2018 年 10 月 25 日（木）実施

変数とデータ型

変数とは

プログラム中の命令を通じて，メインメモリ上にデータを格納する領域を確保し，必要に応じてその場所に格納されるデータを上書きすることが可能である。このような領域を**変数**という。C#言語では，変数の取り扱いに関して，次の様な特徴がある。

1. プログラム中で用いる変数は必ず**宣言**しておく。 ⇒ メインメモリ上にデータを格納する領域を確保せよ，という命令に相当する。

2. 変数の宣言時には，**データ型**を指定する。 ⇒ データ型毎にデータを格納する領域のサイズが固定されている。

   例） `int x;`　// 整数（integer）のデータ型の変数 x を宣言

3. 変数にデータを格納するには，**代入**を行う。

   例） `x=10;`　/* 変数 x に定数 10 を代入（=の右辺を評価し，その値を左辺の変数に格納）
   なお，宣言時に**初期化**することも可能である。 ⇒ `int x=4;` */

4. 代入式の左辺以外に変数名を用いると**参照**が行われ，変数に格納されたデータが取り出されて利用される。

   例） `y = x*2;`　// 変数 x の中身を 2 倍した値を変数 y に代入する。

データ型

C#言語には数値や文字のデータを扱う**値型**と，クラス等を扱う**参照型**とがある。主な値型には，次のものがある。

分類	型名	サイズ	内容
文字型	char	16 ビット	Unicode 文字（U+0000 ～ U+FFFF）
整数型	byte	8 ビット	符号無し整数（0 ～ 255）
	short	16 ビット	符号付き整数（-32768 ～ 32767）
	int	32 ビット	符号付き整数（-2^{31} ～ $2^{31}-1$ = -2147483648 ～ 2147483647）
	long	64 ビット	符号付き整数（-2^{63} ～ $2^{63}-1$）
実数型	float	32 ビット	単精度浮動小数点数　有効桁数 7 桁
	double	64 ビット	倍精度浮動小数点数　有効桁数 15-16 桁
論理型	bool	1 ビット	論理値（true または false）

値型の変数はデータを直接格納するのに対して，参照型の変数はデータ（オブジェクト）への参照を格納する。

C#言語では，1 文字を格納するには，char 型の変数を用いる。その値は Unicode で表される文字コードの数値で，16 ビットの符号無し整数である。

例） char c;　// 文字（character）のデータ型の変数 c を宣言
char c1 = 'A'; // c1 の宣言時に初期値として文字 A を設定（一重引用符で挟む）
char n1 = '神'; // n1 の宣言時に初期値として文字 神 を設定（C#言語では全角文字も 1 文字）

文字列の扱い

String クラス

C#言語では，文字列を扱う String クラスが用意されている。また，String の別名として，参照型の string も用意されている。文字列データは第 2 回の教材に登場した文字列リテラルによって表される。

例） string s;
s = "CUC"; // string s = "CUC"; の様に，宣言時に初期化することも可能である。

書式指定文字列

書式指定文字列とは，内容が実行時に動的に決定される文字列で，Format メソッドを用いて，中括弧内に，実行時に他の値に置換されるプレースホルダーを埋め込むことで，作成される。

例） s = string.Format("{0}×{1}の答えは{2}", i, j, (i * j));

この例では，{0}，{1}，{2}のそれぞれが，プログラムの実行時に i，j，i＊j の値に置き換えられる。

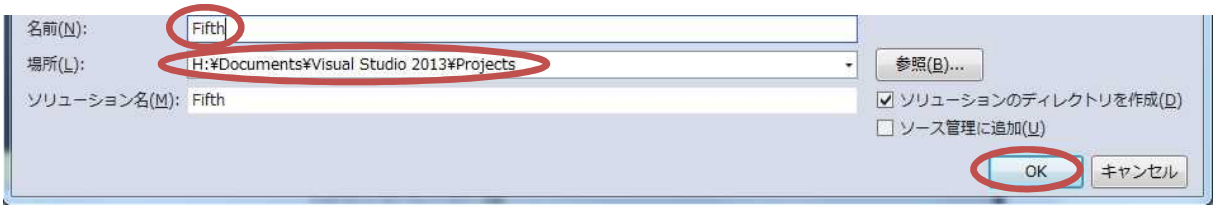
本日の課題

テキストボックスに入力されたデータを整数値や実数値として読み取り，四則演算及び剰余算の結果を書式指定文字列を用いて一繋がりの文字列として文字列変数に代入し，その値をラベルに表示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「Fifth」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

フォームを幅，高さ共に拡げる。[表示] → [ツールボックス] と辿って，ツールボックスを表示する。『コモンコントロール』を選択して展開する。『Label』を選択してから，左にある縦の『ツールボックス』タブをクリックして，メニューを引っ込める。フォームをクリックしてラベルを左上に貼る（『label1』の名前が付く）。フォーム上の『label1』を選択して，プロパティのフォントサイズを 14 ポイントに変更する。（ここまでの詳細は，第 1 回及び第 2 回の教材を参照）

プロパティの『Text』を「label1」から「**a の値**」に書き換える。

同様にして，『label2』，『label3』，『label4』をフォーム上に貼り，プロパティのフォントサイズを 14 ポイントに変更する。また，『label2』のプロパティの『Text』を「label1」から「**b の値**」に書き換える。

ツールボックスで『TextBox』を選択してから，左にある縦の『ツールボックス』タブをクリックして，メニューを引っ込める。フォーム上でドラッグしてテキストボックスを『label1』の右に貼る（『textBox1』の名前が付く）。プロパティのフォントサイズを 14 ポイントに変更する。

同様に『textBox2』を『label2』の右に貼り，プロパティのフォントサイズを 14 ポイントに変更する。

（図は次のページ）

ツールボックスで『Button』を選択してから，左にある縦の『ツールボックス』タブをクリックして，メニューを引っ込める。フォームをクリックしてボタンを左のテキストボックスの下に貼る（『button1』の名前が付く）。『button1』のプロパティのフォントサイズを 14 ポイントに変更する。プロパティの『Text』を「button1」から「**整数計算**」に書き換える。

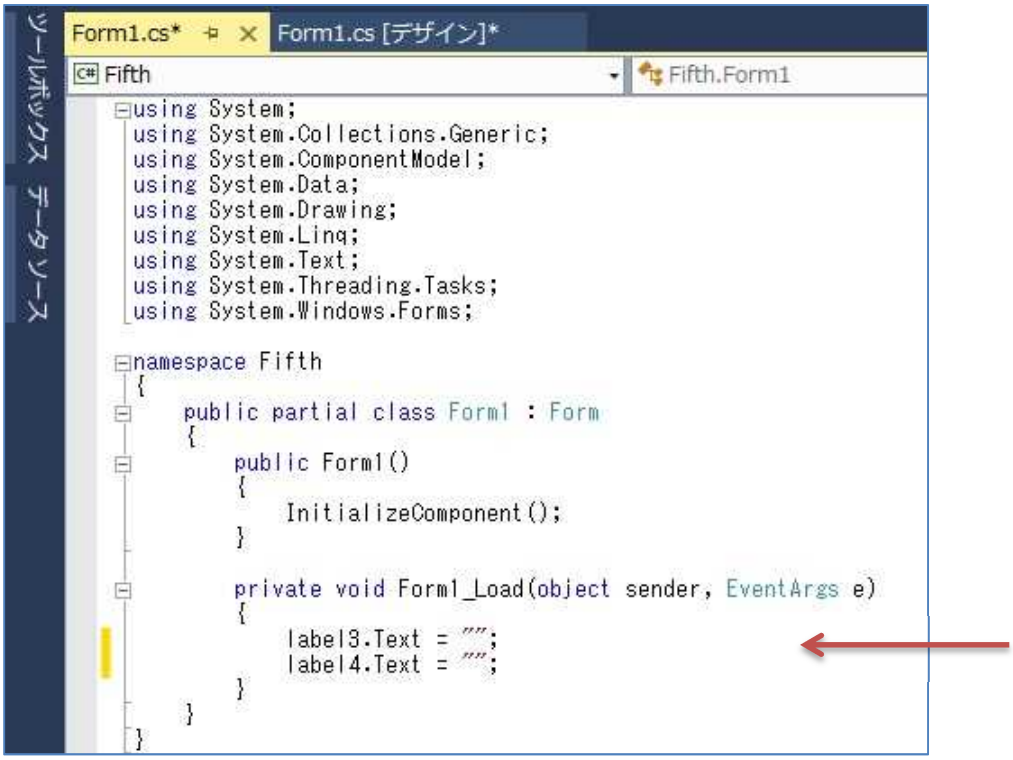
同様に『button2』を右のテキストボックスの下に貼り，プロパティのフォントサイズを 14 ポイントに変更する。プロパティの『Text』を「button2」から「**実数計算**」に書き換える。

3) コーディング

フォーム上の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，『label3』及び『label4』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)
{
    label3.Text = "";
    label4.Text = "";
}
```

（コードエディタの図は次のページ）



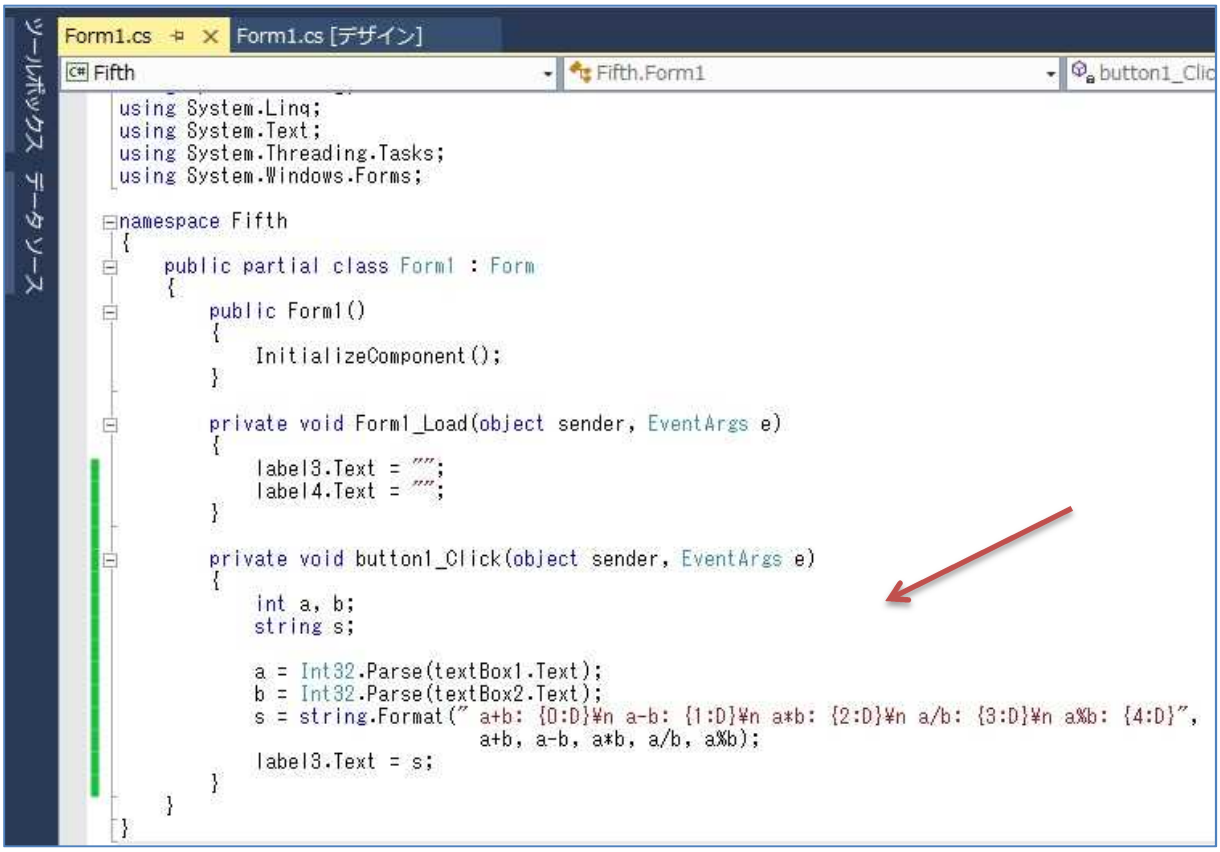
フォームデザイナー上で『button1』をダブルクリックして，Form1.cs のプログラムのソースコードを表示する。button1_Click メソッドのブロック内にボタンがクリックされた際の処理（赤枠の部分）を記述していく。

```
private void button1_Click(object sender, EventArgs e)
{
    int a, b;
    string s;

    a = Int32.Parse(textBox1.Text);
    b = Int32.Parse(textBox2.Text);
    s = string.Format("a+b: {0:D}¥n a-b: {1:D}¥n a*b: {2:D}¥n a/b: {3:D}¥n a%b: {4:D}",
                      a+b, a-b, a*b, a/b, a%b);
    label3.Text = s;
}
```

ここでは，2 つのテキストボックスに入力された文字列をそれぞれ Int32.Parse メソッドで整数値として読み取り，int 型の変数 a，b に代入し，『label3』の Text プロパティの値に一連の計算結果を書式指定文字列として設定している。{0:D}等でコロン（:）の後に記述している D は，10 進法表記（Decimal notation）として表示することを指定している。

（コードエディタの図は次のページ）



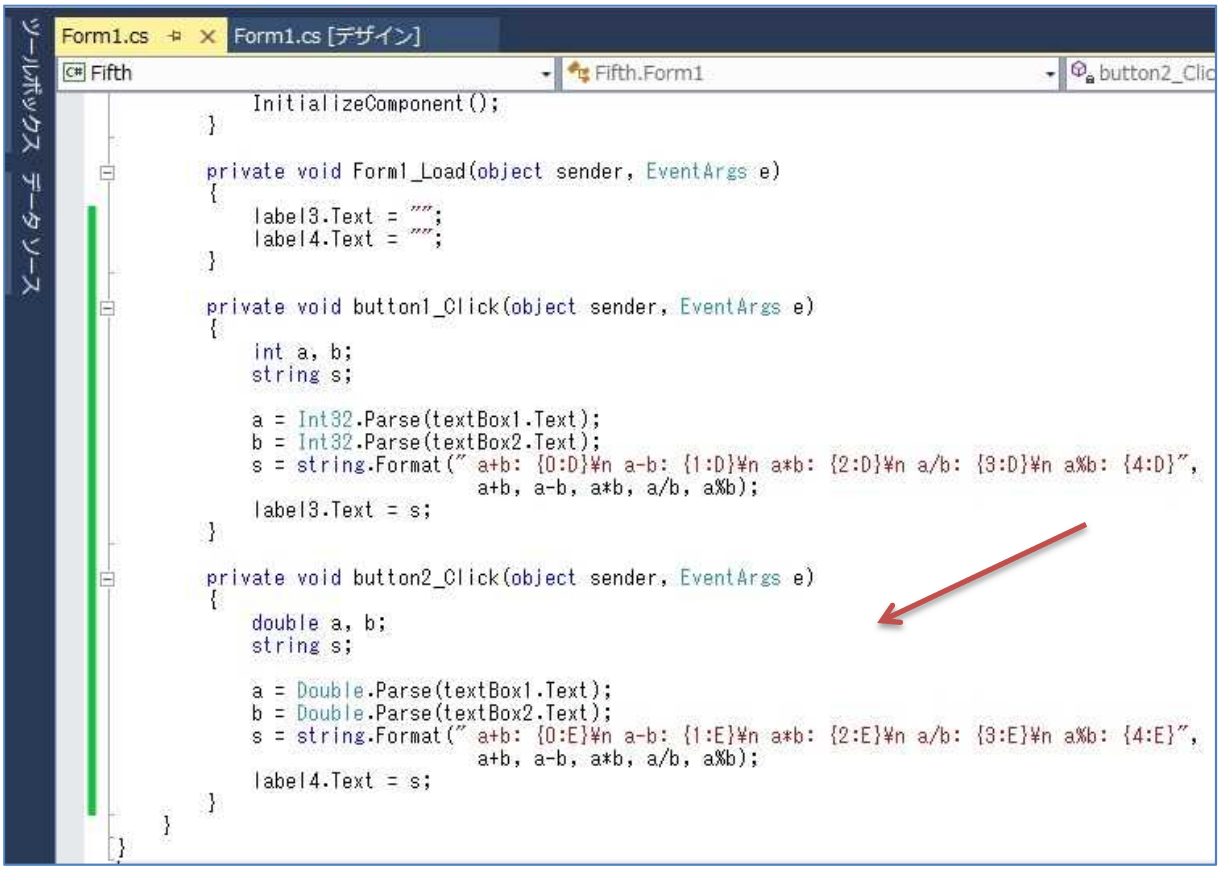
フォームデザイナー上で『button2』をダブルクリックして，Form1.cs のプログラムのソースコードを表示する。button2_Click メソッドのブロック内にボタンがクリックされた際の処理（赤枠の部分）を記述していく。

```
private void button2_Click(object sender, EventArgs e)
{
    double a, b;
    string s;

    a = Double.Parse(textBox1.Text);
    b = Double.Parse(textBox2.Text);
    s = string.Format(" a+b: {0:E}¥n a-b: {1:E}¥n a*b: {2:E}¥n a/b: {3:E}¥n a%b: {4:E}",
                      a+b, a-b, a*b, a/b, a%b);
    label4.Text = s;
}
```

ここでは，2 つのテキストボックスに入力された文字列をそれぞれ Double.Parse メソッドで実数値として読み取り，double 型の変数 a，b に代入し，『label4』の Text プロパティの値に一連の計算結果を書式指定文字列として設定している。{0:E}等でコロン（:）の後に記述している E は，指数表記（Exponential notation）として表示することを指定している。

（コードエディタの図は次のページ）



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，２つのテキストボックスに整数値を入力して，『**整数計算**』ボタンをクリックして，ラベルに計算結果が表示されることを確認する。

次に，一度プログラムを終了させてから再度起動し，2 つのテキストボックスに実数値を入力して，『**実数計算**』ボタンをクリックして，ラベルに計算結果が表示されることを確認する。

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¥Fifth¥Fifth

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。

【余裕のある人向けの課題】

フォームの下の方に「クリア」と表示されたボタンを配置し，プログラムの実行時にそのボタンをクリックすると，『textBox1』，『textBox2』，『label3』及び『label4』の『Text』プロパティに空の文字列（""）を設定するイベントハンドラを作成する。