ＩＣＴ活用研修
エクセルマクロ

栃木県総合教育センター
研究調査部　情報教育支援チーム

目　次

【参考資料】

Microsoft Excel ヘルプ
Microsoft Visual Studioホームページ、https://www.visualstudio.com/ja/
猫とエクセル、http://www.geocities.co.jp/excelgame/
エクセルVBAのクラスモジュールを利用しよう、http://www.geocities.jp/tomtomf/
ExcelVBAを学ぶならmougモーグ、http://www.moug.net/
Office TANAKA、http://officetanaka.net/
いつも隣にITのお仕事、http://tonari-it/pcom/
BASIC MZ-80 SERIES、SHARP株式会社

第１部
1 はじめに
1.1 マクロとは

従来「マクロ」とは、プログラム中の文字列を、あらかじめ定義された規則に従って置換するものであり、Ｃ言語やアセンブラ言語などで用いられていた。これらの言語では、何度も利用されるような一連の処理であっても、多くの記述が必要であり、これをマクロとして複数の命令をまとめることによって、簡略に記述できるようにしたり、プログラムを分かりやすくしたりしていた。これらのことから、現在は、「一連の処理をまとめたもの」を、「マクロ」と呼ぶようになっている。

エクセルにおける「マクロ」とは、このように、エクセルの一連の処理をまとめて自動化する機能のことであり、「Visual BASIC」というプログラミング言語によって記述される。

1.2 Visual Basic

Visual Basic(以下Ｖ Ｂ(ブイビー))は、マイクロソフト社によって作られた言語であり、BASIC[1]言語を基にしたプログラミング言語である。Ｃ言語を基にした Visual C などと同様に、本来の VB は、アプリケーション･プログラムを作成するための言語環境であり、実際に「Microsoft Visual Basic」として販売されていた。現在も「Microsoft Visual Studio 2017」の一部として販売されており、また、無償版である「Microsoft Visual Studio 2017 Community」は、マイクロソフトのホームページからダウンロード[2]できる。アプリケーションを作成する場合は、これらの環境を利用することになる。

VB は、バージョンや環境によって種類があり、上記のようなアプリケーションを作成するためのバージョンとして、1998 年に発売された VB 6.0 やその後の VB．NET(ドットネット)、VB 2005(2005 版から.NET という名前は付かなくなった)、VB 2010、VB 2013、VB 2015、最新版の VB 2017 などがある。

【資料 1】

一方、マクロを作成するための VB は、「**Visual Basic for Application**」（**VBA**）と呼ばれる[3]。また、VB はホームページの記述等でも利用され、この VB は、「Visual Basic Script」(VB Script)と呼ばれている。いずれも VB の簡易版として実装されており、特に VBA は、Excel をはじめ、Word、PowerPoint、Access などでも利用され、Excel や Word などのアプリケーションが異なっても、ほぼ同様の記述方法でプログラミングが可能である（8.4 参照）。

1.2.1　ヘルプの利用

VBのヘルプは、キーボードのF1キーを押すことによって閲覧可能である(**図 1**)。

このヘルプには、エクセルで使われるマクロ言語の説明をはじめ、プログラミングのヒントなど、非常に参考になる内容となっている。また、マクロ中で使用されている特定のキーワード上にカーソルを置いたうえで F1 キーを押すことによって、そのキーワードに関するヘルプを閲覧することもできる。本研修でも、指示がなくても積極的に参照していただきたい。

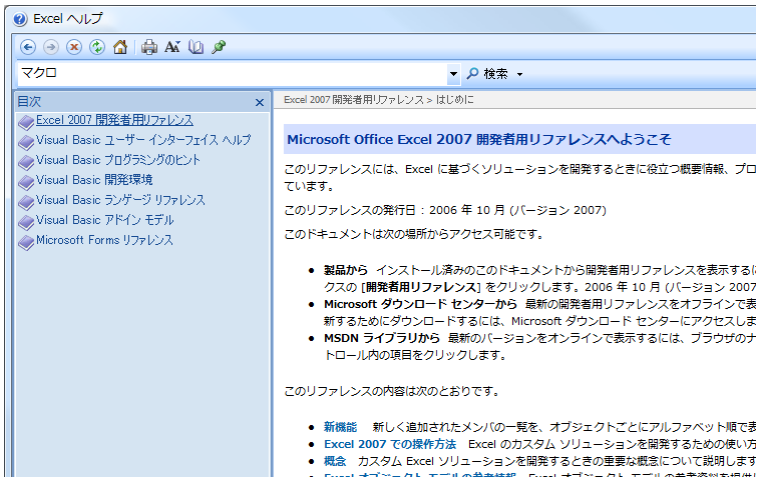


図 1　VB のヘルプ画面

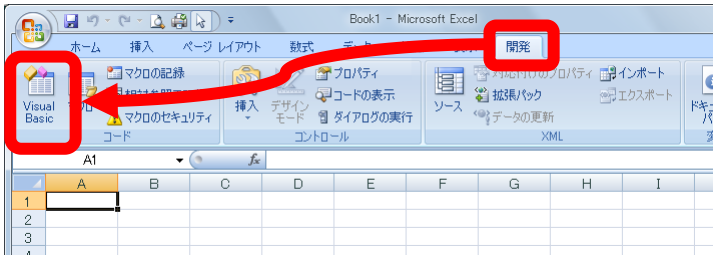
---

[1] 言語としての BASIC は、「Beginner's All-purpose Symbolic Instruction Code」(初心者用のあらゆる目的に使える、記号命令コード)の略であり、すべて大文字で記述される。

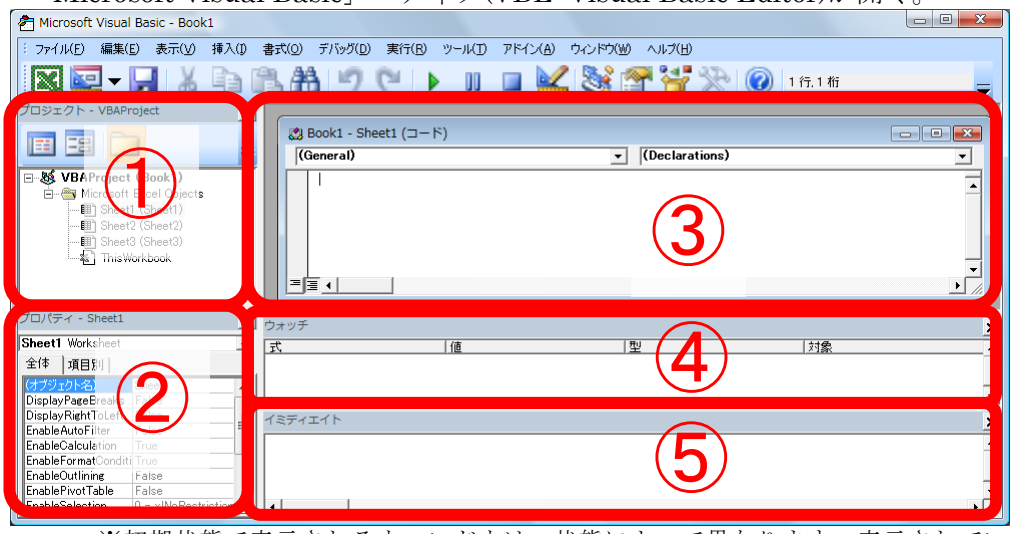
[2] 30 日以上利用する場合は無償のユーザ登録が必要

[3] 似たような言葉で「VBE」という言葉も用いられるが、これは、Visual Basic Editor、すなわち後ほど述べる「Microsoft Visual Basic」のエディタ（編集）環境を示す。

(1)　開発環境の起動（エクセルを**新規作成**で開き、「開発」タブをクリックし、「Visual Basic」をクリックする（「開発」タブが表示されていない場合は、**【資料 2】**の手順にて表示することができる)。



(2)　「Microsoft Visual Basic」エディタ(VBE: Visual Basic Editor)が開く。



※初期状態で表示されるウィンドウは、状態によって異なります。表示されていない場合は、メニューの「表示」から開くことができます。

①プロジェクトエクスプローラ(Project Explorer)
　現在マクロを作成しているブックの情報が表示される。
②プロパティウィンドウ(Property Window)
　現在選択されているプロジェクトやシートのプロパティ（情報）が表示される。
③コードウィンドウ(Code Window)
　マクロプログラムのコードが表示される。
④ウォッチウィンドウ(Watch Window)
　実行中のマクロで使用されている変数の値を監視する。
⑤イミディエイト ウィンドウ(Immediate Window)
　命令を直接実行するためのウィンドウである。

1.2.2　イミディエイト ウィンドウ

イミディエイト ウィンドウを利用することによって、簡単な命令を実行することができる。イミディエイト ウィンドウは、Ctrl + G によって、表示／非表示が切り替えられる。ここでは、このウィンドウを利用し、VB の基礎となる操作を試してみる。なお、イミディエイト ウィンドウに入力した内容は、**保存されない**ので注意が必要である。

例１　イミディエイト ウィンドウの任意の場所でクリックしてカーソルを置き、以下のように入力し、Enter を押す。何が表示されたか？

```
msgbox　"こんにちは、世界"
```

※「msgbox」と「"」は半角、「こんにちは、世界」は全角で入力する。

例２　以下のように**すべて半角**で入力し、Enterを押すと何が表示されるか？

```
?  3+5
```

「?」は、「print」という BASIC の命令の省略形で、「print 3+5」と入力しても同様の結果になる。print 命令は、そのあとにあるものを処理して表示する。

例３　以下のように入力し、Enterを押すと何が表示されるか？例２との違いは何か？

```
?  "3+5"
```

「"」(double quotation：ダブル クォーテーション）で文字や数字をくくると、それらの文字は**文字列**として認識され、処理される。文字列には、０文字や１文字の場合も含まれる。特に、０～９は、値としての「数値」と、文字列としての「数字」で分けて考える必要がある。

例４　以下のように入力し、Enterを押すと何が表示されるか？

```
a=3 : ? a
```

「:」(colon：コロン）をセパレータとして用いることによって、１行に２つ以上の命令を記述すること（マルチ ステートメント）が可能である。この場合は、「a=3」という命令と、「? a」という命令の２つをコロンでつないでいます**【資料 3】**。「a」（変数）については、後述する。

例５　以下のように入力し、Enterを押すと何が表示されるか？

```
a=4: ? "a"
```

例６　以下のように入力し、Enterを押すと何が表示されるか？

```
? ActiveWorkbook.Name
```

「ActiveWorkbook.Name」は、現在のエクセルファイルの名前を保持するプロパティであり、?によって、そのファイル名を表示することができる。「名前を付けて保存」等を行ってない場合には、初期設定の「Book1」がファイル名となる。
　なお、すべて小文字で入力しても、同様の結果となる。

例７　以下のように入力し、Enterを押すと何が表示されるか？

```
? 2 * 2 + 2 / 2 * (2 + 2)
```

計算は、式の優先順位を判定して、計算する。

例８　以下のように入力し、Enterを押すと何が表示されるか？

```
? sqr(2)
```

平方根を求める場合、Excel では SQRT 関数を使うが、VB では SQR 関数を使う。このように、同じ動作をする場合でも、Excel の関数名と VB の関数名が異なる場合があるので、注意が必要である。

例９　あなたは、今年の元旦、公衆電話を掛けるために、友達から１０円借りた。１０円ならばすぐに返せると思ったあなたは、友達と「１日１割の複利」で返すとの約束をした。次の年の元旦にこのことを思い出したあなたは、友達にいくら返せばいいか？

```
a=10 : for i=1 to 365: a = a + a*0.1 : next : ? a
```

「1.2E+3」のような E を用いた表記は「指数表示」と呼ばれる。1.2E+3 の場合は、$1.2\times10^{+3}$（=1.2×1000=1200、すなわち、小数点を右に**３つ**移動する）を意味する。

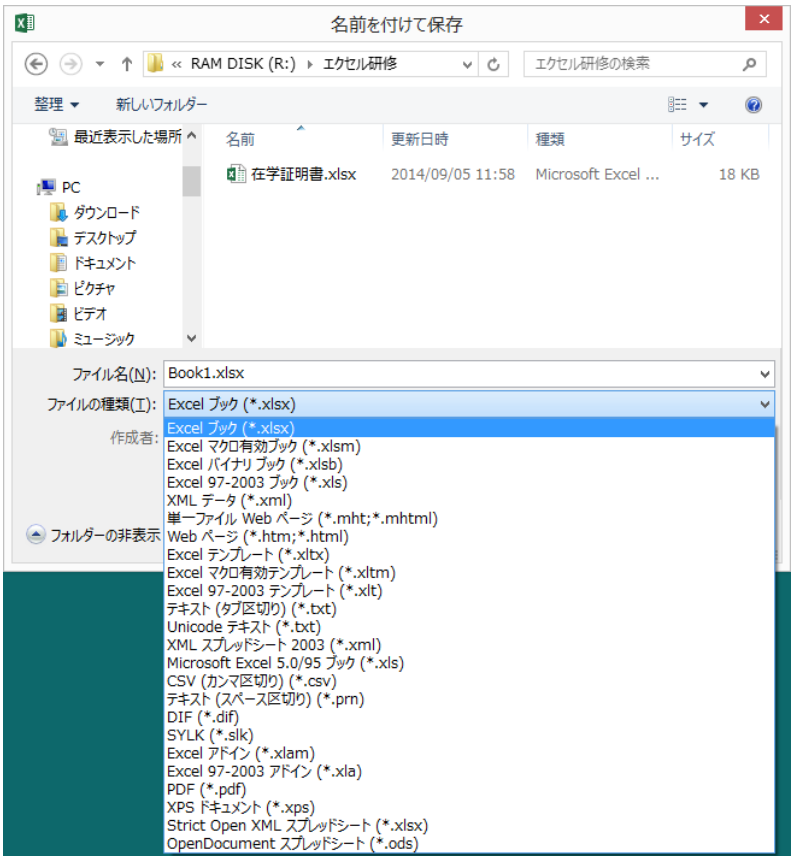
上記の式は、５つの命令からなり、10 が最初に借りた金額、365 が１年の日数である。a=a+a*0.1 で、１割(0.1)の金額を複利になるように加算している。最後の? a で答えを表示している。

なお、このような複利計算は Excel 関数に財務関数として用意されており、この例では、以下の式で求めることができる。

```
=FV(0.1, 365, , -10)
```

1.3 マクロの保存

マクロが完成したら、ファイルとして保存する。VBE にも保存のメニューはあるが、ここでは、エクセルのシート側で保存する。Excel 2007 以降で名前を付けて保存をすると、初期状態ではファイルの種類が「Excel ブック」(拡張子:xlsx)となり、このままではマクロは保存されずに破棄されてしまう。マクロも含めて保存する場合には、ファイルの種類として、「Excel マクロ有効ブック」(拡張子:xlsm)を選択しなければならない。



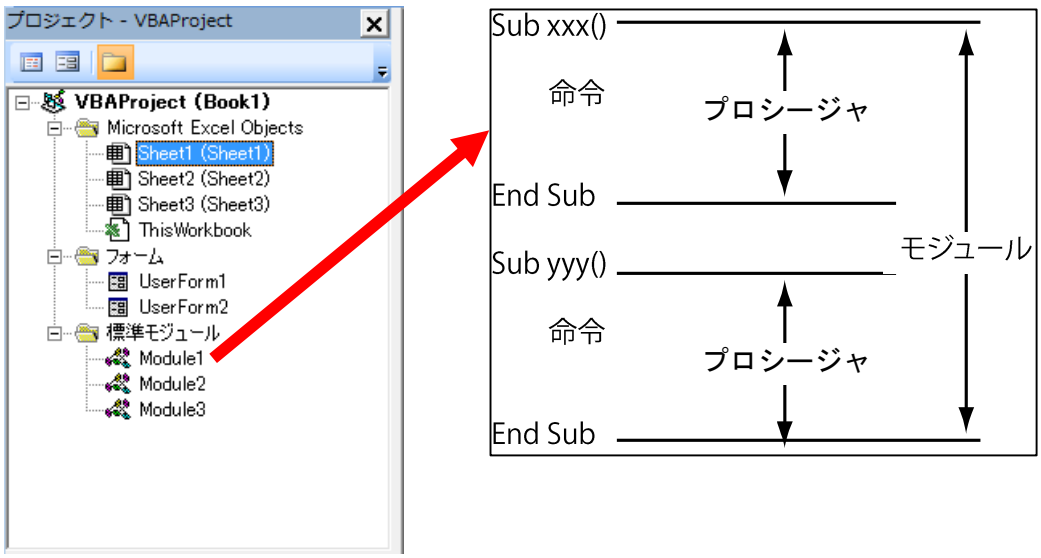
問 1　ここまで作成したブックをファイル名「**prog.xlsm**」として保存しなさい。

2 プログラミング

本研修で取り上げるマクロを含め、一般的に「プログラム[4]」を作成しようとする場合、その言語で使われる命令語やその文法を習得することも必要であるが、プログラムを作成するために最初にすべきことは、「データ構造」と「アルゴリズム」の検討である。「データ構造」とは、処理で使用するデータ群を、どのような形式（構造）でコンピュータの中に記憶させるかを定義することであり、「アルゴリズム」とは、そのデータを用いた処理の手順を定義することである。ここでは、エクセルのマクロを作成するに当たって必要となる、エクセルのファイル構造と、基本的なデータ構造、一般的なアルゴリズムについて説明する。

2.1 エクセルのファイル構造

エクセルでは、１つのファイルを「ブック」と呼び、その中に複数の「シート」が存在している。それぞれのシートは、1 つのオブジェクト（2.3.2 参照）として存在し、そのほかに、ユーザーフォーム（7 章参照）、マクロを記述する標準モジュールなどから構成される。



フォームや標準モジュールには、それぞれにコード（マクロ）を記述することができ、それらは独立して管理される。これのように１つのオブジェクト等に記述されたマクロを、「モジュール」と呼ぶ。

１つのモジュール内には、「プロシージャ」と呼ばれる１まとまりのマクロを複数記述することができ、それぞれをお互いに呼び出しながら処理を実行していく。

2.2 データ構造

データ構造には変数、リンクリスト、二分木などがある[5]。ここでは、データ構造のうち、一般的によく利用される変数（スカラー変数）を取り上げる。

エクセルにおいて、処理に使用する文字や数値を保持するためには、シートのセルに値を保存する場合と、変数（配列を含む）を利用する場合とが考えられる。シートのセルに保存した場合、マクロの実行中にもその値を確認できるが、実行速度は非常に遅くなる。一方、変数に保存した場合、マクロの実行中にその値を直接見ることはできない[6]が、実行速度は非常に速い（3.3 節の **問 3-8** 参照）。

ここでは、エクセルにおける変数の利用方法について説明する。

---

[4] 本テキストでは、一般的な意味での「プログラム」と、エクセルマクロに特化した内容での「マクロ」という言葉を使い分ける。

[5] BASIC 言語では、リンクリスト等を実現するためのメモリアドレス（コンピュータのメモリ上のアドレス。セルのアドレスとは異なる）を取得したり参照したりすることは一般的ではない（VB でのメモリアドレスは、VarPtr 関数を使って取得できる）。

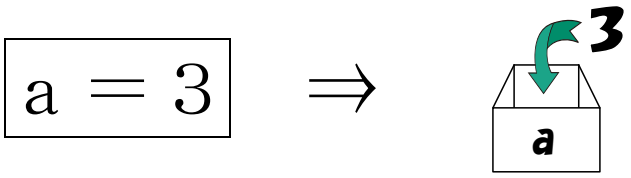
[6] 変数の値を確認したい場合は、4.3 で説明するウォッチウィンドウやイミディエイト ウィンドウを利用して値を表示させる必要がある。

2.2.1　変数と代入

コンピュータのプログラムで使われる変数は、数学で使われる変数と違った考え方をしなければならない。プログラムでは、数値を収容するための「箱」（メモリ）がコンピュータの内部にあり、この箱につけられた名前を**変数**という。また、式で使用される「＝」（イコール[7]）も、数学とは異なり、「代入」の意味で用いられる。

```
a = 3
```

という式（命令）は、ａという名前が付けられた箱に、３という数値を入れなさいという意味になる。



変数につける名前（変数名）には、以下のような決まりがある。

- １文字目に数字は使えない（アルファベットか全角文字を使う）。
- 255 バイト以内（半角で 255 文字、全角で 127 文字）。
- 予約語（for, if, msgbox, DoEvents など）は使えない。
- スペースや次に説明する型宣言文字など、使えない**半角記号**がある（「?」は使えないが、「_」は使える）。

問 2-1　次の文字列のうち、変数名として使えるものに○、使えないものに×をつけなさい。

- abcdefg
- 3.141592
- クラスの平均
- database_読み込み
- 3nen
- 三年
- sannen?
- さんねん？
- jugemujugemugokounosurikirekaijarisuigyonosuigyoumatuunraimatufuuraimatu

(72文字)

2.2.2　変数の種類

VB で使える変数の種類は、**【資料 4】**のようなものがある。この中でよく使われるものとして、Integer 型、Double 型、String 型などがある[8]。変数を使用するときには、Dim 命令[9]を使い、変数名とその型を宣言する。

```
Dim 変数名 as 型名
Dim 変数名 as 型名, 変数名 as 型名, …
```

例

```
Dim i as Integer                整数型変数 i
Dim s as String                 文字列型変数 s
Dim pai as Double               倍精度浮動小数点型変数 pai
Dim 点数 as Byte                バイト型変数　点数
```

「型宣言文字」は、変数や定数等の後に付けて、そのデータ型を表す。例えば、文字列型変数ｓは、「ｓ＄」という名前を使用することにより、その変数が文字列型であることが明示できる。このように、変数を使用するときには、そのデータ型を明示するため、できるだけ型宣言文字を付加したほうがよい**【資料 5】**。なお、Dim 命令や型宣言文字で変数の型を明示しないで使用すると、バリアント型になる。さらに、複数の同一型の変数をまとめて扱える「配列」も利用できる**【資料 6】**。

また、変数は、宣言する場所によって、その参照できる範囲（スコープ）が決定する**【資料 7】**。

---

[7] 記号「＝」は、「equal」（イコール）と読むが、プログラムの意味的に「becomes」（ビカムズ）と読む場合もある。

[8] 一般的にはサイズが増えるほどメモリを消費し、処理も遅くなる。しかし、現在の一般的なコンピュータ（32 ビット以上のＣＰＵを持つコンピュータ）では、メモリも十分に搭載され、また、Long 型および Double 型を直接処理するためのハードウェアが内蔵されているため、無理に Integer 型や Single 型を使う必要はない。

[9] Dim は Dimension(次元)の省略。以前は、配列を宣言する命令であったが、VB では変数の宣言一般に用いられるようになった。

2.3 アルゴリズム

アルゴリズムとは、問題を解く手順を表したものであり、プログラムは、このアルゴリズムを記述したものである。アルゴリズムを視覚的に表すために、しばしばフローチャートやデータフロー図、状態遷移図、PAD などが使われ、処理の流れを考えたり、整理したりする場合に有用となる。フローチャートで用いられる記号を**【資料 8】**に示す。また、アルゴリズムで用いられる構造を

【資料 9】に示す。

2.3.1　実行順序

例えば、三角形の面積を求める場合を箇条書きにすると、以下のような手順となる。

① 底辺を決める
② 高さを決める
③ 底辺と高さをかける
④ ２で割る

このとき、①と②はどちらを先に実行しても問題ないが、③を実行するためには、①と②を先に実行しておかなければならない。このように、処理の流れは、その順番を考えて決めていかなければならない。この処理をフローチャートで表すと、右図のようになる。

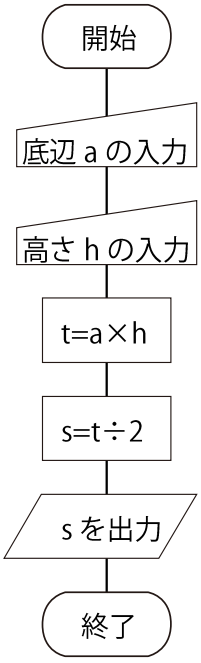


図 2　フローチャートの例

問 2-2　長方形の面積を求める手順を、箇条書きで書きなさい。また、フローチャートを使って書きなさい。変数は、面積 s、高さ h、幅 w とする。

問 2-3　円の面積を求める手順を、以下の公式を使って箇条書きで書きなさい。また、フローチャートを書きなさい。ただし、円周率は 3.14 とし、変数名は、面積 s、半径 r とする。

$$円の面積 = 円周率 \times （半径）^2$$

問 2-4　台形の面積を求める手順を、以下の公式を使って箇条書き及びフローチャートで書きなさい。変数名は、面積 m、上底 j、下底 k、高さ t とする。

$$面積 = \frac{1}{2} \times (上底 + 下底) \times 高さ$$

2.3.2　オブジェクトとプロパティ

VB はオブジェクト指向言語である。オブジェクト指向とは、コンピュータ内で扱われるあらゆるもの（例えば、マウスやキーボード、ディスプレイなどのハードウェアの装置から、アプリケーションの中で使われるエクセルのシートやセルなど）を、「オブジェクト」（モノ）として定義し、それらのオブジェクトとメッセージをやり取りすることによって、処理を実行していく考え方である**【資料 10】**。

この考え方によって、あらゆるオブジェクトをブラックボックスとして定義して「部品化」することにより、再利用することが容易になった。プログラムを記述する場合は、それらの部品の必要なところだけを修正し、それ以外のところは気にせずに利用することができる。

オブジェクトに対し、その状態や性質を表すものを「プロパティ」と呼ぶ。このプロパティによって、そのオブジェクトの色やサイズ、画面上の位置、チェックボックスが選択可能かどうか、などが決まる。さらに、オブジェクトの動作を記述したものを「メソッド」と呼ぶ。オブジェクトには、あらかじめ決められて内蔵されている動作もあるが、ユーザが新たなプロシージャ（3.1 参照）やファンクション（4.4 参照）として付け加える動作もある。さらに、システムからオブジェクトに対し発せられる動作を「イベント」と呼ぶ。これは、マウスのクリックやキーの押し離し、セルの値の変更などがある。

ポイント
オブジェクト…ワークシート、セル、グラフなど、アプリケーションの要素
プロパティ …オブジェクトの属性を表す名前
メ ソ ッ ド…オブジェクトに属するプロシージャやファンクション
イ ベ ン ト…オブジェクトで認識される動作

2.3.3　アルゴリズムの例

よく使われるアルゴリズムとしては、次のようなものがある[10]。

合計・平均を求める
最大値・最小値を求める
素数を求める
最大公約数を求める
サーチ（検索）
ソート（並べ替え）

例えば、クラスからプリントを回収し、それを番号順に並べ替える（ソート）場合を考える。クラスの生徒が５名の場合と４０名の場合で、手順が異なると考えられる。

問 2-5　クラスの生徒が５名の場合、どのように並べ替えるか考えなさい。

問 2-6　クラスの生徒が４０名の場合、どのように並べ替えるか考えなさい。

[10] よく使われるアルゴリズムは、命令（関数）としてあらかじめ用意されている場合が多い。エクセルでは、シート上で使える sum、average、max などや、マクロ上で使える sort などが用意されている。

問 2-7　並べ替えが終わった後、欠席していた１名が提出した場合、どのように並べ替えるか考えなさい。

問 2-8　問 2-5 の方法で４０名のプリントを並べ替えた場合と、問 2-6 の場合とで比べて、どのくらい時間が変わるか考えなさい。

問 2-9　次の文章を読み、下の問い（問 a, b）に答えよ(情報入試研究会　第２回大学情報入試全国模擬試験より)。

次のプログラムは、「正の整数 n を入力したとき、1 から n までの整数を出力する」プログラムである。

```
n に整数値を入力
i を 1 から n まで 1 ずつ増やしながら、くり返し
    i を出力
ここまでが「くり返し」の範囲
```

たとえば、このプログラムを実行して 7 を入力した場合「1 2 3 4 5 6 7」と出力する。

下の解答群を使用して、このプログラムを表現すると、次の通りになる。

ア　ク　セ　コ

解答群

ア　n に整数値を入力
イ　j ← 0
ウ　k ← 1
エ　k ← i
オ　k ← i × i
カ　j ← j + i
キ　j ← j + k
ク　i を 1 から n まで 1 ずつ増やしながら、くり返し
ケ　j を 1 から k まで 1 ずつ増やしながら、くり返し
コ　ここまでが「くり返し」の範囲
サ　もし、i が偶数ならば
シ　もし、i が偶数でないならば
ス　ここまでが「もし」の範囲
セ　i を出力
ソ　j を出力

注意事項

・変数は、プログラム実行開始時点で値が－1 となっているものとする。
・"**くり返し**" で終わる行には、それに対応する"**ここまでが「くり返し」の範囲**" の行が必要である。
・"**もし**" で始まる行には、対応する"**ここまでが「もし」の範囲**" の行が必要である。
・解答は、解答群の記号を左から詰めて記入すること。

問 a) 正の整数 n を入力したとき、1 から n までの数それぞれの 2 乗の和を出力するプログラムを、上の解答群の行を必要なだけ並べて作成せよ。解答群にある行は何回使ってもよい。たとえば 4 を入力すると、1 + 4 + 9 + 16 = 30 なので、「30」と出力する。

問 b) 正の整数 n を入力したとき、1 から n までの数を順番に、奇数については 1 回、偶数についてはその数の回数だけくり返して、出力するプログラムを、上の解答群の行を必要なだけ並べて、次のプログラムを作成せよ。解答群にある行は何回使ってもよい。たとえば 4 を入力すると、「1 2 2 3 4 4 4 4」と出力する。

2.3.4　プログラムの最適化

最終的な結果が正しければ、それは「正しいアルゴリズム」と考えられるが、その処理が 1 秒で終わる場合と、1 日かかる場合とでは、1 秒で終わるほうが「よりよいアルゴリズム」と考えられる。また、1 度しか使わないプログラムであればプログラムが見難くても処理が速いほうがいいが、長期間に渡って同じプログラムを利用する場合や多くの人が変更することが前提となる場合は、分かりやすいプログラムのほうが修正や保守がしやすい。このように、同じ処理を行う場合でも、「よいアルゴリズム」や「よいプログラム」が変わることがある。

速度のみを考えた場合、データをセル上に配置するより、変数として保存しておいたほうが速くなる。特に、データの並べ替えなど、同一のデータを何度も読み書きする場合には、セル上に置いて処理するよりも、最初に配列変数に取り込み、それを変数上で並べ替えて、セルに戻すほうが、速度はかなり速くなる。

プログラムが完成した後、その処理が遅い場合は、プログラムの「最適化」を考えてみるとよい。特に

・繰り返し部分の内側を修正する
・セルの値を、変数に保持して処理する

などの修正によって、劇的に速くなることがある。

3 基本的な処理

3.1 メッセージボックスの表示

最初に、マクロの簡単な例として、メッセージボックスを表示する。

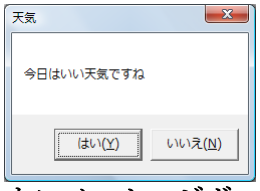


図 3　表示したいメッセージボックス

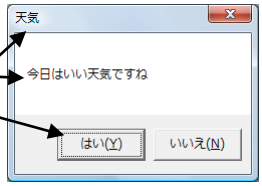
MsgBox 命令の書式

MsgBox プロンプト, ボタン, タイトル

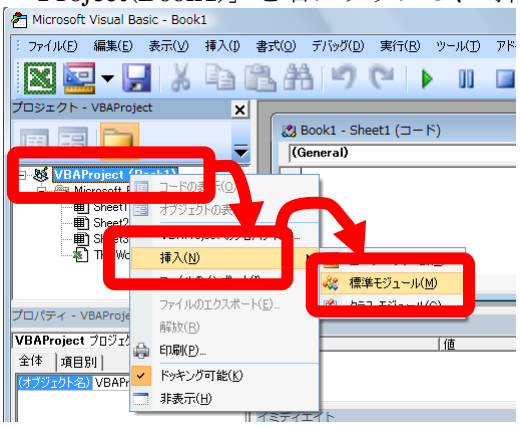
プロンプト：中に表示する文字列

ボタン：表示されるボタンの種類や数

タイトル：上部に表示する文字列



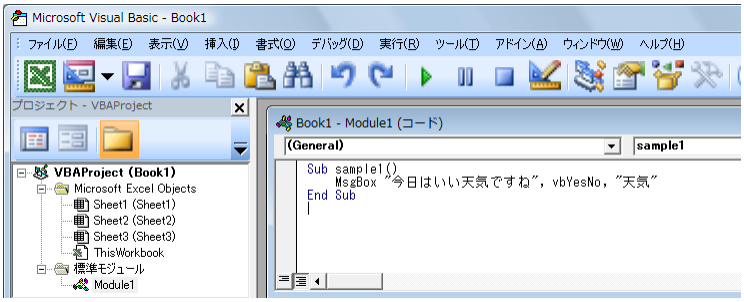
(1) 配布したファイルから「prog1.xlsm」を開き、プロジェクトウィンドウの「VBA Project(Book1)」を右クリックし、「挿入」→「標準モジュール」をクリックする。



(2) コードエディタから、次のように入力する。

```
Sub sample1()
    MsgBox "今日はいい天気ですね", vbYesNo, "天気"
End Sub
```

※キーワードとなっているアルファベットの大文字・小文字は自動で修正される。

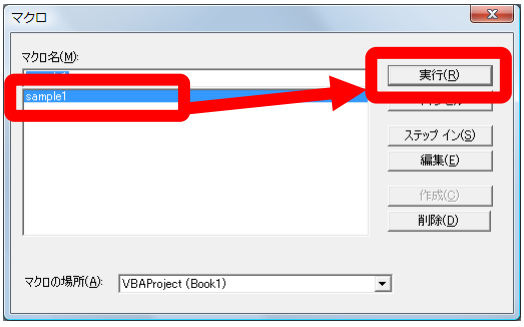


「Sub」は、サブルーチン(sub-routine)の略で、ひとまとまりの処理（プロシージャと呼ぶ）を定義する。ここでは、「sample1」という名前を付けたひとまとまりの処理（ここではメッセージボックスを表示する１行だけ）を定義した。また、「End Sub」は、Sub の定義がここまでであることを示す。sample1 の後ろのカッコについては、4.4 で説明する。

コードエディタの最初に、「Option Explicit」という記述が自動的に挿入されることがある。この命令文は、そのモジュールで使用する変数を、明示的(Explicit)に宣言しなければ使えないようにオプション(Option)を設定するものである。標準状態の Visual Basic では変数を宣言しなくても使用できるが、この設定によって、変数名のスペルミスによる間違いを防ぐことができるようになる。もし挿入されていない場合は、最初に設定しておいたほうがよい。
設定方法は、VBE メニューから「ツール」－「オプション」－「編集」タブ内の「変数の宣言を強制する」にチェックを入れる。

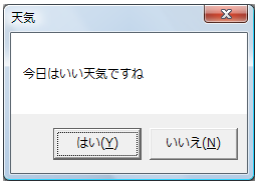
(3) カーソルをSub～End Subの間に置き、実行ボタン をクリックする[11]。

Sub～End Sub の外にカーソルがある場合は、実行すべきマクロが特定できないため、実行するマクロ名を選択するダイアログが表示される。今回は「sample1」を選択し、「実行」をクリックする。

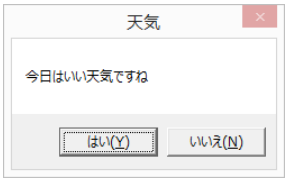


※左図の手順は必要ない場合もある。

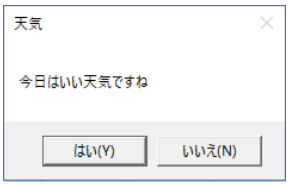
(4) メッセージボックスが表示される。



Windows Vista の場合



Windows 8.1 の場合



Windows10 の場合

3.2 セルの操作

エクセルでは、基本操作としてセルに値をセットしたり、セルの値を読み出したりする必要がある。これらの操作について説明する。

特定のセルを指定するためには、「シート名」と「セルの番地」（アドレス）を指定する必要がある[12]。これらを指定するのが、Sheets オブジェクト、Range オブジェクトである。また、対象となるセルの**値**を表すプロパティは、Value という名称がつけられている。

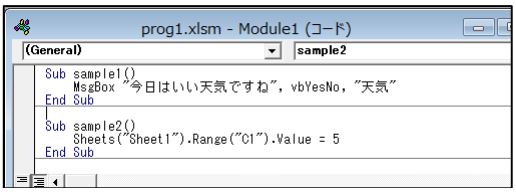
```
Sheets, Range, Value
      Sheets("xxx") : シート名の指定
      Range("C1")   : セルの範囲指定

      Value : セルの「値」の読み出し／書き込みを指定する。
              (セルの色や罫線などを指定することもできる)
```

[11] マクロを実行すると、シートの状態を元に戻すこと(Undo)はできないので、注意する。
[12] 正確には、「ブック名」も Workbooks オブジェクトを使用して指定する必要がある。

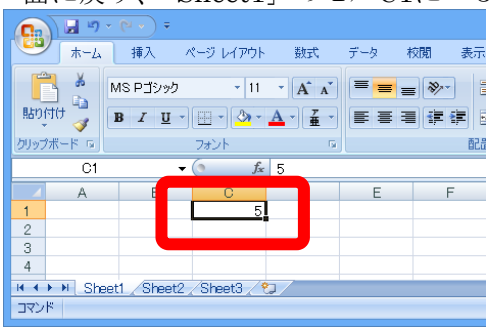
3.2.1　セルの値のセット

(1)　先ほどのsample1のEnd Subに続いて、次のように入力する。

```
Sub sample2()
    Sheets("Sheet1").Range("C1").Value = 5
End Sub
```



(2)　sample2内にカーソルを合わせて実行ボタンを押し、マクロを実行する。エクセルの表の画面に戻り、「Sheet1」のセルC1に「５」が入力されていることを確認する。



セルを指定する場合の記述方法としては、上記の Range のほかに、以下のような記述方法もある。

①　Sheets("Sheet1").Range("C1").Value = 5
②　Sheets("Sheet1").Cells(1,3).Value = 5
③　Sheets("Sheet1").[C1].Value = 5

ここで、②の Cells のカッコ内は、**（行，列）の順**、すなわち（縦，横）の順である。

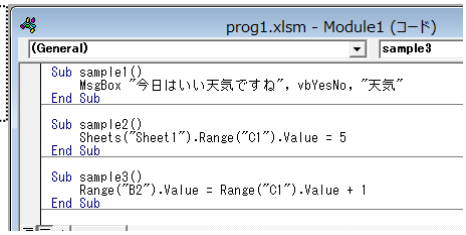
Sheet1 がすでに選択されている（一番前に出てきて表示されている＝アクティブになっている）場合、Sheets(...)の記述を省略することができる。また、値をセルに代入する場合、.Value を省略することができる。よって、以下のような記述をすることも可能である。

①　Range("C1") = 5
②　Cells(1,3) = 5
③　[C1] = 5

3.2.2　セルの値の読み出し

(1)　sample2に続いて、次のsample3を入力する（「…①」は説明のための記述であり、入力する必要はない）。

```
Sub sample3()
    Range("B2").Value = Range("C1").Value  + 1   …①
End Sub
```



①の命令は、"＝"の右辺の値を計算し、左辺に代入する。右辺の Range("C1").Value は、sample2のところで、セル C1 に「５」が代入されたので、式は以下と同じになる。

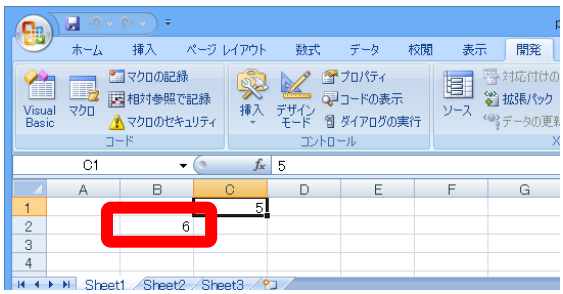
```
Range("B2").Value =  5  + 1
```

さらに、右辺の値を求めると、以下のようになる。

```
Range("B2").Value =  6
```

よって、セルB2には、「６」が設定される。

(2)　sample3を実行すると、エクセルの表は、次のようになることが確認できる。



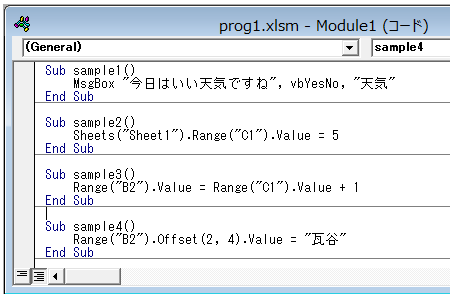
3.2.3　オフセットの指定

オフセット（Offset[13]）を指定することによって、特定のセルの位置を基準として、そこから指定した値だけ移動したセルを読み書きの対象とすることができる。

```
Offset(y, x)
  セルの位置を、現在の位置から相対的に指定する。
        y ：上下方向（行、Row、下が＋、上がー）を指定
        x ：左右方向（列、Column、右が＋、左がー）を指定
```

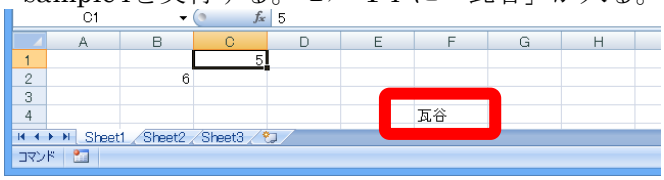
(1)　次のsample4をsample3の下に追加する。

```
Sub sample4()
    Range("B2").Offset(2,4).Value = "瓦谷"　…②
End Sub
```



②では新たに、Offset(2,4)が追加されている。これによって、Range で指定されたセル（"B2"）から、下方向に「２」、右方向に「４」移動したセル（"F4"）が対象となる[14]。また、②の右辺は、「瓦谷」という文字列を指定しているが、文字列を指定する場合には、このように必ずダブルクォーテーション（"）でくくらなければならない。

(2)　sample4を実行する。セル"F4"に「瓦谷」が入る。



問 3-1　セル C3 に 3 を入力するプロシージャを作成しなさい【プロシージャ名は「Sub ex31」とする。以下同様】。

問 3-2　セル D3 に 5 を入力するプロシージャを作成しなさい【Sub ex32】。

[13] マクロで使用する offset([列数],[行数])は、エクセルの表の中で使用する offset(参照,行数,列数,[高さ],[幅])関数とは書式が異なる。この他にも、マクロの命令と表の関数名で同じ名前で書式が異なったり、結果が異なったりするものが多数あるので注意が必要となる。

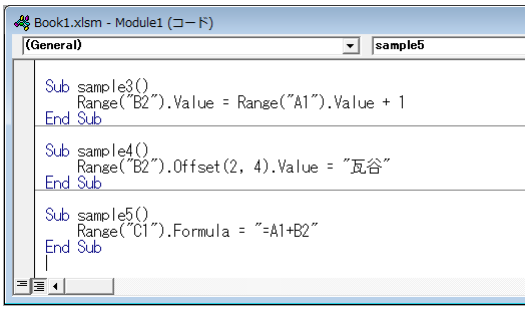
[14] 数学では座標は(x, y)の順番であるが、VB では(y, x)の順番となる。また、コンピュータでは一般的に下向きが正となることが多い。

問 3-3　セルE3に、セルC3の値とセルD3の値の和を入力するプロシージャを作成しなさい【Sub ex33】。

3.2.4　式のセット

セルに**式**を設定する場合は、「～.Formula」メソッドを使い、以下のような記述になる。

```
Sub sample5()
    Range("D4").Formula = "=C1+B2"    …③
End Sub
```



③では、これまでのValue（値）の代わりにFormula（式）となっている。また、右辺は、式を示す文字列としてExcelで入力される式と同様に、「＝」で始まる式がダブルクォーテーションで囲まれて指定されている。

問 3-4　sample5の実行結果はどのようになるか。

問 3-5　問3-3をFormulaメソッドを用いて記述しなさい。

3.3 繰り返し

2.3で述べたように、「繰り返し」は、アルゴリズムの基本構造として重要なものである。VBでの繰り返しには、For～Next文またはDo～Loop文を使用する。一般的な使い分けとして、実行時に繰り返しの回数が決まっている場合にはFor文を、実行時に回数が決まっていない場合にはDo文を使用する。ここではそれぞれの場合についての演習を行う。サンプルデータとして、**prog2.xlsm**を使用する。

3.3.1　For 文

For文は、以下のような構文である。

```
For 変数 = 初期値 To 最終値 [Step 増分]
        命令群
Next [変数]
```

変　数：数を数えるための変数（カウンタ）
初期値：カウンタの最初の値
最終値：カウンタがこの値を超えたら終了となる値
増　分：繰り返しごとにカウンタに加算される値（省略可、省略時は１）

問 3-6　次のプログラムを入力し、実行しなさい。

a)
```
Sub t3_7_a()
    Dim i As Integer

    For i = 1 To 5
        MsgBox i
    Next i

End Sub
```

b)
```
Sub t3_7_b()
    Dim i As Integer

    For i = 1 To 10 step 2
        MsgBox i
    Next i

End Sub
```

c)
```
Sub t3_7_c()
    Dim i As Integer

    For i = 5 To 1 step -1
        MsgBox i
    Next i

End Sub
```

問 3-7　次の条件を満たす For 文を記述しなさい。また、問 3-を参考にしてプログラムを作成・実行し、動作を確認しなさい。

例) 変数 i を 1 から 5 まで繰り返す　　→　　For i=1 to 5

a) 変数 j を 1 から 10 まで繰り返す

b) 変数 k を 10 から 100 まで 10 ずつ増やしながら繰り返す

c) 変数 m を 10 から 1 まで繰り返す

d) 変数 n を 100 から 10 まで 7 ずつ減らしながら繰り返す

e) 変数 p を a から b まで繰り返す

f) 変数 q をセル B1 の値からセル B2 の値まで繰り返す。ただし、セル B1、B2 の値はともに整数とし、セル B1 の値＜セル B2 の値とする。

g) 変数 r をセル C1 の値からセル C2 の値まで繰り返す。ただし、セル C1、C2 の値はともに整数とし、セル C1 の値＞セル C2 の値とする。

Excel では、合計と平均を求める関数として、Sum と Average が用意されている。ここでは、この関数と同等の動作をマクロで実現する。

(1) エクセルを一度終了し、「**prog2.xlsm**」を起動する。
(2) (1)と同様に、Visual Basicエディタを起動する。
(3) プロジェクトウィンドウの「VBA Project(prog2.xlsm)」を右クリックし、「挿入」→「標準モジュール」をクリックする。

(4) 次のsample6を入力し、実行する（「…①」～「…⑩」は入力しない）。

```
Sub sample6()
    Dim s As Integer                                    …①
    Dim i As Integer                                    …②
    Dim nen As Integer                                  …③

    s = 0                                               …④

    For i = 1 To 5                                      …⑤
        nen = Range("M5").Offset(i, 0).Value            …⑥
        s = s + nen                                     …⑦
    Next i                                              …⑧

    MsgBox "合計は" & s & "年です"                       …⑨
    MsgBox "平均は" & (s / 5) & "年です"                 …⑩
End Sub
```

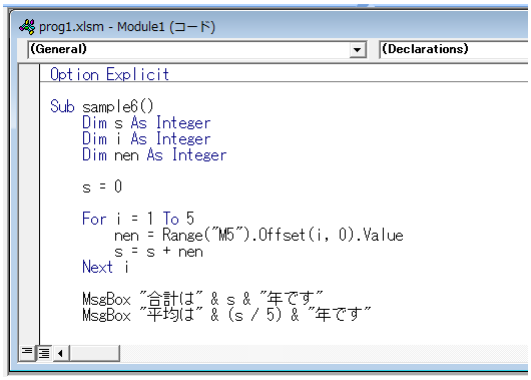
プログラムの説明

①～③：使用する変数を宣言

④：積算する変数をゼロに初期化

⑤：１人目から５人目まで繰り返し
⑥：　シートから、使用年数を取得
⑦：　使用年数を加算
⑧：ここまで、繰り返す

⑨、⑩：合計年数と平均年数を表示



データ構造

変数名	使用目的
s	合計年数を積算する
i	繰り返し回数を保持
nen	個人の使用年数を保持

3.3.2　Do 文

For 文は、対象となる回数があらかじめ分からなければ利用できないため、人数が分からない場合は、For 文の代わりに Do 文を用いる。このとき、繰り返しを終了するための条件（または、続ける条件）が必要となる。Do 文の構文は while の場合と until の場合、さらに、それらを Do の後に記述するか、Loop の後に記述するかで変わる。まとめると以下の通りとなる。

```
Do {While | Until} 条件
        命令群
Loop
  または、
Do
        命令群
Loop {While | Until} 条件

        {While | Until}：While か Until のいずれかを指定する。
                While:条件が成立している間（真の間）は繰り返す。
                Until：条件が成立するまで（偽の間）繰り返す。
        条件：処理を継続／中断する条件を数式等で指定する。
```

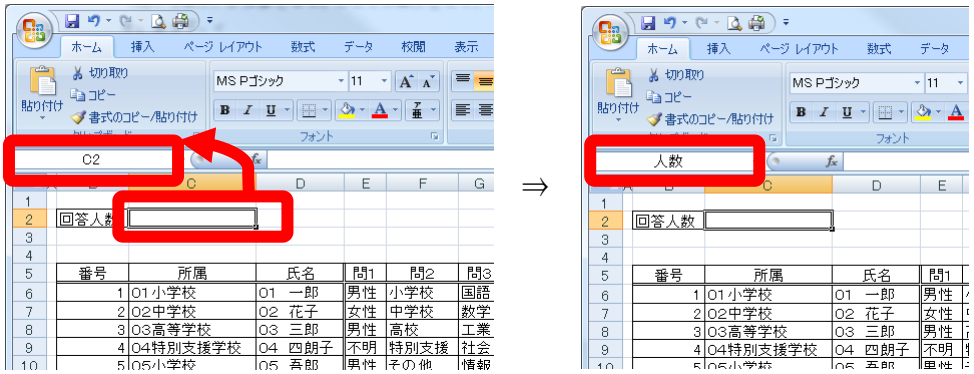
例として、「セルが空白でない間は繰り返す」とすると、次のようになる。

```
Do While Range("M6").Offset(p, 0).Value <> ""
```

「セル M6 から、下に p、右に 0 移動したセルの値」が、「""」でない(<>)間(while)は繰り返す(Do)、となる。ここで、p をゼロから始めるとすると、参照するセルはセル M6 から開始し、下に移動していくことになる。

ここでは、Do 文を用いた例として、sample6 の for 文の代わりに、Do 文を用い、人数を数えながら合計、平均を求めてみる。

(1) セルC2を選択し、名前ボックスに「人数」と入力し、Enterを押す。



このように、セルに名前を付けることによって、マクロから Range("人数")のようにして、その領域を容易に参照することができる。シートの列や行の挿入・削除を行ったとき、指定された名前はそのまま平行移動するが、マクロ内でセルの番地を”A5”のように直接指定している場合は、その部分全てを書き換えなければならない。よって、できるだけセルに名前を付けて管理したほうが保守性の点で有利となる。

(2) 次のsample7をsample6の下に追加し、実行する。

```
Sub sample7()
    Dim s As Integer, p As Integer                          …①
    Dim nen As Integer                                      …②

    s = 0: p = 0                                            …③

    Do While Range("M6").Offset(p, 0).Value <> ""           …④
        nen = Range("M6").Offset(p, 0).Value                …⑤
        s = s + nen                                         …⑥
        p = p + 1                                           …⑦
    Loop                                                    …⑧

    Range("人数").Value = p                                 …⑨

    MsgBox "合計は" & s & "年です"                          …⑩
    MsgBox "平均は" & (s / 5) & "年です"                    …⑪

End Sub
```

このマクロでは、行が短くなるように、これまで行を分けて記述していた内容を、１行にまとめている。①では２つの変数の宣言を「,」でつないで記述し、③では２つの命令を「:」でつないで記述している（1.2.2 参照）。

データ構造

変数名	使用目的
s	合計年数を積算する
p	参照している場所を示す
nen	個人の使用年数を保持

名前	参照範囲	使用目的
人数	基礎データ!C2	アンケート回答人数

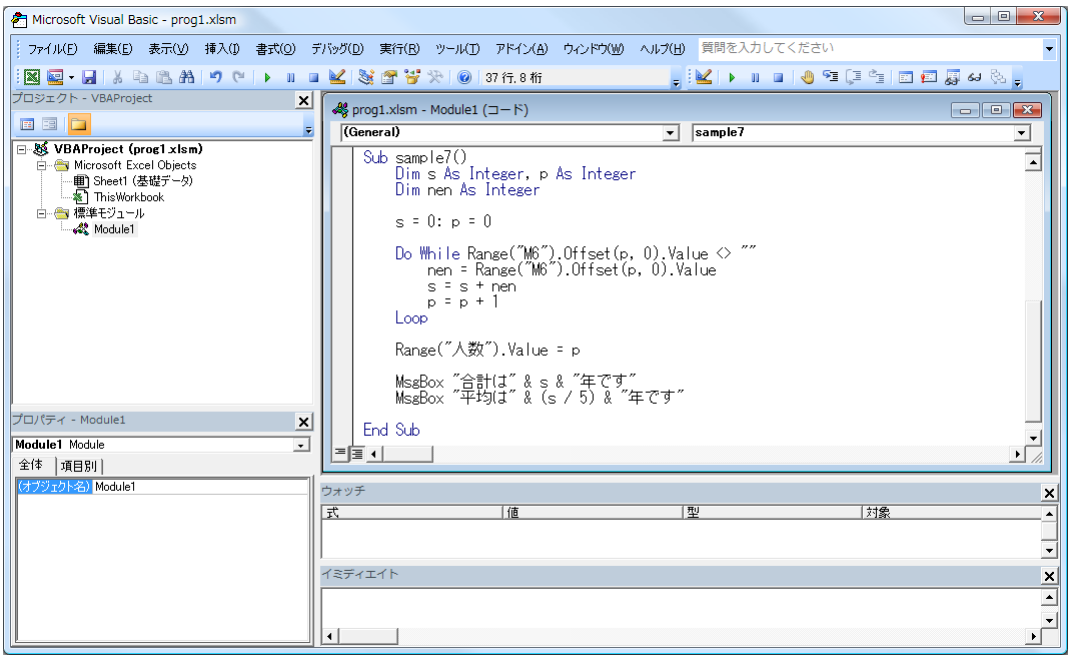


図 4　sample7 の入力画面

なお、For 文は基本的に繰り返しの回数が決まっているため（時間が掛かっても）必ず終了するが、Do 文は繰り返しの条件によっては、永遠に終わらない（無限ループと呼ぶ）ことがあるため注意する必要がある【資料 11】。

問 3-8　次のマクロをそれぞれ追加し、実行結果を比較しなさい。この結果より、値をセルに保存した場合と、変数に保存した場合、どのくらい速度が異なるか評価しなさい。

```
Sub speed1()
    Dim t As Single
    Dim i As Long


    t = Timer

    For i = 1 To 10000          '１万回
        Cells(1, 1) = 2
        Cells(1, 2) = Cells(1, 1) + i
    Next i

    MsgBox (Timer - t) & "  秒"

End Sub
```

```
Sub speed2()
    Dim t As Single
    Dim i As Long
    Dim a As Integer, b As Long

    t = Timer

    For i = 1 To 100000000      '１億回
        a = 2
        b = a + i
    Next i

    MsgBox (Timer - t) & "  秒"

End Sub
```

※プログラム中の「’１万回」「’ １億回」のような、「’ 」（アポストロフィー）の記号以降は「注釈文」として扱われ、プログラムの動作に影響はない。
（よって、「’ 」（アポストロフィー）以下の注釈については、入力しなくても問題ない）。

問 3-96　次の繰り返しを実現するための Do 文を記述しなさい。
a) セルA1の値がゼロでない間は繰り返す

b) セルA1の値がプラスの間は繰り返す

c) セルB1の値がゼロになるまで繰り返す

d) セルB1の値が100を超えるまで繰り返す

e) セルC1の値が100以下かつセルD1の値が100以下の間は繰り返す

3.4 条件分岐

条件分岐も、アルゴリズムの記述で用いられる基本的な構造の１つである。VB での条件分岐には、If 文または Select 文を使用する。

3.4.1　If 文

If 文は以下のような構文を持つ。

```
If　条件　Then　命令１　Else　命令２

または

If　条件　Then
        命令１
Else
        命令２
End If
```

条件：処理を分岐する条件を数式等で指定する。
命令１：条件が成立したときに実行される命令群
命令２：条件が成立しなかったときに実行される命令群（必要ない場合は Else とともに記述しなくてよい）

例：もし、変数aの値が３ならば、aの値を表示する

```
If a=3 Then Print a
```

問 3-7　次の条件分岐を実現するための If 文を記述しなさい。

a) 変数bの値がゼロならば、変数cに３を代入する

b) 変数dの値がゼロでないならば、変数eにd×１０を代入する

c) セルB2の値がゼロならば、変数fにセルB2の値を代入する

d) セルB3の値がゼロより大きいならば、変数gにセルB2の値を代入する

e) セルB4の値が30未満ならば、変数hに変数h＋１の値を代入する

f) セルB2から下にp、右に0移動したセルの値がゼロでないならば、変数sにその値を加える

ここでは、If 文を用いて、「鹿沼市」出身者が何名いるかを数えるプログラムを作成する。

(1) 次のsample8をsample7の下に追加する。

```
Sub sample8()
    Dim i As Integer
    Dim k As Integer

    k = 0

    For i = 1 To 5                                          …①
        If Range("J5").Offset(i, 0).Value = "鹿沼市" Then    …②
            k = k + 1                                       …③
        End If
    Next i

    MsgBox "鹿沼から来た人は" & k & "人です"

End Sub
```

データ構造

変数名	使用目的
i	繰り返し回数を保持
k	鹿沼市出身の人数を保持

①では、5回の繰り返しを行うよう For 文を記述している。②では、セル J5 から、下へ順に値を取得し、"鹿沼市"と比較している。両者が等しい場合、Then 以下の③の命令が実行され、変数 k の値を１つ増加させる。ここでは、Else は省略されている。

問 3-8　sample8 を変更して、鹿沼市以外の出身者の数を数えるマクロを 10 秒で作成しなさい。

3.4.2　複雑な If 文

複数の条件が必要な場合も、同様に記述することができる。

```
Sub sample9()
    Dim i As Integer                                ‘繰り返し回数
    Dim sei As String                               ‘判定した性別

    For i = 1 To 5

        If Range("E5").Offset(i, 0).Value = 1 Then      …①
            sei = "男性"                                …②
        ElseIf Range("E5").Offset(i, 0).Value = 2 Then  …③
            sei = "女性"                                …④
        Else                                            …⑤
            sei = "不明"                                …⑥
        End If                                          …⑦

        Range("E5").Offset(i, 0).Value = sei            …⑧

    Next i

End Sub
```

①の条件では、対象とするセルの値を１と比較し、等しければ、②変数 sei に"男性"を代入する。③そうでなくて、もし(ElseIf)、対象セルが２であれば、④変数 sei に"女性"を代入する。⑤さらにそうでなければ(Else)、⑥変数 sei に"不明"を代入する。この If 文によって代入された値は、⑧によって、シートの対象セルに書き戻される。

3.4.3　Select 文

3.4.2 のように、条件が多い場合、If 文が複雑になり、分かりにくくなる。このような場合は、Select 文を使うことによって、簡単に記述することができる。

Select 文は以下のような構文を持つ。

```
Select Case  式
  Case 値１
    命令１
  Case 値２
    命令２
    ……
  Case Else
    命令ｎ
End Select
```

式　　：評価対象の変数
値１　：変数の値(Is や To を用いて、比較や範囲を指定できる)
命令１：式が値１の場合に実行される命令群
値２　：変数の値
命令２：式が値２の場合に実行される命令群
命令ｎ:式がどの Case の値にも該当しなかった場合に実行される命令群(Case Else と命令ｎは省略可)

Select 文を使用して、sample9 を書き換えると、以下のようになる。

```
Sub sample10()
    Dim i As Integer
    Dim sei As String

    For i = 1 To 5
        Select Case Range("E5").Offset(i, 0).Value      …①
            Case 1                                      …②
                sei = "男性"                            …③
            Case 2                                      …④
                sei = "女性"                            …⑤
            Case Else                                   …⑥
                sei = "不明"                            …⑦
        End Select                                      …⑧

        Range("E5").Offset(i, 0).Value = sei

    Next i
End Sub
```

このマクロでは、①で対象となるセルの値を指定し、この値が、
②１の場合、③変数 sei に"男性"を代入し、
④２の場合、⑤変数 sei に"女性"を代入し、
⑥どれにも当てはまらない場合は、⑦変数 sei に"不明"を代入
している。Select 文の終了は、End Select で指定する。

第２部

第２部では、エクセルのマクロを活用するため、エクセルマクロ上で行うデータの並べ替えの方法、ボタンの配置、マクロからの印刷、ファイルの入力、ユーザーフォームの作成について説明する。これらを組み合わせることにより、最終的に、次のようなアンケート処理システムを構築する。

① 児童生徒に問題やアンケートを記載したエクセルのファイルを配布し、回答・保存させる
② 回答したそれらのファイルを、１つのファイルに取り込む
③ 取り込んだファイル上で、回答の一覧表を作成する
④ 配布用の個票を印刷する
⑤ ②～⑤のうち、どの処理を行うかをメニュー画面から選んで実行する

4 並べ替えとボタンの配置

4.1 並べ替え

シート上にあるデータを並べ替えるためには、Sort メソッドを使う。Sort メソッドの構文は、次のとおりである。

範囲.Sort Key1:=キー, Order1:=順序, Orientation:=行列, Key2:=…

範囲：並べ替えを行う範囲を Range 等で指定する。
キー：並べ替えのキーの範囲を Range 等で指定する。
順序：昇順(xlAscending)か、降順(xlDescending)かを指定する。
行列：列単位(xlSortColumns)か、行単位(xlSortRows)で並べ替えるか指定する。
Key2…：キーと順序は、３つまで指定できる。

例として、**prog2.xlsm** の「基礎データ」シートの範囲「B6 から X10」を並べ替えるマクロは次のようになる。

```
Sub sample11()

    Sheets("基礎データ").Range("b6:x10").Sort _
        Key1:=Range("k6"), Order1:=xlAscending, Orientation:=xlSortColumns

End Sub
```

このマクロでは、

並べ替えのキーとして、「K6」の列（xlSortColumns）
並べ替えの順序がxlAscending（昇順、小さい順）
並べ替えの方向としてxlSortColumns（列、１行を１件のデータとする）

として、並べ替える。すなわち、「基礎データ」シートで、金額が小さい人が上、大きい人が下になるように５人分のデータを並べ替えることができる。

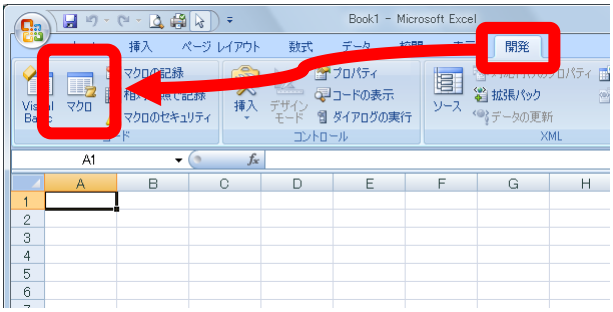
なお、Sort 行が１行で収まらないため、行継続文字「 _」（スペースとアンダースコア）を使って、次の行も論理的に同一行であることを表している[15]。

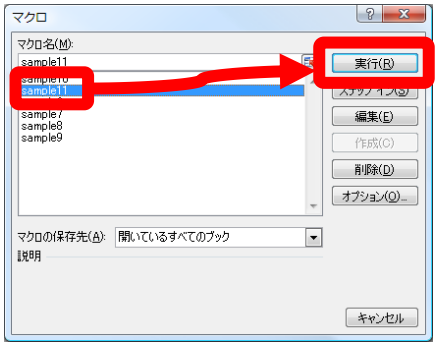
問 4-1　Ｂ列の番号順に並べ替えるマクロ sample12 を作成しなさい。

[15] 行継続文字による行の分割は、25 行まで可能である。

4.2 ボタンの配置

マクロを実行する場合、これまで VB エディタの画面から実行してきたが、多くの人に使ってもらう場合、これは非常に使いづらい。また、「開発」タブから「マクロ」ボタンをクリックし、マクロ名を指定して実行する方法（下図）もあるが、いずれの方法も、「開発」タブの表示が必要であり、また、マクロの名前を覚えておかなければならないなど、不便な面もある。



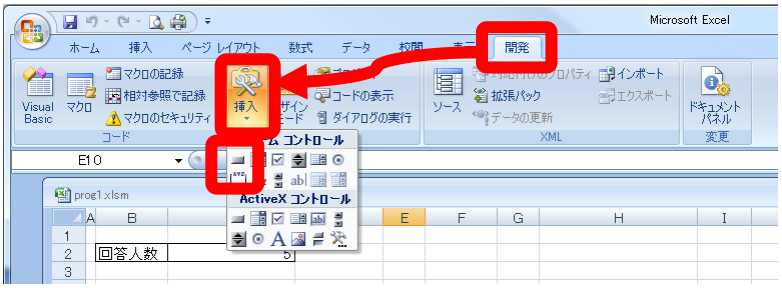


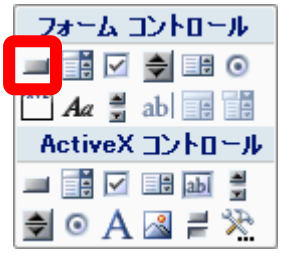
これに対して、シート上にボタンを用意し、そのボタンをクリックするだけでマクロを実行する方法がある。エクセルでは、マクロを実行するためのボタンとして、「ボタン（フォームコントロール）」、「コマンドボタン（ActiveX コントロール）」、「図形」などを使うことができる。この３つの違いをまとめると、**【資料 12】**のようになる。

今回は、標準モジュール内のマクロが実行可能な「ボタン（フォームコントロール）」を利用することとするが、ActiveX コントロールや図形を用いてもほぼ同様のことが可能である。

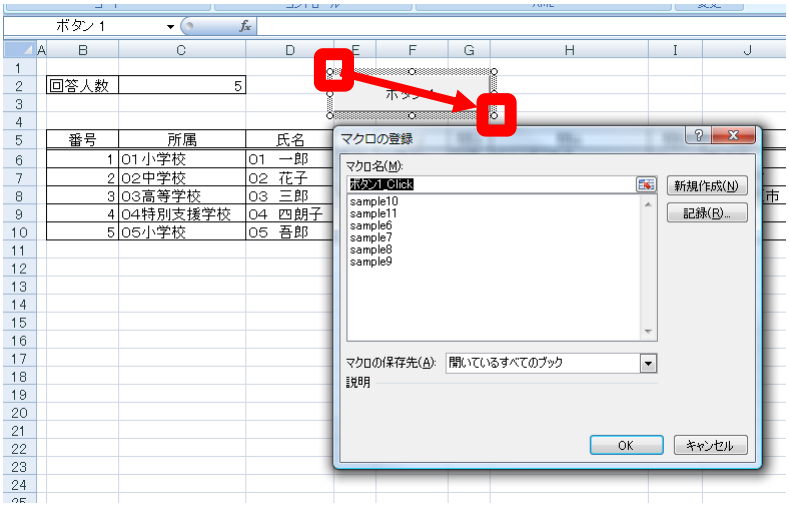
ここでは、例として、先ほど記述したマクロ sample11 を実行するボタンを配置する手順を以下に示す。

(1) 「開発」タブを選択し、「挿入」→フォームコントロールの「ボタン」をクリックする。

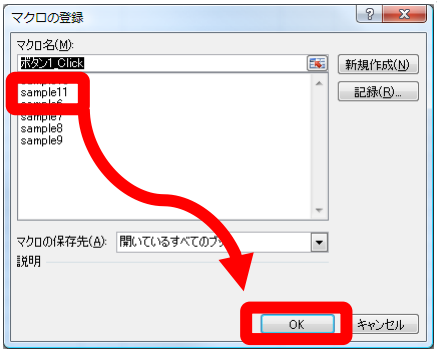




(2) シート上にボタンをドラッグして配置する。「マクロの登録」ウィンドウが表示される。



(3) 実行したいマクロ（ここではsample11）を選択し、ＯＫをクリックする。



(4) ボタンの上で右クリックし、「テキストの編集」を選ぶ。

(5) ボタンのテキストを「お小遣いの金額で並べ替え」に変更する。

(6) ボタン上で右クリックし「テキスト編集の終了」を選ぶ。

以上で、ボタンにマクロを実行させるための設定が終わりとなる。ボタンが選択されていない状態（ボタンに選択枠が出てない状態）でマウスを乗せると、カーソルが指のマークに変化し、この状態でクリックすると、選択したマクロが実行される。

4.3 デバッグ

マクロや計算式の入力ミス、アルゴリズムの間違いなど、マクロが動作しなかったり、想定した結果と異なる実行結果になったりすることがある。このようなときは、マクロのどの場所に問題があるのかを探し出し、適切に修正しなければならない。このような間違い（bug、バグ）を取り除く作業を「デバッグ」(debug、デバグともいう）といい、デバッグに用いられるのが、「デバッグ」メニューである。VB のデバッグでは、ブレークポイントの設定、ステップ実行、変数の値の表示（ウォッチ）などが可能である。

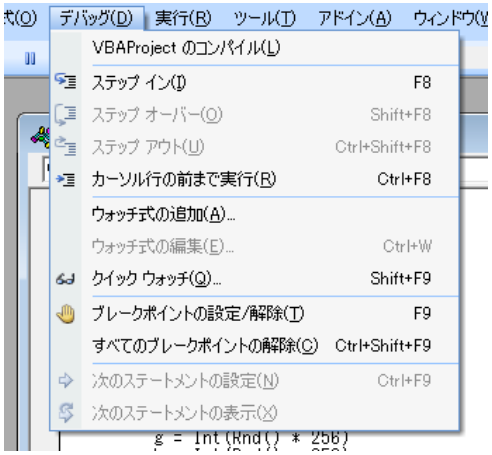


図 5　デバッグメニュー

命令の入力間違いは、プログラム入力時にシステムから指摘されることが多いため、あまり発生しないが、変数名や式の間違い（＋を－として入力しているなど）は、入力時に指摘されることはない。また、アルゴリズムの間違いもシステムから指摘されることがないため、特に大きなプログラムになると、根気強くデバッグしなければならない。それぞれのデバッグ方法については、**【資料 13】**に示す。

4.4 関数と引数

これまで、マクロのプロシージャを定義する場合に、「Sub マクロ名()」の書式を利用してきた。この Sub は、3.1 で述べたように、ひとまとまりのプロシージャを定義する場合に用いられる。同様の命令として、Function がある[16]。これは、呼び出し側に値を返す場合に用いられる。書式は以下のとおりである。

```
Function マクロ名(引数) As 型名

    マクロ名 = 返す値

End Function
```

マクロ名：定義するプロシージャに付ける名前
引数：呼び出し側からプロシージャに渡された値を入れる変数と、その型
型名：このプロシージャから返す値の型
返す値：プロシージャから呼び出し側に返す値
Function を途中で抜け出す場合は、Exit Function を使う。

Sub と Function を用いた例を以下に示す。次のコードを入力し、プロシージャ a を実行する。

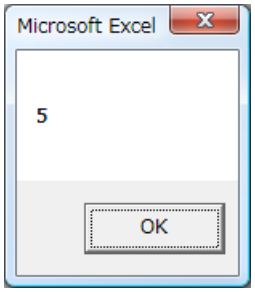
```
Sub a()

    Call b                                          ' b の呼び出し

End Sub

Sub b()
    Dim s As Integer

    s = wa(2, 3)                                    ' wa の呼び出し
    MsgBox s

End Sub

Function wa(x As Integer, y As Integer) As Integer        'wa の定義(引数は x と y)

    wa = x + y

End Function
```

ここでは、a、b、wa の 3 つのプロシージャを定義している。Sub によって定義されたプロシージャは Call 命令を用いて呼び出し、Function によって定義されたプロシージャは関数として式の中に入れることにより呼び出す。プロシージャ a を実行すると、以下の手順で処理が行われる。

① a を実行すると、プロシージャ b を呼び出す
② b では、s = wa(2, 3)の行で wa を呼び出す
③ wa は引数として整数型の x と y を必要とするが、呼び出し側で wa(2, 3)としているため、それぞれ対応する x に 2、y に 3 が代入される
④ wa が実行される
⑤ x＋y(すなわち、2+3)を計算し、その答え 5 をプロシージャ名と同じ変数 wa に代入する。この値がこのプロシージャの値となり呼び出し側に返される
⑥ 呼び出し側の wa(2, 3)は、5 に置き換えられる。すなわち、s = 5 と同等になり、s に 5 が代入される

[16] マクロで定義した Function は、マクロ中から呼び出して使用することは当然できるが、sum や average のような標準で実装されている関数と同等に使用することが可能である。例えば、マクロで Functiuon goukei(s as Integer) As Integer のように宣言してマクロを記述した場合、シートのセルの式として、「=goukei(3)」のように入力して使うことが可能である。

⑦　この s の値は、次の MsgBox によって表示される



⑧　b の実行が終了する
⑨　a に動作が戻る
⑩　マクロの実行が終了する

このように、Function は、戻り値を１つ返すことができる[17]。Sub は、戻り値を返す必要のない場合に使われるが、引数は Function と同様にとることができる。なお、戻り値が必要ない場合（すなわちプロシージャを Sub で定義した場合）は、呼び出し側でプロシージャ名のあとに付けるカッコは不要である。

この例では、a から b を呼び出し、b から wa を呼び出した。このように、呼び出された側がさらに呼び出し側となり、別のプロシージャを呼び出すような形式を、「入れ子」（ネスト）[18]と呼び、VB では、最大７レベルまでの入れ子が可能である。また、自分自身を呼び出すような形式を「再帰」と呼び、特に人工知能等の分野ではよく用いられる。

[17] 2 つ以上の戻り値が必要な場合は、グローバル変数を用いる方法がある。
[18] 入れ子構造には、プロシージャの中でプロシージャを呼び出す他に、if 文の中の if 文や、for 文の中の for 文などでも使われる。

5 印刷

ここでは、マクロから印刷する方法を説明する。

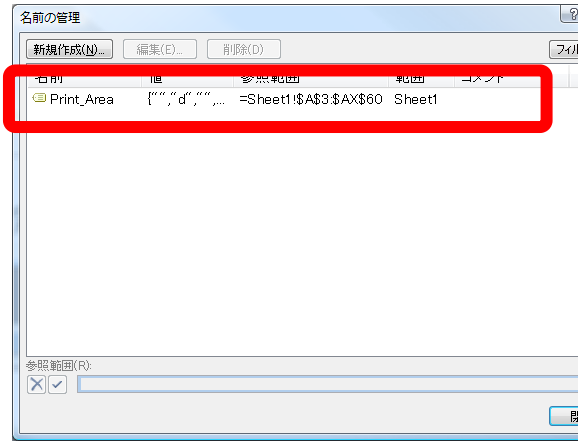
5.1 印刷のための基本操作

5.1.1　範囲の設定

現在のシートの印刷範囲を設定するためには、「PageSetup.PrintArea」プロパティを使用し、以下のように記述する。

```
ActiveSheet.PageSetup.PrintArea = "$A$3:$AX$60"
```

この命令によって、現在のシートの印刷範囲(Print_Area)を、A3 から AX60[19]に設定することができる。



5.1.2　印刷の命令

シートの内容を印刷するためには、印刷のダイアログを出してから印刷する場合と、ダイアログを出さずに直接印刷する場合が考えられる。印刷のダイアログを出さずに直接印刷する場合には、PrintOut メソッドを使い、以下のように記述する。

```
ActiveWindow.SelectedSheets.PrintOut Copies:=1
```

この中で、Copies:=1 は、印刷部数が１部であることを示している。この他にもページの指定など、**【資料 14】**のような指定が可能である。

例：３ページから５ページまで、２部、部単位(Collate)で印刷する場合。

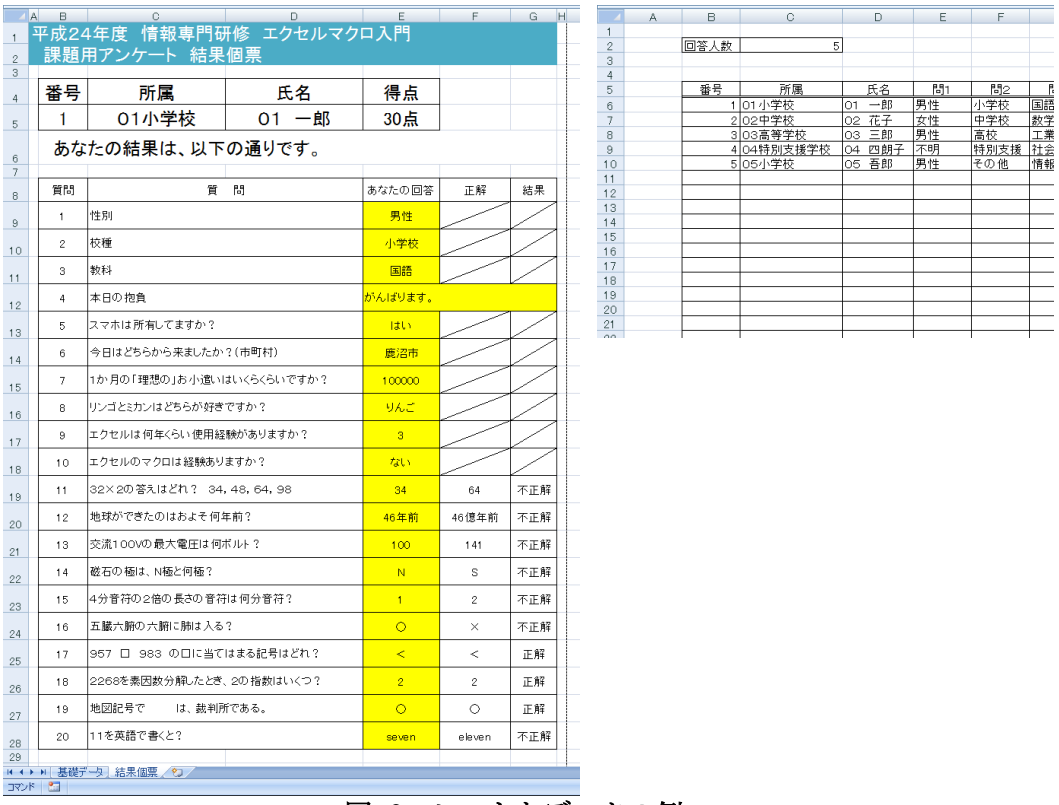
```
ActiveWindow.SelectedSheets.PrintOut From:=3, To:= 5, Copies:=2, Collate:=True
```

問 5-1　11 ページから 15 ページまで、40 部、ページ単位(11 ページを 40 部、12 ページを 40 部…の順番)で印刷したい場合、どのような記述となるか考えなさい。

[19] 範囲の設定には「$」を付けて絶対番地としているが、設定時に明示的に「$」を付けなくても、自動的に絶対番地として設定される。

5.2 連続印刷

全生徒のデータを１枚のシートにデータベースのように保存してある場合、別のシートに印刷の書式を作って、１名ずつ書式に当てはめて印刷することがある。この場合、１名印刷するごとに、表示する出席番号を変更し、さらに印刷を繰り返さなければならない。これを、全生徒のデータを順次読み出しながら、自動で印刷できると便利である。このために、シートに工夫をして、繰り返し命令を使って印刷する。ここでは、「**prog3.xlsm**」の「結果個票」シートを使用し、以下に、この手順を示す。



平成24年度　情報専門研修　エクセルマクロ入門
課題用アンケート　結果個票

番号	所属	氏名	得点
1	01小学校	01　一郎	30点

あなたの結果は、以下の通りです。

質問	質　問	あなたの回答	正解	結果
1	性別	男性		
2	校種	小学校		
3	教科	国語		
4	本日の抱負	がんばります。		
5	スマホは所有してますか？	はい		
6	今日はどちらから来ましたか？(市町村)	鹿沼市		
7	1か月の「理想の」お小遣いはいくらくらいですか？	100000		
8	リンゴとミカンはどちらが好きですか？	りんご		
9	エクセルは何年くらい使用経験がありますか？	3		
10	エクセルのマクロは経験ありますか？	ない		
11	32×2の答えはどれ？　34, 48, 64, 98	34	64	不正解
12	地球ができたのはおよそ何年前？	46年前	46億年前	不正解
13	交流100Vの最大電圧は何ボルト？	100	141	不正解
14	磁石の極は、N極と何極？	N	S	不正解
15	4分音符の2倍の長さの音符は何分音符？	1	2	不正解
16	五臓六腑の六腑に肺は入る？	○	×	不正解
17	957 □ 983 の□に当てはまる記号はどれ？	<	<	正解
18	2268を素因数分解したとき、2の指数はいくつ？	2	2	正解
19	地図記号で　　は、裁判所である。	○	○	正解
20	11を英語で書くと？	seven	eleven	不正解

回答人数　5

番号	所属	氏名	問1	問2
1	01小学校	01　一郎	男性	小学校
2	02中学校	02　花子	女性	中学校
3	03高等学校	03　三郎	男性	高校
4	04特別支援学校	04　四朗子	不明	特別支援
5	05小学校	05　吾郎	男性	その他

図 6　シートとデータの例

5.2.1　結果個票シートの作成

シートを作成する上でのポイントとしては、結果を表示するデータの**番号**をいずれかのセルに設定し、その番号を使って、INDEX 関数で該当するデータを呼び出す。今回の例では、以下のように、セル B5 に番号を設定し、そのセルの値から、基礎データの表の値から、該当するデータを読み出している。

```
E9: =INDEX(基礎データ!$B$6:$X$45, $B$5, B9+3 )
```

INDEX 関数は３つの引数をとる。

「**基礎データ!$B$6:$X$45**」　元となるデータの範囲を示す
「**$B$5**」　番号が存在するセルを指定することにより、行を指定
「**B9+3**」　質問番号(B9)から該当する質問の列を指定

例として B5 に番号１を指定すると、「基礎データ」シートの範囲 B6:X45 の中の１行目（すなわち、シート全体の６行目）を指定している。さらに、例の場合は、質問番号 B9 に１が入力されているため、１＋３＝４列目を指定していることになり、B6:X45 の４列目は（B 列を１として）E 列を指定することになる。この関数によって、「基礎データ」シートの範囲「B6:X45」から、１行４列目であるセルＥ６のデータ「男性」を取得する。同様に、所属、氏名、あなたの回答のセルをすべて埋めることができる。

なお、「正解」の列は、あらかじめ正解となる答えが入力してあり、**結果の列**は「あなたの回答」と「正解」が等しければ「正解」、そうでなければ「不正解」となる IF 関数が入力してある。さ

らに、得点は、結果の列の「正解」の文字の数を数え、その値を 10 倍して、後ろに「点」の文字を付加することにより表示している。

表 1　各セルの内容

セル	セルの値（式）
B5	個票を表示する番号
C5	=INDEX(基礎データ!$B$6:$X$45, $B$5, 2)
E5	=COUNTIF(G19:G28,"正解")*10 & "点"
E9	=INDEX(基礎データ!$B$6:$X$45, $B$5, B9+3)
F19	６４（問１１の正解の値）
G19	=IF(E19=F19, "正解", "不正解")

このようにして作成した結果個票では、セル B5 の番号の値だけを変化させれば、シート内のすべての値を変化させることができるようになる。

5.2.2　連続印刷マクロの作成

このようなシートを作成することによって、セル B5 の値を変更しながら印刷することによって、全人数の印刷が可能となる。マクロの例を以下に示す。

```
Sub 印刷()
    Dim i As Integer                                          '繰り返しのための変数

    Sheets("結果個票").Select                                  'シートの選択
    ActiveSheet.PageSetup.PrintArea = "$a$1:$h$29"            '印刷範囲の設定

    For i = 1 To Range("人数").Value                          '人数分繰り返す

        Range("b5").Value = i                                 '番号欄に記入

        ActiveWindow.SelectedSheets.PrintOut Copies:=1        '印刷のダイアログを出さずに印刷

    Next i

End Sub
```

※範囲名「人数」は、あらかじめセル C2 につけられています。

このマクロでは、最初に、シートを選択して、印刷範囲を指定している。その後、繰り返し命令により１番目から人数の値まで、１つずつ増やしながら、セル B5 に値を設定し、印刷命令を実行している。この命令によって、人数分のシートを印刷することができるようになる。

このマクロを利用して 40 名の生徒のデータを印刷した場合、40 枚分の印刷データが次々とプリンタに送信されることになる**【資料 15】**。

問 5-2　上記のプロシージャを実行するためのボタンを設定しなさい。

5.3 印刷ダイアログの表示

印刷する場合、プリンタを選択するなど、最初に印刷のダイアログを出してから印刷したい場合がある。このようなダイアログを表示するためには、以下のように Application オブジェクトの Dialogs オブジェクトで xlDialogPrint を指定しなければならない[20]。

```
ans = Application.Dialogs(xlDialogPrint).Show(arg12:=2)
```

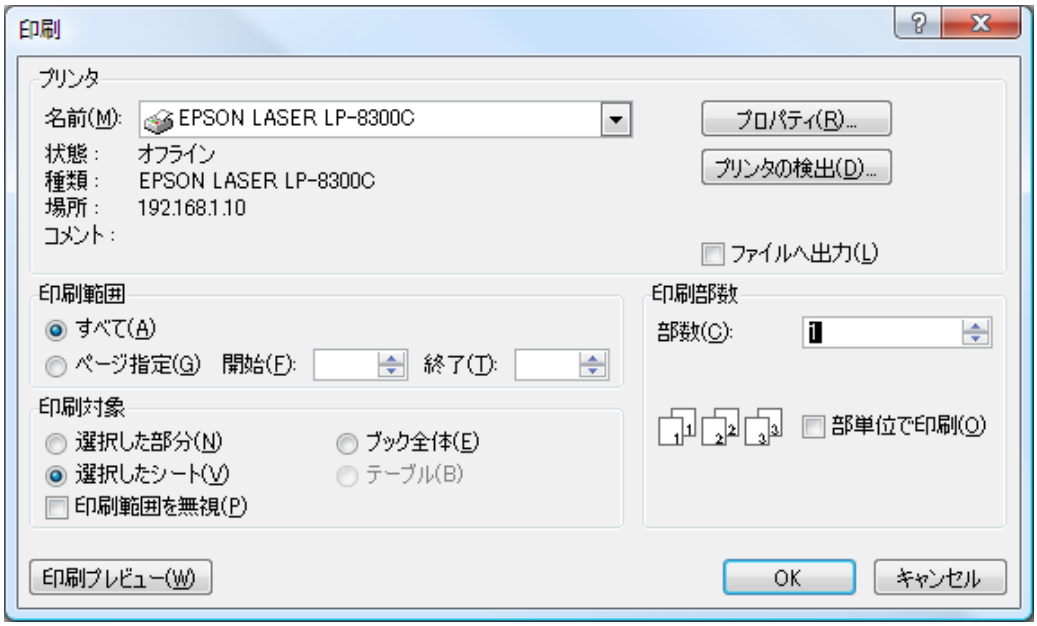


図 7　印刷ダイアログの例

この命令では、xlDialogPrint が印刷のダイアログを指定し、そのダイアログを表示(show)する。また、その戻り値をブール型変数 ans に代入している。戻り値は True または False のいずれかで、True の場合は OK ボタンが押されて印刷して終了したとき、False の場合はキャンセルボタンが押されて印刷せずに終了したときの値となる。

さらに、引数 arg12:=2 は、印刷対象(arg12)として、選択したシート(2)を選択することを表している[21]。この他の引数は、**【資料 16】**のとおりである。

ここでは、最初の１人目を印刷する場合にはダイアログを表示し、２人目以降はダイアログを表示せず、また、マクロ実行時に印刷開始の確認を行うように、マクロを変更し、保存(**prog3.xlsm**)する。

---

[20] 印刷のダイアログの他にも「セルの書式設定」ダイアログや、「ファイルを開く」ダイアログなどエクセルで使われる各種ダイアログを表示させることができる。エクセルのヘルプで、「XlBuiltInDialog」を参照のこと。

[21] 上記の命令は、戻り値を使用しない場合、以下のように記述することも可能である。

```
Application.Dialogs(xlDialogPrint).Show  arg12:=2
```

このように VB では、**戻り値を使用するときに引数にカッコ**を使い、**戻り値を使用しないときにはカッコを記述しない**規則になっている。

```
Sub 印刷_その2()
    Dim msg_ans As Integer                                          '印刷確認メッセージボックスの結果
    Dim i As Integer                                                '繰り返し用変数
    Dim dl_ans As Boolean                                           '印刷ダイアログの結果

    msg_ans = MsgBox("印刷を開始します", vbOKCancel, "確認！！")
    If msg_ans = vbCancel Then End                                  'キャンセルが押されたら終了

    Sheets("結果個票").Select                                       '印刷するシートの選択
    ActiveSheet.PageSetup.PrintArea = "$a$1:$h$29"                  '印刷範囲の設定

    Range("b5").Value = 1                                           '最初の１人目の番号を番号欄に記入

    dl_ans = Application.Dialogs(xlDialogPrint).Show(arg12:=2)      '印刷のダイアログを出す
    If dl_ans = False Then End                                      'キャンセルが押されたら終了

    For i = 2 To Range("人数").Value                                '人数分繰り返す

        Range("b5").Value = i                                       '番号欄に記入

        ActiveWindow.SelectedSheets.printout Copies:=1              '印刷のダイアログを出さずに印刷

    Next i

End Sub
```

このマクロでは、以下の手順によって、印刷を実行している。

① 印刷確認のメッセージボックス（メッセージとして「印刷を開始します」、ボタンの種類を「OK」と 「Cancel」、タイトルを「確認！！」）を表示
② キャンセルが押されて戻ってきた場合には、ENDによってマクロを終了
③ 「OK」で戻ってきた場合には、印刷するシートと、印刷範囲を指定
④ 番号欄に１を設定することによって、１人目の結果を表示
⑤ 印刷のダイアログを表示
⑥ キャンセルで戻ってきた場合（dl_ans=False）には、マクロを終了
⑦ OKで戻ってきた場合には、（１人目の印刷は終わったので）２人目以降を印刷

6 エクセルシートの取り込み

エクセルでは、いくつかのファイルに分けてデータを管理することがある。また、生徒がそれぞれに記入した多数のエクセルファイルを対象に、同一の操作を行いたいこともある。ここでは、このような場合に利用できる、マクロによるシートの取り込みを取り扱う。

最初に、特定のフォルダ（ここでは sample とする）に入っているファイルすべてを対象として、そのファイルを開いたときに最初に表示されるシートを、１つのファイル（同じく **prog4.xlsm** とする）に集める場合を考える。

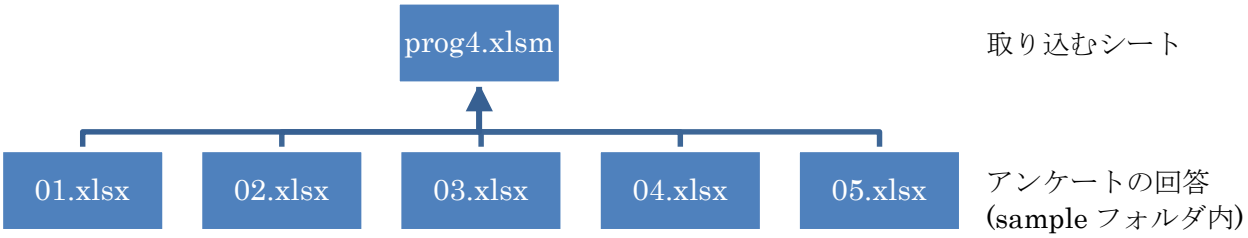


図 8　ファイルの取り込み

6.1 ディレクトリの操作

読み込むファイルを指定する場合、そのファイル名を指定しなければならない。ファイル名が分かっている場合はそのファイル名を直接指定できるが、ファイル名が分からない場合、何らかの方法で指定しなければならない。

ディレクトリ内のファイル名を取得するためには、Dir 関数を利用する。この関数では、ディレクトリ内のファイルの名前を順に取得することができる。

Dir 関数の書式

filename = Dir(パス名, 属性)

パス名：フォルダ名やファイル名（省略可能）

属　性：対象とするファイルの属性の合計値（省略可能、省略すると標準ファイルとなる）

vbNormal	標準ファイル
vbReadOnly	読み取り専用ファイル
vbHidden	隠しファイル
vbSystem	システムファイル
vbVolume	ボリュームラベル
vbDirectory	フォルダ

※Dir 関数を最初に呼び出すときにパス名を省略するとエラーとなる。また、パス名、属性の両方を省略した場合、前回に取得したファイル名の次のファイル名を取得する。

例：D:ドライブのファイルとフォルダを取得

```
filename = Dir( “d:¥” , vbNormal + vbDirectory)
```

この Dir 関数を用いたファイルの指定方法の例を以下に示す。

このマクロでは、まず「ActiveWorkbook.Path」によって、現在のファイル(**prog4.xlsm**)のあるディレクトリ名を取得し、文字型変数 strPATH に代入している。次に、このディレクトリ名に、「¥sample¥」を結合[22]することによって、**prog4.xlsm** のあるディレクトリの中の sample フォルダを指定している。

次に、Dir 関数によって、指定したパスにある１つ目のファイル名を取得し、その文字列が何もなくなるまで (””でない間は)、メッセージボックスでそのファイル名を表示する処理と、次のファイル名を読み込む処理を繰り返す。

[22] 文字列の結合には、「＋」も使うことができるが、あいまいさを避けるため、できる限り「&」を用いる。

```
Sub ファイル名の取得()
    Dim strPATH As String                          'ファイルへのパス
    Dim strFN As String                            'ファイル名

    'ファイル名一覧の作成
    strPATH = ActiveWorkbook.Path                  '現在のワークブックのパスを取得
    strPATH = strPATH & "¥sample¥"                 'パス名に¥sample¥を追加する
    strFN = Dir(strPATH, vbNormal)                 '指定パスにあるファイルを取得

    Do While strFN <> ""                           'ファイル名がなくなるまで繰り返す
        MsgBox "ファイル名：" & strFN              'ファイル名をメッセージボックスで表示
        strFN = Dir()                              '次のファイル名を読み込み
    Loop                                           '繰り返し終わり

End Sub
```

なお、このマクロに使用されている２つの変数名は、先頭が str で始まっている。これは、この変数の型が String 型であることを表しており、このような変数の型を先頭に付ける記述（接頭語、プレフィックス）方法を、ハンガリアン記法と呼ぶ。以前は、この記法を Microsoft が推奨していたことから、多くのプログラムでこの記法が見られる。

6.2 シートの取り込み

マクロを利用せずに、１つのシートを別のエクセルファイルにコピーする場合、次のような手順を行う。

① コピー元となるエクセルファイルを開く
② コピー元シートを選択し、「移動またはコピー」を選ぶ
③ 移動先ブック名を選択し、挿入先シートを選び、コピーする
④ （必要に応じて）コピーしたシートのシート名を変更する
⑤ コピー元となったエクセルファイルを保存せずに閉じる

多くのシートを１つのファイルにまとめたい場合、３Ｄ参照**【資料 17】**を利用する方法もあるが、利用できる関数が限定されていたり、手作業で行うには非常に時間がかかるため、これをマクロを使って自動で行う。このマクロは、次のようになる。

```
Sub ファイルから読み込む()
    Dim strPATH As String                                  'ファイルへのパス
    Dim strFN As String                                    'ファイル名
    Dim cnt As Integer                                     '人数を数える

    strPATH = ActiveWorkbook.Path                          '現在のワークブックのパスを取得
    strPATH = strPATH & "¥sample¥"                         'パス名に¥sample¥を追加する
    strFN = Dir(strPATH, vbNormal)                         '指定パスにあるファイルを取得

    cnt = 0                                                '人数をゼロにする
    Do While strFN <> ""                                   'ファイル名がなくなるまで繰り返す
        cnt = cnt + 1                                      '人数を増やす

        Workbooks.Open Filename:=strPATH & strFN, ReadOnly:=True    'ファイルを開く①
        ActiveSheet.Copy before:=Workbooks("prog4.xlsm").Sheets("tmp")    'シートをコピーする③
        ActiveSheet.Name = "t" & cnt                       'シート名を変更する④

        Workbooks(strFN).Close SaveChanges:=False          'ファイルを保存しないで閉じる⑤

        strFN = Dir()                                      '次のファイル名を読み込み
    Loop                                                   '繰り返し終わり

    Range("人数").Value = cnt                              '最終人数を保存
End Sub
```

このマクロでは、先の「ファイル名の取得」プロシージャを変更し、①～④を挿入したものである（②は③に含まれる）。また、取得した人数を数えながら、"t"とその番号を結合し、シート

名(t1, t2, t3, …)としている。ファイル名は、そのまま prog4.xlsm とする。

ファイルを開くための命令 Workbooks.Open は、ここではファイル名（Filename := strPATH & strFN、パス名とファイル名を結合したもの）と読み込み専用(ReadOnly:=True)であることを指定し、エクセルのファイルを開く。この時、エクセルファイルを開くと、自動的にそのファイルが選択された状態になる。

次の ActiveSheet.Copy 命令は、現在のシート（すなわち、上記で開き、選択された状態になったシート）をコピーするが、コピー先として、ワークブック prog4.xlsm のシート tmp の前(before)を指定している。この命令によって、コピー後のシートが選択された状態となる。

さらに、ActiveSheet.Name によって、選択されたシート（今回は、コピー後のシート）の名前を、「t」と人数を文字として結合したものに変更している。

最後に、先ほどオープンしたワークブック(ファイル名：strFN)を保存せず(SaveChanges:=False)に閉じ(Workbooks.Close)、次のファイル名を取得し、繰り返しを行う。

このように、多数のファイルに分かれたシートを１つのファイルに結合することができる。

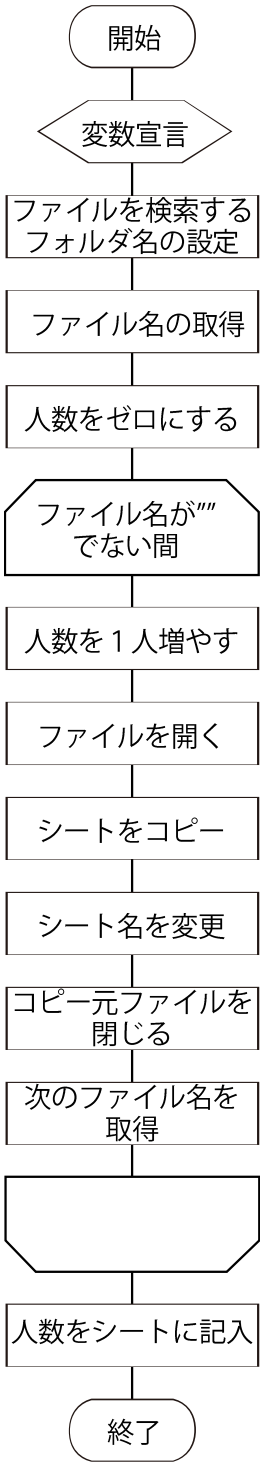


図 9
ファイル読み込み

Open メソッドの書式

Workbooks.Open Filename:=ファイル名, ReadOnly:=読み書きモード

ファイル名：パス名を含むファイル名

読み書きモード：ファイルを読み込みのみ(True)か、読み書き可(False)とするか

Close メソッドの書式

ワークブック名.Close SaveChanges:=保存の有無

ワークブック名：閉じるワークブック（ファイル名。パスは不要）

保存の有無：保存して終了(True)か、保存せずに終了(False)か

Copy メソッドの書式

コピー元シート名.Copy Before :=コピー先シート名

コピー元シート名.Copy After :=コピー先シート名

コピー元シート名：コピーを行うシート。

コピー先シート名：貼り付け先のシート名（ワークブック名を指定することも可）。

コピー先シートの、前に貼り付ける(Before)か、後に貼り付ける(After)のいずれかを選択する。

Name プロパティの設定

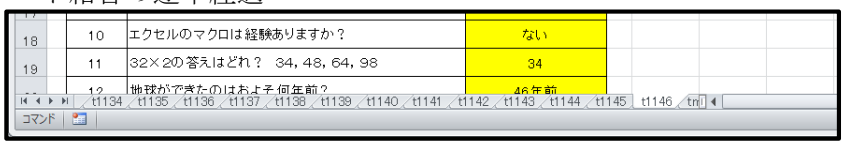
シート名.Name = 新しいシート名

シート名：名前を変更するシート

新しいシート名：変更後の名前

問 6-1　何枚のシートまで結合できるか試しなさい。

シート結合の途中経過



6.3 基礎データ表の作成

ここまでに、複数のエクセルシートから、１つのファイルにまとめる方法を示した(prog4.xlsm)。ここでは、このファイル内の全シートのデータを、１つの表としてまとめる方法を説明する。

```
Sub 表の作成()
    Dim i As Integer                                        '繰り返し変数、何番目を処理しているか
    Dim syozoku As String                                   '所属名
    Dim namae As String                                     '氏名

    Sheets("基礎データ").Select                              '集計先シートの選択
    Range("b6:z100").Value = ""                             '集計先セルのクリア

    For i = 1 To Range("人数").Value                         '人数分繰り返す
        syozoku = Sheets("t" & i).Range("c6").Value         '所属を取得
        namae = Sheets("t" & i).Range("e6").Value           '氏名を取得

        Sheets("基礎データ").Range("b5").Offset(i, 0).Value = i          '番号を集計先に転記
        Sheets("基礎データ").Range("b5").Offset(i, 1).Value = syozoku    '所属を集計先に転記
        Sheets("基礎データ").Range("b5").Offset(i, 2).Value = namae      '氏名を集計先に転記

        Sheets("t" & i).Select                              '転送元シートの選択
        Range("e9:e28").Copy                                'データの範囲をコピー

        Sheets("基礎データ").Select                          '集計先シートの選択
        Sheets("基礎データ").Range("b5").Offset(i, 3).Select '集計先セルの選択
        Selection.PasteSpecial Paste:=xlPasteFormulas, _
            Operation:=xlNone, SkipBlanks:=False, Transpose:=True   '集計先シートへ貼り付け

    Next i                                                  '繰り返しここまで

End Sub
```

このマクロのポイントは、以下の３点である。

１）集計先シートの集計表をクリアすること

これは、このマクロを複数回実行したとき、後から実行したときのほうが人数が少なかった場合、前回のデータが残ってしまうことを防ぐ必要があるためである。ここでは範囲をb6:z100 として、広めの範囲を消去している。

２）シートから変数に値を読み出し、それを別のシートに書き込むための方法

ここでは、所属と氏名を取得しているが、対象となるシート名は毎回変化するため、そのシート名を Sheets オブジェクトで、"t"と番号を結合した名前として指定している。さらに、コピー先は、１つのシートであるが、行が変わるため、offset によって上から行を変化させながら、値を記入している。

３）セルのコピーアンドペースト

最初に、コピー元となるシートを選択し、データの範囲を Copy メソッドでコピーしている。この Copy メソッドは、先のシートコピーのためのメソッドと同じ名前であるが、ここではシート名ではなく、セルの範囲を指定している。その後、コピー先シートを選択して、コピー先セルを１つ選択している。貼り付けは、PasteSpecial メソッドを使用して、貼り付けの条件を指定している。特に、コピー元のデータは列で与えられるが、コピー先は行となるため、行と列の変換(Transpose:=True)を指定している。

Copy メソッドの書式

セルの範囲.Copy

セルの範囲：クリップボードへコピーする範囲を Range 等で指定

PasteSpecial メソッドの書式

セルの範囲.PasteSpecial Paset:=**貼り付ける部分**, Operation:=**貼り付け操作**,
SkipBlanks:=**空白セルの処理**, Transpose:=**行列入れ替え**

セルの範囲：クリップボードから貼り付けるセルの指定
貼り付ける部分：数式、書式、入力規則、すべてなど、何を貼り付けるか指定
貼り付け操作：貼り付けるとき、数値データがどのように計算されるか指定
空白セルの処理：空白セルを貼り付ける(False)、貼り付けない(True)
行列入れ替え：行と列を入れ替える(True)、入れ替えない(False)

問 6-2　集計表（基礎データ）の行列を反転せず、そのまま貼り付けて利用するマクロを作成しなさい（すなわち、全員分のデータが、縦に並ぶ）。

7 ユーザーフォームの作成

これまでに作成した、個票の印刷、ファイル読み込み、シートデータの統合のそれぞれの処理を順に実行するためのユーザーフォームを作成する。

7.1 ユーザーフォームとは

これまで使用したダイアログボックスを表示する MsgBox は、メッセージの内容とボタンの種類程度しか変更することはできなかった。オリジナルのメニューや、エクセルの表形式以外の出力をダイアログボックスとして用いる場合、「ユーザーフォーム」を作成して使用する。このフォームには、テキストやボタン、チェックボックスなどの部品（コントロール）を、自由な大きさ・位置に、貼り付けて使用することができる。

7.2 ユーザーフォームの作成

ここでは、これまでに作成した 3 つの処理を選択して実行するようなフォームを作成する。ここでは、**prog5.xlsm** を使用する。

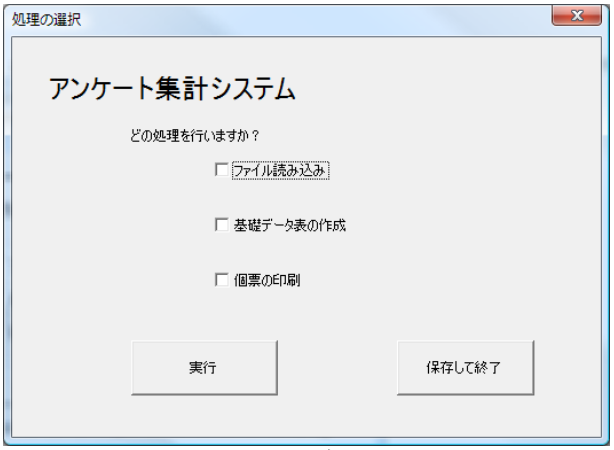
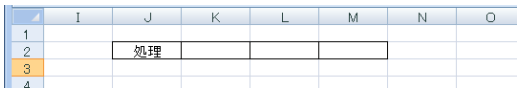


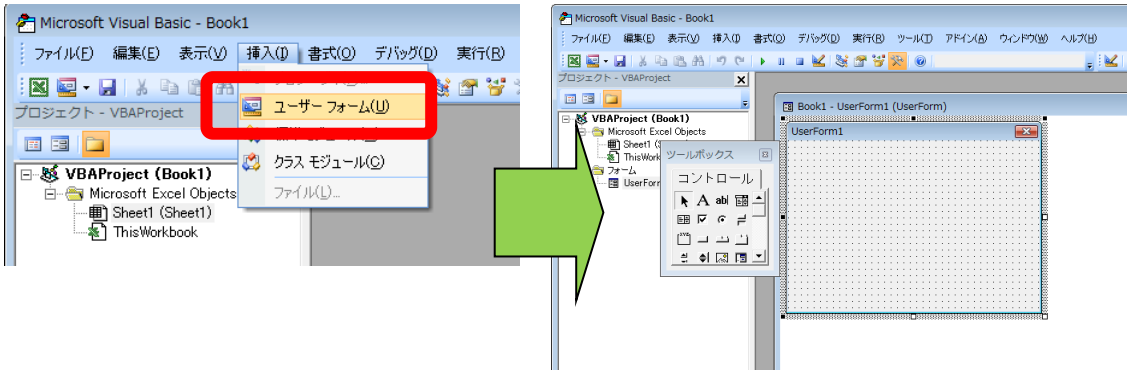
図 10　完成したユーザーフォーム

(1) シート「基礎データ」のセルJ2からM2に、以下のように入力し、設定する。

セル	内容	名前	罫線
J2	処理	（なし）	上下左右
K2	（空欄）	読込処理	上下左右
L2	（空欄）	表作成処理	上下左右
M2	（空欄）	印刷処理	上下左右

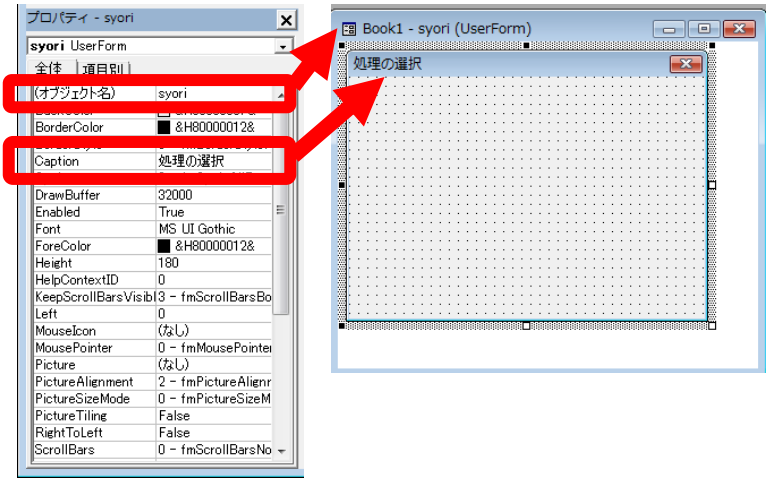


(2) メニューの「挿入」から、「ユーザーフォーム」をクリックする。

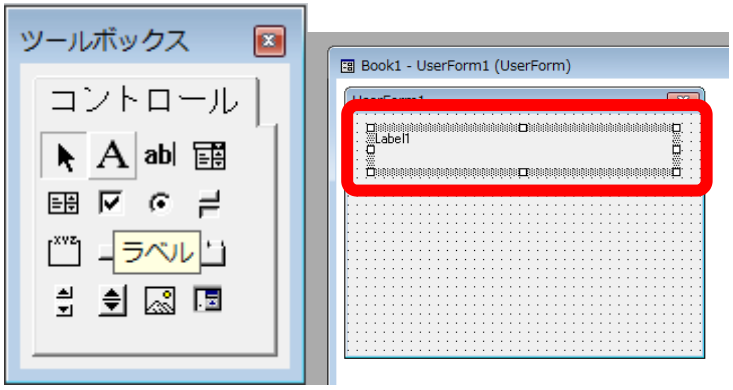


(3) フォーム内で右クリックし、プロパティを表示する。

(4) （オブジェクト名）を「syori」、Captionを「処理の選択」、Heightを「280」、Widthを「390」とする。



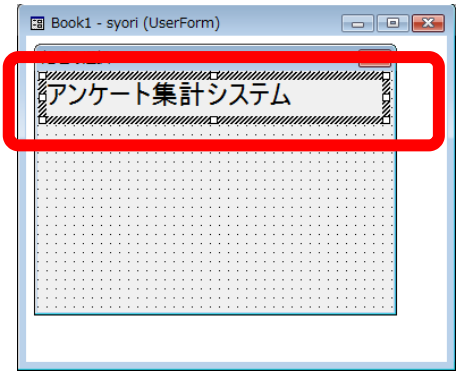
(5) ツールボックスから、A（ラベル）をクリックし、フォームにラベルを貼り付ける。さらに、プロパティで、次の表のように設定する。ラベルオブジェクトは、画面に値(Caption)を表示するためのものである。



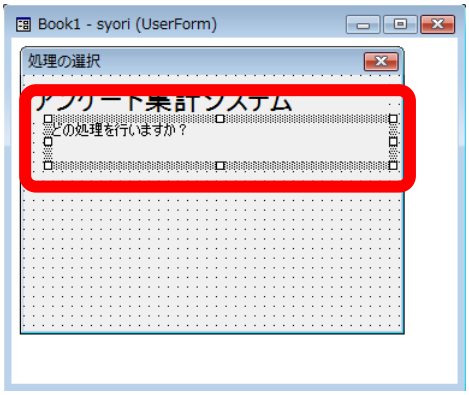
項目	値
Caption	アンケート集計システム
Font	MS UI Gothic、サイズ：18
Left	24
Top	24
Height	24
Width	204

※Left以下は参考値である。この通りでなくてもよい。

オブジェクトの
　左上の点の座標：(Left, Top)
　横縦のサイズ　：(Width, Height)

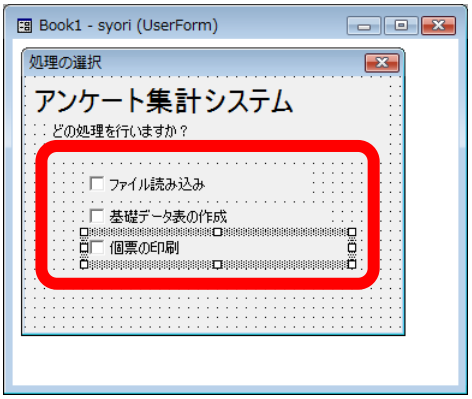


(6) 同様に、もう１つラベルを追加し、Captionを「どの処理を行いますか？」と設定する。

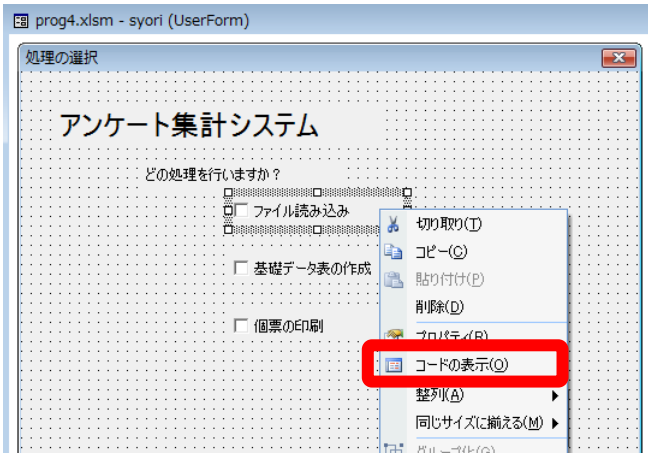


(7) 図のようにチェックボックスを３つ追加し、それぞれ以下のようにプロパティを設定する。チェックボックスは、項目を選択する場合に用いられ、チェックが入っている場合True、チェックが入っていない場合Falseの値を持つ。

	CheckBox1	CheckBox2	CheckBox3
(オブジェクト名)	readfile	maketable	printout
Caption	ファイル読み込み	基礎データ表の作成	個票の印刷
Height	15	15	15
Left	40	40	40
Top	60	80	100



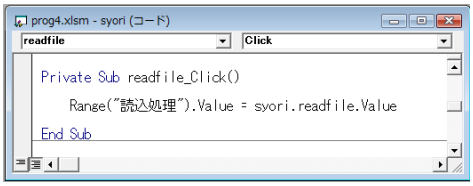
(8) チェックボックス「ファイル読み込み」で右クリックし、「コードの表示」を選択する。



なお、ここで表示されたコードウィンドウは、左のプロジェクトウィンドウを見ると分かる通り、フォーム syori に付与されるコードであり、これまでマクロを記述してきた標準モジュール Module1 とは異なる。すなわち、フォーム「syori」と標準モジュール「Module1」は、互いに相手の変数やモジュール等を直接参照することができない点に注意が必要である。

(9) 以下のコードを入力する。

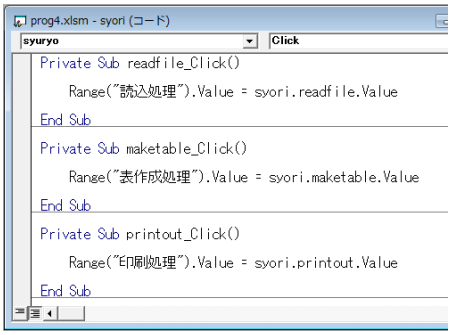
```
Private Sub readfile_Click()

    Range("読込処理").Value = syori.readfile.Value

End Sub
```



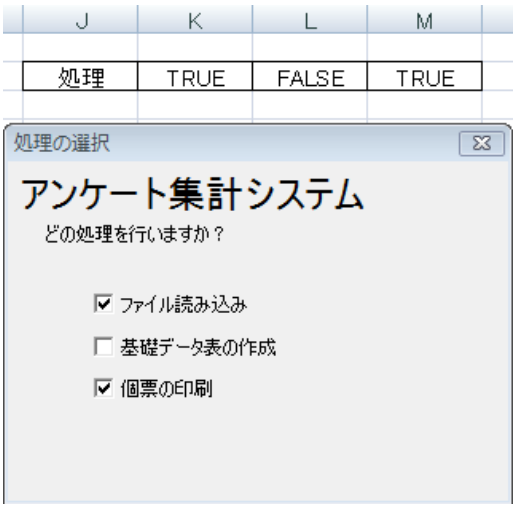
このコードは、シート「基礎データ」のセル K2 に(1)で付けた名前「読込処理」を利用し、そのセルの値を、syori.readfile（syori オブジェクトの、readfile オブジェクト、すなわち先ほど追加したチェックボックス）の値(Value)とする命令である。

(10) 同様に、チェックボックス「基礎データ表の作成」「個票の印刷」で、コードを入力する。

```
Private Sub maketable_Click()

    Range("表作成処理").Value = syori.maketable.Value

End Sub

Private Sub printout_Click()

    Range("印刷処理").Value = syori.printout.Value

End Sub
```

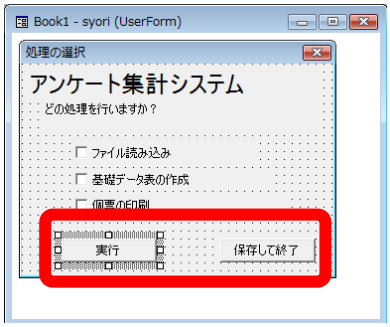


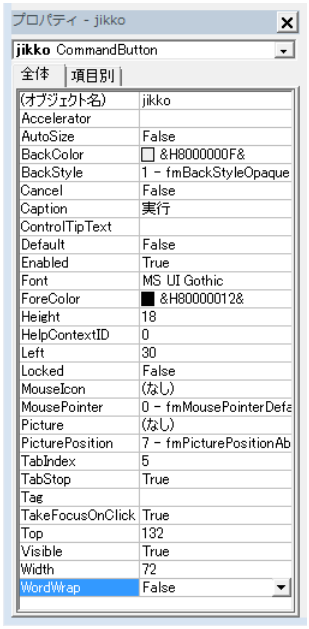
(11) ここまでで、フォームの動作を確認する。最初に、シート「基礎データ」のJ2からM2が見えるように、画面上に表示されているウィンドウを整理・移動する。続いて、フォーム「syori」のウィンドウを選択した状態で、実行ボタン ▶ をクリックすると、syoriフォームが実行状態で表示される。このウィンドウを、先ほどのJ2からM2の下に移動し、チェックボックスのチェックを入れると対応するセルがTRUEに、チェックを外すとFALSEにと、連動して動作することを確認する。終了は、右上の×をクリックする。

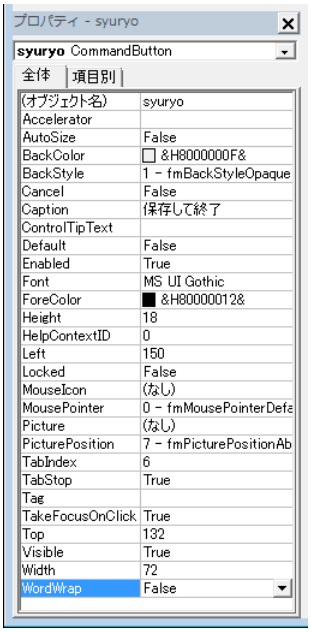


(12) フォームの下部に、コマンドボタンを２つ追加し、以下のように設定する。

	CommandButton1	CommandButton2
(オブジェクト名)	jikko	syuryo
Caption	実行	保存して終了
Height	18	18
Left	30	150
Top	132	132
Width	72	72







(13) 「実行」をダブルクリックしてコード入力画面を表示し、以下のコードを入力する。

```
Private Sub jikko_Click()

    Range("読込処理").Value = syori.readfile.Value
    Range("表作成処理").Value = syori.maketable.Value
    Range("印刷処理").Value = syori.printout.Value

    Unload Me

End Sub
```

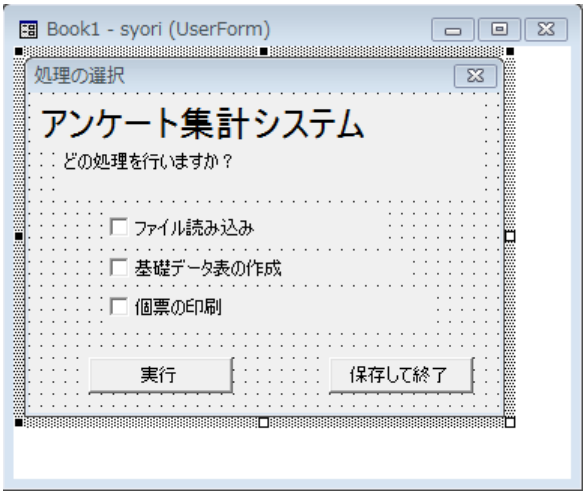
このマクロは、「実行」ボタンがクリックされた時の処理として、各チェックボックスの状態を改めてシートの該当領域に読み込み、フォーム自身(Me)をメモリから削除(Unload)することによって、このフォームの処理を終了する。

(14) 同様に「保存して終了」をダブルクリックし、以下のコードを入力する。

```
Private Sub syuryo_Click()

    ActiveWorkbook.Save              '保存する
    DoEvents                         'OS の処理の実行

    If Workbooks.Count <= 1 Then     'Excel を終了する
        Application.Quit
    End If
    DoEvents                         'OS の処理の実行

    ThisWorkbook.Close False         'Book を閉じる

End Sub
```

「保存して終了」では、最初に、現在のワークブックを保存(ActiveWorkbook.Save)し、残りのワークブックが１つ以下の場合は、エクセル自体を終了する。このために、Application.Quit と ThisWorkbook.Close をこの順に実行しなければならない。各処理の間には、念のため、DoEvents 命令を入れ、各処理を発行した後、OS がそのメッセージ処理を受け取り、実行するようにしている。

ここまでで、ユーザーフォーム syori の設定が終了する。コードおよびフォームは以下の通りとなる。



※Subはどの順序でもよい。

(15) 標準モジュールを選択し、次のコードを入力する。

```
Sub 開始()
    Sheets("基礎データ").Select
    Range("k2:m2").Value = False

    syori.Show

    If Range("読込処理").Value Then Call ファイルから読み込む
    If Range("表作成処理").Value Then Call 表の作成
    If Range("印刷処理").Value Then Call 印刷

End Sub
```

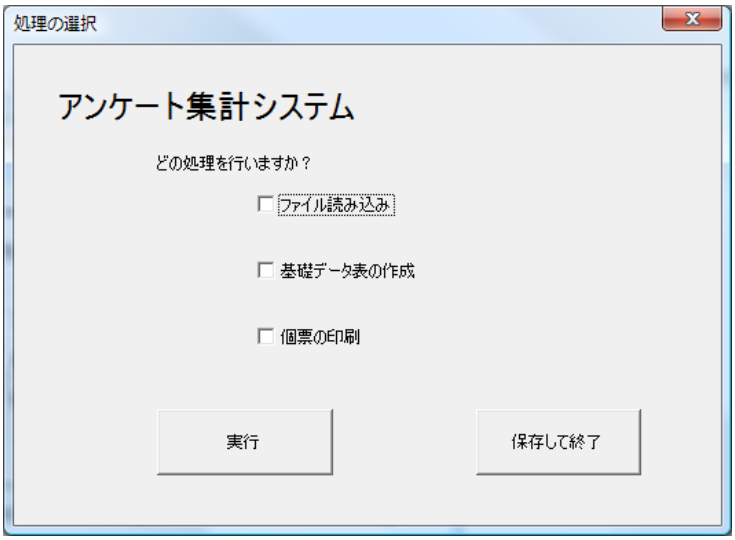
フォームを呼び出すために、ここでは、標準モジュールに、「開始」プロシージャを作成する。この中では、最初に、処理内容を記録する「基礎データ」シートを選択し、各処理の初期値として「False」を設定する。その後、フォーム syori を表示（Show、呼び出し）して、処理を委譲している。フォーム syori が終了すると、「開始」マクロが継続し、各処理の状態が True であれば、その処理を呼び出し(Call)ている。

(16) エクセルを起動したときに、自動的に「開始」プロシージャを実行するようにするため、以下のコードを標準モジュールに入力する。

```
Sub auto_open()

    Call 開始

End Sub
```

プロシージャ auto_open()は、特別なプロシージャ名で、このファイルを起動したときに、自動的に実行したい処理を記述することによって、自動で実行を開始することができる[23]。

(17) ここまで作成したエクセルのファイルを上書き保存して一度終了し、改めて起動して、ユーザーフォームが自動で起動することを確認する。



[23] この他にも、終了時に自動実行する Auto_Close などもある。

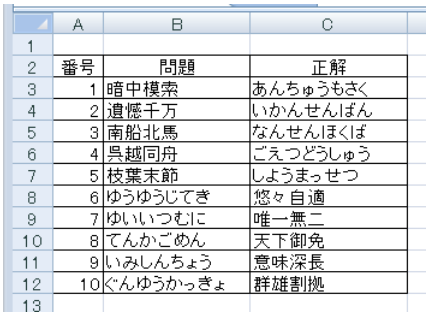
8 その他の例

8.1 問題と解答のシャッフル

あらかじめ用意されたいくつかの問題を、順番を変えて出題したい場合、用意された問題をシャッフルして作成すればよい。ここでは、このような問題用紙および解答を作成する手順について説明する。

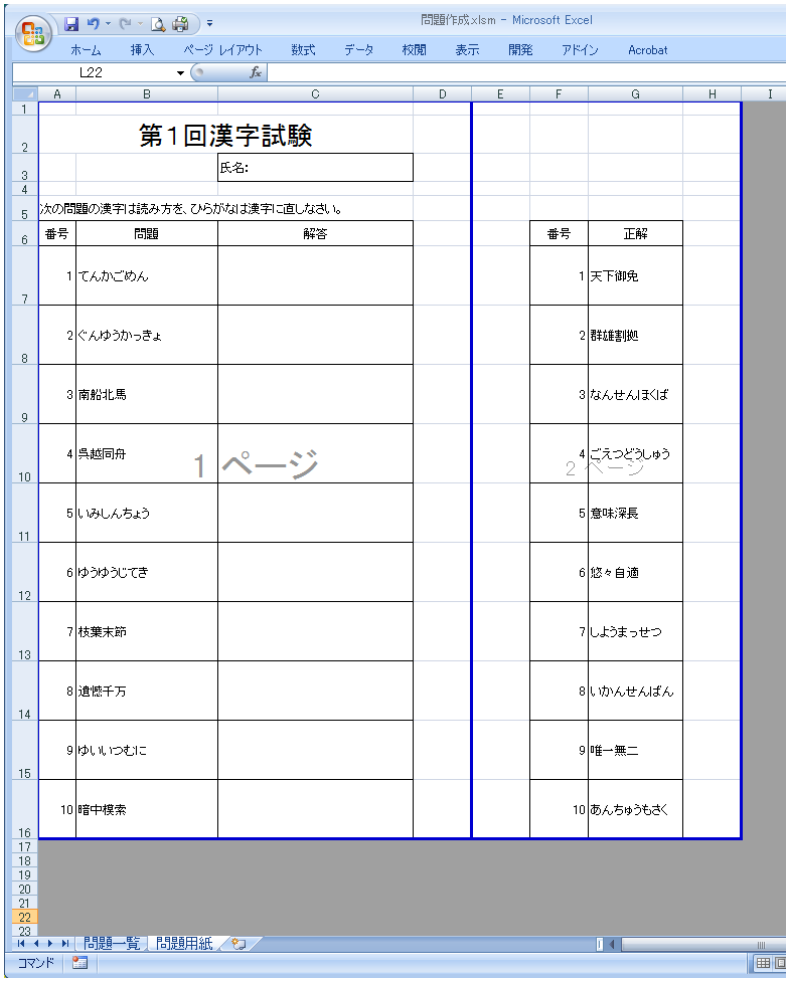
(1) 「新規作成」で新しいブックを作成する。ファイル名「問題作成.xlsm」として保存。

(2) シート「問題一覧」と「問題用紙」を作成し、「問題一覧」に問題を用意する。



番号	問題	正解
1	暗中模索	あんちゅうもさく
2	遺憾千万	いかんせんばん
3	南船北馬	なんせんほくば
4	呉越同舟	ごえつどうしゅう
5	枝葉末節	しようまっせつ
6	ゆうゆうじてき	悠々自適
7	ゆいいつむに	唯一無二
8	てんかごめん	天下御免
9	いみしんちょう	意味深長
10	ぐんゆうかっきょ	群雄割拠

(3) シート「問題用紙」を作成する。改ページプレビューで、試験問題を１ページ目、正解を２ページ目に設定する。



(4) 「開発」タブからVisualBasicを起動し、標準モジュールを挿入する。

(5) 標準モジュールに以下のコードを入力する。

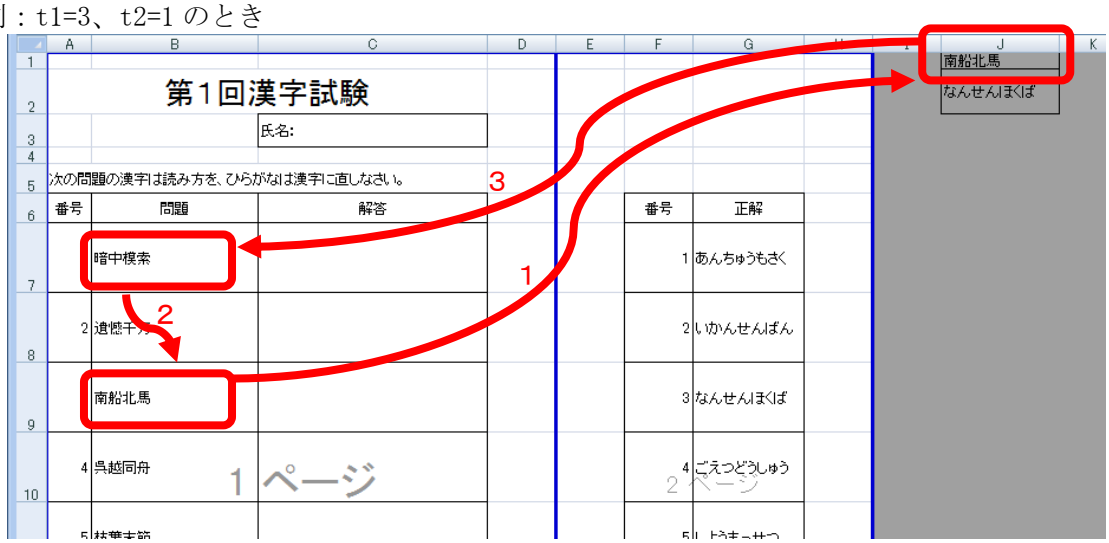
```
Sub 問題作成()
    Dim i As Integer                          '繰返し用変数

    Sheets("問題一覧").Range("b3:b12").Copy '問題の原本を
    Sheets("問題用紙").Range("b7:b16").PasteSpecial '問題用紙にコピー

    Sheets("問題一覧").Range("c3:c12").Copy '解答の原本を
    Sheets("問題用紙").Range("g7:g16").PasteSpecial '問題用紙の２枚目にコピー

    Sheets("問題用紙").Select                 '問題用紙を選択
    Randomize                                 '乱数を初期化

    For i = 1 To 100                          '以下の処理を１００回
        t1 = Int(Rnd() * 10) + 1              '１つ目の乱数生成
        t2 = Int(Rnd() * 10) + 1              '２つ目の乱数生成

        Range("B" & (t1 + 6)).Copy            '１つ目の問題をコピーして
        Range("j1").PasteSpecial              'ｊ１へ避難
        Range("G" & (t1 + 6)).Copy            '１つ目の正解をコピーして
        Range("j2").PasteSpecial              'ｊ２へ避難

        Range("B" & (t2 + 6)).Copy            '２つ目の問題を
        Range("B" & (t1 + 6)).PasteSpecial    '１つ目の欄に貼り付け
        Range("G" & (t2 + 6)).Copy            '２つ目の正解を
        Range("G" & (t1 + 6)).PasteSpecial    '１つ目の欄に貼り付け

        Range("j1").Copy                      '避難した問題を
        Range("B" & (t2 + 6)).PasteSpecial    '２つ目の欄に貼り付け
        Range("j2").Copy                      '避難した正解を
        Range("G" & (t2 + 6)).PasteSpecial    '２つ目の欄に貼り付け

    Next i                                    'ここまで繰り返す

End Sub
```

このマクロでは、問題の元となる「問題一覧」シートから、問題及び正解を「問題用紙」にコピーし、問題用紙側で問題と正解をいっしょに 100 回シャッフルしている。

シャッフルは、２つの乱数[24]を生成し、その値を問題番号として、該当する問題と正解を入れ替えている。問題と正解の入れ替えは、セルの値をコピーして貼り付けるだけで済むが、同時の交換はできないため、セル J1 と J2 を一時退避領域として使用し、１つ目の問題を J1 へ、空いた１つ目の問題領域に２つ目の問題をコピー、空いた J2 の問題領域に退避した問題をコピー、としなければならない。

[24] 次に出現する値が予測できない「でたらめな」数のこと。コンピュータは、「でたらめ」ができないため、計算式を使って乱数のように見える値を返す。このような乱数を疑似乱数と呼ぶ。また、この計算の最初に使われる値を設定することを「乱数の初期化」と呼ぶ。乱数値は、Single 型で、0 以上 1 未満の値を返す。

例：t1=3、t2=1 のとき



(6)　「問題作成」マクロを実行し、正しく実行されることを確認する。

問 8-1　Rnd 関数は、0 以上 1 未満の乱数値を返す。0 ～ 9 の整数の乱数値を得るためには、どのような式を用いればよいか考えなさい。

問 8-2　1 ～ 6 の整数の乱数値を得るためには、どのような式を用いればよいか考えなさい。

問 8-3　101～999 の整数の乱数値を得るためには、どのような式を用いればよいか考えなさい。

問 8-4　1 ～10 の乱数を 10 個生成し、その度数分布（出現回数）を求めよ。乱数を 100 個、1,000 個と増やした場合、度数分布はどのようになるか。

ヒント：数を数えるために、配列 a(10)を使うとよい。

問 8-5　1 ～10 の乱数 2 つ生成し、その和を求めよ。さらに、この乱数を 1,000 回発生させたときの度数分布を求めよ。どのような分布になっているか。

8.2 DLL の読み込み

さまざまな処理を行っていると、VBA の標準的な機能だけでは対応できないことがある。このような場合、外部のプログラム(ダイナミック[25]リンクライブラリ、DLL)を読み込んで利用することができる。DLL には、他言語（C 言語など）で作られたものや、ユーザが記述したもの、あらかじめ用意された Access などのデータベースとのやり取りができるようなものなど、非常に多くの機能が用意されている。

ここでは、簡単な例として、ゲームなどでよく使われる、その瞬間押されているキーの情報をリアルタイムに取得するための命令（関数）を説明する。

(1) 標準モジュールを用意し、次のマクロを入力する。

```
Declare Function GetAsyncKeyState Lib "User32.dll" (ByVal vKey As Long) As Long

Sub test()
    Dim a As Long

    Do While a = 0
        a = GetAsyncKeyState(37)      '「←」(左ボタンを押したときのキーコード)
    Loop

    MsgBox a
End Sub
```

最初の Declare…は、GetAsyncKeyState[26]という関数(引数として、値参照の Long 型変数 vKey を使い、返り値として Long 型を使う)を利用することを宣言しているが、この関数は、”User32.dll”ライブラリに存在しており、この dll を読み込むよう指示している。

GetAsyncKeyState 関数は、キーの番号となる整数の値（仮想キーコード）を引数として取り、そのキーが押されていなければ 0 を、押されていれば 0 以外の値(-32767)を返す関数である。キーとキーコードの対応は**【資料 18】**のとおりとなる。

Declare 命令の書式

Declare {Sub | Function} 関数名 Lib ″ライブラリ名″(引数) As 戻り値の型

{Sub | Function}：Sub か Function かいずれかを選ぶ
関数名：Declare によって定義される関数名
ライブラリ名：関数が含まれる外部のライブラリ名（～.dll 等）
引数：関数で使われる引数
戻り値の型：関数の戻り値の型（Function の場合のみ As 以下が必要）

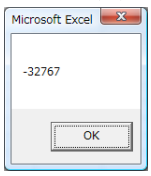
GetAsyncKeyState 関数の書式

GetAsyncKeyState(キー番号)

キー番号：キーボードのキーに割り当てられた長整数値
戻り値は、該当キーが押されていれば 0 以外の値、押されていなければ 0 を返す。

(2) マクロを実行し、任意のキーを押す。

※左矢印キーが押されるまでマクロは実行され続けます。



[25] 「ダイナミック」という言葉は、プログラムの実行中に、という意味で使われることが多い。これと対になる言葉として、プログラムの実行前に準備するものを、「スタティック」と呼ばれる。

[26] この関数は外部関数（C 言語で作られた関数と思われる）のため、大文字・小文字を区別して入力しなければならない。

問 8-6　User32.dll には、ここで使用した GetAsyncKeyState 以外にどのような関数が含まれているかインターネットを利用して調べなさい。

問 8-7　User32.dll 以外にどのような dll が存在するかインターネットを利用して調べなさい。

問 8-8　次のプログラムを入力し、実行しなさい（実行時には、エクセルのシート内容の変化に注意すること。また、キーボードの左矢印、右矢印を押してみること）。

```
Declare Function GetAsyncKeyState Lib "User32.dll" (ByVal vKey As Long) As Long

Sub dokutu()
    Dim x As Integer, s As Integer
    Dim w As Integer, p As Integer
    Dim r As Single

    Range("A1:CB80").Clear                                      '画面消去
    Columns("A:CB").ColumnWidth = ((18 * 72 / 96) - 3.75) / 6   'セル幅設定
    Range("A1").Select                                          'セルはA1を選択しておく

    x = 40                                                      '自分の横座標
    p = 30: w = 20                                              '穴の場所pと幅w
    s = 0                                                       '得点
    end_flag = 0                                                '衝突判定フラグ

    Do While end_flag = 0                                       '衝突してない間は繰り返す

        r = Rnd(1)                                              '乱数の生成
        If r < 0.3 And p > 1 Then                               '乱数が0.3未満で、穴の横位置が１より大きいなら
            p = p - 1                                           '穴の位置を左に１つずらす
        ElseIf r > 0.7 And p + w < 80 Then                      '乱数が0.7より大きくて、穴の右側が80より小さいなら
            p = p + 1                                           '穴の位置を右に１つずらす
        End If

        If GetAsyncKeyState(37) <> 0 And x > 1 Then x = x - 1   'キーボードの左矢印を確認
        If GetAsyncKeyState(39) <> 0 And x < 80 Then x = x + 1  'キーボードの右矢印を確認
        If GetAsyncKeyState(27) <> 0 Then end_flag = 1          'キーボードのESCが押されていたら終了

        Rows(1).Insert                                          'シートの最上行に１行挿入する

        Range("A1:CB1").Value = "■"                             '挿入した行を■で埋める

        If s Mod 20 = 0 And w > 4 Then w = w - 1                '点数が20点ごとに穴の幅を減らす(4まで)
        Range("A1").Offset(0, p).Resize(1, w).Value = ""        '■で埋めたセルを、穴の位置・幅で、消去する

        If Range("A30").Offset(0, x).Value <> "" Then end_flag = 1  '自分の位置に何かあったら、衝突
        Range("A30").Offset(0, x).Value = "A"                   '自分のキャラクターを描く
        s = s + 1                                               '点数を増やす

        DoEvents                                                '制御を一度Windowsに戻す

    Loop                                                        '衝突するまで繰り返す

    MsgBox ("終了です。得点は" & s & "点です")                  '終わったときの点数表示

End Sub
```

8.3 グラフィック操作

VBA では、フォーム上に直線や円、四角などをマクロから直接描画することはできない。このようにグラフィックを利用したい場合は、図（chart）を用いて、図形を描くことができる。

```
Sub graphics()
    Dim ch As Chart                                    ' 図オブジェクト
    Dim a As Shape                                     ' 図形オブジェクト
    Dim i As Integer                                   ' 繰り返し用変数
    Dim x As Integer, y As Integer                     ' 画面上の座標
    Dim r As Byte, g As Byte, b As Byte                ' 色

    Set ch = Charts.Add                                ' シートに図オブジェクトを追加

    For i = 1 To 500                                   ' ５００回繰り返す
        x = Int(Rnd() * 600)                           ' ｘ座標を乱数(０～599)で求める
        y = Int(Rnd() * 400)                           ' ｙ座標を乱数(０～399)で求める
        r = Int(Rnd() * 256)                           ' 色の赤要素を乱数(0～255)で求める
        g = Int(Rnd() * 256)                           ' 色の緑要素を乱数(0～255)で求める
        b = Int(Rnd() * 256)                           ' 色の青要素を乱数(0～255)で求める
        Set a = ch.Shapes.AddShape(msoShapeOval, x, y, 5, 5)     '円を図形として追加する
        a.Fill.ForeColor.RGB = RGB(r, g, b)            ' 塗りつぶしの色を設定する
        a.Line.ForeColor.RGB = RGB(r, g, b)            ' 輪郭線の色を設定する
        DoEvents                                       ' OS に動作を委譲する(描画する)
    Next i

End Sub
```

このマクロでは、最初に、図を描画するシートを追加し、５００回の繰り返しを行っている。繰り返す内容は以下の通りである。

①円を描く座標(x, y)、色(r, g, b)を乱数(rnd)で生成。x は 0～599、y は 0～399、r,g,b はそれぞれ 0～255 とする。

②addshape で、座標(x,y)に、サイズ(5,5)の円(msoShapeOval)を描画する。描画したオブジェクトを a とする。

③描画した図形 a の塗りつぶし色を(r, g, b)に変更する。

④描画した図形 a の線の色を(r, g, b)に変更する。

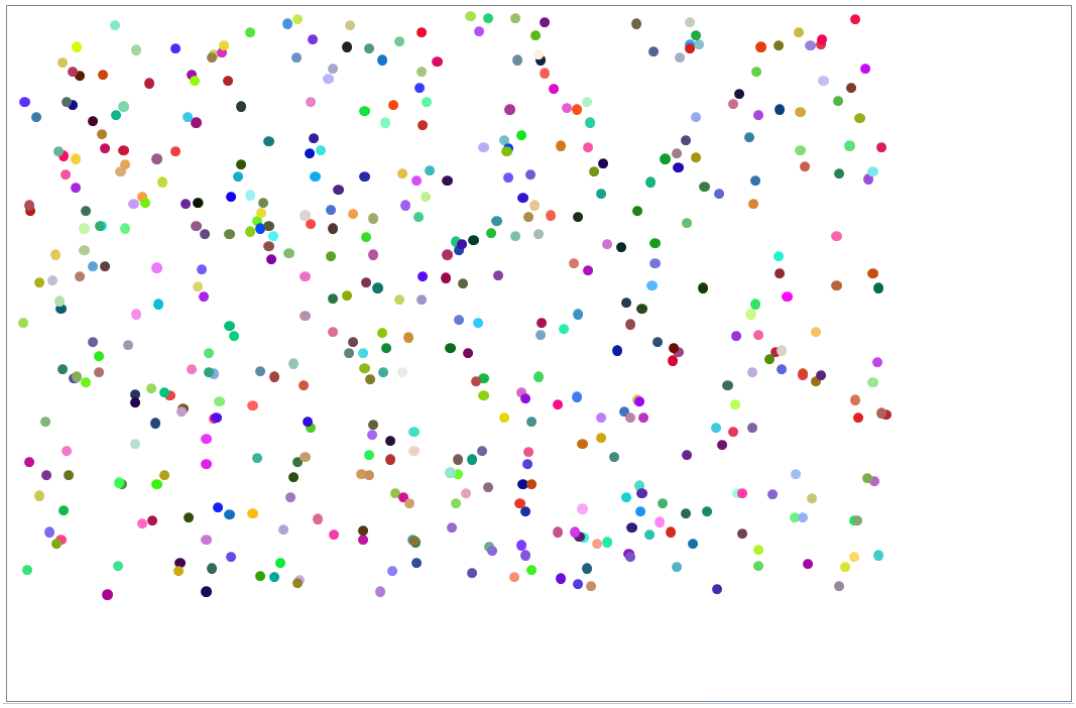


図 11　graphics の実行結果

ここでは、円(msoShapeOval)を描く例を取り上げたが、この他にも、四角形(msoShapeRectangle)、星(msoShape10PointStar)なども描画可能である。また、直線は、ch.Shapes.AddLine(*x1, y1, x2, y2*)で描画することができる（問 8-11)。

問 8-9　座標、大きさが異なる四角形を 100 個描くマクロを作成しなさい。

問 8-10　適当な座標を持つ直線を色を変えて 100 個描くマクロを作成しなさい。

問 8-11　次のマクロを入力し、実行しなさい。

```
Sub g2()
    Dim ch As Chart                 ' 図オブジェクト
    Dim a As Shape                  ' 図形オブジェクト
    Dim i As Integer                ' 繰り返し用変数

    Set ch = Charts.Add             ' シートに図オブジェクトを追加

    For i = 0 To 300 Step 5
        Set a = ch.Shapes.AddLine(i, 0, 0, 300 - i)
        Set a = ch.Shapes.AddLine(300, i, 300 - i, 300)
        DoEvents
    Next i

End Sub
```

8.4 ワードとパワーポイントのマクロ

ワードおよびパワーポイント等でも、エクセルと同様にマクロを利用することができる。その場合、そのマクロで扱うオブジェクト（エクセルの場合、セルやグラフ、ワードの場合は文章など）が変わるため、それらを操作するメソッドやプロパティ等が変更となる。ここでは、比較対象として、ワード及びパワーポイントの簡単なマクロの例を紹介する。

8.4.1　ワードのマクロ

以下のマクロでは、変数 tbl で指定された番号のテーブルの４行目以降を、１行おきに灰色にする。

```
Sub Macro1()

    tbl = 1

    On Error Resume Next

    ActiveDocument.Tables(tbl).Select
    If MsgBox("いいですか？", vbYesNo) = vbNo Then End

    For i = 4 To ActiveDocument.Tables(tbl).Rows.Count Step 2
        For j = 1 To ActiveDocument.Tables(tbl).Columns.Count
            ActiveDocument.Tables(tbl).Cell(i, j).Shading.BackgroundPatternColor = 14803425
        Next

    Next

    On Error GoTo 0

    MsgBox ("終わりました")

End Sub
```

このマクロの中で、On Error Resume Next は、マクロ実行中にエラーが発生した場合、エラー処理（エラーの表示やマクロの停止）を行わず、そのままその次の命令を実行する[27]ための命令であり、On Error Goto 0 は、エラーが発生した場合、通常のエラー処理（エラーメッセージ出力等）を行うように戻す命令である。

このマクロを、エクセルの場合と同様に、標準モジュールに記述し、実行ボタンで実行すればよい。ただし、文書には、大きめの表を用意しておく必要がある。実行結果の一例を以下に示す。

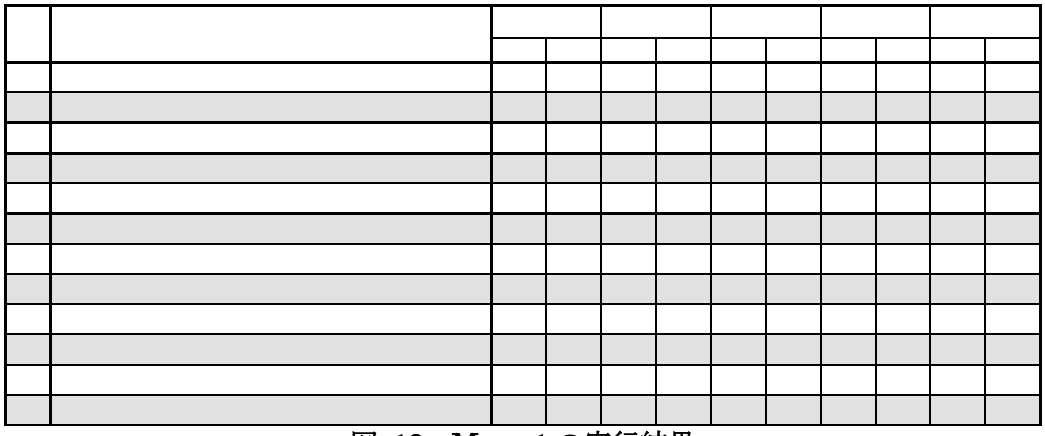


図 12　Macro1 の実行結果

[27] エラーが発生したときに、特定の位置（ラベルの位置）に移動することも可能(On Error Goto *Label* を使う)である。

8. 4. 2パワーポイントのマクロ

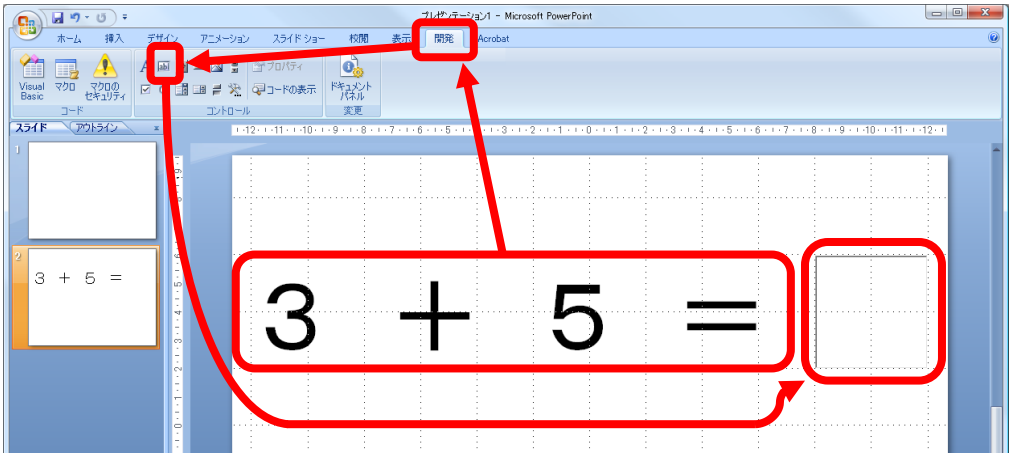
パワーポイントのマクロの例として、問題を出題してその答えをテキストボックスに記入し、正解不正解を判定して○または×のスライドへジャンプするマクロを示す。

(1) コントロールの配置

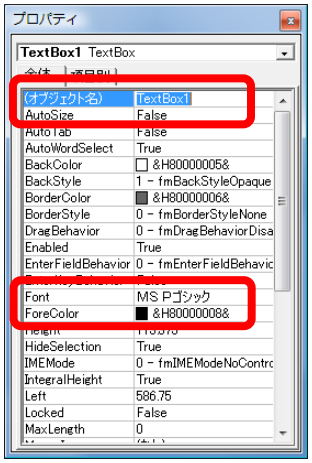
① パワーポイントを起動し、新しいスライドを１枚追加する。

② 問題を書く。96 ポイント等、できるだけ大きく表示する。

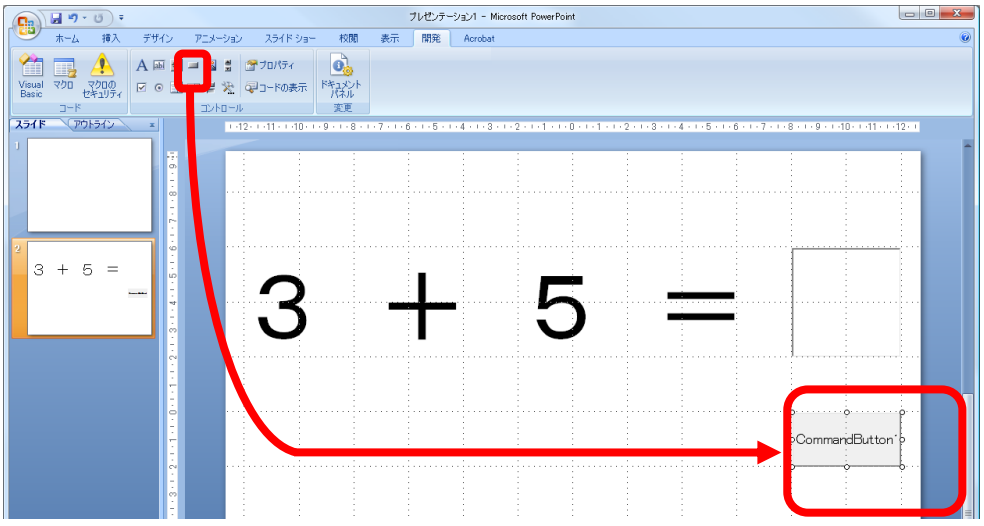
③ 開発タブのコントロールから、「テキストボックス」をクリックし、画面に配置する。



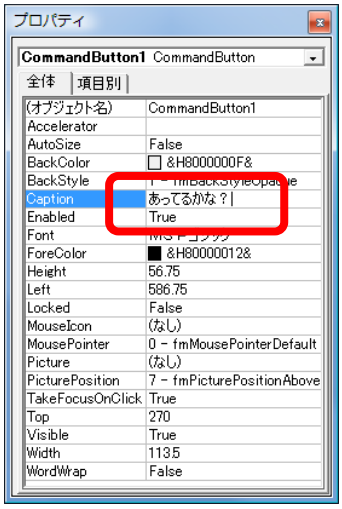
④ 配置したテキストボックス上で右クリックし、プロパティを選ぶ。テキストボックスのオブジェクト名（ここでは TextBox1）を確認し、さらに Font のサイズや色等を設定する。最後に、×をクリックしてプロパティの表示を消す。

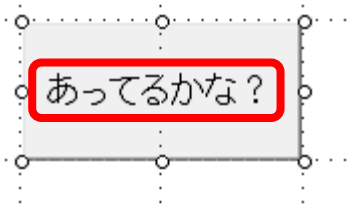


⑤ コントロールから「コマンドボタン」をクリックし、画面に配置する。

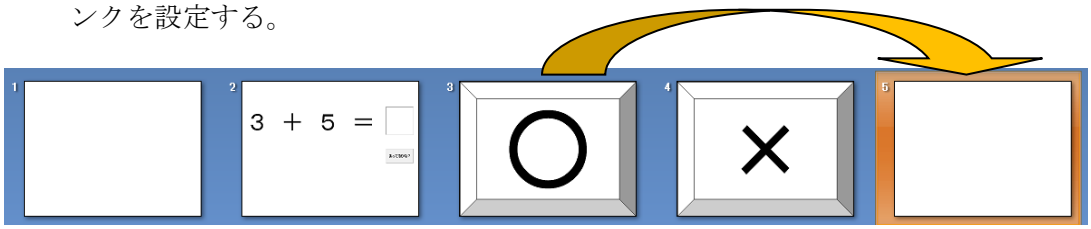


⑥　配置したコマンドボタンで右クリックし、「プロパティ」を表示させ、Caption 欄にボタンに表示する文字を設定する。自動的にボタンに表示された文字が変更される。



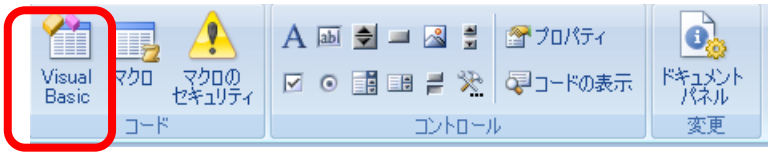


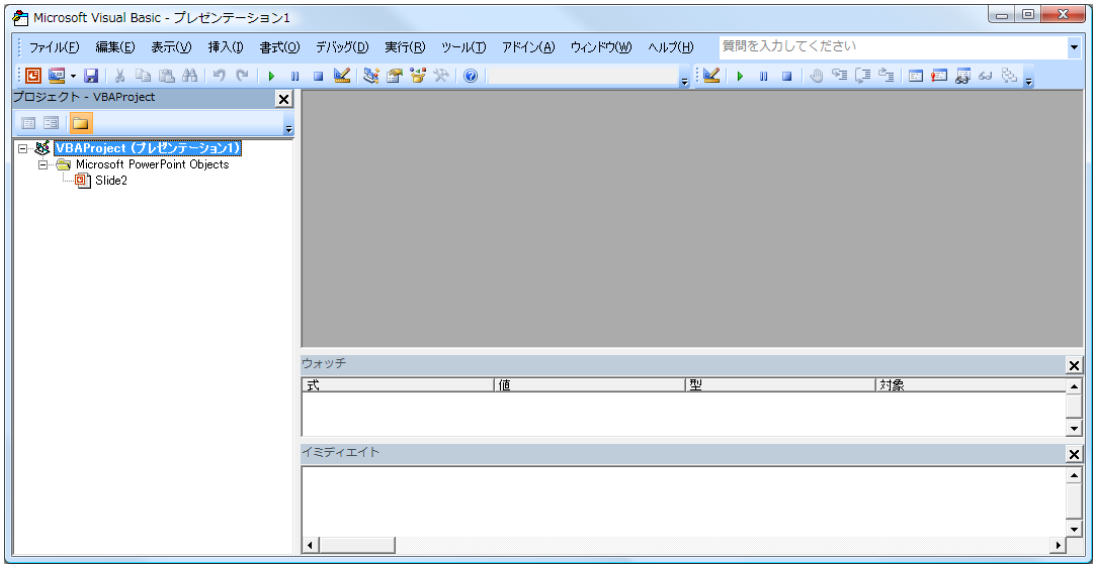
⑦　○、×、次の問題のシートを作成し、○のシートから次の問題に移動するようハイパーリンクを設定する。



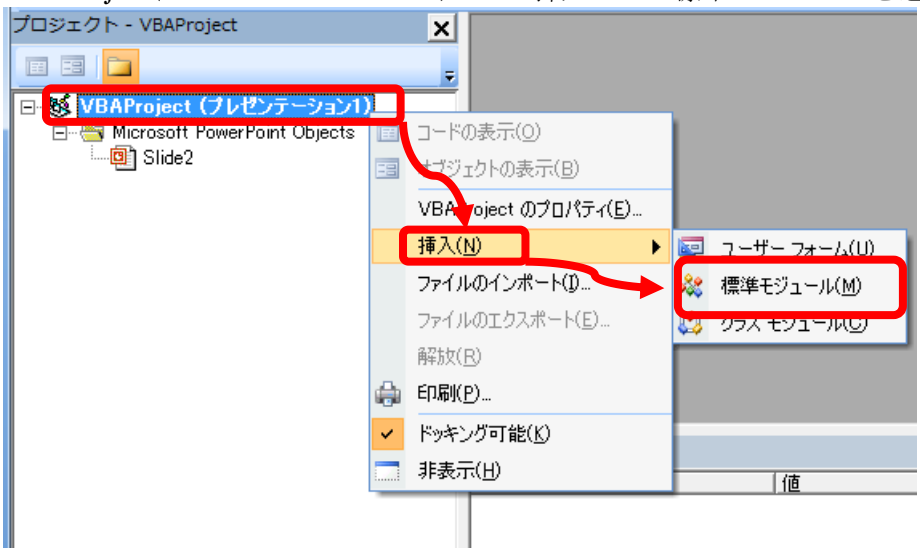
(2)　マクロの記述

①　コードの中の「Visual Basic」ボタンをクリックする。

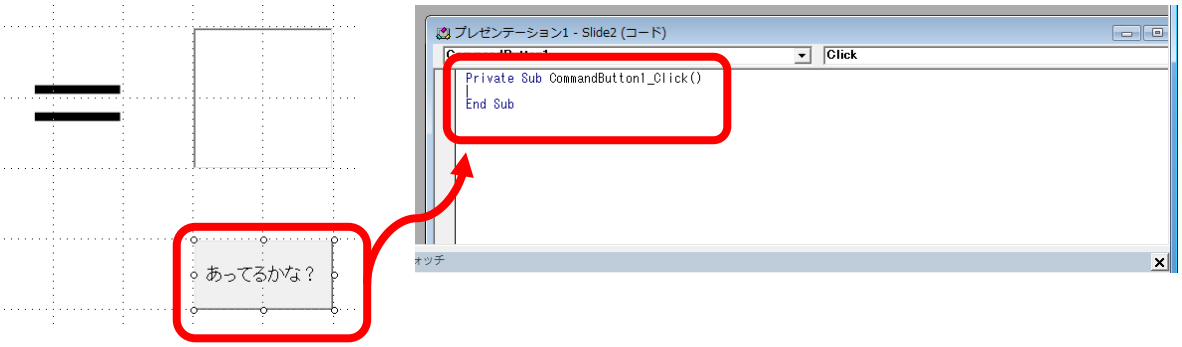




② VBAProject(プレゼンテーション１) → 挿入 → 標準モジュールを選択する。

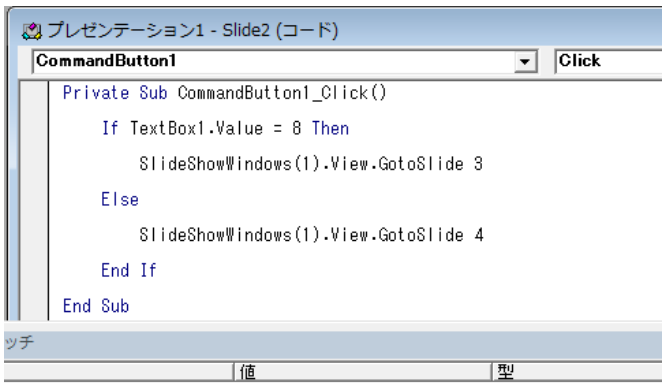


③ (1)の⑤で配置したボタンをダブルクリックし、コードウィンドウを表示する。



④ 以下のようなマクロを記述する。

```
Private Sub CommandButton1_Click()                ･･･ (a)
    If TextBox1.Value = 8 Then                    ･･･ (b)
        SlideShowWindows(1).View.GotoSlide 3      ･･･ (c)
    Else                                          ･･･ (d)
        SlideShowWindows(1).View.GotoSlide 4      ･･･ (e)
    End If                                        ･･･ (f)
End Sub                                           ･･･ (g)
```



マクロの意味

(a) コマンドボタン１がクリックされたら、ここから実行しなさい
(b) もし、TextBox1 の値が８と等しいならば（条件判断）、
(c) スライド３を表示しなさい
(d) そうでなければ（TextBox1 の値が８と等しくないならば）
(e) スライド４を表示しなさい
(f) 条件判断はここまで
(g) コマンドボタン１がクリックされたときに実行するのはここまで

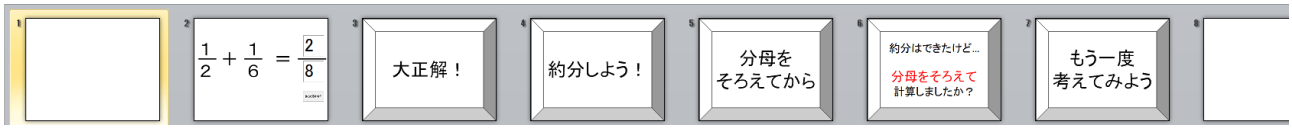
(3) 実行と確認

① スライドショーを実行する。

② 解答欄に８を入力して、「あってるかな？」ボタンをクリックする。〇が表示されるか確認する。

③ 解答欄に８以外の数字を入力して、「あってるかな？」ボタンをクリックする。×が表示されるか確認する。

問 8-12　分数の計算：次のような流れのパワーポイントのスライドとマクロを作成しなさい。



$\frac{1}{2}+\frac{1}{6}$ を表示し、以下の答えによって、場合分けする。

① 答えが $\frac{2}{3}$ のとき → 大正解！

② 答えが $\frac{4}{6}$ のとき → 約分しよう！

③ 答えが $\frac{2}{8}$ のとき → 分母をそろえてから！

④ 答えが $\frac{1}{4}$ のとき → 約分はできたけど、分母をそろえてから計算しましたか？

⑤ 答えがその他のとき → もう一度考えてみよう！

ヒント：

```
Private Sub CommandButton1_Click()
    If TextBox1 = 3 And TextBox2 = 2 Then
        SlideShowWindows(1).View.GotoSlide 3
    ElseIf TextBox1 = 6 And TextBox2 = 4 Then
        SlideShowWindows(1).View.GotoSlide 4
    ElseIf TextBox1 = 8 And TextBox2 = 2 Then
        SlideShowWindows(1).View.GotoSlide 5
    ElseIf TextBox1 = 4 And TextBox2 = 1 Then
        SlideShowWindows(1).View.GotoSlide 6
    Else
        SlideShowWindows(1).View.GotoSlide 7
    End If
End Sub
```

8.5 その他のプログラム

以下に Excel のマクロで動作するプログラムの例を示す。様々な種類のプログラムを、(本テキストからのコピーアンドペーストではなく) 実際にキーボードから入力して、デバッグし、動作を確認することにより、マクロやアルゴリズムに対する一層の理解が進むことが期待できる。

例１

```
Sub sosu()

    Dim i As Integer, j As Integer
    Dim y As Integer

    '素数判定のために、倍数をすべて消していく（エラトステネスのふるい）
    Range("A1:B20000").Clear        'すべての範囲をクリア
    Cells(1, 1) = 1                 '1は素数ではないのでフラグを立てる

    For i = 2 To 10000
        For j = 2 To 10000 / i
            Cells(i * j, 1) = 1     'i の j 倍は素数ではないのでフラグ立て
        Next j
    Next i

    '一覧表作成
    y = 1
    For i = 1 To 10000              '10000 までの数のうち
        If Cells(i, 1) <> 1 Then    'フラグが立ってないのは素数なので
            Cells(y, 2) = i         'Ｂ列にその値を入れて
            y = y + 1               '行を１つ増やす
        End If
    Next i

End Sub
```

例２

```
Sub ketueki()
    Dim aisyo As String, s As String
    Dim k1 As Integer, k2 As Integer
    Dim ans As Integer

    aisyo = "4645605245335231"                                          '相性の定義 A,O,B,AB の順

    s = "の血液型を選択してください" & vbCr                                '入力用プロンプトの作成
    s = s & "  1...A" & vbCr & "  2...O" & vbCr
    s = s & "  3...B" & vbCr & "  4...AB"

    k1 = Application.InputBox(prompt:="あなた" & s, Title:="血液型占い", Type:=1)   '１人目の入力（数値のみ）
    k2 = Application.InputBox(prompt:="相手" & s, Title:="血液型占い", Type:=1)     '２人目の入力（数値のみ）

    ans = Val(Mid(aisyo, (k1 - 1) * 4 + k2, 1))                         '相性の定義から値を取得
    ans = ans * 10 + 40                                                 '値６が 100%になるよう換算
    MsgBox "二人の相性は " & ans & "% です"                               '結果表示

End Sub
```

例３

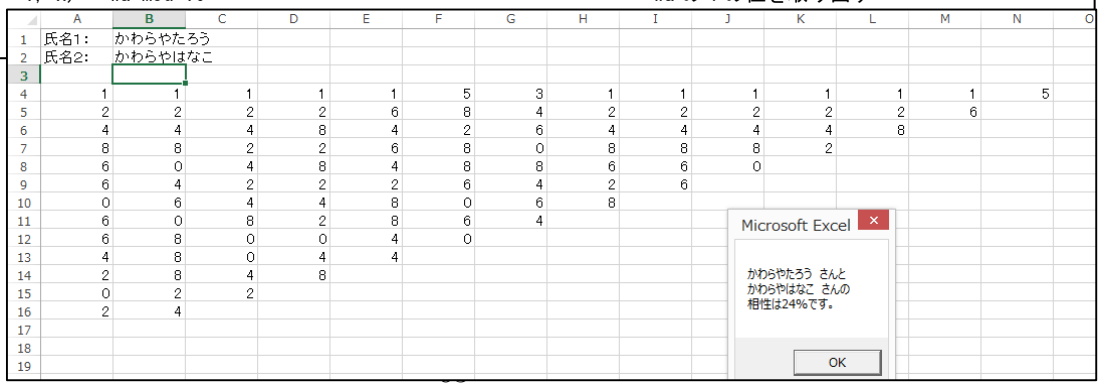
```
Sub maze()
    Dim x As Integer, y As Integer                                      '画面上の道の作り始めの座標(x, y)
    Dim x1 As Integer, y1 As Integer                                    '伸ばした道の座標(x1, y1)
    Dim sx(4) As Integer, sy(4) As Integer                              '進行方向と座標の関係
    Dim s As Integer                                                    '進行方向
    Dim siz As Integer

    Randomize                                                           '乱数の初期化
    siz = 100                                                           'サイズ

    sx(0) = 0: sx(1) = 0: sx(2) = 2: sx(3) = -2             '進行方向のｘ方向の差分(0:下, 1:上, 2:右, 3:左)
    sy(0) = 2: sy(1) = -2: sy(2) = 0: sy(3) = 0                         '進行方向のｙ方向の差分

    Range(Columns(1), Columns(100)).ColumnWidth = 1                     'セルの幅を 10 ポイントに
    Range(Rows(1), Rows(100)).RowHeight = 10                            'セルの高さを 10 ポイントに
    Range("a1:cz100").Interior.ColorIndex = 2                           '100x100 サイズのセルをすべて白に

    Range(Cells(2, 2), Cells(siz - 2, siz - 2)).Interior.ColorIndex = 1 '壁で埋める

    Cells(3, 3).Interior.ColorIndex = 2                                 'スタート位置
    Cells(3, 2).Interior.ColorIndex = 2                                 '入口
    Cells(siz - 3, siz - 2).Interior.ColorIndex = 2                     '出口

    For x = 3 To siz - 3 Step 2                                         '横方向の繰り返し
         For y = 3 To siz - 3 Step 2                                    '縦方向の繰り返し
            x1 = x                                                      '伸ばしていく道のｘ座標
```

（次のページへ続く）

```
            y1 = y                                                      '伸ばしていく道のｙ座標

            Do
                                                                        '周囲の状態を取得
                p = Cells(y1, x1 - 2).Interior.ColorIndex + Cells(y1, x1 + 2).Interior.ColorIndex _
                  + Cells(y1 - 2, x1).Interior.ColorIndex + Cells(y1 + 2, x1).Interior.ColorIndex

                If p <> 8 Then                                          '８の時は、袋小路
                    Do
                        s = Int(Rnd() * 4)                              '進行方向の検索。道でないところを探す
                    Loop While Cells(y1 + sy(s), x1 + sx(s)).Interior.ColorIndex = 2

                    Cells(y1 + sy(s), x1 + sx(s)).Interior.ColorIndex = 2       '道を伸ばす
                    Cells(y1 + sy(s) / 2, x1 + sx(s) / 2).Interior.ColorIndex = 2   '道を伸ばす
                    x1 = x1 + sx(s)                                     '伸ばしていく道のｘ座標の更新
                    y1 = y1 + sy(s)                                     '伸ばしていく道のｙ座標の更新
                    DoEvents                                            'Windows に渡して描画する
                End If
            Loop While p <> 8                                           '袋小路に入るまでは繰り返す
        Next y
    Next x

    MsgBox ("完成しました")                                             '完成のお知らせ
End Sub
```

例４

```
Sub uranai()                                                            '実行開始
    Dim y As Integer, nagasa As Integer                                 'セル B1 と B2 にひらがなで
    Dim kekka As Integer                                                '   名前を入れておくこと
    Dim m As String

    nagasa = str2num()                                                  'ひらがなを数字に変換

    For y = 1 To nagasa - 2                                             '２ケタになるまで繰り返す
        Call keisan(y + 3, nagasa - y)                                  '１行計算
    Next y

    kekka = Cells(y + 3, 1) * 10 + Cells(y + 3, 2)                      '結果の計算(10 の位、１の位)
    m = Range("B1").Value & "  さんと" & vbCr                           '表示する文字列を作成
    m = m & Range("B2").Value & "  さんの" & vbCr
    m = m & "相性は" & kekka & "%です。"
    MsgBox m                                                            '結果の表示
End Sub

Function str2num() As Integer                                           'ひらがなを数字に変換
    Dim st As String, moji As String
    Dim i As Integer, n As Integer

    st = Range("B1").Value & Range("B2").Value                          'セル B1 と B2 に名前が入っているので結合
    Range("A4:AZ50").Clear                                              '計算する場所をクリア

    For i = 1 To Len(st)                                                '２人の文字数分繰り返す
        moji = Mid(st, i, 1)                                            'ｉ番目の文字を取得
        n = 0                                                           'ひらがなが不明ならゼロとする
        If InStr("あかさたなはまやゃらわんがざだばぱ", moji) > 0 Then n = 1 '「あ…ぱ」なら値は１
        If InStr("いきしちにひみいりいぎじぢびぴ", moji) > 0 Then n = 2     '「い…ぴ」なら値は２
        If InStr("うくすつっぬふむゆゅるうぐずづぶぷ", moji) > 0 Then n = 3 '「う…ぶ」なら値は３
        If InStr("えけせてねへめえれえげぜでべぺ", moji) > 0 Then n = 4     '「え…ぺ」なら値は４
        If InStr("おこそとのほもよょろをごぞどぼぽ", moji) > 0 Then n = 5   '「お…ぽ」なら値は５

        Cells(1 + 3, i).Value = n                                       'セルに値を設定
    Next i

    str2num = Len(st)                                                   '関数の値は文字数とする
End Function

Sub keisan(y As Integer, ngs As Integer)                                '１行分の計算(y:行、ngs:長さ)
    Dim x As Integer, wa As Integer

    For x = 1 To ngs                                                    'Ａ列から右端まで繰り返す
        wa = Cells(y, x) + Cells(y, x + 1)                              'x 番目と x+1 番目を加算
        Cells(y + 1, x) = wa Mod 10                                     'wa の１の位を取り出す
    Next x
End Sub
```



9 資料編

【資料 1】

	内部バージョン	備考
Visual Basic 6.0	6.0	
Visual Basic .NET	7.0	.NET フレームワーク
Visual Basic 2005	8.0	
Visual Basic 2008	9.0	
Visual Basic 2010	10.0	
Visual Basic 2012	11.0	
Visual Basic 2015	12.0	
Visual Basic 2017	14.0	忌み番号により 13 をスキップ
Excel 2007 VBA	6.5	Visual Basic 6.0 とほぼ同等
Excel 2010 VBA	7.0	
Excel 2013 VBA	7.X	
Excel 2016 VBA	7.1	

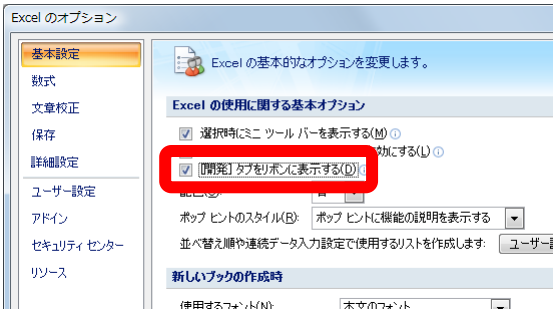
Wikipedia より

【資料 2】

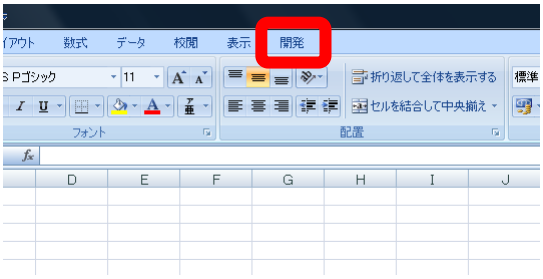
Excel 2007の場合

(1) 左上のオフィスボタン  をクリックし、最下行の「Excelのオプション」をクリックする。

(2) 左のメニューから「基本設定」を選び、「[開発]タブをリボンに表示する」にチェックを入れる。



(3) 「OK」をクリックすると、「開発」タブが表示される。



【資料 3】

以前は、１行ずつ命令を解読・処理していく「インタプリタ型」の BASIC 言語が多く、また、コンピュータの処理速度が遅かったため、このようにたくさんの命令を１行に記述して読み込む回数を減らして、速度向上を図っていた。

現在は、コンピュータの処理速度が十分速くなったこと、オブジェクト指向型言語や「コンパイラ型」（命令をすべて解読・翻訳してから実行する）の環境が増え、また、マルチ ステートメントはプログラムが読みにくく（複雑な制御構造が記述されているプログラムを「スパゲッティ プログラム」と呼ぶ）、継続的な保守が困難になるとの考えが広まったことなどから、現在は用いられなくなった。

【資料 4】

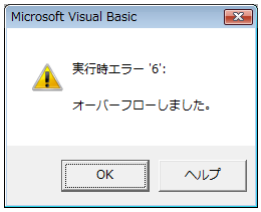
名　称	値の範囲	サイズ(バイト)	型宣言文字(半角)	備　考
ブール型 (Boolean)	真偽値(True / False)	2		変数への代入 0:False、0以外:True 他の変数型への代入 False=0、True=－1
バイト型 (Byte)	0～255の整数	1		0または正の値
整数型 (Integer)	－32,768 ～ ＋32,767	2	%	
長整数型 (Long)	－2,147,483,648 ～ ＋2,147,483,647	4	&	
単精度浮動小数点型 (Single)	-3.4×10^{38} ～ -1.4×10^{-45} $+1.4\times10^{-45}$ ～ $+3.4\times10^{38}$	4	!	IEEE準拠
倍精度浮動小数点型 (Double)	-1.8×10^{308} ～ -4.9×10^{-324} $+4.9\times10^{-324}$ ～ $+1.8\times10^{308}$	8	#	IEEE準拠
文字列型 (String)	全角／半角の文字	最大約2GB	$	約21億文字まで
日付型(Date)	西暦100年1月1日 ～ 西暦9999年12月31日 00:00:00 ～ 23:59:59	8		1899年12月31日00:00:00をゼロとし、整数部分が日付、小数部分を時刻を表す
オブジェクト型 (Object)	Worksheetオブジェクト Rangeオブジェクト　等	4		オブジェクトへのアドレス Set命令を使って参照を代入
通貨型 (Currency)	－922,337,203,685,477.5808 ～ 922,337,203,685,477.5807	8	@	金額に関する計算に使用 15桁の整数部分と４桁の小数部分
10進型 (Decimal)	最大絶対値 79,228,162,514,264,337,593,543,950,335 小数点以下28桁の場合の最大絶対値 7.922 816 251 426 433 759 354 395 033 5 絶対値の最小値(0を除く) 0.000 000 000 000 000 000 000 000 000 1	12		整数部分の桁数によって、小数部分は0～28桁に変化する 10進数で計算するため、範囲内であれば誤差がない
バリアント型 (Variant)	ユーザ定義型および一部の文字列型を除くあらゆるデータ型 特殊型（Empty値、Null値、エラー値、Nothing型等）も格納可能			データ型が宣言されていないすべての変数 VarType関数で型を調べられる
ユーザ定義型				Type命令を使って作成した型。 異なるデータ型（ユーザ定義型を含む）を１つ以上組み合わせたもの

※Excel のヘルプより

【資料 5】

変数ではなく、定数を使用するときも、型の考え方は必要である。例えば、イミディエイトウィンドウで、以下のように入力して実行すると、右のようなエラーが発生する。

```
print 1000*1000 Enter
```



これは、入力した 1000 は、ともに整数型(32767 以下の値を持つ)として扱われ、整数型同士の掛け算を行うため、計算結果も整数型として扱われるが、その範囲を超えてしまうためである。

これを避けるためには、長整数型を利用すればよく、この場合は、以下のように、どちらかの値に長整数型を表す「&」を付加すればよい。

```
print 1000*1000& Enter
1000000
```

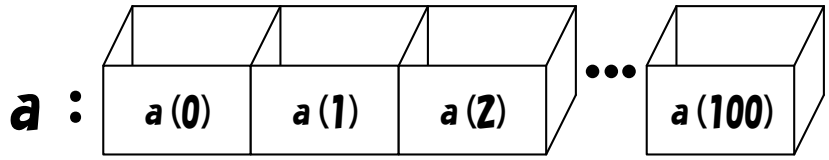
【資料 6】

配列について

４０名のクラスの生徒の成績など、同一型の変数が多数必要となった場合、それぞれの出席番号ごとに変数を一つ一つ宣言するのは効率的ではない。このような場合、配列(Array)を用いることによって、同一型の変数を（メモリが許す限り）必要な数だけ確保することができる。配列を使う場合は、Dim 命令を使用して以下のような宣言をする。

```
Dim a(100) As Integer
```

この宣言によって、101 個分[28]の整数型変数の箱を確保し、a と名前を付けたことになる。それぞれの箱（要素と呼ぶ）を参照するには、a(0)、a(1)、a(2)、a(3)、……、a(100)のように記述する。変数の型や動作については、他の変数と全く同様に扱うことができる。



問 9-1　文字列型変数 b を 11 個確保するための命令を記述しなさい。

問 9-2　倍精度型浮動小数点数型変数 c を 10,000 個確保するための命令を記述しなさい。

上記の配列は、箱が一列に並んだ「１次元配列」であるが、下駄箱のような２次元配列や、奥行きを持つ３次元配列、イメージできるかどうか分からないが４次元配列などを使用することもでき、VB では最大 60 次元まで使用可能となっている。２次元配列 c の宣言は、以下のようになる。

```
Dim c(100,100) As Integer
```

この宣言によって、101×101 個分、計 10,201 個の整数変数の箱を確保し、c と名付けたことになる。同様に、３次元以上の配列もカッコ内にカンマで区切ることによって宣言することが可能となる。

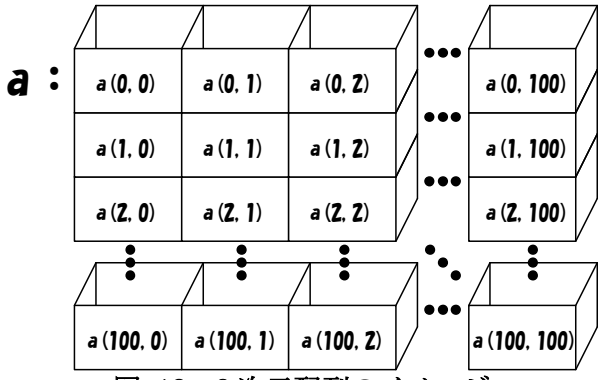


図 13　２次元配列のイメージ

配列は、繰り返し処理（3.3 節）とともに使用すると、その威力を発揮することができる。

[28] 番号＝添え字が０から始まるため。「Option Base 1」と設定すると１からとなる。

【資料 7】

変数の有効範囲

宣言した変数が有効となる範囲をスコープという。例えば、Sub sample1()内に Dim 宣言を置いて宣言した場合、その変数は、その Sub～End Sub の範囲でのみ参照可能（プロシージャレベル変数と呼ぶ）であり、他の Sub からは参照できない。他の Sub から参照する必要がある場合は、Sub の外（一般的にマクロの Option 行の次）に変数宣言（モジュールレベル変数と呼ぶ）を記述すればよい。以下の図では、a がモジュールレベル変数、b がそれぞれプロシージャレベル変数である。

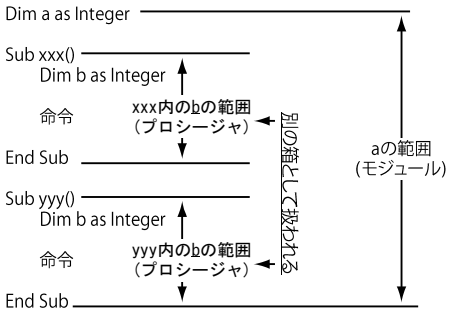


図 14　変数のスコープ

さらに、別のモジュール（同一エクセルファイル内の、別のコードファイル）から参照する必要がある場合は、Dim 命令を用いずに、モジュールレベル変数に Public 宣言（パブリックモジュールレベル変数）を、別のモジュールから参照する必要がない場合は Private 宣言（プライベートモジュールレベル変数）しなければならない。

変数の宣言と同様に、Sub や Function の宣言も、Private や Public として宣言することがある。

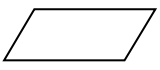
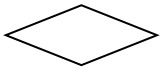
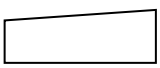
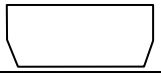
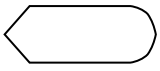
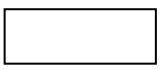
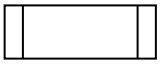
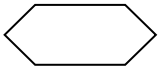
```
例
  Public a as Integer
  Private b as Integer

  Sub xxx()
      Dim c as Integer

  End Sub

  Sub yyy()
      Dim d as Integer

  End Sub
```

	変数 a	変数 b	変数 c	変数 d
他のモジュールから	○	×	×	×
このモジュール内	○	○	×	×
xxx 内	○	○	○	×
yyy 内	○	○	×	○

図 15　モジュールレベル変数のスコープ

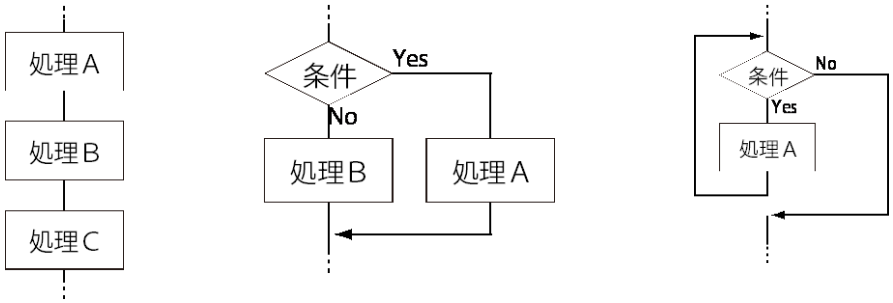
【資料 8】

フローチャートで用いられる記号は以下の通り（JIS X0121 より一部抜粋）。

データ		判断	
書類		ループ始端	
手操作入力		ループ終端	
表示		線	
処理		通信	
定義済み処理		結合子	
準備		端子	

【資料 9】

アルゴリズムで用いられる構造としては、以下の３つの要素が使われ、この３つの構造ですべてのプログラム（アルゴリズム）が記述できることが証明されている。一般的に、条件分岐は if 文を、繰り返しは for 文や while 文を用いることが多い。



逐次　　条件分岐　　繰り返し

図　アルゴリズムで用いられる構造

① 逐次（順次） … 記述された順番に、（上から下へ）実行する。
② 条件分岐 … ある条件が成立するなら処理Ａを、成立しなければ処理Ｂを実行する。
③ 繰り返し … ある条件が成立している間（成立してない間）、処理Ａを繰り返す。

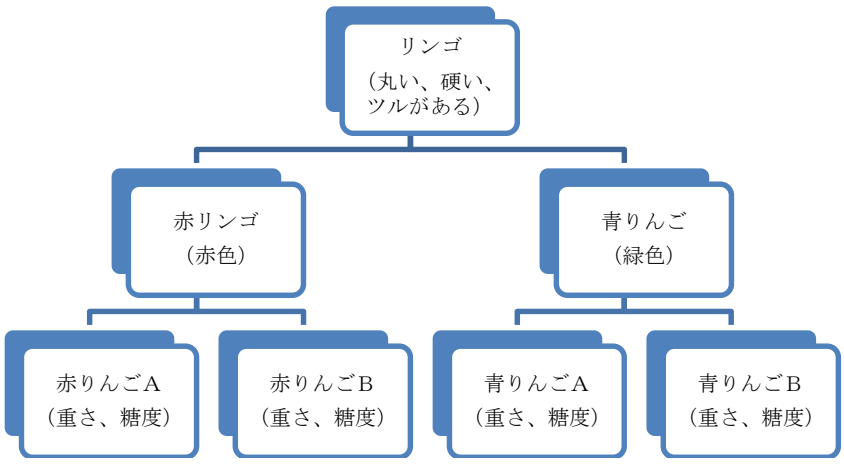
【資料 10】

よく使われるオブジェクトの例として、「リンゴ」を考える場合がある。リンゴは、一般的に、丸い、硬い、上からツルが出ているなどの特徴がある一方で、赤りんごや青りんごなどの見た目の違い、さらに、同じ赤りんごでも、１つ１つ見れば、重さや糖度などが異なる。赤りんごや青りんごは、リンゴ全体の特徴をもち（継承し）、赤りんご１つ１つは重さや糖度は異なるが、赤りんごの特徴を継承している。

一方、リンゴを生のまま食べる、焼く料理、煮る料理など、リンゴに対する料理（操作）は、いろいろある。赤りんごでできるが、青りんごでできない料理（操作）もある（と思う）。

このように、「オブジェクト」には、そのオブジェクトの特徴をデータとして保持し、そのオブジェクトに対して可能な操作が定義されている。これらのデータと操作を合わせて、「クラス」と呼ぶ。すなわち、リンゴというクラスには、特徴を表す重さや糖度を表す「プロパティ」と、焼く、煮るなど、そのオブジェクトに対してできる操作「メソッド」が記述されており、同じクラスでは、これらの特徴が継承されている。一方で、個別の重さや操作などをそれぞれのクラスに設定することもできる。

オブジェクトに属するプロパティやクラスは、「.」（半角小数点）で区切られ、以下のように記述される。



記述例：リンゴ１.重さ ＝150ｇ　　プロパティの設定
　　print リンゴ１.糖度　　プロパティの表示
　　リンゴ１.焼く(10 分)　　メソッドの実行

【資料 11】

エクセルのマクロは、実行が開始されると、マクロを実行することに**専念**（制御を占有）するため、処理の内容によっては、コンピュータに高い負荷を与え、マウスの操作などが困難になることがある。特に、マクロ実行中は、エクセルと同じ環境である VB エディタの停止ボタンを全く受け付けなくなるため、強制終了するしかない場合があり、マクロの開発は特に注意が必要となる。このようなことに備えて、マクロ開発中（一般的に、プログラムの開発中）はこまめに保存するのがよい。

なお、このような場合、マクロの実行から一時的に Windows に処理を譲渡することによって、VB エディタの停止ボタン等を受け付けることが可能となる。このための命令として、マクロで占有していた制御をオペレーティングシステムに渡す「DoEvents」命令がある。この命令を繰り返しの最後(Loop 命令の前など)に挿入することによって、Windows の処理（処理待ち行列に入っているイベントの処理）を行うことができるようになる。ただし、このような処理が入るため、マクロの実行自体は遅くなる。

【資料 12】

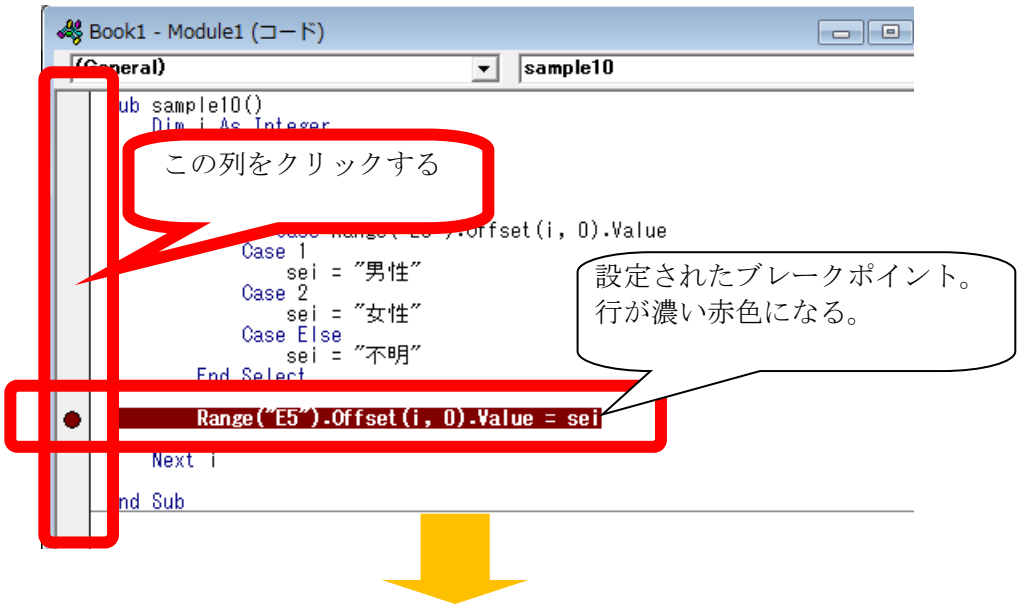
ボタンの種類と動作

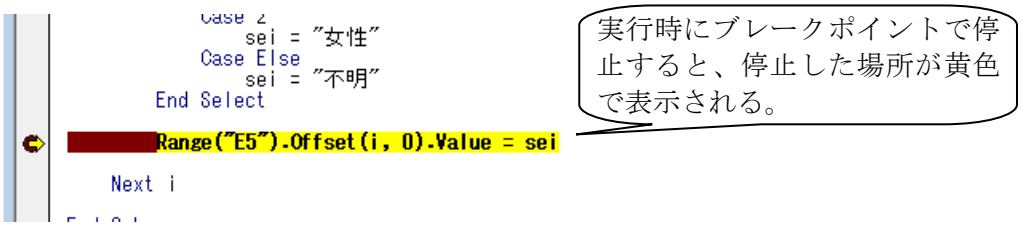
	ボタン （フォームコントロール）	コマンドボタン （ActiveXコントロール）	図形
クリックしたときの動き	あり	あり	なし
標準モジュールの実行	可	不可 ※コードはボタンを置いたシートやフォームに記述される	可
プロパティの設定	不可	可	可
その他	従来の互換性のため	今後標準として使用される	

【資料 13】

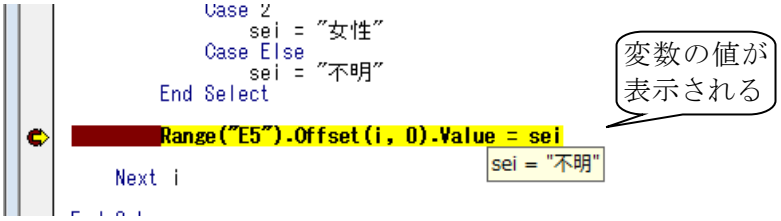
①ブレークポイント

ブレークポイントは、マクロの実行中に、実行を停止したい場所に設定し、停止することができる。ブレークポイントによって停止すると、その時の各変数の値やシートの状態が保たれるため、それぞれの値を確認でき、そこまでの実行に問題がないかどうかを確認することができる。ブレークポイントは、設定したい行の左の場所をクリックするか、設定したい行にカーソルを置き、F9 キーを押すか、デバッグメニューの「ブレークポイントの設定／解除」を選択すれば、設定することができる。





また、実行停止中は、特定の変数にカーソルを合わせるとその値を確認することができる。



②ステップ実行

ステップ実行は、マクロを１行ずつ実行したり、カーソルの前まで実行して停止したりすることができる。この作業によって、各変数がどのように変化していくのかを順に追いかけて（トレースと呼ぶ）確認することができる。

ステップ実行は、デバッグメニュー内にある通り、ステップイン／ステップオーバー／ステップアウト、及びカーソルの前まで実行の 4 種類があり、他プロシージャの呼び出しがあったときに、そのプロシージャの内部までステップ実行するものが「ステップイン」、呼び出されたプロシージャを１つの命令として実行するものが「ステップオーバー」、プロシージャ内の残りの命令をまとめて実行するのが「ステップアウト」となり、状況に応じて使い分けることになる。

それぞれ、キーボードの F8、Shift + F8、Shift + Ctrl + F8、Ctrl + F8 の各キーで１ステップを実行することができる。

③ウォッチ式

変数の値や式の値を確認する場合、上記のようにカーソルを合わせて確認することができるが、あらかじめ確認したい変数を登録しておき、逐一値を見ることもできる。このような変数を登録し、値を確認するためのウィンドウとして、「ウォッチ」ウィンドウがある。ウォッチウィンドウは、「表示」メニューのウォッチウィンドウから表示することができる。また、変数の登録は、登録したい変数上で右クリックし、「ウォッチ式の追加」を選択することで、登録することができる。

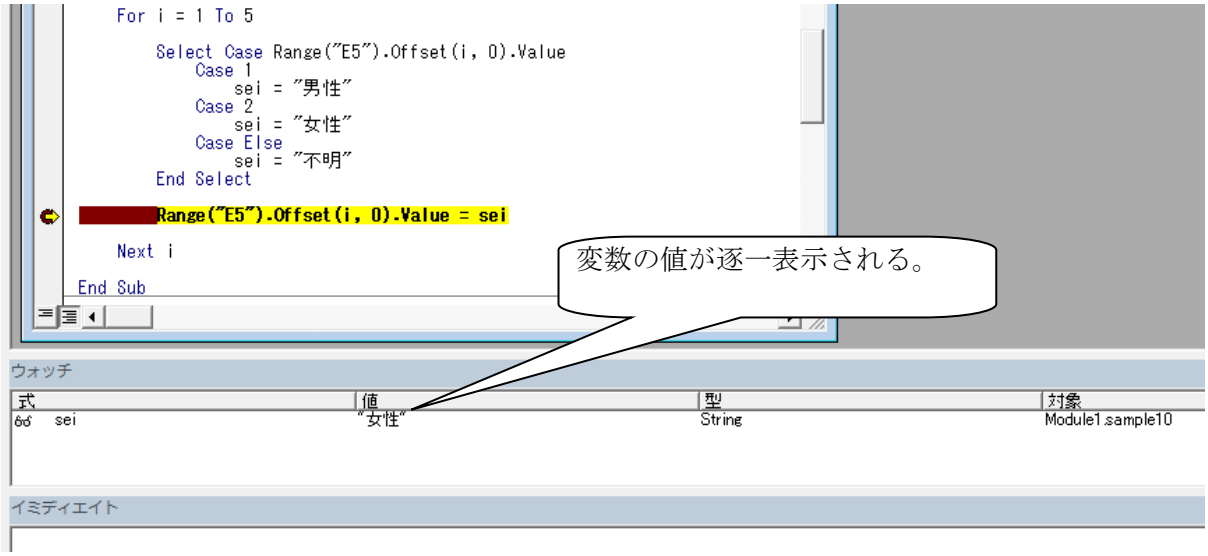


図 16　ウォッチウィンドウ

ウォッチウィンドウではこの他に、式の値が変化した場合に実行を停止するなどの機能もある。

④イミディエイトウィンドウを用いたデバッグ

ウォッチ式で確認できない値の場合は、イミディエイトウィンドウに、「? 」に続いて値を確認したい式を貼り付け、Enter を押すことで確認できることもある。

さらに、変数に特定の値を設定したい場合にも、使用することができる。例えば、変数 sei の値を「不明」としたい場合は、イミディエイトウィンドウで、以下のように入力し、Enter キーを押せばよい。

```
sei = “不明” Enter
```

【資料 14】

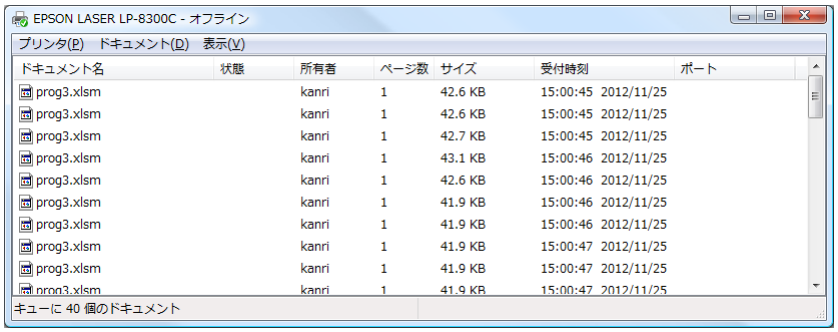
印刷命令の引数

引数の名前	説　明	設定例
From	印刷開始ページ	From := 5
To	印刷終了ページ	To := 10
Preview	印刷プレビュー	Preview :=True
ActivePrinter	アクティブプリンタの指定	ActivePrinter := "EPSON LASER LP-8300C"
PrintToFile	ファイルへ出力	PrintToFile := True
Collate	部単位で印刷	Collate := True
PrToFileName	ファイル名の指定	PrToFileName := "sample1"
IgnorePrintAreas	印刷範囲を無視	IgnorePrintAreas := True

【資料 15】

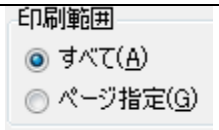
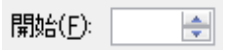
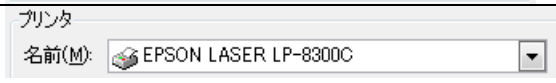
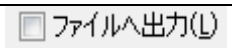
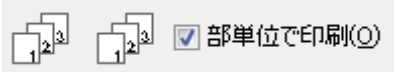
このマクロを利用して 40 名の生徒のデータを印刷した場合、40 枚分の印刷データが次々とプリンタに送信されることになる。このようなデータは、一度パソコンの中のキュー（スプール）に保存され、印刷機の速度に合わせて順次送出されるため、キューがいっぱいとなり、メモリが足りなくなるなどの問題が起きる可能性がある。

また、以前の Windows の仕様では、LAN 経由で一度に送出できる印刷データが 11 件分と制限されていたため、環境によっては 11 件分のデータが送出された後、 5 分程度、送出が停止してしまうこともある。



【資料 16】

印刷ダイアログの引数と役割

引数	役割名	ダイアログ	設定値
arg1	range_num 印刷範囲		1…すべて 2…ページ指定
arg2	from 開始		印刷開始ページ番号
arg3	to 終了		印刷終了ページ番号
arg4	copies 印刷部数		印刷部数
arg5	draft 簡易印刷		True…簡易印刷 False…通常印刷
arg6	preview プレビュー		プレビューの有無
arg7	print_what		
arg8	color カラー印刷		カラー印刷の有無
arg9	feed 紙送り		
arg10	quality 品質		水平方向の解像度
arg11	y_resolution ｙ方向解像度		垂直方向の解像度
arg12	selection 印刷対象		1…選択した部分 2…選択したシート 3…ブック全体 4…テーブル
arg13	printer_text プリンタ名		プリンタ名を""で指定 "EPSON LASER LP-8300C"
arg14	print_to_file ファイルへ出力		0…ファイルへ出力しない 1…ファイルへ出力する
arg15	collate 部単位で印刷		0…ページ単位 1…部単位

【資料 17】

３Ｄ参照

同一形式のシートを１つのファイルに結合した場合、マクロを利用しなくても、複数のシートに渡る処理をエクセル上で実行することが可能である。例えば、アンケート用紙のシートが s1～s40 まである場合、全シートのセル E15 に入力された値の平均を求めたい場合、次のような式で求めることができる。

```
=AVERAGE('s1:s40'!E15)
```

この式では、シート s1 から s40 までの、セル E15 の値の平均を求める。このように、複数のシートの同一範囲を指定し、その値を参照することを「３D 参照」（串刺し集計）という。

式の入力には、「=average(」まで入力し、シート s1 をクリック後、Shift キーを押しながらシート s40 をクリックし、その後セル E15 をクリックすることのよって、入力することもできる。

３D 参照が可能な関数は、以下のとおりである（Rank 関数等含まれないものもある）。

関　数	説　　明
SUM	数値を合計する。
AVERAGE	数値の（数学的な）平均値を計算する。
AVERAGEA	数値、文字列、および論理値を含む値の（数学的な）平均値を返す。
COUNT	数値が含まれるセルの個数を返す。
COUNTA	空白以外のセルの個数を返す。
MAX	値の集合に含まれる最大の値を返す。
MAXA	数値、文字列、および論理値の集合に含まれる最大の値を返す。
MIN	値の集合に含まれる最小の値を返す。
MINA	数値、文字列、および論理値の集合に含まれる最小の値を返す。
PRODUCT	数値の積を計算する。
STDEV	母集団の標本を使って標準偏差を計算する。
STDEVA	数値、文字列、および論理値を含む母集団の標本を使って標準偏差を計算する。
STDEVP	母集団全体の標準偏差を計算する。
STDEVPA	数値、文字列、および論理値の母集団全体の標準偏差を計算する。
VAR	母集団の標本を使って分散を返す。
VARA	数値、文字列、および論理値の母集団の標本を使って分散を返す。
VARP	母集団全体の分散を計算する。
VARPA	数値、文字列、および論理値を含む母集団全体の分散を計算する。

【資料 18】

キーとキーコードの対応

キー	コード	キー	コード	キー	コード	キー	コード	キー	コード
A	65	N	78	0	48(96)	←	37	Esc	27
B	66	O	79	1	49(97)	↑	38	PageUp	33
C	67	P	80	2	50(98)	→	39	PageDown	34
D	68	Q	81	3	51(99)	↓	40	Home	36
E	69	R	82	4	52(100)	スペース	32	Ins	45
F	70	S	83	5	53(101)	BS	8	Del	46
G	71	T	84	6	54(102)	Tab	9	,	188
H	72	U	85	7	55(103)	Enter	13	.	190
I	73	V	86	8	56(104)	Shift	16	/	191
J	74	W	87	9	57(105)	Ctrl	17	¥	220
K	75	X	88	F1	112	Alt	18	-	189
L	76	Y	89	F2	113	:	186	^	222
M	77	Z	90	F12	127	;	187	@	192

索　引

Memo