C#の基本2
～プログラムの制御構造～

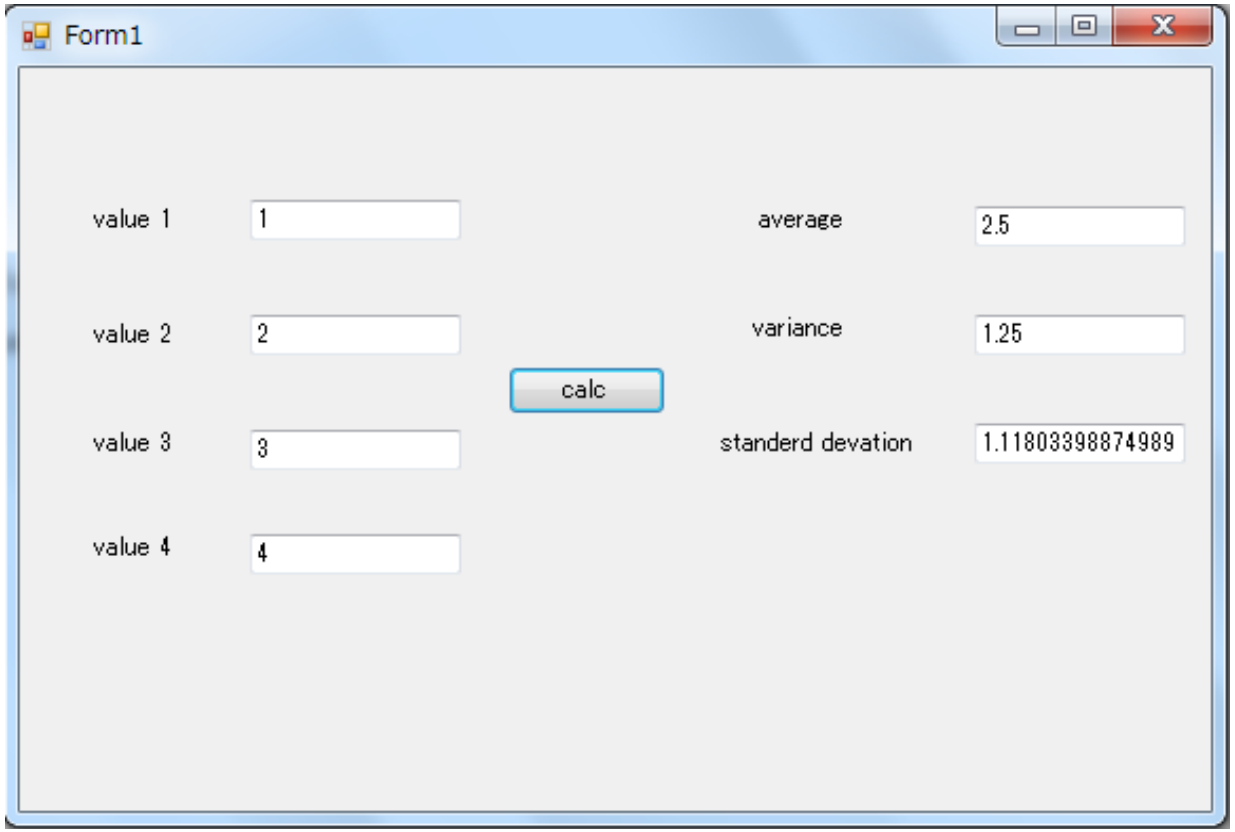
今回学ぶ事

- プログラムの制御構造としての単岐選択処理（If文）、前判定繰り返し処理（for文）について説明を行う。

- また、整数型（int型）等の組み込み型や配列型についても解説を行う。

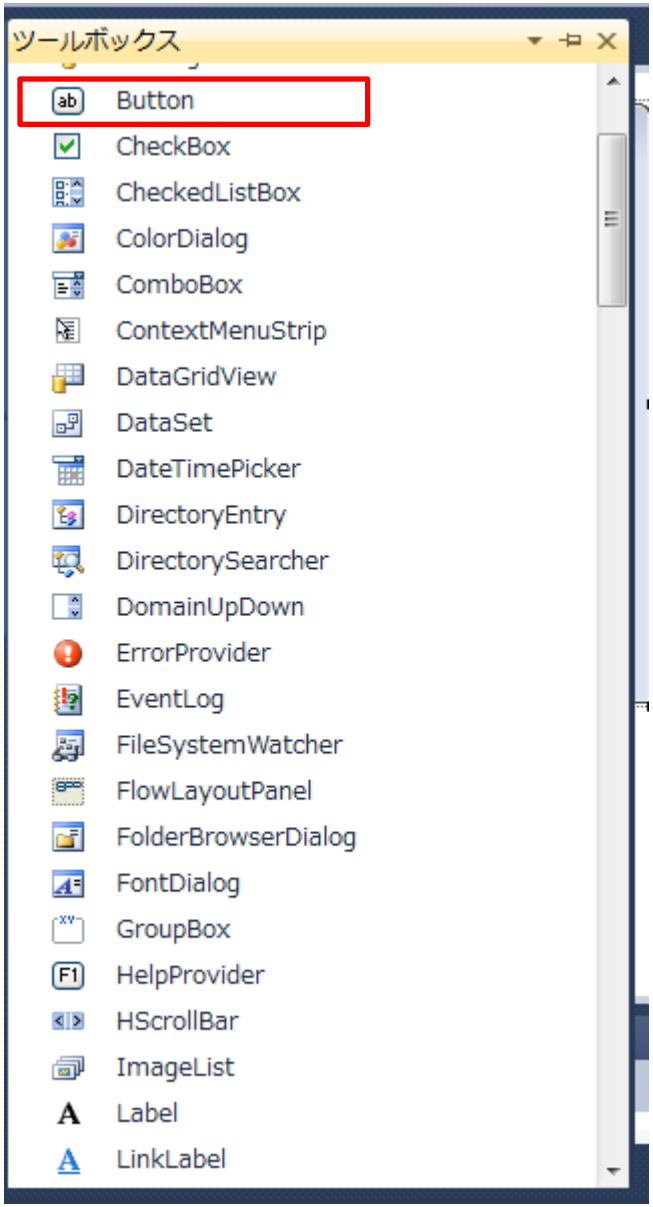
今回作るプログラム

- 入れた文字の平均、分散、標準偏差を表示するプログラム
- このプログラムではcalcボタンを押すと計算を行う（valueは整数に限る）

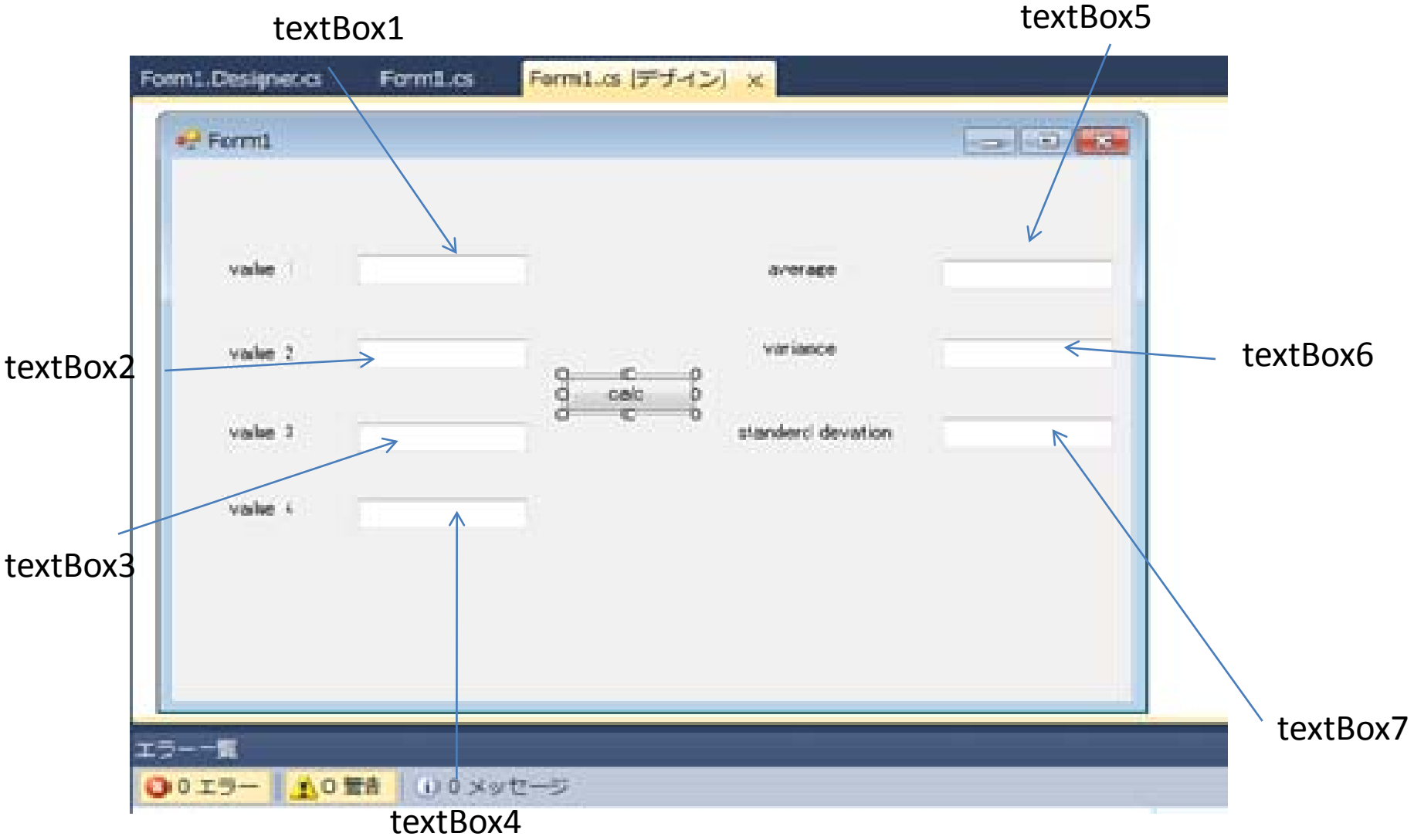


フォーム作り

- まず、label、button、textBoxを配置する。
- ツールボックスより左クリックで選択する。
- そのまま、フォーム上を左クリックすると設置することができる。



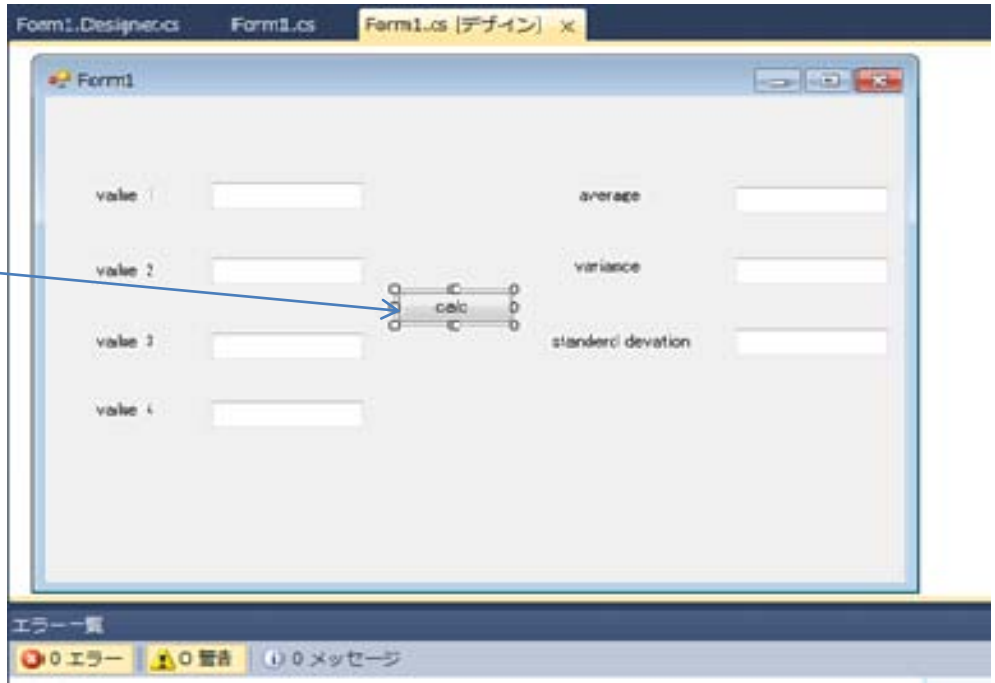
フォーム作り　配置例


textBox1
textBox5
textBox2
textBox6
textBox3
textBox7
textBox4
Form1.Designer.cs
Form1.cs
Form1.cs [デザイン]
Form1
value 1
value 2
value 3
value 4
average
variance
standerd deviation
calc
エラー一覧
0 エラー
0 警告
0 メッセージ

コード記述

- 前回と同様に、buttonをクリックしたときに動作するプログラムを作成する。
- そのため、buttonをダブルクリックしコードを記載する

ダブルクリック



コード記述

- 中央のbuttonをダブルクリックすると以下の様なコードが表示される

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace test
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {

      }


   }
}
```

コード記述

- このコードに以下の様に記述する

```
private void button1_Click(object sender, EventArgs e)
    {
        int value1, value2, value3, value4;
        double sum;
        double ave,v,sd;

        value1 = int.Parse(textBox1.Text);
        value2 = int.Parse(textBox2.Text);
        value3 = int.Parse(textBox3.Text);
        value4 = int.Parse(textBox4.Text);

        sum = value1 + value2 + value3 + value4;
        ave = sum / 4;
        sum= (value1-ave)*(value1-ave)+(value2-ave)*(value2-ave)+(value3-ave)*(value3-ave)+(value4-
    ave)*(value4-ave);
        v = sum / 4;
        sd = Math.Sqrt(v);

        textBox5.Text = ave.ToString();
        textBox6.Text = v.ToString();
        textBox7.Text = sd.ToString();

    }
```

コード解説

```
int    value1, value2, value3, value4;
double    sum;
double    ave,v,sd;
```

- 今回使う数字の宣言を行う。この、value1やsumといった文字に数字を代入することにより計算を行う。
- このように、数字を代入するための文字の事を変数と呼ぶ。
- Int 、doubleなどは組み込み型と呼ばれるものであり文字が整数や実数であることを示す。

コード解説　組み込み型とは

- 以下に代表的な組み込み型を示す
- byte 型（１バイト）は、-128 ～127 の範囲
- short 型（２バイト）は、-32768 ～32767 の範囲
- int 型（４バイト）は、-2147483648 ～2147483647 の範囲
- float型は4バイトの実数型
- double型は８バイトの実数型

- 他にもさまざまな組み込み型が存在する

コード解説　組み込み型とは

- 基本的には以下の様に宣言する

```
int a;
```

- これは、int（整数）型の変数aを以降のプログラムで用いるという意味になる。
- また、複数の変数を用いる場合には以下の様にも宣言することができる。

```
int a,b,c;
```

コード解説

```
value1 = int.Parse(textBox1.Text);
value2 = int.Parse(textBox2.Text);
value3 = int.Parse(textBox3.Text);
value4 = int.Parse(textBox4.Text);
```

- このコードでは、テキストボックスの内容を変数に代入している
- int.Parse(～)を用いることにより文字で書かれた内容を数字（int型）に変換することができる。

コード解説

```
sum = value1 + value2 + value3 + value4;
ave = sum / 4;
```

- このコードでは1文目でsumにすべての和を代入している
- さらに、2文目でaveにsum÷４の値を代入している
- つまり、aveに平均が入ることが分かる

コード解説

```
sum= (value1-ave)*(value1-ave)+(value2-ave)*(value2-ave)+(value3-ave)*(value3-ave)+(value4-ave)*(value4-ave);
v = sum / 4;
```

- 同様にこのコードではvに分散の値が入ることが分かる。
- ここで、＊は掛け算を示している。

コード解説

```
sd = Math.Sqrt(v);
```

- Math.Sqrt(～)は～の平方根を返す関数である。
- この関数は、MathクラスのSqrt関数という意味である。
- つまり、sdには。分散の平方根である標準偏差が入ることになる

コード解説

```
textBox5.Text = ave.ToString();
textBox6.Text = v.ToString();
textBox7.Text = sd.ToString();
```

- ave.ToString()とは、実数型であるaveを文字に変換するという意味である。
- これにより、テキストボックスに記載することができる。

コード改良

- 例えば、100個や1000個の値の計算を行う時などこのコードでは書くのが面倒である。

- そこで、配列型、for文と呼ばれる技術によりコードの簡略化を図る。

- 次ページに改良したコードを示す。

```
private void button1_Click(object sender, EventArgs e)
    {
        int[] value = new int[5];
        double sum;
        double ave,v,sd;
        int n;

        value[0] = int.Parse(textBox1.Text);
        value[1] = int.Parse(textBox2.Text);
        value[2] = int.Parse(textBox3.Text);
        value[3] = int.Parse(textBox4.Text);

        sum = 0;
        n = 0;
        for (int i = 0; i < 4; i++)
        {
            sum = value[i] + sum;
            n＝n+1;
        }
        ave = sum / n;
        n = 0;
        sum = 0;
        for (int i = 0; i < 4; i++)
        {
            sum = (value[i] - ave) * (value[i]- ave)+sum;
            n=n+1;
        }
        v = sum / n;
        sd = Math.Sqrt(v);

        textBox5.Text = ave.ToString();
        textBox6.Text = v.ToString();
        textBox7.Text = sd.ToString();


    }
```

コード解説

```
int[] value = new int[5];
```

- Int[]は変数が配列型であることを示す。
- この1文でint型のvalue[0], value[1], value[2], value[3], value[4]の5個の変数を宣言したのと同様の意味になる。

コード解説　配列型

- 基本形は以下の様な構文である

```
int[] a =new int[（要素数）];
double[] a =new double[（要素数）];
```

- 配列型では添え字（Index）によって変数を指定できるため繰り返し処理を行う時に便利
- newは配列のメモリ領域を確保するという意味である

コード解説　繰り返し処理

```
sum = 0;
  n = 0;
  for (int i = 0; i < 4; i++)
  {
      sum = value[i] + sum;
      n++;
  }
  ave = sum / n;
```

コード解説　繰り返し処理

for（int i=0;i<N;i++){～}

- 変数iが0からN－1（i<Nであるため）まで1ずつ増加しながらN回実行する。
- int iの様にC#では同時に宣言を行うことができる。（C言語ではできなかった）
- 画像処理では最も良く使う構文の一つである

```
sum = 0;
    n = 0;
    for (int i = 0; i < 4; i++)
    {
        sum = value[i] + sum;
        n=n+1;
    }
    ave = sum / n;
```

- つまりこのプログラムではsumにvalue[0]～value[3]までの和が、nには繰り返し回数（今回は4）が入ることになる。
- よって、aveには平均が入ることが分かる。

```
sum = 0;
n = 0;
for (int i = 0; i < 4; i++)
{
    sum = value[i] + sum;
    n=n+1;
}
ave = sum / n;
```

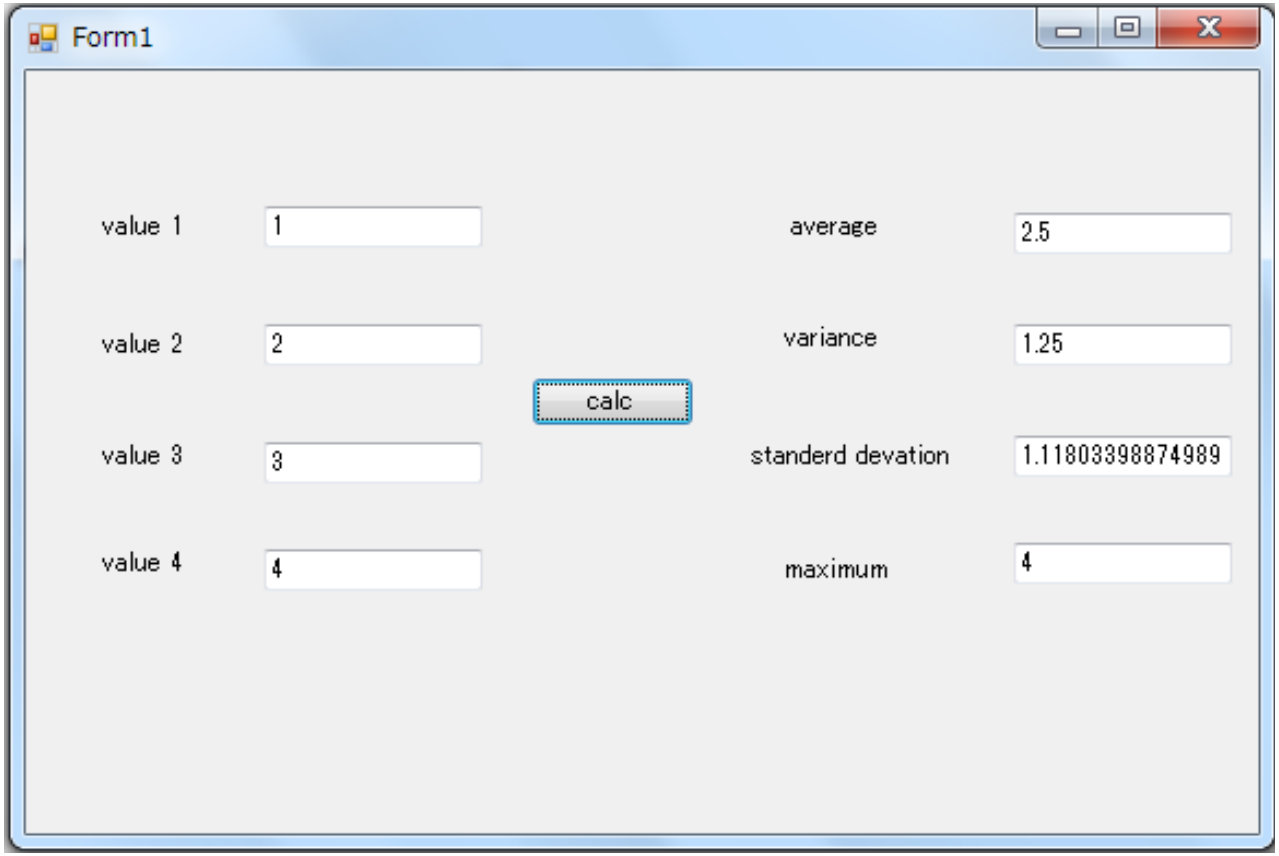
- また、変数は宣言された時点では0が入っているとは限らない。
- sum、nが0でないと困るため始めに0を代入しておく。
- この作業を初期化と呼ぶ。

```
 n = 0;
sum = 0;
for (int i = 0; i < 4; i++)
{
    sum = (value[i] - ave) * (value[i]- ave)+sum;
    n=n+1;
}
v = sum / n;
sd = Math.Sqrt(v);
```

- 同様に、ここでは分散を算出する際にfor文を用いていることが分かる。
- v、sdを求める作業は同じである。

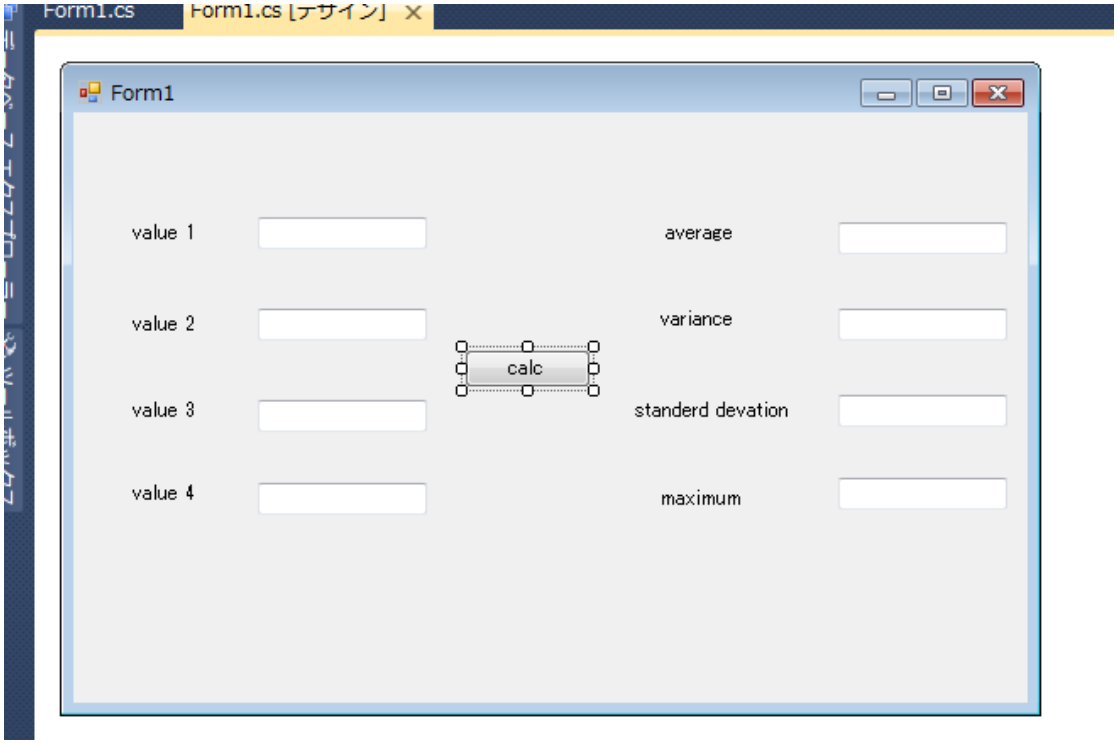
プログラムの追加

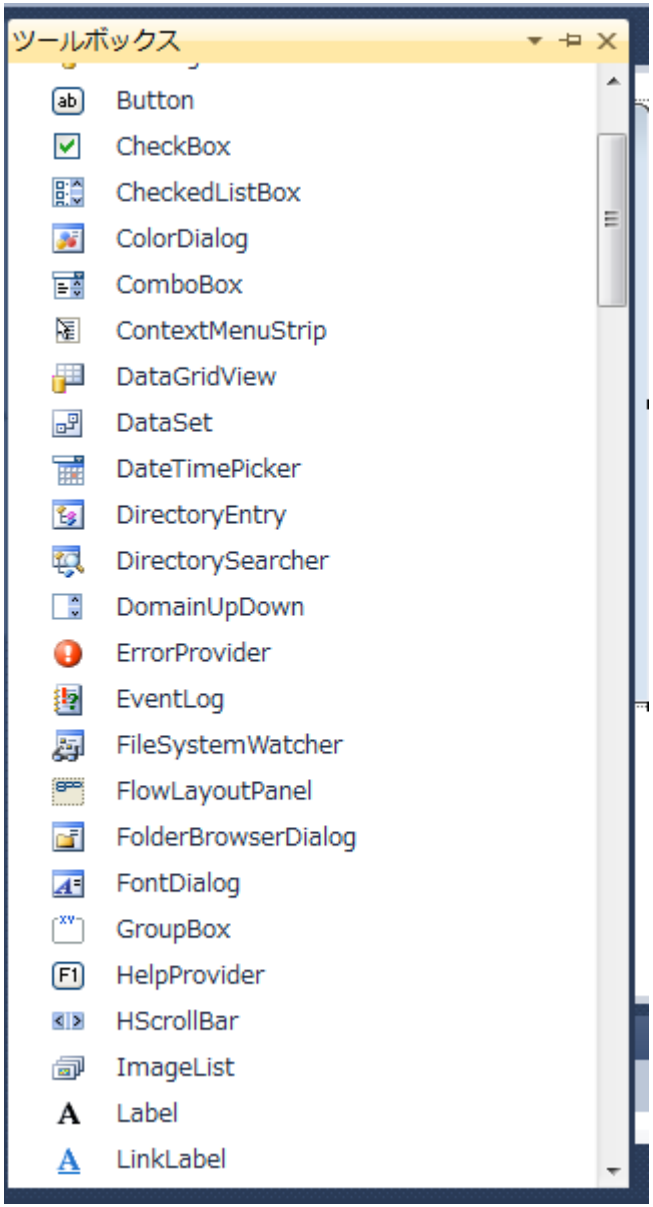
- 次に、value1～value4までのうち最大値となる値を求めるプログラムを作る。（valueは整数に限る）



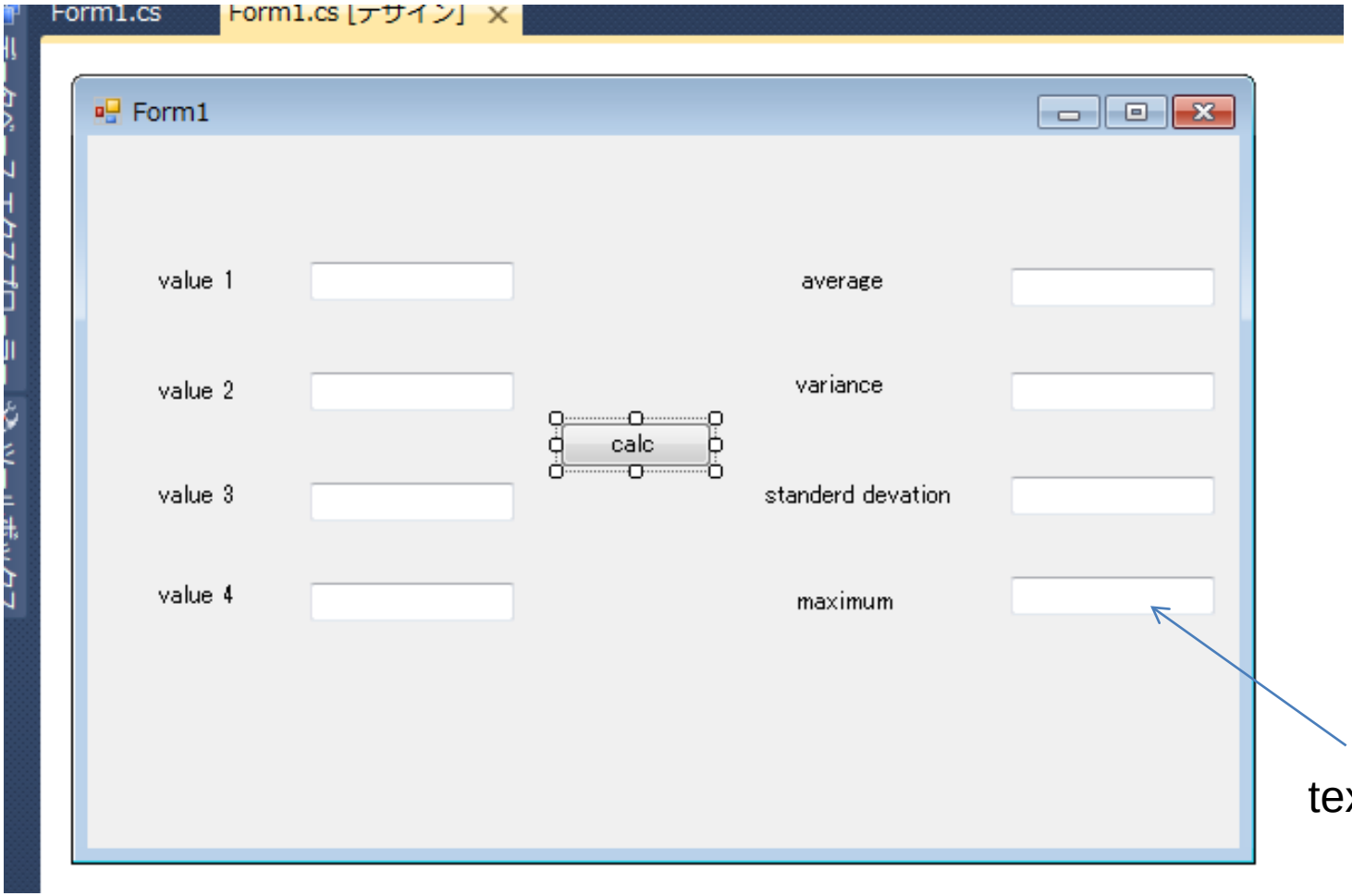
フォーム作り

- textBox、labelを配置する。
- ツールボックスより左クリックで選択する。
- そのまま、フォーム上を左クリックすると設置することができる。





配置例


Form1.cs
Form1.cs [デザイン]
Form1
value 1
value 2
value 3
value 4
calc
average
variance
standerd devation
maximum
textBox8

コードの記載

ここまでのコードの

textBox7.Text = sd.ToString();

の下に以下のコードをそのまま追加する

```
int max=value[0];
for (int i = 0; i < 4; i++)
{
    if (max < value[i]) max = value[i];
}

textBox8.Text = max.ToString();
```

```
int max=value[0];
for (int i = 0; i < 4; i++)
{
    if (max < value[i]) max = value[i];
}

textBox8.Text = max.ToString();
```

- ここでは、if文が用いられているif文の基本構文は以下の様になる。

if (A) B ;

- この文はAが正しければ、Bを行うという意味になる
- また、Bが複数の文に分かれる場合には。

if(A){B; C;}

- このような、書き方もできる。

```
int max=value[0];
        for (int i = 0; i < 4; i++)
        {
            if (max < value[i]) max = value[i];
        }

        textBox8.Text = max.ToString();
```

- つまり、このコードでは“すべてのvalueについて比較を行い今までの最大値（max）より大きければmaxに新しく代入しなさい”という意味になる。

まとめ

- 今回は繰り返し処理としてfor文、選択処理としてif文を取り上げた。
- C#には今回取り上げた制御構文以外にもwhile文、do～while文、switch・case文等様々な制御構文があるため各自調べてください。
- また、組み込み型、配列型に関しても解説を行った。