# はじめに

## プログラミングとは

　プログラミングをマスターすることは「魔法の杖」を手にすることに例えることができます。プログラムを組んでコンピューターに実行させると、コンピューターは命令された通りのことを超高速で実行してくれます。人間が一生かかってもできない計算を、コンピューターが1秒以内で終わらせることも多いでしょう。

　ですから、プログラミングをマスターすることで、仕事の能率が飛躍的にアップする可能性があります。同僚が何日もかかって行っている仕事を、あなたはプログラムを組むことにより一瞬で終わらせ（プログラムを組むための時間は必要ですが）、差をつけることができます。

　プログラミングは素晴らしい威力がありますが、コンピューターは愚直な道具です。気を利かせて仕事をやってくれるということは絶対にありません。プログラムに1文字でも間違いがあれば、全く動かない場合がほとんどです。そして、コンピューターはあなたが思った通りではなく、プログラムに書かれた通りに実行します。思い通りにならないときは、プログラムが間違っています。プログラミングをはじめたての頃は、思い通りにプログラムが動かないことに悩むこともあるでしょう。

　プログラミングは人を夢中にさせる魔力があります。LinuxというOSやGimpなどのアプリは無料で使えます。これはプログラミングをすることに喜びを感じる人がたくさんいることを意味します。皆さんは、プログラムに喜びを感じることができるでしょうか？　そうなったら、教える側としてこれ以上の喜びはありません。

　この授業では、教師に役立ちそうな例題を選んでプログラミングを行います。プログラミング環境としてExcel VBAを用います。

## Excel VBAとは

　Excelは表計算のアプリであり、成績処理など教師の仕事になくてはならない道具です。Excelの数式や関数（平均、合計、標準偏差、順位づけなど成績関係の関数が多数あります）を使いこなすだけでも非常に便利ですが、プログラミングを覚えると、さらに無敵です。Excelはプログラミングをして操作することができます。Excelで使えるプログラミング言語はExcel VBAと呼ばれる言語です。VBAはVisual Basic Applicationの略で、Basic系の言語です。プログラミング言語には色々なものがありますが、1つをマスターすると、他の言語は表現の仕方が多少変わるだけで、本質は同じですから、他の言語をマスターするのは簡単です。Basicは分かりやすい言語ですから、Excel VBAはもっとも実用的で易しいプログラミング言語と言えましょう（ここは突っ込みどころ満点ですが、こう思い込んで次に進んで下さい）。

## 準備

　プログラミングができる状態にしましょう。

　「ファイル」→「オプション」→「リボンのユーザー設定」を押して下さい。右の「開発」のところにチェックを入れて「OK」を押して下さい。これで「開発」というリボンが表示されます。

　「開発」→「コード：Visual Basic」を押して下さい。プログラミング用のウィンドウが生成されます。まずは、眺めてみましょう。左端に「プロジェクト-VBAProject」というウィンドウがあります。閉じてしまったときは「表示」→「プロジェクトエクスプローラ」で復活します。左下に「プロパティ-Sheet1」というウィンドウがあります。このウィンドウはほとんど使わないので閉じてしまっても構いません。下端に「イミディエイト」というウィンドウがあります。これはよく使います。閉じてしまったときは「表示」→「イミディエイトウィンドウ」で復活します。

　プログラムは「標準モジュール」という場所の中に書きます。「挿入」→「標準モジュール」と操作して、標準モジュールを1つ挿入して下さい。「Module 1」という名前がついたと思います。もう一つ挿入すると「Module 2」という名前がつきます。挿入してみて下さい。この中にプログラムを書きます。

　モジュールの切り替えはモジュール名の上でダブルクリックです。シングルクリックでは切り替わりません。このインターフェースは1995年頃から変わっていない古いインターフェースであり、現在の感覚とは異なる点があります。

# 基本命令

## セルにものを入れる

　シート「練習用」を選択して下さい。

　まずは、セルの中に何か入れることから始めましょう。セルA1に数値12、セルB1に文字列abcを入れるプログラムは以下のように書きます。= は右辺のものを左辺に代入します。Module 1の中に書いて下さい。

Sub rei1()

Range("A1") = 12

Range("A2") = "abc"

End Sub

　プログラムは「Sub プログラム名()」という記述ではじめます。このように記述すると、自動的にEnd Subという文字列が入ります。SubからEnd Subまでが1つのプログラムです。rei1 というのはプログラムの名前です。プログラムの名前のつけ方は以下の通りです。

* 使える文字はA～Z, a～z, 数字, \_
* 数字からはじめてはいけない
* 大文字小文字は区別されない

　1つのモジュールに複数のプログラムを書くことができます。ただし、同じ名前のプログラム名をつけることはできません（つけるとエラーが発生します）。

字下げ

　Rangeを含む2行は字下げしています。tabキーを押すと、1ランク字下げします。こうすることで、プログラムが見やすくなります。字下げをしなくてもプログラムは動作しますが、プログラムを見やすくするため、字下げの習慣をつけて下さい。

　このプログラムはセルA1に数値12を入れ、セルA2に文字列 abc を入れます。

　セルの中に入るものは「数値」か「文字列」です。文字列の場合は "abc" のように、ダブルクオーテーションで囲みます。

　「実行」→「サブ／ユーザーフォームの実行」として、プログラムを実行して下さい。セルA1とA2にそれぞれ数値12と文字列abcが入ったことを確認してください。

数値と文字列の区別

　文字列は "abc" のようにダブルクオーテーションで囲む必要があります。

　次のようにプログラムを組むとどうなるでしょう？

Range("A3") = abc

　セルA3には何も入りません。これは何が起こったかというと、セルA3に変数abcの内容を入れています（変数については後ほど詳しく説明します。セルと同じように文字列や数値を入れる箱と考えて下さい）。変数abcの初期値は何も入っていない状態なので、A3には何も入りません。

abc = "yabu"

Range("A3") = abc

とすると、セルA3には yabu という文字が入ります。

エラーへの対処

　次のようにプログラムを変更して下さい。

Renge("A2") = "abc"

　Rangeと書くところ、Rengeと綴りを間違っています。実行すると、エラーが出て中断します。コンピューターは1文字でも間違いがあると、エラーを出して実行してくれません。「実行」→「リセット」でプログラムを実行するモードから抜けた後、誤りを修正して下さい。

　上記のプログラムではセルを指定するのにRange("A1")のような形式を使いました。もう一つの形式があります。

Sub rei2()

Cells(4, 2) = 23

Cells(4, 3) = "def"

End Sub

　Cells(4, 2)は4行目2列目（B列）を表します。この形式は、4, 2といった数値の部分を変数で表すときに便利な形式です。

　列は次のように、数値ではなく文字を使うこともできます。

Sub rei3()

Cells(4, "C") = 23

Cells(4, "D") = "def"

End Sub

## セルの値を読み出す

　これ以降は先頭の「Sub プログラム名()」と末尾の「End Sub」は省略します。まずは、次のプログラムを見て下さい。

Range("B1") = Range("A1")

　セルB1にセルA1の内容を代入しています。

## 変数とは

　前節の例では直接代入しましたが、次のように書くこともできます。

a = Range("A1")

Range("B1") = a

　ここで登場したaを変数といいます。セルと同じように「数値」か「文字列」を入れることができます。

　変数名のつけ方はプログラム名と同じです。

　プログラム名と同じ変数名をつけてはいけません。

　この例では変数を使う意味がほとんどありませんが、aを何回も使う場合やaの値に何らかの処理を加えたい場合などは、このように一旦変数にコピーしてから処理を記述する方がプログラムが見やすくなります。変数の威力は後の節で説明します。

## イミディエイトウィンドウ

　Visual Basicのエディタのウィンドウの下方に「イミディエイト」という名前のウィンドウがあります。ない場合は「表示」→「イミディエイト」で表示させて下さい。この領域は2つの働きがあります。

* この中で単発の命令を実行することができる。
* この中に表示することができる

　イミディエイトウィンドウの中でRange("A1") = "kyokyo" と打ってEnterキーを押しましょう。セルA1に文字列 "kyokyo" が入りました。このように、イミディエイトウィンドウの中で単発の命令を実行することができます。

　次のプログラムを実行しましょう。

a = 10

Debug.Print "a = " & a

　Debug.Printの後に続くものがイミディエイトウィンドウに表示されます。& は文字列を接続します。この例のaのように数値がある場合、文字列に変換してから接続します。

　プログラムの実行の途中で、変数の内容を確認したいときは、イミディエイトウィンドウに出力することで確認することができます。

## プログラムの保存

　これまでに書いたプログラムを保存しましょう。「ファイル」→「名前を付けて保存」とすると、図2.1のダイアログが表示されます。

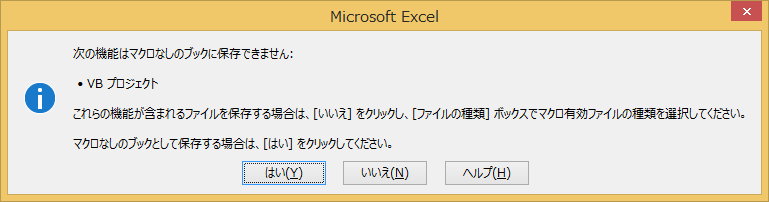


図2.1　保存時の警告

　ここで「はい」を押してはいけません。プログラムは保存されず、Excelのシートのみが保存されます。「名前を付けて保存」のダイアログの「ファイルの種類」で「Excelマクロ有効ブック \*.xlsm」を選びます。

　Visual Studio Basic for Applicationsの左上に「プロジェクト - VBAProjct」というタイトルのウィンドウがあります。ない場合は「表示」→「プロジェクトエクスプローラ」で表示して下さい。

　ここは1つのファイルを構成する要素が示されています。1つのExcelファイルはいくつかのシートと、標準モジュール（存在しない場合もある）から構成されます。

## セルを変数で指定する

　セルC1～C5に数値12を入れるプログラムは以下のようになります。

Range("C1") = 12

Range("C2") = 12

Range("C3") = 12

Range("C4") = 12

Range("C5") = 12

　数字の部分を変数で書くことができます。

i = 1

Range("C" & i) = 12

i = 2

Range("C" & i) = 12

i = 3

Range("C" & i) = 12

i = 4

Range("C" & i) = 12

i = 5

Range("C" & i) = 12

　ここではRange("C" & i) という形式を使いました。& は先ほど出てきましたが、文字列を接続する命令です。数値を書くと自動的に文字列に変換してから接続します。

　次のように書くこともできます。

Range("C" + CStr(i))

　+ と & は働きが少し異なります。+ は数値同士の場合は足し算をし、文字列同士の場合は接続します。文字列 "C" と数値 i を + で結ぶとエラーが起こります。CStr() は括弧の中の数値を文字列に変える命令であり、文字列同士にすることで、エラーを起こさずに接続しています。& は文字列を接続しますが、数値がある場合は文字列に変換してくれます。従って、+ を使うより、& を使った方が簡潔に書けるので、本書では & を使います。

## For文

　次のプログラムを実行してみてください。

For i = 1 To 5

Debug.Print i

Next i

　イミディエイトウィンドウに

1

2

3

4

5

と表示されたと思います。ForとNextは常にペアとなって用いられます。

　For ～ Nextはこの区間の繰り返しを行う命令です。i = 1 To 5と書くと、i = 1, 2, 3, 4, 5と1ずつ増やして5回繰り返します。1が最初の値、5が最後の値です。ループと呼びます。この例から分かるように、Forの後は字下げを行い、ループの範囲を明確にして下さい。

For i = 1 To 5 Step 2

と書くと、iの値を2刻みに増やします。すなわち、i = 1, 3, 5と3回ループを回ります。この場合、i = 1 To 6 Step 2と書いても同じ結果となります。

For i = 5 To 1 Step -1

なら、iの値は5, 4, 3, 2, 1と変化します。

　先ほどセルC1 ～ C5に12を入れるプログラムを書きました。10行ありましたが、このプログラムは以下のように書き直すことができます。

For i = 1 To 5

Range("C" & i) = 12

Next i

　Range("C1") とする代わりに、Range("C" & i) としています。先ほど述べたように & は文字列を接続する命令です。数値の場合は、文字列に自動変換します。

　iは1, 2, 3, 4, 5と変化してFor～Nextの間を5回繰り返すので、"C" & iの部分は "C1", "C2", "C3", "C4", "C5" となります。

　ここではForループ用の変数としてiを使いました。プログラミングの世界ではループ用の変数として、i, j, k, l, m, nの順番に使うのが慣習となっています (初期のプログラミング言語であるFortranではI, J, K, L, M, Nからはじまる変数は整数型変数と決められており、ループ変数にはI～Nを使いました。その名残と思われます)。それ以外の変数名を使っても、プログラムは動作しますが、他の人が読みづらくなってしまうので、皆さんも慣習に従って、i, j, k,... の順番で使うようにして下さい。

【For課題1】

1. シート「For練習」を使います。セルA1～A10の内容をB1～B10にコピーするプログラムを作成しなさい。
2. セルA1～A10の内容をC10～C1に逆順にコピーするプログラムを作成しなさい。すなわち、C1 ← A10, C2 ← A9, C3 ← A8 のように代入します。

（ヒント）Range("C" & i) ではなく、Range("C "& j) として、j = （iを含む数式） という形の式を書けばよい。

## 変数の使い方

　シート「数列」を使います。

＜例題＞

　セルA1からA2, A3と下方へ向かって順番に1, 3, 5, 7....という数値をA10まで入れなさい。

num = 1

For i = 1 To 10

Range("A" & i) = num

num = num + 2

Next i

　numという名前の変数を導入しています。そして、1, 3, 5, 7, ...と変化させています。ここで重要な表現があります。

num = num + 2

です。この1行はプログラミングの神髄を表しています。= は右辺の値を、左辺の変数（またはセル）に代入することを意味しています。もし、num = 3 の状態で、num = num + 2を実行すると、右辺は3 + 2 → 5になります。これを左辺numに代入するのでnumは5になります。すなわち、numを2増やすことを意味しています。

　私はプログラミングを理解することは、i = i + 1といった表現を理解することだと考えています。変数の値を更新しながら処理を進めるという概念を是非とも理解して下さい。

【For課題2】

　B1～B10に1, 2, 4, 8, 16, .... という数列を入れるプログラムを作成しなさい。

（注意）

　プログラムではかけ算は a \* b, 割り算は a / b で表します。

## デバッグ

　For文を以下のように書き換えて下さい。

For i = 0 To 10

　実行すると、図2.2のようなエラーが発生し、プログラムはストップします。



図2.2　エラー発生時の様子

　「終了」を押すと、プログラムは終了します。

　「デバッグ」をクリックすると、図2.3のようにエラーが発生した行が、黄色反転表示になり、矢印で示されます。

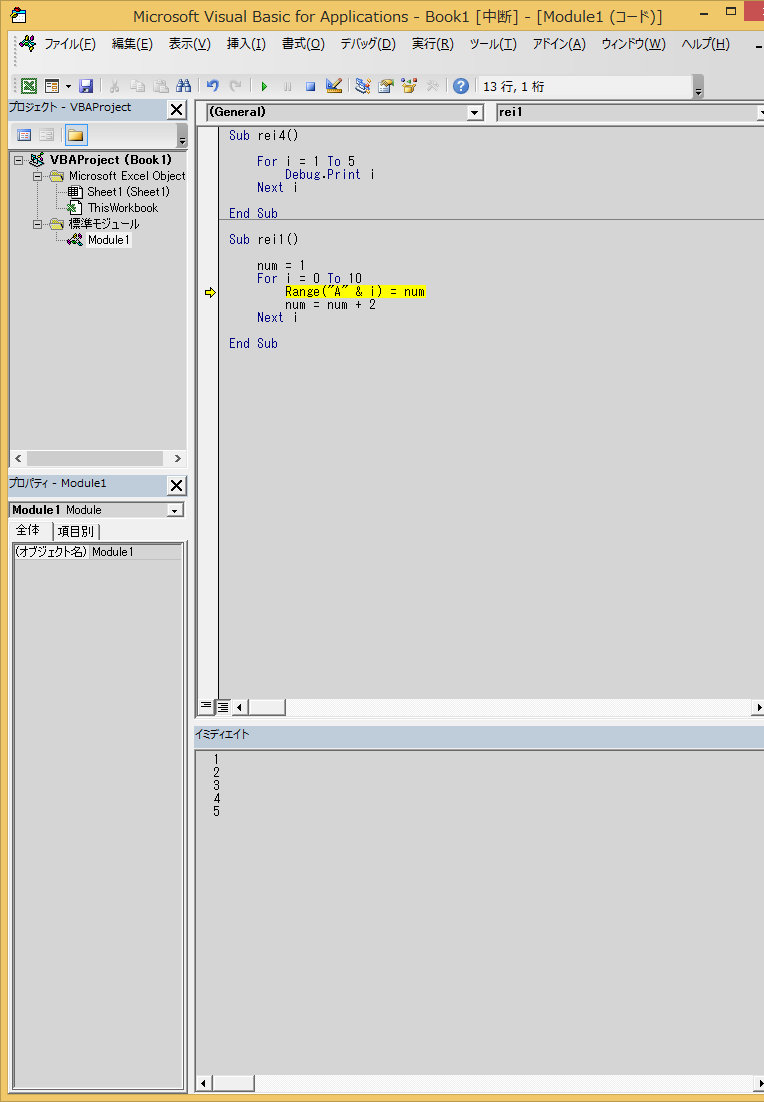


図2.3　「デバッグ」を押したときの状況

　エラーの発生原因を探るため、i の値を表示してみましょう。イミディエイトウィンドウで

Print i

あるいは

? i

と書いてEnterを押します。? は Print の省略形です。i の値として0が得られたと思います。このプログラムのループの1周目はi = 0, num = 1という状態です。セルA0に1を入れようとしますが、A0という名前のセルはないので、エラーが発生しています。このようにプログラムがストップしたときは、イミディエイトウィンドウを使って、変数の値を調べることで解決する場合があります。

　プログラムは中断した状態を継続しています。「実行」→「リセット」で何も実行していない状態に戻ります。

## コメント文と継続行

　プログラムの中に注釈を書くことができます。以下のように、'（アポストロフィー）を書くと、その行の右側は何を書いても無視されます。プログラムが何をしているか分かりやすく解説しておくと、後日プログラムを見直したときや、他の人がプログラムを見るときに役に立ちます。

'

' A1 から A10 まで 10 を入れるプログラム

'

num = 10 ' 空行はどこに入れても良い。プログラムを

' 見やすくするため、適宜空行を入れよう

For i = 1 To 10

Range("A" & i) = num ' コメントは行の途中に書くこともできる

Next i

　これまで学習したプログラムは1行が短いですが、今後1行が長くなるかもしれません。以下のように、\_ (アンダースコア) を書くと、次の行にまたがってプログラムを書くことができます。

Range("A" & i) = \_ ' アンダースコア \_ を入れると、続きを

num ' 次の行に書くことができる。

## If文

　条件が成立するか否かによって、実行する命令を切り替えるのがIf文です。シートを「If練習」に切り替えてください。A列に名前、B列に点数が入っています。C列に加点するか否かが入っています。1が入っている人は加点対象者です。

＜例題＞

　D列に点数を入れなさい。ただし、加点対象者は10点加えます。

For i = 2 To 11

tokuten = Range("B" & i)

katen = Range("C" & i)

If katen = 1 Then

tokuten = tokuten + 10

End If

Range("D" & i) = tokuten

Next i

　データが2～11行目に入っているので、For文の範囲は2 To 11 です。tokuten と katen という変数を導入し、B列とC列の値を代入しています。ここで、If文が出てきました。If文は以下の形式をとります。

If 条件式 Then

Ifが成立したときの処理1

Ifが成立したときの処理2 ' 命令の個数は何個でも良い

Else

Ifが成立しないときの処理1

Ifが成立しないときの処理2

End If

　今回の例題のように、条件が成立しないときの処理が不要な場合、以下のように書きます。

If 条件式 Then

Ifが成立したときの処理1

Ifが成立したときの処理2

End If

　条件が成立しないときのみ処理を行いたいときは、以下のように書きます。

If 条件式 Then

Else

Ifが成立しないときの処理1

Ifが成立しないときの処理2

End If

　条件式は以下のように書きます。文字列の場合は、「等しい」か「等しくない」かです。

a = b　　　　aとbは等しい

a <> b　　　 aとbは等しくない

　数値の場合、以下の6パターンがあります。

a = b　　　　　等しい

a <> b　　　　 等しくない

a < b　　　　　＜

a <= b　　　　 ≦

a > b　　　　　＞

a >= b　　　　 ≧

　>= のように、不等号とイコールを書く場合、不等号を先に書きます。=> と書くと、Visual Basic for Applicationsのエディタが自動的に直してくれます。

　「a > b かつ c > d」のように、2つの条件がどちらも成立するときにThen以下の命令を実行したいときは次のように書きます。

IF a > b And c > d Then

　「5 < a < 10」という条件の場合、そのまま書くとエラーになります。以下のように2つに分解して下さい。

If 5 < a And a < 10 Then

　「a > b または a < c」のように2つの条件のどちらかが成立するときにThen以下の命令を実行したいときは、以下のように書きます。

IF a > b Or c > d Then

　ここで間違えやすいパターンが一つあります。「a = 1またはa = 2」という条件を以下のように書いてはいけません。

If a = 1 Or 2 Then

　これは「a = 1」という条件と「2」という条件のいずれかが成立したときにThen以下を実行することを意味しており、「2」は常に成立する（理由はちょっとややこしいので省略します）ので、このIf文は常に成立します。たとえば、a = 3 として実行すると、Then以下を実行します。

＜例題＞

　E列に最終得点を記入しなさい。原則としてD列の得点を転記するが、100点を超えた場合は、100点にする。

For i = 2 To 11

tensuu = Range("D" & i)

If tensuu > 100 Then ' >= 100 でもよい

tensuu = 100

End If

Range("E" & i) = tensuu

Next i

　tensuuという変数を使うのをやめて以下のように書くこともできますが、私は上のプログラムの方が分かりやすいと思います。

For i = 2 To 11

If Range("D" & i) > 100 Then

Range("E" & i) = 100

Else

Range("E" & i) = Range("D" & i)

End If

Next i

＜例題＞

　F列に合否を記入しなさい。60点以上のとき合格とします。合格のとき "○", 不合格のとき "×” を記入します。同時に、不合格のセルはRGB(255,200,255)の色で塗りつぶし、合格のセルは「塗りつぶしなし」にしなさい。

For i = 2 To 11

tensuu = Range("E" & i)

If tensuu >= 60 Then

Range("F" & i) = "○"

Range("F" & i).Interior.Color = xlNone

Else

Range("F" & i) = "×"

Range("F" & i).Interior.Color = RGB(255, 200, 255)

End If

Next i

　例題の解答から分かるように、セルの塗りつぶしは

Range("A1").Interior.Color = 色の情報

のように指定します。「塗りつぶしなし」のとき、xlNoneを指定し、「塗りつぶしあり」のとき、色をRGB(200,255,255) のようにRGBを0～255の範囲で指定します。

＜例題＞

　E列が80点以上を成績優秀者とする。成績優秀者のリストを作ることにした。成績優秀者の名前のみをA列からH列にコピーしなさい。

For i = 2 To 11

Name = Range("A" & i)

tensuu = Range("E" & i)

If tensuu >= 80 Then ' 80点以上の人は名前を H 列に

Range("H" & j) = Name ' コピーする

End If

Next i

＜例題＞

　成績優秀者をH2, H3, H4,.... のセルに並べなさい。

j = 2

For i = 2 To 11

Name = Range("A" & i)

tensuu = Range("E" & i)

If tensuu >= 80 Then ' 80点以上の人は H 列に記入する

Range("H" & j) = Name ' j は行番号

j = j + 1 ' 記入したら次の行に進める

End If

Next i

　jは成績優秀者を記入するH列の行を表す変数です。最初2からスタートし、1人記入するたびに1増やしています。ここで重要なのは初期値を設定するj = 2はForの手前に書く（ループの外に書く）ことと、j = j + 1はIfの内側に書くことです。これを間違えると正しい結果が得られません。

【If課題1】

1. j = 2の場所をFor文の次にすると、どうなるか。なぜ、そうなるのか。
2. j = j + 1の場所をEnd Ifの後にすると、どうなるか。なぜ、そうなるのか。

【If課題2】

　シート「If課題」に対して、以下の処理を行うプログラムを作成しなさい。

1. D列に英語と数学の合計得点を記入するプログラムを作成しなさい。
2. 数学について、60点未満の人のセルをRGB(255,255,200)で塗りつぶし、そうでない人は塗りつぶしなし、とするプログラム作成しなさい。
3. 英語と数学の両方ともが60点以上の人を合格とします。E列に合否を書き込むプログラムを作成しなさい。合格者は "○", 不合格者は "×" を記入します。
4. 英語が60点未満の人は補習を実施します。補習対象者の名前をG2, G3, G4...に列挙するプログラムを作成しなさい。

## Else If

　シート「Else If練習」を見て下さい。名前と得点が記入されています。プログラムが正しく動くことを確認するため、名前や点数はプログラムの動作確認用の値が入っています。

　B列の値に従ってC列に評価を書き込みます。0～59はF, 60～69はD, 70～79はC, 80～89はB, 90～100はAを書き込みます。IFを使います。点数が低い方から考えても高い方から考えてもいいですが、ここでは高い方から考えることにします。

1. 90以上はA, そうでないなら2.へ
2. 90未満の場合、80以上ならB, そうでないなら3.へ
3. 80未満の場合、70以上ならC, そうでないなら4.へ
4. 70未満の場合、60以上ならD, そうでないならF

　この考えをプログラムにすると以下のようになります。

For i = 2 To 13

tokuten = Range("B" & i)

If tokuten >= 90 Then

hyouka = "A"

Else

If tokuten >= 80 Then

hyouka = "B"

Else

If tokuten >= 70 Then

hyouka = "C"

Else

If tokuten >= 60 Then

hyouka = "D"

Else

hyouka = "F"

End If

End If

End If

End If

Range("C" & i) = hyouka

Next i

　Elseの中にIfが入れ子になっており、見づらいです。これを以下のように書くことができます。

For i = 2 To 13

tokuten = Range("B" & i)

If tokuten >= 90 Then

hyouka = "A"

ElseIf tokuten >= 80 Then

hyouka = "B"

ElseIf tokuten >= 70 Then

hyouka = "C"

ElseIf tokuten >= 60 Then

hyouka = "D"

Else

hyouka = "F"

End If

Range("C" & i) = hyouka

Next i

　ElseIfという命令を使っています。慣れるまではこの形式は違和感があるかも知れませんが、プログラムの定番表現なので、処理が複数に分かれるIfの使い方に慣れて下さい。

　次の課題のための説明です。セルに文字列 A が入っているか否かは次のように書きます。

If Range("D" & i) = "A" Then

　セルに何も入っていないか否かを判定する場合、以下のように書きます。

If Range("D" & i) = "" Then

【ElseIf課題1】

1. D列にレポートの評価値が入っています。E列にレポート点を記入しなさい。レポート点は評価がAのとき10, Bのとき8, Cのとき6, Dのとき4, Eのとき2, 未提出のとき0です。レポート未提出の場合D列のセルは空です。
2. B列とE列の合計得点をF列に記入しなさい。ただし、レポート点が10点の人はボーナス点として10点が加算されます。
3. レポート未提出者の名前をH2, H3...に列挙しなさい。あるセルが空白か否かは以下のように判定します。

　皆さんはこれで、Excel VBAの基礎的な命令のほとんどを修得しました。

　残っている命令はDo While ～ Loop のループだけです。これをマスターしたなら、後は自分のアイデア次第で何でも組むことができるでしょう。

よくあるエラーのパターン

　以下のプログラムはエラーが発生します。どこが間違っているか分かりますか？

For i = 1 To 10

If Range("A" & i) >= 10 Then And Range("B" & i) <= 30 Then

("A" & i) = 12

End If

Nexti

解答

1. If 文の中央付近の Then は不要
2. Range("A" & i) とするところを ("A" & i) としている
3. Nextとiの間にスペースがない。

　次のプログラムは「Nextに対応するForがありません。」というエラーが出ます。

For i = 1 To 10

If Range("A" & i) = "" Then

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

Next i

　しかし、実際にはEnd Ifがないのがエラーの原因です。なぜ、このようなエラーが出るのかは分かりません。

　次のプログラムは何か錯覚をしています。どのような錯覚をしているか分かりますか？

For i = 1 To 10

b = Range("B" & i)

If Range("A" & i) = "A" Then

a = 10

Else

a = 5

End If

Next i

解答

　2行目にa = Range("B" & i) とあります。これは変数aにセル "B" & i の内容を入れることを意味しますが、それを、aに文字列Range("B" & i) を入れることと錯覚した。

　4, 6行目のa = 10とa = 5の部分は、それぞれRange("B" & i) = 10, Range("B" & i) = 5を意図している。