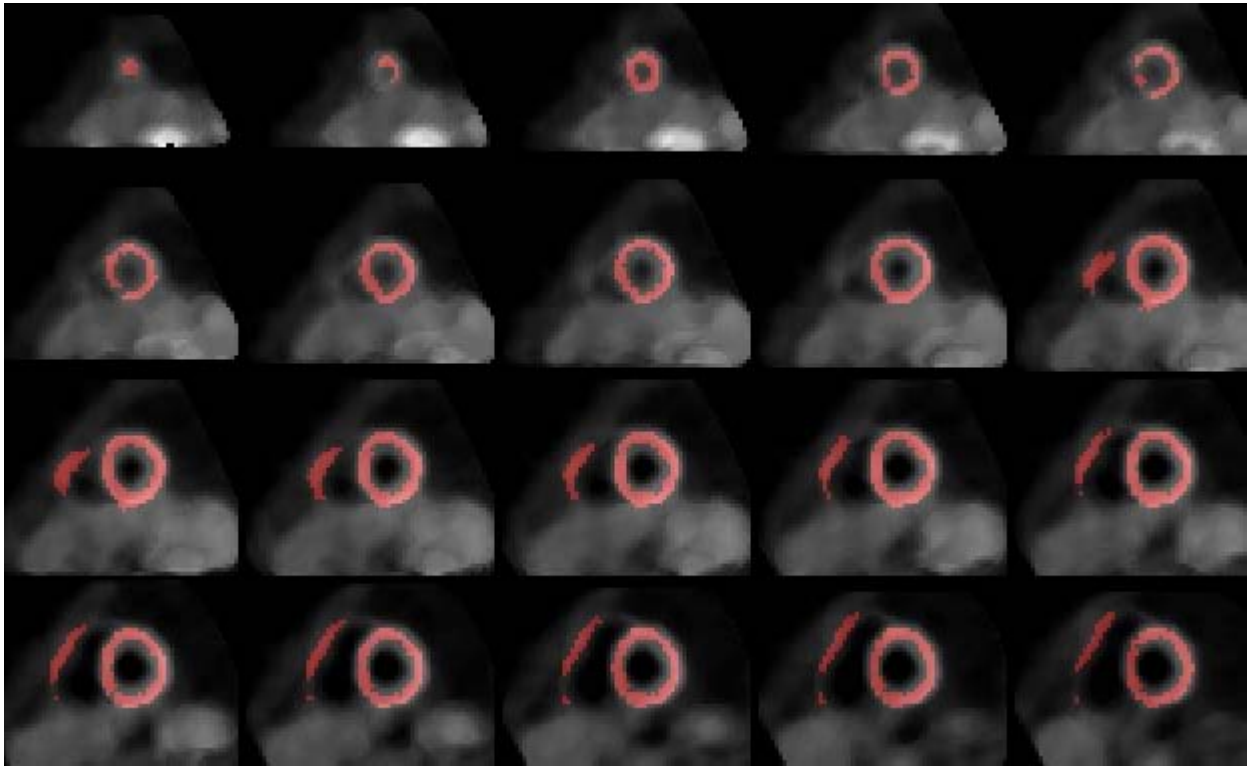
Labelling

Labelling

ある領域をcheckすることをLabellingという。

関心領域ROIは、画像のうち着目したい場所（腫瘍など）をLabellingしたものである。

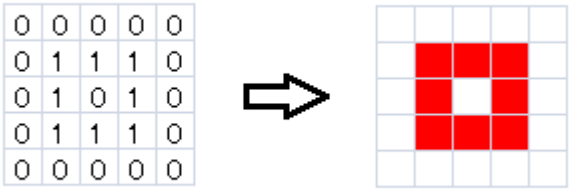
今回は labelling について学び、基本的なコーディングはC++と変わらないということを実感することを目標としたい。



心筋にとったROI。

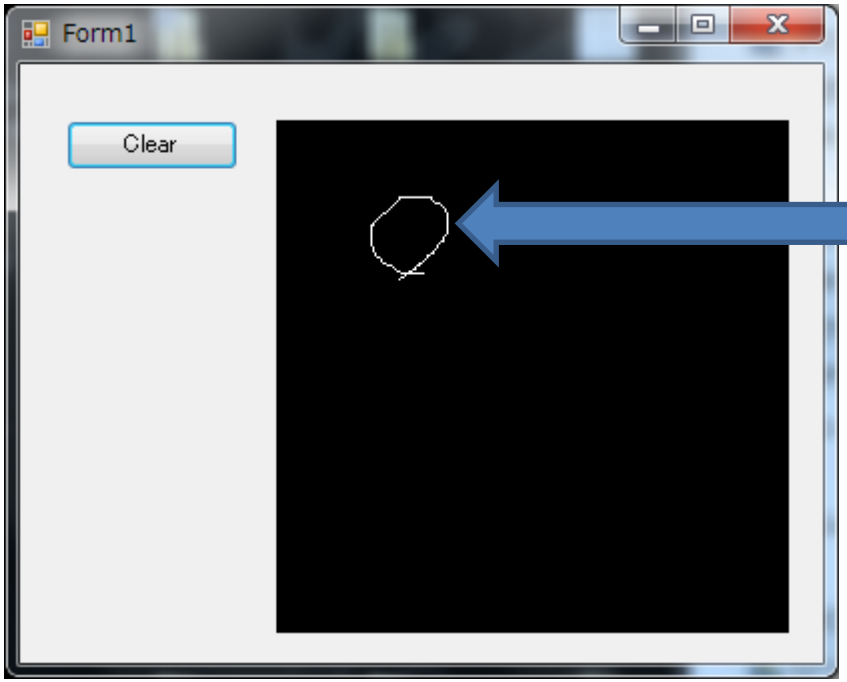
画像配列とは別の配列にROI情報があり、重ね合わせて表示している。

赤色領域＝１　　その他＝０



前回のプログラムから(配列の方)

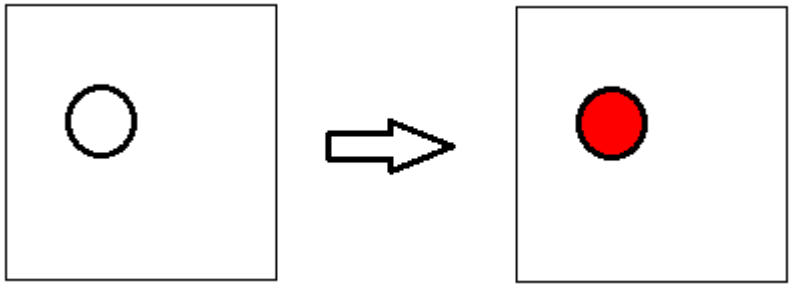
閉じた領域をLabellingしたい。



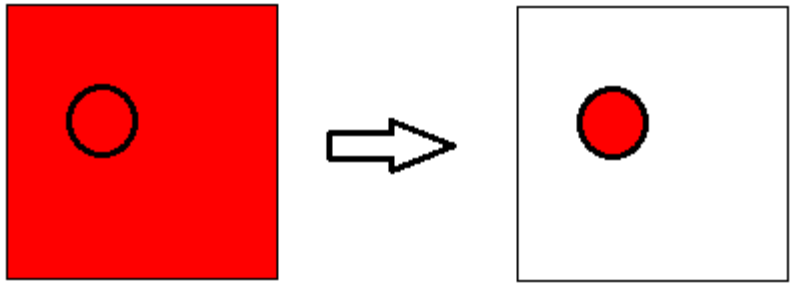
この中

考え方

①中から広げる



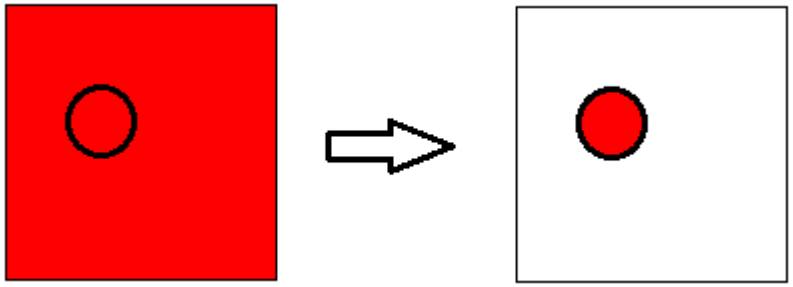
②配列全てに1を入れたのち、周りをけずる



①はクリックなどによる情報が必要。（なくてもできるかも）

今回は②の方が簡単にできそう。

考え方

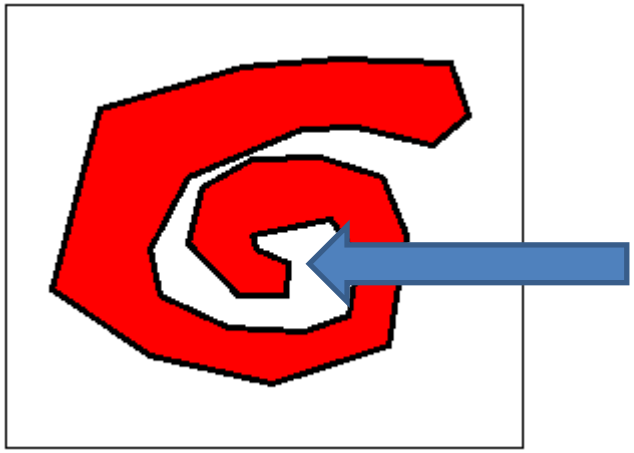


単純に

左→右、右→左、上→下、下→上

と走査して、円（ROIの境界）にぶつかったら終わりという

やり方だと、網羅できない領域が出る恐れがある。

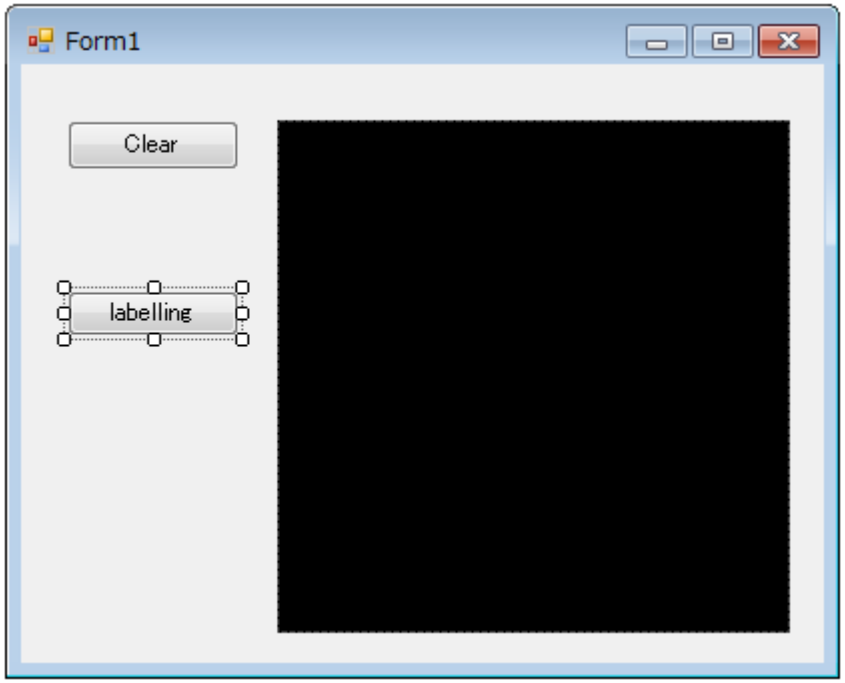


ここまでたどり着かない

可能性あり

Labelling

まずはLabellingボタンを準備。



Labelling

前回のコードのどこかに下を追加。

```
private void button2_Click(object sender, EventArgs e)//labelling
{
    int[,] label = new int[pictureBox1.Width, pictureBox1.Height];
    for (int j = 0; j < pictureBox1.Width; j++)
    {
        for (int i = 0; i < pictureBox1.Height; i++)
        {
            label[i, j] = 1;
        }
    }
}
```

少し格好つけた。.Width, .Height でサイズが入る。

結局は256x256。

やっていることは、全てに1が入っている配列の準備。

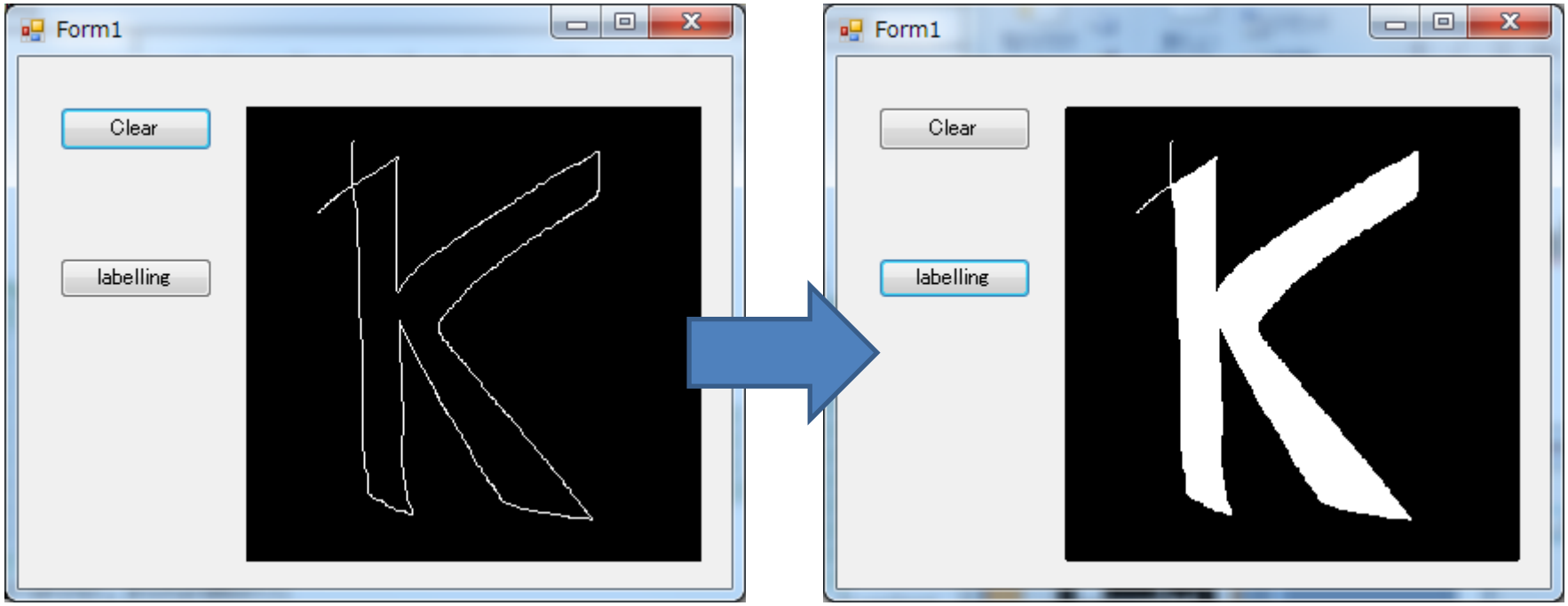
new した段階では0が入っている。

Labelling

```
private void button2_Click(object sender, EventArgs e)//labelling
{
    int[,] label = new int[pictureBox1.Width, pictureBox1.Height];
    for (int j = 0; j < pictureBox1.Width; j++)
    {
        for (int i = 0; i < pictureBox1.Height; i++)
        {
            label[i, j] = 1;
        }
    }
    label[1, 1] = 0;
    while (true)
    {
        int counter = 0;
        for (int j = 1; j < pictureBox1.Width - 1; j++)
        {
            for (int i = 1; i < pictureBox1.Height - 1; i++)
            {
                if (label[i, j] == 0)
                {
                    if (image[i + 1, j] == 0 && label[i + 1, j] == 1) { label[i + 1, j] = 0; counter++; }
                    if (image[i - 1, j] == 0 && label[i - 1, j] == 1) { label[i - 1, j] = 0; counter++; }
                    if (image[i, j + 1] == 0 && label[i, j + 1] == 1) { label[i, j + 1] = 0; counter++; }
                    if (image[i, j - 1] == 0 && label[i, j - 1] == 1) { label[i, j - 1] = 0; counter++; }
                }
            }
        }
        if (counter == 0) break;
    }
    D.Disp(label, pictureBox1);
}
```

Labellingコード。image == 1 でなければ、roi = 0 領域を拡大する、といった内容。
counterで拡大程度を見ており、もう十分広がった（counter = =0）となったときに、whileループをぬける。
roi[1, 1] = 0 は、「隅は関係ないだろう」という適当な考えのもとの始点。

Labelling



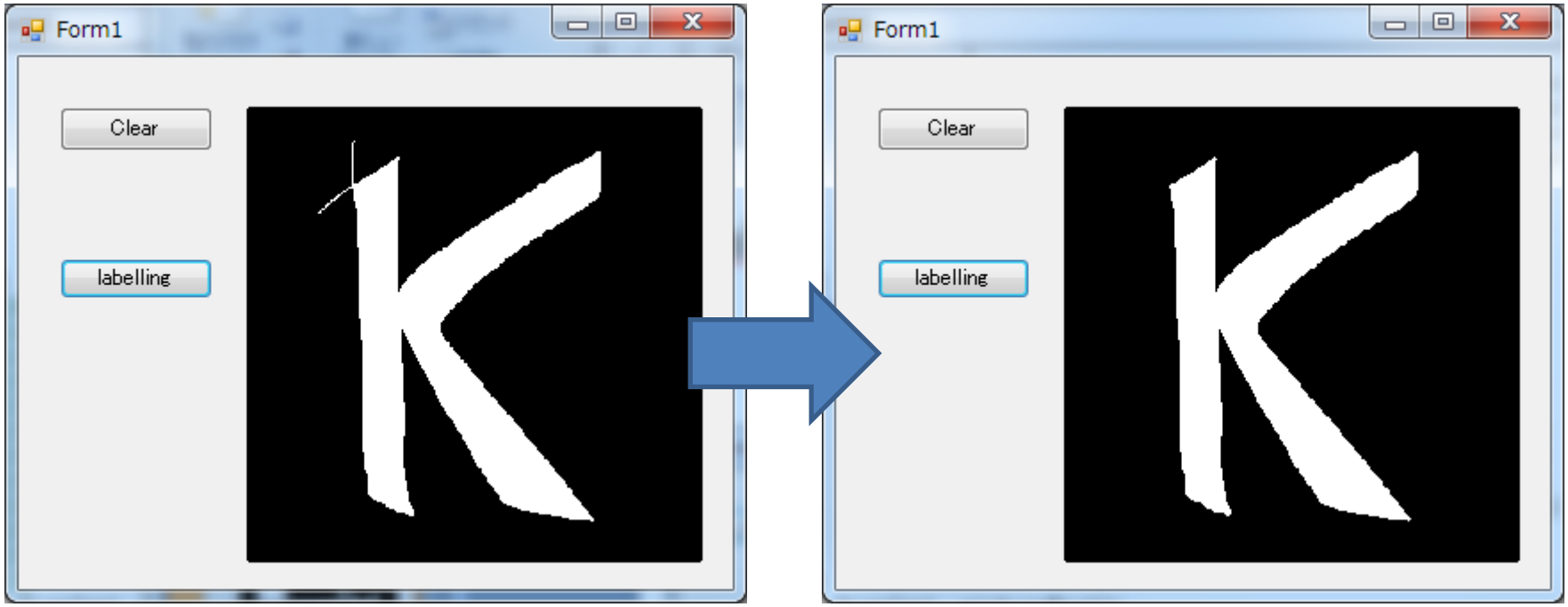
細かいが、左上のでっぱりが気になる。

Labelling

```
private void button2_Click(object sender, EventArgs e)//labelling
{
    int[,] label = new int[pictureBox1.Width, pictureBox1.Height];
    for (int j = 0; j < pictureBox1.Width; j++)
    {
        for (int i = 0; i < pictureBox1.Height; i++)
        {
            label[i, j] = 1;
        }
    }
    label[1, 1] = 0;
    while (true)
    {
        int counter = 0;
        for (int j = 1; j < pictureBox1.Width - 1; j++)
        {
            for (int i = 1; i < pictureBox1.Height - 1; i++)
            {
                if (label[i, j] == 0)
                {
                    if (image[i + 1, j] == 0 && label[i + 1, j] == 1) { label[i + 1, j] = 0; counter++; }
                    if (image[i - 1, j] == 0 && label[i - 1, j] == 1) { label[i - 1, j] = 0; counter++; }
                    if (image[i, j + 1] == 0 && label[i, j + 1] == 1) { label[i, j + 1] = 0; counter++; }
                    if (image[i, j - 1] == 0 && label[i, j - 1] == 1) { label[i, j - 1] = 0; counter++; }
                }
            }
        }
        if (counter == 0) break;
    }
    for (int j = 0; j < pictureBox1.Width; j++)
    {
        for (int i = 0; i < pictureBox1.Height; i++)
        {
            if (image[i, j] == 1
                && (label[i - 1, j] == 0 || image[i - 1, j] == 1)
                && (label[i + 1, j] == 0 || image[i + 1, j] == 1)
                && (label[i, j - 1] == 0 || image[i, j - 1] == 1)
                && (label[i, j + 1] == 0 || image[i, j + 1] == 1))
                label[i, j] = 0;
        }
    }
    D.Disp(label, pictureBox1);
}
```

← 追加。

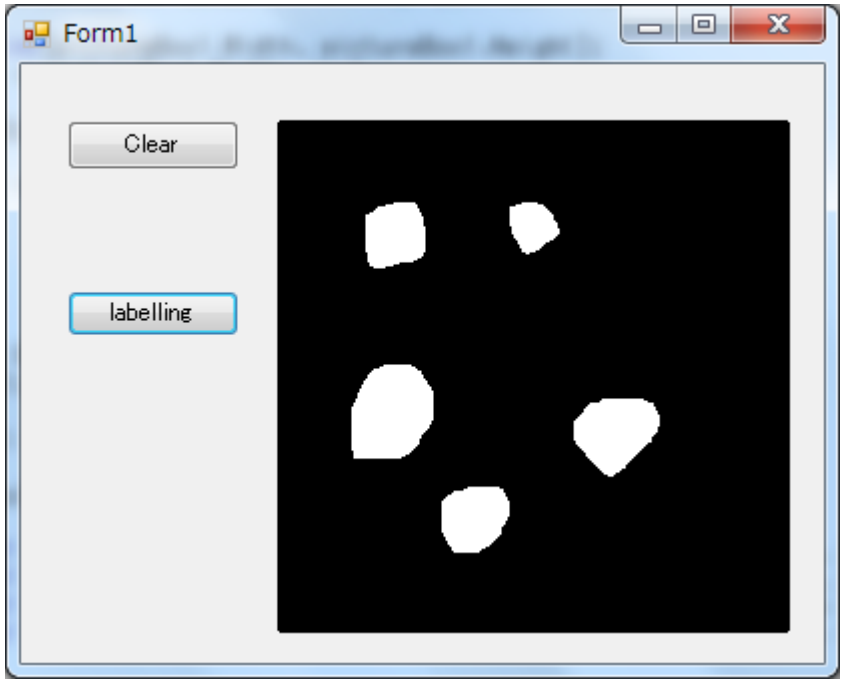
Labelling



改善。
もっと良い方法があるだろう。

Labelling

何個でもできる。



塊の数を数えたい。

Labelling

```
        D.Disp(label, pictureBox1);

        int[,] labels = new int[pictureBox1.Width, pictureBox1.Height];
        int n = 0;
        for (int j = 1; j < pictureBox1.Width - 1; j++)
        {
            for (int i = 1; i < pictureBox1.Height - 1; i++)
            {
                if (label[i, j] == 1 && labels[i, j] == 0)
                {
                    n++;
                    labels[i, j] = n;
                    while (true)
                    {
                        int counter = 0;
                        for (int jj = 1; jj < pictureBox1.Width - 1; jj++)
                        {
                            for (int ii = 1; ii < pictureBox1.Height - 1; ii++)
                            {
                                if (labels[ii, jj] == n)
                                {
                                    if (labels[ii + 1, jj] == 0 && label[ii + 1, jj] == 1) { labels[ii + 1, jj] = n; counter++; }
                                    if (labels[ii - 1, jj] == 0 && label[ii - 1, jj] == 1) { labels[ii - 1, jj] = n; counter++; }
                                    if (labels[ii, jj + 1] == 0 && label[ii, jj + 1] == 1) { labels[ii, jj + 1] = n; counter++; }
                                    if (labels[ii, jj - 1] == 0 && label[ii, jj - 1] == 1) { labels[ii, jj - 1] = n; counter++; }
                                }
                            }
                        }
                        if (counter == 0) break;
                    }
                }
            }
        }
        label1.Text = "number = " + n.ToString();
    }
```

← {0} を使った方が格好良い。

Label==1にぶつかったら、

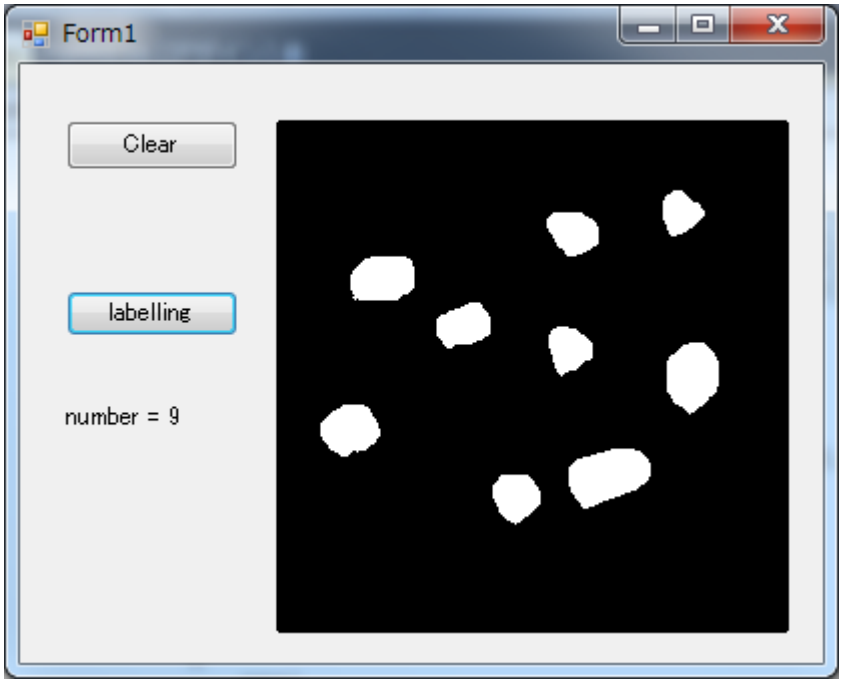
連続領域をcheckする

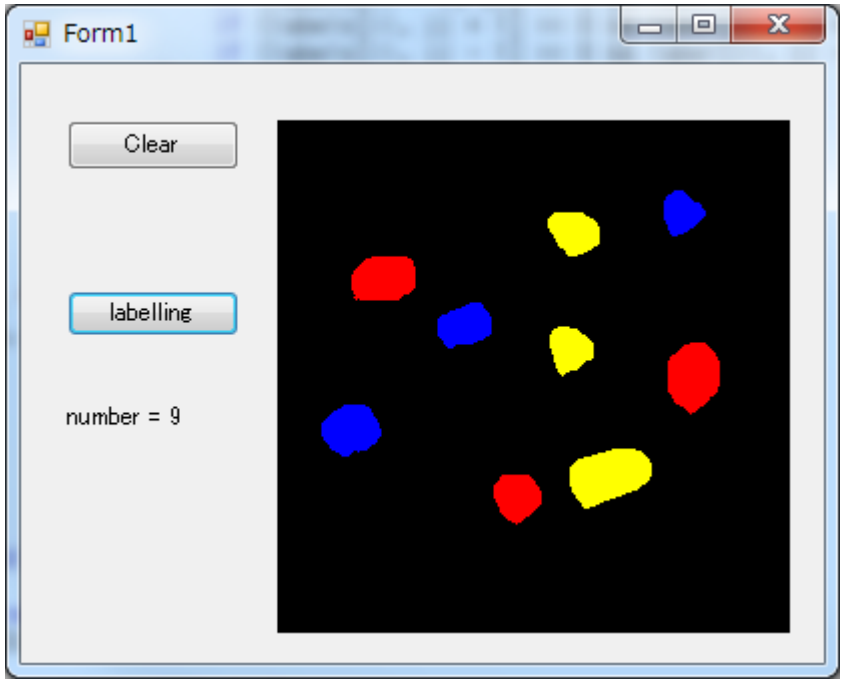
という流れ。

Dispの下に追加。別ボタンとしても問題ない。

その場合は配列宣言を頭（imageと一緒の所）で行う。

Labelling





labelsには、1, 2, 3, …. が入っているので、色分けなどが可能

おわり