科学技術館CanSatプロジェクト
第１期

第2回　データの読み込みと書き込み

今回の流れ

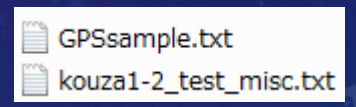
1　テキストファイルの読み込み
2　Excelへのデータ変換
3　GPSモジュールのデータ
4　テキストファイルの書き込み

※教材を利用するすべての皆様へ

今回は途中でデバッグ機能の説明をします。その後の作業で行き詰まることがあったら、デバッグ機能を積極的に活用して、問題の解決を試みてください。

例題など一部のソースコードについては、入力の手間を省けるよう「kouza1-2_test_misc.txt」を用意してあるので、ご活用ください。

また、教材用データファイルとして、GPSsample.txt　を用意してあります。（当日参加の方は「X:¥CanSat」から入手できます。）



※参加者の皆様へ

全部を2時間以内にこなすのは難しいかもしれません。できるところまでやりましょう。

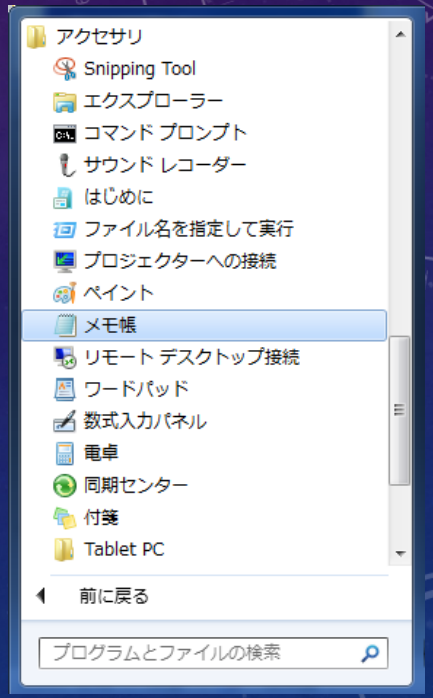
※独習・復習される方へ

この教材は、順番に読んでいただきたいのですが、1回読んで全てを理解しなくても大丈夫です。おそらく、後の方を読んでからでないとわからない所もあります。

実践しながら、繰り返し読むことをお勧めします。

1　テキストファイルの読み込み

・Wordの文書は、テキストファイルではありません。フォントの大きさや色が指定でき、ページのサイズや余白などのデータも入っています。



・テキストファイルとは、文字のデータだけが入っているファイルのことです。スタートメニューのアクセサリ-メモ帳を使うと作れます。

（メモ帳はWindowsに標準添付されていますが、
もっと便利なフリーの「テキストエディタ」もたくさんあります。）

・まずは、VBAに読ませるサンプルファイルを作ることから始めましょう。

メモ帳で作るテキストファイルの内容

・「何が書かれているか」「どのように書かれているか」が
はっきりしていないと、それを読むプログラムは作れません。
この例では、「XY平面上の点の座標を書く」ということにして、
「コメントは行頭に#（ナンバー）をつけて表す。
　座標データは1行に1個で、何行でも書ける。
　X座標とY座標は”,”で区切る。」
という書式を定めておきます。

※「コメント」は人間向けのもので、処理するプログラムからは無視されるようにします。

※「#」は音楽記号の＃とは異なる「ナンバー」という記号ですが、「シャープ」と呼んでも通用します。

・上記のフォーマット（データの書式）にしたがって、
正5角形の頂点データを書くと右のようになります。
メモ帳を開き、この内容を打ち込んで、”sample.txt”というファイル名で保存してください。

例題1：テキストファイルの内容を表示する

・sample.txtを1行ずつ読み込んで表示するプログラムを作ります。

・Excelの空白ブックを1個作ってください。

・標準モジュールを1個追加して、xlsmファイルとして、USBメモリ等に保存してください。

・右の内容を打ち込み、実行してください。

・「'」（ダッシュ）はコメントを表し、処理系には無視されます。

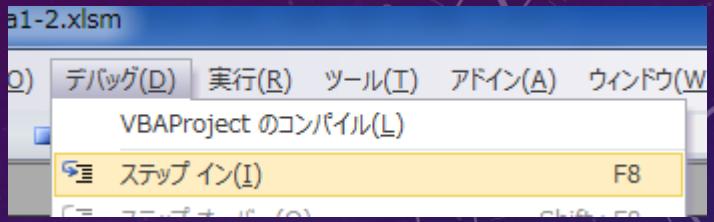
（コメントは人間のためのものです。動作するだけでよければ打ち込まなくてもかまいません。）

・「Dim fn As Str」まで入力すると、アシスト機能が現われます。TABキーでその機能を試しましょう。この機能を使いこなせるようになると、プログラムの打ち込みが簡単になります。

・このアシスト機能が現われるかどうかで、スペルの違いも察することができます。

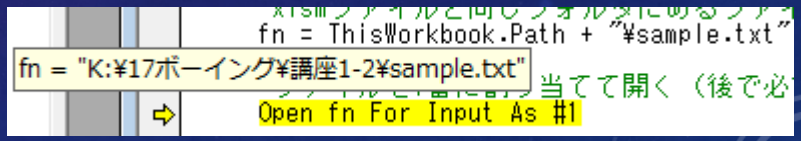
```
Option Explicit

Sub test1()
    Dim fn As String, buf As String

    'xlsmファイルと同じフォルダにあるファイルのフルパス名
    fn = ThisWorkbook.Path + "¥sample.txt"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    Do Until EOF(1)
        Line Input #1, buf   '1行読み込む
        MsgBox buf   '内容を表示する
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

デバッグ機能の活用



- 実行するプロシージャの中にカーソルがある状態で、
  デバッグ-ステップインをクリックすると、デバッグ機能が始動します。（F8キーでも可）
- ステップインを繰り返すと、プログラムが1行ずつ実行されます。
- 黄色の矢印は、実行待ち状態の位置を示します。

- ”fn = …“が実行待ちの時に、マウスカーソルをfnに
  かぶせると、fnの内容が空であることがわかります。
- F8キーで1行実行すると、fnに代入された文字列を
  確認できます。



- この機能によって、プログラムの動作状況を逐一確認できます。
  先ほどのプログラムをすべてステップイン実行し、使っている変数の内容が
  どのタイミングでどのように変わっているかを確認してください。

「フルパス名」を作るための「クラス」「オブジェクト」と「メンバ」

・Excelが実行されるときの「カレントフォルダ」は、Excelのファイルがあるフォルダではありません。そのため、ファイル名だけを指定してもファイルを開くことができません。

・コンピュータが管理しているフォルダ全体の中で、目的のファイルにたどり着くための経路を示す文字列を「フルパス」と呼び、フルパス¥ファイル名のことを「フルパス名」と呼びます。

```
fn = ThisWorkbook.Path + "¥sample.txt"
```

・ワークブック(xlsmファイル)のフルパスは、**ThisWorkbook.Path**によって取得できます。

下記のような事情がありますが、まずは呪文のように覚えてかまいません。

・"ThisWorkbook"は、実行中のワークブックの「オブジェクト」を表す変数であり、ExcelVBAの処理系ではあらかじめ定義されているものです。

・ThisWorkbookの型は「Workbookクラス」です。「クラス」とは、ある一つの事柄に関連するプロシージャと変数をまとめたものです。「モジュール」をまるごと型にしたようなもの、とも言えます。

・LongやStringという型は、変数を定義すると使えるようになります。クラスも、「オブジェクト変数」を定義すると使えるようになります。

・オブジェクトの中身は、文字列や数値ではなく、プロシージャや変数という「メンバ」で構成されます。

それらは、**[オブジェクト名].[メンバ名]** という書き方で、オブジェクトの外部から使うことができます。

・そのようなわけで、WorkbookオブジェクトであるThisWorkbookのメンバ変数であるPathに格納されたフルパスは、上記のコードを書くと使えます。

Openステートメント,Closeステートメント

・Open [ファイル名] For [開き方] As #[ファイル番号]
とすると、ファイルを開きます。開き方は主に下記3種類を使います。

読み込み：Input

新たに最初から書き込み：Output

追加で書き込み：Append

・一旦開いたファイルは、プログラム終了までに
Close #[ファイル番号]　で閉じる必要があります。

```
Option Explicit

Sub test1()
    Dim fn As String, buf As String

    'xlsmファイルと同じフォルダにあるファイルのフルパス名
    fn = ThisWorkbook.Path + "¥sample.txt"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    Do Until EOF(1)
        Line Input #1, buf   '1行読み込む
        MsgBox buf   '内容を表示する
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

Do Untilステートメント

- Do Until [条件式]
  とすると、[条件式]が成り立たない間は、Loopまでを繰り返します。
  （ForでもDoでも、繰り返しの構造は「ループ」と呼ばれます。）

- Ifでもそうですが、[条件式]の部分は、Boolean型の値(TrueまたはFalse)が入ります。そのため、戻り値がBoolean型であれば、関数を条件式の代わりに使うこともできます。

- この例では、EOF(1)がFalseである間は処理を繰り返し、EOF(1)がTrueになったらループを抜けるということになります。

```
Option Explicit

Sub test1()
    Dim fn As String, buf As String

    'xlsmファイルと同じフォルダにあるファイルのフルパス名
    fn = ThisWorkbook.Path + "¥sample.txt"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    Do Until EOF(1)
        Line Input #1, buf   '1行読み込む
        MsgBox buf   '内容を表示する
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

EOF関数、Line Inputステートメント

・ファイルを開く処理は、直接的にはOSが行っていて、開いている間は、ファイル上のアクセス位置が常に管理されています。（メモ帳のカーソルのようなものです）

・EOF([ファイル番号])　は、ファイルの終端に達しているときはTrueを、そうでないときはFalseを返します。

・Line Input [ファイル番号], [読み込みバッファ] で、ファイルの内容を1行ずつ読み込むことができます。

・1行読み込むと、ファイルのアクセス位置が1行後にずれます。それが最後まで達したらEOF()はTrueを返すようになります。

※EOFはEnd Of Fileの略です。

```
Option Explicit

Sub test1()
    Dim fn As String, buf As String

    'xlsmファイルと同じフォルダにあるファイルのフルパス名
    fn = ThisWorkbook.Path + "¥sample.txt"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    Do Until EOF(1)
        Line Input #1, buf   '1行読み込む
        MsgBox buf   '内容を表示する
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

2　Excelへのデータ変換

・VBAでテキストファイルを読み込むだけでは、Excelの機能を生かせません。

・テキストファイルの内容をセルに書き込むと、手で入力したデータと同様に扱えます。

・ただし、データをExcelに適した形に変換する必要があります。
　その理由と方法について、課題を解きながら学びましょう。

課題1：セルにファイルの内容を入れる

・test1の内容をコピペ（コピーしてペースト）して、
　kadai1というプロシージャを作ってください。
・MsgBoxでファイルの内容を1行ずつ表示していますが、その代わりにセルに入れることを課題とします。
・開始位置は(1,1)とします。

・Cells([行番号],[列番号]) = buf　で1行のデータは入れられますが、行番号を変えていかないと、最後の行しかデータが残りません。
・行番号を1ずつ変えていくしくみを、前回の内容を思い出しながら組み込んでください。

＜制限時間　5分＞

```
Sub kadai1()
    Dim fn As String, buf As String

    'xlsmファイルと同じフォルダにあるファイルのフルパス名
    fn = ThisWorkbook.Path + "¥sample.txt"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    Do Until EOF(1)
        Line Input #1, buf   '1行読み込む
        MsgBox buf   '内容を表示する
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

課題1の回答例

・行番号のカウンタとして、変数cを定義します。

・読み込みループに入る直前に1を入れておき、
1行読むたびにcを1ずつ増やします。

・Cellsの行番号としてcを使えば、すべての行が
順番にセルに入っていきます。

	A
1	#X座標,Y座標
2	1.000,0.000
3	0.309,0.951
4	-0.809,0.588
5	-0.809,-0.588
6	0.309,-0.951

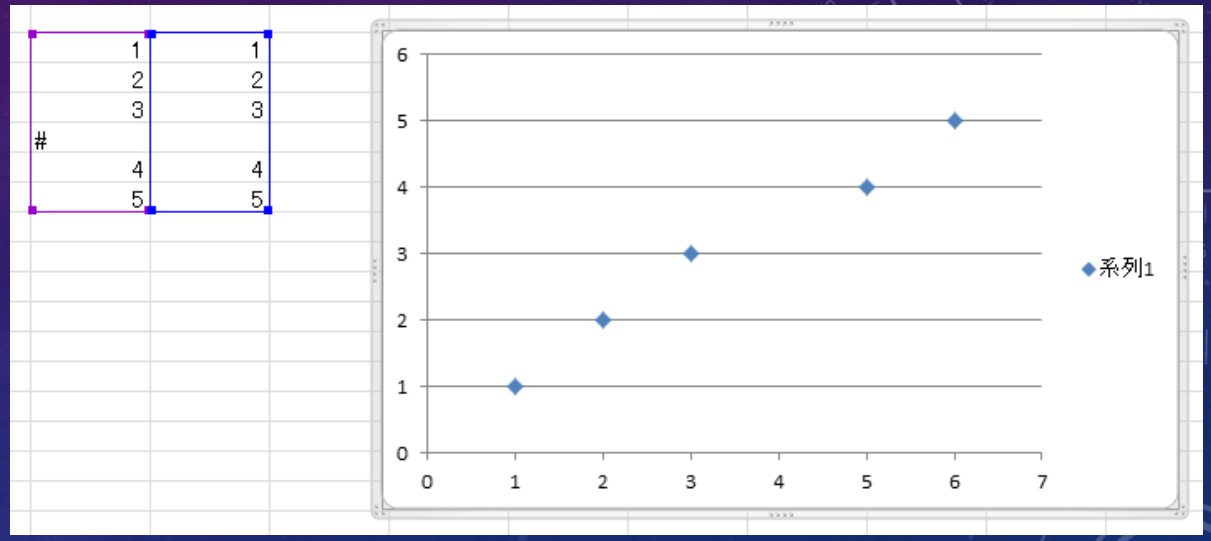
```
Sub kadai1()
    Dim fn As String, buf As String
    Dim c As Long

    'xlsmファイルと同じフォルダにあるファイルのフルパス名
    fn = ThisWorkbook.Path + "¥sample.txt"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 1
    Do Until EOF(1)
        Line Input #1, buf   '1行読み込む
        Cells(c, 1) = buf
        c = c + 1
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

文字列の分解と認識

・テキストファイルを読み込んでセルに入れるところまでできましたが、
Excelのデータとしては、下記のような問題があります。

	A
1	#X座標,Y座標
2	1.000,0.000
3	0.309,0.951
4	−0.809,0.588
5	−0.809,−0.588
6	0.309,−0.951

①現在、X座標とY座標が1個のセルの中に文字列
として入っています。Excelはこれを数値として
扱えないので、X座標とY座標を別々のセルに
入れ、1つずつの数値にしておく必要があります。



②数値データからグラフを作るとき、数値データにコメントが含まれていても、Excelは个のように
グラフを表示しますが、不具合が起き、しかもそれが発見しづらいという危険性があります。

・文字列の分解と認識を適切にすれば、上記2つの問題は解決できます。

今後の作業のための準備1

・今後、ファイルの内容を何度も読み込み、セルに入れる実験を繰り返します。
その結果を正しく見るためには、実験する前に書き込んだセルの内容を消去する必要があります。

・その作業を自動的に行うために、右のプロシージャが必要です。
標準モジュールを1個増やして、そこに右の内容を入れてください。

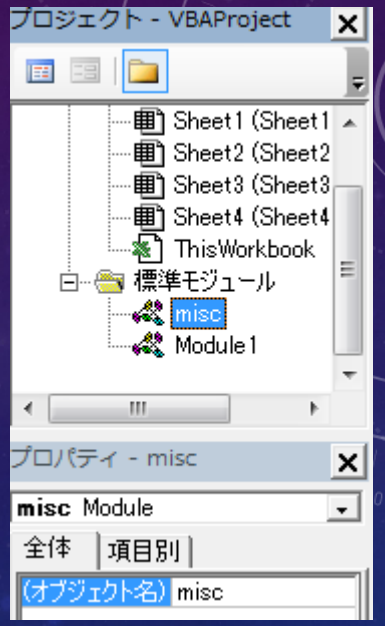
※行頭に「Public」とありますが、これは「他のモジュールから使いますよ」という意味のキーワードです。

```
Option Explicit

'使用中の最後の行番号を取得する
Public Function LastRow() As Long
    Dim ws As Worksheet

    'アクティブになっているシートのオブジェクトへの参照を取得する
    Set ws = ThisWorkbook.ActiveSheet

    '最終行 = 使用中領域の最初の行番号 + 使用中領域の行数 - 1
    LastRow = ws.UsedRange.Row + ws.UsedRange.Rows.Count - 1

End Function
```

```
'使用中の最後の列番号を取得する
Public Function LastColumn() As Long
    Dim ws As Worksheet

    'アクティブになっているシートのオブジェクトへの参照を取得する
    Set ws = ThisWorkbook.ActiveSheet

    '最終列 = 使用中領域の最初の列番号 + 使用中領域の列数 - 1
    LastColumn = ws.UsedRange.Column + ws.UsedRange.Columns.Count - 1

End Function
```

```
'Cells(br,bc)を左上として、Cells(er,ec)を右下とする領域の内容を消去する
Public Sub ClearCells(br As Long, bc As Long, er As Long, ec As Long)
    If er >= br And ec >= bc Then
        Range(Cells(br, bc), Cells(er, ec)).Clear
    End If

End Sub
```

今後の作業のための準備1



・2番目の標準モジュールを追加すると「Module2」という名前になりますが、
この名前は変えることができます。
左側のプロジェクトの中のモジュール名をクリックし、
下の「オブジェクト名」を変えることで、名前を変えられます。
「misc」という名前にしておきましょう。

※miscはmiscellaneous(寄せ集めの、雑多な)の略語で、ソフトウェア開発の
現場ではよく使われる単語です。

例題2：作成した関数の動作確認

・先ほど追加したプロシージャのテストを行います。

・kadai1をコピペしてtest2を作ってください。

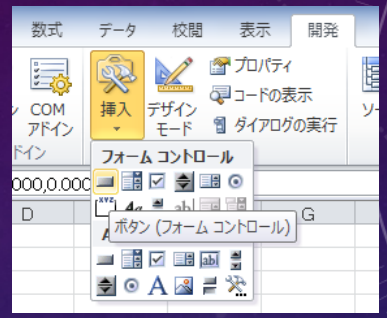
・test2()の始めの方に、右のように ClearCellsとLastRow, LastColumnを使うコードを挿入してください。

※このようにして、細かい部品となるプロシージャと、それを使って目的の作業を行うプロシージャを分けて開発することができます。

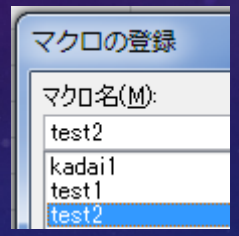
```
Sub test2()
    Dim fn As String, buf As String
    Dim c As Long

    '使用中のセルをすべて消去する
    ClearCells 1, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダにあるファイルのフルパス名
    fn = ThisWorkbook.Path + "¥sample.txt"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 1
    Do Until EOF(1)
        Line Input #1, buf   '1行読み込む
        Cells(c, 1) = buf
        c = c + 1
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

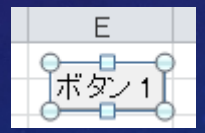
今後の作業のための準備2

・ワークシート画面に移り、開発-挿入から、左上のボタン（フォームコントロール）をクリックします。するとマウスカーソルが十字になります。

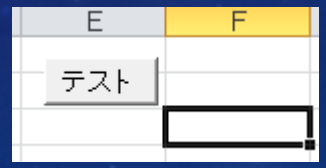


・ワークシートの右の方の空いている所をクリックします。すると
「マクロの登録」ダイアログが出るので、「test2」を選んでから
OKをクリックします。「ボタン 1」などと書かれたボタンが出現しますが、
その直後の状態でクリックすると、ボタンの文字を編集できます。
「テスト」などわかりやすい文字に書き直してください。





・他の何もないセルをクリックすると、ボタンの編集状態が終了します。
その後ボタンをクリックすると、test2()が実行されるようになります。
先ほどの変更箇所が正常に動作することを確認してください。



例題3：コメントの無視

・まずtest2をコピペしてtest3を作ります。

・この例題では、先ほど提示した2つの問題のうち②を解決します。着実な開発手順として、まずコメントの無視に取り組みます。

なぜなら、コメント以外の行にはXY座標が入っているのが前提なので、もう1つの問題解決が単純にできるからです。

・コメントの無視は、Line Inputの後の処理を右のように変更するとできるので、書き換えてください。

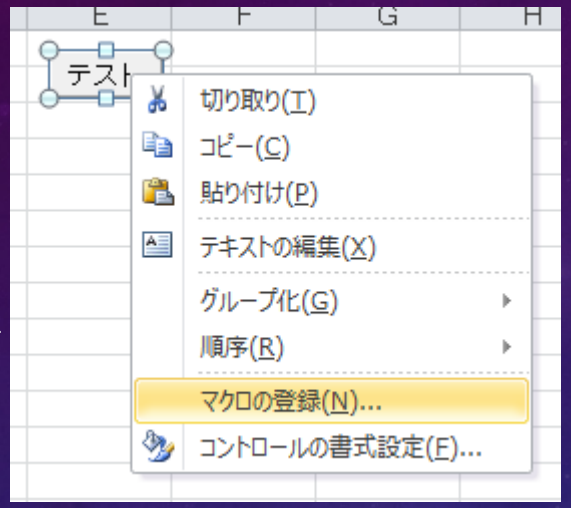
※ ＜＞ は「等しくない」という意味の比較演算子です。

※ 関数Left([文字列],[n])で、左からn番目までの文字が取り出せます。

```
Sub test3()
    Dim fn As String, buf As String
    Dim c As Long

    '使用中のセルをすべて消去する
    ClearCells 1, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダにあるファイルのフルパス名
    fn = ThisWorkbook.Path + "¥sample.txt"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 1
    Do Until EOF(1)
        Line Input #1, buf   '1行読み込む
        If Left(buf, 1) <> "#" Then
            Cells(c, 1) = buf
            c = c + 1
        End If
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

例題3：コメントの無視

・test3をVBE上で実行してもよいのですが、先ほどワークシートに作ったボタンを右クリックし「マクロの登録」を選び、実行するマクロをtest3に変更するのも便利です。



・実行結果は右のようになるはずです。

	A	B
1	1.000,0.000	
2	0.309,0.951	
3	−0.809,0.588	
4	−0.809,−0.588	
5	0.309,−0.951	

```
Sub test3()
    Dim fn As String, buf As String
    Dim c As Long

    '使用中のセルをすべて消去する
    ClearCells 1, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダにあるファイルのフルパス名
    fn = ThisWorkbook.Path + "¥sample.txt"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 1
    Do Until EOF(1)
        Line Input #1, buf   '1行読み込む
        If Left(buf, 1) <> "#" Then
            Cells(c, 1) = buf
            c = c + 1
        End If
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

・デバッグ機能を使って実行過程を確認しましょう。

例題4：文字列の区切りを認識して分割する

- 問題②が解決された結果、[X座標], [Y座標] という形式のテキストが得られています。
- まずtest3をコピペしてtest4を作り、それに機能を追加して、問題①を解決しましょう。
- 関数 Split([文字列],[区切り文字]) は、区切り文字で文字列を分割し、Variant型の配列に入れて返します。
- For Each [変数] In [配列]　は、配列の要素が1個ずつ変数に入れて、繰り返し処理をするステートメント（構文）です。
- Split, For Eachを利用すると問題①を解決できるので、右のようにソースコードを変えてください。
- ソースコードを仕上げたら、デバッグ機能を使って実行過程を確認しましょう。

```
Sub test4()
    Dim fn As String, buf As String
    Dim c As Long
    Dim spBuf As Variant, danpen As Variant, c2 As Long

    '使用中のセルをすべて消去する
    ClearCells 1, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダにあるファイルのフルパス名
    fn = ThisWorkbook.Path + "¥sample.txt"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 1
    Do Until EOF(1)
        Line Input #1, buf  '1行読み込む
        If Left(buf, 1) <> "#" Then

            'Variant型の配列を作り、1行をカンマで
            '区切って分割して入れる
            spBuf = Split(buf, ",")

            c2 = 1
            'spBufの配列の要素を1個ずつ取り出して
            'danpenに入れる処理を繰り返す
            For Each danpen In spBuf
                Cells(c, c2) = danpen
                c2 = c2 + 1
            Next

            c = c + 1
        End If
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

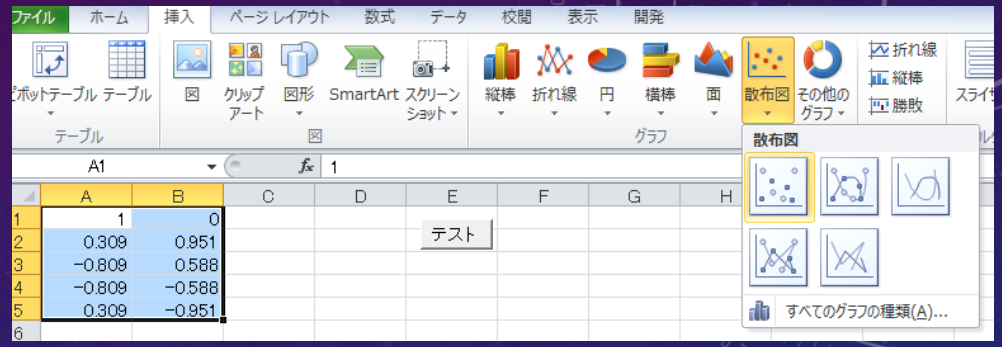
例題4：文字列の区切りを認識して分割する

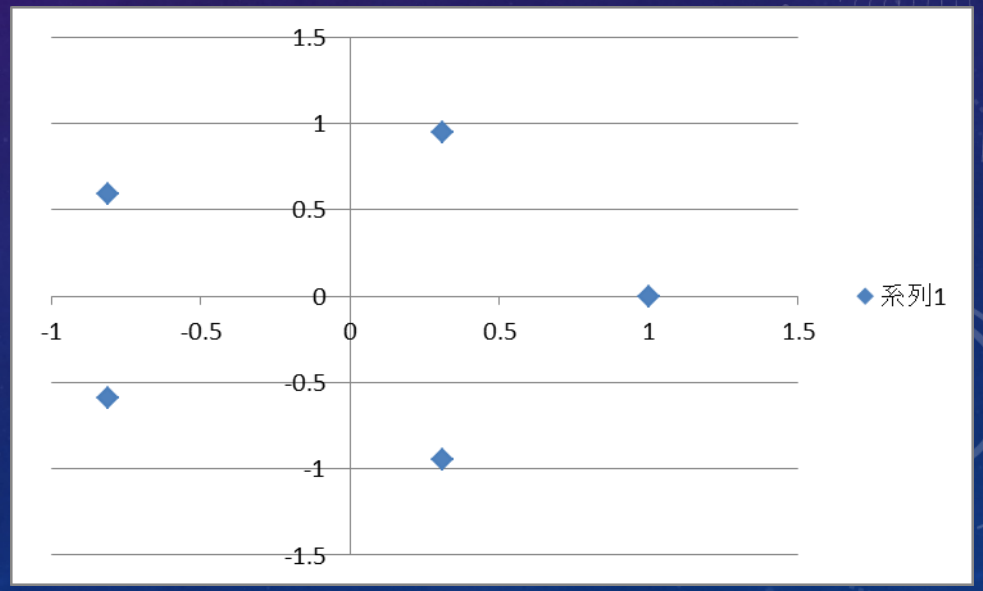
・実行すると、A列にX座標が、B列にY座標が並ぶはずです。

・実は、Splitが自動的に配列の大きさを決めてくれるので、1行の中にデータがいくつ書かれていても対応できるようになっています。

・実行結果が正しければ、ワークシート上で座標データをすべて選択し、挿入-散布図を選び、散布図（マーカーのみ）をクリックすると、右下のようなグラフが描かれます。

・このグラフはもう使わないので、削除しておいてください。





例題5：ファイルを選択するダイアログを使う

・これまではファイル名をソースコードの中に書いていましたが、実行時にユーザーが選択できるようなダイアログを表示させましょう。

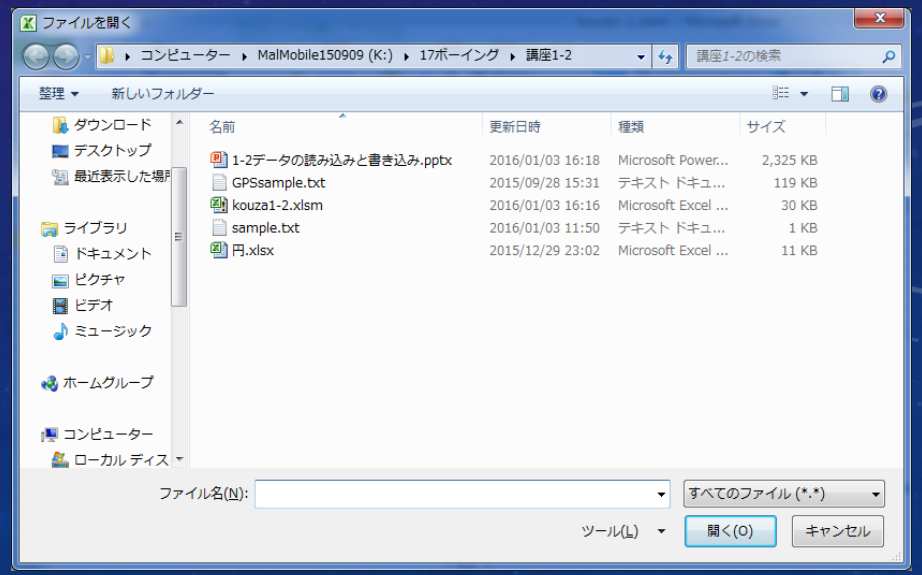
・まずtest4をコピペしてtest5を作ってください。

・ファイル名を決めていた部分を、右のように変えてください。

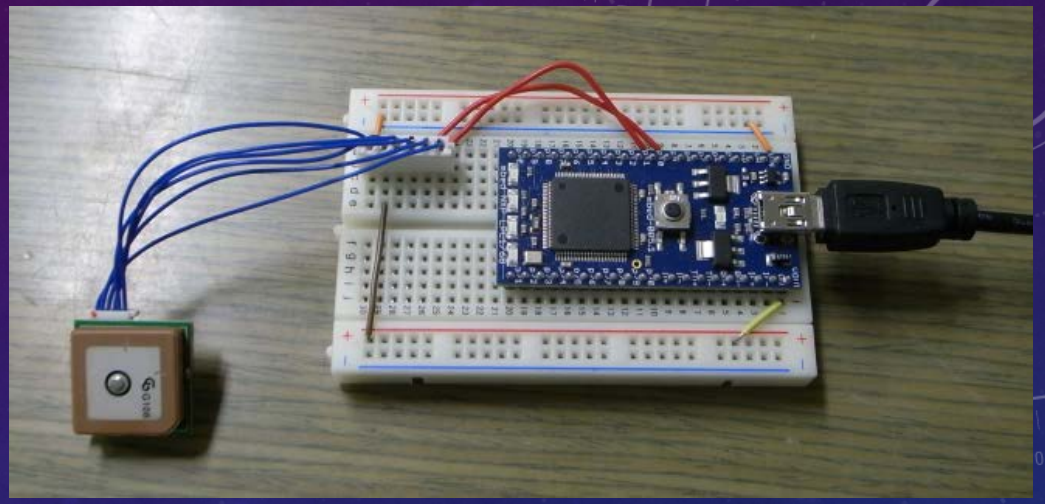
```
Sub test5()
    Dim fn As String, buf As String
    Dim c As Long
    Dim spBuf As Variant, danpen As Variant, c2 As Long

    '使用中のセルをすべて消去する
    ClearCells 1, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダをカレントフォルダにして、
    'ユーザーにファイルを選択させる
    ChDrive ThisWorkbook.Path
    ChDir ThisWorkbook.Path
    fn = Application.GetOpenFilename

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1
```

・実行すると「ファイルを開く」ダイアログが現われ、ファイルを選んで「開く」をクリックする（またはファイルをダブルクリック）すると、そのファイルを読み込んでセルに数値を入れてくれます。

・「キャンセル」をクリックすると、キャンセルはできますが、エラーが出ます。その対処法は今は扱いません。



3　GPSモジュールのデータの使い方



・CanSatは一般に、自分の位置を発信・記録するためのGPSモジュールを搭載します。

・GPS（全地球測位システム）衛星からの微弱なデータを正確に受信し、現在位置を計算するには高度な技術が必要ですが、GPSモジュールは、そのような処理をすべてパッケージ内のマイコンで実行し、データだけを接続された電子機器に送信するようになっています。

・今回は、このモジュールから出力されたデータをテキストファイルとして保存したものを題材として用意してあります。(GPSsample.txt)

NMEAフォーマット

・GPSモジュールからのデータは、「NMEA 0183」という規格に準拠したテキストになっています。

・緯度・経度だけでなく、時刻、衛星捕捉数、高度などを含んで、複数の書式で出力されています。書式の名前は「$GPGGA」のように行頭に書かれています。

・今回は時刻・緯度・経度を取得するために、GGAデータ（$GPGGAの行）だけを使います。

・カンマ区切りなので、今まで作ってきたソースコードを流用できます。

```
$GPGSV,3,2,12,17,30,130,35,10,80,174,,57,68,084,,02,62,313,*76
$GPGSV,3,3,12,04,37,069,,29,33,313,,25,20,311,,23,10,042,*7F
$GPRMC,062437.009,V,,,,,,,280915,,,N*47
$GPGGA,062438.009,,,,,0,00,,,M,0.0,M,,0000*54
$GPGSA,M,1,,,,,,,,,,,,,,,*12
$GPRMC,062438.009,V,,,,,,,280915,,,N*48
$GPGGA,062439.009,,,,,0,00,,,M,0.0,M,,0000*55
$GPGSA,M,1,,,,,,,,,,,,,,,*12
$GPRMC,062439.009,V,,,,,,,280915,,,N*49
$GPGGA,062440.019,3541.5020,N,13945.2098,E,1,04,2.9,-70.8,M,39.4,M,,0000*4E
$GPGSA,M,3,05,06,09,17,,,,,,,,,6.5,2.9,5.8*37
$GPRMC,062440.019,A,3541.5020,N,13945.2098,E,0.66,273.11,280915,,,A*6E
$GPGGA,062441.019,3541.5019,N,13945.2057,E,1,04,2.9,-80.2,M,39.4,M,,0000*43
$GPGSA,M,3,06,09,17,05,,,,,,,,,6.5,2.9,5.8*37
$GPRMC,062441.019,A,3541.5019,N,13945.2057,E,0.40,273.11,280915,,,A*62
$GPGGA,062442.019,3541.5011,N,13945.2062,E,1,05,1.8,-71.2,M,39.4,M,,0000*43
$GPGSA,M,3,06,09,17,02,05,,,,,,,,3.3,1.8,2.8*33
$GPGSV,3,1,12,06,67,352,33,09,58,088,30,17,47,173,34,02,36,309,30*75
$GPGSV,3,2,12,05,19,252,22,12,13,289,29,25,00,323,16,57,82,292,34*7E
$GPGSV,3,3,12,04,65,069,,10,56,268,,23,28,048,,29,12,320,*72
$GPRMC,062442.019,A,3541.5011,N,13945.2062,E,0.41,273.11,280915,,,A*6E
$GPGGA,062443.019,3541.5020,N,13945.2057,E,1,05,1.8,-70.1,M,39.4,M,,0000*44
$GPGSA,M,3,06,09,17,02,05,,,,,,,,3.3,1.8,2.8*33
```

課題2：GGAデータの抽出

・test5をコピペしてkadai2を作ってください。

・1番目の要素が"$GPGGA"になっている行のデータだけを
セルに入れるように、For Eachループ周辺を変更してください。

・それができたら、$GPGGAの行の2,3,4,5,6番目のデータだけを
セルに入れるように変更してください。
（結果的に、For Eachループ・danpen・c2は不要になります。）

※spBuf配列の最初の要素はspBuf(0)になることに注意してください。

※実行してみて、表示されないデータがあっても、慌てないでください。
$GPGGAの後にデータが入ってくるのは182行目以降です。

＜制限時間　5分＞

```
Sub kadai2()
    Dim fn As String, buf As String
    Dim c As Long
    Dim spBuf As Variant, danpen As Variant, c2 As Long

    '使用中のセルをすべて消去する
    ClearCells 1, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダをカレントフォルダにして、
    'ユーザーにファイルを選択させる
    ChDrive ThisWorkbook.Path
    ChDir ThisWorkbook.Path
    fn = Application.GetOpenFilename

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 1
    Do Until EOF(1)
        Line Input #1, buf  '1行読み込む
        If Left(buf, 1) <> "#" Then

            'Variant型の配列を作り、1行をカンマで
            '区切って分割して入れる
            spBuf = Split(buf, ",")

            c2 = 1
            'spBufの配列の要素を1個ずつ取り出して
            'danpenに入れる処理を繰り返す
            For Each danpen In spBuf
                Cells(c, c2) = danpen
                c2 = c2 + 1
            Next

            c = c + 1
        End If
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

課題2の回答例

・右のようにFor Eachループの部分を書き換えると、$GPGGAの2番目～6番目のデータがセルに入ります。

・GPSモジュールは、衛星を捕捉できなくても1秒に1回データを送ります。そのため、データがない行もあります。
このデータを取得した時はたまたま182番目から位置測定ができていましたが、実際の測定開始のタイミングは、状況によって変わります。

・次は、測定できているかどうかの検知が課題となります。

179	62437.01				
180	62438.01				
181	62439.01				
182	62440.02	3541.502	N	13945.21	E
183	62441.02	3541.502	N	13945.21	E
184	62442.02	3541.501	N	13945.21	E
185	62443.02	3541.502	N	13945.21	E
186	62444.02	3541.502	N	13945.2	E

```
Sub kadai2()
    Dim fn As String, buf As String
    Dim c As Long
    Dim spBuf As Variant

    '使用中のセルをすべて消去する
    ClearCells 1, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダをカレントフォルダにして、
    'ユーザーにファイルを選択させる
    ChDrive ThisWorkbook.Path
    ChDir ThisWorkbook.Path
    fn = Application.GetOpenFilename

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 1
    Do Until EOF(1)
        Line Input #1, buf  '1行読み込む
        If Left(buf, 1) <> "#" Then

            'Variant型の配列を作り、1行をカンマで
            '区切って分割して入れる
            spBuf = Split(buf, ",")

            '行頭に"$GPGGA"があったら、データをセルに入れる
            If spBuf(0) = "$GPGGA" Then
                Cells(c, 1) = spBuf(1)  '時刻
                Cells(c, 2) = spBuf(2)  '緯度
                Cells(c, 3) = spBuf(3)  'N:北緯  S:南緯
                Cells(c, 4) = spBuf(4)  '経度
                Cells(c, 5) = spBuf(5)  'E:東経  W:西経
                c = c + 1
            End If

        End If
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

課題3:有効なGGAデータの抽出

・kadai2をコピペしてkadai3を作ってください。

・緯度・経度のデータがある
GGAデータのみ抽出する
ように、 spBufの処理部分
を修正してください。

179	62437.01				
180	62438.01				
181	62439.01				
182	62440.02	3541.502	N	13945.21	E
183	62441.02	3541.502	N	13945.21	E
184	62442.02	3541.501	N	13945.21	E
185	62443.02	3541.502	N	13945.21	E

＜ヒント＞

・緯度・経度はセットになっているので、緯度があるかないかで
判断すればOKです。

・「""」で「文字が含まれない」という意味になります。

＜制限時間　5分＞

```
Sub kadai3()
    Dim fn As String, buf As String
    Dim c As Long
    Dim spBuf As Variant

    '使用中のセルをすべて消去する
    ClearCells 1, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダをカレントフォルダにして、
    'ユーザーにファイルを選択させる
    ChDrive ThisWorkbook.Path
    ChDir ThisWorkbook.Path
    fn = Application.GetOpenFilename

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 1
    Do Until EOF(1)
        Line Input #1, buf  '1行読み込む
        If Left(buf, 1) <> "#" Then

            'Variant型の配列を作り、1行をカンマで
            '区切って分割して入れる
            spBuf = Split(buf, ",")

            '行頭に"$GPGGA"があったら、データをセルに入れる
            If spBuf(0) = "$GPGGA" Then
                Cells(c, 1) = spBuf(1)  '時刻
                Cells(c, 2) = spBuf(2)  '緯度
                Cells(c, 3) = spBuf(3)  'N:北緯  S:南緯
                Cells(c, 4) = spBuf(4)  '経度
                Cells(c, 5) = spBuf(5)  'E:東経  W:西経
                c = c + 1
            End If

        End If
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

課題3の回答

・「spBuf(2) <> ""」は「緯度のデータがなくはない」という意味になります。
これで「緯度のデータがある」という条件を表現できます。

・次は、このデータを人間が読みやすいように変えてみましょう。

```
Sub kadai3()
    Dim fn As String, buf As String
    Dim c As Long
    Dim spBuf As Variant

    '使用中のセルをすべて消去する
    ClearCells 1, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダをカレントフォルダにして、
    'ユーザーにファイルを選択させる
    ChDrive ThisWorkbook.Path
    ChDir ThisWorkbook.Path
    fn = Application.GetOpenFilename

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 1
    Do Until EOF(1)
        Line Input #1, buf  '1行読み込む
        If Left(buf, 1) <> "#" Then

            'Variant型の配列を作り、1行をカンマで
            '区切って分割して入れる
            spBuf = Split(buf, ",")

            '行頭に"$GPGGA"があったら、データをセルに入れる
            If spBuf(0) = "$GPGGA" Then
                If spBuf(2) <> "" Then
                    Cells(c, 1) = spBuf(1)  '時刻
                    Cells(c, 2) = spBuf(2)  '緯度
                    Cells(c, 3) = spBuf(3)  'N:北緯  S:南緯
                    Cells(c, 4) = spBuf(4)  '経度
                    Cells(c, 5) = spBuf(5)  'E:東経  W:西経
                    c = c + 1
                End If
            End If

        End If
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

課題4-1：データの表示形式

・kadai3をコピペしてkadai4を作ってください。

・1行目に、右のように項目名を入れ、ボタンの位置をずらし（マウスの右ボタンでドラッグするとできます）、そしてKadai4を登録してください。

・データの消去および出力を2行目からするように、しかるべき修正をしてください。

＜制限時間　3分＞

※ここまでできたら、まだ下記の問題が解決されていないので、読んでおいてください。

緯度、経度は100倍されているように見えますが、実は十の位から下は60進法になっています。（例：3541.502　+　18.498　＝　3600.000　となる）

これを通常の10進法に変換するには、少し複雑な処理が必要です。

課題4-1の回答

・ClearCellsの最初の引数を2に、
　cの初期値を 2 とすればOKです。

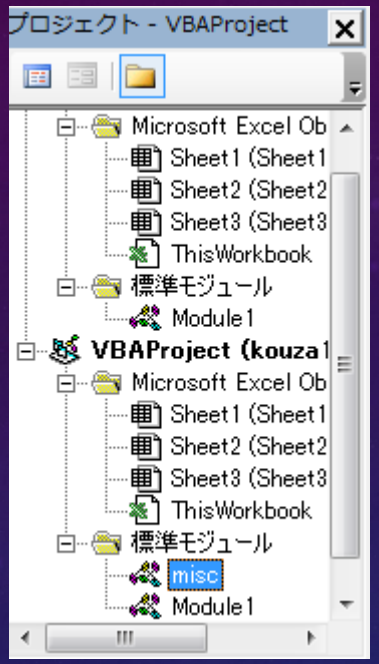
・次は、緯度・経度の数値を普通の10進法に変換
　する関数を作ります。

・今まで数値といえば整数のLong型でしたが、
　実数（小数点が入った数値）を扱うには、
　Single型（単精度浮動小数点型）と、
　Double型（倍精度浮動小数点型）があります。
　今回は、高精度なDouble型を使います。

```
Sub kadai4()
    Dim fn As String, buf As String
    Dim c As Long
    Dim spBuf As Variant

    '使用中のセルをすべて消去する
    ClearCells 2, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダをカレントフォルダにして、
    'ユーザーにファイルを選択させる
    ChDrive ThisWorkbook.Path
    ChDir ThisWorkbook.Path
    fn = Application.GetOpenFilename

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 2
    Do Until EOF(1)
        Line Input #1, buf  '1行読み込む
        If Left(buf, 1) <> "#" Then

            'Variant型の配列を作り、1行をカンマで
            '区切って分割して入れる
            spBuf = Split(buf, ",")

            '行頭に"$GPGGA"があったら、データをセルに入れる
            If spBuf(0) = "$GPGGA" Then
                If spBuf(2) <> "" Then
                    Cells(c, 1) = spBuf(1)  '時刻
                    Cells(c, 2) = spBuf(2)  '緯度
                    Cells(c, 3) = spBuf(3)  'N:北緯  S:南緯
                    Cells(c, 4) = spBuf(4)  '経度
                    Cells(c, 5) = spBuf(5)  'E:東経  W:西経
                    c = c + 1
                End If
            End If

        End If
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

緯度・経度の数値を変換する関数

・モジュール"misc"（課題1の後で追加したモジュール）を開き、




```
'100倍され十の位以下が60進法になった数値を、完全な10進法で元の数値に変換する
Function convDMMtoDeg(src As String) As Double
    Dim tmps As Double, pu As Long, pd As Double

    'データがないときは0を返す
    If src = "" Then
        convDMMtoDeg = 0
        Exit Function
    End If

    tmps = CDbl(src)     '一旦、倍精度浮動小数点数にする
    pu = Int(tmps / 100)     '100で割って商の整数部分だけを取り出す
    pd = (tmps - pu * 100) / 60 '100で割った余りを60で割る
    convDMMtoDeg = CDbl(pu) + pd

End Function
```

右のコードを追加してください。

・終わったら、Module1のkadai4に戻ってください。

課題4-2：緯度・経度を完全に10進法で出力する

・緯度・経度の部分に、先ほどのconvDMMtoDegを組み込んで、変換後の数値がセルに入るようにしてください。

・ただし、convDMMtoDegの引数はString型なので、型変換関数 CStr([数値])を使ってください。

※Cells()に代入する変数型は、数値でも文字でも大丈夫です。

＜制限時間　2分＞

```
Sub kadai4()
    Dim fn As String, buf As String
    Dim c As Long
    Dim spBuf As Variant

    '使用中のセルをすべて消去する
    ClearCells 2, 1, LastRow, LastColumn

    'xlsmファイルと同じフォルダをカレントフォルダにして、
    'ユーザーにファイルを選択させる
    ChDrive ThisWorkbook.Path
    ChDir ThisWorkbook.Path
    fn = Application.GetOpenFilename

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Input As #1

    'ファイルの終端に達するまで繰り返す
    c = 2
    Do Until EOF(1)
        Line Input #1, buf  '1行読み込む
        If Left(buf, 1) <> "#" Then

            'Variant型の配列を作り、1行をカンマで
            '区切って分割して入れる
            spBuf = Split(buf, ",")

            '行頭に"$GPGGA"があったら、データをセルに入れる
            If spBuf(0) = "$GPGGA" Then
                If spBuf(2) <> "" Then
                    Cells(c, 1) = spBuf(1)  '時刻
                    Cells(c, 2) = spBuf(2)  '緯度
                    Cells(c, 3) = spBuf(3)  'N:北緯  S:南緯
                    Cells(c, 4) = spBuf(4)  '経度
                    Cells(c, 5) = spBuf(5)  'E:東経  W:西経
                    c = c + 1
                End If
            End If

        End If
    Loop

    'ファイルを閉じる
    Close #1

End Sub
```

課題4-2の回答

・GGAデータの処理部分を、右のように変えれば
　OKです。

・その結果、右のような出力が得られます。

```
        '行頭に"$GPGGA"があったら、データをセルに入れる
        If spBuf(0) = "$GPGGA" Then
            If spBuf(2) <> "" Then
                Cells(c, 1) = spBuf(1)   '時刻
                Cells(c, 2) = convDMMtoDeg(CStr(spBuf(2)))   '緯度
                Cells(c, 3) = spBuf(3)   'N:北緯  S:南緯
                Cells(c, 4) = convDMMtoDeg(CStr(spBuf(4)))   '経度
                Cells(c, 5) = spBuf(5)   'E:東経  W:西経
                c = c + 1
            End If
        End If
```

	時刻	緯度	N/S	経度	E/W
2	62440.02	35.6917	N	139.7535	E
3	62441.02	35.6917	N	139.7534	E
4	62442.02	35.69169	N	139.7534	E
5	62443.02	35.6917	N	139.7534	E
6	62444.02	35.6917	N	139.7534	E
7	62445.02	35.69166	N	139.7534	E
8	62446.02	35.69166	N	139.7534	E

テスト

4　テキストファイルの書き込み

・テキストファイルの読み込みとほぼ同じやり方で、テキストファイルの書き込みができます。

・定められたフォーマットでテキストファイルを作ると、そのフォーマットに対応したソフトに読み込ませることができます。

・Excelのセルデータから、目的とするフォーマットに適した文字列を作る必要があります。

・どのようなことが必要になるのか、kmlファイルを例として、学びましょう。

例題6:ファイルへの緯度・経度の書き込み

・まずはGPSデータが出力されたセルから、単純に数値を
ファイルに書き込むプログラムを作ります。

・右のようなコードを追加してください。ファイルの読み込みと
書き込みの手順はあまり変わりません。InputをOutputに、
Line InputをPrintにするというのがポイントです。

・実行し、出力されたファイル(sampleGPSdata.kml)を、メモ帳で
開いて確認してください。

・このコードから、緯度が南緯の場合は緯度を−(マイナス)に、
経度が西経の場合は経度を−にする部分を見つけて、
どのように動作するか、デバッグ機能で確認してください。

```
Sub test6()
    Dim fn As String, buf As String, i As Long

    ChDrive ThisWorkbook.Path
    ChDir ThisWorkbook.Path
    fn = ThisWorkbook.Path + "¥sampleGPSdata.kml"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Output As #1

    'データの終端に達するまで繰り返す
    For i = 2 To LastRow
        '経度
        If Cells(i, 5) = "E" Then
            buf = CStr(Cells(i, 4))
        Else
            buf = CStr(-Cells(i, 4))
        End If
        buf = buf + ","
        '緯度
        If Cells(i, 3) = "N" Then
            buf = buf + CStr(Cells(i, 2))
        Else
            buf = buf + CStr(-Cells(i, 2))
        End If
        Print #1, buf
    Next

    'ファイルを閉じる
    Close #1

End Sub
```

kmlファイルのフォーマット

・下のような書式で、緯度・経度の情報を書き連ねたテキストファイルはkmlファイルと呼ばれ、
Google Earthなどのソフトで読み込むことができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Document><name>test</name>
<Style id="style1"><LineStyle><color>FFFF0000</color><width>2</width></LineStyle></Style>
<Placemark><name>test</name><styleUrl>#style1</styleUrl><MultiGeometry><LineString><coordinates>
[経度],[緯度],[高度]
[経度],[緯度],[高度]
[経度],[緯度],[高度]
.
.
.
</coordinates></LineString></MultiGeometry></Placemark></Document></kml>
```

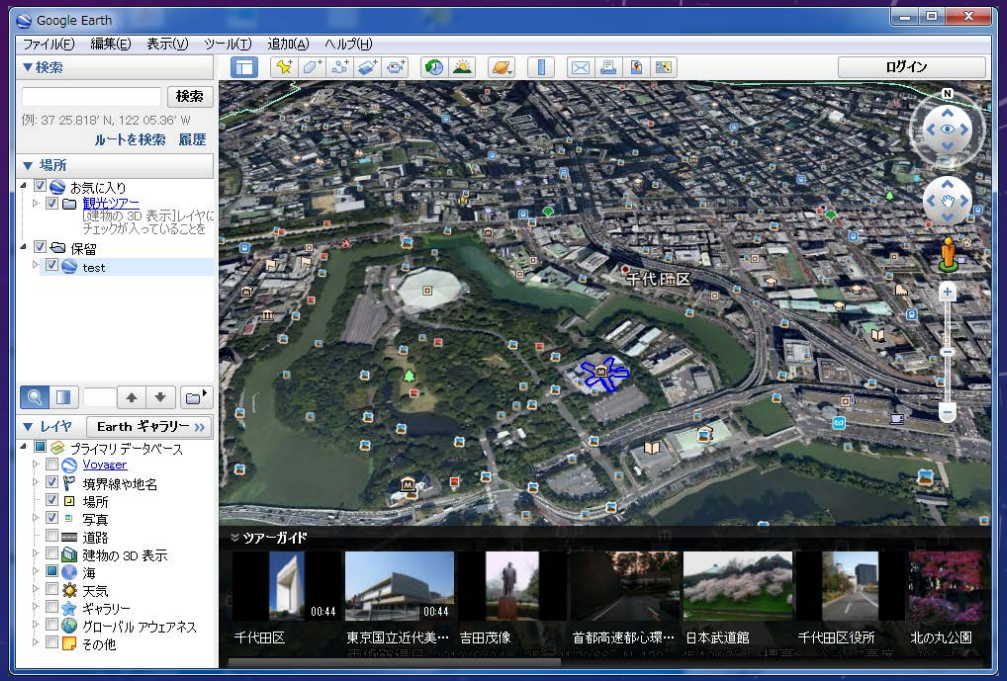
例題7：kmlファイルの出力

・test6をコピペしてtest7を作り、下記のコードを追加してください。（変更部分は緑の枠内です）

```
Sub test7()
    Dim fn As String, buf As String, i As Long

    ChDrive ThisWorkbook.Path
    ChDir ThisWorkbook.Path
    fn = ThisWorkbook.Path + "¥sampleGPSdata.kml"

    'ファイルを1番に割り当てて開く（後で必ず閉じる）
    Open fn For Output As #1

    'kmlファイルの最初の部分を出力
    Print #1, "<?xml version=" + Chr(34) + "1.0" + Chr(34) + " encoding=" + Chr(34) + "UTF-8" + Chr(34) + "?>"
    Print #1, "<kml xmlns=" + Chr(34) + "http://earth.google.com/kml/2.2" + Chr(34) + ">"
    Print #1, "<Document><name>test</name>"
    Print #1, "<Style id=" + Chr(34) + "roadStyle" + Chr(34) + "><LineStyle><color>FFFF0000</color><width>2</width></LineStyle></Style>"
    Print #1, "<Placemark><name>test</name><styleUrl>#roadStyle</styleUrl><MultiGeometry><LineString><coordinates>"

    'データの終端に達するまで繰り返す
    For i = 2 To LastRow
        '経度
        If Cells(i, 5) = "E" Then
            buf = CStr(Cells(i, 4))
        Else
            buf = CStr(-Cells(i, 4))
        End If
        buf = buf + ","
        '緯度
        If Cells(i, 3) = "N" Then
            buf = buf + CStr(Cells(i, 2))
        Else
            buf = buf + CStr(-Cells(i, 2))
        End If
        Print #1, buf + ",0"    '仮の高度(0m)を最後に追加する
    Next

    'kmlファイルの最後の部分を出力
    Print #1, "</coordinates></LineString></MultiGeometry></Placemark></Document></kml>"

    'ファイルを閉じる
    Close #1

End Sub
```

例題7：kmlファイルの出力

・GoogleEarthがインストールされているPCの場合、できたファイル(sampleGPSdata.kml)をエクスプローラ上でダブルクリックすると、自動的に内容が表示されるので、やってみてください。



・「建物の3D表示」のチェックが入っていると、拡大時に軌跡が見えにくくなるので外してみてください。

・エラーが出る場合は、kmlファイルのフォーマットに正しくしたがっているかどうかを確認し、プログラムに必要な修正をして再実行してください。

まとめ

・今回は、Excel VBAでのファイルの読み込み・書き込みを扱いました。
この技術はさまざまに応用できますが、特に研究・開発の現場においては、

「他のソフトや装置からの出力データをExcelで使えるようにできる」

「Excelのデータを他のソフトに読み込ませることができる」

ということが大きな意味を持ちます。

・その意味では、Excelの機能について深く知ることも重要です。
Excelには、科学技術計算に役立つ数学関数が多数内蔵されています。
条件判断を行う関数もあります。（あるいは、作れます）
また、それらをグラフ化する機能も多彩で、CanSatのデータを処理するとき、大いに役立つ可能性があります。ぜひ、Excel自体についても学んでみてください。

＜おわり＞