関西大学『社会学部紀要』第39巻第1号，2007，pp.61-96 ISSN 0287-6817

資料

心理学におけるExcel VBAの利用 その2
―ストループ効果の実験プログラム―

久 本 博 行

Applications of Excel VBA for the Psychology (2)
A Computerized System for the Stroop Effect

Hiroyuki HISAMOTO

Abstract

A computer program for experimentation on the Stroop effect was made.

Key words: Stroop Effect, Excel VBA, Programming

抄 録

Excel VBAを使用し，ストループ効果の実験プログラムを作成する方法が示された。

キーワード：ストループ効果，Excel VBA，プログラミング

１．ストループ効果

ここでは，ストループ効果の実験プログラムを作成します。ストループ効果とは，Stroop, J. R.（1935）によって発見された現象で，例えば「赤」という文字を青色のインクで書いてあるものを提示し，その文字の色（この場合は青）を答えるという課題を行うと，青インクで書かれた四角を提示し，その色（この場合は青）を答える場合に比べ反応速度が遅くなるというものです。これは，「赤」という言葉の情報処理とインクの色の情報処理との間に葛藤が生じ，それによって反応が遅れているのです，このような葛藤を認知的葛藤といいます。ストループ効果の実験方法としては，今挙げた文字の色を答えるものだけでなく，いろいろな課題が考案されています。下にいくつかの例を挙げておきます。

⑴ 数字がいくつか書かれているものを提示し，数字の個数を答えるもので，「７」という数字が５つ書かれていれば５と答えるといったもの。

⑵ 基点に対して文字が上下左右どこに書かれているかを答えるもので，基点の下に「上」と書かれていれば下と答えるといったもの。

⑶ 犬の線画の中に「猫」と書かれていると犬と答えるもの。

⑷ 高い音（175Hz）で「low」あるいは低い音（110Hz）で「high」といい，その音の高さを答えるもの。

⑸ 蛇恐怖症の人とそうでない人に「毒蛇」，「這う」といった言葉を書いたものを示し，その文字の色を答えてもらうと，蛇恐怖症の人はそうでない人より，反応が遅れるといったもの（情動的ストループ効果）。

以上のようにいろいろな課題が考えられており，このストループ効果による研究は，認知，発達，人格，臨床など多くの心理学の分野で行われています。

２．プログラムの概要

ここでは，「あか」，「あお」，「みどり」といった言葉の文字の色を答えるストループ課題を作ってみましょう。刺激語として「あか」，「きいろ」，「みどり」，「あお」の４つを使い，その文字の色が赤ならキーボードの数字の「１」のキーを，黄色なら「２」，緑なら「３」，青なら「４」を押して答えてもらうことにします。実際には，図２.２にあるような刺激を提示し，左上の下線が引かれた言葉から順に答えてもらいます。実験条件としては，統制条件，一致条件，不一致条件の３種類を作ります。図２.２に示すように統制条件では，赤，黄，緑，青の４色の「■」をランダムに提示しています。一致条件では４色の言葉をその

言葉と同じ色でランダムに提示しています。不一致条件では，４色の言葉をその言葉とは異なる色でランダムに提示しています。

プログラムの大まかな流れは，次のようになります。

① タイトルと実験方法の説明

② 被験者番号の入力

③ 刺激提示条件の選択

④ 刺激提示し，実験を開始，実験終了

⑤ 結果をExcelのワークシートへ出力し，①へ戻る。

タイトルと実験方法の説明，刺激提示条件の選択画面は，図２.１のようになります。

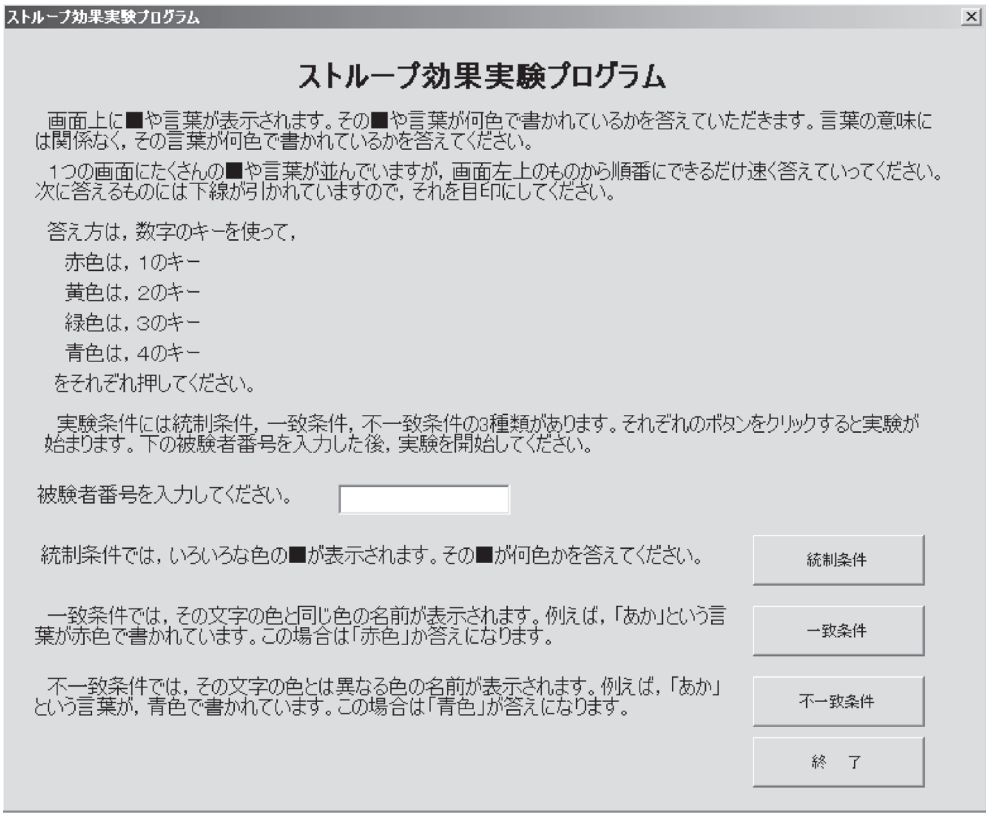
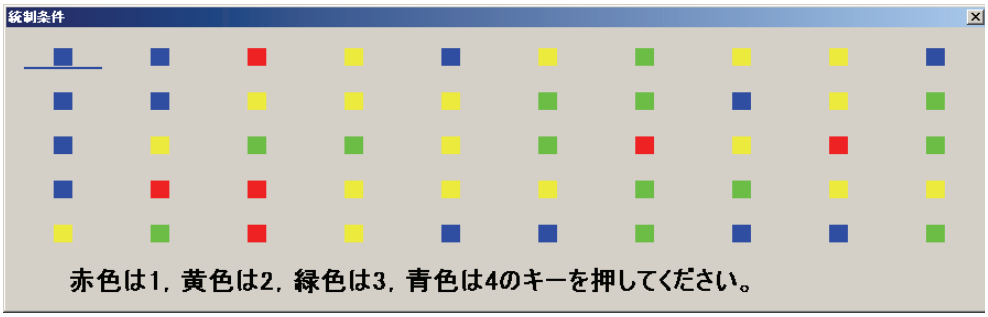
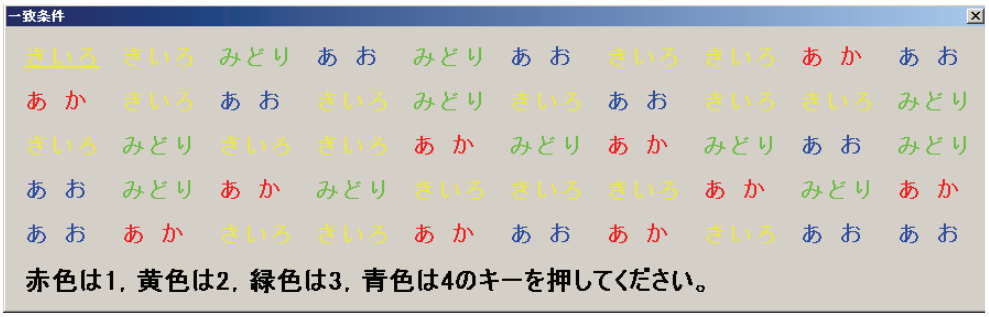


図２.１　タイトルと実験方法の説明，刺激提示条件の選択画面

統制条件の画面



一致条件の画面



不一致条件の画面

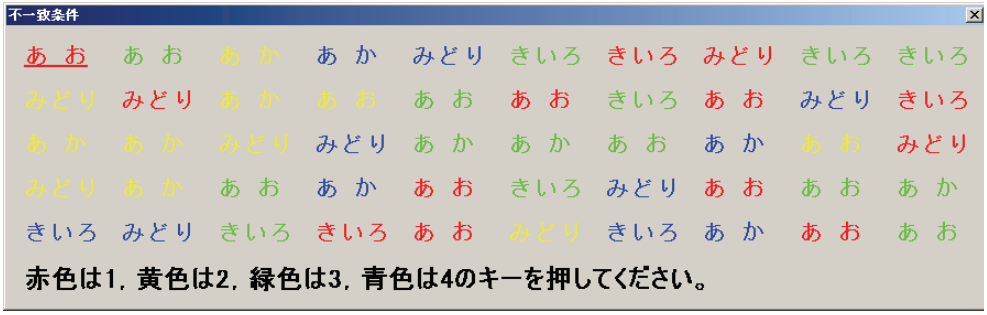
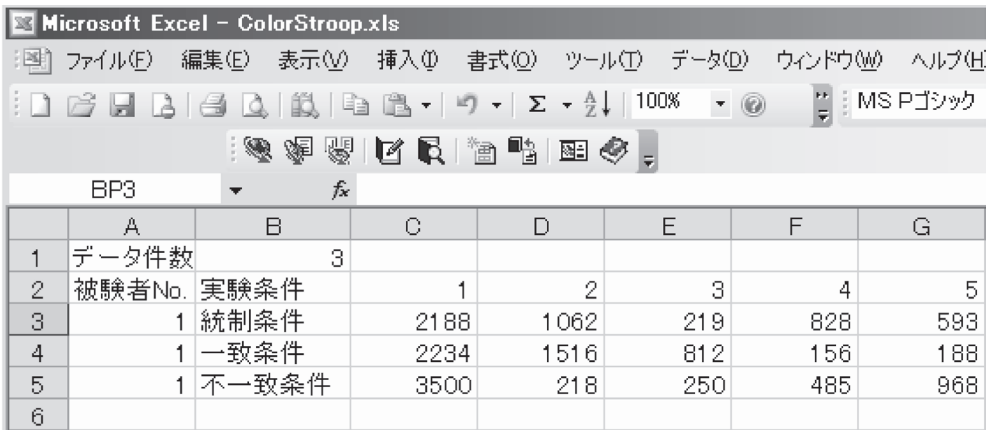


図２.２　刺激提示画面

結果はワークシート上にmSec.単位の反応時間として出力します（図２.３）。出力情報としては，データ件数，被験者No.，実験条件，反応時間の４つです。出力するワークシートは，「StroopData」という名前のワークシートに固定しておきます。データの出力のタイミングとしては，１つの実験条件が終了した後に出力します。

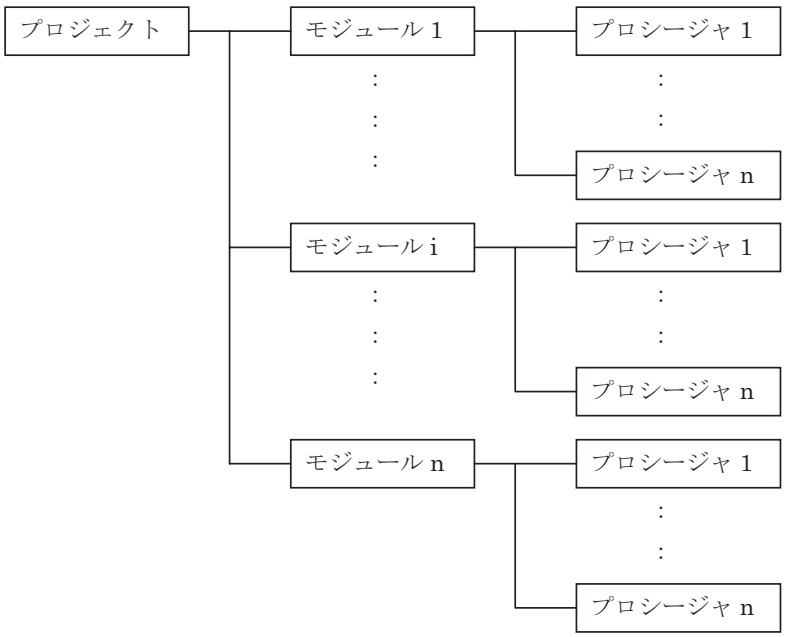


	A	B	C	D	E	F	G
1	データ件数	3					
2	被験者No.	実験条件	1	2	3	4	5
3	1	統制条件	2188	1062	219	828	593
4	1	一致条件	2234	1516	812	156	188
5	1	不一致条件	3500	218	250	485	968
6							

図2.3　結果の出力

プロジェクト，モジュール，プロシージャについて

VBAのプログラムで，最も上位の単位をプロジェクトといいます。その下にモジュールがあり，モジュールの下にサブプロシージャやファンクションプロシージャといったプロシージャがあります。



それでは，ここでモジュールごとに処理内容をまとめると下図のようになります。

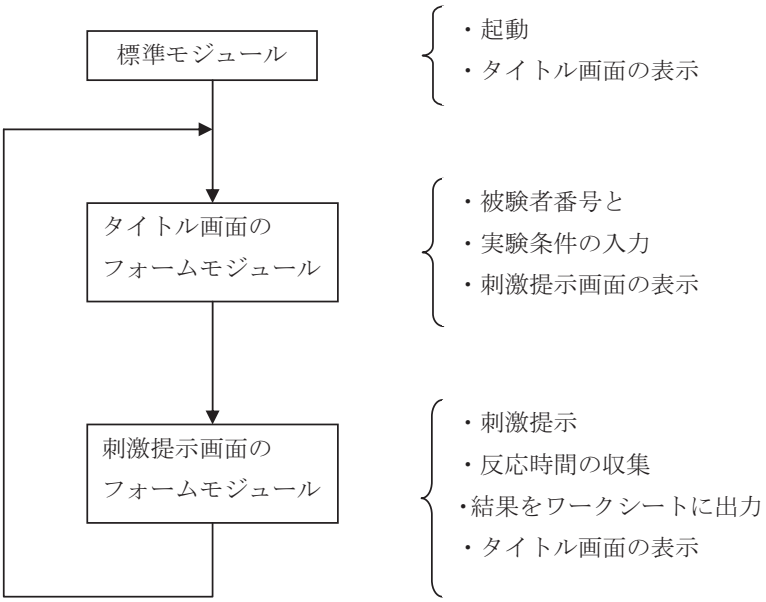


図２.４　処理の流れと各モジュールの処理

３．プログラムの作成

３.１ タイトル画面の作成

［ツール］メニュー →［マクロ］→［Visual Basic Editor］の順にクリックし，Visual Basic Editorを起動します。［ユーザーフォームの挿入］ボタンをクリックし，ユーザーフォームを作ります（図３.１）。そのままでは小さいのでハンドルをドラッグし，図２.１のような実験の説明が入る大きさに拡大します。

次にオブジェクト名とCaptionを設定しておきましょう。オブジェクト名はfrmMainManuとします。オブジェクトの名前は，自分で分かりやすいものであればなんでも構いません。デフォルトではUserForm1となっているので，このままでも問題はありません。しかし，たくさんのオブジェクトを利用する場合，このままでは分かりにくくなることがあります。ですから，自分なりにオブジェクトの名前をつけるルールを作って，それに従って名前をつけておくのがいいでしょう。

ここでは，オブジェクト名の先頭３文字は小文字でオブジェクトの種類を表し，４文字

目からは大文字で始めて，その機能を表す言葉をつけ名前としています。オブジェクトの種類を表す3文字は，以下のようなものです。

ユーザーフォーム	frm
ラベル	lbl
テキストボックス	txt
コマンドボタン	cmd
コンボボックス	cbb
リストボックス	lst
チェックボックス	chk
オプションボタン	opt

ウィンドウ左下にあるプロパティの中の（オブジェクト名）のUserForm1をfrmMainMenuとし，CaptionのUserForm1を「ストループ効果実験プログラム」とします。Captionを変えるとユーザーフォームのウィンドウのタイトルバーを変えることになり，ユーザーフォームのタイトルが「ストループ効果実験プログラム」となったのが分かると思います（図3.2）。

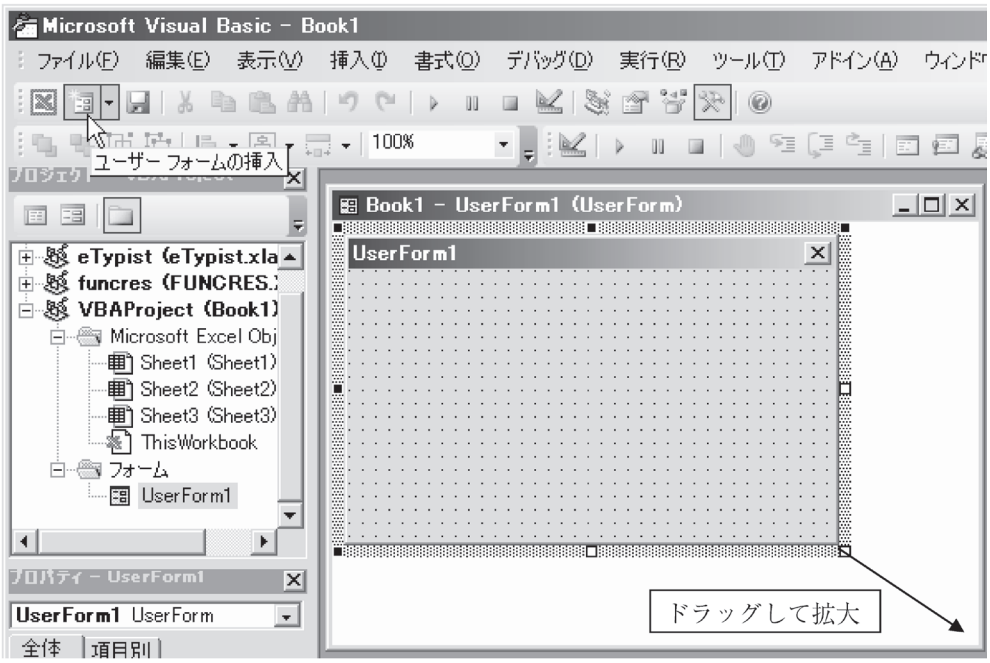


図3.1　ユーザーフォームの挿入と拡大

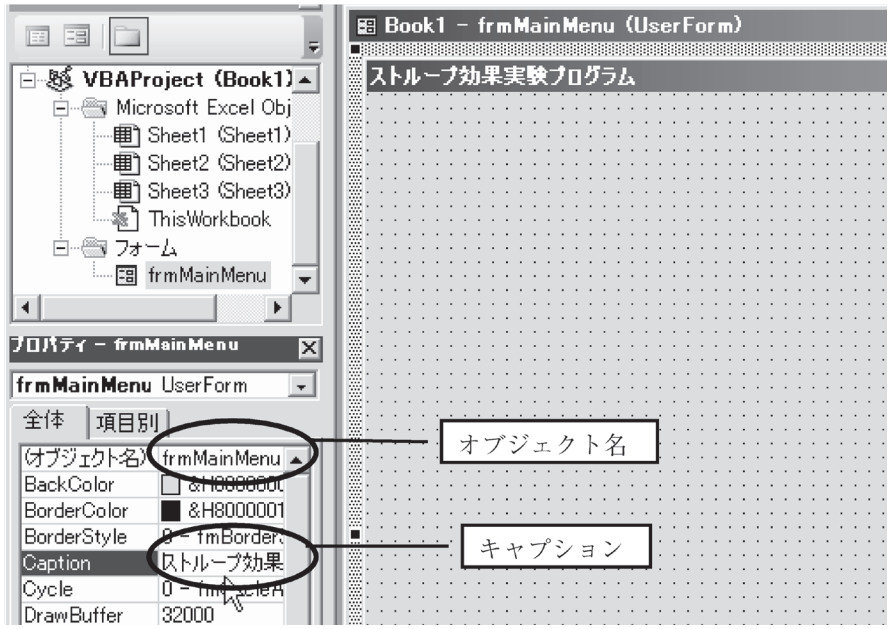


図３.２　オブジェクト名とキャプションの入力

次に図２.１にあるようなタイトルや実験の説明文などを書くのですが，フォーム上に直接文字を書くことはできません。ですから，ラベルをフォーム上に貼り付け，タイトルを入力します。ラベルの貼り方は，テキストボックスやコマンドボタンと同じ要領で，ツールボックスのラベル（図３.３）をクリックし，フォーム上で適当な大きさにドラッグします。ラベルの中に図２.１にあるようなタイトルや実験の説明文を入力します。ラベルの中で改行はできませんので，改行したい場合は別のラベルを新たに貼り付け，そこに文を入力します。

図３.３ ラベル

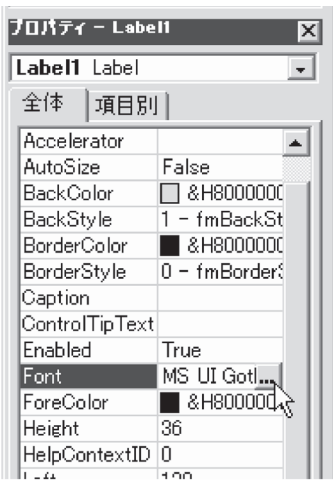


図３.４　フォント

文字の大きさや字体を変えたい場合は，プロパティのFontのところをクリックし，反転表示にするとボタンが出てきます。そのボタンをクリ

ックするとフォント・ダイアログボックスが出てくるので，これを使って文字の大きさや字体を変えます。ラベルの中では，すべての文字は同じ設定になります。ですから，一つのラベルの中で特定の文字だけ大きくしたりすることはできません。文字の色はプロパティのForeColorで設定します。これもラベルの中の特定の文字だけ色を変えるということはできません。もし，そういうことをしたい場合は，その部分だけ別のラベルにするという方法があります。

ラベルは，プログラムの中で制御することはありませんので，ラベルについてはオブジェクト名を設定せずそのままにしておきます。

図3.5　フォント・ダイアログボックス

次に被験者番号を入力するために，テキストボックスを一つ貼り付けます。テキストボックスのオブジェクト名は，txtSsNoとしておきます。このテキストボックスの次にコマンドボタンを4つ図2.1のように貼り付けます。コマンドボタンのキャプションも図のように「統制条件」，「一致条件」，「不一致条件」，「終了」に変えます。コマンドボタンのオブジェクト名は，「統制条件」はcmdControl，「一致条件」はcmdMatch，「不一致条件」はcmdMismatch，「終了」はcmdEndにしておきましょう。

3.2 刺激提示画面作成

次に，刺激提示画面を作ります。刺激提示画面は，図2.2にあるように3種類のものがありますが，実際に作成するフォームは1つだけで，プログラムの中で実験条件によって表示するものを変えていきます。

提示する刺激の個数によってフォームのサイズを調整するので，今は適当なサイズで作ります。フォームのオブジェクト名はfrmColorに，CaptionはColor Naming Stroopとします。そして，図3.6にあるようにラベルを1つ貼ります。そのラベルのCaptionには「赤色は1，黄色は2，緑色は3，青色は4のキーを押してください。」と入力しておきます。フォントサイズは18ぐらいで太字にします。位置は適当であとで変更します。このラベルは，

プログラムの中で位置を設定するので，オブジェクト名をlblInstructionとしておきましょう。

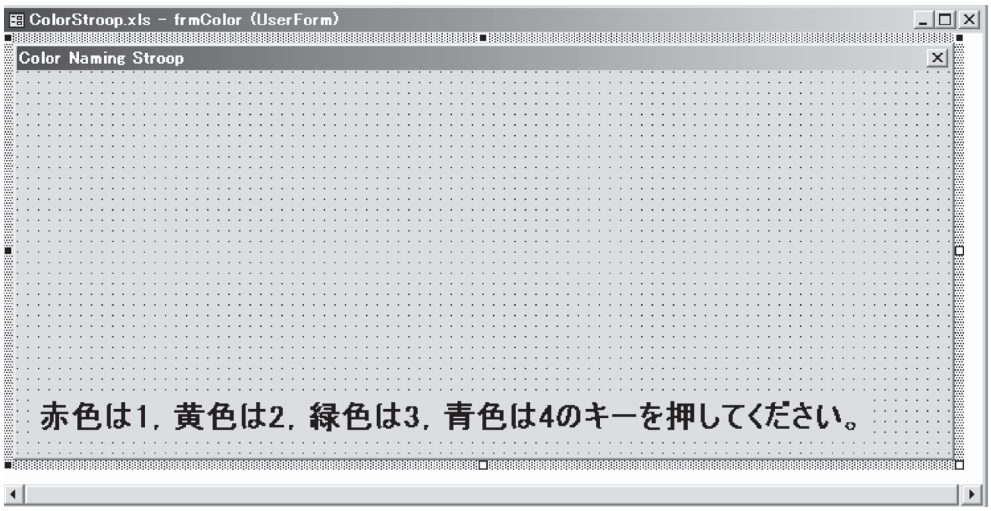


図３.６　刺激提示画面

３.３ 標準モジュールの作成

ユーザーがVBAで作成されたプログラムを起動する場合，Excelの［ツール］メニューから［マクロ］さらに［マクロ］を選びクリックすると，マクロのダイアログボックスが出て，その中から必要なマクロを選び，［実行］ボタンを押すとそのマクロが実行されることになります（図３.７）。その最初に実行されるプログラムは，標準モジュールとして作成します。

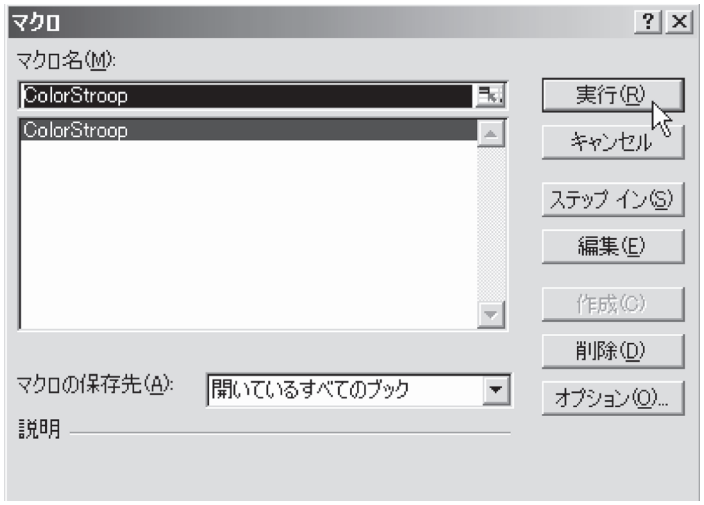


図３.７　マクロ・ダイアログボックス

では，最初に起動されるマクロとなる標準モジュールを作りましょう。Visual Basic Editorの［ツール］メニューから［マクロ］をクリックするとマクロ・ダイアログボックスが出てきます。まだマクロを作っていないので，ダイアログボックスには何も表示されていないはずです。マクロ名の欄に「ColorStroop」と入力してください。これがこのマクロの名前になります。入力すると［作成］ボタンが押せる状態になるので，［作成］ボタンを押します。するとプロジェクトエクスプローラの中に標準モジュールのフォルダーができ，Module 1 がその中に作成され，Module 1 のコードウィンドウが開きます。

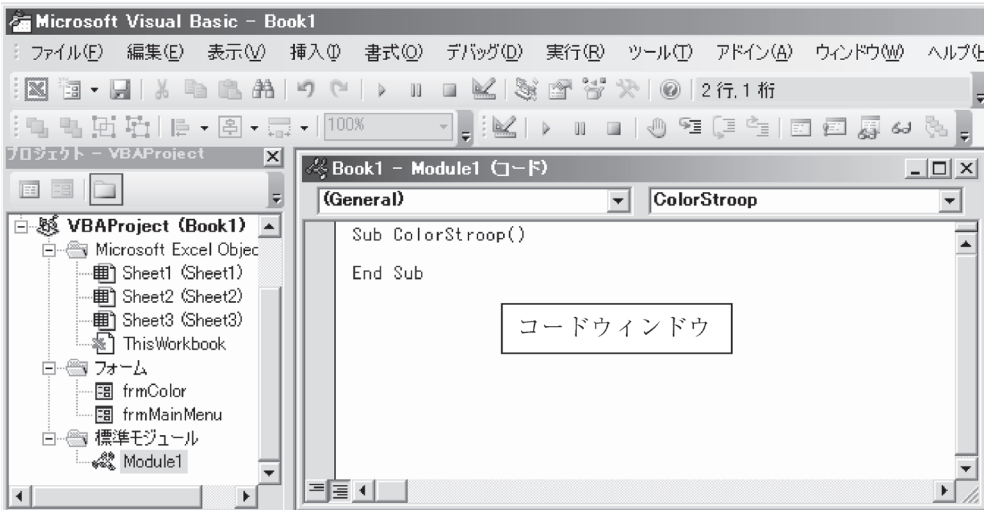


図3.8　標準モジュールのコードウィンドウ

マクロが起動されると，メインメニューのウィンドウを表示するので，そのようにプログラムします。

Sub ColorStroop（）

と

End Sub

の間に

frmMainMenu. Show

と入力します。これは，frmMainMenuというオブジェクトのShowというメソッドを動かしなさいという意味です。Showメソッドは，ユーザーフォームを表示するメソッドで，これを使ってfrmMainMenuを表示させます。

入力の仕方はまず入力したい位置をクリックし，そこにカーソル移動させます。そして，Ctrlキーを押しながら空白（Space）キーを押すとこのプロジェクトのオブジェクトとそのメンバー一覧が出てきますので，そこでfrmmぐらいまで入力すると，frmMainMenuが出てきて反転表示されます。その状態で，Tabキーを押すとfrmMainMenuが入力できます（図３.９）。次に，ピリオドを入力すると今度はfrmMainMenuのメンバー一覧が表示されるので，shぐらいまで入力するとshowが反転表示されますので，Tabキーを押すとshowが入力されます。このようにオブジェクトのメンバー一覧をうまく使うと入力ミスを格段に減らすことができます。なお，frmMainMenuは大文字と小文字を区別せずに入力しても，同一プロジェクト内に同じ名前のオブジェクトや変数などがあれば，自動的に大文字と小文字を書き分けてくれます。

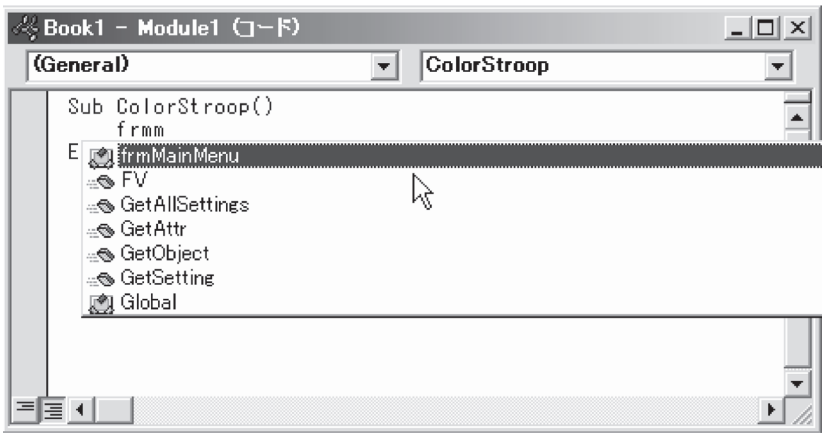


図３.９　オブジェクトの一覧から目的のオブジェクトを選ぶ

さて，下記のようなプログラムができたところで，実際にこれを動かしてみましょう。

```
Sub ColorStroop（）
    frmMainMenu．Show
End Sub
```

［ツール］メニューから［マクロ］を選ぶと図３.７のようなマクロ・ダイアログボックスが表示されますので，［実行］ボタンを押してみてください。そうすると，メインメニューのウィンドウが表示されます。

オブジェクトについて

プログラミングにおいてオブジェクトとは，処理とデータが一体化したものといえるでしょう。これまで，プロパティの値をいろいろと設定してきましたが，このプロパティはオブジェクトのデータにあたるものなのです。そして，オブジェクトの行う処理や動作をメソッドといいます。どのようなプロパティやメソッドがあるかということは,オブジェクトによって異なります。あるオブジェクトのプロパティやメソッドを確認するには，オブジェクト・ブラウザやヘルプを使います。オブジェクト・ブラウザは[表示]メニューから，オブジェクト・ブラウザを選ぶと見ることができます

3．4 タイトル画面のフォームモジュール作成

次にタイトル画面のフォームモジュールを作成します。プロジェクトエクスプローラのfrmMainMenuオブジェクトをクリックし，さらにオブジェクトの表示ボタンをクリックするとfrmMainMenuのオブジェクトが表示されます（図3.10）。

フォーム上に配置されているボタンが，クリックされたときの動作をプログラミングしましょう。

図3.10 オブジェクトの表示

イベントプロシージャについて

ボタンがクリックされるといったイベントに応じて，動作するプロシージャをイベントプロシージャと呼びます。イベントプロシージャは，そのオブジェクトのモジュールに作られます。つまり，ユーザーフォームに貼り付けられたボタンをクリックするイベントプロシージャは，そのフォームモジュールの中に作られることになります。イベントプロシージャの名前は，自由につけることができません。イベントプロシージャの名前は，オブジェクト名＋__（アンダースコア）＋イベント名という形式になっています。

frmMainMenuには４つのボタンがあります。終了ボタンがクリックされた場合は，プログラムを終了させます。他の３つのボタンの処理は，ほとんど同じもので以下のような処理になります。

① 実験条件を変数に記憶する（統制条件：１，一致条件：２，不一致条件：３）
② 入力された被験者番号を変数に記憶する
③ タイトル画面（frmMainMenu）を消す
④ 実験画面を表示する
⑤ 実験が終了した時にタイトル画面を表示する

このうち③～⑤の処理はフォームの制御に関する処理なので，この部分は一つのサブプロシージャを作り，それを３つのボタンのイベントプロシージャで呼び出すことにしましょう。

それでは，終了ボタンの処理からはじめます。終了ボタンをダブルクリックしてください。そうすると，コードウィンドウが開かれ，その中に次のようなプログラムの一部が書かれています。これは，cmdEndというオブジェクトをクリックしたときのイベントプロシージャであることを示しています。

```
Private Sub cmdEnd_Click（）

End Sub
```

終了ボタンをクリックされた場合は，プログラムを終了させるだけでよいので，上の２行の間にEndと入れます。これで終了ボタンのイベントプロシージャは出来上がりです。［Sub/ユーザーフォームの実行］ボタンをクリックすると（図３.11），frmMainMenuが起動し終了ボタンを押すとプログラムを終了することが確認できます。

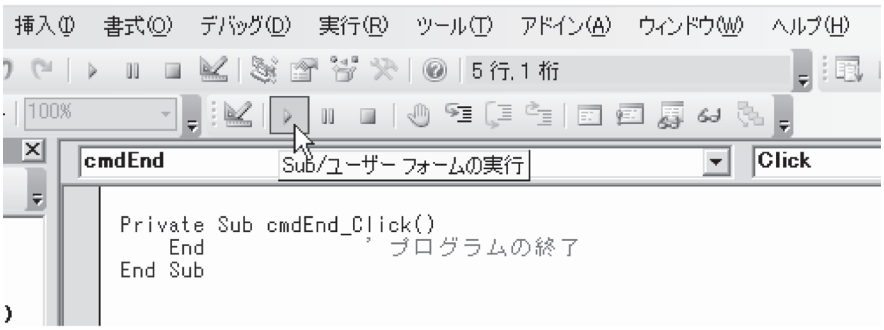


図３.11　Sub/ユーザーフォームの実行ボタン

次に，frmMainMenuのオブジェクトを表示させ，先ほどと同じように統制条件ボタンをダブルクリックします。するとコードウィンドウが開かれ，これも先ほどと同様に下記の２行のプログラムが書かれています。

```
Private Sub cmdControl_Click（）

End Sub
```

このプロシージャでは，前に述べたように実験条件と被験者番号を変数に記憶することと，フォームを制御することを行います。実験条件については，統制条件は１，一致条件は２，不一致条件は３というように変数に記憶し，被験者番号についてはそのまま文字列型変数に記憶しましょう。変数名はそれぞれConditionとSsNoにします。この２つの変数は，もう一つのフォームモジュールfrmColorからもアクセスできる必要があります。frmColorでは，実験条件によって表示する刺激が変わりますし，最後に結果をワークシートに出力する時に被験者番号が必要になるからです。そのため，この２つの変数は標準モジュールでPublilc変数として宣言します。Public変数の詳しい説明については，変数のスコープについての部分を参照してください。

図３.12のようにプロジェクトエクスプローラでModule１をクリックし，さらにコードの表示ボタンをクリックすると，標準モジュールのコードウィンドウが開かれますので，そのサブプロシージャの外側で下記の２行の宣言をしてください。

```
Public Condition As Integer
Public SsNo As String
```

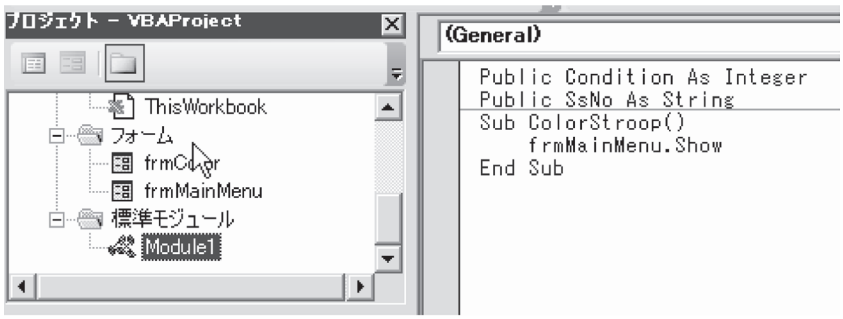


図３.12　Public変数の宣言

変数のスコープについて

変数のスコープとは，変数の値を読み出したり，書き換えたりすることができるプログラムの範囲のことです。今ユーザーフォームXとY，それに標準モジュールZがあるとします。下の表でSub～End Subの間で宣言されている変数は，その宣言されたサブプロシージャの中だけが有効範囲なのです。そのため,サブプロシージャabcで宣言された変数aとdefで宣言されたa，モジュールZのサブプロシージャmnoで宣言されたaはそれぞれ全く別の変数となります。このような変数をプロシージャレベル変数と呼びます。では，フォームXの中のサブプロシージャabcとdefで共通して使いたい変数は，どのように宣言すればよいのでしょうか。その場合は，変数cのようにSub～End Subの外側で宣言します。ですから，変数cはサブプロシージャabcとdefのどちらからもアクセスできる変数となっています。フォームYのモジュールでもSub～End Subの外側で変数cが宣言されています。この変数cもサブプロシージャghiとサブプロシージャjklで共通に使える変数です。このような変数をモジュールレベル変数といい，宣言されたモジュール内で有効です。したがって，フォームXのモジュールのcとフォームYのモジュールのcは全く別の変数ということになります。

フォームXのモジュール	フォームYのモジュール	標準モジュールZ
Dim c As Integer	Dim c As Long	Public r As Integer
Sub abc()	Sub ghi()	Sub mno()
Dim a As Integer	Dim e As Integer	Dim a As Integer
Dim b As String	Dim f As String	Dim i As Long
End Sub	End Sub	End Sub
Sub def()	Sub jkl()	Sub pqr()
Dim a As Long	Dim g As Boolean	Dim j As Single
Dim d As Single	Dim h As Double	Dim k As String
End Sub	End Sub	End Sub

では，異なるモジュール間でデータを共有したい場合は，どのようにすればよいのでしょうか。その場合は，標準モジュールZのところで宣言されているrのように

Publicステートメントをつけてサブプロシージャの外側で宣言します。このrは，このプロジェクトの全てのプロシージャからアクセスすることができます。このような変数をパブリック変数と呼びます。パブリック変数は，必ず標準モジュールで宣言する必要があります。フォームモジュールでパブリック変数を宣言しても，異なるフォームモジュールからはアクセスすることはできません。

変数の寿命について

プロシージャレベル変数は，そのプロシージャが終了する毎に変数の値が初期化されます。例えば，特定のプロシージャが何回起動されたか，そのプロシージャ内で知る必要がある場合，プロシージャレベル変数をそのまま使うと，呼び出されるたびに初期化されますので回数を数えることができません。その場合は，つぎのようにStaticステートメントを使って宣言します。

```
Static Kaisu As Integer
```

モジュールレベル変数やパブリック変数も，プログラムの実行が終了するまでその値が保持されます。

プロジェクトエクスプローラでfrmMainMenuをクリックし，コードの表示ボタンをクリックしてfrmMainMenuのところに戻りましょう。cmdControl_Clickプロシージャで実験条件と被験者番号を変数に記録し，フォームの表示，非表示を制御するサブプロシージャを呼び出します。呼び出すサブプロシージャの名前は，FormControlとします。プログラムは下記のようになります。

```
1 :  Private Sub cmdControl_Click（）
2 :  Condition ＝ 1                ’統制条件
3 :  SsNo ＝ txtSsNo.Text      ’被験者番号を保存
4 :  FormControl
5 :  End Sub
```

プログラムの２行目は変数Conditionに統制条件を示す１を代入しています。２行目はtxtSsNoオブジェクトのTextプロパティの値を変数SsNoに代入し，入力された被験者番号を保存しています。もし，被験者番号を必須にしたい場合は，被験者番号が入力されているかどうか，チェックする必要があります。その場合は，３，４行目を以下のように修正すると被験者番号の未入力を防ぐことができます。

```
3 :  If txtSsNo. Text ＝ "" Then
4 :      MsgBox（"被験者番号を入力してください"）
5 :  Else
6 :      SsNo ＝ txtSsNo. Text ' 被験者番号を保存
7 :      FormControl
8 :  End If
```

３行目の「""」は何もない（Null）を示す表記です。MsgBoxは図３.13のダイアログボックスを表示する関数です。MsgBox関数の構文は以下とおりです。

MsgBox(prompt[, buttons] [, title] [, helpfile, context])

図３.13　メッセージボックス

prompt：表示する文字列で必ず指定します。
buttons：表示するボタンの種類や個数などを指定します。
title：タイトルバーに表示する文字列です。
helpfile：ヘルプを設定するために，使用するヘルプ ファイルの名前を指定します。
context：ヘルプ トピックに指定したコンテキスト番号を表す数式を指定します。

一致条件ボタンと不一致条件ボタンについても，統制条件ボタンと同じようにプログラムします。異なるところは，統制条件では変数Conditionに１を代入しましたが，一致条件では２を不一致条件では３を代入するようにしてください。それ以外は，変わりません。

次に，FormControlサブプロシージャを作ります。処理の概要は以下のとおりです。

① タイトル画面を隠す
　これは，刺激提示画面の邪魔にならないように自分自身を隠します。
② ３秒間の待ち時間を入れる
　いきなり刺激提示画面が出ると被験者が戸惑ったり，最初の反応時間だけが極端に長くなるのを防ぐためです。
③ 刺激提示画面を表示する
④ タイトル画面を表示する

刺激提示画面での実験が終了すると，刺激提示画面を表示した行の次に制御が戻ってくるので，ここでタイトル画面（自分自身）をもう一度表示してやります。
以上をプログラムすると次のようになります。

```
１：  Private Sub FormControl（）
２：      Me. Hide              ’タイトル画面を隠す
３：      Application. Wait（Now + TimeValue（"00:00:3"））’刺激提示画面を表示
          する前に3秒間待つ
４：      frmColor. Show        ’刺激提示画面を表示
５：      Me. Show              ’タイトル画面を表示
６：  End Sub
```

プログラムの２行目のMeは自分自身を示しています。ここでは，frmMainMenuオブジェクトのことで，Hideはユーザーフォームを非表示にするメソッドです。

３行目のWaitメソッドは指定の時刻まで，マクロを停止するメソッドです。Nowは現在時刻を内部表現の値（シリアル値）で返してくれる関数で，TimeValueは引数の文字列を内部表現の値に変換してくれる関数です。ですから，３行目の式では現在時刻から３秒間待つように指定していることになります。

４行目は刺激提示画面を表示しています。刺激提示画面が表示され実験が始まります。そして，実験が終了しfrmColorのモジュールの実行が終ると，frmColorを表示した次の行に制御が戻ってきます。

５行目はもう一度タイトル画面を表示しています。

全体としてfrmMainMenuのモジュールは，図３.14のようになります。

では，ここまでのプログラムの動作を確認してみましょう。［ツール］メニューから［マクロ］を選び，マクロ・ダイアログボックスが出てきたら［実行］ボタンを押してマクロを実行します。被験者番号を入力し，実験条件のいずれかをクリックします。少し間をおいて刺激提示画面が表示されます。刺激提示画面の右上隅のクローズボタンをクリックすると刺激提示画面が消え，タイトル画面がまた出てきます。そして，終了ボタンを押すとマクロが終了します。

```
Private Sub cmdControl_Click()
    Condition = 1    ' 統制条件
    SsNo = txtSsNo.Text '  被験者番号を保存
    FormControl
End Sub
Private Sub cmdEnd_Click()
    End             ' プログラムの終了
End Sub
Private Sub cmdMatch_Click()
    Condition = 2    ' 一致条件
    SsNo = txtSsNo.Text '  被験者番号を保存
    FormControl
End Sub
Private Sub cmdMismatch_Click()
    Condition = 3    ' 不一致条件
    SsNo = txtSsNo.Text '  被験者番号を保存
    FormControl
End Sub
Private Sub FormControl()
    Me.Hide             '  タイトル画面を隠す
    Application.Wait (Now + TimeValue("00:00:3"))   ' 3秒間待つ
    frmColor.Show       '  刺激提示画面を表示
    Me.Show             '  タイトル画面を表示
End Sub
```

図 3.14　frmMainMenuのモジュール

3.5 刺激提示画面のフォームモジュール作成

次に，刺激提示画面のフォームモジュールを作っていきます。

プログラムの流れは以下のようになります。

① 刺激提示画面の作成

② キーボードからの入力反応を待つ

③ 入力反応を受け，反応時間を測定，全ての刺激に対する反応が終了すれば④へ，そうでなければ②へ戻る

④ 結果をワークシートに出力し，frmMainMenuに戻る。

①の処理は，UserForm_Initializeというイベントプロシージャで行います。このイベントプロシージャは，ユーザーフォームを初期化するプロシージャで，Initializeイベントの発生で起動されます。Initializeイベントは，ユーザーフォームが表示される前に発生します。今回の場合ですとfrmMainMenuでfrmColor. Showを実行した段階でこのイベントが発生し，UserForm_Initializeイベントプロシージャが起動されることになります。

②，③，④の処理は，UserForm_KeyPressというイベントプロシージャで行います。このイベントプロシージャは，文字キーを押すことで発生するイベントを処理します。

Keypressイベントについて

Keypressイベントは，印刷可能文字やアルファベットとCtrlキーの組み合わせ，特殊文字とCtrlキーの組み合わせのキーを押したときに発生します。ファンクションキーや矢印キーなどではイベントが発生しません。そうしたキーのイベントを処理したい場合は，KeydownかKeyupのイベントプロシージャを使います。

3.5.1 初期化プロシージャ（UserForm_Initialize）の作成

プロジェクトエクスプローラでfrmColorをクリックし，コードの表示ボタンを押します（図3.15)。コード表示ウィンドウ上部のオブジェクトボックス（オブジェクトを選択するコンボボックス）でUserFormを選び，プロシージャボックス（プロシージャを選択するコンボボックス）でInitializeを選ぶと図3.16のようにUserForm_Initializeイベントプロシージャが作られます。

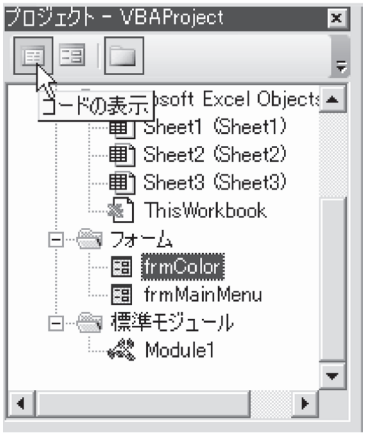


図3.15　frmColorのコードの表示

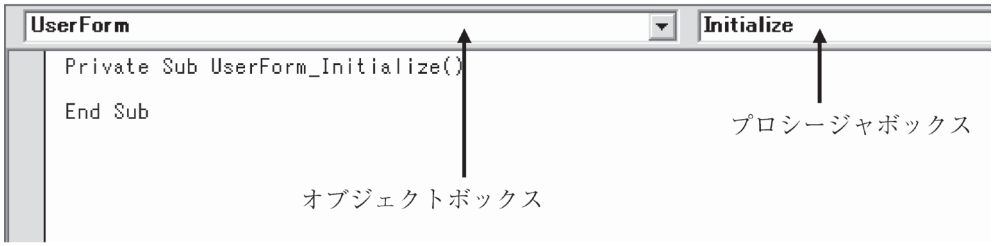


図3.16　UserForm_Initializeプロシージャの作成

まずPrivate Sub UserForm_Initialize（）文の上に図3．17の宣言文を入力します。モジュールの先頭から最初のSub文までの間を宣言セクションといいますが，そこに入力します。

```
 1:Option Base 1
 2:Option Explicit
 3:Const nRow As Integer = 8        ’1画面に表示するラベルの行数
 4:Const nClm As Integer = 8        ’1画面に表示するラベルの列数
 5:Const lblWidth As Integer = 54   ’ラベルの幅
 6:Const lblHeight As Integer = 18  ’ラベルの高さ
 7:Const rowSpace As Integer = 12   ’ラベルとラベルの行の間隔
 8:Const clmSpace As Integer = 12   ’ラベルとラベルの列の間隔
 9:Const FontSize As Integer = 16   ’ラベルに書く文字の大きさ
10:Dim lblItem(nRow, nClm) As MSForms.Label  ’ラベルのオブジェクト配列
11:Dim stmStrings(nRow, nClm, 2) As Integer  ’刺激文字列
12:Dim rowPosition As Integer            ’現在の入力位置の行番号
13:Dim clmPosition As Integer            ’現在の入力位置の列番号
14:Dim startTime As Currency             ’測定開始時刻
15:Dim endTime As Currency               ’測定終了時刻
16:Dim resColor(nRow * nClm) As Integer  ’被験者の答え
17:Dim resTime(nRow * nClm) As Currency      ’被験者の応答時間
18:Dim title As Variant        ’実験条件名の配列
19:Private Declare Function QueryPerformanceCounter Lib "Kernel32" (ByRef X As Currency) As
Boolean
20:Private Declare Function QueryPerformanceFrequency Lib "Kernel32" (ByRef X As Currency)
As Boolean
```

図３.17　frmColorモジュールの宣言セクション

３行目から９行目までの定数は，図３.18に示すようにフォームを構成する上で必要な値を定義したものです。フォーム上に直接文字を書くことはできないので，刺激となる言葉や文字はラベルにしてフォーム上に貼ります。nRowとnClmはラベルを縦横にいくつ貼るかという行数と列数を定義した数値です。この数値を変えることによって，刺激の数をいくつにするか決めることができます。lblWidthとlblHeightは，１枚のラベルの幅と高さを定義したものです。これは，FontSizeで定義している16ポイントの全角文字を３文字書くことができる大きさです。

刺激の数が多いと１つ１つラベルを貼っていき，そのラベルに刺激語を入力するのはひじょうに大変です。そこで，ラベルを配列変数として取り扱うことにより，処理を簡単に済ませます。ラベルを配列変数として宣言しているのが，10行目の文です。これで，nRow × nClm個（ここでは，８×８＝64）のラベルが出来上がったことになります。実際には，このラベルは宣言されただけで，実体はありません。実体を作成し，フォーム上に貼り付けるのはプロシージャの中で行います。

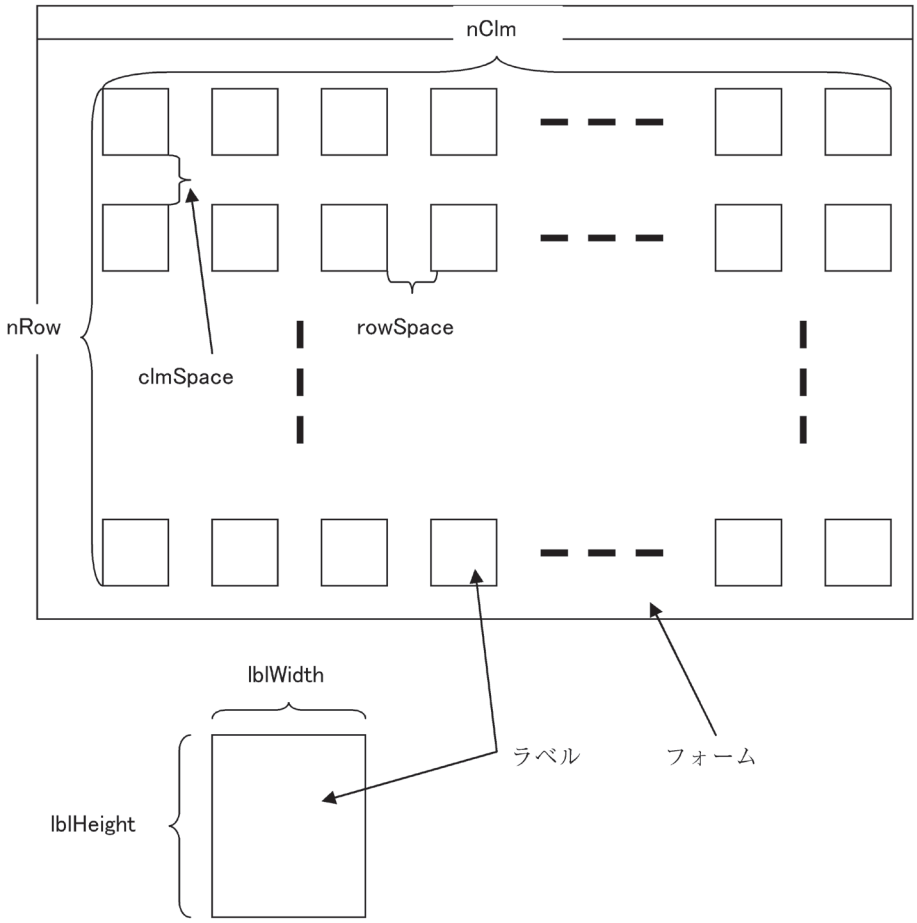


図３.18　宣言された定数の図解

11行目は，刺激となる文字列とその色を記録しておく変数です。stmStrings（i, j, 1）はi，j番目のラベルに表示される文字列，stmStrings（i, j, 2）はi，j番目の文字列の色を記録しています。

12，13行目は現在の反応対象となっているラベルの位置を示す変数です。

14，15行目は反応時間を求めるための刺激提示を開始した時刻と被験者が反応した時刻を記録する変数で，endTime－startTimeが被験者の反応時間となります。この2つの変数がCurrency型であるのは，時間を測定する関数QueryPerformanceCounterの引数が，64ビットの変数であるためです。VBAのLong型は32ビットの長さなのでLong型変数には入りきらないのです。

16，17行目は被験者の反応と反応時間を記録する変数です。被験者の反応はワークシートに出力しませんが，必要な場合はこの変数の内容をワークシートに出力することでどのような誤りが多いかといったことを見ることができます。

18行目は，実験条件名が入る配列です。配列宣言されていませんが，プロシージャの中でArray関数を使って初期化されますので，配列宣言する必要はありません。Array関数については，プロシージャの中で出てきたときに説明いたします。

19, 20行目は時刻を測定する関数QueryPerformanceCounterとQueryPerformanceFrequencyを宣言しています。プロジェクト内にない関数を使用する場合には，関数も変数と同様に宣言する必要があります。これは，Kernel32というダイナミックリンクライブラリに含まれている関数で，引数はCurrency型の変数が一つ，関数の型はBoolean型であることを示しています。Privateとなっていますが，標準モジュールで宣言する場合はPrivateでなくても構いません。フォームモジュールで宣言する場合は，Privateが必要です。

時間の測定について

Windowsでは時間を測定する関数が，Now，Time，Timer，GetTickCount，TimeGetTimeなど各種用意されており，それらのタイマーは，それぞれ精度が異なります。Now，Time，Timerの分解能は秒単位で，GetTickCount，TimeGetTimeは約10msです。ここでは，最も分解能の高いQueryPerformanceCounterを使っています。このタイマーの分解能はシステムによって異なり，1秒あたりの周波数をQueryPerformanceFrequencyで取得することができるのでその逆数が分解能となります。QueryPerformanceCounterの使い方は，下記のマイクロソフトの文書を参考にしてください。

http://support.microsoft.com/kb/172338/ja

なお，WindowsはマルチプログラミングのOSで，いくつものプログラムが平行して動作しているので，その処理時間のため，ひじょうに高い精度で反応時間を測定することは困難です。もし，もっと精度よく測定したい場合は，リアルタイムOSを使うか，あるいはSetPriorityClass関数で実験プログラムの優先順位をあげるといった方法をとる必要があります。

次にUserForm_Initializeのコードを見ていきましょう（図3.19）。

```
1:Private Sub UserForm_Initialize()
2:
3:   Dim i As Integer
4:   Dim j As Integer
5:    Dim lblTop As Integer
6:    Dim lblLeft As Integer
7:
8:   ’刺激語をセット
9:   Dim stmCharacters As Variant        ’刺激文字
10:   If Condition = 1 Then
11:       ’統制条件の場合■をセット
12:       stmCharacters = Array(" ■ ")
13:   Else
14:       ’一致条件，不一致条件では色名をセット
15:       stmCharacters = Array("あ　か", "きいろ", "みどり", "あ　お")
16:   End If
17:
18:   ’刺激に用いる4色を配列stmColorにセット
19:   Dim stmColor As Variant              ’刺激色
20:   ’                赤色     黄色     緑色     青色
21:   stmColor = Array(vbRed, vbYellow, vbGreen, vbBlue)
22:   title = Array("統制条件", "一致条件", "不一致条件")
23:   ’フォームのタイトルバーに実験条件を表示
24:   frmColor.Caption = title(Condition)
25:   ’フォームの幅と高さを計算し，フォームの大きさを決める
26:   frmColor.Width = clmSpace + lblWidth * nClm + clmSpace * nClm
27:   frmColor.Height = rowSpace + lblHeight * nRow + rowSpace * nRow _
28:                     + rowSpace + lblInstruction.Height + rowSpace
29:
30:   Randomize           ’乱数用の新しい初期値を作成
31:
32:   lblTop = rowSpace
33:   lblLeft = clmSpace
34:   For j = 1 To nClm
35:       For i = 1 To nRow
36:           If Condition = 1 Then
37:               ’統制条件では，刺激は1種類 色は4色をランダムに設定
38:               stmStrings(i, j, 1) = 1
39:               stmStrings(i, j, 2) = Int((4 * Rnd()) + 1)
40:           Else
41:               ’一致条件と不一致条件では，刺激をランダムに設定
42:               stmStrings(i, j, 1) = Int((4 * Rnd()) + 1)
43:               If Condition = 2 Then
44:                   ’一致条件では刺激と同じ色を設定
45:                   stmStrings(i, j, 2) = stmStrings(i, j, 1)
```

図3.19　UserForm_Initializeのコード

```
46:             Else
47:                 ’不一致条件の場合文字色を不一致なものにランダムに変更する
48:                 Do
49:                     stmStrings(i, j, 2) = Int((4 * Rnd()) + 1)
50:                 Loop Until stmStrings(i, j, 1) <> stmStrings(i, j, 2)
51:             End If
52:         End If
53:
54:         ’フォーム上にラベルを貼り，ラベルに刺激語を設定する
55:         Set lblItem(i, j) = frmColor.Controls.Add("Forms.Label.1")
56:         lblItem(i, j).Left = lblLeft                ’ラベルのフォーム上での左からの位置
57:         lblItem(i, j).Top = lblTop                  ’ラベルのフォーム上での上からの位置
58:         lblItem(i, j).Height = lblHeight                ’ラベルの縦の長さ
59:         lblItem(i, j).Width = lblWidth                  ’ラベルの横幅
60:         lblItem(i, j).Font.Name = "ＭＳ ゴシック"           ’刺激語をMSゴシックにセット
61:         lblItem(i, j).Font.Bold = True                  ’刺激語を太字にセット
62:         lblItem(i, j).Font.Size = FontSize              ’刺激語のフォントサイズをセット
63:         lblItem(i, j).Caption = stmCharacters(stmStrings(i, j, 1))   ’刺激語をセット
64:         lblItem(i, j).ForeColor = stmColor(stmStrings(i, j, 2))’刺激語の色をセット
65:         lblItem(i, j).FontUnderline = False             ’下線を引かない
66:
67:         lblTop = lblTop + lblHeight + rowSpace
68:
69:     Next i
70:
71:     lblTop = rowSpace
72:     lblLeft = lblLeft + lblWidth + clmSpace
73: Next j
74: lblInstruction.Top = lblTop + (lblHeight + rowSpace) * nRow
75: lblItem(1, 1).FontUnderline = True
76: rowPosition = 1
77: clmPosition = 1
78: ’反応時間の測定開始
79: QueryPerformanceCounter startTime
80:End Sub
```

図3.19　UserForm_Initializeのコード（つづき）

8～22行目は，配列に初期値を代入しています。ここで使っているArray関数は配列に値を代入する関数です。Array関数の構文は以下のとおりです。

変数 = Array(引数)

変数はVariant型で宣言しておく必要があります。

例えば，15行目は下記の４行のプログラムと同じ働きをします。

stmCharacters（1）＝"あ　か"
stmCharacters（2）＝"きいろ"
stmCharacters（3）＝"みどり"
stmCharacters（4）＝"あ　お"

その結果，下図のようにstmCharactersに文字列が代入されることになります。

	(1)	(2)	(3)	(4)
stmCharacters	あか	きいろ	みどり	あお

図３.20　Array関数によって配列へ代入されたデータ

30行目のRandomize文は，乱数を生成するRnd関数に新しい初期値を与える数値演算ステートメントです。

```
Randomize ［引数］
　引数には，任意の数値または数式を指定しますが，省略可能です。
```

34行目から73行目が主要な処理で，刺激となるいろいろな色の言葉が入ったラベルを作り出しているところです。その中で36～52行目が刺激となる言葉やその文字色を乱数で作り出しているところです。Rndが乱数を発生させる関数です。Rnd関数の構文は以下のとおりです。

```
Rnd[(引数)]
　RndはSingle型の0以上1未満の乱数を返す関数です。
　引数は，乱数の初期値ですが省略可能です。
　引数　>0　乱数列の次の乱数を返します。
　　　<0　引数によって決まる同じ値を返します。
　　　=0　直前に生成した乱数を返します。
　　省略時 乱数列の次の乱数を返します。
```

55行目はAddメソッドでラベルの実体を作成し，配列変数に代入しています。図３．17の宣言セクション10行目でラベルオブジェクトの配列を宣言していますが，これは変数の宣言をしただけで，実体としてラベルオブジェクトができあがったわけではありません。この55行目のSetステートメントでラベルの実体を代入しているのです。

56～65行目は，できあがったラベルの大きさや位置，表示する文字列やその色といった

フォームにコントロールを追加

Set オブジェクト型変数= オブジェクト.Add(プログラムID [, 名前 [, 表示]])

オブジェクト型変数：追加するコントロールオブジェクト型変数

Labelオブジェクトを追加する場合は，Labelオブジェクト型変数

オブジェクト：コントロールを追加する対象となるオブジェクト

UserForm1に追加したい場合は，

UserForm1.Controlsとなります。

プログラムID：追加するコントロールオブジェクトのクラスを表すテキスト文字列で，これは下記のように決まっています。

名前：追加するコントロールオブジェクトの名前で省略可能です。

表示：追加するコントロールオブジェクトを表示するか否かを指定するBoolean型の値または式です。省略可能で省略するとTrue（表示）となります。

各種コントロールのプログラムIDは以下のとおりです。

CheckBox	Forms.CheckBox.1
ComboBox	Forms.ComboBox.1
CommandButton	Forms.CommandButton.1
Frame	Forms.Frame.1
Image	Forms.Image.1
Label	Forms.Label.1
ListBox	Forms.ListBox.1
MultiPage	Forms.MultiPage.1
OptionButton	Forms.OptionButton.1
ScrollBar	Forms.ScrollBar.1
SpinButton	Forms.SpinButton.1
TabStrip	Forms.TabStrip.1
TextBox	Forms.TextBox.1
ToggleButton	Forms.ToggleButton.1

各種のプロパティを設定しているところです。

74行目は，「赤色は1，黄色は2，緑色は3，青色は4のキーを押してください。」という操作説明のラベルの位置を設定しているところです。

75行目は，最初の刺激語に下線を引いています。

79行目は，反応時間の測定を開始した時刻を取得しているところです。

これで測定が開始され，次に被験者が反応して何かキーを押すと，KeyPressイベントプロシージャが起動されます。では，KeyPressイベントプロシージャを見ていきます（図3.21）。

```
01:Private Sub UserForm_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
02:    Dim inputChar As String
03:
04:    ’反応時間の測定終了
05:    QueryPerformanceCounter endTime
06:    resTime(clmPosition + (rowPosition - 1) * nClm) = endTime - startTime ’反応時間の記録
07:
08:    ’どのキーを押したかを記録
09:    resColor(clmPosition + (rowPosition - 1) * nClm) = KeyAscii - 48
10:    ’誤反応には，ビープ音を鳴らす
11:    If KeyAscii - 48 <> stmStrings(rowPosition, clmPosition, 2) Then
12:        Beep
13:    End If
14:
15:    If rowPosition >= nRow And clmPosition >= nClm Then ’実験の終了チェック
16:        OutputData      ’データをシートに出力
17:        Unload Me       ’実験用フォームを消去し，もとの画面に戻る
18:        Exit Sub
19:    End If
20:
21:    ’済んだ項目の下線を消去
22:    lblItem(rowPosition, clmPosition).FontUnderline = False
23:
24:    ’次の項目に下線を引く
25:    clmPosition = clmPosition + 1
26:    If clmPosition > nClm Then
27:        clmPosition = 1
28:        rowPosition = rowPosition + 1
29:    End If
30:    lblItem(rowPosition, clmPosition).FontUnderline = True
31:
32:    ’反応時間の測定開始
33:    QueryPerformanceCounter startTime
34:
35:End Sub
```

図3.21　UserForm_KeyPressのコード

４～６行目は，反応の時刻を測定し，反応の時刻から反応測定の開始時刻を引き反応時間をもとめ，それを変数resTimeに保存しています。

９行目は，被験者の反応を記録しています。数字を入力すると仮定して，数字の文字コードから48を引くと，その数値に変換できるのでその性質を利用して押されたキーを文字ではなく数値として保存しています。今回のプログラムでは，この情報を出力していないので，このステップはなくてもよいのですが，誤答の回数やパターンを分析したい場合は，このデータを出力できるように記録しています。

11～13行目は，誤反応に対しビープ音を鳴らしています。Beepステートメントはコンピュータのスピーカを鳴らすもので，鳴る時間や音はハードウェアやシステムによって異なります。

15～19行目は，全試行が終了したかどうかチェックし，終了していれば実験データをワークシートに出力するサブプロシージャOutputDataを呼び出して，実験データを出力し，自分自身であるフォームfrmColorを消去して，このサブプロシージャを終了しています。17行目のUnloadステートメントはオブジェクトをメモリから削除するもので，オブジェクトに関するすべてのメモリが解放されます。Hideメソッドとの違いは，Hideメソッドではオブジェクトが非表示になり，ユーザーがそのアプリケーションとやり取りできなくなりますが，別のプロセスとやり取りしたりTimerイベントを使うことはできます。Unloadでは非表示になるだけではなく，プログラムそのものがメモリから削除され消えてしまいます。Unloadの構文は以下のとおりです。

```
Unload オブジェクト
```

21～30行目は，たくさんの刺激が並んでいるので現在どの刺激に対して反応すべきかを下線を引いて示していますが，その下線を引く位置を変えています。

33行目は，新たな刺激に対する反応時間の測定を開始しています。

次に収集したデータをワークシートに出力するサブプロシージャOutputDataについて見ていきましょう（図３.22）。

12行目はサブプロシージャCheckWorksheetを呼び出し，データを出力するためのワークシートStroopDataの有無をチェックしています。もし，StroopDataというワークシートがなければ，現在のブックに追加し，ある場合はなにもしません。

```
01Private Sub OutputData()
02'  測定データをワークシートに出力するプログラム
03
04   Dim NumberOfData As Integer ' データ件数
05   Dim j As Integer, i As Integer, k As Integer
06   Dim l As Integer, m As Integer
07   Dim NumberOfError As Integer
08   Dim Frequency As Currency
09   Dim sum As Double
10
11   ' StroopData Worksheetの有無をチェック
12   CheckWorksheet
13
14   ' データを出力
15   With ActiveSheet
16      .Cells(1, 1).Value = "データ件数"          ' セルA1に項目名をセット
17      NumberOfData = .Cells(1, 2).Value          ' 既存のデータ数を得る
18      .Cells(1, 2).Value = NumberOfData + 1      ' セルB1のデータ件数を1増やす
19      .Cells(2, 1).Value = "被験者No."           ' セルA2に項目名をセット
20      .Cells(2, 2).Value = "実験条件"            ' セルB2に項目名をセット
21
22      ' 試行回数を出力
23      For i = 1 To nRow * nClm
24         .Cells(2, i + 2).Value = i
25      Next i
26      ' 平均 誤答数
27      .Cells(2, i + 2).Value = "平均"
28      .Cells(2, i + 3).Value = "誤答数"
29
30      j = NumberOfData + 3
31
32      .Cells(j, 1).Value = SsNo                  ' 被験者番号を出力
33      .Cells(j, 2).Value = title(Condition)      ' 実験条件を出力
34
35      ' 反応時間の出力
36      QueryPerformanceFrequency Frequency        '
37      For i = 1 To nRow * nClm
38         .Cells(j, i + 2).Value = CDbl(resTime(i) / Frequency * 1000)
39      Next i
40
41      ' 平均値の出力
42      sum = 0
43      For i = 1 To nRow * nClm
44         sum = sum + .Cells(j, i + 2).Value
45      Next i
46      .Cells(j, i + 2).Value = sum / (nRow * nClm)
47
```

図3.22　OutputDataのコード

```
48        ' 誤答数の出力
49        NumberOfError = 0
50        For i = 1 To nClm * nRow
51            l = i Mod nClm
52            k = (i \ nClm) + 1
53            If l = 0 Then
54                l = nClm: k = i \ nClm
55            End If
56            If resColor(i) <> stmStrings(k, l, 2) Then _
57                    NumberOfError = NumberOfError + 1
58        Next i
59        .Cells(j, nRow * nClm + 4).Value = NumberOfError
60
61    End With
62
63End Sub
```

図 3 .22　OutputDataのコード（つづき）

15〜61行目が実際にワークシートにデータを出力している部分です。15行目のWithステートメントは，ここからEnd Withステートメントまでのオブジェクトの記述を簡略化するためのものです。例えば，16行目の「. Cells(1, 1). Value」は正確には「ActiveSheet. Cells(1, 1). Value」と書かなければいけませんが，With 〜 End Withの間ではWithステートメントで指定されたオブジェクトは省略できるのです。WithとEnd Withの構文は次のとおりです。WithとEnd Withの制御構造の中に，さらにWithとEnd Withの制御構造をつくり，入れ子構造にすることもできます。

```
With  オブジェクト
        :
        :
End With
```

36行目は，高分解能カウンターの周波数を取得しています。QueryPerformanceFrequencyは1秒間の周波数ですので，QueryPerformanceCounterで得られたカウンターの値をこの周波数で割ることで，時間を求めることができます。

38行目では，QueryPerformanceCounterで得られた反応時間を周波数で割り，秒単位の時間になおし，それに1000をかけることでミリ秒の単位にしています。CDblは引数で与えられた数値をDouble型に変換する関数で，反応時間をDouble型に変換しています。

反応時間を格納する変数resTimeはCurrency型ですが，Currency型の値をワークシートに出力する場合，書式を設定しないとうまく表示されません。したがって，Currency型で出力するよりもDouble型で出力したほうが取り扱いやすいので変換しています。CDbl関数の構文は下記のとおりです。

CDbl(式)

　式は，任意の定数，変数を含む数式です。

Double型以外の変換関数としては，下記のものがあります。

関数名	戻り値のデータ型
CBool(式)	Bool型
CByte(式)	Byte型
CCur(式)	Currency型
CDate(式	Date型
CDec(式)	Decimal型
CInt(式)	Integer型
CLng(式)	Long型
CSng(式)	Single型
CVar(式)	Variant型
CStr(式)	String型

41～46行目は，反応時間の平均値を求めそれを出力しています。

48～59行目は，誤答数を求めそれを出力しています。変数resColorには被験者の反応が１～４の値で記録されており，変数stmStringsには提示された刺激の文字と色が同じく１～４の値で記録されているので，２つの変数の値を照合することによって正答か誤答かが判断できます。より詳しく誤答を分析したい場合は，この２つの変数を出力してみてください。

次にサブプロシージャCheckWorksheetについて見てみましょう（図3.23）。

```
01:Private Sub CheckWorksheet()
02:    Dim dataWorksheet As Worksheet
03:    Dim worksheetName As String
04:    worksheetName = "StroopData"    ’データを保存するワークシート名
05:    Dim dataSheet As Boolean    ’True：dataSheet有，False:dataSheet無
06:    dataSheet = False
07:
08:    ’StroopData Worksheetの有無をチェック
09:    For Each dataWorksheet In Worksheets
10:        If dataWorksheet.Name = worksheetName Then
11:            dataSheet = True
12:            Exit For
13:        End If
14:    Next
15:
16:    ’StroopData ワークシートがなければ新たに作る
17:    If dataSheet = False Then
18:        Worksheets.Add                    ’ワークシートの追加
19:        ActiveSheet.Name = worksheetName        ’ワークシートに名前をつける
20:    End If
21:
22:    ’StroopData ワークシートをアクティブにする
23:    Sheets(worksheetName).Activate
24:
25:End Sub
```

図３.23　CheckWorksheetのコード

このサブプロシージャは，「StroopData」という名前のワークシートがあるかどうかをチェックし，なければ新たに作るということをします。

８～14行目の処理が「StroopData」という名前のワークシートを探しているところです。９行目のFor Each … Nextステートメントは，配列やコレクションの各要素に対して一連の繰り返し処理を行うものです。コレクションとは，オブジェクトの集まりのことで例えば，Worksheetsコレクションというのはそのブックにある全てのワークシートのことです。For Each … Next ステートメントの構文は，次のとおりです。

```
For Each 要素 In グループ
        :
        :
Next
```

グループは，配列やコレクションで，要素は配列の変数やコレクションのオブジェクト変数です。

16〜20行目はワークシートを新しく作っているところです。18行目は，Addメソッドを使ってワークシートを挿入しています。ワークシートを挿入する際のAddメソッドの構文は下記のとおりです。

```
ワークシートの挿入
  オブジェクト.Add  [Before/After]  [Count]
    Before/After  Beforeは指定したワークシートの前（左）に挿入します。
                  Afterは指定したワークシートの後（右）に挿入します。
    Count         挿入するワークシートの枚数を指定します。
例
  Worksheets.Add  Before:=Worksheets(“Sheet1”), Count:=5
  Sheet1というワークシートの前に5枚のワークシートを挿入しています。
```

19行目は，挿入したワークシートに名前をつけています。挿入したワークシートにはデフォルトの名前がついており，任意の名前をつけるにはアクティブなワークシートのNameプロパティに名前の文字列を代入します。

23行目は，「StroopData」という名前のワークシートをアクティブにしています。ワークシートを新しく作った場合には，そのワークシートがアクティブになっていますが，既存の場合には必ずしもアクティブになっていない場合があるので念のためここでアクティブにしています。

以上でストループ効果の実験プログラムは，完成です。

このプログラムを使って実験する場合は，Excelのメニューを［ツール］→［マクロ］→［マクロ］の順でクリックし，マクロのダイアログボックスでColorStroopを選び，［実行］ボタンを押すと実験が開始されます。実験を終えると，StroopDataというワークシートにデータが出力されているのでそのデータを分析してください。

参考文献

久本博行（2006）心理学におけるExcel VBAの利用その1―VBAの基本文法―　関西大学社会学部紀要　38（1）　191-221

市川伸一，矢部富美枝（1985）*パーソナルコンピュータによる心理学実験入門*　150-158　東京　培風館

MacLeod, C. M.（1991）Half a Century of Research on the Stroop Effect: An Integrative Review. *Psychological Bulletin* 109 2 163-203

水野りか（2004）*Webを介してできる基礎・認知心理学実験演習*　39-42　京都　ナカニシヤ出版
嶋田博行（1994）　*ストループ効果*　東京　培風館
心理学実験指導研究会（1985）　*実験とテスト＝心理学の基礎　実習編* 58-62　東京　培風館
Stroop, J. R.（1935）Studies of Interference in Serial Verbal Reactions. *Journal of Experimental Psychology* 18 643-662
Williams, J. G., Mathews,A., MacLeod, C.（1996）The Emotional Stroop Task and Psychopathology. *Psychological Bulletin* 120 1 3-24

－2007. 4 . 4 受稿－