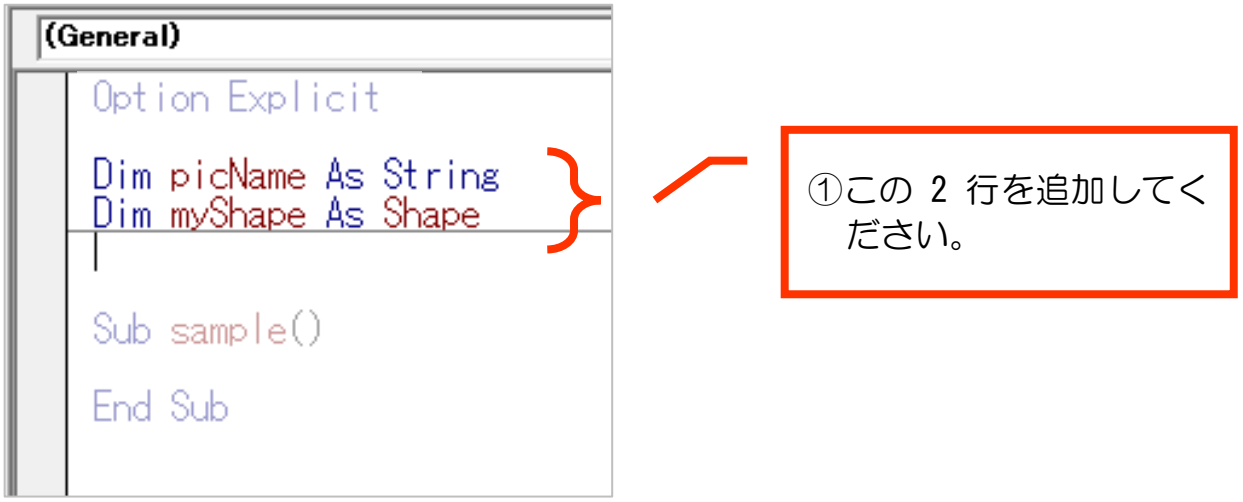
第１章　画像を表示してみよう

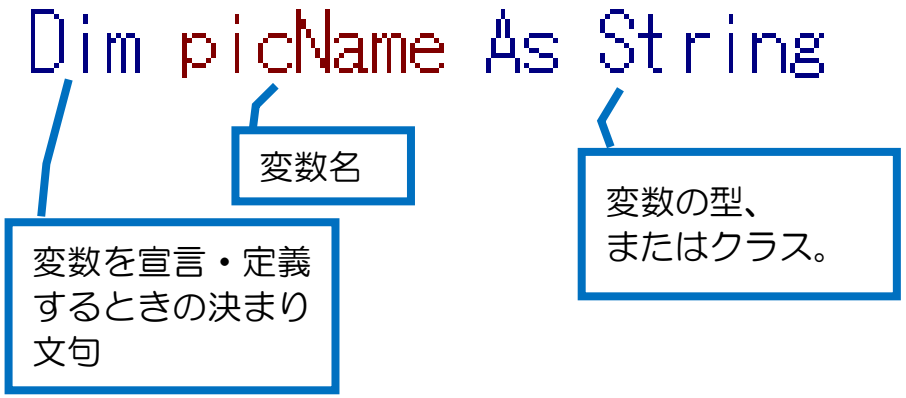
Excel で新規空白ブックを作成してください。
この講座で使用する画像ファイル５２枚入った「card」フォルダと同じフォルダ内にマクロ有効ブック形式で保存してください。
ファイル名は任意です（例では「神経衰弱」としました。）

そのブックで標準モジュールを追加し、sample という名前のプロシージャを作ってください。ここまでが前準備です。

１－１　まずは１枚表示してみよう

１－１－１　変数の宣言





「String」は「文字列」型です。
「picName という変数は、文字列が代入される変数です」と宣言・定義していることになります。
今回は、Excel シート上に表示するカードの画像の画像ファイル名として使います。

「Shape」は「Shape」クラスです。図形やイラストを扱う種類の変数です。
今回は、Excel シート上に表示するカードの画像として使います。

１－１－２　表示したい画像ファイルを指定

card フォルダ内の sa.png を指定してみましょう。

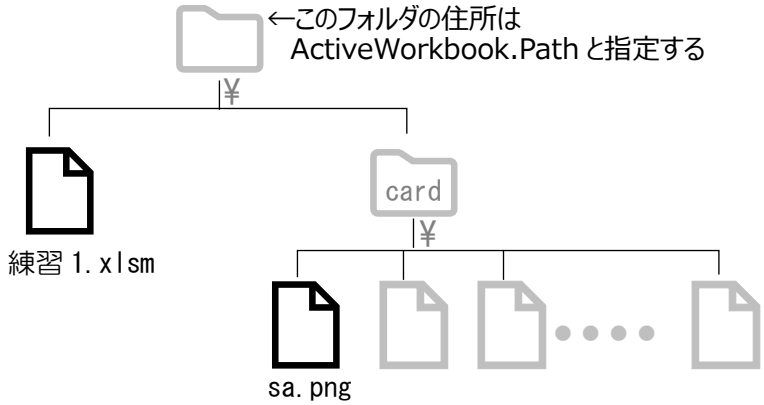
①この１行を追加してください。

```
Sub sample()

    picName = ActiveWorkbook.Path & "¥card¥sa.png"

End Sub
```

この段階では、変数 picName に表示したい画像を指定しただけです。
どこに表示するかについては、次節にて。

【参考】　画像の住所を把握する

画像を表示したい Excel ファイル「神経衰弱」の場所を基準にして、表示したい画像がパソコン内でどのような位置関係（住所）にあるか、把握する必要があります。
今のケースでは、同じフォルダ内にある card フォルダ内の sa.png ですので、



card¥sa.png

と表現します。

【参考】
「¥」は階層を区切る記号です。
「～の中の」と訳すと分かりやすいでしょう。

住所と似ています。大きなくくりから、小さなくくりへ、
〇〇県△△市▽▽区××５－１　●●様
　　¥　　¥　　¥　　¥

個人名がファイルに相当するという例えにすると分かりやすいでしょうか…？

パソコン全体での見た時の住所は、

ActiveWorkbook.Path & “¥card¥sa.png”

で表現できます。

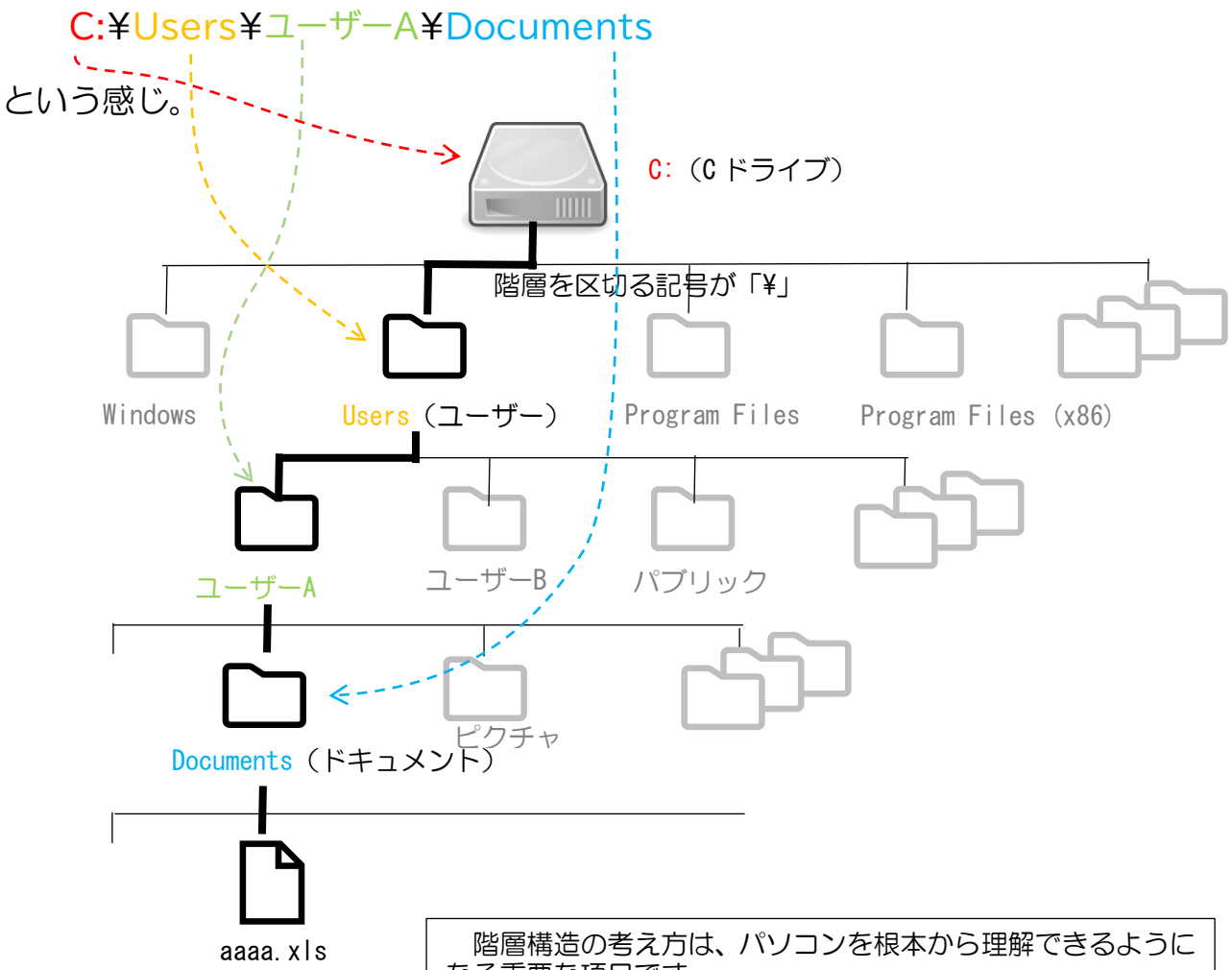
ActiveWorkbook.path はそのファイルが存在する場所を割り出してくれます。

●ActiveWorkbook.Path

　このマクロを含んだ Excel ファイルが置かれているディレクトリ（フォルダ）のパス（場所）のこと。

　プログラム内で画像ファイルを指定しやすいように、マクロを含んだ Excel ファイルと同じディレクトリ（フォルダ）に画像を置いたので、この方法で場所を指定できます。

　例えば、ユーザーA のドキュメントフォルダに「aaaa.xlsx」がある場合、そのパスは



　階層構造の考え方は、パソコンを根本から理解できるようになる重要な項目です。

　理解できると、ファイル操作（例えば写真をデジカメから取り込む、音楽ファイル・イラスト・ソフトウェアなどのダウンロード、作成した文書ファイルなどの保存、など）をスムーズに行えるようになります。

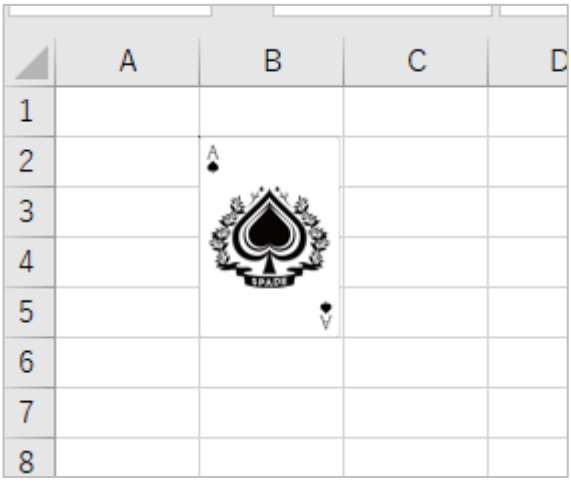
１－１－３　画像をシートに表示

いきなりヘヴィかもしれませんが、続けて次の命令を追加してください。

```
picName = ActiveWorkbook.Path & "¥card¥sa.png"
Set myShape = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
    linktofile:=False, _
    savewithdocument:=True, _
    Left:=Cells(2, 2).Left, _
    Top:=Cells(2, 2).Top, _
    Width:=0, _
    Height:=0)
myShape.ScaleHeight 0.1, msoCTrue
myShape.ScaleWidth 0.1, msoCTrue
```

「 _（スペース＋アンダーバー）」は改行を表します。
1行に書くと長くなって見づらい場合などに、次の行に折り返してもつながった行として認識されます。

打てたら、
プロシージャを実行してみましょう。
下図のように♠A が表示されれば OK です。



【解説】覚えなくていいです！
●画像をシートに配置

```
    Set myShape = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
        linktofile:=False, _
        savewithdocument:=True, _
        Left:=Cells(2, 2).Left, _
        Top:=Cells(2, 2).Top, _
        Width:=0, _
        Height:=0)
```

変数 myShape には、アクティブシート上に配置する画像オブジェクトを設定しています。

```
Set myShape = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
```

アクティブシートの

Shapes オブジェクトに

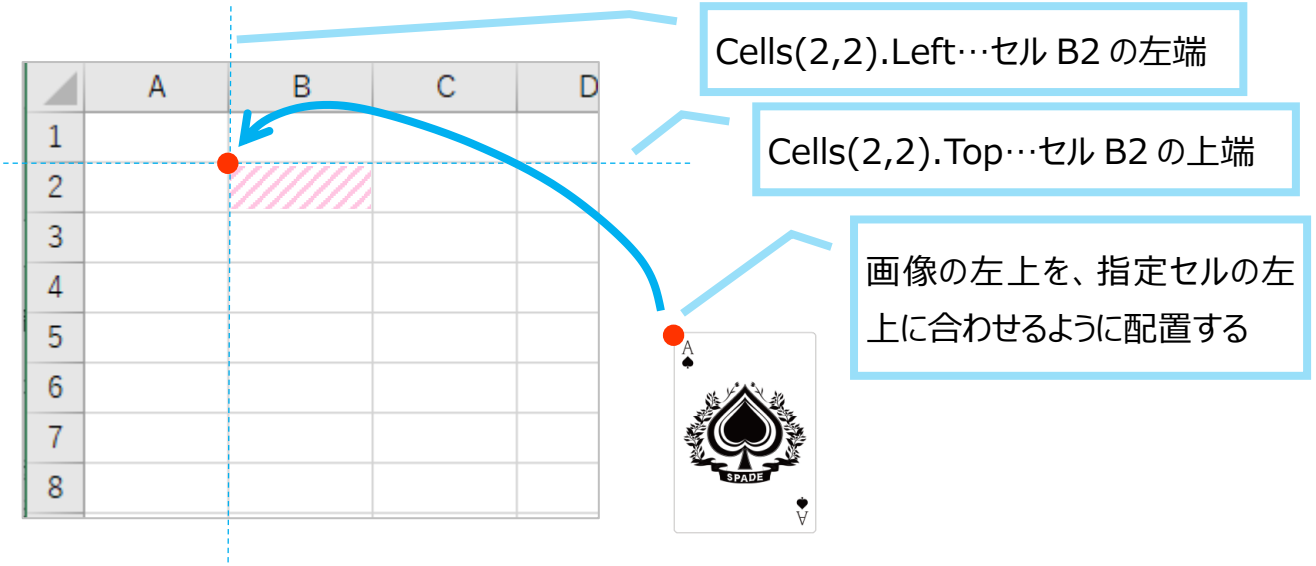
Shapes オブジェクトは、画像・イラスト・図形などを扱うものです。

画像(Picture)を追加(Add)

追加する画像のファイル名を picName で指定

●画像の表示位置を次の２つのパラメータで指定しています。

```
Left:=Cells(2, 2).Left, _
Top:=Cells(2, 2).Top, _
```



Cells(2,2).Left…セル B2 の左端

Cells(2,2).Top…セル B2 の上端

画像の左上を、指定セルの左上に合わせるように配置する

●その他のパラメータは次の表をご参照ください。
（https://tonari-it.com/excel-vba-shapes-addpicture/から引用、加筆しました）

パラメータ	指定する内容
Filename	**エクセルに追加したい画像ファイル**を指定します。
LinkToFile	True 又は False で指定します。**追加した画像ファイルにリンクを張ってエクセルを起動するたびに画像ファイルを読み込むか(True)、指定した画像をコピーしてエクセルに張り付けるか(False)を指定**します。
SaveWithDocument	True 又は False で指定します。**画像ファイルをエクセルと一緒に保存するか(True)、画像ファイルへのリンク情報だけを保存するかを指定**します。

パラメータ	指定する内容
Left	**画像ファイルの左上にあたる位置を、ポイントで指定**します。
Top	**画像ファイルの上にあたる位置を、ポイントで指定**します。
Width	**画像ファイルの幅を指定**します。（ピクセル単位）
Height	**画像ファイルの高さを指定**します。（ピクセル単位）

●画像の大きさ変換

```
        myShape.ScaleHeight 0.1, msoCTrue
        myShape.ScaleWidth 0.1, msoCTrue
```

ScaleHeight 0.1　　画像の縦の大きさを読み込んだ画像の 0.1 にする
ScaleWidth 0.1　　画像の縦の大きさを読み込んだ画像の 0.1 にする
msoCTrue　　元のサイズを基準にして図形を拡大・縮小する

　元の画像の大きさに対して、縦横 10%（0.1 倍）の大きさでエクセルシート上に表示する設定です。

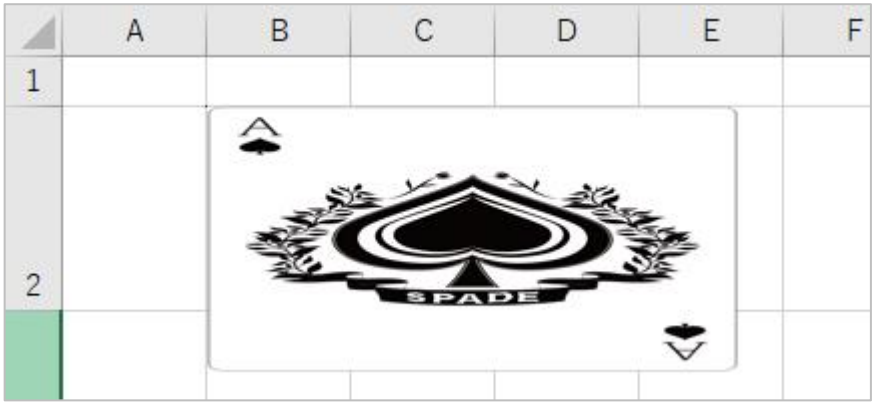
【参考】
　AddPicture メソッドのパラメータ Width と Height は、貼り付ける画像の幅と高さをピクセル単位で指定できます。

　ですが、ピクセル単位だと扱いづらい側面もあるため、Width と Height をダミーでそれぞれ0に、そのあとに ScaleHeight と ScaleWidth メソッドで元画像に対する比率で幅と高さを変更しました。
　ですので、Width と Height はどんな値でも見た目には反映されません。

　興味のある方は、ScaleHeight と ScaleHeight の行をコメントアウトして、Width と Height を適当な数字に指定して実行してみましょう。
　（コメントアウトに関しては、次節１－２をご参照ください。）

```
            Top:=Cells(2, 2).Top, _
            Width:=200, _
            Height:=100)
'       myShape.ScaleHeight 0.1, msoCTrue
'       myShape.ScaleWidth 0.1, msoCTrue
```

　Width:=200, Height:=100 にすると、幅 200 ピクセル、高さ 100 ピクセルで下図のようになります。



１－２　コメントアウト

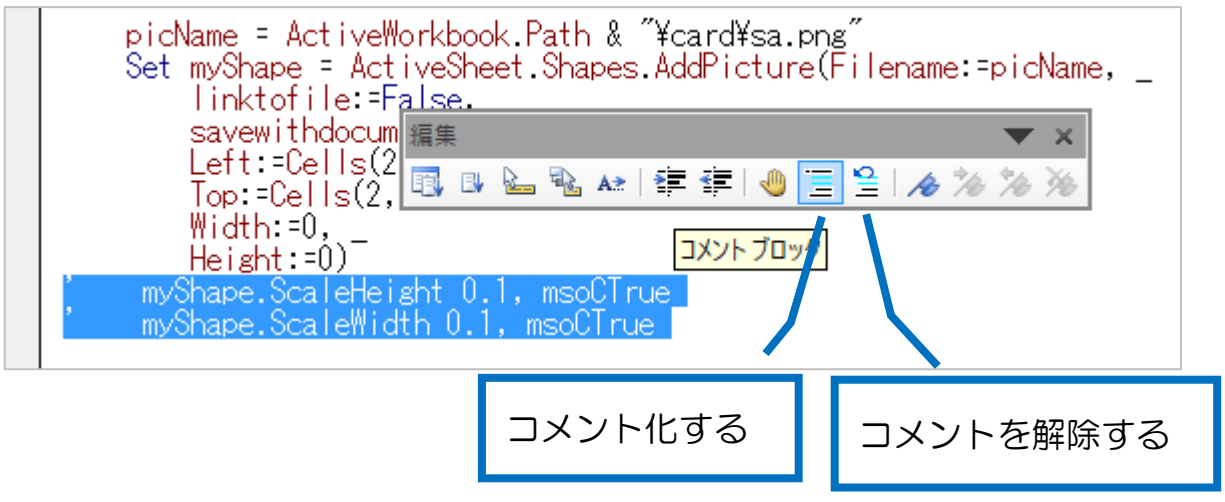
１－２－１　コメントアウト

すでに書いた命令を、コメント化して働かなくさせることができます。

```
            Top:=Cells(2, 2).Top, _
            Width:=200, _
            Height:=100)
'     myShape.ScaleHeight 0.1, msoCTrue
'     myShape.ScaleWidth 0.1, msoCTrue
```

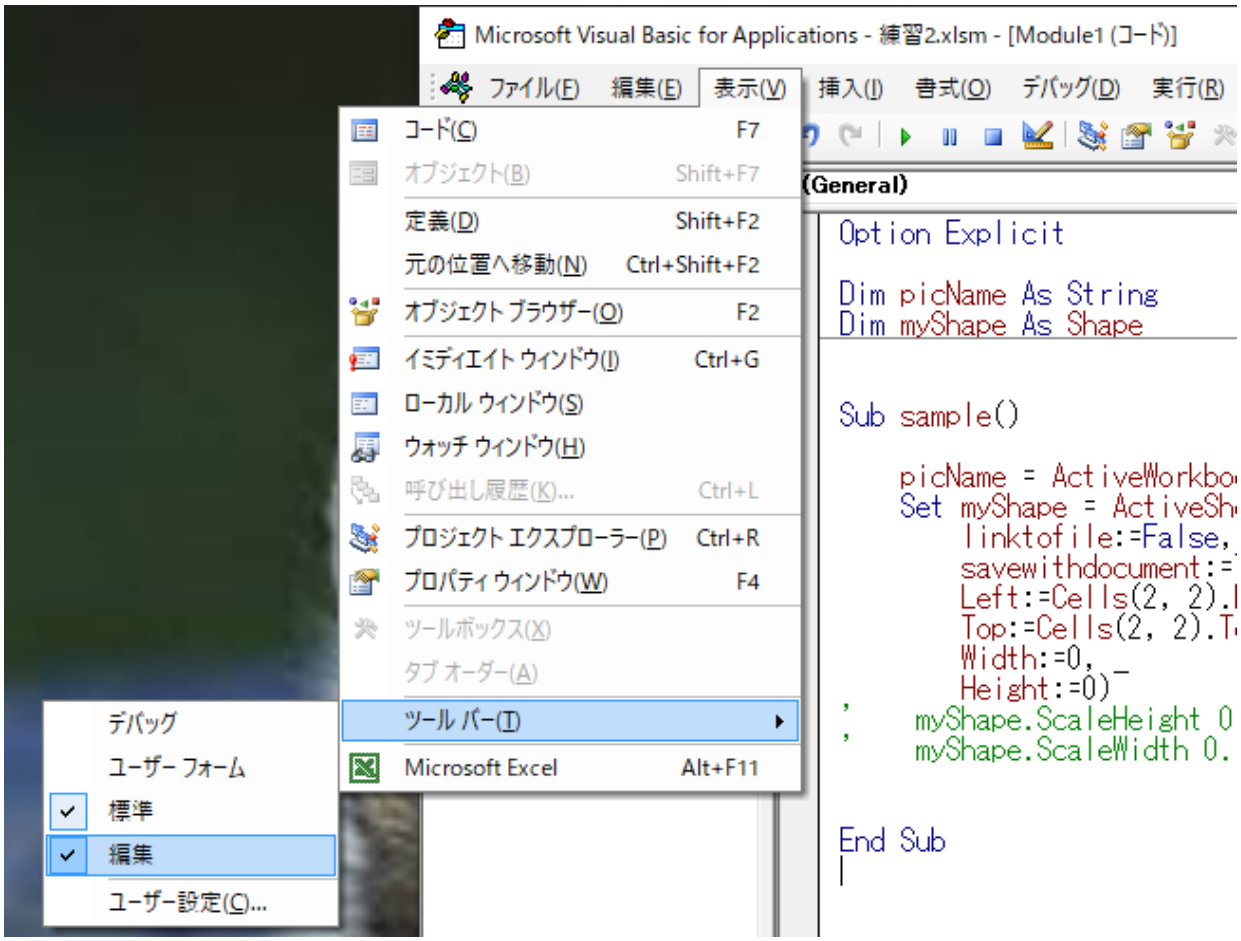
行の先頭に「’」（シングルクォーテーション）を置くと、その行がコメントになります。

手入力で置いてもいいですし、編集ツールバーを使うと複数行の場合は楽です。（編集ツールバーがない場合は、次節１－２－２をご参照ください。）



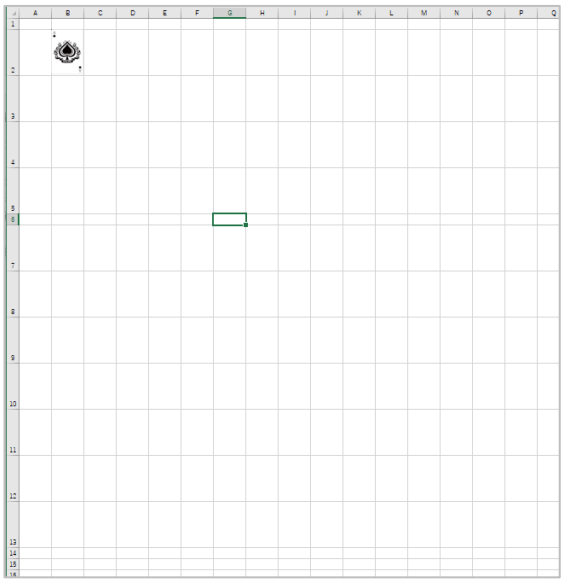
１－２－２　編集ツールバー

　編集ツールバーの表示・非表示の設定は、「表示」→「ツールバー」→「編集」と辿ってください。



１－３　画像を複数表示するときのために、行の高さをあらかじめ設定しておきましょう。（手動）

　２～５行目、7～13 行目の行の高さを 76.5（204 ピクセル）にしてください。VBA ではなく、手動で変更してください。



　カードの画像 1 枚がセル 1 個にほぼ収まる大きさです。

第２章　画像を複数表示してみよう

２－１　配列の概念

トランプ 52 枚（マーク 4 種、A～K の 13 枚）の画像を表示するためには、前章の方法を 52 回繰り返す必要があります。

ですが、これ

```
picName = ActiveWorkbook.Path & "¥card¥sa.png"
Set myShape = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
    linktofile:=False, _
    savewithdocument:=True, _
    Left:=Cells(2, 2).Left, _
    Top:=Cells(2, 2).Top, _
    Width:=0, _
    Height:=0)
myShape.ScaleHeight 0.1, msoCTrue
myShape.ScaleWidth 0.1, msoCTrue
```

を 52 個書くのはあまりにも冗長です。

そこで、

「配列」と「繰り返し」を使って短くまとめます。

まずは準備として、「配列」について考え方を整理しましょう。
練習用に、新規空白ブックを作成し、hairetu プロシージャを作ってください。

```
Option Explicit

Sub hairetu()

End Sub
```

配列を宣言・定義します。変数と同様です。ただし配列の場合、()をつけます。
ここでは、list()を Variant 型で宣言してください。

```
Sub hairetu()
    Dim list() As Variant

End Sub
```

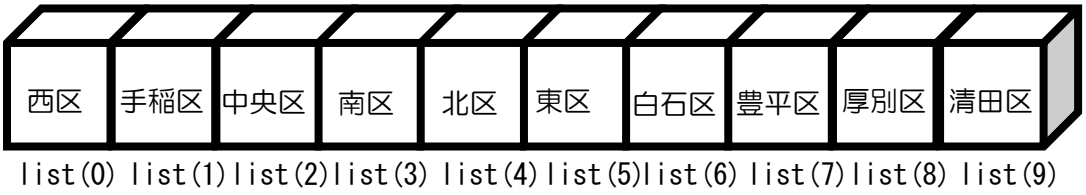
※Variant 型は整数や文字列などいろいろな型の要素を入れられる便利な配列だと思っておいてください。

配列 list()に次のように値を代入し、初期化してください。
（Variant 型の場合、Array()を使用して初期化できます。）

```
Sub hairetu()
    Dim list() As Variant

    list = Array("西区", "手稲区", "中央区", "南区", _
                 "北区", "東区", "白石区", "豊平区", _
                 "厚別区", "清田区")

End Sub
```

長くなるので、見やすさのため 2 回改行しました。
改行は「スペース＋アンダーバー（いずれも半角）」です。
改行せずに 1 行に打ち続けても問題ありません。

配列は、変数の塊です。ワンルームのアパートのように解釈してください。
list は、要素が 10 個ある配列として初期化されました。（10 部屋あるアパート）



各部屋番号は、0から始まり９で終わります。この番号をインデックスとよびます。インデックスは先頭が0、順々に１ずつ増えていきます。
アパート各部屋にそれぞれ住人が入っているように、配列には各要素にそれぞれ値が入っています。

配列の各要素を個別に扱うには、添え字でインデックス番号を指定します。

では、list()の先頭要素を Excel で表示してみましょう。
セル A2 に list の先頭要素（index が 0）を表示するには、次のように書きます。

```
    list = Array("西区", "手稲区", "中央区", "南区", _
                 "北区", "東区", "白石区", "豊平区", _
                 "厚別区", "清田区")

    Cells(2, 1).Value = list(0)
```

	A	B
1		
2	西区	
3		
4		

実行結果が左図のようになることを確認しましょう。

【練習２－１】
セル A3～A11 に、右図のような実行結果が得られるように list の要素すべてを表示してください。

	A	B
1		
2	西区	
3	手稲区	
4	中央区	
5	南区	
6	北区	
7	東区	
8	白石区	
9	豊平区	
10	厚別区	
11	清田区	
12		

２－２　繰り返し（For ループ）

10 回も同じような文を書くのは面倒くさかったことと思います。
各文で、異なるのは行数と配列インデックスだけです。

```
Cells(2, 1).Value = list(0)
Cells(3, 1).Value = list(1)
Cells(4, 1).Value = list(2)
Cells(5, 1).Value = list(3)
Cells(6, 1).Value = list(4)
Cells(7, 1).Value = list(5)
Cells(8, 1).Value = list(6)
Cells(9, 1).Value = list(7)
Cells(10, 1).Value = list(8)
Cells(11, 1).Value = list(9)
```

インデックス…0, 1, 2, 3, 4, 5, 6, 7, 8, 9　　…… i
行数　　…2, 3, 4, 5, 6, 7, 8, 9, 10, 11 …… i + 2

変数 i を使って表現すると、

```
    Cells(i + 2, 1).Value = list(i)
```

となります。

練習２－１で書いた命令を、For ループに書き換えていきましょう。

まず、変数 i を使いますので、宣言します。

```
Sub hairetu()
    Dim list() As Variant
    Dim i As Integer
```

インテジャー
Integerは整数型です。

先ほどの 10 行を、次の 3 行に書き換えてください。

```
    For i = 0 To 9
        Cells(i + 2, 1).Value = list(i)
    Next
```

実行して同じ結果が得られることを確認しましょう。

	A	B
1		
2	西区	
3	手稲区	
4	中央区	
5	南区	
6	北区	
7	東区	
8	白石区	
9	豊平区	
10	厚別区	
11	清田区	
12		

２－３　配列をシャッフルする

２－３－１　変数宣言

二つの変数を追加してください。シャッフルの作業に必要です。

```
Sub hairetu()
    Dim list() As Variant
    Dim i As Integer
    Dim rn As Integer
    Dim tmp
```

r（アール）と n（エヌ）です。

tmp は型指定しなくてかまいません。

先ほどのリスト表示のあとに、もう一つ For ループを作ってください。

```
    For i = 0 To 9
        Cells(i + 2, 1).Value = list(i)
    Next

    For i = 0 To UBound(list)

    Next
```

【解説】UBound

For で変化させる i の最終値を UBound(list)としています。

UBound(配列名)という関数で、指定した配列名の最終インデックス番号を返してくれます。
ですので UBound(list)は、配列 list()の最終インデックス番号、ここでは９になります。

※For i = 0 To 9 でもいいのですが、後ほど関数化して他のプログラムでも使いまわしするために汎用性を持たせた形で書いておきます。

２－３－２　ランダムな数字を作り出す

次のように追記してください。

```
    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)

        MsgBox rn
    Next
```

①この４行を追加

【解説】
処理に必要なのは最初の 2 行だけです。

● Randomize

次に出てくる「Rnd」を使う前に、ランダムな数値を生み出すタネを初期化します。こうしない場合、「Rnd」を使っても再現性のあるランダム値ばかり出てくる場合があります。

● rn = Int(UBound(list) * Rnd)

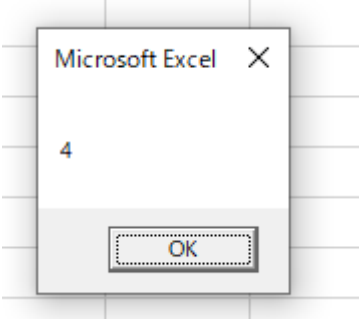
変数 rn にランダムな数字を代入します。
UBound(list)は前節で説明したように、ここでは９になります。
Rnd は、0 以上 1 未満の範囲の Single 型（小数）の乱数です。
Int()は、()内の値の小数部分を切り捨てて整数値にします。

ですので、０～８のランダムな整数値が rn に代入されます。

● MsgBox rn

rn の値をダイヤログボックスに表示します。
rn の値が実際にどうなったかを確認するためだけに書きました。

実行してみましょう。



右図のように値が表示されます。
OK を押すと、次の値が表示されます。
0～9 の For ループ内なので、10 回値がでるまで繰り返されます。
実行するたびに違う値がランダムに表示されることを確認してください。

【練習２－２】
MsgBox rn を削除して、代わりにセル B２から B11 に rn の値が表示されるように書き換えてください。

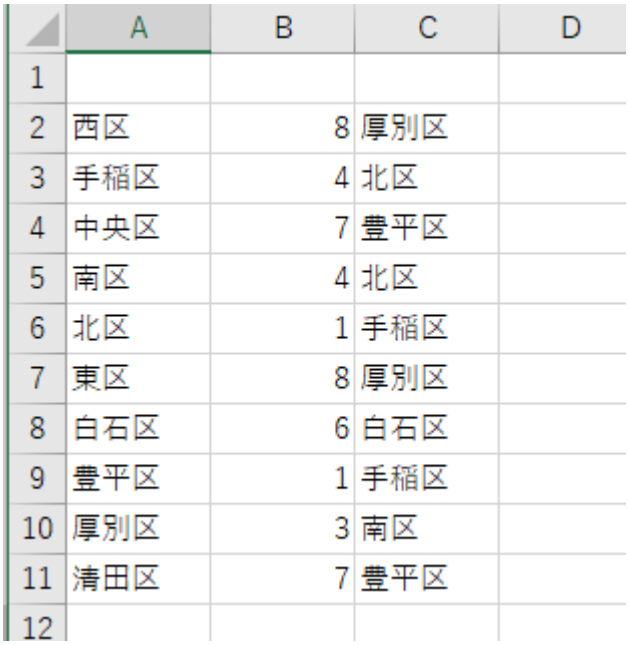
実行結果は右図のようになります。
なお、値はランダムですので、実行するたびに異なります。

	A	B	C
1			
2	西区	0	
3	手稲区	0	
4	中央区	0	
5	南区	3	
6	北区	6	
7	東区	7	
8	白石区	7	
9	豊平区	8	
10	厚別区	7	
11	清田区	3	

（ヒント１）
MsgBox の代わりに Cells を使って、
i = 0 のときは B2（2 行目 2 列目）に表示
i = 1 のときは B3（3 行目 2 列目）に表示
i = 2 のときは B4（4 行目 2 列目）に表示
……　　↑
　　i + 2 行目ですね

【練習２－３】
配列 list の要素をランダムな数値 rn をインデックスに使ってセル C２から C11 に表示する命令を追加してください。

実行結果は右図のようになります。
なお、値はランダムですので、実行するたびに異なります。



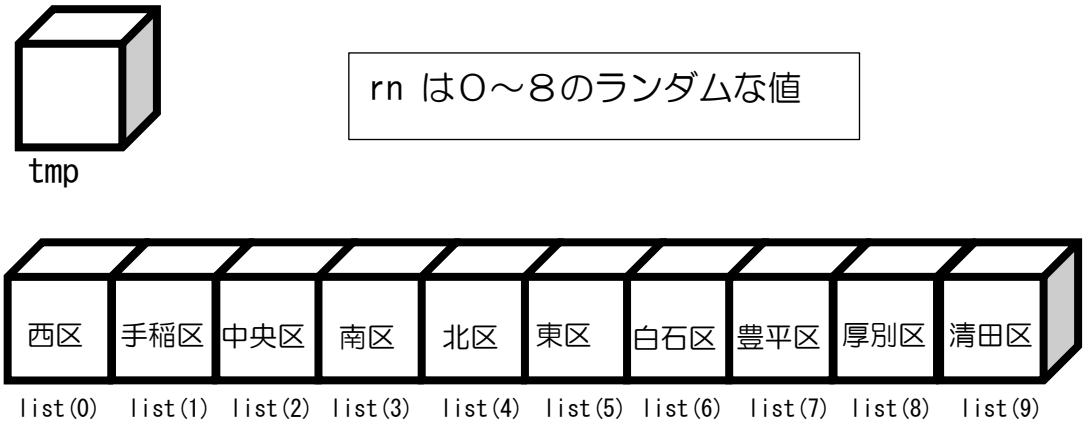
	A	B	C	D
1				
2	西区	8	厚別区	
3	手稲区	4	北区	
4	中央区	7	豊平区	
5	南区	4	北区	
6	北区	1	手稲区	
7	東区	8	厚別区	
8	白石区	6	白石区	
9	豊平区	1	手稲区	
10	厚別区	3	南区	
11	清田区	7	豊平区	
12				

B 列にインデックス番号、C 列にそのインデックスの要素が表示されます。

重複しているものがあっても、ここでは気にしません。
次節で配列内を入れ替える作業をします。

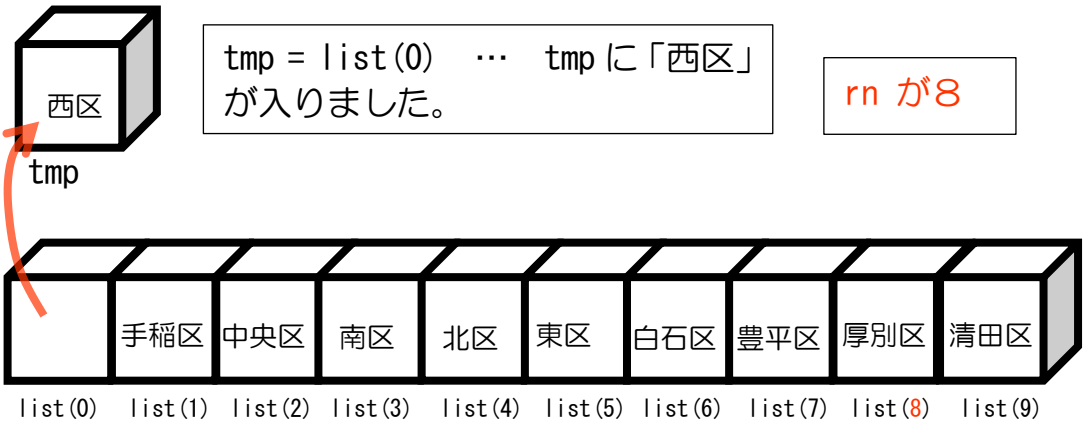
２－３－３　入れ替えの技

tmp という空の変数を用意しています。



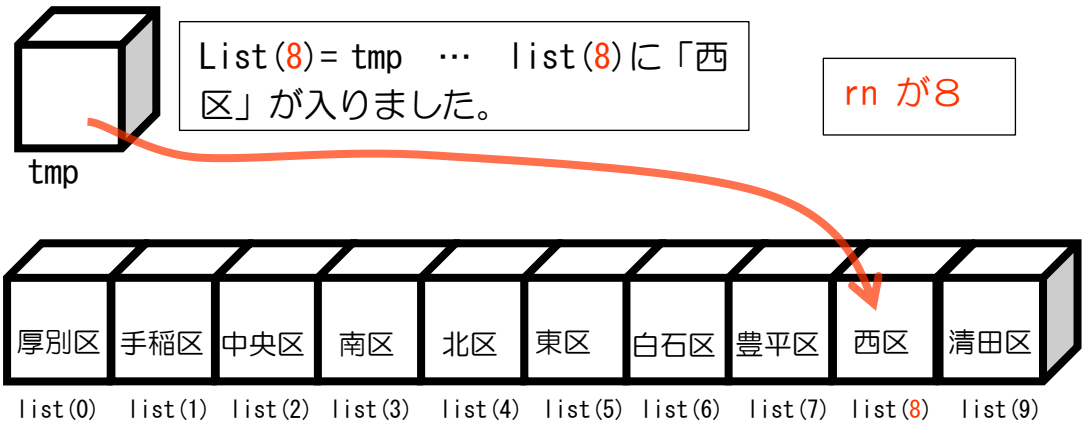
i = 0 のとき　　rn が８になったとします。

① tmp に list(0)の値を一時的に退避させます。



② list(0)に list(8)の値を入れます。

③ list(8)に tmp の値を入れます。
④



この流れを i を 1 つずつ変化させて i = 9 まで繰り返します。

例えば、次の rn の値がまた8になったとしても、list(1)と list(8)を入れ替えるので list(1)が西区、list(8)が手稲区になります。

以上の手順を、次のように追記してください。

```
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
```

①この3行を追加

結果確認のための命令を、次のように用意してから実行してみましょう。

```
    For i = 0 To 9
        Cells(i + 2, 1).Value = list(i)
    Next

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

    For i = 0 To 9
        Cells(i + 2, 3).Value = list(i)
    Next
```

②これをコピーして、

③ループ処理後に貼り付け

④3 列目に書き換え

⑤ここにあった表示用 2 行は削除

右図の実行結果例のように、C 列に入れ替え後のリストが表示されます。
A 列の元のリストと比較して、ランダムに入れ替えられていることを確認してください。

	A	B	C
1			
2	西区		手稲区
3	手稲区		中央区
4	中央区		厚別区
5	南区		豊平区
6	北区		西区
7	東区		北区
8	白石区		南区
9	豊平区		東区
10	厚別区		清田区
11	清田区		白石区
12			

２－３－４　関数化

汎用性を持たせるため、シャッフルする仕組みを関数化しましょう。
あとでコピーして、カードを並べ替えるときにも使えます。

関数は、本体のプロシージャの外に作ります。
次の図のように、まず関数の枠組みを作ってください。

```
Option Explicit

Sub hairetu()
    Dim list() As Variant
    Dim i As Integer
    Dim rn As Integer
    Dim tmp

    list = Array("西区", "手稲区", "中央区", "南区", _
                 "北区", "東区", "白石区", "豊平区")

（中略）

        Cells(i + 2, 3).Value = list(i)
    Next


End Sub

Function Shuffle(list)

End Function
```

hairetu プロシージャ

関数の枠組み
Function 関数名（引数）
～
End Function

この中に、シャッフルする仕組みをコピーしましょう。

```
    For i = 0 To 9
        Cells(i + 2, 1).Value = list(i)
    Next

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

    For i = 0 To 9
        Cells(i + 2, 3).Value = list(i)
    Next

End Sub

Function Shuffle(list)

End Function
```

この部分がシャッフルしている仕組みですから、これを関数の中にコピーします。

次の段階で、もともとのここは書き換えます。

```
    For i = 0 To 9
        Cells(i + 2, 1).Value = list(i)
    Next

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

    For i = 0 To 9
        Cells(i + 2, 3).Value = list(i)
    Next

End Sub

Function Shuffle(list)

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

End Function
```

関数の中は本体のプロシージャとは別世界ですので、関数内で使う**変数を宣言しておく必要**があります。

```
Function Shuffle(list)
    Dim i As Integer
    Dim rn As Integer
    Dim tmp

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

End Function
```

このように、i, rn, tmp を宣言・定義してください。

```
Function Shuffle(list)
    Dim i As Integer
    Dim rn As Integer
    Dim tmp

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

    Shuffle = list

End Function
```

関数の最後で、シャッフルされた list を呼び出し元に戻す命令を書いてください。

```
Sub hairetu()
    Dim list() As Variant
    Dim i As Integer
    Dim rn As Integer
    Dim tmp

    list = Array("西区", "手稲区", "中央区", "南区", _
                 "北区", "東区", "白石区", "豊平区", _
                 "厚別区", "清田区")

    For i = 0 To 9
        Cells(i + 2, 1).Value = list(i)
    Next

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

    For i = 0 To 9
        Cells(i + 2, 3).Value = list(i)
    Next
```

本体 hairetu プロシージャの中では rn, tmp は使わなくなりましたので、定義を削除します。

関数化した部分を、関数を呼び出す形にします。次の図を参照ください。

```
Sub hairetu()
    Dim list() As Variant
    Dim i As Integer

    list = Array("西区", "手稲区", "中央区", "南区", _
                 "北区", "東区", "白石区", "豊平区", _
                 "厚別区", "清田区")

    For i = 0 To 9
        Cells(i + 2, 1).Value = list(i)
    Next

    list = Shuffle(list)

    For i = 0 To 9
        Cells(i + 2, 3).Value = list(i)
    Next


End Sub
```

定義を削除した

関数を呼び出す形にした。

【解説】

① 関数 Shuffle に、シャッフルしたい配列 list を渡します。
（補足動画　https://drmpls.com/shinkei/）

```
list = Shuffle(list)
```

② 配列を受け取った Shuffle 関数は、受け取った配列をシャッフルする仕事をします。

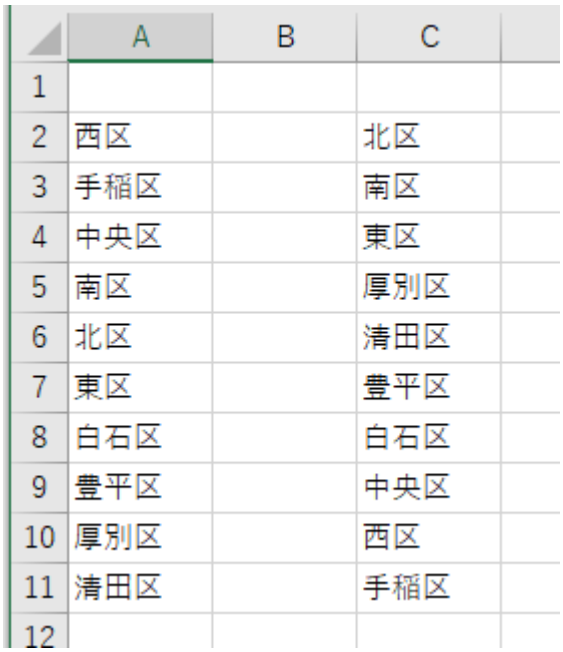
```
Function Shuffle(list)
    Dim i As Integer
    Dim rn As Integer
    Dim tmp

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

    Shuffle = list

End Function
```

③シャッフルし終わった配列を、関数の仕事の結果として出力します。

④結果の出力先は、呼び出し元です。

⑤元の配列に、シャッフルされた配列が代入されます。（上書き）

ここまでできたら、hairetu プロシージャを実行してみましょう。



	A	B	C
1			
2	西区		北区
3	手稲区		南区
4	中央区		東区
5	南区		厚別区
6	北区		清田区
7	東区		豊平区
8	白石区		白石区
9	豊平区		中央区
10	厚別区		西区
11	清田区		手稲区
12			

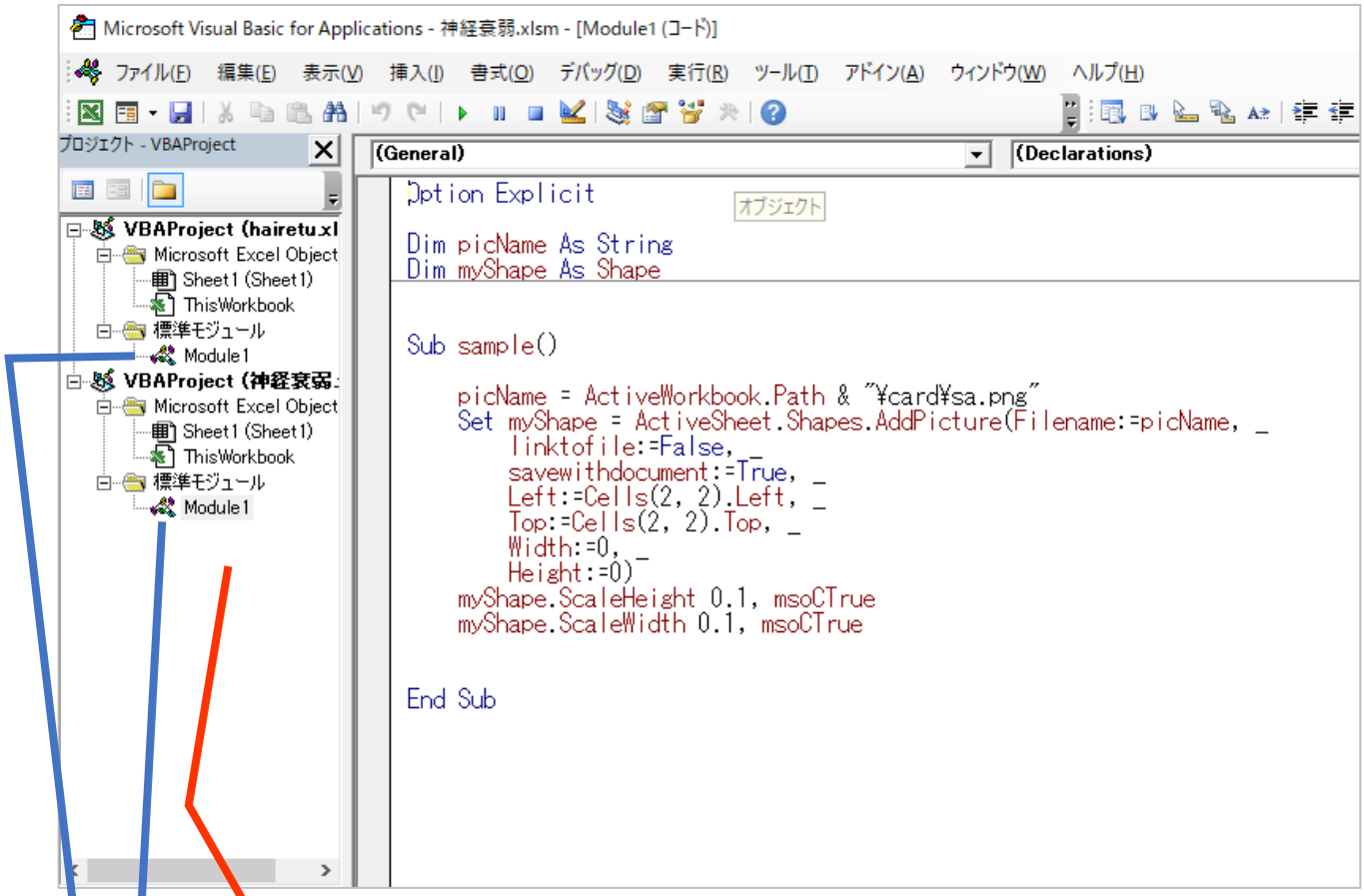
C 列にシャッフルされた結果が表示されれば OK です。

関数化する前と同じ結果（並びはランダムなので異なります）となることがわかります。

２－４　読み込む複数の画像名を配列にする

いよいよ、神経衰弱.xlsm 本体の方で複数画像を表示する手順に入ります。
今開いている hairetu.xlsm は開いたまま、神経衰弱.xlsm を開いてください。

VBE（Visual Basic Editor）では、下図のように表示されます。



hairetu.xlsm、神経衰弱.xlsm の両方が見えます。
操作（表示）したいものをダブルクリックして切り替えられます。

神経衰弱.xlms のプロシージャはこれ

hairetu.xlms のプロシージャはこれ

画像名を格納するための配列 list()を、プロシージャの外側に宣言・定義しましょう。

プロシージャの外側で定義するとグローバル変数として定義されますので、モジュール全体で使えます。（グローバル変数については「【参考】変数の有効範囲　グローバル変数・ローカル変数」（次ページ）をご参照ください。）

```
Option Explicit

Dim picName As String
Dim myShape As Shape
Dim list() As Variant    'カードの名前リスト

Sub sample()

    picName = ActiveWorkbook.Path & "¥card¥sa.png"
    Set myShape = ActiveSheet.Shapes.AddPicture(Filenam
```

次に、プロシージャ内で list()に画像名を格納しましょう。

【練習２－４】
配列 list()に、52 枚の画像ファイル名を格納してください。
別の言い方をすると、配列 list()を 52 枚の画像ファイル名で初期化してください。
（「２－１　配列の概念」を参考にしてみましょう。）

【参考】変数の有効範囲　グローバル変数・ローカル変数

●ローカル変数…プロシージャや関数の中で宣言・定義された変数です。宣言しているプロシージャ内・関数内のみで使えます。別のプロシージャや関数には影響を与えません。

```
Option Explicit

Sub hairetu()
    Dim list() As Variant
    Dim i As Integer

    list = Array("西区", "手稲区", "中央区", "南区", _
                 "北区", "東区", "白石区", "豊平区", _
                 "厚別区", "清田区")

    For i = 0 To 9
        Cells(i + 2, 1).Value = list(i)
    Next

    list = Shuffle(list)

    For i = 0 To 9
        Cells(i + 2, 3).Value = list(i)
    Next

End Sub

Function Shuffle(list)
    Dim i As Integer
    Dim rn As Integer
    Dim tmp

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

    Shuffle = list

End Function
```

hairetu プロシージャ内で宣言された変数は、このプロシージャ内でのみ有効。

Shuffle 関数内で宣言された変数は、この関数内のみで有効

hairetu プロシージャ内と Shuffle 関数内でそれぞれ i が宣言されているとします。この場合、同じ変数名ですが、それぞれの i はそれぞれプロシージャ・関数内のみで有効な変数のため、まったくの別物です。

ですので、hairetu プロシージャ内で i = 0 にしたとしても、Shuffle 関数内での i の値は0にはなりません。

●グローバル変数…モジュール内全体で使える変数です。

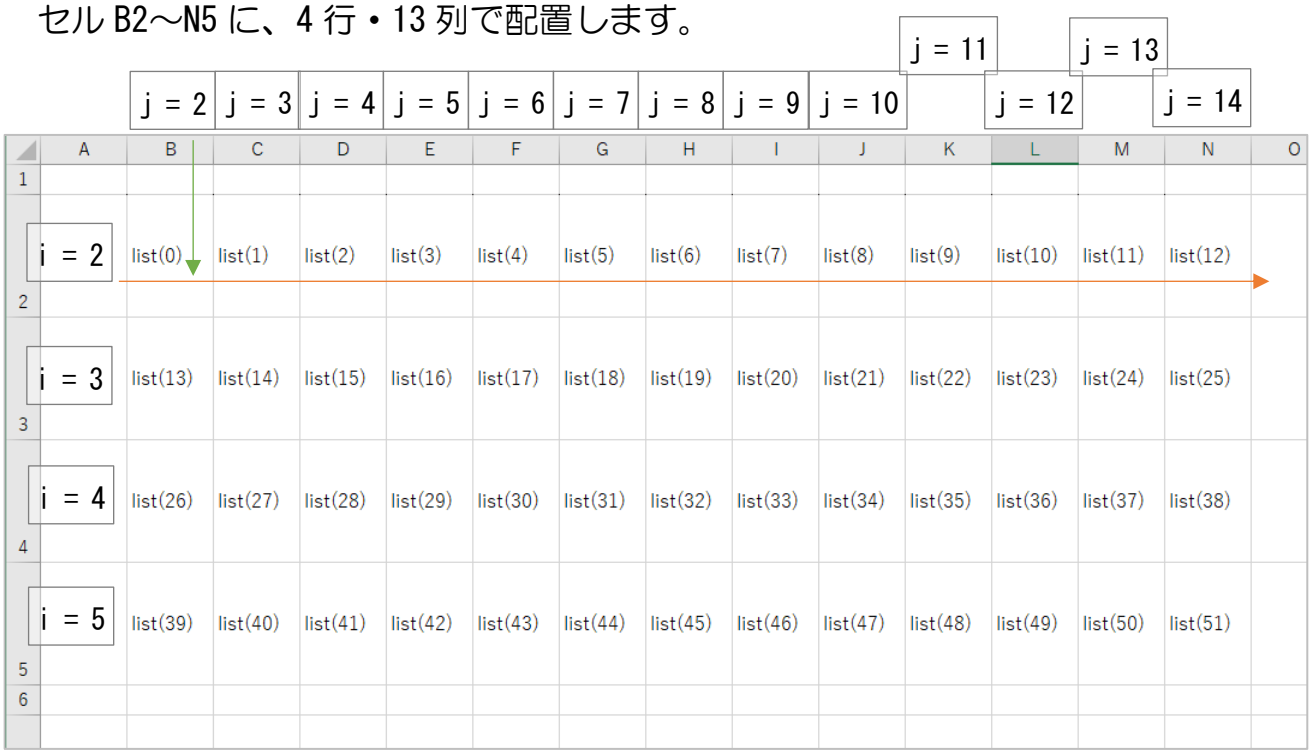
```
Option Explicit

Dim picName As String
Dim myShape As Shape
Dim list() As Variant   'カードの名前リスト

Sub sample()

    picName = ActiveWorkbook.Path & "¥card¥sa.png"
    Set myShape = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
        linktofile:=False, _
        savewithdocument:=True, _
        Left:=Cells(2, 2).Left, _
        Top:=Cells(2, 2).Top, _
        Width:=0, _
        Height:=0)
    myShape.ScaleHeight 0.1, msoCTrue
    myShape.ScaleWidth 0.1, msoCTrue


End Sub
```

モジュールの先頭で宣言された変数はグローバル変数。

モジュール内にあるプロシージャ内でも関数内でも共通に使えます。

２－５　複数の画像を表示する

あとは、52 回（52 枚分）画像表示の処理を繰り返します。
セル B2～N5 に、4 行・13 列で配置します。



■考え方

- 変数 i，j，k を用意します。
- k を 0～51 まで変化させることで、配列 list の要素をそれぞれ指定できるようにします。
- 行を i で表現し、シートの 2 行目～5 行目なので i を２～５でループさせます。
- 列を j で表現し、シートの 2 列目（B 列）から 14 行目（N 列）なので i のループ内で j を２～１４でループさせます。

①i，j，k を整数としてローカル変数で宣言しましょう。

```
Sub sample()
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer

    list = Array("SA.png", "S2.png", "S3
           "HA.png", "H2.png", "H3.png"
           "DA.png", "D2.png", "D3.png"
           "CA.png", "C2.png", "C3.png"


    picName = ActiveWorkbook.Path & "¥ca
```

② k に0をセットしましょう。

```
    list = Array("SA.png", "S2.png", "S3.png", "S4.pr
           "HA.png", "H2.png", "H3.png", "H4.png", "
           "DA.png", "D2.png", "D3.png", "D4.png", "
           "CA.png", "C2.png", "C3.png", "C4.png",

    k = 0

    picName = ActiveWorkbook.Path & "¥card¥sa.png"
    Set myShape = ActiveSheet.Shapes.AddPicture(Filer
        linktofile:=False, _
        savewithdocument:=True, _
        Left:=Cells(2, 2).Left, _
        Top:=Cells(2, 2).Top, _
        Width:=0, _
        Height:=0)
    myShape.ScaleHeight 0.1, msoCTrue
```

③ 画像表示の部分を挟みこむように i のループを作りましょう。

```
    k = 0
    For i = 2 To 5

        picName = ActiveWorkbook.Path &
        Set myShape = ActiveSheet.Shapes.AddPic
            linktofile:=False, _
            savewithdocument:=True, _
            Left:=Cells(2, 2).Left, _
            Top:=Cells(2, 2).Top, _
            Width:=0, _
            Height:=0)
        myShape.ScaleHeight 0.1, msoCTrue
        myShape.ScaleWidth 0.1, msoCTrue

    Next i
```

画像表示の部分

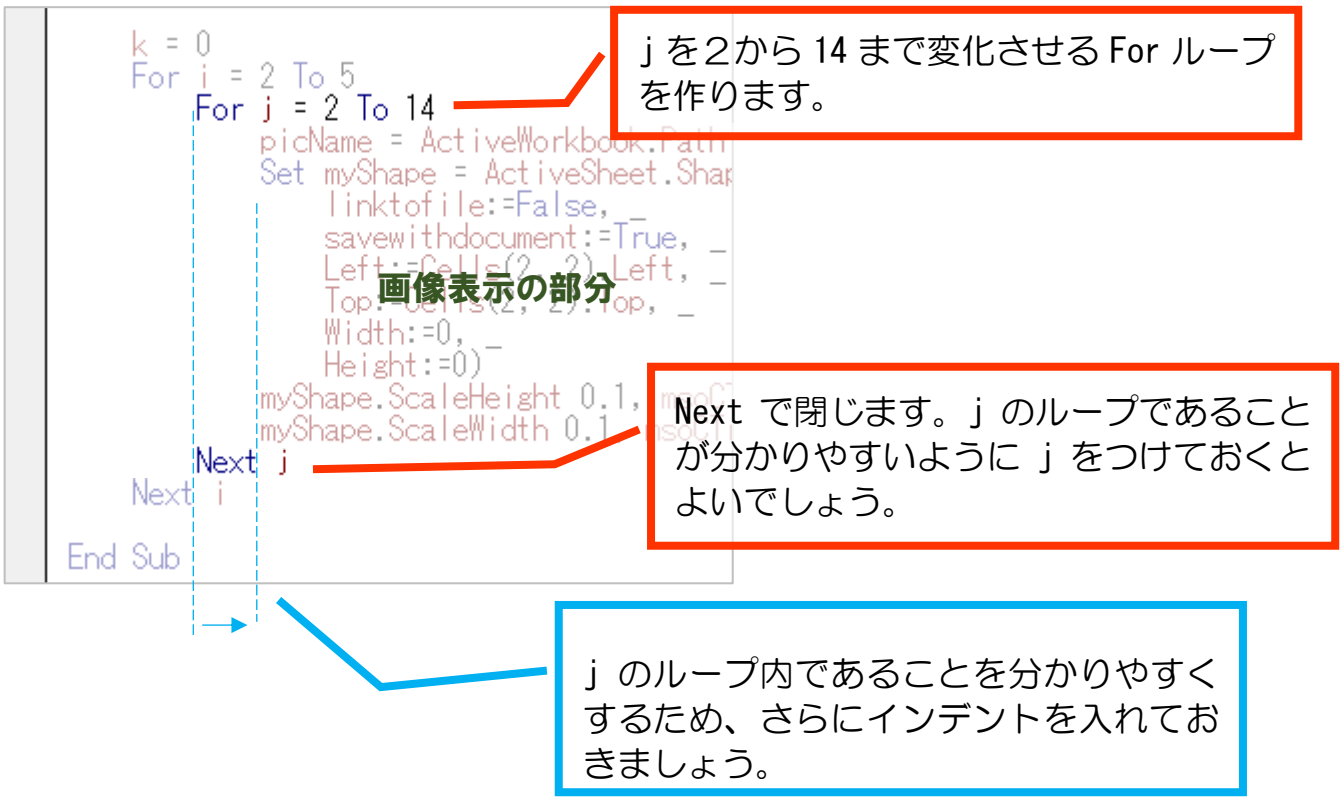
i を２から５まで変化させる For ループを作ります。

Next で閉じます。i のループであることが分かりやすいように i をつけておくとよいでしょう。

画像表示の部分は、For ループの内部であることを分かりやすくするため、インデントを入れておいてください。

※インデントとは、この場面では全体的に右にずらすことです。

④ さらに画像表示の部分を挟みこむように j のループを作りましょう。



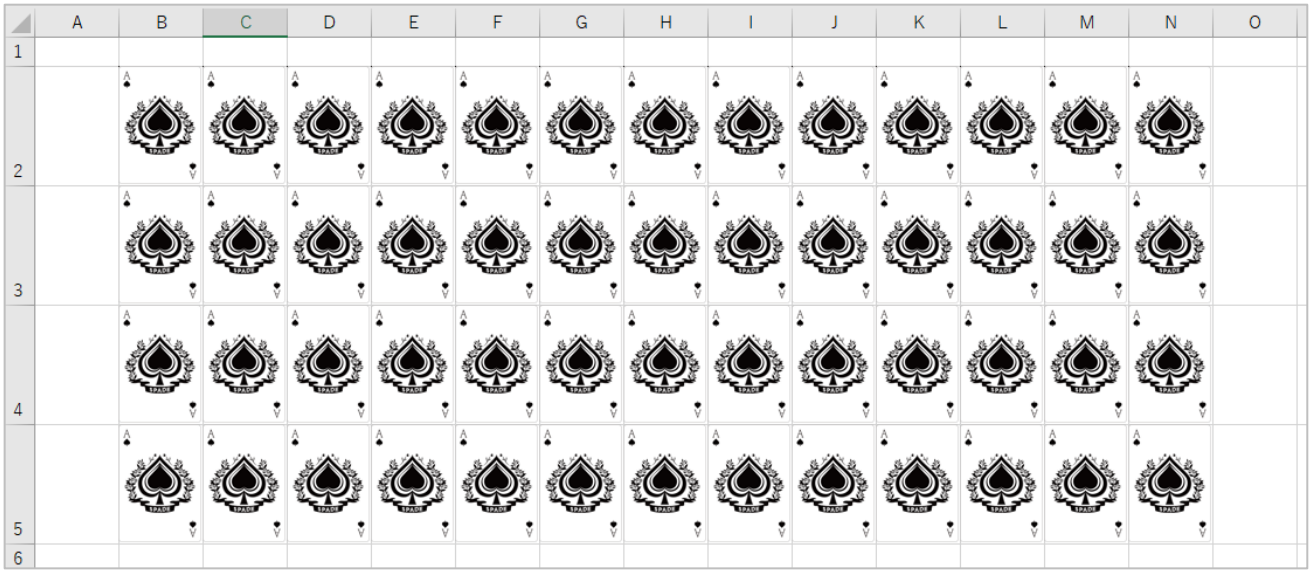
⑤picName に list()の要素を割り当てるように書き換えてください。

```
k = 0
For i = 2 To 5
    For j = 2 To 14
        picName = ActiveWorkbook.Path & "¥card¥" & list(k)
        Set myShape = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
            linktofile:=False, _
            savewithdocument:=True, _
            Left:=Cells(i, j).Left, _
            Top:=Cells(i, j).Top, _
            Width:=0, _
            Height:=0)
        myShape.ScaleHeight 0.1, msoCTrue
        myShape.ScaleWidth 0.1, msoCTrue
    Next j
Next i
```

⑥画像の表示位置を、i 行目、j 列目に書き換えてください。

これで実行してみましょう。

全部♠A が表示されました。



それもそのはず。k を変化させていません。（k = 0 のまま）

でも、５２枚の画像表示は成功！

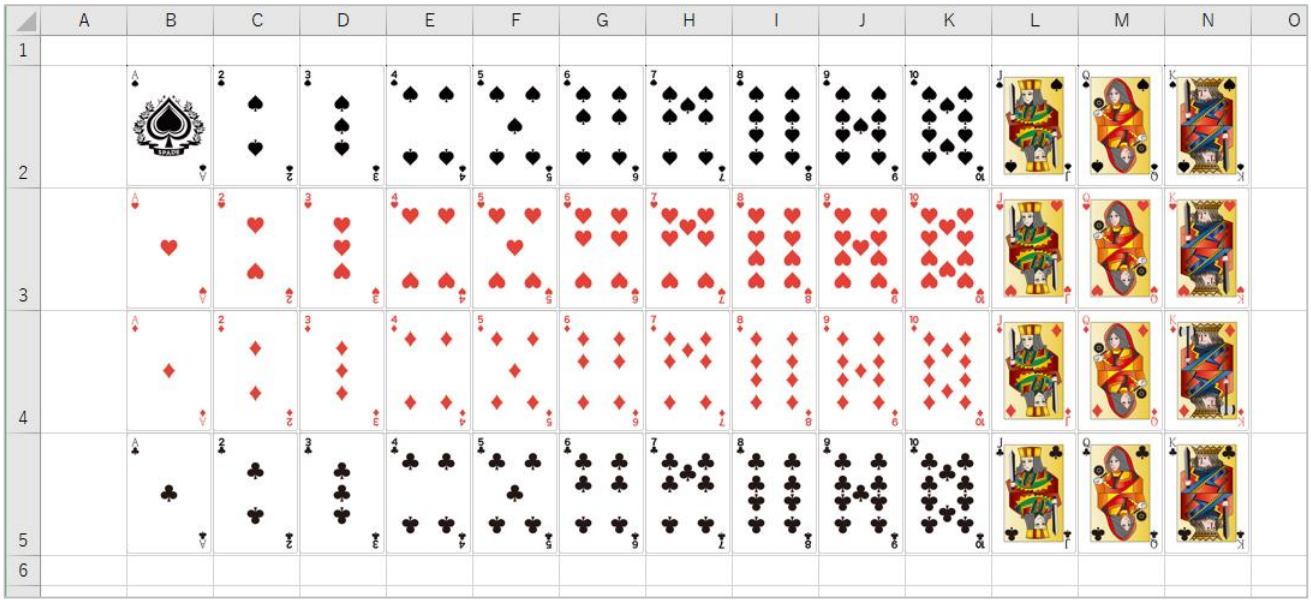
では、
⑦ k を１ずつ増やしていきましょう。

```
    k = 0
    For i = 2 To 5
        For j = 2 To 14
            picName = ActiveWorkbook.Path & "¥card¥" & list(k)
            Set myShape = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
                linktofile:=False, _
                savewithdocument:=True, _
                Left:=Cells(i, j).Left, _
                Top:=Cells(i, j).Top, _
                Width:=0, _
                Height:=0)
            myShape.ScaleHeight 0.1, msoCTrue
            myShape.ScaleWidth 0.1, msoCTrue

            k = k + 1
        Next j
    Next i
```

画像を表示した直後で、k を 1 増やしてください。

実行してみましょう。

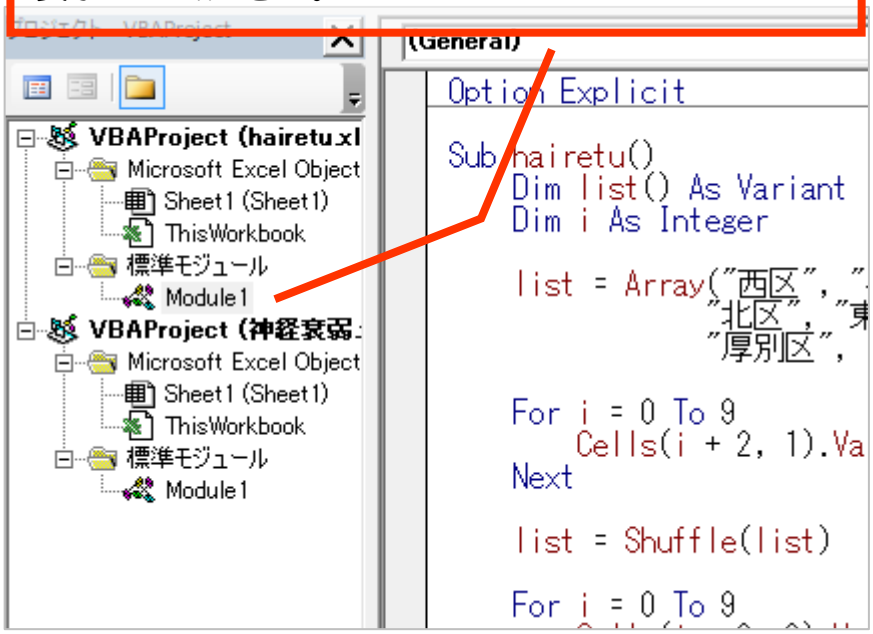
こうなれば OK です。



2－6　画像名の配列をシャッフルする

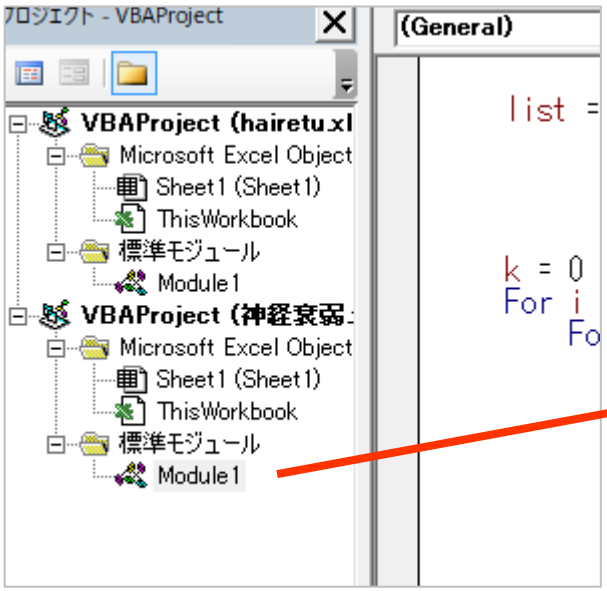
　2－3節で作ったシャッフル関数を持ってきて、画像表示するループの前段階で list をシャッフルしてから画像表示してみましょう。
　シャッフルする関数は hairetu.xlsm で作りましたので、そこからコピーして使いましょう。

① hairetu.xlsm の Module1 をダブルクリックして表示してください。



```
    list = Shuffle(list)

    For i = 0 To 9
        Cells(i + 2, 3).Value = list(i)
    Next


End Sub


Function Shuffle(list)
    Dim i As Integer
    Dim rn As Integer
    Dim tmp

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

    Shuffle = list

End Function
```

②Shuffle 関数をコピーしてください。



③神経衰弱.xlsm の Module1 をダブルクリックして表示してください。

```
                Height:=0)
            myShape.ScaleHeight 0.1, msoCTrue
            myShape.ScaleWidth 0.1, msoCTrue

            k = k + 1
        Next j
    Next i

End Sub


Function Shuffle(list)
    Dim i As Integer
    Dim rn As Integer
    Dim tmp

    For i = 0 To UBound(list)
        Randomize
        rn = Int(UBound(list) * Rnd)
        tmp = list(i)
        list(i) = list(rn)
        list(rn) = tmp
    Next

    Shuffle = list

End Function
```

画像表示しているプロシージャ

④画像表示しているプロシージャの外側に Shuffle 関数を貼り付けてください。

⑤プロシージャ本体から Shuffle 関数を呼び出しましょう

```
    list = Array("SA.png", "S2.png", "S3.p
               "HA.png", "H2.png", "H3.png",
               "DA.png", "D2.png", "D3.png",
               "CA.png", "C2.png", "C3.png",

    list = Shuffle(list)

    k = 0
    For i = 2 To 5
        For j = 2 To 14
            picName = ActiveWorkbook.Path
            Set myShape = ActiveSheet.Shap
```

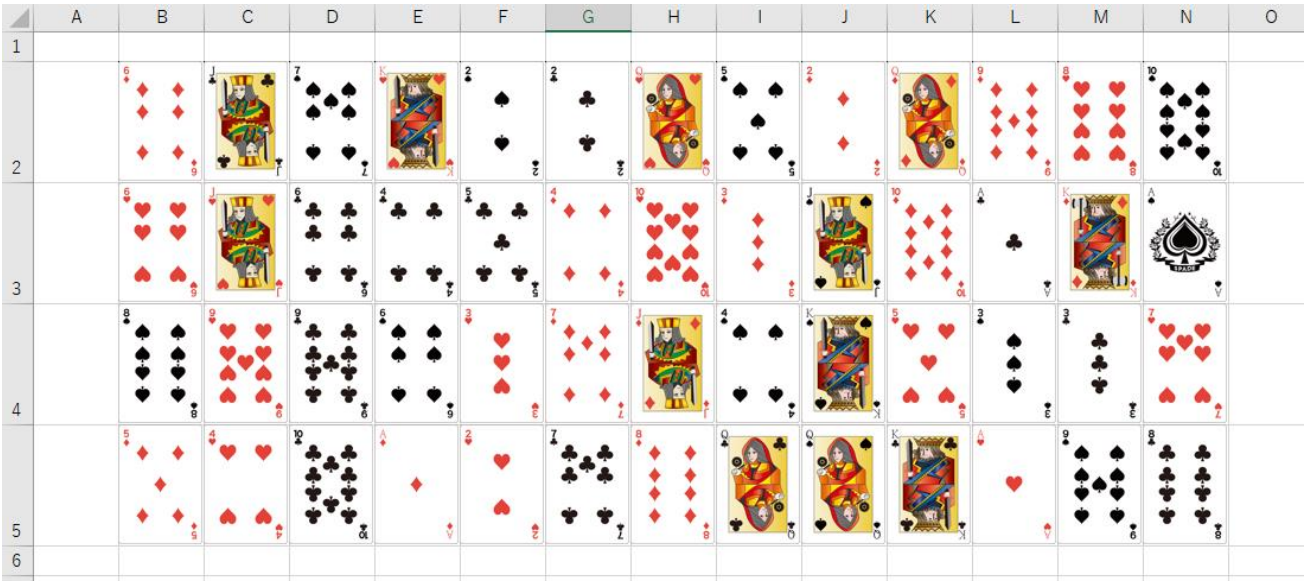
list 初期化部分

list 初期化後、画像表示する前で Shuffle 関数に list を突っ込んでシャッフルさせてください。

画像表示部分

ここまでできたら実行してみましょう。

カードがランダムに表示されれば OK です。



２－７　伏せたカード（裏面）を表示する

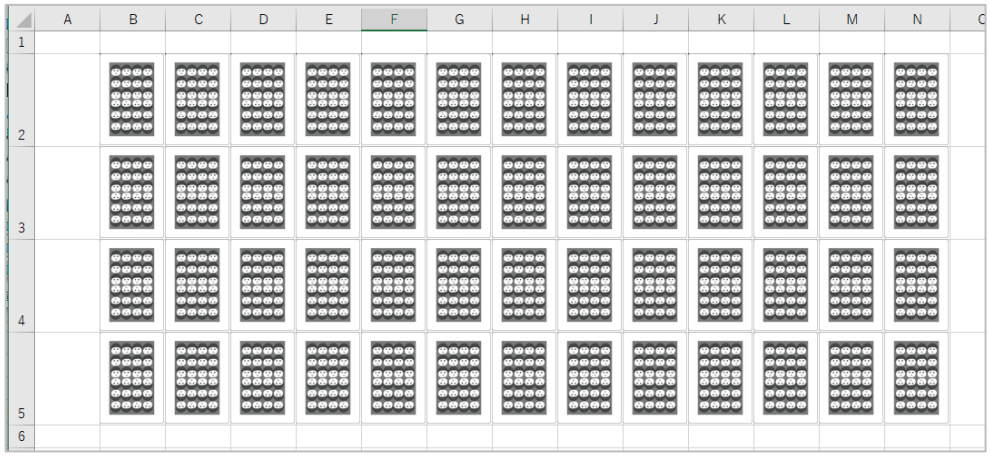
神経衰弱ですから、実際はカードの数字は見えないように伏せて置かれます。
ですので、伏せた状態のカード（裏面）の画像を 52 枚表示するようにしましょう。

【練習２－５】
オモテ面（図柄の面）がランダムに表示されるように作ったプロシージャを改造して、52 枚の裏面を表示するようにしてください。

（ヒント）

- 表面の画像ファイルは、ura.png です。
- picName にその画像ファイルの在処をセットするとよいです。「１－１－２ 表示したい画像ファイルを指定」を参考にしてください。

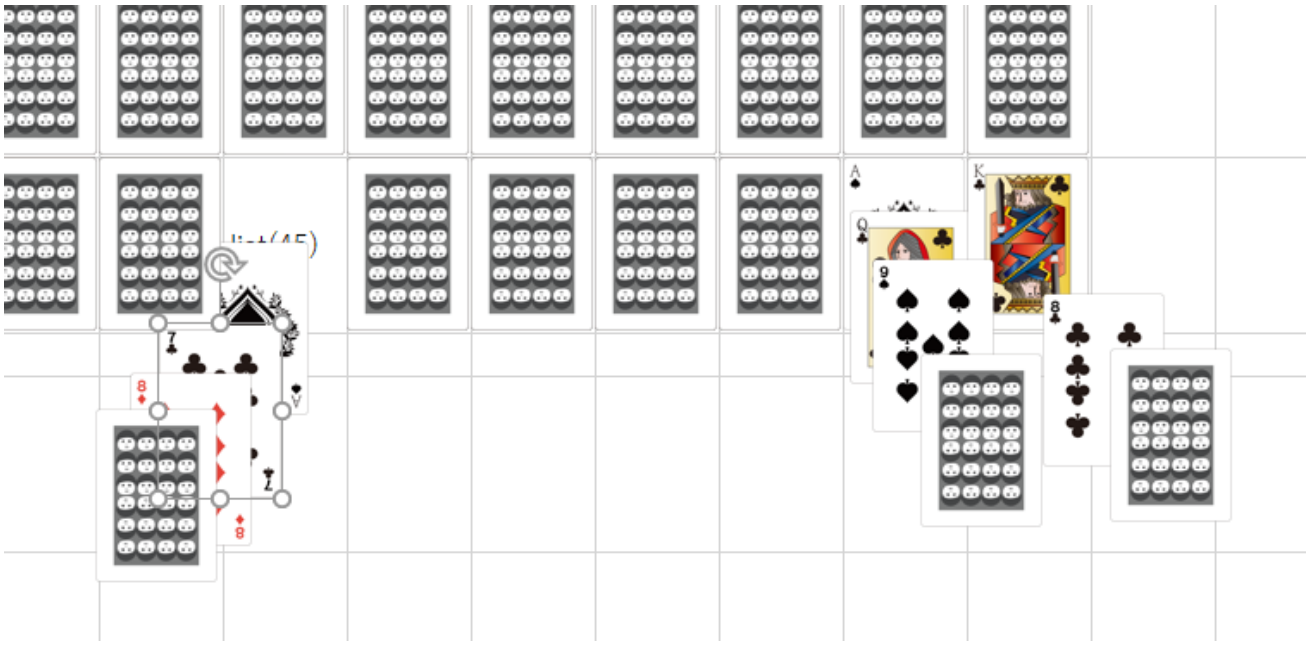
実行して、下図のようになれば OK です。



第３章　あらかじめ、すべての画像を消去しておく

３－１　概要

現段階では、過去に表示した画像が消えずに隠れています。
画像をドラッグしてずらすと、それが残っていることがすぐにわかります。



プロシージャを実行するたび・新たに画像を配置する前にすべて削除する仕組みを作っていきます。

３－２　すべての画像を削除する関数を作る

モジュールの最後に次の関数を追加してください。この段階ではまだ枠組みのみです。

```
    Next
    Shuffle = list
End Function

Function allShapeDelete()
End Function
```

① allShapeDelete という名前で関数をつくりましょう。

```
Function allShapeDelete()
    Dim shp As Shape

End Function
```

② 画像を扱いますので、Shape 型の変数をひとつ用意します。

③ この 3 行を追記してください。

```
Function allShapeDelete()
    Dim shp As Shape

    For Each shp In ActiveSheet.Shapes
        If shp.Type = msoPicture Then shp.Delete
    Next shp

End Function
```

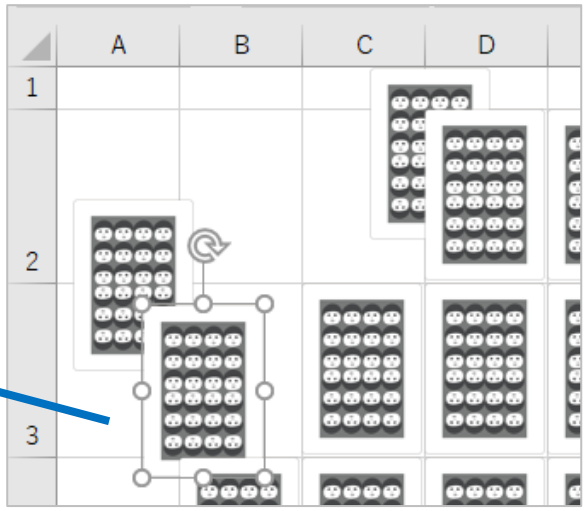
※この関数についての解説は、次ページをご参照ください。
概要としては、シートにあるすべての画像を消去する処理を行っています。

本体プロシージャの変数定義直後あたりでこの関数を呼び出してください。

```
Sub sample()
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer

    allShapeDelete

    list = Array("SA.png", "S
                 "HA.png", "H2.png
                 "DA.png", "D2.png
```

④ allShapeDelete を呼び出す。

実行して、画像をずらしてみましょう。過去に配置した図がないことを確認しましょう。

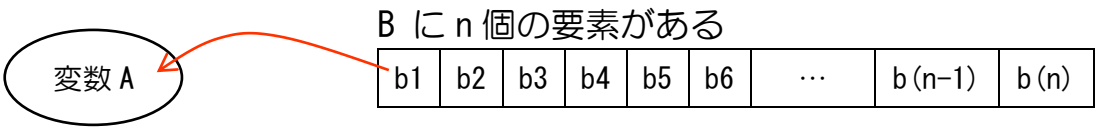


ずらしてみても下に隠れていませんね。

For Each shp In ActiveSheet.Shapes の解説

For Each A In B

B に要素が存在する限り順番に A に入れてループ処理する



1 回目の処理は、変数 A に要素 b1 が入る
2 回目の処理は、変数 A に要素 b2 が入る
3 回目の処理は、変数 A に要素 b3 が入る
…
n-1 回目の処理は、変数 A に要素 b(n-1)が入る
n 回目の処理は、変数 A に要素 b(n)が入る

という流れで、要素の数だけループ処理します。

```
For Each shp In ActiveSheet.Shapes
```

ActiveSheet（現在操作しているシート）の Shapes（図形）１つを

変数 shp に代入します

```
    If shp.Type = msoPicture Then shp.Delete
```

もし変数 shp のタイプが | 画像 | だったら | shp を Delete（消去）

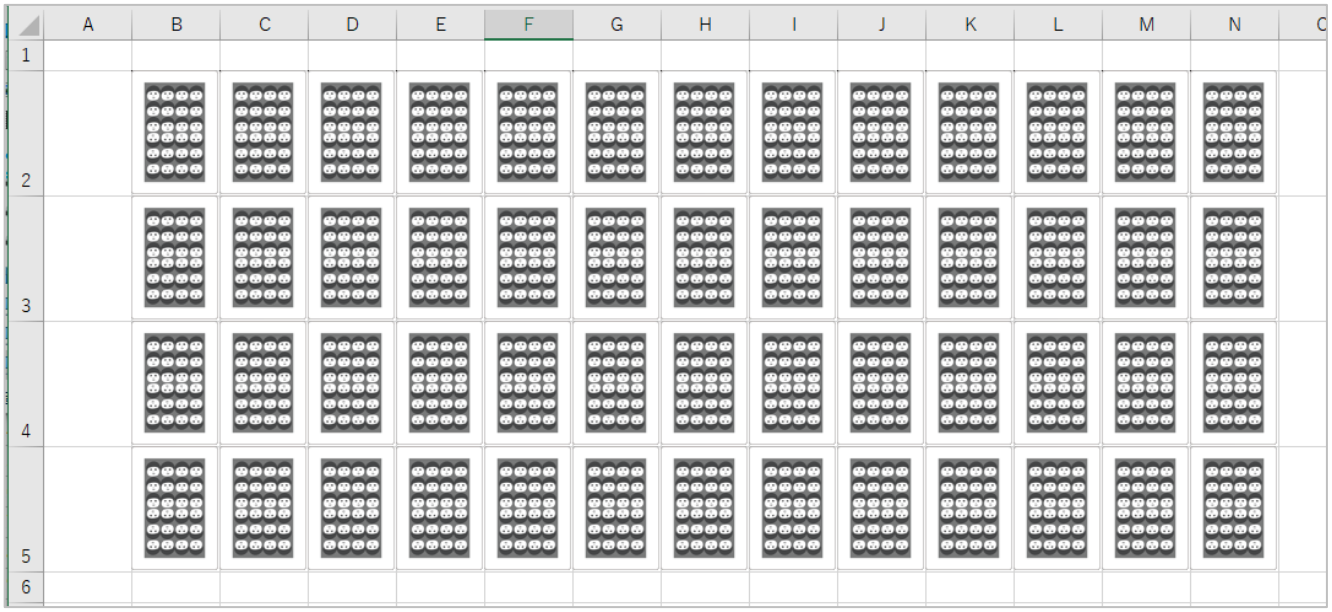
```
Next shp
```

繰り返します。For Each ループの先頭に戻ります。先頭に戻った時に、ActiveSheet にもう画像が無い状態になると shp に代入できませんから、このループが終わります。

第４章　クリックされたら画像を変える

トランプをめくる処理を追加していきます。
クリックされたら、絵柄の画像に切り替わるようにします。

その際、「何がクリックされたか」プログラム側でクリックされた画像を特定・識別する必要があります。



４－１　画像を配列で扱えるようにする

個々の画像を識別するために、画像それぞれに名前を設定しましょう。
そのために、まずは myShape を配列にしましょう。

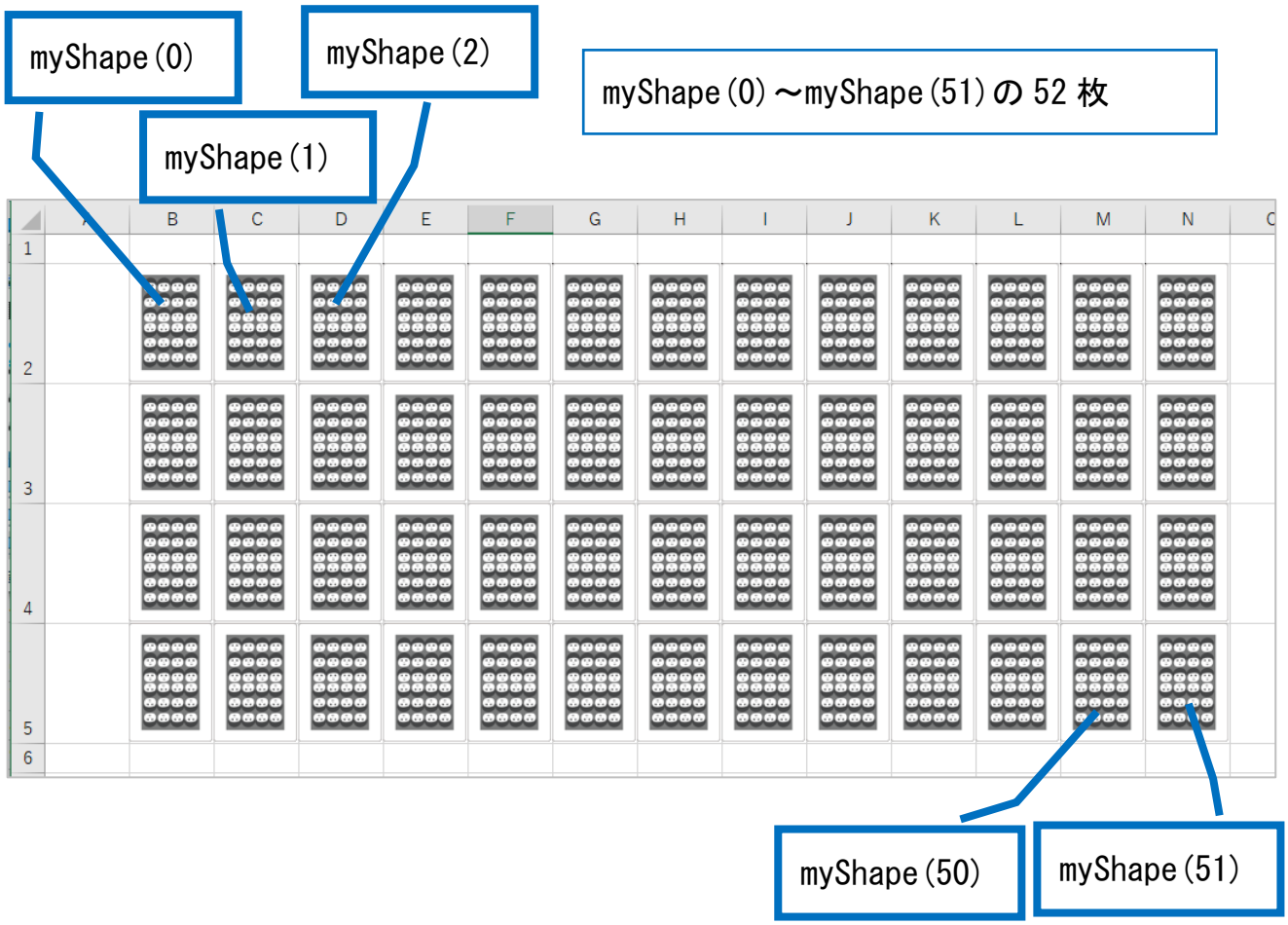
```
Option Explicit

Dim picName As String
Dim myShape(51) As Shape
Dim list() As Variant    'カードの名前リスト
```

宣言部分で、最終インデックス番号 51 の配列に書き換えます。

```
        picName = ActiveWorkbook.Path & "¥card¥ura.p
        Set myShape(k) = ActiveSheet.Shapes.AddPictu
            linktofile:=False, _
            savewithdocument:=True, _
            Left:=Cells(i, j).Left, _
            Top:=Cells(i, j).Top, _
            Width:=0, _
            Height:=0)
        myShape(k).ScaleHeight 0.1, msoCTrue
        myShape(k).ScaleWidth 0.1, msoCTrue

        k = k + 1
```

myShape だった部分を、配列インデックス(k)を追加してインデックス指定する形にします。

これで、配列 myShape では個々の画像をインデックス番号で識別できるようになりました。



４－２　個々の画像に画像名を割り当てる

```
    k = 0
    For i = 2 To 5
        For j = 2 To 14
            picName = ActiveWorkbook.Path & "¥card¥ura.png
            Set myShape(k) = ActiveSheet.Shapes.AddPicture
                linktofile:=False, _
                savewithdocument:=True, _
                Left:=Cells(i, j).Left, _
                Top:=Cells(i, j).Top, _
                Width:=0, _
                Height:=0)
            myShape(k).ScaleHeight 0.1, msoCTrue
            myShape(k).ScaleWidth 0.1, msoCTrue
            myShape(k).Name = list(k)

            k = k + 1
        Next j
    Next i
```

myShape(k) の名前プロパティに、画像名リスト list(k) から画像名を割り当てましょう。

これは、後の処理でクリックされた画像に切り替えられるようにする準備のひとつです。

４－３　OnAction で画像がクリックされたときの処理を呼び出す

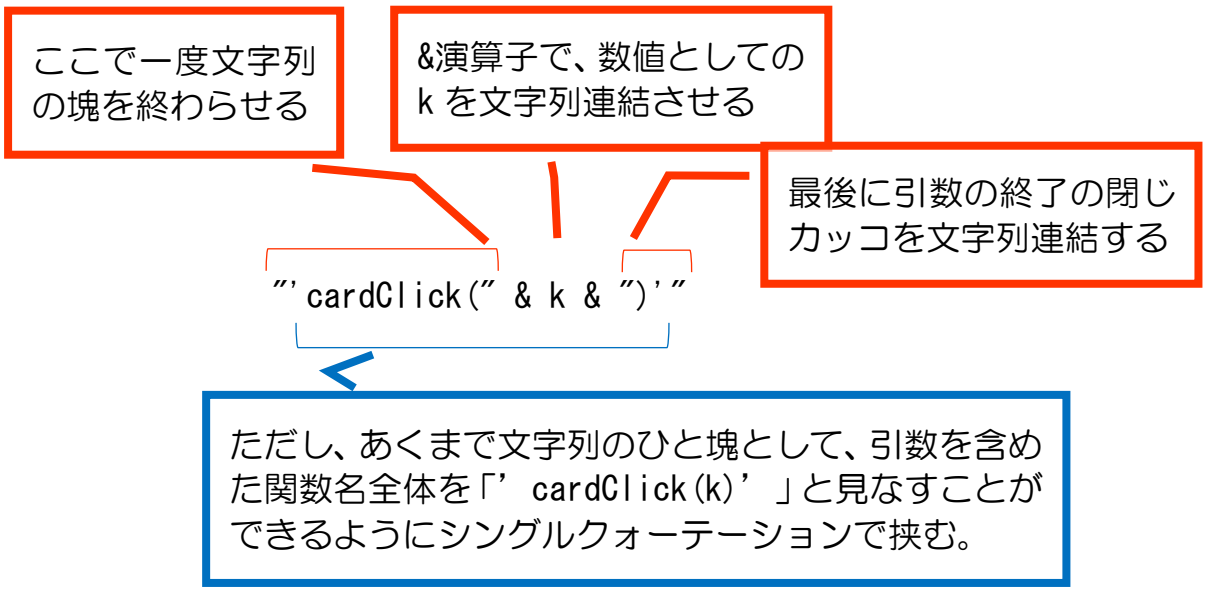
呼び出し側として、この１行を追加してください。

```
            Width:=0, _
            Height:=0)
        myShape(k).ScaleHeight 0.1, msoCTrue
        myShape(k).ScaleWidth 0.1, msoCTrue
        myShape(k).Name = list(k)
        myShape(k).OnAction = "'cardClick(" & k & ")'"

        k = k + 1
    Next j
```

この部分、大変複雑な記述になっています。

その理由は…

- 何番目の画像がクリックされたか、インデックス番号を cardClick 関数のに引数（ひきすう）として渡したい。
- OnAction の場合、関数を指定するときに「"関数名"」という感じでダブルクォーテーションで挟む必要がある。
- ダブルクォーテーションで挟むと、通常は文字列扱いのため、単純に「cardClick(k)」としたのでは引数「k」を数値として渡せない。



…といった事情によります。

芦原ウラ話
これを解決するのに１週間くらいかかりました。

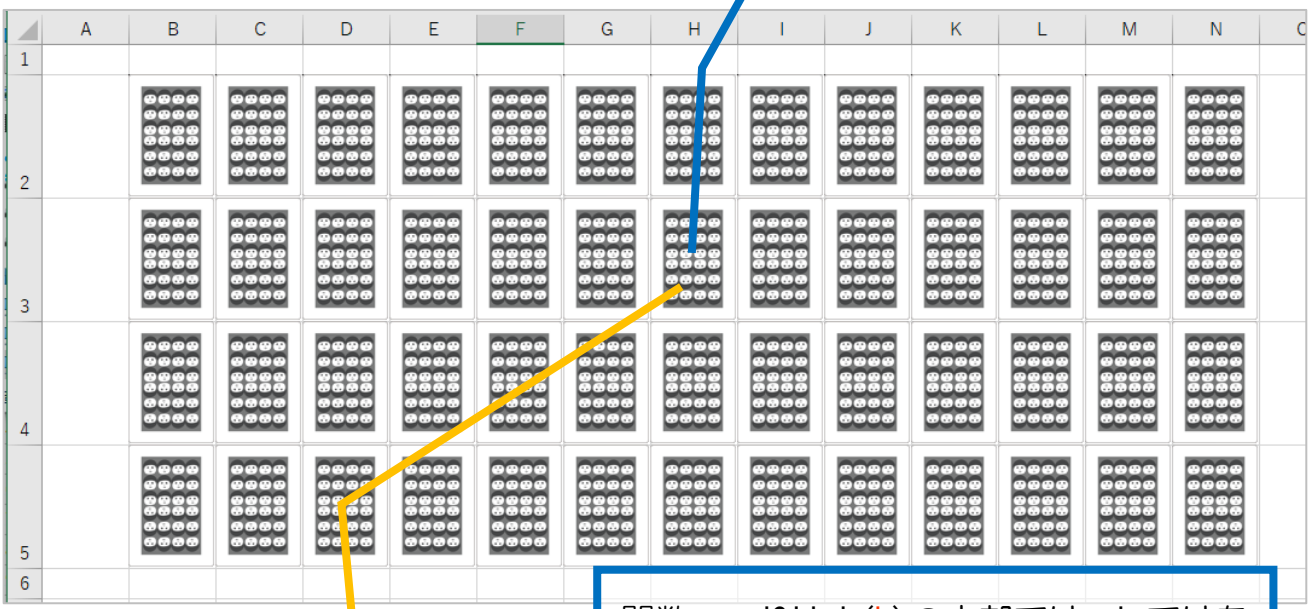
４－４　OnAction から呼び出される関数をつくる

```
            myShape(k).ScaleWidth 0.1, msoCTrue
            myShape(k).Name = list(k)
            myShape(k).OnAction = "'cardClick(" & k & ")'"

            k = k + 1
        Next j
    Next i

End Sub


Function cardClick(cardNo)
    MsgBox Application.Caller
End Function
```

本体プロシージャのすぐ後（どこでもいいのですが）に cardClick(cardNo) 関数を作ってください。

本体プロシージャで渡している k は、関数の内部では cardNo として扱うこととします。「何番目のカードがクリックされたか」という情報のため、cardNo という名称にしておくと分かりやすくなります。

クリックされたカード myShape(k) のインデックス番号 k が cardClick(k) として渡される。



関数 cardClick(k) の内部では、k ではなく cardNo という変数で扱われる。

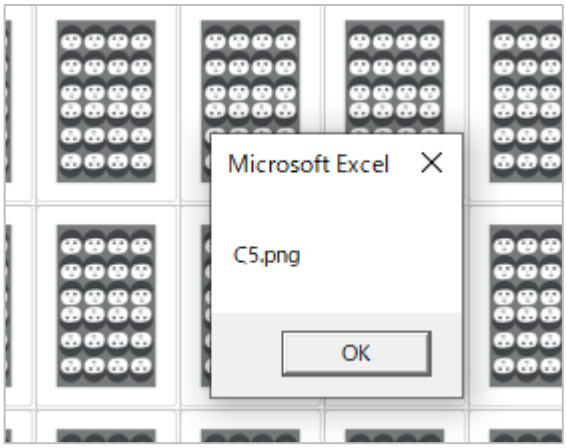
クリックされたカード myShape(k) には myShape(k).Name = list(k) としてめくったときの画像ファイル名が割り当てられている。

関数内のこの１行

```
MsgBox Application.Caller
```

は、関数を呼び出したものの名前・すなわちクリックされた画像に割り当てられている名前（ファイル名に相当）をメッセージボックスで表示します。

実行して、任意の画像をクリックしてみましょう。



上図のように、画像ファイル名がメッセージボックスで表示されることと思います。
ここまでできれば、画像ファイル名の特定もできているので表示するだけです。

４－５　画像ファイル表示を関数化

cardClick 関数内で画像表示することになるのですが、また Set myShape～の 10 行程度を書くのは冗長です。
何度も出てくる同じような作業は関数化して呼び出して使う運用にしましょう。

画像表示処理の元ネタ部分はこれです。これの Set MyShape～の部分を関数化します。

```
            picName = ActiveWorkbook.Path & "¥card¥ura.png"
            Set myShape(k) = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
                linktofile:=False, _
                savewithdocument:=True, _
                Left:=Cells(i, j).Left, _
                Top:=Cells(i, j).Top, _
                Width:=0, _
                Height:=0)
            myShape(k).ScaleHeight 0.1, msoCTrue
            myShape(k).ScaleWidth 0.1, msoCTrue
            myShape(k).Name = list(k)
            myShape(k).OnAction = "'cardClick(" & k & ")'"
```

関数化する際に、引数として渡す必要があるのは
picName（画像ファイルのパス・ファイル名）
i（表示されている行番号）　→　関数内では row とします。

j（表示されている列番号）　→　関数内では col とします。
k（カードのインデックス番号）　→　関数内では cardNo とします。

関数としては下図のようになります。
関数名は任意ですが、例として picSet とします。引数・変数を適宜書き換えてください。

```
Function picSet(picName, row, col, cardNo)

    Set myShape(cardNo) = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
        linktofile:=False, _
        savewithdocument:=True, _
        Left:=Cells(row, col).Left, _
        Top:=Cells(row, col).Top, _
        Width:=0, _
        Height:=0)
    myShape(cardNo).ScaleHeight 0.1, msoCTrue
    myShape(cardNo).ScaleWidth 0.1, msoCTrue
    myShape(cardNo).Name = list(cardNo)
    myShape(cardNo).OnAction = "'cardClick(" & cardNo & ")'"

End Function
```

本体プロシージャの呼び出し側はかなりシンプルになります。

```
Call picSet(picName, i, j, k)
```

として呼び出してください。

```
    k = 0
    For i = 2 To 5
        For j = 2 To 14
            picName = ActiveWorkbook.Path & "¥card¥ura.png"
            Call picSet(picName, i, j, k)
            k = k + 1
        Next j
    Next i
```

関数を呼び出す1行のみに書き換える。

ここまでできたら、実行してください。
関数化する前と同じ動きをすれば OK です。

４－６　クリックされた画像を消去

画像がクリックされると、画像のインデックス番号を引数として cardClick 関数が呼び出されます。
cardClick 関数内では、インデックス番号は cardNo として扱われます。

すなわち、myShape(cardNo)がクリックされた画像です。
このオブジェクトを Delete メソッドを働かすと削除できます。

```
Function cardClick(cardNo)
    MsgBox Application.Caller
    myShape(cardNo).Delete
End Function
```

この１行を追加してください。

実行して、画像をクリックしてみましょう。
メッセージボックスが表示されたら OK をクリックし、画像が削除されることを確認できたら OK です。

４－７　めくられたときの画像ファイルを表示する

　クリックされて消去された画像の位置に、めくられたときの画像ファイルを表示しましょう。

　画像表示は、４－５節で関数化した picSet 関数を使うことができます。

```
picSet(picName, row, col, cardNo)
```

ですので、次の４つの引数を渡して呼び出すと画像表示できます。

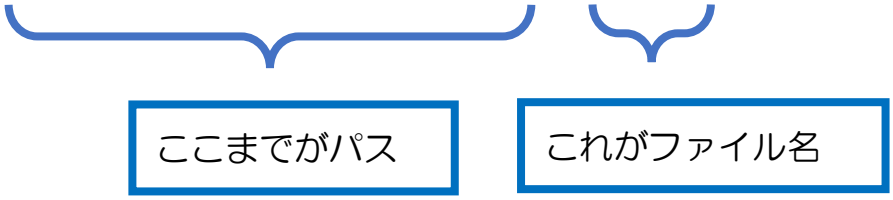
picName（画像ファイルのパス・ファイル名）
row（表示したい行番号）
col（表示したい列番号）
cardNo（カードのインデックス番号）

では、渡す引数を順番に考えていきましょう。

４－７－１　画像ファイルのパス・ファイル名

　「２－５　複数の画像を表示する」（p.28）を参考に、画像ファイルのパス・ファイル名は次のように指定しています。

```
picName = ActiveWorkbook.Path & "¥card¥" & list(k)
```



　今回は、ファイル名は Application.Caller で取得できています。（メッセージボックスで表示されていますよね。）

　ですので、

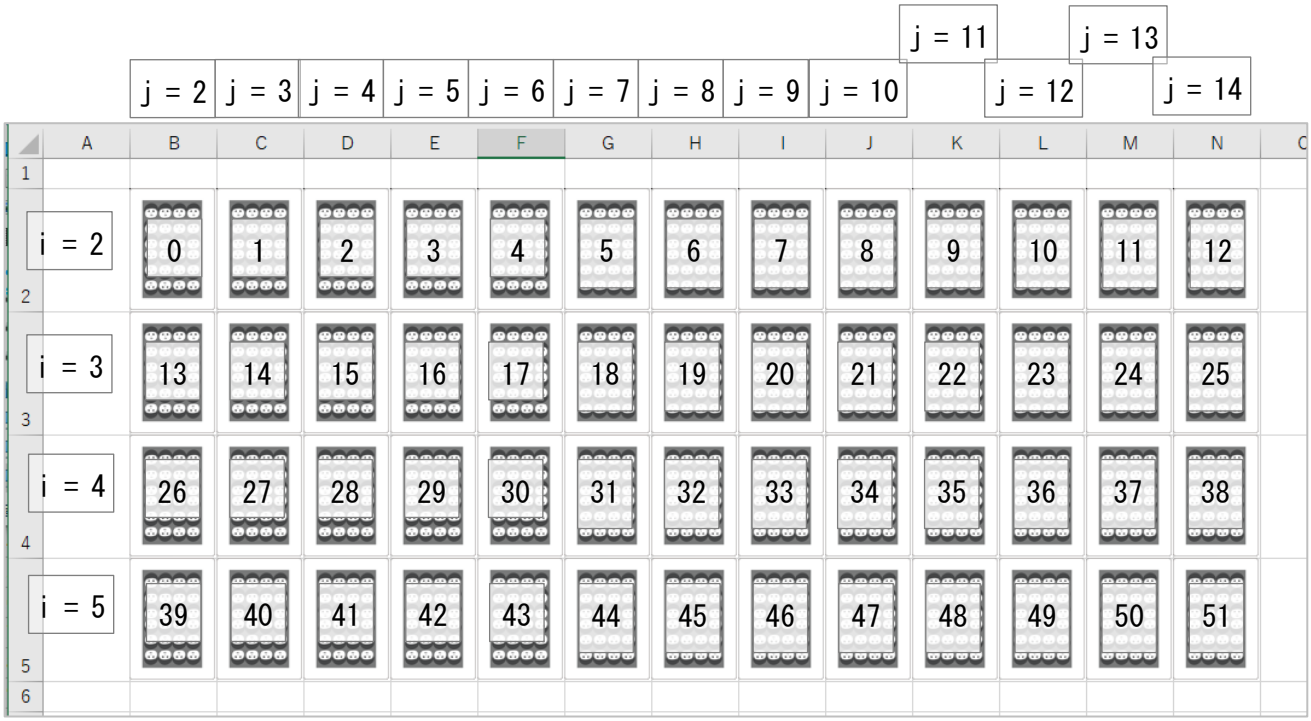
```
picName = ActiveWorkbook.Path & "¥card¥" & Application.Caller
```

としましょう。

４－７－２　消去された画像の行番号・列番号を割り出す

クリックされて消去された画像の位置が何行目・何列目なのか、割り出します。

cardNo は 0～51 です。



次の式で割り出すことができます。

```
行番号　　i = myShape(cardNo).TopLeftCell.row
列番号　　j = myShape(cardNo).TopLeftCell.Column
```

Shape オブジェクトの TopLeftCell プロパティはそのオブジェクトの左上が属するセル番号を Range で示します。そのセル番号からさらに行番号を取得するには row プロパティ、列番号を取得するには Column プロパティを指定します。

４－７－３　画像表示

【練習４－７－３①】

前節までで求めた picName（画像ファイルパス・ファイル名）、行番号、列番号、cardNo を引数として setPic 関数を使って画像表示する命令を cardClick 関数内に追加してください。

必要な変数はローカル変数として定義してください。

【練習４－７－３②】

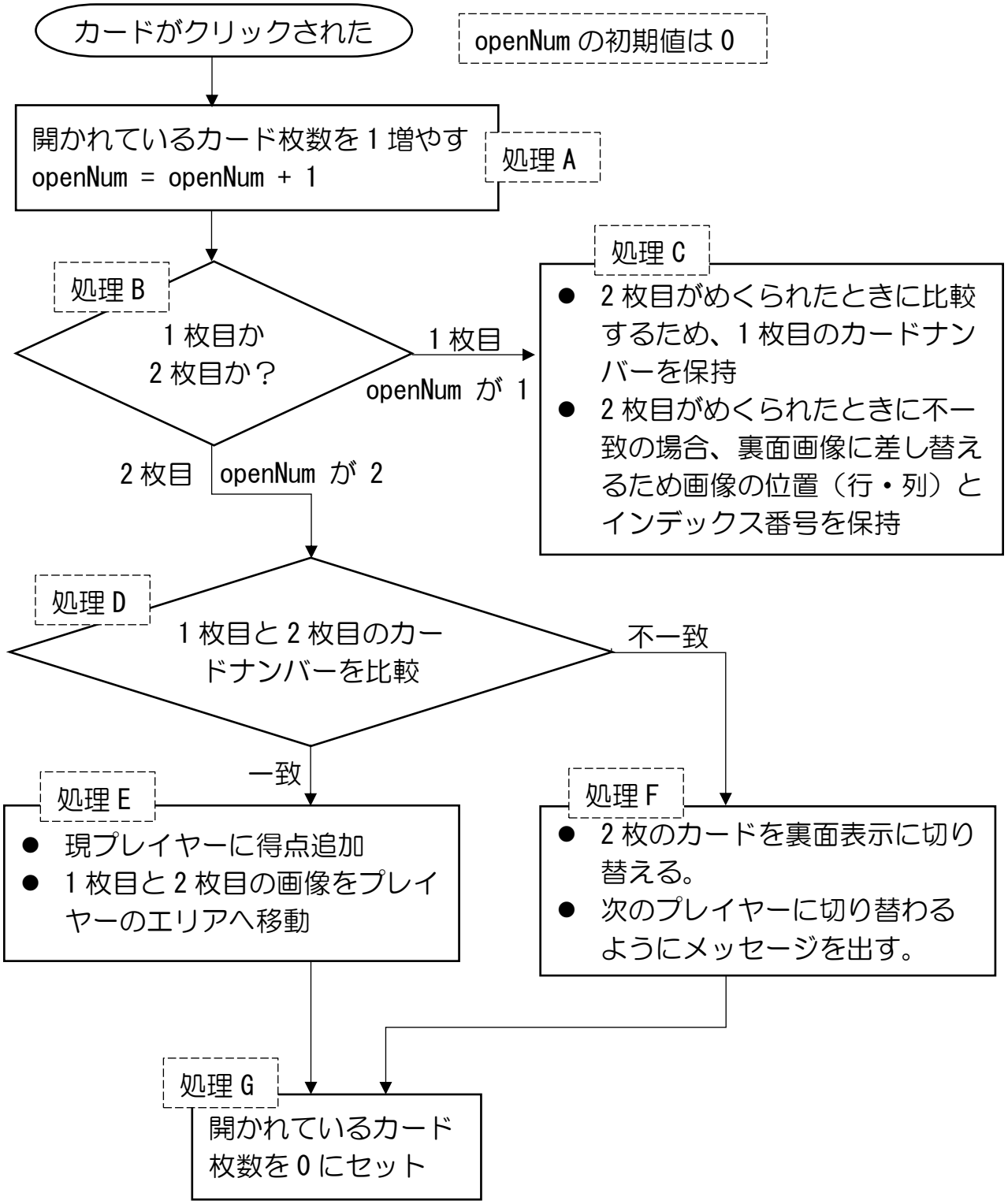
画像をクリックするたびにメッセージボックスが表示されています。表示されないようにしてください。

（ヒント）メッセージボックスを表示している命令を削除してください。

第５章　ゲームとしての設計

５－１　処理の流れ

ゲームとして成立させるおおまかな処理の流れは以下のようになります。



５－２　開かれているカード枚数

５－２－１　開かれているカード枚数を扱う変数

開かれているカード枚数は、モジュール全体で使いますのでグローバル変数にします。

```
Option Explicit

Dim picName As String
Dim myShape(51) As Shape
Dim list() As Variant   'カードの名前リスト
Dim openNum As Integer     '開かれているカード枚数
```

モジュール先頭の変数宣言が並んでいる場所に、この１行を追加してください。

openNum の初期値は、開かれている枚数が 0 なので 0 とします。
本体プロシージャの冒頭付近で 0 に初期化しましょう。

```
Sub sample()
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer

    openNum = 0    '開かれているカード枚数を0で初期化
    allShapeDelete
```

この１行を追加してください。

５－２－２　【処理 A】開かれているカード枚数を 1 増やす

cardClick 関数内で openNum を 1 増やしましょう。

```
    j = cardNo - 13 * Int(cardNo / 13) + 2

    Call picSet(picName, i, j, cardNo)

    openNum = openNum + 1
```

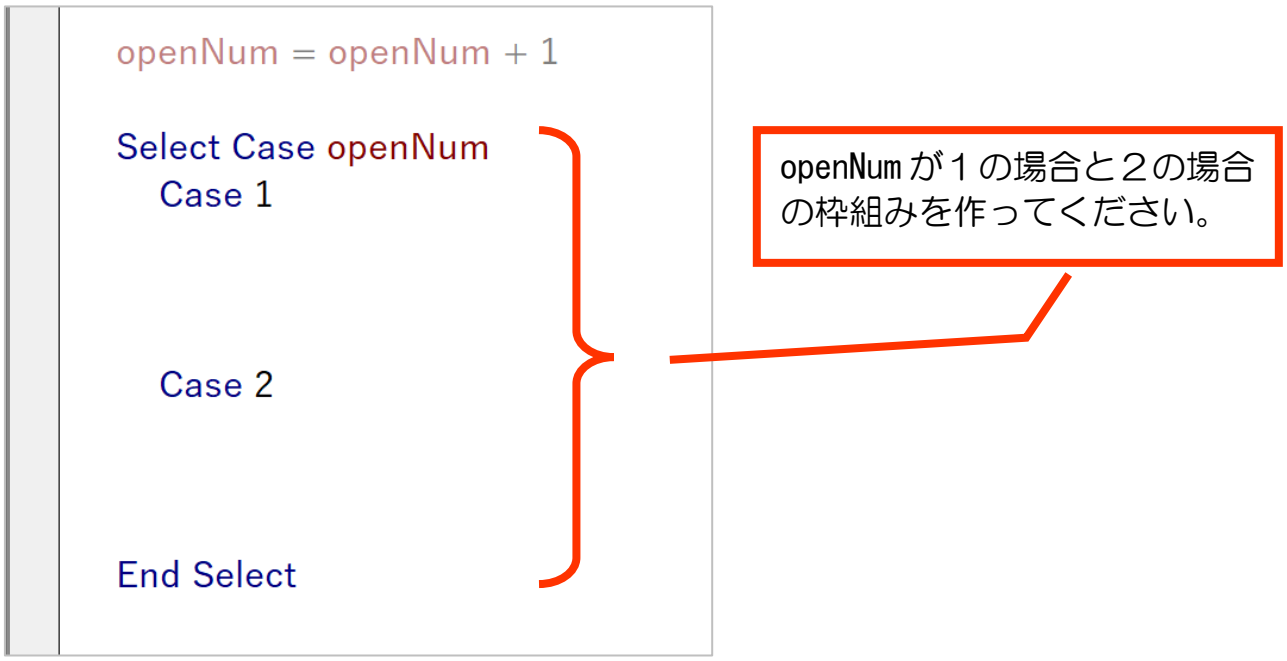
画像表示の後あたりで openNum を１増やしましょう。

初期値が 0 なので、画像が 1 回クリックされれば openNum は１、もう 1 回クリックされれば２、…というふうに開かれている枚数を表せます。

５－３　１枚目か２枚目かの条件分岐

５－３－１　【処理 B】１枚目か２枚目かの条件分岐

Select Case 構文を使って、openNum の値で条件分岐しましょう。



Select Case 構文

```
Select Case A
    Case B
        A の値が B のときの処理
    Case C
        A の値が C のときの処理
End Select
```

というように、条件によって異なる処理を行う場合に使います。

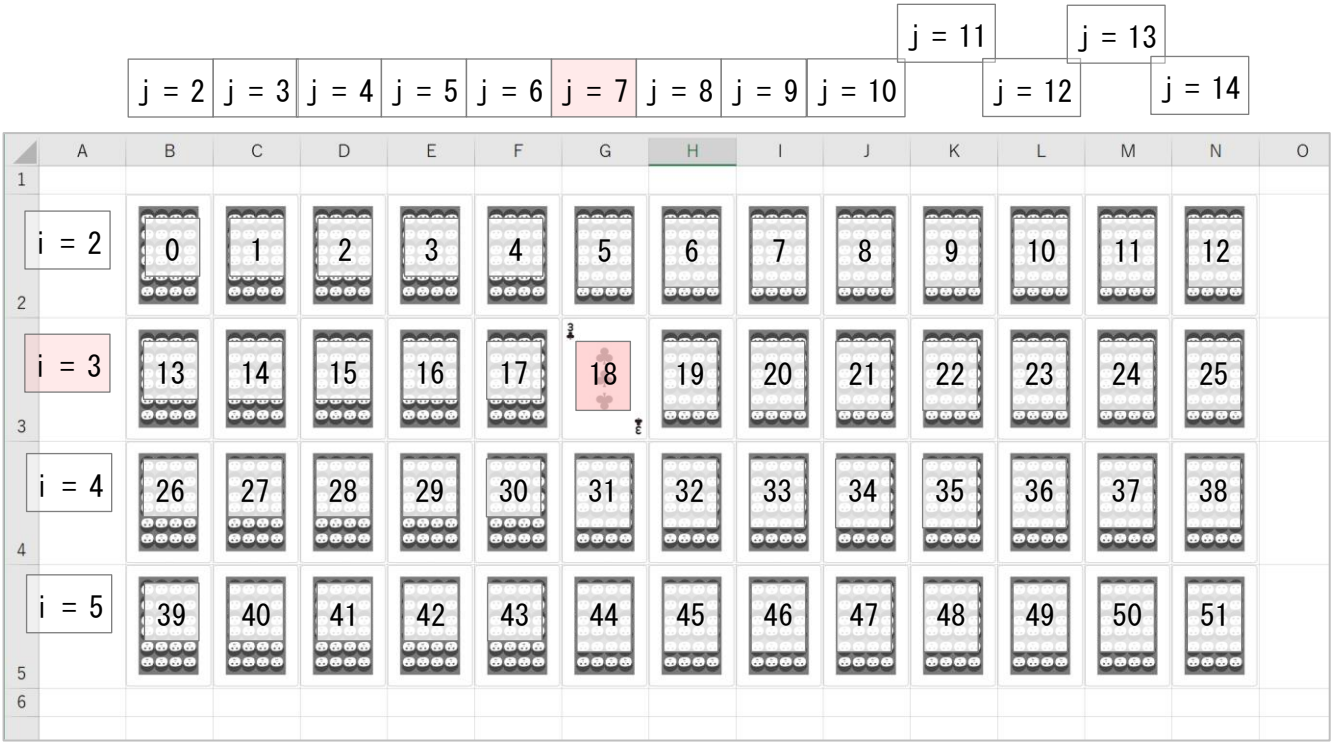
５－３－２　【処理 C】１枚目のときの処理

```
Dim list() As Variant   'カードの名前リスト
Dim openNum As Integer      '開かれているカード枚数
Dim target1row As Integer     '1枚目の行番号を保持
Dim target1col As Integer     '1枚目の列番号を保持
Dim target1 As String         '1枚目のカードナンバー
Dim index1 As Integer         '1枚目のカードのリスト内インデックスを保持
```

グローバル変数としてこの４つを宣言してください。

※target1 を String・文字列型にしているのは、数値（2～9）以外に a（エース）, j（ジャック）, q（クイーン）, k（キング）も扱うためです。

例えば、下図のようにセル G3（3 行目、7 列目）の画像が 1 枚目としてクリックされた場合、



target1row は３ ‥‥‥‥ target1row = i
target1col は７ ‥‥‥‥ target1col = j
index1 は 18‥‥‥‥‥‥‥ index1 = cardNo

target1 は、c3.png（オブジェクト名・ファイル名）のカードナンバー部分に相当する 3。

オブジェクト名は Application.Caller で取得し、2 文字目から 1 文字だけ取り出す。すなわち、

```
target1 = Mid(Application.Caller, 2, 1)
```

これらを Case 1 の処理に書きましょう。

```
Select Case openNum
    Case 1
        target1row = i
        target1col = j
        index1 = cardNo
        target1 = Mid(Application.Caller, 2, 1)
    Case 2



End Select
```

処理 C としての追加部分

動作確認用に、メッセージボックスで target1 を表示する命令を書き足して動作確認してみましょう。

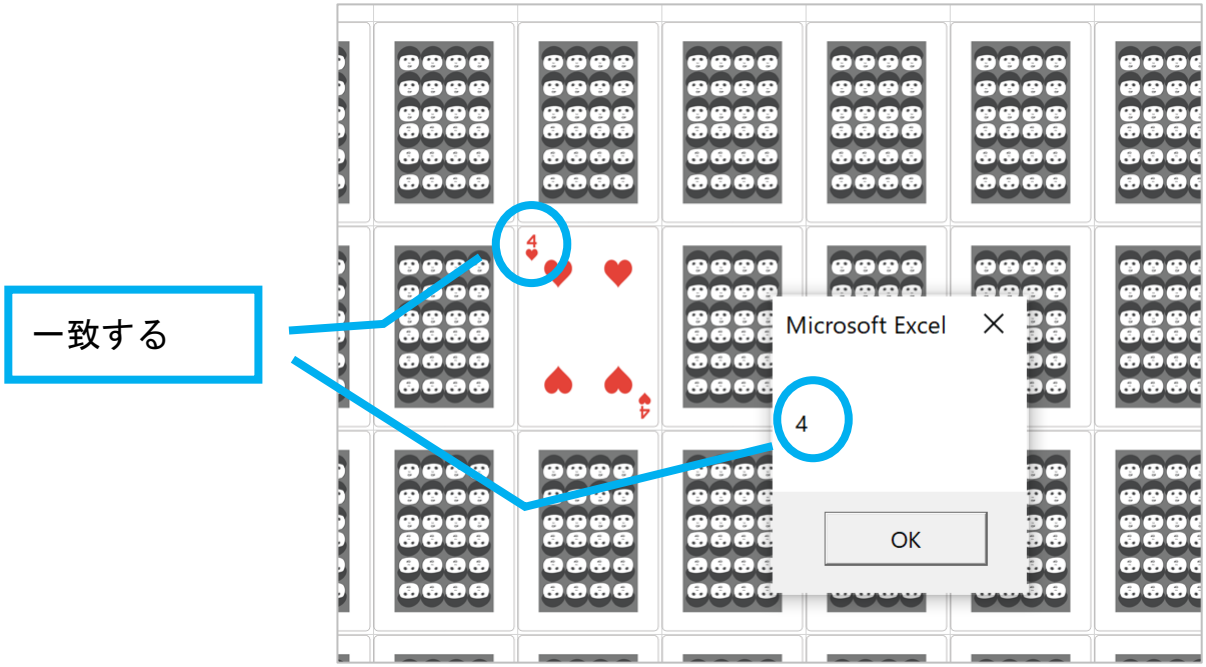
```
        index1 = cardNo
        target1 = Mid(Application.Caller, 2, 1)

        MsgBox target1
```

動作確認用のメッセージボックス追加

【確認事項】

1 回目のクリックで、メッセージボックスにカードのナンバー部分が表示される。

Select Case 構文の Case 1 への分岐ができていることが確認できます。



2 回目以降のクリックでは、メッセージボックスが表示されない。

Select Case 構文の Case 2 への分岐ができていることが確認できます。

※確認が終わったら、動作確認用メッセージボックス表示の行はコメントアウトしておきましょう。削除してしまってもいいです。

５－４　２枚目のときの処理

５－４－１　【処理 D】１枚目と２枚目のカードナンバーを比較

①カードナンバーを比較しますので、2 枚目のカードナンバーを変数 target2 に代入しましょう。
　まずは変数 target2 をグローバル変数として宣言・定義しましょう。

```
Dim target1 As String          '1枚目のカードナンバー
Dim index1 As Integer          '1枚目のカードのリスト内インデックスを保持
Dim target2 As String          '2枚目のカードナンバー
```

追加部分

　次に、target2 に 2 枚目のカードナンバーを代入しましょう。
【練習５－４】
５－３－２を参考に、target2 にカードナンバーを代入してください。

②target1 と target2 が同じかどうかを、IF 構文を使って比較します。

```
    Case 2
        target2 = Mid(Application.Caller, 2, 1)

        If target1 = target2 Then
            処理 E
        Else
            処理 F
        End If

    End Select
```

IF 構文の枠組みをこのように作ってください。

　処理 E と F はプレイヤーが複数人いることを想定していますので、次章にてプレイヤーの追加とともに作っていきます。

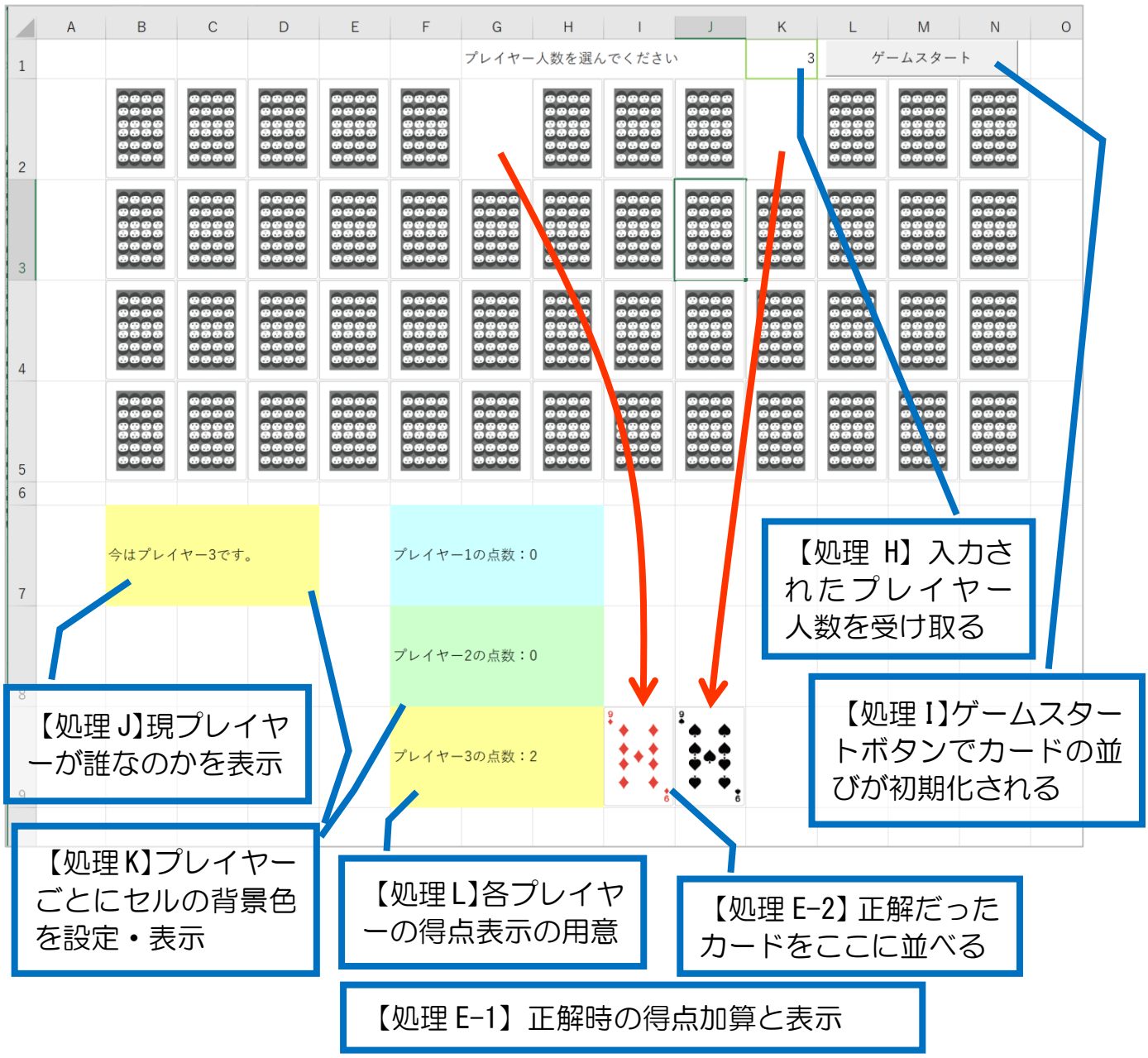
第６章　プレイヤーをつくる

６－１　概要

複数人で遊ぶことを想定します。
５－１で示した処理の流れに加え、下図に示すように処理 H～L を行います。

２枚めくって同じナンバーだったら得点追加。【処理 E-1】
現プレイヤーの得点表示エリアにその２枚のカードを移動。【処理 E-2】
そのプレイヤーが続けてめくります。

処理 E



次のプレイヤーに切り替わるようにメッセージを出します。【処理 F-1】
めくった２枚が異なる数字の場合、２枚のカードを裏面表示にきりかえます。【処理 F-2】

処理 F

６－２　【処理 H】入力されたプレイヤー人数を受け取る

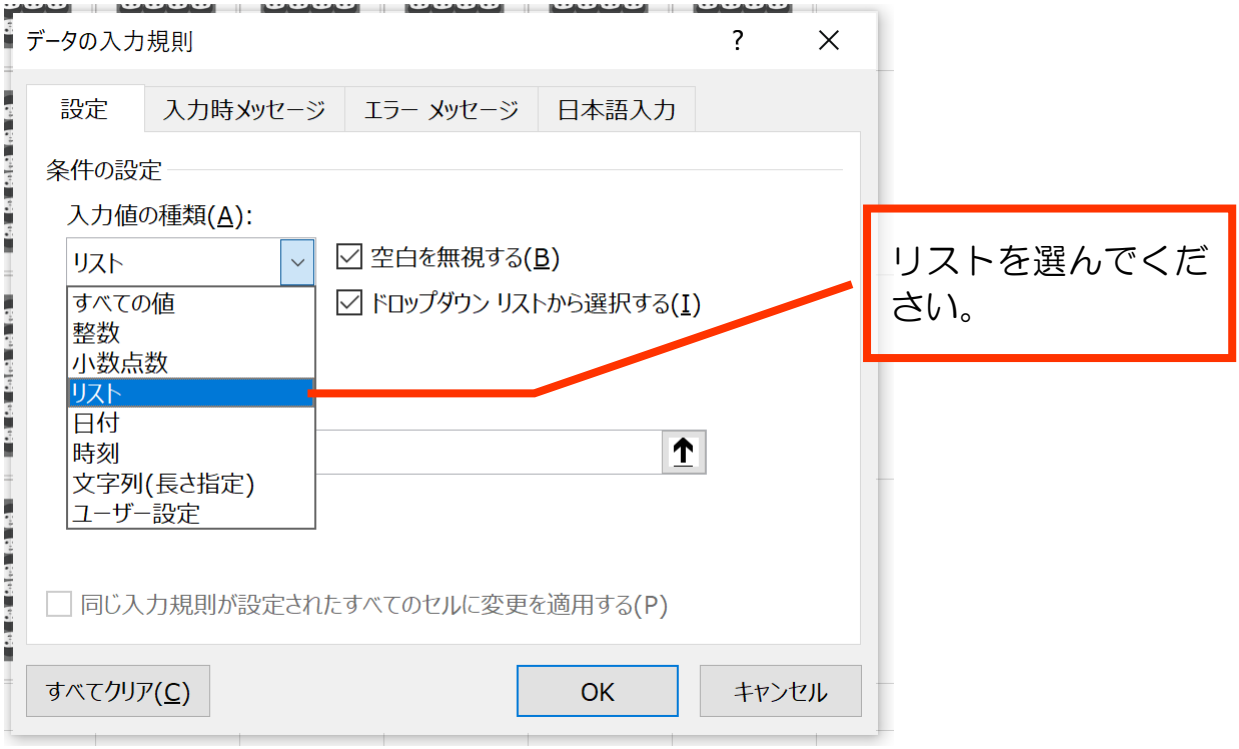
６－２－１　プレイヤー人数を入力してもらう

シートの１行目に、プレイヤー人数を入力してもらう仕組みを作ります。

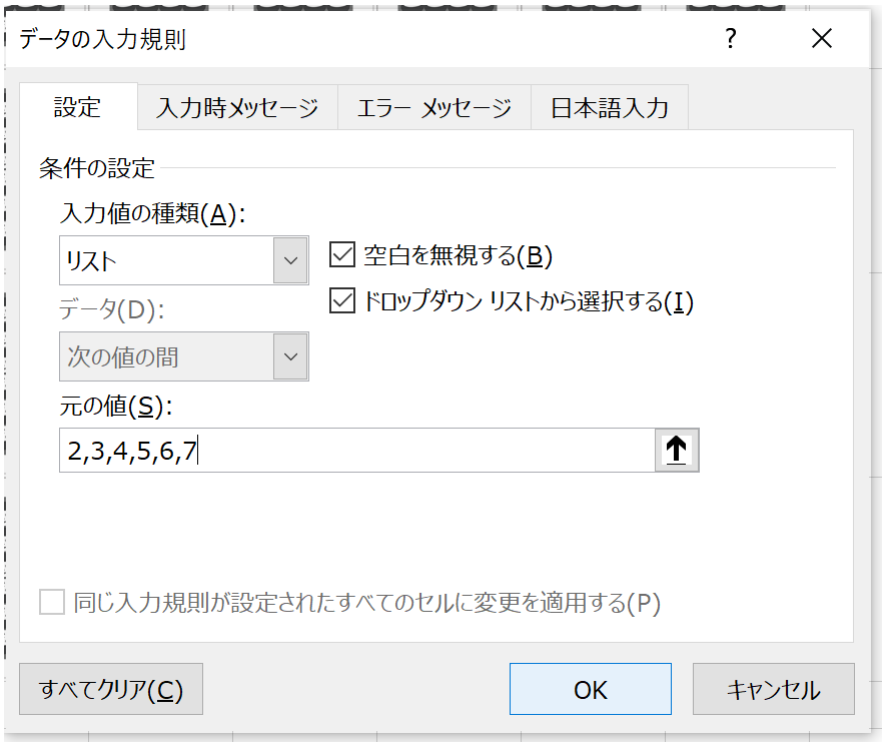
①セル G1 に「プレイヤー人数を選んでください」と入力してください。
１行目の高さが狭い場合は適宜任意の幅に広げてください。（サンプルでは 30 にしています。）

②セル K1 に入力規則を設定しましょう。
「データ」タブ→「データツール」グループ→「データの入力規則」

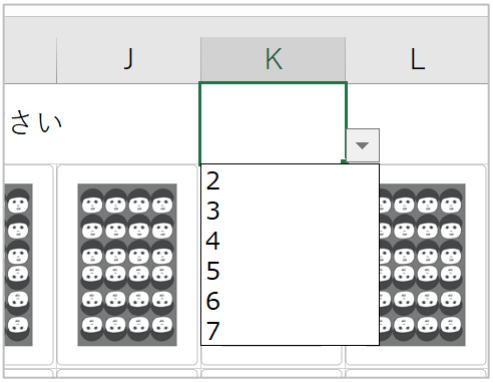
③「設定」タブにて、入力値の種類を「リスト」にしてください。



④「元の値」欄に半角で「2, 3, 4, 5, 6, 7」と入力し、OK をクリックしてください。



⑤セル K3 が、２～７のみをプルダウンメニューから選べる形になりました。



こうすることで、プレイヤー人数が２～７に限定されます。

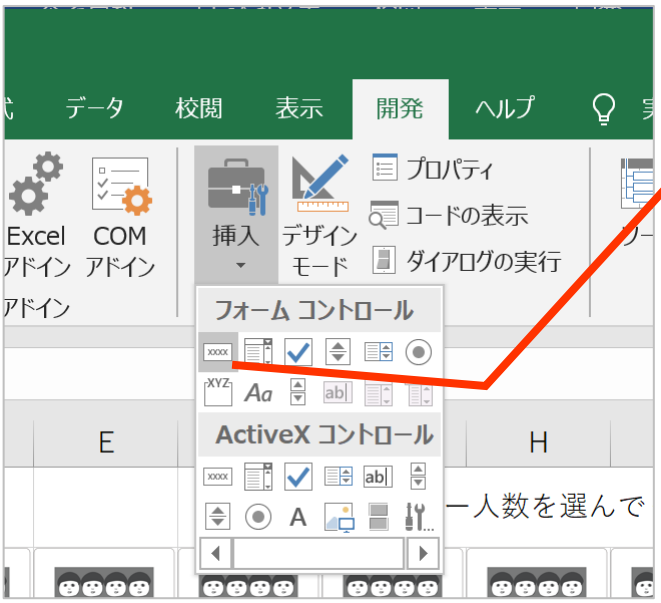
６－２－２　プレイヤー人数をプロシージャで受け取る

【練習６－２】
変数 player をグローバル変数・整数型で定義してください。
そして、セル K3 に入力されたプレイヤー人数を player に代入してください。
（アドバイス）
player に人数を代入するタイミングは、ゲーム開始直後で良いでしょう。

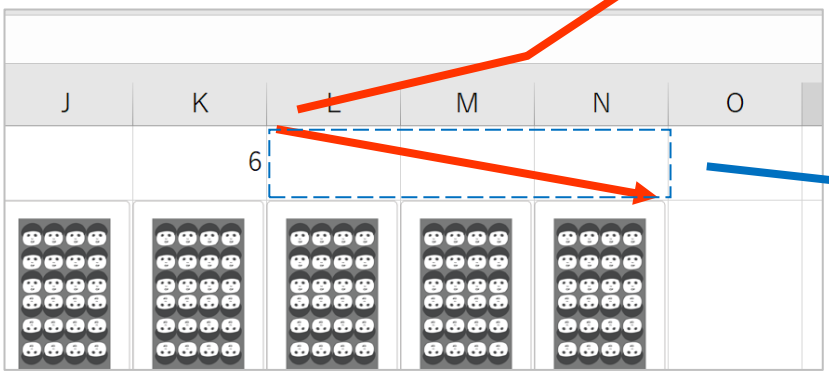
なお、本当に受け取れているかどうかの確認のため、セル M1 あたりに player の値を表示してみましょう。人数を変更してからプロシージャを実行して、入力された数値が M1 に表示されれば確認になります。（確認後はこの命令を消去してください。）

６－３　【処理（アイ）Ｉ】ゲームスタートボタンでカードの並びを初期化

ボタンを追加していきます。

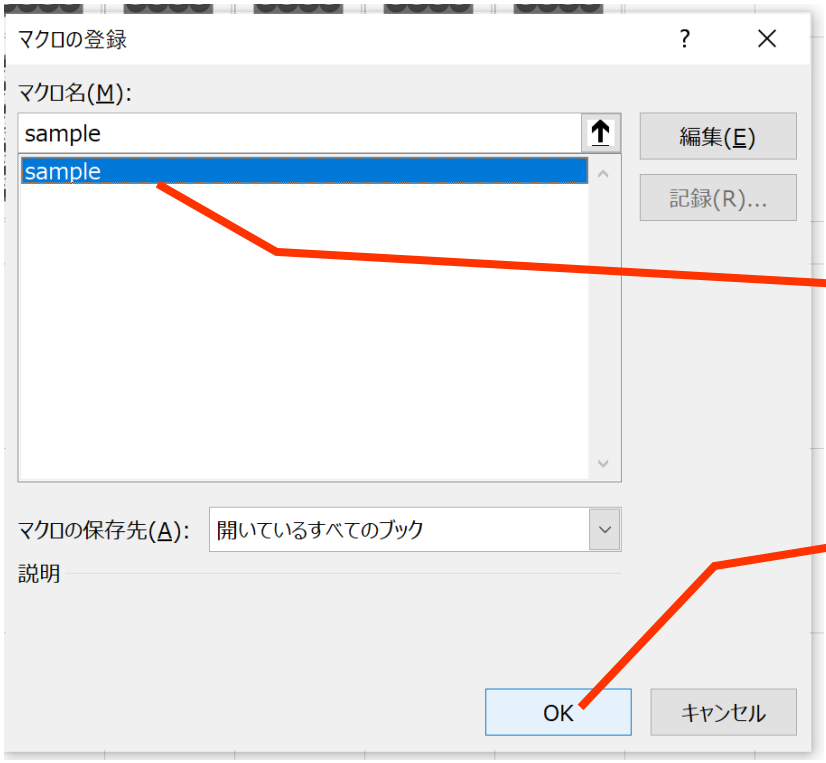


①ボタン（フォームコントロール）をクリックしてください。



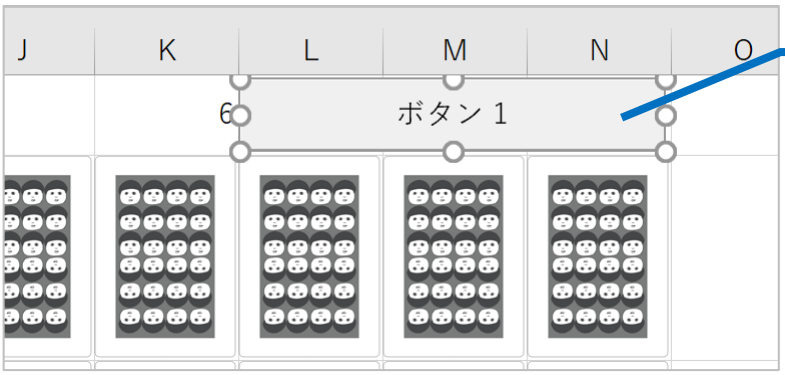
②セル L1 の左上からセル N1 の右下あたりまでドラッグしてください。

この範囲に四角いボタンが生成されますが、マウスを離すとマクロの登録」ダイヤログボックスが出てきます。

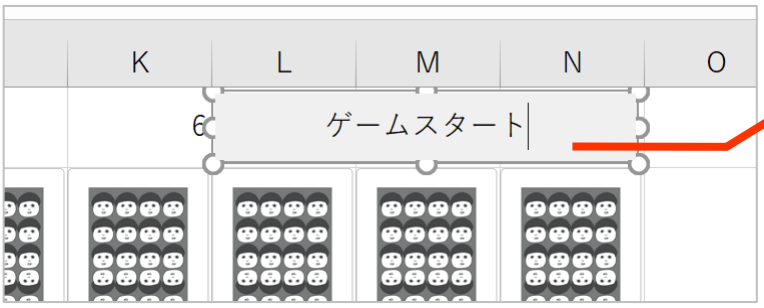


③メインのプロシージャ（例では sample）をクリックして、選択状態にしてください。

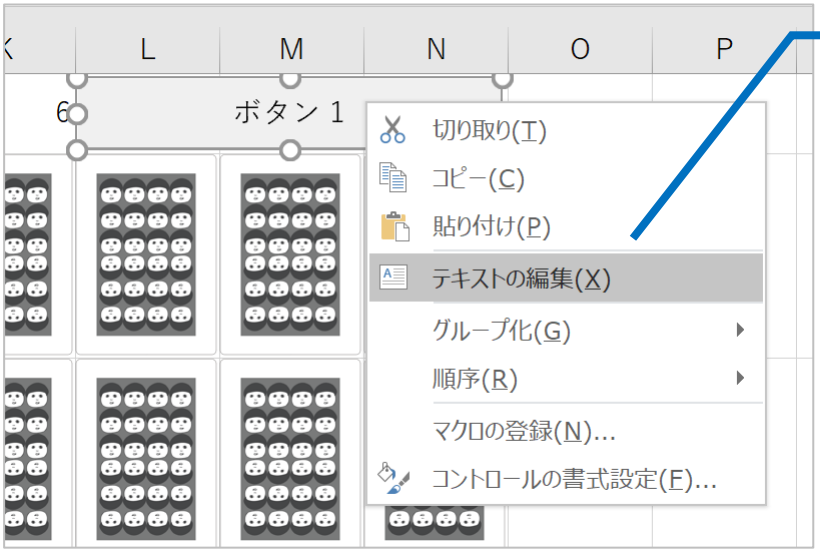
④「OK」をクリックしてください。



ボタンが出来ます。



⑤ボタンに表示されている文字を「ゲームスタート」に書き換えてください。



カーソルが出ていなくて書き換えられないときは、ボタンを右クリックして「テキストの編集」を選んでください。

これで、この「ゲームスタート」ボタンをクリックするとカードの並びが初期化されるようになりました。

６－４　【処理 J】現プレイヤーが誰なのかを表示する

①まずはプレイヤーを識別するための変数を作りましょう。

【練習６－４－１】
プレイヤーを識別するための変数 playerNum を宣言・定義しましょう。整数型にしてください。モジュール全体で使うので、グローバル変数にしておきましょう。

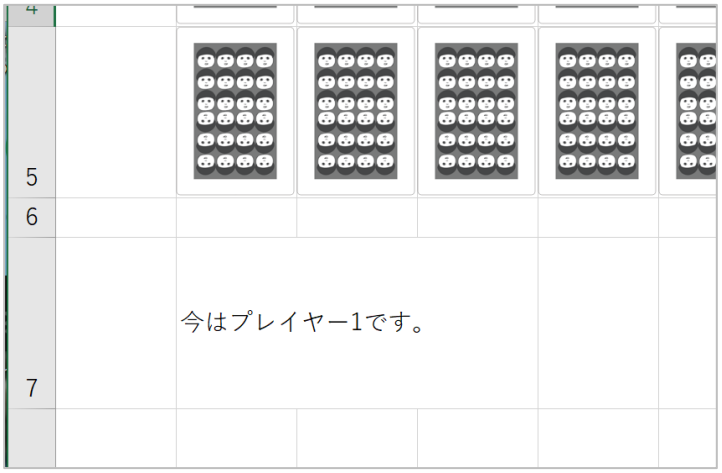
②プレイヤー番号を初期化しましょう。
　プレイヤー番号は１からプレイヤー人数までです。プレイヤー人数は６－２で作ったように、ユーザーによって入力された値（変数 player に代入されている値）です。
　まずはプレイヤー番号（playerNum）を１で初期化します。

【練習６－４－２】
playerNum に１を代入してください。
どのタイミングで行えばよいか、考えて追加してみましょう。
（ヒント）
諸々初期化しているタイミングが適当です。

③セル B7 に現プレイヤーが誰なのかを表示しましょう。

【練習６－４－３】
　セル B7 に「今はプレイヤー（番号）です。」と表示してください。
　（番号）の部分は変数 playerNum です。
　文字列と変数の連結を用いて作ってみましょう。
　どのタイミングで行えばよいか、考えて追加してみましょう。
（ヒント）
playerNum 初期化と同様のタイミングで問題ありません。
　（後々、プレイヤーが切り替わった際には、そのタイミングで改めてプレイヤーが誰なのか書き換えます。）

　ここまでできたら、動作確認してみましょう。
　「ゲームスタート」ボタンを押して、セル B7 に「今はプレイヤー1 です。」と表示されれば OK です。



６－５　【処理 K】プレイヤーごとにセルの背景色を設定・表示

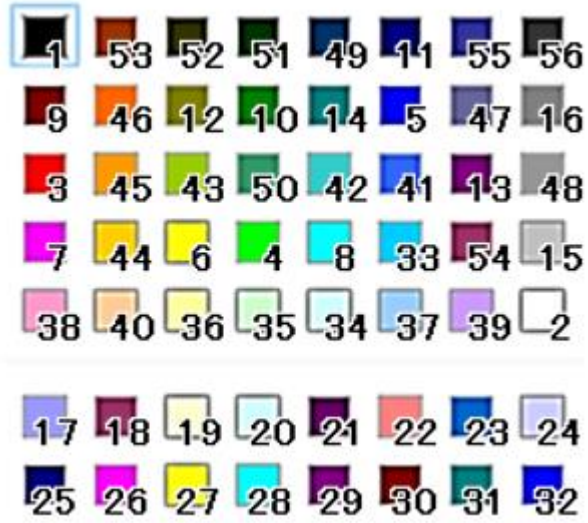
６－５－１　カラーインデックスで色の指定

セルをある色で塗りつぶす場合、Cells または Range で特定セル・セル範囲に対してカラーインデックスを指定する方法があります。

（セル範囲）.Interior.ColorIndex = （カラーインデックス番号）

という書式になります。

カラーインデックスは下図のようにあらかじめ用意されています。カラーインデックス番号を指定するだけでその色を使うことができるので、大変手軽に使える色指定方法です。



「【VBA 入門】ColorIndex の使い方と色見本一覧(色番号、RGB)」より引用
https://www.sejuku.net/blog/32288

34～40 の７つ連続が淡い色でいい感じですので、プレイヤー7 人にそれぞれ割り当てます。

プレイヤー番号	ColorIndex
1	34
2	35
3	36
4	37
5	38
6	39
7	40

33 にプレイヤー番号を足すことで、使いたい ColorIndex 番号になります。

プログラムで表現する場合、33 を ColorIndex の底上げ値（定数）的な扱いとして、グローバル変数に

Dim COLOR_INDEX_ADD As Integer　…(処理 K-1)

宣言をしましょう。
その上で、

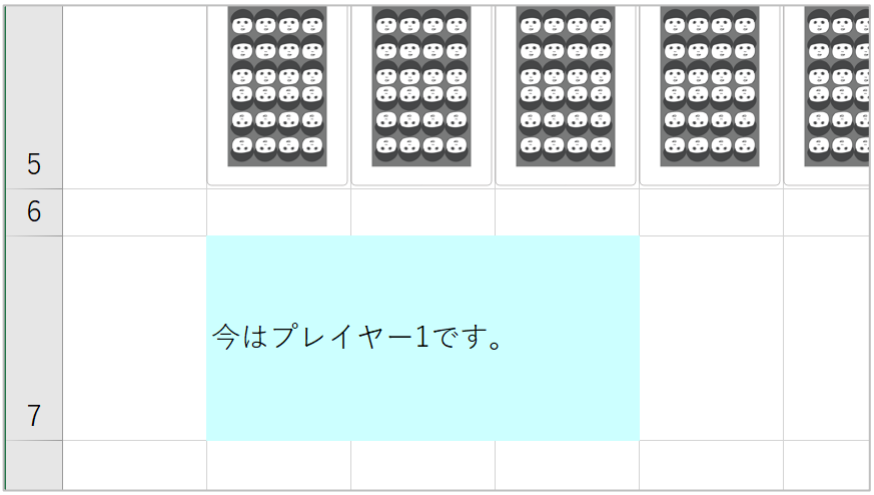
COLOR_INDEX_ADD = 33　…(処理 K-2)
と初期化し、

playerNum + COLOR_INDEX_ADD　…(処理 K-3)

で使いたい ColorIndex を表現できます。

【練習６－５】
処理 K-1, K-2, K-3 の命令を追加してください。
処理 K-3 は、「今はプレイヤー１です。」と表示するセル B7 を含めて B7～D7 のセルの塗りつぶしとして ColorIndex を指定してください。

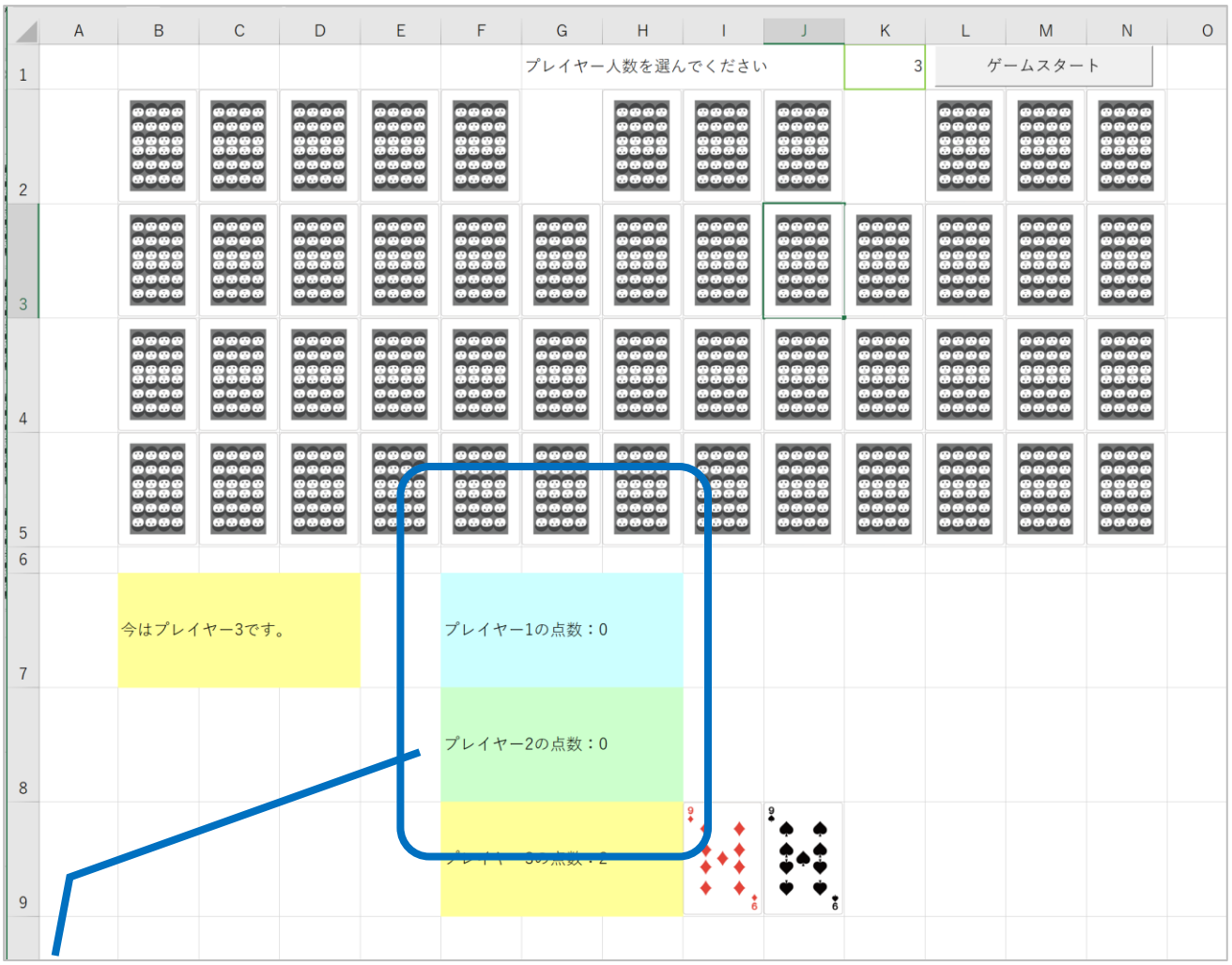
ここまでできたら動作確認してみましょう。
ColorIndex34 の水色っぽい色でセル B7～D7 が塗りつぶされれば OK です。



６－６　【処理Ｌ】各プレイヤーの得点表示を用意

６－６－１　概要

この節でやることの説明です。



このような感じで、各プレイヤーの得点表示機能をつくりましょう。

処理Ｌの内容をさらに具体的に細かく分解すると次のようになります。

【L-1】得点表示はプレイヤーの人数分。２名なら２名分、７名なら７名分。

【L-2】各プレイヤーごとにセルを塗りつぶす。色は６－５のとおり、IndexColor で指定。

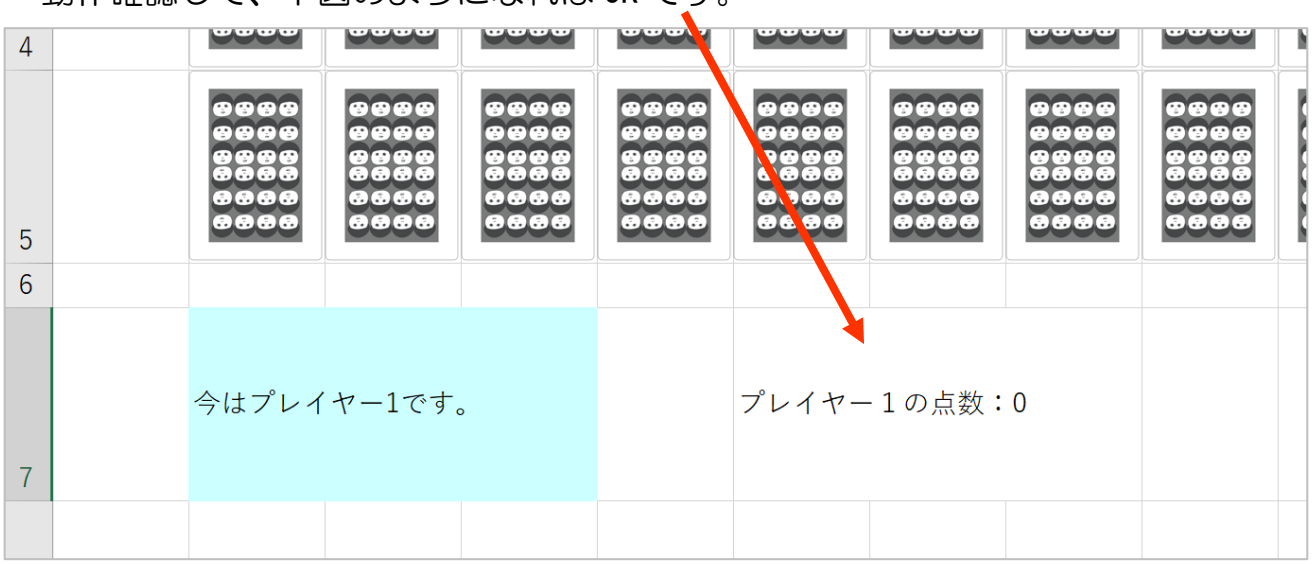
次節から手順に沿って作っていきます。

６－６－２　【処理 L-１】プレイヤーごとの得点表示欄をつくる

まず最初の手順として、プレイヤー１のみ作ってみましょう。

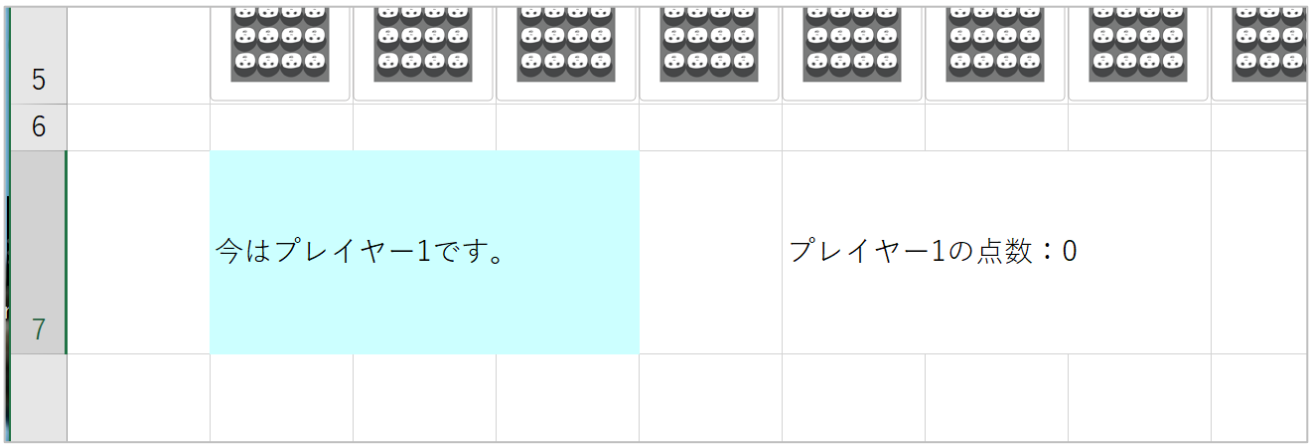
【練習６－６－２①】
セル F7 に、「プレイヤー１の点数：0」と表示される命令を追加してください。
「0」は半角にしてください。（得点は数値のため）
タイミングとしては「ゲームスタート」ボタンが押されたときです。

動作確認して、下図のようになれば OK です。



【練習６－６－２②】
変数ｉを使って、i=1 とし、「プレイヤー１の点数：0」の部分をｉを使った表記に書き換えてください。
文字列と変数の連結を上手に使ってください。

実行結果はほぼ変わりません。



【練習６－６－２③】
　プレイヤー人数分、８行目、９行目、・・・に表示できるように書き換えていきましょう。
　プレイヤー人数は、変数 player に格納されています。
　For ループで、1～player まで i の値を変化させて、下図のように表示する仕組みを作ってください。

←プレイヤー人数を 6 に設定した場合

　プレイヤー人数を 2 に設定すればプレイヤー2 まで、7 に設定すればプレイヤー7 まで表示します。

６－６－３　【処理 L-2】各プレイヤーごとにセルを塗りつぶす

【練習６－６－３①】

　前練習で作った For ループ内に、各プレイヤーごとの色でセルを塗りつぶす命令を追加してください。

　練習６－５にて、プレイヤーごとの ColorIndex での色指定をしたことを参考にしてみましょう。

　まずは F 列のみ、塗りつぶしてください。

　7 人だと右図のようになります。

【練習６－６－３②】

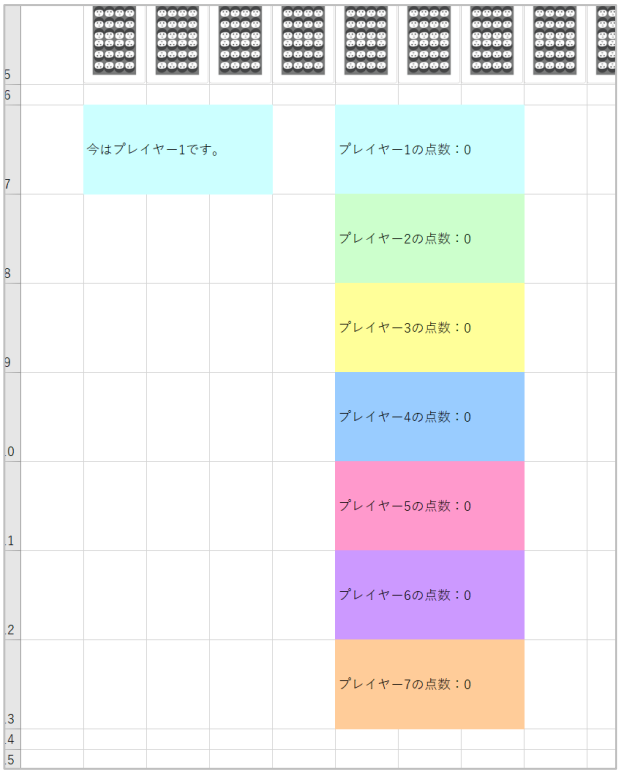
　G 列、H 列も塗りつぶした方が見栄えがいいので、塗りつぶしましょう。

　塗りつぶしの命令の行

```
Cells(6 + i, 6).Interior.ColorIndex = i + COLOR_INDEX_ADD
```

をさらに j を使った For ループに入れて、j で列を F（6 列目）～H（8 列目）と変化させてください。

　動作確認して右図のようになれば OK です。



６－７　得点表示欄の初期化

現段階では、例えばプレイヤーが 7 人だったあとに 2 人で新しいゲームを始めた場合、プレイヤー７までの得点欄が表示されたままになってしまっています。
ですので、ゲーム開始時にセル F7～H13 の領域のデータと塗りつぶし色をなくしましょう。

６－７－１　データの消去　ClearContents

次の命令を追加してください。
COLOR_INDEX_ADD = 33 の後あたりで良いです。

```
Range("F7:H13").ClearContents
```

６－７－２　セルの塗りつぶしを無しにする

セルの塗りつぶしを無しにするのも、ColorIndex でできます。０を指定します。

```
Range("F7:H13").Interior.ColorIndex = 0
```

を、データ消去に引き続いて追加してください。

```
    playerNum = 1                                プレイヤー番号を１で初期化
    COLOR_INDEX_ADD = 33            'カラーインデックス底上げ値を33に指定
    Range("F7:H13").ClearContents
    Range("F7:H13").Interior.ColorIndex = 0
```

プレイヤー人数を変えて動作確認してみましょう。
得点表示欄が人数分だけ表示されれば OK です。

第７章　めくられた２枚が違うナンバーのときの処理【処理 F】【処理 G】

動作確認のしやすさを考慮して、処理 E よりも先に処理 F を作ります。
ついでに、処理 G も作っておきます。

これらの処理は、カードが 2 枚めくられたときですので、cardClick 関数内に追加します。

５－４－１の②で target1 と target2 が同じかどうかの条件分岐を作りました。（そのときの図をもう一度示します。）この図で「処理 F」の部分になります。
（処理 G については後述します。）

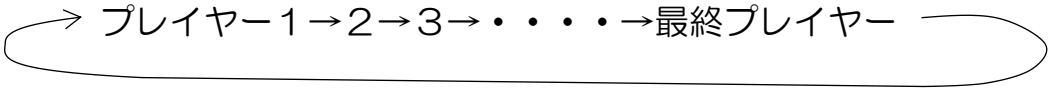
```
    Case 2
        target2 = Mid(Application.Caller, 2, 1)

        If target1 = target2 Then
                処理 E
        Else
                処理 F
        End If

    End Select
```

処理の流れを次のようにします。
【処理 F-0】プレイヤー番号を次のプレイヤーに変更
【処理 F-1】「残念！次は、プレイヤー（番号）です」とメッセージボックスを表示
【処理 F-2】開かれた２枚のカードの画像を、裏の画像に変更

ひとつひとつ、作っていきましょう。
すべて、処理 F の場所に追加します。

７－１　【処理 F-O】プレイヤー番号を次のプレイヤーに変更

【練習７－１－１】
プレイヤーは、



という順番で回ります。

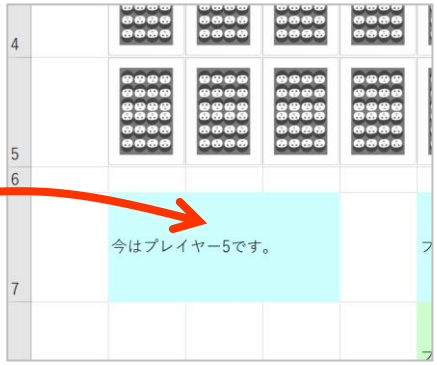
- 現プレイヤーが最終プレイヤーだったら、次のプレイヤーはプレイヤー１
- 現プレイヤーが最終プレイヤーじゃなかったら、プレイヤー番号が１増える

という感じです。
playerNum をこの条件で切り替える命令を追加してください。

（ヒント）
もし、playerNum が最終プレイヤー番号と一致するなら
playerNum を１にする
そうでなければ
playerNum を１増やす

【練習７－１－２】
セル B7 を、切り替わった新しいプレイヤー番号で書き換える命令を追加してください。
（ヒント）練習６－４－３でやっていますので、参考にしてみましょう。

問題７－１－１、７－１－２の両方ができたら、動作確認してみましょう。
開いた２枚のカードが一致しない場合、ここが次のプレイヤー番号（最終プレイヤーの次は 1）に変われば OK です。



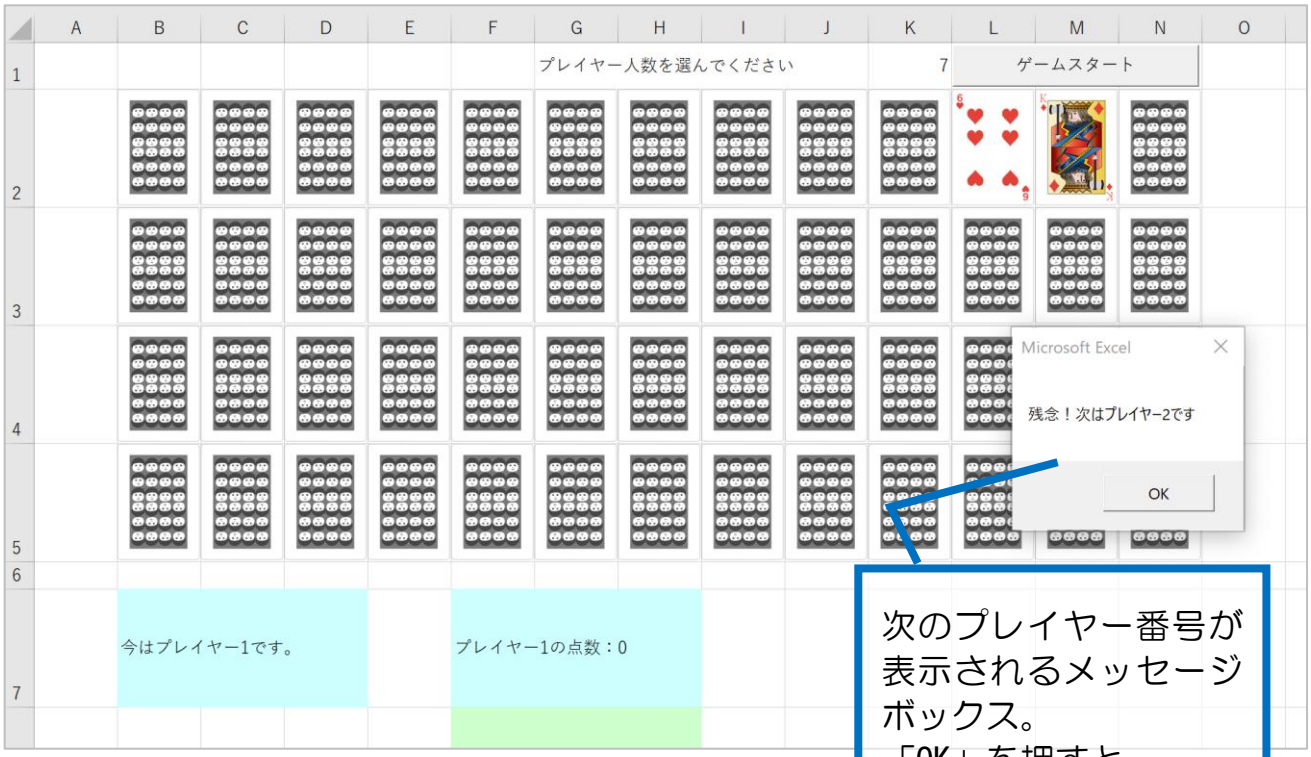
【練習７－１－３】
セル B7 をプレイヤー番号固有の色で塗りつぶしてください。
（ヒント）

- 練習６－６－３①でやっていますので、参考にしてみましょう。
- Range でセル範囲指定するとよいです。
- 各変数はこの場面に合うように。

動作確認で、右図のようにプレイヤーごとに塗りつぶし色が変われば OK です。

７－２　【処理 F-1】「残念！次は、プレイヤー（番号）です」とメッセージボックスを表示

【練習７－２】
　セル B7 の「今はプレイヤー（番号）です。」表示が変わる前に、「残念！次は、プレイヤー（番号）です」とメッセージボックスを表示してください。



…という動きになります。

７－３　【処理 F-2】開かれた２枚のカードの画像を、裏の画像に変更

７－３－１　クリックされた１枚目を裏の画像にする

処理 C（５－１：処理の流れ参照）にて、

●2 枚目がめくられたときに不一致の場合、裏面画像に差し替えるため画像の位置（行・列）とインデックス番号を保持

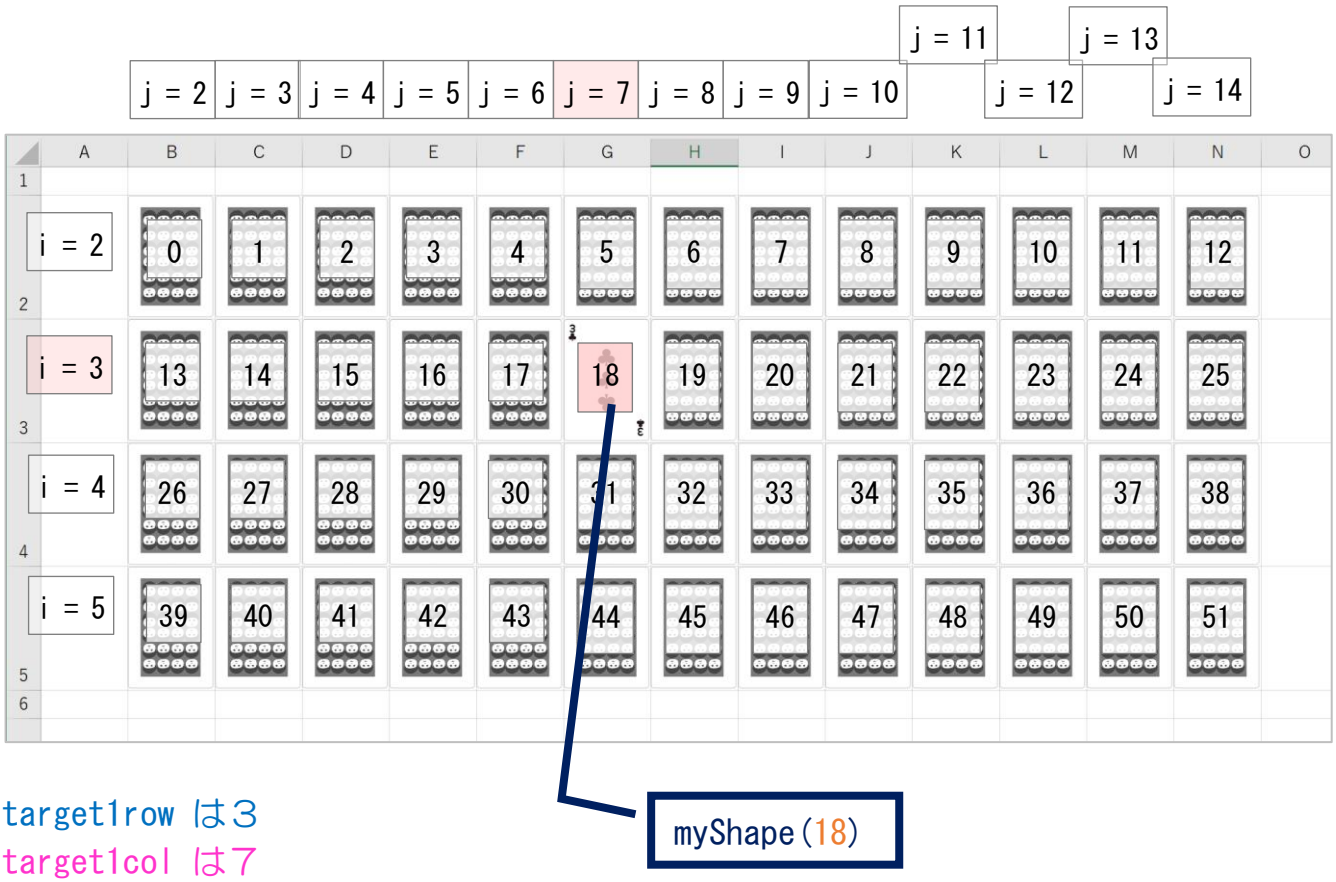
ということをしています。これは１枚目のカードの位置情報になります。
復習がてら、もう一度整理しましょう。

１枚目の行・列・インデックス番号を扱う変数は次のように定義されています。

```
Dim list() As Variant   'カードの名前リスト
Dim openNum As Integer      '開かれているカード枚数
Dim target1row As Integer     '1枚目の行番号を保持
Dim target1col As Integer     '1枚目の列番号を保持
Dim target1 As String         '1枚目のカードナンバー
Dim index1 As Integer         '1枚目のカードのリスト内インデックスを保持
```

行　列　インデックス番号

例として、クリックされた１枚目がセル G3 の位置にある画像だとしましょう。



target1row は３
target1col は７
index1 は 18

という情報が保持されます。（詳しくは５－３－２を参照ください。）

この情報があれば裏面の画像に差し替えるのは簡単で、次の手順となります。

A) 今ある画像を消去

B) 消去された位置に、裏面画像をセット。

【練習７－３－１①】

手順 B)を行うには、裏面画像のパス・ファイル名を取得を事前に行っておく必要があります。

そこで前準備として、変数 picName に裏画像のパス・ファイル名をセットしてください。

（ヒント）

「１－１－２　表示したい画像ファイルを指定」を参照してください。

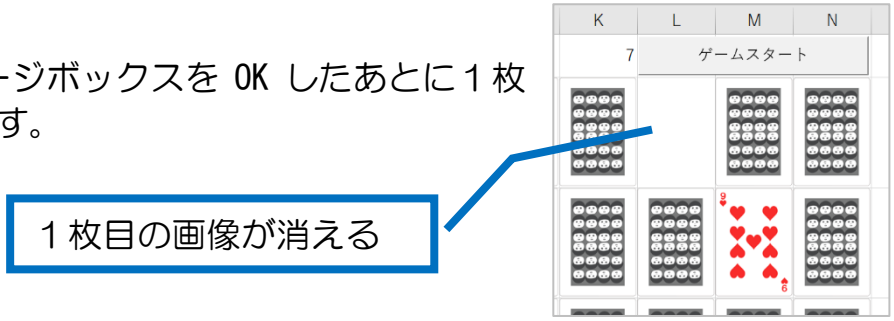
【練習７－３－１②】

クリックされた１枚目の画像を消去する命令を追加してください。

（ヒント）

「４－６　クリックされた画像を消去」を参照してください。

動作確認では、メッセージボックスを OK したあとに１枚目の画像が消えれば OK です。



【練習７－３－１③】

消去された位置に裏の画像を表示してください。

（ヒント）

picSet 関数を使って画像を表示できます。

「４－７　めくられたときの画像ファイルを表示する」を参考にしましょう。

picSet 関数には、４つの引数を渡す必要があります

- 画像ファイルのパス・ファイル名（練習７－３－１で取得済）
- 表示したい行番号
- 表示したい列番号
- カードのインデックス番号

} 処理 C にて取得済

動作確認して、メッセージボックスを OK したあとに１枚目の画像が裏面の画像になれば OK です。

７－３－２　クリックされた２枚目を裏の画像にする

２枚目に関しても、そのカードの位置情報（行番号・列番号）、インデックス番号がわかれば１枚目と同じ手順

A） 今ある画像を消去

B） 消去された位置に、裏面画像をセット。

で裏面画像に切り替えることができます。

【練習７－３－２①】

クリックされた２枚目の、（A）行番号、（B）列番号、（C）インデックス番号、を示す変数はそれぞれ何でしょうか？

【練習７－３－２②】

前練習で答えた変数と picName を用いて、クリックされた２枚目の画像を裏面画像に切り替えてください。

ここまでできると、１枚目と２枚目が違ったら裏面画像に戻り、次のプレイヤーに切り替わります。

ですが、次にめくると延々とめくり続けるだけになってしまっています。

これは、開かれているカード枚数（openNum）が増え続けていっているためです。

次節の処理を追加しましょう。

７－４　【処理 G】開かれているカード枚数を０にリセット

【練習７－４】

開かれているカード枚数を０にリセットする命令を追加してください。

（ヒント）

追加場所は、2 枚目のカードが開かれて一連の処理が終わった後。「５－１　処理の流れ」のフローチャートを参照。

開かれているカード枚数を示す変数は？

ここまでできると、２枚めくって不正解だったらカードが裏面画像になり、また 2 枚めくって不正解なら裏面画像になり、・・・・が繰り返されます。

いきぬきのひととき

第８章　めくられた２枚が同じナンバーのときの処理【処理 E】

８－１　【事前準備】各プレイヤーの得点

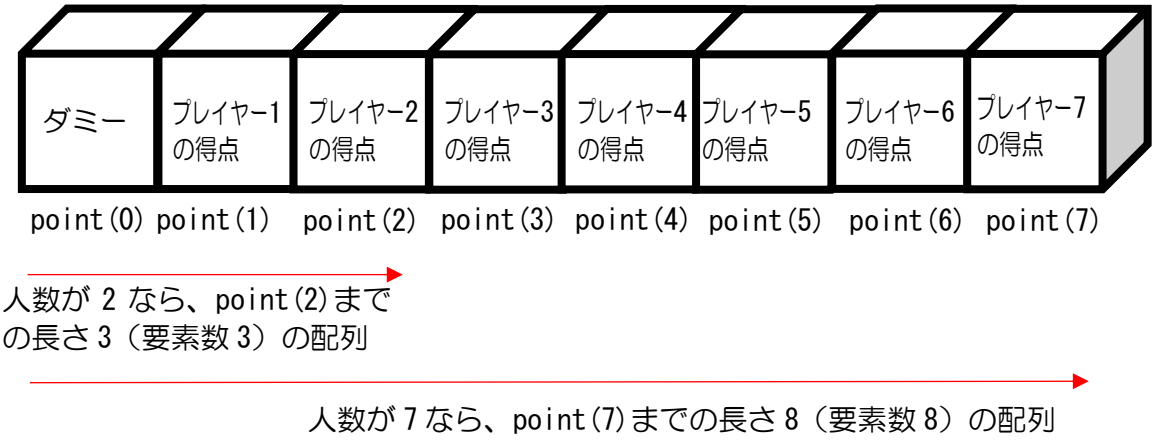
８－１－１　各プレイヤーの得点を格納する配列を用意する

グローバル変数で、整数型の配列 point() を宣言・定義しましょう。

```
Dim player As Integer                プレイヤー人数
Dim playerNum As Integer             'プレイヤー番号
Dim point() As Integer               '各プレイヤーの得点

Dim COLOR_INDEX_ADD As Integer       'カラーインデックス底上げ値
```

これを追加

配列の要素数はこの段階では指定しません。プレイヤー人数が入力されて「ゲームスタート」ボタンが押されたタイミングで配列の要素数を決定します。（次節）



８－１－２　配列 point() の要素数を指定する

配列の要素数を改めて設定するには、次のようにします。

ReDim 配列名(最終インデックス番号)

【練習７－２－２】
point() の要素数を設定してください。最終インデックス番号は変数 player です。
（ヒント）
タイミングとしては、player の値が設定された後です。
各種変数等を初期化しているタイミングが適当です。

８－１－３　各プレイヤーの得点を 0 で初期化

【練習７－２－３】
For ループを使って、各プレイヤーの得点を 0 で初期化してください。
（ヒント）
変数 i を 0 から配列 point()の最終インデックス番号（player）まで変化させます。
point(i)に 0 を代入してください。

８－２　【処理 E-１】カード２枚ゲットで２点追加

この処理は、カードが 2 枚めくられたときですので、cardClick 関数内に追加します。

５－４－１の②で target1 と target2 が同じかどうかの条件分岐を作りました。（そのときの図をもう一度示します。）この図で「処理 E」の部分になります。

```
    Case 2
        target2 = Mid(Application.Caller, 2, 1)

        If target1 = target2 Then
                処理 E
        Else
                処理 F
        End If

    End Select
```

処理の流れを次のようにします。
A) 「おめでとう！２枚ゲット」メッセージボックスを表示
B) 各プレイヤーに点数加算
C) 加算後の点数表示

ひとつひとつ、作っていきましょう。
すべて、処理 E の場所に追加します。

８－２－１　「おめでとう！２枚ゲット」メッセージボックスを表示

【練習８－２－１】
「おめでとう！２枚ゲット」とメッセージボックスを表示する仕組みを追加してください。

　動作確認してみましょう。
　正解の時に「おめでとう！～」表示が出れば OK です。

８－２－２　各プレイヤーに点数加算

　各プレイヤーの得点は、配列 point() です。８－３－１で、各プレイヤーの得点は０に初期化しています。
【練習８－２－２】
　正解の場合、現プレイヤーの得点に 2 を足してください。

（ヒント）
現プレイヤーのプレイヤー番号は playerNum ですので、point(playerNum) が現プレイヤーの得点です。

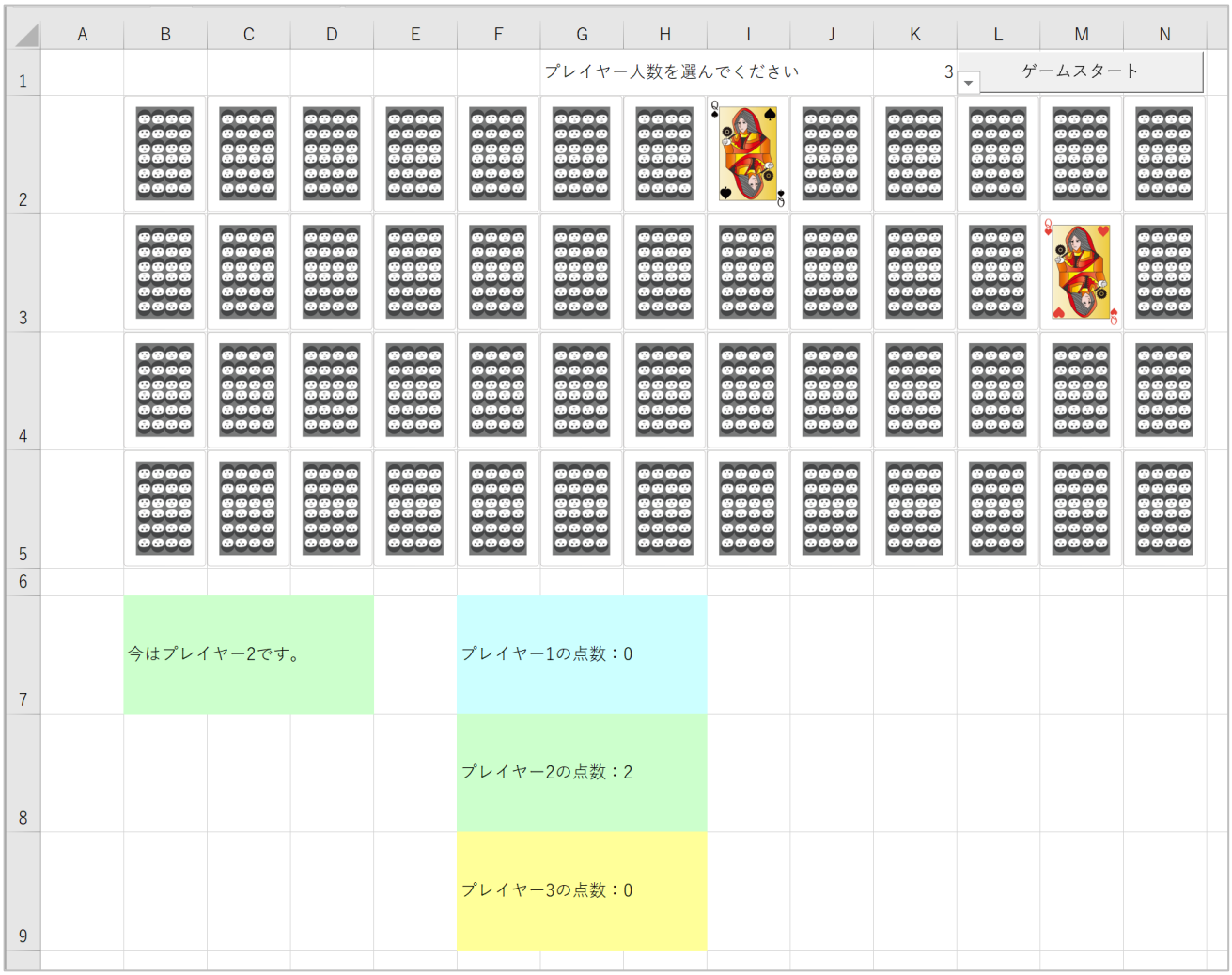
（動作確認は、８－２－３までやってからにしましょう）

８－２－３　加算後の点数表示

【練習８－２－３】
　F 列の各プレイヤーの得点表示部を得点加算後の値に書き換える命令を追加してください。

（ヒント）
　練習６－６－２③を参考にしてみましょう。

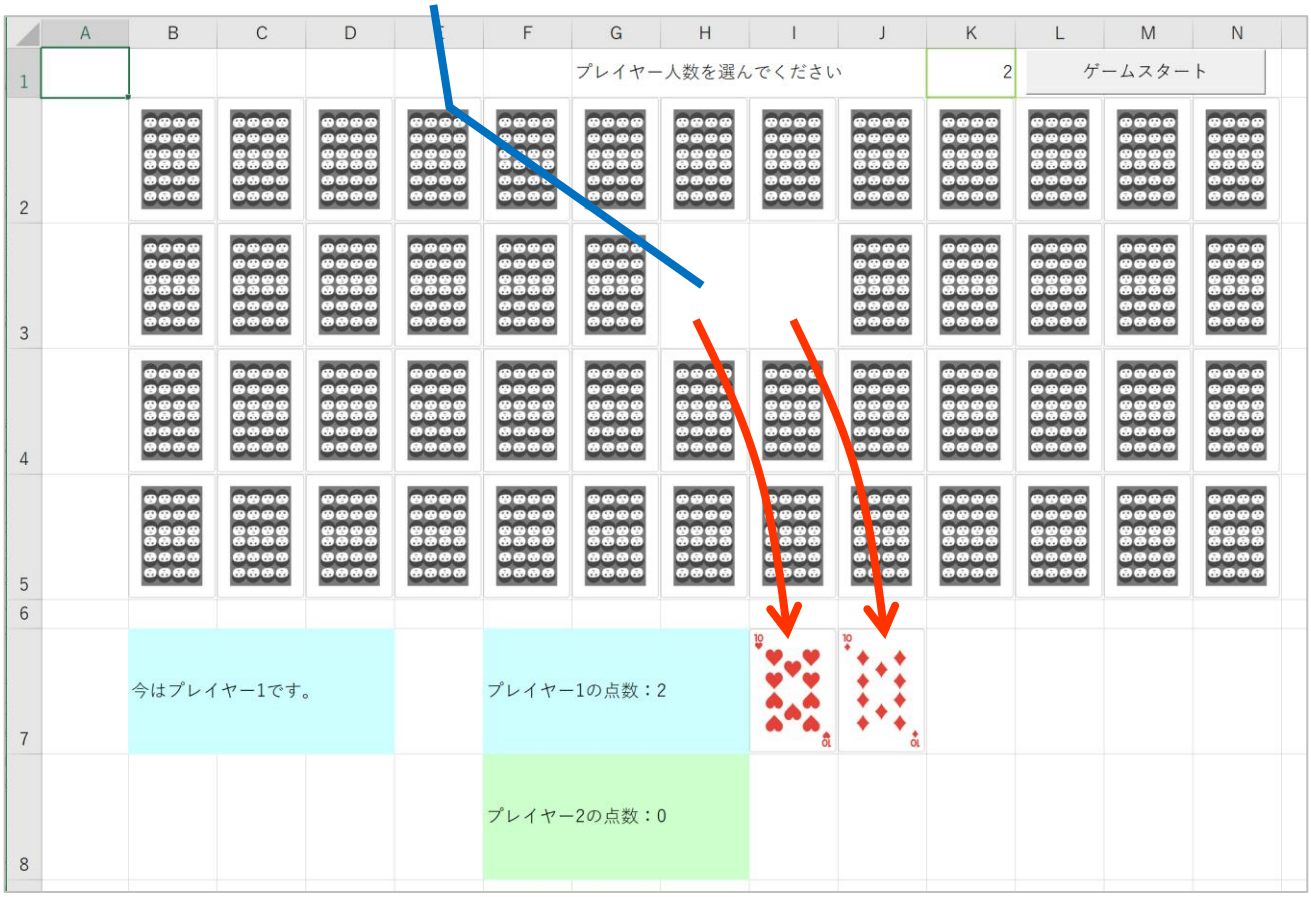
　ここまでできたら動作確認してください。
　正解の時にそのときのプレイヤーの得点が加算されて表示されれば OK です。



８－３　【処理 E-2】正解のカードを得点エリアに並べる

８－３－１　カードの移動先

正解の２枚のカードを、正解したプレイヤーの得点表示部に移動します。



カードの移動先の行・列は次のように整理できます。

例	一般化
プレイヤー1 で得点が 2 点になった場合 カード１枚目の移動先は 7 行目・9 列目 カード２枚目の移動先は 7 行目・10 列目	プレイヤーplayerNum で得点が point(playerNum)点になった場合
7 = 6 + 1 行目 …変数 trow 2 枚目　10 =（8 + 2）列目 …変数 tcol 1 枚目　9 =（tcol - 1）列目	trow = 6 + playerNum 行目　…(a) 2 枚目 tcol = 8 + point(playerNum) 列目 …(b) 1 枚目 tcol - 1 列目

【練習８－３－１】
変数 trow と tcol を整数型で cardClick 関数に宣言し、(a)と(b)の式を追加してください。

８－３－２　画像の表示位置

カード画像は、Shape 型オブジェクト myShape 配列です。
このオブジェクトのパラメータである Left と Top で表示位置を指定できます。
（詳しくは１－１－３をご参照ください。）

1 枚目のインデックス番号は？　index1
2 枚目のインデックス番号は？　cardNo

カードの移動先は、前節にて整理した trow と tcol を使って指定できます。

1 枚目を移動してみましょう。
まずは横位置について、次のように命令を追加してください。

```
'画像の移動
trow = 6 + playerNum
tcol = 8 + point(playerNum)
myShape(index1).Left = Cells(trow, tcol - 1).Left   '1枚目の移動先・横位置
```

これを追加

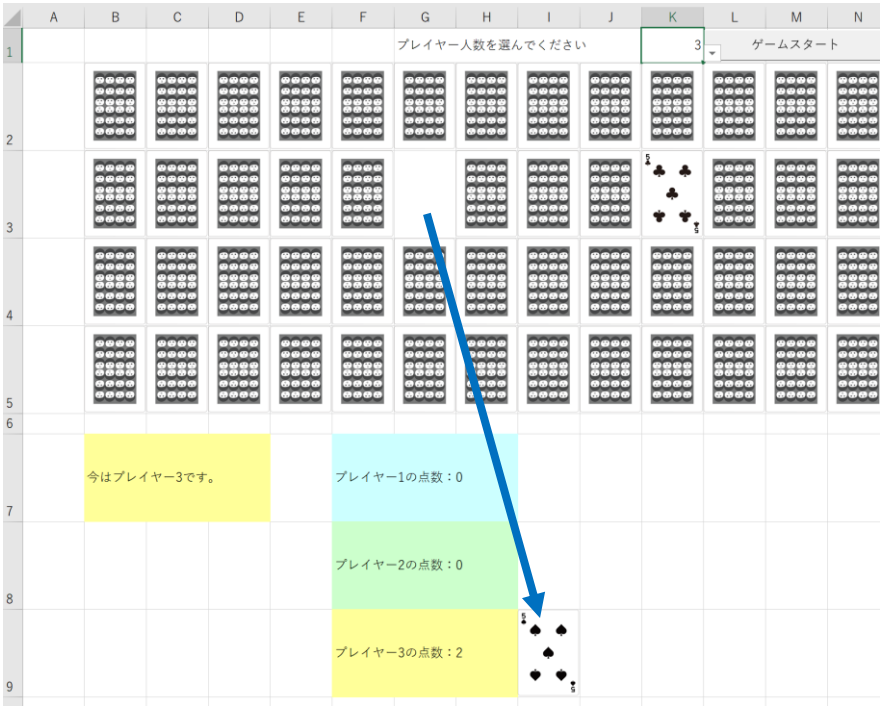
1 枚目のインデックス番号

横位置

セル指定

【練習８－３－２①】
1 枚目の移動先の縦方向位置を指定する命令を追加してください。

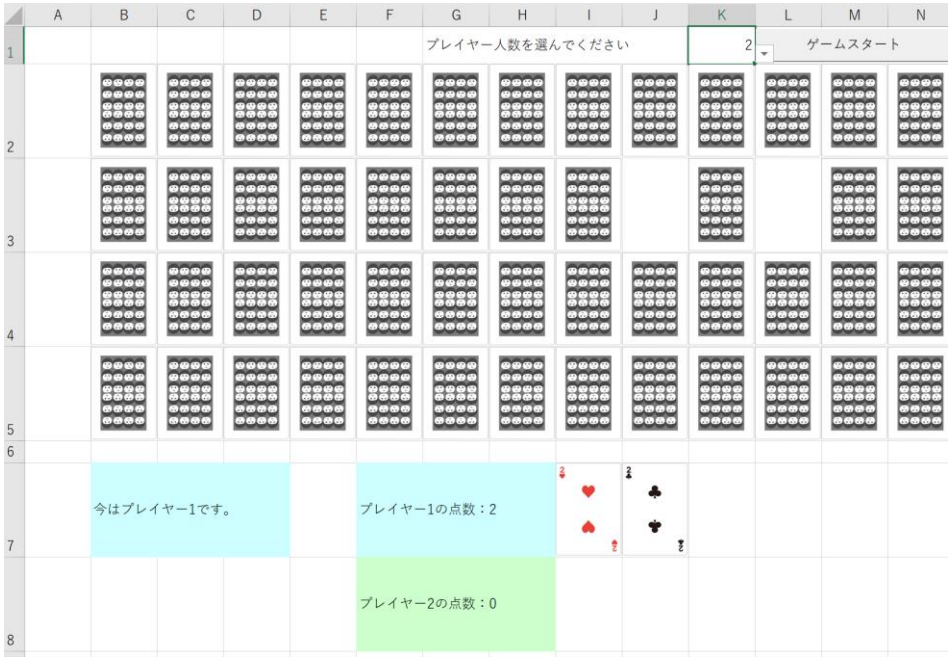
動作確認して、1 枚目のカードがそのときのプレイヤーの得点領域に移動すれば OK です。



【練習８－３－２②】

2 枚目の移動先を指定する命令を追加してください。

縦位置・横位置の両方ともになりますので、2 行追加になります。



これでほぼ完成に近いのですが、次のような考え落ちがあります。

考え落ち①　開かれた画像をもう一回クリックすると、正解になってしまう。

考え落ち②　得点エリアに移動した画像をクリックすると、元の位置に戻ってしまう。

最後にこれらを修正して仕上げとしましょう。

第９章　仕上げ

９－１　考え落ち①　開かれた画像をもう一回クリックすると、正解になってしまうのを防ぐ

９－１－１　インデックス番号の比較

2 枚目クリックのときに 1 枚目のカードそのものをクリックするということは、同じインデックス番号のカードをクリックしていることになります。

それを判定するには、1 枚目のインデックス番号（index1）と 2 枚目（cardNo）のインデックス番号が同じかどうかを見ると良いです。

違う場合のみ、Select Case 構文内の openNum が２の場合の処理を行うようにしましょう。

index1 と cardNo が違う場合のみ、case 2 の処理に入るようにするには、
IF index1 <> cardNo Then～
End IF
の中にこの処理を入れると良い。

```
Select Case openNum
   Case 1
      target1row = i
      target1col = j
      index1 = cardNo
      target1 = Mid(Application.Caller, 2, 1)

   Case 2
      target2 = Mid(Application.Caller, 2, 1)

      If target1 = target2 Then
         MsgBox "おめでとう！２枚ゲット"
         point(playerNum) = point(playerNum) + 2
         Cells(6 + playerNum, 6).Value = "プレイヤー" & playerNum & "の点数：" & point(playerNum)

         '画像の移動
         trow = 6 + playerNum
         tcol = 8 + point(playerNum)
         myShape(index1).Left = Cells(trow, tcol - 1).Left   '1枚目の移動先・横位置
         myShape(index1).Top = Cells(trow, tcol - 1).Top     '1枚目の移動先・縦位置
         myShape(cardNo).Left = Cells(trow, tcol).Left       '2枚目の移動先・横位置
         myShape(cardNo).Top = Cells(trow, tcol).Top         '2枚目の移動先・縦位置

      Else
         If playerNum = player Then
            playerNum = 1
         Else
            playerNum = playerNum + 1
         End If

         MsgBox "残念！次はプレイヤー" & playerNum & "です"

         Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
         Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

         picName = ActiveWorkbook.Path & "¥card¥ura.png"

         myShape(index1).Delete
         Call picSet(picName, target1row, target1col, index1)

         myShape(cardNo).Delete
         Call picSet(picName, i, j, cardNo)


      End If


      openNum = 0
End Select
```

```
    Case 2
      If index1 <> cardNo Then       '1枚目と2枚目で違うものがクリックされた場合
        target2 = Mid(Application.Caller, 2, 1)

        If target1 = target2 Then
          MsgBox "おめでとう！２枚ゲット"
          point(playerNum) = point(playerNum) + 2
          Cells(6 + playerNum, 6).Value = "プレイヤー" & playerNum & "の点数：" & point(playerNum)

          '画像の移動
          trow = 6 + playerNum
          tcol = 8 + point(playerNum)
          myShape(index1).Left = Cells(trow, tcol - 1).Left   '1枚目の移動先・横位置
          myShape(index1).Top = Cells(trow, tcol - 1).Top     '1枚目の移動先・縦位置
          myShape(cardNo).Left = Cells(trow, tcol).Left       '2枚目の移動先・横位置
          myShape(cardNo).Top = Cells(trow, tcol).Top         '2枚目の移動先・縦位置

        Else
          If playerNum = player Then
            playerNum = 1
          Else
            playerNum = playerNum + 1
          End If

          MsgBox "残念！次はプレイヤー" & playerNum & "です"

          Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
          Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

          picName = ActiveWorkbook.Path & "¥card¥ura.png"

          myShape(index1).Delete
          Call picSet(picName, target1row, target1col, index1)

          myShape(cardNo).Delete
          Call picSet(picName, i, j, cardNo)


        End If


        openNum = 0
      End If        '1枚目と2枚目で違うものがクリックされた場合のIFの終り
    End Select
```

他にも End If があるので、どの If に対応するものなのか、分かりやすくコメントを書いておくことをお勧めします。

これで動作確認してみましょう。

「あれ？？」と思うことがあると思います。

1 枚目と同じカードをもう一回クリックしたとき、得点は入らなくなりましたが・・・

９－１－２　開かれ続けてしまう現象の修正

同じカードを 2 回クリックしたあと、他のカードをクリックしても果てしなく開かれてしまい、正解・不正解の判定がされません。
判定に入るには、openNum が２である必要があります。（今まさに IF で囲んだ部分）
判定に入らないということは、openNum が２になっていない可能性が疑われます。

詳しく見ていきましょう。カードがクリックされると、cardClick 関数が呼ばれます。

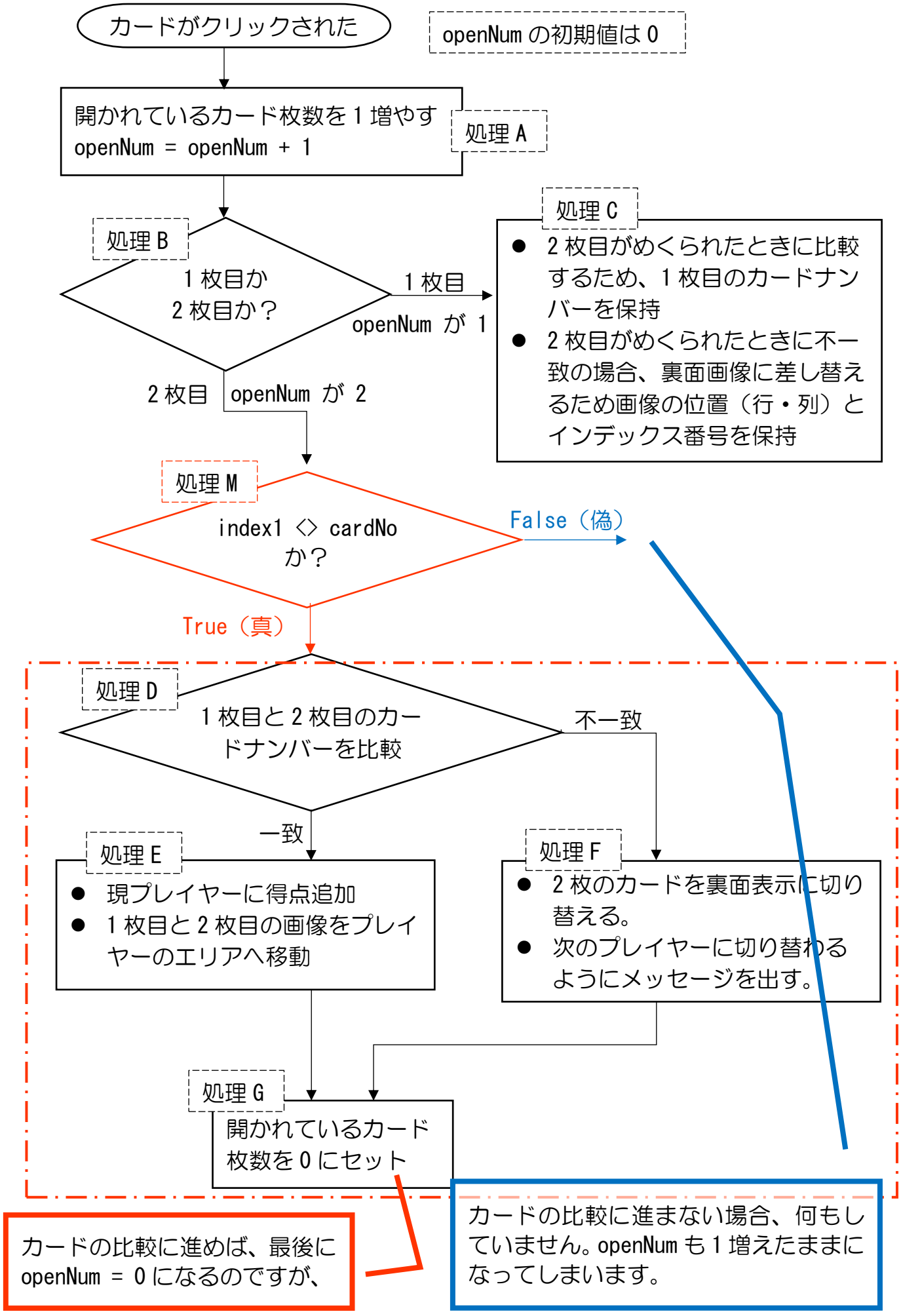
```
Function cardClick(cardNo)
    Dim i As Integer
    Dim j As Integer
    Dim trow As Integer
    Dim tcol As Integer

    myShape(cardNo).Delete

    picName = ActiveWorkbook.Path & "¥card¥" & Application.Caller
    i = Int(cardNo / 13) + 2
    j = cardNo - 13 * Int(cardNo / 13) + 2

    Call picSet(picName, i, j, cardNo)

    openNum = openNum + 1

    Select Case openNum
        Case 1
            target1row = i
            target1col = j
            index1 = cardNo
```

cardClick 関数に入ると、この部分で openNum を 1 増やします。（処理 A）

次のページにフローチャート（処理の流れ）を示します。
赤い部分が９－１－１で追加された部分（処理 M とします）です。

処理 M で index1 <> cardNo が True で正解・不正解の判定処理に進めば、最終的に openNum が０にリセットされますが、False の場合は何も処理されません。同じものをクリックされた場合 1 枚しか開かれていないのに openNum が 2 のままとなってしまうのです。
さらに次にカードがクリックされると、また openNum が 1 増えて 3 になります。あとは増えていく一方なので、正解・不正解の判定処理にまったく入ることができません。

ですので、False のときに openNum を１にしてあげると良いです。この処理を Else で追加しましょう。


カードがクリックされた
openNum の初期値は 0
開かれているカード枚数を 1 増やす
openNum = openNum + 1
処理 A
処理 B
1 枚目か
2 枚目か？
1 枚目
openNum が 1
処理 C
● 2 枚目がめくられたときに比較するため、1 枚目のカードナンバーを保持
● 2 枚目がめくられたときに不一致の場合、裏面画像に差し替えるため画像の位置（行・列）とインデックス番号を保持
2 枚目
openNum が 2
処理 M
index1 ◇ cardNo
か？
False（偽）
True（真）
処理 D
1 枚目と 2 枚目のカードナンバーを比較
不一致
一致
処理 E
● 現プレイヤーに得点追加
● 1 枚目と 2 枚目の画像をプレイヤーのエリアへ移動
処理 F
● 2 枚のカードを裏面表示に切り替える。
● 次のプレイヤーに切り替わるようにメッセージを出す。
処理 G
開かれているカード枚数を 0 にセット
カードの比較に進めば、最後に openNum = 0 になるのですが、
カードの比較に進まない場合、何もしていません。openNum も 1 増えたままになってしまいます。

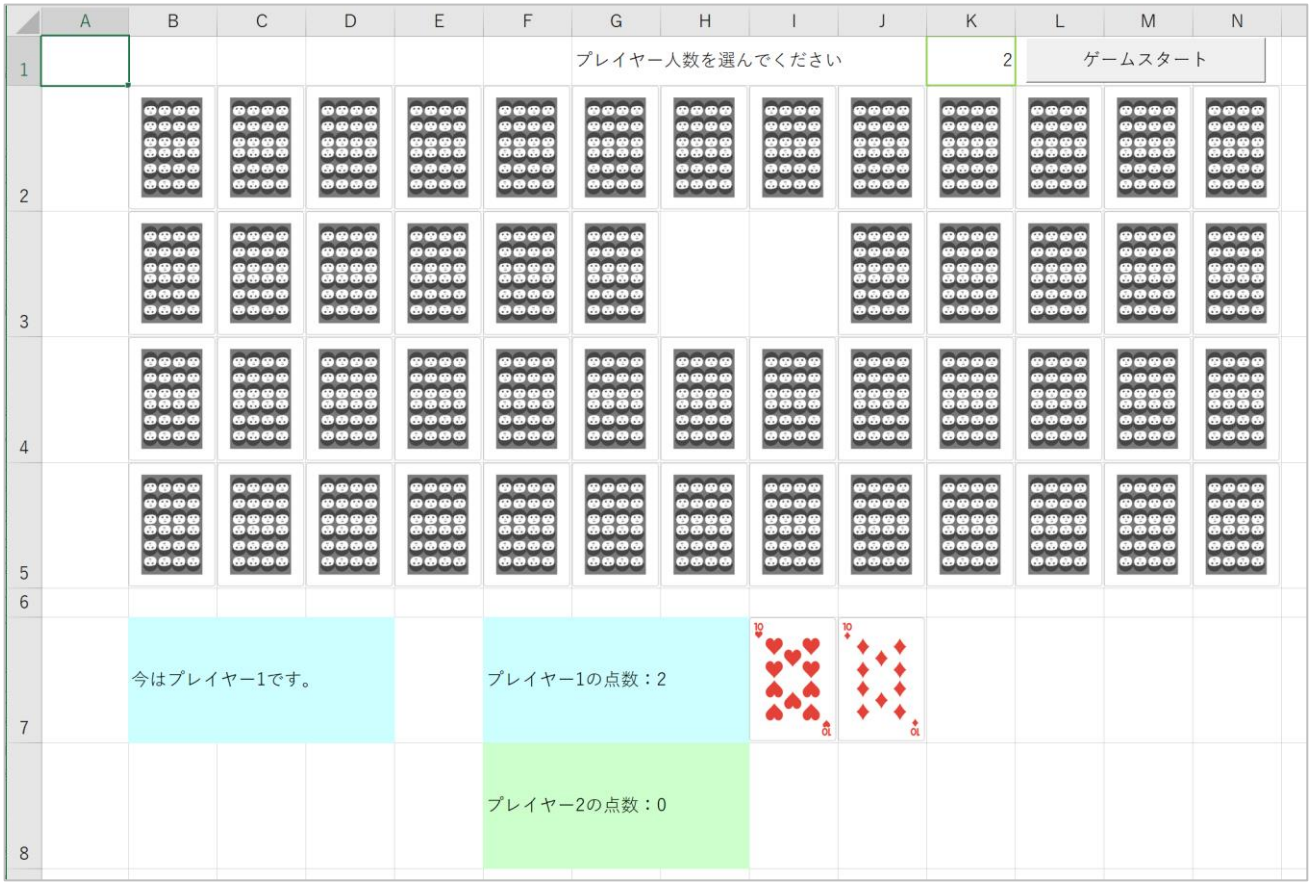
```
    Case 2
      If index1 <> cardNo Then       '1枚目と2枚目で違うものがクリックされた場合
        target2 = Mid(Application.Caller, 2, 1)

        If target1 = target2 Then
          MsgBox "おめでとう！２枚ゲット"
          point(playerNum) = point(playerNum) + 2
          Cells(6 + playerNum, 6).Value = "プレイヤー" & playerNum & "の点数：" & point(playerNum)

          '画像の移動
          trow = 6 + playerNum
          tcol = 8 + point(playerNum)
          myShape(index1).Left = Cells(trow, tcol - 1).Left   '1枚目の移動先・横位置
          myShape(index1).Top = Cells(trow, tcol - 1).Top     '1枚目の移動先・縦位置
          myShape(cardNo).Left = Cells(trow, tcol).Left       '2枚目の移動先・横位置
          myShape(cardNo).Top = Cells(trow, tcol).Top         '2枚目の移動先・縦位置

        Else
          If playerNum = player Then
            playerNum = 1
          Else
            playerNum = playerNum + 1
          End If

          MsgBox "残念！次はプレイヤー" & playerNum & "です"

          Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
          Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

          picName = ActiveWorkbook.Path & "¥card¥ura.png"

          myShape(index1).Delete
          Call picSet(picName, target1row, target1col, index1)

          myShape(cardNo).Delete
          Call picSet(picName, i, j, cardNo)


        End If


        openNum = 0
      Else
        openNum = 1
      End If        '1枚目と2枚目で違うものがクリックされた場合のIFの終り
    End Select
```

End If の前に、
Else
　openNum = 1
を追加してください。

動作確認してみましょう。
１枚目にクリックして開かれたカードを何回もクリックしたあと、別のカードをクリックしてみて、正解・不正解の判定が出れば OK です。

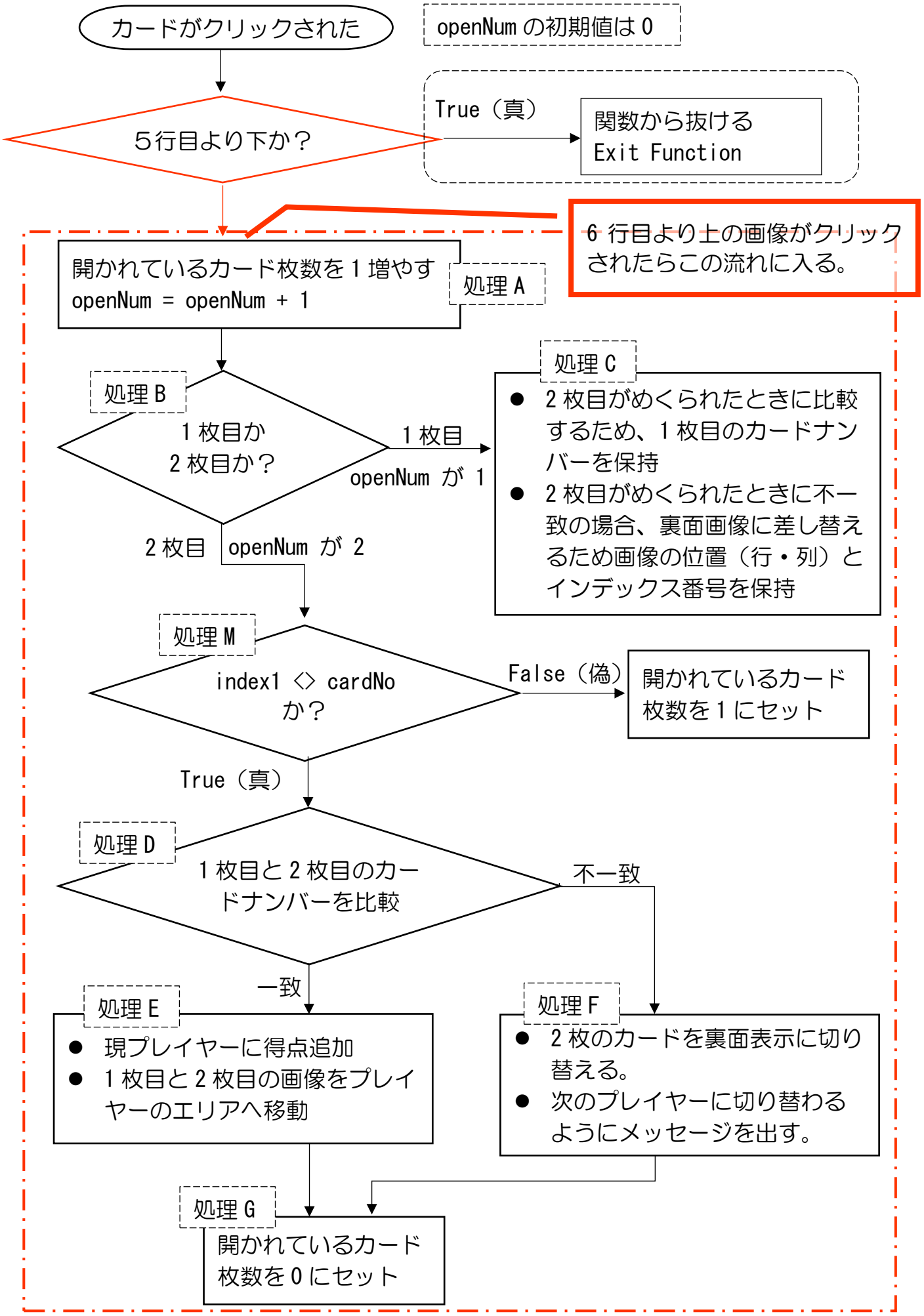
９－２　考え落ち②　得点エリアに移動した画像をクリックすると、元の位置に戻ってしまうのを防ぐ

９－２－１　考え方と準備



クリックされた画像が 6 行目より上にあるときのみ、cardClick 関数の中身が実行されるようにすればよいです。

次ページのフローチャートに示すように、cardClick 関数の中身が実行される前に、クリックされた画像が 6 行目より上にあるかどうかの判定を挟みましょう。


カードがクリックされた
openNum の初期値は 0
5行目より下か？
True（真）
関数から抜ける
Exit Function
6 行目より上の画像がクリックされたらこの流れに入る。
開かれているカード枚数を 1 増やす
openNum = openNum + 1
処理 A
処理 B
1 枚目か
2 枚目か？
1 枚目
openNum が 1
処理 C
● 2 枚目がめくられたときに比較するため、1 枚目のカードナンバーを保持
● 2 枚目がめくられたときに不一致の場合、裏面画像に差し替えるため画像の位置（行・列）とインデックス番号を保持
2 枚目
openNum が 2
処理 M
index1 <> cardNo
か？
False（偽）
開かれているカード枚数を 1 にセット
True（真）
処理 D
1 枚目と 2 枚目のカードナンバーを比較
不一致
一致
処理 E
● 現プレイヤーに得点追加
● 1 枚目と 2 枚目の画像をプレイヤーのエリアへ移動
処理 F
● 2 枚のカードを裏面表示に切り替える。
● 次のプレイヤーに切り替わるようにメッセージを出す。
処理 G
開かれているカード枚数を 0 にセット

```
Function cardClick(cardNo)
    Dim i As Integer
    Dim j As Integer
    Dim trow As Integer
    Dim tcol As Integer
```

cardClick 関数の中身（変数宣言を除く）に入る前に、５行目より下がクリックされたかどうか判定する IF 構文を入れる。

```
    myShape(cardNo).Delete

    picName = ActiveWorkbook.Path & "¥card¥" & Application.Caller
    i = Int(cardNo / 13) + 2
    j = cardNo - 13 * Int(cardNo / 13) + 2

    Call picSet(picName, i, j, cardNo)

    openNum = openNum + 1

    Select Case openNum
        Case 1
            target1row = i
            target1col = j
            index1 = cardNo
            target1 = Mid(Application.Caller, 2, 1)

'           MsgBox target1

        Case 2
            If index1 <> cardNo Then        '1枚目と2枚目で違うものがクリックされた場合
                target2 = Mid(Application.Caller, 2, 1)

                If target1 = target2 Then
                    MsgBox "おめでとう！２枚ゲット"
                    point(playerNum) = point(playerNum) + 2

                    （中略）

                    myShape(index1).Delete
                    Call picSet(picName, target1row, target1col, index1)

                    myShape(cardNo).Delete
                    Call picSet(picName, i, j, cardNo)


                End If


                openNum = 0
            Else
                openNum = 1
            End If          '1枚目と2枚目で違うものがクリックされた場合のIFの終り
    End Select


End Function
```

具体的には、次のページから示します。

まず準備です。

```
Function cardClick(cardNo)
    Dim i As Integer
    Dim j As Integer
    Dim trow As Integer
    Dim tcol As Integer

    

    myShape(cardNo).Delete

    picName = ActiveWorkbook.Path & "¥card¥" & Application.Caller
    i = Int(cardNo / 13) + 2
    j = cardNo - 13 * Int(cardNo / 13) + 2

    Call picSet(picName, i, j, cardNo)

    openNum = openNum + 1

    Select Case openNum
        Case 1
            target1row = i
```

If 構文を入れるスペースを 1～2 行あけます。

９－２－２　クリックされたカードの縦方向位置（座標）

クリックされたカードは、cardClick 関数の中では myShape(cardNo)です。
このカードの縦方向位置（座標）は次のように表現できます。

myShape(cardNo).Top　……(1)

次に、5 行目の縦方向位置（座標）は次のように表現できます。

cells(5, 1).Top　……(2)

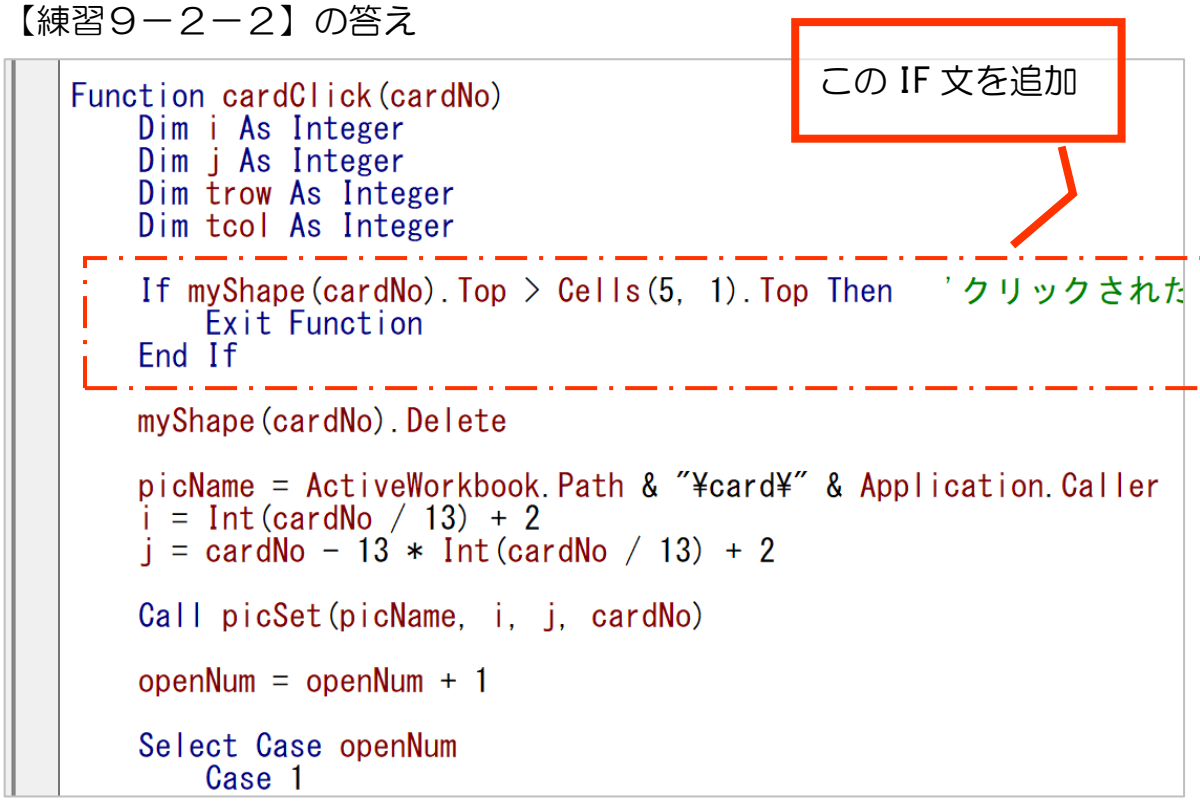
【練習９－２－２】
クリックされたカードが 5 行目より下の場合は cardClick 関数から抜けるように、上記(1)と(2)を使って IF 構文を作ってください。
関数から抜ける命令は、

Exit Function

の一文です。

答えは次のページにありますが、少し自力で考えてみましょう。

【練習９－２－２】の答え




```
Function cardClick(cardNo)
    Dim i As Integer
    Dim j As Integer
    Dim trow As Integer
    Dim tcol As Integer

    If myShape(cardNo).Top > Cells(5, 1).Top Then   'クリックされた
        Exit Function
    End If

    myShape(cardNo).Delete

    picName = ActiveWorkbook.Path & "¥card¥" & Application.Caller
    i = Int(cardNo / 13) + 2
    j = cardNo - 13 * Int(cardNo / 13) + 2

    Call picSet(picName, i, j, cardNo)

    openNum = openNum + 1

    Select Case openNum
        Case 1
```

ここまでできたら、動作確認してみましょう。

- 得点エリアに移動したカードをクリックしても何も変化が起きないこと
- 6 行目より上の裏返しのカードをクリックしたらめくられること

この 2 つが確認できれば OK です。

これで完成です！

おめでとうございます。おつかれさまでした。

練習の回答例

【練習２－１】

```
Cells(2, 1).Value = list(0)
Cells(3, 1).Value = list(1)
Cells(4, 1).Value = list(2)
Cells(5, 1).Value = list(3)
Cells(6, 1).Value = list(4)
Cells(7, 1).Value = list(5)
Cells(8, 1).Value = list(6)
Cells(9, 1).Value = list(7)
Cells(10, 1).Value = list(8)
Cells(11, 1).Value = list(9)
```

【練習２－２】

```
For i = 0 To UBound(list)
    Randomize
    rn = Int(UBound(list) * Rnd)

    Cells(i + 2, 2).Value = rn
Next
```

書き換え部分はこの１行

【練習２－３】

```
For i = 0 To UBound(list)
    Randomize
    rn = Int(UBound(list) * Rnd)

    Cells(i + 2, 2).Value = rn
    Cells(i + 2, 3).Value = list(rn)
Next
```

追記部分はこの 1 行

【練習２－４】

```
Option Explicit

Dim picName As String
Dim myShape As Shape
Dim list() As Variant   'カードの名前リスト


Sub sample()

    list = Array("SA.png", "S2.png", "S3.png", "S4.png", "S5.png", "S6.png", "S7.png", "S8.png", "S9.png", "S10.png", "SJ.png", "SQ.png", "SK.png", _
           "HA.png", "H2.png", "H3.png", "H4.png", "H5.png", "H6.png", "H7.png", "H8.png", "H9.png", "H10.png", "HJ.png", "HQ.png", "HK.png", _
           "DA.png", "D2.png", "D3.png", "D4.png", "D5.png", "D6.png", "D7.png", "D8.png", "D9.png", "D10.png", "DJ.png", "DQ.png", "DK.png", _
           "CA.png", "C2.png", "C3.png", "C4.png", "C5.png", "C6.png", "C7.png", "C8.png", "C9.png", "C10.png", "CJ.png", "CQ.png", "CK.png")


    picName = ActiveWorkbook.Path & "¥card¥sa.png"
    Set myShape = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
        linktofile:=False, _
```

↑こうなるのですが、見づらいと思いますので配列を初期化する部分のみ下に文字情報として記載します。

```
list = Array("SA.png", "S2.png", "S3.png", "S4.png", "S5.png", "S6.png", "S7.png", "S8.png", "S9.png", "S10.png", "SJ.png", "SQ.png", "SK.png", _
           "HA.png", "H2.png", "H3.png", "H4.png", "H5.png", "H6.png", "H7.png", "H8.png", "H9.png", "H10.png", "HJ.png", "HQ.png", "HK.png", _
           "DA.png", "D2.png", "D3.png", "D4.png", "D5.png", "D6.png", "D7.png", "D8.png", "D9.png", "D10.png", "DJ.png", "DQ.png", "DK.png", _
           "CA.png", "C2.png", "C3.png", "C4.png", "C5.png", "C6.png", "C7.png", "C8.png", "C9.png", "C10.png", "CJ.png", "CQ.png", "CK.png")
```

【練習２－５】

この部分をこのように書き換える

```
        For j = 2 To 14
            picName = ActiveWorkbook.Path & "¥card¥ura.png"
            Set myShape = ActiveSheet.Shapes.AddPicture(Filename:=picName, _
                linktofile:=False, _
```

【練習４－７－３①②】

```
Function cardClick(cardNo)
    Dim i As Integer
    Dim j As Integer

    MsgBox Application.Caller
    myShape(cardNo).Delete

    picName = ActiveWorkbook.Path & "¥card¥" & Application.Caller
    i = Int(cardNo / 13) + 2
    j = cardNo - 13 * Int(cardNo / 13) + 2

    Call picSet(picName, i, j, cardNo)

End Function
```

ローカル変数として、i, j を追加。整数型。

引数で渡す項目を準備

picSet 関数を呼び、画像表示。

②のメッセージボックスを表示しなくするためには、この行を削除する。
　または、コメントアウトでもよいです。

【練習５－４】

```
        Case 2
            target2 = Mid(Application.Caller, 2, 1)
```

【練習６－２】
定義部分

```
Dim index1 As Integer       '1枚目のカードのリスト内
Dim target2 As String       '2枚目のカードナンバー
Dim player As Integer       'プレイヤー人数
```

これを追加

値を受け取り、player に代入する部分。

```
Sub sample()
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer

    openNum = 0     '開かれているカード枚数を0で初期化
    player = Cells(1, 11).Value        'プレイヤー人数を受け取る
    allShapeDelete

    list = Array("SA.png", "S2.png", "S3.png", "S4.png", "S5.png", "S
```

これを追加

確認用表示

```
    openNum = 0     '開かれているカード枚数を0で初期化
    player = Cells(1, 11).Value        'プレイヤー人数を受け取る
    Cells(1, 13).Value = player        '確認用
    allShapeDelete
```

これを追加
確認後は消去 or コメントアウト

【練習６－４－１】

```
Dim index1 As Integer          '1枚目のカードのリスト内イ
Dim target2 As String          '2枚目のカードナンバー
Dim player As Integer          'プレイヤー人数
Dim playerNum As Integer       'プレイヤー番号
```

これを追加

【練習６－４－２】

```
    openNum = 0     '開かれているカード枚数を0で初期化
    player = Cells(1, 11).Value        'プレイヤー人数を受け取る
    playerNum = 1                      'プレイヤー番号を１で初期化
    allShapeDelete
```

これを追加

【練習６－４－３】

```
    openNum = 0     '開かれているカード枚数を0で初期化
    player = Cells(1, 11).Value        'プレイヤー人数を受け取る
    playerNum = 1                      'プレイヤー番号を１で初期化
    Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
```

これを追加

【練習６－５】
処理 K-1

```
Dim player As Integer           'プレイヤー人数
Dim playerNum As Integer        'プレイヤー番号

Dim COLOR_INDEX_ADD As Integer     'カラーインデックス底上げ値
```

これを追加

処理 K-2、K-3

```
openNum = 0     '開かれているカード枚数を0で初期化
player = Cells(1, 11).Value        'プレイヤー人数を受け取る
playerNum = 1                      'プレイヤー番号を１で初期化
COLOR_INDEX_ADD = 33          'カラーインデックス底上げ値を33に指定

Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

allShapeDelete
```

K-2 はこれ

K-3 はこれ

【練習６－６－２①】

```
Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

Cells(7, 6).Value = "プレイヤー１の点数：0"

allShapeDelete
```

これを追加。
タイミングとしては
このあたりが妥当。

【練習６－６－２②】

```
Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

i = 1
Cells(7, 6).Value = "プレイヤー" & i & "の点数：0"

allShapeDelete
```

ｉ ＝ １を追加。
&演算子で文字列と ｉ
を連結。

【練習６－６－２③】

```
    Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

    For i = 1 To player
        Cells(6 + i, 6).Value = "プレイヤー" & i & "の点数：0"
    Next i

    allShapeDelete
```

For ループにする。

行数指定は、こうすることで i=1 のとき 7 行目、あとは順次 8，9，…と指定できる。

【練習６－６－３①】

```
    For i = 1 To player
        Cells(6 + i, 6).Value = "プレイヤー" & i & "の点数：0"
        Cells(6 + i, 6).Interior.ColorIndex = i + COLOR_INDEX_ADD
    Next i
```

これを追加

【練習６－６－３②】

```
    For i = 1 To player
        Cells(6 + i, 6).Value = "プレイヤー" & i & "の点数：0"
        For j = 6 To 8
            Cells(6 + i, j).Interior.ColorIndex = i + COLOR_INDEX_ADD
        Next j
    Next i
```

ｊ を使った For ループにする。ｊの変化は 6～8。

ここを ｊ にすることで、列方向に変化させる。

【練習７―１―１】【練習７―１－２】【練習７―１－３】

```
If target1 = target2 Then

Else
    If playerNum = player Then
        playerNum = 1
    Else
        playerNum = playerNum + 1
    End If

    Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
    Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD
End If
```

７―１－１

７―１－２

７―１－３

【練習７－２】

```
If target1 = target2 Then

Else
    If playerNum = player Then
        playerNum = 1
    Else
        playerNum = playerNum + 1
    End If

    MsgBox "残念！次はプレイヤー" & playerNum & "です"

    Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
    Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD
End If
```

このタイミングで入れましょう。

【練習７－３－１①】

```
    MsgBox "残念！次はプレイヤー" & playerNum & "です"

    Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
    Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

    picName = ActiveWorkbook.Path & "¥card¥ura.png"
End If
```

これを追加

【練習７－３－１②】

```
    MsgBox "残念！次はプレイヤー" & playerNum & "です"

    Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
    Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

    picName = ActiveWorkbook.Path & "¥card¥ura.png"

    myShape(index1).Delete
```

これを追加

【練習７－３－１③】

```
    picName = ActiveWorkbook.Path & "¥card¥ura.png"

    myShape(index1).Delete
    Call picSet(picName, target1row, target1col, index1)

End If
```

これを追加

【練習７－３－２①】
行番号・・・i
列番号・・・j
インデックス番号・・・cardNo

【練習７－３－２②】

```
    Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
    Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

    picName = ActiveWorkbook.Path & "¥card¥ura.png"

    myShape(index1).Delete
    Call picSet(picName, target1row, target1col, index1)

    myShape(cardNo).Delete
    Call picSet(picName, i, j, cardNo)
```

この２行を追加
消去して表示。

【練習７－４】

```
    Select Case openNum
        Case 1
            target1row = i
            target1col = j
            index1 = cardNo
            target1 = Mid(Application.Caller, 2, 1)

             MsgBox target1

        Case 2
            target2 = Mid(Application.Caller, 2, 1)

            If target1 = target2 Then

            Else
                If playerNum = player Then
                    playerNum = 1
                Else
                    playerNum = playerNum + 1
                End If

                MsgBox "残念！次はプレイヤー" & playerNum & "です"

                Cells(7, 2).Value = "今はプレイヤー" & playerNum & "です。"
                Range("B7:D7").Interior.ColorIndex = playerNum + COLOR_INDEX_ADD

                picName = ActiveWorkbook.Path & "¥card¥ura.png"

                myShape(index1).Delete
                Call picSet(picName, target1row, target1col, index1)

                myShape(cardNo).Delete
                Call picSet(picName, i, j, cardNo)


            End If

            openNum = 0
    End Select
```

1枚目が開かれたときの処理

2枚目が開かれたときの処理

これを追加

【練習８－１－２】

```
For i = 1 To player
   Cells(6 + i, 6).Value = "プレイヤー" & i & "の点数：0"
   For j = 6 To 8
      Cells(6 + i, j).Interior.ColorIndex = i + COLOR_INDEX_ADD
   Next j
Next i

ReDim point(player)

allShapeDelete
```

これを追加。
練習７－１－３②の後に追加しました。

【練習８－１－３】

```
    ReDim point(player)
    For i = 0 To player
        point(i) = 0
    Next i

    allShapeDelete
```

ReDim のあとで For ループを回してください。

【練習８－２－１】

これを追加

```
            If target1 = target2 Then
                MsgBox "おめでとう！２枚ゲット"
            Else
                If playerNum = player Then
                    playerNum = 1
                Else
                    playerNum = playerNum + 1
```

【練習８－２－２】

これを追加

```
            If target1 = target2 Then
                MsgBox "おめでとう！２枚ゲット"
                point(playerNum) = point(playerNum) + 2
```

【練習８－２－３】

この行を追加

```
If target1 = target2 Then
    MsgBox "おめでとう！２枚ゲット"
    point(playerNum) = point(playerNum) + 2
    Cells(6 + playerNum, 6).Value = "プレイヤー" & playerNum & "の点数：" & point(playerNum)

Else
```

【練習８－３－１】

宣言・定義部分

```
Function cardClick(cardNo)
    Dim i As Integer
    Dim j As Integer
    Dim trow As Integer
    Dim tcol As Integer

    myShape(cardNo).Delete
```

この２行を追加

(a) (b) 部分

```
point(playerNum) = point(playerNum) + 2
Cells(6 + playerNum, 6).Value = "プレイヤー" & playerNum & "の点数：

'画像の移動
trow = 6 + playerNum
tcol = 8 + point(playerNum)
```

(a)

(b)

【練習８－３－２①】

```
'画像の移動
trow = 6 + playerNum
tcol = 8 + point(playerNum)
myShape(index1).Left = Cells(trow, tcol - 1).Left   '1枚目の移動先・横位置
myShape(index1).Top = Cells(trow, tcol - 1).Top     '1枚目の移動先・縦位置
```

この行を追加