2018 年 11 月 29 日（木）実施

反復構造のプログラム

1)　0 回以上の繰り返しのプログラム

for 文

C#言語で 0 回以上の繰り返しのプログラムを実現するための文として，for 文と while 文とが用意されている。for 文の構文は次の様になる。

```
for（ 初期化処理; 継続条件式; 更新処理 ）{ 文 }
```

先ず，初期化処理を実行し，続いて継続条件式を評価する。ここで，継続条件式が true であれば，文を実行する。次に更新処理を実行した上で，再度，継続条件式を評価する，という繰り返しを行う。継続条件式が true でなければ，文を実行せず，for 文から抜け出す。最初から継続条件式が true でなければ，全く文を実行しないので，0 回以上の繰り返しと呼ばれる。

for 文は通常，**特定の処理を決まった回数繰り返す目的**で用いられる。

＊ for 文の括弧の中の 3 つのセクションには，それぞれ，順に initializer，condition，iterator という呼び名が付けられている。

while 文

while 文の構文は次の様になる。

```
while（ 継続条件式 ）{ 文 }
```

先ず，継続条件式を評価し，true であれば文を実行し，再度，継続条件式を評価する，という繰り返しを行う。継続条件式が true でなければ，文を実行せず，while 文から抜け出す。最初から継続条件式が true でなければ，全く文を実行しないので，0 回以上の繰り返しと呼ばれる。

while 文は通常，**処理を特定の状態になるまで繰り返す目的**で用いられる。

2)　1 回以上の繰り返しのプログラム

do 文

C#言語で 1 回以上の繰り返しのプログラムを実現するための文として，do 文が用意されている。do 文の構文は次の様になる。

```
do { 文 } while（ 継続条件式 ）;
```

先ず文を実行し，続いて継続条件式を評価し，true であれば再度文を実行し，継続条件式を評価する，という繰り返しを行う。継続条件式が true でなくなれば，文を実行せず，do 文から抜け出す。最初から継続条件式が true でなくとも，先ず文を実行するので，1 回以上の繰り返しと呼ばれる。

do 文は通常，**ある処理を実施し，その処理を特定の状態になるまで繰り返す目的**で用いられる。

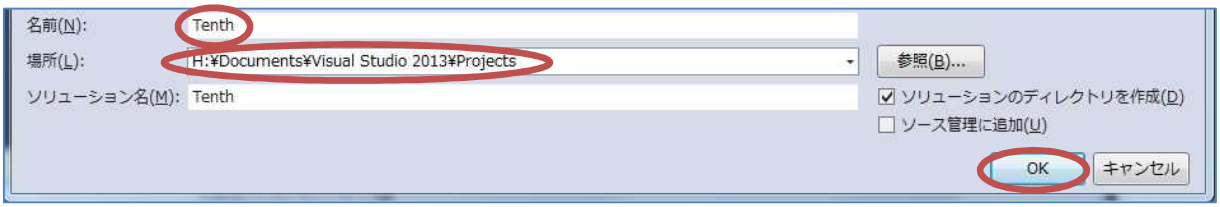
本日の課題

反復構造のプログラムとして，0 回以上の繰り返しのプログラム，及び 1 回以上の繰り返しのプログラムの特徴を学ぶ。

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「Tenth」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 1 つ，テキストボックスを 1 つ，ボタンを 3 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　「while 文」
　　Icon　自作のアイコン
【label1】Text　「お年はお幾つですか？」
【button1】Text　「入力」
【button2】Text　「for 文」
【button3】Text　「do 文」

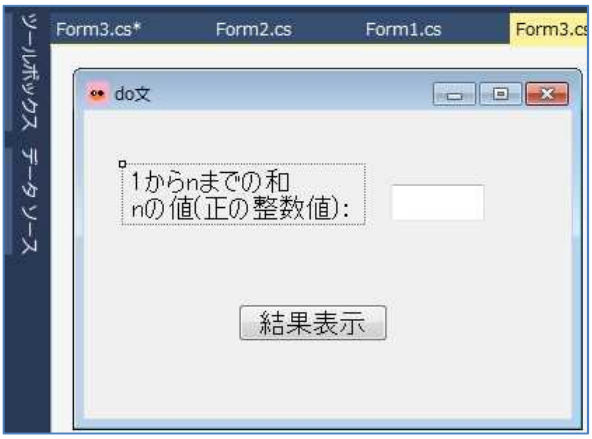
［プロジェクト］→［Windows フォームの追加］を選択して Form2 を追加し（方法は第 6 回の教材を参照），ラベルを 1 つ，テキストボックスを 1 つ，ボタンを 2 つ貼り付ける。それぞれのプロパティは次の様に設定する。

【Form2】Text　「for 文」
　　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 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 「**do 文**」
　　Icon　自作のアイコン
【label1】Text 「**1 から n までの和**
　　n の値（正の整数値）：」
【button1】Text 「**結果表示**」

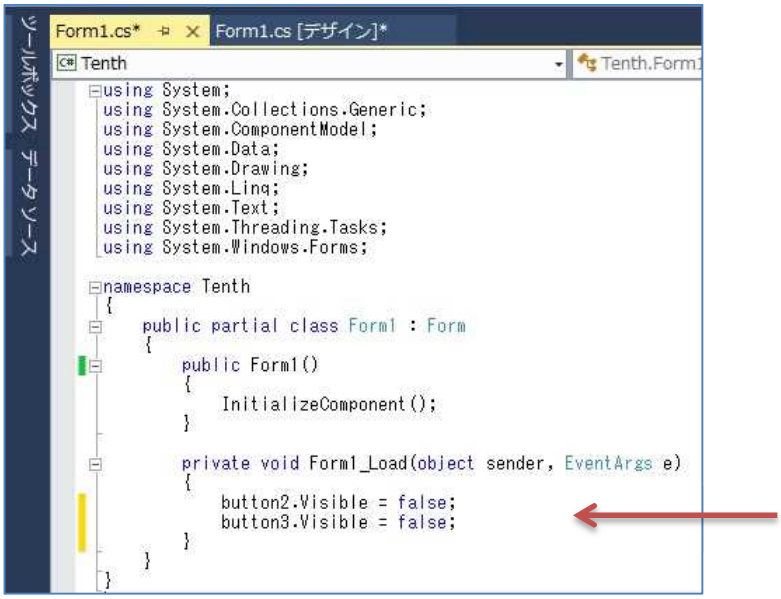


＊ ラベルの Text プロパティで改行する方法については，第 4 回の教材の p.5 を参照。

3）コーディング

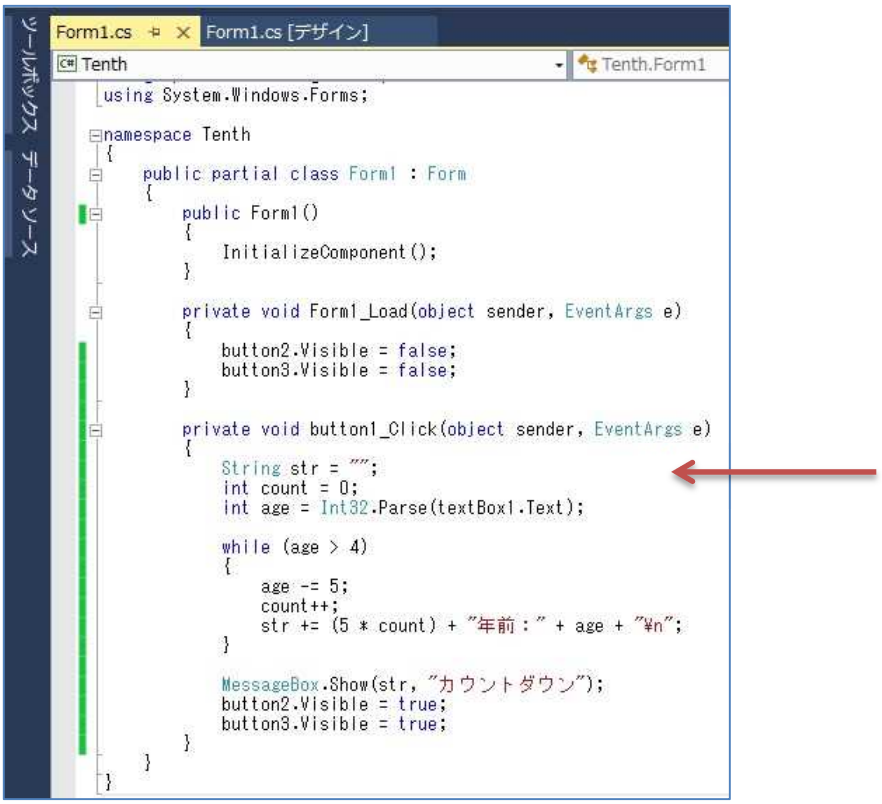
Form1 のフォームデザイナー上でコントロールが貼られていない箇所をダブルクリックして Form1.cs のプログラムのソースコードを表示する。Form1_Load メソッドのブロック内に Form1 が読み込まれた際の処理として，『button2』及び『button3』の『Visible』プロパティに「false」を設定して非表示にする処理（赤枠の部分）を記述する。

```
private void Form1_Load(object sender, EventArgs e)
{
    button2.Visible = false;
    button3.Visible = false;
}
```



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。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。ここで，str はメッセージボックスに表示する文字列，count は繰り返しの数を数えるカウンタ，age は初期値としてテキストボックスに入力された年齢を設定し，繰り返しの度にそこから 5 ずつ減らしていくための変数である。

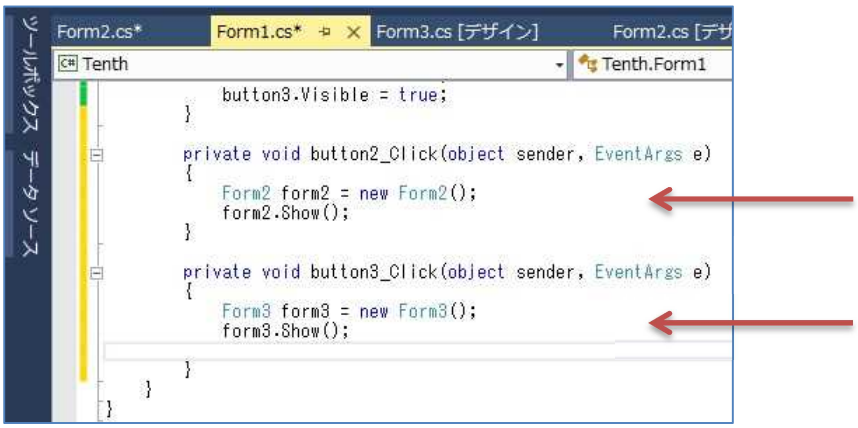
```
private void button1_Click(object sender, EventArgs e)
{
    String str = "";
    int count = 0;
    int age = Int32.Parse(textBox1.Text);

    while (age > 4)
    {
        age -= 5;
        count++;
        str += (5 * count) + "年前：" + age + "¥n";
    }

    MessageBox.Show(str, "カウントダウン");
    button2.Visible = true;
    button3.Visible = true;
}
```



更に，フォームデザイナー上で『button2』及び『button3』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。button2_Click メソッド及び button3_Click メソッドのブロ

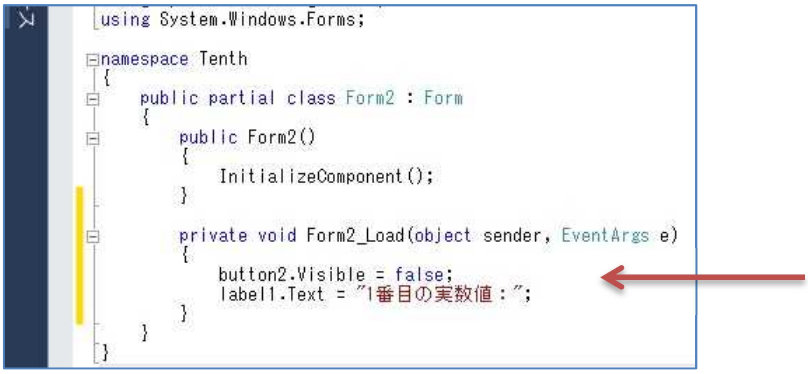
ック内にそれぞれのボタンがクリックされた際の処理（赤枠の部分）を記述していく。

```
private void button2_Click(object sender, EventArgs e)
{
    Form2 form2 = new Form2();
    form2.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    Form3 form3 = new Form3();
    form3.Show();
}
```



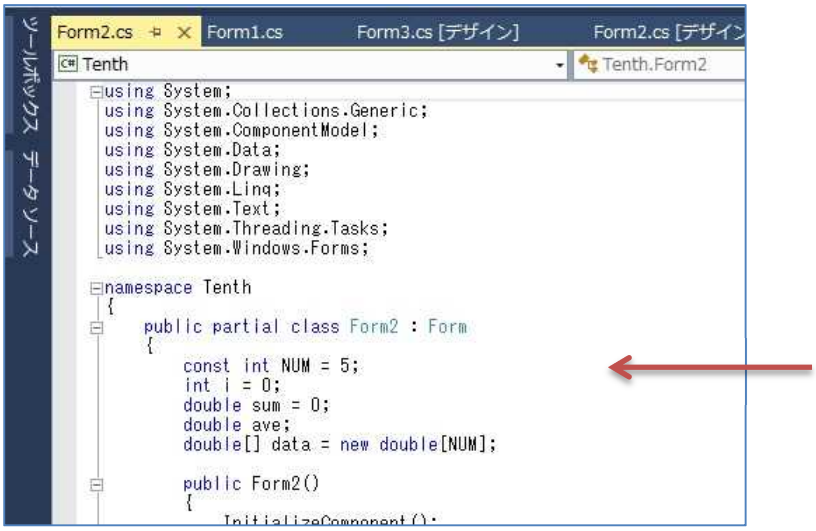
Form2 のフォームデザイナー上でコントロールが貼られていない箇所をダブルクリックして Form2.cs のプログラムのソースコードを表示する。Form2_Load メソッドのブロック内に Form2 が読み込まれた際の処理として，『button2』の『Visible』プロパティに「false」を設定して非表示にする処理及び『label1』の『Text』プロパティに初期の文字列を設定する処理（赤枠の部分）を記述する。

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



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，i，sum，ave，data の役割は前回の Form2.cs のプログラムと同様であるが，sum 及び data のデータ型を今回のプログラムでは double 型としている。

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

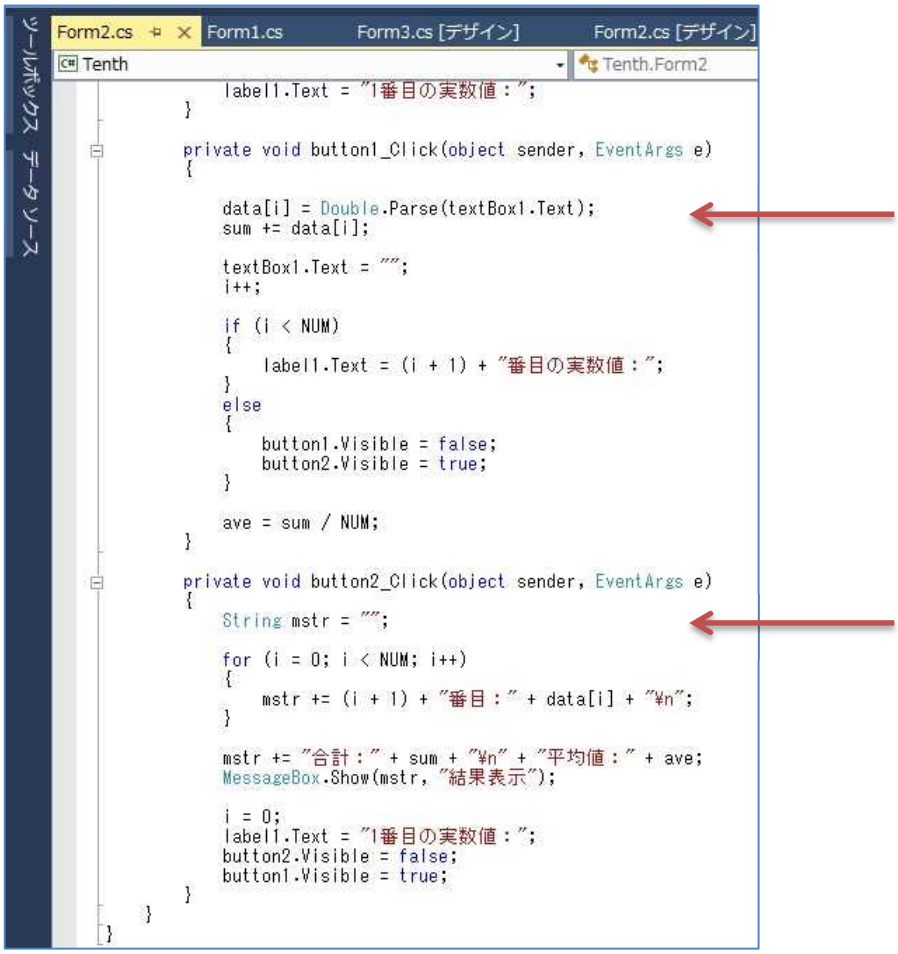


次に，コードにイベントハンドラを作成する。button1_Click メソッドのブロック内にボタンがクリックされた際の処理（赤枠の部分）を記述していく。

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

    textBox1.Text = "";
    i++;

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

    ave = sum / NUM;
}
```

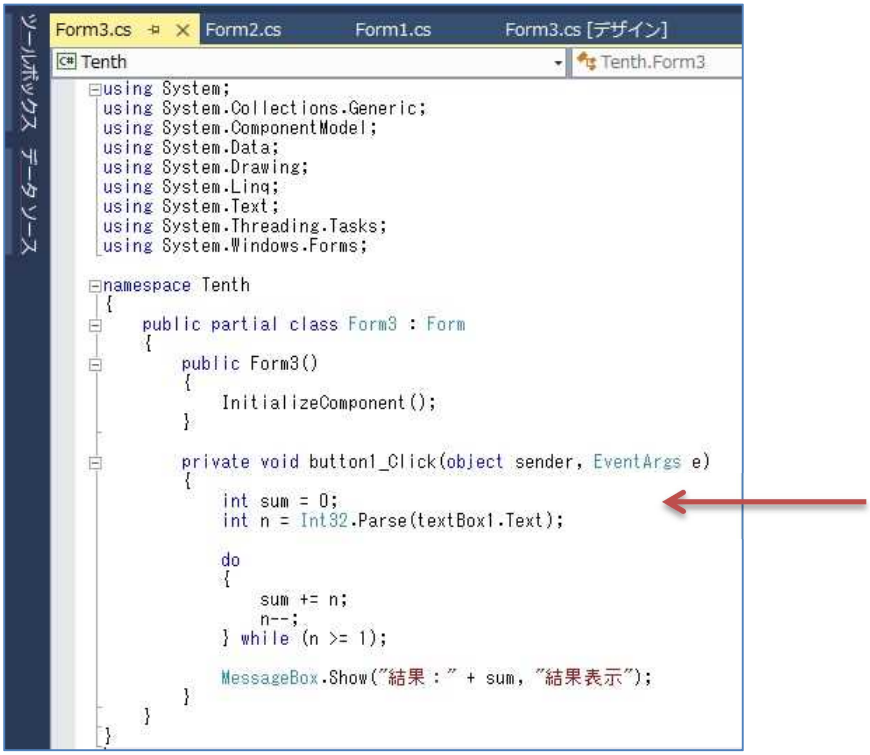
同様に，Form2 のフォームデザイナー上で『button2』をダブルクリックして，コードにイベントハンドラを作成する。button2_Click メソッドのブロック内にボタンがクリックされた際の処理（赤枠の部分）を記述していく。ここで，mstr はメッセージボックスに表示する文字列を格納しておくための変数である。

```
private void button2_Click(object sender, EventArgs e)
{
    String mstr = "";

    for (i = 0; i < NUM; i++)
    {
        mstr += (i + 1) + "番目：" + data[i] + "¥n";
    }

    mstr += "合計：" + sum + "¥n" + "平均値：" + ave;
    MessageBox.Show(mstr, "結果表示");

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



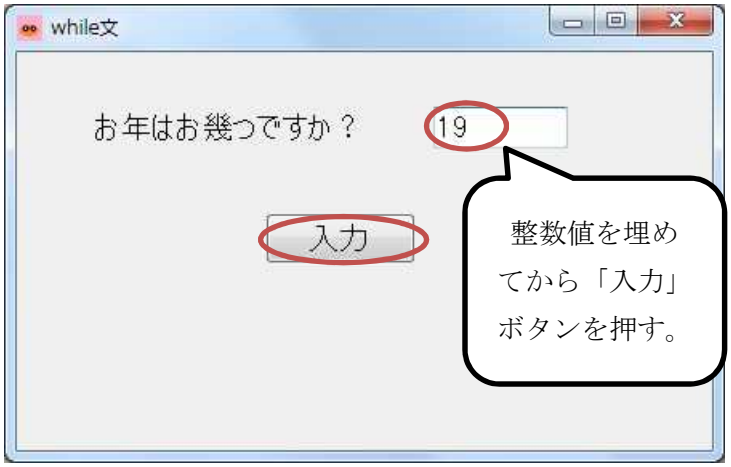
Form3 のフォームデザイナー上で『button1』をダブルクリックして，Form3.cs のプログラムのソースコードを表示し，コードにイベントハンドラを作成する。button1_Click メソッドのブロック内にボタンがクリックされた際の処理（赤枠の部分）を記述していく。ここで，n は 1 からいくつまでの和を求めるかをテキストボックスに入力された値で決めて初期値とし，繰り返しの度に値を減らしていくための変数で，sum は n から 1 までの値を足し込んでいくための変数である。

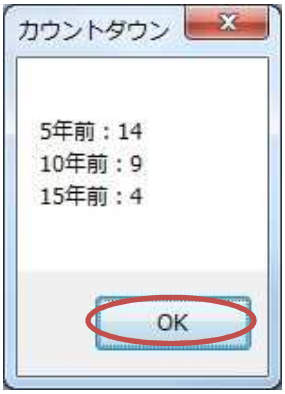
```
private void button1_Click(object sender, EventArgs e)
{
    int sum = 0;
    int n = Int32.Parse(textBox1.Text);

    do
    {
        sum += n;
        n--;
    } while (n >= 1);

    MessageBox.Show("結果：" + sum, "結果表示");
}
```



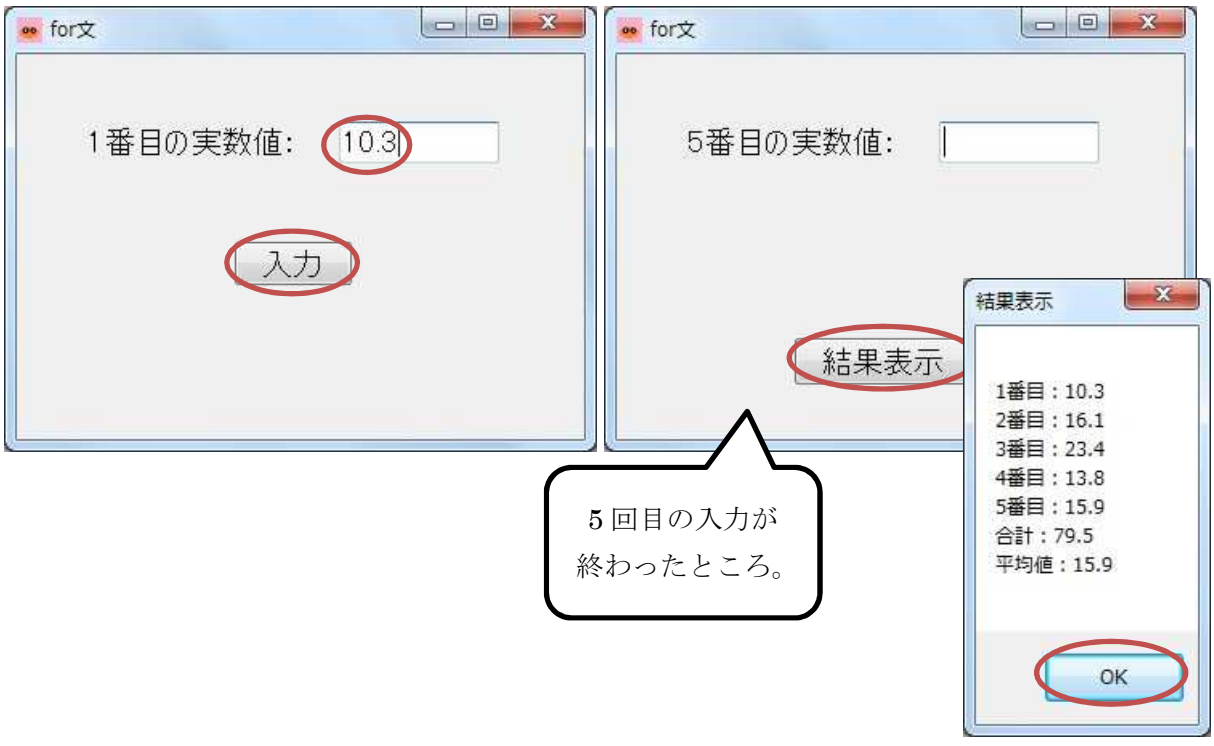
4）プログラムの実行・最終確認

『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。



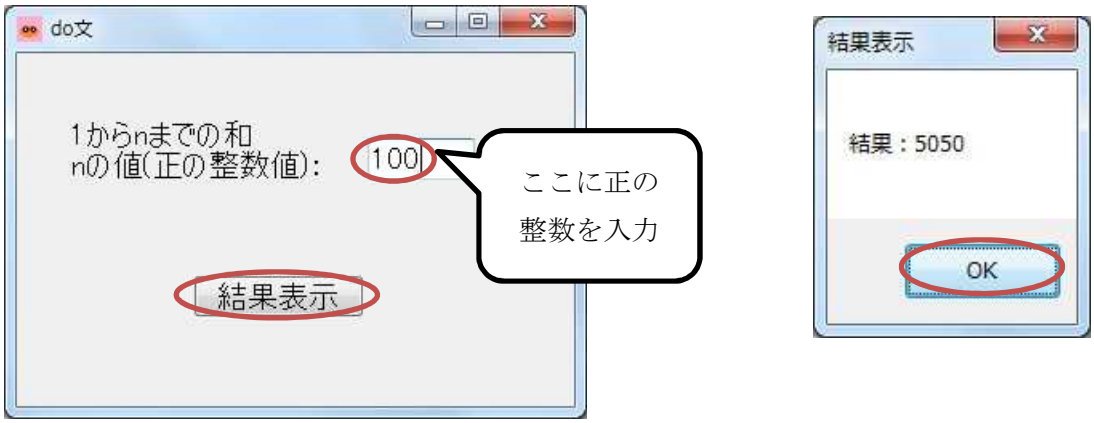


今回のプログラムでは，実数値をテキストボックスに入力して入力ボタンを押すという操作を 5 回繰り返すと結果表示のボタンが現れる。（図はその一例）



次に，do 文の動作確認をする。ここでは，いくつかの正の整数を入力して正しい結果を得られているかどうか確かめる。（図はその中の一例）

＊ 1 から n までの和は，n×(n+1)÷2 で求められる。この式で検算してみる。



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¥Tenth¥Tenth

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_10th.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。