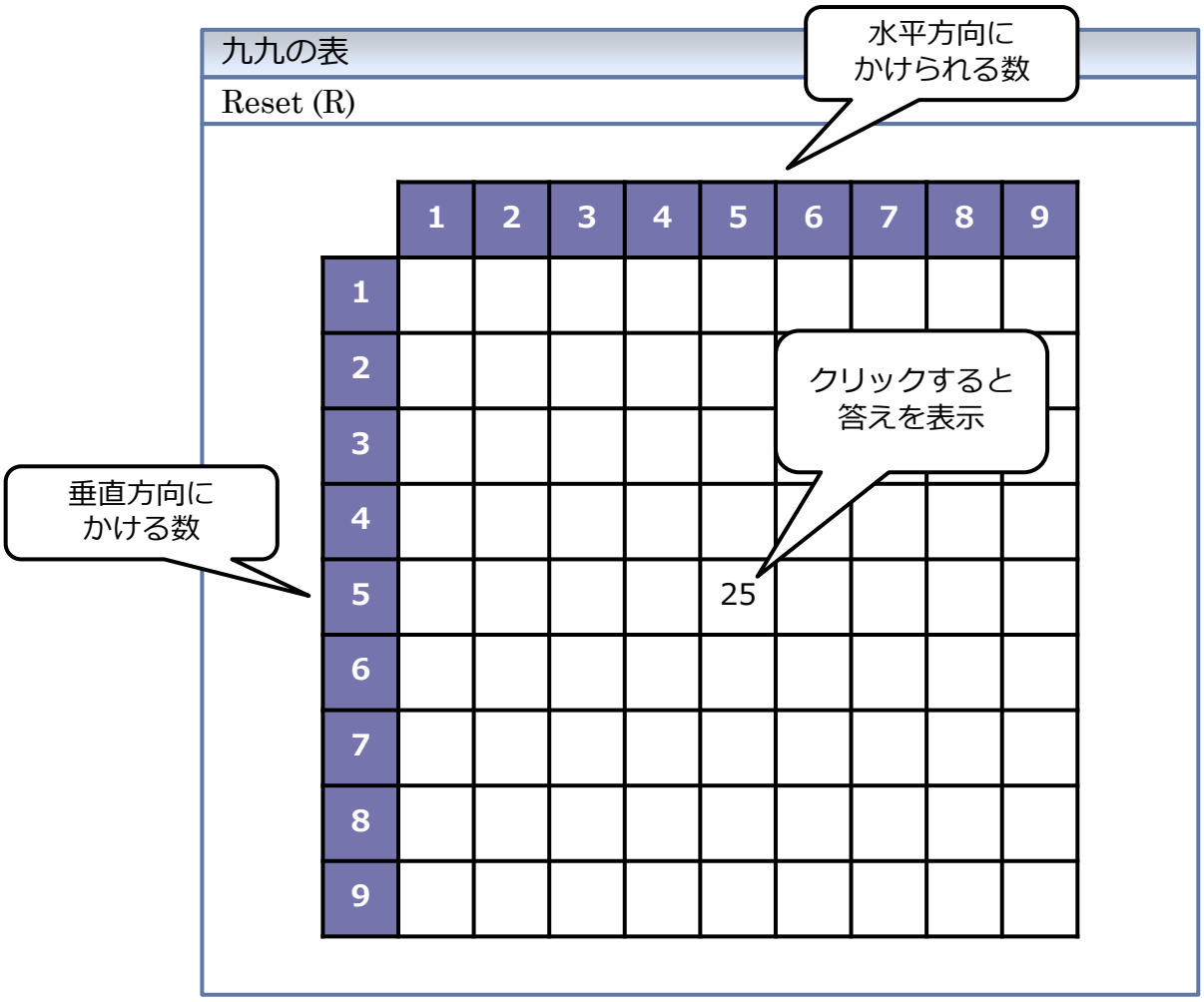
【06】 コントロール配列を使う
九九の表

1 今回作成するアプリケーションの概要

九九の表を表示する
ただ表示しただけでは面白くないので、起動時には答えを表示しないで、クリックしたら表示するようにする

◆ 行われる動作
[1] 起動すると、九九の表のうち、かける数とかけられる数の部分をのみ表示
答えの部分は表示しない
[2] 答えの部分をクリックすると、答えを表示
[3] メニュー「Reset」をクリックすると答えの部分を空白に戻す



◆ 使用者とコンピュータの関係をまとめる

[使用者 ← コンピュータ] 九九の表を表示
[使用者 → コンピュータ] 表の表示されていないところをクリック
[使用者 ← コンピュータ] 答えを表示

[使用者 → コンピュータ] 「Reset」をクリック
[使用者 ← コンピュータ] 答えをすべて非表示に

◆ 必要なコントロールは次の通り
- ◆ Label コントロール（かけられる数、かける数、答えを表示）
  かけられる数 と かける数 を表示するのに一次元のコントロール配列
  答えを表示するのに二次元のコントロール配列
- ◆ MenuStrip コントロール（メニューを表示、実行、今回は「Reset」のみ）

コントロール配列 とは

- ◆ Label、TextBox、PictureBox などのコントロールを配列にしたもの
- ◆ ツールボックスから貼りつけてフォーム上に設置することができない
  → プログラム中で宣言、定義を行う。

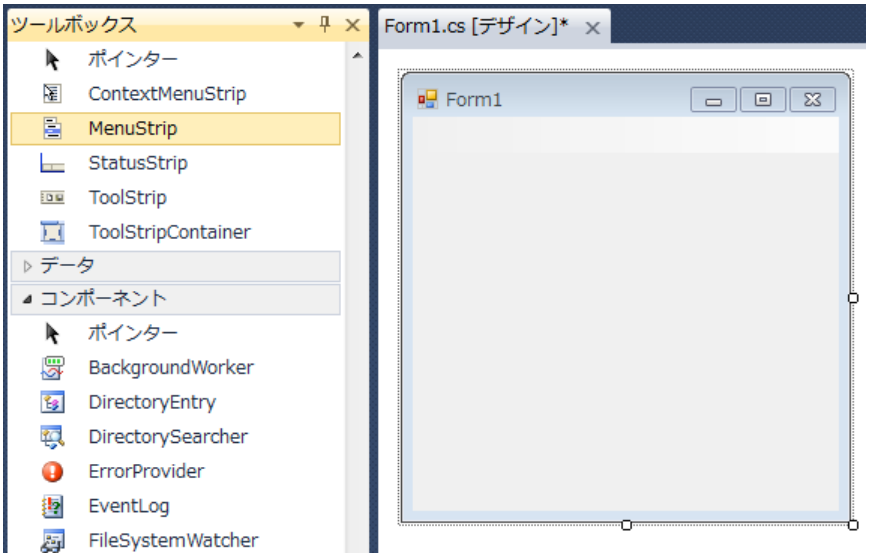
2 Visual Studio 2010 の起動と新規プロジェクトの作成

■ 1 回目でやったとおり、Visual Studio 2010 を起動
[1]「スタート」→ [2] 「すべてのプログラム」→ [3] 「Visual Studio 2010」のフォルダ
→ [4] 「Visual Studio 2010」のアイコン

■ 新規プロジェクトも 1 回目の手順で作成
[1] メニュー →「ファイル」→ [2] 「新規作成」→ [3] 「プロジェクト」
[4]「Visual C#」→ [5] 「Windowsフォームアプリケーション」
[6]「プロジェクト名」を入力 **（今回は JKJ06）**
[7]「参照…」をクリックして、プロジェクトを保存する場所を選択
[8]「ソリューションのディレクトリを作成」のチェックは はずす
[9]「OK」をクリック

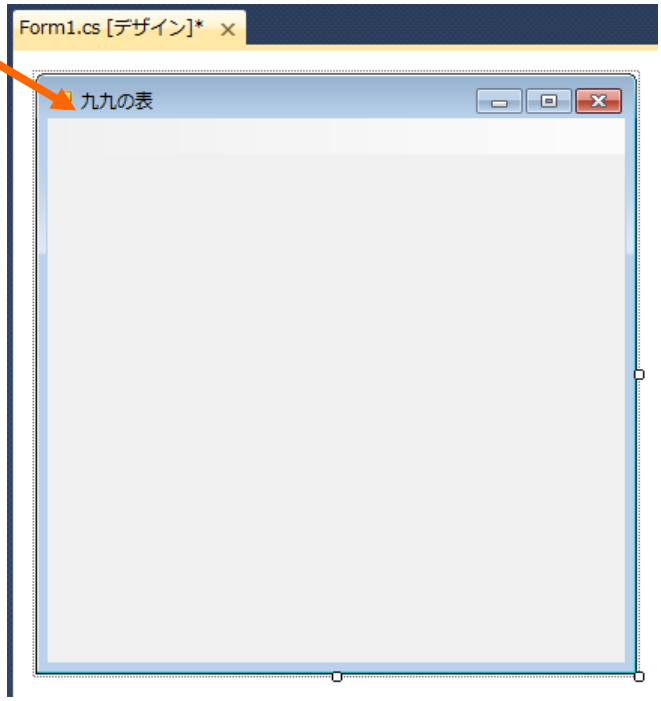
3 コントロールの配置

- **MenuStrip コントロール** をダブルクリックして追加する。
  （MenuStrip コントロールはフォーム上には配置されない）



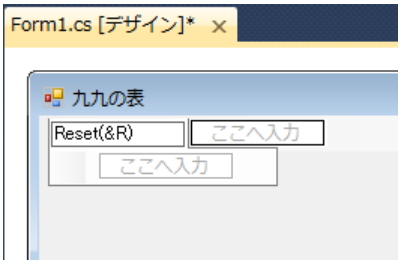
- フォームのコントロールのプロパティを次のように設定する

Name	Form1
Size	400, 400
Text	九九の表



4 メニューの作成

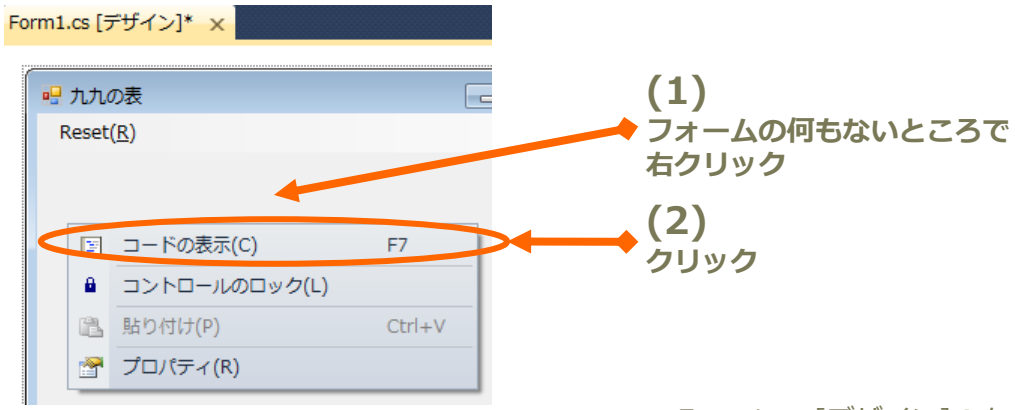
- これまでと同様にして、メニューに「Reset(&R)」を追加する



5 コードを表示して、プログラムを入力できるようにする

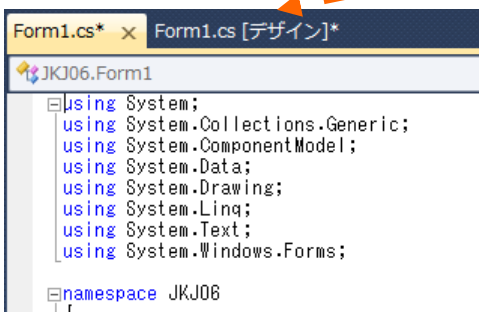
- コード（プログラム）を表示して、入力できるようにする

(1) フォームの何もないところで 右クリック
(2) 表示されたメニューの「コードを表示」をクリック



- コード（プログラム）が表示

Form1.cs [デザイン]のタグを
クリックすると、フォームに
コントロールが貼れるデザインへ戻る



6 コントロール配列をプログラミングするときの手順

(1) コントロール配列のフィールドを宣言する
どのメンバ関数でも利用可能なように、メンバ変数を宣言するのと同じ場所で宣言する

(2) コントロール配列の作成（数も決定する）

(3) コントロールのインスタンス（実体）を作成
配列の要素の数だけ作成する

(4) コントロールのプロパティを設定

(5) イベントを発生したときに呼び出す関数を関連付ける

(6) フォームに追加

(5) イベントを発生したときに呼び出す関数を定義

7 かけられる数（水平方向）のラベルの生成と表示

- 次の通りにプログラムを入力

```
namespace JKJ06
{
    public partial class Form1 : Form
    {
        Label[] lblHorizontal;

        public Form1()
        {
            InitializeComponent();

            // ラベルの配列の生成
            lblHorizontal = new Label[9];
            for (int i = 0; i < lblHorizontal.Length; i++)
            {
                // ラベルのインスタンスを生成
                lblHorizontal[i] = new Label();
                // プロパティを設定
                lblHorizontal[i].Left = 30 + 30 * i;
                lblHorizontal[i].Top = 30;
                lblHorizontal[i].Width = 30;
                lblHorizontal[i].Height = 30;
                lblHorizontal[i].BorderStyle = BorderStyle.FixedSingle;
                lblHorizontal[i].BackColor = Color.LightGray;
                lblHorizontal[i].TextAlign = ContentAlignment.MiddleCenter;
                lblHorizontal[i].Text = (i+1).ToString();
                // フォームに追加する
                Controls.Add(lblHorizontal[i]);
            }
        }
    }
}
```

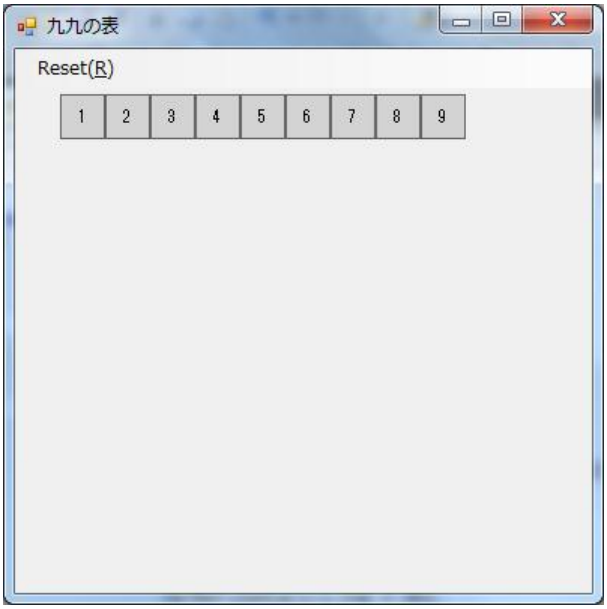
この１行を入力

lblHorizontal 配列の大きさ

配列の index より 1 大きく

この部分を入力

- 実行してみると、水平方向にラベルが 9 個表示される



8 かける数（垂直方向）のラベルの生成と表示

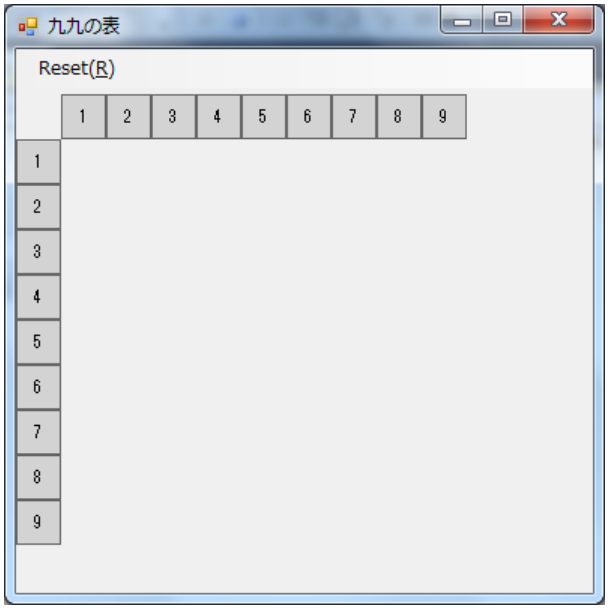
- 次の通りにプログラムを入力

```
namespace JKJ06
{
    public partial class Form1 : Form
    {
        Label[] lblHorizontal;

        Label[] lblVertical;

        public Form1()
        {
            InitializeComponent();

            // ラベルの配列の生成
            lblHorizontal = new Label[9];
            for (int i = 0; i < lblHorizontal.Length; i++)
            {

                    （略）

                // フォームに追加する
                Controls.Add(lblHorizontal[i]);
            }

            // ラベルの配列の生成
            lblVertical = new Label[9];
            for (int j = 0; j < lblVertical.Length; j ++)
            {
                // ラベルのインスタンスを生成
                lblVertical[j] = new Label();
                // プロパティを設定
                lblVertical[j].Left = 0;
                lblVertical[j].Top = 30 + 30 + 30 * j;
                lblVertical[j].Width = 30;
                lblVertical[j].Height = 30;
                lblVertical[j].BorderStyle = BorderStyle.FixedSingle;
                lblVertical[j].BackColor = Color.LightGray;
                lblVertical[j].TextAlign = ContentAlignment.MiddleCenter;
                lblVertical[j].Text = (j + 1).ToString();
                // フォームに追加する
                Controls.Add(lblVertical[j]);
            }

        }
    }
}
```

この１行を入力

この部分を入力

- 実行してみると、垂直方向にもラベルが 9 個表示される



9 答えを表示するラベルの生成と表示

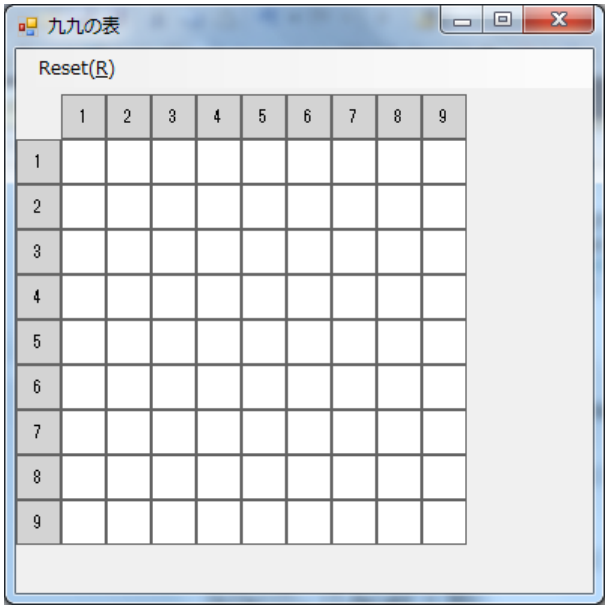
- 次の通りにプログラムを入力

```
namespace JKJ06
{
    public partial class Form1 : Form
    {
        Label[] lblHorizontal;
        Label[] lblVertical;

        Label[,] lblCell;

        public Form1()
        {
            InitializeComponent();

            // ラベルの配列の生成
            lblHorizontal = new Label[9];
                    (略)
                // フォームに追加する
                Controls.Add(lblHorizontal[i]);
            }

            // ラベルの配列の生成
            lblVertical = new Label[9];
                    (略)
                // フォームに追加する
                Controls.Add(lblVertical[j]);
            }

            // ラベルの配列の生成（2次元）j
            lblCell = new Label[lblHorizontal.Length, lblVertical.Length];
            for (int j = 0; j < lblVertical.Length; j++)
            {
                for (int i = 0; i < lblHorizontal.Length; i++)
                {
                    lblCell[i, j] = new Label();
                    lblCell[i, j].Left = 30 + 30 * i;
                    lblCell[i, j].Top = 30 + 30 + 30 * j;
                    lblCell[i, j].Width = 30;
                    lblCell[i, j].Height = 30;
                    lblCell[i, j].BorderStyle = BorderStyle.FixedSingle;
                    lblCell[i, j].BackColor = Color.White;
                    lblCell[i, j].TextAlign = ContentAlignment.MiddleCenter;

                    Controls.Add(lblCell[i, j]);
                }
            }

        }
    }
}
```

この１行を入力

この部分を入力

- 実行してみると、縦 9 個、横 9 個の格子が表示される



ラベルの生成と表示を行っただけなので、クリックしてもなにも起こらない。

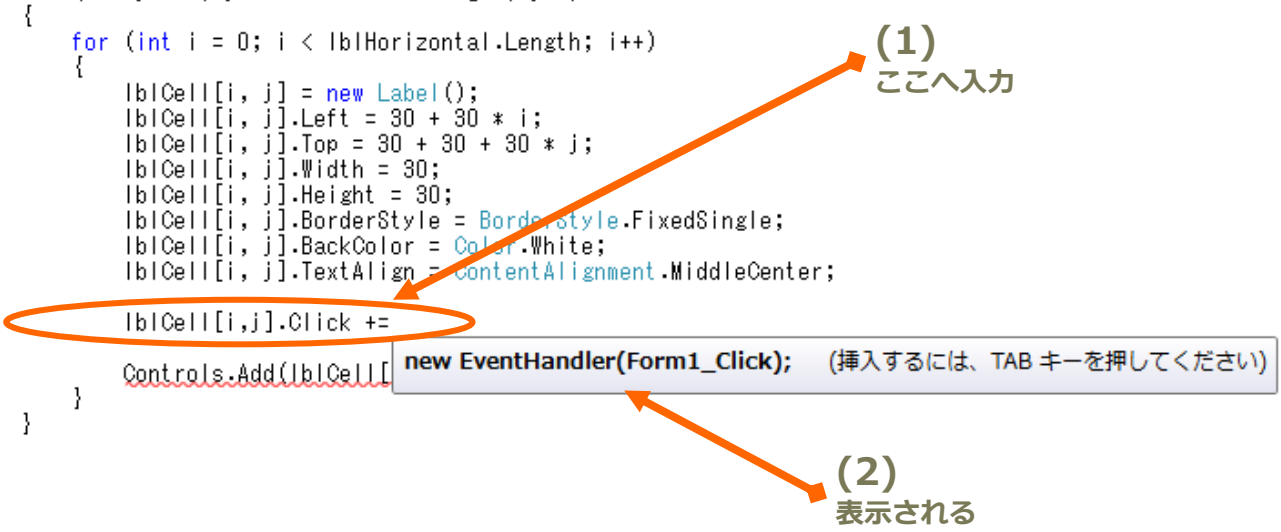
１０ 答えを表示するラベルへのイベントの追加

- コントロールのイベント（今回はクリック）を追加する。

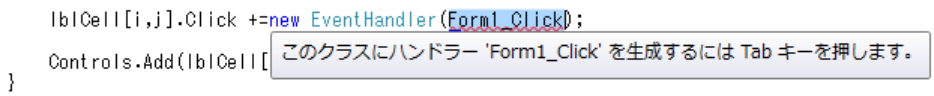
(1) まず、以下の場所に次の行を入力する。

lblCell[I,j] .Click +=

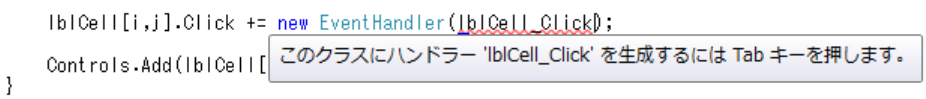
(2) ここまで入力すると、（挿入するには、TAB キーを押してください）の
メッセージが表示される。



(3) TAB キー を押すと、「このクラスに ～ 押します」 のメッセージが表示される。



(4) Form1_Click の部分を lblCell_Click に変更する。
（TAB キーを押さないで、そのまま lblCell_Click と入力すればいい）



(5)「このクラスにハンドラー ‘lblCell_Click’ を生成するには Tab キーを押します。」
というメッセージが表示される

(6) Tab キーを押すと、 lblCell_Click 関数が自動的に生成される。

(7) 自動的に生成された lblCell_Click 関数は次のようになっている。

```
                lblCell[i, j].BackColor = Color.White;
                lblCell[i, j].TextAlign = ContentAlignment.MiddleCenter;

                lblCell[i, j].Click += new EventHandler(lblCell_Click);

                Controls.Add(lblCell[i, j]);
            }
        }

    }

    void lblCell_Click(object sender, EventArgs e)
    {
        throw new NotImplementedException();
    }
}
```

自動生成された lblCell_Click 関数

(8) 自動的に生成された lblCell_Click 関数を次のように記述する

自動的に入力された部分

```
void lblCell_Click(object sender, EventArgs e)
{
    // throw new NotImplementedException();
    for (int j = 0; j < lblVertical.Length; j++)
    {
        for (int i = 0; i < lblHorizontal.Length; i++)
        {
            if (sender.Equals(lblCell[i, j]))
            {

                int sj = int.Parse(lblVertical[j].Text);
                int si = int.Parse(lblHorizontal[i].Text);
                lblCell[i, j].Text = (si * sj).ToString();

            }
        }
    }
}
```

クリックされたコントロールの情報が入っている

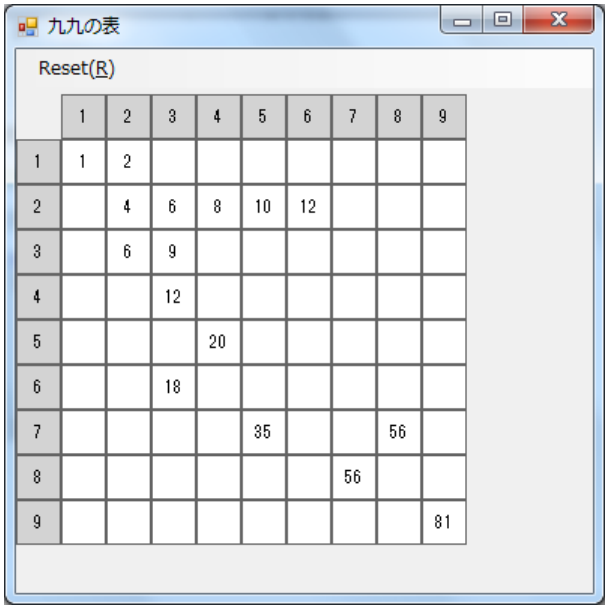
この行を消去するかコメントにすること

クリックされたコントロールとどの lblCell [i , j] が一致するか調べている

この部分を入力

自動的に入力された部分

- 実行してみると、縦 9 個、横 9 個の格子が表示される。
  また、クリックしたところの答えが表示される。



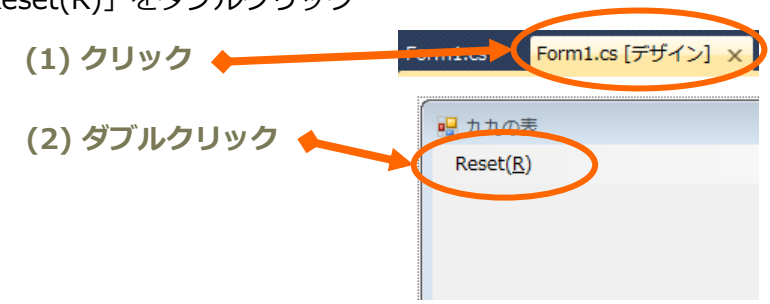
	1	2	3	4	5	6	7	8	9
1	1	2							
2		4	6	8	10	12			
3		6	9						
4			12						
5				20					
6			18						
7					35			56	
8							56		
9									81

11 「Reset(R)」を選択したときのプログラムを入力

- メニューの「Reset(E)」をクリックされたときの処理
  = 答えを表示した Label コントロールの2次元配列 lblCell

(1) Form1 [デザイン] をクリックして、Form1 のデザインをするタグに戻る

(2) 「Reset(R)」をダブルクリック



(3) lblCell の表示をリセットするプログラムを入力する

自動的に入力された部分

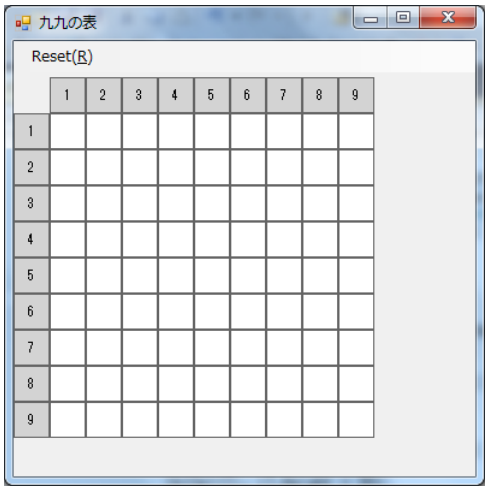
```
private void resetRToolStripMenuItem_Click(object sender, EventArgs e)
    {
        for (int j = 0; j < lblVertical.Length; j++)
        {
            for (int i = 0; i < lblHorizontal.Length; i++)
            {
                lblCell[i, j].Text  = String.Empty;
            }
        }
    }
```

この部分を入力

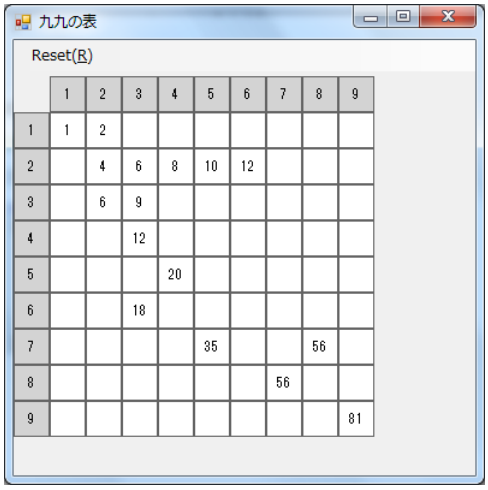
自動的に入力された部分

１２ とりあえず完成

- プログラムを実行すると、かけられる数が水平にとかける数が水平にある表が表示される



- 表の空いている部分をクリックすると、掛け算した答えが表示される



- Reset(R) をクリックすると、すべての答えが消去される

１３ まとめ

■ コントロール配列
- ◆ コントロールの配列を利用することができる
- ◆ ただし、ツールボックスからフォームに貼りつける方法では利用できない
- ◆ プログラム中でコントロールの配列を生成することで利用できる
- ◆ その手順は
  - (1) コントロール配列のフィールドを宣言する
  - (2) コントロール配列の作成（数も決定する）
  - (3) コントロールのインスタンス（実体）を作成
  - (4) コントロールのプロパティを設定
  - (5) イベントを発生したときに呼び出す関数を関連付ける
  - (6) フォームに追加
  - (5) イベントを発生したときに呼び出す関数を定義
- ◆ イベントを発生したときに呼び出す関数 を複数のコントロールで共有できる
  - ◆ 呼び出し元のコントロールは引数の object sender で参照することができる
  - ◆ どのコントロールから呼び出したか知りたいときは、sender.Equal( ) 関数の引数にそのコントロールを与え、true であるかどうかを確認する

■ 今回使ったコントロール
- ◆ Label コントロール（文字を表示する）
- ◆ MenuStrip コントロール（メニューを管理する）

１４ 追加課題

1 かけられる数（水平方向）とかける数（垂直方向）をランダムに並び替えた表を作成できるようにする。「ランダム」ボタンやメニューの項目を作っても構わない。

2 かけられる数（水平方向）とかける数（垂直方向）の数を 9 から変更できるようにする。
変更する値入力する方法は TextBox コントロールを利用してもいいし、別のフォームを開いても構わない。

3 答えを表示する Label コントロールを TextBox コントロールに変更し、百ます計算を実施できるプログラムへ変更する。