心理学におけるExcel VBAの利用 その1 : VBAの基本文法

関西大学『社会学部紀要』第38巻第1号，2006，pp.191-221　　ISSN 0287-6817

資料

心理学におけるExcel VBAの利用 その1
—VBAの基本文法—

久 本 博 行

Applications of Excel VBA for the Psychology (1)
VBA Programming

Hiroyuki HISAMOTO

Abstract

An experimental system in psychology using Excel VBA(Visual Basic Applications) is proposed, and the basic grammar of VBA is introduced.

Key words: Excel VBA, Psychological Experiments, Programming

抄　　録

Excel VBA（Visual Basic Applications）によって，心理学のコンピュータ実験システムを作成することが提案され，その1としてVBAの基本文法が紹介された。

キーワード：Excel VBA，心理学実験，プログラミング

はじめに

現在，心理学の実験にはコンピュータが使われることが多くなってきている。これは，コンピュータによる刺激提示や提示時間の制御，反応時間の測定などが容易なためである。また，コンピュータ上で実験が行われ，データが収集されるとそのデータをすぐに分析することができる，という利点もある。

コンピュータによる心理学実験システムとしては，Cedrus社のSuper Labが有名であり多くの実験で用いられている。しかし，Super Labは高価で，システムは柔軟性に富むものであるが，それゆえに設定も複雑である。また，コンピュータによる心理学実験についての著作は数多く発表されている（田中　1975，吉村・山上　1983，市川・矢部　1985，野沢　1986，金子ら　1990，Dixon, M.R. & MacLin, O.H. 2003，北村・坂本　2004，水野りか　2004）が，この中で1970年代から90年代に発表されたものはBASICで作られたものがほとんどで，Windowsに対応していない。Dixon & MacLinのものはVisual Basic .Netが必要であること，北村・坂本のものはVisual Basic6.0，HSP，Delphiといった各種のプログラミング言語を使っているため初心者には難しく，さらにVisual Basic6.0やDelphiの開発環境が必要である。水野はWebブラウザ上で動く心理学実験システムをJAVAとPerlで作成しているが，これは開発環境はフリーで入手しやすいのであるが，JAVAやPerlを学習するのが難しいという難点がある。もちろん，北村・坂本や水野らのプログラムをそのまま利用するうえでは問題はないのであるが，同様なプログラムを開発しようとすると上記のような障害がある。

したがって，開発環境が手軽に手に入り，プログラミングが容易な言語で実験のシステムを構築することが多くの実験者にとって有益であると考えられる。そこでフリーウェアではないが，多くのWindowsシステムにバンドルされており，プログラミングも比較的容易なMicrosoft社のExcelのVBA（Visual Basic for Applications）を使い，いくつかの心理学実験用ソフトやデータ分析用ソフトを作成した。これらは，心理学実験を行おうとするものが，そのままでも使えるが，さらに自分の実験向きにプログラムを改良したり，サンプルプログラムとして利用できるようなものを目指した。今回は「その1」として自分でVBAのプログラミングが可能になるように，VBAの基本文法を紹介する。

また，今後は「錯視」，「Stroop効果」などの実験ソフトウェアや「SD法」，「一対比較法」などの刺激提示とデータ収集用ソフトウェア，「因子分析」などのデータ解析ソフトウェアを順次発表していきたい。

1.Excel VBAの操作

1.1 VBAとは

VBAとは，Visual Basic for Applicationsの略でMicrosoft社のWord，Excel，Access，Power Pointといったオフィス製品で使われるプログラミング言語です。VBAを利用するとOfficeを使ったアプリケーション・ソフトを開発することができます。ここでは，Excelのアプリケーションを開発していきますが，この知識は他のOffice製品のアプリケーションを作る時にも役立ちます。

1.2 マクロのセキュリティ

Excelでマクロを利用する場合には，はじめにマクロのセキュリティ設定をしなければなりません。マクロのセキュリティレベルを変えるには，[ツール] メニューをクリックし，次に［マクロ］をポイントし［セキュリティ］をクリックします。

図1.2.1

そうすると，次の図のようなセキュリティ設定のダイアログボックスが出てきます。通常，デフォルト（既定値）ではセキュリティレベルは「高（H)」に設定されており，自

分で作成したマクロを実行することができません。そこで，セキュリティレベルを「中(M)」にします。そして，「OK」ボタンをクリックします。この状態ではまだセキュリティレベルは変わっていませんので，いったんExcelを終了し，もう一度Excelを起動しなおします。これで，セキュリティレベルの変更が完了しました。

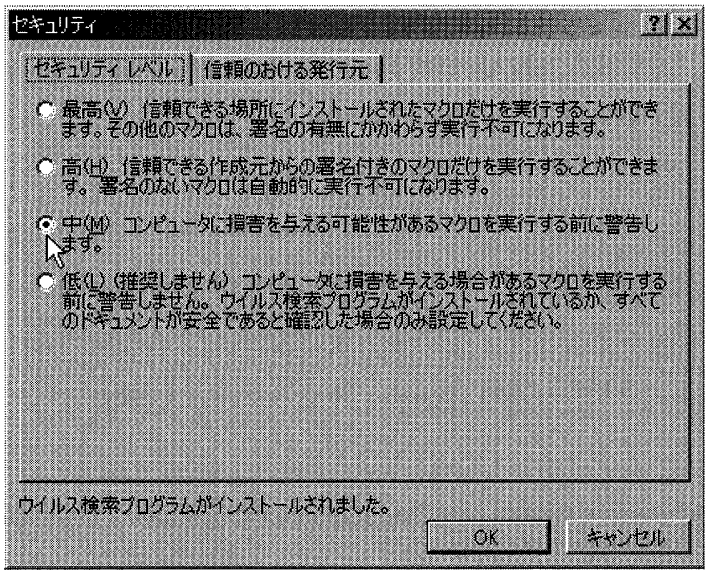


図1.2.2

次にマクロを含むExcelのファイルを開こうとすると，下図のようなセキュリティ警告が出ます。そのときは，自分が作ったような信頼できるマクロであれば「マクロを有効にする（E)」をクリックしファイルを開きます。

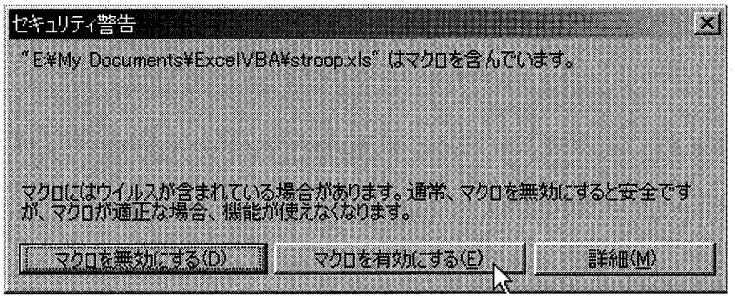


図1.2.3

1.3 Visual Basic Editorの起動

マクロを作成するには，まずマクロを作成するツールVisual Basic Editor（以下VBEと略）を起動します。［ツール］メニューをクリックし，［マクロ］→［Visual Basic Editor］の順でクリックします。（Altキーを押しながら，F11を押してもVBEを起動できます。）

マクロを作る他のやり方として，Excelでは「マクロ記録」という方法があります。これは，マクロに行わせたい操作を実際に行い，その一連の操作手順を記録し，マクロとして利用するものです。これは簡単にマクロが作成できますが，使える機能が限られているのでここでは取り上げません。

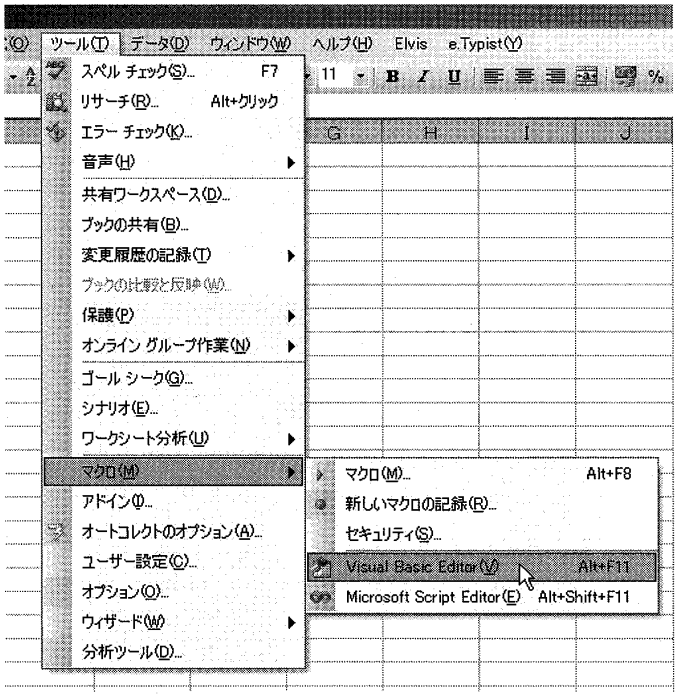


図1.3.1

1.4 やさしい入門

では，実際にプログラムを書いてみましょう。図1.4.1のような形のフォームを作り，「はじめてのプログラム」というボタンをクリックすると，「Hello World!」と表示されるようにします。

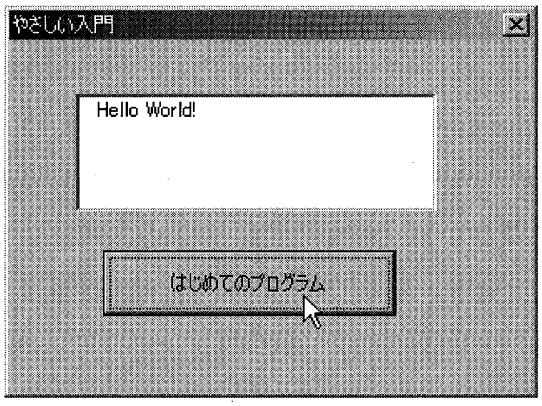


図１．４．１

１．４．１　ユーザーフォームの作成

「挿入」ボタンから「ユーザーフォーム」を選び（図１．４．２），ユーザーフォームを作ります。この操作は「挿入」メニューから「ユーザーフォーム」を選んでもできます。

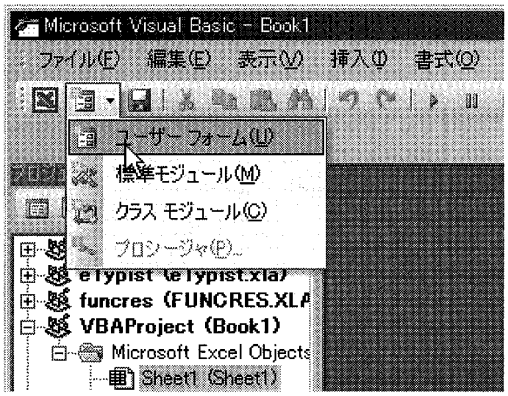


図１．４．２

ここで少し，Visual Basic Editorの画面（図１．４．３）の説明をしておきましょう。プロジェクトエクスプローラは，そのブックに含まれるオブジェクトやモジュールの階層構造を表したものです。

プロパティウィンドウは，選択されているオブジェクトのプロパティを見たり，編集したりするためのものです。フォームウィンドウは，フォームを編集するためのもので，ツールボックスはフォームに貼り付けるいろいろなコントロールを用意しているものです。ツールボックスが非表示の場合は，標準ツールバーのツールボックスボタンを押すと表

示されます。作業ウィンドウはヘルプを表示します。

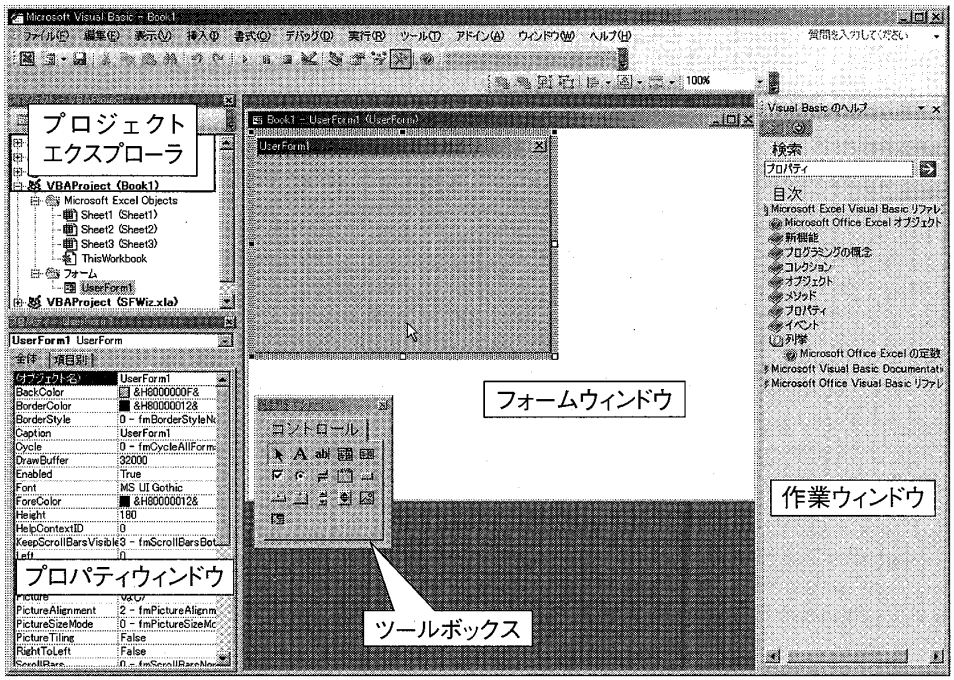


図1.4.3

「Hello World!」と表示が出ている部分は，テキストボックスと呼ばれるものです。このテキストボックスをフォームに貼り付けます。ツールボックスのテキストボックスをクリックし（図1.4.4），フォーム上で適当な大きさになるようにドラッグします。（図1.4.5）

図1.4.4

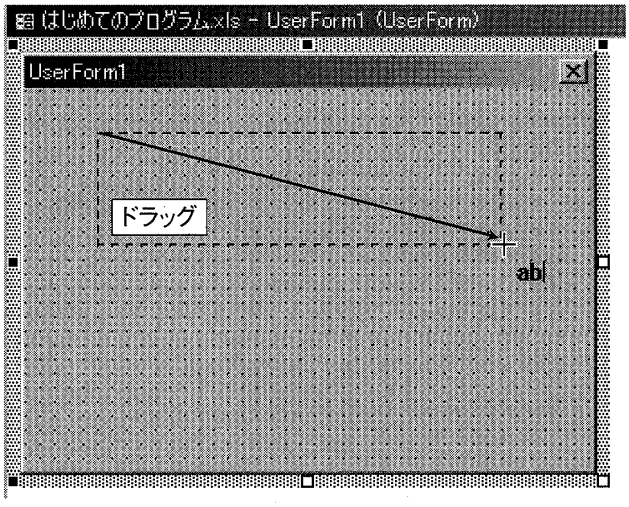


図1.4.5

次に，「初めてのプログラム」というボタンを作ります。このボタンのことをコマンドボタンといいますが，これをテキストボックスの同じ要領で貼り付けます。ツールボックスのコマンドボタンをクリックし（図1.4.6），フォーム上で適当な大きさになるようにドラッグします。（図1.4.7）

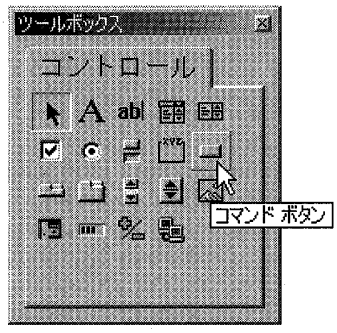


図1.4.6

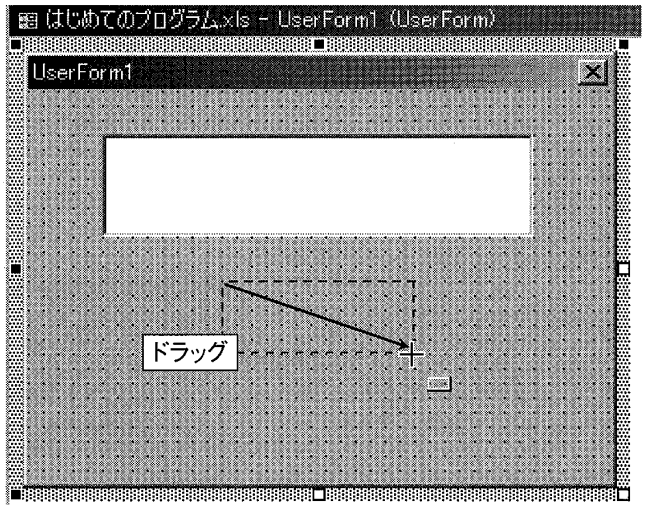


図1.4.7

次に，フォームのタイトルバーに表示されている名前が「UserForm1」となっていますので，これを「やさしい入門」に，コマンドボタンには「CommandButton1」と表示され

ていますので，これを「初めてのプログラム」に変更します。

まず，フォーム上でテキストボックスやコマンドボタンのないところならどこでも構いませんからクリックし，フォームを選択状態にします。そして画面左下のプロパティ・ウィンドウの「Caption」欄を「やさしい入門」に書き換えます（図1.4.8）。これでユーザーフォームのタイトルバーの表示が「UserForm1」から「やさしい入門」に変わりました。コマンドボタンについても同様です。コマンドボタンを一度クリックし，選択状態にするとコマンドボタンのプロパティがプロパティ・ウィンドウに表示されますので，「Caption」欄を「はじめてのプログラム」に書き換えます（図1.4.9）。プロパティでは，この他に文字の色，サイズ，そのオブジェクトの幅，高さ，背景色などの属性が定義されています。プロパティはこのようにいろいろな属性を定義しているもので，あるオブジェクトを選択状態にすると，プロパティ・ウィンドウにそのプロパティが表示されるような仕組みになっています。なお，プロパティを変更するのは，今のようにプロパティ・ウィンドウでもできますし，プログラムからもできます。

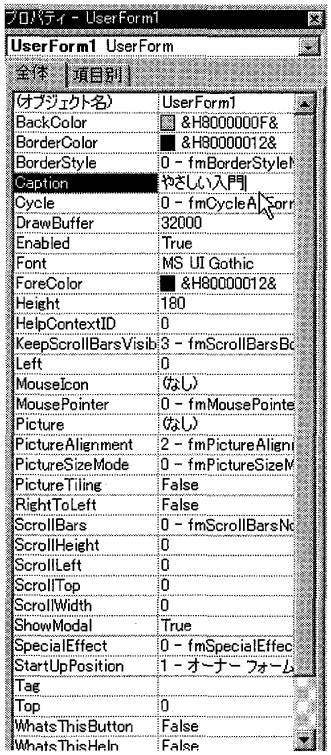


図1.4.8

図1.4.9

１．４．２　プログラムの作成

では，次にコマンドボタンを押すと「Hello World!」と表示されるようにプログラムを作ってみます。

先ほど作った「はじめてのプログラム」というコマンドボタンをダブルクリックします（図１．４．10)。そうすると，図１．４．11にあるようなコードウィンドウが表示されます。このコードウィンドウでコマンドボタンが押されたときの動作を記述します。

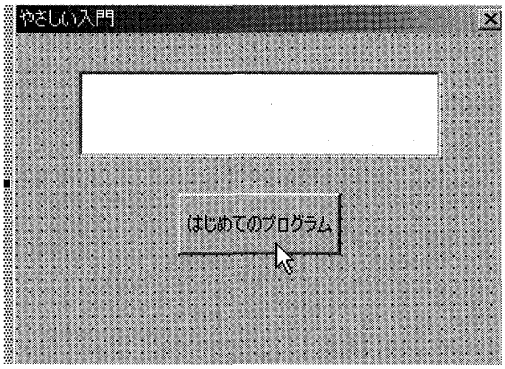


図１．４．10

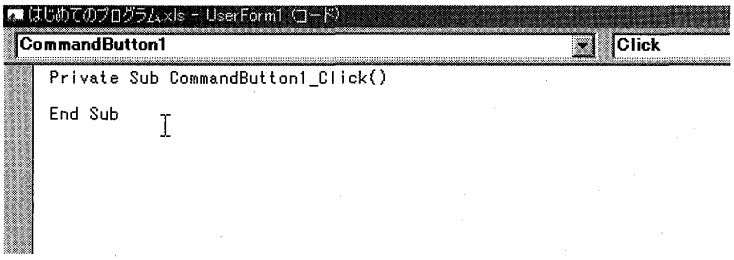


図１．４．11

Private Sub CommandButton1_Click( )

と

End Sub

の２行の間に

TextBox1.Text = "Hello World!"

と入力します。これで出来上がりです。

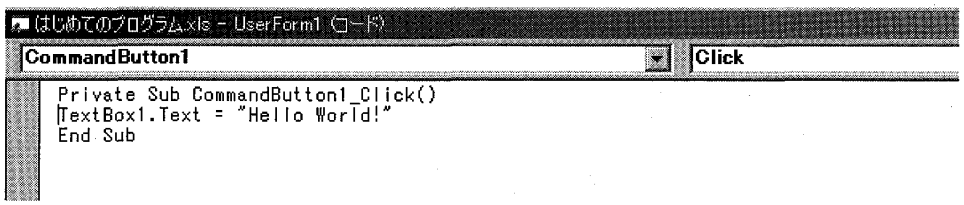


図１．４．12

プログラムの詳しい説明の前に，このプログラムを動かしてみましょう。ツールバーにある「Sub/ユーザーフォームの実行」ボタンをクリックしてください（図1.4.13)。そして，「はじめてのプログラム」のボタンをクリックしましょう。「Hello World!」と表示されましたか（図1.4.14)。

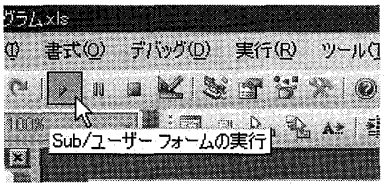


図1.4.13

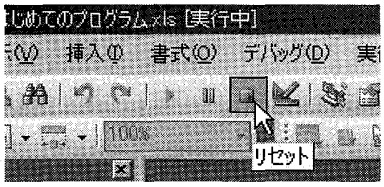


図1.4.15

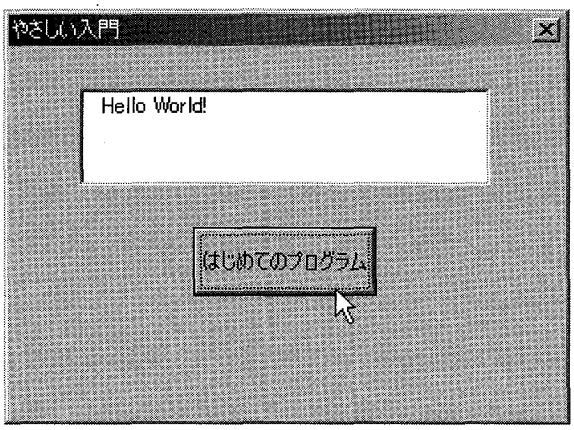


図1.4.14

もし，うまく表示されなかったり，エラーが表示された場合は，左記のプログラムとよく見比べてください。正しく実行できれば，リセットボタン（図1.4.15）を押して停止します。

では，プログラムについて説明しておきましょう。プログラムの1行目は，このプログラムがCommandButton1をクリックされたときの動作を定義しているプログラムであることを示しています。

```
Private Sub CommandButton1_Click()
   TextBox1.Text = "Hello World!"
End Sub
```

2行目が実際にテキストボックスに「Hello World!」を表示している部分です。TextBox1.TextはTextBox1というオブジェクトのなかのTextプロパティを示しており，＝は代入の記号で，Textプロパティに""で囲まれた文字列の「Hello World!」を代入しなさい，ということになります。文字列を指定するときには，""で囲むことになっており，""で囲まれた文字列はプログラムではなく，データとして取り扱われます。

3行目はこのプログラムの終わりを示す部分です。

2　基本文法

ここでは，Excel Visual Basic for Applications（VBAと略）の基本的な文法を説明します。なお，文法の説明の中で［　］で囲まれた部分は，省力可能であることを示しています。

2.1　変数

変数とはプログラム中でa,bなどのように名前がつけられた，数値や文字列を収納しておく記憶領域です。変数の名前のつけ方には規則があります。

・名前の先頭は数字以外の文字（英字，漢字，ひらがな，カタカナ）で始めます。
・名前には空白や&，#などの記号（_アンダースコアを除く）を使うことはできません。
・名前は255文字（半角の場合）以内でなければなりません。
・Visual Basicの関数，ステートメント，およびメソッドと同じ名前を使うことはできません。

正しい変数名の例

a
heikin
利息

悪い変数名の例

1gakki……数字で始まっている
abc@def…@が使われている
right……Visual Basicに同じ関数名がある

2.2　データ型

VBAで取り扱う変数には，表2.2のようなデータ型があります。通常，取り扱うデータが整数の場合は，Integer型かLong型を，実数型の場合はSingle型かDouble型を使います。金銭の計算を行う場合は，Currency型を使います。用途に合わせることと，その変数型で表現できる範囲を考え合わせてデータ型を選択します。

表2.2 データ型

データ型	サイズ	説　　明
Byte（バイト型）	1バイト	0～255の整数
Boolean（ブール型）	2バイト	TrueあるいはFalseのみです。数値をブール型に変換すると0はFalseで0以外の値はTrueとなります。また，ブール型の変数を数値に変換するとTrueは－1，Falseは0となります。
Integer（整数型）	2バイト	-32,768～32,767
Long（長整数型）	4バイト	-2,147,483,648～2,147,483,647
Single （短精度浮動小数点数型）	4バイト	－3.402823E38～－1.401298E－45（負の値） 1.401298E－45～3.402823E38（正の値）
Double （倍精度浮動小数点数型）	8バイト	－1.29769313486231E308～ －4.94065645841247E－324（負の値） 4.94065645841247E－324～1.29769313486231E308（正の値）
String（文字列型）	10バイト＋ 文字列の長さ	0～2GB
Currency（通貨型）	8バイト	－922,337,203,685,477.5808～922,337,203,685,477.5807
Decimal（10進型）	14バイト	小数部分を持たない数値の場合， －79,228,162,514,264,337,593,543,950,335～ 79,228,162,514,264,337,593,543,950,335 小数点以下28桁の数値の場合， －7.9228162514264337593543950335～ 7.9228162514264337593543950335
Date（日付型）	8バイト	西暦100年1月1日～西暦9999年12月31日の範囲の日付と，0:00:00～23:59:59の範囲の時刻
Object（オブジェクト型）	4バイト	オブジェクトを参照するためのアドレスです。
Variant（バリアント型）		数値の場合は，倍精度浮動小数点数型と同じで，文字列の場合は，文字列型と同じです。

2.3 変数の宣言

変数は使う前に必ず宣言してから使うようにしましょう。VBAでは変数は宣言しなくても使えますが，宣言してから使うようにした方が，間違いを発見しやすくなります。

変数の宣言には，Dim文を使います。Dimの構文は以下の通りです。

```
Dim 変数名 [As データ型]
```

[ ] で囲まれた部分は省略可能です。

例

```
Dim Mean As Single
Dim i As Integer
Dim x As Double, y As Double, s As String
```

悪い例

```
Dim j, k As Integer  'この場合，jはInteger型ですがkはVariant型になります。
```

変数を使う場合に必ず宣言を必要とするように強制するには，モジュールの先頭で次のように記述します。

```
Option Explicit
```

この宣言をしておくと，宣言をせずに変数を使うとエラーメッセージが表示されます（図2.3.1）。例えば，タイプミスで変数の綴りを間違えた場合でも，Option Explicitの宣言をしておくと，タイプミスした変数は宣言されていないのでエラーになり，すぐに見つけることができます。

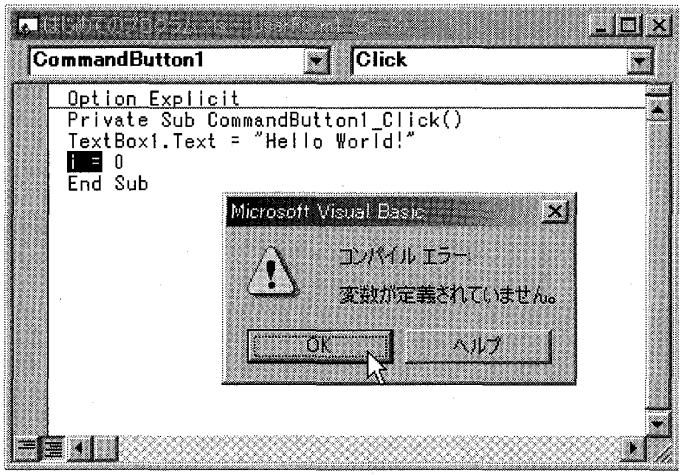


図2.3.1

2.4 定数

一般的には，プログラム上に直接記述された数値や文字列のことを定数と呼びますが，ここではユーザーが定数を定義する方法について説明します。ユーザーが定数を宣言する場合は，Constキーワードを使います。

```
[Public/Private] Const 定数名 [As データ型] = 値
```

例

```
Const TaxRate As Single = 0.5
Const Morning As String = "おはようございます。"
```

VBAで既に定義されている定数もあります。非常に数多くの定数が定義されていますので，詳しくはHelpを見てください。

VBAで定義されている定数の例

```
'色の定数
    vbBlack   '黒色
    vbRed     '赤色
    vbWhite   '白色   など
'キーコードの定数
    vbKeyA    'アルファベットのA
    vbKey1    '数字の1
    vbKeyF1   'ファンクションキーF1など
```

なお，比較的よく使われる円周率については，PI ()というワークシート関数を呼び出すことで3.14159265358979という値を得ることができます。

2.5 演算子

VBAで使われる演算子には，以下のようなものがあります。

表2.5.1 演算子の一覧表

演算子	機　　能	使用例
算術演算子		a＝10, b＝4とすると
+	2つの数値の和を求める	a + b　　14
-	2つの数値の差を求める	a − b　　6
*	2つの数値の積を求める	a * b　　40
/	2つの数値の商を求める	a / b　　2.5
¥	2つの数値の商の整数部を求める	a ¥ b　　2
Mod	2つの数値の割り算の余りを求める	a Mod b　　2
^	べき乗を求める	a ^ b（aをb乗する）10000
文字列連結演算子		a = "Good", b = "Morning"とすると
&	2つの文字列を連結する	a & b　　"Good Morning"
+		a + b　　"Good Morning"
比較演算子		a＝5 b＝8とすると
＝	等しい	a ＝ b　　False
＜＞	等しくない	a ＜＞ b　　True
＜	より小さい	a ＜ b　　True
＞	より大きい	a ＞ b　　False
＜＝	以下	a ＜＝ b　　True
＞＝	以上	a ＞＝ b　　False
Like	文字列の比較を行う	a = "Good", b = "G*"の場合（*はメタ文字） a Like b　はTrue
Is	オブジェクト変数の比較を行う	Dim a Object, b Object　Set b Object = a Objectの場合 a Is b　　True
論理演算子		a＝2, b＝3, c＝4, d＝5とすると
Not	論理否定	Not a ＝ b　　True
And	論理積	a ＜＝ b And c ＜＞ d　　True
Or	論理和	a ＝ b Or c ＝ d　　False
Xor	排他的論理和	a ＜＝ b Xor c ＜＝ d　　False
Eqv	論理等価演算	a ＜＝ b Eqv c ＜＝ d　　True
Imp	論理包含演算	a ＜＝ b Imp c ＞＝ d　　False

2.6 配列

2.6.1 1次元配列

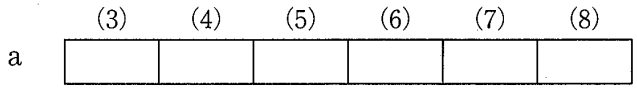
配列は，たくさんの変数を1つの変数名で利用できるようにしたもの，と考えることができます。線形代数のベクトルや行列を，思い浮かべてもらってもいいでしょう。

配列の宣言は，次のように行います。

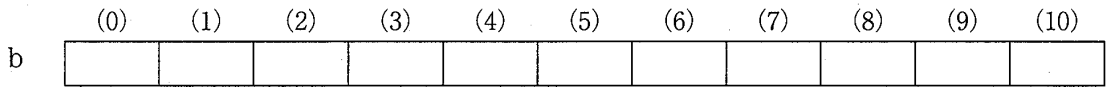
```
Dim 変数名（要素の初めの番号 To 要素の終わりの番号）As データ型
```

例

```
Dim a(3 To 8) As Single     'a(3) ～ a(8)まで6つのSingle型の要素を持つ配列を
                            '宣言したことになります。
```




```
Dim b(10) As Integer        '要素の初めの番号とToを省略した場合，
                            '要素は0から始まるとみなされます。
                            'この場合はb(0)～b(10)までの11の要素を
                            '持つ配列を宣言したことになります。
```



要素の初めの番号を1からはじめたい場合には，モジュールの先頭で次のように書きます。

```
Option Base 1
```

この宣言で全ての配列要素の先頭は，(1) から始まるようになります。この指定がなければ (0) から始まります。指定できるのは1か0しかありません。

2.6.2 多次元配列

2次元以上の配列も作ることができます。

例

```
Dim x(4, 5) As Long      '下記の5行6列の表のような配列を宣言したことになります。
```

	(0)	(1)	(2)	(3)	(4)	(5)
(0)	x (0,0)	x (0,1)	x (0,2)	x (0,3)	x (0,4)	x (0,5)
(1)	x (1,0)	x (1,1)	x (1,2)	x (1,3)	x (1,4)	x (1,5)
(2)	x (2,0)	x (2,1)	x (2,2)	x (2,3)	x (2,4)	x (2,5)
(3)	x (3,0)	x (3,1)	x (3,2)	x (3,3)	x (3,4)	x (3,5)
(4)	x (4,0)	x (4,1)	x (4,2)	x (4,3)	x (4,4)	x (4,5)

この表からも分かるように，2次元配列の宣言では

Dim 変数名（最後の行番号，最後の列番号）

という形になります。

また3次元以上の配列も利用することができます。次元数の最大値は60です。

例

```
Dim x(3, 5, 7) As Double        '3次元の配列
Dim y(2, 4, 6, 8) As String     '4次元の配列
```

配列の要素数の最大値は，使用しているコンピュータのメモリの容量によって決まります。上限を超えて大きな配列を宣言すると「メモリが不足しています。」（図2.7.2）というエラーが出ます。エラーが出なくても，不必要に大きな配列を宣言すると実行速度が極端に低下しますので注意しましょう。

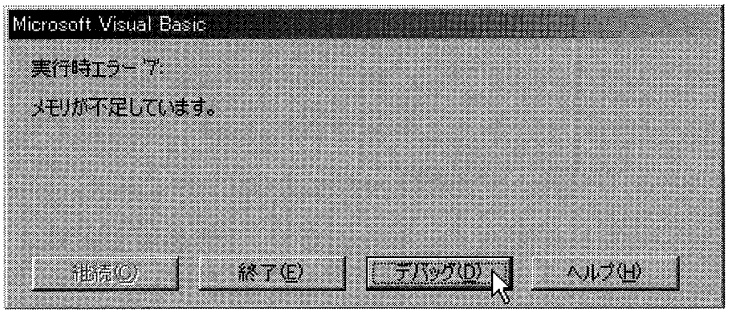


図2.6.2

2.6.3 動的配列

必要な配列の大きさが，プログラムの実行時にならないと分からない場合にはどうすればいいのでしょうか。そのような場合には，動的配列を使うことができます。

動的配列は，最初に要素数を指定しないで配列を宣言しておき，実行時に要素数が確定したときに，その配列を再宣言します。

例

```
Dim z() As Double              '要素数を指定しないで配列を宣言しておく
.
.
.
ReDim z(n, m) As Double        '要素数が決まった時に，もう一度宣言しなおす。
```

ReDim文の構文は，下記のとおりです。この文は配列の大きさや次元数を変えるために何度でも使うことができます。

ReDim [Preserve] 変数名（要素数）[As データ型]

ReDim文はDim文とほぼ同じですが，違うのはPreserveというオプションを指定できることです。Preserveをつけると，既に配列に格納されているデータを保持したまま配列の大きさや次元数を変えることができます。Preserveがない場合は，配列は初期化されます。

2.7 コメント

コメントというのは，プログラムを読みやすくするために記述する，解説文のことです。コメント文は，プログラムの実行には何も影響を与えません。プログラムを作成している時点では，よく分かっているつもりでも，後で読み返すとどのような処理をしているのかさっぱり分からない，というようなことがあります。そのようなことを防ぐために，きちんと分かりやすいコメントを書く習慣を付けましょう。

「Rem」あるいは「'（シングルクォーテーション）」に続く文字列は，コメントとみなされ，緑色の文字で表されます。

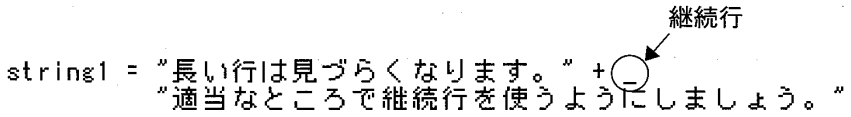
例

```
Rem Remに続く文字列はコメント文です。
' コメント文はプログラムを分かりやすくする解説です。
a = b + c   ' 文の途中からでもコメントを入れることができます
```

2.8 継続行

プログラムの1行が非常に長くなると，画面をスクロールしてみなければならず，見づらくなってしまいます。そういう場合は，1行のプログラムを2行以上に書くことができます。「 _（半角の空白とアンダースコア）」を行の最後にすると，次の行と続いていて同一の行とみなされます。

例



2.9 条件判断

条件判断とは，変数の値などによってプログラム中の処理を変えるような操作をいいます。コンピュータが他の機械と大きく異なるのは，この条件判断ができるところにあるのです。

2.9.1 If文

If文の構文は以下のような2通りのものがあります。

```
If 条件式 Then 実行文 [Else 実行文 EndIf]
```

```
If 条件式 Then 実行文 [ElseIf 条件式 Then実行文 [Else実行文 ] EndIf]
```

Ifのあとの条件式には，比較演算子や論理演算子を使って答えが真（True）か偽（False）であるような式か，あるいはBoolean型の変数が入ります。条件式の答えが真であれば，Thenの後の実行文を実行し，偽であれば何もしないか，Elseの後の実行文あるいはElseIfのあとの条件式の評価を行います。

例

```
If a = 10 Then b = 10          '変数aの値が10と等しい場合変数bにも10を代入する

If a = 10 Then                 ' 変数aの値が10と等しい場合
    b = 10                     ' 変数bに10を代入する
Else                           ' そうでなければ
```

```
    b = 5                    ' 変数bに5を代入する
End If

a = b = 10                   '変数bが10と等しい場合aにはTrueが，bが10等しくな
                             'ければaにはFalseが入る

Dim a As Boolean             '変数aをBooleanとして宣言
  :
If a Then c = 5              ' Boolean型の変数の場合は，その変数が
                             'TrueかFalseかでIfの判断が行われ，
                             'ここではaがTrueであればc = 5 が実行されます。
```

次はElseIfの例ですが，ElseIfを使った場合，注意しなければいけないのはIfやElseIfの条件式のうち一つでも真の場合があれば，それ以降の条件式については判定を行わないことです。下記の例はtokutenが80以上ならseisekiに"優"を，70以上80未満の場合は"良"，60以上70未満は"可"，60未満は"不可"を代入するプログラムです。

ElseIfの正しい例

```
If tokuten >= 80 Then        '変数tokutenの値が80以上であるなら
    seiseki = "優"           '変数seisekiiに"優"を代入する
ElseIf tokuten >= 70 Then    '80以上でなく，70以上であるなら
    seiseki = "良"           '変数seisekiiに"良"を代入する
ElseIf tokuten >= 60 Then    '80以上でも70以上でもなく60以上なら
    seiseki = "可"           '変数seisekiiに"可"を代入する
Else                         '上記の条件のいずれにも偽である場合
    seiseki = "不可"         '変数seisekiiに"不可"を代入する
End If
```

下記との誤った例では，最初にtokutenが60以上かどうか判定され，60以上の場合は全て"可"となり，60未満のものだけが"不可"となってしまいます。それは，ElseIfではいづれかの条件式が真（True）となると，それ以降に記述されているElseIfの条件式は，評価されなくなるからです。

ElseIfの誤った例

```
If tokuten >= 60 Then
    seiseki = "可"
ElseIf tokuten >= 70 Then
    seiseki = "良"
ElseIf tokuten >= 80 Then
    seiseki = "優"
Else
    seiseki = "不可"
End If
```

2．9．2　Select Case文

Select Case文は，Elseifと同様に多方向への分岐を行う制御文です。構文は以下のとおりです。

```
Select Case 式1
    Case 式2
            実行文
              :
            実行文
    Case 式3
            実行文
              :
            実行文
    Case 式n
              :
              :
              :
    Case Else
              :
              :
              :
EndSelect
```

Select Case文の式１には，式や変数を記述します。この式１と式２～式nまでで等しい式のCaseに続く文を実行します。Case Elseは必要がなければ，記述しなくてもかまいませんが，想定外の値が出た場合Case Elseに跳ぶようにしておくと予期せぬエラーを防ぐことができます。

Elseifで用いたものと同じ例を，Select Caseで記述してみましょう。下記の例はtokutenが80以上ならseisekiに"優"を，70以上80未満の場合は"良"，60以上70未満は"可"，60未満は"不可"を代入するプログラムです。例１のプログラムは数値の大きいものから判定しています。それに対して，例２のものは数値の小さいものから判定しています。例３は，値の範囲を指定して分岐するものです。

ここでIs, Toというキーワードが使われているのに注意して下さい。Isは比較式を使って値を指定したい場合に使います。Toは最小値と最大値を指定し，値の範囲を示したい場合に使います。

例4はもっとも一般的によく使われるSelect Case文の使い方です。

例1

```
Select Case tokuten
    Case Is >= 80
        seiseki = "優"
    Case Is >= 70
        seiseki = "良"
    Case Is >= 60
        seiseki = "可"
    Case Is < 60
        seiseki = "不可"
    Case Else
        seiseki = "入力エラー"
EndSelect
```

例2

```
Select Case tokuten
    Case Is < 60
        seiseki = "不可"
    Case Is < 70
        seiseki = "可"
    Case Is < 80
        seiseki = "良"
    Case Is >= 80
        seiseki = "優"
    Case Else
        seiseki = "入力エラー"
EndSelect
```

例3

```
Select Case tokuten
    Case 80 To 100
        seiseki = "優"
    Case 70 To 79
        seiseki = "良"
    Case 60 To 69
        seiseki = "可"
    Case 0 To 60
        seiseki = "不可"
    Case Else
        seiseki = "入力エラー"
EndSelect
```

例4

```
Select Case i
    Case 1
        days = 31
    Case 2
        days = 28
    Case 3
        days = 31
    Case 4
        days = 30
```

```
    Case 5
        days = 31
    Case 6
        days = 30
    Case 7
        days = 31
    Case 8
        days = 31
    Case 9
        days = 30
    Case 10
        days = 31
    Case 11
        days = 30
    Case 12
        days = 31
EndSelect
```

2.10 繰り返し

条件判断がコンピュータのコンピュータらしいところなのですが，繰り返しはコンピュータが機械であることを思い出させてくれるところで，同じ処理を何度も何度も繰り返し実行していくところです。

2.10.1 For … Next

For … Nextは反復回数が，事前に分かっている場合に使われる繰り返しの方法です。

For … Nextは次のような構文です。

```
For 変数 = 初期値 To 終値 [Step 増分]
   実行文
    …
   実行文
Next [変数]
```

例1

アクティブなシート上に下図のようなデータがある場合，これらのデータを1次元配列の変数に代入するプログラムを作ってみましょう。

	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5	6	7	8	9	10

図2.10.1 データの表

```
Dim a(1 To 10) As Integer
Dim i As Integer

    For i = 1 To 10
        a(i) = Cells(1, i)
    Next i
```

例2

次にシート上に下図のような2次元のデータがある場合，このデータを2次元配列に代入するプログラムを作ってみましょう。この場合は，For … Nextの繰り返しを二重に使います。

	A	B
1	1	2
2	2	3
3	3	4

図2.10.2　データの表

```
Dim a(1 To 3, 1 To 2) As Integer
Dim j As Integer, i As Integer

For j = 1 To 2
    For i = 1 To 3
        a(i, j) = Cells(i, j)
    Next i
Next j
```

上のプログラムでは，外側の繰り返しで初めにjが1となり，次に内側の繰り返しでiが1，2，3と変わっていき，iが3までくると内側の繰り返しが終了し，外側の繰り返しでjが2となります。a(i, j)は表2.10.1のように変化していきます。

表2.10.1　繰り返しの値の変化

繰り返しの回数	j	i	a(i,j)の値
1	1	1	1
2	1	2	2
3	1	3	3
4	2	1	2
5	2	2	3
6	2	3	4

なお，例１，例２では「Step 増分」の部分が省略されています。省略された場合の増分は，既定値１となります。次のように増分を２として，１つ飛ばしに指定したり，負の値を指定し大きいものから，小さいものへと繰り返すこともできます。

例３　増分が２刻みの場合

```
For i = 2 To 10 Step 2
    …
Next i
```

例４　増分が負の値

```
For i = 10 To 1 Step -1
    …
Next i
```

ここで，例１と例２で出てきたCellsというキーワードについて説明しておきます。Cellsとは，ワークシート上のセルの属性（プロパティ）を示すもので，そのセルの値，大きさ，色,フォント，フォントサイズといったプロパティがあるところです。

シート上の特定のセルのデータを取得するには，Cellsプロパティを使います。Cellsプロパティの構文は，下記のとおりです。

[オブジェクト名.] Cells（行番号,列番号）

オブジェクト名は，通常はシート名が入りますが，何も指定がなければアクティブなシートが指定されたとみなされます。また，列番号は左から順に数えた場合の列番号を指定します。ですから，列Aは列番号１，列Bは２，列Cは３というように指定します。

例

```
Worksheets("Sheet1").Cells(4, 5) = 10    ' Sheet1のセルE4に10を代入しています。
```

2.10.2　Do … Loop

For … Next文は，繰り返しの回数が分かっている場合に使いますが，繰り返しの回数が分からないが，定められた条件式が真の間繰り返したり，条件式が真になるまで繰り返したりする場合，Do … Loopを使います。Do … Loopには，条件式が真の間処理を繰り返すWhile型と，条件式が真になるまで処理を繰り返すUntil型があり，それぞれに繰り返しの最初で条件判断を行うものと最後で条件判断を行うものの計4つのタイプがあります。繰り返しの最初で条件判断を行う場合は，条件によってはその繰り返しが1回も実行されない場合があるのに対して，条件判断を最後に行う場合は，1回目の繰り返しは無条件に実行されるという違いがあります。

表2.10.3　Do … Loopの4つのタイプ

	While型　条件式が真の間反復	Until型　条件式が真になるまで反復
最初に条件判断	Do While 条件式 　実行文 　… 　実行文 Loop	Do Until 条件式 　実行文 　… 　実行文 Loop
最後に条件判断	Do 　実行文 　… 　実行文 Loop While 条件式	Do 　実行文 　… 　実行文 Loop Until 条件式

例1

アクティブなシート上に下図のようなデータがある場合，これらのデータを1次元配列の変数に代入するプログラムを作ってみましょう。その際，データがいくつあるか分からないが，データの無いセルがあれば，そこでデータは終わりにします。データが無いのは""（ダブルクォーテーション2つ）で示されます。

	A	B	C	D	E	F	G	H	I	J	K
1	1	2	3	4	5	6	7	8	9	10	

図2.10.2

Whileによる前判定

```
Sub t()
    Dim i As Integer
    Dim j As Integer
    Dim a(1 To 20) As Integer

    i = 1
    j = 1

    Do While Cells(1, j) <> ""
        a(i) = Cells(1, j)
        j = j + 1
        i = i + 1
    Loop

End Sub
```

Untilによる前判定

```
Sub t()
    Dim i As Integer
    Dim j As Integer
    Dim a(1 To 20) As Integer

    i = 1
    j = 1

    Do Until Cells(1, j) = ""
        a(i) = Cells(1, j)
        j = j + 1
        i = i + 1
    Loop

End Sub
```

Whileによる後判定

```
Sub t()
    Dim i As Integer
    Dim j As Integer
    Dim a(1 To 20) As Integer

    i = 1
    j = 1

    Do
        a(i) = Cells(1, j)
        j = j + 1
        i = i + 1
    Loop While Cells(1, j) <> ""

End Sub
```

Untilによる後判定

```
Sub t()
    Dim i As Integer
    Dim j As Integer
    Dim a(1 To 20) As Integer

    i = 1
    j = 1

    Do
        a(i) = Cells(1, j)
        j = j + 1
        i = i + 1
    Loop Until Cells(1, j) = ""

End Sub
```

2.11 その他の制御文

2.11.1 GoTo文

GoTo文は無条件に分岐する文です。構文は次のような形式で，GoTo文のあとに記述されたラベルと同じラベルの行へ制御が移ります。

```
  GoTo ラベル
  …
ラベル:
```

例1

下記の例ではCellの値が負のときNegativeというラベルの行まで跳びます。ですから，Cellの値が負の場合は，それ以降のFor … Nextは実行されなくなります。

```
Sub t()
Dim a(1 To 10) As Integer
Dim i As Integer

    For i = 1 To 10
        If Cells(1, i) < 0 Then GoTo Negative
        a(i) = Cells(1, i)
    Next i
Negative:

End Sub
```

2.11.2　Exit文

For … Next, Do … Loopなどの繰り返しから途中で抜け出したり，Subプロシージャや関数プロシージャから抜け出す文です。構文としては，以下の5種類があります。

表2.11.2　各種のExit文

文	説　　明
Exit For*	For … Nextの繰り返しから抜け，Next文の次に制御が移ります。
Exit Do*	Do … Loopの繰り返しから抜け，Loop文の次に制御が移ります。
Exit Sub	Exit文を含むSubプロシージャから抜け，そのSubプロシージャを呼び出した文の次の文へ制御が移ります。
Exit Function	Exit文を含むFunctionプロシージャから抜け，そのFunctionプロシージャを呼び出した文の次の文へ制御が移ります。
Exit Property	Exit文を含むPropertyプロシージャから抜け，そのPropertyプロシージャを呼び出した文の次の文へ制御が移ります。

*上記のExit文のうちExit ForやExit Doの場合，二重以上の繰り返しの中でExit文を使うとそのExit文を含む最も内側の繰り返しから抜け出します。

例1

下記の例は二重の繰り返しの中でExit文が使われていますが，この場合Exit文が実行されると，次に実行されるのはNext　jの文です。外側の繰り返しまでは抜け出すことはありません。

```
Sub t()
Dim a(1 To 3, 1 To 2) As Integer
Dim j As Integer, i As Integer

  For j = 1 To 2
      For i = 1 To 3
          If Cells(i, j) < 0 Then Exit For ──┐
          a(i, j) = Cells(i, j)             │
      Next i                                │
  Next j ←──────────────────────────────────┘
                                    ここへ抜け出す
End Sub
```

2.12 Subプロシージャ

Sub文の構文は下記のとおりです。Sub文はSubプロシージャを作るときに使います。

```
Sub 名前（[引数1,引数2,…,引数n]）
        …
        …
End Sub
```

Sub文では，Subプロシージャの名前と名前の後の（）が必要です。引数リストは，引数がなければ必要がありませんが，（）はその場合でも必要です。

Subプロシージャの呼び出し方法には，２通りの方法があります。１つはCall文を使う方法と使わない方法です。

例

下記のようなExample 1 というSubプロシージャを呼び出します。

```
Sub Example1(a As Integer)
    a = a ^ 2
End Sub
```

Call文を使った呼び出しでは，引数を（）で括る必要があります。

```
Call Example1(a)     ' Call文を使った呼び出し
Example1 a           ' Call文を使わない呼び出し
```

VBAのSubプロシージャは，自分自身を呼び出す再帰呼び出しが可能です。再帰呼び出しの例は，次のFunction文で示します。

2.13 Function文

Function文の構文は下記のとおりです。Function文はFunctionプロシージャを作るときに使います。Functionプロシージャと Subプロシージャの違いは，Functionプロシージャには戻り値がありそれによって呼び出したプロシージャに計算結果などの値を戻すことができるという点です。もちろん，Subプロシージャでも引数を使うことによって値を戻すことはできますが，Functionプロシージャの場合はそれ以外に戻り値という形で呼び出したプロシージャに値を戻すことができるのです。

```
Function 関数名（[引数1,引数2,…,引数n]）[As 型]
        …
        …
    [関数名 = 戻り値]
        …
End Function
```

例1　階乗の計算を行う関数です。

```
Function Factorial(n As Integer) As Long
    Dim t As Long
    Dim i As Integer

    t = 1
    For i = 2 To n
        t = t * i
    Next i

    Factorial = t    '計算結果を戻り値として戻している

End Function
```

Functionプロシージャの呼び出しは，下の例のように関数名の次に（）で括って引数を記述します。引数がない場合でも（）は必要です。

例2　関数の呼び出し方

例1の関数を呼び出しています。

```
a = Factorial(5)
```

FunctionプロシージャもSubプロシージャと同様に，再帰呼び出しができます。再帰呼び出しは，自分自身を呼び出すものです。次の例3は，例1の階乗を行う関数を再帰呼び出しの形に書き換えたものです。

例3　再帰呼び出しによる階乗の計算

```
Function Factorial(n As Long) As Long
' 再帰呼び出しによる階乗の計算

    If n = 0 Then
        Factorial = 1
    Else
        Factorial = n * Factorial(n - 1)
    End If

End Function
```

再帰呼び出しは，プログラムの実行効率から考えると必ずしも良くありませんが，再帰呼び出しを使ったほうが，よりわかりやすいプログラムになるアルゴリズムも多いので，うまく使い分ける必要があります。

自分で関数を作成する場合は，Function文を使いますが，多くの便利な関数がVBAで用意されています。また，ワークシートで使われる関数の一部はVBAの中で利用することもできます。ワークシートで使われる関数をワークシート関数と呼びますが，その使い方は標準のVBA関数と異なります。

ワークシート関数は次のような形で呼び出します。

```
Application.WorksheetFunction.関数名（[引数]）
```

例

```
mean = Application.WorksheetFunction.Average(Range("B2:B51"))
```

上記の例では，セルB２～B51までの数値の平均を，ワークシート関数を使って求めています。このときにRange（"B2:B51"）はRangeオブジェクトといい，セル範囲を指定するときに使います。ワークシート関数を使う場合は，必要になることが多いのでよく覚えておいてください。

参考文献

アスキー書籍編集部　編 1999　*Excel VBA 2000　リファレンス*　アスキー　東京

Dixon,M.R. & MacLin,O.H. 2003 *VisualBasic for Behavioral Psychologists.* Context Press Reno

市川伸一，　矢部富美枝編　1985　*パーソナル・コンピュータによる心理学実験入門*　ブレーン出版　東京金子秀彬，吉田俊郎，伊田政司，森山哲美　1990　*心理学に必要なコンピュータ技術*　北樹出版　東京

北村英哉，坂本正浩編　2004*パーソナル・コンピュータによる心理学実験入門：誰でもすぐにできるコンピュータ実験*　ナカニシヤ出版　京都水野りか　2004　*Webを介してできる基礎・認知心理学実験演習*　ナカニシヤ出版　京都

野沢晨　1986　*パソコンBASIC心理学実験*　東海大学出版会　東京

田中良久　1975　*BASIC入門　行動科学のためのコンピュータ・プログラミング入門*　東京大学出版会　東京

吉村浩一，山上暁　1983　*BASIC入門*　ナカニシヤ出版　京都

—2006.6.19受稿—