はじめてのBASIC

プログラマーからひとこと

BASICは不親切です。
あ、えらい人に怒られるとこわいので、きちんと言いなおします。
プチコンで使える、「なんていうか古いほうのBASIC」は不親切です。

買って最初にさーてどう作るのかなーと思ったら、黒い画面とキーボードだけ出てきてあとは好きにやれとか、ビックリするんじゃないでしょうか。
もっとアイコンとかボタンとかツールとか、グラフィカルなユーザーインターフェースがないとさー、とか思うんじゃないですか。

思いますか。今どきの若者ですね。
きっとモテるんでしょうね。

わたしはちっともモテませんでした。

そんなことはどうでもいいんです。
ここまで読んで、あなたは「そういうのはいいから、さっさとプチコンをやらせなさいよ」と思ったかもしれません。
そこです。
プチコンがわざわざ不親切なBASICを使っているのにも、実はそういう理由があるのです。
この文章だって、できるだけカンタンにていねいに書こうとしたらこうなったのです。
でも、読んでるあなたにとって、そんなことはどうでもいいですよね。そういうことです。

アイコンとかボタンとかツールとか、親切になればなるほど「そういうのはいいから、さっさとやらせなさいよ」ということは増えていきます。
アイコンをタッチしてあっちにドラッグしてこっちのライブラリからぐりぐりしたりするよりも、ダダダッとキーボードを打つほうがずっと早く思った通りにいったりするものなのです。
まあ、BASICをよく知っていればの話ですけど。

そこで、これからBASICのことを色々おぼえてもらおうと思います。
簡単におぼえてもらうために、奥の深い話とかはめったにしません。
もしもあなたの知り合いにBASICやプログラムについて超くわしい人がいても、このページのことは秘密にしておいてください。笑われると恥ずかしいですから。

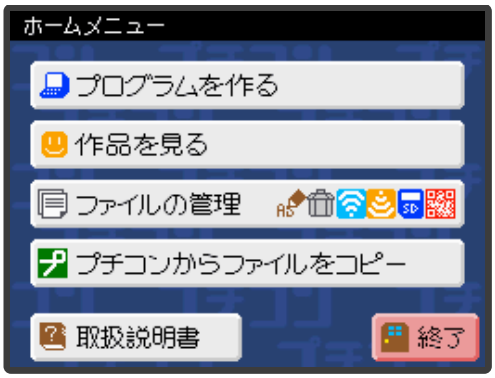
遠い遠いむかし、はるかかなたの銀河系で……

（この章ではものすごく最初の簡単なところから始めています。「さすがにそれは分かるよ！」という人のために、最後まで飛ばすボタンをおいておきます。 とばし読み ）


ではソフトを起動してみるところからはじめてみようか。


そこからかよ！　ミチノリが長そうだぜ！


まあまあ。最初の画面から「プログラムを作る」を選べばプログラムのはじまりだね。



とにかく画面は出てきたけど、ここから何をすればいいの？　READY って書いてあるけど……。「準備(READY)よし」？


テメエにもあきれたもんだゼ！　オレにはこのぐらいのこと、サッパリわからねえな！


自信たっぷりに全面降伏してきたね。まずは文字を表示してみようかな？


そこまで甘く見るんじゃねェぞ！　このキーボードから……オレの名前だ！

キー入力 (1)

ワシじゃワシじゃ、プチコンでおなじみのハカセじゃ。
どうやらワンパク君は下画面のキーボードを打つことには気がついているようじゃのう。
アルファベットを打つのはカンタンじゃと思うが、ここではよく使う「特殊(とくしゅ)キー」について説明しておくぞい。
だいたいパソコンのキーボードのようなモノじゃが、プチコンだけのテクニックもあるから、おさらい気分で読んでくれい。

スペースキー 

１文字だけ空白の「スペース」をあけるキーじゃな。

バックスペースキー 
「後ろ向きのスペースキー」というところかの。書いた字を消すのに使うのじゃ。DSiのＹボタンでも同じコトができるぞい。

アルファベットキー  、**記号キー**  、**カナキー** 
それぞれ文字のタイプを変えるキーじゃ。一番よく使うのはアルファベットじゃろうか。

シフトキー 

キーボードのスミに青色で書かれている字を書くときは、コレを押してからタッチじゃ。DSiのＬボタンかＲボタンを押しっぱなしでもシフトキーのかわりになるが、コレはなかなかベンリじゃぞい。

キャプスロックキー 
コレを押すと、シフトキーが押しっぱなしになるのじゃ。もう１回押せばカイジョされるぞい。

エンターキー 

DSiのＡボタンでもかわりになるんじゃが、このキーのことは、これからセツメイがあるじゃろう。

タッチでキーを打つのは正解だね。ここで、"ＷＡＮＰＡＫＵ"のあとに Enter キーを押してみよう。

「Enter（エンター）」は英語で「入る」っていう意味だけど……？

「コマンドを入れる」っていう言い方もあるよね。そういう「入る（Enter）」なのさ。メールっぽく言えば「送信」かな。

プチコンにオレからのアツいメッセージを送信するってワケか！　オーケー、Enterだ！

```
WANPAKU
Syntax error
OK
```

……？

（あちゃー……）

……よ、読めねェ！　シュンタ……エックス……イロロ……？

そこからなんだ。

これは「シンタックス（文法）・エラー」。おおざっぱに言いかえれば、「何を言っているのかわかりません」ということかな。

なんだと!?　オレのメッセージが伝わってねえのか？

日本語や英語があるように、プチコンにはBASIC語で言わないといけないのさ。書き直してみよう。

```
PRINT"WANPAKU"
```

「" 」はキーボードの上から２段目、左から２つ目だね。よし、Enter！

```
PRINT"WANPAKU"
WANPAKU
OK
```

ちゃんと表示されたよ！

……。

どうしたのワンパク君、ナットクしてない顔じゃないか。

イヤ……ワルかねェんだけどよォ……ジュウジツ感に欠けるっつーか……。

たしかに……まあ……エラーになる前と結果はあんまり変わらないよね……。

じゃあ、こういうのはどうかな？

```
CLS↵
```

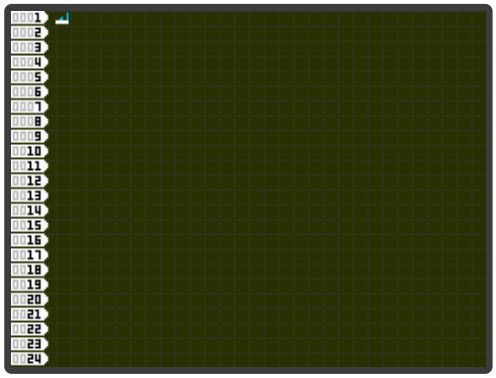
！　イッキに画面が消えたじゃねェか！

今度は操作してるって感じだろう。これはＣＬＳ命令。「シー・エル・エス」と呼ぶ人も多いけど、もともとは「クリア・スクリーン（画面を消す）」の略だよ。

た……たしかに一歩レベルアップした感がなくもないゼ！　だが……こんなチョウシで１回１回Enterって、プログラムのイメージとなんか違うっつーか……。

映画やテレビで見ると、もっと何十行もいちどに打ってるよね。

なるほど。１つの操作のたびに、いちいちＯＫを待つのはおかしいかもね。じゃあ、本格的にプログラムをしてみよう！



編集 ボタンを押すと、この編集モード画面になるんだ。左にDOCOMOからずっと番号が続いているね。この番号の順に命令をこなすようになっているんだよ。

わ……分かるような分かんねェような話だが、とにかく今までオボエたコトを全部書いてみるゼ！

キー入力 (2)

ワシじゃワシじゃ、プチコンといえばハカセと言われとるハカセじゃ。
編集モードに入ったことじゃし、さらに１歩上の特殊キーをショウカイするぞい。

カーソルキー ↑↓←→

文字を書く場所が「カーソル」じゃ。画面でチカチカしとる「▁」が見えるじゃろ？　それじゃ、それじゃ。そのカーソルを上下左右に動かすキーがカーソルキーじゃな。
もっともカーソルキーのかわりに、DSiの十字ボタンを使うのがカンタンでオススメじゃぞい。

デリートキー DEL

バックスペースキーに似ておるが、こっちはカーソルが重なっている字を消すのじゃ。使い分けるとイガイにいいものじゃぞ。

インサートキー INS

フダンは文字の上でキー入力すると、字がはさまるように書き足されるじゃろう（なんのコトかわからなければ、ジッサイにやってみい）。

そこでこのキーじゃが、押すとカーソルのカタチが変わって、インサートモードになるぞい。このモードでは文字の上でキー入力すると、そのまま上書きされるのじゃ。

タブキー TAB

４文字ぶんのスペースをいっぺんに入れるキーじゃ。何の役に立つのかわからんか？　ウーム……人によってはベンリなことがあるんじゃが……。

```
0001 CLS
0002 PRINT"WANPAKU"
```

あれ？　今度は`CLS`と打っても画面が消えないね。

編集モードでできることはプログラムを打つだけで、結果は出ないのさ。プログラムの結果を出すために、ここで ▶実行 ボタンで実行モードに戻しておこう。

```
OK
_
```

さっきの画面にモドってきたんだな。だがこれはこれでナニも変わってねェじゃねえか？

じゃあ、もうひとつ新しいことをおぼえよう。

```
RUN
```

```
WANPAKU
OK
_
```

あっ、`OK`がひとつしか出てないよ！　今までだったら画面が消えた後にもいちど`OK`が出ていたよね。

そのとおり。編集モードで１行目に`0001 CLS`、２行目に`0002 PRINT"WANPAKU"`と書いたろう。その２つを上から順に、つづけて実行したんだね。これがプログラムというものさ。

さっき使った`RUN`ってのがプログラムを動かす命令ってワケか！　ライト・ナウ！

そういえば英語で「RUN（ラン）」は「走る」って意味だけど……？

コンピューターがプログラムを上から順に走っていくようなイメージかな。たまに「ゲームを走らせる」という言い方を聞くだろう。あれはもともとこの`RUN`命令からきているんだよ。

さっきからテメーら、コマかいコトにこだわりすぎだぜ！　ヨウするに命令をこのノリでオボエてけば、BASICが使えるってこった！

今回のまとめ

実行モードと編集モード

のボタンで切り替えます。
編集モードではプログラムを打ち込み、実行モードではプログラムを実行します。

PRINT命令

```
PRINT"（好きな文字）"
```

中に書いてある文字を表示します。

CLS命令

```
CLS
```

画面から文字をぜんぶ消します。

RUN命令

```
RUN
```

プログラムを実行します。

キー入力

だいたい見た通りですが、「十字ボタンでカーソル移動」「Ｙボタンで１文字消す」「Ｌボタンでシフトキーのかわり」あたりはおぼえておくといいでしょう。

変数のはなし

プログラマーからひとこと

プログラムの世界では「変数」ということばがあります。
変数、ヘンなことばですね。「変化する数」という意味らしいですが、そんなことを言われても、むしろわかりづらいです。
このわかりづらいものを、まずはおぼえなければいけません。たいていのプログラムは変数で動いているからです。

変数をいじるのに、昔はよくＬＥＴ命令なんて使ったものです。「LET」を日本語で言うと「こうやっといてね」みたいな意味ですが、すごくアバウトですね。
そんなものをいちいち書くのもめんどくさかったのか、今はＬＥＴ命令が省略されるという、なんだかよくわからないことになっています。プチコンの説明書で「＝（LET）」という、ひとめで命令とはわからない見出しになっているのもそのせいです。ややこしいですね。

いいんです、もうなりたちとか考えずにおおざっぱにおぼえましょう。

最初におぼえる変数


というわけで、プログラムの前に変数についておぼえておこう！


なにが「というわけ」だ！　ヒントすら見えてこねェぞ！


じゃあまずは、カンタンな変数の実例を見てみようか。

```
A=10
```

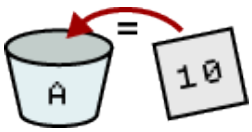

これがどういう意味かわかるかな？　ヒントは、この中では「A が変数」！


変数がなんなのかはサッパリだが、イコールでつながってんだからＡが10なんじゃねェの？　……「Ａが10」ってナンだ？


自分で言っていてわからなくなってきたかな。でもそれでほぼ正解なんだよ。


けっきょく「変数A」ってなんなのか、だよね。


では変数について説明しようか。
変数はよくバケツにたとえられるね。ここでは、A という名前のバケツがあると考えよう。
A というバケツの中に１０と書かれた紙を入れる、それがA＝１０という命令だったんだね。




ふうん、算数で使うイコールは「両方が同じ」だけど、BASICの= はちょっと意味が違うんだね。


そういうこと。「バケツの中に入れる」のがBASICの= なんだ。


で、そのバケツが何の役に立つんだ？　今んトコ、メモを入れただけのタダのゴミ箱じゃねえか！


たしかに使わないとイミがないよね。こういうプログラムにしてみよう。

```
0001 A=10
0002 PRINT A
```

あれ？　PRINTの使い方が前におぼえたのとちょっと違う……？

PRINT"A"のマチガイじゃねェのか？　RUNで実行してみようぜ！

```
10
OK
```

Aじゃなく10って表示されたよ！

２行目のPRINT　Aは、バケツAの中に入れておいた10という紙を読みあげていると考えればいいよ。
PRINT文で変数を使うときは、" で囲まないのがルールだね。

だんだんワカってきたぜ。こういうコトだな？

```
0001 A=10
0002 B=20
0003 PRINT A
0004 PRINT B
```

バケツを２コ用意してみたぜ！　ラクショーじゃねえか！

```
10
20
OK
```

いま新しいバケツの名前をBにしたけど、変数にはどんな名前でもつけられるのかな？

アルファベット16文字までなら何でもＯＫ！……と言いたいところだけど、BASICが使う命令は変数にできないんだ。たとえば変数「RUN」なんてのは作れないね。

まあ、あたりまえと言えばあたりまえかな。使えるのはアルファベットだけ？

数字も使えるけど、最初の１文字だけはアルファベットじゃないと変数には使えないね。変数の名前でありそうなのはざっと、こんな感じかな？

```
0001 A1=1
0002 B123=2
0003 ABCDEFG=3
0004 ABC123DEF=4
```

そんな豆チシキの間にも、オレはドンヨクに新しいコトをためすぜ！　バケツに入れる紙にオレの名前をキザミこんでやろう！

```
0001 A=WANPAKU
0002 PRINT A
```

前にPRINT"WANPAKU"って書いたが、これでもいいワケだろ？　RUNだ！

```
0
OK
```

？？？

なんでも応用してみるのは大事なことだよ。エライぞ、ワンパク君！　うまく動かなかったら、そのゲンインを探るのもレベルアップのヒケツさ。

思いきり上からメセンでホメやがって！　何がマチガってるってんだ！

実は変数でも、数字を入れるバケツと文字を入れるバケツは少し違うんだ。さっきのプログラムを正しく書くと、こうなるね。

```
0001 A$="WANPAKU"
0002 PRINT A$
```

```
WANPAKU
OK
```

バケツの名前の最後に$マークがついて……

紙に書く字は" でカコむってコトか。コレはPRINTでやったコトと似てるな。

$マークを使う変数は「文字変数」や「文字列型の変数」と言ったりするね。とりあえず文字を入れる変数には$マーク、とおぼえておけば大丈夫！

でもよォ、コレ変数じゃなくてもデキるコトばっかりじゃねェの？　そのままPRINT"10"とかPRINT"WANPAKU"で十分じゃねえか!?

たしかに変数が何の役に立つのかは、まだわからないよね。それじゃあ、次からは変数を使ったサンプルを見ていこう！

今回のまとめ

変数とは

名前のついたバケツに、字を書いた紙を入れる。それが変数です。
A=10と書けば、Aというバケツに１０という紙を入れることになります。
変数には「文字変数」というパターンもあります。A$="ABC"というように、名前に$をつけて「" 」で囲んだ文字を入れるのが文字変数です。

サンプルプログラム１

プログラマーからひとこと

いよいよ本格的なプログラムのはじまりです。ちょっとワクワクしますね。
ここからが長いと考えると、ちょっとウンザリしますね。

でもプログラムはおぼえたことが増えていくと、どんどん楽しくなってきます。
そのための準備運動だと思ってここはガマンしましょう。

今回用意したプログラムは、ここまでのおさらいでもありますが、ちょっと新しいこともいくつか出てきます。
どれもあとあと大事になってくることなので、ひとつひとつ身につけていきましょう。

LOAD命令

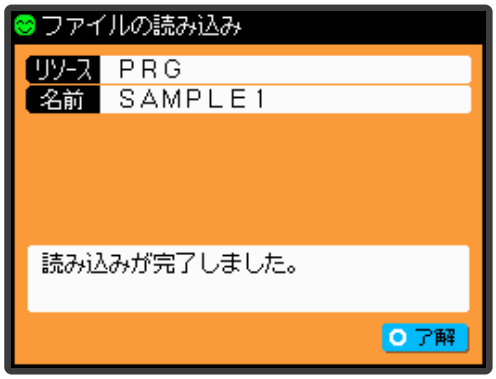
ほっほっほ。ショクン、がんばっておるかな？　ワシからのプレゼントとしてサンプルプログラムを用意したぞい。


プチコンにはサンプルプログラムが最初からいくつか入っているんだよね。


そのトオリだぜ！　どうやって見ればいいのかも知らねえけどな！


そういえば方法はまだ教えてなかったね。実行モードでこう入力してみよう！

```
LOAD"SAMPLE1"
```




ゲームのセーブデータを「ロード」するのと同じ、英語の「LOAD」なんだね。


ミミザワリな効果音がついてるじゃねえか。パンクスピリットを感じるゼ！

パンクかどうかはわからんが、昔のBASICではこんなふうに音声化したデータをロードしていたのじゃよ。この画面で流れる音はさすがにただのカザリじゃがな。


画面にいろいろ書いてあるけど、「リソース」とかは、なに？


うーん、今はとりあえず気にしないでいいかな。まずは読み込み成功なので右下の 了解 ボタンを押して先に進めよう。


オイ、OKしか出てきてねえぞ！　失敗してんじゃねェのか？

気が早いなあ。編集モードにしてごらん。

```
'
' SAMPLE1
'  コンソールヘノモジ゛シュッリョク
'
VISIBLE 1,1,0,0,0,0
CLS:COLOR 0
'
'---- タンジュンナ モジ゛ノヒョウシ゛
PRINT"☻:ニコチャン"
PRINT"ALPHABET"
PRINT"12345";"カタカナ"
'
'---- イロシテイ
COLOR 12:PRINT"♥";
COLOR 13:PRINT"♦";
COLOR 11:PRINT"♠";
COLOR 9:PRINT"♣"
COLOR 0
'
'---- ヘンスウ ヲ ツカウ
APPLE=56
ORANGE=123
'
'---- ヒョウシ゛イチヲシテイ
```

ウ、ウオォ、なんだか今まで見たコトのない世界じゃねえか。コレは逃げるしかねえ！

ＣＬＳやＰＲＩＮＴ命令は見おぼえがあるけど、それ以外にもいろいろ……

まあまあ。そうコワがるモノでもないんじゃよ。どれも知れば「なーんだ」というくらいの命令じゃ。ためしにＲＵＮしてみればよかろう。

```
☻:ニコチャン
ALPHABET
12345ｶﾀｶﾅ
♥♦♠♣


      ﾘﾝｺﾞ  = 56ｺ
      ﾐｶﾝ   = 123ｺ
      ------------
      ｺﾞｳｹｲ= 179

PUSH ❶ BUTTON
OK
```

キホン的には、文字を表示するプログラムなんだね。色つきになってるところがあるけど、どうやってるんだろう？

グムム……下の方では足し算をやってんのか？　オレは算数とか勉強みてえのはエンリョしたいゼ。

だいじょうぶだよワンパク君。計算するのはキミじゃなくプチコンだから。

ナニ？　計算もプログラムでやるってのか、聞きずてならねェな!?

じゃあ１行目から順に見ていこう！

編集モードのベンリ機能

ワシじゃワシじゃ、プチコン界のアイドルことハカセじゃ。
とうとう長めのプログラムのトウジョウじゃのう。そんな時にベンリなテクニックを教えておくぞい。

１画面スクロール

シフトキー＋カーソルキーで、１画面ぶんスクロールさせられるぞい。
つまり……DSiのＬボタンを押しながら十字ボタンで、サクサク大きくスクロールできるというわけじゃな。

１行削除

シフトキー＋バックスペースキーで、カーソルのある１行を一気に消せるのじゃ。
もちろん、DSiのＬボタンを押しながらＹボタンでもいいし、Ｌボタン＋画面のバックスペースキーという合わせ技も使えるぞい。

１行コピー＆ペースト

画面の下の方に  ボタンがあるじゃろう。この下ボタンを押すと、  ボタンが出てくるぞい。
 ボタンでその行がキオクされるから、好きなトコロで  ボタンを押せばソコにコピーした行がはりつけられるのじゃ。１行まるごとのコピペじゃな。

REM命令


プログラムのあちこちに「’」で始まる行があるのがわかるよね。


ホントだ。１～４行目、８行目、13行目……

```
0001 ’ ┌──────────┐
0002 ’ │SAMPLE1    │
0003 ’ │ ｺﾝｿｰﾙﾍﾉﾓｼﾞｼｭｯﾘｮｸ │
0004 ’ └──────────┘

0008 ’ｰｰｰ ﾀﾝｼﾞｭﾝﾅ ﾓｼﾞﾉﾋｮｳｼﾞ

0013 ’ｰｰｰ ｲﾛｼﾃｲ
```


で、ケッキョクこの「’」はナンだってんだ？


こうやって「’」と書くと、その行はプログラムで何も起きなくなるんだよ。


ハ、ハァ!?　イミねえじゃねえか！


ははは。プチコンにとっては意味がないよね。でも、ボクらがプログラムを読むときにはベンリだろう？


そうかあ……８行目から後は「**ﾀﾝｼﾞｭﾝﾅ　ﾓｼﾞﾉﾋｮｳｼﾞ**」、13行目からは「**ｲﾛｼﾃｲ**」をするって、メモしてあるんだね。


そういうこと！　この「’」はもともと**REM**と書くものだったんだ。REMとは英語のREMARK（リマーク）（コメント）の略だから、「プログラムに自分のコメントを書くための命令」ってところだね。もっとも、４文字使って「**REM　**」と書くよりもシンプルに「’」と書く方が多いかな。


「**ーーー**」と線を書いておるのも、プログラム的には意味はないが、見やすくするためのクフウじゃな。１行目のように記号を使ってちょっと見た目を良くしたり、REM命令はある意味プログラマーのこだわりポイントじゃぞ。


そこなんだがよォ、「**ｼｭｯﾘｮｸ**」って、なんだ？　「シュツリョク」じゃねえのか？

```
0003 ’ │ ｺﾝｿｰﾙﾍﾉﾓｼﾞｼｭｯﾘｮｸ │
```


やってもうたあああああああ

VISIBLE命令、コロン、COLOR命令


REM文が続いた次の５行目でまた新しい命令が出てくるね。「**VISIBLE**」……？

```
0005 VISIBLE 1,1,0,0,0,0
```


む。これはショクンにはまだ早いかの。言ってしまえば画面上に表示するもの・しないものを決めているのじゃが……今のところ文字ぐらいしか表示しとらんからのう。

じゃあナンのためにプログラムに入れてるんだ？　ムダじゃねえか！

他のプログラムを実行した後だと、その画面がまだ残っていることがあるからね。
もっともリセットすればそれも消えるし、いまボクらがやっているコトの間はあまり関係ないのさ。

今はこう書いておけば役にたつこともある、くらいにおぼえておくのがいいのかな。
じゃあ次の６行目を見てみようか。

```
00005 CLS:COLOR 0
```

CLSはもう知ってるけど、その後はなんだろう？

まず「：（コロン）」が注目だね。
「：」は２つの命令を１行に書くための「つなぎ」なんだ。だからこの行は、こう書いてあると考えていいんだよ。

```
00005 CLS
00007 COLOR 0
```

コロンを使わないと、どうしてもタテに長いプログラムになってしまうのう。ここでコロンを使うのは、リストをコンパクトにまとめるクフウじゃ。
ほかにも、区切りのいい命令まとめて１行におさめたい時なんかにも使うわい。

ひとつワカッたが、そうなると「COLOR　0」の方はナンなんだ？

COLOR（カラー）は色のことだよね。でも色をゼロにするって……？

これは「０番の色を選ぶ」と考えるといいよ。プチコンでは基本カラーとして

こんなふうに０番から15番まで色が決まってるんだ。

０番は白だから、つまり白をエラんで、……だから、どうだってんだ？

いちどえらんだ色はその後もずっとえらびっぱなしになるんだ。つまりこの行から後は、またCOLORで決めるまで、ずっと白色で字を表示することになるね。

てことは……オッ、やっとわかる行が出てきたゼ！　９～10行目で白色で文字をPRINTしてるってワケだ！

```
00009 PRINT"😊：ニコチャン"
00010 PRINT"ALPHABET"
```

この２行はカンタンじゃな。一応、記号とカナとアルファベットを全部入れてみたぞい。

前から思ってたんだけどよォ、😊って「ニコチャン」マークってほど笑ってなくねェか？

そこは気がつかんかったあああああああ

セミコロン

11行もPRINT文だけど、いつもとちょっと違うね。

```
00011 PRINT"12345";"ｶﾀｶﾅ"
```

この途中にある記号はさっきの「：」に似てるけど、別のもの？

この記号「；」は「セミコロン」と言って、ＰＲＩＮＴ命令でよく使われるんだ。
カンタンに言えば、" を閉じた後に、改行せずに文字をくっつけるんだよ。

あんまりカンタンじゃねーぞ！　もっとワカるように言いヤガレー！

ためしにこう書いてみようか。

```
0011 PRINT"12345"
0012 PRINT"ｶﾀｶﾅ"
```

こうしてＲＵＮするとどうなるかな？

```
12345
ｶﾀｶﾅ
```

ＰＲＩＮＴで書くたびに、次の行に表示されるんだね。

では、元にもどしてＲＵＮしよう。

```
0011 PRINT"12345";"ｶﾀｶﾅ"
```

```
12345ｶﾀｶﾅ
```

くっついてる！　たしかにくっついてるぜ！　……だからナンだってんだ!!　そのままＰＲＩＮＴ"１２３４５ｶﾀｶﾅ"って書けばすむコトじゃねーのか!?

まあまあ。それじゃ、；の意味がわかってきたところで、こういう書き方はどうかな？

```
0011 PRINT"12345";
0012 PRINT"ｶﾀｶﾅ"
```

```
12345ｶﾀｶﾅ
```

２つのＰＲＩＮＴでも、１つにくっついて表示されるんだね。

イヤイヤ、これも１行でＰＲＩＮＴ"１２３４５ｶﾀｶﾅ"って書けばイイってのは変わらねェぞ！

これはいちばんカンタンな例だからね。じゃあ、この先で「；」が本当に役に立つところを見てみよう。

```
0014 COLOR 12:PRINT"♥";
0015 COLOR 13:PRINT"♦";
0016 COLOR 11:PRINT"♠";
0017 COLOR 9:PRINT"✣"
```

いままで習ったことがいくつも入っているね。まずＣＯＬＯＲで色を決めて、その色で記号をＰＲＩＮＴして、「；」で……

なるほど！　そういうワケか！

気が付いたようだね。ＰＲＩＮＴ文の最後に「；」を付けておくと、次にＰＲＩＮＴするときにも、くっついたままで表示されるんだ。こんなふうにね。

```
♥♦♠✣
```

あいだにCOLOR命令が入ってもやっぱりくっつくってコトか……。
コロンといいセミコロンといい、イミはちがっても２つの行をくっつけるのは同じってトコだな。

ワンパク君、なかなかスルドイのう。
この後で出てくるが、セミコロンにはもう１つダイジな使い方があるのじゃよ。

たし算

21行と22行は、変数だね。

```
00021 APPLE=56
00022 ORANGE=123
```

今度の変数の名前はずいぶん長いけど、リンゴ(APPLE)とミカン(ORANGE)になっているね。
プログラムを実行した画面でリンゴとミカンの数を表示しているから、どうしてこういうバケツの名前にしたか、だいたいわかるんじゃないかな？

つまり変数APPLEには、リンゴの数を入れて、変数ORANGEには、ミカンの数を入れたってことだね。

スターップ！　テメエらめんどクセエぜ！　バケツにそれぞれリンゴ56コとミカン123コを入れたってことだろ、要するによォ！

ものすごくランボウにたとえたが、まさしくそういうコトじゃな。こんなふうに書くこともできるぞい。ちょっと書きかえてみい。

```
00021 APPLE=20+36
```

よ……よけいややこしくなってんじゃねェか！

ちょっと待って、コレは足し算になってるのかな？

さよう。20個のリンゴと36個のリンゴを、バケツにまとめて入れたら56個。変数APPLEには56という数字が入っておることになるな。

PRINT 20+36みたいな使い方もできるよ。これを実行モードで書いてみると……

```
PRINT 20+36
56
OK
```

おいおい、ちょっとした計算機じゃねえか！　なんかスゲエまだるっこしいけど！

今はまだ電卓を使ったほうがラクなレベルじゃが、プログラムの世界ではこの計算がなにかと役に立つんじゃよ。まあおいおいワカってくるじゃろう。

LOCATE命令

25行目にある、こりゃナンだ？

```
00025 LOCATE 5,10
```

ロ…カ……テ？

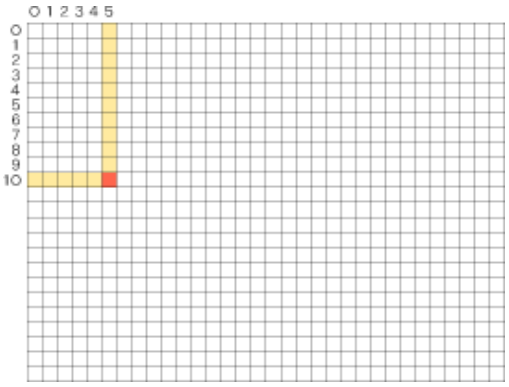
初めて出てきた命令だね。読み方は「ロケート」。英語で「場所を決める」って意味さ。

たしかにRUNすると、この先の文字が表示される場所が変わってるね。

文字をどこに出すか決める命令ってワケか。だが、しくみがサッパリだぜ！　なんだこの５とか１０とかの数字？

プチコンの上画面はヨコに32文字ぶん、タテに24文字ぶんあるのは知ってるね。マス目にするとこうかな。

そして、左上のマスをゼロとして、順に数字をふっていくとこうなるね。



つまりLOCATE　5,10は「左から６文字目、上から11文字目」に表示するということになるね。

ゼロから始まるから、６文字目をあらわす数字は「５」になるんだね。イメージした数より１少なくなるのは注意しないといけないなあ。

ワナみてえなモンじゃねェか！　なんで左上を１にしなかったんだ？

こればっかりは、「コンピューターの世界ではゼロから始まるのがお約束」としか言いようがないのう。ゼロで始まる数え方はコレ以外の命令でもちょくちょく出てくるから、よーくおぼえておいた方がいいじゃろうな。

セミコロンと変数

とにかく文字を出す場所はよくワカったぜ！　そのPRINT文はコレ、どういうこった？

```
PRINT"ﾘﾝｺﾞ　=　";APPLE;"ｺ"
```

セミコロンはもう出てきたね。

そんでAPPLEってのはたしか変数だったよな。それがセミコロンでくっつく……？

そうか！　変数APPLEには56って数字が入ってるから、これはこう書いてるのと同じなんだね。

```
PRINT"ﾘﾝｺﾞ　=　";56;"ｺ"
```

たしかにRUNしても同じように表示されてるぜ！　ナルホドな、PRINT文で文字と変数をつないで書きたいときは、セミコロンでつなぐしかねえってコトか……。

わかってきたじゃないか。そうやって26行と28行でリンゴとミカンの数を表示してるんだね。32行ではまた違った変数の使い方をしているよ。

```
00EE TOTAL=APPLE+ORANGE
00EE PRINT"ｺﾞｳｹｲ=  ";TOTAL
```

このTOTALっていうのは、やっぱり変数なのかな？

何が変数で何が命令か、初心者にはなかなか区別がむずかしいね。英字のあとにイコールが付いていたら、その英字はたいてい変数だって考えると早いかもしれないな。

APPLEもORANGEも変数……で、＋ってのは足し算か？

その通り！　変数の中に数字が入っていれば、そのまま計算式が使えるんだ。ここでTOTAL=APPLE+ORANGEと書いてあるのは、中の数字におきかえればTOTAL=56+123ということになるね。

それを33行目で表示してるんだね！

```
00EE PRINT"ｺﾞｳｹｲ=  ";TOTAL
```

それにしても……ナンのために変数を使ってんのか、オレはまだナットクいってねェぜ。サイショからゼンブ数字で書いてもいいんじゃねえか？

ホッホッホ。気持ちはワカらんでもないわい。
じゃが、たびたびリンゴやミカンの数が変わるとしたらどうじゃ？　まず上のﾘﾝｺﾞ＝の数を書きかえて、次はミカンの数、最後に合計の数を順に書きかえねばならんな。これが変数なら、APPLEとORANGEの数を書き換えるだけで、最後の合計分は自動になるのじゃぞ。

プログラムの中でリンゴやミカンの数が何度も出るほど、手間がはぶけるってことだね。

ウーム……。カンゼンにナットクしたワケじゃねーが……。

まだモヤモヤしておるようじゃな。よかろう、そこがスッキリするようなサンプルを次までに用意しておくぞい。さらば、カミングスーンじゃ！

あ、行っちゃった。

いまさらだけど、博士って何の仕事してる人なんだろうね。

END命令、SAVE命令

あのジジイ、最後のあたりはスッとばして帰りやがった。
まァ、このへんはワカるようなワカらねえような……。

```
00E5 '--- オワリ
00E6 PRINT ""
00E7 PRINT "ー"*32;
00E8 PRINT "PUSH 0 BUTTON"
```

イヤ、やっぱりワカんねえな。

ず、ズイブン早くあきらめたね。
どれもPRINTしてるから、画面に出すメッセージのサイゴを書いているのはマチガイないんじゃないかな。

そうだね。ただ今までと書き方がちょっとちがうから、フシギに思うかな？

```
PRINT ""
```

この行なんかは、画面に何も書いてないようにも見えるよね。

その通りだゼ！　"ではさんだナカミを書くハズが、中には何もねェじゃねえか！

ムダなことをしてるようにも見えるけど……あっ！　この行の終わりには；がないんだ！

わかってきたかな。終わりに；が書かれていないＰＲＩＮＴは……

次の行にいくんだったな！　ツマリ、この行は「１行だけ先にすすめる」ための行ってワケだ。

というコトは、次の行では長い線を引いているんだよね。


```
PRINT "━"＊32;
```

？？？

プログラムを動かした画面を見ればココで━の字を32コ書いてるのはわかるんだけど……＊３２って書くの？

うーん、それで合ってるんだけど、なんで＊記号なのかはまだヒミツ、にしておこうかな。

ナンだそりゃ？

この先はちょっとムズカしい話がつづくからね。そもそもこのあたりはプログラムがイキナリ終わってビックリしないように、後からつけたした部分だから、他のところよりムズカしい命令が多いんだ。

イ、インテリ君もけっこうミもフタもないコトを言うよね……。

あいだはとばすけど、せっかくだから最後の行だけはおぼえておこうか。

```
END
```

ENDエンド命令は、プログラムを終わりにするという、名前のとおりの命令だよ。

そのまんまだね……。

しかしコレ、どうせ最後の行だろ？　ワザワザそんな命令入れなくても、自動的に終わるんじゃねえか？

だよね。かといってプログラムの途中に書いても、そこから先が実行されなくなるからムダになっちゃうし……

今はまだそう思うよね。だけどけっこうダイジな命令なんだ。またすぐに出てくるから、それまではこういう命令があ

るとだけおぼえておこう！

さっきからそんなのバッカリだな。オレのナカにモヤモヤがチクセキされていくぜ！

キソをおぼえておけば、応用がラクになるのがプログラムさ。この使い方はキソのキソ。応用に入る前におぼえておく方がトクだよ。

チッ、オリコウさんぶりやがって！　しかしつねにラクな道を選ぶのがオレだ。今のうちにオボエとくぜ！

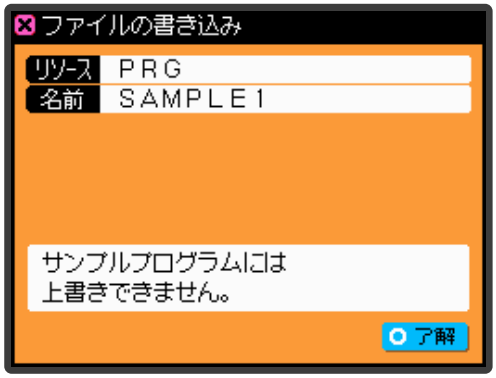
……。

今回のサンプルプログラムはほとんどいじってないけど、最後だしプログラムのセーブ方法もおぼえておこうか。

ロードがＬＯＡＤだったということは、セーブもＳＡＶＥ？

つまりそのままロードとセーブを置きかえて、こういうコトだな！

```
SAVE"SAMPLE1"
```



上書きできねェだと!?　ナットクいかねえな！

サンプルプログラムだけは、上書き禁止になってるんだ。

ヘンに改造したままセーブすると、サンプルプログラムがなくなっちゃうものね。でも書き換えたプログラムは保存できないのかな？

大丈夫、プチコンでは自由にファイル名を付けられるよ！

```
SAVE"TORIAEZU"
OK
```

とりあえずすぎるファイル名だな、オイ！

８文字以内で英数字であればなんでもいいよ。逆にこうして名前を付けたプログラムは、上書きでセーブされるから気をつけないとね。

テキトーにテストしたプログラムを大事なファイルに上書きしてタイヘンなことになったこともあるんじゃあああああ

（帰ったんじゃなかったっけ……）

実行モードのベンリ機能

ワシじゃワシじゃ、プチコン界のドンことハカセじゃ。
実行モードにもダイブなれてきたことじゃろう。ちょっとしたテクニックをさずけておこうかのう。

ログのまき戻し

カーソルキー上下で、これまでに入力した命令をさかのぼれるのじゃ。
何度も似たような命令を打ち込むなら、十字ボタンでチョイチョイとさかのぼって、書きかえてEnterするのがカンタンじゃな。
セーブやロードをそのまま実行すれば、書きかわってしまうので注意じゃぞ。

ファンクションキー

画面の上にある５つのキーがファンクションキーじゃ。

見てのとおりキーを１回打つだけで、登録されたコマンドが入力されるラクチン仕様じゃ。「RUN」はワシもよく使うのう。
ジツはこのキーのナカミも「KEY」命令で書きかえられるのじゃが……キョウミがあれば、説明書をさがしてくれい。

今回のまとめ

LOAD命令

```
LOAD"（ファイル名）"
```

中に書いてあるファイル名のプログラムをロードします。

SAVE命令

```
SAVE"（ファイル名）"
```

中に書いてあるファイル名で、プログラムをセーブします。

REM命令

```
REM　（好きなことば）
```

```
’（好きなことば）
```

自由なコメントを書きのこせます。

VISIBLE命令

```
VISIBLE 1,1,0,0,0,0
```

とりあえずこう書いておけば今はまちがいない、とだけおぼえておきましょう。

COLOR命令

```
COLOR　（番号）
```

文字の色を変更します。

LOCATE命令

```
LOCATE　（横方向）,（縦方向）
```

横０～31、縦０～23まで好きな位置に、文字の表示位置を決めます。

END命令

プログラムを終了します。

たし算

```
PRINT 1+1
```

```
A=1+1
```

```
PRINT B+C
```

```
A=B+C
```

このように数字や変数で計算したり、変数の中に入れることができます。

コロン（:）

```
（好きな命令）：（好きな命令）
```

ふたつの文を１行につないで書けます。

セミコロン（;）

```
PRINT "（好きな文字）";"（好きな文字）"
```

```
PRINT "（好きな文字）";
```

セミコロンの後には、文字がくっついて表示されます。

編集モードのベンリ機能

「Ｌボタン＋十字ボタンで１画面スクロール」「Ｌボタン＋Ｙボタン（バックスペースキー）で１行消す」「  ボタンで１行まるごとコピー・ペーストボタン表示」とおぼえておくといいでしょう。

実行モードのベンリ機能

「十字ボタン上下で、これまでの命令を読みもどし」「画面の上にある５つのキーは短縮ボタン」と、おおざっぱにおぼえておくと便利です。

ラベルとＧＯＴＯのはなし

プログラマーからひとこと

あなたがここを読んでいるということは、サンプルプログラム１をやりとげたということでしょう。
読み流しただけという人も、まあ、それはそれでいいです。

ずいぶんおぼえる事がたくさんあって、大変だったかもしれません。
なにごとも最初が一番おぼえることがあって大変なものです。
ここからはだいぶラクになると言えなくもないです。

そうは言ってもまだいくつか大事なことが残っているので、つまみ食いしながらおぼえていきましょう。
サンプルプログラム１よりはだいぶ少ないはずですよ。

ラベルとGOTO命令

```
0001 PRINT"AAA"
0002 PRINT"BBB"
```


こんなプログラムなら、もうセツメイしなくてもいいよね。


オレたちをバカにしてんのか？　ラクショーすぎてワラっちまうぜ！


ＡＡＡと表示して、次の行でＢＢＢと表示するだけのプログラムだね。

```
AAA
BBB
OK
```


それでは、これをもとに「ラベル」についておぼえようか。

```
0001 @A
0002 PRINT"AAA"
0003 @B
0004 PRINT"BBB"
```


この「@」で始まる行が「ラベル」だよ。


サッパリなんだかワカらねーが、こういう時はムヤミにＲＵＮするぜ！

```
AAA
BBB
OK
```


ん？　さっきと変わんねェぞ！　……ハハーン、ラベルって言い方といい、こいつは……


コメントに使う「'」と同じ、プログラムにイミのない記号なのかな？


おしい！　というより、今はまだコメントと同じイミしかないと言うべきかな。

その何でもワカったような言い方、ハナにつくぜ！　違いを見せてみやがれ！

１行目に、こう足してみるとどうかな？

```
0001 GOTO @B
0002 @A
0003 PRINT"AAA"
0004 @B
0005 PRINT"BBB"
```

「GOTO」……？

GOTO……ゴ・ト……テメエ、全国のゴトーさんにナニかモンクがあんのか？

そういうイミじゃないんだけど。とりあえずＲＵＮしてみようか。

```
BBB
OK
```

あれ？　「ＡＡＡ」が表示されなくなったよ？

なんてこった……ゴトーもバカにできねえな……

ええと、これはGO TO（ゴー・トゥ）と読むんだよ。「～に行け」という意味だね。

１行目のＧＯＴＯ　＠Ｂは、そのまま読めば「『ラベル＠Ｂ』に行け」ってこと……？

そうか！　それならツジツマが合うぜ！　１行目のＧＯＴＯ命令で、イキナリ「＠Ｂ」と書いてある４行目までジャンプしたわけだ。ジャンプしたから３行目のＰＲＩＮＴ"ＡＡＡ"は飛ばされる……つまり、「ＡＡＡ」が表示されることもないってワケだ！

ここまではカンペキに分かったようだね。
それじゃあ応用問題を出してみようか。「ＢＢＢ」「ＡＡＡ」の順に表示したい時は、どうすればいいかな？

考えるまでもねェぜ！　「ＢＢＢ」と表示したアトで、＠Ａに飛ばせばイイんだろ？

```
0001 GOTO @B
0002 @A
0003 PRINT"AAA"
0004 @B
0005 PRINT"BBB"
0006 GOTO @A
```

最後の行にＧＯＴＯ　＠Ａを書きたせばバッチリだぜ！

……あれ？　これだと……

これがどうしたってんだ？　ＲＵＮするぜ！

ウ、ウォオ……!?

やっぱり！　２行目まで飛んでから、また６行目で２行目に飛ぶから、いつまでたっても止まらないよ！

そう、GOTOをくり返すプログラムを組んじゃうと、無限にくり返し続けるね。こういうのを「無限ループ」というんだ。

やっちまった……こりゃリセットか……なんともノーフューチャーだぜ。

おっと待った。こういう時はあせらずSELECTボタンを押そう。

あれ、けっこうカンタンに止まるんだね。

おぼえておこうね。プログラムを途中で止める時はSELECTボタン。

SELECTボタンとSTARTボタン

ワシじゃワシじゃ、プチコンではブイブイいわせとるハカセじゃ。
インテリ君、SELECTボタンとはイキナリひとつ飛ばしで教えるではないか。そこらも合わせて、DSiの２つのボタンのセツメイじゃぞい。

途中停止

もともとプログラムを止めるのは「エスケープキー」を使うものだったのじゃ。その代わりになるのがDSiのSELECTボタンなんじゃな。そうそう、この停止ボタン[×停止]でもよいぞ。
もっとも、ワシもふだんはSELECTを使うタイプじゃな。なんかラクなんじゃよ。

簡易実行

ジツはRUN命令を使わなくても、DSiのSTARTボタンを押すとプログラムの実行になるのじゃ。編集モードからボタン１つでイキナリじゃから、こりゃラクじゃな。

さて、無限ループをしないように、PRINT"AAA" の後に１行ふやしてみようか。

```
0001 GOTO @B
```

```
0002 @A
0003 PRINT"AAA"
0004 END
0005 @B
0006 PRINT"BBB"
0007 GOTO @A
```

あっ、前におぼえたプログラムを終わらせるＥＮＤ命令だね。こういうトキにＥＮＤ命令にイミが出てくるんだね。

```
BBB
AAA
OK
```

フムム……しかしコレ、こうして見るとＧＯＴＯ使わなくてもフツーにこう書けばすむんじゃねェか？

```
0001 PRINT"BBB"
0002 PRINT"AAA"
```

その通り。あっちに行ったりこっちに行ったりするプログラムは、見る方もこんがらがるから「スパゲッティプログラム」と呼ばれて、良くないものとされているね。
じゃあ何のためにラベルとＧＯＴＯ命令があるのか？　それは次のサンプルで確かめていこう！

ケッキョクまたそのパターンかよ！

今回のまとめ

ラベル

先頭@の後に英数字で、ラベルになります。

GOTO命令

```
GOTO @（ラベル名）
```

指定したラベルまでジャンプします。

SELECTボタンとSTARTボタン

SELECTボタンでプログラム停止、STARTボタンでプログラム実行（RUN）です。

サンプルプログラム２

プログラマーからひとこと

このサンプルでは、いよいよ条件分岐について学びます。
あ、ついカッコつけてむずかしい言い方をしてしまいました。
このサンプルでは、いよいよIF～THEN文について学びます。二度と条件分岐なんて言いません。

IF～THEN文は、もうこれさえマスターしたらBASICはほとんど乗り越えたようなものです。
ちょっと言いすぎた気もしますが、それくらい大事な命令です。

それから、ずいぶんもったいぶりましたが、変数がなんの役に立つのかもこのサンプルであきらかになります。
なかなか大事なサンプルプログラムですね。
そのわりにまったく役に立つけはいのないプログラムですが。

役に立つように見えないプログラムでも、そのリストから学べることは意外にあるものです。
この講座もまた、そういうものです。
たぶん。

計算記号

やあショクン、いかしたサンプルプログラムを持ってきたぞい。

上の方では役にたたねえプログラムとか書いてあるけどな、まあやってみるぜ！

```
LOAD"SAMPLE2"
RUN
```

これを読んでいるPCなどの前のみんなも、サンプルをＲＵＮしてためしてみよう！

```
ｹｲｻﾝCOMPUTER(ﾆｺﾁｬﾝ☻)
☻:ｺﾝﾆﾁﾊ ﾆｺﾁｬﾝﾃﾞｽ
☻:ﾎﾟｸﾉｼｯﾓﾝﾆ ｺﾀｴﾃﾈ

☻:1ﾂﾒﾉ ｽｳｼﾞﾊ?
12
☻:2ﾂﾒﾉ ｽｳｼﾞﾊ?
56
☻:(1)12ﾄ(2)56ﾃﾞｽﾈ
+ → (1)ﾄ(2)ﾖﾀｽ
- → (1)カラ(2)ヲヒク
/ → (1)ヲ(2)ﾃﾞﾜﾙ
% → (1)ヲ(2)ﾃﾞﾜｯﾀｱﾏﾘ
* → (1)ﾄ(2)ヲカケル
☻:ｷｺﾟｳﾊ(+-/%*)?
+
☻:ｵｺﾀｴｼﾏｽ･･･
12+56=68

☻:1ﾂﾒﾉ ｽｳｼﾞﾊ?
```

なんてこった、ビックリするほどおもしろくねえ！

ずいぶんストレートじゃな……

カンタンに言えば、計算機のプログラムになるのかな？　数字(1)と数字(2)を入力して、それを足すか引くか……ひととおりの計算ができるんだね。

たすひくかけるわる
＋ －　×　÷のいわゆる「四則演算」と、割り算の「あまり」を出すことができるね。

これならケータイに入ってる電卓でいいんじゃねェかってのはおいといて……

ワンパク君、ワシはそんなつめたいキミを見たくはなかったぞ……

「＋」と「－」以外の計算に使う記号がヘンなんだよな。なんで「×」の計算するのに記号が「＊」なんだ？

アスタリスク
プログラムの世界では、たいてい×記号のかわりに　＊　記号を使うね。

何十年も前のコンピュータには×記号がなかったので、しかたなく＊記号で代用したと言われておるな。今でもそのままというのはヘンな気もするが、まあタンジュンな言いかえじゃよ。

じゃあ「÷」の計算に「／」を使ってるのは？

スラッシュ
「　／　」だね。まず割り算が、分数で表わせるのは知ってるだろう。

へ？

4÷2は、分数で書くと $\frac{4}{2}$。これをヨコに倒すように書くと……

なるほど、「4／2」になるわけだ！

お、オウ……。

ワンパク君……それは小学校中学年でもわかるはずじゃぞ。

オ、オレをそんな目で見るんじゃねェー！　じゃあ割り算のあまりを出すのが「%」ってのはなんなんだコノヤロー！　パーセントってのはそういうモンじゃねェだろ！

むむむ。これは例によってコンピューター界のお約束としか言えんのう。なぜか「7％3」と書くと「7を3で割ったあまりを出す」ということになるのじゃよ。ワシもくわしくは知らんが、タンジュンに「%」の記号があまっていたから、なんて話もあってキョウミぶかいところじゃ。

なりたちはどうでもいいけど、BASICの計算でもこの記号は使うからおぼえておいてソンはないんじゃないかな。

インテリ君もけっこうバッサリくるのう……

省略形

サンプルプログラムの最初のほうはもうわかるね。

```
'
' ┌──────────────┐
' │SAMPLE2       │
' │ ﾓｼﾞｭｳﾘｮｸﾄ ﾌﾟﾝｷ │
' └──────────────┘
VISIBLE 1,1,0,0,0,0
CLS:COLOR 0
PRINT"┌──────────────┐"
PRINT"│ｹｲｻﾝCOMPUTER(ﾆｺﾁｬﾝ☺)│"
PRINT"└──────────────┘"
PRINT"☺:ｺﾝﾆﾁﾊ ﾆｺﾁｬﾝﾃﾞｽ"
PRINT"☺:ﾎﾞｸﾉｼｯﾓﾝﾆ ｺﾀｴﾃﾈ"
```

カンタンすぎてアクビが出るぜ！　要するに画面を消してメッセージを出してるだけだな！

```
0012 '---
0013 @LOOP
0014 PRINT
0015 COLOR 9
```

12～15行も今まで出てきた命令だね。14行目でただＰＲＩＮＴとだけ書いてあるのが気になるけど……。

これは１行あけてるのさ。ＰＲＩＮＴ命令の後には自動的に次の行にいくからね。ちょっとした省略かな。

まあＰＲＩＮＴ"" なんてイチイチ書くのもメンドくせェしな。

省略できるところはスグ省略する、というのは良くも悪くもプログラマーというもののクセじゃろうな。プログラムが見やすくなることもあれば、ナニしとるのかわからなくなることもある。ショクンには「見やすい省略」のほうをココロガケてほしいのう。

ＲＥＭが' になるのも一種の省略形だね。実はＰＲＩＮＴも「？」と書いて省略できるんだよ。

あっ、本当だ。？"ＡＡＡ" と書いただけでＡＡＡって表示されたよ！

これもなんでPRINTが「？」になるんだって、聞いてもしかたねェんだろうな……。

INPUT命令

16行目ではじめて見る命令が出てくるね。

```
0016 INPUT"☻:1ツメノ スウジ ハ";NO1
```

インプット
INPUTという名前と、プログラムを実行した結果から、なんとなく使い方はわかるんじゃないかな？

「入力する」という意味だし、キーボードから入力してもらうトキに使うんだよね。

……ＩＮＰＵＴのアトに「" 」でかこった文字を表示して、で、セミコロンもまあわかるぜ。そのアトにくる「ＮＯ１」ってナンだ？

これは変数だね。名前はなんでもよかったんだけど、１つ目の数字だからＮＯ１というところかな。ＩＮＰＵＴ命令では、こうやって最後にセミコロンでつないで変数をきめておくんだ。

ワカんねえのはソコだぜ！　ココで変数が出るのはなぜだ？

ＩＮＰＵＴ命令で使う変数には、キー入力された文字が入ることになっているよ。

この16行目でいえば、プレイヤーが打った「１つ目の数字」が変数ＮＯ１に入るんだね。

フムフム……次のＩＮＰＵＴも似たようなモンで、22～23行目でその変数をＰＲＩＮＴしてるワケだな……。

```
0022 PRINT"☻:(1)";NO1;"ト";
0023 PRINT"(2)";NO2;"ラ ヌネ"
```

さようワンパク君、変数のダイジさがワカッてきたのではないかな？

ん？　変数なんてメンドクセェから、チョクセツ数字で書きゃいいじゃねえかって話か？　そりゃコレだって数字で……オ？
……わ、わからねエぞ！　なんてこった！　どんな数字が入力されるのか、オレには読めねえ！

その通りじゃ。ジッサイにプレイヤーがキーを打つまで、プログラマーにはそのナカミはわからぬ。そこで変数というバケツを用意しておいて、とにかく中に何か数字が入っていると想像してプログラムするのじゃよ。

そういうコトだったのか！　たしかに……コイツは変数じゃねえとハナシにもならねえ……。

ＩＮＰＵＴ命令にかぎらず、プレイヤーに入力をまかせる時や、プログラムの進行によって変化する時なんかは、変数を「ナカミは予想できないけど、中に何か入っているモノ」として使うね。

これが変数……オレの中でナニか大きなムーブメントがおきた気がするゼ！　これが初期衝動ってヤツか！

それはどうか知らんが、ワカッてくれてワシもうれしいぞ。寝ないでサンプルプログラムを書いたカイがあったわい。

（本当に何の仕事をしている人なんだろう……）

IF～THEN命令

31行目も、同じようにＩＮＰＵＴ命令だね。

```
00E1 INPUT"ﾓｰﾄﾞ:ｷｺﾞｳﾊ(+-/%*)";K$
```

おぼえてたかな？　変数で数字以外の字を入れるときは、後ろに＄がついた「文字変数」を使うんだったよね。

ここで入力してもらった記号を文字変数Ｋ＄に入れるわけだね。

そのＫ＄がカンケイしてるのはわかるが……34行から先はどういうイミだ？

```
00E4 MARK=0
00E5 IF K$=="+" THEN MARK=1
00E6 IF K$=="-" THEN MARK=2
00E7 IF K$=="/" THEN MARK=3
00E8 IF K$=="%" THEN MARK=4
00E9 IF K$=="*" THEN MARK=5
00EA ON MARK GOTO @SKIP,@PLUS,@MINUS,@DIV,@MOD,
@MUL
```

34行目はいちおうフツウに変数かな。変数ＭＡＲＫに０を入れて……

35行目からは新しい命令だね。こういうのは「IF～THEN文」と呼ばれていて、日本語にすると「もしも～なら～」というところかな。

「もしもＫ＄＝＝"＋"ならＭＡＲＫ＝１」ってコトか。ＭＡＲＫ＝１はワカるぜ。フツーの変数だからな。だがＫ＄＝＝"＋"ってナンだ？　「＝＝」ってオカシイだろコンチクショー！

これまたプログラムの世界だけのヤクソクごとじゃな。まあイコールが１コの「＝」は変数のために使ってしまったので、別の記号として２コつないだ「＝＝」を使った、というのがそもそものなりたちと言われておる。

つまり、もともと「＝＝」はイコール記号だった……？

ジッサイに、ワシの若いころBASICではこういう時はＩＦ　Ｋ＄＝"＋"　ＴＨＥＮと書いたものじゃよ。「＝＝」は新しめのヒョウゲンじゃな。

イロイロ聞きずてならねェな！　＝＝が……変数に入れるイコールじゃなくて、オレたちのよく知ってる方のイコール記号だってことは、……ああヤヤコシイぜ……。
ツマリ、「Ｋ＄＝＝"＋"」てのは「Ｋ＄と"＋"は同じもの」ってイミかよ？

ミゴトにたどりついたじゃないか、ワンパク君！　「もしもＫ＄のナカミが"＋"なら、ＭＡＲＫ＝１」というのが35行目のイミなんだよ。

うまくユウドウされた気もするが、ワルい気はしねェな。

このＩＦ～ＴＨＥＮ文、プログラムではダイジじゃぞい。どのキーが押されたか、敵にタマが当たったかどうか、アイテムを持っているか。すべてＩＦ～ＴＨＥＮで調べると言ってもいいじゃろう！

イキオイで言いすぎてる気もするが、ポイントなのは本当っぽいな！　それで、このアトはどうなるんだ？

ON～GOTO命令

```
00E4 MARK=0
00E5 IF K$=="+" THEN MARK=1
00E6 IF K$=="-" THEN MARK=2
00E7 IF K$=="/" THEN MARK=3
00E8 IF K$=="%" THEN MARK=4
00E9 IF K$=="*" THEN MARK=5
00E0 ON MARK GOTO @SKIP,@PLUS,@MINUS,@DIV,@MOD,
@MUL
```

34～39行まではもうわかってるね。

変数Ｋ＄に入った文字によって、変数ＭＡＲＫの数字を変えているんだね。

ワカんねーのは40行だぜ！　またしても新しいメイレイかよ！

よく見ると、それほど新しくもないんじゃないかな？　最初のＯＮはともかく、ＭＡＲＫは変数の名前だし、ＧＯＴＯ　＠～の形はよくおぼえてるよね。

にしても「ON（オン）」って言われてもよォ。電源でもオンにすんのか？

このON～GOTO文では、「～しだいで～に行け」という意味になるかな。40行目で言いかえると、「変数ＭＡＲＫしだいで、＠ＳＫＩＰ、＠ＰＬＵＳ、～、＠ＭＵＬへ行け」となるね。

変数しだいで……。もしかして、34～39行でＭＡＲＫの数を順に並べたのと関係ある!?

そう、ON～GOTO文では変数が０のとき、１のとき、２のとき……と順にGOTOの行き先を決めているんだ。
40行目では変数ＭＡＲＫが０ならラベル＠ＳＫＩＰへ、１なら＠ＰＬＵＳへ……と順に飛ばしているんだね。IF～THEN文とも少し似てるんじゃないかな？

フーム……このタメだけに変数を０から順にふるってのはまだるっこしくて気にくわねェが……

そうだね。本当はこのプログラムでわざわざON～GOTOを使う必要はないんだ。

（イ、インテリ君！　そんな痛いトコロを……！）

でもON～GOTOにはもっとプログラムの知識がふえていくと、ベンリな使い道が出てくるからね。今のうちにおぼえておいてソンはないさ。

（ム……この男、テキかミカタか……ユダンならぬヤツじゃぞい……）

例外処理

ゴホン、ON～GOTO文でそれぞれの飛び先でやるコトはだいたいソウゾウできるじゃろう。

イチバンわかりづれェのは、ＭＡＲＫがゼロだったトキだな。

まずＭＡＲＫがゼロになるのはどういう時か？　そこを考えればわかりやすいね。

ええと、35行～39行のどれにも当たらなかった時だけ、34行で決めたゼロのままで40行のON～GOTO文にたどりつくんだね。

```
00E4 MARK=0
00E5 IF K$=="+" THEN MARK=1
00E6 IF K$=="-" THEN MARK=2
00E7 IF K$=="/" THEN MARK=3
00E8 IF K$=="%" THEN MARK=4
00E9 IF K$=="*" THEN MARK=5
00EA ON MARK GOTO @SKIP,@PLUS,@MINUS,@DIV,@MOD,
@MUL
```

それは言いかえると、Ｋ$が＋－／％＊のどれでもなかったということだね。

ん？　Ｋ$には「：ｷｺﾞ ｳﾊ(＋－／％＊)？」と聞かれて入力した字が入ってるハズだろ……なのにどれでもないってコトは……

そうか！　打ちまちがいってコトだね！　そのときだけＭＡＲＫがゼロになって、＠ＳＫＩＰまで飛ばされるんだ。

たとえばＡと打たれたら打ちまちがいだね。ホカにもＢやＣ、ＡＡ、ＢＢＢ、と「＋－／％＊」以外に打ち間違いのパターンはいくらでもあって、いちいちIF～THEN文でチェックしきれないよね。
だから34行で最初にＭＡＲＫ＝０と決めてしまって、35～39行では正しい記号が打たれたときだけＭＡＲＫが変わるように作ってあるんだね。

そしてＭＡＲＫがゼロのまま変わらなければ、正しい記号が打たれていないとハッキリ言いきれるわけだ。

そういうこと。プログラム用語では「例外処理（れいがいしょり）」というんだ。

ムズかしいコトバはいいけどよォ、そうなるとこのプログラムで最初に数字を聞いたトキの例外処理はどうなってんだ？　数字以外を入れることだってあるよな！

たとえば16行だね。ＩＮＰＵＴ文で変数が文字変数じゃないから、数字以外が入力されるとエラーが自動で出るよ。

```
?Redo from start
```

チッ、また英語かよ！　BASICの自動にマカせるってのがそもそも気にくわねェが……

ワンパク君、イガイにこだわるタイプじゃのう。まあサンプルがあんまり長くなってもナンじゃし、ここはひとつオンビンにな？

BEEP命令

```
00E2 @SKIP
```

```
00040 BEEP 4
00050 PRINT"ﾛ:シラナイキコﾞ ウテﾞ ス・・コﾞ メン"
00060 PRINT
00070 GOTO @MARK
```

そしてこれが例外処理で飛んできた@SKIPだね。

エラーメッセージをPRINTして、@MARKに飛んで入力をやり直すのはもうわかるね。43行のBEEP　4が新しい命令かな。

たしかに見たことのねェ命令だが、オレにはだいたいワカッたぜ！　コレは音を出す命令だな！

（ワンパク君が英語に反応できた!?）

ジッサイにプログラム動かして、例外処理に飛ぶよう「W」って入力してやったからな！

（そういうコトか……）

プログラムの動きから、命令のイミを見つけ出す……それもジョウタツのひとつの方法じゃな。そればかりだとこの講座がいらなくなってしまうので、ひかえめにオネガイしたいところじゃが。

BEEP(ビープ)命令は、文字どおり「ビーッという音を鳴らす」命令だね。BEEPの後の数字は、ゼロから69まであってそれぞれ違ったサウンドが鳴るようになっているよ。

パンクスピリットがヨビサマされるぜ！　４以外の番号はどんな音なんだ？

```
BEEP 0
OK
BEEP 1
OK
BEEP 2
OK
```

ワンパク君のスイッチが入ってしまったようじゃが、説明書の「10 サウンド関係の表」の中で「プリセット効果音」という表に70種類の音の名前がイチオウ書いてあるから、そちらも見てほしいのう。

ループ

あとはカンタンだね。ON～GOTOで記号どおりに飛んで、記号どおりに計算するだけさ。たとえばこんな感じにね。

```
00040 @PLUS
00050 PRINT NO1;"+";NO2;"=";NO1+NO2
00060 GOTO @LOOP
```

49行はいくつも変数と文字があって見づらいけど、たし算の結果を書いているんだね。

PRINTの中でも計算ってできるんだな。オレはてっきり

```
00040 NO3=NO1+NO2
00050 PRINT NO1;"+";NO2;"=";NO3
```

みたいに書かないとヤバいのかって思ってたぜ。

イガイに変数は自由に書けるものさ。ワンパク君の書きかたでもマチガイじゃないけど、スッキリしているのはこっちと言えるかな。

そのあとは、＠ＬＯＯＰにＧＯＴＯで戻るんだね。

そういえばこのプログラム、計算が終わったらまたやり直すんだな。コレ、無限ループになるんじゃねェか？

SELECTボタンで止めることをゼンテイに作ってあるからのう。そういうプログラムはかつてのBASICではごくフツウにあることじゃったのじゃよ。あえて終了コマンドを用意しておくのもシンシ的でいいことではあるんじゃが。

今回のまとめ

計算記号

プログラムでは「＋」「－」「＊（×）」「／（÷）」、そしてＡをＢで割ったあまりを「Ａ％Ｂ」と書いて出す「％」が使えます。

省略形

たとえば「ＰＲＩＮＴ""」と書くところは「ＰＲＩＮＴ」と書くだけですませられます。
「ＰＲＩＮＴ」も「？」で代用できます。

INPUT命令

```
ＩＮＰＵＴ　"（メッセージ）"；（変数）
```

キーボードから入力を待って、入力された字を変数に入れます。

IF～THEN命令

```
ＩＦ　（条件文。Ａ＄＝＝"Ａ"など）　ＴＨＥＮ　（命令）
```

条件文（この場合、変数Ａ＄の内容がＡであること）が正しければ、命令を実行します。

ON～GOTO命令

```
ＯＮ　（変数）　ＧＯＴＯ　＠（ラベル），＠（ラベル）……
```

変数が０のとき最初のラベルに、１のとき次のラベルに……と、変数の内容によって飛び先を切りかえます。

例外処理

意味はいろいろありますが、ここではユーザーの入力がプログラムで必要なものと違った場合に使いました。
このサンプルではIF文をすりぬけたものを全部「例外」として処理しています。

BEEP命令

```
ＢＥＥＰ　（数字）
```

０～69までのサウンドが鳴ります。

くりかえし命令と配列変数

プログラマーからひとこと

この章では、FOR～NEXT文について学びます。

FOR～NEXT文は、それがあるだけでプログラムリストを読むのがぐっと難しくなるというおそろしい命令です。
プログラムした人は意味があって使っているしその意味もわかっているのですが、そのリストを読むほうはたまったものじゃありません。

それでも使うのはなぜかといえば、とてもベンリだからです。
これがあるだけでプログラムの手間はものすごく少なくなります。
あんがい悪くないですね。プラマイでいえばプラスじゃないでしょうか。

さらにこの章では、DIM命令、いわゆる配列変数についても学びます。
配列変数というものはパッと頭で理解するのがなかなかむずかしく、プログラムをおぼえはじめた人はみんなひっかかるポイントです。
正直わたしも何度か「こんなものなけりゃいいのに」と思いました。
「なんの役にたつんだ？」とも思いました。
「なくていいじゃん」とも思いました。
でも、そういうものが意外とけっこう役にたつということは、ここまでこの講座を読んできた人なら知っているはずです。
くやしいことにこれが本当に役にたつのです。

この章では、できるだけじっくりと配列変数のことをおぼえていきます。
あんまりじっくりすぎて途中でめんどくさくなるかもしれませんが、カベをのりこえるためだと思ってじっくり読んでみてください。

FOR～NEXT命令

「くりかえし命令」と言ってどんな命令を想像するかな？

GOTOで作る無限ループなんか、「くりかえし」になるんじゃねえか？

```
0001 @MUGEN
0002 PRINT"クリカエシ"
0003 GOTO @MUGEN
```

たしかに何度もくりかえしているね。じゃあ、無限ループじゃなく10回くり返したら途中でループをやめる、とするとどうすればいいかな？

ムリじゃねえの？

ず、ずいぶんアキラメが早いね。ええと、変数を使って、ループのたびに変数の中身を変えて、IF～THEN文で変数によって別のラベルまで……

ウォオー！　話を聞いてるだけでめんどクセエうえに、言ってるイミもよくわかんねえぜ！

なかなか難しいよね。そこで役に立つのがFOR～NEXT(フォー　ネクスト)命令なんだ。
まずこのプログラムを見てごらん。

```
0001 FOR A=1 TO 10
```

```
0002 PRINT"クソカエシ"
0003 NEXT
```

ほとんどゼンブ新しい命令じゃねえか！　とりあえずRUNしてみるか。

```
OK
RUN
クソカエシ
クソカエシ
クソカエシ
クソカエシ
クソカエシ
クソカエシ
クソカエシ
クソカエシ
クソカエシ
OK
```

ちゃんとクソカエシを10回書いたところで終わったね。

３行のプログラムにしちゃナカナカやるじゃねえか。どうやらクワシク話をきかなきゃならねェようだな！

英語で「FOR（フォー）」は「〜から」、「TO（トゥ）」は「〜まで」という意味になるね。
FOR A=1 TO 10は「A=1から10まで」と読んでもいいだろう。
そして「NEXT（ネクスト）」は「次にいく」というところかな。「次のくりかえしを始める」って考えればこれはわかるよね。

NEXTの方はだいたいワカッたぜ。しかしFORの方、「A=1から10まで」ってのはイミわからねェな。フツウ「1から10まで」じゃねえのか？

あれ？　たしかBASICでイコール記号を使うのは変数のときだけだから……？

よくおぼえていたね。そう、このAは変数なんだ。

なんでココで変数が出てくんだ……？
話が見えてこねェぞ！　ワカるようにセツメイしやがれ！

ジッサイにプログラムで見てもらうのが早いかな？　ちょっと書きかえてみよう。

```
0001 FOR A=1 TO 10
0002 PRINT"クソカエシ";A
0003 NEXT
```

これをRUNするとこうなるね。

```
OK
RUN
クソカエシ1
クソカエシ2
クソカエシ3
クソカエシ4
クソカエシ5
クソカエシ6
クソカエシ7
クソカエシ8
クソカエシ9
クソカエシ10
OK
```

１から10まで、くりかえすたびに変数Aがふえている、のかな？

そう考えてマチガイじゃないね。これがＦＯＲ～ＴＯのむずかしいトコロなんだけど、「Ａ＝１から始めて、変数Ａが１０になるまで、変数Ａを１ふやしながらくりかえす」というのがＦＯＲ　Ａ＝１　ＴＯ　１０の正しいイミなんだ。

ズイブンいろいろなコトがはぶいて書いてあんだな、ＦＯＲ～ＴＯ文にはよォ！　わかりづれェ！

そのまま書いても長い命令になって、打ちづらいだろうしね。これくらいがちょうどいいんじゃないかな。

おっと待てよ、オレのギモンがカイケツしてないままだったぜ！　なんで変数がかならず出てくんだ？　はじめの方で使ったこのプログラムだけは、変数いらねえじゃねえか！

```
FOR A=1 TO 10
PRINT"クリカエシ"
NEXT
```

それはもっともかもしれないね。FOR～NEXT文では変数を使うことの方がずっと多いから……としか言えないかな。じっさい、変数を使わないループなら変数のことはムシすればいいんだし、別にモンダイはないだろう？

ウムム……タシカに。

うっかり別のところで変数Ａを使ってると、ＦＯＲ命令で変数のナカミが変わっちゃうから、それだけは気をつけないとね。

なんだかマルメこまれた気がしねェでもないが……

次はこのFOR～NEXT文と相性のいい、DIM命令について説明するよ！

配列変数

まずサイショに、「配列変数（はいれつへんすう）」について知っておこうか。

イヤなヨカンだな……。漢字が４コ以上つづいたコトバはムズカシいモンと決まってるぜ！

まあまあ。いままでにおぼえた変数とたいした違いはないんだよ。

ホントに？

使いかたがトクシュだからとまどうかもしれないけど、変数と同じように使うものだってコトさえ忘れなければ、けっこうカンタンなはずさ。
まず、いつもの変数を考えてみようか。

```
APPLE=56
PRINT"リンゴ";APPLE;"コ"
```

おぼえてるかな？　サンプルプログラム１でこんな使いかたをしたよね。

オボエてなくたって、リストを読めばイミはわかるぜ。えーと、"ＲＵＮ"！

```
リンゴ 56コ
OK
```

そうそう、変数ＡＰＰＬＥのナカミ５６をＰＲＩＮＴするんだぜ！

（ワンパク君、わすれかけてたな……）

これを配列変数に変えてみよう。

```
CLEAR:DIM APPLE(1)
APPLE(0)=56
PRINT"ﾘﾝｺﾞ";APPLE(0);"ｺ"
```

１行目からさっそく新しい命令が２つ……

ああ、そこは今は気にしないでいいトコロだから、２行目から読むようにしてね。

ン？　じゃあ変わったのは、変数のウシロに（０）がついただけってコトか？　それとも変数のウシロに（０）がつくと、何か変数がチガってくんのか？

いいや、APPLE（０）のナカミは56でマチガイないよ。

```
ﾘﾝｺﾞ56ｺ
OK
```

じゃあナマエ以外ゼンブ同じじゃねェかよ！

変数と同じだって言ったじゃないか。こういうものなんだよ。

うーん……これはワンパク君じゃなくても、フツウの変数でいいじゃないかって思うな。

よし、その気持ちをわすれずに、配列変数の使い方その２にいってみよう！

```
CLEAR:DIM APPLE(2)
APPLE(0)=56
APPLE(1)=32
PRINT"ﾘﾝｺﾞ";APPLE(0);"ｺ"
PRINT"ﾘﾝｺﾞ";APPLE(1);"ｺ"
```

やっぱり１行目はムシして読んでね！

テメー……まさか、ソレもこのフツウの変数と同じだって言うんじゃねェだろうな……

```
APPLE0=56
APPLE1=32
PRINT"ﾘﾝｺﾞ";APPLE0;"ｺ"
PRINT"ﾘﾝｺﾞ";APPLE1;"ｺ"
```

合ってるよワンパク君！　ちゃんとわかってるじゃないか。

ウオォー！　ガキあつかいもタイガイにしやがれー！　変数のナマエが違うだけじゃねえか!!

そうそう、名前くらいのちがいだって思えばいいんだよ。教えやすくてたすかるなあ。

チッ、ここまではテメーのオモワクどおりってワケか。だが、そろそろ配列変数だけができるワザを言わねえと、オレもダマっちゃいないぜ！

配列変数ならではって言うと……こういうのはどうかな？

```
CLEAR:DIM APPLE(32)
APPLE(1)=56
APPLE(2)=32
APPLE(3)=71
```

```
    :
APPLE(31)=45

PRINT"1ニチメ リンゴ";APPLE(1);"コ"
PRINT"2ニチメ リンゴ";APPLE(2);"コ"
PRINT"3ニチメ リンゴ";APPLE(3);"コ"
    :
PRINT"31ニチメ リンゴ";APPLE(31);"コ"
```

あんまり長いから途中ははぶいたけど、31日分のリンゴを数えるプログラムなのはわかるよね。

つまり……カッコの中の数字があるからわかりやすいって感じ……？

フザケんじゃねエー！　だからソレ、こう書いても同じじゃねえか！

```
APPLE1=56
APPLE2=32
APPLE3=71
    :
APPLE31=45

PRINT"1ニチメ リンゴ";APPLE1;"コ"
PRINT"2ニチメ リンゴ";APPLE2;"コ"
PRINT"3ニチメ リンゴ";APPLE3;"コ"
    :
PRINT"31ニチメ リンゴ";APPLE31;"コ"
```

そうだよワンパク君！　よくここまでガマンしたね！　ここまでは本当に変数といっしょなんだ。これから、配列変数だけの使い方に変えていくよ。

お、おう……？　ガマンしたオボエはあんまりねェが、とにかくのぞむトコロだぜ！

```
CLEAR:DIM APPLE(32)
APPLE(1)=56
APPLE(2)=32
APPLE(3)=71
    :
APPLE(31)=45

FOR DAY=1 TO 31
PRINT DAY;"ニチメ リンゴ";APPLE(DAY);"コ"
NEXT
```

!!

わかってきたかな。34行目からはおぼえたばかりのFOR～NEXTを使っているね。

変数DAYを1から順にふやしながら、31までくりかえすんだよね。

変数DAYが1のトキは、35行目のイミはPRINT 1;"ニチメ リンゴ";APPLE(1);"コ"ってコトになる……
だがFOR～NEXTでくりかえしてDAYが2になったら、同じ35行目でもPRINT 2;"ニチメ リンゴ";APPLE(2);"コ"に……！
これが31まで自動でくりかえされるってワケか！

これは配列変数じゃないとできないだろう。こういう使い方もできるね。

```
INPUT"ナンニチメヲ シラベマスカ";DAY
```

```
00035 PRINT DAY;"ニチメ リンゴ";APPLE(DAY);"コ"
```

INPUTで数字を変数DAYに入れて、34行目で配列変数APPLE(DAY)をPRINTするわけだね。
うーん、これを普通の変数でやろうとすると、変数名を入力させて……いやいや配列変数じゃないとダメか……

みんなもわかったみたいだね。つまり配列変数は「変数名に変数を使える」ベンリな変数なんだ。

ヘンスウメイをヘンスウにツカえるヘンスウ……

ごめんごめん、「変数」というコトバばっかりでわかりづらかったね。こういうふうに、口で説明するのがムズカシいのが配列変数なんだ。だから今回はできるだけジッセン的に説明したよ。

ヘッ、なかなかヤルじゃねえかクソッタレー！

それ、ほめてるの？

DIM命令、CLEAR命令

配列変数についてわかってきたところで、DIM命令もおぼえておこう。

配列変数を使うときはいつも１行目に書いてあったね。

```
00001 CLEAR:DIM APPLE(32)
```

DIM APPLE(32)というのは、これから配列変数をAPPLE(0)～APPLE(31)まで用意する、という命令だよ。

いちいち前もって数を決めておかないといけないの？

そう。このことを「配 列 宣 言（はいれつせんげん）」と言うんだけど、かならず配列宣言するのはちょっとめんどうではあるね。

メンドウなコトなんざやってられるか！　イキナリ書けばいいじゃねえか！

```
00001 APPLE(31)=1
```

```
Subscript out of range
(1)
OK
```

というエラーが出るんだよね。

のヤロー！　こうすりゃいいんだろ！　負けたキブンだぜ！

```
00001 DIM APPLE(32)
00002 APPLE(31)=1
```

```
OK
```

使うのはAPPLE(31)までなのに、DIM APPLE(32)と書かなきゃいけないのがフシギなんだけど……。

そのDIMだと、用意されるのはAPPLE(0)～APPLE(31)だからね。数を数えてみるとわかるよ。

用意される配列変数の数	1個	2個	3個	……	30個	31個	32個
必要な配列宣言	DIM (1)	DIM (2)	DIM (3)	……	DIM (30)	DIM (31)	DIM (32)
使える最大の配列変数	APPLE (0)	APPLE (1)	APPLE (2)	……	APPLE (29)	APPLE (30)	APPLE (31)

……タシカに、32コあるな。しかし言われねえとピンとこねえハナシだぜ！

LOCATE命令でもそうだったけど、ゼロから数えるから、頭で考える数より１つズレちゃうんだね。これは気をつけないといけないかな。

いっそありえないくらい大きく宣言したら？

ところが配列のこの数字は、「合計で」262144個までと決まってるんだ。たとえばDIM APPLE(262144)と宣言したあとだと、DIM ORANGE(1)くらいでもエラーになってしまうんだよ。

ヤッカイだな、まったく！　まあホドホドに使うくらいならダイジョウブなんだろ？

```
0001 DIM APPLE(300)
0002 APPLE(31)=1
```

```
RUN
Duplicate definition (1,
 DIM)
OK
```

!?　な、ナンだ？　３００個テイドでエラーは出ねェハズだぞ！

そう感じるのもモットモだね。でもこれは配列変数の「二重定義」になっちゃうんだ。

？？？

たとえば、こう書くのはよくないって、なんとなく思うんじゃないかな？

```
0001 DIM APPLE(32)
0002 APPLE(31)=1
0003 DIM APPLE(300)
```

まあ……いちどナカミを1に決めたAPPLE(31)が３行目でどうなっちゃうのか、不安になるね。

だから、プチコンでは同じ配列変数の名前で２度DIMするとエラーになるようになってるんだ。
そこでさっきのワンパク君だけど……

オレは最初に……

```
0001 DIM APPLE(32)
```

って書いてRUNして、次に

```
0001 DIM APPLE(300)
```

にしてRUNしたが……、まさか、ソレか？

プログラムが終わっても変数の情報は消えないから、それでもエラーになっちゃうんだ。

そんなコト言われてもどうすりゃいいってんだ！　これハマリじゃねえか！

だいじょうぶ。変数の情報が残ってるのがモンダイなんだから、これをリセットすればいいんだ。

```
0001 CLEAR
0002 DIM APPLE(300)
```

インテリ君がかならず１行目に書いていた、CLEAR(クリア)命令だね！

日本語では「かたづける」と言えばいいかな。こう書くだけで、変数はぜんぶリセットされるんだ。

めんどクセェことをはぶくとだ、プログラムにＤＩＭ命令を書くトキは、かならず前にＣＬＥＡＲを書いときゃいいってことだな？

ワンパク君のまとめ方はランボウだけど、今回はボクもサンセイかな。

そうそう、ＣＬＥＡＲを使うならプログラムのはじめの方がオススメだってことも言っておかないとね。あんまり途中で使うと、ダイジな変数もいっしょにクリアされちゃうかもしれないよ。

チッ、なにかとメンドウだぜ！
しかしまあワリと今回はジュウジツしたな。デカいヤマを乗りこえたって感じだぜ！

あ、でもその前にもう１つ、おぼえておきたいコトが……

マジでか。

DATA命令・READ命令

配列変数の説明に使ったこのプログラムだけど……

```
0001 CLEAR:DIM APPLE(32)
0002 APPLE(1)=56
0003 APPLE(2)=32
0004 APPLE(3)=71
     :
0032 APPLE(31)=45
0033 FOR DAY=1 TO 31
0034 PRINT DAY;"ニチメ リンゴ";APPLE(DAY);"コ"
0035 NEXT
```

いま見直しても、２～32行目がずいぶん長いって思うよね。

でもコロンで２行や３行をまとめても、まだけっこう長いしなあ。

こういうデータをまとめたいときに役にたつのがDATA(データ)とREAD(リード)のコンビなのさ。
まずはＤＡＴＡの使い方を先に見てみよう！

```
0002 DATA 56,32,71,40,64,73,61,10,45,80
0003 DATA 25,71,63,54,98,23,14,66,58,47
0004 DATA 94,36,31,26,45,71,24,76,66,58,46
```

ホントにデータって感じだね。

やってるコトは「，」で区切りながらリンゴの数を順に並べてるだけ……みたいだな。

リンゴの数の種類は全部で31。４行目だけ１つ飛び出してるけど、これはいいの？

ＤＡＴＡ文はかなり自由だから、１行に何個データがあってもいいんだ。２行目では10個、３行目も10個、４行目では11個。プログラムしていて見やすければそれが一番だと思うよ。

だがイマんとこ、ただ数をズラッと書いてるだけじゃねえか！　ＤＡＴＡの使いミチにはほど遠いゼ！

そこがポイントだね。ＤＡＴＡ命令はそのままだと何のイミもないんだ。そこでＲＥＡＤ命令の出番になるよ。

READ（リード）、日本語で言えば「読む」だね。

いちばんカンタンなＲＥＡＤ命令の使いかたはこうかな。

```
00005 READ APPLE
```

さっきのＤＡＴＡ文の後にこう書くと、最初のデータ５６が変数ＡＰＰＬＥに入るよ！

31コあるデータなのに、入る数字は最初の１つだけかよ。どういうこった！

いくつもデータを読み込むなら、ＲＥＡＤのあとに変数を並べて書くね。今回使うのは配列変数だから……

```
00005 READ APPLE(1),APPLE(2),APPLE(3)
00006 READ APPLE(4),APPLE(5),APPLE(6)
      :
```

こんなふうに10行ほど続けると全部のデータが読み込めるよ。

ゼンゼン長ったらしいゼ、まだまだ使えたもんじゃねェな！

あれ、待ってよ。配列変数っていうことは……？

そう、気がついたようだね。今のＲＥＡＤ文を短くまとめると、こうなるんだ。

```
00005 FOR DAY=1 TO 31
00006 READ APPLE(DAY)
00007 NEXT
```

ココでもまたFOR～NEXTか……。オボエたての命令だが、ずいぶんアチコチで使うんだな。

特に配列変数やＲＥＡＤにはベンリな命令だからね。31行あった文が６行におさまったのはFOR～NEXT文のおかげだね。

いままでのプログラムリストを全部書いてみると、こうなるね。

```
00001 CLEAR:DIM APPLE(32)
00002 DATA 56,32,71,40,64,73,61,10,45,80
00003 DATA 25,71,63,54,98,23,14,66,58,47
00004 DATA 94,36,31,26,45,71,24,76,66,58,46
00005 FOR DAY=1 TO 31
00006 READ APPLE(DAY)
00007 NEXT
00008 FOR DAY=1 TO 31
00009 PRINT DAY;"ﾆﾁﾒ ﾘﾝｺﾞ";APPLE(DAY);"ｺ"
00010 NEXT
```

ウオォ、いつのまにか10行まで短くなってたのか！

5〜10行目はかぶってるところがあるから、まとめちゃおう。

```
CLEAR:DIM APPLE(32)
FOR DAY=1 TO 31
READ APPLE(DAY)
PRINT DAY;"ニチメ リンゴ";APPLE(DAY);"コ"
NEXT
DATA 56,32,71,40,64,73,61,10,45,80
DATA 25,71,63,54,98,23,14,66,58,47
DATA 94,36,31,26,45,71,24,76,66,58,46
```

こんなふうにDATAはプログラムリストのどこに置いてもいいんだよ。READすると自動的にリストの中で最初にあるデータをさがして順番に読みこむからね。

オイオイ、8行におさまっちまったぞ！

やっていることは同じでも、はじめのころの63行あったプログラムとくらべると、かなりコンパクトになったろう。

あんまり短くしすぎると書いてる自分もこんがらがるので考えものじゃが、目で追えるていどならセイリしてまとめるのは良いことじゃな。こういうのもプログラムのひとつのダイゴミじゃ。

あ、ハカセ。いつから……？

ジジイの言うこともワカる気がするぜ。ケッキョク配列変数もFOR〜NEXTもナシで書いた方がワカりやすさじゃイチバンなんじゃねえか？ とか思ったモンだが……

今までの話を全部だいなしにしそうな発言だね。

しかし、BASICの命令をクシして、めんどクセェことをゼンブはぶいたのはロックだぜ！　オレのパンクスピリットにもウッタエるモノがあったな！

ワカッてくれたようでうれしいのう。そんなパンクキッズのキミのために、次のサンプルプログラムはサウンドをフィーチャーしてみたぞい。

やってやろうじゃねェかコノヤロー！　さっさと見せやがれェ！

（なんだろうこのテンカイ……）

今回のまとめ

FOR〜NEXT命令

`FOR （変数）=（最初の数） TO （終わる数）`

`NEXT`

最初の数から始めて、変数を1ずつふやしながら、終わる数までFOR〜NEXTまでをくり返します。

配列変数

`A（（数字））`という形の変数が配列変数です。たとえばA(1)とA(2)はまったく別のものとして、違う数が中に入ります。

DIM命令

`DIM （変数）（（数））`

配列変数に使う数を決める「配列宣言」をします。たとえばDIM A(10)ならA(0)～A(9)まで10個の配列変数が用意されます。

CLEAR命令

```
CLEAR
```

変数をリセットします。DIM命令の「二重定義」をしないために、プログラムのはじめのうちに使っておくのがいいでしょう。

READ命令・DATA命令

```
DATA （データ），（データ），……
```

変数に入れるデータ（数字など）を「，」でくぎって連続で書くことができます。

```
READ （変数）
```

変数にDATA文で書いたデータを読みこみます。データはプログラムの中で先の方にあるものから順に呼びだされます。

サンプルプログラム３（前）

プログラマーからひとこと

配列変数のことはもう大丈夫ですか？
もはやこの章ではあたりまえのように配列がFOR文といっしょに使われています。
一歩ひきかえすのもいいかもしれません。何度も読んだり、自分で打ってみて、ついでにちょっと書き換えたりしてみて、自分のものになることもあるものです。

心配のしすぎだったでしょうか。

前回はかなりプログラムっぽいことをやったので、今回はちょっと方向をかえてリアルタイムのキー入力や、十字ボタンの操作なんてやってみます。
「リアルタイム」というのは、まあいろいろ意味はあるんですが、いまは「押したらすぐ動く」みたいなものと思っていいんじゃないでしょうか。
けっこうゲームっぽいですね。

そういう意味ではけっこう大事なことをまなぶサンプルかもしれません。
けっこう長くなっているので、今回と次回の２つにわけて見ていきましょう。

あと、BEEP命令で音を出すヒントもちょっとついてますよ。

SYSBEEP変数

サンプルプログラム３は、まあちょっとした楽器プログラムじゃな。

```
LOAD"SAMPLE3"
RUN
```

これを読んでいるPCなどの前のみんなも、サンプルを**RUN**してためしてみよう！　前も言ったけど。
今回はプログラムリストをDSiで見ながら読むとわかりやすいよ！

ピアノ風のキーボードが１オクターブと、ドラムが４音使えるんだね。

実はもっとエキスパート向きのMMLという音楽機能もプチコンmkIIにはあるのじゃが……まあ、最初のウチはあっさり目におぼえるのがよかろう。

ツーバスやハイハットを考えてねェあたり好感がもてるぜ。だがピアノなんざ資産階級のオモチャじゃねえか！　いただけねえな！

ワンパク君はいささか音楽的にかたよったエイキョウばかり受けている気がするんじゃが、まあピアノがイヤならプログラムを改造するというコトもできるぞい。

プログラムをサイショから見ていこうか。まず気になるのはどこかな？

1～21行まではだいたいおなじみの画面表示だね。7行目が引っかかるなあ。

```
00007 SYSBEEP=FALSE
```

「=」を使ってるってことは、変数？

いいところに気がついたね。このSYSBEEPは「システム変数」という、変数の一種なんだ。システム変数はプチコンの動作にかかわるデータが入るようになっているよ。

ウンザリだぜ！　配列変数をやっとオボえたってのに、また変数のバリエーションかよ！

今はこんがらがりそうだから変数のことは考えずに、命令として丸暗記しておいた方が早いかもしれないね。
「SYSBEEP=FALSE」と書くと、プチコンのシステム音が鳴らなくなるんだよ。

オボエるのがそれだけなら、まあカンタンかな。このプログラムでキーボードを打っても、いつもの「ポツッ」ていう音が出なかったのはSYSBEEPを使ったからなんだね。

そういやBEEPはサウンドを鳴らす命令だったな。SYSBEEPの「SYS（シス）」ってのはシステムの「シス」ってコトか？

「FALSE（フォールス）」は「まちがった」って意味だと思ったけど……

プログラムの世界ではちょっと意味が変わって「偽（ぎ）」などと言うのじゃが、どう見てもイッパンテキではないのう。おおざっぱなオボエかたじゃが、アリ・ナシの「ナシ」がFALSE、とオボエると今後なにかとワカりやすいぞい。

システムのBEEPが「ナシ」、つまりシステム音を鳴らさない、ってコトか……ワカりやすいが、じゃあ鳴らすようにモドすにはどうすりゃイイんだ？

イッキにプログラムのサイゴになるが、89行で使っておるな。

```
00089 SYSBEEP=TRUE
```

「TRUE（トゥルー）」は「真（しん）」と言うが、これも「アリ」とオボエておけばよかろう。

ワカモノコトバをムリに使ってるカンジもするが、ワカりやすいことはタシカだな！

それ、ワシに言っとるのかの……？

インデント

23～43行は、配列とFOR～NEXT、READ・DATAを使ったおなじみのプログラムだね。

配列変数N$()にキーボードを左から順に入れてるのはだいたいワカったぜ。たぶん、そのウチこの変数を使うんだろ。

40行で使っている変数KCNTは何なんだろう？

```
00040 KCNT=20
00041 FOR I=0 TO KCNT-1
```

ケンバンの数、すなわち20コと考えてもらえばいいかの。「Kenban（ケンバン）」の「CouNT（カウント）」じゃからKCNTじゃ。変数の名前を決めるのはけっこうナヤむところじゃわい。

おっと待った、こういうコトにはオレはうるさいゼ！　わざわざ変数にしなくても、FOR I=0 TO 19でいいんじゃねェか!?

う、ウム。このアトにも１回しか使っておらんし、そうとも言えるのう。じゃが、コレはケンバンの数をふやす改造をするトキにベンリなように作っておるのじゃよ。

39行でDIM N$(20)と変数を使わずに配列宣言をしているから、カンペキじゃないけどね。

```
00EE KCNT=20
00F0 DIM N$(KCNT)
```

となっていればもっと良かったんじゃないかな。

インテリ君のレイセイなシテキは、ときどきワシの心をスルドクえぐるぞい……。

まァそのヘンはいいとしてもよォ、42行で１文字スペースがあいてんのはナンかイミあんのか？

```
00F1 FOR I=0 TO KCNT-1
00F2   READ N$(I)
00F3 NEXT I
```

これはプログラム用語で「インデント」というヤツじゃな。カンタンに言えば、FORやGOTOで「くり返しているナカミ」のブブンだけ、１文字スペースを空けておるのじゃ。
これはまだ１行じゃからあまりイミはないが、ナカミが何行もあったりFORの中にFORがあったりするトキには、なかなか見やすくなっててイイものじゃぞい。

「'」のコメントもそうだけど、プログラムがどう動いているのか、リストを他人が見てもすぐわかるように書くのはダイジだよね。
そのわりに次の46〜47行の変数のイミが分かりづらいけど……

```
00F6 F=4096/12
00F7 V=22
```

インテリ君は上げてから落とすのがウマいのう……。

BEEP命令のピッチ


```
00F6 F=4096/12
00F7 V=22
```


この変数のイミを教えるには、まずBEEP命令についてもっとクワしく説明がヒツヨウじゃな。
BEEP 22でピアノの音を鳴らす、ココまでは知っておるハズじゃな？

このプログラムでも使っている音色ですね。

その先はボクが説明しようか。BEEP命令にはもっとフクザツな使い方があって、ピッチを決めることもできるんだ。

ビ……？

「PITCH（ピッチ）」は「音高」とも言うんだけど、カンタンに言えば音の高さのことだね。

音の……タカ……？

ワンパク君は音楽にはくわしいと思っていただけに、さすがにその反応にはオドロキを隠せないよ。
たとえばドレミの「ミ」の音は「ド」より「高い」という言い方をするよね。ドからレ、レからミ、ファソラシド……と先に行くほどピッチが高くなる、と思えばだいたいマチガイじゃないよ。

音楽リロンを気にしてるようじゃパンクじゃねーゼ！　もっとストレートにセツメイしやがれ！

たとえば`BEEP 22,0`はピアノのドの音が鳴って、`BEEP 22,682`でレの音になると言えば実感できるかな？

「,」の後の数字がピッチになっているんだね。

ちょっとセンモン的な話になるけど、ピッチが`-8192`だと２オクターブ下、`8192`で２オクターブ上になるよ。
つまり半音は4096÷12という……

それイジョウ難しいコトをさえずるようなら、オレのベースが火をふくゼ！

ワンパク君にとってのベースギターは人をナグる武器だからなあ。

うーん、じゃあこう説明しよう。ドレミをセイカクに言うと「ド・ド＃・レ・レ＃・ミ・ファ・ファ＃・ソ・ソ＃・ラ・ラ＃・シ」だね。最初の「ド」がピッチ０だと考えて、ひとつ右に行くごとにピッチの数字が`4096/12`ずつ増えるとおぼえるといいんじゃないかな。

フーム……まったくリクツがワカった気はしねえが、そういうモンだとわりきってオボエればいいってコトだな！

（ワンパク君のロックと、ボクらが知ってるロックは少しちがうのかもしれない……）

ハナシはワカったようじゃな。そこで46行目にモドるが、`F=4096/12`というのはまさにピッチを増やすキジュンの数、`V=22`はピアノの音色の番号というワケじゃ。あとあと`BEEP`命令でこの変数を使っておるからオボエておいてくれい。

つまり変数`V`の数を変えればピアノ以外の音も出せるってワケだな。`V=20`なんかロックでいいんじゃねェか？

ここを読んでるみんなも数字を変えて`RUN`してみよう！

BTRIG()命令

プログラムリストは50行からメインループに入るね。

その「メインループ」ってのはナンだ？　メインなのはなんとなくオレにもワカるけどよォ、ループするってコトは無限ループになったりすんのか？

そうマチガっとるワケでもないぞい。じゃが、やってはならない無限ループとちがって、これはヒツヨウなループなのじゃ。

ちょっと実感がわかないけど、先を読むとわかってくるのかな。

まずドラムが、54～63行目か。

```
0054 '--- ﾄﾞﾗﾑ
0055 B=BTRIG()
0056 IF B AND 1 THEN BEEP 52
0057 IF B AND 2 THEN BEEP 53
```

```
00058 IF B AND 4 THEN BEEP 62
00059 IF B AND 8 THEN BEEP 25
00060 IF B==64 GOTO @EXIT
```

ワカらねえコトだらけで、ドコから手をつければいいのかもワカらねえぜ！

そういうことなら、まず56行目の**B=BTRIG()**がポイントだね。こう書くと、DSiの何ボタンが押されたか、変数**B**に入るようになっているんだ。

ボタンっていうと、DSiのAボタンみたいな？

DSiのボタンはひととおり全部だね。この表を見てごらん。

十字ボタン↑	1	**Aボタン**	16	**Lボタン**	256
十字ボタン↓	2	**Bボタン**	32	**Rボタン**	512
十字ボタン←	4	**Xボタン**	64	**START**	1024
十字ボタン→	8	**Yボタン**	128	**SELECT**	2048

たとえば十字ボタンの上を押したときは、変数**B**には**1**が入る。右ボタンなら**4**だね。

ナンだってこんなハンパな数なんだ？　２の次は３でいいじゃねェか！　ジュンバンに数字をふれば１～12ですむハナシだろ？

じゃあ十字ボタンの左上を押したときを考えてごらん。左は**4**、上は**1**だね。だから４＋１で、変数には**5**が入ることになるんだ。

同時押しされた時を考えて、わざと数字に間をあけてるってコト？

フツウはまずないことだけど、上と下が同時に押されれば１＋２で**3**、さらに左も押せば３＋４で**7**だね。こういうふうにどんな組み合わせで何個ボタンを押しても、同じ数字が出てこないようにできてるんだよ。もともとはビット演算という考えかたが……

おっと、ヤヤコシそうなハナシはゴメンだぜ！　とにかくシクミはワカッたぜ。タブンその後のIF文で変数**B**にあわせた音を鳴らしたりしてんだろ？

たしかにほぼそのトオリじゃ。ワンパク君のいろいろすっとばしたリカイリョクには、野生のたくましさを感じるのう。

メインループ

このままプログラムリストの先に進んでもいいけど、その前にさっきもちらっと出てきた「ループ」の考え方を知っておくほうがよさそうだね。

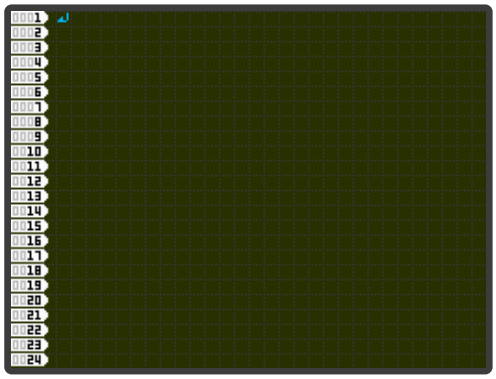
ナンだかフクザツなハナシのヨカンを感じるぜ。

いやいや、フクザツにならないための予習さ。ちょっと実行モードで**NEW**と打ってごらん。

```
NEW
OK
```

これがナンだってんだ？

あ、編集モードにすると……



プ、プログラムリストがゼンブ消えてるじゃねェか！

プログラムリストを消すのがＮＥＷ命令なのか……マチガって使ったらコワいね。

サンプルプログラムはまたＬＯＡＤしなおせばいいけど、自分のプログラムは消す前にＳＡＶＥを忘れずにね！
じゃあループの話にもどって、このプログラムを打ってみよう。プログラムのイミは分かるかな？

```
0001 INPUT"ニュウリョク";A
0002 PRINT"コタエハ";A
```

またずいぶんシンプルな……。

オレをバカにしてんのか？　ヨウするにＩＮＰＵＴで変数Ａを入れさせて、それをＰＲＩＮＴしてるだけじゃねェか！

```
RUN
ニュウリョク?
1
コタエハ1
OK
```

その通り。じゃあ、ＢＵＴＴＯＮ（）命令を使って同じことをするにはどうすればいいかな？

１行ふえちまうが、こういうコトだな？

```
0001 PRINT"ニュウリョク"
0002 A=BTRIG()
0003 PRINT"コタエハ";A
```

```
RUN
ニュウリョク
コタエハ0
OK
```

あれ？　なにもしないうちにプログラムが終わっちゃったよ。

どういうコトだチクショー！　ボタンを押してねェんだからそりゃ「コタエハ０」だろうぜ！

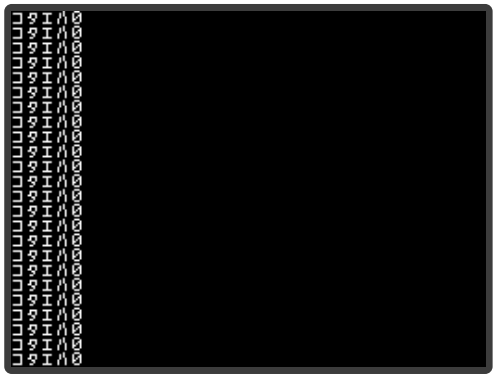
そこだよ！　ＩＮＰＵＴ命令とちがって、ＢＴＲＩＧ（）命令では「ボタンを押してない」ことも入力のウチなんだ。
だからＲＵＮしたとたんに終わっちゃうんだね。

じゃあどうすりゃボタンを押すまで待ってもらえるってんだ、ナットクできねえぞ！

何回でもＢＴＲＩＧ（）命令をやり直すってことだよね。こうすればいいのかな？

```
PRINT"ニュウリョク"
@LOOP
A=BTRIG()
PRINT"コタエハ";A
GOTO @LOOP
```

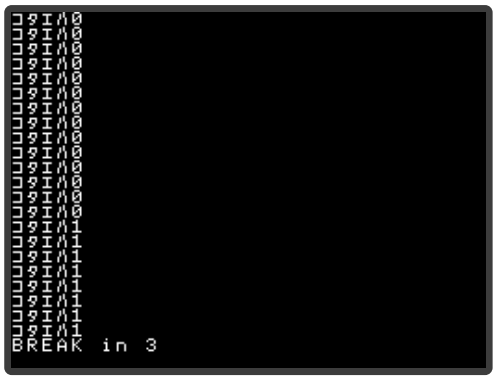
……あ、これじゃダメだ……。



やっぱり。PRINT文もループの中に入ってるからどんどん書き続けちゃう……

でもなかなか悪くないよ。ボタン入力の結果をPRINTするのはデキてるからね。

タシカに、ボタンを押したシュンカンだけは「コタエハ」の後にちゃんと数字が入ってる……ような気がする……チクショウ、見づれえんだよ！
ボタンを押すと同時にSELECTボタンで止めてやる！



……ン？

なんだかヘンだね。ボタンは押しっぱなしなのに、これだと何度もボタンを押してるような……？

そこに気が付いたか。そうBTRIG()命令なら、本当はボタンを押したその時だけ変数に数字が入る命令だったよね。

うまく動いてねえじゃねェか！

そこで、ちゃんとBTRIG()命令が動くように、もう１つ命令を入れるよ。

```
VSYNC 1
```

VSYNC……？

読みカタすらもワカらねェぜ！

タシカに読みづらいが、これは「ブイ・シンク」と読むのがイッパンテキじゃな。
もともとは「ヴァーティカル（Vertical）・シンクロナイゼーション（SYNCronization）」のことで……

よしワカッた、そのセツメイはいらねェ！

まあそんなトコじゃろうな……。
ザックリ言えば、プログラムを一時停止する命令じゃ。VSYNC 1なら60分の１秒だけ止まり、VSYNC 60ならピッタリ１秒止まるコトになっとるぞい。

つまり、この行でも60分の１秒だけプログラムを止めてるってこと……？

プログラムが止まると、BTRIG()命令がうまく動くってのが、よくワカんねェな……。

プチコンはすごいスピードでプログラムを動かしているよね。特にこんな短いループなんかは、こうしてちょっとだけスローダウンさせておかないと、BTRIG()命令が終わる前にもう１回BTRIG()命令がはじまっちゃう……ような、そんなカンジかな。

フーム……そのセツメイはわからないでもねェが、そのワリにスッキリと言いきらねえじゃねェか。

すごくおおざっぱに説明したから、プログラムにくわしい人にはツッコミどころなんだよね。まあ深く考えずに、BTRIG()命令のループにはVSYNCをセットで入れるようにしておくとうまく動くよ！

そう言われるとわかりやすいけど、話はすっとばしたなあ。

この命令はアクションゲームなんかによく使われておる。今はまだこのテイドしか使うところがないが、そのウチずいぶん助かる命令になるじゃろう。

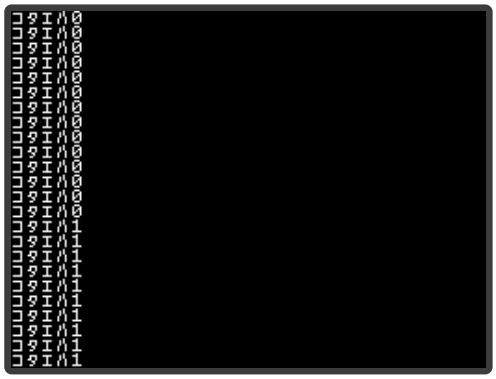
そんなものかなあ……あれ？　BTRIG()命令がうまく動くと、うまく動いてるかどうかがよく見えなくなるね……。

一瞬で通りすぎちゃうから、見づらいどころじゃないかな？　BTRIG()命令のかわりにBUTTON()を使ってみよう。

```
PRINT"ニュウリョク"
@LOOP
A=BUTTON()
PRINT"コタエハ";A
VSYNC 1
GOTO @LOOP
```

BUTTON（ボタン）って、そのまんまボタンって意味？

そのまんまさ。BTRIG()命令はボタンを押したシュンカンだけ、変数に数字が入ったよね。BUTTON()命令ではとにかくボタンが押してあれば、いつでも変数に数字が入るんだ。プログラムをRUNして見てみよう！



ナルホド。ボタンを押してる間はずっと変数Aのナカミに数字が入るってワケか。

ここまででわかったけど、このタイプの命令は何回もくり返さないと思ったように動かないね。そのためにはGOTOでループを作らないといけないんだ。

さよう。こういうリアルタイムで入力を待つプログラムは、入力を待っているとアッというまに終わってしまうので、ループを作って入力を待ち続けるのがキホンじゃな。

ハハーン、それが「メインループ」になるってワケか！

ジッサイにはこれ以外にもすぐ通りすぎちゃう命令や、くり返さないとイミがないコトもプログラムにはたくさんあって、そういうのをまとめてひとつのループにするのが「メインループ」になるね。
まあ、プログラムをおぼえていくうちに自然に身につくハズさ。

今つくったプログラムは無限ループだったけど、本当にやりたかったように作るとどうなるの？

まだ見おぼえのない書き方もまじるけど、これでいいんじゃないかな。

```
PRINT"ニュウリョク"
@LOOP
A=BUTTON()
IF A!=0 THEN PRINT"コタエハ";A:END
GOTO @LOOP
```

みんなも打ってみてね！

ウゥ……マジで見たことねェ書き方しやがって、ヨウシャのねえヤロウだぜ！

比較演算子

この行がモンダイなんだね。

```
IF A!=0 THEN PRINT"コタエハ";A:END
```

今までの話から、これが「もしも変数Aがゼロじゃなければ〜する」ってIF文だとはなんとなく分かるんだけど……

気になるのは「変数**!=**数字」のカタチだよね。

「**==**」もビックリだったけど、「**!=**」だって初めて見るよ！

ほっほっほ。ソコはそうムズカシいモノでもないぞ。いわゆる「≠（ノットイコール）」（等号否定）」と同じイミで、まあ右と左が「ちがう」というくらいのことじゃな。

まさか、またムカシのコンピューターに「≠」の字がなかったから「**!=**」と書いたとか、そんなことじゃねェだろうな……。

それでだいたい合っておるそうじゃ。

コ、コンピューターにタイするゲンソウがどんどんウシナわれていくぜェ……。

ほかにも「＜」（小なり）「＞」（大なり）を使うこともあるが、「　≦　」（小なりイコール）は「＜＝」、「　≧　」（大なりイコール）は「＞＝」と書くのじゃ。

そんなのばっかりか！　なんかもう、コンカイはオレもいろいろダキョウしてる気がするぜェ……。

そろそろ元気がなくなってきたかな。色々おぼえたし、いったん一休みしよう！

今回のまとめ

インデント

行の先頭に空白スペースを入れることを「インデント」といいます。プログラムの動作には関係ありませんが、リストを見やすくするのに使います。FOR〜NEXTの中に使うのがいちばん一般的ですね。

NEW命令

```
NEW
```

実行モードで使います。プログラムが消去されます。

SYSBEEP変数

```
SYSBEEP=FALSE
```

```
SYSBEEP=TRUE
```

`FALSE`ならシステム音を消し、`TRUE`ならシステム音が鳴るようになります。

BEEP命令のピッチ

```
BEEP　（音色），（ピッチ）
```

ピッチ`0`が基本の音、`-8192`だと２オクターブ下、`8192`で２オクターブ上になります。おおざっぱに「ドレミで半音上げるには、ピッチを`4096/12`だけ増やす」とおぼえておくといいでしょう。

BTRIG()命令

```
変数=BTRIG()
```

ボタンが押された瞬間、変数に押されたボタンの情報が入ります。

十字ボタン↑	1	**Aボタン**	16	**Lボタン**	256
十字ボタン↓	2	**Bボタン**	32	**Rボタン**	512
十字ボタン←	4	**Xボタン**	64	**START**	1024
十字ボタン→	8	**Yボタン**	128	**SELECT**	2048

たとえば左上に押せば4+1で`5`が、何も押さなければ`0`が入ります。

VSYNC命令

```
VSYNC　（数）
```

プログラムを一時停止します。`VSYNC 60`で１秒、`VSYNC 1`なら60分の１秒だけ遅くなります。

BUTTON()命令

```
変数=BUTTON()
```

BTRIG()命令とよく似ています。
ボタンが押されている間ずっと、変数に押されたボタンの情報が入ります。

比較演算子

IF～THEN命令の条件文で使います。

変数AとBを比べる場合

A==B	AとBの内容が同じ
A!=B	AとBの内容が違う
A<B	AよりBが大きい
A<=B	AよりBが大きいか、同じ
A>B	AよりBが小さい
A>=B	AよりBが小さいか、同じ

ＩＦ　（変数）！＝（数）　ＴＨＥＮ　（命令）

変数と数が一致しないときだけ命令を実行します。

サンプルプログラム３（後）

プログラマーからひとこと

サンプルプログラム３を見ていくのも後半になりました。
と言っても、前回までで難しいところはだいたいおさえたので、今回はわりとスルスルいくような気がします。

中にはかなりおおざっぱな説明も出てきますが、リクツ付けはそのうち、レベルが上がってきたころにやるので、それまでは「こーゆーもんなんだな」という感じでおぼえてください。
ないしょですが、プログラマーでもそうやっておぼえた事というのは、けっこうあります。
ちなみに私も前回ここで指摘されるまで、「フォールス」または「フォルス」と読む「False」を「ファルス」と間違って読みつづけていました。あんがいそういうものです。

ビット演算子

さて、メインループの話からダイブずれてしまったようじゃな。そろそろ`LOAD"SAMPLE3"`でサンプルプログラムにモドるとするかの。
前回は、ドラムを鳴らすこのあたりで止まっていたんじゃったな。

```
0054 '--- ドラム
0055 B=BTRIG()
0056 IF B AND 1 THEN BEEP 52
0057 IF B AND 2 THEN BEEP 53
0058 IF B AND 4 THEN BEEP 62
0059 IF B AND 8 THEN BEEP 25
0060 IF B==64 GOTO @EXIT
```

ケッキョク、ここに出てくるIF文の使い方はワカんねえまんまだぜ！

いちばん多く使われてるのは、このタイプの文だね。

```
0056 IF B AND 1 THEN BEEP 52
```

今まで習ったことのフンイキから、「上ボタンを押したら52番の音色で`BEEP`」ということはわかるけど……

「`IF B AND 1 THEN`」ってナンだよコンチクショー！　おかしいにもホドがあるじゃねえか！

これはリクツは気にせず丸ごとおぼえた方が早いかなあ。`BUTTON()`したバアイ、「`IF B AND 1`」と書くと、「Ｂの中に１がまじっている」というイミになるんだ。

「まじっている」？

とくに十字ボタンの入力なんかにベンリな使いかただね。たとえば押されたボタンが右上だったとしよう。すると変数に入る数字は１＋８で９だよね。

十字ボタン↑	1
十字ボタン↓	2
十字ボタン←	4

十字ボタン→ | 8

そうなるとコマるのは、「IF B==1」じゃヒットしないことだね。
ところが「IF B AND 1」や「IF B AND 8」と書くと、ちゃんとヒットするんだ。フシギだね。

これをセツメイするには、まず「ビット演算」についてセツメイせねばならん。ならんのじゃが、あんまりムズカしいので人に教えづらいのじゃあああああ

初心者向きのサンプルプログラムなのに、ちょくちょくフクザツなコトをしようとするよな。セツメイできねェなら、サイショからやるなよ！

まあまあ。こういうベンリな使いかたがあるんだから、ふかく考えずに使えばいいのさ。

テメーのワリキリもなんだかスゲエな……。

連続入力を止めるために

まだ61行目が残ってるね。

```
0061 VSYNC 1
```

前の章でおぼえた命令だね。

ツマリ、BTRIG()命令をマトモに動かすために……エート……60分の１秒だけプログラムを止めてるんだったか。

もっとくわしく言うと、VSYNCで止まっている間でも、BEEPの音は鳴りつづけるのがポイントだね。

ということは、VSYNCの直前で鳴らしたBEEPの音が、60分の１秒間は何があっても鳴りつづける……？

言い方はスゴそうでも、イマイチたいしたことが起きてねえぞ？

実際にためしてみるのが早いかな。ためしに61行の先頭に「'」をつけてみよう。

```
0060 'VSYNC 1
```

これでVSYNC命令は使われなくなったよね。RUNしてドラムを一度に鳴らしてごらん。

ウォオー！　鳴らしまくってやるゼェー！

みんなもワンパク君と一緒にためしてみてね！

ワンパク君のロックはデタラメに音をならせばとにかくロックだからなあ。……あれ？　なんだか今までより音が大きいような……？

そうなんだよね。VSYNCがないと、BTRIG()が「何度もボタンを押してる」ようなことになるのはおぼえてるかな？　つまり、人間に出せないようなスピードで何度もBEEPが鳴ってるってコトさ。

格闘ゲームっぽく言うと技を出した瞬間に、同じ技でキャンセルしてるってコトか。ナルホド……「ドカーン」の「ド」だけくり返し聞こえると、なんだか大きな音に感じるワケだな。

ワンパク君らしいけど、ナカナカわかりやすいたとえだね。じゃあ、さっきの「'」は消して、フツウに鳴るようにしておこう。

先頭に「'」をつけてとりあえずテストしてみる、というのはタンジュンじゃが、ダイジなテクニックじゃな。
プログラム用語では「コメントアウト」と言うのじゃが、このキカイにおぼえておくといいぞい。

INKEY$()命令

64行から、気になることが書いてあるね。

```
00064 '--- FOR-NEXT ﾃﾞ
00065 '--- ﾄﾁｭｳｶﾗﾇｹﾙﾄ
00066 '--- ﾅｲﾌﾟ ﾒﾓﾘｶﾞ ﾍﾙﾉﾃﾞ
00067 '--- GOTO ﾖｯｶｯﾀﾙｰﾌﾟ
```

う、うむ。ソコはちょっとヤッカイなモンダイなのじゃが、ひとまずその先の70行から片づけてもらおうかの。

```
00070 K$=INKEY$()
```

カタチはなんとなくＢＵＴＴＯＮ（）命令と似ているね。

そう！　ＩＮＫＥＹ＄（）命令は、ＢＵＴＴＯＮ（）命令のキーボードバージョンだと考えていいだろうね。「IN KEY（イン・キー）」つまりキー入力というコトだね。

ツマリ、70行だと変数Ｋ＄に押したキーの……ナニが入るんだ？

あ、それはＢＵＴＴＯＮ（）命令よりカンタンだね。押した文字がそのまま変数に入るんだ。キーボードの「A」を押していたら、Ｋ＄にはＡという文字が入るよ。

チッ、フカヨミしすぎたか！　しかし、おかげで71行目のやってるコトはよくワカッたぜ。

```
00071 IF K$=="" GOTO @LOOP
```

変数Ｋ＄がカラ……つまりキーが押されてなければ、おなじみの＠ＬＯＯＰに飛んでくり返しってことだな！

FOR～NEXT文からの脱出とスタック溜まり

このあたりまではＢＵＴＴＯＮ（）命令と本当によく似ているね。問題はこの後だけど……

ゴホン。ここはヒトアシさきに、カンタンに書き直したプログラムの方を見てもらおうかの。

```
00072 FOR I=0 TO KCNT-1
00073   IF K$==N$(I) GOTO @PLAY
00074 NEXT
```

これを読めば、ナニをしとる行かはワカるはずじゃの？

……たしか配列変数Ｎ＄（）には（０）～（１９）まで、ケンバンに使う字が入ってたハズだな。

変数ＫＣＮＴ－１はケンバンの数をコンピューター風に数えて、１９のはずだよね。

変数Ｋ＄に入ってるのは押されたキー。つまり……20回ブンＮ＄（）を変えながらくり返しチェックして、どこかで押されたキーがケンバンのキーと同じだったら＠ＰＬＡＹに飛ぶ、ってコトだな！

おお……ワンパク君、すっかりリッパになったのう……。

そりゃいいんだけどよォ、ジッサイのサンプルはこうじゃねェよな。またしくじったのか？

おお……ズバズバ言うトコは変わっとらんのう……。

```
I=0
@KLOOP
  IF K$==N$(I) GOTO @PLAY
  I=I+1
IF I<KCNT GOTO @KLOOP
GOTO @LOOP
```

よーく見なおしてもらえばワカると思うが、ココで書いているコトはFOR～NEXT文で書いたコトと同じなのじゃ。

75行目の「**I = I + 1**」てのが初めて見るパターンだが……

変数のルールにそって考えると、なんとなく分かるね。バケツ**I**に、それまでの**I**の中身プラス**1**を入れるってことでしょ？

そういうこと。こういうふうに変数の中では、同じ変数を使った計算もできるんだけど、なれないとピンとこないかもね。
もしも変数**I**のナカミが**2**だったときに**I = I + 1**すると、**I = 2 + 1**というわけで、**3**が変数**I**のナカミになるね。

FOR～NEXT文も、変数に１ずつプラスしてくワケだしな。
よし、このブブンがFOR～NEXTと同じコトをやってるのはワカッた。で、なんでそんなコトすんだ？　FOR～NEXTの方がカンタンじゃねェか！

ココがムズカしいトコロでのう……。
押されたキーとケンバンのデータがマッチすれば**@PLAY**に飛ぶのはワカッとるな？
じゃがFOR～NEXTのくり返しが終わらないウチにトチュウで飛び出して、またFOR～NEXTをゼロからくり返しはじめて……、とやっておると、プログラムの中に「トチュウで止まったままのFOR文」がドンドンたまっていくのじゃよ。

つまり、いわゆるスタックがたまってメモリをアッパクするんですね。

？？？

まあモノはためしじゃ、サンプルをFOR～NEXT方式に書きかえて、ジャンジャンキーボードをたたいてみい。

ウオォー！　オレのレンシャが火をふくゼェー！

```
Out of memory (72, FOR)
OK
```

ゼエ、ゼエ……た、タシカにエラーが出たな。

メッタにないコトではあるんじゃが……。とにかくFOR～NEXTをトチュウで抜け出すのはあまりコンピューターにやさしくないというコトと、もし自作のプログラムで**Out of memory**エラーが出たらコレをうたがってもいい、とはココロのスミでおぼえておいていいかもしれんのう。

BEEP命令で再生

さあ、ここまで来たらもうあとはキーボードどおりに音を鳴らすだけだね！

なんだかんだで、けっこう長かったなあ。

```
@PLAY
P=F*(I)-4096
BEEP V,P
GOTO @LOOP
```

82行で使っている変数**V**は最初のほうで出てきた**BEEP**命令の音色、変数**P**は**BEEP**命令の「ピッチ」だね。

その変数Ｐを決めてるのが81行目ってワケだな。もうずいぶんサカノボるが、変数Ｆは46行目で決めてたんだったか。

```
00046 F=4096/12
```

１音ずらす数字が「４０９６／１２」なんだったね。そして変数Ｉは０がキーボードの「'」、１がキーボードの「A」、……と対応してるから……

Ａキーを押せばＦ＊１、つまりフダンより１音上のピッチになるってコトだ！

……あれ？　－４０９６って何だろう？

```
00081 P=F*(I)-4096
```

ウム。Ｐ＝Ｆ＊（Ｉ）のままじゃと、ちょっと音がカルいかと思ってのう。－４０９６してちょうど１オクターブだけ下げておいたのじゃ。

重低音……ってホドじゃねェが、そこはワカるぜ。いっそ－４００００くらいにしたらどうなるんだ？

```
Out of range (82, BEEP)
OK
```

ウウ……ＢＥＥＰ命令のピッチは-8192～8192までがゲンカイなのじゃよ……。

「Out of range（アウト・オブ・レンジ）」というのは「ハンイからはずれた」という意味のエラーメッセージだね。

インテリ君のすばやい進行は、ときにザンコクじゃ……。

ループ脱出とEND命令

86行目からは、ラベル＠ＥＸＩＴだね。

```
00082 @EXIT
00083 SYSBEEP=TRUE
00084 END
```

＠ＥＸＩＴ……。そんなラベル、どっかで見たような……見てねェような……

ははは。ＢＵＴＴＯＮ（）命令を使ったところで、ボタン判定をしていたじゃないか。

```
00050 IF B==64 GOTO @EXIT
```

オゥ、これだ！　そういやあの時はドラムのコトばっかり考えてて、このＧＯＴＯのコトはすっかり忘れてたぜ！　オレとしたことが！

ＢＵＴＴＯＮ（）命令で６４が入ったということは、たしかＸボタンを押したってことだったよね。

さよう。このプログラムでは、Ｘボタンでプログラム終了じゃったな。と言えばあとはワカるじゃろう？

さすがにラクショーだぜ！　87行でＳＹＳＢＥＥＰ＝ＴＲＵＥして音を元にモドして、88行でＥＮＤだな！

ポイントは、Ｘボタンを押すことで、50～77行目の間をグルグル回っていたメインループからぬけ出しているコトだね。

そういうコト！　ハッキリ言えば88行目のＥＮＤはなくてもいいんだけど、ループから飛びだしてプログラムを終わらせるにはピッタリだね。

それはモチロンじゃし、この後にラベルを増やして新しいキノウを増やすときにはますますＥＮＤがダイジになるじゃろうな。

```
0088 @EXIT
0087 SYSBEEP=TRUE
0086 END
0085 @USO
0084 SYSBEEP=FALSE
```

たとえばこういうプログラムじゃと、88行目にＥＮＤがなければ89行から先まで勝手に進んでしまうわい。

フーム……。ラベルでループを使ったプログラムじゃ、ループを抜け出すのと、そこでＥＮＤするのがダイジになるワケだな……。

ワ、ワンパク君！　どうしたんじゃそのリッパなまとめぶりは……！

さすがに長びいたからよォ、このヘンでＥＮＤっぽく終わらせるのがイイ気がしたんだゼ。ノーフューチャー！

いつも通りだったね。

今回のまとめ

コメントアウト

行の先頭に「'」と書くだけで（あたりまえですが）その行がなかったことになります。プログラムの動作テストのときなどに便利です。

INKEY$()命令

```
（文字変数）＝ＩＮＫＥＹ＄（）
```

変数に押されたキーが入ります。

ビット演算子

むずかしいので、使い方だけおぼえましょう。BTRIG()命令やBUTTON()命令で使う変数なら、この形が使えます。

```
ＩＦ　（変数）　ＡＮＤ　（数）　ＴＨＥＮ　（命令）
```

変数がボタンに対応した数を含んでいるときだけ、命令を実行します。
BTRIG()命令やBUTTON()命令以外でもこれが使えると思ったら大まちがいです。

FOR～NEXT文からの脱出とスタック溜まり

くわしいことをはぶくと、FOR～NEXT文の中からGOTOで飛び出して、またFOR～NEXT文の中から……とくり返すのは、あんまりコンピューターに優しくなく、激しくやりすぎると「Ｏｕｔ　ｏｆ　ｍｅｍｏｒｙ」エラーが出ることもあります。めったにないことですが、このサンプルプログラムの方法で回避はできますよ。

RND()命令

プログラマーからひとこと

あなたがここを読んでいるということは、おそらくもうIF～THEN文を使いこなせるようになったころだと思います。

いや、ビット演算子とかは、まあ、いいんです。「IF 条件文 THEN 命令」の形をおぼえていればそれでＯＫです。
おめでとう、それをおぼえたあなたはかなりイケてます。

このIF～THEN文と、ある命令さえあれば、もうゲームがひとつできてしまうのです。
けっきょくもうひとつ命令をおぼえるのかよ！　と思うかもしれないですが、ここまで読んできたあなたにとっては、もうよゆうすぎるくらいの簡単さです。
重要な命令なのに簡単なんて、わりと世の中ちょろいもんです。
その命令は、RND()命令といいます。

でたらめな数

アール・エヌ・ディー命令……。リパブリック(Republic)・ナウ(Now)・デッド(Dead)のこと……か。

いやいやいや。

ワンパク君は英語がニガテでも、そういうボキャブラリーは持ってるんだね。でもRNDは「でたらめ」って意味の「ランダム(RaNDom)」の略だよ。

なんだと？　デタラメをオレに教えようってのか！　なかなかパンクだぜ、いいドキョウじゃねえか！

なにもかもオカシイような……。やる気になってるからいいか。

じゃあカンタンな例から説明するよ。

```
A=RND(10)
```

RND()命令は、こういうカタチで使うんだ。

左のAは変数だね。

こうやって()でかこんだ数字と命令の組み合わせで、変数にトクベツなアクションをさせる命令は「関数(かんすう)」というんだけど……ややこしいから、今は気にしなくてもいいかな。

シンセツなんだかフシンセツなんだかわかんねェな！　まあいいや、ツマリA=RND(10)と書いたら変数Aに10でRND()した数が入るんだろ？　よくワカんねェけど。

ダイナミックなリカイだけど、かなり合ってるよ。じゃあRND関数でなにが起きるのか、このプログラムで見てみよう！

```
0001 A=RND(10)
0002 PRINT A
```

```
RUN
```

```
8
OK
```

変数Aに8が入った……ということだよね。

ところが、もう1回RUNすると、どうなるかな？

```
RUN
7
OK
```

ン？　同じプログラムなのになんでAに入る数字が変わってんだ？

```
RUN
4
OK
RUN
0
OK
RUN
9
OK
```

オイオイ、RUNするタビに違うコトになってるじゃねーか！　デタラメもいいとこだぜ……。デタラメ……？
ハッ!?

もう分かってきたかな。RND()は、「でたらめな数字を出す」という関数なんだ。
いまのプログラムは()の中に10が入ってるから、0～9までの10コのうちから数字をひとつ出すんだね。

ゼロから始まるから、RND(10)と書いても一番大きい数字は9までなんだね。

またコンピューターの「ゼロから数えはじめる」クセかよ……いつもながら、ピンとこねェぜ……。
まてよ……じゃあ、ゼロはいらねー、1から10で数えたいってトキはどうすりゃいいんだ？

えぇと、A=RND(11)にすれば……あれ、ダメだ。これでもゼロは出てきちゃうか。

ちょっと頭のタイソウになるから、これを読んでる人もどうすればいいのか考えてみてね！

テメエ、その学習っぽさはハナにつくぜ！　いいからサッサと教えやがれー！

では、答を見てみようかな。こういうコトさ！

```
A=RND(10)+1
```

……ン？

ああ！　そうか！　0～9のどの数字が出ても、それに1を足したら1～10になるね！

モニタの前のみんなは気が付いていたかな？
同じように、10から100までの数字を出したいときは、RND(91)+10と書けばいいね。

ウーム……カンタンに言ってくれるが、コンピューター式の数え方はアタマがイテえぜ。
だいたい、この命令がなんのヤクに立つってんだ？　ケッキョク、数字をデタラメに言ってるだけだろ？

いちばんよく見るランダムの使い方といえば、RPGのダメージかな。もちろんデタラメな数字だけがすべてじゃないけ

ど、同じステータスのモンスターから同じ攻撃を受けても、ダメージの数はちょっとずつ違うだろう？

よく「乱数（らんすう）」って言われるやつだね。

なるほど「ラン」ダムだから、「ラン」スウってことか。

そ、それはちがうんじゃないかな……。

ほかにもアクションゲームの敵の動きにも使われたりするよ。たとえばコッソリ`RND(2)`で０～１の乱数を出してみて、０なら左に、１なら右に向かう、とかね。

言われてみれば、落ちゲーで次に落ちてくるブロックも乱数だしな……。オイオイ、コイツはかなり使える命令じゃねェのか!?

さよう。ほとんどのゲームにはどこかで乱数が使われていると言っていいじゃろうな。せっかくじゃから、次回はジッサイに乱数を使ったゲームのサンプルプログラムも見せてしんぜよう。

今回のまとめ

RND()関数

`（変数）=RND(最大数)`

ゼロから最大数マイナス１までのでたらめな数字を変数に入れます。

`A=RND(10)`で変数Ａに０～９までの数字のうちどれかが入ります。

`A=RND(10)+1`とすれば、１～10になります。

サンプルプログラム４

プログラマーからひとこと

１～３と少しずつやることをふやしてきたサンプルプログラムも、ついにゲームをプログラムするところまで来ました。

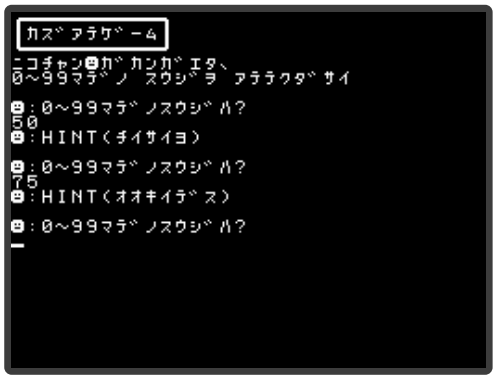
ここで出てくるゲームは「数当てゲーム」とか「ハイ・ロー・ゲーム」とか言われる、ものすごく短いゲームプログラムです。
ぶっちゃけそれほどおもしろいゲームじゃないんですが、その短さだけはほめてあげていいと思います。
私もいちばん最初に作ったのがこのゲームでした。何百人ものプログラマーたちが、最初にこのあんまりおもしろくはないけど、とりあえずプログラムが短くて簡単なゲームを作って自信をつけていったものです。

今回はいつもと違って、サンプルプログラムで新しく出てくる命令はほとんどありません。
どうやってゲームを作ればいいのか、そこを楽しんで読んでみてください。

数当てゲームを遊んでみよう

サンプルプログラム４はムカシからつたわるゲーム、数当てゲームじゃ。ためしに遊んでみるとよかろう。

```
LOAD"SAMPLE4"
RUN
```




```
ニコチャン😐ガ゛ カンカ゛ エタ、
0～99マテ゛ノ スウシ゛ヲ アテテクタ゛サイ
😐:0～99マテ゛ノスウシ゛ハ?
```

１～100じゃなく０～99ってあたり、さっきオボエたコトがなんとなくカブってくるぜ！　さては＋１するのをサボりやがったな？

う、ウム。遊んだだけでBASICのクセからプログラムをスイリできるとは、ワンパク君もなかなかプログラムがワカってきたのう。よいコトじゃぞい。

なんだかゴマかされた気もするが……とにかく数字を打つぜ！　６６なんてどうだ？

```
😐:HINT(ﾁｲｻｲﾖ)
```

アァ？　ハズれたってことか！

HINT
ヒントっていうのは、ワンパク君の６６が🅱の考えた数字よりも小さいってことだね。

正解は66より大きい数……つまり、67～99のうちどれかってコトさ。一気にしぼりこめたね。

メンドくせーコトはヌキでいくぜ！　ザックリ８０でどうだ！

```
🅱:HINT(ｵｵｷｲﾃﾞｽ)
```

ということは、正解は80よりもっと小さい……

あわせて考えれば、67～79のうちどれかだね！

となると、オレのカンが７２とつげているぜ！

……というカンジで、数字をしぼっていくのじゃな。３人のアツいプレーはちょっと飛ばすぞい。

```
🅱:ｱﾀﾘ!!!!
```

イェー！　７手目で当たりが出たぜ！

タンジュンだけど、ロンリ的と言えなくもないゲームだね。攻略法はすぐに見つかってあとは作業になるけど、ミスをへらすのはそれなりに楽しいんじゃないかな。

インテリ君、キミのコトバはあたってるだけにグサグサくるぞい……

BGMPLAY命令、BGMSTOP命令

それじゃあプログラムを見てみようか。はじめに表示されるメッセージまでは、もうプログラムをお手本にしないでも自分で書けるくらいだよね。

```
0001 '┌──────────┐
0002 '|SAMPLE4     |
0003 '|  ｶｽﾞｱﾃｹﾞｰﾑ  |
0004 '└──────────┘
0005 VISIBLE 1,1,0,0,0,0
0006 CLS:COLOR 0:BGMSTOP
0007 PRINT"┌──────────┐"
0008 PRINT"|ｶｽﾞｱﾃｹﾞｰﾑ|"
0009 PRINT"└──────────┘"
0010 PRINT"ニコチャン🅱ｶﾞ ｶﾝｶﾞｴﾀ､"
0011 PRINT"0～99ﾏﾃﾞﾉ ｽｳｼﾞｦ ｱﾃﾃｸﾀﾞｻｲ"
```

さらっと飛ばしやがったが、たしかに’(REM)文だのＰＲＩＮＴだの、いつものヤツってところだな。

あれ、でも１つだけ新しい命令があるよ。

```
0006 CLS:COLOR 0:BGMSTOP
```

このＢＧＭＳＴＯＰはまだ習ってないよね？

フム、ＣＬＳじゃのＣＯＬＯＲ　０じゃの、プログラムをまっさらにしている部分のことじゃの。こういうのはプログラムでは「初期化（しょきか）」と言うのじゃが……

ハナシがなげェぜ！　つまりコレもプログラムのジュンビってコトだな？

ま、まあそのとおりじゃ。

コレだけだとよくわからないよね。本当は**BGMPLAY**という命令とセットで使われる命令なんだ。

また新しいのが出てきやがった……ヤッカイなヨカンがするぜ。

そうムズカシくはないよ。ためしに、こう書きかえてみよう。

```
CLS:COLOR 0:BGMPLAY 1
```

あれ!? **RUN**したら音楽が鳴りはじめたよ！

BGMPLAYの「BGM」はわかるよね。あえて言えば「バックグランドミュージック（Back Ground Music）」、つまりこういう後ろでかかってる音楽のことだけど……

ガキあつかいのセツメイはゴメンだぜ！　だが、「PLAY」はサッパリわからねーな！

わからないんだ。

プレイ（PLAY）と読めばわかるんじゃないかな。オーディオプレイヤーにも「再生」ボタンにPLAYと書いてあったりするよね。

そもそもオーディオプレイヤーってコトバに「プレイ」って入ってるじゃねーか！　つまり、**BGMPLAY**はBGMを鳴らす命令ってワケか？

そういうコト！　プチコンには**BGMPLAY 0**～**BGMPLAY 29**までの30曲が入っているよ。どんな曲なのかはだいたいココに書いてあるかな。

このプログラムで数字を変えて**RUN**してもスグに曲がきけるね。

それはいいが、オイちょっと待てよ？　コレ曲が止まらねェぞ！　マズいじゃねェか！

BGMだから、ループして終わらない曲も多いんだね。

そこで使うのが**BGMSTOP**命令！　これで**BGMPLAY**した曲がスグに止まるんだ。

このサンプルプログラムで**BGMSTOP**を使っているのは、もしこれより前に**BGMPLAY**していたら止めておく、というシンセツ心じゃな。
せっかくじゃから、今回はプログラムをちょっと書きかえて、ゲーム中にBGMをかけるようにしてみようかの。

```
CLS:COLOR 0:BGMPLAY 1

BGMSTOP:END
```

ゲームのおわりと同時にBGMも止まるようにしておいたぞい。

これを**END**：**BGMSTOP**と書いちゃうと、BGMを止める前にプログラムが終わっちゃうから気をつけてね！

条件式のANDとOR

セイカイの数字を決めるのは14行目だな。

```
'--- ｾｲｶｲﾉｽｳｼﾞ ﾖｶﾝｶﾞ ｴﾙ
ANS=RND(100)
```

変数ANSに0～99までのどれかを入れる。ナニもムズカシいことはねェぜ！

そのあとはいよいよプログラムの中心部だね。

```
@RETRY
PRINT""
INPUT"ｵﾚｻﾏ:0～99ﾏﾃﾞﾉｽｳｼﾞﾊﾝ";NO
IF NO<0 OR NO>99 THEN @RETRY
IF NO==ANS THEN @BINGO
IF NO<ANS THEN @SMALL
```

ラベルがあって、1行あけて、変数NOに数字を入力させて……19行目まではカンタンだね。

だが20行目のIF文の使い方は見たことがねェぜ、また新しい書き方かよチキショー！

IF NO<0 OR NO>99 THENという使い方だね。
この行を2つに分けて考えてみようか。
IF NO<0 THEN @RETRYや、IF NO>99 THEN @RETRYと書かれればイミはわかるよね。

ただのIF文じゃねえか、ラクショーだぜ！　「もしも変数NOのナカミが0より小さければ」「もしも変数NOのナカミが99より大きければ」どっちでもラベル@RETRYに行くってコトだろ？

モンダイは間に入っていたORだね。

それがカンタンなのさ。IF NO<0 OR NO>99 THEN @RETRYと書いたら、それは「もしも変数NOのナカミが0より小さければ」「もしも変数NOのナカミが99より大きければ」どっちでもラベル@RETRYに行くってコトなんだからね。

そりゃオレの言ったこととソックリ同じじゃねェか！……ハッ!?

わかってきたかな。「OR（オア）」というのは英語で「または」ってこと。「IF ～ OR ～ THEN」は「もしも～または～ならば」というイミになるね。

20行目で言えば「もしもNO<0またはNO>99ならば@RETRYに行く」ってコトだな。

0より小さくても99より多くても、ゲームには関係ない数字だから@RETRYに帰ってやりなおすんだね。そうか、前にやった「例外処理」だね！

ウム。まあ1000と入力されてもゲームが止まるワケではないが、ルールでは0～99の数字でやりくりすることになっておるからこうしとるワケじゃな。

サンプルでは使ってないけど、「IF ～ AND ～ THEN」という使い方もあるんだよ。「AND（アンド）」はもうわかるよね。つまり……

「もしも～で、さらに～ならば」ってトコロだな！　コトバどおりだぜ、ワカりやすいじゃねェか！

どうやらバッチリのようじゃな。IF文のORとANDは使いこなせばなかなかベンリじゃぞい。
ジツはこの2つは「論理積」「論理和」と言って、もっとオクのフカい使い方もあるのじゃが……さすがにムズカシすぎるのでいまは考えんでよかろう。

残りの行はこれまでのおさらいでわかるね。

```
IF NO==ANS THEN @BINGO
IF NO<ANS THEN @SMALL
```

21行目の入力された変数NOと答の変数ANSが同じってコトは、正解したってコトだね。この先にあるラベル@BINGOに行くんだ。

22行目は入力された変数NOが答の変数ANSより小せえトキ……ま、そのマンマじゃねェの？　ラベル@SMALLに行く、と。

どちらの条件も数当てゲームのキホンじゃな。

あれ？　でも、答に当たった時と、答より小さい時……まだ「答より大きい時」のIF文がないよ!?

ほっほっほ。これも例外処理のトキにオボエたやり方じゃぞい。
答に当たった時はラベル@BINGOに行く。答より小さければラベル@SMALLに行く。……もし、変数NOが答より大きかったら、どっちのラベルにも行かないハズじゃな？

そのままプログラムの先に行くと……あっ！

さよう。その２つの条件をすり抜けたということは、変数NOは答より大きいに決まっておる。じゃから、IF文を使うまでもなく、そのまま24行の@BIGまで進んでいいのじゃよ。

３つのサブルーチン

24～27行までは、入力された数が答より大きかった時のプログラムだね。

```
0024 @BIG
0025 BEEP 4
0026 PRINT"☻:HINT(オオキイデス)"
0027 GOTO @RETRY
```

ムズカシイことは何もなさそうだね。音を鳴らして、文字を書いて、また入力のためにラベル@RETRYに戻る……。

うむうむ。さすがにここまで来ると、みんなよくワカってきたようじゃな。

29～32行目もほとんど同じだな！

```
0029 @SMALL
0030 BEEP 6
0031 PRINT"☻:HINT(チイサイヨ)"
0032 GOTO @RETRY
```

入力された数が答より小さいトキはこうするってダケだぜ！

もう気がついてると思うけど、ラベルの名前に使ってる英語は「BIG（大きい）」「SMALL（小さい）」「RETRY（やり直し）」というふうに、ナカミをあらわしてるんだ。

ラベルの名前はわかりやすいのがイチバンじゃな。
ところで29～32行はラベル@SMALLに飛んできて、ひととおり動いたらまた@RETRYに帰るワケじゃが、こういう行って帰ってくる小さなプログラムのブロックを「サブルーチン」と呼ぶのじゃぞ。豆チシキじゃ。

チッ！　ただでさえイロイロつめこんでるサイチュウなのに、いらねえチシキまでふやされちゃタマらねーぜ！

まあまあ。サブルーチンの考え方は、プログラムにはダイジなものさ。これからも出てくるセンモン用語だから、オボエておいてソンはないよ。
ルーチン（routine）は日本語で「きまりきった仕事」という意味になるけど、コンピューター用語では「ひと通りのきまった処理」ってところかな。

「処理（ショリ）」ってコトバがスデにセンモン用語っぽいが……ヨウするに「コンピューターにやらせるコト」は全部ショリか。

34行から終わりまでの４行は、どこかに帰るワケじゃないけど、小さなブロックだからサブルーチンと言っていいのかな？

```
@BINGO
BEEP 34
PRINT"●:アタリ！！！"
BGMSTOP:END
```

「BINGO（ビンゴ）（当たり）」っていうラベル名どおり、当たった時のプログラムだよね。

まあ「行って帰る」のがサブルーチンとは言ったが、あまりムズカシく考えず、これもサブルーチンのひとつでいいじゃろうな。
@BIGも「行って帰る」の「行って」があるかというとビミョウじゃが、これもサブルーチンではあるしのう。

ザックリしすぎて、ワカるようなワカんねえようなハナシだぜェ……。

こういうのはフンイキでおぼえておけばＯＫ！　もう少しあとでサブルーチンがクローズアップされるから、その時まではザックリでいこうね。

もうひとくふう

さて、ここまででサンプルプログラムがどう書かれているかはゼンブ見おわったかのう。

RND命令で決めた答を、INPUT文で当てさせるのが中心でしたね。

当たったかどうかはIF～THEN文でしらべて……

デカいか、小せえか、当たったか、サブルーチンでケッカを出すってワケだ。
けっこうタンジュンじゃねえの？

ここまで来たショクンなら、そうじゃろう。思えばズイブン成長したものじゃわい。
このさいじゃから、ひとつテストを出してみようかの。

テ、テスト!?

このゲームは少ない回数で答にたどりつくほどエラいわけじゃが、「何回使ってクリアしたか」が表示されておらん。
つまりはスコアじゃな。そこで、ゲームクリアしたら「何回使ってクリアしたか」表示するようにしてもらおうかの。

つ、つまりゲームとしておもしろくするためのクフウってヤツか。そういうコトならノゾむところだぜ！

（わりとタンジュンだなあ……）

１回答えるたびに数字がふえていく、つまり数字がヘンカしていく……ということは、何を使えばいいかもうわかるね。

オウ！　変数のデバンってワケだな！

とりあえず、変数名はSCORE（スコア）でいいかな。

ウッカリ「スコレ」と読みそうなことをのぞけば、わかりやすいぜ。

う、うん。とりあえず、スコアを書くところだけやっておこうか。

```
00ET PRINT SCORE;"カイテ゛クリア"
00EE BGMSTOP:END
```

いまの37行をひとつ下にずらして、その上に書いておいたよ。

まだトチュウだが、このままクリアするとどうなるんだ？

```
RUN
```

ワンパク君、なかなかサエておるな。カンセイするまで待たずに、ちょっとずつテストしてみるのもプログラムにはダイジじゃぞ。

```
ﾖ:アタリ!!!!
0カイテ゛クリア
OK
```

ウォオー！　ハイスコアをたたき出しちまったぜ！　スゲエ！

……どんだけテンネンなのじゃ。アキラカにそのケッカはおかしいぞい。

そういえば変数は、なにもしないままだとゼロになるんだったね。じゃあ、ダイジな「１回答えるたびに１増える」プログラムを足していこう。

プレイヤーが答えるというと、この行だね。

```
00IE INPUT"ﾖ:0～99ﾏﾃ゛ ﾉｽｳｼ゛ ﾊ";NO
```

そうだね。このあとに変数ＳＣＯＲＥを１ふやすことにしよう！

いまゼロの変数を１にふやす……ツマリＳＣＯＲＥ＝１……か？

……それだと、２にふやすことができないんじゃないかな。

この方法はサンプルプログラム３で出てきたね。ＳＣＯＲＥ＝ＳＣＯＲＥ＋１と書けば、１ずつふえていくハズだよ。

```
00Z0 SCORE=SCORE+1
```

お、オウ。オレとしたコトが、ウッカリしていたぜ。
とにかく、コレでスコアが出るようになったハズだぜ！

```
RUN
```

フフフ。それはどうじゃろうな？

ナニをキショク悪いワラい方してやがる……。見やがれ、うまく動いてるゼ！

```
ﾖ:アタリ!!!!
7カイテ゛クリア
OK
```

うん。そこまではモンダイないんだけど……

```
RUN
```

……あっ！　そうか！

なんのハナシだ？　もう１回やっても……オ、オォ!?

```
ﾎﾝ:アタリ!!!!
15カイﾃﾞ ｸﾘｱ
OK
```

な、なぜだ！　いまオレはタシカに８回でクリアしたハズだぜ！　サスガに15回もかけるほどねぼけちゃいねえ！

アリガチなシッパイじゃったな。変数はプログラムが**END**で終わっても、変わらずにプチコンの中にノコっているのじゃよ。

……つまり、最初にクリアした７回からまた数えはじめて、１ずつプラスしちまってたのか……。
だが、ゲンインがワカればコワくねえゼ！　ゲームを始めるたびにリセットすりゃいいんだろ？

```
SCORE=0
```

ゲームが始まるマエにこう書いとけば、カイケツだぜ！

ワンパク君……！　イガイにもセイカイじゃぞ！　これでテストはクリアーじゃ。

もっともボクのシュミとしては、プログラムのはじめに**CLEAR**命令を使って、変数をまるごとリセットする方が好きかな。

```
CLS:COLOR 0:BGMPLAY 1:CLEAR
```

インテリ君……！　どっちかと言えば、そっちの方がなおセイカイじゃ！

ナットクいかねェー!!

ウーム……。このチョウシでプログラムにまた何か書きたしていくと、変数が増えるたびにリセットの命令を書くのがメンドウじゃからな。**CLEAR**と１回書いておけば、もうシンパイないというコトじゃ。

チッ……、フロントマンのおカブをウバわれちまったぜ。こうなったらもうブンなぐるしかねェか？

キミのパンクスピリットはかたよってると思うぞい……。

今回のまとめ

BGMPLAY命令、BGMSTOP命令

```
BGMPLAY （曲番号）
```

0～29の音楽を鳴らします。

```
BGMSTOP
```

BGMPLAY命令で再生している音楽を止めます。

条件式のANDとOR

```
IF （条件式） OR （条件式） THEN （命令）
```

条件式のどちらかが一致していれば命令を実行します。

```
IF （条件式） AND （条件式） THEN （命令）
```

全部の条件式が一致したときだけ命令を実行します。

ビット演算

プログラマーからひとこと

この章では、「ビット演算（えんざん）」というものをおぼえます。
前にちらっと出てきましたね。
その時は、むずかしいので説明はしませんでした。
そうです、ビット演算はむずかしいのです。
あんまりむずかしいので、飛ばして読んでもいいです。ほら、ここにボタンも用意しましたよ。 とばし読み

これまでもむずかしい話はいろいろしました。どれもむずかしいかわりに、かならず必要なものでしたから、それはいいと思うんです。
配列変数なんて、かなりめんどくさかったですが、ないと困るものでしたよね。
ところがビット演算は使わないでも、まあなんとかなるんです。
私もはじめにおぼえたときは、「なくていいじゃん」と思いました。
どうです、とばし読みしたくなってきたでしょう。

じゃあ何のためにおぼえるの？　いくつか理由はあります。

1．コンピューターの考え方がわかる
私たちがものを考えるとき、だいたいイメージとか日本語でものを考えるように、コンピューターは「ビット」でものを考えます。
つまり、ビットの考え方を理解すると、ぐっとコンピューターの考え方に近づけると言えるでしょう。
こう言われてヤル気が出てきた人は、わりとプログラマー向きな性格だと思います。
ぶっちゃけ、コンピューターの考え方がわかってもモテませんよ。そんなに役に立たないですよ。

2．短いプログラムが書ける
ビット演算をうまく使うと、IF文を何行も使うようなプログラムも１行でぐっとシンプルに書けるようになります。
これはアツいですね。と、私は思いますが、それは私がプログラマーだからかもしれません。
プログラマー的にはシンプルにスパッと書けるとプロっぽくてうれしいものですが、でもちょっとくらいプログラム長くたって別にいいじゃん、と言われると、まあ、それもそうだな、なんて気もします。

3．速いプログラムが書ける
コンピューターは「ビット」でものを考える、とさっき書きました。
ビット演算はビットを使ったプログラムなので、コンピューター的にはラクっていうか、やりやすいものみたいです。
おおざっぱに言って、ビット演算を使ったプログラムと、同じことをビット演算を使わずにやったプログラムでは、ビット演算した方が速くなります。
よくゲームで「処理落ち」とか「重い」とか言うでしょ。アレが少なくなります。
もっとも、この講座で教えているレベルだと、違いがまったくわからないていどの速さだったりもします。高レベルなプログラムになるほど必要になってくるんですが、いま必要かっていうと別にそうでもないんですね。

4．人のプログラムが読みやすくなる
けっきょく一番だいじなのは、ここかもしれません。
ビット演算を使ってるプログラムは、ビット演算の考え方を知らないと、何が書いてあるのかよくわかりません。
そして参考にしたいプログラムにかぎって、ビット演算を使ってるのです。これはプログラムあるあるです。
おおむねすぐれたプログラマーというのは、コンピューターが好きで、短くて速

く動くプログラムを書きたがるものです。つまりビット演算をしたがるものなのです。
すぐれたプログラマーが書いたプログラムを読むのはとても勉強になりますが、書いてあることがわからないとあまり勉強になりません。そこでビット演算の初歩だけでもおぼえておくといい、というわけです。

どうです？　ビット演算を知りたくなってきましたか？
やっぱりそうでもないですか？
その気持ちもとてもよくわかるので、ここにボタンがあるのです。 とばし読み

まあ、今までの話をきいて、じゃあおぼえてみるか、と思った人だけが読むくらいのサジかげんでちょうどいいような気がします。

２進数


いいかげんにしろ！　あれだけムズかしいとかスグには役に立たねェとか言われたアトじゃ、そうそうヤル気は起きねーぞ……


ではビット演算のキソから始めようか！　ワクワクするよね！


コ……コイツ、ものすげえタフか、それともオレのハナシをきいてねェのか……。


まあ、ここはスナオに話にのっておこうか。


ビット演算をリカイするには、その前に「２進数」のことを知らないといけないね。


２進数？


ボクらがふだんイシキせずに使ってる数字は、「10進数」でできているんだ。これがコンピューターだと、２進数になるんだね。


それにしちゃ、今までBASICでフツーに数字を使ってなかったか？　アレも２進数ってやつなのか？


BASICではふつう、ボクらにもわかりやすいように10進数しか出てこないんだ。ボクらの目に見えないだけで、コンピューターの中では２進数で動いているんだよ。


そうか……わざわざオボエなくていいっていうのは、そういうトコロなのかあ。10進数と２進数はどう違うの？


ボクらの10進数では、０・１・２・３・４・５・６・７・８・９の10コの数字の組み合わせで数をつくるよね。これが２進数だと、「０」と「１」の２コの数字しか使わないんだ。


エ、……ハァ!?　それじゃモノを数えられねェだろ！


そうでもないんだな。
ここからは、区別しやすいよう２進数には線を引いておくよ。
たとえば10進数で「０」「１」の次は「２」だよね。
２進数だと、「**0**」「**1**」の次は「**10**」になるんだ。


ジ、10だと!?


２進数だと**10**と書いて「イチゼロ」とか読む方がいいんだけど……


うーん……急にチシキをつめこまれてる気がしないでもないぞ。

もうちょっとユックリいこうか。
10進数では、使える10コの数字の中で一番大きいのは「9」だよね。9より大きい数を書くときはどうするかな？

そりゃトウゼン、1ケタ位（くらい）が上がって「１０」だゼ！　……ン？　１０？

そう、２進数も同じようなものなのさ。

ええと、２進数だと使える２コの数字の中で一番大きいのは「**1**」。それより大きい数を書くときは……

位が上がって、「**10**」になるワケか！

そういうコト。このルールだけオボエれば、２進数はカンペキさ。

ホント!?　うーん、**10**の次は１ケタ目が上がるから、**11**。その次は、もう上げられる位が残ってないから、位上がりして**100**……。

オイオイ、とんでもねえハイペースでケタが上がってるぞ！　ホントにコレで合ってるのか？

バッチリだよ！　０〜10までを数えた表がコレになるね。

10進数	２進数
0	**0**
1	**1**
2	**10**
3	**11**
4	**100**
5	**101**
6	**110**
7	**111**
8	**1000**
9	**1001**
１０	**1010**

こんなカンジで、どんどんケタが上がるから、**1010**と書いて「イチマルイチマル」とか読むんだね。

フーム……じゃあ10進数で「1976」は、２進数だとどうなるんだ？

……。

……。

Google検索してみようか！

ナンだソレ！　ググんのかよ！

しかも、本当にそれで答が出るんだ！

まあ、２進数をいちいち手で計算するのはタイヘンだからね。検索とかアプリで出すのがフツウなのさ。

「1976 = **0b11110111000**」か……。この「**0b**」ってのはナンだ？

10進数と区別するための記号だね。1976をただ２進数で言えば**11110111000**だけど、これだと10進数の「111億1千11万1千」にも見えるから、アタマに**0b**と付けるのがマナーってことかな。

だいたいわかってきたけど、それにしてもメンドくさい数え方だね……。

もともとコンピューターはスイッチがON(オン)かOFF(オフ)かの組み合わせで作られたからねえ。この２つの組み合わせを「ビット」と言うんだ。

ん……ビット？

ワスれるトコだったぜ、アブねえ！　「ビット演算」のハナシはドコ行ったんだ？

初歩のビット演算

それじゃあ、プチコンで使うビット演算のサンプルを見てもらおうか。

```
0001 A=5 AND 3
0002 PRINT A
```

Aは変数だね。これを**RUN**すると、変数**A**に**1**が入るよ！

ハ、ハァ!?

```
RUN
1
OK
```

ANDっていうから、てっきり5+3みたいなコトかと思ったけど……

5 AND 3はそういう計算とはちがうモノだってオボエておこうね。
答だけ見ても何だかわからないだろうから、トチュウの計算式も見てみよう！

```
  101
AND 011
  = 001
```

えーと……

なんだコレ！　ワカるフンイキがまったくねェぞ！　**5 AND 3**のドコからこの式になるってんだ！

いやいや、たぶん２進数がここで出てくるんだよね……。

そのとおり。**5**を２進数にすると**101**。**3**を２進数にすると**011**になるね。

……**3**を２進数にすると**11**じゃなかったか？

アタマにつけるゼロは、いくらつけてもいいんだ。**101**と位をそろえるために、１コつけておいたよ。

ウーム……つまり、**5 AND 3**を２進数にすると、

```
  101
AND 011
```

ココまではまあワカッたぜ。だがコタエが**001**ってのは、ナットクいかねェな！

習ったことのない計算だから、そう思うのもトウゼンだね。
ビット演算の「AND」では、「両方とも**1**になっている位だけ**1**になる」とオボエると……カンタン……かなあ？

……かなあ？

ダレもナットクいってねェじゃねえか！

うーん。「位」がポイントなんだよね。たとえば１ケタ目は……

```
  101
AND 011
```

両方とも「**1**」になっている位……だから……

```
  101
AND 011
  = ??1
```

答も１になるってコト？

いいよ！　そこまでワカれば、あともカンタンなハズ！

つまり２ケタ目も、こういうコトか？

```
  101
AND 011
  = ??1
```

コレはかたっぽが「**0**」、もうかたっぽは「**1**」だから、

```
  101
AND 011
  = ?01
```

ゼロがくるワケか？

そのとおり！　３ケタ目はもうわかるよね。
片方が「**1**」、もう片方が「**0**」ということは……

```
  101
AND 011
  = 001
```

これで**001**までは答が出たね。

その**001**は２進数だから、10進数に直すと……

1が変数**A**に入るってコトか。

計算のしかたさえワカれば、あんがいラクだろう。たとえば**B=7 AND 11**だとどうなるかな？

ツギツギに出してきやがって！　まあヤリ方はワカッてるぜ。まず２進数に変えるんだろ？
７は**111**、11は**1011**だから、こうなるゼ！

```
  0111
AND 1011
  = ????
```

で、「両方とも**1**になっている位」をさがすと……

```
   0111
AND 1011
  = ??11
```

1ケタ目と2ケタ目が「1」になるね。のこった位はトウゼン「0」だから……

答はこうなるね。

```
   0111
AND 1011
  = 0011
```

0011、つまり10進数で言いかえると3が変数Bに入ることになるね。どうだい、思ったよりはカンタンだろう？

やり方ドオリにやれば、まあデキねえコトはねェが……トンデモなくメンドくせェぜ！

ANDのほかにも、「OR」や「XOR」の計算もあるんだけど……

ウオォー！　メンドくせェー!!

ORとXOR、NOTのビット演算

ワンパク君、すっかりOverflowエラーというトコロじゃな。
などとBASICネタもクールに使いこなすワシじゃ、ハカセじゃ。ビット演算の「OR」も「XOR」も「AND」とあまり変わらんぞい。ものはためしじゃ、ジッサイの計算式を見てもらおうかの。

OR

```
   0101
OR 0011
 = 0111
```

ワカるかの？　「どちらか片方でも1がある位は1になる」のがOR(オア)の計算じゃ。

XOR

```
   0101
XOR 0011
  = 0110
```

XORじゃと、「両方とも1になっている位は0になり、片方にだけ1がある位は1になる」わけじゃ。ちょっとヤヤコシイのう。
そうそう、「XOR」は「exclusive OR」略して「エクスオア」とか「エックスオア」などと読むのじゃぞ。伝説の剣かなんかの名前っぽくてカッコイイわい。

NOT

```
NOT 1101
  = 0010
```

NOT(ノット)はちょっとトクシュじゃな。2つの数をくらべたりせず、ただ0と1がひっくり返るのがNOTじゃよ。

ビット演算の実例

ゼェ、ゼェ……。アタマがワレそうだぜ……。

ひととおりオボエたけれど、けっきょくコレって何につかうの？

たしかに、何の役に立つのかわからないとハリアイがないよね。
じゃあ、ちょっと前に出てきたこのプログラムをおぼえてるかな？

```
B=BUTTON()
```

```
00058 IF B AND 1 THEN BEEP 52
00059 IF B AND 2 THEN BEEP 53
00060 IF B AND 4 THEN BEEP 62
00061 IF B AND 8 THEN BEEP 25
```

たしかサンプルプログラム３で出てきたね。
BUTTON()命令とIF文の組み合わせで、十字ボタンの押された方向を調べるんだったっけ。

「IF B AND 1」と書くと、「Bの中に１がまじっている」というイミになると習ったよね。

十字ボタンの右上が押されたら、BUTTON()命令で変数Bのナカミには１（上）＋８（右）の9が入る……。
で、Bのナカミが9でもなぜか「IF B AND 1」や「IF B AND 8」と書くとアタリになる、って話だったな……。
あらためて考えると、ワケがわからねェぜ！

あれ、だけどこのB AND 1って、ビット演算の書き方じゃないかな？

そのとおり！　もう１度、このIF文をビット演算の考え方で見てみると、イミがわかってくると思うよ。

チクショウ、またビット演算すんのかよ！
まず右上が押されて変数Bに9が入ったとするだろ……で、58行目がIF B AND 1 THEN……。
これは変数のナカミを考えると、IF 9 AND 1 THENってコトだよな。

9 AND 1は、こう計算するんだったよね。

```
  1001
AND 0001
  = 0001
```

答は0001、10進数に直すと「1」になるね。

じゃあ次の59行は……IF B AND 2 THENだから、9 AND 2をビット演算すると……

```
  1001
AND 0010
  = 0000
```

オ、オォ!?　キレイにゼロになったぜ。

だんだんカラクリが見えてきたね。「B AND 4」を調べる60行だと、

```
  1001
AND 0100
  = 0000
```

やっぱりゼロ。
「B AND 8」を調べる61行なら、

```
  1001
AND 1000
  = 1000
```

これを10進数に直すと「8」になるね。

下方向（AND 2）も左方向（AND 4）もビット演算のケッカはゼロ……。上（AND 1）や右（AND 8）だと、ゼロにはならねェんだな。

ソコだね！
ちょっと話がそれるようだけど、IF （変数） THENというカタチのIF文は、「変数がゼロじゃなければ」というイミになるんだよ。
ここと合わせて考えてみよう。

ってコトは……58行と61行のIF文の中にあるビット演算のケッカがゼロじゃないから……

……THENのサキの命令がジッコウされるってワケか！

そういうコト！
BUTTON() 命令で変数に入る数字には、はじめからビット演算しやすい数字が選ばれているんだよ。

フムフム。たしかに変数B に8（右ボタン）が入った時は、B AND 8以外ではゼロになるね。

それ以外のどの数でも、ちゃんとウマく動くようにできてるよ。なかなかベンリだろう。

ナルホド……サンプルプログラム３のころのギモンが、やっとトケたってカンジだぜ。
しかしよォ……

どうしたのワンパク君。スッキリしないカオじゃないか。

BUTTON() 命令はもともとコレのために作られてるけどよォ、こんなコトをイチから作るのってムズカシくねえか？

たしかに、頭で考えてココまで作るのはタイヘンそうだよね。

うん、サスガになにもかも２進数で考えてプログラムに組むのはムズカシいだろうね。だけどビット演算にはいくつか必勝法になるパターンがあるんだ。BUTTON() 命令で使ってる数の組み合わせもそのひとつだよ。

つまり……その必勝パターンと使い方をオボエれば、ウマくプログラムに使えるってのか？

そういうコトなんだけど……

？

その話をするともっと長くなっちゃうから、また今度にしよう！

ウォオー!!　今回の話がオレのプラスになったのかどうか、サッパリわからねー!!

なかなかタイヘンじゃったようじゃのう。しかしワンパク君、ここでオボエたコトはけっしてムダにはならんぞい。ツギのサンプルプログラムでもビット演算がさっそく出てくるくらいじゃ。

今回のまとめ

２進数

０と１の組み合わせでできた数え方が２進数です。アタマに「0b」と付けて区別したりします。3を２進数で言うとどうかとか、111100を10進数（ふつう）に言いなおすとどうかとか、検索して調べれば早いんじゃないでしょうか。

ビット演算（AND）

7 AND 11のような形で使います。
これを２進数におきかえると「111 AND 1011」。さらに見やすい式の形にすると、

```
  0111
AND 1011
  = 0011
```

「両方とも１になっている位だけ、１になる」のがANDを使ったビット演算です。

この場合、答の0011を10進数に変えれば「3」、つまり `A=7 AND 11` と書けば変数Aに3が入ります。

ビット演算（OR）

```
  0101
OR 0011
 = 0111
```

「どちらかでも１がある位は、１になる」のがORを使ったビット演算です。

ビット演算（XOR）

```
   0101
XOR 0011
  = 0110
```

「両方とも１になっている位は０になり、片方にだけ１がある位は１になる」のがXORを使ったビット演算です。

ビット演算（NOT）

```
NOT 1101
  = 0010
```

２つの数をくらべずに、ただ０と１がひっくり返るのがNOTを使ったビット演算です。

サブルーチン

プログラマーからひとこと

これまでいろいろな命令をおぼえてきましたが、そのクライマックスになりそうなのが、これから出てくる「GOSUB」命令です。
ここまでおぼえれば初心者のプログラムはもう最後といっていいかもしれません。

ゴーサブ
GOSUB。
英語としてもわりとヘンなんですが、もともとの意味は「GO to SUB routine（ゴー・トゥ・サブ・ルーチン）」といって、まあ「サブルーチンに行け」的なことだったみたいです。

サブルーチンといえば、前にもちょっとだけ出てきましたね。
「行って帰ってくる小さなプログラムのブロック」なんて説明もしましたが、おぼえてましたか？
この「行って帰ってくる」をカンタンにするのがGOSUB命令ですが、くわしくはインテリ君の説明にまかせましょう。

GOSUB命令

サブルーチンのハナシはもう終わったんじゃねェのか？　ヨウするに「行って帰ってくる小さなプログラムのブロック」がサブルーチンなんだろ？

そこまでは合ってるよ。これからおぼえるGOSUB命令は、それをもっとラクにするための命令だね。

たしかサブルーチンは、こういうものだったよね。

```
0001 'メインループ
0002 @MAIN
0003 PRINT"ココハ メインループ テ゛ス"
0004 GOTO @SUB
0005 @MAIN2
0006 GOTO @MAIN
0007
0008 'サブルーチン
0009 @SUB
0010 PRINT"ココハ サブルーチン テ゛ス"
0011 GOTO @MAIN2
```

これだとホントにサブルーチンにするイミのない、グルグル回るだけのプログラムだけど……。

うん、でもおさらいとしてはこれでいいんじゃないかな。４行目でサブルーチン@SUBに行って、11行目でメインループの中の@MAIN2に帰るんだね。

コレをカンタンにするのがGOSUB命令だってのか？　スデにかなりカンタンな気もするが……

まあ見てみようよ。GOSUB命令を使うと、こういうプログラムになるね。

```
0001 'メインループ
0002 @MAIN
0003 PRINT"ココハ メインループ テ゛ス"
0004 GOSUB @SUB
0005
```

```
0006 GOTO @MAIN
0007
0008 'ｻﾌﾞﾙｰﾁﾝ
0009 @SUB
0010 PRINT"ココハ ｻﾌﾞﾙｰﾁﾝ ﾃﾞｽ"
0011 RETURN
```

４行目でGOSUB命令を使っているのがわかるね。

それだけじゃねェぞ、５行目にあったラベルが消えてんじゃねェか！　これじゃサブルーチンから戻ってこれなく……アレ？

サブルーチンの最後が、GOTO命令じゃなくRETURNになっているね。

GOSUB命令は、RETURN命令とセットで使うんだ。
プログラムを見ればだいたい分かるかな？　GOSUBのあとにRETURNがあると、GOSUBしたすぐアトまで自動で帰るんだね。

……リクツはワカッたぜ。あんがいカンタンなモンなんだな。

サブルーチンに行くたびにラベルを付けなくてよくなるのは、たしかにベンリかな。

しかしコレ……サブルーチンって、ホントにやらなきゃダメか？　ワザワザGOTOやGOSUBで飛ばさなくても、メインループの中でやればいいコトじゃねェか？

いいトコロに気が付いたね。ジツはここまでで使ったサブルーチンは、たしかにサブルーチンの良さがちゃんと出てないんだな。
そこで、このプログラムを見てもらおうか。

```
0001 'ﾒｲﾝﾙｰﾌﾟ
0002 @MAIN
0003 A=1
0004 GOSUB @SUB
0005 A=2
0006 GOSUB @SUB
0007 GOTO @MAIN
0008
0009 'ｻﾌﾞﾙｰﾁﾝ
0010 @SUB
0011 PRINT A;"カイメ ノ ｻﾌﾞﾙｰﾁﾝ"
0012 RETURN
```

ドコがポイントか、わかるかな？

サブルーチンに行く前に、変数Aのナカミを変えているね。それにメインループの中から２回、同じサブルーチンに飛んでいるよ。

それからサブルーチンの中でも変数Aを使ってるぜ！　つまり……サッパリわからねーな！

そ、そうなの？
まず同じサブルーチンでも、RETURNで帰る行がちがうというのが１つ目のポイントかな。

同じRETURNなのに、４行目から飛んできたときは４行目に帰ってきて、６行目から飛んできたときは６行目に帰るんだね。ちゃんとしてるなあ。

タシカにそれと同じコトをGOTOでやるのはメンドクセェな……。アチコチから飛ぶんならGOSUBを使えってコトか。

もう１つのポイントは、１コだけのサブルーチンでも、ＧＯＳＵＢする前の変数によってＰＲＩＮＴする字が変わるコトだね。

ＰＲＩＮＴにかぎらず、変数によってやることを変えられるんだね。

そういうコトだね。いろいろなプログラムを見てみるとわかるけど、決まった動きを、変数のナカミによって変えながらできるのがサブルーチンの良さだね。

つまりサブルーチンってのは、プログラムのどこから、どんなジョウタイでも、「オイこれやっとけ」って言えば空気を読みながらひととおりのシゴトをこなすってワケだな。

どうしたのワンパク君、よくわかってるじゃないか！

いや、リソウのパシリってこんなカンジじゃねえかって考えたら……

……

……

今回のまとめ

GOSUB～RETURN命令

`GOSUB @（ラベル）`

`RETURN`

GOSUBで指定したラベルまでジャンプします。その後にRETURNと書いてあれば、GOSUBの直後まで戻ります。

サンプルプログラム７（その１）

プログラマーからひとこと

いよいよサンプルプログラム７、この講座ではこれが最後のサンプルです。
サンプルプログラム５や６はやってませんけど……。
うーん、エピソード１～３を飛ばしてエピソード４からストーリーが始まる的な？
いや、そういうものでもないですね。

サンプル５と６は、「グラフィック面」や「ＢＧ面」といった、これまでとはだいぶ違うものの話になるので、この講座ではいったんおいておきます。
新しくおぼえる命令がドーンと出てくるので、そういうのって思わずくじけちゃいそうじゃないですか。

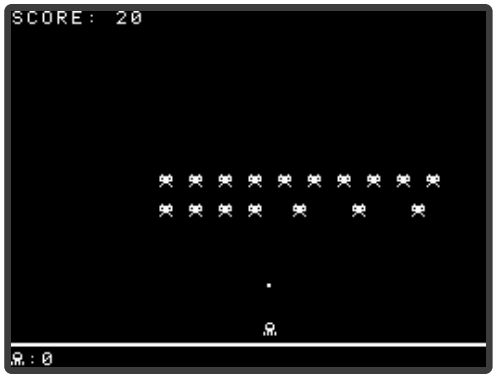
それよりは先にスパッとアクションゲームを作ってしまいましょう。
そうです、そういえばサンプルプログラム７はゲームです。それもアクションゲームです。
いよいよって感じですね。
だいたいは、これまでにおぼえたことの組み合わせで、できるようになっています。

と言っても、応用そしてまた応用なのはたしかなので、何回にもわけてじっくりやっていくことにしましょう。
どんなゲームなのか、とりあえず見てみましょう。

サンプルを遊んでみよう

いよいよクライマックスというカンジじゃのう。これは今のショクンにピッタリのサンプルプログラムじゃろう。

```
LOAD"SAMPLE7"
OK
RUN
```



自機を左右に動かしながら、上からせまる敵をショットで打つゲームだね。

十字ボタン左右で移動。画面タッチでショットっていうのはめずらしいね。

いや……ちょっと待て、コレおかしいぞ！　どうやっても負けにならねーじゃねェか！

じゃから、タイトルが「侵略未遂（しんりゃくみすい）」となっとるじゃろ。カレらは、最後までシンリャクできなかったのじゃよ。カナしい話じゃな。

いやそうじゃねェだろ！　ナンのための残機ヒョウジだよ！　だいたい敵をゼンブ倒しても特にナニも起きねー！

ウム……、コレはそもそもユーザーショクンのチカラで、未完成のプログラムを完成させようというシュシなのじゃ……が……ちょっとムリがあったじゃろうか。

まあ、そういうコトなら……いいの……かな？

そういうことにしておこうよ。今のところ気になるポイントは、
・敵にせめ込まれてもダメージがない
・敵をゼンブ倒しても終わらない
この２つかな？

・敵のスピードが早すぎる
ってのもモンダイだぜ。

・Ａボタンでショットにならない
のもやっぱりやりづらいよね。

ワシが言うのもなんじゃが、テキカクなシテキじゃな。
ホカにもプラスアルファのためにいくつか変数をしこんでおるんじゃが……

それはイイや、コレ以上ふやすとタイヘンそうだしな！

ワシが言うコトでないかもしれんが、テキカクじゃのう……。

再現をはじめてみよう

プログラムを改造するには、まずプログラムをリカイした方がいいよね。せっかく総まとめのプログラムなんだから、イチから自分で作るつもりでやってみようか！

カンタンそうに言うなあ……。

ムズカシく見えても、だいたいはこれまでのおさらいでデキるプログラムだよ。
たとえば、プログラムのサイショに何をすればいいのかはもうわかるよね。

そこはイマまでのサンプルと同じパターンでイケるだろうな。

```
0001 '
0002 '┌──────────┐
0003 '│SAMPLE7     │
0004 '│  シンリャクミスイ  │
0005 '└──────────┘
0006 VISIBLE 1,1,0,0,0,0
0007 CLS:COLOR 0
0008 CLEAR
```

だいたいこんなモンか。本当はこのアトにＤＩＭ命令がくるんだろうが……。

```
0008 DIM ???
```

配列変数のヨテイが立ってないから、トクに思いつかねェぜ！

そのあたりは後で出てくるだろうから、今はまだいいかな。
REMの' を行のはじめに入れて、メモとして残しておこうか。

```
0008 'DIM ???
```

いわゆる「コメントアウト」じゃな。プログラムを組みながらなら、こういうコメントの使い方もユウコウじゃぞ。

動かしてみよう

ウーム……とりあえずプレイヤー機を出してはみたが……

```
0010 '---
0011 LOCATE 10,21:PRINT"A";
0012 LOCATE 0,22
001E PRINT"━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━";
```

サンプルプログラム１のころにおぼえたコトだね。

このプレイヤーを動かすプログラムがいるだろうね。いまのうちにメインループを作っておこう。

```
0020 '---
0021 @LOOP
0022  GOSUB @MYSHIP
0023 GOTO @LOOP
```

このラベル@MYSHIP（自　機（マイ・シップ））で、プレイヤーを動かすサブルーチンを作るんだね。

サンプル３の方法で左右に動かすなら、こんなカンジだろ？

```
0025 '--- ｼﾞﾌﾞﾝ
0026 @MYSHIP
0027 B=BUTTON()
0028 IF B AND 4 THEN ???
0029 IF B AND 8 THEN ???
002D RETURN
```

だが、わからねえ！　このIF文をどうすりゃ、プレイヤーが動くんだ？

そういえば、まだキャラクターが動くタイプのプログラムはやったことがなかったね。少しじっくり作ってみようか。

```
0011 LOCATE 10,21:PRINT"A";
```

まずはこの行に注目かな。

イチバンはじめのトコロじゃねェか！　オレのヤリ方にモンクでもあるってのかよ！

このLOCATEの数字を変えると、プレイヤー機のバショも変わるのはなんとなくイメージできるかな？

まあ、それくらいなら。
たとえばLOCATE 11,21:PRINT"A";と書けばもう１文字ぶん右に行くよね。

ソレと押されたボタンを組み合わせるのか……ツマリ、BUTTON()が押されたらLOCATEの数字を変える……？　しかしLOCATEの数字を変えるには……

あ、変数を使うんだ！

！　……「数」が「変」わるから「変数」。そういうコトか……！

そうマチガってるワケでもないかな。数字がたびたび変わるようなトコロは、変数で作っておくとベンリだね。

ということは、この行も……？

```
0011 LOCATE 10,21:PRINT"A";
```

こんな感じがいいんじゃないかな。

```
0011 PX=10:PY=21
0012 LOCATE PX,PY:PRINT"A";
```

変数の名前に使ったPはプレイヤー(Player)のP。XとYはヨコ方向・タテ方向のイミだよ。

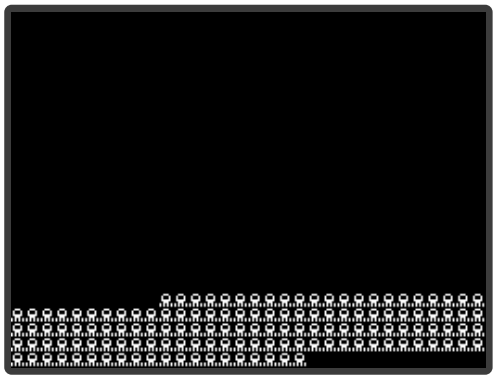
数学で「座標」のヨコ方向に「x」、タテ方向に「y」を使うコトがモトモトのイミかのう。コンピューターのセカイでもヨコといえばX、タテといえばY、というコトが多いのじゃ。

リクツはワカッてきたぜ。ボタンのあたりは、こうすりゃイイんだな？

```
0025 '--- ｼﾞﾌﾞﾝ
0026 @MYSHIP
0027 B=BUTTON()
0028 IF B AND 4 THEN PX=PX-1
0029 IF B AND 8 THEN PX=PX+1
0030 LOCATE PX,PY:PRINT"A";
0031 RETURN
```

左ボタンを押せばＰＸから１引いて、右ボタンを押せばＰＸに１足すってワケだ。そのアトにLOCATEしてPRINTすればバッチリだぜ！

さて、どうかのう？　ためしにRUNしてみるかの。



ウ、ウォオー！　ナンだコリャアー！

アイデアはまちがってなかったけど、いくつかダイジなところが抜けていたね。
まず、B=BUTTON()から戻ってくるまでのループが早すぎるってこと。ボタンをチョンと押しただけで、ものすごいスピードで何度も押したことになってるんだな。

そういえばサンプルプログラム３でも、そんなコトがあったね。

となると、思い出すこともあるんじゃないかな？

```
0021 @LOOP
0022  GOSUB @MYSHIP
0023  VSYNC 1
0024 GOTO @LOOP
```

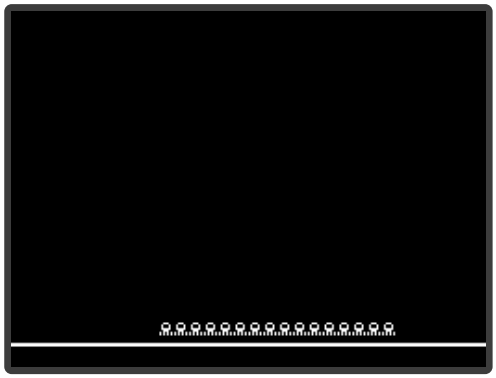
メインループの中にVSYNC命令をふやしたよ。

60分の１秒だけプログラムの動きを止めるんだったね。早すぎるループをわざと遅らせるのはワカるけど……60分の

1秒って、そんなあっと言う間でいいの？

そのくらいでも、アクションゲームならイガイとちょうど良かったりするんだよ。ものたりなければ、増やせばすむコトだしね。

ナルホドな。とにかくモンダイはカイケツだぜ。RUN！



カイケツしきってねェー！

動かし方のコツを学ぼう

ナニがおきておるのか、23行目をVSYNC 20ぐらいに変えて見るとハッキリするじゃろう。
こういう「重し（ウェイト）」をかけるのは、テストにはユウコウじゃぞい。

な……ナンとなくはワカッたな。ヨコに動かすと、プレイヤー機が分身するみたいに増えていっちまうんだ。

いちど画面にPRINTした字は、消さなきゃずっとそこに残っちゃうのさ。

言われてみれば当たり前だけど……じゃあどうやって消せばいいの？

それはカンタンさ。"　" つまりスペースを同じ場所にPRINTすれば、Aはなくなるってワケだね。

消すっつーか、上書きするっつーか、だな。
このプログラムだと、モトの場所にスペースを書けばイイんだから……ン？　ブツブツ……ボタンが押されたらもうPXを変えてるワケで……モトの場所を消さないとイケねェんだけど、ボタンを押してないのに消したらマズいワケで……
イマイチわかんねェぞ！

これは、もっとプログラムを書き足さないといけないね。

```
00025 '--- ｼﾞﾌﾞﾝ
00026 @MYSHIP
00027 B=BUTTON():VX=0
00028 IF B AND 4 THEN VX=-1
00029 IF B AND 8 THEN VX=1
00030 LOCATE PX,PY:PRINT" "
00031 PX=PX+VX
00032 LOCATE PX,PY:PRINT"A";
00033 RETURN
```

わかるかな？　クッションになる変数VXを新しく作ったよ。

ええと、ふだん変数VXは0で、ボタンを押したときだけ-1か1になるんだね。

変数ＰＸはまだ変わってねェから、まずスペースで消して……あらためてＰＸにＶＸを合わせてからＬＯＣＡＴＥしなおしてｗをＰＲＩＮＴするってワケか！
ワレながら、英語の多いコトバをしゃべっちまったぜ！

英語ついでに言うと、変数ＶＸの「Ｖ」は「動く方向と量」という意味の「ベクター（Vector）」のコトじゃな。プログラムではよく使われるコトバじゃが、広義のベクトルと違って……

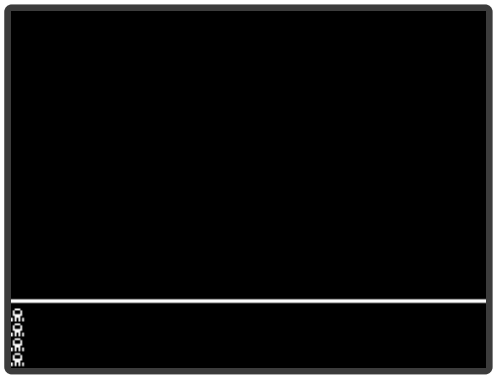
ちょ、ちょっと待って！　センモン的なハナシはいいんじゃないかな。
それより、ボタンが押されてない時にもいったんスペースで消して、同じ場所にもういちどｗを書いているよね。これはいいの？

ＲＵＮしてみるとわかるけど、すごいスピードで消しては書いてるから、そう気にはならないね。
でも、それがベストかというとビミョウかな。こう書くのが正しいのかもね。

```
0025 '--- ｼﾞﾌﾞﾝ
0026 @MYSHIP
0027 B=BUTTON():VX=0
0028 IF B AND 4 THEN VX=-1
0029 IF B AND 8 THEN VX=1
0030 IF VX==0 THEN @NOTMOVE
0031 LOCATE PX,PY:PRINT" "
0032 PX=PX+VX
0033 LOCATE PX,PY:PRINT"ｗ";
0034 '---
0035 @NOTMOVE
0036 RETURN
```

ボタンが押されていないときは、イッキにラベル＠ＮＯＴＭＯＶＥ（NOT MOVE 動かない）まで飛んで、プレイヤー機をＰＲＩＮＴし直さないことにしたよ。

コレで動きはバッチリってコトだな。ＲＵＮしてみるゼ！



チ、チキショー！
画面のハジまで動かしたら、オカシなコトになっちまったぜ！

それは、変数ＰＸが画面を飛びこしちゃったのがゲンインかな？

ＰＸが０のときに－１しちゃったり、ＰＸが３１のときに＋１しちゃったり、だね。

テメーら、もっとわかりやすいコトバでしゃべりヤガレー！　画面のハジまで行ったら、それよりムコウに動かさないようにすればいいだけじゃねえか！

これはワンパク君の言うとおりだね。ｗをＰＲＩＮＴする前に、IF文を２行用意しておこうか。

```
0032 PX=PX+VX
0033 IF PX<0 THEN PX=0
```

```
IF PX>31 THEN PX=31
LOCATE PX,PY:PRINT"Ω";
```

これでプレイヤーの動きはモンダイなくなったね。

コレはコレで、動いてねェプレイヤー機を消しては書いちまってるが……、モトのサンプルプログラムがこうだからシカタねェのか？

し、しもうたァ――ッ！

ま、まあそのあたりはコンゴのカダイとして……。
いったん、ココまでのプログラムをまとめておこう。
どういうイミなのか考えながらジッサイに打ち込んでみると、もっとよくワカると思うよ！　モニタの前のみんなもやってみてね！

プログラムリストを表示

今回のまとめ

キャラクターを動かすには

たとえばLOCATEの位置を変数にして、BUTTON()の入力に合わせてIF文で変数を足したり引いたりすると、動いて見えます。
動く前にいた場所に、空白スペースを上書きして消してやると自然に見えそうですね。
LOCATEする場所が画面をはみださないように注意！

VSYNC命令とアクションゲームの関係

プログラムを一時停止するVSYNC命令で、スピードを落とせます。早すぎて操作が追いつかない時にはとてもよく使いますね。

サンプルプログラム７ (その１)でできたリスト

```
'
' ┌──────┐
' │SAMPLE7 │
' │ シンリャクミスイ │
' └──────┘
'
VISIBLE 1,1,0,0,0,0
CLS:COLOR 0
CLEAR
'DIM ???

'---
PX=10:PY=21
LOCATE PX,PY:PRINT"Ⓐ";
LOCATE 0,22
PRINT"━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━";

@LOOP
 GOSUB @MYSHIP
 VSYNC 1
GOTO @LOOP

'--- ｼﾞﾌﾞﾝ
@MYSHIP
B=BUTTON():VX=0
IF B AND 4 THEN VX=-1
IF B AND 8 THEN VX=1
IF VX==0 THEN @NOTMOVE
LOCATE PX,PY:PRINT" "
PX=PX+VX
IF PX<0 THEN PX=0
IF PX>31 THEN PX=31
LOCATE PX,PY:PRINT"Ⓐ";
'---
@NOTMOVE
RETURN
```

サンプルプログラム7（その2）

プログラマーからひとこと

前回は、キャラを動かす方法を重点的に学びました。
たぶん今のあなたなら、画面の中の主役キャラ（たとえそれが「🚢」とかだったとしても）を十字ボタンで上下左右に動かすのもカンタンなんじゃないでしょうか。
そう考えるとかなりレベルアップしたって気分になりませんか。

今回は、フラグの考えを中心に学んでいきます。
ゲーム用語で「フラグ」ということばを聞いたことはありませんか？　あのフラグです。
これまたゲームプログラムらしいフンイキが出てきましたね。
これを読み終えたとき、あなたはまた一歩レベルアップすると言えるんじゃないでしょうか。
それを楽しみに読んでいきましょう。

弾を出してみよう

しかしプレイヤーキャラ１つ動かすだけでも、ケッコウ考えるコトが多いもんだぜ……。

ホントだね。でも、やり方がだいぶわかったから、次からは応用でうまくできるんじゃないかな？

ちょうど応用になりそうなトコロがあるよ。画面タッチで出るショットのプログラムを作ってみようか。

```
0021 @LOOP
0022  GOSUB @MYSHIP
0023  GOSUB @MYSHOT
0024  WAIT(1)
0025 GOTO @LOOP
```

これもサブルーチン@MYSHOT（自弾（マイ・ショット））として用意することにしたよ。

まずはフツウにサブルーチンをはじめるとして……

```
0040 '--- タマ
0041 @MYSHOT
```

……オイちょっと待て、画面タッチってイガイとまだ考えたコトなかったぞ！

それもそうだったね。
画面タッチにはシステム変数TCHSTを使うよ。

```
0042 IF TCHST==TRUE THEN ???
```

TCHST==TRUEだとタッチされている、TCHST==FALSEだとタッチされてない、というイミさ。

TRUEは「アリ」、FALSEは「ナシ」だったね、そういえば。

このシステム変数TCHSTという名前はタッチのジョウタイ、つまり「タッチ（Touch）・ステータス（Status）」のリャクじゃな。

どうもシステム変数ってヤツは、フツウの変数と見た目がイッショでワカりづらいぜェ……。

たしかに一見、区別がないからねえ。説明書のシステム変数の表を見てみると、全部でこれだけだから、丸暗記するのがイチバンなのかなあ。

これればっかりは、しかたないかのう。システム変数はどれもなかなかベンリなモノなので、なれておくのもイイじゃろう。

話をもどすと、システム変数ＴＣＨＳＴでタッチされたかどうか調べるんだったよね。

```
0042 IF TCHST==TRUE THEN VY=-1
```

あっ、これだとずっとタッチしていないとショットが動かなくなっちゃうか……。

プレイヤー機のプログラムそのままじゃダメってコトかよ……応用問題とか、オレのキライな話だぜ！

プレイヤー機は「止まっている」「１文字ぶん動く」の２つだけでプログラムできたけど、ショットには３つのパターンが考えられるね。

A	**発射した**	タッチされてすぐ	プレイヤー機の上に出る
B	**ショット移動中**	発射されてから消えるまで	１文字分ずつ上に上がる
C	**何もない**	ショットが画面中にない	スキップ

フーム……あいかわらずワカるような、ワカんねェような……。
とりあえず、一番カンタンなＣの「何もない」だけ作るゼ！

```
0040 '--- タマ
0041 @MYSHOT
0042 IF TCHST==FALSE THEN @MSSKIP
0043 '---
0044 @MSSKIP
0045 RETURN
```

ＭiSsile　　ＳＫＩＰ
@ＭＳＳＫＩＰってラベル名は、「ミサイルをスキップする」ってコトだぜ。

ミサイルとかそういう英語はよく分かるんだね。それにしても、ホントにカンタンなトコだけ……。

まあ分かるトコロはさっさと片付けるというのも、プログラム作りのテクニックのひとつじゃよ。
次にカンタンなのはＡの「発射した」じゃが……

それって、こういうこと？

```
0040 '--- タマ
0041 @MYSHOT
0042 IF TCHST==FALSE THEN @MSSKIP
0043 MX=PX:MY=PY
0044 MY=MY-1
0045 LOCATE MX,MY:PRINT"・"
0046 '---
0047 @MSSKIP
0048 RETURN
```

Ｍissile
とりあえずミサイルの場所ってことで、43行目に変数ＭＸとＭＹを用意したよ！

42行目のIF文で「タッチしていないとき」は@ＭＳＳＫＩＰに行くことになってるから、43行目から先はかならず「タッチされているとき」になるね。
だから、ココから書いていくのはリクツに合ってるよ。

プレイヤー機の場所（ＰＸとＰＹ）のすぐ上ってことは、ヨコ方向はＭＸ＝ＰＸで、タテ方向はＭＹ＝ＰＹにした後ＭＹ＝ＭＹ－１でいいよね。

で、45行目でソコにショットの"・"を出してるワケだ。合ってるんじゃねェか？

うーん、でもコレだと何回でも撃てちゃうな。本当はショットが出てる間はもう撃てないはずなんだけど……。

ウム、なにがマズいのかよくワカっておる。ダイジなことじゃぞい。
ショットは敵に当たって消えることもあるしのう。どうやって「A.発射した」と「B.ショット移動中」の区別をつけるかとカンガエると、ムズカシいのう？

ここは、フラグを使いましょう。

ン？　フラグ？

フラグを立ててみよう

フラグというのは旗(はた)のことじゃが、プログラム用語ではもっとトクシュなイミになるのじゃ。

ゲームでもときどき「フラグを立てる」って言い方をするよね。あれのこと？

だいたい同じイミだよ。ジッサイに見てもらうのが早いかな。
まず変数をひとつ用意しておくね。ミサイル(Missile)・ステータス(Status)で**ＭＳＴ**という名前にしておこう。

```
0004 MST=1
```

スガスガしいほど、ナニをしたいのかワカらねえぜ！

だと思ったから、ボクらにワカるルールを作っておこう。変数**ＭＳＴ**が**１**のときは「ショットが画面上にある」、変数**ＭＳＴ**が**０**のときは「ショットが画面から消えている」、ということにするよ。

自分ルールかよ。マスマスややこしいんじゃねェのか？

これがフラグというものさ。もう少し進むと、わかってくるハズだよ。

```
0041 @MYSHOT
0042 IF TCHST==FALSE THEN @MSSKIP
0043 MST=1
0044 MX=PX:MY=PY
0045 MY=MY-1
0046 IF MY<3 THEN MST=0:GOTO @MSSKIP
0047 LOCATE MX,MY:PRINT"・"
```

46行目にIF文を足して、ショットのタテ位置（変数**ＭＹ**）が３より小さくなったら**ＭＳＴ＝０**にすることにしたよ。

画面の上の方まで行ったら、ショットが消えるから**ＭＳＴ＝０**にするんだね。

ついでに**＠ＭＳＳＫＩＰ**に行くのもナットクだぜ。画面の上までショットを**ＰＲＩＮＴ**してもシカタねえもんな。

ココから一気に書きたすよ。

```
0041 @MYSHOT
0042 IF MST==1 THEN @MSMOVE
0043 IF TCHST==FALSE THEN @MSSKIP
0044 MST=1
0045 MX=PX:MY=PY
0046 GOTO @MSMOVE2
0047 '---
```

```
@MSMOVE
LOCATE MX,MY:PRINT"  "
'---
@MSMOVE2
MY=MY-1
IF MY<3 THEN MST=0:GOTO @MSSKIP
LOCATE MX,MY:PRINT"・"
```

ラベルに使った**MOVE**というのは「動く」というイミの「MOVE（ムーブ）」だよ。

オイオイ、イキナリプログラムが増えすぎてついてけねェぞ！

まずは、さっきの表をもういちど見てみよう。

A	**発射した**	タッチされてすぐ	プレイヤー機の上に出る
B	**ショット移動中**	発射されてから消えるまで	1文字分ずつ上に上がる
C	**何もない**	ショットが画面中にない	スキップ

たとえば、フラグの変数**MST**が**0**のとき（自分ルールで「ショットが画面から消えている」）に、画面をタッチするとどうなるかな？

そりゃトウゼン「A.発射した」じゃねーか！　……ン？　トウゼンっていえばトウゼンだが、**MST**が**0**のトキは、「B.ショット移動中」にはならねェわけだな……そういえば。

それがフラグのベンリなところだね。「変数**MST**が**0**のときはショットが画面から消えている」という自分ルールが決めてあるから、ぐっと区別がラクになっているんだ。

なるほどね。「A.発射した」のパターンだと、まず42行目と43行目のIF文でひっかからないから、そのままリストを下に進んで……ショットを発射するから、変数**MST**を**1**にしてるんだ！

そのあとプレイヤー機の上にショットを書いている方法は、もうわかるよね。あいだに**GOTO**をはさんでいるけど、やってるコトはほとんど今までと同じさ。

となるとモンダイは、「B.ショット移動中」のパターンか？

それはフラグの変数**MST**が**1**のとき、だよね。プログラムをなぞってごらん。

えぇと、42行目ですぐに**@MSMOVE**に飛んで、画面に書いてあるショットを**"　"**で消しちゃうのか。

そのまま**MY=MY-1**して、**PRINT"・"**するから……ショットが上に上がるワケだな。ここはプレイヤー機を動かすヤリ方と似てるからワカる気がするぜ。

だいたいフラグの使い方は見えてきたようじゃの。変数に入っている数字でジョウタイをチェックするのが、おおざっぱなフラグの考え方じゃ。
数字は自分ルールでかまわんが、イッパンテキには**1**だと「アリ」、**0**だと「ナシ」で、フラグが**1**になることを「フラグ（旗）が立つ」と言うのじゃな。

遠くから土地を見て、旗が立っているかどうかでそこに何があるのかチェックする、というイメージなんだろうね。
ココを読んでるみんなはフラグについてわかったかな？　変数**MST**が**0**のときと**1**のときで、どう動きが変わるか注意して読み直してみるとわかりやすいよ！

ム……なんだかキョウイクテキすぎる言い方が気にいらねェが、ゲーム用語になってるくらいだし、ダイジなコトなのはたしかか……。

（その通りだけど、ゲーム用語かどうかがキジュンなのかなあ……）

じゃあショットのプログラムができたことだし、ココまでのプログラムをまとめてセイリしておこうか。

メインループに入るより先に、13行目で変数を用意しておいたよ。

プログラムリストを表示


ン？　0じゃなくてFALSEなのか？　文字変数……ってワケじゃねェよな。


ホントだ。今までフラグに使っていた変数MSTにはゼンブ1や0じゃなくてTRUEやFALSEが入っているよ。


まとめるついでに、そこだけ変えてみたんだけど、わかるかな？
変数にFALSEと入れると0、TRUEと入れると1のイミになるんだ。


……ん？


たとえばMST=FALSEと書いたあとにPRINT　MSTすると、0って出るんだけど……


オイ！　それじゃハジメからMST=0って書けばいいじゃねェか！


ふぉっふぉっふぉ。リクツで言えばそうかもしれんが、MST=FALSEと書いたほうがハッキリ「アリ」か「ナシ」かワカりやすいじゃろう？


また「アリ」（TRUE）「ナシ」（FALSE）のハナシかよ……。


ビット演算のトキにも話したが、もともとコンピューターはON / OFFのアリ・ナシの組み合わせで作られたものじゃからな。このテのコトが多いのも、トウゼンなのじゃよ。

今回のまとめ

TCHST変数

システム変数TCHSTには、常に画面タッチされているかどうかの情報が入りつづけます。0（FALSE）ならタッチされていなく、1（TRUE）ならタッチされています。

```
A=TCHST
```

変数Aに画面タッチされたかどうかが数字で入ります。

```
IF　TCHST==TRUE　THEN　（命令）
```

画面タッチされていれば、命令を実行します。

フラグを使うとは

おおざっぱに言えば、自分ルールにしたがって決まった数を変数に入れるようにすれば、それがフラグです。
たとえばゲーム開始時に変数A=0として、「プレイヤーがカギを取ったら変数A=1にする」というルールを決めてプログラムしておけば、ドアの前でIF　Aで調べるとドアを開くべきかどうかすぐにわかりようになります。このとき変数Aが1になるのを「フラグが立つ」と言うこともあります。

サンプルプログラム７ (その２)でできたリスト

```
01 '
02 ' ┌──────┐
03 ' │SAMPLE7 │
04 ' │ シンリャクミスイ │
05 ' └──────┘
06 VISIBLE 1,1,0,0,0,0
07 CLS:COLOR 0
08 CLEAR
09 'DIM ???
10
11 '---
12 PX=10:PY=21
13 LOCATE PX,PY:PRINT"Ｍ";
14 MX=PX:MY=PY:MST=FALSE
15 LOCATE 0,22
16 PRINT"━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━";
17
18 @LOOP
19  GOSUB @MYSHIP
20  GOSUB @MYSHOT
21  VSYNC 1
22 GOTO @LOOP
23
24 '--- ｼﾞﾌﾞﾝ
25 @MYSHIP
26 B=BUTTON():VX=0
27 IF B AND 4 THEN VX=-1
28 IF B AND 8 THEN VX=1
29 IF VX==0 THEN @NOTMOVE
30 LOCATE PX,PY:PRINT" "
31 PX=PX+VX
32 IF PX<0 THEN PX=0
33 IF PX>31 THEN PX=31
34 LOCATE PX,PY:PRINT"Ｍ";
35 '---
36 @NOTMOVE
37 RETURN
38 '--- ﾀﾏ
39 @MYSHOT
40 IF MST==TRUE THEN @MSMOVE
41 IF TCHST==FALSE THEN @MSSKIP
42 MST=TRUE
43 MX=PX:MY=PY
44 GOTO @MSMOVE2
45 '---
46 @MSMOVE
47 LOCATE MX,MY:PRINT" "
48 '---
49 @MSMOVE2
50 MY=MY-1
51 IF MY<3 THEN MST=FALSE:GOTO @MSSKIP
52 LOCATE MX,MY:PRINT"･"
53 '---
54 @MSSKIP
55 RETURN
```

サンプルプログラム７（その３）

プログラマーからひとこと

サンプルプログラム７を見ていくのも、とうとう３回目です。
全４回でまだ１回分ありますが、今回がクライマックスです。
なぜならページの長さがいちばん長いからです。
ボス戦前のダンジョンとかこんな感じですよね。

今回は１章ごとにけっこうドンドン進みます。
そのスピードにおいていかれることもあるかもしれませんが、そんな時ははじめに戻って、きっちりわかったあたりで次に進むくらいがいいかもしれません。
なんだか勉強っぽくてイヤかもしれませんが、なにしろラストダンジョンです。
ワンパク君たちが迷うところではいっしょに迷いつつ、いっしょに戦っていきましょう。
ところで、「ワンパク君」とかあたりまえのように呼んでますが、あれ、本名なの？

敵を出してみよう

ここまででプレイヤー側の動きはだいたいできあがったね。いよいよ敵の動きのプログラムに入ってみようか！

今までよりも大変そうなヨカン……。

ははは。たしかにタクサンの敵がジグザグに動き回るから、これまでよりもフクザツではあるね。
最初はカンタンに、スタートしたときのならび方から始めようか。

カンタンって言っても、こういうワケじゃねェよな？

```
0010 LOCATE 0,3
0011 PRINT "👾 👾 👾 👾 👾 👾 👾 👾 👾 👾"
0012 LOCATE 0,5
0013 PRINT "👾 👾 👾 👾 👾 👾 👾 👾 👾 👾"
```

サスガにココまでカンタンじゃねェとは、オレにもワカるぜ。

たしかに、そうかも。サンプルをＲＵＮしてみると、敵👾は１匹ずつ順に動いてるんだよね。ショットでやられたらその敵だけが消えるわけだし……。

つまり１匹ずつ、あわせて20匹ぶんの変数がいるわけだね。……こう言えば、今までにおぼえたコトが役に立つんじゃないかな？

そうか！　20個の変数をヨウイするんだな！

い、いや……

……じゃなくって、配列変数を使うんだよね。

そうそう、ソコだよ！

```
0014 DIM EX(20):DIM EY(20):DIM ED(20)
```

ゲームで使うのはこんなところかな。敵は英語で「Enemy（エネミー）」だから、変数の頭文字には「E」をつけることにするよ。

EXとEYはやっぱり、ヨコ（X）とタテ（Y）かな？

そうそう。ヨコがXでタテがYという変数の名付け方は、はっきりルールがあるワケじゃないけど、プログラマーの人はたいていこう書くね。おぼえておいていいんじゃないかな。
変数EDの「D」は「Dynamics（ダイナミクス）」のDなんだけど、……まあ、「動き方」くらいに考えておけばいいかな。

やけにハッキリしない言い方じゃねェか。ナニか言いたくないイミでもあんのか!?

「力学」とか「変動」とか……

おベンキョウの話だったか、キタイはずれだぜ！

ま、まあ分かりにくいのは本当かな。とにかく変数EDは敵がどう動くかのフラグにするってことだね？

そういうコト。とりあえず最初は0にしておいていいかな。

ネンのタメに、ここで配列変数をどう使うのかまとめておこうかのう。
たとえば１匹めの敵は配列の番号を(0)に統一するのじゃ。こやつのいる場所を調べたければ、横方向なら配列変数EX(0)、縦方向ならEY(0)じゃな。同じように２匹めの敵なら、EX(1)とEY(1)で場所が決まっているワケじゃ。
それとタダの書き方のモンダイじゃが、

```
0000 DIM EX(20):DIM EY(20):DIM ED(20)
```

というふうにコロン（:）で分けなくても、

```
0000 DIM EX(20),EY(20),ED(20)
```

と、ひとつのDIM命令でスッキリ書くこともできるぞい。

じゃあ、その３つの配列変数を使いながら敵をナラベてみるゼ！

```
0011 FOR I=0 TO 9
0012   EX(I)=I*2
0013   EY(I)=3
0013   ED(I)=0
0014   LOCATE X(I),Y(I)
0015   PRINT "■";
0016 NEXT
0017 FOR I=10 TO 19
0018   EX(I)=I*2-20
0019   EY(I)=5
0020   ED(I)=0
0021   LOCATE X(I),Y(I)
0022   PRINT "■";
0023 NEXT
```

……け、けっこうメンドくせェな、コレ！

イヤイヤ、なかなかジョウデキじゃぞい。トクに18行目なんかはクロウのアトがみえるのう。

敵が２列にならんでいるから、FOR文を２回に分けて作らないといけなかったんだね。
たしかに「FOR I=0 TO 19」ひとつで終わらせる方法ってないもんなあ。IF文で変数Iが10を超えたら……いやいや、それもプログラムが長くなっちゃう……

このタメだけに似たような行を２つ書くのがなんたってメンドクセェぜ！

実は、すごく短くする方法があるんだけど……

な、ナニ!?　キキズテならねえな！

短かくしてみよう

さっきのワンパク君のプログラムでも必要なコトはできてるから、わざわざ短くしなくてもいいんだけど……

イヤイヤ、さっきのプログラムを打つのはかなりメンドウだったんだぜ。短くなるならカンゲイじゃねェか？

ならば、こう書くとかなり短くなるのう。

```
FOR I=0 TO 19
 EX(I)=FLOOR(I%10)*2
 EY(I)=FLOOR(I/10)*2+3
 ED(I)=0
 LOCATE EX(I),EY(I)
 PRINT "☻"
NEXT
```

ハ……ハァ!?

たしかに短いけど……よくわからない……

そうか、まだ習ってねェ**FLOOR**がカギだな！　これにスゲェ意味が……

FLOORは「切り捨てて整数にする」命令だよ。たとえば**A=FLOOR(1.75)**と書けば変数**A**には1.75から「.75」を切り捨てて**1**が入るね。

ふ、フツウだぜ！

「FLOOR」すなわち「たいらにする」というイミじゃから、整数にするというイミに変わったのじゃな。
12行目の計算をカンタンに言いなおすと、「変数**I**を10で割ったあまりから小数点以下を切りすてて２倍する」となるのう。

ぜんぜんカンタンじゃねー！

うーん、なるほど……たしかに**EX(0)=0**、**EX(1)=2**、**EX(2)=4**と２ずつ増えていって、**EX(10)**になったとたん**EX(10)=0**に戻っているね。やりたかったコト通りではあるんだけど……なんだかフシギだなあ。

13行目も、「**I**を10で割って小数点以下を切りすてたら、２倍して３を足す」だけじゃがこれで敵の１～10匹めまではタテ位置**3**行目に、11～20匹めまではタテ位置**5**行目にそろうというワケじゃ。

ウ……ウウ……。ケイサンしてみたらタシカにその通りだったが……セイカツの中でまったく出てこねェ式にもホドがあるぜ！

あんまりパッと出てくる計算じゃないかもね。
でも「**FLOOR(I/N)**なら**I**が『**N**の倍数』になるたび１増える」。
「**FLOOR(I%N)**なら**I**が『**N**の倍数』になるたびゼロまでおり返しながら１増えていく」。
原理まで考えなくても、こう丸アンキしておくとプログラムで使えるバメンはけっこう多いよ。

たとえば時計を考えてみい。１時間は60分で、０分〜59分をくり返すものじゃが、この「分」は`FLOOR(I%60)`で書きあらわせるのう。「時間」は`FLOOR(I/60)`と書けばいいワケじゃ。
かなりカタいたとえになってしまったが、サンプルプログラムのように、敵をセイレツさせるようなコトにも応用できるんじゃぞい。

プログラマーの生活のチエってところかな。

そういうコト。そうそう、ここでは「`FLOOR(I%N)`」なんて書いたけど、本当は「割り算のあまり」に少数が出てくるハズはないから、「`I%N`」と書くだけでいいんだよ。

ワシは……ワシは、見た目をそろえておきたかったのじゃあああああ

敵を動かそう

敵をならべた後は、イヨイヨ動かすワケだが……

今までとちがって１方向や２方向じゃないのが、ナヤむところだね。

迷ったら、まずセイリしてみよう！

0	右に動く	X=31になるまでX+1
1	下におりる	1回だけY+1
2	左に動く	X=0になるまでX-1
3	下におりる	1回だけY+1

キホン的には、このくり返しだよね。

そう言われれば、そう……なのか？

ショットで撃たれたり、下までおりて上にもどるのが抜けてるけど……

そこはもっとアトから考えよう！　まずこの表にある動きをクリアーしないとそこまでたどりつかないしね。

そのワリキリ、キライじゃねえな！
じゃあとにかくサブルーチン「ALIEN（エイリアン）」をメインルーチンから呼び出すことにするぜ！

```
00EE GOSUB @ALIEN
```

で、アトは……１匹ずつ動かすコトだけ考えるぜ！

```
00E0 '--- シンリャクシャ
00E1 @ALIEN
00E2 FOR I=0 TO 19
00E3  LOCATE EX(I),EY(I):PRINT" "
00E4  GOSUB @EMOVE
00E5 NEXT
00E6 RETURN
```

動かすトキのキホンだから、"　" で消すだけは消しといたぜ。
こっからサキはナガくなりそうなんで`@EMOVE`に`GOSUB`させといたが、モンダイはどの方向に動かすか……だな。

なかなかワルくないよ。飛んだ先でまた別のラベルに飛ぶけど、こうすると分かりやすいんじゃないかな。

```
00E8 @EMOVE
00E9 ON ED(I) GOTO @RI,@DW,@LF,@DW
```

ED(I)=0	右に動く	X=31になるまでX+1
ED(I)=1	下におりる	1回だけY+1
ED(I)=2	左に動く	X=0になるまでX-1
ED(I)=3	下におりる	1回だけY+1

この表のとおりだね。フラグに使ってる変数**ED(I)**の数しだいで、ラベル**@RI**、**@DW**、**@LF**、**@DW**のどれかに飛ぶってやり方さ。

ああそうか、ラベルはそれぞれ「右(RIght)」「下(DoWn)」「左(LeFt)」になってるんだ。

とりあえず、右に動かすコトだけ考えるゼ！

```
00910 @RI
00920 EX(I)=EX(I)+1
00930 LOCATE EX(I),EY(I)
00940 PRINT "☻"
01000 RETURN
```

コレで……まあ、右にだけはウゴくだろうな。

ずいぶんドンドン行くなあ。
でも、のこりの方向も同じように動かせそうだね。

```
01020 @DW
01030 EY(I)=EY(I)+1
01040 LOCATE EX(I),EY(I)
01050 PRINT "☻"
01060 RETURN
```

やってるコトがカブる行が多いから、ゼンブをまとめるとこうなるかな。

```
00910 @RI
00920 EX(I)=EX(I)+1
00930 GOTO @EPUT
00940
00950 @DW
00960 EY(I)=EY(I)+1
00970 GOTO @EPUT
00980
00990 @LF
01000 EX(I)=EX(I)-1
01010 GOTO @EPUT
01020
01030 @EPUT
01040 LOCATE EX(I),EY(I)
01050 PRINT "☻"
01060 RETURN
```

ラベル名の「PUT(プット)」は日本語では「置く」というイミになるかな。敵キャラ(E)を画面に置くルーチンってことだね。

ココまではよし。……というか、まだ右に行くコトしかデキてねェな。フラグが**ED(I)=0**のまんまだからな。

いま**RUN**しても、右ハジまで来たら表示がおかしくなるよ。

マカセとけ！　たとえば**@RI**なら、右ハジに来たときだけフラグを変えてやりゃいいんだな？

```
00910 @RI
00920 EX(I)=EX(I)+1
00930 IF EX(I)<31 THEN @EPUT
```

```
00094 ED(I)=1
00095 GOTO @EPUT
```

てコトは、@LFでも同じようにできるハズだぜ！

```
01001 @LF
01002 EX(I)=EX(I)-1
01003 IF EX(I)>0 THEN @EPUT
01004 ED(I)=3
01005 GOTO @EPUT
```

ふむふむ。

ED(I)=0	右に動く	X=31になるまでX+1	@RI
ED(I)=1	下におりる	1回だけY+1	@DW
ED(I)=2	左に動く	X=0になるまでX-1	@LF
ED(I)=3	下におりる	1回だけY+1	@DW

フラグED(I)が1や3になると、次はラベル@DWに行くことになるね。

@DWでは下におりるのは１回だけだから、すぐフラグを変えることになるね。

```
00097 @DW
00098 EY(I)=EY(I)+1
00099 IF ED(I)==1 THEN ED(I)=2
01000 IF ED(I)==3 THEN ED(I)=0
01001 GOTO @EPUT
```

こうやってIF文を使うことになるのかな？

ン？　……ああそうか、さっきまで右に動いていたとき（ED(I)==1）と、さっきまで左に動いていたとき（ED(I)==3）で、次に立てるフラグがカワッてくんのか。

せっかくだから、今までででおぼえたコトを使って、もっとシンプルにしてみようか。

```
00097 @DW
00098 EY(I)=EY(I)+1
00099 ED(I)=(ED(I)+1) AND 3
01000 GOTO @EPUT
```

……ナニ？

あ、ビット演算かあ……。

ケッ！　オリコウさんドモはいつもこうだ！　ウーム、2のトキは２進数で10で、3は２進数で11だから……ブツブツ……

えーと、ようするに99行ではED(I)のナカミが１だけふえるけど、4になったときだけは0にもどされるってコトだよね？

ワリとカンタンなんだけどなあ。

デキるヤツの考えのオシツケはマッピラだぜ！

ヒドい言われようだよ。まあ、ビット演算の話はもっと後からくわしくおぼえることにしようか。
そうそう、画面の下まで行ったらまた上から出てくる動きも足しておこうね。

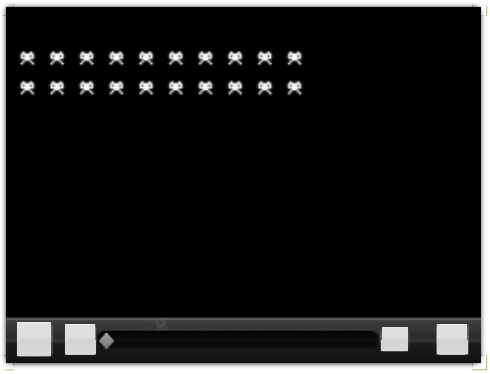
```
IF EY(I)>20 THEN EY(I)=3
```

ナルホドな。コレでとりあえず、敵の動きはひととおりデキたか……。
ずいぶんトントンビョウシじゃねェか。**RUN**してみるか！

……なんか、ちがう……

ウオォー！　ナンだこの動き、ナンかチガうぞコレ、っつかキモチワリー！　なんかキモチワリー！

ふぉっふぉっふぉ。ようやく気づいたようじゃのう。めざしていた正しい動きはこっちじゃよ。



そうだコレだ。敵は下におりるトキは、１匹ずつじゃなくマトメて下におりるんだ。チクショウ、どこでマチガエたんだ!?

ここまでにミスがあったワケじゃないよ。まだそういう動きをプログラムしてないっていうだけさ。

足りないところをつけ足してみよう

ちょっとアタマをヒネるところだよ。いいかい？
今までは「１匹ずつ、右ハジ・左ハジに来たら下に行く」ことにしていたね。

うん、そのハズ。……そうだよね？

ムシロ、そのドコがワルいのかがワカらねーぜ！

ところが、ボクらがやろうとしてたのは「どれか１匹が右ハジ・左ハジに来たら、次のターンでは敵全部がその場から下に行く」動きなんだよ。

……！

つまりアレか、今までやってたのは行列を作って歩く「ムカデ歩き」の動きで、本当にやりたいのは行進の「右向け、右！」でゼンインがいつでも同じ方向をむく動きってコトか？

わりとヨウチじゃが、悪くないたとえじゃぞい。ココまでわかれば、アトはできるんではないかの？

待て待て待て、やるべきコトはだいたいワカったが、けっこうコンランするぜ！

急にハッソウを変えるところだからね。順にかたづけていこうか。

まず、方向を変えるきっかけは「どれか１匹が右ハジ・左ハジに来たら」だよね。ここから考えてみようよ。

それは、ツマリ……@ＲＩや@ＬＦの中の、IF文にひっかからなかったトキじゃねェか！

```
00191 @RI
00192 EX(I)=EX(I)+1
00193 IF EX(I)<31 THEN @EPUT
00194 ED(I)=1
00195 GOTO @EPUT
```

この94行目のあたりだな！

そうだね。ここでＥＤ(Ｉ)のフラグを変えてるけど、これだと(Ｉ)に入った１匹ずつ、右ハジにつくまでフラグが変わらないね。

ソレじゃマズいんだよな……そのカワリに、ゼンインのフラグを変える方法……？

もう１つ、大きなフラグを作ったらいいんじゃないかな！

```
00194 EDW=TRUE
```

94行目を新しい変数ＥＤＷのフラグを立てるように書きかえたよ。
このフラグが立ってる時は、どこにいる敵でもフラグＥＤ(Ｉ)を下におりるように変えたらどうかな？

フラグＥＤ(Ｉ)を変えるフラグがＥＤＷか……ドンドンややこしくなってくるぜェ……。

ここがサイゴのカベだよ、がんばろう！

画面のハジに来たことはＥＤＷで分かるようになったから、「次のターンで敵ぜんぶがその場から下に行く」動きをプログラムすればいいんだよね。
さっきのビット演算ＥＤ(Ｉ)=(ＥＤ(Ｉ)+１) ＡＮＤ ３を使えばそれでいいかな。

ムムム……。次のターンってことは、だいぶ前にモドるがこのあたりか。

```
00180 '--- シンリャクシャ
00181 @ALIEN
00182 FOR I=0 TO 19
00183  LOCATE EX(I),EY(I):PRINT" "
00184  GOSUB @EMOVE
00185 NEXT
00186 RETURN
```

ココのFOR～NEXTで敵を動かしてたハズだぜ！

いちど敵のターンが終わってからＥＤ(Ｉ)のフラグを変えるわけだから……アレ？　フラグを変えていい場所がないよ!?

ナニ言ってんだ……オ、オォ!?
なんてこった、タシカにそうだぜ！　83行目のアトでフラグを変えてもシッパイだし、85行目のアトじゃFOR～NEXTループが終わってるからＥＤ(Ｉ)って書いてもナンにもならねェぞ！

ソコがモンダイだね。フラグＥＤＷが立つのは敵のターンのトチュウだから、まず敵のターンが終わるまで待たなきゃ

いけない。でも敵のターンが終わったFOR～NEXTの外だとED(I)を変えるコトができない。

ハマリじゃねェか！　どうしようもねえ！

うーん、FOR～NEXTが終わったあとに、もう１回こういうFOR～NEXTを置くのはどう？

```
00087 FOR I=0 TO 19
00088  IF EDW THEN ED(I)=(ED(I)+1) AND 3
00089 NEXT
00090 EDW=FALSE
```

ED(I)を変えるためだけのFOR～NEXTなんだけど……

やるじゃねェか。コレでバッチリに見えるぜ？

なかなかワルくないぞい。しかし、似たようなFOR～NEXTを２回ツカうのはカイゼンのヨチありじゃ。

？

このテイドの長さなら気にならんじゃろうが、FOR～NEXT文は増えれば増えるほど動作に時間がかかってしまうモノなのじゃ。ミライあるショクンには、これでマンゾクせずにFOR～NEXTの少ないプログラムをメザしてもらいたいのう。

ムズカしいんだなあ。

そうなるとマスマスどうすりゃイイのかワカらねえじゃねーか！

ここは……もう１つフラグを使うことにしようか。

ED(I)	敵１匹ずつ(I)の動きを決めるフラグ
EDW	次のターンで敵ぜんぶが下におりるフラグ
MD	今のターンで敵１匹ずつ(I)下におりるフラグ

この表のとおり、モンダイの動きをコントロールするための変数MDを使ってみるよ。

フラグMDが１匹ずつ動かすのはイイのか……？　ああ、１匹ずつと言っても、FOR～NEXTのターンまるごとMDフラグで動かしてるから、ケッキョク敵ゼンブまとめて動いてるのか……
マジでヤヤコシイな、ついてこれてるのか自分でもワカらねーぜ！

プログラムとしては、こういうことになるね。よく見直してみよう。

```
00080 '--- シンリャクシャ
00081 @ALIEN
00082 MD=FALSE
00083 IF EDW THEN MD=EDW:EDW=FALSE
00084 '---
00085 FOR I=0 TO 19
00086  LOCATE EX(I),EY(I):PRINT" "
00087  IF MD==FALSE THEN @EMV
00088  '---
00089  ED(I)=(ED(I)+1) AND 3
00090  '---
00091  @EMV
00092  GOSUB @EMOVE
00093 NEXT
00094 RETURN
```

EDWフラグのことはFOR～NEXTの外でかたづけて、ジッサイにED(I)を変えるのはMDフラグにマカせてあるのさ。

83行目でＥＤＷフラグのナカミをＭＤフラグに引きつがせて、ＥＤＷフラグじたいは初期化してるんだね。

ココまでのプログラムを全部書くと、こんな感じかな。

プログラムリストを表示

ナンドでも言うが、ヤヤコシイぜ！

ひとつずつモンダイをカイケツしていくうちに、いつしかこうなってしまったワケじゃが、だれにでも見やすいプログラムかといえばギモンかもしれん。
全てがアタマに入ったところで、あらためて見やすいプログラムに書きなおすのもダイゴミじゃぞ。今すぐやるべきコトではないかもしれんが……

この講座がゼンブ終わったあとで、みんなやってみてね！

マル投げのフンイキがプンプンにおうぜ！

今回のまとめ

配列宣言をいちどにすませる

```
DIM A(10),B(10),C(10)
```

ＤＩＭの後に，で区切って配列宣言を並べることができます。

FLOOR命令

```
（変数）=FLOOR(数)
```

数から小数点以下を切り捨てて整数にします。

```
FOR I=0 TO 100
 A(I)=FLOOR(I/5)
NEXT
```

こう書くと、変数Ａ（Ｉ）に入る数は、変数Ｉが「５の倍数」になるたび１増えます。
Ａ（０）から順に、0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,……と入っていくわけです。

```
FOR I=0 TO 100
 A(I)=I%5
NEXT
```

こう書けば、変数Ａ（Ｉ）に入る数は１ずつ増えていき、それが「５の倍数」になるたび０に戻ります。
Ａ（０）から順に、0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,……と入っていくわけです。

サンプルプログラム７ (その３)でできたリスト

```
'
' ┌──────┐
' │SAMPLE7 │
' │ シンリャクミスイ │
' └──────┘
'
VISIBLE 1,1,0,0,0,0
CLS:COLOR 0
CLEAR
DIM EX(20),EY(20),ED(20)

'---
EMAX=20
FOR I=0 TO EMAX-1
 EX(I)=FLOOR(I%10)*2
 EY(I)=FLOOR(I/10)*2+3
 ED(I)=0
 LOCATE EX(I),EY(I)
 PRINT "☠"
NEXT
EDW=FALSE
'---
PX=10:PY=21
LOCATE PX,PY:PRINT"Ѧ";
MX=PX:MY=PY:MST=FALSE
LOCATE 0,22
PRINT"━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━";

@LOOP
 GOSUB @MYSHIP
 GOSUB @MYSHOT
 GOSUB @ALIEN
 VSYNC 1
GOTO @LOOP

'--- ｼﾞﾌﾞﾝ
@MYSHIP
B=BUTTON():VX=0
IF B AND 4 THEN VX=-1
IF B AND 8 THEN VX=1
IF VX==0 THEN @NOTMOVE
LOCATE PX,PY:PRINT" "
PX=PX+VX
IF PX<0 THEN PX=0
IF PX>31 THEN PX=31
LOCATE PX,PY:PRINT"Ѧ";
'---
@NOTMOVE
RETURN
'--- ﾀﾏ
@MYSHOT
IF MST==TRUE THEN @MSMOVE
IF TCHST==FALSE THEN @MSSKIP
MST=TRUE
MX=PX:MY=PY
GOTO @MSMOVE2
'---
@MSMOVE
LOCATE MX,MY:PRINT" "
'---
@MSMOVE2
MY=MY-1
IF MY<3 THEN MST=FALSE:GOTO @MSSKIP
LOCATE MX,MY:PRINT"･"
'---
@MSSKIP
RETURN

'--- ｼﾝﾘｬｸｼｬ
@ALIEN
MD=FALSE
IF EDW THEN MD=EDW:EDW=FALSE
'---
FOR I=0 TO EMAX-1
 LOCATE EX(I),EY(I):PRINT" "
 IF MD==FALSE THEN @EMV
 '---
 ED(I)=(ED(I)+1) AND 3
```

```
  '---
  @EMV
  GOSUB @EMOVE
NEXT
RETURN
'---
@EMOVE
ON ED(I) GOTO @RI,@DW,@LF,@DW

@DW
EY(I)=EY(I)+1
IF EY(I)>20 THEN EY(I)=3
ED(I)=(ED(I)+1) AND 3
GOTO @EPUT

@RI
EX(I)=EX(I)+1
IF EX(I)<31 THEN @EPUT
EDW=TRUE
GOTO @EPUT

@LF
EX(I)=EX(I)-1
IF EX(I)>0 THEN @EPUT
EDW=TRUE

@EPUT
LOCATE EX(I),EY(I)
PRINT "☠"
RETURN
```

サンプルプログラム7（その4）

プログラマーからひとこと

ついにサンプルプログラムを見ていく旅もここで終わりです。
RPGなら光とか青空とか出てくるムービーが始まるころですが、まだ今回もサンプルプログラムがゲームとして完成してはいないので、青空がふいにくもって真のラスボスが出てくる場面と言ったほうがいいでしょうか。
そこはどうだっていいですね。

今回はゲーム用語のいわゆる「当たり判定」から始まります。
思えばゲームっぽい話がたくさん出てくるサンプルプログラムでした。
あなたのプログラムのウデもかなり上がったんじゃないでしょうか。
これが終わったら、自分でカンタンなゲームを作ってみるのもいいかもしれません。

いまのは死亡フラグみたいなセリフでしたが、フラグのことも学んだあなたなら、このていどのフラグはFALSEにできるはずです。

当り判定を付けよう

いままでのプログラムリストをゼンブまとめてみると、こうだったよね。

プログラムリストを表示

変数ＥＭＡＸを敵の数に使うようになってるんだね。

敵の数をいつ変えたくなってもいいワケじゃな。前にもちらっと話したが、変数にできるトコロは変数にしておくのはダイジな心くばりじゃ。

まだデキてねェところは……ショットが敵に当たったトキ、か。

プログラムでショットが敵に当たるというのは、どういうコトかを考えるとわかるんじゃないかな？

うーん、ショットの場所と、敵の場所が重なったとき……かな？

ショットの場所は変数ＭＸとＭＹだったよな……。敵の場所はＥＸ（Ｉ）とＥＹ（Ｉ）だから……。つまり、こんなカンジか!?

```
IF MX==EX(I) AND MY==EY(I) THEN GOSUB @EOUT
```

配列変数（Ｉ）をツカッてるから、敵をウゴかすFOR～NEXTの中に入れないとイケねェな。

CHKCHR()命令

ワシじゃワシじゃ、今回はやたら顔を出すハカセじゃ。
ここでは話のナガレ的に出てこなかったが、当たり判定にベンリなＣＨＫＣＨＲ（）という命令があるのを知っておったかな？
このサンプルにはカンケイない話じゃが、カンタンにセツメイしておくぞい。

CHKCHR()命令

たとえばA=CHKCHR(0,0)と書くと、そのトキ画面の(0,0)のバショにナニが表示されているか変数Aに入るのじゃ。
(0,0)というのはLOCATE命令で使うのと同じヨコ方向とタテ方向の数じゃな。

文字コード

A=CHKCHR(0,0)と書いたトキに変数Aに入るのは、「ASCII(アスキー)文字コード」という数字じゃ。ナンのことかワカらんじゃろうが、ココのいちばん下を見てくれい。左上から順に0、1、2……と数がわりふられているのじゃぞ。今回のサンプルで使っている敵「👾」なら文字コードは6番じゃな。
もし今回のサンプルのように、ショットの場所(MX,MY)に敵「👾（文字コード006）」がいるかどうかチェックしたければ、IF CHKCHR(MX,MY)==006 THEN～とやればいいワケじゃな。

ここまではいい調子じゃないか。敵にショットが当たったコトが分かったら、次は敵がやられるルーチン@EOUTだね。

このサンプルだと、ショットが当たった敵はすぐに消えるから、敵を空白スペースで上書きすればいいのかな。

```
@EOUT
LOCATE EX(I),EY(I):PRINT" "
RETURN
```

タシカにそうだが……ちょっと待てよ。コレだと、次に敵のターンがきたトタンに、消えた敵もまた出てきちまうぜ！

そうか、敵を動かすためにいつも書きなおしていたんだっけ。
いちどショットが当たった敵はPRINTしちゃダメってことだなあ。

これも、いままでの応用でなんとかなるんじゃないかな？

そうか！　フラグを使うんだな！

敵1匹ずつに割りあたってるフラグっていうと、ED(I)だね。

ED(I)=0	右に動く	X=31になるまでX+1	@RI
ED(I)=1	下におりる	1回だけY+1	@DW
ED(I)=2	左に動く	X=0になるまでX-1	@LF
ED(I)=3	下におりる	1回だけY+1	@DW

消えたっていうトクベツなフラグだから、数字には-1をあてはめようか。

ED(I)=-1	消えた	表示しない	@EOUT

110行に追加しておこう。

```
@EOUT
LOCATE EX(I),EY(I):PRINT" "
ED(I)=-1
RETURN
```

これでフラグは立ったから、次は敵の表示のトキにそのフラグを調べるのが……えーと、ドコがいいんだ？

👾を書いて消すなら、FORループの中ならどこでもいいとも言えるけど。でもどうせなら、もう表示しないってくらいがいいだろうね。

じゃあ、最初の方だね。

```
FOR I=0 TO EMAX-1
  IF ED(I)==-1 THEN @PASS
```

```
00E1  @PASS
00E2 NEXT
00E3 RETURN
```

これでED(I)フラグが-1のときは表示もされずに、次の敵まで飛ばされるよ！

待てよ……？　ハハァ、ピンときたぜ！
こうなってくると敵が減れば減るほどごっそりスキップする行が多くなるよな。つまり、敵をタオせばタオすほど、ゲームのスピードが早くなるってワケだな！

ウムウム。どうやらショクンも、プログラムがダイブ身についてきたようじゃな。

負ける気がしねェゼ！

じゃが、せっかくならココで、よりコウリツのいいプログラムにチャレンジしてみんかな？

処理のスキップをおぼえておこう

コウリツときやがったか。プログラムが動けばそれでいいんじゃねェの？

ム……ム。そういう考えもたしかにアリじゃが。
ひとつにはプログラムの動作スピードが早くなるテクニックなので、オボエておけばアトアト「重い」プログラムを動かすのに助かる、というコトはあるぞい。

アクションゲームにある「処理落ち」とか、そういうやつ？

まあそんなトコロじゃな。もうひとつのメリットとしては、プログラムが見やすくなるコトもある、とも言えるのう。

「見やすくなるコトもある」……？

逆に見にくくなるコトも、ときにはあるかも……

えー……

ま、まァまァ！　じゃからこそ、シンプルプログラムのイマのうちにオボエておこうという話じゃよ。なに、たいしてムズカしいコトではないぞい。

ハカセが言いたいのは、たぶんこの部分ですね。

```
00E0  IF MX==EX(I) AND MY==EY(I) THEN GOSUB @EO
UT
```

さすがはインテリ君じゃな。さよう、ソコは「敵がやられたとハッキリ決まったら、やられルーチンに行って帰ってくる。それ以外は普通に進む」となっておる。

それでフツウじゃねえか？

しかし、こうすればどうじゃ。

```
00E0  IF MX!=EX(I) THEN @PASS
00E1  IF MY!=EY(I) THEN @PASS
00E2  '--- ｼﾎﾞｳ
00E3  LOCATE EX(I),EY(I):PRINT" "
00E4  ED(I)=-1
```

```
0085  @PASS
0086 NEXT
```

ン？　……ンンン？　83～84行目は、さっき作ったやられたトキのプログラムだよな。
80行目でMX！＝EX（I）ってコトは、ショットのヨコ位置と敵のヨコ位置がチガうってコトだろ。ツマリ当たってないから、@PASSに飛ぶよな。

81行目のMY！＝EY（I）もタテ位置になっただけで、同じことだね。
そのどっちでもないって時は、かならずショットが当たっているって言えるから、そのままやられたことになる……

さよう。もともとのMX＝＝EX（I） AND MY＝＝EY（I）という条件式は、ハッキリしとるがそれだけチェックに時間がかかるものじゃ。
新しいプログラムじゃと、カンタンな理由さえあればザクザク切っているのがワカるじゃろう。

やってるコトも、GOTOで飛ぶだけだしね。
それに、まだ敵のやられチェックをカンタンにする方法は残っているよ。

ツマリ今は「敵がゼッタイにやられてない」条件をサガせばいいってコトか？　……やられてないトキ……逆に言えば、やられるのはショットが当たったトキだけ……よし！　ゼンゼンわからねえ！

ものすごくあきらめちゃったけど、たぶん「ショットが画面に出ていない間は、敵はやられない」と言えるんじゃないかな。

ソコだね！　ショットは変数MSTで表示フラグをカンリしていたから……あとはカンタンだよね。

```
0080  IF MST==FALSE THEN @PASS
```

ナルホドな。言われてみりゃ、そのとおりだぜ。……てことは、何も言われずにコレいきなり見たら、どういうイミなのかワカりづらくねェか？
スナオにIF MX＝＝EX（I） AND MY＝＝EY（I）って書いてあった方がカンタンじゃねえの？

ぜんぶヒテイされてもうたが……そのキモチも、ワカらんでもないわい。スピードを気にせんでいい初心者のコロなら、ムリしてまでやるコトではないのはタシカじゃ。

でも、人のプログラムリストを見るときに「どうしてこんな書き方を？」ってギモンはなくなるんじゃないかな。こういうプログラムの書き方があると知っておくのはムダじゃないよ。

ナイスフォローじゃ、インテリ君！
プログラムの書き方にはヒトそれぞれイロイロなポリシーがあるものじゃが、そういう考え方を知っておくのもひとつのベンキョウじゃぞ。

ベンキョウってヒビキは気にくわねェが……ツギからこういうスキップのしかたを見てもナットクできるのはタシカか。

当たり処理の仕上げをしよう

あれ？　このプログラムRUNしてみると、ときどきショット１発で何匹も倒せることがあるよ。

ホントだな。こりゃアレだぜ、パワーアップとかによくある「カンツウダン」の動きになってるな！

「貫通弾」だね。シューティングゲームだと、敵に当たったらそこでショットは消える方がオーソドックスかな？　このゲームならそうしないとカンタンすぎるしね。

ええと、敵に当たったらすぐにショットを消すわけだ。
さっきもやったけどショットの表示フラグは変数MSTだったね。

```
00E3  '--- ｼﾎﾟｳ
00E4  LOCATE EX(I),EY(I):PRINT" "
00E5  ED(I)=-1
00E6  MST=FALSE
```

１行ふやしただけか。イガイにアッサリいくんだな。

フラグはうまく使うと、いろいろラクになるものさ。

あとは、敵を倒したらスコアも上げないとね。

コレはそうムズカしくなさそうな気がするぜ。
とりあえず、サイショに画面の上の方に出しておくんだろ？

```
00E2  LOCATE 0,0
00E7  PRINT"SCORE: ";SC
```

変数ＳＣをスコアってことにしておいたぜ。

そして、敵がやられた時に10点プラスすれば完成かな。

```
00E5  '--- ｼﾎﾟｳ
00E6  SC=SC+10
00E7  LOCATE 7,0:PRINT SC
```

うむ、それで合っておる。合っておるのじゃが……。

ナンだ？　ＲＵＮしても、ナニもモンダイはなさそうだぜ？

しかし、ワシとしてはこういうプログラムをテイアンしたい！

```
00E5  '--- ｼﾎﾟｳ
00E6  SV=SV+10
      
0110  @SCORE
0111  IF SV==0 THEN RETURN
0112  SC=SC+10
0113  SV=SV-10
0114  LOCATE 7,0:PRINT SC
0115  RETURN
```

メインループから＠ＳＣＯＲＥにＧＯＳＵＢすると考えてくれい。

どういうこった？　敵がやられたトキは、ただ変数ＳＶに10点プラスして終わりかよ？

そのあとで、変数ＳＣを変えてＰＲＩＮＴするのはサブルーチン＠ＳＣＯＲＥでやるんだね。そこで変数ＳＶも－１０して、またゼロに戻る……

イミがワカらねェな！　プログラムが長くなっただけじゃねえか！

先に言っちゃうと、これはＬＯＡＤしたサンプルプログラムだとこうなってるんですよね。

さよう。なぜワシがこんな一見ムダに見えるプログラムにしたのか、ソコがワカるかの？

ムググ……サイゴになってイガイなナゾトキだぜ……。どう見てもムダにしか見えねェが……。

いやジッサイ、このプログラムのままじゃと、ホントにムダなのじゃ。

バカにしてんのか！

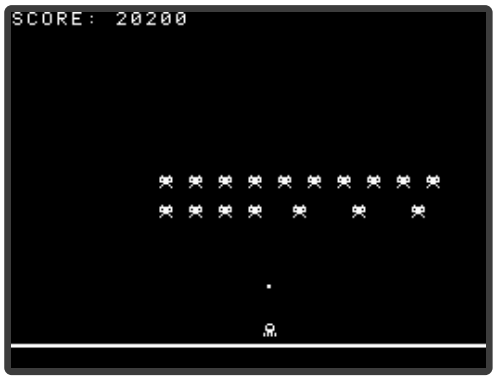
このサンプルプログラムはユーザーのショクンの手でカンセイさせるためのモノじゃからのう。そこでイミが出てくるように作ってあるのじゃよ。

うーん。そうは言っても、どういう時にヤクに立つのか……

こういう何のためにあるのか分からないプログラムは、関係のありそうな数をデタラメでも変えてみると、あんがい目に見えて分かるコトもあるよ。

というと、たとえば敵がやられたトキの`SV=SV+10`を`SV=SV+1`に変えてみる、とかいうコトか？
ヤミクモに`RUN`してやろうじゃねェか！

画面の前のみんなもジッサイに変えてためしてみよう！



ウ、ウォオ!?　敵を１回撃ったら、スコアが上がりツヅケて止まらねー！

そうか……考えてみればこうなるのはトウゼンではあるよね。

アァ？　ブツブツ……`SV`が１になってそのアト１０引いてまたモドッて０じゃねェからまた１０……フムフム。そりゃスコアがいつまでも上がりっぱなしにもなるな！

みんなもプログラムの流れを読むと、何が起きたのかだいたいわかるよね！

わざとデタラメにやっておるのじゃから、おかしな動きになるのはトウゼンじゃな。
しかし、このスコアを見てナニかに気付いたのではないかの？

いや、ベツに……

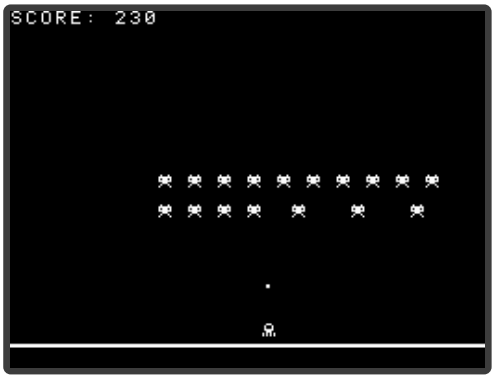
ココまでのナガレがダイナシになるのう……

あっ！　スコアが10点ずつアニメーションで増えるように見えるよ！

そりゃタシカにそう見えるけどよォ……。メインループを１回通るたびに10点増やしてんだからトウゼンじゃ……ハッ!?

いいトコロに気付いたね。ギャクに言えば、メインループを１回通るたびに点を増やすようにすると、アニメーションで点数が増えて見えるワケだ。

ナゾがとけてきたぜ……こうすりゃハッキリするな。さっき`SV=SV+1`にしたトコロを`SV=SV+100`にして`RUN`だ！



うむ！　止まった画面ではよくワカらんが、ミゴトに１匹倒すごとに100点が入り、それが10点ずつアニメーションしておる。

サブルーチン@SCOREを通るたびに、変数SVがカラになるまで10点ずつ増えてるからな！　コレはメインループの中でナンドも通ることにイミがあったってワケだ！

モトモトは、たとえば１匹だけ高得点の敵がいるとして、そいつを倒したときにボーナスっぽく、と作ったブブンだったんじゃが、コレももちろんセイカイじゃぞい。

まさかサイゴにこんなクイズがノコッてたとはな……。

プログラムをヒトに渡す時は分かりにくい処理にはコメントを付けておくのがキホンだけど、初心者向けのカダイとしては、まあ良かったんじゃないかな。

ケッキョクそういうアツカイなのかのう……？

今回のまとめ

当たり判定

いちばんオーソドックスな当たり判定は、ぶつけるモノの位置とぶつかられるモノの位置を比べることです。ヨコ座標とタテ座標が一致すれば、それは当たっているということになります。

CHKCHR()命令

（変数）＝CHKCHR(横座標,縦座標)

画面の指定した座標に、何のキャラクターが表示されているかが、文字コードで変数に入ります。

ビット演算の実習

プログラマーからひとこと

ふたたびビット演算の話です。

「もういいよ！　あんまり使う場面ないし！」と思うかもしれませんが、うん、まあ、いままであんまり使う場面ありませんでしたね。
そこで今回は、もっと役立つビット演算の使い方をいくつかおぼえてみよう、という回です。

これで意外と役に立つものなんですよ。
例によって、これさえおぼえておけば、人のプログラムを読むときにグンと読みやすさがアップしますし。

難しいようで、わかってみるとカンタンなことでもあるので、ANDやORの計算に慣れながら読んでいってみてください。

ループする数


前回までのサンプルプログラムで、ビット演算が出てきたのはおぼえてるかな？


たしか、`A=(A+1) AND 3`ってカタチだったよね。


そうそう。コレでどういう効果が出るかは、おぼえてる？


この行を通るたびに変数`A`の数が`0`から１ずつ増えるんだけど、`4`まで増えたときには`0`に戻されるんだった、かな？


そういうコトだね。では、このビット演算を……


テメエら、リズムだけでサクサクおさらいするのもイイカゲンにしやがれ！　このオレもついてこれてねェぞ！


ヒドいなあ。じゃあ、いまの`AND 3`がうまく動くしくみを見直してみることから始めるかい。


復習ってのもおベンキョウくさくってあんまり好きじゃねェが……
まず、サイショに変数`A`が`0`だったトキはどうなるんだ？


`A=(A+1) AND 3`ってことは、言いかえると`1 AND 3`の計算になるよね。


それを２進数の式にすると、こうなるね。

```
  01
AND 11
  = 01
```

`AND`を使った計算はまだ忘れてないかな？


上からメセンがニクニクしいぜ！　同じ位をクラベて、両方`1`なら答も`1`、`0`が片方でもあったら`0`になるってヤツだろ。


そこまでワカッてるなら一気にいこうか。変数`A`に入る数字が`0`から`3`までのパターンを並べてみるよ。

それぞれ+1されるから、1 AND 3 ～ 4 AND 3の式になるね。

```
   01      10      11      100
AND 11  AND 11  AND 11  AND 011
  = 01    = 10    = 11    = 000
```

4 AND 3のときだけ、結果がゼロになるのがわかるだろう。

ホントだな。
ギャクに1～3までは、AND 3されても、そのままナニゴトもなく変化しねェのがフシギだぜ。

そうか、11みたいに全部1で並んだ数でANDすると、数はなにも変わらないでそのまま出てくるんだね。

おっ、いいトコロに目をつけたね。そのとおりで、たとえばAND 2だと、そうウマくはいかないんだ。

```
   01      10      11      100
AND 10  AND 10  AND 10  AND 010
  = 00    = 10    = 10    = 000
```

タシカになんだか、スッキリしねえケッカになるな。

これがAND 7だとどうかな？ 7を2進数になおすと111で、1が並ぶ数字だね。

```
   001     010   …    111     1000
AND 111 AND 111   … AND 111 AND 0111
  = 001   = 010   …   = 111     0000
```

やっぱり3ケタまではゼッタイ変化しないで、上の数が4ケタになったとたんにゼロになるよ！

……ANDと1が並ぶ数を使うと、ある数まではフツウに増えて、ある数まで増えたらゼロにリセットされるってコトなのか？

悪くないリカイだよ。
A=(A+1) AND 7が「8回ループするたびに1回だけゼロになる」ものと考えると、IF A==FALSE THENとすれば8回に1度だけトクベツなコトができるね。

かならず8回目にゼロに戻ってくるのもベンリかもね。ON A GOTOを使ったプログラムでも、自動的にループになるよ。

タシカにサンプルプログラムでやったコトも同じか……。
で、111みたく1が並ぶ数ってのはナニとナニがあるんだ？ 3と7はワカッたが、いちいち探すのはメンドくせェぜ！

だいじょうぶ、それにはパターンがあるんだ。先にそういう数を見せた方が早いかな？

2進数にすると
1の連続になる数字

10進数	2進数
1	1
3	11
7	111
15	1111
31	11111
63	111111
127	1111111
255	11111111

:	:


1、3、7、15、31……うーん、パターンがあるような、ないような……


ナンだか数字パズルみたいになってきたな。ニガテなテンカイだぜ！


ゼロから数え始めるコンピューターのクセに合わせて、数を１ずつ増やすとどうかな？


ん？　2、4、8、16、32……あっ！　ぜんぶ倍になってるよ！


２の倍は４、４の倍は８、８の倍は16……いわゆる「２のべき乗」だね。２のべき乗マイナス１と考えれば、アンガイおぼえやすいだろう。「64」や「256」なんかはゲームでも見たことある数じゃないかな？


64分の１の確率でアイテムをドロップ、とかアリガチじゃな。
コンピューターとアイショウのいい数じゃからのう。ムカシはバキ●ラを256回撃つと倒せるというウワサが……


ケッ、ムカシ話にはキョウミがねえぜ！


ま、まあそれもこれもビットの考えカタにそっとるのじゃよ。


この数だと「63」にWAIT命令を組みあわせると、こんな風にもできるね。

```
@LOOP
WAIT(1)
A=(A+1) AND 63
PRINT A
GOTO @LOOP
```


ＷＡＩＴ（１）は60分の１秒だけ待つってコトだろ。それとコレとどうカンケイが……


ビット演算では64回に１回リセットだから……１秒と4/60秒だけループした時に、変数Ａがゼロにリセットされることになるね。


つまり「だいたい１秒」でリセットされるカウンターだね。ここではただのＰＲＩＮＴだけど、ゲームのタイマーや１秒ごとにたまるダメージとか、使いどころはあるんじゃないかな。


オイオイ、「だいたい１秒」って、そんなテキトウでいいのか!?


目に見えるワケではないからのう、体感時間としてはアリじゃ。ぶっちゃけプログラムにはよくある手じゃよ。
その後、プログラムを解析されて「なんちゃって１秒」だとバレるところまでふくめてワンセットじゃ。


は、はかりしれねえセカイだぜェ……。

BUTTON()命令の種明かし


ン？　さっきの「2、4、8、16、32……」って数、マエにも見た気がするな……


そういえば、ＢＵＴＴＯＮ（）命令に使われてる数字がそうだったよ！

十字ボタン↑	1	Aボタン	16	Lボタン	256
十字ボタン↓	2	Bボタン	32	Rボタン	512
十字ボタン←	4	Xボタン	64	START	1024

十字ボタン→	8		Yボタン	128		SELECT	2048

ソレもビット演算の考え方を使ってるんだけど、しくみは飛ばしておぼえていたね。ためしに２進数に直してみるよ。

10進数	2進数
1	1
2	10
4	100
8	1000
16	10000
32	100000
:	:

そうか、２進数にするとキリがいい数だったんだな。ソコまではワカったが……コレとビット演算がカンケイあんのか？

これだけだと、まだわかりづらいね。２進数のケタをそろえてみよう。

10進数	2進数
1	000001
2	000010
4	000100
8	001000
16	010000
32	100000
:	:

先頭にゼロをつけただけで、数が変わったわけじゃないよ。

ははあ、右からジュンバンに１ケタずつ、ボタンと1のペアができてるんだね。こうやって見ると数字っていうよりは、記号みたいだよ。

タシカに。それぞれのケタがスイッチで、0が「オフ」で1が「オン」だって考え方があるんだ。
だから、こういう表でも1になっているケタを「ビットが立っている」と言ったりするね。

0から1になるのが、「ビットが立つ」ってことか。

000010なら第１ビットが立っている、なんて言うね。１ケタ目が第０ビット、２ケタ目なら第１ビットさ。

またゼロから始める数えカタかよ……ギョウカイ用語はいいから、サッサとハナシをススメやがれ！

じゃあ、たとえば「十字ボタン↑」と「Ａボタン」が同時押しされた時で考えてみよう。
10進数なら１＋１６で１７が変数に入るけど、この数を２進数にするとどう見えるかな？

えーと、こうなるかな。

10進数	2進数
1	000001
16	010000
1+16=17	010001

お、オォ？　そのまま組み合わさったみてェな数になったぜ！

こういう「1、2、4、8、……」と倍々になっている数のトクチョウだね。数をたすとビットが立ったままキレイに合わさるんだ。

ますます記号みたいだね。
つまりBUTTON()命令だと、第0ビットが「十字ボタン↑」のスイッチになっていて、第4ビットは「Aボタン」のスイッチってことだね。

十字ボタン↑	1	000000000001
十字ボタン↓	2	000000000010
十字ボタン←	4	000000000100
十字ボタン→	8	000000001000
Aボタン	16	000000010000
Bボタン	32	000000100000
Xボタン	64	000001000000
Yボタン	128	000010000000
Lボタン	256	000100000000
Rボタン	512	001000000000
START	1024	010000000000
SELECT	2048	100000000000

ナカナカおもしれえハナシだが……だからナンのイミがあるんだって気もするぜ！

ここからがベンリなところだよ。
BUTTON()命令で言えば、Aボタンが押されているかをチェックしたい時はどうするんだったかな？

何もカンガエずに、IF文でAND　16と書けばイイって、教わったな。

```
@LOOP
B=BUTTON()
IF B AND 16 THEN PRINT"ｶﾞｵｻﾚﾃﾏｽ"
GOTO @LOOP
```

じゃあ、今度はちゃんと見てみよう。
いま出てきた「十字ボタン↑」と「Aボタン」が同時押しされた時の数「010001」に、AND　16のビット演算をしてみるね。

```
  010001
AND 010000
  = 010000
```

あっ！　Aボタンのスイッチになってる第4ビットだけが立ってるよ！

わかってきたかな。チェックしたいケタに1を、それ以外のケタに0を入れてANDで演算すれば、そのビットが立っているかどうかスグにわかるんだ。
このトクチョウをうまく使ったのが、BUTTON()命令とANDを使ったIF文だね。

ゴクリ……てことは、第5ビットのBボタンが押されてるかどうかで考えると……

```
  010001
AND 100000
  = 000000
```

ヤッパリ答はゼロ、第５ビットは立ってねえってコトがワカるぜ！

「1、2、4、8、……」という数で組み立てておくと、こういうチェックにベンリなのがわかるね。
頭で考えるよりも、こういう数のパターンをひとつおぼえておくといいよ！

ビット演算を使ったフラグ

フーム……このパターンをBUTTON()命令イガイに使うとすると、トウゼン変数だよな。フラグなんかにベンリそうな気はするぜ！
どうするのがイイのかはサッパリ思いつかねェけど。

スッキリあきらめたね。
まあ、さっきのBUTTON()命令だって、「上ボタンを押すと第０ビットのフラグが立つ」と考えればリッパにフラグさ。その応用で作ればいいよ。

ナルホド、そういうミカタもあるか……。

さすがにBUTTON()なみに11ケタいるとは思えないけど、４ケタあれば、４つのオン・オフをいちどに見られるフラグになるワケだね。

イヤちょっと待てよ！　BUTTON()ならボタンを押せばカッテにフラグが立ってくれたがよォ、自分のプログラムの中でフラグを立てたり消したりするのは、どうやりゃいいんだ？

たしかに、10進数みたくA=A+1みたいに書くわけにはいかないよね。第２ビットを1にするには、ええと……

フラグを立てるならビット演算のORを使うのがキホンかな。
たとえばまっさらな変数Aがあったとしよう。A=0だから、２進数でも0000なのはわかるよね。
この３ケタ目（第２ビット）を立てるには、こうするよ。

```
  0000
OR 0100
 = 0100
```

ここでORに使った0100は、10進数で言いかえれば４。
つまり、A=A　OR　4と書けば、第２ビットが立つことになるね。

ORを使うのはワカッたが、イチイチ10進数に直すのがカッタルイぜ！

なれないウチは並んだ２進数の右ハジから、「1、2、4、8、……」と倍にしながら数えていくのがカンタンかもしれんな。10進数で右ハジから「一、十、百、千、万、……」とケタを数えるノリでいけるぞい。

第２ビットが立っているときに、もういちどOR　4すると……

```
  0100
OR 0100
 = 0100
```

やっぱりそのままだね。ORはビットを立てるために使うってコトか。

オン・オフを逆にしたい時は、XORがオススメだね。
ためしに1100と1000、どっちにも0100でXORしてみるよ。

```
   1100        1000
XOR 0100    XOR 0100
 = 1000      = 1100
```

1でXORした第２ビットだけがひっくり返って、ホカのケタはそのまま残るのがわかるだろう。

XORって使い方がムズカシいと思ってたけど、こういう時にベンリなんだね。

じゃあ、とにかく０にしたいってトキは……そうか、ＡＮＤだな？

```
  0100      1111
AND 1011  AND 1011
  = 0000    = 1011
```

1011でＡＮＤ、10進数で言えばＡＮＤ　１１だな。
これでホカのケタはそのままで、第２ビットだけが０になったぜ！

なかなかワカッてきたではないか。
さっきの復習じゃが、ＡＮＤを使ったフラグチェックには、こういうテクニックもあるのう。

```
  1110
AND 0100
  = 0100
```

ＡＮＤ　４すると、第２ビット以外はゼロになったじゃろう。

ＢＵＴＴＯＮ（）で使ったテクだな！　第２ビット以外をゼロにすれば、第２ビットのオン・オフだけがハッキリする。ツマリ……エー……

そうか！　そうなると、ＩＦ　Ａ　ＴＨＥＮの形が使えるんだね。

そのトオリ。１行にまとめると、ＩＦ　Ａ　ＡＮＤ　４　ＴＨＥＮ～だね。
もちろんホカにもＡ＝Ａ　ＡＮＤ　４：ＩＦ　Ａ＝＝ＦＡＬＳＥ　ＴＨＥＮ～ なんかでもいいけど、とにかく決まったケタのナカミだけがスグにわかるよね。

ホカのケタがまるまるゼロになっちまうが……、ああ、そこは変数のナカミをあらかじめコピーしとけばいいのか？

そうじゃな、Ａ＝Ｂとか逃がしてから、Ｂ＝Ｂ　ＡＮＤ　４などとするのがアンゼンじゃろうの。

まだまだイロイロなテクニックはあるよ。なかなかオクの深い世界だろう。

とっつきはワルいが、使いミチはワカッてきた気がするぜ。てコトは、このアトすぐに使う場面があるワケだな？

いや、そうともかぎらんが。

えー……？

今回のまとめ

ループする数

```
A=(A+1) AND 3
```

こう書くと、Ａ＝０から始めて、この式を３＋１回くり返したとき、変数Ａは０に戻ります。
３以外に、７、15、31、63……と、「２のべき乗」マイナス１の数で同じパターンが使えます。

ビット演算を使ったフラグ

たとえば２進数の0000（10進数の０）を「４つ並んだフラグ」と見なせば、４種類のオン・オフを管理できます。
0001（10進数の１）なら「第０ビットが立った」状態、0100（10進数の４）なら「第２ビットが立った」状態と言えます。
他のケタを変化させずに第２ビットだけを1にしたければＯＲ　４（２進数の０１００）、第２ビットだけを0にしたければＡＮＤ　１１（２進数の１０１１）することになるでしょう。
第２ビットの状態だけを反転させるには、ＸＯＲ　４（２進数の０１００）がいいでしょう。
わざとＡＮＤ　４（２進数の０１００）することで、第２ビット以外を0にして、第２ビットの状態

をハッキリさせる方法もあります。
こういったあるひとケタだけが**1**になっている数は、10進数に直せば「1、2、4、8……」と倍々に増えていく数になります。

サンプルプログラム７の改造

プログラマーからひとこと

いよいよサンプルプログラム７ともこれでおわかれです。

時間をかけてサンプルがどう動いているのか見てきただけあって、今回はけっこうカンタンに話が終わるような気がします。
それだけみなさんもスキルアップしたということです。おめでとう！

前回いきなりサンプル７と関係ないビット演算の話が出てきておどろいたかもしれませんが、ちゃんと今回の話に使いますよ。
それもこれもふくめて、これまでの講座の総まとめです。
泣けるＢＧＭなどをかけながら読んでみてください。

ゲームオーバーを作ってみよう

さて、サンプルプログラム７のつくり方はだいたい分かったけれど……

イヨイヨ、このロクでもねえプログラムをマシにカイゾウするってワケだな！

ソコまで言わんでもよかろうに……

いちどロードしなおしておこうか。

```
LOAD"SAMPLE7"
```

どこから手をつけたらいいのかな？

マッサキにおかしいトコロと言ったら、ゲームオーバーにならねェとこだろうな。

セイカクには、ゲームオーバーになるのは「敵が自機のいる２１行目まで下りてきた時」ということになるかな。

チッ、コマカいやつだぜ！

どういうプログラムにするか、前もってハッキリとイメージしておくのはダイジじゃぞい。
まあ、思いつきでドンドン打ち込むのもひとつのダイゴミではあるが。

こまかく考えていくと、先頭の１匹が２１行目についたタイミングがいいのかな。２０行目のはじっこだとまだ早いって気がするし、何匹も２１行目に乗っていると遅いかんじだもんね。

うむうむ、そういうコトじゃ。こういうのを「仕様（しよう）を決める」と言うのじゃが、できるだけミノガシの少ない仕様を決められるのが良いプログラマーのジョウケンじゃぞい。

ムムム……アトは……そうだな、画面に「GAME OVER（ゲームオーバー）」って書かねェとな！　そしてＥＮＤだ！

そうか、それはたしかにダイジだね。ちゃんとゲームオーバーだってわからないと。

まずは、そういう仕様で作ってみようか。まずサイショにすべきコトは……

もともとムダだった、自機のノコり表示をヤメようぜ！　コメントアウトしとくか。

```
00026 'LOCATE 0,23
00027 'PRINT"ﾊｲｽｺｱ:";LEFT;
```

そのトオリじゃが、あいかわらずコトバがキビしいのう……。

プログラムの中で敵が２１行目におりてくるとすると……@DWルーチンになるのかな。

```
01200 @DW
01210 EY(I)=EY(I)+1
01220 IF EY(I)>20 THEN EY(I)=3
01230 ED(I)=(ED(I)+1) AND 3
01240 GOTO @EPUT
```

122行目では敵が画面の下から上にワープするように作ってあるけど、今回の改造ではココはいらないね。

```
01220 'IF EY(I)>20 THEN EY(I)=3
```

とりあえずコメントアウトしよう。

せっかくIF EY(I)>20 THENなら、ソコでゲームオーバーにすりゃイイんじゃねェの？

オシイけど、ココではまだ敵が21行目に表示されていないんだ。さっき決めた仕様どおりに作るならハズレだね。

チッ、コマカいハナシはニガテだぜ！　ウゴいた敵を書いてるのは、そのアトの@EPUTルーチンだったか。

```
01370 @EPUT
01380 LOCATE EX(I),EY(I)
01390 PRINT "ﾊｲ"
01400 RETURN
```

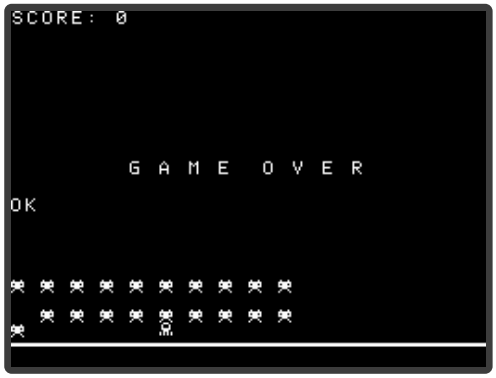
どう見ても、139行目のアトにIF文を１行書けばイイはずだぜ！

```
01400 IF EY(I)>20 THEN @GOVER
```

たいしてやるコトもなさそうだが、イチオウ新しくゲーム(Game)オーバー(OVER)サブルーチンに飛ばしといたぜ。

あとはムズカシイことは何もなさそうだね。

```
01420 @GOVER
01430 LOCATE 8,11
01440 PRINT "G A M E  O V E R"
01450 END
```



ウム。キチンとできておる。

しかし、やってみるとジミな気もするな。ドウセならこのぐらいは……

```
@GOVER
LOCATE PX,PY
COLOR 7
PRINT "■"
LOCATE 8,11
COLOR 13
PRINT "G A M E  O V E R"
END
```

うむうむ。いくら仕様を前もって決めていても、作ってみるともっとやりたいコトは出てくるモノじゃ。ためらわず新しく作りはじめる心もダイジじゃぞ。

言ってるコトがバラバラな気もするけど……

あと、自分でワケがワカらなくなる前に、ホドホドで止めておく心もやっぱりダイジじゃな。

どうしろってんだ！

どの考え方も同じくらい大切なものさ。うまくバランスをとってプログラムしていきたいね。

画面リセット

このコラムもこれがサイゴかもしれんと思うと……ワシじゃ、コラムのヌシことハカセじゃ。
さっきのプログラムでは`COLOR`を`15`にモドさないで`END`しておるから、その先にナニを打っても赤い字になるのう。`COLOR`命令を何度も使うプログラムじゃと、いつも`COLOR 15`で終われるとはかぎらん。そこで、ベンリなワザを教えておくぞい。

“Rボタン ＋ 左ボタン ＋ STARTボタン”

こうDSiのボタンを押せば、画面がリセットされるのじゃ。
`ACLS`と命令を打っても同じことじゃが、キーボードを使うよりラクじゃろ？

スピードを変えてみよう

次はどこを改造するのがいいかな。ゲームとしてちゃんとしてないのは……

そりゃ、敵のスピードが早すぎるコトだな！　ゲームオーバーを作ってマスマスよくワカッたが、敵をオソくしねえとハナシにならねェぜ！

そう言えば、スピードをコントロールしてる部分がもうあったような……

！　タシカに、オレとしたコトがワスレてたぜ！

```
@LOOP
```

```
0036 GOSUB @MYSHIP
0037 GOSUB @MYSHOT
0038 GOSUB @ALIEN
0039 GOSUB @SCORE
0040 WAIT(1)
0041 GOTO @LOOP
```

メインループの中の**WAIT(1)**で1/60秒オソくしてたんだったな。じゃあ、コレをもっとオソくすりゃイイんじゃねェか？

```
0040 WAIT(20)
```

あれ？　なんだか……

```
RUN
```

……あ、そうか。

し、シマッタ！　自機がオソいのは100歩ゆずっても、ショットがオソいのだけはガマンならねー！
というかコレ、ただゲームのテンポがユルくなったダケだな！

そういうコトだね。**WAIT**はその場でポーズするようなモノだから、ゲーム全体のスピードが変わってしまうんだ。

WAITじゃなくて、敵だけが遅くなるような改造にしないとダメなんだね。

ムムム……自分やショットはウゴくが、敵はウゴかない……ん？　そう考えるとアンガイよくあるパターンじゃねえか？

そう、敵を動かす部分をスキップすればいいってコトだね。

```
0117 @EMOVE
0118 IF ??? THEN @EPUT
0119 ON ED(I) GOTO @RI,@DW,@LF,@DW
```

ある条件のトキ以外は敵の場所を変えないことにしたよ。

ずっとスキップしっぱなしじゃ動かなくなっちゃうから、このIF文で「何回かに１回だけ動く」ようにすればいいってことか……。

だいたいハナシが見えてきたぜ。こうすりゃイイんだな！

```
0085 '--- シンリャクシャ
0086 @ALIEN
0087 ESTOP=ESTOP+1
```

まず敵をウゴかすオオモトのところに、変数**ESTOP**をヨウイするぜ！　コイツは敵を止(STOP)めるためのフラグだな。

```
0088 IF ESTOP==9 THEN ESTOP=0
```

こうやって、10回に１回は**0**にモドるようにしておくぜ！　アトは、スキップするジョウケンを……

```
0119 @EMOVE
0120 IF ESTOP THEN @EPUT
```

IF ESTOP THEN、ツマリ**ESTOP**がゼロじゃなければスキップする、ってコトにすればデキアガリじゃねえか!?

おっ、やったじゃないか！　これで敵のスピードだけが10分の１になったね。

ワレながら、ジョウタツぶりがミゴトだぜ……。

あれ？　そういえばこの10回に１回は0にもどるってパターンは……

どうやらオボエていたようじゃのう。ワンパク君のプログラムの、ドコを変えればいいかわかったかの？

オレのプログラムのシュウセイからのスタートかよ！

この88行目はやっぱりけずって……

```
88ET ESTOP=ESTOP+1
88EE IF ESTOP==9 THEN ESTOP=0
```

そしたら、数がフエるイッポウじゃねェか？

うーん、タシカにそうなんだけど。

```
88IE IF ESTOP%9 THEN @EPUT
```

スキップの条件をこう書きかえれば、10回に１回だけFLOOR(ESTOP%9)の答が０になる……つまりスキップしないってコトになるよね。

「I=I+1でループするとき、I%Nと書くとIが『Nの倍数』になるたびゼロまでおり返しながら１増えていく」。前におぼえたホウソクだね。

やってるコトは同じじゃが、IF文がひとつ消えてコンピューターにやさしく、短いプログラムになったのう。
ESTOPの数はサイゴまで増えっぱなしじゃが、このゲームではメモリーのゲンカイまで増える前にゲームオーバーになるから、これでもよかろう。

そこまでコウリツをモトめるその考え方、オレは気にくわねェな！

タシカに、ワンパク君のプログラムの方が初心者にはわかりやすいとも言えるわい。
どこまでプログラムにコウリツを求めるべきか、それはプログラマーにとってエイエンのテーマじゃ。

どれがセイカイとも言えない、ってコトかな。ちなみにスキップする回数はちょっと変わるけど、こういう方法もあるよ。

```
88ET ESTOP=(ESTOP+1) AND 7
     :
88IE IF ESTOP THEN @EPUT
```

ビット演算のくわしいセツメイはもうはぶくけど、９回に１回だけESTOPがゼロになってスキップしなくなるプログラムだね。
コンピューター的にはわり算よりビット演算の方が早いから、ボクはI%Nを使うよりもスキかなあ。

ケッキョク、プログラムするヤツのスキズキってコトか……。

セイカイがひとつとかぎらんのも、プログラムのおもしろさじゃな。プログラマー２人に合作させるとおたがいのビガクがぶつかって、見ていてオモシロいもんじゃよ。

……やじうま？

クリア画面をつくってみよう

スピードが変わって、だいぶゲームらしくなってきたな！

敵が少なくなると、デタラメに撃っても当たらなくなるのがよくわかるね。だけどあんまりじっくり当てようとしていると、敵にせめこまれてゲームオーバーになっちゃう。ゲームにはこういうカケヒキがないとね。

なんだかハジメてホメられた気がするわい……。

こうなってくると、敵を全滅させた時に何もないのがさびしいね。ゲームクリア画面を作っておこうよ。

そうムズカシくないんじゃねェか？　とりあえず敵の数をチェックするための変数がいるな。

```
0012 ELEFT=EMAX
```

敵の最大数を決める変数**EMAX**がもともとあったから、新しい敵(Enemy)の残り(LEFT)変数**ELEFT**に使うことにするぜ！

敵が１匹やられるたびに、変数**ELEFT**を１へらせばいいね。

```
0110 ELEFT=ELEFT-1
```

あとはこの数がゼロになったら、ゲームエンド(Game END)でいいかな。

```
0111 IF ELEFT==FALSE THEN @GEND
```

そこからは新しいサブルーチンになるね。

```
0157 @GEND
0158 LOCATE 11,11
0159 PRINT "ｼﾝﾘｬｸ ﾐｽｲ"
0160 END
```

こんなところかな？

```
SCORE: 190


            ｼﾝﾘｬｸ ﾐｽｲ
OK
```

さすがにサクサクいくようになったのう。
クリアしたその時、はじめてゲームタイトルが出てそのイミが明らかになる。ワシのノゾミどおりの、ハリウッド的クールなエンシュツじゃ。

そこまで考えてなかったけど……。

あっと、まだ終わりじゃないよ。ひとつわすれてることがあったね。

そうか？　トクにおかしいトコロはなさそうだが……

```
SCORE: 190
```

あっ！　最後の敵を撃った点が入ってないよ！

最後の敵が撃たれたらスグに**@GEND**に飛んでいるから、こうなっちゃうね。

じゃあ、@GENDに飛んだサキですぐに点を入れればイイんじゃねェか？

```
0157 @GEND
0158 GOSUB @SCORE
```

ず……ずいぶんストレートな方法でカイケツしてきたね。

@SCOREに行って10点プラスして帰ってくるワケじゃな。10点以上の敵を倒してクリアしたトキはどうする、というモンダイはあるが……そもそも10点以上の敵がいないので、マチガイなくこれでカイケツはしておるな。

いいの？　ハカセにしてはアッサリしてるけど。

何点の敵を打ってもＯＫなプログラムを組むのがリソウではあるんじゃが……、ショクンならフラグをうまく使えばスグにできるじゃろうしな。10点以上の敵を出すヨテイもない今は、これでよかろう。

（さっき自分も気がつかなかったコトを、ゴマカそうとしてるんじゃねェか……？）

ショットをボタン操作に変えてみよう

あとは、ショットを画面タッチからＡボタンに切り替えれば、ボクらのキボウもだいたいかなったことになるかな。

ショットは70行目で入力チェックしていたんだっけ。

```
0070 IF TCHST==FALSE THEN @MSSKIP
```

システム変数TCHSTがFALSEだったら（タッチされてなかったら）ショットをスキップする、っていうやり方だね。

そのままBUTTON()に置きかえればすむコトじゃねェか！　……ン？
この前にB=BUTTON()してるから、IF B AND 16と書けばＡボタンを押してるのはワカるが……

そのままそう書くと、ＴＨＥＮの後のイミがちがっちゃうね。
スキップするのは変えたくないから、ここには「IF （Ａボタンが押されていない） THEN @MSSKIP」と書きたいけど……。

えーと、ビット演算のときにおぼえたけど、BUTTON()で変数に入る数は、２進数に直すと第０ビットから１ケタずつ、スイッチになってるんだったよね。

十字ボタン↑	1	000001
十字ボタン↓	2	000010
十字ボタン←	4	000100
十字ボタン→	8	001000
Ａボタン	16	010000
Ｂボタン	32	100000
：	：	：

たとえば「十字ボタン→」と「Ａボタン」が同時押しされていれば、変数に入る数は２進数では011000になるね。
これにAND 16すると……

```
  011000
AND 010000
  = 010000
```

Aボタンの第４ビットだけが立っているかどうかわかる、というのがおさらいになるね。

オサライかよ、オレのキライなナガレだぜ……。

うーん……そして、Aボタンが押されていない時にAND 16すると……

```
   001000
AND 010000
  = 000000
```

ほかのボタンを押していても、答はゼロになるわけだね。あたりまえだけど……

オ、オイ！　ソレじゃねえか!?

え？

！　（……ワンパク君！　気づいたようじゃな……！）

AND 16すりゃ、Aボタンを押してないトキにはゼッタイ0になるんだろ？
イマまで使ったことのねェやり方だけどよォ、それなら「IF　（Aボタンが押されていない）　THEN @MSSKIP」ってのも書けそうじゃねェか！

```
IF B AND 16==FALSE THEN @MSSKIP
```

……ム。ヘンな書き方になった気もするが……

```
RUN
```

チクショウ！　合ってると思ったんだが……ウマく動いてねえ！

いやいや、考え方はバッチリだよ。

ナニ？

その書き方だと、「B=TRUE」AND「16==FALSE」のように読めちゃうから、おかしくなるんだね。
カッコでかこんで、こうして書けばいいんだよ。

```
IF (B AND 16)==FALSE THEN @MSSKIP
```

ホントだ！　これならうまく動くよ！

ワンパク君……イージーミスでつまずいたものの、あれだけニガテだったビット演算の考え方をキチンとリカイしたようじゃのう。ワシはうれしいぞい。

ダテにあのメンドクセェ講座をくぐり抜けてきたワケじゃねえぜ！

まとめのようなもの

おぼえたコトは、なんでもムダにはならないってコトだね。
ビット演算にかぎらず、このBASIC講座で勉強したことは、イガイなところで役に立ったり、応用しだいでもっとイロイロな使いみちがあるハズさ。

そういうコトじゃな。読者のショクンも、このサンプルをさらに改造するコトから始めてみてはどうかな？
これまでの講座でおぼえた命令を組み合わせると、イガイにやりたいコトができたりするものじゃ。

たしかに、まだ改造したいポイントはいっぱいあるね。敵の種類をふやしてみたいし、プレイヤーが１機でおわりじゃ

なくてもいいよね。

もっとカラフルにするのもよさそうじゃな。敵がやられた時にバクハツパターンを出すなどもいいじゃろう。
こういう見た目のエンシュツはイガイに考えるところが多く、うまくいった時のジュウジツ感も大きいのでおすすめじゃぞ。

やりたいコトに手が届かない時は、説明書をながめてみるのもいいかもね。
ムズカシそうだけど、ここまで読んだ今なら、何が書いてあるのかカナリわかるようになってるハズだよ。

……テメーら！　最後だけマジメな話でまとめて、さもキョウイク的な講座だったってカオしてんじゃねえぞ！

今回のまとめ

画面リセット

“Rボタン ＋ 左ボタン ＋ STARTボタン”を押せば、画面がリセットされます。

まとめ方

最後のほうで、みんないい子になって教育的なことでまとめれば、まるでいい講座だったように思い込めるものです。