# 配列・文字処理・数学関数・オブジェクト

## 配列

　配列変数というタイプの変数があります。以下のように使います。

Dim a(3)

a(0) = 0

a(1) = 10

a(2) = 20

a(3) = 30

Debug.Print a(0)

Debug.Print a(1)

Debug.Print a(2)

Debug.Print a(3)

　a(0), a(1), a(2), a(3) は配列変数です。配列変数はこのようにカッコ付きの変数で、カッコの中に整数を書きます。配列変数を使うには宣言が必要です。Dim a(3)が配列の宣言です。この宣言でa(0), a(1), a(2), a(3) の4個の変数が使用可能になります。この例の場合、Dim a(3) As Longのように型を書いていないので、a(0)～a(3) はVariant型です。

　配列変数が便利なところは、カッコの中（添字と呼びます）を変数で表せることです。上記のプログラムを変数を使って書き直すと次のようになります。

Dim a(3)

For i = 0 To 3

a(i) = i \* 10

Next i

For i = 0 To 3

Debug.Print a(i)

Next i

　Excel VBAでa(10)と宣言すると、a(0)～a(10)まで11個の変数が使えますが、これは他の言語とは異なります。C言語ではa[10]と宣言すると、a[0]～a[9]まで10個の変数が使えます。Fortranではa(10)と宣言するとa(1)～a(10)まで10個の変数が使えます。10と宣言して11個使えるというExcel VBAは少し風変わりな処理系です。

　a(i) のように添字が1個の配列を1次元配列と呼びます。

Dim a(10, 20)

と宣言すると、添字が2個ある配列変数を扱うことができます。2次元配列と呼びます。使うことはあまりないと思いますが、3次元や4次元の配列も使用可能です。

　C, Javaなどの言語と異なり、Excelではセルという2次元配列が最初から用意されています。従って、配列変数を使う機会は他の言語よりは少ないかもしれません。

　「セル」と「2次元配列」を比べると、「2次元配列」の方がアクセス速度が速い（セルに値を代入すると画面表示が変更されるので画面処理の時間が必要ですが、2次元配列に値を代入するのは瞬時です）という長所があります。プログラムを高速化するため、セルの値を一旦2次元配列にコピーし、その2次元配列に対して処理を行った後、セルに戻すという技法があります。

　後の節で文字列を分解する関数を学習しますが、分解結果は1次元配列で受け取ります。

## 文字列の処理

　Excel VBAには文字列を処理する関数があります。下のプログラムを実行してみて下さい。

moji = "abcあいうえおxyz"

Range("a1") = Len(moji) ' 文字列の長さ

Range("a2") = Left(moji, 4) ' 左端から4文字取り出す

Range("a3") = Right(moji, 5) ' 右端から5文字取り出す

Range("a4") = Mid(moji, 3, 2) ' 3文字目から2文字取り出す

Range("a5") = InStr(moji, "c") ' 文字列 "c"は何文字目か？

Range("a6") = InStr(moji, "p") ' 文字列 "p"（含まない）は

' 何文字目か？

Range("a7") = InStr(moji, "cあ") ' 文字列 "cあ" は何文字目か？

　文字列を扱う関数の働きは以下の通りです。全角文字と半角文字はどちらも1文字とカウントされます。

* Len(s)：文字列sの長さを返す
* Left(s, n)：文字列sの左端からn文字取り出す
* Right(s, n)：文字列sの右端からn文字取り出す
* Mid(s, n, m)：文字列sのn文字目からm文字取り出す
* Instr(s, c)：文字列sの中に文字列 c が含まれる場合は、何文字目かを返す。含まれない場合は0を返す。複数回含まれる場合は最初に見つかった位置を返す。検索する文字列は "cあ" のように2文字以上でも構わない。

　文字列を扱う場合、文字列の右端や左端に空白が含まれる場合、それをあらかじめ除去しておきたい場合があります。Trimを使います

moji2 = " abc de fgh ijk " ' Trimの違いを調べる

Range("b1") = moji2

Range("b2") = Trim(moji2)

Range("b3") = WorksheetFunction.Trim(moji2)

　Trimは文字列の左右に空白が含まれている場合に、それを切り落とします。ただし、注意事項があります。Excel VBAに備わっているTrim関数とワークシート関数Trim（セルの中に =Trim(A2) のように書く関数）は動作が異なります。違いは以下の通りです。

* Excel VBAのTrim関数………文字列の左右の空白だけを切り落とし、文字列の中間に含まれている空白はそのままです。
* WorksheetFunctionのTrim関数………左右の空白を切り落とす処理に加えて、文字列の中間に2個以上の空白が連続している場合、1個の空白に置換します。Split関数を使うときなどにこの処理が必要です。

【例題】

　シート「文字列操作」のA列に名字と名前が入っています。全角空白で区切られています。名字の部分と名前の部分に分解しなさい。

For i = 2 To 10

moji = Range("A" & i)

a = InStr(moji, "　") ' ダブルクオーテーションの中は全角空白

Debug.Print "i = " & i & " a = " & a

If a <> 0 Then

l = Len(moji)

myouji = Left(moji, a - 1)

namae = Right(moji, l - a)

Range("B" & i) = myouji

Range("C" & i) = namae

End If

Next i

　空セル、あるいは空白を含まない文字列が来た場合はa = 0となります。その場合は、何も処理をしないようにIf文を使っています。

　文字処理としてはLen(), Left(), Right(), Mid(), InStr()の他にSplit()を知っておくと便利です。シート「文字列操作2」を見て下さい。A1に入っている文字列を "," を区切り文字として分解して、B列に並べます。

moji = Range("A1")

moji\_list = Split(moji, ",")

a = LBound(moji\_list)

b = UBound(moji\_list)

Debug.Print a & " " & b

j = 1

For i = a To b

Range("B" & j) = moji\_list(i)

j = j + 1

Next i

　Split(moji, ",") と書くと、文字列を格納した変数を、区切り文字 "," で分解して配列moji\_listに格納します。この場合は、Dim moji\_list(100) のように配列の宣言は不要です。LBound(moji\_list)は配列moji\_list() の添字の下限を取得し、UBoundは上限を取得します。Splitで得た配列の添字の下限は0なので、上記のプログラムは冗長かもしれません。配列の下限を0と決め打ちしても構いません。

【課題1】

　シート「文字列操作2」のA10に英語の文章が入っています。これを単語ごとに分解してB10, B11, B12, .... に並べるプログラムを作りなさい。

## テキストファイルの読み書き

　Excel VBAでテキストファイルを読み込み、書き出しすることができます。まずは、ファイルを書き出してみましょう。セルA1の内容をファイルに書き出してみます。

filename = ThisWorkbook.Path & "\out.txt"

Open filename For Output As #1 ' ファイルを作成し装置番号1を

' 割り当てる

Debug.Print "A1 : " & Range("A1")

Print #1, "A1 : " & Range("A1")

Close #1 ' ファイルを閉じる

　まず作成するファイルの名前をfilenameという変数に入れます。

　ThisWorkbook.Pathはxlsmファイルが存在するフォルダ名（例えば"C:\Users\ユーザー名\Documents\フォルダ名"）に置換されます[[1]](#footnote-1)。その文字列に "\out.txt" という文字列を連結します。"\" はフォルダとフォルダを区切る文字です[[2]](#footnote-2)。filenameの内容は"C:\Users\ユーザー名\Documents\フォルダ名\out.txt" のようになります。

Open filename For Output As #1

と書くと、filenameという名前のファイルを作成し、装置番号1番に割り当てます。

　ファイルに書き出すPrint文とイミディエイトウィンドウに書き出すDebug.Printの使い方は同じです。対比させるため、Debug.Printを併記しています。

Print #1, ○〇〇

と書くと、○○○の内容をファイルに書き出します。書き出す文字列や変数名は & で接続します。セルA1の内容が「数値」「文字列」のどちらでも接続できます（数値の場合は文字列に変換してから接続します）。

　ファイルに書き出し終わったら、最後に装置番号1番を閉じます。

Close #1

　これを怠ると、ファイルは作成されますが、内容は空という状態になります。Excelを終了すると、Close #1が自動的に実行され、ファイルの内容が確定します。

　複数のファイルを同時にオープンするときは、#2, #3のように別の装置番号を使います。

　A1～A10のセルの内容を行番号付きで書き出してみましょう。

filename = ThisWorkbook.Path & "\out.txt"

Open filename For Output As #1

For i = 1 To 10

line = i & ":" & Range("A" & i)

Print #1, line

Next

Close #1

　1行分の内容を変数lineに入れ、Print #1, lineでファイルに書き出しています。

　こんなこともできます。A列に入っている名前のファイルを作成し、内容としてB列の文字列を入れます。各個人用にファイルを作成するときなどに使えます。

For i = 1 To 4

filename = ThisWorkbook.Path & "\" & Range("A" & i)

Open filename For Output As #1

Print #1, Range("B" & i)

Close #1

Next i

　次は、ファイルの内容を読み取るプログラムの書き方を示します。以下のような内容がテキストファイルin.txtに入っていることを仮定します。名前の手前や数値の後ろには半角空白が入っています。

yabu 10 20

aoki 30 40

sinjyou 50 60

suzuki 70 80

　ファイルから1行読み取って、セルA1に入れるプログラムは以下のように書きます。

filename = ThisWorkbook.Path & "\in.txt"

Open filename For Input As #1

Line Input #1, line ' 1行読み取って変数lineに入れる

Range("A1") = line

Close #1

　行数が分からないファイルを、ファイルの末尾まで読み取って、A1, A2, A3....の列に入れるプログラムは以下のように書きます。Do Until EOF(1) は定番表現であり、装置番号1をファイルの終わりまで読むことを表します。

i = 1

Do Until EOF(1)

Line Input #1, line ' line に1行分入る

Range("A" & i) = line

line = line + 1

Loop

　上記の処理に加えて、空白で区切られた文字列を分解して、A, B, C列に入れるには、以下のように書きます。主要部分を記します。

i = 1

Do Until EOF(1) ' ファイルの最後を見つけるまで繰り返す

Line Input #1, line ' 1行読み取って変数lineに入れる

line = WorksheetFunction.Trim(line) ' 左右の空白を削除し

' 途中の複数の空白を1個にする

buf = Split(line, " ") ' 空白で区切られている文字列を分解して

' buf(0),buf(1).... に格納する

For j = 0 To UBound(buf) ' UBound(buf) は配列の添え字の上限

Cells(i, 1 + j) = buf(j)

Next j

i = i + 1

Loop

　スペースで区切られた文字列を分けるため、WorksheetFunctionのTrim関数を使っています。

line = WorksheetFunction.Trim(" abc 10 20 ")

を実行すると、lineは

"abc 10 20"

となります。左右の空白が切り取られ、文字列と文字列を区切っていた空白は1個だけになります。Excel VBAのTrim関数は左右の空白は切り落としますが、abc と 10 の間のスペースは（3個）はそのままになるので、この場合には使えません。

　Split関数は文字列を分解します。Split関数の第1引数は入力する文字列、第2引数は区切り文字です。分解した結果は配列変数となって返ってきます。buf = Split(line, " ") と書くと、buf(0), buf(1), buf(2),..... に分解された文字列が入ります。この場合のみ宣言せずに配列変数が使えます。配列変数buf( ) の添字の上限を調べるのがUBound(buf)です。buf(0), buf(1), ..... buf(4)まで文字列が入っている場合、UBound(buf) は4になります。

　j のForループを用いて、分解した結果をセルに入れます。

　ここでは、空白で区切られた例を扱いましたが、CSVファイルの場合、コンマで区切られています。CSVファイルを読み取る場合、WorksheetFunction.Trimの処理は不要で、以下のように書きます。

buf = Split(line, ",") ' buf(0),buf(1).... に格納される

　本節では、VBAを使ってファイルの書き出しと読み込みを行いました。ほぼ同等のことを手動で行うこともできます。

　読み込みはOffice 2016などの場合、「データ」→「外部データの取り込み」→「テキスト」です。Office 365の場合、「ファイル」→「オプション」→「データ」で「レガシデータインポートウィザードの表示」の「テキストから（レガシ）」にチェックを入れてから、「データ」→「データの取得と変換：データの取得V」→「従来のウィザード」→「テキストから」です。

　書き出しは、「ファイル」→「名前を付けて保存」です。ファイル形式として以下の4つの形式があります。CSV形式がポピュラーです。

* テキスト（タブ区切り）(\*.txt)………半角文字は1字1byte, 全角文字は1字2byteで漢字コードはSJISです。改行は0d 0aの2byteです。
* Unicodeテキスト(\*.txt)………タブ区切りで漢字コードはUTF16-Little Endianです。全角文字も半角文字も1字2byteです。改行は0d 00 0a 00の4byteです。
* CSV（カンマ区切り）(\*.csv)………漢字コードはSJISです。
* テキスト（スペース区切り）(\*.prn)………漢字コードはSJISです。1セル8 byte固定で、それを超えた文字数は切り捨てられます。あまり使い道はなさそうです。

## 数学関数

　基本的な数学関数には以下のような関数があります。

Int(x) 整数化 Int(2.5) --> 2 Int(-2.5) ---> -3

Sqr(x) 平方根

Abs(x) 絶対値

Log(x) 底をeとする自然対数

　Excelのワークシート関数にも同等の機能を持ったものが揃っていますので、そちらを使っても構いません。Logはワークシート関数とVBA関数とで底が異なっていますので、使用時は注意して下さい。ワークシート関数は以下のように使います。

WorksheetFunction.Round(x, 0) ' 小数点以下1桁目を四捨五入して整数にする

WorksheetFunction.Ln(x) ' 底はe

WorksheetFunction.Log(x) ' 底は10

## オブジェクト変数の使用

　本節の内容は高度なので、必要なときに読み返して下さい。

　Range("A2") などをオブジェクトと呼びます。これを変数のように扱うことができます。まずは、下の例を見て下さい。

Dim a As Object ' As Range でもよい

Set a = Range("A1")

a.Interior.Color = RGB(255, 200, 200)

　aはオブジェクト変数です。オブジェクト変数は必ず宣言する必要があります。

　2行目でオブジェクト変数にRange("A1") を代入しています。普通の変数とは異なり、オブジェクト変数への代入はSetが必要です。

　Rangeオブジェクトが含むセルの個数は1個とは限りません。Range("A1") なら一つですが、Range("A1:B3") ならA1, A2, A3, B1, B2, B3の6個です。Range("A1:B2,D4") ならA1, A2, B2, B3, D4の5個です。

　複数のセルを含むRangeオブジェクトを1個1個のセルに分解するには以下のようにします。次のプログラムを実行してみて下さい。

Dim a As Object, c As Object

Set a = Range("A1:C2")

i = 1 ' オブジェクトの順番を調べるために

For Each c In a ' 1, 2, 3, 4, .... と書き込む

c.Value = i ' .Value は省略可能

i = i + 1

Next c

　For Eachという新しいループが登場しました。

For Each c In a

においてcはObject型かVariant型であることが必要です。aはObjectか配列であることが必要です。上のように書くと、aの要素から1つずつ取り出してcに入れます。

　Range("A1:C2") は6個のセルを含みます。この場合、For Each のループは2行×3列=6回実行されます。cはRange("A1") → Range("B1") → Range("C1") → Range("A2") → Range("B2") → Range("C2") の順番になります。上記のプログラムを実行することでその順番が分かります。

　Selectionは現在選択されているRangeオブジェクトを表します。次のように使います。

Dim a As Object

Set a = Selection

Debug.Print a.Address

Debug.print a.Count

　a.Addressで "A1:B3" のような「範囲を表す文字列」を取り出すことができます。

　a.Count でセルの個数を取得できます。

　a(1), a(2),... a(n) で個々のセルにアクセスできます（注：a(0) にアクセスするとエラーが発生します）。ただし、この方法は領域が長方形のときしか使えません。「Ctrl + 左クリック」で飛び飛びの場所を指定した場合には使うことができません。

　「ctrl + 左クリック」を2回行って2つのセルを選択した状態で、その2つのセルの内容を入れ替えるには以下のように書きます。

Dim a As Object, c As Object

Dim address(2)

Set a = Selection

i = 1

For Each c In a

address(i) = c.address

i = i + 1

Next c

tmp = Range(address(1)).Value

Range(address(1)).Value = Range(address(2)).Value

Range(address(2)).Value = tmp

　上記のプログラムで、末尾3行の .Value は省略可能です。

　Excelでオブジェクト変数を使うのは、筆者の経験では以下のような場合です。

* 選択した複数のセルを扱うとき
* テキストボックス、図形要素を扱うとき

1. 「Office 365のExcel」「OneDriveを使用している」の2つの条件が満たされたとき、ThisWorkbook.Pathの値はhttps://xxxxx.jp-mysharepoint.com/personal/xxxxx\_cc\_nara/Documents/....という値になり、プログラムを実行すると「ファイルが見つかりません」というエラーが出ます。そのときは、OneDriveの設定のOfficeタブにおいて「Office アプリケーションを使用して、開いているOfficeファイルを同期する」のチェックを外して下さい。 [↑](#footnote-ref-1)
2. ThisWorkbook.Pathを省略して、

   filename = "out.txt"

   と書くと、現在のフォルダの直下にout.txtを作成します。現在のフォルダはイミディエイトウィンドウでPrint CurDirと打ってEnterを押すと得られます。Excelの場合、ドキュメントフォルダの直下です。

   filename = "\out.txt"

   と書くと、ドキュメントフォルダがあるドライブのルートにout.txtを作成します。 [↑](#footnote-ref-2)