愛知県立三谷水産高等学校　教諭　山田　学

水産海洋技術〔ゲームプログラミング〕　No.01

■ キー入力　Inkey$

どのボタンが押されているかを判断するために、キー入力操作の仕方を学びます。
プログラム実行後に、文字や数値を入力する命令に Input がありましたが、
Input は、入力されるまでプログラムが停止するため、ゲーム等でキャラクターを動かしたり
するには、使いにくいです。ここで使用する命令が、Inkey$ です。

（使い方）　A$=Inkey$　　命令実行時に押されているキーが A$に代入される。

Step01　キー入力　Inkey$

```
Cls
A$=Inkey$
```

実行しても、プログラムが一瞬で終了してしまいます。
Inkey$ は、プログラムがその場で停止しないため、工夫が必要です。

Step02　キー入力の工夫

```
Cls
A$=""
While(A$="")
  A$=Inkey$
WEnd
```

While 命令を使用し、A$ が空っぽ(何も入力されていない)間、繰り返すようにします。
ボタンが押されると、A$ に文字が代入されるため、繰り返しを終了します。

Step03　入力された文字を表示する

```
Cls
A$=""
While(A$="")
  A$=Inkey$
WEnd
Print A$
```

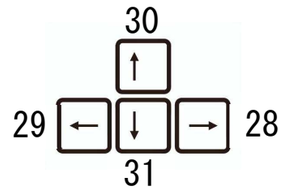
Print A$ とすれば、代入された文字を表示できます。
しかし、矢印キーや、Enter キー、ESC キーなど、特殊なキーの表示ができません。
特殊なキーが押されたとき、それを判断する場合には、キーコードに変換して処理します。

Step04　キーコードで表示する

```
Cls
A$=""
While(A$="")
  A$=Inkey$
WEnd
Print ASC(A$)
```

ASC( ) で囲うと、文字を文字コードに変換して表示できます。

これを応用して、カーソルキーのキーコードを調べて判断することができます。



※生徒に調べさせて覚えておくよう指示するとよいです。

Step05　キーコードで表示する

```
Cls
A$=""
While(A$="")
  A$=Inkey$
WEnd
If ASC(A$)=28 Then Print "右"
```

右カーソルキーを押すと、"右"と表示されます。

問題　左・上・下 も同様にプログラムしてみよう。

Step06　グラフィック表示してみよう

```
GScreen(400,400)
X=200:Y=200
PSet(X,Y),0
```

画面の中央に黒い点を描きます。　※ PSet(,) は、1 ドット点を描く命令

ST という変数を用意して、0の間繰り返すようにします。

```
GScreen(400,400)
X=200:Y=200
ST=0
While(ST=0)
  PSet(X,Y),0
WEnd
```

繰返しの中でキー入力を行い、Enter キー(13 番)が押されたら、ST を 1 にして繰返しを
終了するようにします。

```
GScreen(400,400)
X=200:Y=200
ST=0
While(ST=0)
  PSet(X,Y),0
  A$=""
  While(A$="")
    A$=Inkey$
  WEnd
  If ASC(A$)=13 Then
    ST=1
  End If
Wend
```

右カーソルキー(28 番)が押されたら、X の値を +1 するようにします。

```
GScreen(400,400)
X=200:Y=200
ST=0
While(ST=0)
  PSet(X,Y),0
  A$=""
  While(A$="")
    A$=Inkey$
  WEnd
  If ASC(A$)=13 Then
    ST=1
  End If
  If ASC(A$)=28 Then
    X=X+1
  End If
WEnd
```

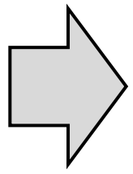
右カーソルキーを押すと、黒の点が右に移動していきます。
また、Enter キーを押すと、プログラムが終了します。

問題　左・上・下 も移動するようにプログラムしてみよう。

ゲーム化します

Step07　自動的に動いていき、カーソルキーで向きを変えられるように変更する

```
GScreen(400,400)
X=200:Y=200
ST=0
While(ST=0)
  PSet(X,Y),0
  A$=""
  While(A$="")
    A$=Inkey$
  WEnd
  If ASC(A$)=13 Then
    ST=1
  End If
  If ASC(A$)=28 Then
    X=X+1
  End If
  If ASC(A$)=29 Then
    X=X-1
  End If
  If ASC(A$)=30 Then
    Y=Y-1
  End If
  If ASC(A$)=31 Then
    Y=Y+1
  End If
WEnd
```



```
GScreen(400,400)
X=200:Y=200:MX=1:MY=0
ST=0
While(ST=0)
  A$=""
  While(A$="")
    A$=Inkey$
    X=X+MX
    Y=Y+MY
    PSet(X,Y),0
  WEnd
  If ASC(A$)=13 Then
    ST=1
  End If
  If ASC(A$)=28 Then
    MX=1:MY=0
  End If
  If ASC(A$)=29 Then
    MX=-1:MY=0
  End If
  If ASC(A$)=30 Then
    MX=0:MY=-1
  End If
  If ASC(A$)=31 Then
    MX=0:MY=1
  End If
WEnd
```



```
GScreen(400,400)
X=200:Y=200:MX=1:MY=0
ST=0
While(ST=0)
  A$=""
  While(A$="")
    A$=Inkey$
    X=X+MX
    Y=Y+MY
    PSet(X,Y),0
  WEnd
  If ASC(A$)=13 Then
    ST=1
  End If
  If ASC(A$)=28 Then
    MX=1:MY=0
  End If
  If ASC(A$)=29 Then
    MX=-1:MY=0
  End If
  If ASC(A$)=30 Then
    MX=0:MY=-1
  End If
  If ASC(A$)=31 Then
    MX=0:MY=1
  End If
WEnd
```

移動方向用の変数を用意します。

何もキーを押していない時に、自動的にその方向へ移動させます。

カーソルキーを押したときは、移動方向を変更します。
MX は横の移動方向
MY は縦の移動方向

Step08　ゲームオーバーを考える

ゲームオーバーとなる場合は、以下の２点があります。
①　自分の線にあたった　（点を描く際、すでに黒色の点が書かれている）
②　画面からはみ出した　（変数ＸとＹが、０より小さい、399 より大きいときゲームオーバーとする）

①は、GetRGBPixel(x,y) 命令によって、その場所の色情報を 16 進数で取得できるため、黒色(#000000)なら "GAME OVER" とします。

```
If GetRGBPixel(x,y)="#000000" Then
   Cls:Print "GAME OVER"
   End
End If
```

②は、ＸとＹの値が、範囲外かどうかで判断します。

```
If X<0 Or X>399 Or Y<0 Or Y>399 Then
   Cls:Print "GAME OVER"
   End
End If
```

①は点を描く前、②点を描いた後に挿入します。

問題　変数 SC を使って、SCORE が最後に表示されるようにしてみよう。

愛知県立三谷水産高等学校　教諭　山田　学

水産海洋技術〔ゲームプログラミング〕　No.02

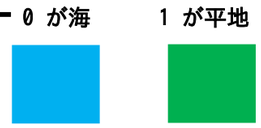
■ マップスクロール

自分でマップを作成し、そのマップを歩くことができるプログラムを作成します。
今回の画面表示は３×３とします。

Step01　マップデータを一次元配列で作成します。

```
GScreen(300,300)
Dim A$(10)
A$( 1)="0000000000"
A$( 2)="0110000110"
A$( 3)="0011111100"
A$( 4)="0001111000"
A$( 5)="0011001110"
A$( 6)="0111001110"
A$( 7)="0011111000"
A$( 8)="0001111110"
A$( 9)="0111000010"
A$(10)="0000000000"
```

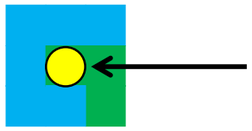
0 が海　　1 が平地



Step02　現在地を設定します。

```
X=2:Y=2
```

Step03　自分を中心に３×３のマップを表示します。



○が自分の位置（X,Y）
その手前から、一つ先までを表示するため
横は X-1～X+1 まで
縦は Y-1～Y+1 までを表示する。

```
For I=-1 To 1
  For J=-1 To 1
    C=Val(Mid$(A$(Y+I)),X+J,1))
    If C=0 Then
      Line((J+1)*100,(I+1)*100)-((J+2)*100,(I+2)*100),9,BF
    End If
    If C=1 Then
      Line((J+1)*100,(I+1)*100)-((J+2)*100,(I+2)*100),10,BF
    End If
  Next
Next
```

自分が中心にいるように見せるために、黄色の円を描く。

```
Circle(150,150),49,14,,,,F
```

Step04　キー入力をします。

```
K$=""
While(K$="")
  K$=Inkey$
WEnd
Select Case ASC(K$)
  Case 28:X=X+1
  Case 29:X=X-1
  Case 30:Y=Y-1
  Case 31:Y=Y+1
End Select
```

前回 If 命令で行っていた判断を
Select 命令で記述しました。

以降、マップ表示の部分までを ESC(コード 27)が押されるまで繰り返すようにする。

```
GScreen(300,300)
Dim A$(10)
A$( 1)="0000000000"
A$( 2)="0110000110"
A$( 3)="0011111100"
A$( 4)="0001111000"
A$( 5)="0011001110"
A$( 6)="0111001110"
A$( 7)="0011111000"
A$( 8)="0001111110"
A$( 9)="0111000010"
A$(10)="0000000000"
X=2:Y=2
ST=0
While(ST=0)
  For I=-1 To 1
    For J=-1 To 1
      C=Val(Mid$(A$(Y+I),X+J,1))
      If C=0 Then
        Line((J+1)*100,(I+1)*100)-((J+2)*100,(I+2)*100),9,BF
      End If
      If C=1 Then
        Line((J+1)*100,(I+1)*100)-((J+2)*100,(I+2)*100),10,BF
      End If
    Next
  Next
  Circle(150,150),49,14,,,,F
  K$=""
  While(K$="")
    K$=Inkey$
  WEnd
  Select Case ASC(K$)
    Case 27:ST=1
    Case 28:X=X+1
    Case 29:X=X-1
    Case 30:Y=Y-1
    Case 31:Y=Y+1
  End Select
WEnd
```

Step05　海の上を歩けないようにする。

キーを押して、ＸまたはＹの値を変化させる前に、
ＢＸ＝Ｘ：ＢＹ＝Ｙ
を実行して、ＸとＹの値をＢＸとＢＹにコピーしておく。

その後、ＸまたはＹの値を変化させた後に、マップを調べて、海ならば
Ｘ＝ＢＸ：Ｙ＝ＢＹ
として、変更前の値に戻すことで、移動していないことにする。

```
BX=X:BY=Y                    ← X と Y の値をコピーしておく
Select Case ASC(K$)
  Case 27:ST=1
  Case 28:X=X+1
  Case 29:X=X-1
  Case 30:Y=Y-1
  Case 31:Y=Y+1
End Select
C=Val(Mid$(A$(Y),X,1))       ← 移動後のマップを調べる
If C=0 Then                  ← 海ならば
  X=BX:Y=BY                     X と Y の値をもとに戻す
End If
```

これで、海に入れなくなります。

問題１　マップを大きくしてみよう。

横に広げる場合は、そのまま数字を増やしてください。

（例）　A$(1)=”00000000000000000000000000000000”

縦に広げる場合は、配列の数を増やして、行を増やしていきます。

```
（例）　Dim A$(30)
        A$(11)=”0111111110”
        A$(12)=”0111111110”
```

問題２　マップを拡張してみよう。

マップデータ中に２を追加し、２のマップは濃い緑（色番号２）で表示する。

マップデータ中に３を追加し、３のマップは灰色（色番号８）で表示する。

（例）

```
A$( 1)=”0000000000”
A$( 2)=”0112000210”
A$( 3)=”0012222100”
A$( 4)=”0001221000”
A$( 5)=”0013001130”
A$( 6)=”0111001110”
A$( 7)=”0011111000”
A$( 8)=”0001113110”
A$( 9)=”0111000010”
A$(10)=”0000000000”
```


完成プログラム

```
GScreen(300,300)
Dim A$(10)
A$( 1)=”0000000000”
A$( 2)=”0112000210”
A$( 3)=”0012222100”
A$( 4)=”0001221000”
A$( 5)=”0013001130”
A$( 6)=”0111001110”
A$( 7)=”0011111000”
A$( 8)=”0001113110”
A$( 9)=”0111000010”
A$(10)=”0000000000”
X=2:Y=2
ST=0
While(ST=0)
  For I=-1 To 1
    For J=-1 To 1
      C=Val(Mid$(A$(Y+I),X+J,1))
      If C=0 Then
        Line((J+1)*100,(I+1)*100)-((J+2)*100,(I+2)*100),9,BF
      End If
      If C=1 Then
        Line((J+1)*100,(I+1)*100)-((J+2)*100,(I+2)*100),10,BF
      End If
      If C=2 Then
        Line((J+1)*100,(I+1)*100)-((J+2)*100,(I+2)*100),2,BF
      End If
      If C=3 Then
        Line((J+1)*100,(I+1)*100)-((J+2)*100,(I+2)*100),8,BF
      End If
    Next
  Next
  Circle(150,150),49,14,,,,F
  K$=””
  While(K$=””)
    K$=Inkey$
  WEnd
  BX=X:BY=Y
  Select Case ASC(K$)
    Case 27:ST=1
    Case 28:X=X+1
    Case 29:X=X-1
    Case 30:Y=Y-1
    Case 31:Y=Y+1
  End Select
  C=Val(Mid$(A$(Y),X,1))
  If C=0 Then
    X=BX:Y=BY
  End If
WEnd
```

愛知県立三谷水産高等学校　教諭　山田　学

水産海洋技術〔ゲームプログラミング〕　No.03

■ 棒倒し法アルゴリズムを使用した迷路ゲーム

教科書 P116～P117 の棒倒し法アルゴリズムを使用した、迷路自動作成ゲームです。

Step01　棒倒しアルゴリズムを使って二次元配列に迷路を作成します。

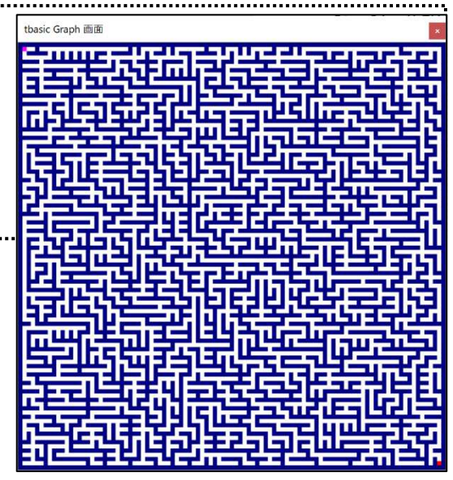
```
Randomize
Gscreen(507,507)
DIM M(101,101)
For I=1 To 101 Step 1
  M(I,1)=1:M(I,101)=1:M(1,I)=1:M(101,I)=1
Next
For Y=3 To 99 Step 2
  For X=3 To 99 Step 2
    M(X,Y)=1
    XT=0:YT=0
    While(M(X+XT,Y+YT)=1)
      If Y=3 Then
        A=Int(Rnd*4)
      Else
        A=Int(Rnd*3)+1
      End If
      Select Case A
        Case 0:XT=0 :YT=-1
        Case 1:XT=1 :YT=0
        Case 2:XT=0 :YT=1
        Case 3:XT=-1:YT=0
      End Select
    Wend
    M(X+XT,Y+YT)=1
  Next
Next
```

四方を壁で囲む

棒倒しアルゴリズムで迷路作成
〔教科書 P116～P117〕

Step02　作成した迷路をグラフィックで画面に表示する。

```
For Y=1 To 101 Step 1
  For X=1 To 101 Step 1
    If M(X,Y)=1 Then
      Line(X*5-5,Y*5-5)-(X*5,Y*5),1,BF
    End If
  Next
Next
Line(99*5,99*5)-(100*5,100*5),12,BF
```



Step03　カーソルキーで迷路を歩けるようにする。

画面右下の赤い部分(X=99,Y=99 の位置)に到達するとゴールです。

```
X=2:Y=2:ST=0
While(ST=0)
  Line(X*5-5,Y*5-5)-(X*5,Y*5),13,BF
  If X=99 And Y=99 Then
    Cls:Print "ゴール"
    End
  End If
  A$=""
  While(A$="")
    A$=Inkey$
  Wend
  BX=X:BY=Y
  Select Case ASC(A$)
    Case 27:ST=1
    Case 28:X=X+1
    Case 29:X=X-1
    Case 30:Y=Y-1
    Case 31:Y=Y+1
  End Select
  If M(X,Y)=1 Then
    X=BX:Y=BY
  End If
  Line(BX*5-5,BY*5-5)-(BX*5,BY*5),14,BF
Wend
```

座標(99,99)ならゴール

キー入力処理

押した方向によって X と Y の値を変更
ESC(27 番)を押したとき、ST の値を 1 にして、繰返しを抜けるようにする。

移動先が壁(ﾏｯﾌﾟﾃﾞｰﾀ 1)ならば、移動する前の X,Y のバックアップ BX,BY に戻す

キー入力の詳細や壁を貫通できないようにする処理については、ゲームプログラミング No.02 のプリントで解説しています。

問題　スタート位置とゴールの位置を変更してみよう。

水産海洋技術〔ゲームプログラミング〕　No.04

■ ブラックジャック(BlackJack)の作成

トランプを使った、ブラックジャック(BlackJack)の作成をします。
〔ルール〕
最初に２枚のカードを配ります。
カードの合計が 21 に近いほど強く、21 を超えると負けとなります。
カードの 10 以上は、10 として加算し、１は１または 11 として加算できます。

Step01　グラフィック画面の初期設定

```
Randomize
GBackColor=2:GForeColor=15
GScreen(600,100)
```

Step02　カードを作成します

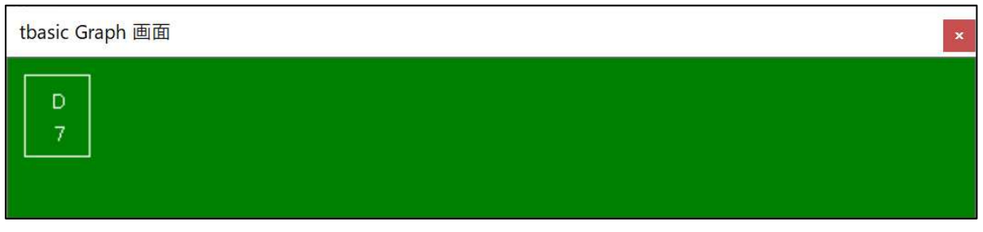
```
Dim T(52),M(52)                          ← 配列Ｔをカードの数字、Ｍをマークとします
For I=1 to 13                            ← カードの数字は１～13
  For J=1 to 4                           ← マークは４種類　1:ﾀﾞｲﾔ　2:ﾊｰﾄ　3:ｸﾛｰﾊﾞ　4:ｽﾍﾟｰﾄﾞ
    T((I-1)*4+J)=I                       ← 配列の１番目～52 番目となるように計算しています。
    M((I-1)*4+J)=J
  Next
Next
```

Step03　カードをシャッフルします

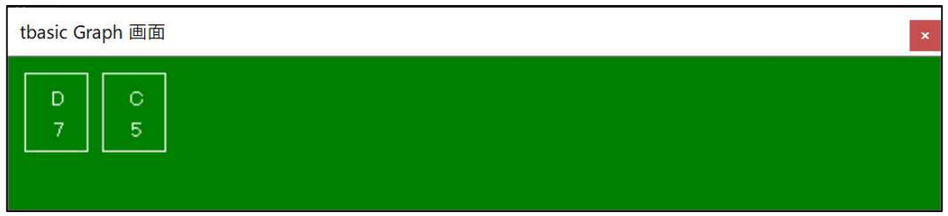
```
For I=1 to 100                           ← 100 回シャッフルします。
  A=Int(Rnd*52)+1                        ← 1～52 のランダムを２つ出して、
  B=Int(Rnd*52)+1                          その２枚を入れ替えます。
  C=T(A):T(A)=T(B):T(B)=C                ← カードの数字の入れ替え
  C=M(A):M(A)=M(B):M(B)=C                ← マークの入れ替え
Next
```

Step04　カードを表示します

```
Line(10,10)-(50,60),15,B                 ← 白い四角を描画
GLocate(28,20):GPrint Mid$("DHSC",M(1),1)← １枚目のマークを DHSC で表示
GLocate(25,40):GPrint T(1)               ← １枚目のカードの数字を表示
```

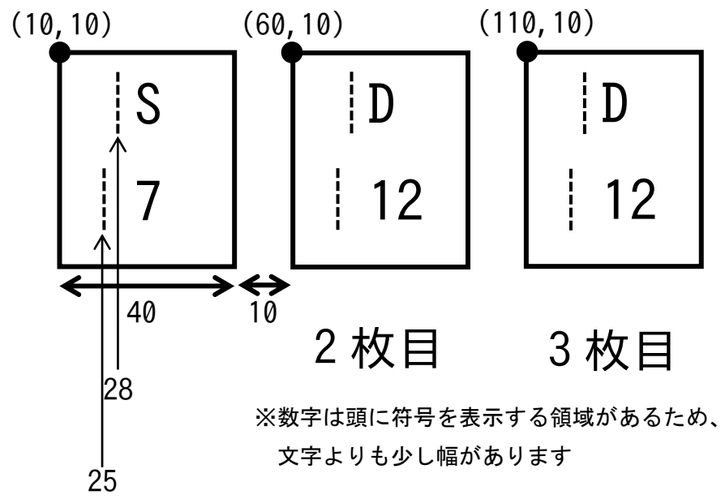


Step05　２枚目以降のカードを表示します



２枚目以降のカードは、右にずれて表示させたいため、計算をします。
何枚目のカードかを、変数Ｃを使って記憶します。

```
C=2
Line(C*50-40,10)-(C*50,60),15,B
GLocate(C*50-22,20):GPrint Mid$("DHSC",M(C),1)
GLocate(C*50-25,40):GPrint T(C)
```



※数字は頭に符号を表示する領域があるため、
文字よりも少し幅があります

Mid$(　)は、文字列の、指定番目から、指定文字数抜き出す命令。

（例）　A$=Mid$("ABCDEFG",2,3)　　← 文字列の２番目から３文字を A$に代入する

カードのマーク(配列 M)は、
１：ダイヤ(D)、２：ハート(H)、３：スペード(S)、４：クローバ(C)、
としているため、

```
Mid$("DHSC",M(C),1)
```

によって、マークの１～４を、文字の DHSC に変換している。

Step06　もう一枚引くかどうかをキー入力する


```
GLocate(100,80):GPrint ”もう一枚引きますか？［Y］はい［その他］いいえ”
K$=””
While(K$=””)
  K$=Inkey$
WEnd
C=C+1                                ← 次のカードとするため C を加算する。
```

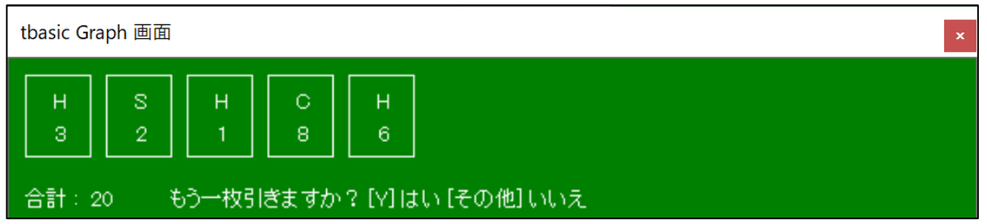
Step07　″Y″以外が押されるまで繰り返しとする

ここまでのプログラムの全体像

```
Randomize
GBackColor=2:GForeColor=15
GScreen(600,100)
Dim T(52),M(52)
For I=1 to 13
  For J=1 to 4
    T((I-1)*4+J)=I
    M((I-1)*4+J)=J
  Next
Next
For I=1 to 100
  A=Int(Rnd*52)+1
  B=Int(Rnd*52)+1
  C=T(A):T(A)=T(B):T(B)=C
  C=M(A):M(A)=M(B):M(B)=C
Next
Line(10,10)-(50,60),15,B
GLocate(28,20):GPrint Mid$(”DHSC”,M(1),1)
GLocate(25,40):GPrint T(1)
C=2
K$=”1”
While(K$=”1”)
  Line(C*50-40,10)-(C*50,60),15,B
  GLocate(C*50-22,20):GPrint Mid$(”DHSC”,M(C),1)
  GLocate(C*50-25,40):GPrint T(C)
  GLocate(100,80):GPrint ”もう一枚引きますか？［Y］はい［その他］いいえ”
  K$=””
  While(K$=””)
    K$=Inkey$
  WEnd
  C=C+1
WEnd
```

プログラム前半部分

Step08　合計値の計算と、21 を超えたときの負け判定



プログラム前半部分

```
C=2
K$=”1”
While(K$=”1”)
  Line(C*50-40,10)-(C*50,60),15,B
  GLocate(C*50-22,20):GPrint Mid$(”DHSC”,M(C),1)
  GLocate(C*50-25,40):GPrint T(C)
  SUM=0
  For I=1 To C
    If T(I)>=10 Then
      SUM=SUM+10
    ElseIf T(I)=1 Then
      SUM=SUM+11
    Else
      SUM=SUM+T(I)
    End If
  Next
  If SUM>21 Then
    SUM=0
    For I=1 To C
      If T(I)>=10 Then
        SUM=SUM+10
      Else
        SUM=SUM+T(I)
      End If
    Next
  End If
  Line(0,80)-(600,100),2,BF
  GLocate(10,80):GPrint ”合計 : ”+Str$(SUM)
  If SUM>21 Then
    GLocate(100,80):GPrint ”あなたの負けです。”
    End
  End If
  GLocate(100,80):GPrint ”もう一枚引きますか？［Y］はい［その他］いいえ”
  K$=””
  While(K$=””)
    K$=Inkey$
  WEnd
  C=C+1
WEnd
```

合計の計算
カードの数字 10 以上は 10 として加算
カードの 1 は、11 として加算、
それ以外は数字の値のまま加算

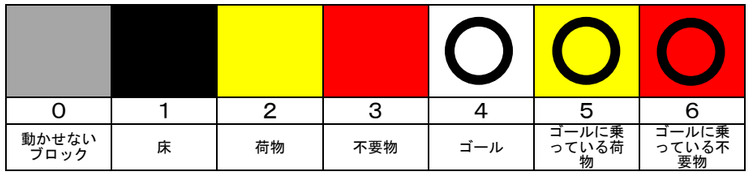
合計が 21 を超えていたら、
カードの数字 1 を 1 として計算し直す

水産海洋技術〔ゲームプログラミング〕　No.05

■ 倉庫番の作成

ゲーム「倉庫番」とは、すべての荷物をゴール地点まで運べばクリアとなるパズルゲームです。荷物の他に、外壁など押せないブロックと、荷物以外の押せるブロックが存在します。また、荷物は引くことができず、２個以上を押すこともできません。

今回は、以下のように荷物やブロックの値を設定する。



0	1	2	3	4	5	6
動かせないブロック	床	荷物	不要物	ゴール	ゴールに乗っている荷物	ゴールに乗っている不要物

Step01　パズルデータの設定

```
GBackColor=Black : GScreen(600,500)
Dim M(12,10)                              ← 横 12 マス、縦 10 マス
For I=1 To 10
  Read A$
  For J=1 To 12
    M(J,I)=Val(Mid$(A$,J,1))              ← Data から読み取って、１つずつ
  Next                                      ２次元配列に格納する。
Next
Data ”000000000000”
Data ”000001110000”
Data ”000111010000”
Data ”011102111000”
Data ”010243401000”
Data ”011240421100”
Data ”001044420100”
Data ”001022011100”
Data ”001111110000”
Data ”000000000000”
X=6:Y=9:Z=7                               ← 自分の位置 X,Y と荷物の総数 Z
```

Step02　画面表示を行うサブルーチンを作成

```
Sub Hyouzi(X,Y,M())                       ← 自分の位置とパズルデータを受け取る
  Cls 3
  For I=1 To 10
    For J=1 To 12
      Select Case M(J,I)                  ← 値によって色を変えて表示する
        Case 0 : Line(J*50-50,I*50-50)-(J*50,I*50),8,BF
        Case 2 : Line(J*50-50,I*50-50)-(J*50,I*50),14,BF
        Case 3 : Line(J*50-50,I*50-50)-(J*50,I*50),12,BF
        Case 4 : Circle(J*50-25,I*50-25),10,14
        Case 5 : Line(J*50-50,I*50-50)-(J*50,I*50),14,BF
        Case 6 : Line(J*50-50,I*50-50)-(J*50,I*50),12,BF
      End Select
    Next
  Next
  Circle(X*50-25,Y*50-25),23,10:Circle(X*50-25,Y*50-25),15,10,,,,F
End Sub
```

Step03　キー入力と移動処理

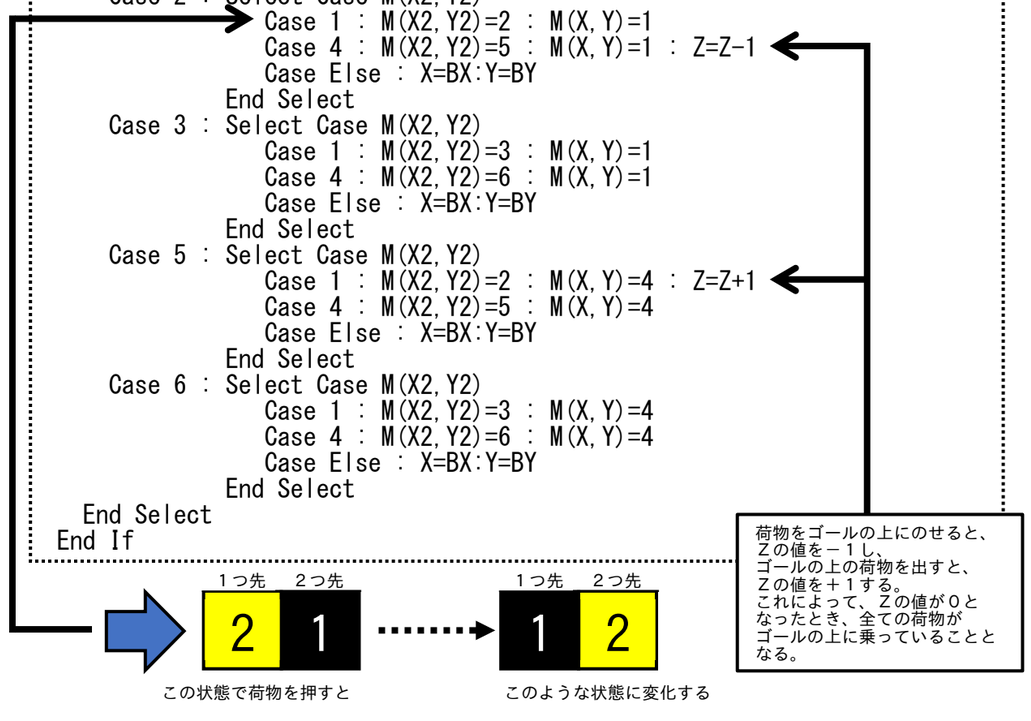
```
Call Hyouzi(X,Y,M())                      ← サブルーチンへ飛んで画面表示
A$=””
While(A$=””)
  A$=Inkey$
WEnd
BX=X:BY=Y
Select Case Asc(A$)
  Case 27 : End
  Case 28 : X=X+1:X2=X+1:Y2=Y             ← X,Y は移動先の座標へ変更
  Case 29 : X=X-1:X2=X-1:Y2=Y             ← X2,Y2 は２つ先の座標を代入
  Case 30 : Y=Y-1:X2=X:Y2=Y-1
  Case 31 : Y=Y+1:X2=X:Y2=Y+1
End Select
```

倉庫番では、荷物を押すときもう一つ先を見て押せるか押せないかを判断するため、1 つ先の座標と 2 つ先の座標を求めている。

Step04　荷物の移動処理

移動先（１つ先）が荷物か不要物で、その先（２つ先）が床かゴールなら押すことができる。
それ以外の組み合わせは、すべて押すことができない。

```
If M(X,Y)=0 Then                          ← 移動先が動かせないブロックのとき
  X=BX:Y=BY                               ← 移動前の値に戻す
Else
  Select Case M(X,Y)                      ← M(X,Y):1 つ先    M(X2,Y2):2 つ先
    Case 2 : Select Case M(X2,Y2)
               Case 1 : M(X2,Y2)=2 : M(X,Y)=1
               Case 4 : M(X2,Y2)=5 : M(X,Y)=1 : Z=Z-1
               Case Else : X=BX:Y=BY
             End Select
    Case 3 : Select Case M(X2,Y2)
               Case 1 : M(X2,Y2)=3 : M(X,Y)=1
               Case 4 : M(X2,Y2)=6 : M(X,Y)=1
               Case Else : X=BX:Y=BY
             End Select
    Case 5 : Select Case M(X2,Y2)
               Case 1 : M(X2,Y2)=2 : M(X,Y)=4 : Z=Z+1
               Case 4 : M(X2,Y2)=5 : M(X,Y)=4
               Case Else : X=BX:Y=BY
             End Select
    Case 6 : Select Case M(X2,Y2)
               Case 1 : M(X2,Y2)=3 : M(X,Y)=4
               Case 4 : M(X2,Y2)=6 : M(X,Y)=4
               Case Else : X=BX:Y=BY
             End Select
  End Select
End If
```



Step05　Ｚの値が０になるまで繰り返す

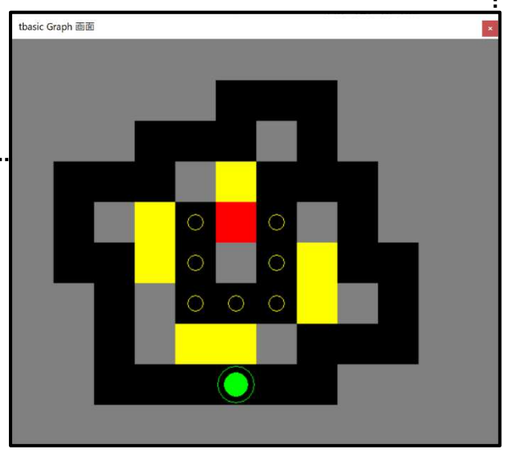
Step03～Step04 の処理を、Ｚの値が０となるまで繰り返す While 命令を入れる。
繰返し While 命令を抜けたとき、全ての荷物がゴールの上に乗っていることになる。

全体のプログラム

```
GBackColor=Black : GScreen(600,500)
Dim M(12,10)
For I=1 To 10
  Read A$
  For J=1 To 12
    M(J,I)=Val(Mid$(A$,J,1))
  Next
Next
Data ”000000000000”
Data ”000001110000”
Data ”000111010000”
Data ”011102111000”
Data ”010243401000”
Data ”011240421100”
Data ”001044420100”
Data ”001022011100”
Data ”001111110000”
Data ”000000000000”
X=6:Y=9:Z=7
```

```
Sub Hyouzi(X,Y,M())
  Cls 3
  For I=1 To 10
    For J=1 To 12
      Select Case M(J,I)
        Case 0 : Line(J*50-50,I*50-50)-(J*50,I*50),8,BF
        Case 2 : Line(J*50-50,I*50-50)-(J*50,I*50),14,BF
        Case 3 : Line(J*50-50,I*50-50)-(J*50,I*50),12,BF
        Case 4 : Circle(J*50-25,I*50-25),10,14
        Case 5 : Line(J*50-50,I*50-50)-(J*50,I*50),14,BF
        Case 6 : Line(J*50-50,I*50-50)-(J*50,I*50),12,BF
      End Select
    Next
  Next
  Circle(X*50-25,Y*50-25),23,10:Circle(X*50-25,Y*50-25),15,10,,,,F
End Sub
```

```
While(Z>0)
  Call Hyouzi(X,Y,M())
  A$=””
  While(A$=””)
    A$=Inkey$
  WEnd
  BX=X:BY=Y
  Select Case Asc(A$)
    Case 27 : End
    Case 28 : X=X+1:X2=X+1:Y2=Y
    Case 29 : X=X-1:X2=X-1:Y2=Y
    Case 30 : Y=Y-1:X2=X:Y2=Y-1
    Case 31 : Y=Y+1:X2=X:Y2=Y+1
  End Select
  If M(X,Y)=0 Then
    X=BX:Y=BY
  Else
    Select Case M(X,Y)
      Case 2 : Select Case M(X2,Y2)
                 Case 1 : M(X2,Y2)=2 : M(X,Y)=1
                 Case 4 : M(X2,Y2)=5 : M(X,Y)=1 : Z=Z-1
                 Case Else : X=BX:Y=BY
               End Select
      Case 3 : Select Case M(X2,Y2)
                 Case 1 : M(X2,Y2)=3 : M(X,Y)=1
                 Case 4 : M(X2,Y2)=6 : M(X,Y)=1
                 Case Else : X=BX:Y=BY
               End Select
      Case 5 : Select Case M(X2,Y2)
                 Case 1 : M(X2,Y2)=2 : M(X,Y)=4 : Z=Z+1
                 Case 4 : M(X2,Y2)=5 : M(X,Y)=4
                 Case Else : X=BX:Y=BY
               End Select
      Case 6 : Select Case M(X2,Y2)
                 Case 1 : M(X2,Y2)=3 : M(X,Y)=4
                 Case 4 : M(X2,Y2)=6 : M(X,Y)=4
                 Case Else : X=BX:Y=BY
               End Select
    End Select
  End If
WEnd
Call Hyouzi(X,Y,M())
Print ”ゲームクリアです。”
End
```



完成したら、インターネットで倉庫番のパズル面をしらべて、入力してみよう。