Excel VBAを用いたプログラミング　第3部

奈良教育大学　薮 哲郎

最終更新　2016.7.4

# ユーザーインターフェースの作成

フォームの利用

　これまではマクロを実行させるのにVisual Basic Editorのウィンドウで「実行」→「Sub/ユーザーフォームの実行」としていました。以下のようにすると、シート上のボタンを押すことでマクロを実行できます。

1. あらかじめマクロ（Subではじまり End Sub で終わるひとまとまりのプログラム）を作成しておきます。
2. 「開発」→「コントロール：挿入▼」→「フォームコントロール：ボタン」を選択し、ドラッグしてシート上にボタンを作成します。このときaltキーを押すとボタンの外周位置をセルの境界に合わせます。
3. そのボタンと結びつけるマクロ名を選びます。

　ボタンの上にカーソルを移動させると、カーソルが手のマークに変わります。この状態でボタンをクリックすると、ボタンに結びつけられたマクロを実行します。

　ボタンを右クリックすると「ボタンに表示されるテキストの編集」「ボタンに結びつけるマクロを変更する」「ボタンのサイズを変更する」などができます。

＜課題1＞

　シート「カウンタ」を使用します。

1. 1回実行するとA1の数値が1増えるマクロを作成しなさい。
2. シート上にボタンを1個配置し、そのマクロと結びつけなさい。これにより、ボタンを1回クリックすると、A1の数値が1増えるカウンタが作成できました。ボタンの上に表示されるテキストは「up」としなさい。
3. 1回クリックするとA1の数値が1減るマクロを作成しなさい。ただし、負の値にはならないようにしなさい。新たにボタンと作成し、このマクロと結びつけなさい。ボタンの上に表示されるテキストは「down」としなさい。
4. ボタンをもう一つ作り、そのボタンを押すとA1の数値が0となるようプログラムを作りなさい。ボタンの上に表示されるテキストは「clear」としなさい。

## メッセージボックスとインプットボックス

　エラーなどのメッセージを表示したり、Yes/Noを問い合わせるためのメッセージボックスを表示する方法を以下に示します。

' OKボタンのみのメッセージボックス

MsgBox("表示するメッセージ")

' OK/Cancelを選ぶメッセージボックス

ret = MsgBox("表示するメッセージ", vbOKCancel)

'　　retの値はOKのとき1, Cancelのとき2

' Yes/Noを選ぶメッセージボックス

ret = MsgBox("表示するメッセージ", vbYesNo)

'　　Yes のとき6, No のとき7

' Yes/No/Cancelを選ぶメッセージボックス

ret = MsgBox("表示するメッセージ", vbYesNoCancel)

'　　Yes のとき6, No のとき7, Cancelのとき2

　MsgBoxの引数について知りたいときは、カーソルをMsgBoxの上に置いてF1キーを押してヘルプを表示して下さい。

　表示する文字列は変数を用いて動的に生成することも出来ます。

MsgBox("a の値は " & a & " です。")

　文字列を連結する場合、& の代わりに + でも連結できます。ただし + を使う場合は、CStr(a) と書かないとエラーになります。

　InputBoxを使うと文字列を入力することができます。

name = InputBox("あなたの名前を入れて下さい。")

# マクロからマクロを呼び出す

　次のように「call マクロ名」と書くと、マクロから別のマクロを呼び出すことができます。Call macro2はSub macro2() にジャンプせよという命令です。macro2の実行が終了した後は、Call macro2の次の行から実行を再開します。なお、callは省略することができますが、付けた方が良いでしょう。

Sub macro1()

MsgBox ("今から macro2 を呼びます")

Call macro2

MsgBox ("macro2 から戻ってきました")

End Sub

Sub macro2()

MsgBox ("macro2 が呼ばれました")

End Sub

　共通する処理を1つのマクロにまとめることで、プログラムが読みやすくなり、変更するときも楽になります。

　次のように変数を渡すこともできます。

Sub macro3()

a = 2

b = 3

Debug.Print "at macro 3a:"; a; b

Call macro4(a, b)

Debug.Print "at macro 3b: "; a; b

End Sub

Sub macro4(c, d)

Debug.Print "at macro 4a:"; c; d

c = c \* 2

d = d + 10

Debug.Print "at macro 4b:"; c; d

End Sub

　このプログラムの実行結果は下記のようになります。

at macro 3a: 2 3

at macro 4a: 2 3

at macro 4b: 4 13

at macro 3b: 4 13

　変数a, b はmacro3で値を代入しています。これをcall macro4(a, b) としてmacro4に渡すと、macro4 においては c, d として認識されます。macro4の中で変数 c, d の値を変更すると、macro3 における a, b の値が書き換わります。

＜課題2＞

　先ほどの課題において、カウンタを5増やすマクロとそれに対応するボタンを作成しなさい。ただし、そのマクロ内においては、セルの値を5増やすのではなく、セルの値を1増やすマクロを5回呼びなさい。

＜課題3＞

　シート「電子サイコロ」を使用します。

　空のマクロsaikoro\_furuとsaikoro\_clearを作りなさい。

　ボタンを2つ作り、一つは「サイコロを振る」、もう一つは「クリア」とテキストを入れ、空のマクロと結びつけなさい。

1. 「クリア」ボタンを押すとB1～B6に0を入れなさい。
2. 「サイコロを振る」ボタンを押すと以下の処理を行いなさい。まず、1～6の乱数を発生させ、E1に書き込みます。Excelの関数Randbetweenを利用して、Worksheetfunction.Randbetween(1,6) と書きます。次に、1が出たらB1の値を1増やします。2～6の目が出たときも同様にして、B1～B6に出た目の累積回数を入れなさい。
3. B7にサイコロを振った合計回数を入れる関数を記入しなさい。
4. C1～C6にそれぞれ1～6の目が出た割合を入れる数式を記入しなさい。
5. 以下のようなグラフを作成しなさい。棒グラフの縦軸の最小値と最大値は自動ではなく、手動で設定しなさい。
6. もう一つボタンを作り、「サイコロを10回振る」としなさい。そのボタンを押したなら、「サイコロを振る」で作成したマクロを10回呼びなさい。

　　

＜課題4＞

　シート「野球」を使用します。

　strike(), ball(), hit(), out() という4つの空のマクロを作りなさい。次にボタンを4つ作り「ストライク」「ボール」「ヒット」「アウト」と表示し、空のマクロに結びつけなさい。

* 「ストライク」を押すとA2を1増やします。増やした結果A2の値が3になったなら、「三振です」というメッセージボックスを表示し、A2 = 0, B2 = 0としてC2を1増やします。C2を増やした結果、3になったら、「チェンジです」というメッセージボックスを表示し、A2 = 0, B2 = 0, C2 = 0としなさい。
* 「ボール」を押すとB2の値を1増やします。増やした結果、B2が4になったなら、「フォアボールです」というメッセージボックスを表示し、A2 = 0, B2 = 0にしなさい。
* 「ヒット」を押すとA2 = 0, B2 = 0とします。
* 「アウト」を押すとA2 = 0, B2 = 0としてC2の値を1増やします。C2を増やした結果、3になったなら「チェンジです」というメッセージボックスを表示し、C2 = 0としなさい。
* 「チェンジです」と表示してA2, B2, C2の3つのセルに0を入れる部分が2箇所あります。この部分を1つのマクロとして独立させなさい。

# 乱数

　サイコロを振ったときに出る目の数のように、ランダムな数値を乱数と言います。WorkSheetFunction.Randbetweenを使うのが手軽ですが、VBAのRnd() 関数を使う方法もあります。以下のようにします。

Randomize ' 乱数を使用する前に Randomize を実行する。

a = Rnd() ' 0 以上 1 未満の乱数を発生させ a に入れる

b = Int(Rnd() \* 6) + 1 ' 1～6 までの乱数を発生させ b に入れる

　関数Rnd() は0以上1未満の数を返します。1から6までの乱数が必要な場合は、Int(Rnd() \* 6)+1 とします。Rnd() で得られた値を6倍すると0～5.9999999…… の数値が得られるので、Intをとると0～5までの乱数が得られます。それに1を足すことで1～6までの乱数を得ます。

　Randomize命令はRnd() 関数の種を設定します。この例のように引数を省略するとタイマーを手がかりとした値を設定します。Randomizeは原則としてプログラム開始後1回だけ呼べばよいのですが、1回しか呼ばない場合、乱数に偏りが生じる場合があります。Rnd() の直前に毎回呼ぶと偏りはなくなります。

# 文字列と空白

　Excelのセルや変数に入るデータのうち、代表的な2つは以下の通りです。

* 数値
* 文字列

　文字列は "abc" のように前後をダブルクォーテーションで囲んで表します。文字列の中に " が含まれる場合は、"ab""c" のように " を2個続けることで、1個の " を表します。

　文字列を扱う場合、気をつけないといけないことが1つあります。セルの中に文字列が入っている場合、"abc" と "abc "（cの後に半角空白が1つ入っている）は人間が見ると区別がつきませんが、コンピュータは別の文字列として区別します。半角空白は1個の文字として認識されるからです。

　以下のプログラムを実行すると、イミディエイトウィンドウには false と表示されます。

Range("A1") = "abc " ' c の後に半角空白が1つある

If Range("A1") = "abc" Then ' abc の前後には空白はない

Debug.Print "true"

Else

Debug.Print "false"

End If

　セルA1の見た目は "abc" ですが、内容は "abc " です（c の次に半角空白が1個）。通常は、文字列の前後の空白は無視したい場合が多いですから、以下のようにTrim関数を使います。

If Trim(Range("A1")) = "abc" Then

　Trim関数は文字列の前後に空白（全角の空白を含む）があるとき、その空白を削除します。上記のようにTrim関数を使うと、イミディエイトウィンドウにはtrueと表示されます。

　大変ややこしいのですが、ワークシート関数にもTrim関数があり、VBAのTrim関数とは働きが少し異なります。ワークシート関数のTrimは文字列の途中に複数の空白があるとき1つにします。従って、

moji = " abc def "

str1 = Trim(moji)

str2 = WorksheetFunction.Trim(moji)

Debug.Print "<"; str1; ">"

Debug.Print "<"; str2; ">"

の実行結果は

<abc def>

<abc def>

となります。

# 変数の宣言

　これまでは変数の宣言を省略してきました。この場合、変数名のタイプミスによるバグが発生する可能性があります。例えば変数valueの値をセルA1に入れるつもりで、Range("A1") = valua（末尾のeをタイプミスしてaと書いた）と書くと、VBAでは変数の初期値は長さ0の文字列なので、セルA1がクリアされます。

　これを回避する方法として、標準モジュールの最初の行に

Option Explicit

と書くと変数の宣言が必須となります。

　あるいは「ツール」→「オプション」→「エディタ」で「変数の宣言を強制する」にチェックを入れてから、新たにモジュールを挿入するとOption Explicitがモジュールの先頭に挿入されます。

　変数名の宣言を必須にしておいて、Range("A1") = valuaという文を書くと、「valuaという変数は宣言されていないので使えない」というエラーが発生し、タイプミスによるバグを防ぐことができます。

　変数の宣言例を示します。

Dim a, b, c

Dim d As Long, e As Double, f As String, g As Variant

Dim h, i As Long

　変数を宣言は「Dim」を使います。Dimの後に変数名を列挙します。変数を2個以上宣言する場合は「,」で区切ります。

　変数の型を指定する場合、変数名の後に「As 変数の型」と書きます。代表的な型としては以下に示す型があります。

Int 　　2バイトの整数型（-32768～32767までしか表せない）

Long 　　4バイトの整数型

Double 　　8バイトの実数型

String 　　文字列型

Variant 　　何でも入る型

　As以下を省略するとVariant型とみなされます。上の例ではa, b, c は型が省略されているのでVariant型です。注意すべきは3行目です。h は型が省略されているのでVariant型となります。h, i の両方をLong型変数としたい場合は、

Dim h As Long, i As Long

と書く必要があります。

　Option Explicitを省略し、変数を宣言しない場合、全ての変数はVariant型となります。実はこれまでに扱ってきた変数は全てVariant型です。

　「変数の型」という新しい概念が登場しました。整数型変数（Int, Long）には整数しか入りませんし、文字型変数（String）には文字列しか入りません。

　整数型変数に実数（Double）を代入すると小数点以下が切り捨てられます。整数型変数に文字列を代入するとエラーが発生します。

　文字型変数に数値「1.23」を代入すると、文字列 "1.23" に自動変換されてから格納されます。

　Variant型は何でも入るので一見便利なように見えますが、欠点もあります。「整数型」「実数型」のように数値のみを格納する変数に文字列を代入すると「型が一致しません」というエラーが発生します。Variant型の使用はそのようなエラーの検知を不可能にするので、バグを見落とす可能性が高くなります。

　なお、またVariant型変数の読み書きはLong, Double型の変数より若干時間がかかりますが、本演習で行う程度のプログラムの場合は全く問題とはなりません。

# 数値と文字列

数値と文字列の自動変換

　数値「123」と文字列 "123" は異なるものです。また文字列 "123" と " 123" は異なります（先頭のスペースの有無に注意）。

　Excel VBAには型の自動変換機能があります。

Dim num As Long, str As String, i As Long, a As Double

num = 123

str = num

i = str

a = str

　文字型変数strに整数123を代入すると、自動的に文字列 "123" に変換されます。

　整数型変数iに文字列 "123" を代入すると、自動的に数値123に変換されます。

　実数型変数aに文字列 "123" を代入すると、自動的に数値123に変換されます。

　型変換を明示する場合は以下のように書きます。

str = CStr(num)

i = CLng(str)

a = CDbl(str)

　If文において「=」で比較する左右の型が異なる場合、自動的な型変換が行われます。

Dim a As String, b As Long

a = "100"

b = 200

If a = 10 Then ①

......

If b = "100" Then ②

......

If b = "abc" Then ③

......

　① では文字型変数a（中身は "100"）と数値10が比較されています。変数aに格納された文字列 "100" は数値100に自動変換されてから比較されます。

　②では整数型変数b（中身は200）と文字列 "100" が比較されています。文字列 "100" は数値100に自動変換されてから比較されます。

　③は整数型変数b（中身は200）と文字列 "abc" が比較されています。文字列 "abc" は数値に直すことができず、数値と文字列の比較はできないので、「型が一致しません」のエラーが発生します。

## 文字列処理用関数

　文字列を処理するための関数として既にTrimを学習しました。文字列を扱う代表的な関数の使用例を以下に示します。

Dim moji As String

Dim str As String, str2 As String, str3 As String, str4 As String

Dim num As Long, leng1 As Long, leng2 As Long

Dim bunkai() As String　あるいは bunkai As Variant

moji = " abcdefghij "

leng1 = Len(moji) ' 文字列の長さ（文字数）を数える

str = Trim(moji) ' 前後の空白を切り落とす

leng2 = Len(str) ' 文字列の長さ（文字数）を数える

num = InStr(str, "d") ' 文字 "d" が存在するなら何文字目かを返す

str2 = Left(str, 2) ' 左から2文字切り出す

str3 = Right(str, 3) ' 右から3文字切り出す

str4 = Mid(str, 3, 2) ' 左から3文字目から2文字切り出す

Debug.Print leng1; leng2; num

Debug.Print str; "---"; str2; "---"; str3; "---"; str4

moji = "abc def ghi"

bunkai = Split(moji, " ") ' 半角空白を区切り文字として文字列を分解

Debug.Print bunkai(0); "---"; bunkai(1); "---"; bunkai(2)

　Lenは文字列の長さを与えます。a～jの10文字の手前に半角空白が2個、後に半角空白が2個あります。よって、leng1は14です。

　Trimは前後の空白を切り落とすので、strは "abcdefghij" が入ります。leng2は10です。

　InStr(str,"d") はstrの中に文字列 "d" が含まれる場合は、何文字目から "d" と一致するかを返します。この場合は4となります。含まれない場合は0となります。

　Left(str,2) は左から2文字を切り出すので、str2は "ab" が入ります。

　Right(str,3) は右から3文字を切り出すので、str3は "hij" が入ります。

　Mid(str,3,2) は左から3文字目から2文字切り出すので、str4は "cd" が入ります。Mid(str,3) とすると3文字目から最後までを切り出します。

　Splitは与えられた文字列を指定された区切り文字で分割します。この例では " "（半角スペース）が区切り文字です。結果はVariant型かString型変数で受けます。この場合、bunkai(0) = "abc", bunkai(1) = "def", bunkai(2) = "ghi" となります。変数bunkaiの宣言方法が、Variant型かString型かで異なります。それぞれ「Dim bunkai() As String」「Dim bunkai As Variant」です。そうでないとエラーになります。Variant型は配列を格納できるというのが理由のようですが、分かりにくいです。

＜課題5＞

　シート「グループ分け」を使用します。

(1) A列を名字と名前に分けてC列とD列に入れなさい。SplitあるいはInstr, Left, Rightを組み合わせます。名字と名前は全角空白で区切られています。

(2) B列を名字と名前に分けてE列とF列に入れなさい。

(3) ひらがなの名前（F列）に基づいて全体をソートしなさい（手作業で行う）。

(4) 先頭から4人ずつで1つの班を作ります。班員の名前をI, J, K, L列に書き込みなさい。

(5) I, J, K, L列をつなげたものをM列に入れる数式を書きなさい。ただし、全角空白を挟んで名前を繋げなさい。セルの内容をつなげるには　&　を使うかCONCATENATE関数を使います。例えば、セルI2とJ2を全角空白をはさんで接続する場合、

= I2 & "　" & J2と書きます。CONCATENATE関数を使用する場合は

= CONCATENATE(I2,"　",J2)　と書きます。



# 配列

　配列という変数があります。以下のように使います。

Dim a(10)

For i = 0 To 10

a(i) = i

Next i

　Dim a(10) と宣言するとa(0), a(1), a(2).....a(10) という11個の変数を使えるようになります。一方で、aという変数は使えません。宣言はどの場所でしても構いませんが、マクロの冒頭で行うのが分かりやすいと思います。

　配列変数のメリットはa(i) のiの部分を変数として変化させることができる点にあります。

　Excelではセルという2次元配列が最初から利用できるので、配列変数を使うことはあまりないかも知れません。

　配列変数を型も含めて宣言するには以下のようにします。

Dim a(10) As Long, b(10) As String, c(10) As Variant

（参考）

　C言語でa[10]と宣言するとa[0]～a[9]までの10個の配列変数が使えます。Fortranでa(10)と宣言するとa(1)～a(10)までの10個の配列変数が使えます。VBAではa(10)という宣言で11個の配列変数が使えるのは、他の言語とは異なっているので注意して下さい。私はa(10)と宣言してもa(0)は使わないことが多いです。Option Base 1 とモジュールの先頭で記述するとa(1)からになりますが、ファイルの先頭を意識する必要があるというのが何となく嫌なので私は使っていません。

# 関数

　自作関数は以下のように記述して使用します。

Function myfunc(x)

y = x \* 2

myfunc = y

End Function

Sub test()

a = 2

b = myfunc(a)

Debug.Print b

End Sub

　関数は値を返すマクロです。Function で書き始めます。上記のように関数myfuncの中にmyfunc = 5という記述があった場合、この関数の返り値は5となります。関数に渡す数値（ここではx）を引数（ひきすう）と言います。引数の個数は2個以上あっても構いませんし、0個でも構いません。

＜例題1＞

　電気料金は以下の表で決まります。A列に電気使用量が入っています。使用電力量を引数として与えると、電気料金を返す関数を完成させなさい。また、この関数を利用して、シート「電気料金」のB列を完成させるプログラムを作りなさい。

|  |  |
| --- | --- |
| 使用電力量 | 価格 |
| ～ 15kWh | 400円 |
| 15kWh ～ 120kWh | 25円／kWh |
| 120kWh ～ 300kWh | 30円／kWh |
| 300kWh ～ | 35円／kWh |

Function denki\_ryokin(x0)

x = x0

ryokin = 0

If x >= 300 Then

ryokin = 35 \* (x - 300)

x = 300

End If

If x >= 120 Then

ryokin = ryokin + 30 \* (x - 120)

x = 120

End If

If x >= 15 Then

ryokin = ryokin + 25 \* (x - 15)

x = 120

End If

ryokin = ryokin + 400

denki\_ryokin = ryokin

End Function

Sub ryokin\_list()

Range("a65536").End(xlUp).Select

last\_row = ActiveCell.Row

For i = 2 To last\_row

Range("b" & i) = denki\_ryokin(Range("a" & i))

Next i

End Sub

# その他

＜課題6＞

　シート「検索」を利用します。本課題はプログラムを組む課題ではなく、便利な関数であるVLookupの使い方を学ぶ課題です。

* F4に入れる値をドロップダウンリストから選べるようにします。a2～a36から選べるようにしなさい。ドロップダウンリストの作成法は「データ」→「データツール：データの入力規則」→「設定：入力値の種類：リスト」です。
* F4に入れた名前コードを持つ人の「名前」「レポート提出」「テスト点」をそれぞれG4, H4, I4に入れなさい。VLookupの構文は以下の通りです。

　=VLookup(探す値, 探索する範囲　探す値は1列目に含む, 表示したい値が何列目か)

　VLookupは大変便利な関数です。VBAの中で使うときは、範囲を以下のように指定します。

Range("G4") = WorksheetFunction.VLookup(\_ ' \_ は継続行を表す

Range("F4"), Range("A2:D36"), 2)