Excel版ラテン語動詞活用表の作成

布施　温

Making of Latin Verb Conjugation Table by EXCEL

Yutaka FUSE

Abstract

After finishing "Spanish Verb Conjugation Table by EXCEL" I have intended to make "Latin Verb Conjugation Table" using also Excel as an instrument of developing a large scale of "Macro program".

The reasons why I have chosen EXCEL are following ;

1) Excel is installed in all Windows machines.

2) The width of columns can be fixed automatically.

3) We can represent long and short vowels in printing.

4) The VBA of Excel is almost the same one with Visual Basic.

5) The VBA of Excel has string treating functions enough and satisfactorily.

The usage of it is very simple ; putting the first person, singular, present of indicative mood all in short vowel in G2 cell you can get instantaneously fully conjugated 146 forms of the verb with long and short vowels.

This program is intended to help the beginner or intermediate student of Latin language to get acquaintance of verb forms which is important and indispensable to Latin language.

The program is based on the book titled "The Big Gold Book of Latin Verbs, 555 fully Conjugated Verbs" of McGraw Hill. This software can treat about 2,100 verbs that means four times as large as the

book above mentioned.

I am afraid, however, there may be many mistakes of data. I hope therefore, with your collaboration, to make this program a perfect one.

0．初めに

N88BASICでラテン語の活用表を作り、1990年5月のロマンス語学会で発表したことがある。その時は、長母音と短母音の区別をつけるために、それらをデータとしてプログラム内に納めたのだが、プログラム領域に65536バイト(64K)、データ領域も同じく65536(64K)バイトという大変にきつい制限があったために、約700語しか表示できなかった。短母音はそのまま表示すればよいので何の問題もないが、長母音はプログラム内では例えば[a]の長母音はカタカナの[ア]で示して、表示の際はその16進データが画像として表示できるようにした。

その後パソコンはWindowsの時代となり、NECも独自路線を捨ててMicrosoft陣営に移った。それに応じてWindows用に移植を試みたいと思ったが、特殊文字のVisual Basicでの扱い方が分からなかったし、また汎用性を確保するべく16進での表示は避けたかったので、永い間実行に移せないでいた。

しかし、Windowsも2000, XPとバージョンが新しくなるようになってからはスペイン語などのいわゆる特殊文字もシステムとしてサポートするようになった。Word等のワープロソフトでは以前から使えるようになっていたけれども、マクロ内ではそのままでは扱えないので、なかなか計画が進まなかった。最近になってようやく表示方法が分かったので、まずはスペイン語動詞活用表を作った。ただし、Visual Basicなどの言語ではなくExcelを使った。その理由は次の通りである。

1）Excelは、Windowsにはほぼ間違いなく付いている。Visual Basicなどの言語ソフトをインストールする必要がない。
2）長い動詞形でもセル枠を自動的に移動せるので、表示位置を指定する必要がない。
3）印刷もExcelに任せることができる。

4）Excel の VBA は Visual Basic そのものだから今までの経験が生かせる。
5）Excel には文字列処理関数が十分に備わっている。

1．長母音の表示方法

Excel のセル内には直接入力はできないが、次のようにすれば表示が可能になる。

1）Word を起動して、Alt キーを押しながら、テンキーから0257と入力すれば長母音の a が出る。必ずテンキーから入力すること。アルファベット上の数字キーでは不可。

長母音は次のように入力する。

ā− Alt+0257 → ChrW(257)
ē− Alt+0275 → ChrW(275)
ī− Alt+0299 → ChrW(299)
ō− Alt+0333 → ChrW(333)
ū− Alt+0363 → ChrW(363)

2) Word で表示された長母音を Excel の適当なセルにコピーしておく。

なお、Excel の VBA(Visual Basic for Applications)では、上の表の矢印のように表現するか、コピーした表のセル番地を指定する。ラテン語では長母音、短母音の区別は大切なので、データ入力時には例えば a の長母音は、aa と入力して後で長母音に変換すると仕事が早い。

3) このようにすれば、発音記号を含むかなりの特殊文字が扱える。ちなみにそれらの一覧表を Excel で作成してみよう。

VBA エディタをクリック、挿入、標準モジュール内に次のコードを入力

```
1 : Sub tokushu_moji()
2 :         Dim i as Integer
3 :             For i=161 To 651
4 :           Cells(i−160, 1) =ChrW(i)
5 :       Next i
6 : End Sub
```

プログラムの説明を簡単にする。左端の番号は説明のためにつけた。

1行：プロシージャの名称

2行：Dim i as Integer　　'変数iの型宣言

3行：For i=161 To 651　　'iを161から651まで変化させる。

4行：Cells(i−160, 1)＝ChrW(i)

Cellsプロパティでセルを選択するのだが、使い方は、Cells(行, 列)の順序になる。

最初には、Cells(1, 1)＝“A1”にChrW(161)＝！の逆さま(！)が入る。

5行：Next i　　'次のiに移る

次にセル、Cells (2, 1) ＝“A2”にChrW(162)＝¢が入る。以下同様。

6行：プロシージャに名称を付けると自動的に入る。

なおモジュールとは宣言やプロシージャの塊のことである。プログラム全体と言っても良い。プロシージャは個別の処理を担当するプログラムである。

これでA列に特殊文字が入るので、B1セルに、ChrW161、B2セルにChrW162と入力し、B1セルとB2セルを選択しマウスをそのままB2セルの右下隅に移動しマウスポインタが小さな十字架になったのを確認してA1の最終セルまでドラッグして離す。そうすれば番号が自動的に連番になる。所々欠番があるのでそれらは削除する。ChrW(651)位までで十分であろう。

2．動詞の選択

これが一番の問題であろう。ラテン語は専門でないのでどの動詞を選ぶか迷うところである。最初は、日本で唯一のラテン語辞書、研究社版「羅和辞典」からすべての動詞を抽出した。約5,000語になった。全部を表示するようにするのか、それとも選択するのか。目標は初級、中級レベルを対象とした活用表を作ることである。しかしそれではこの中からどれを選べば良いのか、この段階で大部時間を取られた。それに文法書や辞書に載っている活用表はほんの一部分の動詞しか載っていない。そこで詳しいラテン語動詞活用表の本はないかと探してみた。以前にスペイン語動詞活用表を作成した際に、“501 Spanish Verbs” を使ったことがあるのでそのラテ

ン語版はないかと調べてみると、同じ出版社、Barron's Educational Series, Inc. から "501 Latin Verbs"(以下「501語」と略す)が出ていることが分かった。さらに、McGraw-Hill から"The Big Gold Book of Latin Verbs. 555 fully Conjugated Verbs"(以下「555語」と略す)にはタイトルの555動詞を含めて、2,500語のリストが載っていた。しかしそのうちの約400語は動詞変化のパターンが明示されておらず、慎重に調べないといけないので、今回は約2,100語を扱うことにした。この400語と以前に抽出した動詞を約500語加えて、3,000語を表示することを目標とする。この数は必要かつ十分なものであろう。

3．全体の構造

Excelのワークシート3枚を使用する。

1枚目のシートにはには、ラテン語動詞、現在1人称単数形を、G2セルに入力するように指定する。

2枚目のシートには、動詞活用語尾のデータが入る。まずは5種類は最低必要。第一変化動詞、第二変化動詞、第三変化動詞、第三変化動詞と第四変化動詞の混合、第四変化動詞である。

さらに不規則動詞の、sum、eo, fero と予備にさらに2列の語尾形を入れた。

3枚目のシートには、約2100語の動詞データが入っている。5列を使用する。詳細は4.3で述べる。

4．各シートの説明

4.1　1枚目のシート

このシートはどのような配置になっているかを示す。この配置は、前述の「555語」に倣っている。原則として、このプログラムは「555語」を基にしており、また配置は「501語」より見やすい。

B列とC列には能動相が入る。G列とI列には受動相が入る。H列とI列は受動相完了形で、語形が長くなることを配慮して開けてある、と言うよりは完了形の枠として、G列とH列を、I列とJ列を結合してある。

なお、これらの枠は0の2)で述べたように、自動的に変えるようにできる。これは非常にありがたいことである。さもないと動詞毎に表示枠幅を指定せねばならないが、これは大変な作業を伴うし、またその作業は本質的でない。

表示の実際は、このソフトを試して頂くかあるいは、添付の「活用枠.xls」で確かめて欲しい。また、それを印刷するときには、「ページ設定」「シート」の順で開いて「行列番号」にチェックを入れると行列番号も印刷される。大変便利な機能である。

4.2　2枚目のシート

これは先ほど述べた。必要なら「活用語尾.xls」を見て頂きたい。

4.3　3枚目のシート

2100語のうちからいくつかをピックアップしてご覧に入れる。

do	70				
eo	63				
ago	31	ag	ēg	act	
amo	11	am			
sto	14	st	stet	stat	
sum	61				
dico	90	dīc	dix	dict	
fero	64	fer	tul	lāt	
meto	34	met	messu	mess	
nolo	71				
pono	31	pōn	posu	posit	
peto	31	pet	petīv(peti)		petīt
repo	33	rēp	reps		
volo	90				
conor	12	cōnor			

etc.

第1列は、長短の区別無しのラテン語動詞現在1人称単数形を入れる。もっともlicet, ningitなどのように3人称単数を入れることもある。母音

の長短を区別しない理由は、実際の文には表示されていないし、また初学者にはそこまでの要求は出来ないからである。

第２列には、動詞変化のパターンを示す数字が入る。ここで第何変化動詞か、その変化パターンは何かを所得する。

第３列には現在形の語幹が、第４列には完了形語幹が、第５列には分詞語幹が入る。ラテン語はある時制内では語幹変化がない。これはプログラミング作業を大いに軽減することにつながる。なお peto の完了形は()内に別の語形が示されている。２つあるからである。

dō, eō, sum, nōlō, vōlō などは現在1人称単数形しかデータとしては入れてない。これらは不規則だから個別に処理するからである。

amō は現在語幹しか表示されていない。現在語幹が分かれば他の語幹は決まってしまうからである。第1類所相動詞の cōnor も同様である。

中には repō のように分詞語幹がないものもある。このような動詞の場合は分詞語幹の部分の活用がないからそれなりの処理をする必要がある。

このようにしてすべての動詞のデータを入力するのだが、大切なのは、文字数の短い動詞をシート３の先に並べることだ。そうでないと、例えば、sum が intersum の後に置かれたりすると sum と入力しても intersum の活用が表示されてしまうからである。

それを避けるのは極めて簡単だ。シート３の第６列(A6)に、次のような式を書く。

=len(A1)

A1セルに、eo が入っていれば２という値が戻ってくる。fero ならば４である。

A6セルをコピーして、最後の行までコピーする。これでA列すべての文字数が出るので後はデータの並べ替えをするだけだ。第１優先順位はF列、第２順位どこでも良いがA列、後はB列にでもすれば良いだろう。

このプログラムは「555語」を基にしているとは述べたが、必ずしもすべてそうしたわけではない。例えば、cōnficiō は「555語」では、conficiō となっていたが、参照した辞書３冊では長母音となっていたからそれに従った。

5．プログラミングの実際

5.1 宣言部

```
Option Explicit
Const n As Integer=2500
Public i As Integer
Public j As Integer
Public tipo As Integer
Public pls As String
Public gokan As String
Public Inf As String
Public Present As String
Public Perfect As String
Public Participle As String
Public WS1 As Worksheet
Public WS2 As Worksheet
Public WS3 As Worksheet

Option Base1
Dim Conj (146) As String
```

これからのプログラミングで使う変数の型宣言を行う。
これをしないと、プロシージャ毎に変数の型宣言をせねばならない。

Option Explicit

これを宣言することによって、このプログラムで使用するすべての変数を明示することになる。逆に宣言がないときはエラーがでる。変数の誤入力があればチェックがかかる。

Const n As Integer=2500

Constステートメントで、これから扱う動詞の数を宣言する。
最終的には3000語を目指すが今回は2100語、余裕を見て2500とする。この数字を変更するだけで使用するメモリが大幅に変化するに注意しよう。
例えばN=3000の場合は751Kバイト、2500の場合は612Kバイトであ

る。従ってこの数字はプログラム確定後に変更するのが良い。

Public ステートメントでこのブック内の全モジュール内のすべてのプロシージャで参照できるようになる。つまりプログラム全体でこの変数が通用することになる。

Integer は整数型の変数である。

String は文字列型の変数である。

Dim Conj (146) As String で Conj は文字列型の変数で箱を146個用意させる。

Option Base 1で配列の番号が1から使えるようになる。そうでないと配列は0から始まるのでプログラム中に混乱が起きるかも知れない、それを避けるための工夫である。

5.2 セル入力時にマクロを実行する

```
Sub Worksheet_Change (ByVal Target As Range)
    If Target(1). Address= "$G$1" And Target(1).Value
                                <> Empty Then
      Call find
    End If
End Sub
```

このマクロ(プログラム)を実行するには、現在形1人称単数形を入力せねばならない。つまりデータ入力を促さねばならない。そのためには次のようなことが考えられる。

1) InputBox 関数を使う

2) セル入力時に即実行する。

1) InputBox 関数を使えば確かにデータ入力はできる。しかし毎回このボックスが開くのは大変に煩わしい。それに今回は長母音記号の入力は行わないが、他のロマンス系言語のようにアクセント記号が必要な場合は、それらの入力がこのボックス内では出来ないので、結局 InputBox 関数は使えない。またラテン語の長母音記号はかなり特殊なので、入力モードもそれなりに設定せねばならない。となると2)しか残らない。

このプロシージャの意味は次の通りである。

"G1セル(絶対参照のG1セル)に何かのデータが空でない限りは(何かの

データが入力されれば)、findプロシージャを呼び出せ”ということである。

なお、この部分のプロシージャは「イベントプロシージャ」なので、必ずシート1内に作成されねばならないことに注意しよう。マクロは通常、標準モジュールに書かれる。

最初はこの部分の処理が分からなかったのだが、Excelのメーリング・グループのMOUGに質問したところ直ちに解答が帰ってきた。大変にありがたいことである。

5.3 変化パターンにより分岐する

```
Sub find()

    Set WS1=Worksheets("sheet1")
    Set WS2=Worksheets("sheet2")
    Set WS3=Worksheets("sheet3")

    j=1
    Do While j <= n
        If WS3.Cells (j,1) =WS1.Range("G1").Value Then
            Exit Do
        End If
        j = j+1
    Loop
    If j <= n Then
        tipo = WS3.Cells (j,2)
    Else
        tipo=0
    End If

    If tipo=11 Then
        Call amare              'First regular ( a 1−11)
    ElseIf tipo=12 Then
        Call conor              'First deponent ( a 2−12)
```

```
ElseIf tipo=13 Then
    Call labo                ' First no-passive (a 3-13)
ElseIf tipo=14 Then
    Call spero               'First3-6in passive (a 4-14)
ElseIf tipo=18 Then
    Call first_others1       'Fist others (a 8-18)
ElseIf tipo=21 Then
    Call monere              'Second (B 1-21)
ElseIf tipo=22 Then
    Call vereor              'Second deponent (b 2-22)
ElseIf tipo=23 Then
    Call studeo              'Second no-passive (b 3-23)
ElseIf tipo=24 Then
    Call caveo               'Second3-6in-passive (b 4-24)
ElseIf tipo=25 Then
    Call sedeo               'Second3in-passive (b 5-25)
ElseIf tipo=31 Then
        Call ago             'Third (c 1-31)
ElseIf tipo=32 Then
    Call loquor              'Third deponent (c 2-32)
ElseIf tipo=33 Then
    Call resido              'Third no-passive (c 3-33)
ElseIf tipo=34 Then
    Call eruo                'Third  3-6in-passive (c 4-34)
ElseIf tipo=35 Then
    Call surgo               'Third  3in-passive (c 5-35)
ElseIf tipo=41 Then
    Call accipio             'Mixed 3rd & 4 th (d 1-41)
ElseIf tipo=42 Then
    Call aggredior           'Mixed 3rd & 4th deponent (d 2-42)
ElseIf tipo=51 Then
    Call audio               'Fourth regular (e 1-51)
ElseIf tipo=52 Then
```

```
        Call mentior            'Fourth deponent ( e 2−52)
    ElseIf tipo=54 Then
        Call sentio             'Fourth3−6in passive ( e 4−64)
    ElseIf tipo=58 Then
        Call fourth_others      'Fourth others ( e 8−48)
    ElseIf tipo=61 Then
        Call esse               'sum
    ElseIf tipo=62 Then
            Call possum         'possum
    ElseIf tipo=63 Then
        Call eo                 'eo
    ElseIf tipo=64 Then
        Call fero               'fero
    ElseIf tipo=65 Then
        Call facio              'facio
    ElseIf tipo=70 Then
        Call irreg1             'Irregular1 : for,aio,ave,decet etc.
    ElseIf tipo=71 Then
        Call irreg2             'Irregular2 : nolo,malo,queo,memini
    ElseIf tipo=90 Then
        Call doble              'volo(1), volo(irreg)
    ElseIf tipo=0 Then
        MsgBox "その動詞は登録されておりません。あるいは入力間違
                いです。"
        Range("G1").Select
    End If
End Sub
```

ここで大切なのは次の部分である。

```
j=1
Do While j <= n
    If WS3.Cells(j, 1)=WS1.Range("G1").Value Then
        Exit Do
```

```
        End If
        j = j+1
    Loop
    If j <= n Then
        tipo = WS3.Cells(j, 2)
    Else
        tipo=0
    End If
```

ワークシート１のG1セルに入力した文字が、ワークシート３のA列に入力されている文字と一致するまで調べよ。一致したら、このループを抜けよ。その際にはパターン番号(ここでは tipo)を所得してその番号のプロシージャに呼び出せという意味である。

もし見つからなければ、“その動詞は登録されておりません。あるいは入力間違いです。” というメッセージボックスが開かれ再度入力を促すことになる。これは入力間違いをチェックするには大変に有効な措置である。

なお、この部分は最初は、VLOOPUP 関数を使おうと考えていたが、その返り値の処理が分からず、また MOUG に質問したところこれまたすぐに上級者からの解答が得られた。プログラムの主要な部分は多くの人の助言で出来ている。また上記の部分は、VLOOKUP 関数と同じ作用をする。

パターンは一応上記のように22に分けたが、恣意的なもので絶対的なものでは決してない。もっと細かく分けてもよいし少なくてもよい。要は処理が簡単に済むようにすればよいのだ。第一変化動詞には10番台の数字を、第二変化動詞には20番台の数字をというようにして、以下同様に割り振った。どれくらいのパターン分けが必要になるかは未定だから空き番号を用意しておくのが賢明だ。最終的には類型により同じ末尾の番号になるように、元の数字を置換した。検索、置換はコンピュータの得意とする機能である。

明らかに不規則な動詞はデータをそのまま表示させるようにした方が便利である。

5.4 辞書、文法書だけでは語形変化は分からない

ここで、辞書、文法書だけでは動詞活用の正しい変化形は得られないことを指摘しておきたい。例えば、sperō(期待する)は、すべての辞書で語形変化の部分は次のように記述されている。

spērō, āre, āvī, ātum

これに従えば、まったく規則変化をするようだ。しかし「555語」、「501語」ともに、受動相では3人称単複のみの変化しかない。つまり、辞書、文法書だけでは正確な変化形は求められないと言える。

また、ここでtipo 90の部分に気を付けて頂きたい。

5.5 同形動詞の処理

入力時には、長母音と短母音の区別なしに行う。とすると、本来、母音の長短で区別していた動詞あるいは全く同形の動詞の処理はどうなるのだろうと言う疑問が生じる。

その部分の処理がdobleプロシージャである。

```
If pls= "volo" Then
    Range("C2")="vol" & ChrW(333) & "飛ぶ"
    Range("D2")="v" & ChrW(333) & "l" & ChrW(333) &"～
                したい"
ElseIf pls= "lego" Then
    Range("C2")="l" & ChrW(275) & "g" & ChrW(333) &"派
                遣する"
    Range("D2")="leg" & ChrW(333) & "集める"
ElseIf pls="appello" Then
    Range("C2")="appell" & ChrW(333) & "呼ぶ"
    Range("D2")="appell" & ChrW(333) & "持ってくる"
        :              :              :
ElseIf pls="sero" Then
    Range("C2")="ser" & ChrW(333) & "結合する"
    Range("D2")="ser" & ChrW(333) & "種をまく"
```

voloと入力するとそれに該当する動詞は、volō(飛ぶ)とvōlō(～したい)

の２動詞がある。どちらの動詞を表示させるか指示を与えねばならない。そこでC2セルには volō をD2セルには vōlō を長短母音記号を明示して表示、数字の１，２でもって選択させることにした。１ならば volō を、２ならば vōlō を処理することになる。次の部分である。

Msg＝“C2セルの動詞を選ぶときは１を、D2セルの動詞を選ぶときは２を押して下さい。”
Title＝“動詞を番号で選択”
GetStr = InputBox (Msg, Title)

```
Select Case GetStr
    Case“1”
        Call Caso1
    Case“2”
        Call Caso2
    Case“”
    Case Else
        MsgBox“入力間違いです。”
End Select
```

この場合１を選ぶと、Caso 1プロシージャで

```
If pls＝“volo” Then
    Call amare
```

volō は第一変化動詞だから、第一変化動詞処理プロシージャ、amare を呼び出す。しかし volō はシート３では、分岐番号が90として指示がないから現在語幹をどこかで指定せねばならない。それは amare プロシージャの次の所で行う。 pls は Present First Singular のつもりで表記した。

```
If pls＝“volo” Then
    Present＝“vol”
```

ここで現在語幹を得る。後は amare と同じ処理だが、volō には受動相

変化形がない。それを処理するのが次の箇所だ。

```
If pls="volo" Then
For i=74 To 146
    Conj (i) =""
Next
```

受動相の変化形には74番から146番までの番号が割り振ってある。それを表示させないためには、何のデータも含ませないようにする。"" の部分はそこが空であることを意味する。73の形を一気に処理できるのが変数を使う強みであり理由である。

もう一つの vōlō は、Caso 2プロシージャに飛んで、

```
If pls="volo" Then
    Call irreg 2
```

今度は、不規則動詞処理プロシージャの ireeg2で変化形を得て、hyouji プロシージャに飛び、実際に画面に表示されることになる。

volō、vōlō のように語形も変化パターンが異なる動詞はある意味では処理しやすい。しかし語形もパターンも同じような動詞の場合はどのようにすればよいか。その例として serō を見てみよう。いずれも第三変化動詞である。C2セルとD2セルでは全く同じ表示になる。

```
ElseIf pls="sero" Then
    Range("C2")="ser" & ChrW(333) & "結合する"
    Range("D2")="ser" & ChrW(333) & "種をまく"
```

これではどうしようもないが、"結合する"の場合はCasolプロシージャで

```
ElseIf pls="sero" Then
    Call eruo
```

第三変化動詞で受動相内では3人称単複しかない、“eruo”プロシージャで処理する。

“種をまく”の場合はCaso 2プロシージャで

```
ElseIf pls= “sero” Then
    Call surgo
```

第三変化動詞で受動相内では3人称単数しかない“surgo”プロシージャで処理するようにする。

しかし“種をまく”は実際には受動相で単複の変化形があるので、しかるべく対応がなされることになる。このように全く同形、同パターンの動詞の場合は、処理するプロシージャを変えないとうまく処理できないことに注意しよう。なお母音の長短を区別せずに入力すると、同形の動詞がかなりでてくるようだ。また母音の長短に関係なく同形の動詞も沢山出てきそうだ。同形処理のプロシージャをいくつか増やすようになるかも知れない。

6　処理プロシージャのサンプル

それでは実際に処理はどのように行われるか見てみよう。

一番簡単な、第一規則動詞、amareプロシージャはつぎのようになっている。

6.1

```
Private Sub amare()’11
    Set WS1=Worksheets(“sheet 1”)
    Set WS2=Worksheets(“sheet 2”)
    Set WS3=Worksheets(“sheet 3”)
```

先頭のPrivate SubのPrivateは、記述したモジュール内の他のプロシージャからしか参照できない。いわゆるサブルーチン的な働きをする。Public Subはすべてのモジュールのすべてのプロシージャから参照できると

いう違いがある。

セットステートメントは、 WS1でもってWorksheets("sheet1")を表す。
これはプログラムの作成に役立ち、可読性を大いに高める。これを使わないと、ワークシート1を参照するのに、いちいちWorksheets("sheet1")と表現せねばならない。
なお、この部分は各プロシージャに出てくるので、冒頭の宣言部で処理しようと何回か試みたがうまくいかなかった。Setステートメントは宣言部では使えないようだ。

6.2

```
If pls="volo" Then
    Present="vol"
ElseIf pls="lego" Then
    Present="l" & ChrW(275) & "g"
        :       :       :
Else : Present=WS3.Cells(j,3)
End If
Inf=Present & WS2.Cells(66,1)
Perfect=Present & WS2.Range("A61") & "v"
Participle=Present & WS2.Range("A61") & "t"
End If
```

この部分は、同形動詞の処理部分と、第一変化規則動詞の処理部分である。
これらの動詞は現在語幹さえ分かればあとは処理できるので、現在語幹を導き出している。

```
For i=1 To 18
    Conj(i)=Present & WS2.Cells(i,1)
Next
ここでは能動相直説法現在形、未完了過去形、未来形を取得。
For i=19 To 36
```

```
    Conj(i) = Perfect & WS2.Cells(i, 1)
Next
ここでは能動相直説法完了形、過去完了形、未来完了形を取得。
For i = 37 To 48
    Conj(i) = Present & WS2.Cells(i, 1)
Next
  ここでは能動相接続法現在形、未完了過去形を取得。
For i = 49 To 60
    Conj(i) = Perfect & WS2.Cells(i, 1)
Next
  ここでは能動相接続法完了形、過去完了形を取得
For i = 61 To 65
    Conj(i) = Present & WS2.Cells(i, 1)
Next
  ここでは５つの命令形を所得。
Conj(66) = Inf
Conj(67) = Participle & WS2.Cells(67, 1)
Conj(68) = Perfect & WS2.Cells(68, 1)
Conj(69) = Present & WS2.Cells(69, 1)
Conj(70) = Participle & WS2.Cells(70, 1)
Conj(71) = ""
Conj(72) = Present & WS2.Cells(72, 1)
Conj(73) = Participle & WS2.Cells(73, 1)
ここでは不定詞、分詞類を取得
For i = 74 To 91
    Conj(i) = Present & WS2.Cells(i, 1)
Next
ここでは受動相現在形、未完了過去形、未来形を所得。
For i = 92 To 109
    Conj(i) = Participle & WS2.Cells(i, 1)
Next
ここでは受動相完了形、過去完了形、未来完了形を取得。
For i = 110 To 121
```

```
        Conj(i)=Present & WS2.Cells(i, 1)
    Next
      ここでは受動相接続法現在形、未完了過去形を取得。
    For i=122 To 133
        Conj(i)=Participle & WS2.Cells(i, 1)
    Next
      ここでは受動相接続法完了形、過去完了形を取得。
    For i=134 To 137
        Conj(i)=Present & WS2.Cells(i, 1)
    Next
      ここでは4つの命令形を所得。
    Conj(138)=Present & WS2.Cells(134, 1)
    Conj(139)="[" & Participle & WS2.Cells(139, 1)
    Conj(140)=Participle & WS2.Cells(146, 1)
    Conj(141)=""
    Conj(142)=""
    Conj(143)=Participle & WS2.Cells(147, 1)
    Conj(144)=""
    Conj(145)=Present & WS2.Cells(145, 1)
    Conj(146)=""
    ここでは不定法、分詞類を取得。

    Call Hyouji
End Sub
```

すべての変化形を所得したら、Hyouji プロシージャを呼び出してシート1にその形が表示されることになる。

Hyouji プロシージャではどのセルにどの語形が入るかを指示している。

表示が終われば、またG2セルにフォーカスが移動して再度入力を促すことになる。

```
    Range("C:C"). Columns.AutoFit
    Range("D:D"). Columns.AutoFit
    Range("G:G"). Columns.AutoFit
```

```
Range("I:I").  Columns.AutoFit

pls=""
Range("C2：G2")=""
Range("G1").Select
```

これは Hyouji プロシージャの最後の部分である。

Autofit メソッドでもって表示幅を自動的に調整してくれる、大変にありがたいメソッドである。

プログラムは全体で3,900行になるし、約119,000バイトになりとてもここへは載せられない。とにかくメモリを気にせずに済むことはありがたい。

またこの記述が一番優れているとは言えない。もっと簡単に済ませることもできるだろう。例えば表示番号を順番に、直説法現在語幹、完了語幹、接続法現在語幹、完了語幹と割り振ったが、現在語幹はまとめて通し番号にしてもよいし、完了語幹もまとめてもよい。そうすればもう少し短く記述できるだろう。

このようにして、2,100語の動詞すべてを処理していくことになる。プログラムの構成は出来ているので後はどの動詞をどのように処理するかの問題である。とにかく根気が必要であるし、根気だけが必要でもある。

7　プログラムの意義

ここで何のためにこのプログラムを作成したかを述べておきたい。

ラテン語の教科書や辞書には動詞活用表が載ってはいる。しかしごく一部の動詞だけである。

volō や vōlō に受動形がないことなどはどこにも記載がない。

また既に述べたが、ある動詞の受動相で３人称単複あるいは３人称単数しかないなどもまったく分からない。

ここに動詞活用表の存在理由がある。何冊もの本が出版される理由があるし大変に分厚いものになる。それをコンピュータで行えば、瞬時に表示、参照できるし、単語数も必要に応じていくらでも増やせる。限にこのソフトでも「555語」の約４倍の処理能力がある。本の形ではとても出せない

ことができる。それだけでも価値があるのではないかと思う。

8　言語にとって簡単とは何か

このプログラムは単語の選択が決まってからは約1ヶ月で出来た。まだ未処理の動詞も400ほどあるが、時間の問題である。

プログラミングは簡単だろうという予想は前からあった。

なぜならラテン語は、同じ時制内では語幹変化がないからである。

確かに一つの動詞で146通りの変化形(分詞類を変化させればもっと増える)は数が多いと言えるだろう。しかし活用語尾はすぐに慣れてしまうから面倒ではあるが困難とは言えない。これは規則性が高いからであるとも言える。

ラテン語と比較して現在のロマンス語例えばスペイン語を例に取ると、動詞 "tener" は直説法現在形だけでも、"teng-o", "tien-es", "ten-emos" と語幹が3つ変化する。さらに不定過去では、"tuv-e", etc. また未来形、過去未来形では、"tendr-é" と代わる。同じ時制で同じ語幹ならまだしも、同じ時制で語幹が異なるのはかなり複雑である。

スペイン語はラテン語と比べると他の要素もあり確かに簡単ではあるが、動詞変化に関してだけならどちらが複雑かは単純には言えない。規則性が高ければそれなりに簡単と言えるかも知れないからだ。

ところで英語はどうだろうか。動詞は語形を3個覚えればよいので一見簡単そうに見える。しかしロマンス系の言語を勉強した者にとっては簡単とは言えないのではないだろうか。その理由の一つに文法性の度合いがあるのではないか。英語はほとんど文法らしいものがないようにも見える。しかし言語だから何らかの約束事が無ければコミュニケーションはできない。英語の場合それは慣用表現によるところが大きいのではないだろうか。ここで考えて欲しいのは文法規則は閉じられた系であるのに対して、慣用表現は開かれた系であると言うことだ。開かれた系には無数の要素が入りうる。その場その場での表現を覚えていくのはかえって大変に困難なことではないだろうか。

さらに言わせてもらえば、英語はインドヨーロッパ語族のなかでもかなり特殊な言語と言えるかも知れない。その特殊性が他の言語にも当てはまるかは大いに疑問があると言えるのではないか。

よくある言語は簡単だ、あるいは難しいと言うことがあるが、ある面で簡単なら他の面で複雑なこともあるのが通例であるから、単純には答えは出てこないであろう。従って自然言語については簡単な言語、難しい言語という区別は意味がないと言える。

9 最後に

このプログラムの主要な部分は、MOUG の方々の助言により出来ていることは既に述べた。ここに改めて謝意を述べたい。そのうちに私も助言者になり、お返しをしたい。

参考文献および参照対象

1）動詞活用表

Betts, Gavin & Franklin, Daniel：*"The Big Gold Book of Latin Verbs. 555 fully Conjugated Verbs"*, McGraw-Hill, 2004

E. Prior, Richard & Wholberg, Joseph：*"501 Latin Verbs. fully conjugated in all the tenses"*, Barron's Educational Series, Inc. 1995

2）辞書

田中秀央：「羅和辞典」、研究社、1961

Lewis, T.Charlton：*"An Elementary Latin Dictionary"*, Oxford Univ. Press

KIdd, D.A. & Wade Mary：*"Collins Latin Dictionary plus Grammar"*, Collins, 2004

3）文法書

泉井久之助：「ラテン広文典」、白水社、1962

松平千秋、国原吉之助：「新ラテン文法」、南江堂、1968

4）その他

布施温：「スペイン語動詞活用表」、2003年11月、スペイン語研究者のメーリング・リストに発表及び、Excel メーリング・リスト、MOUG