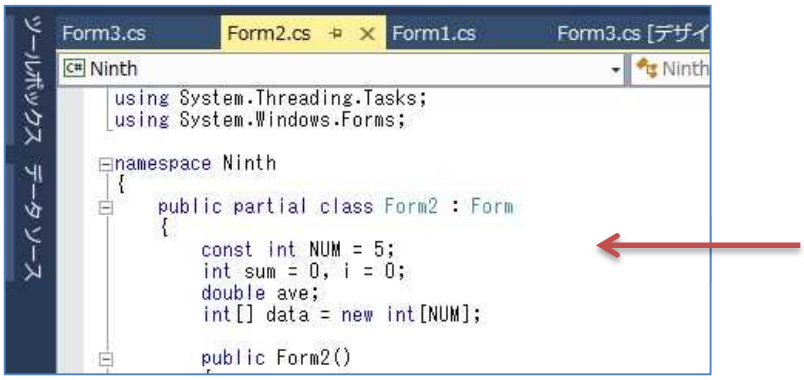
Form2 のフォームデザイナー上でコントロールが貼られていない箇所をダブルクリックして Form2.cs のプログラムのソースコードを表示する。Form2_Load メソッドのブロック内に Form2 が読み込まれた際の処理として，『button2』の『Visible』プロパティに「false」を設定して非表示にする処理及び『label1』の『Text』プロパティに初期の文字列を設定する処理（赤枠の部分）を記述する。

```
private void Form2_Load(object sender, EventArgs e)
{
    button2.Visible = false;
    label1.Text = "1 番目の整数値：";
}
```



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する。先ず，クラス全体に適用可能な定数，変数及び配列の宣言をクラスの冒頭に記述する。NUM は配列の要素数を表す定数（const キーワードを付けるとこの後値を変えられなくなる），sum はデータの合計を格納しておく変数，i は配列の添え字を変化させるための変数，ave は平均値を格納しておく変数，data はテキストボックスから入力された整数データを格納しておく配列変数である。

```
const int NUM = 5;
int sum = 0, i = 0;
double ave;
int[] data = new int[NUM];
```



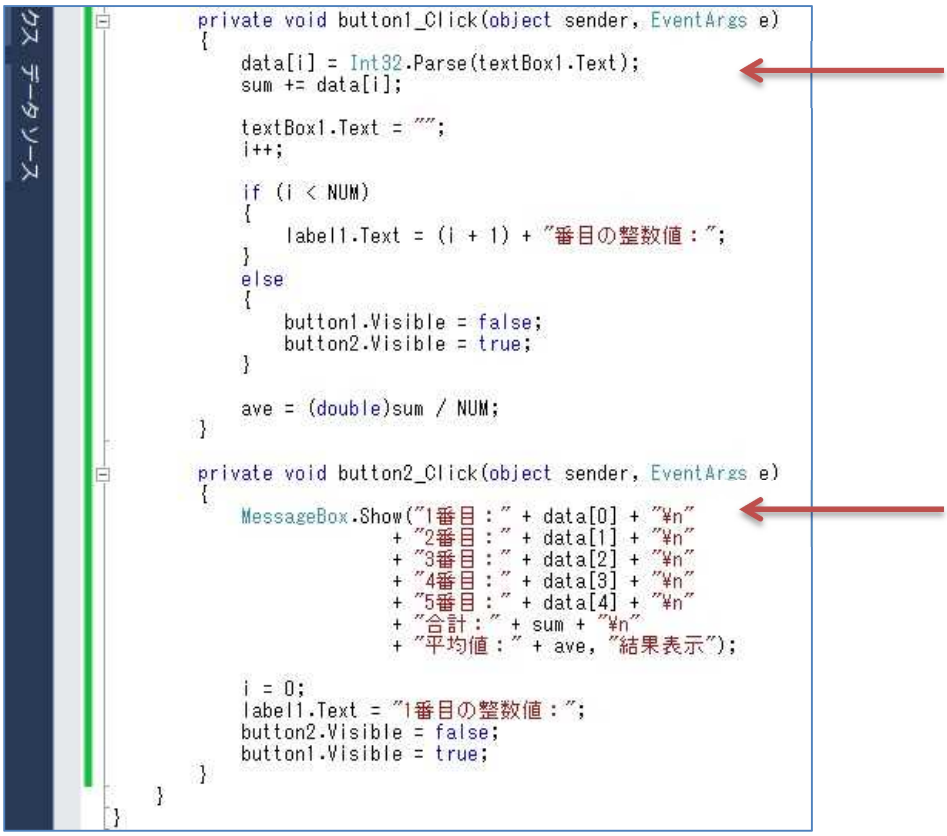
次に，コードにイベントハンドラを作成する。button1_Click メソッドのブロック内にボタンがクリックされた際の処理（赤枠の部分）を記述していく。

```
private void button1_Click(object sender, EventArgs e)
{
    data[i] = Int32.Parse(textBox1.Text);
    sum += data[i];

    textBox1.Text = "";
    i++;

    if (i < NUM)
    {
        label1.Text = (i + 1) + "番目の整数値：";
    }
    else
    {
        button1.Visible = false;
        button2.Visible = true;
    }

    ave = (double)sum / NUM;
}
```

同様に，Form2 の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。

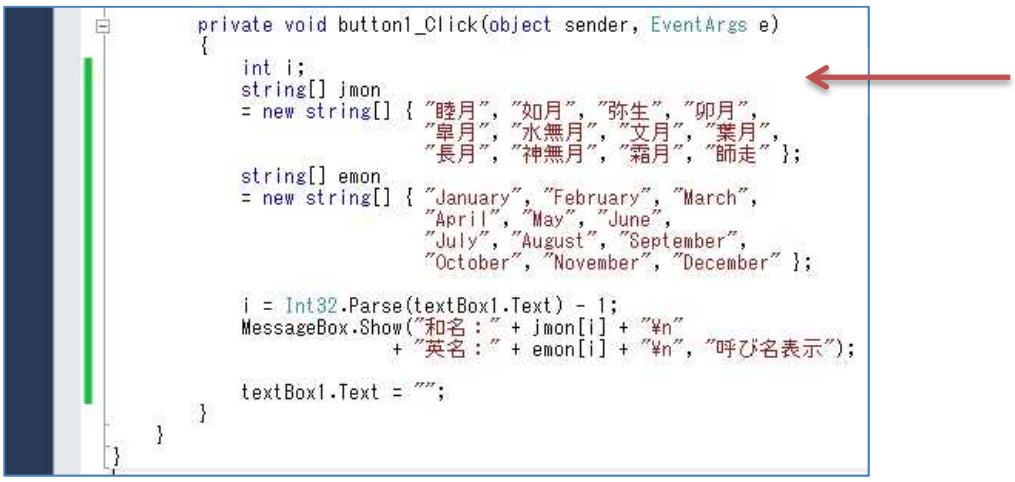
```
private void button2_Click(object sender, EventArgs e)
{
    MessageBox.Show("1 番目 : " + data[0] + "¥n"
                  + "2 番目 : " + data[1] + "¥n"
                  + "3 番目 : " + data[2] + "¥n"
                  + "4 番目 : " + data[3] + "¥n"
                  + "5 番目 : " + data[4] + "¥n"
                  + "合計 : " + sum + "¥n"
                  + "平均値 : " + ave, "結果表示");

    i = 0;
    label1.Text = "1 番目の整数値 : ";
    button2.Visible = false;
    button1.Visible = true;
}
```



```
private void button1_Click(object sender, EventArgs e)
{
    data[i] = Int32.Parse(textBox1.Text);
    sum += data[i];

    textBox1.Text = "";
    i++;

    if (i < NUM)
    {
        label1.Text = (i + 1) + "番目の整数値 : ";
    }
    else
    {
        button1.Visible = false;
        button2.Visible = true;
    }

    ave = (double)sum / NUM;
}

private void button2_Click(object sender, EventArgs e)
{
    MessageBox.Show("1番目 : " + data[0] + "¥n"
                  + "2番目 : " + data[1] + "¥n"
                  + "3番目 : " + data[2] + "¥n"
                  + "4番目 : " + data[3] + "¥n"
                  + "5番目 : " + data[4] + "¥n"
                  + "合計 : " + sum + "¥n"
                  + "平均値 : " + ave, "結果表示");

    i = 0;
    label1.Text = "1番目の整数値 : ";
    button2.Visible = false;
    button1.Visible = true;
}
}
}
```

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。

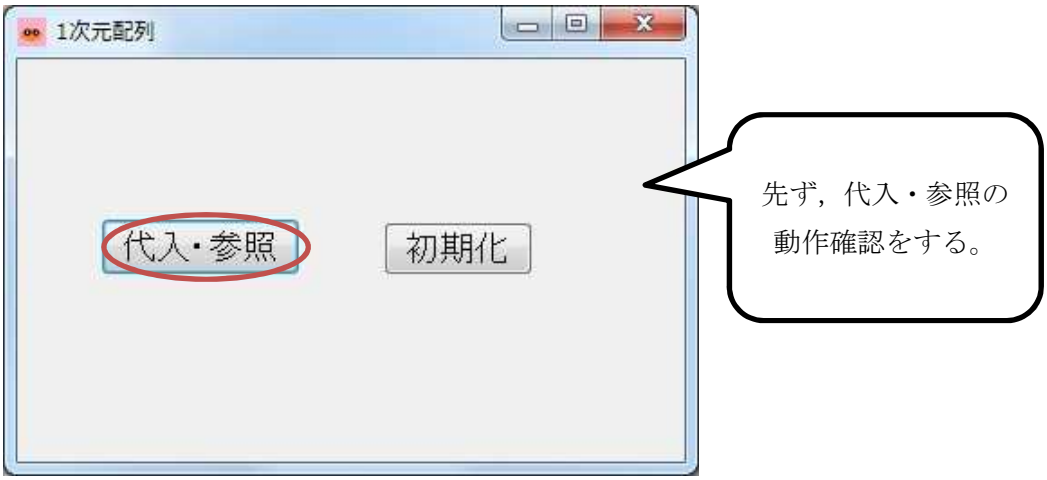
```
private void button1_Click(object sender, EventArgs e)
{
    int i;
    string[] jmon
    = new string[] { "睦月", "如月", "弥生", "卯月",
                     "皐月", "水無月", "文月", "葉月",
                     "長月", "神無月", "霜月", "師走" };
    string[] emon
    = new string[] { "January", "February", "March",
                     "April", "May", "June",
                     "July", "August", "September",
                     "October", "November", "December" };

    i = Int32.Parse(textBox1.Text) - 1;
    MessageBox.Show("和名：" + jmon[i] + "¥n"
                  + "英名：" + emon[i] + "¥n", "呼び名表示");

    textBox1.Text = "";
}
```



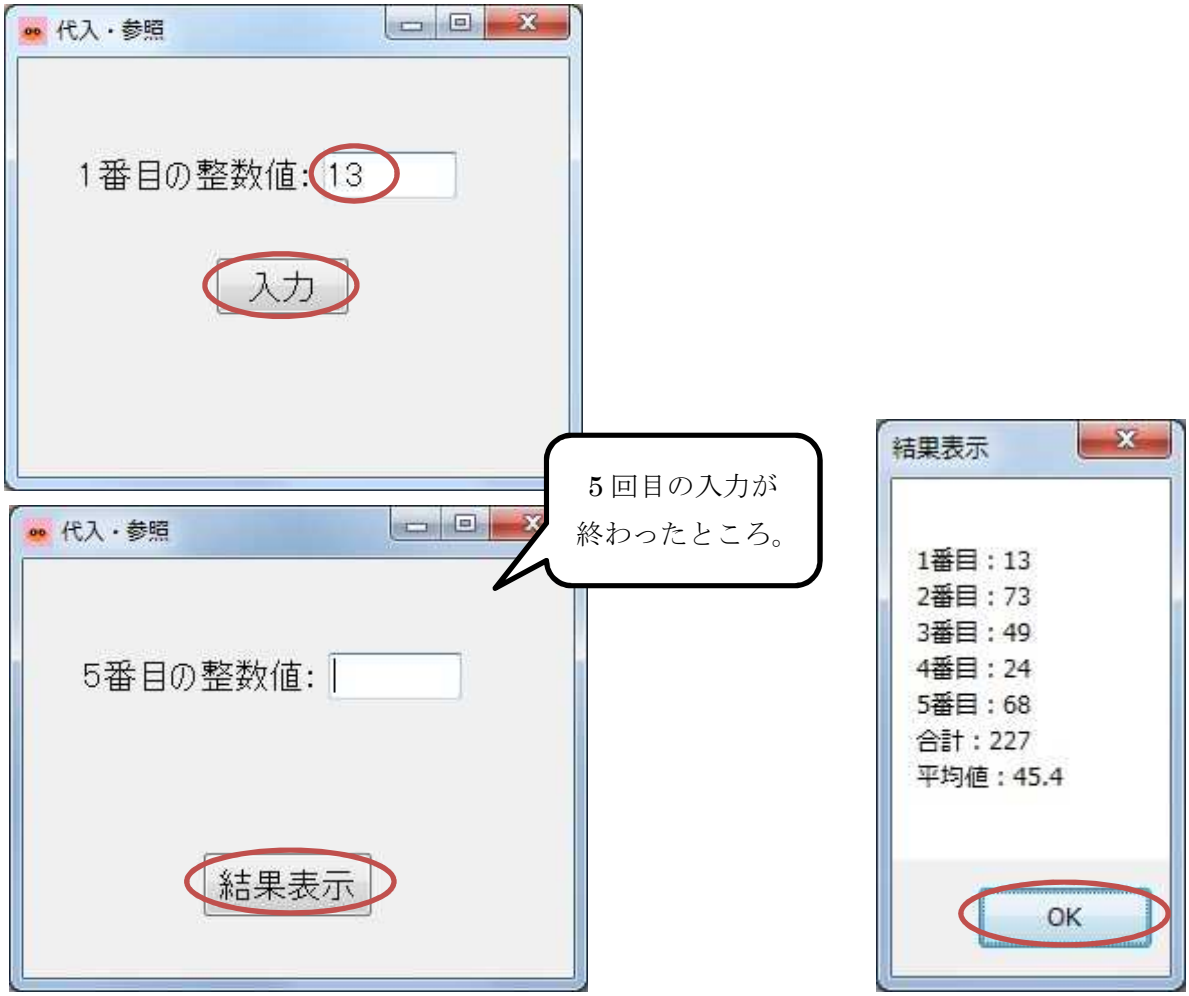
```
        private void button1_Click(object sender, EventArgs e)
        {
            int i;
            string[] jmon
            = new string[] { "睦月", "如月", "弥生", "卯月",
                             "皐月", "水無月", "文月", "葉月",
                             "長月", "神無月", "霜月", "師走" };
            string[] emon
            = new string[] { "January", "February", "March",
                             "April", "May", "June",
                             "July", "August", "September",
                             "October", "November", "December" };

            i = Int32.Parse(textBox1.Text) - 1;
            MessageBox.Show("和名：" + jmon[i] + "¥n"
                          + "英名：" + emon[i] + "¥n", "呼び名表示");

            textBox1.Text = "";
        }
    }
}
```

4) プログラムの実行・最終確認

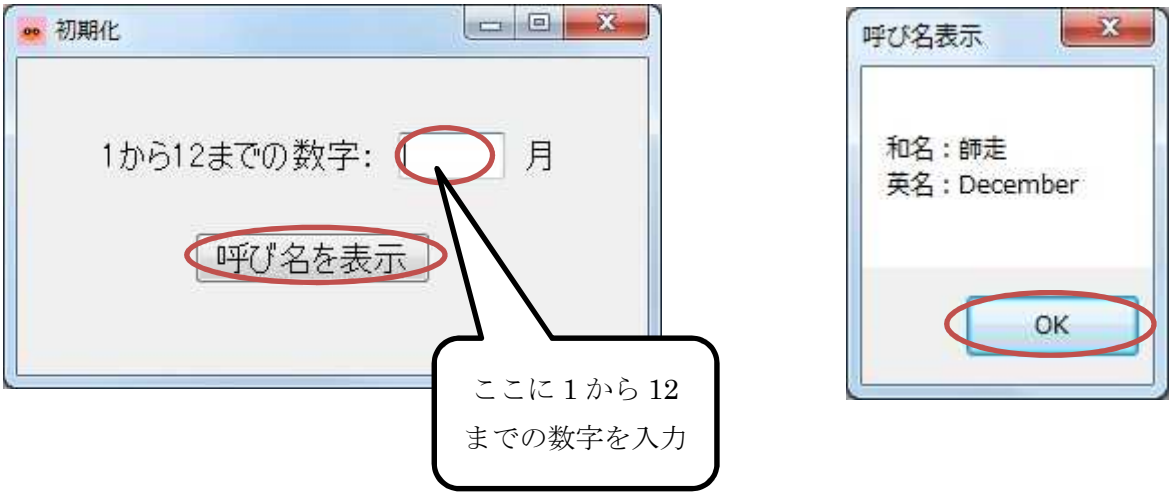
『すべてを保存』ボタンを押してから，『開始』ボタンを押して，プログラムを実行する。エラーが出ている場合には，修正してから保存，開始と進む。



今回のプログラムでは，整数値をテキストボックスに入力して入力ボタンを押すという操作を 5 回繰り返すと結果表示のボタンが現れる。(図はその一例)



次に，初期化の動作確認をする。ここでは，1 から 12 までの数字を入力してそれぞれ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¥Ninth¥Ninth

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_9th.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。