flag 変数によるプログラム制御
Program Control by Flag Variable

上　山　義　尚
Yoshitaka Ueyama

目　次

1．flag 変数 (flag variable)とは

フラグ(flag)とは,「記号，標識として用い，通常，棒などに取り付けられている，いろいろな大きさ，色形のきれ」すなわち「旗」である。

変数(variable)とは，もともと「変わりやすいもの」「変わる」「可変の」という形容詞であるが「変わるもの」「変えられるもの」という名詞に転じ，数学の分野では「ある与えられた情報・データ集合の中のいずれかを代表的に示す記号」という意味に使われている。いわゆる数学でいう「変数」である。

コンピュータ(computer)の分野では数学上の意味を拡張して variable すなわち変数という言葉が使われている。しかし，コンピュータで用いる変数は主記憶装置のメモリ番地に対応するものであり，プログラムの実行中に変化する値を格納する入れ物として用い

られる。(注：1)

コンピュータにおけるフラグは，広義には情報の処理や解読で使用するある種のマーカー(marker)，あるいは特定の条件の存在や状況を示す信号をいう。フラグは通信，プログラミング(programming)，情報処理などの分野で用いられる。

フラグはデータ(data)がどのような状態になっているのかを示すためにデータに付属している標識(data)であったり，値や状態の比較結果によりイベント種類を選択するためにハードウエア(hardware)やソフトウエア(software)によってセットされるビット標識(bit data)であったりする。

プログラミングで用いるフラグは，フラグ専用の変数にプログラム(program)またはデータの状態を識別表示する標識をセットして用いる。

このように特殊なフラグ専用の変数を flag 変数(flag variable)という，または，標識として0か1の数値を代入して用いるケースが多いので bit 変数(bit variable)，あるいは，データ，ソフト，ハード等の状態を記憶•記録することから，フラグレジスタ(flag register)と呼ばれることもある。(注：2)

flag 変数に標識となるデータを付属させることを，通常，「フラグを立てる」と表現する。プログラミングにおける flag 変数は，プログラムの分岐時点においてプログラムの流れをどちらの処理に誘導するか決定あるいは設定するために用いられる。

flag 変数に標識としてある値を代入し，その値を参照することにより「フラグが立っている」か「フラグが立っていないか」を知ることでプログラムの流れを制御する。

flag 変数を上手に活用することによりプログラム自体に判断能力を与えることが可能となる。

自己判断能力を備えたユーザー・フレンドリー(user friendly)なシステムを構築するには，flag 変数を応用したプログラミング手法が不可欠である。

しかしながら，フラグに関する詳しい資料や解説書または論文等は無く，flag 変数はベテラン•プログラマー(veteran programmer)によって経験則的に用いられ，その活用法はプログラマーからプログラマーへと伝承されているにすぎない。

従って，プログラマーやシステム•エンジニア(system engineer)を目指す経験の浅い若者にとっては，flag 変数に関する専門知識やその用い方を修得することは大変困難なことと言える。

本論においては，プログラム構築に関して用いられるフラグ変数の型および宣言方法等の標準的設定を例示し，その活用法・応用法については具体的な事例を挙げながらフラグ変数によるプログラム制御手法のノウハウを論述する。これらによりフラグ変数によるプログラム制御の定型化と標準化を図るものである。

2．フラグ変数のデータ型(data type)

変数の宣言をする際には，その変数が扱うデータは数値であるか文字であるかなどデータの種類や属性，すなわちデータ型を指定する。データ型には

整数型(short integer)	2バイト
長整数型(long integer)	4バイト
単精度浮動小数点数型(single)	4バイト
倍精度浮動小数点数型(double)	8バイト
通貨型(currency)	8バイト
日付型(date)	8バイト
文字型(string)	可変長または固定長
バリアント型(variant)	2バイト＋文字列の長さ

などいろいろな型があるが，通常，フラグ変数に代入する標識は 0 か 1 の数値であるため，フラグ変数の型は2バイトの整数型(short integer)を指定する。Long 型，single 型，string 型，variant 型などを使用することも出来るが，メモリ消費が少なく処理速度の速い integer 型(C 言語では int 型)とするのがよい。

誤ったデータ代入の予防，記憶容量サイズの適正化，処理速度の向上などのため変数のデータ型は明示的に指定すべきである。

3．名前の指定と拡張子(name and extension)

プログラムの構成は大変複雑なものが多く，時間を置くとプログラム作成者自身がそのリストを見てもよく解らないことがある。まして，他人の作成したプログラムは解読不能に近いことが多い。従って，プログラムの全機能を階層的部分機能に分解して構築する構造化プログラミング(structured programming)が理解しやすく，数多く用いられる変数名も判別し易い名前を付けることが望ましい。

フラグ変数はプログラムのいろいろな場面で利用されるため，複数のフラグ変数が必要である。従って，そのフラグ変数が何の識別のために用いられるフラグ変数なのか，容易に判断出来るような名前を付けるべきである。また，フラグ変数が通常の変数と異なることが一目瞭然に識別出来るよう，フラグ変数には専用の拡張子を付属させることを薦めたい。拡張子には，アンダーラインと小文字の flag(_flag)を付けるよう統一するとよい。

例えば，データを入力したかどうかを識別するフラグ変数は

DATAINPUT_flag

のように命名する。同様にして，データを読み込んだかどうかを識別するフラグ変数，データを保存したかどうかを識別するフラグ変数は

DATALOAD_flag
DATASAVE_flag

のようにするとよい。拡張子が付いているのでフラグ変数であることが一目でわかり，その名前から何の為のフラグ変数であるかを知ることができる。

4．フラグ変数の適用範囲と宣言(type statement)

フラグ変数はあらゆるプロシージャ(procedure)，モジュール(module)或いは関数(function) から参照される。

従ってフラグ変数は，Visual Basic , VBA , Lotus Script 等の言語では，すべてのプロシージャおよびモジュール内で通用するパブリック変数(public variable)としてプロシージャ外で明示的に宣言する。

またC言語の場合は，広域変数(グローバル変数 global variable)として関数外で明示的に宣言する。

フラグ変数は適用範囲の一番広い変数として

Public　　Visual Basic , VBA , Lotus Script

または

extern　　C言語

の変数宣言子を用いて明示的に宣言しなければならない。

下記にフラグ変数の宣言を例示する。

```
'--------------------------------
' flag 変数
'--------------------------------
```

```
'
Public DATAINPUT_flag As Integer
Public DATALOAD_flag  As Integer
Public DATASAVE_flag  As Integer
'
'--------------------------------
```

(Visual Basic , VBA , Lotus Script の例)

5．標識定数(sign constant)の適用範囲と宣言

通常，フラグ変数には 0 または 1 の数値を標識として代入する。例えば，データが入力されている場合にはグラフ画面をみることができ，データが入力されていない場合にはグラフ画面に移行することができないように設定するには，まず，データが入力された時点でデータ入力を識別するフラグ変数(DATAINPUT_flag)に 1 を代入し

```
DATAINPUT_flag = 1
```

グラフ画面の選択ルーチンに於いて

```
IF DATAINPUT_flag = 1 THEN
    CALL GRAPH_SHOW
ELSE
    EXIT SUB
END IF
```

のように IF 文によって識別分岐する。

しかしながら，0 または 1 の数値を用いるより，UP と DOWN，または ON と OFF 等の変数に 0 または 1 をセットしフラグ変数専用の標識定数として宣言した上で，フラグ変数に代入した方が理解し易い。

標識用の定数もフラグ変数と同様に，整数型(short integer)のパブリックな広域定数としてプロシージャ外(または関数外)で宣言し，値の変更ができない変数として指定する const 修飾子を付けて定数化する。

例えば

```
'--------------------------------
' Public 定数
'--------------------------------
'
Public Const UP   As Integer = 1
Public Const DOWN As Integer = 0
'
'--------------------------------
```

のように宣言する。以後 UP = 1, DOWN = 0 の値は変更できなくなり，フラグ変数には「旗が立っているか立っていないか」の識別標識として UP または DOWN を代入すればよい。

ON または OFF 等と命名してもよいが ON は予約語として登録されているケースが多く使用できないことがあるので注意を要する。

前記の IF 文は,

```
DATAINPUT_flag = UP
```

とデータ入力時点で標識定数 UP を代入し

```
IF DATAINPUT_flag = UP THEN
    CALL GRAPH_SHOW
ELSE
    EXIT SUB
END IF
```

と書き替えることができる。

「データ入力フラグが立っていたら(データは入力済みであるので)グラフ画面へ，そうでなければ(データが入力されていなければ)中止」とステートメントの意味が分かり易くなる。

また，フラグ変数を使って 2 つ以上の選択枝に分岐するような場合には，標識定数のかわりに 1，2，3 ～ 10 など複数の数値を直接代入すればよい。

6．フラグ変数の初期化(initialize)

変数の初期化とは変数に 0 が設定(代入)されることである。プロシージャ内或いは関数内の変数はそのプロシージャ或いは関数に入る毎に初期化される。

プロシージャ外或いは関数外で宣言される外部変数は，そのプログラムの実行が開始されたときに一度だけ初期化される。これを自動初期化または暗黙の初期化といい，各変数は自動的にその値が 0 に設定される。

しかし，フラグ変数の初期化は明示的に行った方が分かり易い。プログラムの途中ですべてのフラグ変数を初期化したい場合もあるので，フラグ変数初期化のサブ・ルーチンを準備しておく。

そして，プログラムがロードされたとき一度だけ自動的に実行される自動実行プロシージャ内で明示的に flag 初期化サブ・ルーチンを実行することを習慣付けるとよい。

VBA(Visual Basic for Applications Edition)の例では

```
'--- 自動実行プロシージャ ---
Sub Auto_Open()
    Call flag 初期化
End Sub
```

と自動実行プロシージャ Auto_Open()で flag 初期化のプロシージャを Call(呼出して実行)し，各フラグ変数を明示的に初期化する。

勿論，フラグ変数の初期化プロシージャは別途用意しておかなければならない。

```
'--- flag の初期化 ---
Sub flag 初期化()
    DATAINPUT_flag = DOWN
    DATALOAD_flag  = DOWN
    DATASAVE_flag  = DOWN
End Sub
```

Lotus Script では自動実行サブ・ルーチンは

```
Sub Initialize
```

```
    Call flag 初期化
End Sub
```

となる。

C言語，Visual Basic等自動実行サブ・ルーチンの設定されていない言語使用の場合は，関数外またはプロシージャ外で下記のように明示的に初期化しておくとよい。

```
'--------------------------------
' Public 定数
'--------------------------------
'
Public Const UP   As Integer = 1
Public Const DOWN As Integer = 0
'
'--------------------------------
' flag 変数
'--------------------------------
'
Public DATAINPUT_flag As Integer
Public DATALOAD_flag  As Integer
Public DATASAVE_flag  As Integer
'
DATAINPUT_flag = DOWN
DATALOAD_flag  = DOWN
DATASAVE_flag  = DOWN
'
'--------------------------------
```

プロシージャ，サブルーチン，関数の実行(呼出)にはCall文を省略してもよいが，前例に於いては，プロシージャの実行でCall文を意図的に記載した。

```
Call文　プロシージャ名
Call文　サブルーチン名
```

Call 文　関数名

と Call 文を正式に記述することによりサブルーチンの実行文であることが判然として，プログラムの解読が平易となる。

7．フラグ変数によるプログラム制御

(1) Lotus 1－2－3 Macro

Lotus 1－2－3 W 5 J では，まだ Script 言語がサポートされていないため変数を使用することができず，旧来のマクロ(Macro)言語で工夫するしかない。このような場合には，セル(Cell)に範囲名を付けて変数の代用とする。

表－1

	A	B	C	D	E	F
1						
2			Flag 範囲名	範囲名付きセル		
3			DATAINPUT_flag			
4			DATALOAD_flag			
5			DATASAVE_flag			
6						
7			標識範囲名	範囲名付きセル		
8			UP	1		
9			DOWN	0		
10						

表－1はマクロ記録用シートの一部のセルに範囲名を付けフラグ変数と標識定数を設定した場面である。D 列は範囲名の付いたセル，C 列はその名前の覚え書きである。

D3番地　　DATAINPUT_flag という名前の付いたセル
D4番地　　DATALOAD_flag　という名前の付いたセル
D5番地　　DATASAVE_flag　という名前の付いたセル
D8番地　　UP　　という名前で 1 が入力されている，定数 UP＝1 である。
D9番地　　DOWN という名前で 0 が入力されている，定数 DOWN＝0 である。

以上が1-2-3マクロにおけるフラグ変数と標識定数の宣言に相当する。フラグの初期化は

```
¥0  {PANELOFF}
    {flag 初期化. SUB}
    {EDIT-GOTO A:A1}
    {QUIT}

flag 初期化. SUB  {LET DATAINPUT_flag,DOWN}
                  {LET DATALOAD_flag,DOWN}
                  {LET DATASAVE_flag,DOWN}
                  {RETURN}
```

と自動実行マクロ ¥0 と flag 初期化サブルーチンで行う。

さて，ここに損益計算書のデータを入力して経営分析表や分析結果をグラフ表示する，メニュー(menu)選択形式の「経営分析」アプリケーションがあるとする。当然のことながら，まだデータが入力されていない状態では分析表や分析グラフは正しく表示されない。

データを入力するか分析後保存済みのデータを読み込まないと，分析表やグラフ表示のメニューボタンを選択しても反応しないように flag を用いて設定したい。データを入力すると DATAINPUT_flag，分析済みデータを読込むと DATALOAD_flag に標識定数 UP が代入される。このケースのフローチャートを図－1に示す。

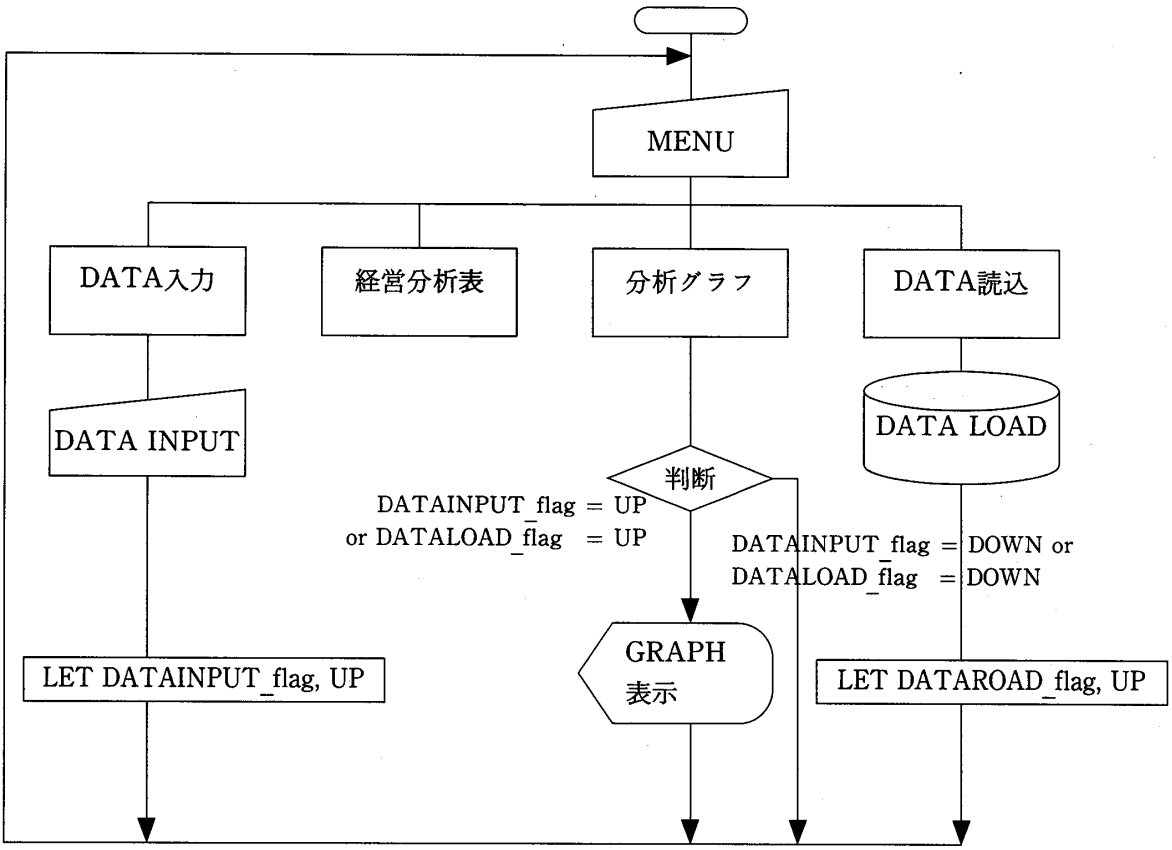
この時の分析グラフマクロは

```
分析 GRAPH    {IF DATAINPUT_flag=DOWN #OR# DATALOAD_flag=DOWN} {BRANCH MENU 画面}
              {EDIT-GOTO D : A1}
              {QUIT}

MENU 画面     {EDIT-GOTO A:A1}
              {QUIT}
```

となって，DATAINPUT_flag か DATALOAD_flag が立っていないとグラフ表示シート(D Sheet)に移動できず，分析グラフは見ることができない。なお，分析表の表示や分析データの保存ルーチンでも同様にしてフラグによるプロセス制御を行うとよい。

また，前記マクロは次のように書いても結果は同じであるが，前例の方が理解し易い。



図－1

分析 GRAPH　{IF DATAINPUT_flag=UP #OR# DATALOAD_flag=UP}{EDIT-GOTO D : A1}
{QUIT}

アプリケーションマクロ(Application Macro)でフラグを活用することにより未完成のグラフや分析表を見せる事を未然に防ぐことが可能となる。

(2) Lotus Script

Lotus 1－2－3 98ではアプリケーション開発言語 Script がサポートされて，Basic やC言語同様に変数が使えるようになった。広域変数や定数は(Globals)オブジェクトの(Declarations)スクリプトまたは(Options)スクリプトで宣言する。

```
(Globals)　　(Declarations)
'--- flag 変数 ---
'
Public DATAINPUT_flag As Integer
```

```
Public DATALOAD_flag  As Integer
Public DATASAVE_flag  As Integer
Public UP   As Integer
Public DOWN As Integer

'--- 標識定数 ---
'
Public Const UP   = 1
Public Const DOWN= 0
```

フラグ変数の初期化はブック起動時に自動実行する(Globals)オブジェクトの Initialize スクリプトで行う。

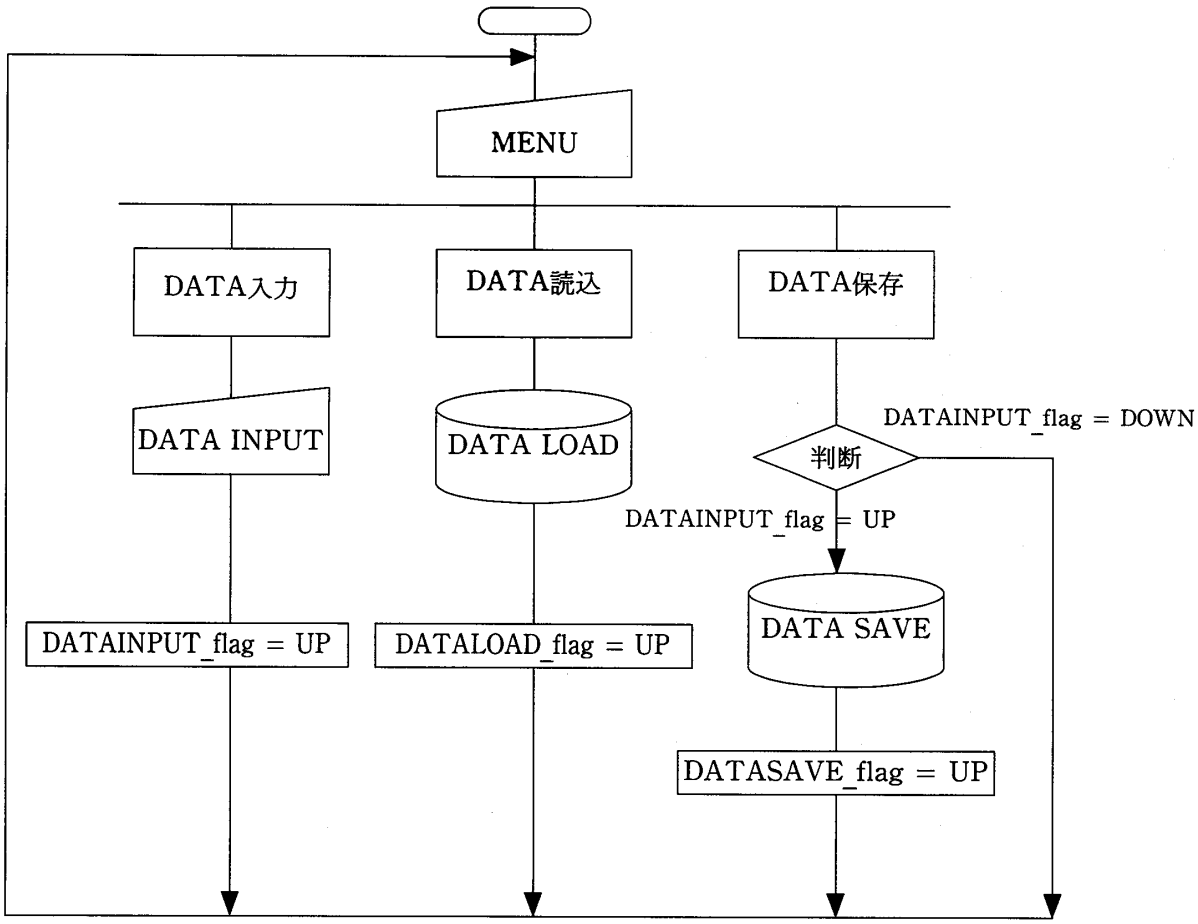
```
'--- 自動実行 Script ---
'
Sub Initialize
    Call flag 初期化
    Call POFF
    Call MENU 画面
End Sub

--- flag 変数の初期化 ---
'
Sub flag 初期化
    DATAINPUT_flag = DOWN
    DATALOAD_flag  = DOWN
    DATASAVE_flag  = DOWN
Sub End
```

なお，POFF，MENU 画面 はパネルオフ，メニュー画面表示のプロシージャ名である。

前例の「経営分析」アプリケーションで入力したデータを保存するプロシージャを考えてみる。データが入力されていない場合は保存ボタンをクリックしても反応しないように設定する。データが入力された時点で DATAINPUT_flag を立て，データを保存したら

DATASAVE_flag を立てる。



図－2

図－2がそのフローチャートである。データが未入力(DATAINPUT_flag = DOWN)の場合は DATA 保存 のメニューボタンをクリックしても作動せず，メニュー選択画面に戻る。

```
Sub DATA 保存 Click(Source As Buttoncontrol)
    If DATAINPUT_flag = DOWN Then Exit Sub
    Call 解放
    Call DATA_SAVE
    Call 保護
End Sub
```

これはメニューの DATA 保存 ボタンをクリックした時のボタンコントロールスクリプトである。解放，保護は画面解放•保護のプロシージャ，DATA_SAVE はデータ保存

のプロシージャで，それぞれ(Globals)オブジェクトの(Declarations)スクリプトに用意する。

If 文と flag 変数で空っぽのデータを保存する愚を回避することができる。

(3) Excel97 VBA (Visual Basic Applications Edition)

VBA は Basic 言語を基準にした汎用性のあるマクロ言語である。VBA のフラグ変数と標識定数の宣言およびフラグ変数の初期化についてはすでに述べた。

人間は時として，ついうっかり忘れたり間違えたりするものである。前例の「経営分析」アプリケーションにおいて，経営データを入力して経営分析を行ったがそのデータを保存しないで，うっかり[終了]ボタンをクリックしてしまったような場合に備えて，フラグ変数を用いた救助策を用意したい。

フラグ変数は，データを入力したとき　DATAINPUT_flag = UP
　　データを保存したとき　DATASAVE_flag = UP

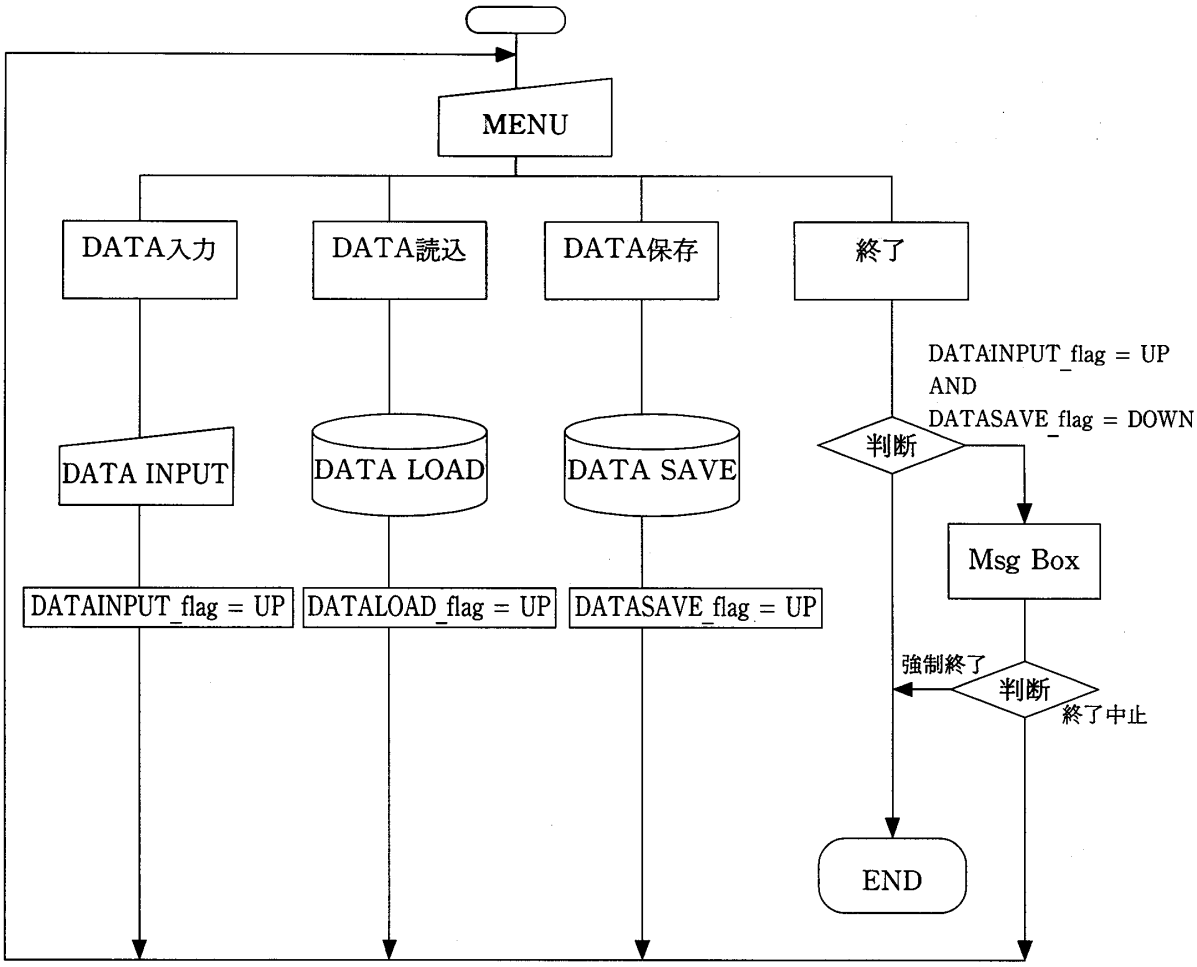
に設定される。[終了]ボタンをクリックしたとき，

DATAINPUT_flag = UP で DATASAVE_flag = DOWN

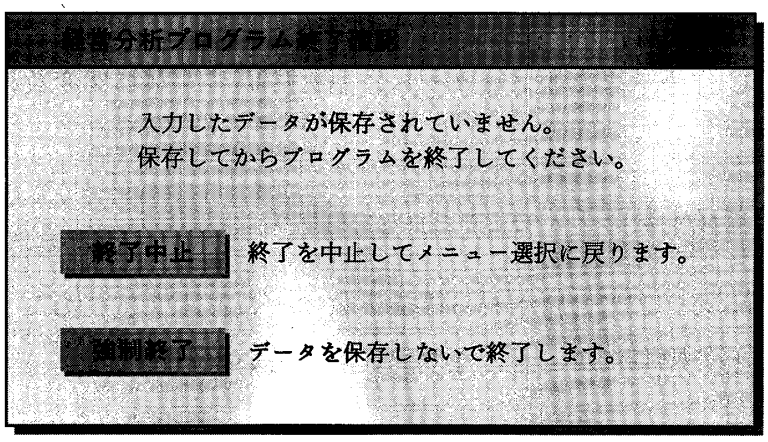
であれば，データは入力されているが，そのデータはまだ保存されていない状態と判断できる。その場合には注意のメッセージボックスを表示するようプログラミングする。図－3にそのフローチャートを示す。

メッセージボックスはユーザーフォームで作成する。図－4はそのサンプルである。このメッセージボックスは VBA のプログラム終了プロシージャで LOAD する。

```
Sub PROGRAM_END()
    If DATAINPUT_flag = UP And DATASAVE_flag = DOWN Then
        Load 終了確認 BOX
        終了確認 BOX.Show
    Else
        ActiveWorkbook.Close(False)
    End If
End Sub
```



図－3



図－4

プログラム終了のメニュー選択時点で，入力データが保存されていればプログラムは終了するが，データは入力されているのに保存が実行されていない(DATAINPUT_flag = UP And DATASAVE_flag = DOWN)場合は，図－4のメッセージボックスが表示される。

終了中止 ボタンをクリックするとメニュー選択画面へ戻りデータを保存することができる。テストランやデータ保存の必要がない場合は 強制終了 ボタンをクリックすればプログラムは強制的に終了する。

各ボタンにはそれぞれユーザーフォームのクリックボタン用コードを次のように記述しなければならない。 終了中止 ボタンには

```
Private Sub 終了中止_Click()
    Unload 終了確認
End Sub
```

強制終了 ボタンのプロシージャは

```
Private Sub 強制終了_Click()
    Unload 終了確認
    ActiveWorkbook.Close(False)
End Sub
```

である。

うっかりミスやコンピュータ操作に不慣れなユーザーをサポートする，親切設計のプログラミングにフラグ変数操作は欠かせぬものである。

Visual Basic，C 言語，その他プログラム言語に於いてもフラグ変数の使い方は同じである。

8．フラグ変数と標識定数のまとめ

フラグ変数とそのフラグに付属させる標識定数は，これまで述べたように一定の基準に従って記述することで標準化が可能となり，理解し易いものとなる。

(1) フラグ変数の型は 整数型(short integer)とする。
(2) フラグ変数の名前は分かり易く命名し，フラグ変数であることが判別し易いよう拡張子として _flag を付ける。
(3) フラグ変数の適用範囲はパブリックな広域変数とし，プロシージャ外または関数外で明示的に宣言する。

```
Public DATAINPUT_flag As Integer
```

(4) フラグ変数に付属させる標識はパブリックな定数として宣言する。

```
Public Const UP   As Integer = 1
Public Const DOWN As Integer = 0
```

(5) フラグ変数は明示的に初期化する。

```
Sub flag 初期化()
    DATAINPUT_flag = DOWN
    DATALOAD_flag  = DOWN
    DATASAVE_flag  = DOWN
End Sub
```

(6) フラグ変数によるプログラムコントロールは，フラグ変数に代入されている標識を識別判断して If ～ Then ～ Else 文等の制御文で行う。
複数に分岐するプログラム制御には標識定数(UP,DOWN 等)の代わりに1,2,3 ～ 10の数値を直接フラグ変数に代入し，Select Case 文，Switch Case 文等の複数分岐制御文を用いてコントロールする。

以上がフラグ変数と標識定数およびフラグ変数によるプログラム制御の要約である。

これらフラグによる認識や制御は社会生活においてもいろいろな場面で用いられている。例えば，交差点における交通信号も信号というフラグである。

(フラグ)	(認識)
赤信号	危険・止まれ
黄信号	注意
青信号	安全・進め

歩行者や自動車の運転者は，赤信号のフラグが立っていると脳細胞のフラグレジスタに「危険・止まれ」のフラグが立ち(「危険・止まれ」と認識し)，交差点の手前で停止する。

その他日常生活上では，自動車のウインカー，自動車のストップランプ，エレベータの

階数表示盤，「在室」・「外出中」などの表示板等視覚に訴えるフラグ，自動車のクラクション，電話のベル等聴覚に訴えるフラグなど数多くのフラグ(標識)が活用されている。

プログラミングにおけるフラグ変数は実社会での「認識と制御」と同様，プログラム制御のツールとして，各種プログラム言語のあらゆる場面で応用することができる。

フラグ変数の活用で複雑なプログラム制御が可能となり，ユーザー・フレンドリーなプログラムも容易に構築することができる。プログラム自体がその働きをコントロールし学習するアプリケーションの開発などフラグ変数の応用範囲は奥深く幅広いものである。

注：1　　出典　図解コンピュータ用語辞典
注：2　　出典　コンピュータ用語辞典およびパソコン用語辞典

参考文献

Excel 97 VBA リファレンス　マイクロソフト(株)
Microsoft Visual Basic プログラミングガイド　マイクロソフト(株)
Lotus 1-2-3 R5J ユーザースガイド　Lotus Development Corporation
Lotus Script ハンドブック　Lotus Development Corporation
新C言語入門　林　晴比古　著　ソフトバンク(株)
図解コンピュータ用語辞典　(株)富士書房
コンピュータ用語辞典　マイクロソフト(株)
パソコン用語辞典　技術評論社