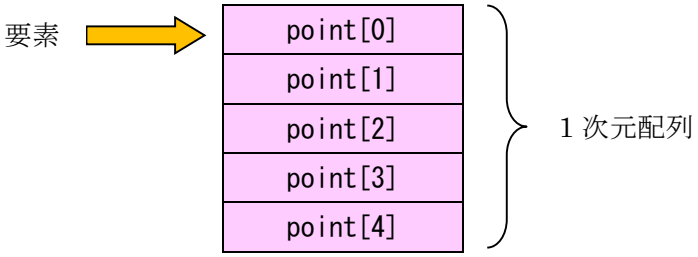
2023 年 6 月 8 日（木）実施

配列

同種のデータ型を有する複数のデータ（要素）を番号付けして，ひとまとまりの対象として扱うものを**配列**と呼ぶ。

1 次元配列

要素が 1 つの添え字（インデックス）で番号付けされるものを **1 次元配列**と呼ぶ。



1 次元配列の取り扱いに関して，次のような特徴がある。

1. プログラム中で用いる**配列変数**（配列の本体を参照する参照型の変数）は必ず**宣言**しておく。

   例）`int[] point;  /* int 型の配列変数 point を宣言 */`

2. 配列の本体を生成し，配列変数に代入して参照させる。

   例）`point = new int[5];  /* int 型の 5 個の要素を持つ配列を生成` ⇒ 要素数は配列が生成された時点で定まる `*/`

3. 配列変数を宣言する際に，配列の本体を生成することが出来る。

   例）`int[] point = new int[5];`

4. 配列の生成時に、各要素には初期値（値型の場合は 0，bool 型の場合は false，参照型の場合は null）が設定される。

5. 配列変数を宣言する際に，中括弧{ }の間に初期値を与えて**初期化**することが出来る。

   例）`int[] point = new int[] { 98, 76, 85, 67, 59 };` ⇒ この場合には，要素数は初期値の個数で決定される。右辺の `new int[]`を省略した書き方も許容される。

6. 配列の要素は **0 番から[要素数]－1 番まで**の添え字を用いて表される。

多次元配列

要素が 2 つ以上の添え字で番号付けされるものを**多次元配列**と呼ぶ。次の文は 1 つ目の添え字が 0，1，2，3 の範囲，2 つ目の添え字が 0，1 の範囲となる 2 次元配列の宣言の例である。

```
int[,] a = new int[4, 2];
```

また，次の文は，2 次元配列の初期化の例である。

```
int[,] a = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
```

同様に，3 次元配列の宣言及び初期化の例は次の様になる。

```
int[, ,] b = new int[2, 2, 3];
int[, ,] b = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } },
                           { { 7, 8, 9 }, { 10, 11, 12 } } }; /* 長いので折り返し */
```

本日の課題

今回の授業では，レジスタアプリの作成を通じて，様々な 1 次元配列を取り扱う方法を学ぶ。

手順

1) プロジェクトの作成

第 1 回の教材の pp.3-4 と同様にして，新規のプロジェクトを作成する。

今回は，『プロジェクト名』を「JimboRegister」（Jimbo の箇所は自分の名前に置き換える）とする。また，今後，暫くは今回作成したプロジェクトに機能を追加していく。

2) コントロールの配置・プロパティの設定

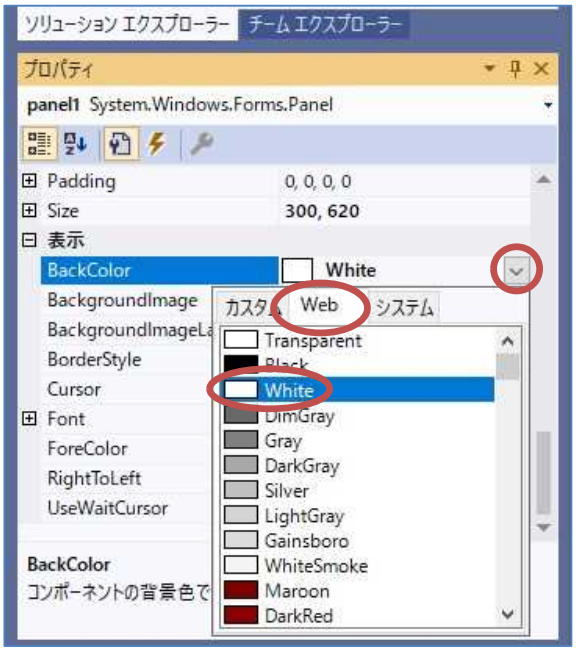
先ず，フォームをクリックすることでフォーム（Form1）を選択し，『プロパティウィンドウ』で，次のプロパティを設定する。

Size：(1216, 660)　← (X, Y)　　＊括弧は入力しない。

Font：12pt

Text："JimboRegister"　← 文字列リテラルの中身を入力　　＊二重引用符は入力しない。

次に，[ツールボックス]→[コンテナ]から Panel を選択してフォームに配置する。パネル（panel1）のプロパティを次の様に設定する。



BackColor：White

Location：(0, 0)

Size：(300, 620)

ここで，BackColor はオプションメニューで『Web』のタブを開いて選択する。

同様にして，もう一枚のパネル（panel2）をフォームに配置し，プロパティを次の様に設定する。

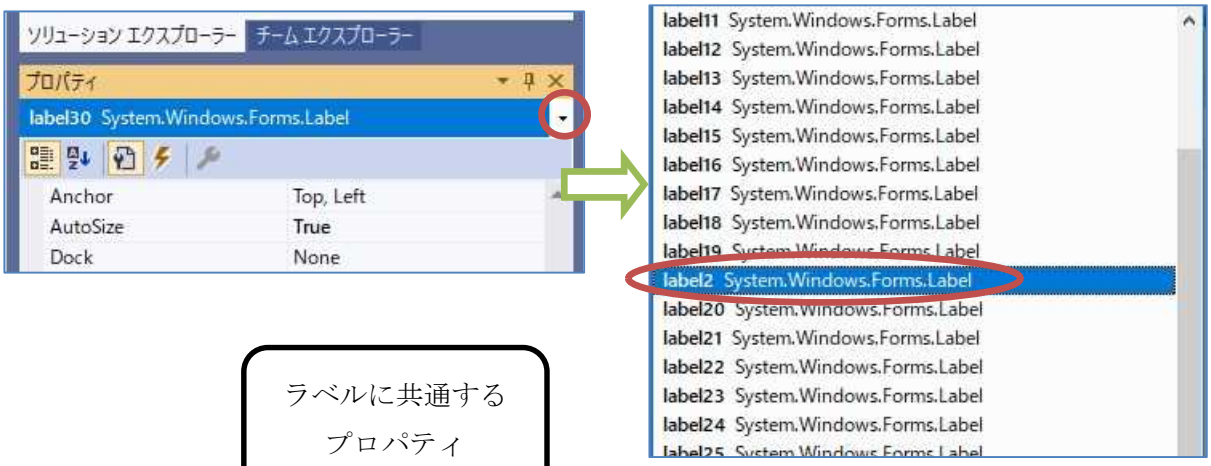
BackColor：MistyRose

Location：(300, 0)

Size：(900, 620)

続けて，panel1 上にラベル（label1～label30）及び商品個数のアップ・ダウン用のボタン（button1～button20）を配置し，プロパティを次の様に設定する。この時，ラベル及びボタンを 1 個だけ配置し，それぞれをコピーして必要な個数分貼ってからプロパティを設定する。

それぞれのコントロールのプロパティを設定するには，プロパティウィンドウの 1 番上の窓でコントロールを選択することが可能である。



ラベルに共通する
プロパティ

【ラベルのプロパティ】　Font：12pt

label1		label2		label3	
Location	(10, 10)	Location	(45, 35)	Location	(110, 35)
label4		label5		label6	
Location	(10, 65)	Location	(45, 90)	Location	(110, 90)
label7		label8		label9	
Location	(10, 120)	Location	(45, 145)	Location	(110, 145)
label10		label11		label12	
Location	(10, 175)	Location	(45, 200)	Location	(110, 200)
label13		label14		label15	
Location	(10, 230)	Location	(45, 255)	Location	(110, 255)
label16		label17		label18	
Location	(10, 285)	Location	(45, 310)	Location	(110, 310)
label19		label20		label21	
Location	(10, 340)	Location	(45, 365)	Location	(110, 365)
label22		label23		label24	
Location	(10, 395)	Location	(45, 420)	Location	(110, 420)
label25		label26		label27	
Location	(10, 450)	Location	(45, 475)	Location	(110, 475)
label28		label29		label30	
Location	(10, 505)	Location	(45, 530)	Location	(110, 530)

ボタンは先ずアップボタンを 1 個配置してプロパティを設定した上でコピーし，9 個分貼り付けて Location プロパティを設定する。続いてダウンボタンを 1 個配置してプロパティを設定した上でコピーし，9 個分貼り付けて Location プロパティを設定する。

【アップボタンのプロパティ】　Font：9pt　Size (Width：18, Height：18)　Text：″△″

button1		button2		button3		button4	
Location	(2, 35)	Location	(2, 90)	Location	(2, 145)	Location	(2, 200)
button5		button6		button7		button8	
Location	(2, 255)	Location	(2, 310)	Location	(2, 365)	Location	(2, 420)
button9		button10					
Location	(2, 475)	Location	(2, 530)				

【ダウンボタンのプロパティ】　Font：9pt　Size (Width：18, Height：18)　Text：″▼″

button11		button12		button13		button14	
Location	(22, 35)	Location	(22, 90)	Location	(22, 145)	Location	(22, 200)
button15		button16		button17		button18	
Location	(22, 255)	Location	(22, 310)	Location	(22, 365)	Location	(22, 420)
button19		button20					
Location	(22, 475)	Location	(22, 530)				

更に，panel1 上に合計表示用のラベル（label31）を配置し，プロパティを次の様に設定する。
Font：18pt
Location：(110，570)

panel2 上に 4 個の商品カテゴリラベル（label32～label35）を配置し，プロパティを次の様に設定する。

【商品カテゴリラベルのプロパティ】　AutoSize：false　Size：(100，70)
TextAlign：MiddleCenter　Font：20pt

label32		label33		label34		label35	
BackColor	Red	BackColor	Silver	BackColor	LawnGreen	BackColor	Orange
Location	(20, 20)	Location	(20, 120)	Location	(20, 220)	Location	(20, 320)
Text	″肉類″	Text	″魚貝類″	Text	″野菜″	Text	″飲料″

続けて，panel2 上に商品ボタン（button21～button32）を配置し，プロパティを次の様に設定する。

【商品ボタンのプロパティ】BackColor：Snow　Size：(100, 70)

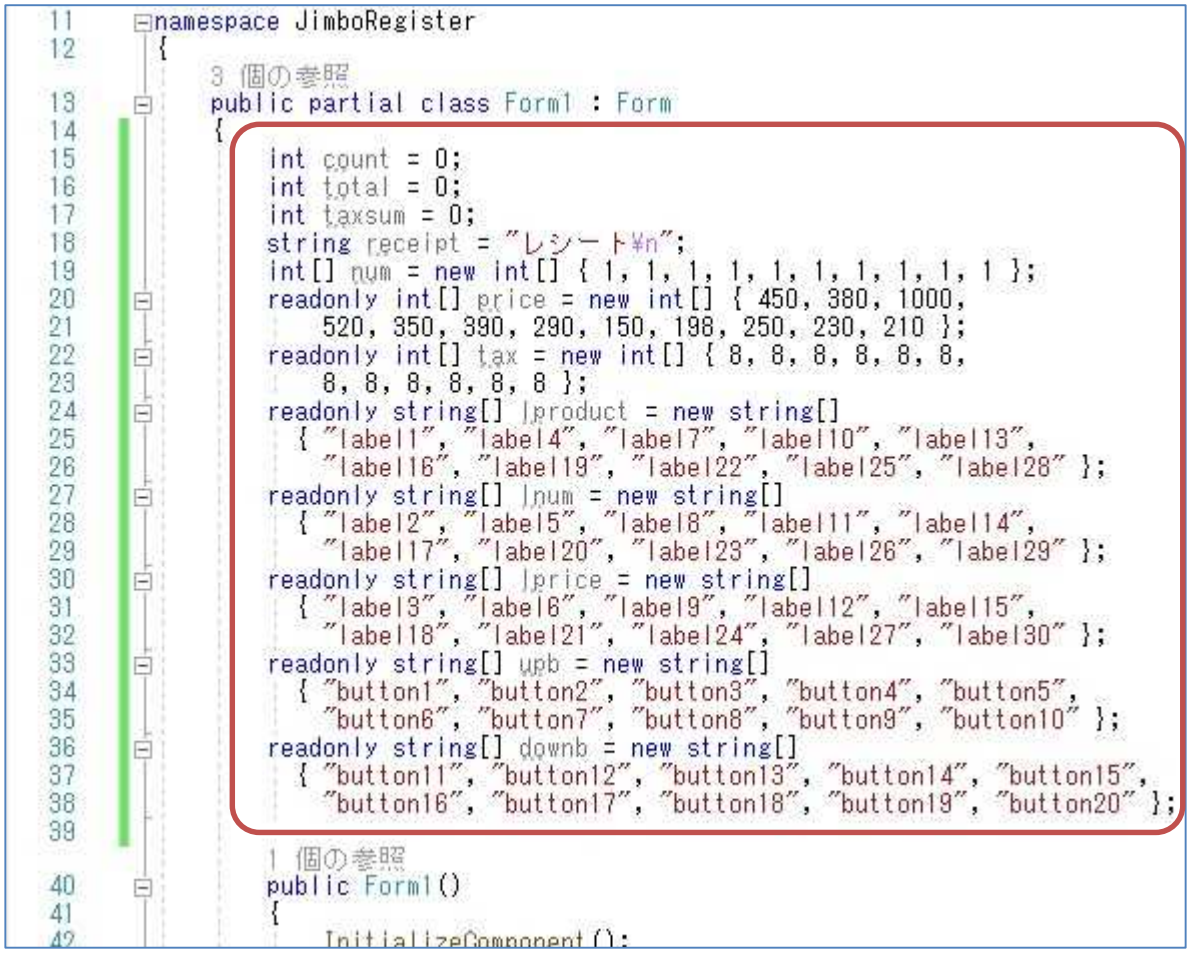
button21		button22		button23	
Location	(150, 20)	Location	(280, 20)	Location	(410, 20)
Text	″豚ロース肉 200g″	Text	″鶏モモ肉 300g″	Text	″牛フィレ肉 150g″
button24		button25		button26	
Location	(150, 120)	Location	(280, 120)	Location	(410, 120)
Text	″さば 2 尾″	Text	″さけ 2 切れ″	Text	″あじ 2 尾″
button27		button28		button29	
Location	(150, 220)	Location	(280, 220)	Location	(410, 220)
Text	″トマト 1 袋″	Text	″レタス 1 個″	Text	″ピーマン 1 袋″
button30		button31		button32	
Location	(150, 320)	Location	(280, 320)	Location	(410, 320)
Text	″オレンジジュース 1L パック″	Text	″牛乳 1L パック″	Text	″飲むヨーグルト 1L パック″

更に，panel2 上に会計処理用のボタン (button33) を配置し，プロパティを次の様に設定する。

BackColor：Khaki
Font：24pt
Location：(25, 530)
Size：(200, 70)
Text：″会計″

3) コーディング

先ず，『表示』→『コード』と選択してコードエディタを開き，クラスのフィールドで変数及び配列を宣言する。ここで，readonly 修飾子はプログラムの中で値が書き換えられないことを表す。



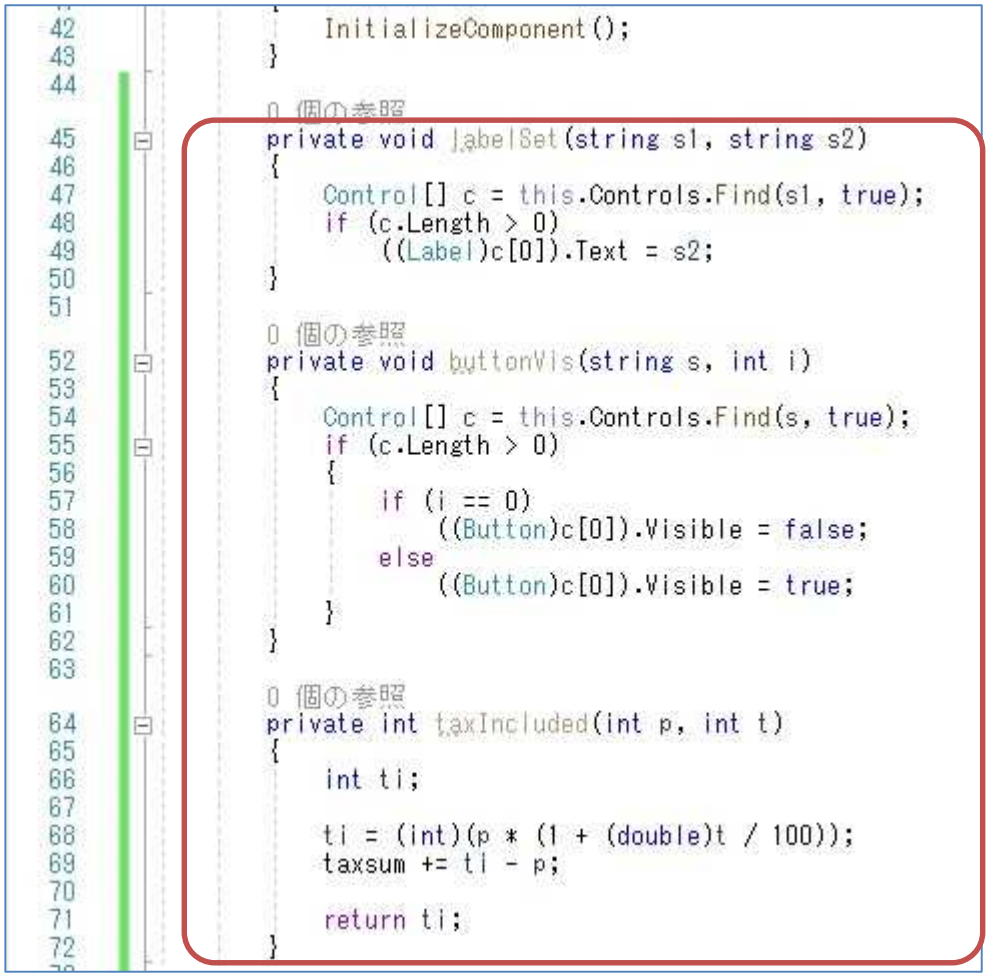
```
namespace JimboRegister
{
    3 個の参照
    public partial class Form1 : Form
    {
        int count = 0;
        int total = 0;
        int taxsum = 0;
        string receipt = "レシート¥n";
        int[] num = new int[] { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
        readonly int[] price = new int[] { 450, 380, 1000,
            520, 350, 390, 290, 150, 198, 250, 230, 210 };
        readonly int[] tax = new int[] { 8, 8, 8, 8, 8, 8,
            8, 8, 8, 8, 8, 8 };
        readonly string[] lproduct = new string[]
          { "label1", "label4", "label7", "label10", "label13",
            "label16", "label19", "label22", "label25", "label28" };
        readonly string[] lnum = new string[]
          { "label2", "label5", "label8", "label11", "label14",
            "label17", "label20", "label23", "label26", "label29" };
        readonly string[] lprice = new string[]
          { "label3", "label6", "label9", "label12", "label15",
            "label18", "label21", "label24", "label27", "label30" };
        readonly string[] upb = new string[]
          { "button1", "button2", "button3", "button4", "button5",
            "button6", "button7", "button8", "button9", "button10" };
        readonly string[] downb = new string[]
          { "button11", "button12", "button13", "button14", "button15",
            "button16", "button17", "button18", "button19", "button20" };

        1 個の参照
        public Form1()
        {
            InitializeComponent();
```

それぞれの変数の意味と名称との対応は，カウンタcount，合計金額 total，税額合計 taxsum，レシート記載内容 receipt である。また，配列の意味と名称との対応は，個数 num，税抜き価格 price，税率(%) tax，商品名表示用ラベル名 lproduct，個数表示用ラベル名 lnum，税抜き価格表示用ラベル名 lprice，アップボタン名 upb，ダウンボタン名 downbである。配列の宣言は，それぞれ1行で書いても良いが，ここでは上の図が教材に収まる様に途中で改行している。

次に， labelSetメソッド， buttonVisメソッド及びtaxIncludedメソッドの3個のメソッドを一から作成する。ここで，labelSetメソッドは，第1引数でラベル名を受け取り，第2引数でそのラベルに設定する文字列を受け取る。C#で用意されているControls.Findメソッドはフォームに配置されているコントロールから第1引数で指定された文字列にある名前のコントロールを検索し，見つかればそれを戻すものである。buttonVisメソッドは，第1引数でボタン名を受け取り，第2引数でその可視属性（Visibleプロパティ）のオンオフのスウィッチ（1 はオン，0 はオフ）を受け取る。taxIncludedメソッドは，第1引数で税抜き価格を受け取り，第2引数で税率(%)を受け取り，税込み価格を戻す。『○○個の参照』は後にメソッドが呼び出された箇所の数に変わる。

（図は次のページ）



```
        InitializeComponent();
    }

    0 個の参照
    private void labelSet(string s1, string s2)
    {
        Control[] c = this.Controls.Find(s1, true);
        if (c.Length > 0)
            ((Label)c[0]).Text = s2;
    }

    0 個の参照
    private void buttonVis(string s, int i)
    {
        Control[] c = this.Controls.Find(s, true);
        if (c.Length > 0)
        {
            if (i == 0)
                ((Button)c[0]).Visible = false;
            else
                ((Button)c[0]).Visible = true;
        }
    }

    0 個の参照
    private int taxIncluded(int p, int t)
    {
        int ti;

        ti = (int)(p * (1 + (double)t / 100));
        taxsum += ti - p;

        return ti;
    }
```

なお，ここで(Label)，(Button)，(int)及び(double)の様に半角文字の括弧( )でのデータ型の名前を挟んだものを**型キャスト**と呼び，その後ろの変数のデータ型をその場だけ括弧内に指定したデータ型に変換する機能を有する。特に，int 型の変数同士の演算では結果は整数しか得られないが，型キャスト(double)を利用して，int 型の変数を参照して得られるデータを一時的に double 型のデータに変換することで，実数化したデータによる演算が可能となる。taxIncluded メソッドでは税率(%)を小数に変換して（例えば 8% → 0.08）税込み価格を実数で求めた上で，(int)によって整数化して，小数点以下を切り捨てている。

フォームデザイナー上でフォームの上の枠をダブルクリックして，イベントハンドラ Form1_Load の処理内容（赤枠の部分）を記述する。

（図は次のページ）

```
        1 個の参照
74      private void Form1_Load(object sender, EventArgs e)
75      {
76          labelSet(lproduct[0], ""); labelSet(lproduct[1], ""); labelSet(lproduct[2], "");
77          labelSet(lproduct[3], ""); labelSet(lproduct[4], ""); labelSet(lproduct[5], "");
78          labelSet(lproduct[6], ""); labelSet(lproduct[7], ""); labelSet(lproduct[8], "");
79          labelSet(lproduct[9], "");
80          labelSet(lnum[0], ""); labelSet(lnum[1], ""); labelSet(lnum[2], "");
81          labelSet(lnum[3], ""); labelSet(lnum[4], ""); labelSet(lnum[5], "");
82          labelSet(lnum[6], ""); labelSet(lnum[7], ""); labelSet(lnum[8], "");
83          labelSet(lnum[9], "");
84          labelSet(lprice[0], ""); labelSet(lprice[1], ""); labelSet(lprice[2], "");
85          labelSet(lprice[3], ""); labelSet(lprice[4], ""); labelSet(lprice[5], "");
86          labelSet(lprice[6], ""); labelSet(lprice[7], ""); labelSet(lprice[8], "");
87          labelSet(lprice[9], "");
88          labelSet("label31", "");
89
90          buttonVis(upb[0], 0); buttonVis(upb[1], 0); buttonVis(upb[2], 0);
91          buttonVis(upb[3], 0); buttonVis(upb[4], 0); buttonVis(upb[5], 0);
92          buttonVis(upb[6], 0); buttonVis(upb[7], 0); buttonVis(upb[8], 0);
93          buttonVis(upb[9], 0);
94          buttonVis(downb[0], 0); buttonVis(downb[1], 0); buttonVis(downb[2], 0);
95          buttonVis(downb[3], 0); buttonVis(downb[4], 0); buttonVis(downb[5], 0);
96          buttonVis(downb[6], 0); buttonVis(downb[7], 0); buttonVis(downb[8], 0);
97          buttonVis(downb[9], 0);
98      }
```

続けて，フォームデザイナー上で「豚ロース肉 200g」と書かれた button21 をダブルクリックして，イベントハンドラ button21_Click の処理内容（赤枠の部分）を記述する。

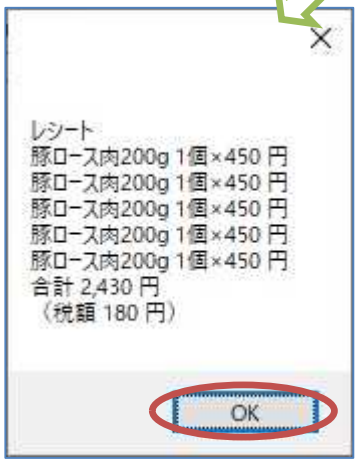
ここで，Format メソッドは, 指定された形式に基づいてオブジェクトの値を文字列に変換し，別の文字列に挿入する。書式指定文字列に関しては，第 3 回の教材の p.2 で紹介したが，{0}は書式指定項目で，その位置に文字列値が挿入されるオブジェクト（カンマの後に書かれた変数や配列の要素）の添え字（1 個目は 0，2 個目は 1）である。{0:#,0}と書いた場合，数値が 3 桁区切りとなる。

```
        1 個の参照
100     private void button21_Click(object sender, EventArgs e)
101     {
102         labelSet(lproduct[count], button21.Text);
103         buttonVis(upb[count], 1);
104         buttonVis(downb[count], 1);
105         labelSet(lnum[count], num[0].ToString());
106         labelSet(lprice[count], "×" + String.Format("{0:#,0} 円", price[0]));
107         total += taxIncluded(price[0] * num[0], tax[0]);
108         labelSet("label31", "合計 " + String.Format("{0:#,0} 円", total));
109         receipt += button21.Text + " " + num[0].ToString() + "個×"
110             + String.Format("{0:#,0} 円¥n", price[0]);
111         count++;
112     }
```

最後に，フォームデザイナー上で「会計」と書かれた button33 をダブルクリックして，イベントハンドラ button33_Click の処理内容（赤枠の部分）を記述する。

```
113
        1 個の参照
114     private void button33_Click(object sender, EventArgs e)
115     {
116         MessageBox.Show(receipt + "合計 " + String.Format("{0:#,0} 円¥n", total)
117             + " (税額 " + String.Format("{0:#,0} 円", taxsum) + ") ");
118     }
119 }
120 }
```

ここで，『Form1.cs の保存』ボタンを押してから，『開始』ボタンを押して，プログラムを実行する。



提出物：

1）フォームのデザインファイル Form1.Designer.cs をメールに添付して提出する。

2）コードエディタで編集したソースファイル Form1.cs をメールに添付して提出する。

3）実行して，アプリが起動後，button21 を複数回クリックした時点のスクリーンショット 第 7 回実行結果.jpg（.png も可）をメールに添付して提出する。

4）上の状態の後，button33 をクリックして表示されたレシートのスクリーンショット 第 7 回レシート.jpg（.png も可）をメールに添付して提出する。

5）質問を記述したファイル Prog2_Questions_7th.txt に解答を書き込んで保存し，メールに添付して提出する。