Ｃ＃早見表

2023年12月2日　　16:34

お願い
・この早見表は、C++からC#に移行する課程で筆者の主観に基づいて作られた覚え書きです。
・広く一般ユーザ様に向けたわかりやすい資料ではありません。
・皆さん一人一人が自分専用の「早見表」を学習の過程で作り上げることをおすすめします。

フォーム上でリターンキーを検出する
フォームにタイマーを追加する
カレンダー
ファイルの日付時刻を取得する
レクタングル
コンテキストメニューの作成
コンテキストメニューを無効にする
マウスの右、左を識別する
マウス左クリックでメニューを開く
コンボボックス
パラメータの参照渡し
タイマー処理
遅延処理

グラフィック処理
グラフィックのペンカラーをＲＧＢで指示する
テキストカラー取得関数

メッセージボックスの表示と選択
グラフィックでJPG画像を配置する
グラフィックテキストの囲み座標取得
グラフィック要素の範囲選択
ピクチャボックスに写真を配置する
グラフィック謎現象
クリップボードに文字列を代入する
タブの保持がされないとものすごく不便です
TXDB_マクロの実行
シリアル通信
タイムアウトのエラー取得
受信処理
送信処理
例外処理

ここから先は、テキストデータベースのアブの専用です

疑問点、判らないことなど

com_で統一したprivate関数は範囲内で利用可能だが、publicとの違いがわからない
グローバル変数のstatic の解放条件などが不明
関数のstatic も役割が不明　付けても付けなくても同じような気がする

調査待ち

画像ファイルの保存時などに、指定されたファイルが保存できるファイルの拡張子かどうかをチェックするロ

ジックです。
LINQを使えば1行で書けます。

```
// 例１）Falseが返ります
var res1 = new string[]{".BMP", ".JPG", ".GIF", ".PNG",
"GIF"}.Contains(System.IO.Path.GetExtension("c:/test1.gif").ToUpper());
// 例２）Trueが返ります
var res２ = new string[]{".BMP", ".JPG", ".GIF", ".PNG",
"GIF"}.Contains(System.IO.Path.GetExtension("c:/test2.txt").ToUpper());
```

LINQ便利すぎる

変数の種類

short cnt;　16ビット整数-32000 から +32000

int x,y;　　32ビット整数

int型に格納できる値の範囲	
最小値	-2,147,483,648
最大値	2,147,483,647

long rec;　　64ビット整数

long型に格納できる範囲	
最小値	-9,223,372,036,854,775,808
最大値	9,223,372,036,854,775,807

double dx,dy;　64ビット実数

正確な小数計算をしたい時は「decimal型」

decimal型はdouble型と違い、正確な小数計算が行えます。
お金に関する計算をシステムで行うなど、小数以下も正確に計算したいときにはdecimalを使用しましょう。

ただし、decimalは処理速度などに影響がでますので多用はお勧めしません。**精度の高い計算がしたい時だけ使用するようにしましょう。**

string　　文字列

string[]　　文字配列

自動変数{}(ブレースと呼ぶ)の内部でのみ有効、外に出ると消滅する

グローバル変数

```
    public partial class Form1 : Form
    {
    //この内側出あればどこでも有効である
    // プログラムの終了で消滅する
  }

namespace Goma_den8_ans
{
    public partial class Form1 : Form
    {
        // グローバル変数領域
        // 関数名はすべて

        public static string[] Prm_nam_tbl = new string[100 + 1];
```

変数のローカルルール

・自動変数は全部小文字、グローバル変数は先頭の1文字あるいは2文字とかを、大文字で表記する
・こうすることで、一目で、どちらかを識別出来る
・グローバル変数は、意味のわかりやすい、長めの命名をするとよい
・関数に指示するパラメータ変数は、必ず先頭にprm_ を付け、役割が判るようにする
・グローバル変数はむやみに増やさない事を心がける、自動変数で処理出来る物は必ず自動変数で処理をする
・関数の書式は、必ず先頭を大文字にする事を要求されるから、グローバル関数の書式を新たに
考えなければいけない　先頭3文字すべてを大文字にするとか？

```
if(lst[0] == "[[Format]]")
if (lst[0] == "[[Standard]]")

        Glo.Select_lst_ans = ""; // 新規連続中止
        Glo.Prm_tbl[8] = "";    // 前回動作メインモード
        //Glo.Prm_tbl[17] = "";   // ビュー対象
```

ツリービューの色変更

```
treeView1.Nodes[i].ForeColor = Glo.Color_tbl[1];
```

・メッセージ関数
・変数の種類
・自動変数とグローバル変数
・変数の名前
・ボタンコントロールを追加する
・文字列
・整数
・文字配列
・if
・for
・while()
・swich()
・brake
・continue
・文字列を数値に変換する
・数値を文字列に変換する
・任意の区切り文字で文字列を分解する

・文字列を検索する
・任意の位置から文字を取り出す
・配列を逆順にする

コードの見通しを良くする工夫

```
45          // エラーメッセージ
            7 個の参照
46          public static void err_msg_hsp(int prm_y,string prm_msg)...
50
51          // LO  MO  の先頭タイプ分解
            10 個の参照
52          public static string get_lst_box_cst_tbl_type(int prm_y,string prm_type_cod)...
64
65          // LO  MO  の代入コード分解
            10 個の参照
66          public static int get_lst_box_cst_tbl_cod(int prm_y,string prm_type,string prm_type_cod)...
83
```

・共用関数が長々と表示されると、全体の見通しが悪くなるので、修正が終わったらこのように折りたたんでいます。
・内部を確認する場合は、　　をクリックすると、展開して表示されます

答えが戻るパラメータ

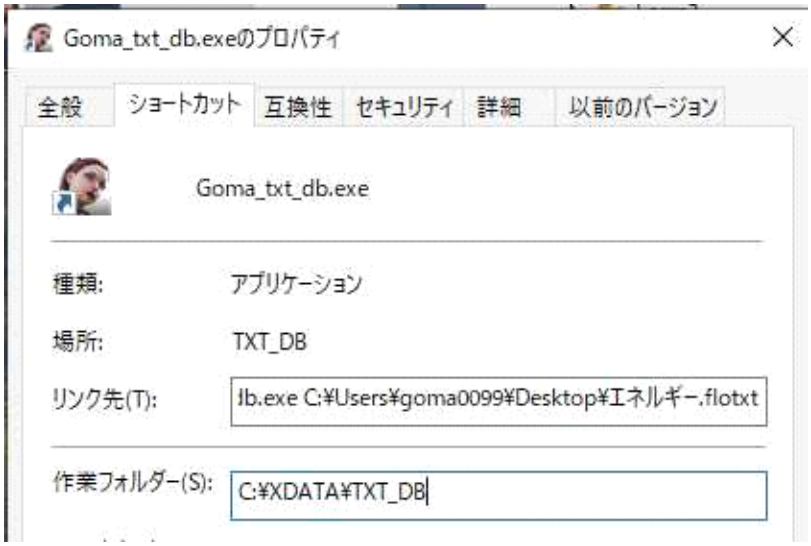
```
com_get_path("起動",ref fil_path);

private void com_get_path(string prm_type,ref string prm_parh)
{
   prm_path = "E:\\ゴマ電子";
}
```

複数起動の禁止処理


```
namespace Goma_txt_db
{
    internal static class Program
    {
        private static Mutex _mutex;

        /// <summary>
        /// アプリケーションのメイン エントリ ポイントです。
        /// </summary>
        [STAThread]
        static void Main()
        {
```

```
            //string[] cmds;

            _mutex = new System.Threading.Mutex(false, "GOMA_TXT_DB_EXE");
            //ミューテックスの所有権を要求する
            // Program.cs main() にペーストする
            // _mutex のエラーは、ALT+Enter で解消する
            if (_mutex.WaitOne(0, false) == false)
            {
                MessageBox.Show("本ソフトウェアは複数起動できません。");
                return;
            }

            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            //cmds = System.Environment.GetCommandLineArgs();
            // [0] プログラムパス [1] 引数1 [2] 引数2
            //MessageBox.Show(cmds[1]);

            Application.Run(new Form1());
        }
    }
}
```

起動パラメータの取得



・ショートカットで、特定のファイルを指定して実行する
・重複起動の後に、下記コードを追加して、表示するところまでは成功したが、Form1() にパラメータ指定するとプログラム自体が起動しない
・エクスプローラに拡張子 .flotxt を登録して、ファイルをダブルクリックして実行すると、表示までは成功するがプログラム自体が起動しない

・機会があれば調査をしてほしい
・今のところ、ぜひ　必要な機能でもない

```
    internal static class Program
    {
        private static Mutex _mutex;

        /// <summary>
        /// アプリケーションのメイン エントリ ポイントです。
        /// </summary>
        [STAThread]
        static void Main()
```

```
        {
            string[] cmds;

            _mutex = new System.Threading.Mutex(false, "GOMA_TXT_DB_EXE");
            //ミューテックスの所有権を要求する
            // Program.cs main() にペーストする
            // _mutex のエラーは、ALT+Enter で解消する
            if (_mutex.WaitOne(0, false) == false)
            {
                MessageBox.Show("本ソフトウェアは複数起動できません。");
                return;
            }

            Application.EnableVisualStyles();
     Application.SetCompatibleTextRenderingDefault(false);

            cmds = System.Environment.GetCommandLineArgs();
            // [0] プログラムパス [1] 引数1 [2] 引数2
            //MessageBox.Show(cmds[1]);         // ここまでは成功したが

            Application.Run(new Form1());
        }
    }
```

自作モニタ関数

・デバッグ機能の代わりに利用する変数などのモニタ関数です
msg("ここで画面表示して止まります");

・事例では、画面に文字列を表示し、処理が止まります
・練習用のリストボックスlistBox1 がある場合は、処理を止めないでモニタ出来ます

```
namespace Goma_txt_db
{
    public partial class Form1 : Form
    {
        public static void msg(String prm_msg)
        {
            //listBox1.Items.Add(prm_msg);
            MessageBox.Show(prm_msg);
        }
```

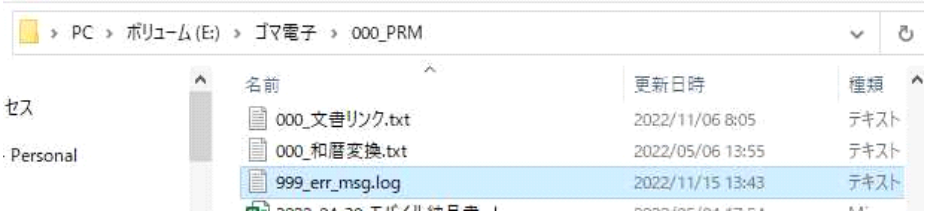
自作モニタ関数２

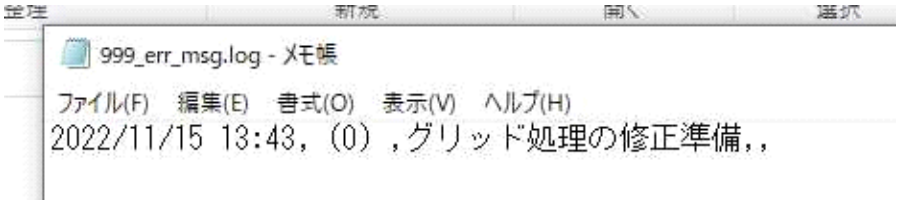
・デバッグ機能の代わりに利用する変数などのモニタ関数です
msg2("経路調査","","");
・画面を止めずに、変数、動作経路、タイミングなどをモニタします。
・カレンダーが記録されます
・タイミングをモニタするため、実行連番が記録されます。
・起動フォルダを取得し、Msg_path文字列変数に代入する必要があります

```
namespace Goma_txt_db
{
    public partial class Form1 : Form
    {
                public static int Msg_cnt = 0;
        public static string Msg_path = "";

        public static void msg2(String prm_msg, String prm_msg2, String prm_msg3)
        {
            // プログラムの問題調査の為のモニタ関数
            //    ログファイルに記録される
```

```
            // 起動先頭位置のファイルモニタの場合、パラメータ読込のエラー調査を
            // するばあい、パスを外します C:ドライブにファイルが生成されます
            //string fil_path = "999_err_msg.log";

            string fil_path = Msg_path + "¥¥999_err_msg.log";
            //string fil_path = Sub_nam_tbl[1] + "¥¥999_err_msg.log";
            int yy_all, mm, dd, tim_h, tim_m;
            string txt_buf;

            DateTime now = DateTime.Now;     // カレンダー取得

            yy_all = now.Year;
            mm = now.Month;
            dd = now.Day;
            tim_h = now.Hour;
            tim_m = now.Minute;

            Encoding enc = Encoding.GetEncoding("Shift_JIS");
            using (StreamWriter wrt = new StreamWriter(fil_path, true, enc))
            {
                txt_buf = yy_all.ToString() + "/" + mm.ToString() + "/" + dd.ToString()
                    + " " + tim_h.ToString() + ":" + tim_m.ToString()
                + ", (" + Msg_cnt.ToString() + ") ," + prm_msg + "," + prm_msg2 + "," + prm_msg3;
                wrt.WriteLine(txt_buf);

                // クローズは必要ないらしい、括弧の外に出た時点でクローズされる
                // 書式を統一するか、検討する事
            }
            ++Msg_cnt;
        }
```

・記録されたモニタはこの事例では下記のフォルダに補算されます





起動フォルダ取得

```
//Glo.Sub_nam_tbl[0] = Directory.GetCurrentDirectory();
//msg(Glo.Sub_nam_tbl[0]);

Glo.Fil_nam_tbl[0] = Glo.Sub_nam_tbl[0] + "\\000_TXT_DBパラメータ.txt";
Glo.Fil_nam_tbl[1] = Glo.Sub_nam_tbl[0] + "\\000_グリッド名一覧.txt";

if (com_prm_red(Glo.Fil_nam_tbl[0]) == false) ++err_cnt;  // パラメータの読込

// システムデータフォルダ
```

```
Glo.Sub_nam_tbl[1] = Glo.Prm_tbl[2] + "\\" + "000_システム定義";
Directory.CreateDirectory(Glo.Sub_nam_tbl[1]);
//msg(Glo.Sub_nam_tbl[1]);

// フォルダを生成する
Glo.Sub_nam_tbl[5] = Glo.Sub_nam_tbl[1] + "\\" + "グリッド配置";
Directory.CreateDirectory(Glo.Sub_nam_tbl[5]);
//msg(Glo.Sub_nam_tbl[5]);

Glo.Sub_nam_tbl[6] = Glo.Sub_nam_tbl[1] + "\\" + "ハッシュ";
Directory.CreateDirectory(Glo.Sub_nam_tbl[6]);

Glo.Sub_nam_tbl[7] = Glo.Sub_nam_tbl[1] + "\\" + "文書原稿";
Directory.CreateDirectory(Glo.Sub_nam_tbl[7]);

Glo.Sub_nam_tbl[8] = Glo.Sub_nam_tbl[1] + "\\" + "カレンダー";
Directory.CreateDirectory(Glo.Sub_nam_tbl[8]);

Glo.Sub_nam_tbl[9] = Glo.Sub_nam_tbl[1] + "\\" + "パス定義";
Directory.CreateDirectory(Glo.Sub_nam_tbl[9]);

Glo.Sub_nam_tbl[10] = Glo.Sub_nam_tbl[1] + "\\" + "頻度";
Directory.CreateDirectory(Glo.Sub_nam_tbl[10]);

Glo.Sub_nam_tbl[11] = Glo.Sub_nam_tbl[1] + "\\" + "クリップボード";
Directory.CreateDirectory(Glo.Sub_nam_tbl[11]);

Glo.Sub_nam_tbl[20] = "E:\\ゴマ電子\\000_システム定義\\キー文書";
Directory.CreateDirectory(Glo.Sub_nam_tbl[20]);

Glo.Sub_nam_tbl[21] = Glo.Sub_nam_tbl[0] + "\\" + "エクセル変換";
Directory.CreateDirectory(Glo.Sub_nam_tbl[21]);


  // 0.現在グリッド名
  // 1.フォーカスタイプ  1:グリッド一覧 2:フォルダツリー
  // 2.
  // 3.簡易一覧タイプ
  // 4.
  // 5.選択下位部分パス
  // 6.選択フォルダフルパス
  // 7.選択済みフォルダノード名
  // 8.前回動作選択　フォルダ生成　リンク文書生成　フォルダ検索　キーワード検索
  // 9.
  // 10.
  // 11.
  // 12.
  // 13.
  // 14.
  // 15.現在行_Row
  // 16.現在列_Column
  // 17.ビュー対象のテキストファイル
  // 18.グリッド対象のテキストファイル
  // 19.
  // 20.
  // 21.
  // 22.
  // 23.
  // 24.
  // 25.
  // 26.
```

```
    // 27.
    // 28.
    // 29.
    // 30.
    // 31.
    // 32.
    // 33.
    // 34.
    // 35.
    // 36.
    // 37.
    // 38.
    // 39.
    // 40.和暦 xx
    // 41.西暦
    // 42.月
    // 43.日
    // 44.ツリーデータ先頭パス
    // 45.コンピュータ名
    // 46.ユーザ名
    // 47.グリッド書式編集  0:データモード 1:グリッド編集モード
    // 48.
    // 49.
    // 50.
    // 51.
    // 52.
    // 53.
    // 54.
    // 55.
    // 56.
    // 57.
    // 58.
    // 59.
```

配列定義

```
public static string[] Sub_nam_tbl = new string[50];
string[] nam_tbl = new string[50];

int[,] dat_tbl = new int[4, 2];       // 2次元配列

string[,] Dat_tbl = new string[10,20];

Dat_tbl.GetLength(0);         //= 10;
Dat_tbl.GetLength(1);         //= 20;
```

★　過去の習慣が不要になる事例
・配列のサイズはLength; Length(0); などで取得出来るため、外部でサイズ指定は必要ない
・関数の内部で、配列のサイズを自動的に取得出来る

2次元配列をパラメータで渡す方式

```
string[,] dat_tbl = new string[2,50];

void tbl_set(string[,] prm_dat_tbl)
{
    int xtbl_max = prm_dat_tbl.GetLength(0);
    int ytbl_max = prm_dat_tbl.GetLength(1);

}
```

配列のコピー
// Array.Copyメソッドでコピー
Array.Copy(a, i, b, j, n);
// 配列aの(i+1)番目の要素を配列bの(j+1)番目の位置へと、順にn個をコピーする。

ArrayクラスのCopyメソッド
CopyToメソッド
BufferクラスのBlockCopyメソッド

LINQのConcatメソッド

```
byte[] buf = new byte[2048];
int[] tbl = new int[10];
Array.Copy(tbl,0,buf,0,10);     // int配列をbyte配列にコピーする
```

配列の件数を取得する

kensu = tbl.Length;

```
            string[] cut_tbl = new string[20];
            int key_max;
            string msg_cst = "行動記録 開発作業 パソコン";

            cut_tbl = msg_cst.Split(" ");
            key_max = cut_tbl.Length;

            EditText edittext1 = FindViewById<EditText>(Resource.Id.editText1);
            edittext1.Text = key_max.ToString();
```

上記の結果は 3 でした

・分解する文字列が空欄の場合

```
            string[] cut_tbl = new string[20];
            int key_max;
            string msg_cst = "";

            cut_tbl = msg_cst.Split(" ");
            key_max = cut_tbl.Length;

            EditText edittext1 = FindViewById<EditText>(Resource.Id.editText1);
            edittext1.Text = key_max.ToString() + "(" + cut_tbl[0]+")";
```

上記の結果は1()　でした

文字列の長さを取得する

```
string msg_cst = "abc";
len = msg_cst.Length;
```

文字列を数値に変換

```
int cnt = int.Parse(Glo.Prm_tbl[0]);
```

文字列を数値に変換可能か、確認する処理

```
int i = 0;
string s = "108";
bool result = int.TryParse(s, out i); //i now = 108
```

★　例外を発生させずに変換する方法

```
cst_th = Prm_tbl[50].Substring(0, 2);
cst_tm = Prm_tbl[50].Substring(2, 2);
if (int.TryParse(cst_th, out th) == true && int.TryParse(cst_tm, out tm))
{
```

}

説明書から抜粋
int i = 0;
bool b = int.TryParse("123", out i);
b → true、i → 123

bool b = int.TryParse("ABC", out i);
b → false、i → 0

文字列の検索
cnt = moto_cst.IndexOf("文字", 13); // 指示された位置からの発見位置
msg(cnt.ToString());

文字列が含まれるか判定する

```
if (prm_bun_cst.Contains(mast_key_cst) == true) return (true);
else return (false);

            // テストモニタ
            string key = "行動記録 開発業務 スマホ ";
            string dat = "行動記録 開発業務 スマホ ";

            if (key.Trim().Contains(dat.Trim()) == true) msg_cst += "一致しました" + "¥r¥n";
            else msg_cst += "-------------" + "¥r¥n";
```

上記の結果は　「一致しました」です

String.Contains メソッド

オーバーロード
Contains(Char)
指定した文字がこの文字列内に存在するかどうかを示す値を返します。

Contains(String)
指定した部分文字列がこの文字列内に存在するかどうかを示す値を返します。

Contains(Char, StringComparison)
指定された比較規則を使用して、指定された文字がこの文字列内に含まれるかどうかを示す値を返します。

Contains(String, StringComparison)
指定された比較規則を使用して、指定された文字列がこの文字列内に含まれるかどうかを示す値を返します。

public bool Contains (string value);

文字列に半角数値が含まれているか確認する

```
Msg(Regex.IsMatch("aa123", @"^[0-9]+$").ToString());        False
Msg(Regex.IsMatch("123", @"^[0-9]+$").ToString());         True
```

正規表現パターンは次のとおりです。

コピー

```
^[A-Z0-9]\d{2}[A-Z0-9](-\d{3}){2}[A-Z0-9]$
```

次の表に、正規表現パターンがどのように解釈されるかを示します。

パターン	説明
`^`	文字列の先頭から照合を開始します。
`[A-Z0-9]`	から までの任意の 1 つのアルファベット文字`A Z`、または任意の数字と一致します。
`\d{2}`	2 つの数字と一致します。
`[A-Z0-9]`	から までの任意の 1 つのアルファベット文字`A Z`、または任意の数字と一致します。
`-`	ハイフンと一致します。
`\d{3}`	正確に 3 つの数字と一致します。
`(-\d{3}){2}`	ハイフンと 3 つの数字を検索し、このパターンの 2 つの出現箇所と一致します。
`[A-Z0-9]`	から までの任意の 1 つのアルファベット文字`A Z`、または任意の数字と一致します。
`$`	入力文字列の末尾で照合を終了します。

文字列を抜き出す
prm_cell_name.Substring(0,5);

文字列を特定のコードで区切る

文字列を特定のコードで区切った後の件数取得
cell_kosu = txt_buf.Split(',').Length;

文字列から空白を除外する
msg_cst.Trim();　　// 前後の削除
msg_cst.TrimStart();　// 前方の削除
msg_cst.TrimEnd();　// 後方の削除

文字列をカンマで区切る
txt_buf.Split(',')[0];

文字列を全角スペース、半角スペース、どちらでも区切る
msg_cst = moto_cst.Split(' ', '')[i];

パスを￥で区切り編集部ベースを作る
string[] fil_lst;
string[] cut_lst;

文字配列に分解した文字配列を受け取る
cut_lst = fil_lst[i].Split('\\');

それによって、最終ブロックを下記の一行で取得可能
max2 = cut_lst.Length;
end_cst = cut_lst[max2-1]; // パス分解後の最終文字列

end_cst = cut_lst[max2 - 1];
listBox1.Items.Add(end_cst);
listBox1.Items.Add(end_cst.Split('.')[0]); // 拡張子を除いたファイル名
listBox1.Items.Add(end_cst.Split('.')[1]); // 拡張子

文字列の半角大文字、半角小文字変換

```
string msg_cst = "c:\\test2.txt";
msg(msg_cst.ToUpper());       // 大文字変換

msg_cst = "C:\\TEST2.TXT";
msg(msg_cst.ToLower());       // 小文字変換
```

文字列からchar型配列に変換する

```
char[] cha_tbl = prm_file_path.ToCharArray();
string ans_cst;
for (y=0;y<cha_tbl.Length;y++)
{
    if (cha_tbl[y] == ':') cha_tbl[y]  = '_';
    if (cha_tbl[y] == '¥¥') cha_tbl[y] = '_';
    if (cha_tbl[y] == '.') cha_tbl[y]  = '_';
}
ans_cst = new string(cha_tbl);
```

文字列からCRLFを除外する

```
string get_file_path = prm_file_path.Replace("¥r", "").Replace("¥n", "");
```

文字列から囲み記号を判定する

```
if(get_txt.Substring(0,1) == "[")
{
    get_txt_len = get_txt.Length;
    if(get_txt.Substring(get_txt_len-1,1) == "]")
    {
        type_cod = 3;
    }
}
```

数値を文字列に変換

```
Glo.Prm_tbl[0] = cnt.ToString();

cnt = moto_cst.IndexOf("文字");      //先頭からの発見位置
msg(cnt.ToString());
```

数値を文字列に変換する時の書式

```
(12345).ToString("D") = "12345"
(12345).ToString("D8") = "00012345"
(-12345).ToString("D6") = "-012345"
(12345).ToString("D4") = "12345"

(10000).ToString("C") = "\10,000"
(-100.5).ToString("C") = "-\101"
(12345.5555).ToString("C2") = "\12,345.56"

(123.456).ToString("E") = "1.234560E+002"
(0.0).ToString("e") = "0.000000e+000"
(-123.45).ToString("E8") = "-1.23450000E+002"
(0.12345).ToString("E3") = "1.235E-001"

(123.456).ToString("F") = "123.46"
(1).ToString("F") = "1.00"
(-123.456789).ToString("F4") = "-123.4568"
(0.1).ToString("F6") = "0.100000"
```

(1234.56789).ToString("N") = "1,234.57"
(10000).ToString("N") = "10,000.00"
(-1234.56789).ToString("n4") = "-1,234.5679"

(1).ToString("P") = "100.00%"
(-0.123456789).ToString("p") = "-12.35%"
(100).ToString("P") = "10,000.00%"
(0.00001).ToString("P") = "0.00%"
(0.123456789).ToString("P4") = "12.3457%"
(0.125).ToString("P0") = "13%"

(10).ToString("X") = "A"
(-1).ToString("x") = "ffffffff"
(10).ToString("X4") = "000A"

利用可能なドライブ一覧を取得する

```
string[] drives = Directory.GetLogicalDrives();

            //DriveInfo driveInfo = new DriveInfo("C:¥¥");
            DriveInfo driveInfo = new DriveInfo("E:¥¥");
            //DriveInfo driveInfo = new DriveInfo("¥¥¥¥goma001¥¥goma0099¥¥");

            listBox1.Items.Add("ボリュームラベル :"+driveInfo.VolumeLabel);
            listBox1.Items.Add("ファイルシステム名 :" + driveInfo.DriveFormat);
            listBox1.Items.Add("種類 :"+ driveInfo.DriveType.ToString());
            listBox1.Items.Add("容量 :"+ driveInfo.TotalSize.ToString());
            listBox1.Items.Add("空き領域 :"+ driveInfo.TotalFreeSpace.ToString());
            listBox1.Items.Add("準備 :"+ driveInfo.IsReady.ToString());
            listBox1.Items.Add("ルート :"+ driveInfo.RootDirectory.Name);
```

・ネットワークドライブは、X:\などに割り当てが必要かもしれません

フォルダの一覧を取得する

```
string[] fold_lst;
int fold_lst_kosu;

//配下のフォルダ全てを取得する
fold_lst = System.IO.Directory.GetDirectories(prm_root_path
   , "*", System.IO.SearchOption.AllDirectories);
fold_lst_kosu = fold_lst.Length;

// 直下のフォルダのみ
fold_lst = System.IO.Directory.GetDirectories(moto_path
   , "*", System.IO.SearchOption.TopDirectoryOnly);
   fold_lst_kosu = fold_lst.Length;

配列の最大数max = fil_tbl.Length;
```

フォルダの存在検査

```
if(Directory.Exists(new_path) == true)
{ msg("同じフォルダ名があります　中止します"); return;}
```

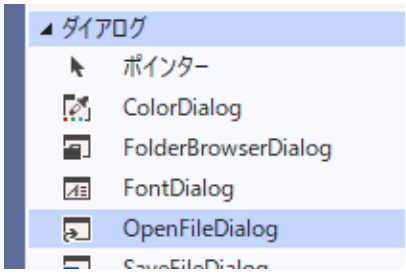
フォルダ生成

```
Directory.CreateDirectory(Glo.Sub_nam_tbl[1]);
```

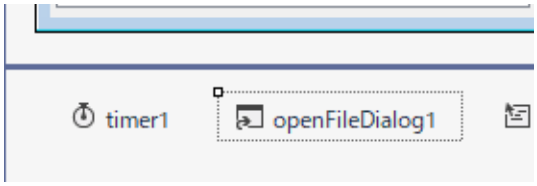
配下のサブフォルダ取得

```
string[] sub_lst;

sub_lst = System.IO.Directory.GetDirectories("E:\\ゴマ電子", "*", System.IO.SearchOption.AllDirectories);
//取得結果をリストボックスに一括表示
listBox1.Items.AddRange(sub_lst);
```
(配下の根っこまで全て検索する)

ファイル選択のコントロールを利用する



・ドラッグして、フォームに配置する



ファイル一覧の取得

件数定義はなしフルパスが取れるので除去必要

```
string[] fil_lst;
```

・フォルダ一覧の取得には、トップのみと、配下全てがあった

```
// 配下フォルダのファイルは含まない
fil_lst = System.IO.Directory.GetFiles(fild_path, "*.*", System.IO.SearchOption.TopDirectoryOnly);
```

★　これは、配下のフォルダも含むので処理してはいけない

```
fil_lst = System.IO.Directory.GetFiles(Glo.Sub_nam_tbl[2]
, "*.txt", System.IO.SearchOption.AllDirectories);
```

ファイルパスから拡張子を取得する

拡張子を取得するには、System.IOの Pathクラスを使用します。
Path.GetExtension((ファイル名文字列));
で拡張子を取得できます。
GetExtension()メソッドは、"."を含む拡張子の文字列を返します。

ファイルパスからファイル名を抽出する

```
string filePath = Path.GetFileName(@"C:Samurai\samurai.txt");
```
samurai.txtの部分が取得出来る

```
// 拡張子を除外したファイル名を取得する
fil_name =  Path.GetFileNameWithoutExtension(fil_lst[i]);
```

一覧リストのパスを除外する

```
int fil_lst_kosu,x,y;
string[] cst_tbl;

// パスを除外し、ファイル名だけにする
for (y = 0; y < fil_lst_kosu; y++)
{
cst_tbl = fil_lst[y].Split('\\');  // フォルダをバラバラに分割し配列に代入する
fil_nam = cst_tbl[cst_tbl.Length - 1];  // 最後の要素にファイル名が残る
fil_lst[y] = fil_nam;          // ファイル名だけを再度保存する
}
```

ファイル削除

```
File.Delete(fil_path);
```

ファイルのコピー

```
File.Copy(in_path, out_path);

File.Copy(moto_path, saki_path,true); // オーバーライトコピー
```

ファイル名の変更

ファイルの存在検査

```
if (File.Exists(prm_fil_path) == true)
{
}
```

テキストファイルの終端判定

```
if (red.EndOfStream == true) break;
```

ファイルレコード読込

```
Encoding enc = Encoding.GetEncoding("Shift_JIS");
using (StreamReader red = new StreamReader(file_path, enc))
{
    if (red.EndOfStream == true) break;
    txt_buf = red.ReadLine();

    sta = prm_cut_name.IndexOf(prm_key_lst[x]);
    if (0 <= sta) { ++find_lst[x]; break; }
}
```

```
Encoding enc = Encoding.GetEncoding("Shift_JIS");
using(StreamReader red = new StreamReader(txt_path, enc))
{
  // 従来通りの動きを維持する為、パスを読込する
  Glo.Sub_nam_tbl[0] = red.ReadLine();



}
```

今後は、出来るだけこの書式に統一します。
シンプルだし、Close()を忘れる心配
上記のサンプルコードは1行だけなので括弧も省略出来るかもしれない

```
Encoding enc = Encoding.GetEncoding("Shift_JIS");

StreamReader red = new StreamReader(prm_fil_path, enc);
if (red.EndOfStream == true) break;
txt_buf = sr.ReadLine();
sr.Close();    // usingを使う場合は不要です
```

ファイルレコード書き込み

true ：前回の後方に追記してくれる(特別な理由がなければ　こちら)
false：前回の書き込みを削除してしまう

```
Encoding enc = Encoding.GetEncoding("Shift_JIS");
using (StreamWriter wrt = new StreamWriter(fil_path, true, enc))
{
    txt_buf = yy_all.ToString() + "/" + mm.ToString()
     + "/" + dd.ToString() + " " + tim_h.ToString()
     + ":" + tim_m.ToString() + "," + prm_msg + ","
     + prm_msg2 + "," + prm_msg3;
    wrt.WriteLine(txt_buf);

    // クローズは必要ないらしい、括弧の外に出た時点でクローズされる
    // 書式を統一するか、検討する事
}

Encoding enc = Encoding.GetEncoding("Shift_JIS");
StreamWriter writer = new StreamWriter(rcv_path, true, enc);

writer.WriteLine("テスト,123");     // テキストを書き込む
writer.WriteLine("テスト文字列,aac");     // テキストを書き込む
writer.Close();             // ファイルを閉じる
```

バイナリファイルの処理調査

・読み書きには、バイト配列を使うようだ

```
byte[] buf  = new byte[1024];
int 変数をバイナリ配列にコピーする
int cnt:
buf = BitConverter.GetBytes(cnt);
```

システム情報取得

```
Glo.Prm_nam_tbl[20] = "10.コンピュータ名";
Glo.Prm_tbl[20] = Environment.MachineName;
Glo.Prm_nam_tbl[21] = "11.ユーザ名";
Glo.Prm_tbl[21] = Environment.UserName;
```

プロセス関連

```
System.Diagnostics.Process[] ps =
    System.Diagnostics.Process.GetProcessesByName("notepad");

foreach (System.Diagnostics.Process p in ps)
{
    //プロセスを強制的に終了させる
    p.Kill();
}
```

```
// シャットダウン実行

//プロセスを強制的に終了させる
System.Diagnostics.Process[] ps = System.Diagnostics.Process.GetProcessesByName("DENSHIN8");
foreach (System.Diagnostics.Process p in ps)p.Kill();
```

★"Denshin8.exe"を使って失敗した、拡張子はいらない

シャットダウン実行

```
System.Diagnostics.Process.Start("shutdown.exe","/s /f");
```

★　自動メールサーバーからは実行出来なかった

グリッド処理

```
dataGridView1.ColumnCount = 2;     // 個数定義がないとエラーになりexeが起動しない

dataGridView1.Columns[0].HeaderText = "名称"; // 名称設定
dataGridView1.Columns[0].Name = "名称";       // 名称設定
dataGridView1.Columns[0].Width = 8 * 20;        // 幅設定


dataGridView1.Columns[1].HeaderText = "コピー内容"; // 名称設定
dataGridView1.Columns[1].Name = "コピー内容";       // 名称設定
dataGridView1.Columns[1].Width = 8 * 60;        // 幅設定


prm_grid.Columns[x].HeaderText = Grid_form_name[x];         // 名称設定
prm_grid.Columns[x].Name = Grid_form_name[x];         // 名称設定
prm_grid.Columns[x].Width = 8 * int.Parse(Grid_form_len[x]);     // 幅設定

dataGridView1.Rows.Clear();   // 全行消去
dataGridView1.Columns.Clear();  // 全カラム消去

private void dataGridView1_CellContentClick_1(object sender, DataGridViewCellEventArgs e)
      {
// データグリッドのセルをマウスでクリックした
// 漢字入力モードのコントロール
switch (e.ColumnIndex)
{
case 0:
case 1:
//この列はIME無効(半角英数のみ)
dataGridView1.ImeMode = System.Windows.Forms.ImeMode.Disable;
break;
default:
//この列は日本語入力ON(ひらがな)
dataGridView1.ImeMode = System.Windows.Forms.ImeMode.Hiragana;
break;
}
}

dataGridView1.ColumnCount = 6;

// 画面に表示される名称
dataGridView1.Columns[0].HeaderText = "開始";

// 値代入の為のＩＤ
dataGridView1.Columns[0].Name = "開始";
```

```
// カラム幅
dataGridView1.Columns[0].Width = 5 * 8;

dataGridView1.Columns[0].ToolTipText = "半角で時分を入力します";

// 行追加
dataGridView1.Rows.Add();

// セル代入
dataGridView1.Rows[0].Cells["開始"].Value = "4/7";

// セル値の読み取り
msg_cst = (string)dataGridView1.Rows[0].Cells["内容"].Value;

// 登録行数の取得
lin_kosu = dataGridView1.Rows.Count;
```

セルの位置を取得する場所には制限がある
下記は成功したが、別の場所ではRowIndexが使えない

グリッドセルをマウスクリックして位置を取得する

```
private void dataGridView1_RowEnter(object sender, DataGridViewCellEventArgs e)
{
   int x = e.RowIndex;    // グリッド行取得
   int y = e.ColumnIndex; // グリッド列取得
   string cell_name;
```

任意のセルを選択状態にする

```
dataGridView1.Rows[1].Selected = true;

//すべてのセルを並べ替え禁止にする
foreach (DataGridViewColumn c in dataGridView1.Columns)
   c.SortMode = DataGridViewColumnSortMode.NotSortable;
```

フォルダを開く、アプリを開く

```
            //起動フォルダを開く
            string gen_folder = Directory.GetCurrentDirectory();
            System.Diagnostics.Process.Start("EXPLORER.EXE", gen_folder);


System.Diagnostics.Process.Start("EXPLORER.EXE", txt_buf);

System.Diagnostics.Process.Start("EXPLORER.EXE", Glo.Sub_nam_tbl[0]);

//テキスト編集
System.Diagnostics.Process.Start("NOTEPAD.EXE", Glo.Fil_nam_tbl[0]);

//    ワードやエクセルを実行する
System.Diagnostics.Process.Start("EXPLORER.EXE", fil_path);


// メール送信実行
System.Diagnostics.Process.Start("goma_mail_snd.exe");

msg_cst.Format("Backup.exe \"%s\" \"%s\" -s -c -d -r -e --nonotify", cst_tbl[0], Day_path);
```

```
// SOTA実行
//複数のコピー元の処理では、待機する必要あり
System.Diagnostics.Process p;

//ファイルを開いて終了まで待機する
p =    System.Diagnostics.Process.Start("C:\\XDATA\\Backup\\Backup.EXE"
, "\"C:\\XDATA\\EXP\\\" \"E:\\TEST\\EXP\\\" -s -c -d -r -e --nonotify");
p.WaitForExit();

p = System.Diagnostics.Process.Start("C:\\XDATA\\Backup\\Backup.EXE"
, "\"C:\\XDEN8\\IN\\\" \"E:\\TEST\\IN\\\" -s -c -d -r -e --nonotify");
p.WaitForExit();
```

ツリービュー

```
TreeNode tn;

// 親ノード検索
node_kosu = treeView2.GetNodeCount(false);   // 親ノード件数

neme = treeView2.Nodes[i].Text;   // ノード名称

tn = treeView2.Nodes[i]; // 親ノートハンドル取得

tn = treeView2.Nodes.Add(node_name, node_name); // 新規ノード追加
tn.ForeColor = Color.FromArgb(50,50, 220);   // 色指定

node_kosu = tn.GetNodeCount(false);   // 配下ノード件数取得

// tnのハンドルにノードを追加する
tn = tn.Nodes.Add(node_name, node_name);


tn = treeView2.SelectedNode;   // 選択反転しているノードのハンドルを取得
node_tbl[0] = tn.Text;       // 選択されているノード名取得

tn = tn.Parent;           // 親ノードのハンドルを取得する

Glo.Prm_tbl[0] = treeView1.SelectedNode.Text;   //選択された名称を取得

tn.ToolTipText = Glo.Grid_nam[i];   //説明文

フォーカスを失うと、選択位置が解らなくなる
プロパティの HideSelection を False に変更する

マウスを乗せると説明文表示が出ない
プロパティの ShowNodesToolTips を True に変更する

// メインツリービューのセレクト変更　選択したい番号が判明している場合
treeView1.SelectedNode = treeView1.Nodes[1];

// ノードの名称で指示する場合　わわあ　賢いね
treeView1.SelectedNode = treeView1.Nodes["フォルダ検索"];
```

ツリービューのイベント調査

・ツリービューなどのコントロールには、イベント関数を定義することが出来ます。

・たとえば、マウスをクリックすることで実行される異変と関数は少なくとも書きの3種類があり、実行タイミングとか役割の調査が必要でした。
・マウスをクリックして実行されるイベントのタイミングは、下記のモニタ結果で知ることが出来ました。
・各イベントがなぜ別々に実行されるのか？役割は何かなどは、実験しながら体験してゆきます

```
2022/5/11 1:39, (0) ,treeView1_Click(),,
2022/5/11 1:39, (1) ,treeView1_MouseClick,,
2022/5/11 1:39, (2) ,treeView1_AfterSelect(),,
```

リストボックス

```
// 文字列を追加する
listBox1.Items.Add(Mac_txt_tbl[y]);

   // 選択されたリストボックスのインデックスを取得する
int sel_cnt = listBox1.SelectedIndex;

   // 選択されていいるリストボックスの文字列を取得する
string sel_cst = listBox1.Text;

// インデックスを指定して、選択状態にする
listBox2.SetSelected(z, true);
```

関数内部でlistBox2のように直接指定出来ない
下記のようにパラメータで渡す必要がある

```
switch (tbl_cod[1])
{
case 2: cpy_cst_tbl_to_lst_box(tbl_cod[0], listBox2); break;
case 3: cpy_cst_tbl_to_lst_box(tbl_cod[0], listBox3); break;
case 4: cpy_cst_tbl_to_lst_box(tbl_cod[0], listBox4); break;
case 5: cpy_cst_tbl_to_lst_box(tbl_cod[0], listBox5); break;
}
 public static void cpy_cst_tbl_to_lst_box(int prm_cst_xtbl_no,ListBox prm_lst_box)
{
prm_lst_box.Items.Clear();
}
```

リストボックスの最終行に移動する

```
// 最終行に移動
listBox1.SelectedIndex = listBox1.Items.Count - 1;
```

テキストボックス

テキストボックス内部でエンターキーが押された

```
private void richTextBox1_KeyDown(object sender, KeyEventArgs e)
{
      // 検索キー取得

      if (e.KeyCode == Keys.Enter)          // テキストボックスでエンターキーが押された
      {
```

フォーム

Form.ActiveForm.Close();　// フォームを終了します

フォームタイトル
this.Text = "コンテキストメニュー　テスト実行";

表示位置を呼び出し前に指示する
//form2.StartPosition = FormStartPosition.CenterParent;

フォームを最小化する
this.WindowState = FormWindowState.Minimized;

フォーム上マウスイベントの順番　左クリックの場合
2022/5/8 2:0, (0) ,Form1_MouseDown,,
2022/5/8 2:0, (1) ,Form1_Click,,
2022/5/8 2:0, (2) ,Form1_MouseClick,,

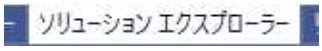
 // 下位フォーム呼び出し準備
Form2 form2 = new Form2(Glo.Sub_nam_tbl[1] + "\\000_文書リンク.txt");
form2.ShowDialog();　　// 下位フォーム呼び出し実行

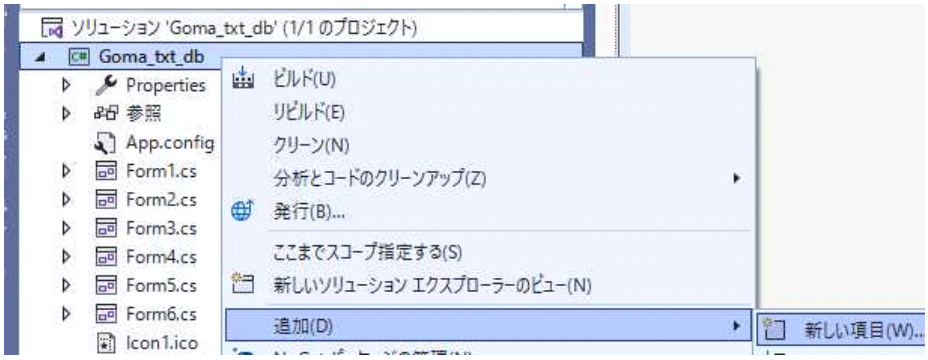
//フォームにツールチップを設定する

1. フォーム (Form1) にボタン (button1) を配置します。
2. フォーム (Form1) にツールチップ (toolTip1) を配置します。
private void Form1_Load(object sender, EventArgs e)
{
 toolTip1.SetToolTip(textBox1, "これは必須入力です。");
}


```
private void Form4_FormClosing(object sender, FormClosingEventArgs e)
{
    // フォームが閉じるときに処理します
}
```

フォームを追加する

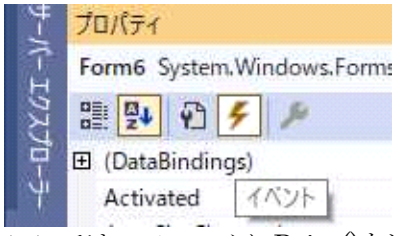




フォームを再表示する

```
this.Refresh();
```

フォーム、コントロールのイベントコードを自動生成する



ここでは、フォームにPaint()を追加する事例です

・最初は未登録であり、このように右側が空欄です。
・イベントの名称　ここではPaint をダブルクリックすると、下記のように、イベント名が追加されます

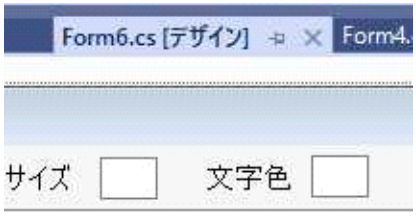
```
1 個の参照
private void Form6_Paint(object sender, PaintEventArgs e)
{

}
```

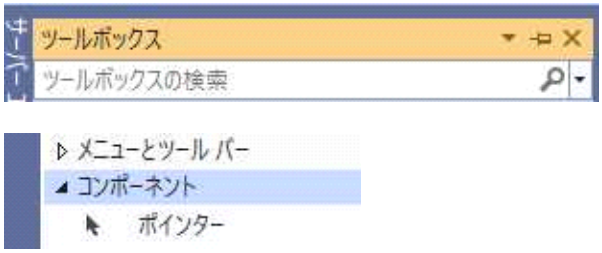
このように、コードが自動生成されます。　宣言などの処理も自動です。

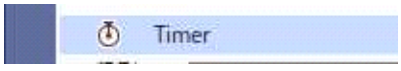
フォーム上でリターンキーを検出する

失敗した

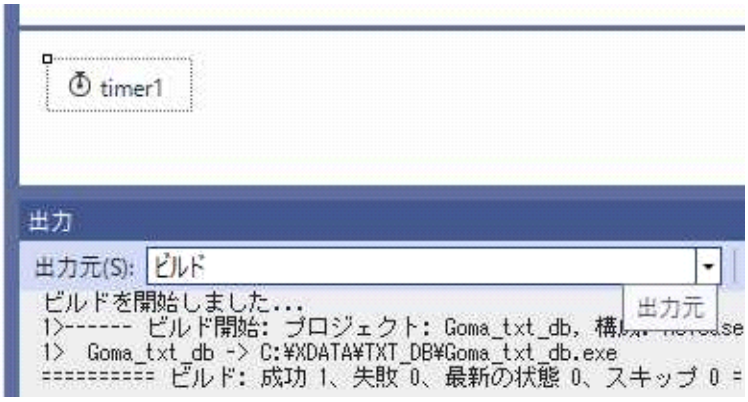
フォームにタイマーを追加する

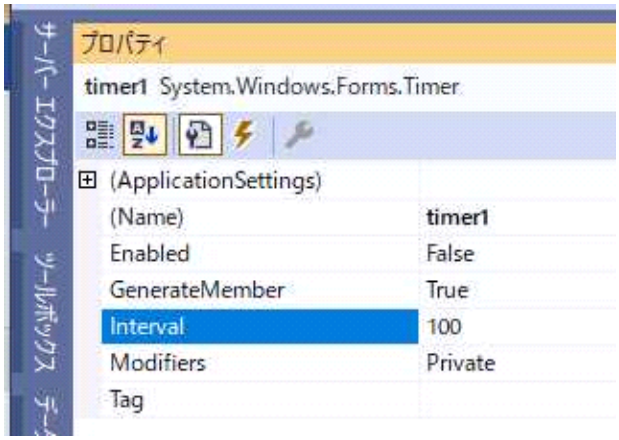






・フォームの上をクリックすると、タイマーが追加されます






```
timer1.Start();     // タイマーを開始します

private void timer1_Tick(object sender, EventArgs e)
{
    // テキストビュー依頼を実行する
    timer1.Stop();       // 停止
         com_main_tit_dsp();      // タイトル表示
         // テキストビュー依頼判定
         if (0 < Prm_tbl[19].Length)
    {
        // 例外エラーのため、停止する
        com_grid_on_txt_view();
        Prm_tbl[19] = "";
    }
    timer1.Start();      // スタート
}
```

カレンダー

```
   DateTime.Now

   DateTime now = DateTime.Now;

   yy_all = now.Year;
   mm = now.Month;
   dd = now.Day;
   tim_h = now.Hour;
   tim_m = now.Minute;
      week = (int)now.DayOfWeek;                  // 曜日はキャストするだけでよい 0:日曜

         DateTime dt = DateTime.Now;
         Glo.Prm_tbl[41] = dt.Year.ToString();
         Glo.Prm_tbl[42] = dt.Month.ToString();
         Glo.Prm_tbl[43] = dt.Day.ToString();

// 32 秒加算する
   dtBirth = dtBirth.AddSeconds(32);

DateTime from_dt = new DateTime(2015, 8, 4);
DateTime to_dt = new DateTime(yy_all, mm, dd);
day_renban = (long)(to_dt - from_dt).TotalDays; // 日数の差
```

ファイルの日付時刻を取得する

```
DateTime dt;
string fil_nam;

dt = File.GetLastWriteTime(fil_lst[y]);
```

ファイルの日付時刻で、写真ファイルの名称を生成する

```
fil_nam = "P_" + dt.Year.ToString("D4") + dt.Month.ToString("D2")+
 dt.Day.ToString("D2")+"_"+dt.Hour.ToString("D2")+
dt.Minute.ToString("D2")+dt.Second.ToString("D2")+".jpg";
```

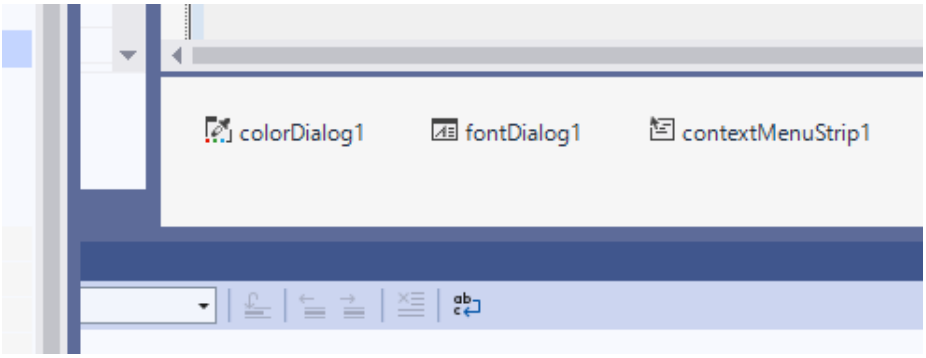
レクタングル

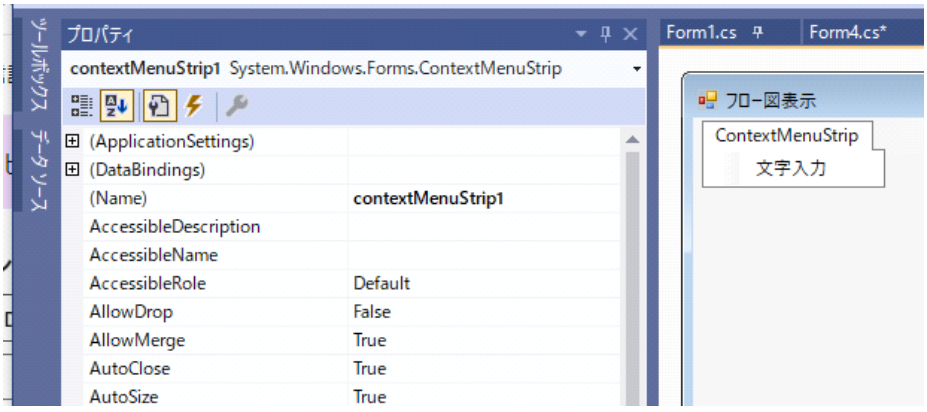
```
Rectangle r = new Rectangle();
r.Width = 50;
r.Height = 50;
```

コンテキストメニューの作成

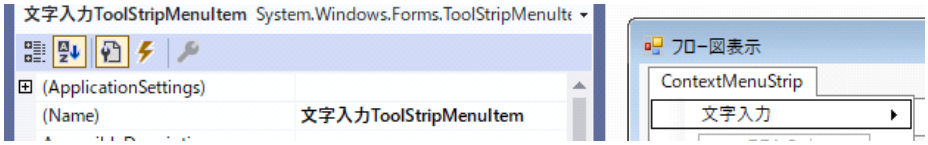
・ツールボックスから、コンテキストメニューを選択する

・フォーム上でクリックすると、下記のように配置される



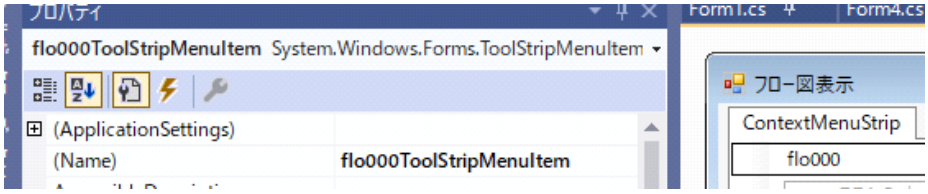


・これは失敗の事例です。最初に漢字で名称入力すると、コードデータに漢字が出てしまう

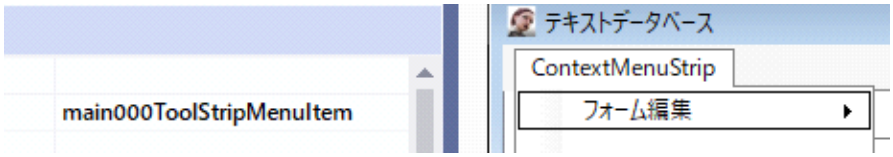


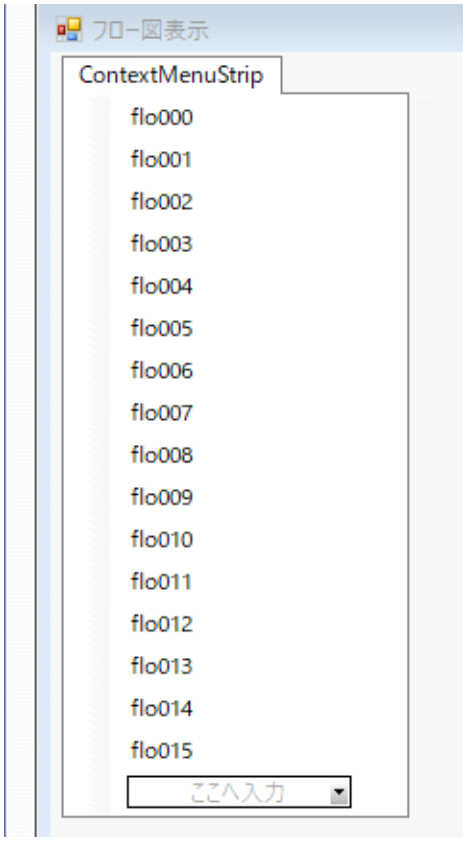
・このまま処理したら動かなくなりそうなので、一度削除します

フロー処理の番号で作成します。



・このように、アルファベットの管理コードが生成されます。後方の文字列は意味は分からない
・管理プログラムでも下記のように、残したままで処理しているのでこのままにします





・このまま、それぞれのメニューをダブルクリックし、イベントコードを生成します

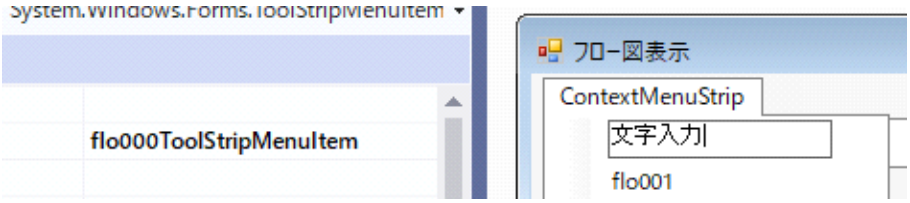
```
1 個の参照
private void flo000ToolStripMenuItem_Click(object sender, EventArgs e)
{

}
```

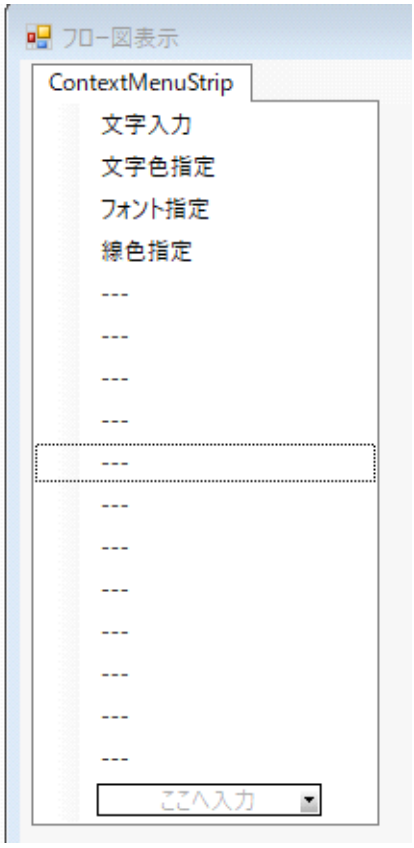
class System.Object
.NET Framework クラス階層のすべてのクラスを

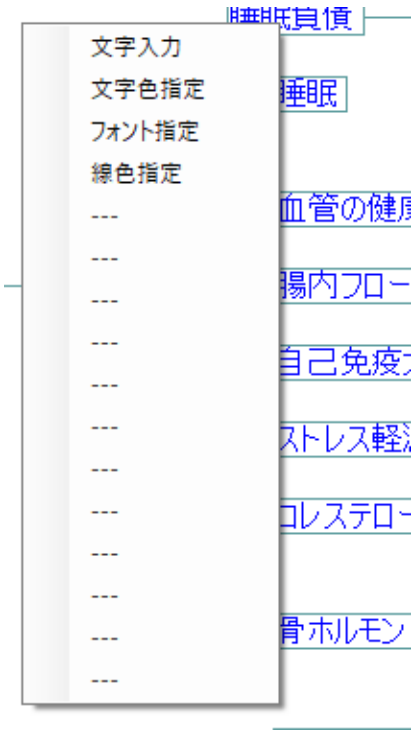
・すべてのメニューコードを生成します。並んで生成されるので後で管理がしやすいのです。

```
private void flo015ToolStripMenuItem_Click(object sender, EventArgs e)
{

}
```

・イベントコードが生成されたら、メニューのタイトルを日本語に変更します








```
public Form4(string prm_fil_path,string prm_fil_nam)
{
    InitializeComponent();

    this.ContextMenuStrip = contextMenuStrip1;  // コンテキストメニューメイン処理
```

・この処理を追加すると、右クリックでメニューが表示されます

コンテキストメニューを無効にする

```
this.ContextMenuStrip = null;
```

マウスの右、左を識別する

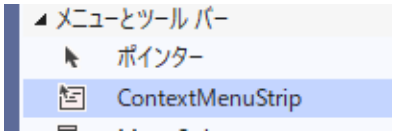
```
private void Form1_MouseClick(object sender, MouseEventArgs e)
{
  // ダイアログ　マウスクリック
  // 右でも、左でも反応する

  if (e.Button == MouseButtons.Left) msg("左");
  if (e.Button == MouseButtons.Right) msg("右");
}
```

★Form1_Click() では使えない

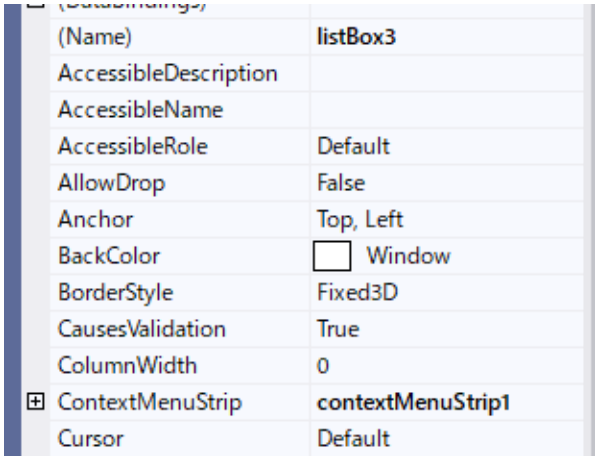
マウス左クリックでメニューを開く

```
private void Form4_MouseDown(object sender, MouseEventArgs e)
{
            gx = (short)e.X;
            gy = (short)e.Y;
            this.contextMenuStrip1.Show(gx, gy);
```

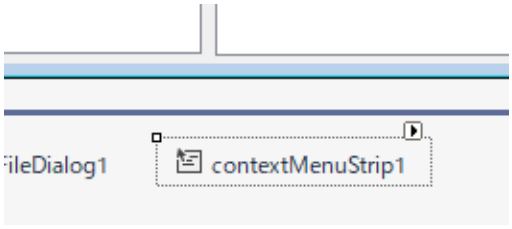
メニューの新規作成



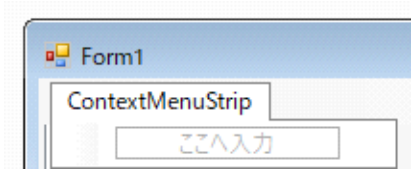
・対象フォームへドラッグして配置する

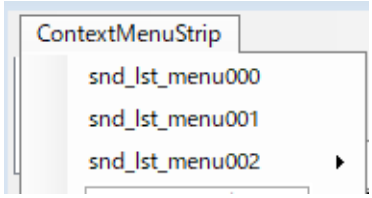


・この事例では、listBox3にメニューを割り当てする



・ドラッグしたアイテムにフォーカスを移動すると、フォームの左上に、メニュー登録の枠が表示される

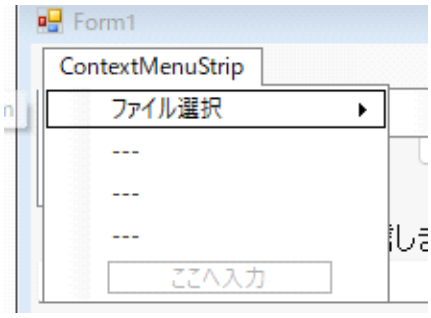




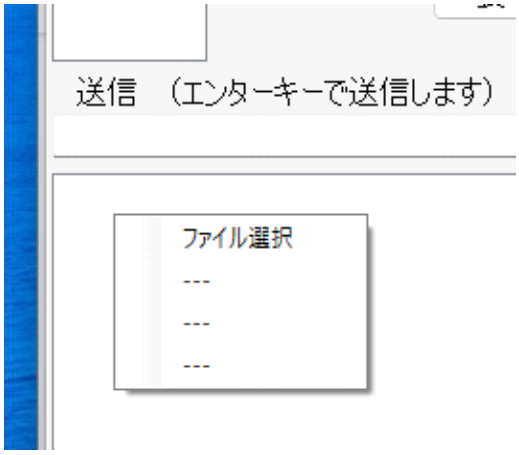
★　ここには最初日本語を入力してはいけない

```
        1 個の参照
335     private void sndlstmenu000ToolStripMenuItem_Click(object sender, EventArgs e)
336     {
337
338     }
339
        1 個の参照
340     private void sndlstmenu001ToolStripMenuItem_Click(object sender, EventArgs e)
341     {
342
343     }
344
        1 個の参照
345     private void sndlstmenu002ToolStripMenuItem_Click(object sender, EventArgs e)
346     {
347
348     }
349
        1 個の参照
350     private void sndlstmenu003ToolStripMenuItem_Click(object sender, EventArgs e)
351     {
352
353     }
```

★　メニューをダブルクリックして、上記のように、イベントを生成した後、タイトルを変更する



・イベント関数が生成されたら、タイトルは自由に変更できます



・このように、右クリックで登録したメニューが表示されます

コンボボックス

```
comboBox1.Items.Add("クリップボード");
comboBox1.Items.Add("頻度一覧");
comboBox1.SelectedIndex = int.Parse(Glo.Prm_tbl[3]);　// 前回インデックス再生

private void comboBox1_SelectedValueChanged(object sender, EventArgs e)
{
   // コンボボックス変更
   Glo.Prm_tbl[3] = comboBox1.SelectedIndex.ToString();
}

// パラメータに保存された文字から、インデックスを再生する
comboBox1.SelectedIndex = comboBox1.FindString(Glo.Prm_tbl[1]);
```

パラメータの参照渡し

```
void Method(ref int refArgument)
{
   refArgument = refArgument + 44;
}

int number = 1;
Method(ref number);
Console.WriteLine(number);
// Output: 45
```

タイマー処理

```
timer1.Start();
```

```
private void timer1_Tick(object sender, EventArgs e)
{
   timer1.Stop();       // 停止

   this.Text = "----メール解析、返信中----";

   System.Threading.Thread.Sleep(3000);

   this.Text = "自動メール解析";

   timer1.Start();      // スタート

}
```

遅延処理

```
Thread.Sleep(5000);
```

グラフィック処理

```
        private void Form4_Paint(object sender, PaintEventArgs e)
        {
            // Graphics オブジェクトを取得
            Graphics g = e.Graphics;

            // 青色, 太さ 2 のペンを定義
            Pen pen = new Pen(Color.Blue, 1);

            // (20, 20) から (200, 200) まで直線を描画
            g.DrawLine(pen, 20, 20, 200, 200);

            // 図形を描画
            e.Graphics.DrawRectangle(pen, 20, 20, 200, 200); // 長方形
            e.Graphics.DrawEllipse(pen, 20, 20, 200, 200);   // 楕円

            // フォントを定義
            Font font1 = new Font("Times New Roman", 4, FontStyle.Regular);
            Font font2 = new Font("Times New Roman", 5, FontStyle.Regular);
            Font font3 = new Font("Times New Roman", 6, FontStyle.Regular);
            Font font4 = new Font("Times New Roman", 7, FontStyle.Regular);

            // (20, 20) の位置からテキストを描画
            e.Graphics.DrawString("グラフィックテスト", font1, SystemBrushes.WindowText, 320, 20);
            e.Graphics.DrawString("グラフィックテスト", font2, SystemBrushes.WindowText, 320, 40);
            e.Graphics.DrawString("グラフィックテスト", font3, SystemBrushes.WindowText, 320, 60);
            e.Graphics.DrawString("グラフィックテスト", font4, SystemBrushes.WindowText, 320, 80);

            // ペンを破棄
            pen.Dispose();
        }
```

```
e.Graphics.DrawString("青色の文字です", font1, Brushes.Blue, gx, gy);
```

```
Color color = Color.FromArgb(R,G,B);
```

グラフィックのペンカラーをＲＧＢで指示する

```
Color pen_color;
pen_color = Color.FromArgb(150, 150, 255);
```

テキストカラー取得関数

```
public Brush get_txt_col(int prm_col_no)
{
    Brush tcol = Brushes.Black;

    switch(prm_col_no)
    {
        case 0: tcol = Brushes.Black; break;
        case 1: tcol = Brushes.Red; break;
        case 2: tcol = Brushes.Green; break;
        case 3: tcol = Brushes.Blue; break;
        case 4: tcol = Brushes.AliceBlue; break;
        case 5: tcol = Brushes.AntiqueWhite; break;
        case 6: tcol = Brushes.Aqua; break;
        case 7: tcol = Brushes.Aquamarine; break;
        case 8: tcol = Brushes.Azure; break;
        case 9: tcol = Brushes.Beige; break;
        case 10: tcol = Brushes.Bisque; break;
        case 11: tcol = Brushes.BlanchedAlmond; break;
        case 12: tcol = Brushes.BlueViolet; break;
        case 13: tcol = Brushes.Brown; break;
        case 14: tcol = Brushes.BurlyWood; break;
        case 15: tcol = Brushes.CadetBlue; break;
        case 16: tcol = Brushes.Chartreuse; break;
        case 17: tcol = Brushes.Chocolate; break;
        case 18: tcol = Brushes.Coral; break;
        case 19: tcol = Brushes.CornflowerBlue; break;
        case 20: tcol = Brushes.Cornsilk; break;
        case 21: tcol = Brushes.Crimson; break;
        case 22: tcol = Brushes.Cyan; break;
        case 23: tcol = Brushes.DarkBlue; break;
        case 24: tcol = Brushes.DarkCyan; break;
        case 25: tcol = Brushes.DarkGoldenrod; break;
        case 26: tcol = Brushes.DarkGray; break;
        case 27: tcol = Brushes.DarkGreen; break;
        case 28: tcol = Brushes.DarkKhaki; break;
        case 29: tcol = Brushes.DarkMagenta; break;
        case 30: tcol = Brushes.DarkOliveGreen; break;
        case 31: tcol = Brushes.DarkOrange; break;
        case 32: tcol = Brushes.DarkOrchid; break;
        case 33: tcol = Brushes.DarkRed; break;
        case 34: tcol = Brushes.DarkSalmon; break;
        case 35: tcol = Brushes.DarkSeaGreen; break;
        case 36: tcol = Brushes.DarkSlateBlue; break;
        case 37: tcol = Brushes.DarkSlateGray; break;
        case 38: tcol = Brushes.DarkTurquoise; break;
        case 39: tcol = Brushes.DarkViolet; break;
        case 40: tcol = Brushes.DeepPink; break;
        case 41: tcol = Brushes.DeepSkyBlue; break;
        case 42: tcol = Brushes.DimGray; break;
        case 43: tcol = Brushes.DodgerBlue; break;
        case 44: tcol = Brushes.Firebrick; break;
        case 45: tcol = Brushes.FloralWhite; break;
        case 46: tcol = Brushes.ForestGreen; break;
        case 47: tcol = Brushes.Fuchsia; break;
        case 48: tcol = Brushes.Gainsboro; break;
        case 49: tcol = Brushes.GhostWhite; break;
        case 50: tcol = Brushes.Gold; break;
        case 51: tcol = Brushes.Goldenrod; break;
        case 52: tcol = Brushes.Gray; break;
        case 53: tcol = Brushes.GreenYellow; break;
        case 54: tcol = Brushes.Honeydew; break;
```

```
case 55: tcol = Brushes.HotPink; break;
case 56: tcol = Brushes.IndianRed; break;
case 57: tcol = Brushes.Indigo; break;
case 58: tcol = Brushes.Ivory; break;
case 59: tcol = Brushes.Khaki; break;
case 60: tcol = Brushes.Lavender; break;
case 61: tcol = Brushes.LavenderBlush; break;
case 62: tcol = Brushes.LawnGreen; break;
case 63: tcol = Brushes.LemonChiffon; break;
case 64: tcol = Brushes.LightBlue; break;
case 65: tcol = Brushes.LightCoral; break;
case 66: tcol = Brushes.LightCyan; break;
case 67: tcol = Brushes.LightGoldenrodYellow; break;
case 68: tcol = Brushes.LightGray; break;
case 69: tcol = Brushes.LightGreen; break;
case 70: tcol = Brushes.LightPink; break;
case 71: tcol = Brushes.LightSalmon; break;
case 72: tcol = Brushes.LightSeaGreen; break;
case 73: tcol = Brushes.LightSkyBlue; break;
case 74: tcol = Brushes.LightSlateGray; break;
case 75: tcol = Brushes.LightSteelBlue; break;
case 76: tcol = Brushes.LightYellow; break;
case 77: tcol = Brushes.Lime; break;
case 78: tcol = Brushes.LimeGreen; break;
case 79: tcol = Brushes.Linen; break;
case 80: tcol = Brushes.Magenta; break;
case 81: tcol = Brushes.Maroon; break;
case 82: tcol = Brushes.MediumAquamarine; break;
case 83: tcol = Brushes.MediumBlue; break;
case 84: tcol = Brushes.MediumOrchid; break;
case 85: tcol = Brushes.MediumPurple; break;
case 86: tcol = Brushes.MediumSeaGreen; break;
case 87: tcol = Brushes.MediumSlateBlue; break;
case 88: tcol = Brushes.MediumSpringGreen; break;
case 89: tcol = Brushes.MediumTurquoise; break;
case 90: tcol = Brushes.MediumVioletRed; break;
case 91: tcol = Brushes.MidnightBlue; break;
case 92: tcol = Brushes.MintCream; break;
case 93: tcol = Brushes.MistyRose; break;
case 94: tcol = Brushes.Moccasin; break;
case 95: tcol = Brushes.NavajoWhite; break;
case 96: tcol = Brushes.Navy; break;
case 97: tcol = Brushes.OldLace; break;
case 98: tcol = Brushes.Olive; break;
case 99: tcol = Brushes.OliveDrab; break;
case 100: tcol = Brushes.Orange; break;
case 101: tcol = Brushes.OrangeRed; break;
case 102: tcol = Brushes.Orchid; break;
case 103: tcol = Brushes.PaleGoldenrod; break;
case 104: tcol = Brushes.PaleGreen; break;
case 105: tcol = Brushes.PaleTurquoise; break;
case 106: tcol = Brushes.PaleVioletRed; break;
case 107: tcol = Brushes.PapayaWhip; break;
case 108: tcol = Brushes.PeachPuff; break;
case 109: tcol = Brushes.Peru; break;
case 110: tcol = Brushes.Pink; break;
case 111: tcol = Brushes.Plum; break;
case 112: tcol = Brushes.PowderBlue; break;
case 113: tcol = Brushes.Purple; break;
case 114: tcol = Brushes.RosyBrown; break;
case 115: tcol = Brushes.RoyalBlue; break;
case 116: tcol = Brushes.SaddleBrown; break;
case 117: tcol = Brushes.Salmon; break;
case 118: tcol = Brushes.SandyBrown; break;
case 119: tcol = Brushes.SeaGreen; break;
```

```
                case 120: tcol = Brushes.SeaShell; break;
                case 121: tcol = Brushes.Sienna; break;
                case 122: tcol = Brushes.Silver; break;
                case 123: tcol = Brushes.SkyBlue; break;
                case 124: tcol = Brushes.SlateBlue; break;
                case 125: tcol = Brushes.SlateGray; break;
                case 126: tcol = Brushes.Snow; break;
                case 127: tcol = Brushes.SpringGreen; break;
                case 128: tcol = Brushes.SteelBlue; break;
                case 129: tcol = Brushes.Tan; break;
                case 130: tcol = Brushes.Teal; break;
                case 131: tcol = Brushes.Thistle; break;
                case 132: tcol = Brushes.Tomato; break;
                case 133: tcol = Brushes.Transparent; break;
                case 134: tcol = Brushes.Turquoise; break;
                case 135: tcol = Brushes.Violet; break;
                case 136: tcol = Brushes.Wheat; break;
                case 137: tcol = Brushes.White; break;
                case 138: tcol = Brushes.WhiteSmoke; break;
                case 139: tcol = Brushes.Yellow; break;
                case 140: tcol = Brushes.YellowGreen; break;
            }


            return tcol;
        }
```

メッセージボックスの表示と選択

```
//メッセージボックスを表示する
MessageBox.Show("正しい値を入力してください。",
   "エラー",
   MessageBoxButtons.OK,
   MessageBoxIcon.Error);

DialogResult ans_flg;
msg_cst = fil_name + ".txt  新しく生成しますか";
ans_flg = MessageBox.Show(msg_cst,"確認"
,MessageBoxButtons.OKCancel, MessageBoxIcon..Question);
if(ans_flg == DialogResult.OK)
{
      File.Copy("000_原稿.gen", fil_path);
      System.Diagnostics.Process.Start("EXPLORER.EXE", Msg_path);


}



//メッセージボックスを表示する
DialogResult result = MessageBox.Show("ファイルを上書きしますか？",
   "質問",
   MessageBoxButtons.YesNoCancel,
   MessageBoxIcon.Exclamation,
   MessageBoxDefaultButton.Button2);

//何が選択されたか調べる
if (result == DialogResult.Yes)
{
```

```
    //「はい」が選択された時
    Console.WriteLine("「はい」が選択されました");
}
else if (result == DialogResult.No)
{
    //「いいえ」が選択された時
    Console.WriteLine("「いいえ」が選択されました");
}
else if (result == DialogResult.Cancel)
{
    //「キャンセル」が選択された時
    Console.WriteLine("「キャンセル」が選択されました");
}
```

グラフィックでJPG画像を配置する

```
//画像ファイルを読み込んで、Imageオブジェクトを作成する
System.Drawing.Image img = System.Drawing.Image.FromFile(@"C:\test.jpg");

//画像を表示する
PictureBox1.Image = img;
```

グラフィックテキストの囲み座標取得

```
SizeF str_siz;
StringFormat sf = new StringFormat();
Font font1 = new Font("Times New Roman", font_siz, FontStyle.Regular);

str_siz = g.MeasureString(Flo_moj_tbl[saki_no], font1, 1000, sf); // テキストサイズ取得
point_cnt = txt_box_to_box_lin_get1(moto_p, (short)str_siz[0].Width, (short)str_siz[0].Height
    , saki_p, (short)str_siz.Width, (short)str_siz.Height,point_tbl);
```

★　これらの処理は、今のところ、Paint() 内部でのみ利用可能
Graphics g;を外部変数定義したが、使えなかった

グラフィック要素の範囲選択

・マウスのドラッグで四角い囲み範囲を指定して要素を検索する

ピクチャボックスに写真を配置する

```
private void Form4_Paint(object sender, PaintEventArgs e)
{

//System.Drawing.Image img;
//img = System.Drawing.Image.FromFile("E:¥¥ゴマ電子¥¥病院¥¥提出書類¥¥現金精算¥¥2022年¥¥P_20220207_143148.jpg");
//pictureBox1.Image = img;

}
```

グラフィック謎現象

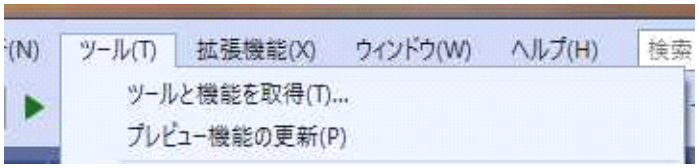
```
            //g.Dispose();          // これを実行するとペイント全体がエラーになる
```

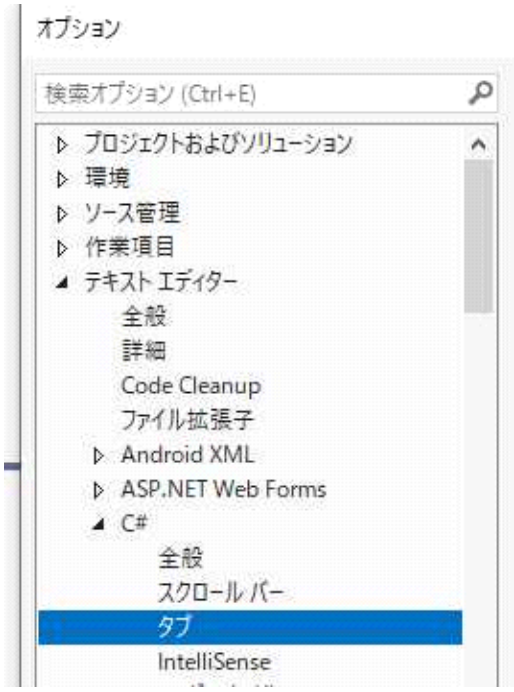
クリップボードに文字列を代入する

```
string txt_buf;
```

```
txt_buf = dt.Year.ToString("D4") + "/" + dt.Month.ToString("D2") + "/" + dt.Day.ToString("D2")
      + " (" + week_tbl[week] + "曜日) " + Prm_tbl[48] + " ";

Clipboard.SetText(txt_buf);     // クリップボードに文字列を代入
this.WindowState = FormWindowState.Minimized; // ワードなどにアクセスするため画面を最小にする
```

タブの保持がされないとものすごく不便です





ここから先は、テキストデータベースアプリ専用です

TXDB_マクロの実行

```
private void dataGridView1_DoubleClick(object sender, EventArgs e)

    case ".mactxt":
        // 下位フォーム呼び出し準備
        Form5 form5 = new Form5(fil_path, Prm_tbl[17]);
        form5.ShowDialog();          // 下位フォーム呼び出し実行
        break;

private void button1_mac_go_Click(object sender, EventArgs e)
```

シリアル通信

タイムアウトのエラー取得

2024.3.29 FRI goma0099 -2632288-

```
public Form1()
{
        InitializeComponent();

        SerialPort = new SerialPort();

        SerialPort.PortName = "COM3";
        SerialPort.BaudRate = 115200;

        SerialPort.Parity = Parity.None;
        SerialPort.DataBits = 8;
        SerialPort.StopBits = StopBits.One;
        SerialPort.ReadTimeout = 500;
        SerialPort.WriteTimeout = 500;
```

```
        // タイムアウトを設定すると例外エラーがポップアップされ使えなかったが
        // エラーを処理する方法が判明しリストボックスに表示ができた
        // そうすると、無反応時間が解消した

        SerialPort.Open();     // マイコンが接続されていないと起動しない
```

受信処理

```
if (0 < SerialPort.BytesToRead)
{
        try
        {
                red_txt = SerialPort.ReadLine();
                len = red_txt.Length;

                // 受信テキストを配列に分解する
                cha_tbl = red_txt.ToCharArray();
                len2 = cha_tbl.Length;

                //listBox1.Items.Add("受信len=" + len.ToString() + "/" + len2.ToString());

                // 表示できない文字コードを除外する　記号も除外されるか？
                msg_cst = "";
                for (i = 0; i < len2; i++)
                {
                        if (cha_tbl[i] < ' ') continue;     // 半角スペース以下は無視する
                        msg_cst += cha_tbl[i].ToString();
                }
                listBox1.Items.Add(msg_cst);   // 連続受信は成功した
                                                                // マインの送信が止まると　フリーズしてしまう
        }
        catch (Exception ex)
        {
                listBox1.Items.Add("rcv_err_msg= "+ ex.Message);
        }

}
```

送信処理

```
try
{
        SerialPort.WriteLine(red_txt);
        listBox1.Items.Add("snd_txt -> " +  textBox1.Text);

}
```

```
catch (Exception ex)
{
        listBox1.Items.Add("err_cod= "+ ex.Message);
}
```
★　この時点でマイコンへの送信は失敗している　パソコン同士では正常である

例外処理

```
try
{
        red_txt = SerialPort.ReadLine();
        //len = red_txt.Length;

        // 受信テキストを配列に分解する
        cha_tbl = red_txt.ToCharArray();
        len = cha_tbl.Length;

        //listBox1.Items.Add("受信len=" + len.ToString() + "/" + len2.ToString());

        // 表示できない文字コードを除外する　記号も除外されるか？
        msg_cst = "";
        for (i = 0; i < len; i++)
        {
                if (cha_tbl[i] < ' ') continue;    // 半角スペース以下は無視する
                msg_cst += cha_tbl[i].ToString();
        }
        listBox1.Items.Add(msg_cst);   // 連続受信は成功した
                                                        // マインコンの送信が止まると　フリーズしてしまう
                                                        // タイムアウト処理を追加したら、フリーズは解決した
}
catch (Exception ex)
{
        if (0 < ex.Message.Length) ++rcv_err_cnt;
        //listBox1.Items.Add("rcv_err_msg= "+ ex.Message);
}
```

終端です
