2024.7.12

# 基本的なアルゴリズム

　本書では原則としてDimは省略しますが、例題を入力するときにDimとして宣言すべき変数名が分かっていた方が、プログラムの入力が容易なので、本章で掲載するほとんどのプログラムはDimを省略せずに書きます。

## 値の交換

【例題】

　シート「値の交換」を使います。A1 ～ A10とB1 ～ B10を入れ替えなさい。

【解答】

 Dim i, tmp

 For i = 1 To 10

 tmp = Range("A" & i)

 Range("A" & i) = Range("B" & i)

 Range("B" & i) = tmp

 Next i

　変数の値を交換するときの重要な原則があります。



(a) ダメな方法



(b) 適切な方法

図3.1　値の交換

　Range("A" & i) と Range("B" & i) を入れ替えようと思って、

 Range("A" & i) = Range("B" & i) ①

 Range("B" & i) = Range("A" & i) ②

と書くのはダメです。図3.1 (a) のように ① を実行した直後の段階で、最初にRange("A" & i) に入っていた値が破壊されてしまいます。次の ② の実行時にRange("A" & i) に入っている値は ① で代入されたRange("B" & i) の値ですから、Range("B" & i) の値は変化しません。

　値を入れ替えるときは図3.1 (b) のように組みます。最初に ① を実行して、あらかじめセルRange("A" & i) の値を待避用変数tmpに入れます。次に ② を実行してRange("A" & i) にRange("B" & i) の値を代入し、最後に ③ を実行してRange("B" & i) に待避しておいたRange("A" & i) の値を代入します。tmpはtemporary （一時的な）の略語です。プログラムの中にtmpあるいはtempという名前の変数を見つけたときは、一時的な値を格納する変数であると判断してよいでしょう。その他に、一時的な値を格納する変数名としてはbuf（緩衝器を表すbufferの略）もよく使われます。

　このように、2つの変数（セル）の値を交換するには、値を一時的に保持する変数が1個必要です。

　この例題においては、Aの値を待避させましたが、以下のようにBの値を待避させても構いません。

 tmp = Range("B" & i)

 Range("B" & i) = Range("A" & i)

 Range("A" & i) = tmp

## 合計と平均

　関数を使っても人数、合計点、平均点を求めることはできますが、本節ではプログラムを組んで求めます。

【例題】

　シート「合計と平均」を使います。A列に名前、B列に得点が入っています。人数をb15, 合計点をb16, 平均をb17に入れるプログラムを作りなさい。

【解答】

 Dim i, num, sum

 num = 0 ' 人数をカウントする変数

 sum = 0 ' 合計点を加算する変数

 For i = 2 To 13 ' 2 行目から 13 行目に点数が入っている

 num = num + 1

 sum = sum + Range("B" & i)

 Next i

 Range("b15") = num

 Range("b16") = sum

 Range("b17") = sum / num

　numは人数をカウントするための変数、sumは合計得点を加算するための変数です。どちらも初期値は0です。そして、1人分の得点を加算したとき、

 num = num + 1

 sum = sum + Range("B" & i)

という計算を実行しています。人数が12人なのでnum = num + 1の計算をせずに、

 Range("B17") = sum / 12

としても同じ値が得られます。しかし、欠席者などノーカウントにすべき人がいる場合に対応するには、この例のように書く必要があります。

　C列は欠席者がいる場合の得点です。欠席者の欄は "欠" が入っています。人数、得点、平均点を求めるにはForの内側を以下のように変更します。結果を書き込む列はC列に変更して下さい。

 If Range("C" & i) <> "欠" Then

 num = num + 1

 sum = sum + Range("C" & i)

 End If

　以下のように書いても動作は同じです。

 If Range("C" & i) = "欠" Then

 Else

 num = num + 1

 sum = sum + Range("C" & i)

 End If

## 最大値と最小値

【例題】

　シート「最大と最小」を使います。B列に数値が入っています。最大値を求めてB18に書き込みなさい。ただし、得点は0～100の範囲内にあるものとします。

　アルゴリズムのフローチャートを図3.2に示します。本来はフローチャート記号の枠内にコードは書きませんが、ここでは枠の上側に処理の内容、下側にコードを記入しています。



図3.2　最大値を求めるアルゴリズム

　

(a) セルの内容が暫定最大値より小さい　　(b) セルの内容が暫定最大値より大きい

図3.3　アルゴリズムの解説

　maxは暫定最大値を入れるための変数です。その時点までに見つけた最大値を保持します。Ifにおける処理内容を図示したのが図3.3です。暫定最大値とセルを比べ、暫定最大値の方が大きい場合、何もしません。暫定最大値よりもセル内の値が大きいときは、その値をコピーします。イコールのときは値をコピーしても暫定最大値の値は変わらないので、値のコピーはしてもしなくても構いません。不要な処理を実行する必要はないので、図3.2ではコピーをする条件に = は含めていません。

【解答】

 Dim i, max

 max = -999 ' 暫定最大値

 For i = 2 To 15

 If Range("B" & i) > max Then ' 暫定最大値より大きな値を発見

 max = Range("B" & i) ' したら暫定最大値を更新する

 End If

 Next i

 Range("B18") = max

　コードの説明をします。現在処理する行を保持する変数がiです。i行目までに見つけた最大値を入れる変数がmaxです。maxより大きな値を見つけたときは、maxを更新します。4行目の > を >= に変更しても結果は同じです。

　暫定最大値maxには得点の下限値0より小さな数を入れておきます。ここでは-999という極端な値を入れていますが-1でも構いません。maxの初期値として0を入れるのはダメです。計算終了時にmaxが0のままだったとき、「全員欠席」だったのか、「出席者が全員0点」だったのか区別できません。A列の値を順に見ていき、maxより大きな値を見つけたときは、maxを更新します。

　「暫定最大値maxの初期値」を「点数の範囲の下限より小さい値に設定する」という方法はあらかじめ点数の範囲が分からない場合には使えません。点数の範囲が分からない場合は「最初に見つけた値を暫定最大値に設定する」という処理が必要です。このとき「1人目の値を最大暫定値に設定する」とやってしまうと、本例題では欠席者は想定していませんが、1人目が欠席者だったときに動作がおかしくなります。「時々誤った値が得られるプログラム」というタチの悪いバグが混入してしまいます。最初に見つけた値を暫定最大値にする方法については、フラグ変数のところで学習します。

【注意】

　"abc" のような文字列と数値を比べることは、本来はできません。文字列と数値を > や < で比較したときは、エラーを出してプログラムをストップするべきですが、Excel VBAは比較できてしまいます。そして、If 文において "abc" のような文字列は「いかなる数値よりも大きい」と判定され、"" は「いかなる数値よりも小さい」と判定されます。これはExcel VBAの欠陥と言えるでしょう。

## 変数の使用

　変数は数値や文字列を入れておくための箱です。セルと同じと考えてよいです。変数はセルと比べると、以下のメリット・デメリットがあります。

● メリット

・max, sumのように、意味のある名前を付けることができる。Range("A1") では何を表すのか分からない[[1]](#footnote-1)。

・Range("A1") のようにタイプするより、文字が少ないのでタイプが楽。

・代入が高速[[2]](#footnote-2)

● デメリット

・内容を見るにはセルに代入するか、Debug.Print を使ってイミディエイトウィンドウに表示させる必要がある。

【例題】

　最大値を求めるプログラムに付け足して、最高得点獲得者の名前をB23, B24, B25 ...のようにB列23行目以降に並べるプログラムを作成しなさい。

【解答】

 j = 23

 For i = 2 To 15

 If Range("B" & i) = max Then

 Range("B" & j) = Range("A" & i)

 j = j + 1

 End If

 End If

　最大値を求めるForループが終わった後、最高点獲得者をピックアップするForループを組みます。最高点獲得者が複数いる場合に対応するには、このように2回ループを組む必要があります。

【発展】

　If Range("A1") = "" Then と書くと、A1に半角や全角の空白が入っていたときに条件が成立しません。「セルの中は」「セルの中は空白のみ」のどちらにもマッチするには、以下のように書きます。

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

　Trim( ) は括弧の中の文字列の左右の空白を切り落とす命令です。例えば、

 a = Trim(" abc de ")

 b = Trim(" ") ' 半角空白と全角空白のみの文字列

とすると、a の内容は "abc de" となります。左右の空白が切り落とされました。b の内容は "" となります。Trimは文字列を扱う関数です。文字列を扱う関数は後の章で学習します。

【例題】

　シート「班分け」を使います。A列の生徒を4つの班に分けます。B列2行目から下方にかけて1 2 3 4 1 2 3 4 1 2 .... と記入するプログラムを作成しなさい。

【解答】

 Dim i, a

 a = 1

 For i = 2 To 15

 Range("B" & i) = a

 a = a + 1

 If a = 5 Then

 a = 1

 End If

 Next i

　班を表すaという変数を導入し、初期値を1とし、1ずつカウントアップし、5になると、1に戻しています。If a = 5 Thenは If a >= 5 Then としても動作は同じです。このパターンは今後の例題や課題の解法として頻出するので、マスターして下さい。

　余りをとる関数を使う方法もあります。3.8節で学習します。

【例題】

　シート「並べ替え」を使います。A2～A26に名前が入っています。4つの班に分けます。A2から順にD2, E2, F2, G2, D3, E3, F3, G3, D4, E4, .... の順に並べるプログラムを作りなさい。

【解答】

 Dim i, row, column

 row = 2

 column = 4

 For i = 2 To 26

 Cells(row, column) = Range("A" & i)

 column = column + 1 ' 次に入れる場所は 1 つ右の列

 If column = 8 Then ' 8 列目 (H) を示したときは、

 row = row + 1 ' 次の行の

 column = 4 ' 4 列目 (D)

 End If

 Next i

　横の列を変数で指定するにはCells(i, j) の形式を使うことが必要です。名前を入れるセルの行と列をそれぞれ変数row, columnで表します。

* 変数名として英語のrow (行), column (列) を使いましたが、英語が嫌いな人はgyou, retuという変数名を使っても構いません。

## Do Whileループ

【例題】

　シート「無限ループ」を使います。A列に1, 3, 9, 27.... と値が3倍ずつ増える数列を入れなさい。ただし、10000未満の範囲とします。

【解答】

 Dim i, num

 num = 1 ' 1, 3, 9, 27 と変化する値を入れる変数

 i = 1 ' 次に記入する行番号を記憶する変数

 Do While True ' ループのはじまり

 Range("A" & i) = num

 i = i + 1

 num = num \* 3

 If num >= 10000 Then

 Exit Do ' Exit Doでループから抜ける

 End If

 Loop ' ループの終わり

　Forループは実行する回数が決められたループでした。もう一つのループの組み方がWhileループです。Whileループは条件が成立している間、繰り返します。一般形は以下の通りです。

 Do While 条件

 処理1

 処理2

 ....

 Loop

　ループの範囲はDo While ～ Loopです。Do While の時点で条件が成立するか否かを判定し、成立する場合はLoopまでの処理を行い、そうでない場合はLoopの次の行から処理を行います。1回Loopまでの処理を終えたら、再びDo Whileの場所に戻って条件をチェックします。

　解答はDo While True ～ Loopと書いています。Trueは条件が成立することを意味します。このループは常に条件が成立するので、無限に繰り返します。無限ループと言います。ループから脱出するにはExit Doを実行します。Exit Doを実行すると、Loopの次の文を実行します。Do While True ではなくDoとだけ書いても同じです。

※　プログラムが終わらなくなってしまったら？

　Whileループを組むと、プログラムが終わらなくなってしまうことがあります。ループからの脱出条件が適切でない場合、無限に実行を続けてしまいます。そのようなときは、ESCキーを押して下さい。実行を中断します。

　Excel VBAの画面が薄くなり、ESCキーを押しても止まらない場合があります。そのときはESCキーを押しながらタスクバー上のアプリのアイコンをランダムに何回かクリックします。[[3]](#footnote-3)。

　これで止まると思いますが、それでも止まらない場合は、「Ctrl」「Alt」「Delete」の3つのキーを同時に押すか、タスクバー上で右クリックして、タスクマネージャーを起動し、Excelを選択して「タスクの終了」をクリックして下さい。タスクバーからExcelを終了させた場合、プログラムが保存されず失われる可能性が高いですが、仕方ありません。

　無限ループを含むプログラムを開発するとき、For i = 1 To 10000 のように、大きな数のForとExit Forを用いてプログラムを開発し、プログラムが正常動作することを確認した後、For → Do While True, Next → Loop, Exit For → Exit Doという変換を施す方法があります。

※　無限ループを最初に学習することについて

　本書ではWhileループの学習において、Do While Trueで無限ループを組み「ループから抜ける条件を記述する」というスタイルを最初に学習します。脱出条件はループの先頭、中間、末尾のどこに書いてもOKです。

　一方、通常のプログラミングの教科書はDo While i < 5のように「ループを継続する条件を記述する」というスタイルを最初に学習します。このパターンの場合、ループの先頭に条件を書くことになります。

　本書が特殊な構成になっている理由は4.9節で説明します。

## 二重ループ

　Forループの中にもう一つForループを作ってみます。2重ループと言います。

 Dim i, j

 For i = 1 To 2

 For j = 1 To 3

 Debug.Print "i = " & i & " j = " & j

 Next j

 Next i

　イミディエイトウィンドウは以下のようになります。

i = 1 j = 1

i = 1 j = 2

i = 1 j = 3

i = 2 j = 1

i = 2 j = 2

i = 2 j = 3

　外側のループ (ループ変数はi) と内側のループ (ループ変数はj) の関係をよく見て下さい。i = 1 の状態で j = 1, 2, 3と変化し、次に i = 2 の状態で j = 1, 2, 3 と変化します。ループは2×3 = 6回実行されます。

　シート「2重ループ」を使います。次のプログラムを実行するとどうなるでしょう？　まずは、実行する前にどうなるかを予想し、次に実行してみて下さい。

 Dim i, j, num

 num = 1

 For i = 1 To 10

 For j = 1 To 10

 Cells(i, j) = num

 num = num + 1

 Next j

 Next i

【例題】

　シート「並列抵抗」を用います。電気回路において、2個の抵抗の値をa, bとするとき、並列接続したときの合成抵抗rは

$$r=\frac{a×b}{a+b}$$

で得られます。aとbが1～100 Ω（整数）のとき、合成抵抗が整数になる組み合わせを全て求め、A列とB列に2つの抵抗値を入れ、C列に合成抵抗値を入れなさい。

　ただし「a=3, b=6」という組み合わせと「a=6, b=3」という組み合わせは重複しているので、a≦bの場合のみを対象とします。

【解答】

　ある数rが整数か否かは次式で判定します。

If r = Int(r) Then

　Int(　　) は括弧の中の整数部分を取り出します。たとえば、Int(4.6) → 4となります。Int(4.6) と 4.6 は等しくないので、r が整数でない場合、Ifは不成立となります。

　変数rが整数か否かを判定するのに、rとInt(r)を比べるというのは、とてもスマートな方法だと思いませんか？　私は初めてこの方法を見たとき「何て賢い方法なんだ！」と感動しました。

　先ほどのInt(　　) の説明は実は不正確でした。Intを正確に表現すると「( ) の中の数値を超えない最大の整数を返す」となります。引数が負の場合、以下のような結果となります。

Int(-1.1) → -2

Int(-2.5) → -3

Int(-3.8) → -4

　a≦b という条件の下で、aとbを1～100まで変化させるにはどうすれば良いでしょうか？　言い換えると、ループの変数a, bが以下のように変化するにはどうすれば良いでしょうか？

 a = 1 b = 1 ' aが1のときbは1～100

 a = 1 b = 2

 a = 1 b = 3

 a = 1 ......

 a = 1 ......

 a = 1 b = 100

 a = 2 b = 2 ' aが2のときbは2～100

 a = 2 b = 3

 a = 2 b = 4

 a = 2 ......

 a = 2 ......

 a = 2 b = 100

 a = 3 b = 3 ' aが3のときbは3～100

 a = 3 b = 4

 a = 3 b = 5

 a = 3 ......

 a = 3 ......

 a = 3 b = 100

　上記のループは以下のように組みます。これまでループ変数の初期値や終了値は定数でした。このように、変数とすることもできます。なお、ループ変数はi, j, k,...を使うと以前に述べましたが、数式に合わせるため、この例題はループ変数にa, bを用います。

 Dim a, b

 For a = 1 To 100

 For b = a To 100 ' ループ変数bの初期値はa

 Next b

 Next a

　あとは、合成抵抗を求め、整数となる組み合わせを、並べるだけです。「見つけた組み合わせを書き込む行番号」を保持する変数iが必要です。

 Dim i, a, b, r

 i = 2

 For a = 1 To 100

 For b = a To 100

 r = (a \* b) / (a + b)

 If r = Int(r) Then

 Range("A" & i) = a

 Range("B" & i) = b

 Range("C" & i) = r

 i = i + 1

 End If

 Next b

 Next a

　上記の (a \* b) / (a + b) は a \* b / (a + b) と書いても同じです。括弧を付けると以下のメリットがあります。

* 分子と分母の範囲が明確に区別され、式が見やすい
* (a \* b + c) のように分子が変化した場合、a \* b / (a + b) の表記法だと 括弧を付け忘れて a \* b + c / (a + b) と書いてしまう恐れがあるが、上記の表記法だと間違えることはない。

【例題】

　シート「レポート提出」を使います。G列にレポート提出者の番号のリストがあります。このリストに基づいて、C列にレポート提出者は "〇", 未提出者は "×" を記入するプログラムを作りなさい。

【解答1】

　「G列から番号を1つずつ取り出し、取り出した番号の学生の行に〇を入れる」という考え方で、C列を埋めます。2重ループを使っており、外側のループがG列（レポート提出者のリスト）、内側のループがA列（全学生）です。

　G列の最後まで見終わった後、未提出者の欄に "×" を入れます。

 Dim i, j, num

 For i = 2 To 10 ' i は G 列の行番号

 num = Range("G" & i) ' G列から 1 つ番号を取り出す

 For j = 2 To 14 ' j は A 列の行番号

 If num = Range("A" & j) Then ' A 列の該当する行を見つけた

 Range("c" & j) = "〇"

 End If

 Next j

 Next i

' 未提出者の欄に "×" を入れる

 For j = 2 To 14

 If Range("C" & j) = "" Then

 Range("C" & j) = "×"

 End If

 Next j

　上記の方法は人間のやり方をシミュレートした方法でした。コンピュータの変数は紙とは異なり、書き換えが容易なので、最初に全員 "×" を入れておくという方法もあります。If を使わなくて済むので、少しシンプルです。

【解答1別解】

 Dim i, j, num

' 最初に全員 "×" をつける

 For j = 2 To 14

 Range("C" & j) = "×"

 Next j

' 提出者を "○" にする

 For i = 2 To 10 ' i は G 列の行番号

 num = Range("G" & i) ' G列から 1 つ番号を取り出す

 For j = 2 To 14 ' j は A 列の行番号

 If num = Range("A" & j) Then ' A 列の該当する行を見つけた

 Range("c" & j) = "〇"

 End If

 Next j

 Next i

【解答2】

　解答1はG列のレポート提出者について、提出者に○をつけていきました。

　別の方法として、名簿番号順に一人一人について「レポートを出したか否か」を判定する方法があります。以下のようなプログラムになります。外側のループがA列、内側のループがG列です。

 Dim i, j, num

 For i = 2 to 14

 num = Range("A" & i)

 For j = 2 to 10

 If num = Range("G" & j) Then ' 提出を確認

 Range("C" & i) = "○"

 End If

 Next j

 If Range("C" & i) = "" Then ' 提出を確認できない人は×を入れる

 Range("C" & i) = "×"

 End If

 Next i

【解答2　別解】

　最初に "×" を入れておくと、If文はなくても良いです。

 Dim i, j, num

 For i = 2 to 14

 num = Range("A" & i)

 Range("C" & i) = "×"

 For j = 2 to 10

 If num = Range("G" & j) Then ' 提出を確認

 Range("C" & i) = "○"

 End If

 Next j

 Next i

## フラグ

　フラグと呼ばれる変数の使い方があります。フラグ変数は二者択一の値をとります。通常は0か1です。yesかnoかを表します[[4]](#footnote-4)。通常は0がno, 1がyesを表します。

　2重ループの節でシート「レポート提出」を扱いました。解答1は人間のやり方をシミュレートした方法でした。解答2はフラグ変数を使って以下のように書くことができます。

【解答2　フラグ変数版】

 Dim i, j, num, find

 For i = 2 To 14 ' i は A 列の行番号

 num = Range("A" & i) ' 名簿番号を取り出す

 find = 0 ' フラグ変数を初期化する

 ' 0:見つからない（初期値） 1:見つけた

 For j = 2 To 10 ' j は G 列の行番号

 If num = Range("G" & j) Then

 find = 1 ' 見つけたら find = 1

 End If

 Next j

' j のループが終わった時点でレポート提出者は find = 1

' 未提出者は find = 0

 If find = 0 Then

 Range("C" & i) = "×"

 Else

 Range("C" & i) = "○"

 End If

 Next i

　ここではフラグ変数findを「0:未提出 1:提出」として使います。最初に0を設定しておきます。A列から1人取り出し、名簿番号をnumに入れます。次にG列を1つ1つ見てゆき、見つかったならfind = 1に設定します。jのループを終えた時点で、名簿番号numを見つけたなら1, そうでないなら0という値がfindに入っています。最後にfindの値によって、"○" か "×" を書き込みます。

【例題】

　入力ミスにより、G列に書き込まれている名簿番号のうち、在籍しない学生がいます。

　解答1の解答を改造し、名簿番号にない学生を見つけたとき、イミディエイトウィンドウに「名簿番号〇は名簿にありません」と表示するよう改造しなさい。

【解答1　フラグ追加版】

　改良したプログラムを以下に示します。

 Dim i, j, num, flag

 For i = 2 To 10

 num = Range("G" & i)

 flag = 0

 For j = 2 To 14

 If num = Range("A" & j) Then

 Range("C" & j) = "〇"

 flag = 1

 End If

 Next j

 If flag = 0 Then

 Debug.Print "名簿番号 " & num & " は名簿にありません"

 End If

 Next i

【例題】

　シート「最大」を使います。最高点をB17に書き入れるプログラムを作成しなさい。

　3.3節で最大値を求める方法を学習しました。3.3節では「得点の取りうる最小値より小さい値を、暫定最大値として設定する」という方法を用いました。

　ここでは得点の取りうる範囲が分からないことを仮定します。従って、最初に見つけた値を暫定最大値に設定しなさい。ただし、欠席者がいるので、暫定最大値は2行目とは限りません。

【解答】

 Dim i, max, set\_max, score

 set\_max = 0 ' set\_max = 0 は max を

 ' 設定していない状態

 For i = 2 To 15

 score = Range("B" & i)

 If score <> "欠" Then ' 欠席者でない場合に処理を行う

 If set\_max = 0 Then ' 初めて数値を見つけた

 set\_max = 1

 max = score

 EndIf

 If score > max Then ' 暫定最大値より大きな値を見つけたとき

 max = score

 End If

 End If

 Next i

 Range("B17") = max

　Range("B" & i) を何回も書くのは面倒なので、scoreという変数に入れています。set\_max = 0のIfが成立するときにscore > max は成立しないので、以下のように書いた方が良いかも知れません。

 If set\_max = 0 Then ' 初めて数値を見つけた

 set\_max = 1

 max = score

 ElseIf score > max Then ' 2 人目以降

 max = score

 End If

　3.3節では変数maxに最初に-9999のようなデータが取り得る範囲よりも小さな値を入れておきました。-9999というのは異常な数値なので、特別な意味（ここでは最大値の初期値）を持つことを想起させます。maxという変数にflagを兼用させていると考えることもできます。使用する変数を1個節約できますが、変数maxに2つの意味を持たせることになり、プログラムを読む人を混乱させるかもしれません。

【例題】

　シート「重複削除」のA列に数値が入っています。上から順番にB列にコピーします。ただし、一度出現した数値は無視しなさい。

【解答】

　以下のように考えます。A列のi行目 (以下セルAiと表す) をB列にコピーするとき、B列はrow行目まで埋まっていると仮定します。セルAiと同じ値の数値がB1～Bjまでの間に存在すれば、コピーはしません。そうでないとき、jを1増やして、セルAiをセルBjにコピーします。

 Dim i, j, row, num, find

 row = 0

 For i = 1 To 30

 num = Range("A" & i)

 find = 0

 For j = 1 To row ' 既に現れた数かチェックする

 If num = Range("B" & j) Then

 find = 1

 End If

 Next j

 If find = 0 Then ' 1回も現れてないならコピーする

 row = row + 1

 Range("B" & row) = num

 End If

 Next i

## 余りをとる

　a÷bの余りをcに入れるには、以下のようにModを使います。

 c = a Mod b

【例題】

　シート「余り」を使います。名前の列と1行目は既に書いてあります。B列の2行目～15行目を右のように埋めるプログラムを作成しなさい。

【解答】

　3.4節で変数を使った方法を以下のように学習しました。

 Dim i, j

 j = 1

 For i = 2 To 15

 Range("B" & i) = j

 j = j + 1

 If j = 5 Then

 j = 1

 End If

 Next i

　余りをとる演算Modを使うと、以下のようにスマートに書けます。ただし、プログラムの分かりやすさは変数 j を使う方が分かりやすいと思います。

 For i = 2 To 15

 Range("B" & i) = (i - 2) Mod 4 + 1

 Next i

　割り算の余りをとる命令を持たないプログラミング言語を使うことはないと思いますが、自力で余りを求める方法を紹介しておきましょう。整数部分を取り出す関数Intを使うとa ÷ bの余りcは以下のように書けます。

 c = a - Int(a / b) \* b

ユークリッドの互除法 (Euclidean Algorithm)

　最大公約数を求めるアルゴリズムとして、ユークリッドの互除法という巧妙な方法があります。aとb（ただしa > b）の最大公約数を求めるアルゴリズムを図3.4に示します。



図3.4　ユークリッドの互除法

　文字にすると以下のようになります。

1. a ÷ bの余りをamariとする。
2. amari = 0 ならbが最大公約数である。
3. そうでないなら、変数の値を a ← b, b ← amari と更新し、1. に戻る。

【例題】

　シート「互除法」を使います。セルA1とA2に入れた数値の最大公約数をユークリッドの互除法を用いて求め、A3に入れるプログラムを作成しなさい。ただしA1 > A2を仮定します。

【解答】

 Dim a, b, amari

 a = Range("A1")

 b = Range("A2")

 Do While True

 amari = a Mod b

 If amari = 0 Then

 Exit Do

 End If

 a = b

 b = amari

 Loop

 Range("A3") = b

　IfからLoopの手前までは以下のように書いても動作は同じです。

 If amari = 0 Then

 Exit Do

 Else

 a = b

 b = amari

 End If

　読者はどちらが良いプログラムだと思いますか？　筆者は「余りが0のときは処理を終了してループを抜ける。そうでない場合は、通常動作」という考え方に従うなら、前者が良いと考えます。

## 状態変数

　Excelの表を処理するプログラムではあまり出てこないかも知れませんが、制御系のプログラムやゲームのプログラムを組む場合、現在の状態を保持する変数（状態変数と呼ぶ）を作り、その値によって処理内容を変えるのは、プログラムの常套手段です。これをステートマシン（状態機械）と呼びます。

【例題】



　シート「状態変数」を見て下さい。A列に文字が入っています。上のように、連続して文字列が入っているブロックごとにC, D, E, F,.....にコピーするプログラムを作成しなさい。

【解答】

　フローチャートを図3.5に示します。状態変数jyotaiは「コピー対象文字列探索中」「コピー中」の2つの状態のいずれかをとります。この例題の場合、状態は2個なので、フラグ変数と考えることもできます。



図3.5　状態変数のフローチャート

　プログラムを以下に示します。

 Dim i, a, jyotai, row, column

 jyotai = 0

 column = 3

 For i = 1 To 27

 a = Range("A" & i)

 If jyotai = 0 Then ' 状態：文字未発見

 If a <> "" Then ' 文字発見

 jyotai = 1 ' 状態：文字コピー中

 row = 1 ' 1 行目から書く

 Cells(row, column) = a

 row = row + 1

 End If

 Else ' 状態：文字コピー中

 If a = "" Then ' 空セル発見

 jyotai = 0 ' 状態：文字未発見

 column = column + 1 ' 次の列へ

 Else

 Cells(row, column) = a ' 文字をコピー

 row = row + 1 ' 次の行へ

 End If

 End If

 Next i

　jyotai という変数が状態変数です。jyotai = 0 のとき文字を探している（空セルを読んでいる）状態、jyotai = 1 のとき文字コピー中です。

1. オブジェクト変数を使って、たとえば Set a1 = Range("A1") と書くと、Range("A1") と書く代わりに a1 と書くことができます。このようにオブジェクト変数の使用により、「名前が分かりにくい」と次の項目である「タイプする文字数が多い」の問題は解消できますが、オブジェクト変数の概念は少し難解なので本書では扱いません。 [↑](#footnote-ref-1)
2. セルに値を代入すると、画面を更新する必要がありますが、変数の場合その必要がないためであると思われます。本テキストで学習するような簡単なプログラムでは、その差は体感できないかも知れません。 [↑](#footnote-ref-2)
3. <https://www.excel-chunchun.com/entry/2019/01/13/010851>　「えくせるちゅんちゅん」という名前のブログです。著者は「ことりちゅん」という方です。Hatena Blog id:Kotori-ChunChun [↑](#footnote-ref-3)
4. 本来はBoolean型変数を使うべきですが、「変数の型」「TrueとFalseという値の学習」などをまだしていないので、ここでは普通の変数を使います。 [↑](#footnote-ref-4)