1. 簡単なプログラミング

簡単なプログラミングの実習をします。例題による練習をしながら、少しずつレベルを上げていきます。【実習】はかならず自分で挑戦してから解答を見てください。いろいろな解法があるので正解は 1 つではありません。自分の解法と解答を比べてみましょう。

1.1. コード

さっそくコードウィンドウに簡単なコードを書いてその出力を確かめてみましょう。簡単な出力の実験です。ABC という文字をセル D3 に出力させます。

```
Sub p11() ‘簡単な出力の実験
   Cells(3, 4) = “ABC” ‘ABC という文字をセル D3 に出力
End Sub
```

このコードは、

```
Sub [プログラム名]
   [statements]
End Sub
```

という構造になっています[1]。このようにプログラムの本体を Sub と End Sub で囲みます。Sub の後に**プログラム名**を書きます[2]。この枠内にある[statements]がプログラムの本体です。Cells(3, 4)は「座標(3, 4)のセル」、つまり第 3 行第 4 列のセルを示します[3]。Excel シートでは D3（D 列 3 行）にあたります。D3 のように列(D)・行(3)のように示す

[1] Sub [プログラム名]を入力すると自動的に End Sub がつきます。

[2] 同じモジュールの中にプログラム名が同じコードを書くことはできません。p12, p12x, xz, …などのように名前の一部を付加したり、変更してあれば大丈夫です。プログラム名の終わりにかっこ()をつけていますが、このことは後で説明します。プログラム名は、たとえば A1, B5 などのセル名と重なるものと使うと実行時のトラブルの原因となります。ここでは記号のようなプログラム名を使っていますが、実際は、たとえば NewFile や「新シート準備」のように、プログラムの機能を 1 語で示すような名前をつけると、後で参照するときに便利です。

[3] 「行」はシートの最上段から下に向かって数えていきます。「列」はシートの最左端から右に向かって数えていきます。

方法と、Cells(3,4)のように行(3)・列(4)で示す方法があります。両者は行と列の順番が異なるので注意が必要です。プログラムでは一般にCells(3, 4)のような指定の仕方をしたほうが操作がしやすいので、この方法をとります。

次の「=“ABC”」は、ABC という文字をイコール(=)の左の項（ここでは Cells(3,4)）に入れることを意味します。一般にプログラム内の「*a*=*b*」は *a* に *b* の値を**代入する**ことを意味します。

結果

現在選択されているシートを見ると次が出力されています。

	A	B	C	D
1				
2				
3				ABC

【課題 1.1a】次のコードを実行するとどのような結果になるか予想しなさい。その実行結果を確かめさない。

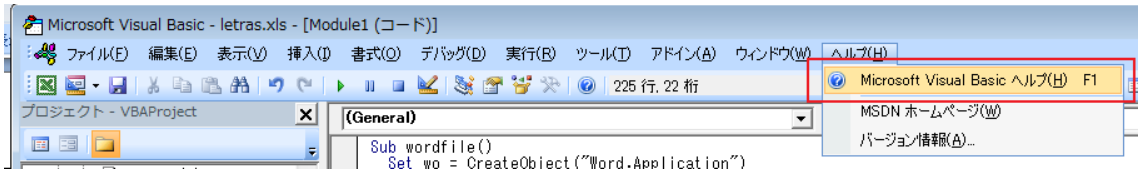
```
Sub e11a() ‘簡単な出力の実験
   Cells(1, 1) = “Sheet”
   Cells(1, 2) = “Row”
End Sub
```

【課題 1.1b】ABC という文字を B2 のセルに出力し、DEF という文字を B3 のセルに出力するコードを書きなさい。

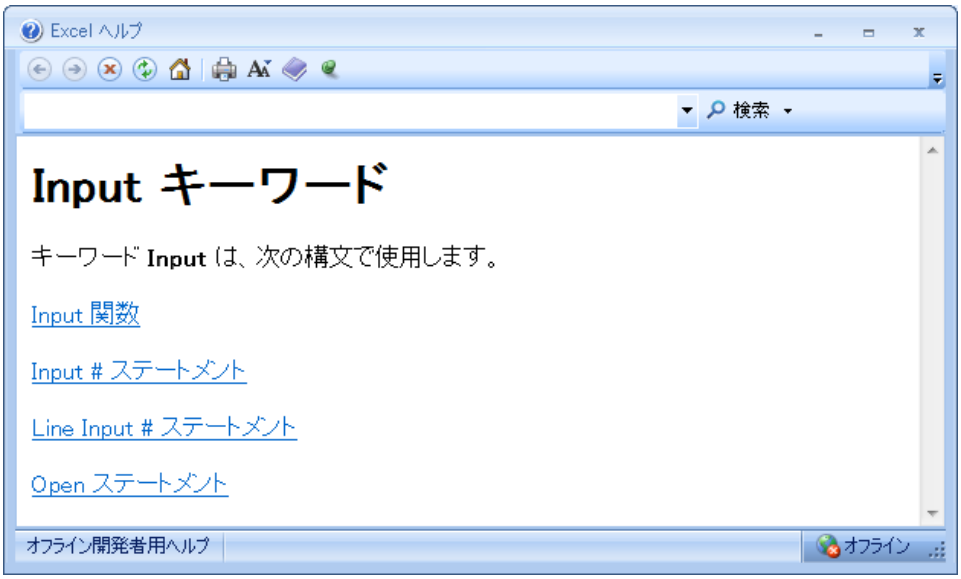
1.2. ヘルプ

コードは英語を使っていますが、その技術的な意味については、VBE のヘルプが役に立ちます。

＊「ヘルプ」→「Microsoft Visual Basic ヘルプ(H) F1」



＊たとえば、Input で検索すると次の画面が現れます[4]。



それぞれの技術的な説明は最初はわかりにくいかも知れませんが、少しずつ慣れていくはずです。巻末の参考書も役に立ちます。インターネットでも多くのサイトから有用な情報が送られていますから、検索してください。

1.3. 変数・関数・制御

ここではプログラムに必要な基本的な要素として、変数と関数と制御を扱います。変数には定数や変数の内容を代入することができます。関数によって、一定の値を変数に返すことができます。制御はプログラムの流れを決めます。これらの要素を使うことによってプログラムは多くの機能を果たします。

1.3.1. 変数

数値データの変数

はじめに数値の変数を扱います。ここでは数値のうち整数を扱います。整数には次の 2 種類の**データ型**があります。

[4] ショートカットは、コードの中の Input にカーソルを置き F1 を押します。

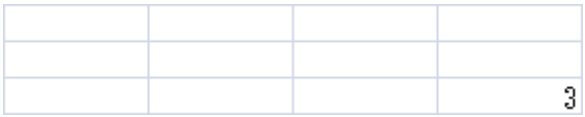
整数型（Integer）　　-32,768 ~ 32,767 (約 3 万 2 千)
長整数型（Long）　　-2,147,483,648 ~ 2,147,483,647 (約 20 億)

これらのデータ型は、扱う数値の大きさによって使い分けます。正確な数値の範囲を覚えておく必要はありませんが、数値が 32000 以下である予想できるときは整数型を使い、それを超えるときは長整数型を使うとよいでしょう。データ型はプログラムの最初の**宣言文**で示します。変数の宣言文は Dim で始め、As の後にデータ型を示します。次のようにコードの先頭に Option Explicit をつけると、かならず変数のデータ型を宣言しなければなりません。

コード

```
Sub p131a() '足し算
   Dim intA As Integer '整数型
   Dim intB As Integer '整数型

   intA = 1 'intA に 1 を代入
   intB = 2 'intB に 2 を代入

   Cells(3, 4) = intA + intB '足し算の結果をセル D3 に出力
End Sub
```

結果



演算子

上のコードの intA + intB で使った「+」は足すことを示す**演算子**です。演算子には次の記号が使われます。

- 足し算：例 intA + intB
- 引き算：例 intA - intB
- 掛け算：例 intA * intB
- 割り算：例 intA / intB

【課題 1.3.1a】上のコードに演算子を使った文を書き足して、次の結果を出すコードを書きなさい。

足し算	引き算	掛け算	割り算
3	-1	2	0.5

（割り算の結果が代入されるのは宣言文にある変数ではなく、Cells なので整数でなくてもかまいません。）

文字データの変数

文字（列）も変数に代入することができます。文字なのに「変数」というのは少しわかりにくいかも知れませんが、「変数」が variable であることを理解すればよいでしょう。つまり、変数に値として文字が代入されるのです。

文字データの変数には次のデータ型を使います。

- 可変長文字列型(String)：0 バイトから 2 ギガバイトまでの範囲

型宣言文字

変数の後に次の文字を加えることによって、データ型を宣言することができます。次のデータ型には型宣言文字があります。

%:　整数型(Integer)
&　長整数型(Long)
$　文字型(String)

Dim で、これらの記号をつけた変数を宣言すれば、As Integer などの指定は不要になります。このような型宣言文字を宣言文や他の代入文などで最初でたとえば A$のように指定しておけば、以降は A だけで文字型を示していることになります。ここでは、繰り返しの For...Next などで使う、i&, j&などは簡略化して i, j としますが、他はデータ型を意識するために、かならず書くようにします[5]。

コード

```
Sub p131b() '文字データ
  Dim A$, B$

  A$ = "太郎は" 'A$に代入
  B$ = "学生です。" 'B$に代入
```

[5] プログラムが読みやすくなります。

```
    Cells(2, 3) = A$ & B$ '連結させて出力
End Sub
```

文字列を連結させるにはアンド(&)の記号を使います。

結果

		太郎は学生です。	

【課題 1.3.1b】上のコードを書き換えて、次の結果をセル A1 に出力するコードを書きなさい。

太郎は学生です。
花子は学生です。

1.3.2. 関数

数学で使う「**関数**」は数字データを扱い、たとえば、f(x) = 2x + 1 のように書きます。このような関数の中で使われる変数 x を「**引数**」（ひきすう）といいます。そして関数は引数に従って一定の値を示すことを「**返す**」という表現を使います。たとえば上の f(x)の引数 x が 1 のときは関数 f(x)は 3 を返し, x が 2 ならば関数 f(x)は 5 を返します。引数が 2 つ以上のときもあります。たとえば、関数 f(x, y) = 2x + y の場合、x=1, y = 2 のとき f(x, y)は 4 を返します。

このような関数の考え方はプログラミングでも使われます。上のような数学の関数は f(x) = 2x + 1 のように自分で定義しますが、プログラミングで使う関数は多くの場合、すでに用意されているものを使います。

ここでは文字データを扱う次のような文字列操作関数を扱います。

Len(*A$*)	文字列 *A$*の文字数を返す
Left(*A$, B%*)	文字列 *A$*の左から *B%*文字分を返す
Right(*A$, B%*)	文字列 *A$*の右から *B%*文字分を返す
Mid(*A$, B%, C%*)	文字列 *A$*の左から数えて *B%*文字の位置から *C%* 文字分を返す

たとえば Left(“abcdefghi”, 3) は“abcdefghi”という文字列の左から 3 文字分である “abc” を返します。また、Mid(“abcdefghi”, 3, 2)は”cd” を返します。

コード

```
Sub p13c() '関数 Left
   Dim A$

   A$ = "In principio creavit Deus caelum et terram." 'A$に文字列を代入

   Cells(1, 1) = A$ '文字列全体を出力
   Cells(2, 1) = Len(A$) '文字列の長さを出力
   Cells(3, 1) = Left(A$, 7) '左の 7 文字分を出力
End Sub
```

Left 関数で切り出した文字数は空白も含めます。

結果

In principio creavit Deus caelum et terram.				
43				
In prin				
pri				

【課題 1.3.1b】次のコードを読み、結果を予想しなさい。その後で、実行して確かめなさい。

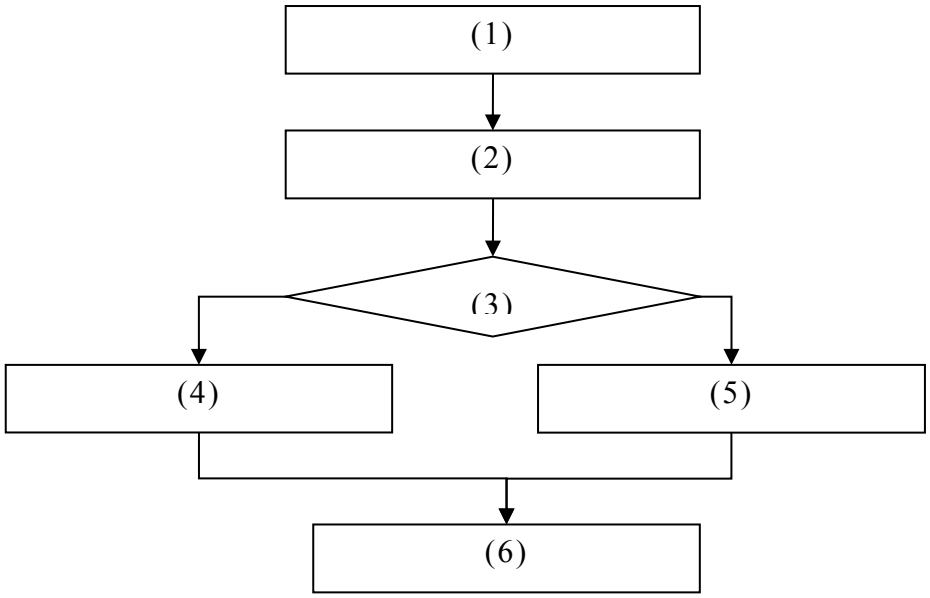
```
Sub p131x() '関数 Left
   Dim A$

   A$ = "In principio creavit Deus caelum et terram." 'A$に文字列を代入

   Cells(1, 1) = A$ '文字列全体を出力
   Cells(2, 1) = Len(A$) '文字列の長さを出力
   Cells(3, 1) = Left(A$, 7) '左の 7 文字分を出力
   Cells(4, 1) = Mid(A$, 4, 3) '文字列の 4 番目から 3 文字を出力
End Sub
```

1.3.3. 制御

プログラムの流れを条件判断や繰り返しを指定して**制御**します。

条件判断

プログラムは基本的に上から下に向かって処理を進めますが、次のように途中で条件によって流れを分けることがあります。



上の図で(3)の部分の条件によってプログラムの流れは(4)、または(5)に分岐します。以下のコード内の If はこのような**条件判断**に使われます。

コード

```
Sub p133a() '文字列の長さの判断
   Dim A$

   A$ = "principio" 'A$に文字列を代入

   Cells(1, 1) = A$ '文字列全体を出力

   If Len(A$) <= 8 Then
      Cells(2, 1) = "比較的短い単語です。" '文字列の長さの判断を出力
   Else
      Cells(2, 1) = "比較的長い単語です。" '文字列の長さの判断を出力
   End If
End Sub
```

結果

繰り返し

プログラムの流れの中で同じ処理を何度でも繰り返すことが必要なときは次の For *i* = *a* To *b*... Next という制御文を使います。For と Next で囲まれた処理を、*i* の値が *a* から 1 つずつ増えて *b* の値になるまで繰り返します。

コード

```
Sub p23b() '逆順
   Dim A$, S$, i&  ‘宣言

   A$ = "TERRAM" 'A$に文字列を代入

   For i = 1 To Len(A$)  ‘A$の文字数まで繰り返す
      S$ = Mid(A$, i, 1) & S$  ‘i 番目の 1 文字を前に連結
   Next

   Cells(1, 1) = A$ '文字列全体を出力
   Cells(2, 1) = S$ '逆順を出力
End Sub
```

上のコードの例では Mid で切り出した文字(1 文字)を strS に前から連結させていきます。ここでは A$に“TERRAM”が代入されていますから、Len(A$)は 6 です。i の増加に従って Mid(A$, i, 1)と S$がどのように変化するのかを次に示します。

i	**Mid(A$, i, 1)**	**S$**
1	T	T
2	E	ET
3	R	RET
4	R	RRET
5	A	ARRET
6	M	MARRET

結果

TERRAM
MARRET

【課題 1.3.3】上のコードを書き換えて、次の結果を出すコードを書きなさい。

i	文字	文字列
1	T	T
2	E	ET
3	R	RET
4	R	RRET
5	A	ARRET
6	M	MARRET

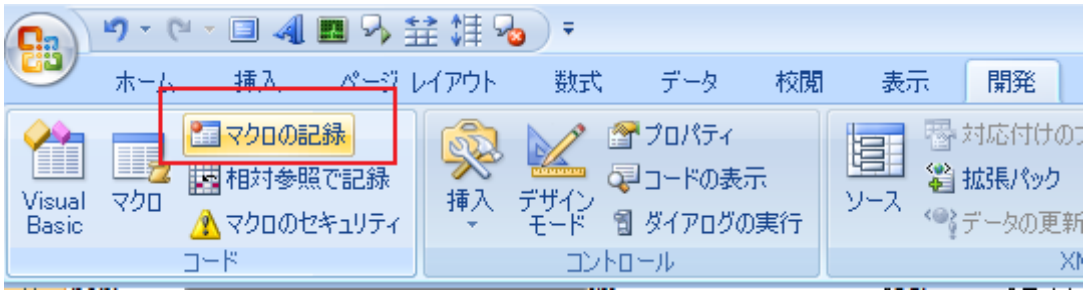
1.4. マクロの記録

プログラムのコードは「マクロの記録」によって知ることができます。

1.4.1. ワークシートの挿入

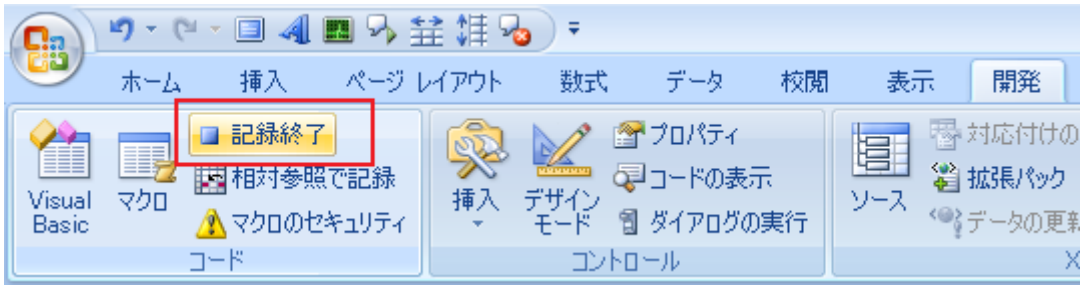
はじめに、シートの挿入という簡単な操作を例に「マクロの記録」を実行します。

(1) 「開発」→「コード」→「マクロの記録」をクリック



(2) Shift+F11 を押すと、現在のシートの左にシートが挿入されます。

(3) 「開発」→「コード」→「記録終了」のボタンを押してください。このボタンは「マクロの記録」と同じ位置にあります。



さて、この後でマクロの VBE のコード画面を見ると次のような一連のコードが自動的に書き込まれています。[Alt]+F11 によってコードが見られます。これはユーザーが行った操作のコードを記録したものです。

コード

```
Sub Macro1()
'
' Macro1 Macro
'


'
    Sheets.Add
End Sub
```

このコードの最初にある Macro1 の数字 1 は以前に作成されたマクロの数によって変化します。次に続く 3 行と、その後 1 行置いて 1 行はシングルコーテーション(')ではじまるコメント文です。この部分はプログラムの実行にかかわりません。以下では省略します。ここでは Sheets.Add に注目しましょう。これは Sheets を対象にして、それを追加(Add)するという意味です。

次に上の(2)の段階で、シートのタブの最後にある「ワークシートの挿入」タブを押してみましょう。

これを記録すると以下のコードになります。

コード

```
Sub Macro2()
   Sheets.Add After:=Sheets(Sheets.Count)
End Sub
```

ここでは Sheets.Add に続く After:=でワークシートを追加する位置を決めます。この場合は Sheets(Sheets.Count)の後ということですが、これは現在扱っているワークブックの中にあるシート数を Sheets.Count で数え、そこで返された数、たとえば 14 を引数として Sheets(引数)にあてまめます。そうすると、シートがシート番号 14 番の後に追加されるのです。Count が Sheets **コレクション**がもっている数という**プロパティ**です。

ここで使われている Add は、その前にある Sheets に加え、Count はそれを数えています。この Add のように、その前にある要素に一定の操作をする機能があるものを「**メソッド**」と呼びます。

【課題 1.4.1a】次のコードを実行し、結果を確かめなさい。この結果からコードの意味を解釈しなさい。

```
Sub Macro3()
   Sheets.Add Before:=Sheets(1)
End Sub
```

【課題 1.4.1b】次のコードを実行し、結果を確かめなさい。この結果からコードの意味を解釈しなさい。

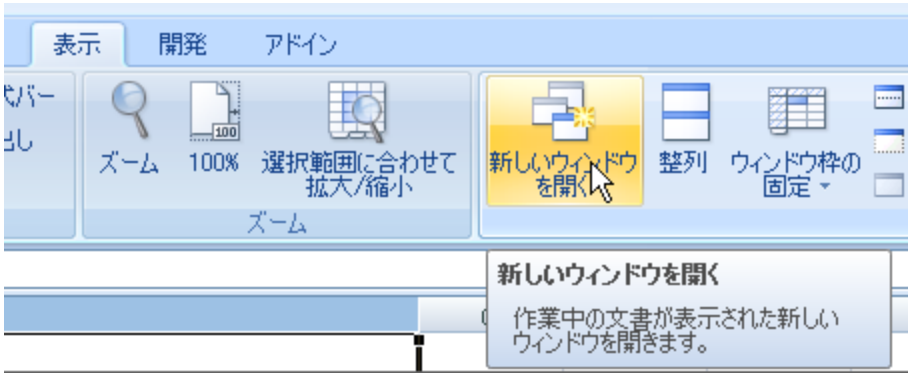
```
Sub Macro4()
   Sheets.Add Count:=3
End Sub
```

【課題 1.4.1c】マクロの記録によってシートのコピーのコードを探しなさい。「現シート」を示すコードは ActiveSheet です。

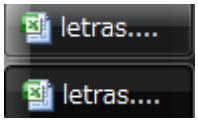
1.4.2. ウィンドウ

ウィンドウを開く操作のコードを探しましょう。

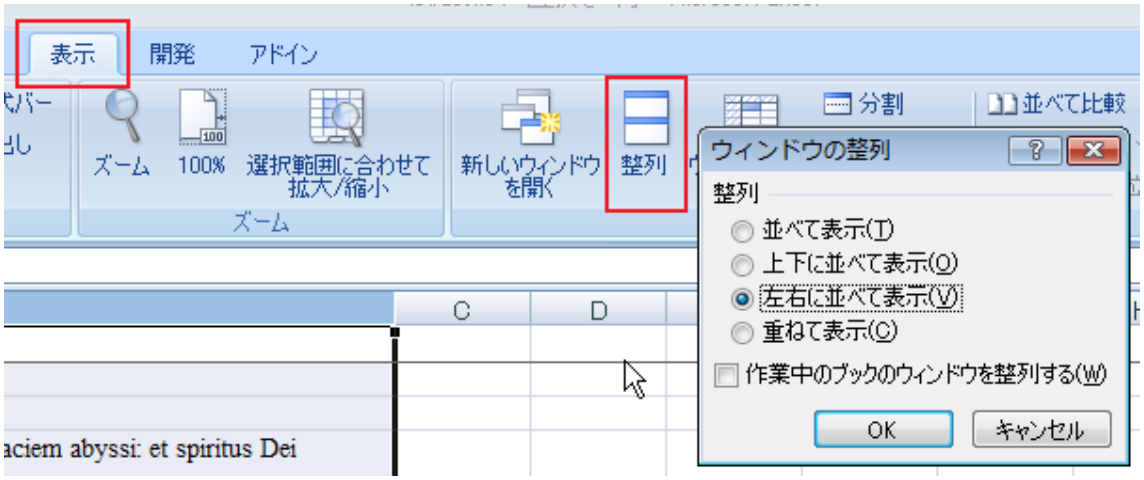
(1)「開発」→「コード」→「マクロの記録」をクリック
(2)「表示」→「ウィンドウ」→「新しいウィンドウを開く」



結果：同じワークブックが 2 つのウィンドウに表示されます[6]。



(3)「表示」→「ウィンドウ」→「整列」→「左右に並べて表示」



(4) 「開発」→「コード」→「記録終了」

コード

```
Sub Macro7()
    ActiveWindow.NewWindow
    Windows.Arrange ArrangeStyle:=xlVertical
End Sub
```

最初のコメント文は省略しました。ここでは「新しいウィンドウを開く」と「左右に並べて表示」という 2 つの操作をしました。それぞれが 1 行ずつ、全体で 2 行のコードになっています。NewWindow と

[6] 一方を閉じ、残ったウィンドウを最大化すれば、元の状態に戻ります。

Arrange というメソッドに注目しましょう

【課題 1.4.2】「ウィンドウの整列」で「上下に並べて表示」を選択し、そのコードを確かめ、全体を解釈しなさい。

1.4.3. 並べ替え

次に、並べ替えの操作をします。並べ替えの操作は、「データ」→「並べ替えとフィルタ」グループで「昇順」を選びます。

これまでと同じ方法で、並べ替えのコードを調べましょう。[A1:D25] ほどの規模の適当なシートを使って、D1 のセルを選択します。

結果

	A	B	C	D
1	c.1-2	代表形	補足	品詞
2	vacua	vacuus	a, um	adj
3	unus	u:nus	a, um	adj
4	unum	u:nus	a, um	adj
5	universo	u:niversus	a, um	adj
6	universis	u:niversus	a, um	adj
7	universam	u:niversus	a, um	adj

コード[7]

```
ActiveWorkbook.Worksheets("Voc").Sort.SortFields.Clear
ActiveWorkbook.Worksheets("Voc").Sort.SortFields.Add
Key:=Range("D1"), _
    SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:= _
    xlSortTextAsNumbers
With ActiveWorkbook.Worksheets("Voc").Sort
    .SetRange Range("A1:G399")
    .Header = xlYes
    .MatchCase = False
    .Orientation = xlTopToBottom
    .SortMethod = xlPinYin
    .Apply
End With
```

全体のプログラムは、

ActiveWorkbook.Worksheets("Voc").Sort.SortFields.Clear

から始まります。この中の ActiveWorkbook.Worksheets("Voc")は現在のブックの Voc というシートを指しています。その後にある Sort が重要です。この Sort という言葉について VBE の**ヘルプ**を見ると、次の情報が得られます[8]。

式 .Sort(Key1, Order1, Key2, Type, Order2, Key3, Order3, Header, OrderCustom, MatchCase, Orientation, SortMethod, DataOption1, DataOption2, DataOption3)

これは、Range を対象にして、(Key1, ...)という条件で Sort をするという意味です。上の式では(...)の中で順番が指定されていますが、先に見たマクロの記録によるコードを見ると Key:=というような書き方

[7] Excel2007 のコードです。Excel2003 では異なるコードが記録されます。この場合の Excel2007 のコードを Excel2003 で使うことはできません。Excel2003 では、次のヘルプで示されるコード形式、または Sub sort01()のコードを使ってください。

[8] Sort という語の一部にカーソルを置いて、[F1]キーを押すとヘルプ画面が立ち上がります。その中の「Range.Sort メソッド」をクリックしてください。

もあることがわかります。Key はソートの「キー」を指しますが、他にここで重要な条件は、Header の有無、大小文字の区別(MatchCase)、そしてオーダー（昇順 xlAscending・降順 xlDescending）です。

キーの位置は先のマクロの記録では Range("D1")というレンジで示されています。ここでは列を指定して並べ替えるので、これを Columns(4)と書くこともできます。ヘッダーがなければ Header の指定は必要ありません。オーダーは昇順ならば指定する必要はありません。向きも必要に応じて Orientation（ソートの向き：上下＝xlTopToBottom, 左右：xlLeftToRight）も使います。上下ならば指定する必要はありません。

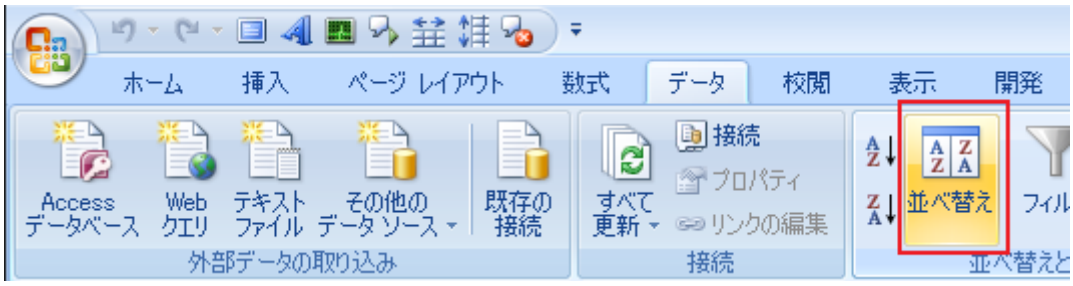
上のコードを書き換えて、次のようなコードにまとめることができます。

```
Sub sort01()
   ActiveSheet.UsedRange.Sort Header:=xlYes, MatchCase:=False, _
      Key1:=Columns(2), Order1:=xlDescending, _
      Key2:=Columns(1), Order2:=xlAscending, _
      Orientation:=xlTopToBottom
      'ソート（ヘッダー：有、大小文字区別：無、キー1,2：カラム、昇順、向き：上下）
End Sub
```

ここでは第 1 キーを B 列、第 2 キーを A 列にしてありますから、「代表形」を第 1 キーとし、それが同じときは「C.1-2」を第 2 キーとして並べ替えます。

【課題 1.4.3a】上のコードの条件（ヘッダー、大小文字、キー、オーダー、向き）を変えて実験し、気づいたことを書きなさい。

【課題 1.4.3b】次の「並べ替え」のボタンを押して、並べ替えの条件を変えてマクロを記録しなさい。上のコードとの違いに注意して気づいたことを書きなさい。



1.5. 行と列・セル・レンジ

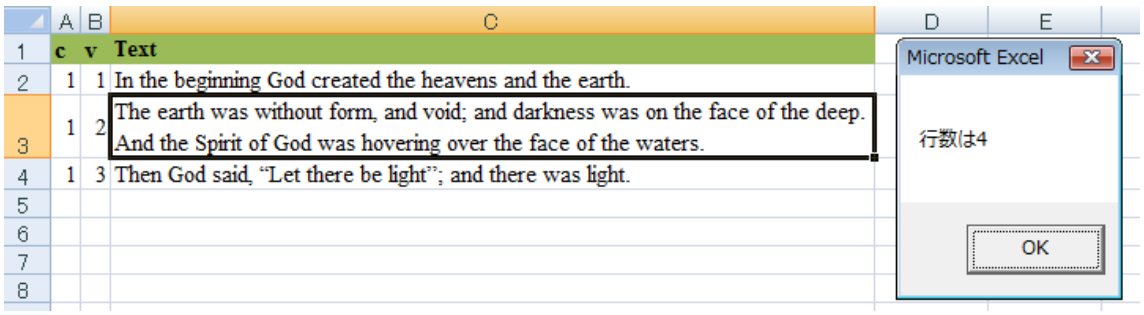
1.5.1. 行と列の操作

ワークシートにあるデータの行数や列数を数えるプログラムを作ります。それをメッセージボックスに出力します。メッセージボックスの OK ボタンを押すまでは、他の操作ができません。

コード：

```
Sub p341a() '使用している行数
   MsgBox "行数は" & ActiveSheet.UsedRange.Rows.Count
End Sub
```

結果



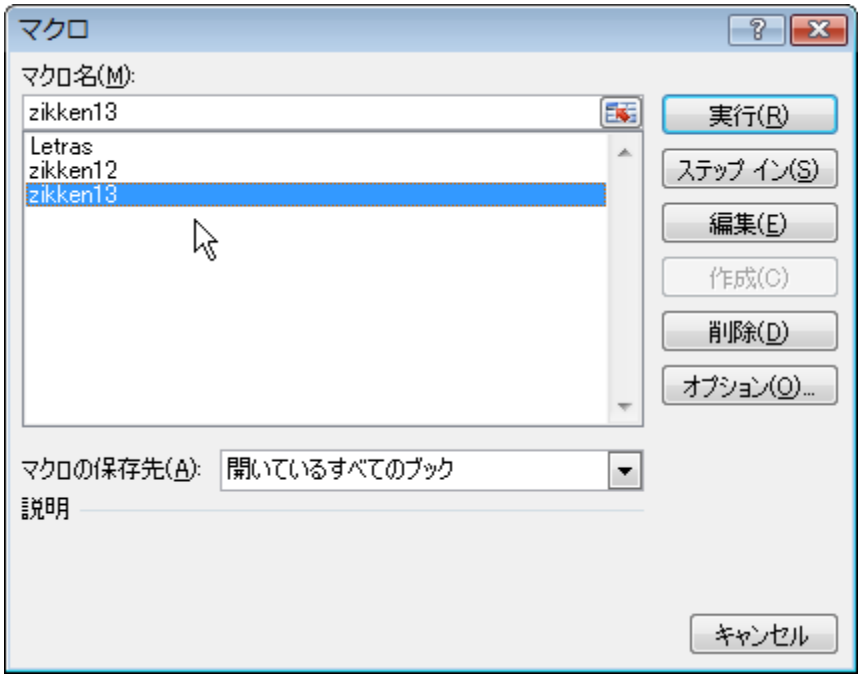
【課題 1.5.1a】列の数をメッセージボックスに出力するコードを書き、実験しなさい。「列」は Columns を使います。

シートから実行

これまでは VBE からプログラムを実行していましたが、次にシートから実行してみましょう。[Alt]+F5 でマクロのフォームを出し、マクロ名を選択し「実行」ボタンを押します。マクロ名を直接ダブルクリ

ックしてもよいでしょう。

＊シートから[Alt]+F8→該当するマクロ名を選んで実行



【課題 1.5.1b】VBE から実行する方法とシートから実行する方法を比較して気づいたことを書きなさい。

1.5.2. セルの操作

セルは行と列で指定されます。たとえば C2 は、第 2 行、第 3 列にあるセルを指します。私たちは先に特定のセルに文字列を出力する方法を見ました（→1.1）。ここでは、For ... Next を使って、複数のセルに出力する方法を扱います。

データ：c.1

(C:V)　Text
(1:1)　En el principio creó Dios los cielos y la tierra.
(1:2)　La tierra estaba desordenada y vacía, las tinieblas estaban sobre la faz del abismo y el espíritu de Dios se movía sobre la faz de las aguas.
(1:3)　Dijo Dios: «Sea la luz». Y fue la luz.

コード：

```
Sub p152() '使用している行数
```

```
    Dim SheetName$, RowCnt&, i& 'シート名、使用行数、作業用変数

    SheetName$ = ActiveSheet.Name '現シート名

    ActiveSheet.Copy After:=Sheets(Sheets.Count) '新シートにコピー
    RowCnt& = ActiveSheet.UsedRange.Rows.Count '使用行数

    Columns(1).Insert '1 列挿入
    Cells(1, 1) = "Sheet" 'タイトル

    For i = 2 To RowCnt& '行数まで繰り返す
        Cells(i, 1) = SheetName$ 'シート名を出力
    Next
End Sub
```

結果：

Sheet	(C:V)	Text
Sheet3	(1:1)	En el principio creó Dios los cielos y la tierra.
Sheet3	(1:2)	La tierra estaba desordenada y vacía, las tinieblas estaban sobre la faz del abismo y el espíritu de Dios se movía sobre la faz de las aguas.
Sheet3	(1:3)	Dijo Dios: «Sea la luz». Y fue la luz.

SheetName$ = ActiveSheet.Name によって、現在のシート名をSheetName$に代入しておきます。

ActiveSheet.Copy After:=Sheets(Sheets.Count)では、現在のシートをコピーし、現在存在するすべてのシートの後に、つまり一番右の位置にペーストします。

RowCnt& = ActiveSheet.UsedRange.Rows.Count によって現在のシートで使用している領域の行数を RowCnt&に代入します。Columns(1).Insert は第 1 列、つまり A 列の前に 1 行挿入するという意味です Cells(1, 1) = "Sheet"でセル A1 に"Sheet"という文字列を出力します。次の For … Next による制御文で i がステップ 1 で 2 から行数(RowCnt&)になるまで(2, 3, …), For … Next の処理が繰り返されます。はじめは i = 2 なので、Cells(2, 1)に、先に代入しておいたシート名SheetName$を出力します。以下同様に i が 3, 4, ….と続き、RowCnt&になるまでこの処理が繰り返されます。

【課題 1.5.2】上のコードに加えて、次のように連続番号(No.)を出力するプログラムを書きなさい。

Sheet	No.	(C:V)	Text
Sheet3	1	(1:1)	En el principio creó Dios los cielos y la tierra.
Sheet3	2	(1:2)	La tierra estaba desordenada y vacía, las tinieblas estaban sobre la faz del abismo y el espíritu de Dios se movía sobre la faz de las aguas.
Sheet3	3	(1:3)	Dijo Dios: «Sea la luz». Y fue la luz.

1.5.3. レンジの操作

レンジ(Range)を使うことによって、シート内のデータの入力や出力を一括して処理することができます。たとえば、Range(Cells(1,1), Cells(1,3))は Cells(1,1)から Cells(1,3)を含む範囲を指します。つまり、A1 から C1 までの範囲です。次は最初の 1 行にあるタイトル全体の背景色を指定するコードです。

コード：

```
Sub p153
   [A1:B1].Interior.Color = RGB(150, 255, 0) '背景色を指定
End Sub
```

レンジ[A1:C1]は Range(Cells(1, 1), Cells(1, 3))のように指定することもできます。これは次のコードと同じ結果になります。

```
Sub p153b
   For i = 1 to 2
      Cells(1, i).Interior.Color = RGB(150, 255, 0) '背景色を指定
   Next
End Sub
```

結果：

	A	B
1	(C:V)	Text
2	(1:0)	Cap.1
3	(1:1)	In principio creavit Deus caelum et terram.
4	(1:2)	Terra autem erat inanis et vacua: et tenebrae erant super faciem abyssi: et spiritus Dei ferebatur super aquas.
5	(1:3)	Dixitque Deus. Fiat lux. Et facta est lux.

上のコードで Interior.Color の部分が背景色を代入する部分です。色を指定する方法の 1 つとして、**RGB 関数**を使う方法があります。これは 3 つの引数を使いますが、それぞれ赤(R)、緑(G)、青(B)に対応します。以下は代表的な色を出すためのそれぞれの引数です。

色	R	G	B
赤	255	0	0
緑	0	255	0
青	0	0	255
白	255	255	255
黒	0	0	0
シアン	0	255	255
マゼンタ	255	0	255
黄色	255	255	0

他にも上のコードで指定したように、3 つの引数を組み合わせることによって微妙な色合いを使うことができます。引数と実際の色の関係は、「テーマの色」の「その他の塗りつぶしの色」で確認することができます。

＊適当なセルを選択し、右クリック→「テーマの色」→「その他の塗りつぶしの色」→「ユーザー設定」

【課題 1.5.3】以下に示すように A 列が太字（ボールド）にするコードの Range の部分(*)を完成させなさい。（太字にするためのコードはマクロの記録を使えばわかります。）

```
Sub e153()
   Dim RowCnt& '使用行数

   RowCnt& = ActiveSheet.UsedRange.Rows.Count '使用行数
```

```
    Range(Cells(*, *), Cells(*, *)).Font.Bold = True '範囲を太字に
End Sub
```

結果：

	A	B
1	(C:V)	Text
2	(1:0)	Cap.1
3	(1:1)	In principio creavit Deus caelum et terram.
4	(1:2)	Terra autem erat inanis et vacua: et tenebrae erant super faciem abyssi: et spiritus Dei ferebatur super aquas.
5	(1:3)	Dixitque Deus. Fiat lux. Et facta est lux.

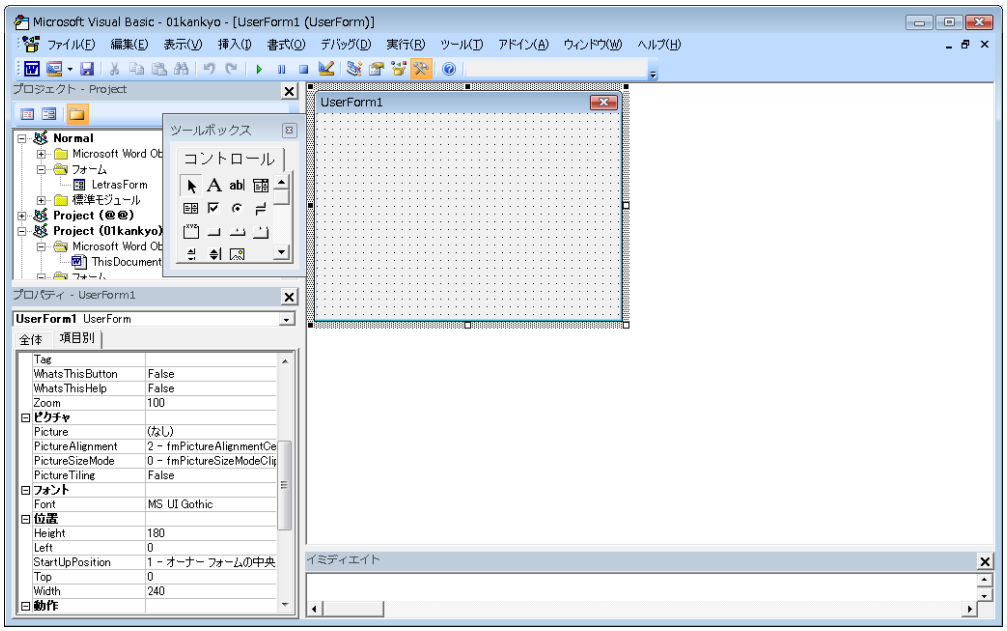
1.6. ユーザーフォームとイベント

ユーザーがプログラムを使いやすいようするために、ユーザーフォームを作成しましょう。はじめにユーザーフォームを作成し、その後このユーザーフォームを標準モジュールから立ち上げます。

1.6.1. ユーザーフォームの作成

ユーザーフォームを作成するにはプロジェクトウィンドー内のブックに、ユーザーフォームを挿入します。

＊「挿入」→「ユーザーフォーム」



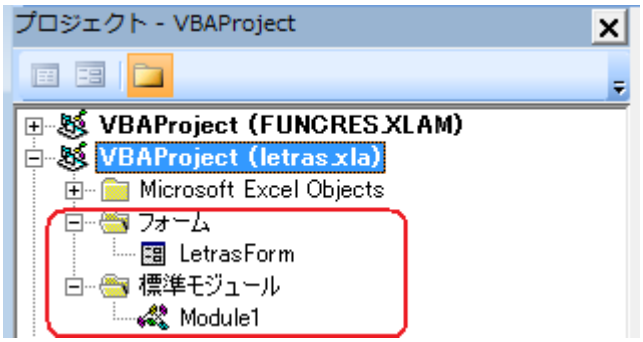
ユーザーフォームを起動

＊「標準モジュール」に次のコードを書き込み、F5 で実行します。

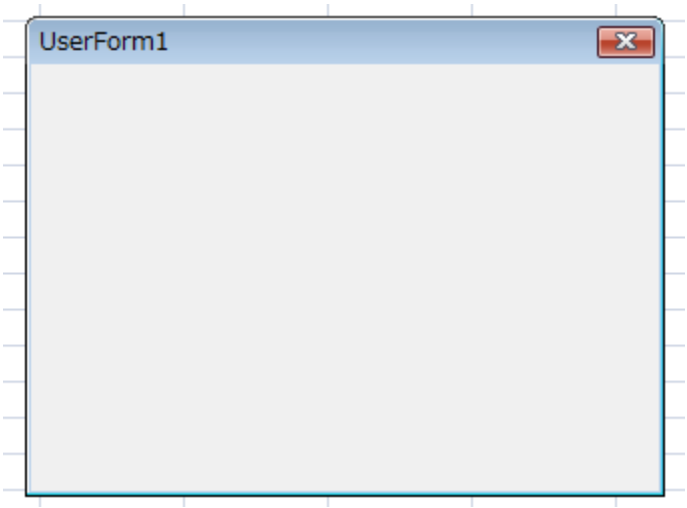
コード

```
Sub p13() 'ユーザーフォーム
    UserForm1.Show 'ユーザーフォームを立ち上げる
End Sub
```

標準モジュールとフォーム



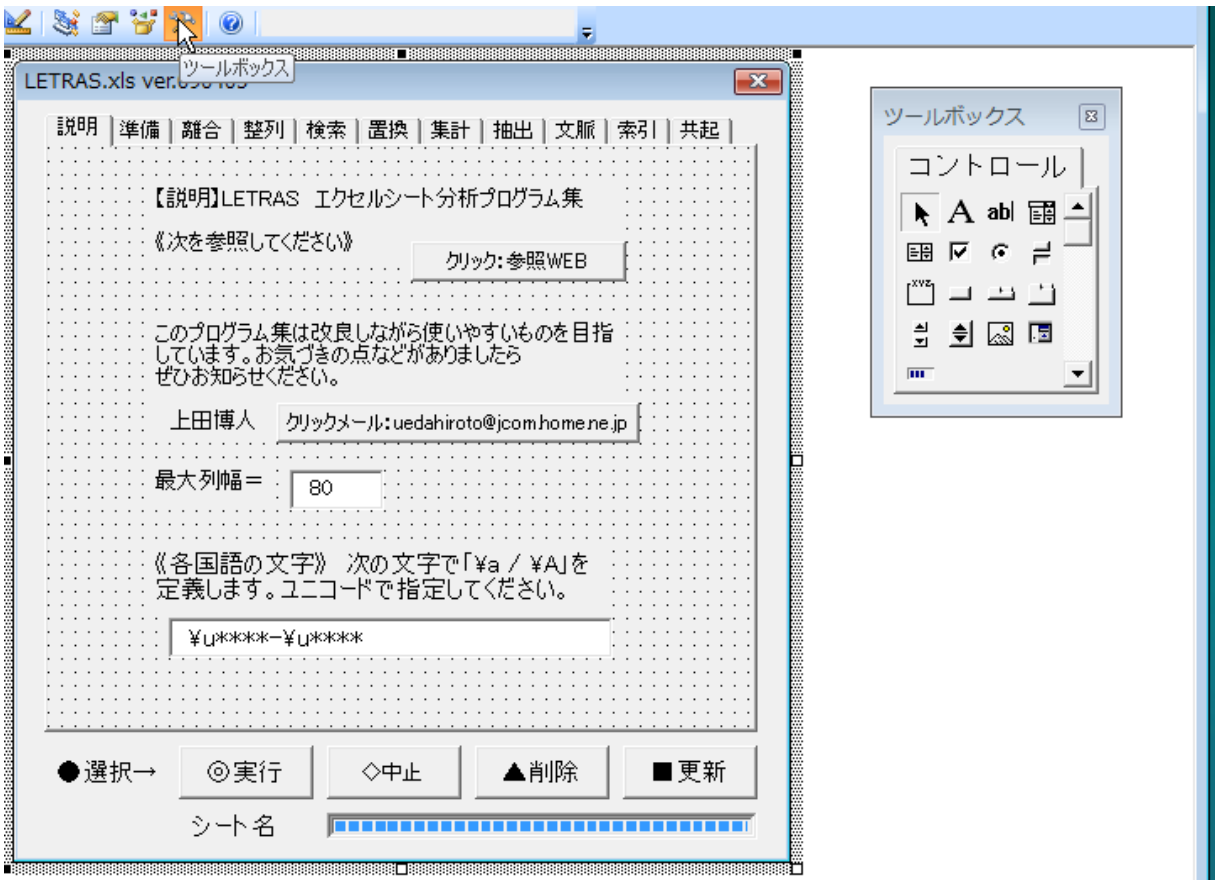
結果：



【課題 1.6.1a】上のコードを解釈しなさい。

【課題 1.6.1b】フォームの大きさを変えて実験しなさい。

1.6.2. ツールボックスとコントロール



コントロール

ラベル	説明や簡単な指示を書き込みます。
テキストボックス	データをキーインで入力したり、データを表示したりします。
コンボボックス	リストの一覧から選択したり、一覧にないデータを入力したりします。
リストボックス	リストの一覧から選択します。
チェックボックス	チェックを入れて選択します。
オプションボタン	フレームコントロールと組み合わせて、選択項目の中から１つを選択くします。
フレーム	コントロールをグループ化します。
コマンドボタン	ボタンをクリックして一定の処理をします。
マルチページ	コントロールをページごとにグループ化します。

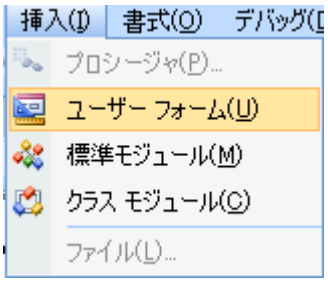
他に、トグル、タブストリップ、スクロールバー、スピン、イメージなどがあります。

【課題 1.6.2】フォームにそれぞれのコントロールを配置し、その動作を確かめましょう。

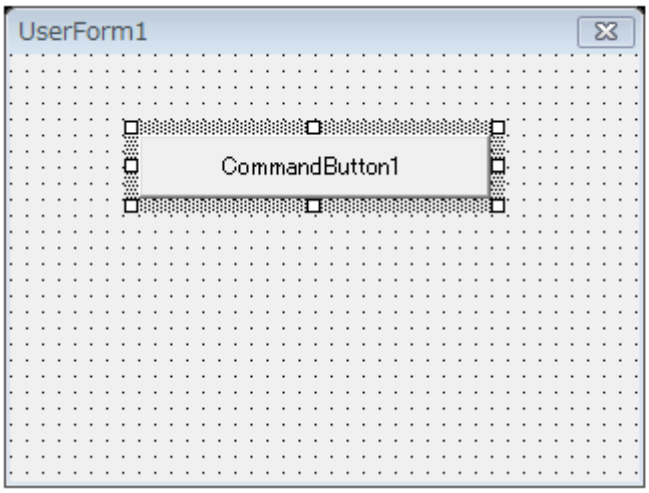
1.6.3. コマンドボタンとイベント

コマンドボタンをクリックすることで、メッセージボックスが現れたり、ユーザーフォームを終了させたりするプログラムを作ります。

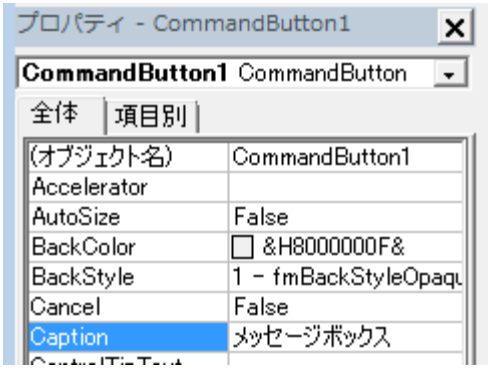
(1) VBE にユーザーフォームを挿入します。



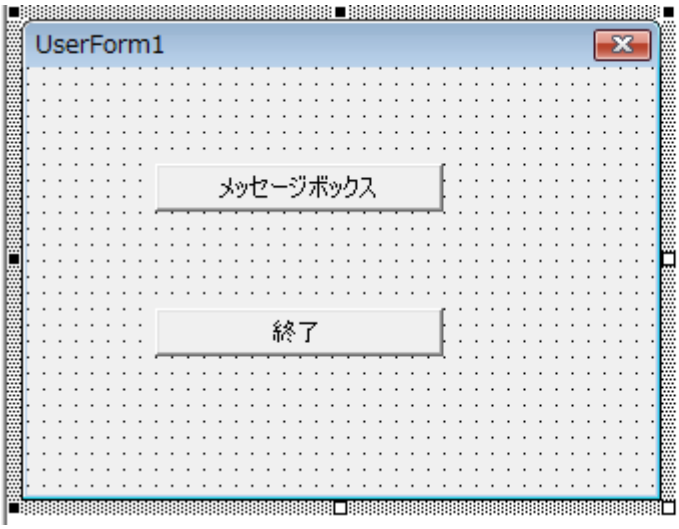
(2) ユーザーフォームにコマンドボタンを設定しします。



(3) プロパティーウィンドウの Caption で、キャプションを「メッセージボックス」とします。



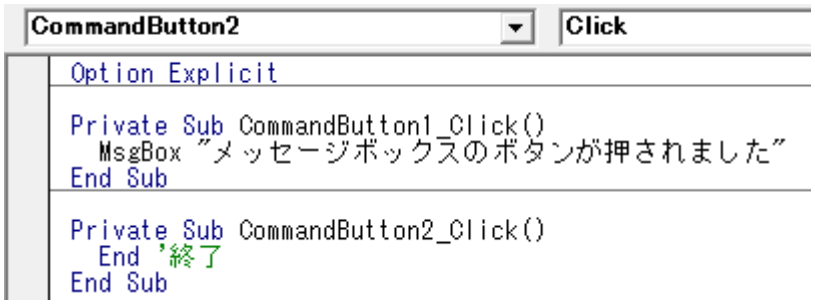
(4) 同様にして、「終了」のボタンを作ります。



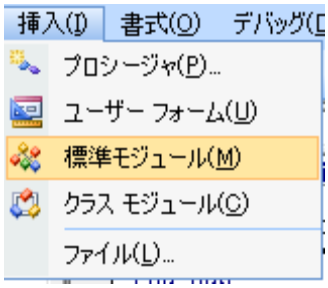
(5) 「メッセージボックス」をダブルクリックして、このボタンをクリックしたときに起動する内容のコードを書きます。ここでは、次のようにメッセージボックスを起動するようにします。

```
Option Explicit

Private Sub CommandButton1_Click()
  MsgBox "メッセージボックスのボタンが押されました"
End Sub
```

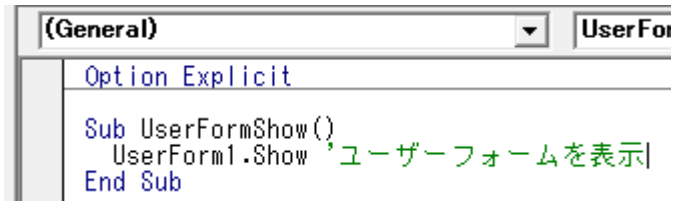
(6) 同様にして、「終了」ボタンによってユーザーフォームを終了させるコードを書き込みます。



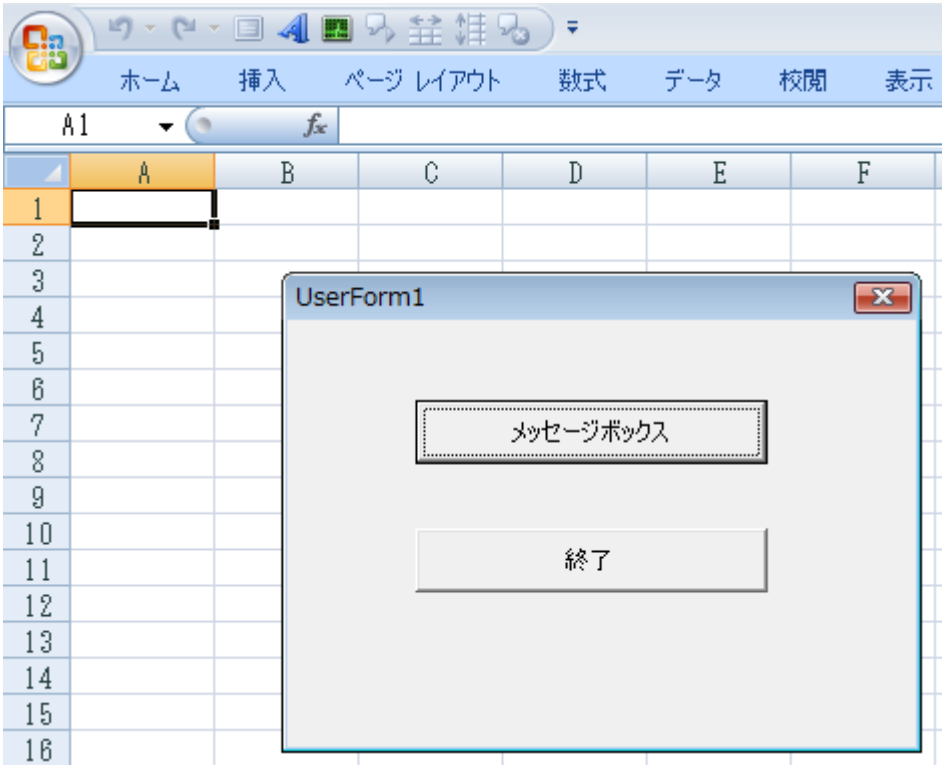
(7) VBE で標準モジュールを挿入します。



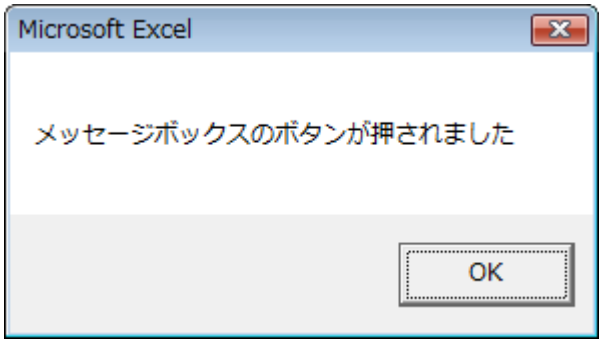
(8) 標準モジュールに、このユーザーフォームを起動するコードを書き込みます。



(9) 動作を確認するために、シートに戻って、Alt+F8, UserFormShow を選択します。



(10) 「メッセージボックス」のボタンをクリックします。



(11) 「終了」のボタンをクリックして、終了します。

【課題 1.6.3】これまでに扱ったコードの中から 1 つ選び、ユーザーフォームから実行するプログラムを作りなさい。

1.7. 正規表現

検索・置換という文字列の処理のためには正規表現が有用です。一定のパターンによる検索式や置換式を書くことにより、さまざまな検

索・置換を高速で実行します。正規表現のパターンについては、第Ⅱ部(LETRAS)を参照してください。

1.7.1. テスト

次のデータを使って、たとえばパターンに一致する語があるかどうかをテストするプログラムを作ります。ここではパターンをインプットボックスで指定します。

データ：

```
(C:V)  Text
(1:0)  *Cap.1
(1:1)  In principio creavit Deus caelum & terram.
(1:2)  Terra autem erat inanis & vacua: & tenebrae erant super faciem abyssi: & spiritus Dei ferebatur super aquas.
(1:3)  Dixitque Deus. Fiat lux. Et facta est lux.
```

コード：

```
Sub p171() '正規表現でテスト
  Dim objRE As Object, RE$, i&

  Set objRE = CreateObject("VBScript.RegExp")
  'VBScript オブジェクトを生成 (1)
  objRE.IgnoreCase = True '大小文字を区別しない (2)

  RE$ = InputBox("例：母音で始まる語", _
    "正規表現を指定してください。", "¥b[aeiou]¥w*")
  ‘インプットボックス (3)

  If RE$ = "" Then End 'キャンセル (4)

  objRE.Pattern = RE$ ‘パタン (5)

  For i = 2 To ActiveSheet.UsedRange.Rows.Count
  '2 から使用行数まで (6)
    MsgBox i & ": " & objRE.Test(Cells(i, 2))
    '行番号とテスト結果を出力 (7)
```

```
    Next

    Set objRE = Nothing 'VBScript のオブジェクトを解放 (8)
End Sub
```

解説：

(1) Set objRE = CreateObject("VBScript.RegExp")によって VBScript のオブジェクトを生成します。この手続きによって以下の正規表現による様々な検索と置換が可能になります。
(2) objRE.IgnoreCase を True に指定すると大小文字の違いを無視します。たとえば、¥b[aeiou]¥w*によって小文字だけでなく、大文字の母音で始まる語(In, Et など)も検索されます。
(3) InputBox は入力を促すインタフェースであり、入力した文字列を返す関数です。主な引数は最初の 3 つの文字型のデータです。ラベル、タイトル行、そしてテキストボックスに示す文字列のデフォルトをダブルコーテーションで囲んで指定します。
(4) キャンセルボタンを押すと、返り値がないので、プログラムを終了します。
(5) objRE.Pattern = RE$によって、パタンを設定します。
(6) For i = 2 To ActiveSheet.UsedRange.Rows.Count によって、Next までの処理を i&が 2 から使用行数になるまで繰り返します[9]。
(7) MsgBox i& & ": " & objRE.Test(Cells(i, 2))によって、行番号とテスト結果をメッセージボックスに出力します。ここで、

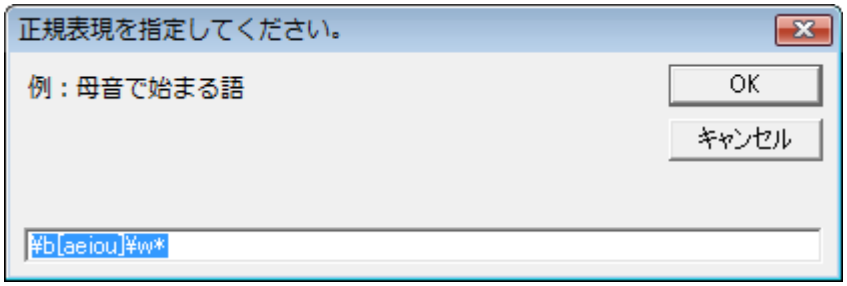
    objRE.Test(Cells(i, 2))

は各セルにあるデータについて、先に設定したパタンに一致する文字列があるかどうかをチェックします[10]。

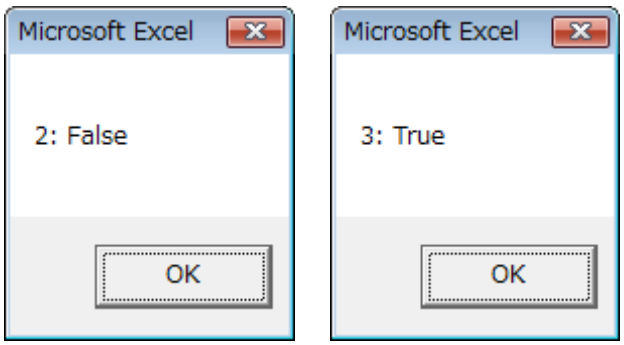
結果：

はじめに次のインプットボックスが現れます。

[9] このように変数を使わないで、毎回 Count のプロパティーを参照させると処理が遅くなります。ここではプログラムを簡単にするために直接に Count プロパティを代入しています。
[10] ここでの出力は True または False ですが、これは文字列ではなく、真(True)または偽(False)を示すブール型の変数です。



結果は最初(2 行目)は False になり、4 行目のときに True になります。



【課題 1.7.1a】インプットボックスに他のパタンを指定して、実験しなさい。

【課題 1.7.1b】次のように C 列にテストの結果を出力するコードを書きなさい。

(C:V)	Text	
(1:0)	*Cap.1	FALSE
(1:1)	In principio creavit Deus caelum & terram.	TRUE
(1:2)	Terra autem erat inanis & vacua: & tenebrae erant super faciem abyssi: & spiritus Dei ferebatur super aquas.	TRUE
(1:3)	Dixitque Deus. Fiat lux. Et facta est lux.	TRUE

1.7.2. 置換

正規表現を使うことにより、文字列の置換が一般化できて、とても便利になります。たとえば、母音 a, e, i, o, u を、それぞれ{*a*}, {*e*}, {*i*}, {*o*}, {*u*}に変換するためには、a=>{*a*}, e=>{*e*}, …のように一つずつ変換するのではなく、([aeiou])=>{*$1*}という 1 つの置換式にまとめこむことができます[11]。

[11] ここでは、[aeoiu]は a, e, i, o, u の中のいずれかの文字を示し、(…)はそれを後で参照させます。=>は左のパタンを右のパタンに置換するときに使います。ここでは、{*$1*}の$1 によって、([aeiou])を参照します。

コード：

```
Sub p172() '正規表現で置換
   Dim objRE As Object, RE$, i& '変数を宣言

   Set objRE = CreateObject("VBScript.RegExp")
   'VBScript のオブジェクトを生成
   objRE.IgnoreCase = True '大小文字を区別しない
   objRE.Global = True 'グローバル

   RE$ = InputBox("例：母音で始まる語", _
      "正規表現を指定してください。", "(¥b[aeiou]¥w*)=>{*$1*}")
   'インプットボックス (1)

   If InStr(RE$, "=>") <= 1 Then End 'キャンセル (2)

   objRE.Pattern = Left$(RE$, InStr(RE$, "=>") - 1) 'パタンを設定(3)

   For i = 2 To ActiveSheet.UsedRange.Rows.Count
   '2 から使用行数まで繰り返す
      MsgBox i & ": " & objRE.Replace(Cells(i, 2), _
         Mid$(RE$, InStr(RE$, "=>") + 2))
      '行番号と置換の結果を出力 (4)
   Next

   Set objRE = Nothing ‘VBScript のオブジェクトを解放
End Sub
```

解説：

(1) インプットボックスのテキストボックスのデフォルトは、
(¥b[aeiou]¥w*)=>{*$1*}
です。これは、母音で始まる語を{*...*}で囲む、という意味です。

(2) インプットボックスの返り値(RE$)に“=>”がなかったり、または“=>”で始まっていたりするときはキャンセルします。関数 Instr(*a$, b$*)は文字列 *a$*の中で *b$*が占める位置を返します。b$が見つからないときはゼロ(0)を返します。

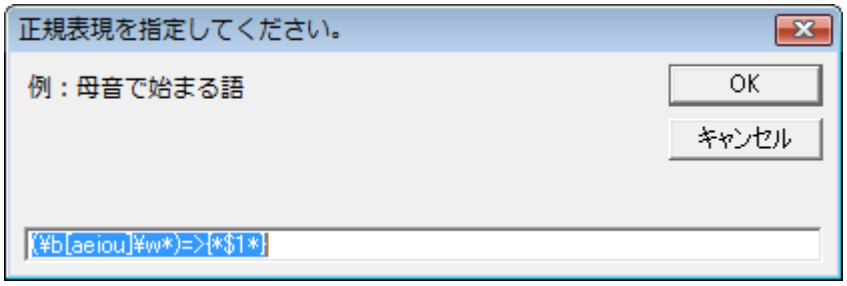
(3) パタンは RE$の中の“=>”の直前の位置までです。関数 Left$(a$, b&)は文字列 a$の中の b&の位置までの文字列を返します。ここでは b&は Instr で“=>”の位置を求め、それから 1 を引いた数となります[12]。

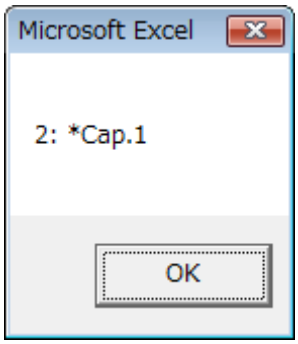
(4) 行番号と置換の結果を出力します。置換の結果は、

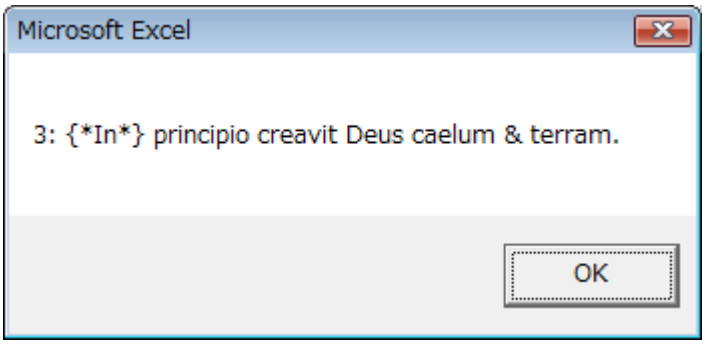
objRE.Replace(*a$, b$*)

で返します。*a$*は対象の文字列で、*b$*は置換文字列（パタン）です。ここでは、*a$*が Cells(i, 2)、*b$*が Mid$(RE$, InStr(RE$, "=>") + 2)となっています。関数 Mid$(*a$, b&*)は文字列 *a$*の中で、*b&*番目の文字以降を切り取った文字列を返します。ここで *b&*は InStr(RE$, "=>") + 2 となっていますが、これはインプットボックスで入力された文字列の中から、“=>”の位置を調べ、それに 2 を足した数を示します[13]。

結果：



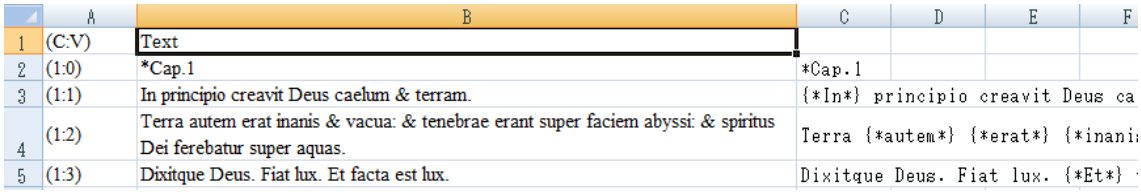




【課題 1.7.2a】インプットボックスに他のパタンを指定して、実験しなさい。

【課題 1.7.2b】次のように C 列に置換の結果を出力するコードを書きなさい。

[12] 1 を引かないと、”=”が含まれてしまいます。
[13] 2 を足すのは、“=>”が 2 文字分あるためです。2 を足さないと、返り値に”=>”が含まれてしまいます。



	A	B	C	D	E	F
1	(C:V)	Text				
2	(1:0)	*Cap.1	*Cap.1			
3	(1:1)	In principio creavit Deus caelum & terram.	{*In*} principio creavit Deus ca			
4	(1:2)	Terra autem erat inanis & vacua: & tenebrae erant super faciem abyssi: & spiritus Dei ferebatur super aquas.	Terra {*autem*} {*erat*} {*inani			
5	(1:3)	Dixitque Deus. Fiat lux. Et facta est lux.	Dixitque Deus. Fiat lux. {*Et*}			

1.7.3. 検索

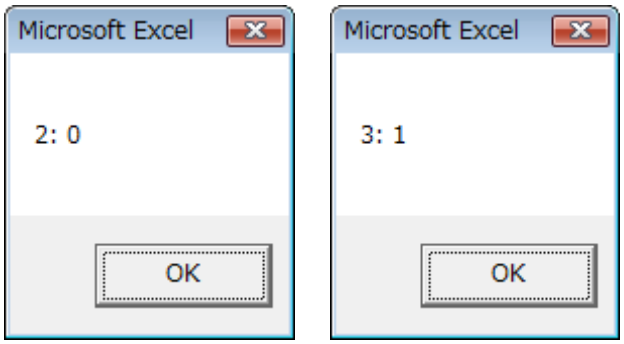
パターンに一致する文字列の数を数えます。

コード：

```
Sub p173() '正規表現でカウント
   Dim objRE As Object, Matches As Object, RE$, i&

   Set objRE = CreateObject("VBScript.RegExp")
   'VBScript のオブジェクトを生成
   objRE.IgnoreCase = True '大小文字を区別しない
   objRE.Global = True 'グローバル  (1)

   RE$ = InputBox("例：母音で始まる語", _
      "正規表現を指定してください。", "¥b[aeiou]¥w*")
    ‘インプットボックス
   If RE$ = "" Then End 'キャンセル

   objRE.Pattern = RE$ 'パタンを設定

   For i = 2 To ActiveSheet.UsedRange.Rows.Count
   '2 から使用行数まで繰り返す
      Set Matches = objRE.Execute(Cells(i, 2))
      'Matches コレクションをセット  (2)

      MsgBox i & ": " & Matches.Count
      '行番号とカウントの結果を出力  (3)
   Next

   Set objRE = Nothing ‘VBScript のオブジェクトを解放
End Sub
```

解説：

(1) objRE.Global を True にすると、最初に一致する語だけでなく、全体を検索します。
(2) Matches はパタンにマッチしたすべての文字列を集めたものです。
(3) 行番号と Matches の数を出力します。



1.7.4. 検索された文字列

正規表現によって検索された文字列（キーワード）を出力するプログラムを作ります。検索の対象となる文字列に含まれるすべてのキーワードを出力します。

コード：

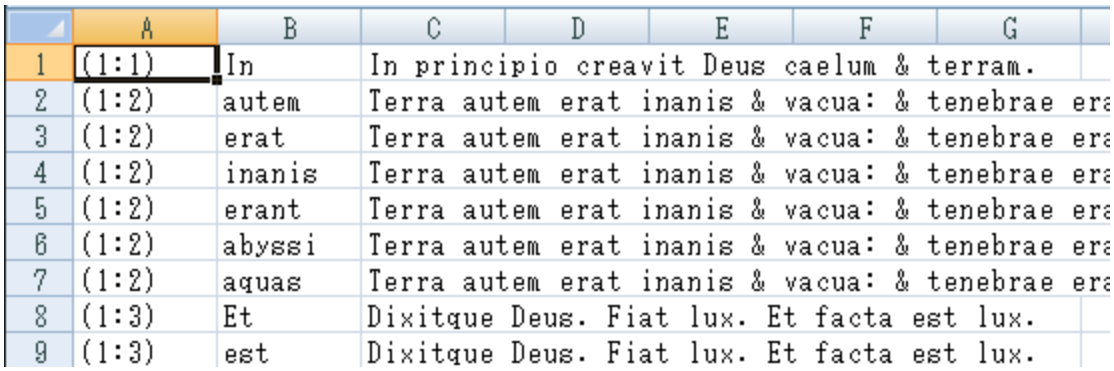
```
Sub p174a() '正規表現で検索された文字列
   Dim objRE As Object, Match As Object, Matches As Object
   '正規表現オブジェクト  (1)
   Dim Sheet$, RE$, i&, k& 'シート名、正規表現、作業用  (2)

   RE$ = InputBox("例：母音で始まる語", _
      "正規表現を指定してください。", "¥b[aeiou]¥w*")
    'インプットボックス
   If RE$ = "" Then End 'キャンセル

   Sheet$ = ActiveSheet.Name '現シート名  (3)
   Sheets.Add After:=Sheets(Sheets.Count) 'シートを追加  (4)

   Set objRE = CreateObject("VBScript.RegExp")
   'VBScript のオブジェクトを生成
   objRE.IgnoreCase = True '大小文字を区別しない
```

```
   objRE.Global = True 'グローバル
   objRE.Pattern = RE$ 'パタンを設定

   For i = 2 To Sheets(Sheet$).UsedRange.Rows.Count
   '2 から使用行数まで繰り返す
      Set Matches = objRE.Execute(Sheets(Sheet$).Cells(i, 2))
      'Matches コレクションをセット

      For Each Match In Matches
      'Matches コレクションのそれぞれの Match について (5)
         k = k + 1 '出力行 (6)
         Cells(k, 1) = Sheets(Sheet$).Cells(i, 1) 'ID (7)
         Cells(k, 2) = Match.Value 'キーワード (8)
         Cells(k, 3) = Sheets(Sheet$).Cells(i, 2) '段落 (9)
      Next
   Next

   Set objRE = Nothing ‘VBScript のオブジェクトを解放
End Sub
```

解説：

(1) 正規表現のオブジェクトとして、Match を追加します。Match については→(4)
(2) シート名 Sheet$と作業用の変数 k&を追加します。
(3) 現シート名を代入します。これは、次にシートを追加したときに、現シート名が代わってしまうので、データ用のシート名を保持しておくためです。
(4) シートをブックの最後に追加します。
(5) Matches コレクションのそれぞれの Match について、Next までの処理をします。これは、すこしわかりにくいかもしれませんが、(8)で Match を使うために必要です。
(6) 出力行をインクリメントします。
(7) データの ID を該当する位置にコピーして出力します。
(8) 検索された文字列をキーワードとして、該当する位置に出力します。
(9) データの段落を該当する位置にコピーして出力します。

結果：



	A	B	C
1	(1:1)	In	In principio creavit Deus caelum & terram.
2	(1:2)	autem	Terra autem erat inanis & vacua: & tenebrae era
3	(1:2)	erat	Terra autem erat inanis & vacua: & tenebrae era
4	(1:2)	inanis	Terra autem erat inanis & vacua: & tenebrae era
5	(1:2)	erant	Terra autem erat inanis & vacua: & tenebrae era
6	(1:2)	abyssi	Terra autem erat inanis & vacua: & tenebrae era
7	(1:2)	aquas	Terra autem erat inanis & vacua: & tenebrae era
8	(1:3)	Et	Dixitque Deus. Fiat lux. Et facta est lux.
9	(1:3)	est	Dixitque Deus. Fiat lux. Et facta est lux.

(3) 検索された文字列の位置と長さ FirstIndex, Length

検索された文字列の位置と長さが求められれば、それを使って、出力した段落に、<*...*>のような記号をつけて、際立たせることができます。また、色を変えて出力することもできます。

コード：

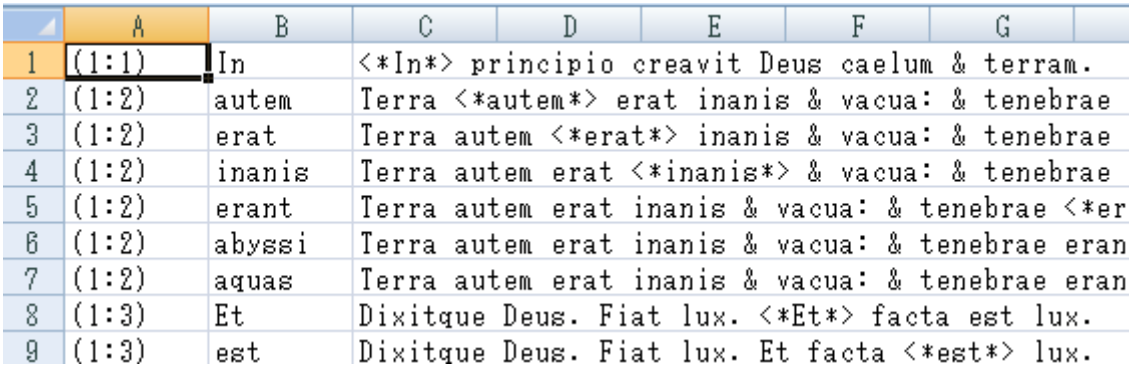
```
Sub p174b() '正規表現で検索された文字列の位置と長さ
   Dim objRE As Object, Match As Object, Matches As Object
   '正規表現オブジェクト
   Dim Sheet$, RE$, Par$, i&, k& 'シート名、正規表現、作業用 (1)

   RE$ = InputBox("例：母音で始まる語", _
      "正規表現を指定してください。", "¥b[aeiou]¥w*")
    'インプットボックス
   If RE$ = "" Then End 'キャンセル

   Sheet$ = ActiveSheet.Name '現シート名
   Sheets.Add After:=Sheets(Sheets.Count) 'シートを追加

   Set objRE = CreateObject("VBScript.RegExp")
   'VBScript のオブジェクトを生成
   objRE.IgnoreCase = True '大小文字を区別しない
   objRE.Global = True 'グローバル)
   objRE.Pattern = RE$ 'パタンを設定

   For i = 2 To Sheets(Sheet$).UsedRange.Rows.Count
   '2 から使用行数まで繰り返す
```

```
    Par$ = Sheets(Sheet$).Cells(i, 2) '対象のセル  (2)
    Set Matches = objRE.Execute(Par$)
    'Matches コレクションをセット)

    For Each Match In Matches
    'Matches コレクションのそれぞれの Match について
      k = k + 1 '出力行
      Cells(k, 1) = Sheets(Sheet$).Cells(i, 1) 'ID
      Cells(k, 2) = Match.Value 'キーワード
      Cells(k, 3) = Left$(Par$, Match.Firstindex) _
        & "<*" & Match.Value & "*>" _
        & Mid$(Par$, Match.Firstindex + Match.Length + 1) '段落(3)
    Next
  Next

  Set objRE = Nothing ‘VBScript のオブジェクトを解放
End Sub
```

解説：

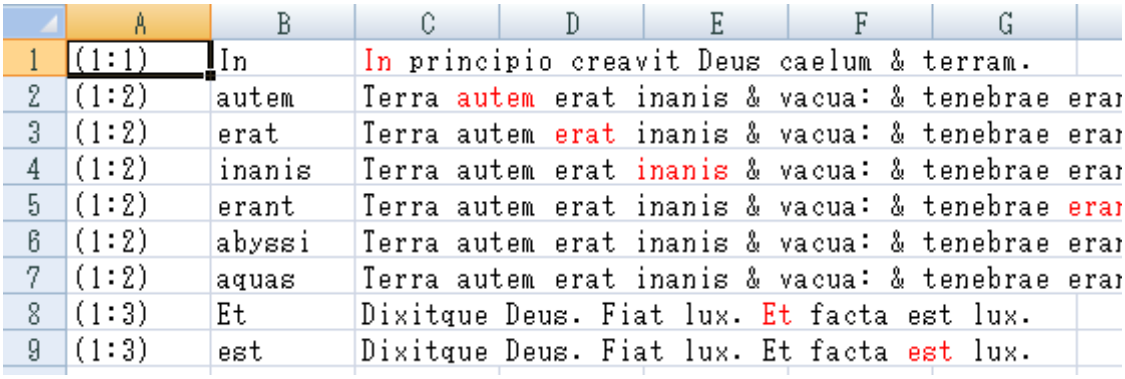
(1) 対象のセルを複数回扱うので、Par$という変数を用意しておきます。
(2) 対象のセルを Par$に代入します。
(3) 検索された文字列（キーワード）の前にある文字列、キーワード、キーワードの後にある文字列を探して、キーワードを<*...*>で囲みます。キーワードの前にある文字列は Left$(Par$, Match.Firstindex)になり、Left$(Par$, Match.Firstindex-1)ではありません。これは Match.Firstindex が検索された文字列の 1 文字目の位置が 0 になる規則があるためです。キーワードの後にある文字列は Par$の中で、Match.Firstindex + Match.Length + 1 から以後の文字列となります。+1 が必要なのは、やはり、Match.Firstindex が検索された文字列の 1 文字目の位置が 0 になる規則によるものです。

結果：



	A	B	C
1	(1:1)	In	<*In*> principio creavit Deus caelum & terram.
2	(1:2)	autem	Terra <*autem*> erat inanis & vacua: & tenebrae
3	(1:2)	erat	Terra autem <*erat*> inanis & vacua: & tenebrae
4	(1:2)	inanis	Terra autem erat <*inanis*> & vacua: & tenebrae
5	(1:2)	erant	Terra autem erat inanis & vacua: & tenebrae <*er
6	(1:2)	abyssi	Terra autem erat inanis & vacua: & tenebrae eran
7	(1:2)	aquas	Terra autem erat inanis & vacua: & tenebrae eran
8	(1:3)	Et	Dixitque Deus. Fiat lux. <*Et*> facta est lux.
9	(1:3)	est	Dixitque Deus. Fiat lux. Et facta <*est*> lux.

【課題 1.7.4】次のように C 列にキーワードを赤字で出力させるためのコードを書きなさい。



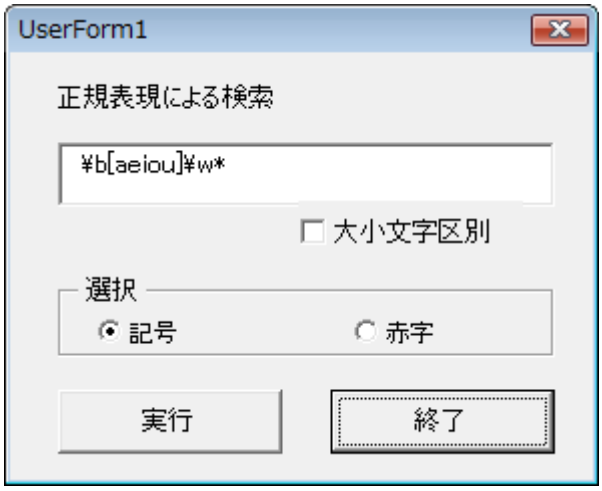
	A	B	C
1	(1:1)	In	In principio creavit Deus caelum & terram.
2	(1:2)	autem	Terra autem erat inanis & vacua: & tenebrae erar
3	(1:2)	erat	Terra autem erat inanis & vacua: & tenebrae erar
4	(1:2)	inanis	Terra autem erat inanis & vacua: & tenebrae erar
5	(1:2)	erant	Terra autem erat inanis & vacua: & tenebrae erar
6	(1:2)	abyssi	Terra autem erat inanis & vacua: & tenebrae erar
7	(1:2)	aquas	Terra autem erat inanis & vacua: & tenebrae erar
8	(1:3)	Et	Dixitque Deus. Fiat lux. Et facta est lux.
9	(1:3)	est	Dixitque Deus. Fiat lux. Et facta est lux.

次のコードを参考にして、*a, b, c, d* を決めなさい。

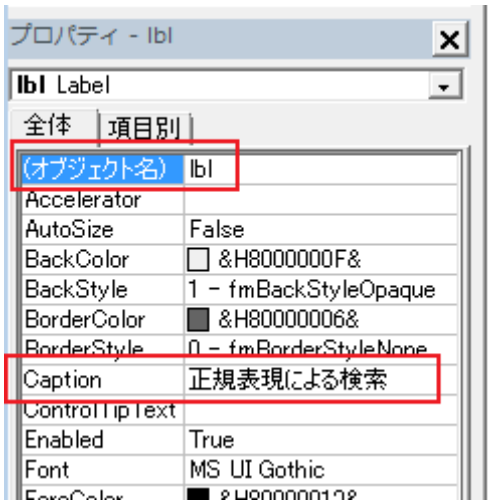
```
Cells(a, b) = Par$ '段落
Cells(a, b).Characters(Start:=c, Length:=d).Font.ColorIndex = 3
'フォントを赤に
```

1.7.5. ユーザーフォーム

次のユーザーフォームからプログラムを実行させます。はじめにユーザーフォームを作成します。



ここでは、次のコントロールを使います。プロパティウィンドーでオブジェクト名(lbl, txtRE, chkCap, frm, optSig, optRed, cmdExec, cmdEnd と Caption（「正規表現による検索」、「選択」、「記号」、「赤字」、「実行」、「終了」）を正しく設定してください。テキストボックスもデフォルト値(¥b[aeiou]¥w*)は、Text で指定します。



ラベル	lbl「正規表現による検索」
テキストボックス	txtRE
大小文字区別	chkCap
フレーム	frm「選択」
オプションボタン 1	optSig「記号」
オプションボタン 2	optRed「赤」
コマンドボタン 1	cmdExec「実行」
コマンドボタン 2	cmdEnd「終了」

コード：

```
Option Explicit

Private Sub cmdExec_Click()
   Dim objRE As Object, Match As Object, Matches As Object
   '正規表現オブジェクト
   Dim Sheet$, RE$, Par$, i&, k& 'シート名、正規表現、作業用

   RE$ = txtRE '正規表現  (2)
   If RE$ = "" Then End 'キャンセル

   Sheet$ = ActiveSheet.Name '現シート名
   Sheets.Add After:=Sheets(Sheets.Count) 'シートを追加

   Set objRE = CreateObject("VBScript.RegExp")
   'VBScript のオブジェクトを生成
   objRE.IgnoreCase = Not chkCap '大小文字を区別  (1)
   objRE.Global = True 'グローバル)
   objRE.Pattern = RE$ 'パタンを設定

   For i = 2 To Sheets(Sheet$).UsedRange.Rows.Count
   '2 から使用行数まで繰り返す
      Par$ = Sheets(Sheet$).Cells(i, 2) '対象の段落
      Set Matches = objRE.Execute(Par$)
      'Matches コレクションをセット)

      For Each Match In Matches
      'Matches コレクションのそれぞれの Match について
         k = k + 1 '出力行
         Cells(k, 1) = Sheets(Sheet$).Cells(i, 1) 'ID
         Cells(k, 2) = Match.Value 'キーワード
         Cells(k, 3) = Par$ '段落

         If optSig Then‘「記号」オプション  (3)
            Cells(k, 3) = Left$(Par$, Match.Firstindex) _
               & "<*" & Match.Value & "*>" _
```

```
                & Mid$(Par$, Match.Firstindex + Match.Length + 1) '段落
        Else‘「赤字」オプション(4)
            Cells(k, 3).Characters(Start:=Match.Firstindex + 1, _
            Length:=Match.Length).Font.ColorIndex = 3 'フォントを赤に
        End If
      Next
   Next

   Set objRE = Nothing ‘VBScript のオブジェクトを解放
End Sub

Private Sub cmdEnd_Click()
   End '終了
End Sub
```

解説：

(1) 大小文字の区別をするときは chkCap の値が True になるので、これを False にして、objRE.IgnoreCase に代入します。逆も同じです。
(2) テキストボックス(txtRE)の値を RE$に代入します。
(3)「記号」オプションボタンが True のときは、次の出力になります。
(4)「赤字」オプションボタンが True のときは、次の出力になります。

【課題 1.7.5a】3.6.2 で扱った置換のためのユーザーフォームを作り、そのコードを書きなさい。「記号」と「赤字」のオプションは不要です。

【課題 1.7.5b】置換と検索を統合するユーザーフォームを作り、そのコードを書きなさい。

1.8. 連想配列

Excel の外部にある VBScript の「ディクショナリ」を組み込むことによって**連想配列**（文字列をキーとした配列）を使うことができます。

1.8.1. 配列とディクショナリ

ふつうの配列は整数をキーとして、文字列や数を項目にします。たとえば、D$(1) = ”in”という代入文では、D$()という文字列の配列のキー＝1 に”in”という文字列が代入されます。一方、ディクショナリオブ

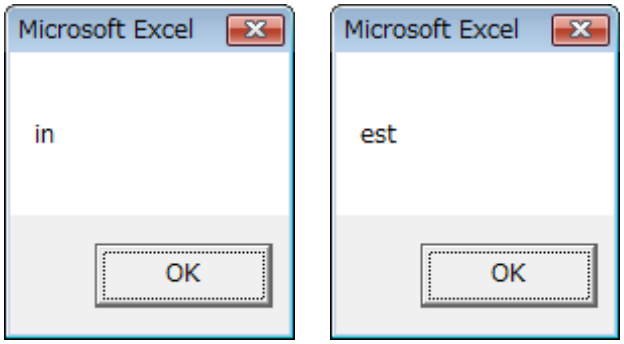
ジェクトを使うと整数ではなくて文字列を配列のキーとすることができます。

コード：

```
Sub p181() 'ディクショナリで検索
   Dim objDic As Object, D$(2) 'ディクショナリオブジェクト、配列

   D$(1) = "in" '配列に格納 (1)
   D$(2) = "est" '配列に格納 (2)

   MsgBox D$(1) '出力 (3)
   MsgBox D$(2) '出力 (4)

   Set objDic = CreateObject("Scripting.Dictionary")
   'ディクショナリオブジェクトを生成 (5)

   objDic("in") = 1 '配列に格納 (6)
   objDic("est") = 2 '配列に格納 (7)

   MsgBox objDic("in") '出力 (8)
   MsgBox objDic("est") '出力 (9)

   Set objDic = Nothing 'VBScript のオブジェクトを解放 (10)
End Sub
```

解説：

(1-2) 配列のそれぞれの位置（キーによって示す）に、文字列を代入します。
(3-4) 配列のそれぞれの位置に格納されたデータを次のように出力します。

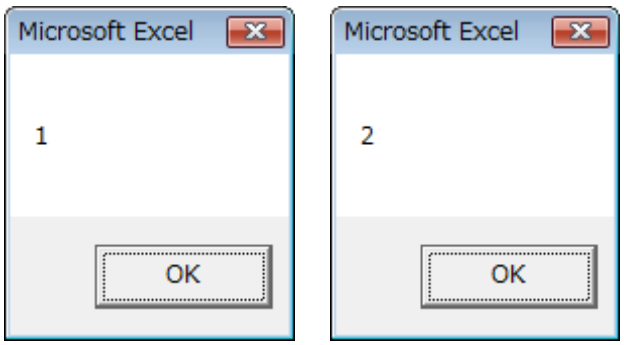


最初は、D$(1)に代入された値(in)を示しています。次は、D$(2)に代入された値(est)を示しています。

(5) ディクショナリオブジェクトを生成します。これはディクショナリを使うための手続きです。

(6-7) それぞれ、”in”, “est”という文字列をキーとして、1, 2 という数を代入します。

(8-9) 配列のそれぞれの位置に格納されたデータを次のように出力します。



(10) VBScript のオブジェクトを解放します。

【課題 1.8.1】上のコードの(1-4), (6-9)の一部を変えて、実験しなさい。

1.8.2. 単語の有無

私たちが主として扱うのは文字列による言語データですが、数字をキーにして、文字列を代入しておくと、数字をキーとした配列では、その内容（文字列）を引き出して、それが検索している対象であるかどうかをチェックする必要があります。一方、ディクショナリを使って文字列をキーとすることで、そのキーが存在することが一瞬でわかります。

コード：

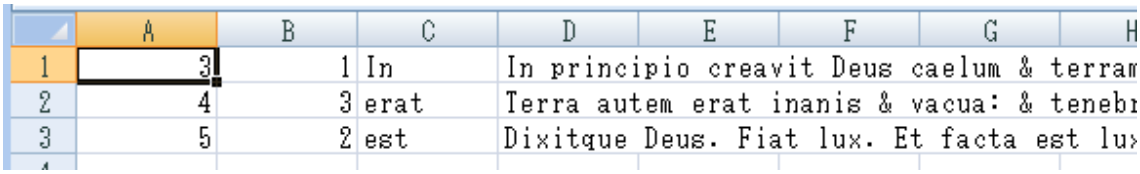
```
Sub p_372() 'ディクショナリで検索
   Dim objRE As Object, Match As Object, Matches As Object
   '正規表現オブジェクト
   Dim objDic As Object 'ディクショナリオブジェクト
   Dim Sheet$, i&, k& 'シート名、正規表現、作業用 (1)

   Set objRE = CreateObject("VBScript.RegExp")
   'VBScript のオブジェクトを生成
   Set objDic = CreateObject("Scripting.Dictionary")
   'ディクショナリオブジェクトを生成

   Sheet$ = ActiveSheet.Name '現シート名
   Sheets.Add After:=Sheets(Sheets.Count) 'シートを追加

   objDic("in") = 1 '配列に格納
   objDic("est") = 2 '配列に格納
   objDic("erat") = 3 '配列に格納

   objRE.IgnoreCase = True '大小文字を区別しない
   objRE.Global = True 'グローバル)
   objRE.Pattern = "¥w+" 'パタン（単語）を設定

   For i = 2 To Sheets(Sheet$).UsedRange.Rows.Count
   '2 から使用行数まで繰り返す
      Set Matches = objRE.Execute(Sheets(Sheet$).Cells(i, 2))
      'Matches コレクションをセット)

      For Each Match In Matches
      'Matches コレクションのそれぞれの Match について
         If objDic.Exists(LCase(Match.Value)) Then
         ‘文字列が objDic に存在すれば (1)
            k = k + 1 'k をインクリメント
            Cells(k, 1) = i '入力行
            Cells(k, 2) = objDic(LCase(Match.Value)) '語番号 (2)
            Cells(k, 3) = Match.Value '語
```

```
            Cells(k, 4) = Sheets(Sheet$).Cells(i, 2) '段落
          End If
        Next
    Next

    Set objRE = Nothing 'VBScript のオブジェクトを解放
    Set objDic = Nothing 'VBScript のオブジェクトを解放
End Sub
```

解説：

(1) マッチした文字列を小文字に変換した文字列が objDic に存在すれば、以下の出力の処理に入ります。マッチした文字列を小文字に変換に変換することで、データの中に In や Est のような大文字語が含まれていても、ディクショナリで検索されます。大文字と小文字を区別するときは、If objDic.Exists(Match.Value) Then とします。
(2) ディクショナリには語の番号が入力されていますから、ここでマッチした文字列に対応する語の番号が出力されます。

結果：



	A	B	C	D
1	3	1	In	In principio creavit Deus caelum & terram
2	4	3	erat	Terra autem erat inanis & vacua: & tenebr
3	5	2	est	Dixitque Deus. Fiat lux. Et facta est lux

【課題 1.8.2】上のコードの(1), (2)の一部を変えて、大小文字を区別して検索するプログラムを作成しなさい。

1.8.3. 単語リスト

インプットボックスに検索する単語のリストを記入します。単語はブランク（空白）などで区切って置きます。インプットボックスに入力された文字列を正規表現によって語に分割し、それぞれの語をディクショナリの配列に格納します。

コード：

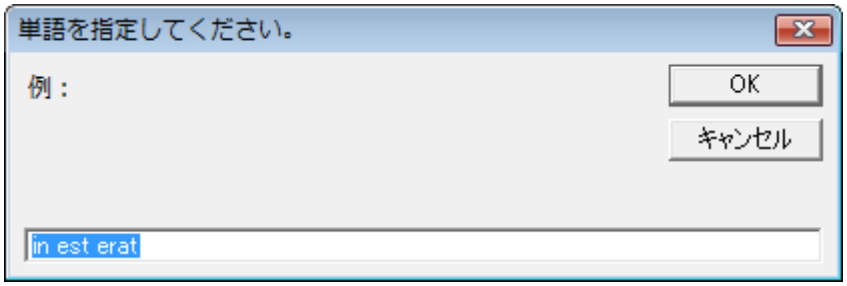
```
Sub p_183() 'ディクショナリで検索
    Dim objRE As Object, Match As Object, Matches As Object
```

```
'正規表現オブジェクト
Dim objDic As Object 'ディクショナリオブジェクト
Dim Sheet$, KW$, i&, k& 'シート名、正規表現、作業用 (1)

Set objRE = CreateObject("VBScript.RegExp")
'VBScript のオブジェクトを生成
Set objDic = CreateObject("Scripting.Dictionary")
'ディクショナリオブジェクトを生成

KW$ = InputBox("例：", "単語を指定してください。",  _
   "in est erat") 'インプットボックス (1)
If KW$ = "" Then End 'キャンセル

Sheet$ = ActiveSheet.Name '現シート名
Sheets.Add After:=Sheets(Sheets.Count) 'シートを追加

objRE.IgnoreCase = True '大小文字を区別しない
objRE.Global = True 'グローバル
objRE.Pattern = "¥w+" 'パタン（単語）を設定

Set Matches = objRE.Execute(KW$)
'Matches コレクションをセット (2)

For Each Match In Matches
'Matches コレクションのそれぞれの Match について (3)
   k = k + 1 'k をインクリメント (4)
   objDic(LCase(Match.Value)) = k '配列に格納 (5)
Next

k = 0 'k を初期化 (6)

For i = 2 To Sheets(Sheet$).UsedRange.Rows.Count
'2 から使用行数まで繰り返す
   Set Matches = objRE.Execute(Sheets(Sheet$).Cells(i, 2))
   'Matches コレクションをセット)

   For Each Match In Matches
```

```
      'Matches コレクションのそれぞれの Match について
        If objDic.Exists(LCase(Match.Value)) Then
          k = k + 1 'k をインクリメント
          Cells(k, 1) = i '入力行
          Cells(k, 2) = objDic(LCase(Match.Value)) '語番号
          Cells(k, 3) = Match.Value '語
          Cells(k, 4) = Sheets(Sheet$).Cells(i, 2) '段落
        End If
      Next
    Next

    Set objRE = Nothing 'VBScript のオブジェクトを解放
    Set objDic = Nothing 'VBScript のオブジェクトを解放
End Sub
```

解説：

(1) 検索する語のリストを入力するためのインプットボックス関数です。これは次のように出現します。返り値は KW$に代入されます。



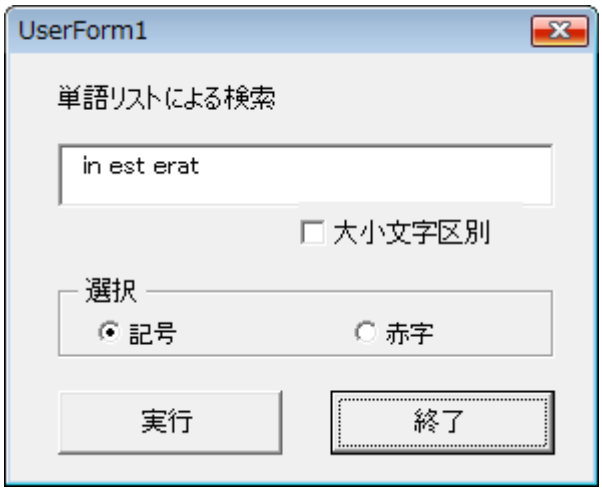
(2) 入力された文字列 KW$について、Matches コレクションをセットします。
(3) Matches コレクションのそれぞれの Match について、以下を実行します。
(4) 配列に格納する整数 k をインクリメントします。
(5) 配列のキーをマッチした文字列を小文字に変換した文字列を使い、そのキーに整数 k を格納します。
(6) k を出力行として再利用するので、0 に初期化します。

【課題 1.8.3】上のコードの一部を変更して、次のように、第 1 行をタイトル行として出力するプログラムを書きなさい。

ID	Paragrah	Key Word	
3	1	In	In principio creavit Deus caelum & terram.
4	3	erat	Terra autem erat inanis & vacua: & tenebrae erant super faciem abyssi: & spiritus Dei ferebatur super aquas.
5	2	est	Dixitque Deus. Fiat lux. Et facta est lux.

1.8.4. ユーザーフォーム

次のユーザーフォームからプログラムを実行させます。はじめにユーザーフォームを作成します。



それぞれのコントロールは、先の「正規表現による検索」を同じ物を使います[14]。

ラベル	lbl「単語リストによる検索」
テキストボックス	txtRE
大小文字区別	chkCap
フレーム	frm「選択」
オプションボタン 1	optSig「記号」
オプションボタン 2	optRed「赤」
コマンドボタン 1	cmdExec「実行」
コマンドボタン 2	cmdEnd「終了」

[14] ワークブックをコピーしてもかまいません。

コード：

```
Private Sub cmdExec_Click()
   Dim objRE As Object, Match As Object, Matches As Object
   '正規表現オブジェクト
   Dim objDic As Object 'ディクショナリオブジェクト
   Dim Sheet$, Par$, KW$, M$, i&, k&
   'シート名、段落、キーワード、マッチ文字列、作業用

   Set objRE = CreateObject("VBScript.RegExp")
   'VBScript のオブジェクトを生成
   Set objDic = CreateObject("Scripting.Dictionary")
   'ディクショナリオブジェクトを生成

   KW$ = txtRE '単語リスト
   If KW$ = "" Then End 'キャンセル

   Sheet$ = ActiveSheet.Name '現シート名
   Sheets.Add After:=Sheets(Sheets.Count) 'シートを追加

   objRE.IgnoreCase = True '大小文字を区別しない
   objRE.Global = True 'グローバル
   objRE.Pattern = "¥w+" 'パタン（単語）を設定

   Set Matches = objRE.Execute(KW$)
   'Matches コレクションをセット

   For Each Match In Matches
   'Matches コレクションのそれぞれの Match について
      M$ = Match.Value 'マッチ文字列  (1)
      If Not chkCap Then M$ = LCase(M$)
      '大小文字を区別しないならばマッチ文字列を小文字に  (2)
      objDic(M$) = “” '配列に格納  (3)
   Next

   Cells(1, 1) = "ID" 'ID (4)
   Cells(1, 2) = "Key Word" ‘(5)
```

```
  Cells(1, 3) = "Paragraph" ‘(6)

  k = 1 'k を初期化 (7)

  For i = 2 To Sheets(Sheet$).UsedRange.Rows.Count
  '2 から使用行数まで繰り返す
    Par$ = Sheets(Sheet$).Cells(i, 2) '対象の段落
    Set Matches = objRE.Execute(Par$)
    'Matches コレクションをセット

    For Each Match In Matches
    'Matches コレクションのそれぞれの Match について
      M$ = Match.Value 'マッチ文字列 (8)
      If Not chkCap Then M$ = LCase(M$)
      '大小文字を区別しないならばマッチ文字列を小文字に (9)
      If objDic.Exists(M$) Then
      '文字列が objDic に存在すれば (10)
        k = k + 1 'k をインクリメント
        Cells(k, 1) = Sheets(Sheet$).Cells(i, 1) 'ID
        Cells(k, 2) = Match.Value 'キーワード
        Cells(k, 3) = Par$ '段落

        If optSig Then
          Cells(k, 3) = Left$(Par$, Match.Firstindex) _
            & "<*" & Match.Value & "*>" _
            & Mid$(Par$, Match.Firstindex + Match.Length + 1) '段落
        Else
          Cells(k, 3).Characters(Start:=Match.Firstindex + 1, _
          Length:=Match.Length).Font.ColorIndex = 3 'フォントを赤
に
        End If
      End If
    Next
  Next

  Set objRE = Nothing 'VBScript のオブジェクトを解放
  Set objDic = Nothing 'VBScript のオブジェクトを解放
```

```
End Sub

Private Sub cmdEnd_Click()
   End '終了
End Sub
```

解説：

先の「正規表現による検索」とほとんど同じですが、次が異なります。

(1-3) 大小文字を区別するときは、マッチ文字列を単語リストに加え、大小文字を区別しないときはマッチ文字列を小文字に変換します。文字列関数 Lcase(*A$*)は、*A$*を小文字に変換して返します。ここでは、配列に""（ヌル）を代入しています。これは、後でキー（の存在）だけを参照するので、とくに項目が必要でないためです。

(4-7) 第 1 行をタイトル行とするためのコードです。

(8-10) 対象とする段落内の単語が、ディクショナリの配列内の単語と一致するかどうかを見るために、大小文字の区別をするかどうかで、マッチ文字列の扱いが代わります。

結果(1)：大小文字を区別しない、記号

	A	B	C	D	E	F	G	H
1	ID	Key Word	Paragraph					
2	(1:1)	In	<*In*> principio creavit Deus caelum & terram.					
3	(1:2)	erat	Terra autem <*erat*> inanis & vacua: & tenebrae er					
4	(1:3)	est	Dixitque Deus. Fiat lux. Et facta <*est*> lux.					
5								

結果(2)：大小文字を区別する、赤字

	A	B	C	D	E	F	G	H
1	ID	Key Word	Paragraph					
2	(1:2)	erat	Terra autem erat inanis & vacua: & tenebrae erant					
3	(1:3)	est	Dixitque Deus. Fiat lux. Et facta est lux.					
4								

1.9. プログラムの構成

1.9.1. 読みやすさ

次の 2 つのコードの動作は同じですが、コード 2 のほうが読みやす

いと言えるでしょう[15]。

コード 1

```
  WinCnt% = WinCnt% + 1 'クリック回数

  If WinCnt% Mod 3 = 1 Then ‘3 の除算の余りが 1 ならば
    C3$ = ActiveCell.Address '現在のカーソルの位置
    ActiveWindow.NewWindow '新しいウィンドウを開く
    ActiveWorkbook.Windows.Arrange ArrangeStyle:=xlVertical
    'ウィンドウを縦に並べる
    Range("A2").Select: ActiveWindow.FreezePanes = True
    'ウィンドー枠を固定
    Range(C3$).Select '以前のウィンドウのカーソル位置に移動
  ElseIf WinCnt% Mod 3 = 2 Then ‘3 の除算の余りが 2 ならば
    ActiveWorkbook.Windows.Arrange ArrangeStyle:=xlHorizontal
    'ウィンドウを横に並べる
  ElseIf WinCnt% Mod 3 = 0 Then ‘3 の除算の余りが 0 ならば
    If Windows.Count > 1 Then
      ActiveWindow.Close 'ウィンドウを閉じる
      ActiveWindow.WindowState = xlMaximized '最大化
    End If
  End If
```

コード 2

```
  Select Case WinCnt% Mod 3 '3 の除算の余りによって以下を選択
    Case 0:
      If Windows.Count > 1 Then 'ウィンドウが複数のときは…
        ActiveWindow.Close 'ウィンドウを閉じる
        ActiveWindow.WindowState = xlMaximized '最大化
      End If
```

[15] これは次のコードの[...]の部分です。

```
Private Sub cmbAnaWindow_Click()
'ウィンドウ・コマンドボタンをクリック
  [...]
End Sub
```

```
    Case 1:
        C3$ = ActiveCell.Address '現在のカーソルの位置
        ActiveWindow.NewWindow '新しいウィンドウを開く
        ActiveWorkbook.Windows.Arrange ArrangeStyle:=xlVertical
        'ウィンドウを縦に並べる
        Range(C3$).Select '以前のウィンドウのカーソル位置に移動

    Case 2:
        ActiveWorkbook.Windows.Arrange ArrangeStyle:=xlHorizontal
        'ウィンドウを横に並べる
End Select
```

このように読みやすいコードを書くことは、コードの説明や管理のために重要です。主な留意点として次が挙げられます[16]。

- コメント文を書く
- 段落を意識して改行とインデントをする
- 同じ階層に属する行の行頭位置を揃える[17]
- わかりやすい変数名を使う[18]
- なるべく短いコードを書く[19]

【課題 1.9.1】適当なコマンドボタンを作って、上の 2 つのコードの動きを確かめなさい。

1.9.2. 構造化

プログラム全体を部分（**モジュール**）に分け、それらを**構造化**すると全体の動きがわかりやすくなり、ミスも防げます。また、大きなプログラムを作るとき、最初は小さな機能だけが実行できるようなモ

[16] 他に、構造化という重要な留意点がありますが、これについては次節で扱います。
[17] VBE のタブの初期設定は 4 文字ですが、ここでは 2 文字に変更してあります。VBE の「ツール」→「オプション」→「編集」→「タブ間隔」を 2 とする。
[18] ここでは文字型($)、短整数型(%)、長整数型(&)以外は、たとえば varI のように語頭に変数型を示す文字を使っています。
[19] 長くなるときは、次で扱う構造化をします。1 つのモジュールの長さは、たとえば 40 行以内にすると読みやすいでしょう。

ジュールを作り、それに他の機能のモジュールを付け足していくように作り上げていくとよいでしょう。初めから大きなプロジェクトを展開すると、うまく動かないときに原因がどこにあるのか見つけにくくなります。

このようにモジュールを積み上げていく、という方法をとるとき、次の方法が有効です。

- プログラムの流れの制御
- サブルーチン
- ユーザー定義関数

プログラムは基本的に上から順に実行されますが、**条件**によって実行の流れが分岐したり、ループによって処理が繰り返されたりします。たとえば、次のプログラムでは、オプションボタン optPrepTitle, optPrepNum, optPrepSepInt のどれか 1 つが True であることを条件に、処理が 3 つのモジュールに分岐しています[20]。

```
Sub ZYUNBI() '■資料：準備
  If optPrepTitle Then '●A1:題名を付与
    [処理]
  End If

  If optPrepNum Then '●A2:番号を付与
    [処理]

    For i = 2 To Rw '使用中の行数まで繰り返す
      [処理]
    Next
  End If

  If optPrepSepInt Then '●B1:分離 ID から統合 ID へ
    If Cl <= Val(cmbPrepSepInt.text) Then
      [処理]
    End If
```

[20] If ... End If と同様に条件分岐をする Select Case ... End Select については先に扱いました。モジュールの境界には 1 行の空白を置くと全体がわかりやすくなります。

```
    [処理]
  End If
End Sub
```

さらに「A2:番号を付与」には For ... Next の繰り返しのモジュールがあり、「B1:分離 ID から統合 ID へ」には、If ... End If の条件分岐によるモジュールがあります。

大きなプログラムは、その一部のモジュールを**サブルーチン**にしておくとわかりやすくなります。サブルーチンは元のプログラム（最初に実行されるプログラム：メインルーチン）から呼び出されます。サブルーチンからさらに別のサブルーチンを呼び出すこともあります。サブルーチンがとくに有用なのは、それが複数のプログラムから呼び出されるときです。

```
Public Sub cmdExec_Click() '◎「実行」ボタンを押したとき
  (...)
  Call ColumnsAutofit '▼行幅・行高を自動調整
End Sub

Sub ZYUNBI() '■資料：準備
  (...)

  If optPrepAutofit Then '●E.行幅・行高を自動調整
    Call ColumnsAutofit '行幅・行高を自動調整
  End If
End Sub

Sub ColumnsAutofit() '行幅・行高を自動調整
  ActiveSheet.UsedRange.Font.Name = cmbFontName.Value 'フォント名
  ActiveSheet.UsedRange.Font.Size = cmbFontSize.Value 'フォントサイズ
  ActiveSheet.UsedRange.Columns.ColumnWidth = Val(txtColMax)
  '列幅を最大値に
  ActiveSheet.UsedRange.Columns.WrapText = True '折り返し
  ActiveSheet.UsedRange.Columns.AutoFit '列幅を自動調整
  ActiveSheet.UsedRange.Rows.AutoFit '行高を自動調整
  Range("A2").Select: ActiveWindow.FreezePanes = True
```

```
  'ウィンドー枠を固定
End Sub
```

上のコードでは、ColumnsAutofit というサブルーチンが cmdExec_Click というイベントによって起動するプログラムから呼び出され、さらに、ZYUNBI というサブルーチンからも呼び出されています。

以前に、あらかじめ用意されている関数について扱いました。あらかじめ用意されている関数ではなく、プログラムの中で、ユーザー（プログラマー）があれば便利だと思う関数を作成することができます。たとえば、次の ReplaceRE は、対象、検索文字、置換文字、大小文字区別、という 4 つの引数によって正規表現置換を実行する関数です。対象の文字列が、これらの条件に従って置換された値（文字列）が返されます。

```
Sub YOMIKOMI() '■資料：読み込み
  (...)
  varK = ReplaceRE(varK, " ", "_123", False) 'データ
  (...)

Function ReplaceRE(ByVal T$, S1$, S2$, bolIC As Boolean) As String
'関数：正規表現による置換（対象、検索文字、置換文字、大小文字区別）
  objWK.IgnoreCase = Not bolIC '大小文字区別を判定
  objWK.Pattern = S1$ '検索パターン
  ReplaceRE = objWK.Replace(T$, S2$) '置換・代入
End Function
```

1.9.3. 高速化

プログラムが正しく実行されれば一応完了したことになりますが、見直すと無駄な処理が見つかることがあります。

変数を使うことによって不必要な計算を繰り返すことを防ぎ、実行が高速化することができます。たとえば、次の 2 つのコードを比べてみましょう。

コード 1

```
Sub p_383a()
```

```
  For i = 1 To ActiveSheet.UsedRange.Rows.Count
  '現シートの行数まで繰り返す
    Cells(i, 1) = i '連続番号を書き込む
  Next
End Sub
```

コード 2

```
Sub p_383b()
  Dim Rw& '現シートの行数

  Rw& = ActiveSheet.UsedRange.Rows.Count '現シートの行数

  For i = 1 To Rw& まで繰り返す
    Cells(i, 1) = i '連続番号を書き込む
  Next
End Sub
```

コード 1 では、For ... Next のループを繰り返すごとに、現シートの行数をカウントしています。一方、コード 2 では Rw&という変数に一度だけ現シートの行数を計算して代入し、For ... Next のループではこれを繰り返して使っています。計算するという動作が 1 回きりなので、コード 2 のほうが高速です。少ないデータでは差があまり感じられませんが、大量のデータとなったときに差が顕著になります。

一方、上のように変数を使うと行数が増え、また変数の内容にも注意しておかなければならなくなります。このように高速化と読みやすさにはそれぞれ一長一短がある場合があります。どちらを優先させるかはケースバイケースです。

【課題 1.9.3a】データの量を多くして、上の 2 つのコードの実行時間を比較しなさい。

上のコードでは Cells という個々のセルを単位として処理していますが、**レンジ**(Range)という「面」（2 次元）の単位を使うことにより処理を高速化することができます[21]。とくに次のように、一括して処理するときに有効です。

```
Sub p_383c()
```

[21] 上のコードで使っている UsedRange も 1 つの「レンジ」です。

```
    Dim Rw& '現シートの行数

    Rw = ActiveSheet.UsedRange.Rows.Count '現シートの行数
    Range(Cells(1, 1), Cells(Rw, 1)) = ActiveSheet.Name '現シート名
End Sub
```

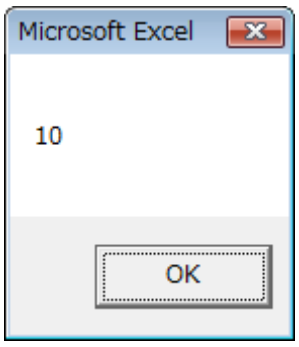
Range は、上のように始点と終点のセルを指定することにより、それらを含む長方形の範囲になります。このように Range を使うと、コードも比較的単純になります。

ワークシート関数はエクセルシートで使われる関数です。たとえば、次のシートの A1:B5 の範囲の最大値を求めるには、コード p_383d を使います。

1	8
5	9
3	10
6	5
2	2

```
Sub p_383d() '最大値
    MsgBox WorksheetFunction.Max(Range(Cells(1, 1), Cells(5, 2)))
End Sub
```

結果：



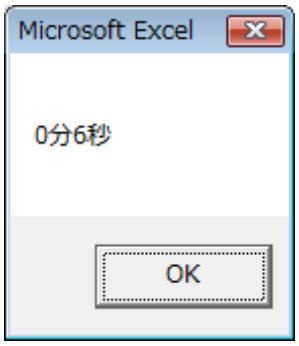
ワークシート関数を使わないと、コードが複雑になり、実行時間も増加することになります。

最後に、処理の時間を計算するコードを示します。これを使うことによってさまざまな処理の経過時間を比較することができます。

```
Sub k()
    varStart = Time '開始時
```

```
    For i = 1 To 10000
        For j = 1 To 10
            Cells(i, j) = i '処理
        Next
    Next

    MsgBox Minute(Time - varStart) & "分" _
        & Second(Time - varStart) & "秒" '経過時間
End Sub
```

結果：



2. 課題コードの解答

【課題 1.1b】

```
Sub e11b()
   Cells(2, 2) = "ABC" 'セル B2
   Cells(3, 2) = "DEF" 'セル B3
End Sub
```

【課題 1.3.1a】

```
Sub e131a()
   Dim intA As Integer '整数型
   Dim intB As Integer '整数型

   intA = 1 'intA に 1 を代入
   intB = 2 'intB に 2 を代入

   Cells(1, 1) = "足し算"
   Cells(1, 2) = "引き算"
   Cells(1, 3) = "掛け算"
   Cells(1, 4) = "割り算"
   Cells(2, 1) = intA + intB
   Cells(2, 2) = intA - intB
   Cells(2, 3) = intA * intB
   Cells(2, 4) = intA / intB
End Sub
```

【課題 1.3.1b】

```
Sub e131b()
   Dim A$, B$, C$

   A$ = "太郎は" 'A$に代入
   B$ = "学生です。" 'B$に代入
   C$ = "花子は" 'C$に代入
```

```
   Cells(1, 1) = A$ & B$ '連結させて出力
   Cells(2, 1) = C$ & B$ '連結させて出力
End Sub
```

【課題 1.3.3】

```
Sub e133() '逆順
   Dim A$, S$, i& '宣言

   A$ = "TERRAM" 'A$に文字列を代入

   For i = 1 To Len(A$) 'A$の文字数まで繰り返す
      Cells(i, 1) = i '番号
      Cells(i, 2) = Mid(A$, i, 1) '文字
      S$ = Mid(A$, i, 1) & S$ 'i 番目の 1 文字を前に連結
      Cells(i, 3) = S$ '文字列
   Next
End Sub
```

【課題 1.5.1a】

```
Sub e151a() '使用している列数
   MsgBox "列数は" & ActiveSheet.UsedRange.Columns.Count
End Sub
```

【課題 1.5.2】

```
Sub e152() '使用している行数
 Dim SheetName$, RowCnt&, i& 'シート名、使用行数、作業用変数

   SheetName$ = ActiveSheet.Name 'シート名
   ActiveSheet.Copy After:=Sheets(Sheets.Count) '新シートにコピー
   RowCnt& = ActiveSheet.UsedRange.Rows.Count '使用行数
   Columns(1).Insert '1 列目の前に 1 列挿入
   Cells(1, 1) = "Sheet" 'タイトル

   For i = 2 To RowCnt& '2 から使用行数まで
      Cells(i, 1) = SheetName$ 'シート名を出力
   Next
```

```
   Columns(2).Insert '2 列目の前に 1 列挿入
   Cells(1, 2) = "No." 'タイトル

   For i = 2 To RowCnt& '2 から使用行数まで
      Cells(i, 2) = i - 1 '2 列目に連続番号を出力
   Next
End Sub
```

【課題 1.5.3】

```
Sub e153()
   Dim RowCnt& '使用行数

   RowCnt& = ActiveSheet.UsedRange.Rows.Count '使用行数

   Range(Cells(1, 1), Cells(RowCnt&, 1)).Font.Bold = True '範囲を太字に
End Sub
```

【課題 1.7.1b】

```
Sub e171() '正規表現でテスト
   Dim objRE As Object, RE$, i&

   Set objRE = CreateObject("VBScript.RegExp") 'VBScript のオブジェクトを生成

   RE$ = InputBox("例：母音で始まる語", "正規表現を指定してください。", "¥b[aeiou]¥w*")
   If RE$ = "" Then End 'キャンセル

   objRE.Pattern = RE$

   For i = 2 To ActiveSheet.UsedRange.Rows.Count '2 から使用行数まで
      Cells(i, 3) = objRE.Test(Cells(i, 2)) '行番号とテスト結果を出力
   Next
End Sub
```

【課題 1.7.2b】

```
Sub e172() '正規表現で置換
   Dim objRE As Object, RE$, i& '変数を宣言

   Set objRE = CreateObject("VBScript.RegExp")
   'VBScript のオブジェクトを生成
   objRE.IgnoreCase = True '大小文字を区別しない
   objRE.Global = True 'グローバル

   RE$ = InputBox("例：母音で始まる語", _
      "正規表現を指定してください。", "(¥b[aeiou]¥w*)=>{*$1*}")
   'インプットボックス

   If InStr(RE$, "=>") <= 1 Then End 'キャンセル

   objRE.Pattern = Left$(RE$, InStr(RE$, "=>") - 1) 'パタンを設定

   For i = 2 To ActiveSheet.UsedRange.Rows.Count
   '2 から使用行数まで繰り返す
      Cells(i, 3) = objRE.Replace(Cells(i, 2), _
         Mid$(RE$, InStr(RE$, "=>") + 2)) '行番号と置換の結果を出力
   Next
End Sub
```

【課題 1.7.4】

```
Sub e174() '正規表現で検索された文字列の位置と長さ
   Dim objRE As Object, Match As Object, Matches As Object
   '正規表現オブジェクト
   Dim Sheet$, RE$, Par$, i&, k& 'シート名、正規表現、作業用 (1)

   RE$ = InputBox("例：母音で始まる語", _
      "正規表現を指定してください。", "¥b[aeiou]¥w*")
    'インプットボックス
   If RE$ = "" Then End 'キャンセル

   Sheet$ = ActiveSheet.Name '現シート名
```

```
   Sheets.Add After:=Sheets(Sheets.Count) 'シートを追加

   Set objRE = CreateObject("VBScript.RegExp")
   'VBScript のオブジェクトを生成
   objRE.IgnoreCase = True '大小文字を区別しない
   objRE.Global = True 'グローバル)
   objRE.Pattern = RE$ 'パタンを設定

   For i = 2 To Sheets(Sheet$).UsedRange.Rows.Count
   '2 から使用行数まで繰り返す
      Par$ = Sheets(Sheet$).Cells(i, 2) '対象のセル (2)
      Set Matches = objRE.Execute(Par$)
      'Matches コレクションをセット)

      For Each Match In Matches
      'Matches コレクションのそれぞれの Match について
         k = k + 1 '出力行
         Cells(k, 1) = Sheets(Sheet$).Cells(i, 1) 'ID
         Cells(k, 2) = Match.Value 'キーワード

         Cells(k, 3) = Par$ '段落
         Cells(k, 3).Characters(Start:=Match.Firstindex + 1,
Length:=Match.Length).Font.ColorIndex = 3
         'フォントを赤に
      Next
   Next

   Set objRE = Nothing 'VBScript のオブジェクトを解放
End Sub
```

【課題 1.8.2】

```
(…)
If objDic.Exists(Match.Value) Then
         ‘文字列が objDic に存在すれば (1)
            k = k + 1 'k をインクリメント
            Cells(k, 1) = i '入力行
            Cells(k, 2) = objDic(Match.Value) '語番号 (2)
```

```
(…)
```

【課題 1.8.3】

```
[A1] = "ID": [B1] = "Paragrah": [C1] = "Key Word"

k = 1 'k を初期化 (6)
```