7つの約束と7つ命令で本格プログラミングを
-Excelで本物っぽいプログラミングに挑戦-

今回は本格的なプログラミングに挑戦していきます。

多くのルールを守って野球をやれる人ならば、7つの約束と7つの命令しかないプログラムは簡単に組めるはずです。

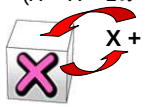
プログラムの7つの約束 (Excel VBScript)

(1)プログラムは「Sub プログラム名」から始まり「End Sub」で終わる。
(2) 命令は行の上から順番に実行される。
(3) プログラムの四則演算(＋－×÷)には+-*/の記号を使う。計算の順番は通常と同様乗算除算が加算減算より先に行われる。カッコ()で計算の順番を変えることができる。
(4) プログラムの中で一時的に値を記録できる変数を定義できる。変数名はアルファベットから始まる文字で定義する。また変数は数式の中でも使用できる。X*5+1など。
(5) プログラムで使う=は、右辺の計算結果を左辺の変数に入れる意味を持っている。
(6) 変数は配列としても定義できる。Dim 変数名(配列の数)
(7) 文字は""で囲んで定義する。 “Hello”など

X = 1

Xと名前をつけた箱(変数)に1を入れる

X = X +1
(X←X + 1のイメージ)



1.初めにXの箱(変数)の中を取り出し+1する。
2. 1.の計算結果をXの箱(変数)に入れなおす。

約束の(1)-(3)はあまり難しくないかと思います。(6)は難しいので、今回の授業ではやりません。
(4)と(5)は普通の数学の=とは違う意味なのでイメージを示してみました。

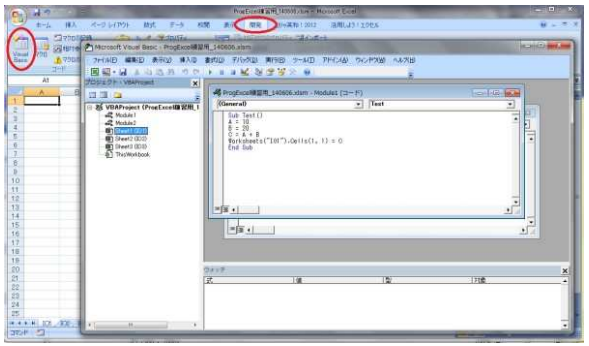
プログラムで使う7つの命令 (Excel VBScript)

(1) **IF..Then**という命令がある。
If 条件 Then
　実効する命令(複数可)
End If　の形式で使い、条件が正しければ命令を実行する。
(2) **If..Then..Else**という命令がある。
If 条件 Then
　実効する命令1(複数可)
Else
　実効する命令2
End Ifの形式で使い、条件が正しければ命令1を、正しくなければ命令2を実行する。
(3) **For..Next**という命令がある。
For 変数 = 初期値　to 終了値
　実行する命令(複数可)
Nextの形式で使い、変数の値を初期値から終了値まで1づつ増やしながら命令を繰り返し実行する。
(4) **Do While ... Loop**という命令がある。
Do While 条件
　実行する命令(複数可)
Loop の形式で使い、条件が正しい間、命令を繰り返して実行する。

(5) 条件は
数式又は変数 比較演算子 数式又は変数
で指定し、比較演算子は**=, <>, >, <, >=,<=**が使用できる。
(6) Excelでは、シート内のセルを
Worksheets("シート名").Cells(行番号,列番号)
の形式で変数と同様に使用できる。
(7) 指定した変数の整数部分の取り出し、Sin, Cosなどを計算するプログラムが**関数**として組み込まれて利用できる。

日本語だと
(2)もし…ならば…他の場合
(3)それぞれ…次に
(4) ..の間実行…繰り返す
という意味になり、すぐにわかるんですが

7つの約束と7つの命令の練習(1) プログラムを作る



プログラムを作る言語はいろいろありますが、授業ではExcelの中で使えるVBSciriptを使います。
ビデオの見ながら、足し算をするプログラムを入力して実行してみてください。
うまくいったら、友達通しでプログラムを意味を考えましょう。(ProgExcel練習用_140620.xlsm)を使います。

```
Sub Test()
A = 10
B = 20
C = A + B
Worksheets("IO1").Cells(1, 1) = C
End Sub
```

初めのプログラム

四則演算

早く終わった人は、ビデオを見ながら四則演算(+-*/)を計算するプログラムを作りましょう

7つの約束と7つの命令の練習(2) データを入力する

```
Sub Test2()
A = 10
B = 20
Worksheets("IO1").Cells(1, 1) = A + B
Worksheets("IO1").Cells(2, 1) = A - B
Worksheets("IO1").Cells(3, 1) = A * B
Worksheets("IO1").Cells(4, 1) = A / B
End Sub
```

```
Sub Test3()
A = Worksheets("IO1").Cells(1, 1)
B = Worksheets("IO1").Cells(1, 2)
Worksheets("IO1").Cells(1, 1) = A + B
Worksheets("IO1").Cells(2, 1) = A - B
Worksheets("IO1").Cells(3, 1) = A * B
Worksheets("IO1").Cells(4, 1) = A / B
End Sub
```

Module3

	A	B
1	80	70
2	60	10
3	700	
4	7	
5		

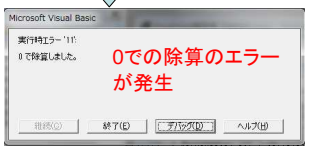
← A = Worksheets("IO1").Cells(1, 1)
← B = Worksheets("IO1").Cells(1, 2)

左側は四則演算を計算するプログラムです。この場合、計算したい数値はプログラムの中で定義しているので、別の数値を計算したい時は、プログラムの中の数字を変更する必要があります。
右側のプログラムはこの点を改良したものです。AとBに入れる数値はExcelのシートから取り出しています。シートの値を変更することによって、プログラムを変更することなく、いろいろな数値の計算ができます。

7つの約束と7つの命令の練習(3) If… Then…

	A	B
1	70	70
2	70	0
3	0	
4		
5		

← Bに0が入るようにシートに入力



改良

```
Sub Test4()
A = Worksheets("IO1").Cells(1, 2)
B = Worksheets("IO1").Cells(2, 2)
Worksheets("IO1").Cells(1, 1) = A + B
Worksheets("IO1").Cells(2, 1) = A - B
Worksheets("IO1").Cells(3, 1) = A * B
If B <> 0 Then
  Worksheets("IO1").Cells(4, 1) = A / B
End If

End Sub
```

Module4

前スライドのプログラムでBに0が設定させるようにExcelのシートに入力すると、実行時にエラーが発生します。これは、0での割り算が起きるためです。
これをIF命令を使って、0以外の時に割り算を実行するように変更すると、このエラーを回避することができます。

○ 比較演算子 △ の条件を指定。
○ = △ (等しい場合は)
○ <> △ (等しくない場合)
○ > △ (○が大きい)
○ < △ (○が小さい:未満)
○ >= △ (○が大きいか等しい:以上)
○ <= △ (○が小さいか等しい:以下)

7つの約束と7つの命令の練習(4) If… Then…Else

```
Sub Test5()
A = Worksheets("IO1").Cells(1, 2)
B = Worksheets("IO1").Cells(2, 2)
Worksheets("IO1").Cells(1, 1) = A + B
Worksheets("IO1").Cells(2, 1) = A - B
Worksheets("IO1").Cells(3, 1) = A * B
If B = 0 Then
  Worksheets("IO1").Cells(4, 1) = "0での割り算"
Else
  Worksheets("IO1").Cells(4, 1) = A / B
End If
End Sub
```

Module5

	A	B
1	60	40
2	20	20
3	800	
4	2	
5		

	A	B
1	70	70
2	70	0
3	0	
4	0での割り算	
5		

← 0での割り算の指定

前のプログラムでは0での割り算が発生した時は、特に何も表示されません。IF.. Then…Else命令を使うと、0の時も、対応したメッセージを表示することができるようになります。

7つの約束と7つの命令の練習(5) For… Next

```
Sub Next1()
S = 0
For A = 1 To 10
 S = S + A
Next
Worksheets("IO1").Cells(1, 1) = S
End Sub
```

Module6

	A
1	55
2	

← 実行結果

```
Sub Next2()
S = 0
For A = 1 To 10
 Worksheets("IO1").Cells(A, 1) = A
 S = S + A
Next
Worksheets("IO1").Cells(A, 1) = S
End Sub
```

Module7

	A
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	55
12	

← 実行結果

For.. Next命令を使うと、同じ処理を繰り返して実行することができます。Module7でわかるように、Aという変数が1づつ増加して10まで、結局10回繰り返されます。

フローチャートの書き方

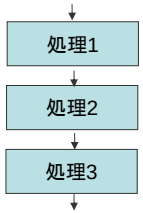
プログラムの開始と終わり

(開始)

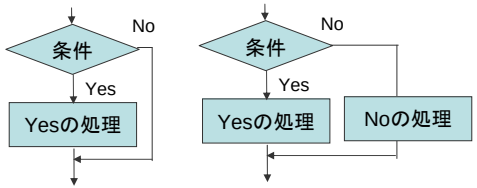
(終了)

処理の流れの表記(アルゴリズム)

逐次型(直線型)

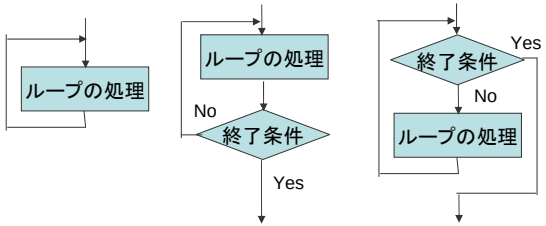


分岐型



プログラムや人間の判断などのアルゴリズムは基本的に、逐次型、分岐型、ループ型の組み合わせで表現できますね。

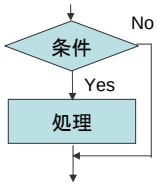
ループ型(繰返し型)



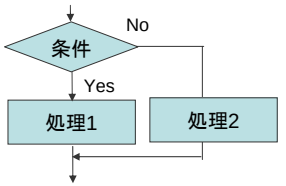
無限ループ　後判断ループ　前判断ループ

フローチャートの書き方と命令の関係

分岐型

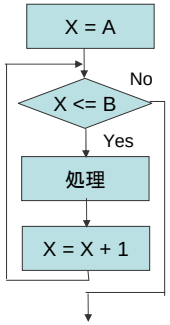


IF 条件 Then
処理
End If



IF 条件 Then
処理1
Else
処理2
End If

ループ型(繰返し型)



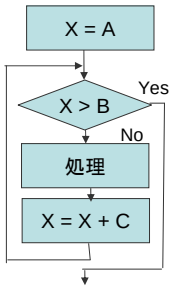
For 変数(X) = 初期値(A) To 終了値(B)
処理
Next

フローチャートと今回勉強している命令の関係を示しました。
For.. Nextは少し複雑ですね。

演習: 10以下の偶数の合計を表示するプログラム

For.. Next命令の例で1から10までの合計を計算するプログラムを示しました。こんどは2+4+6..+10のように10以下の偶数の合計を表示するプログラムを作成してください。プログラムは一つだけではなく、いろいろな方法で表示するプログラムを複数開発してみましょう。(下記の内容も参考にしてください)

参考1: For..Nextのより詳細の指定



For 変数(X) = 初期値(A) To 終了値(B) Step 増分(C)
処理
Next

参考2: 整数部分を取り出すINT関数

A = 1.5
B = INT(A)　　<= Bには1が入る

演習: 10以下の偶数の合計を表示するプログラム例

```
Sub SUME1()
Worksheets("IO1").Cells(1, 1) = 30
End Sub
```

```
Sub SUME2()
Worksheets("IO1").Cells(1, 1) = 2 + 4 + 6 + 8 + 10
End Sub
```

```
Sub SUME3()
S = 0
For A = 2 To 10 Step 2
 S = S + A
Next
Worksheets("IO1").Cells(1, 1) = S
End Sub
```

```
Sub SUME4()
S = 0
For A = 1 To 5
 S = S + A * 2
Next
Worksheets("IO1").Cells(1, 1) = S
End Sub
```

```
Sub SUME5()
S = 0
X = 0
For A = 1 To 5
 X = X + 2
 S = S + X
Next
Worksheets("IO1").Cells(1, 1) = S
End Sub
```

```
Sub SUME6()
S = 0
For A = 1 To 10
   If A = Int(A / 2) * 2 Then
      S = S + A
   End If
Next
Worksheets("IO1").Cells(1, 1) = S
End Sub
```

いくつ作ることができましたか?、いんちきみたいなものもありますが「合計を表示する」という意味で、すべて正解です。どれが一番良いかは、それを利用する場面によって判断されます。

少し実用的なプログラム(1) 合計を計算する

```
Sub APP1()
S = 0
For A = 1 To 10
 X = Worksheets("IO1").Cells(A, 1)
 S = S + X
Next
Worksheets("IO1").Cells(1, 2) = S
End Sub
```

	A	B
1	78	788
2	90	
3	100	
4	87	
5	34	
6	69	
7	83	
8	93	
9	72	
10	82	
11		

← 合計

← 合計を計算する10個の値

Module8

前のFor.. Nextのプログラムを少し改良してみました。Excelのシートに入力した値の合計を求めるプログラムです。
なんとなく実用的なプログラムですね。

7つの約束と7つの命令の練習(5) Do… Loop 値の数がわからない合計を計算する

```
Sub APP2()
S = 0
A = 1
X = Worksheets("IO1").Cells(A, 1)
Do While X <> 999
 S = S + X
 A = A + 1
 X = Worksheets("IO1").Cells(A, 1)
Loop
Worksheets("IO1").Cells(1, 2) = S
End Sub
```

	A	B
1	78	458
2	90	
3	100	
4	87	
5	34	
6	69	
7	999	
8		

← 合計

← 合計を計算する値

← 終了を判断する値

Module9

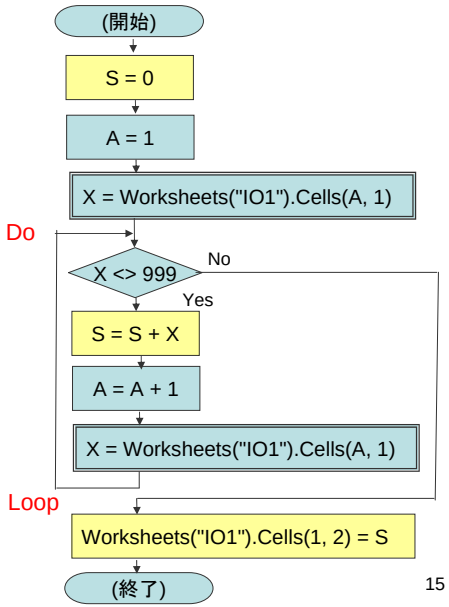
合計する値の数がわからない場合があります。このような時、最後に計算を終了するような目印をつけて、そこまで計算するというようなプログラムを作ることができます。
例えば、テストの点数だと100点までなので、999を終了の目印にすることができます。
そしてDo While … Loop命令を使うと、この終了の値を上手に判断することができます。

Do… Loop 値の数がわからない合計を計算する。のフローチャート

```
Sub APP2()
S = 0
A = 1
X = Worksheets("IO1").Cells(A, 1)
Do While X <> 999
 S = S + X
 A = A + 1
 X = Worksheets("IO1").Cells(A, 1)
Loop
Worksheets("IO1").Cells(1, 2) = S
End Sub
```

Module9

Do.. Loop命令に入る前と、Do..Loop命令の最後で、条件を確認するXの値を入れ替えることがポイントです。



少し実用的なプログラム(2) 九九の表を作る

```
Sub APP99()
For I = 1 To 9
  For J = 1 To 9
    Worksheets("IO1").Cells(I, J) = I * J
  Next
Next
End Sub
```

	A	B	C	D	E	F	G	H	I
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Module10

For.. Next命令の中にさらにFor..Next命令を入れることもできます。こんな簡単なプログラムでも九九の表をつくることができます。
このような繰り返しのプログラムはコンピュータの特異なことですね。

少し実用的なプログラム(3) 正三角形の判断

```
Sub APPT()
A = Worksheets("IO1").Cells(1, 1)
B = Worksheets("IO1").Cells(2, 1)
C = Worksheets("IO1").Cells(3, 1)
If A = B Then
  If B = C Then
    Worksheets("IO1").Cells(4, 1) = "正三角形です"
  Else
    Worksheets("IO1").Cells(4, 1) = "正三角形以外の値です"
  End If
Else
    Worksheets("IO1").Cells(4, 1) = "正三角形以外の値です"
End If
End Sub
```

Module11

	A	B	C
1	20		
2	20		
3	20		
4	正三角形です		

	A	B	C
1	20		
2	20		
3	10		
4	正三角形以外の値です		

If… End If命令の中にIf End Ifを入れることもできます。こうすることにより複雑な判断ができるようになります。
　またIf命令の中にFor..Next命令、For..Next命令の中にIf命令を入れて組み合わせて使うこともちろんできます。

演習: 3つの値でできる三角形の判断

演習A 正三角形、二等辺三角形を判断する。
　前のプログラムをもとに、3つの値を設定して
・正三角形です。
・二等辺三角形です。
・他の三角形の組み合わせです。
と判断・表示するプログラムを作成します。

演習B 三角形ができない場合も判断する。
　3つの値には0以上の整数(0, 1, 2, 3, …)を指定することとして、
・正三角形です。
・二等辺三角形です。
・不等辺三角形です。
・三角形ができません。
・辺として不正な長さが指定されています。
と判断・表示するプログラムを作成します。
例えば、どれか一つでも0が指定されていたら、最後のメッセージを表示します。また、2,3,6の組み合わせでは三角形ができません。

原理的に、今回勉強したプログラムの範囲で、数値を扱う、どんなプログラムも作成することができます。