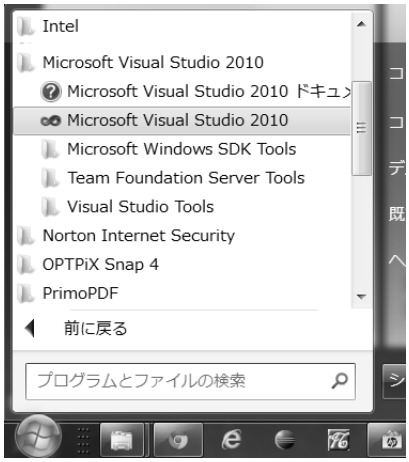
C#とは？

マイクロソフト社によって 2000 年に発表された比較的新しいオブジェクト指向プログラミング言語である。開発ツールである Visual Studio を使うことで、Windows 向け、Web 向け、WindowsPhone 向けといった様々なビジネスアプリケーションや Web サービス、ゲームなどを開発できる。

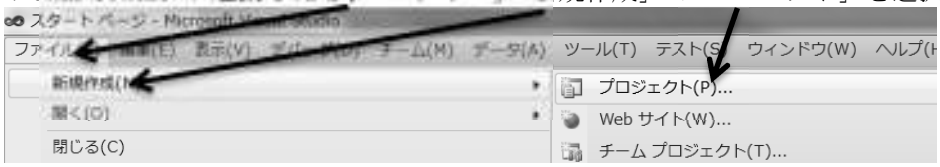
【Mono】プロジェクトというオープンソース環境を使うことで、Linux、FreeBSD、UNIX、Mac OS X、Solaris といった、マイクロソフト以外のプラットフォーム向けのアプリケーションの開発も可能で、ゲーム開発ツールの【Unity】もスクリプト言語の一つとして C#を採用している。文法が Java 言語に近いため、C#を習得すれば Java 言語への移行も容易である。

１．新規プロジェクトの作成

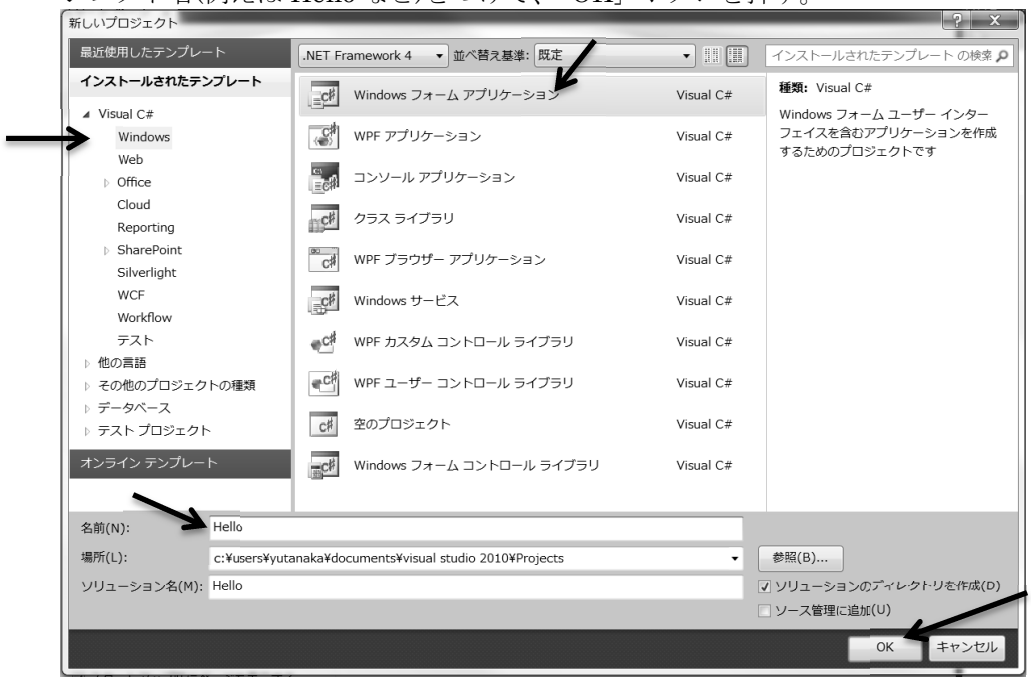
- 「スタート」「すべてのプログラム」「Microsoft Visual Studio 2010」「Microsoft Visual Studio 2010」とクリックしていき、「Visual Studio」を起動する。



- Visual Studio が起動したら、「ファイル」「新規作成」「プロジェクト」を選択。

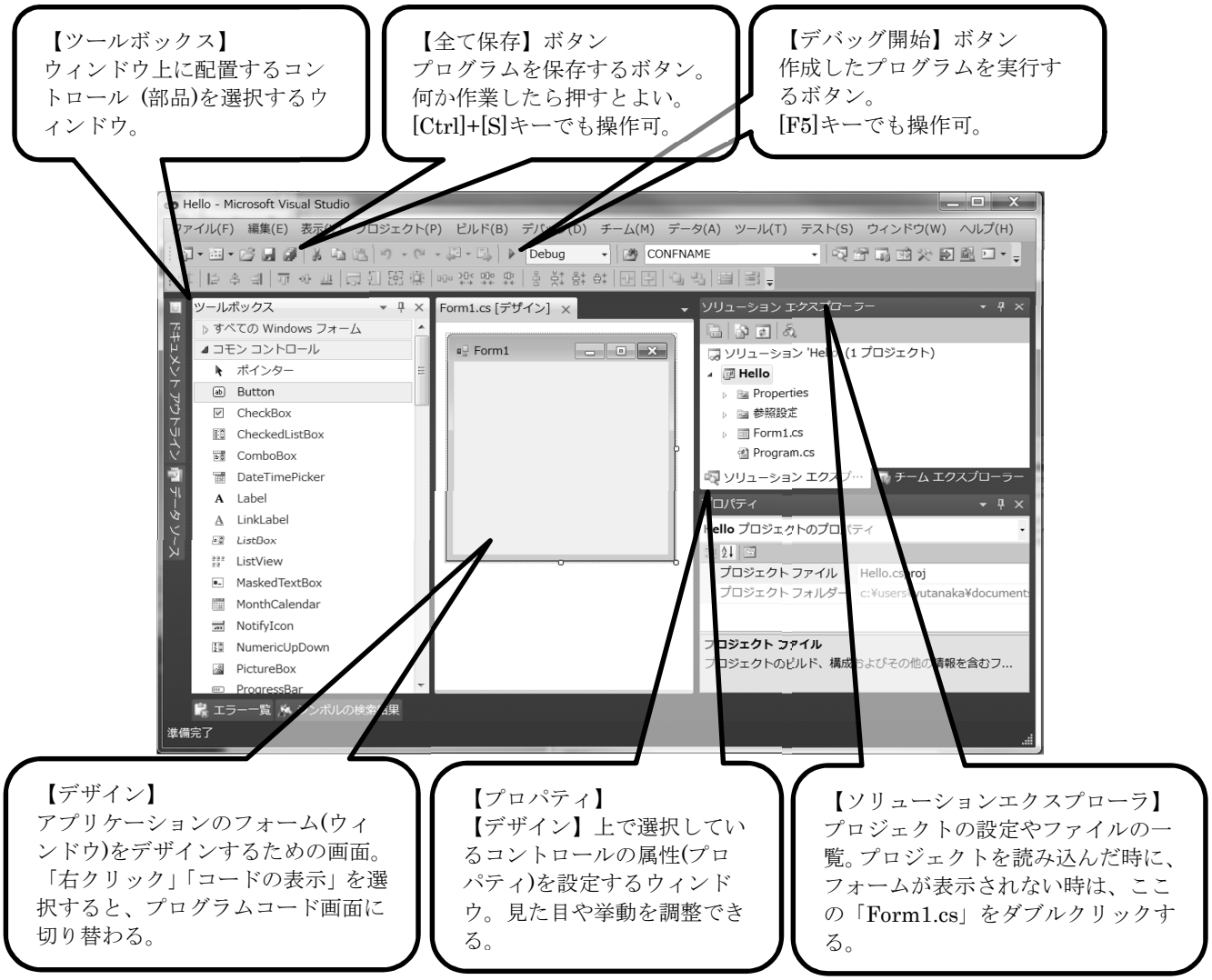


- 「Visual C#」「Windows」「Windows フォーム アプリケーション」を選択したら、「名前」の欄にプロジェクト名(例えば Hello など)をつけて、「OK」ボタンを押す。



2．画面レイアウト

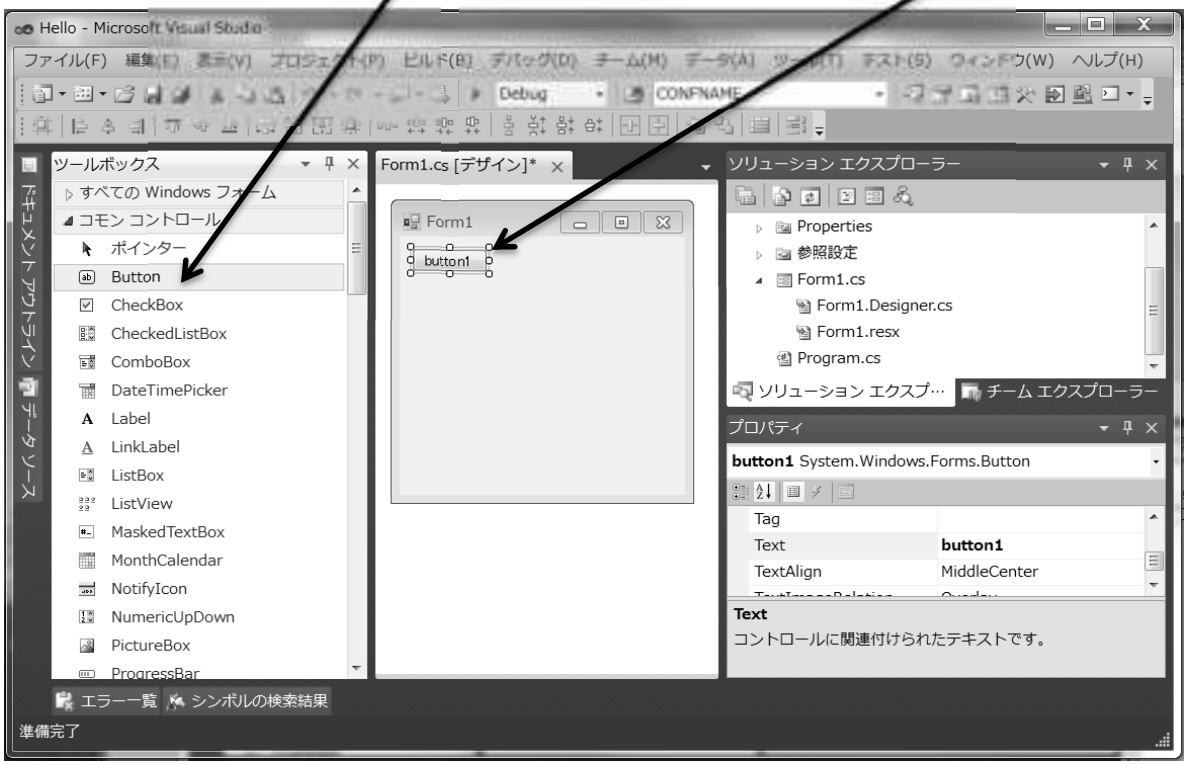
Visual Studio の標準的な画面レイアウトは以下の通り。



「ツールボックス」「ソリューションエクスプローラ」「プロパティ」ウィンドウが表示されない時は、「表示」メニューから表示させたいウィンドウを選択する。

３．コンポーネントの配置

- ツールボックスの「Button」をクリックして選択し、フォーム上の置きたい場所でクリックすると、フォームにボタンが現れる。



- このように、フォーム上に配置できるパーツのことを Visual Studio では「コントロール」と呼ぶ。

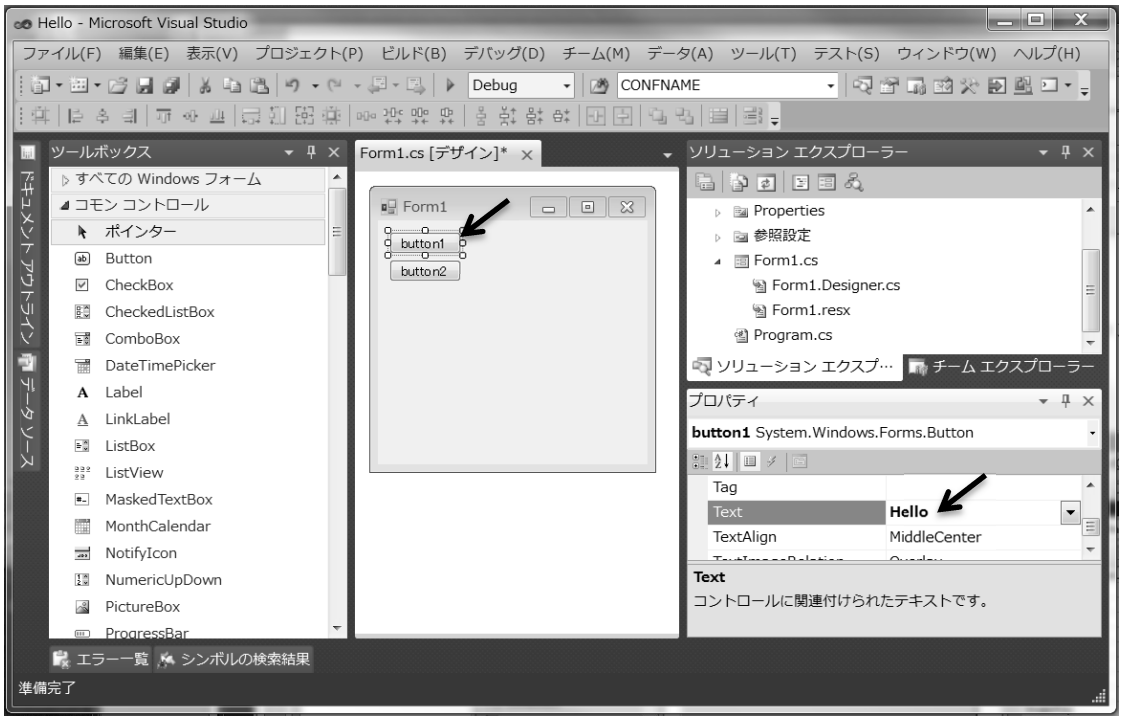
演習１：フォーム上にボタンを２つ配置しよう。
演習２：【デバッグ開始】ボタンを押すか、[F5]キーを押して、プログラムを実行してみよう。
演習３：実行されているウィンドウの右上の×ボタンを押すか、「デバッグ」「デバッグの停止」を選んで、プログラムを停止しよう。

4.プロパティの変更

コントロールの属性(見た目や挙動)は「プロパティ」ウィンドウで変更できる。変更したいコントロールをクリックして選び、「プロパティ」ウィンドウに表示させて編集する。

コントロールに表示される文字列は「Text」というプロパティで編集できる。以下の通りに操作して、button1を「Hello」と表示されるように変更してみよう。

- フォーム上で、「button1」をクリックする。
- 「プロパティ」ウィンドウから「Text」プロパティを探して、「button1」を「Hello」に書き換える。



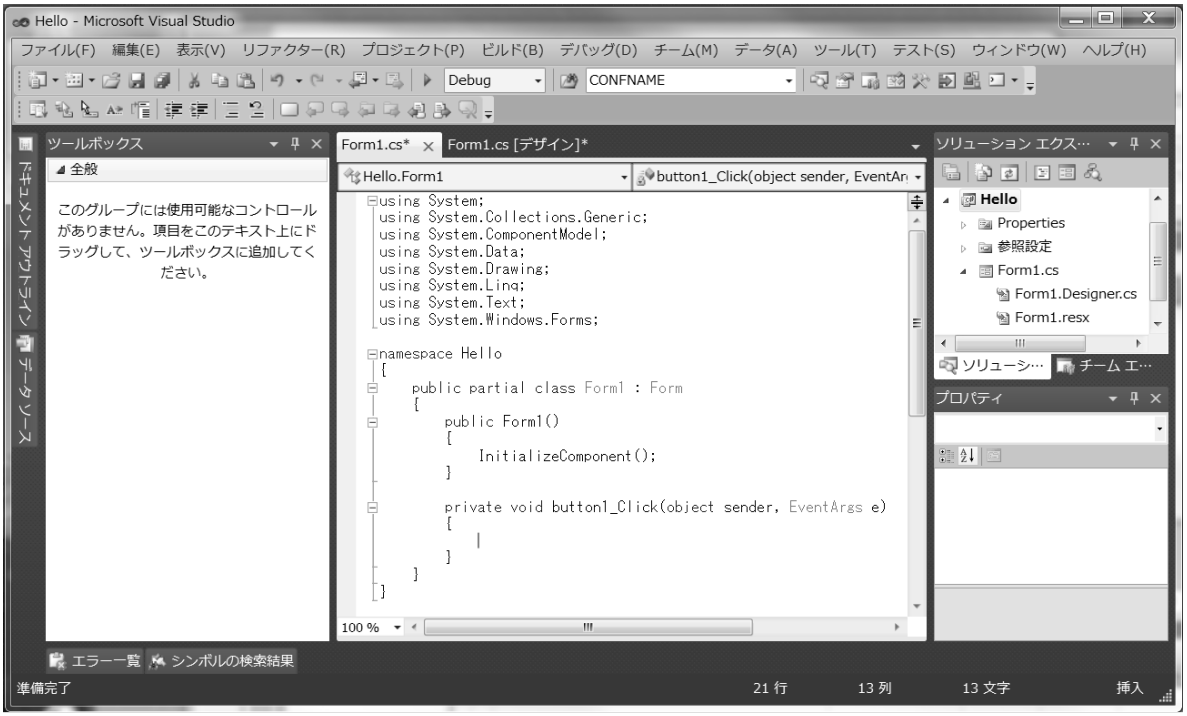
演習 1：「button2」の文字を Bye に変更しよう。

演習 2：その他のプロパティを適当にいじってみて、以下の操作方法を調べてみよう。

- 文字の大きさの変更
- 文字の色の変更
- フォントの変更
  - 反映しないフォントもある。日本語フォント(MS や HG から始まるフォント)で試してみよ。
- ボタンの色の変更

5．イベントの定義

- コントロールが「ある状態になったとき」、特定の処理が実行される。このきっかけとなる状態変化をイベントと呼ぶ。
- 標準的なコントロールのイベントは、そのコントロールをダブルクリックすると記述を始められる。

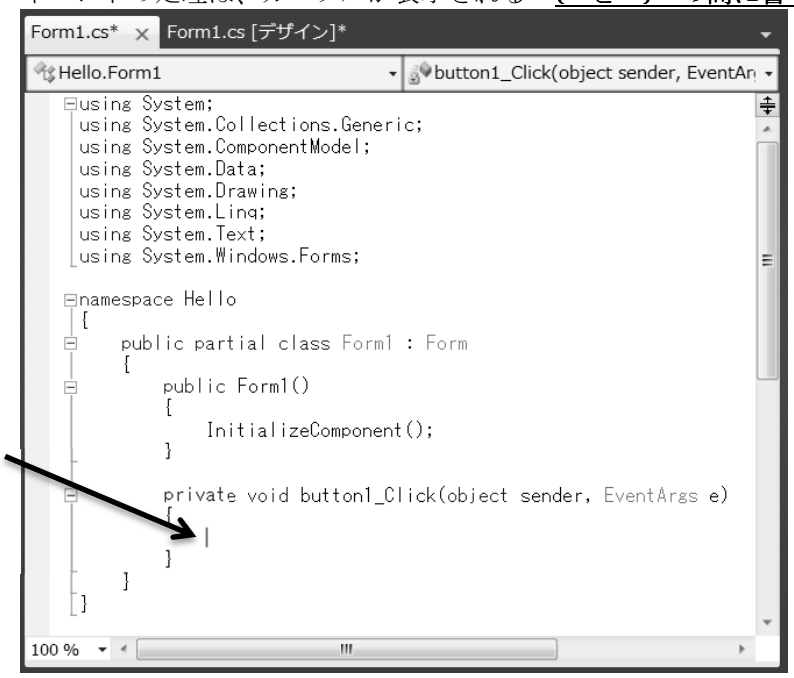


演習：Hello ボタンをダブルクリックしてイベントの処理の記述に入ろう。（ボタンの標準イベントは「ボタンがクリックされたら」）

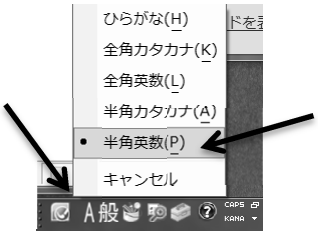
６．イベントの記述

コードを書く際の注意点！！

- イベントの処理は、カーソルが表示される　**{　と　}　の間に書く**。



- プログラムは「半角英数」で入力すること。漢字変換は使わない。



- 意味が分からないうちは、Visual Studio が自動的に作成したコードは消したり、変更したりしない。
- 大文字小文字を区別する。
- 一つの「文」は「；」で終了する。書き忘れに注意。
- ある程度入力すると入力支援機構が働く。上下キーで選択して、「Tab」や「Enter」で表示できる。入力が楽になる上に、間違いを減らせるので、積極的に利用しよう。

演習：Hello ボタンのクリック時の処理に

```
MessageBox.Show( "Hello World" );
```

と記述しよう。

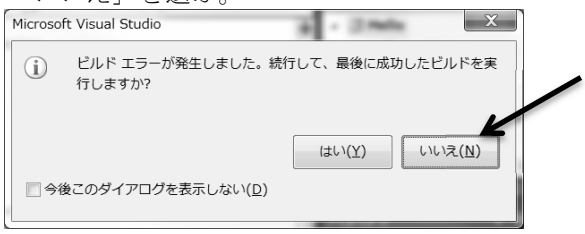
“(ダブルクォーテーション)は、[Shift]+[2]で入力する。

7．実行・デバッグ

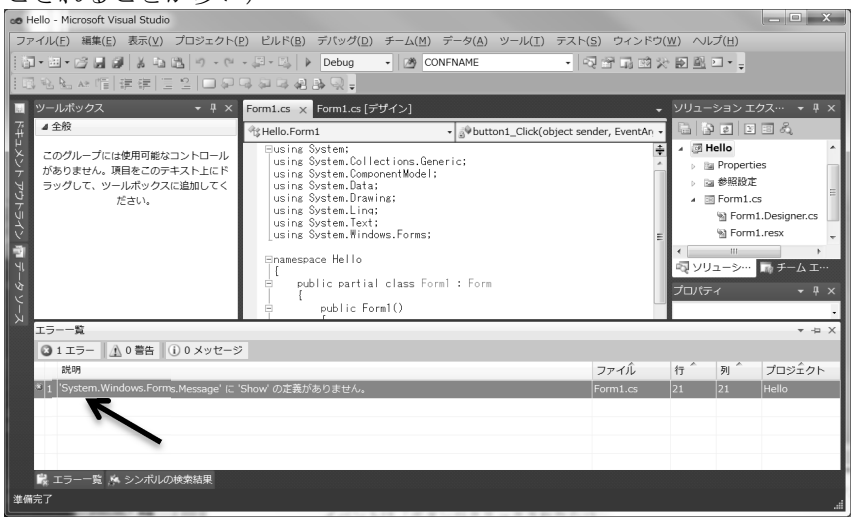
- 自作プログラムが完成！ これまでと同じように「実行」ボタンで実行できる。

もしもエラーが出たら？

- 「いいえ」を選ぶ。



- 最初のエラーメッセージをダブルクリックする。(後のほうのエラーは、最初のエラーによって引き起こされることが多い)

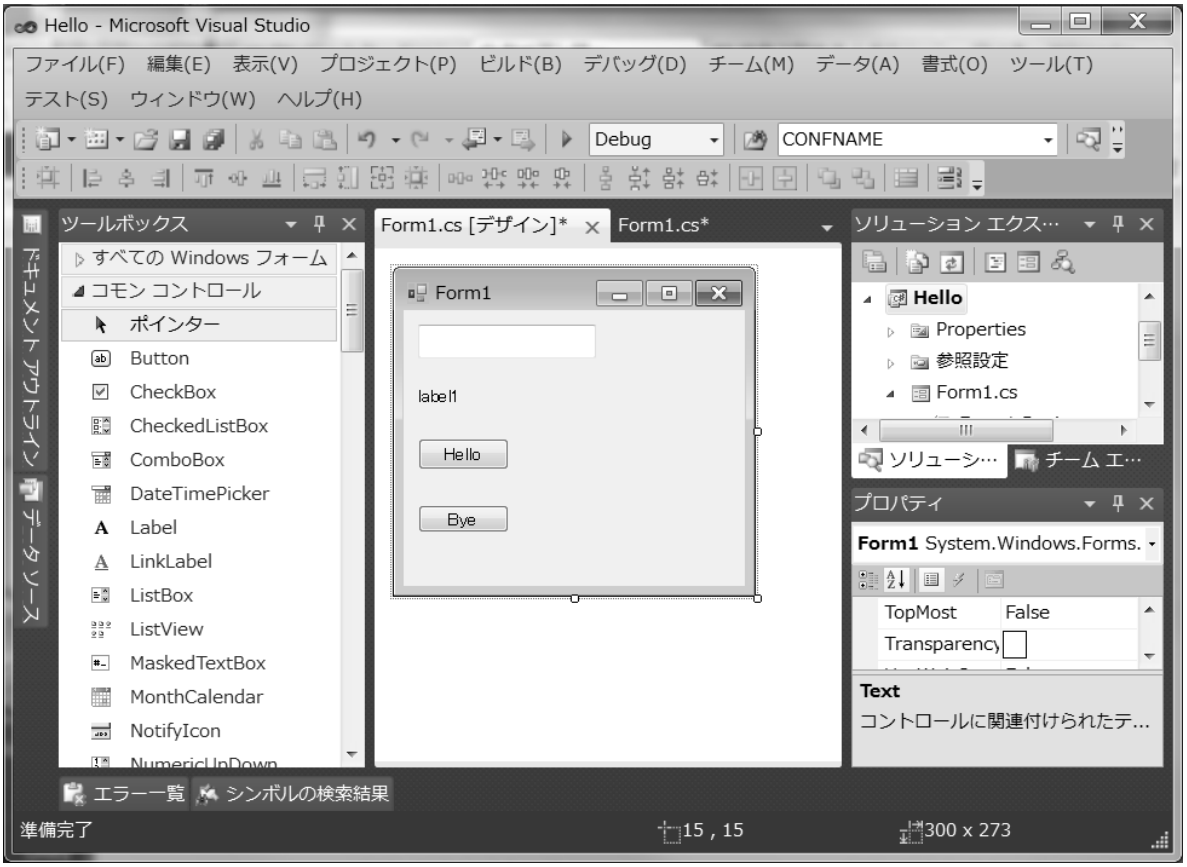


- エラー行が表示されるので、以下のようなミスがないか、探そう。
  - スペル間違い
  - 大文字小文字の間違い
  - 全角で入力していた
  - 行の最後の「;」が抜けている
  - 「;」と「:」の間違い
  - 「.」と「,」の間違い
  - エラーが大量に出る場合は｛や｝を消した可能性が高い
- エラー行以外でエラーが出る場合もあるので、エラー行に間違いが見つからない場合は、上下数行も確認する。よくあるミスは以下の通り。

演習：「Bye」ボタンに対応して、「さようなら」とメッセージを表示するプログラムを作成しよう。
”(ダブルクォーテーション)で囲まれた範囲では、全角や日本語の使用が可能である。

８．文字入出力コントロールの配置

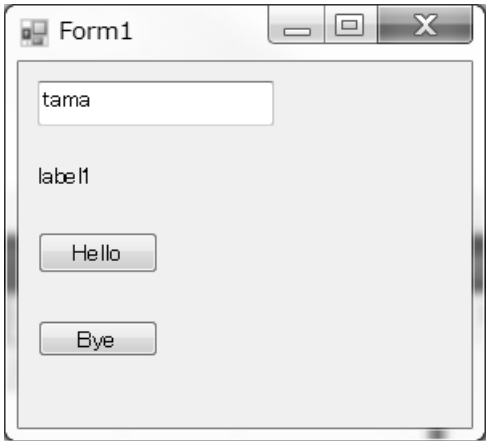
- 文字の入力には「TextBox」コントロールを使う。出力にも使える。
- 文字の表示だけを行うには「Label」コントロールを使う。
- 「Name」プロパティが、そのコントロールの名前である。



演習：TextBox と Label を一つずつ配置しなさい。それぞれのコントロールに自動的につけられる名前を確認しなさい。

９．文字入出力

- Textbox と Label は、「Text」プロパティに表示されている文字列を持っている。
- C# では「＋」記号で文字列をつなぐ。
- 「＝」記号で、右辺の値を左辺に代入する。



演習：Hello ボタンのクリック時処理に

```
label1.Text = "Hello, " + textBox1.Text;
```

と記述し、実行してみよう。（MessageBox は消去）

演習：Bye ボタンのクリック時処理を記述し、さようならメッセージがラベルに表示されるように変更しよう。

演習：Hello ボタンのクリック時処理で

```
textBox1.Text = "Hello, " + textBox1.Text;
```

とすると何が起きるか、予測し、実験してみよう。

１０．変数・数値変換

- 変数は、値を入れておく「箱」のようなもの。
- 「型名 変数名；」と宣言する。
- C#では「文字列(string)」と「数値(int,double)」とに代入互換性はない。
- 数値→文字列：数値変数の持っている ToString() を呼び出す。
- 文字列→数値（整数値）：int.Parse(文字列) を呼び出す。

演習：TextBox を２つ、ボタンを１つ配置して、「age」ボタンに次のコードを記述してみよう。

```
int age;
age = 2012 － int.Parse(textBox2.Text);
textBox3.Text = age.ToString();
```

textBox2, textBox3 は、適宜、自分のプログラムに合わせること。

変数には、記録したい値の種類に応じた型が存在する。一先ず、以下の３つの型を覚えておこう。

- string 型・・・「abc」や「こんにちは」といった文字列を入れておける。
- int 型・・・±21 億程度の範囲の整数を入れておける。
- double 型・・・実数を入れておける。

１１．ここまでのまとめ

演習：新しくプロジェクトを作成し、平成年と西暦年を変換表示するプログラムを作ってみよう。

ヒント

1. 「１０．変数・数値変換」の演習のプログラムを参考にせよ。
2. プログラムを組む前に、自分自身で、西暦から平成への変換や、その逆をどうやって求めるかを考えてみよ。

１２．条件分岐（１）

- C# の条件分岐は、if と switch の２種類ある。
- if は結果が２つに分かれる。
- 構文：

```
if ( 条件式 )
{
        条件が正しいときの処理
}
else
{
        条件が正しくないときの処理
}
```

- それぞれの処理が「１文だけ」のときは {} で括らなくてよい。
- else 以降は省略できる。
- if の処理の中に if が出てきてもよい。そのような場合、（if の数＋１）通りに結果は分かれる。

演習：TextBox を２つと Button を１つ配置し、ボタンに次のコードを記述してみよう。

```
int age;
age = int.Parse(textBox1.Text);
if (age >= 30)
{
        textBox2.Text = "出られる";
}
else
{
        textBox2.Text = "出られない";
}
```

演習：age に年齢を入力してから Check ボタンを押すと、その年齢でできることを表示する次のようなプログラムを作成しなさい。

１３．多重分岐

- ある変数の値に応じて、多くの異なった処理を行う場合、switch を使う。
- 構文：

```
switch ( 値 )
{
      case 値１：
            値１のときの処理
            break;
      case 値２：
            値２のときの処理
            break;
      ～～～ 以下同様 ～～～
      default ：
            それ以外のときの処理
            break;
}
```

演習：西暦年を入力して、十二支を出力するプログラムを作成しなさい。答えを１２個に分けるためには、１２で割ったあまりを利用するのは、占いのときと一緒。

```
int year;
year = int.Parse(textBox1.Text);
switch (year % 12)
{
      case 6:
            textBox2.Text = "とら";
            break;
      case 7:
            textBox2.Text = "う";
            break;
}
```

課題：年齢を入力してもらい、可能性のある十二支を２つ出力するプログラムを作成しなさい。

１４．繰り返し

- 特定の処理を繰り返す場合、for 文、while 文、do ～ while 文を用いる。
- 繰り返し回数が分かっているとき：

```
for( i=0; i<繰り返し回数; i++ )
{
        繰り返す処理
}
```

- ある数値範囲全てについて処理するとき：

```
for( i=初期値; i<=終了値; i++ )
{
        繰り返す処理
}
```

- ある条件が成り立っているあいだ繰り返すとき（ただし、一度も処理しなくても繰り返し条件が決まるとき）

```
while( 条件 )
{
        繰り返す処理
}
```

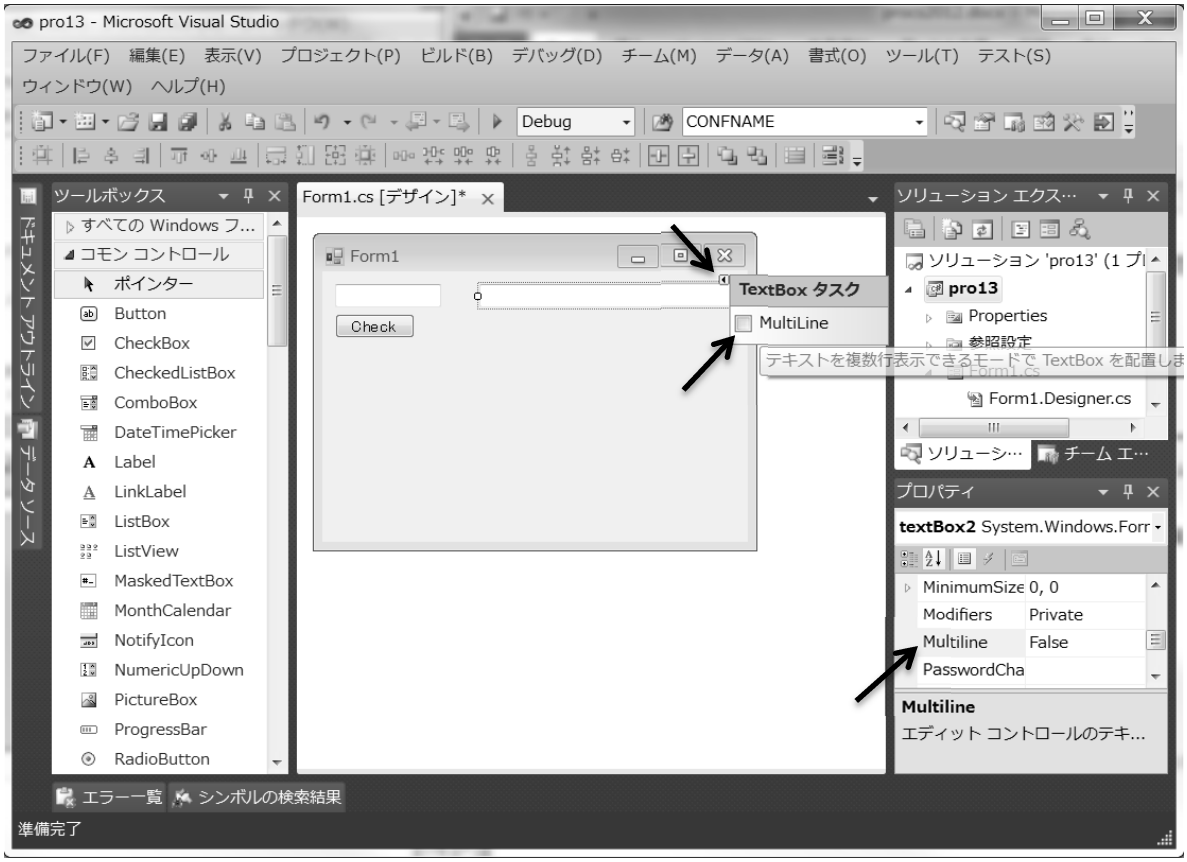
- ある条件が成り立っているあいだ繰り返すとき（ただし、一度処理を実行しないと繰り返し条件が定まらないとき）

```
do
{
        繰り返す処理
}
while( 条件 );
```

１５．繰り返し（回数が分かっているときの書き方）

演習：数値を入力して、その回数だけ「こんにちは」と表示するプログラムを作ってみよう。

テキストボックスを２つ、ボタンを一つ配置する。
複数行出力できるように、出力用のテキストボックスに「MultiLine」の属性を与える。




```
int i, n;
n = int.Parse(textBox1.Text);
for (i = 0; i < n; i++)
{
        textBox2.AppendText("こんにちは¥n");
}
```

「￥ｎ」は、Ｃ＃の文字列で「改行」を表す特殊表記。（￥ｎがないとどうなるか試してみよう。）

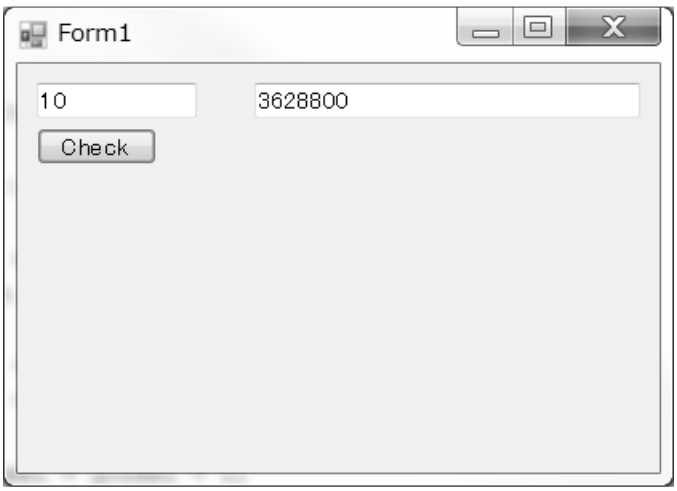
１６．繰り返し（値の範囲が分かっているときの書き方）

演習：数値を入力して、１からその数字までの合計を求めるプログラムを作ってみよう。

```
int i, n, goukei;
n = int.Parse(textBox1.Text);
goukei = 0;
for (i = 1; i <= n; i++)
{
      goukei = goukei + i;
}
textBox2.Text = goukei.ToString();
```

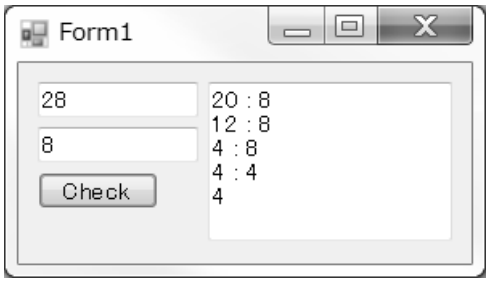
課題：数値を入力して、１からその数字までを全て掛け算するプログラムに変更してみよう。（これは「階乗」という計算で「！」記号で表す。たとえば、3! = 6, 10! = 3628800）

課題：このプログラムを使って、15! を計算してみよう。20! を計算してみよう。何が起きるだろうか？



１７．繰り返し（終了条件が処理前に決められる場合）

演習：入力された２つの数字のうち、「大きいほうの数字を小さいほうの数字分だけ減らす」ことを、両方が等しくなるまで繰り返し、そのときの数字を表示するプログラムを作成しなさい。(これは「ユークリッドの互除法」と呼ばれる計算手法を引き算だけで実現したもの)



※ 入力時点で両方の数字が等しければ、引き算の処理を行わなくても終了できる
→ while 構文

```
int n1, n2;
n1 = int.Parse(textBox1.Text);
n2 = int.Parse(textBox2.Text);
textBox3.Clear();
while (n1 != n2)
{
	if (n1 > n2)
		n1 = n1 - n2;
	else
		n2 = n2 - n1;
	textBox3.AppendText(
		n1.ToString() + " : " + n2.ToString() + "¥n");
}
textBox3.AppendText( n1.ToString() );
```

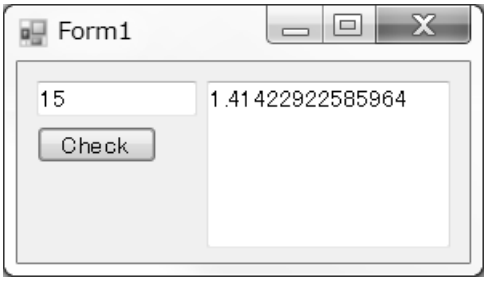
※ この例では textBox3 を MultiLine にして経過を見られるようにした。
※ 「等しくない」は、C＃では「！＝」で表す。
※ while 文の処理の中に if 文が入ってもよい。このように、命令をうまく組み合わせて目的の動作を実現していく。

課題：「大きいほうの数字を、『大きいほうの数字を小さいほうの数字で割った余り』にする」のが本来の計算方法です。本来の計算方法で計算するよう修正してみよう。

課題：10000 以上の数で、お互いに素である（この計算結果が１になる）ような数字のペアを１つ見つけてみよう。

１８．繰り返し（終了条件が処理前に決められない場合）

演習：「x を適当な数から始めて、次の x を x＝(2+x*x)/(2*x) で求める。新しい x と古い x との差の絶対値が 0.001 以下になったら終了（差の絶対値が 0.001 より大きいあいだ繰り返し）し、その x の値を表示する」というプログラムを作成しなさい。（これは「ニュートン法」と呼ばれる手法による方程式の解法）



※ 次の x を求めないと、終了してよいかどうか分からない
→ do ～ while 構文

```
double x, newx;
textBox2.Clear();
newx = double.Parse(textBox1.Text);
do
{
        x = newx;
        newx = (2 + x * x) / (2 * x);
}
while (Math.Abs(newx - x) > 0.001);
textBox2.Text = x.ToString();
```

※ 小数を扱うときは double 型の変数を用いる。
※ 差の絶対値は Math.Abs 関数を用いて求める。
※ do ～ while には最後に「；」が必要。

課題：「次の x を求める式」を x＝(3+x*x)/(2*x)、x＝(5+x*x)/(2*x) などとして答を求めてみよう。