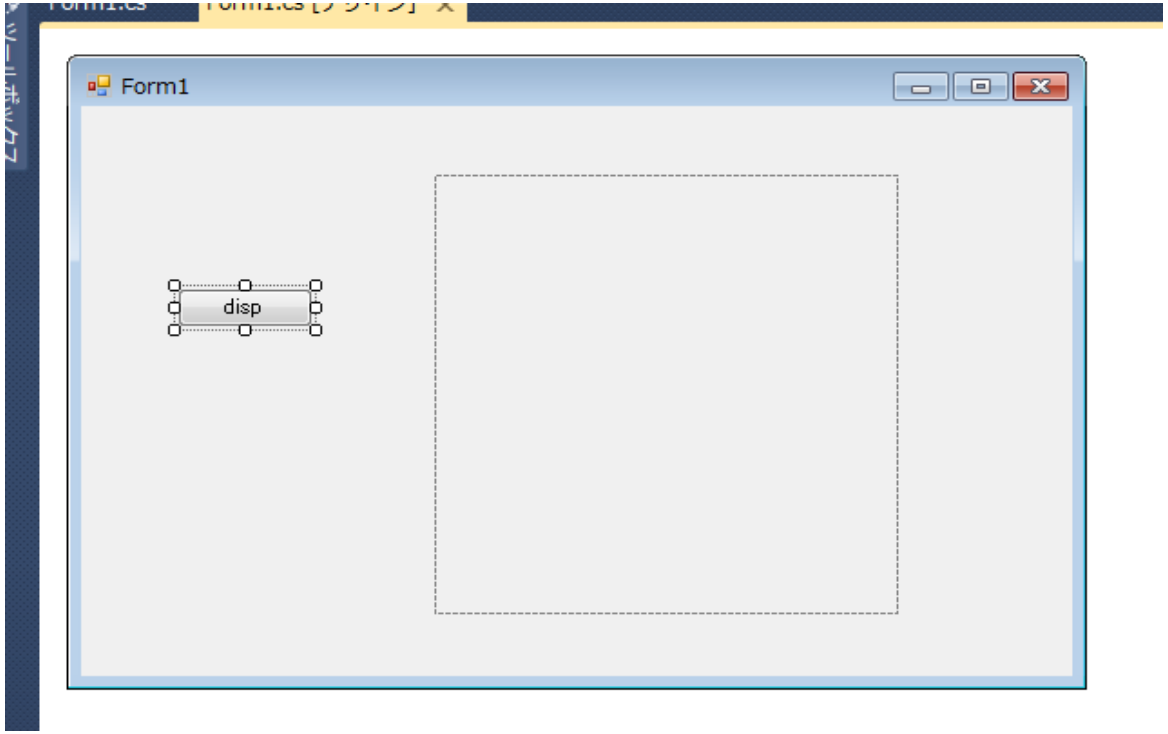
C#の基本
～ファイル読み込み～

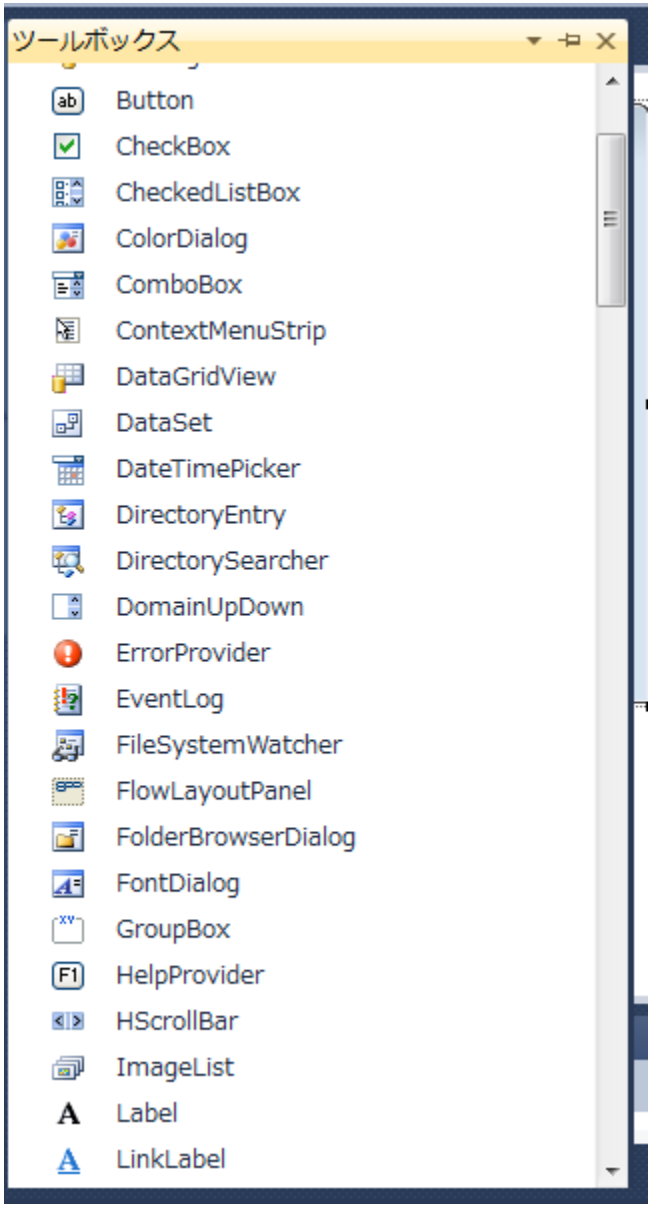
今回学ぶ事

- 今回はファイル読み書きに必要BinaryReaderクラスについて記載する。
- ファイル参照ダイアログであるOpenFileDialogクラスについても理解を深める。
- また、Bitmapクラスを用いたBitmapファイルの読み込み方法についても学ぶ。

フォーム作り

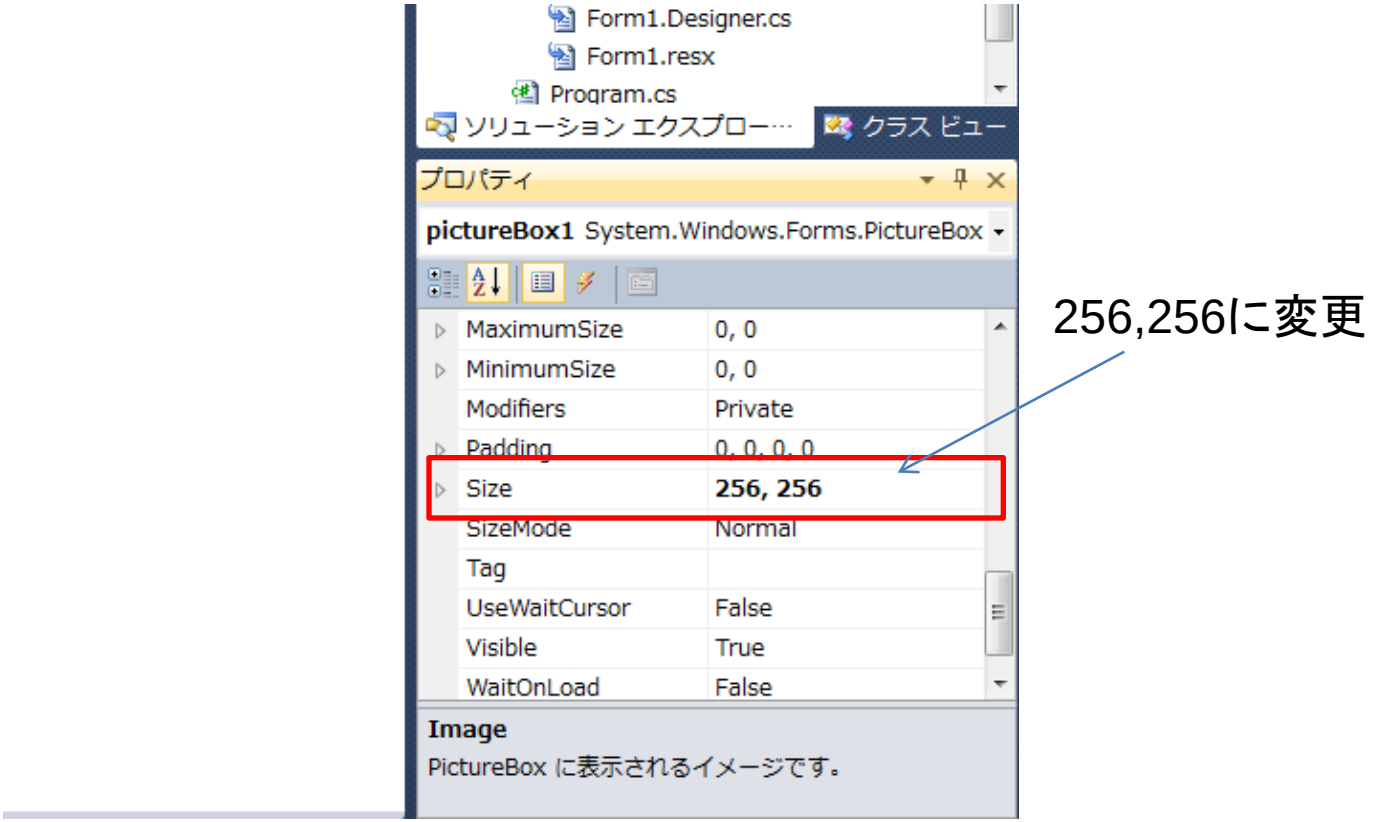
- まず、label、pictureBoxを配置する。
- ツールボックスより左クリックで選択する。
- そのまま、フォーム上を左クリックすると設置することができる。



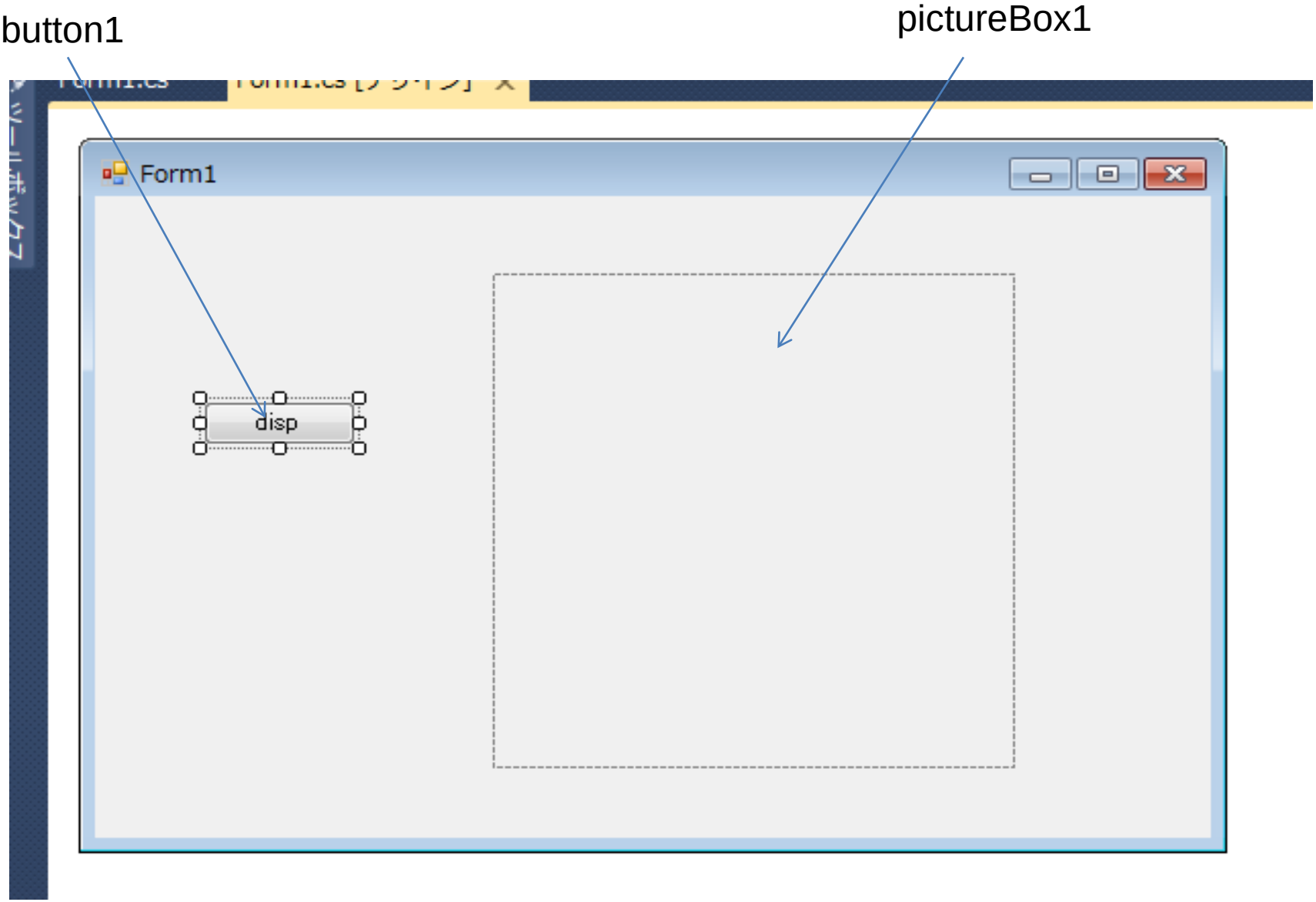


フォーム作り

- pictureBoxを右クリック→プロパティより”Size”の値を”256,256”に変更する。

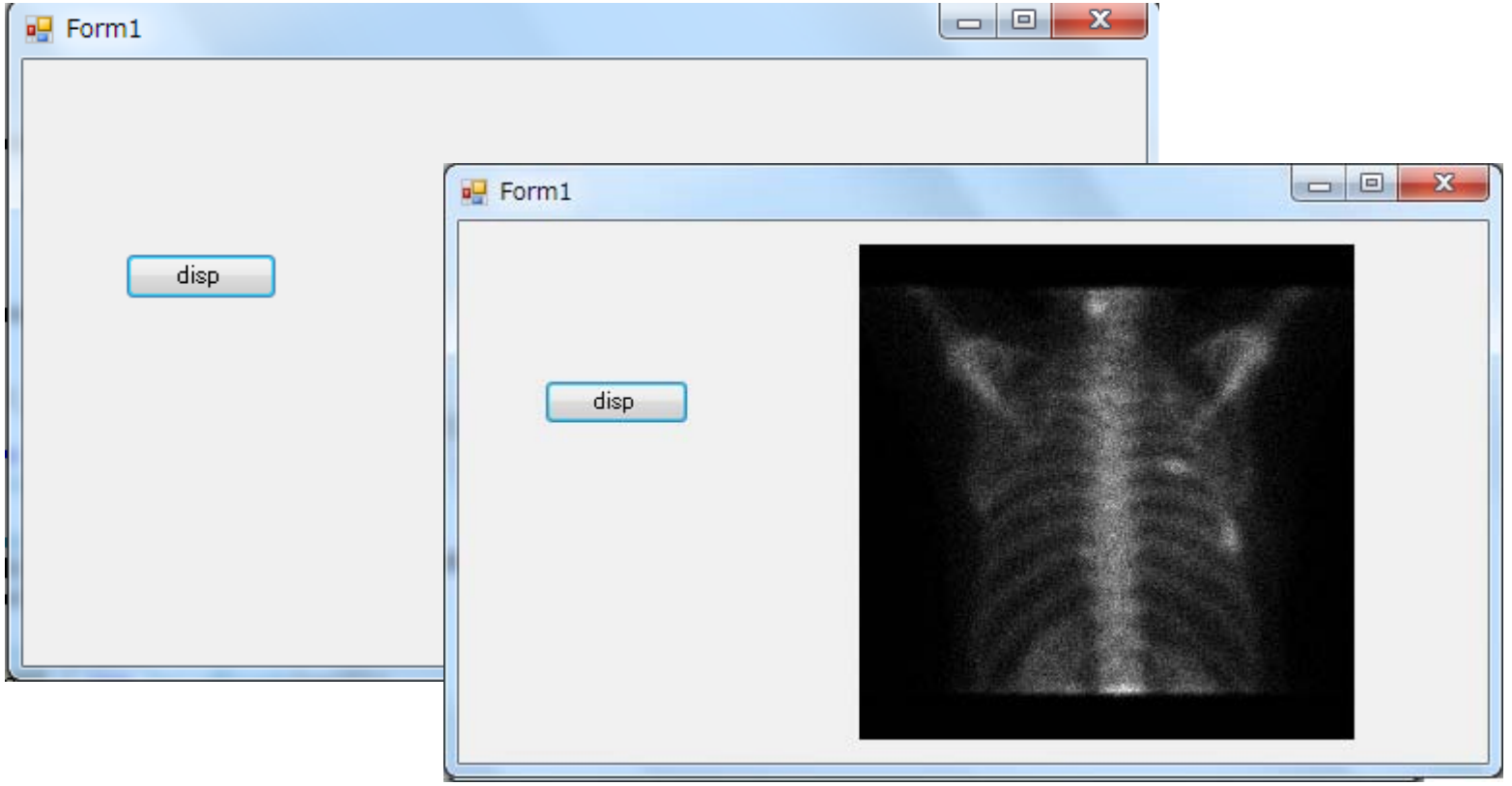


フォーム作り



今回作るプログラム

バイナリデータを読み込んで画像を表示するプログラム。
ファイルにはプログラムフォルダ内のbone1~6を使用する。



コード記述

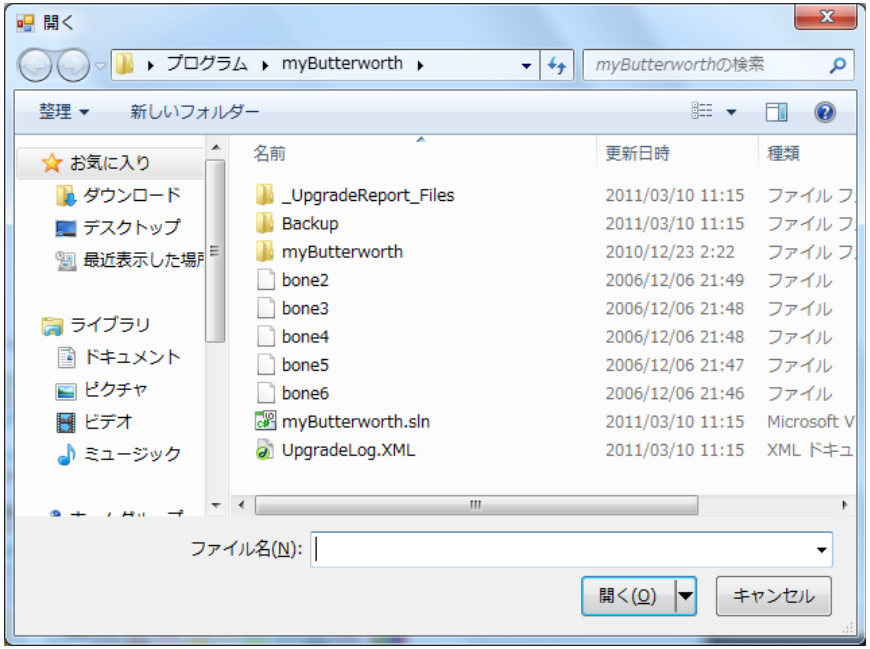
- button1をダブルクリックし、
  Private void button1_Click(object sender, EventArgs e
  を表示させる。
- コード全体に関してはprogram4aの

Private void button1_Click(object sender, EventArgs e)
の中を参照する。

コード解説　OpenFileDialogクラス

OpenFileDialog ofd = new OpenFileDialog();

ファイル参照のダイアログのクラス。つまり、
以下の様なフォームの事



コード解説

```
int[,] img = new int[256, 256];
```

256×256の2次元の配列を宣言する。

（Lecture3で解説済み）

コード解説　ShowDialogメソッド

if (ofd.ShowDialog() == DialogResult.OK){～}

OpenFileDialogクラスであるofdをShowDialogメソッドを用いることで表示させる事が出来る。

また、if (ofd.ShowDialog() ==DialogResult.OK)は「ダイアログが開かれた時に、何かファイルが選ばれたなら以下の処理を行え」という判断を行っている。

コード解説　BinaryReaderクラス

```
BinaryReader　B　=　new　BinaryReader(ofd.OpenFile());
```

- BinaryReaderクラスは読み込んだファイルを1バイト、2バイトずつ等形式を決めて読み込む。

- この文は、OpenFileDialogクラスで選択されたファイルからデータを読み込むBというクラスを宣言したことに相当する。

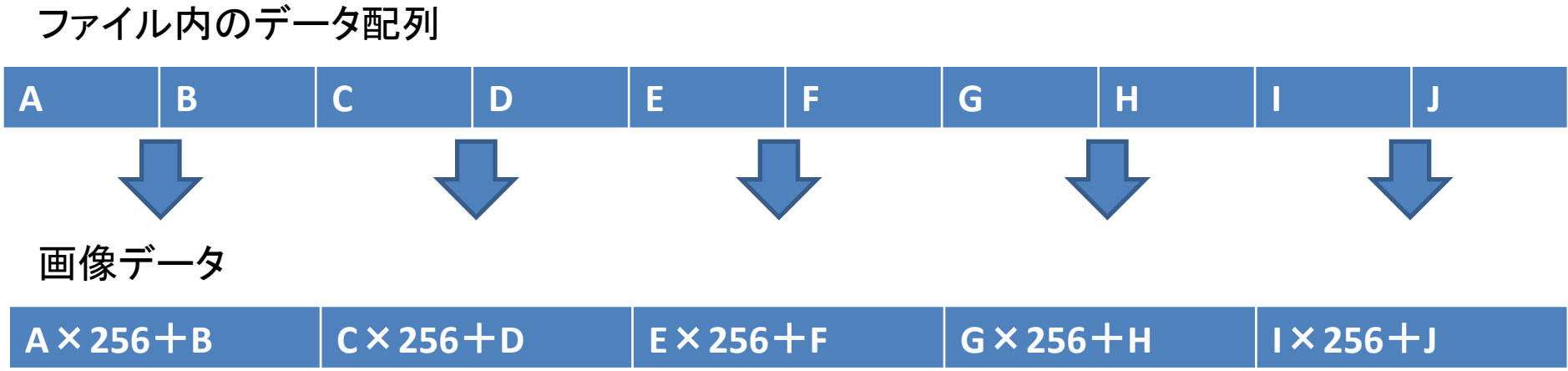
コード解説　BinaryReaderクラス

```
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;
using System.IO;
```

- このクラスを使うためにはコード上部にSystem.IO名前空間を追加する必要がある。
- System.IO名前空間はファイル入出力に関するクラスが記載されている。

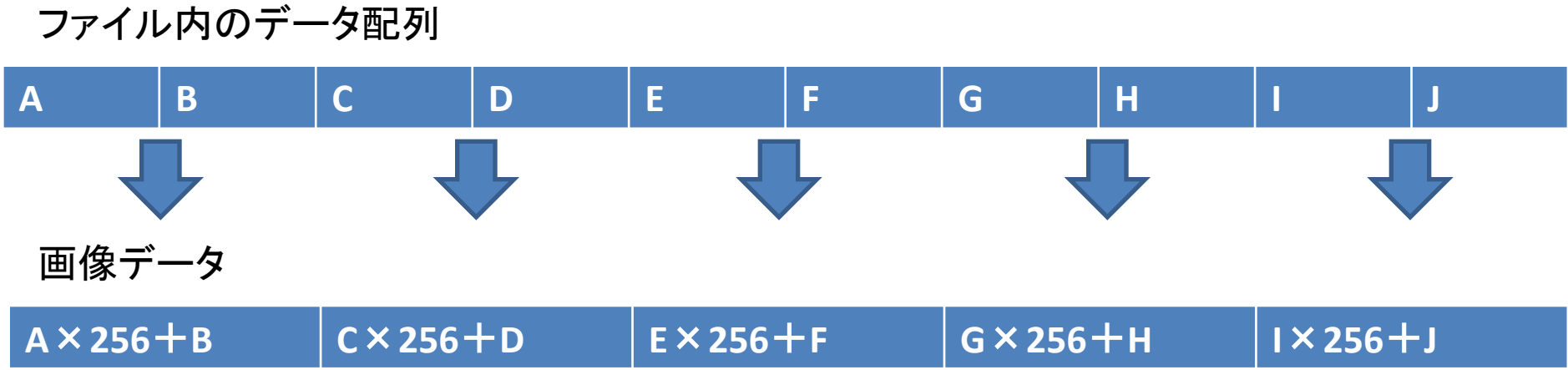
ファイルの構造

- 以下にboneファイルの構造を図示する。



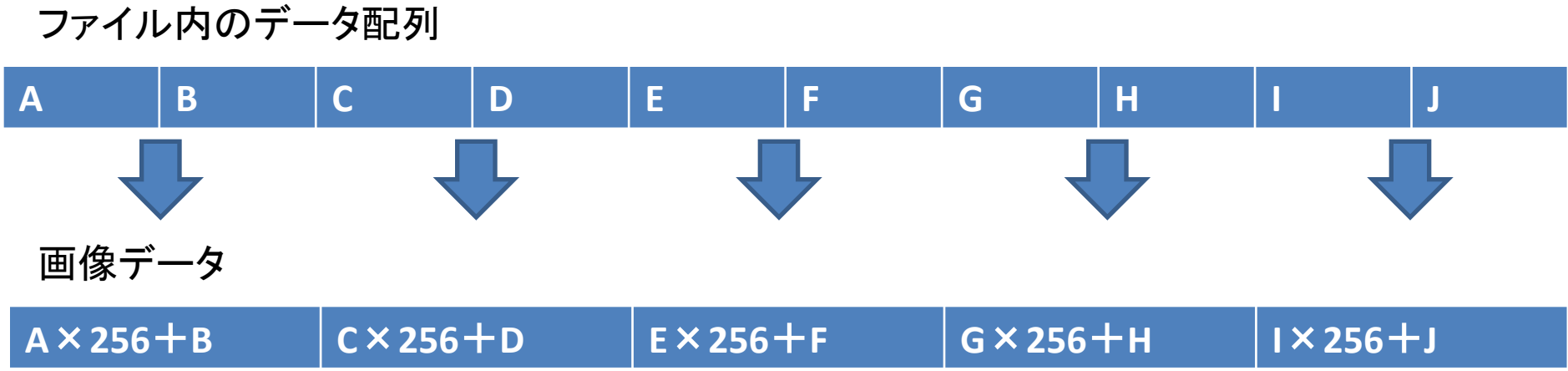
- ファイルはすべて1バイトのデータの配列になっている。
- そのため、表示する画像データの値を得るためには読み込み方を工夫しなければならない

ファイルの構造　ReadByteメソッド



- ReadByteメソッドはファイルから1Byte読み込み、次に読む場所を一つ進めるという働きをする関数である。
- 例えば、ReadByteメソッド1回目はAが読み込まれ、2回目はBが読み込まれる。

ファイルの構造　ReadByteメソッド



- 今回欲しい情報はA×256＋Bであるため

  B.ReadByte() * 256 + B.ReadByte();

と書くことができる。

- これを繰り返すと欲しい情報を得ることができる

コード解説

- 以上の事を理解すると、以下の様なコードで256×256の画像の配列imgを得ることができると分かる。

```
for (int j = 0; j < 256; j++)
  {
    for (int i = 0; i < 256; i++)
    {
      img[i, j] = B.ReadByte() * 256 + B.ReadByte();
    }
  }
```

コード解説

B.Close();

- BinaryReaderクラスは読み込み終わったらファイルを閉じる必要がある。
- ファイルを壊さないように必ず入れるようにする。

コード解説

```
Int    max = 0;

       for (int j = 0; j < 256; j++)
       {
          for (int i = 0; i < 256; i++)
          {
             if (max < img[i, j]) max = img[i, j];
          }
       }
```

- このコードではまず、整数0が代入されているmaxを宣言する。
- そこに、img一つ一つに対し、maxより大きければ値を更新していく。
- つまり、maxにはimgの最大値が入る。

コード解説

```
Bitmap bmp = new Bitmap(256, 256);

    for (int j = 0; j < 256; j++)
    {
        for (int i = 0; i < 256; i++)
        {
            int count = img[i, j] * 255 / max;
            Color col = Color.FromArgb(count, count, count);
            bmp.SetPixel(i, j, col);
        }
    }
```

- 赤字以外は第3スライドで説明済み

コード解説

```
int count = Img[i, j] * 255 / max;
```

- 色の値は0～255でありそれを超えるとエラーとなる。
- そのため、この行では画像の値の最大値を255に縮小している。

コード解説

```
pictureBox1.Image = bmp;
pictureBox1.Refresh();
```

- 画像の描画（第3回スライドで説明済み）

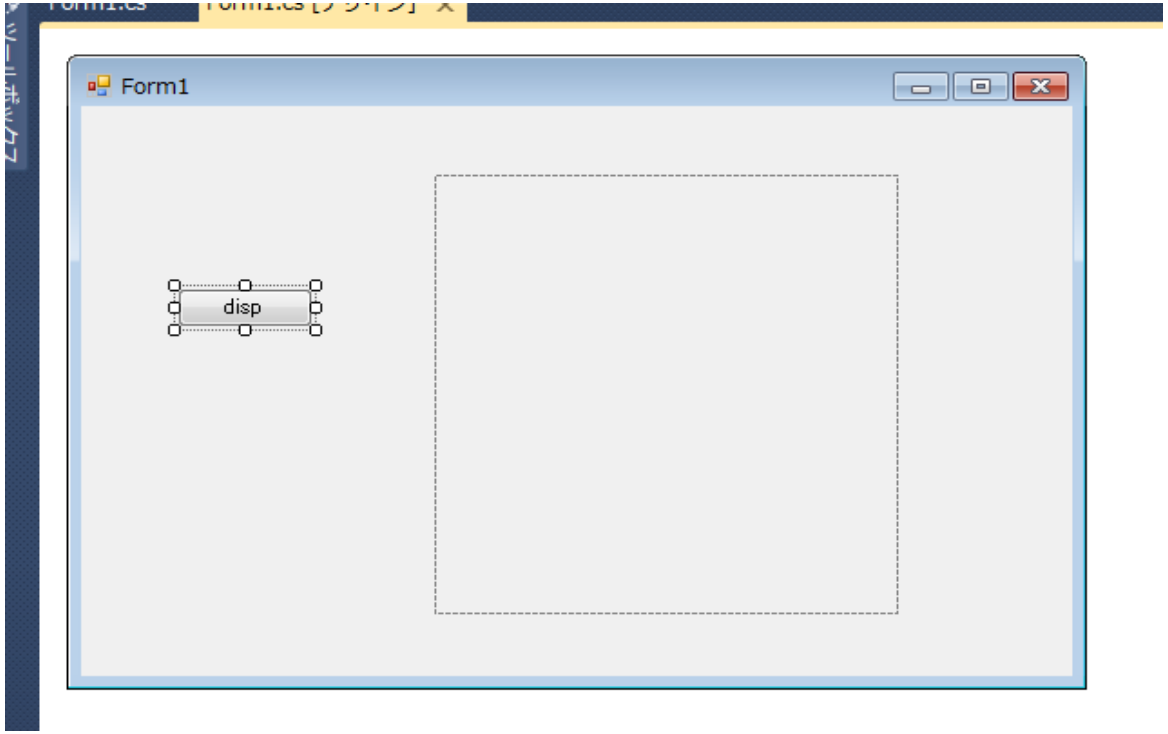
Bitmapファイルの読み込み

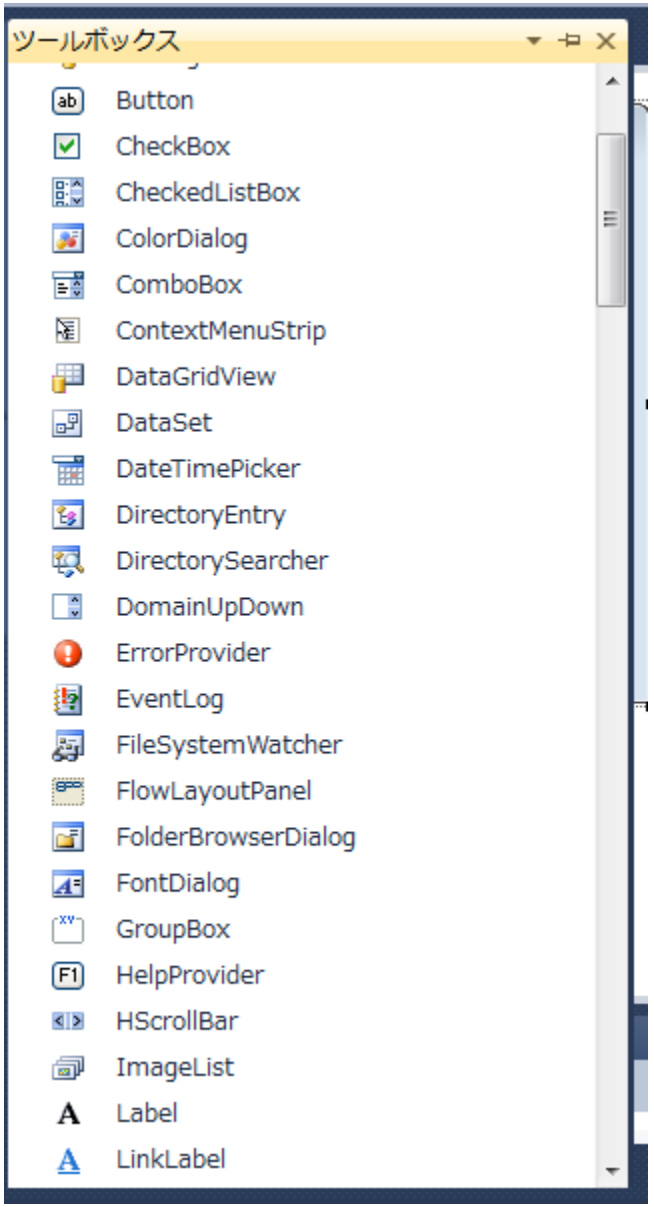
- Bitmap、JPEG、PNG・・等の代表的な画像フォーマットに関してはBitmapクラスを用いることにより簡単に読み取ることができる。

- 次は、このような形式の決まっている画像ファイルの読み取り方法について解説する。

フォーム作り

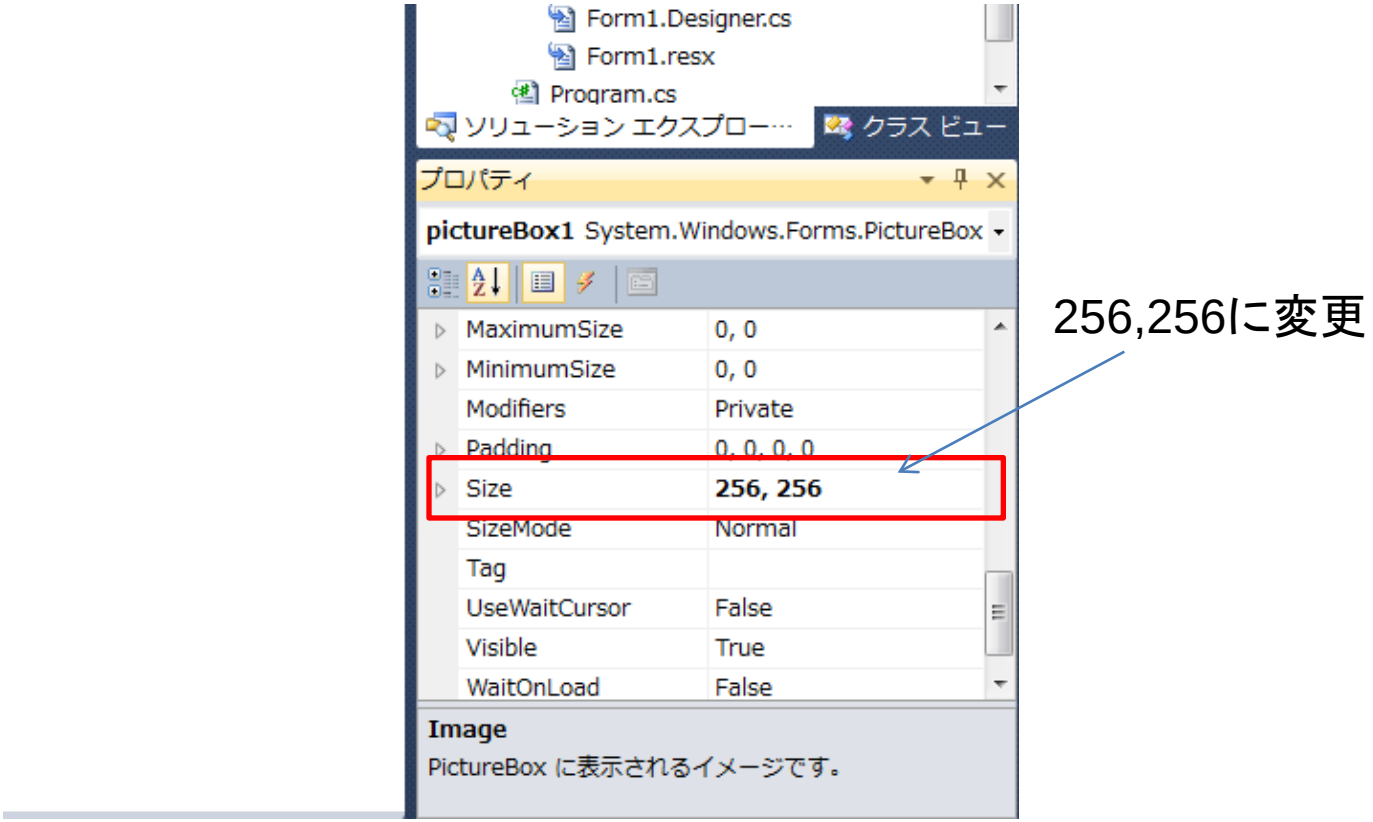
- まず、label、pictureBoxを配置する。
- ツールボックスより左クリックで選択する。
- そのまま、フォーム上を左クリックすると設置することができる。



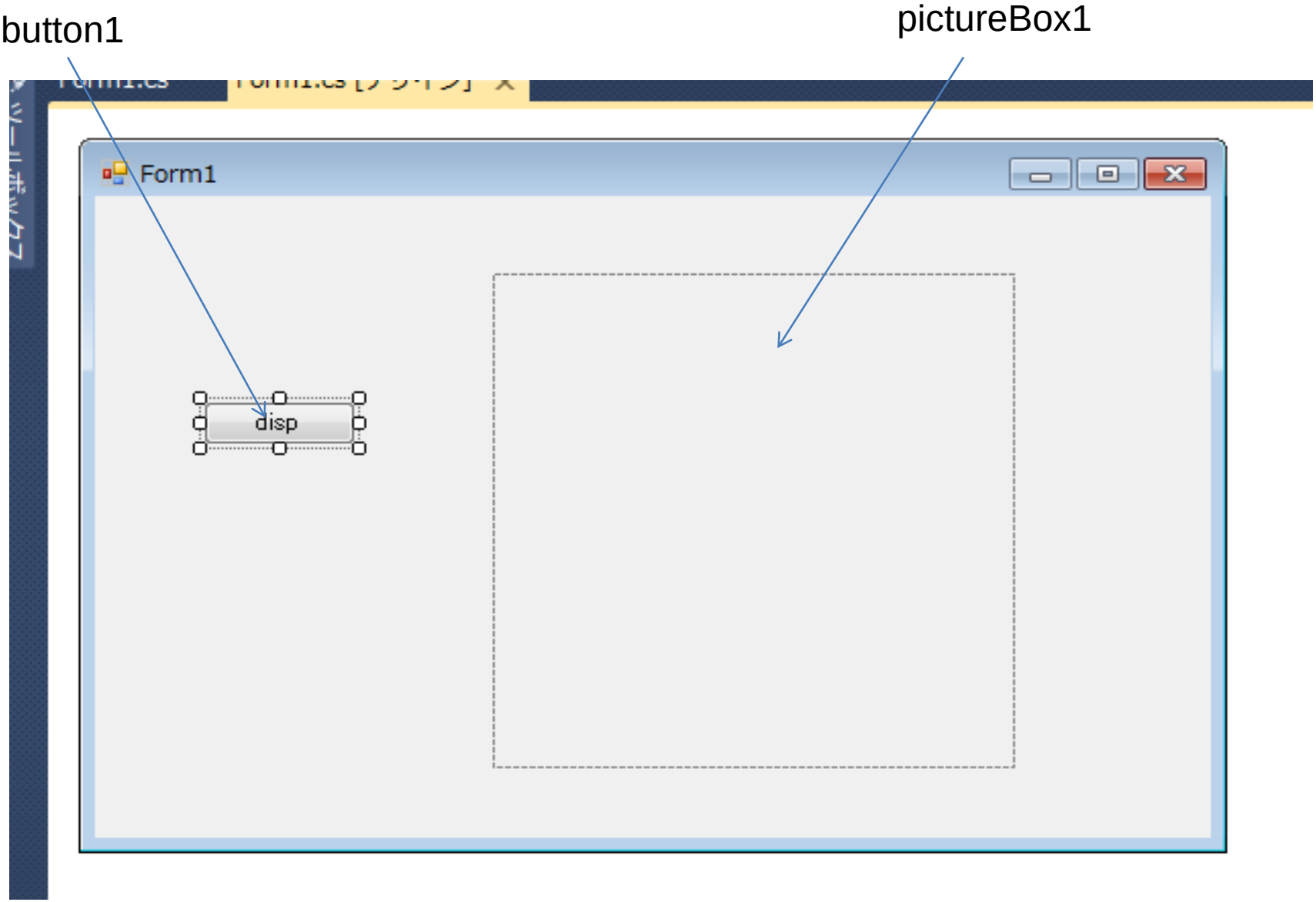


フォーム作り

- pictureBoxを右クリック→プロパティより”Size”の値を”256,256”に変更する。

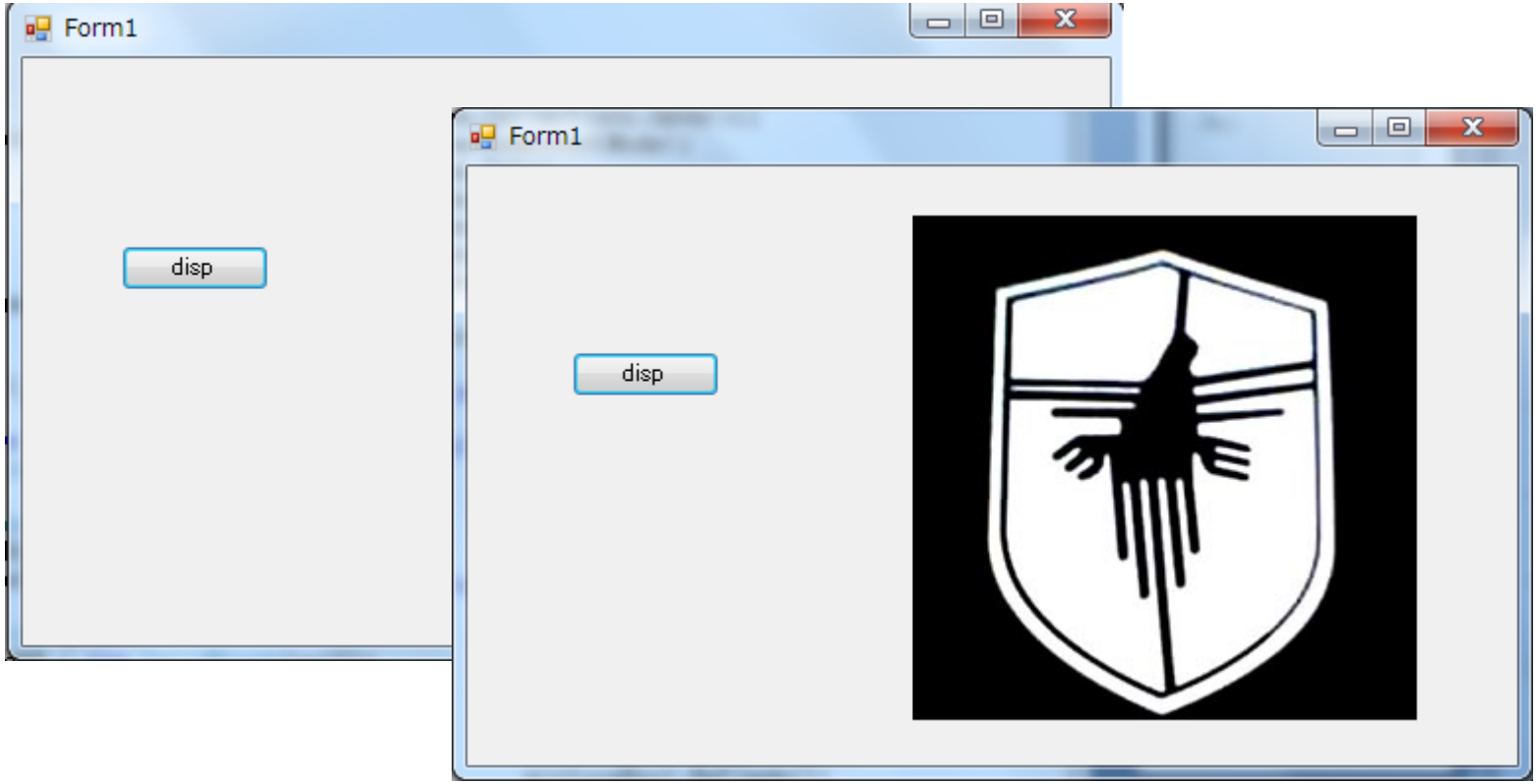


フォーム作り



今回作るプログラム

- 選択したBitmap,JPEGファイルの画像を描画するプログラムを作る
- 選択するファイルはBitmap,JPEGファイルなら何でも良い。
- 以下に、プログラムフォルダ内のsample.jpgを表示させた場合の例を示す。



コード記述

- 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

private void button1_Click(object sender, EventArgs e)の中に以下の様なコードを記述する。

```
private void button1_Click(object sender, EventArgs e)
    {
        OpenFileDialog ofd = new OpenFileDialog();
        if (ofd.ShowDialog() == DialogResult.OK)
        {
            Bitmap bmp = new Bitmap(ofd.FileName);
            pictureBox1.Image = bmp;
            pictureBox1.Refresh();

        }
    }
```

コード解説

```
OpenFileDialog ofd = new OpenFileDialog();
if (ofd.ShowDialog() == DialogResult.OK)
{
        ～
}
```

- ファイル参照ダイアログのofdを宣言し、“開く”を行ったなら処理を行いなさいという意味になる。（すでに解説済み）

コード解説　Bitmapクラス

- Bitmap bmp = new Bitmap(ofd.FileName);

- ここまで、Bitmapファイルは（幅、高さ）で初期化していた。
- しかし、ここでは選んだ画像で初期化している。
- このように、初期化の方法は1つではなく様々な方法がある。（この仕組みをオーバーロードと呼ぶ）

コード解説　OpenFileDialog

- 前半の方法ではOpenFileメソッドを使用していたが、今回の方法ではFileNameプロパティを使用している。
- これはどの様に使い分ければ良いのだろうか？

コード解説　OpenFileDialog

- OpenFileメソッドは主に1バイトずつ等形を決めて読み取る場合に用いる。（BinaryReaderクラスを使いたいときと覚えても良い）
- FileNameプロパティはファイルのパス（例：C¥Users¥desktop)が欲しい時に用いる。（Bitmapクラスを初期化するときにはこちらが必要になる）
- 詳しい使い分けは、Microsoftのリファレンスを確認する

コード解説　OpenFileDialog

- また、OpenFile()に対しFileNameには()が無い。

- これは、メソッドとプロパティの違いでありこれは使いたいコードがどちらかによって()の有無が変わる。

コード解説

```
pictureBox1.Image = bmp;
pictureBox1.Refresh();
```

- 選択した画像を示しているBitmapクラスを描画させている。

まとめ

- 今回はファイル処理について解説を行った。
- ここでは、バイナリファイルの処理方法についてしか扱わないが、テキストファイルの読み書きを行うクラスを数多くサポートされているため興味のある人は調べてみてください。
- また、Bitmap、JEPG等の形式の決まった画像ファイルの扱い方の解説も行った。