非公式プチコン講座

第 2 巻

この「非公式初心者講座」は、Miiverse などに絶えず訪れ続けている「プチコン 3 号や BIG を買ってみたけど、何をしていいかわからない、お金を無駄にしたかもしれない、タスケテ」という迷える初心者の方々の最初の一歩（とっかかり）となるかもしれない講座です。プチコンまとめ Wiki と内容はほぼ同じです。

Takumi

プチコン 3 号&BIG 非公式初心者講座

プチコン 3 号&BIG の世界へようこそ!

この「非公式初心者講座」は、Miiverse などに絶えず訪れ続けている**「プチコン 3 号や BIG を買ってみたけど、何をしていいかわからない、お金を無駄にしたかもしれない、タスケテ」**という迷える初心者の方々の最初の一歩（とっかかり）となるかもしれないページです。

自分でプログラムをつくる前に

かなりはじめのほうのはなし

プチコン 3 号(BIG)は自分で BASIC プログラムを作るためのソフトですが、内蔵のサンプルプログラムで遊んだり、他の人の作品をダウンロードしたりもできます。ひたすら遊ぶガワになるのも、ある意味、プチコンのひとつの楽しみかたです。
そのやりかたは、公式の初心者講座「かなりはじめのほうのはなし」で解説しています。作品のダウンロードに必要な公開キーは、この Wiki の投稿プログラムコーナーや Twitter、Petitverse などで探せますので、いろいろ楽しんでみてください。
また、トップメニューの「Web プチコン入門」ボタンから、公式の SMILE BASIC 講座を読むこともできます。まだ最初のほうの内容しかありませんが、わかりやすく書かれているので参考になるのではないかと思います。

説明書のこともお忘れなく

プチコン 3 号や BIG を起動している間、本体の HOME ボタンを押すと、プチコン 3 号(BIG)の電子説明書を読むことができます（ちなみに、こちらからも読めます）。プログラムを学ぶための説明はほとんどありません（初心者のかたの参考になりません）が、プチコンの操作のしかた、使いかたについては、説明書らしくひととおり書かれています。

非公式初心者講座

ここまではわかったけど、もっとプチコンを使ってみたいみなさまへ。ではいよいよ、BASIC の基礎の基礎を、ハカセとユカイなナカマたち（  ）と一緒に学んでいきましょう!

はじめに

ほっほっほ、ワシがプチコンでユウメイジンなハカセじゃ。ヨロシクじゃぞい。
この本は、印刷が可能じゃから、紙で見たい場合はそうしておくれ。

そして、そのトナリでプチコン 3 号や BIG を起動させて、ジッサイに操作しながら、ジックリこの講座を読んでいこう！　キモチをあせらずにね！

もくじ

※ この講座は、ニンテンドーDSi 向けの前作「プチコン mkII」の公式初心者講座の内容を、「プチコン 3 号」向けに再構成したものです。ハカセたちのキャラクター、および初心者講座の内容は、株式会社スマイルブームより使用許可をいただいたうえで使用しています。また、この PDF は Wiki 内の講座をほぼ引用したものです。
※本書では、以下の命令は色を変えています。

サンプルプログラム 2

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"SYS/EX2CALC"
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！

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。

＋－×÷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。

これならスマホ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？

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？

「／」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」という意味になるのじゃ。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し、もっとも、プチコン 3 号では「％」の記号は、あまりを出す計算には使えないから、いったん忘れよう。

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。

```
'┌──────────────────┐
'| EXAMPLE2         |
'| けいさん COMPUTER |
'└──────────────────┘
XSCREEN 0
DISPLAY 0
VISIBLE 1,1,1.1
CLS
GCLS
PRINT "┌──────────────────────────────┐"
PRINT "|けいさん COMPUTER(ニコチャン)|"
PRINT "└──────────────────────────────┘"
PRINT "※:こんにちは ニコチャン です"
PRINT "ぼくの しつもんに こたえてね"
```

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！

DISPLAYやGCLSって、はじめて見る命令だけど……？

これも画面を消したり、上画面を使うよって指示したりしてるのじゃが……ここではそんなにダイジなものではないから、今は気にしなくていいぞい。

```
'---
@LOOP
PRINT
COLOR 9
```

16～19 行も今まで出てきた命令だね。18 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……。

I NPUT 命令

20 行目ではじめて見る命令が出てくるね。

```
20 INPUT "※：1つめのすうじは" ;NO1
```

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 20 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、26～27 行目でその変数をPRINTしてるワケだな……。

```
26 PRINT "(1)";NO1;"  と ";
27 PRINT "(2)";NO2;"  ですね"
```

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ぜ！　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

36 行目も、同じようにINPUT命令だね。

```
36 INPUT"＊：きごうは（＋－／％＊）";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……39 行から先はどういうイミだ？

```
39 MARK=0
40 IF  K$=="+"  THEN  MARK=1
41 IF  K$=="-"  THEN  MARK=2
42 IF  K$=="/"  THEN  MARK=3
43 IF  K$=="%"  THEN  MARK=4
44 IF  K$=="*"  THEN  MARK=5
45 ON  MARK  GOTO  @SKIP,@PLUS,@MINUS,@DIV,@MOD
,@MUL
```

39 行目はいちおうフツウに変数かな。変数MARKに0を入れて……

40 行目からは新しい命令だね。こういうのは「IF（イフ）～THEN（ゼン）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 40 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

```
39 MARK=0
40 IF  K$=="+"  THEN  MARK=1
41 IF  K$=="-"  THEN  MARK=2
42 IF  K$=="/"  THEN  MARK=3
43 IF  K$=="%"  THEN  MARK=4
44 IF  K$=="*"  THEN  MARK=5
45 ON  MARK  GOTO  @SKIP,@PLUS,@MINUS,@DIV,@MOD
,@MUL
```

39～44 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 45 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」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。45 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、39～44 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。
45 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、40 行～44 行のどれにも当たらなかった時だけ、39 行で決めたゼロのままで 45 行の ON～GOTO 文にたどりつくんだね。

```
39 MARK=0
40 IF  K$=="+"  THEN  MARK=1
41 IF  K$=="-"  THEN  MARK=2
42 IF  K$=="/"  THEN  MARK=3
43 IF  K$=="%"  THEN  MARK=4
```

```
IF K$=="*" THEN MARK=5
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「米：きごうは（＋－／％＊）？」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 39 行で最初にMARK＝0と決めてしまって、40～44 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」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 20 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

```
48 @SKIP
49 BEEP  4
50 PRINT"しらないきごうです...ゴメン"
51 PRINT
52 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。49 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 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133 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、画面の下にある 

SMILE(スマイル)ボタンを押すと、スマイルツールという便利なものが起動して、サウンドの一覧が表示されるのじゃ。タッチペンひとつで音の確認ができるから、そちらも見て欲しいのう。

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。

```
54 @PLUS
55 PRINT NO1;"+";NO2;"=";NO1+NO2
56 GOTO @LOOP
```

55 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

```
NO3=NO1+NO2
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？

START 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。

MOD 演算子

```
66 @MOD
67 PRINT  NO1;" %" ;NO2;" =" ;NO1  MOD  NO2
68 GOTO  @LOOP
```

そういえば、ここ、あまりの計算のはずだけど、MODって書いてあるんだね。これは新しい命令？

さっきは「%」だって言ってたのに、ハナシがチガくねェか？

よく気づいたのう。「MOD」は命令ではなくて、「＋」や「－」と同じ計算記号の一種じゃ。3 号よりも前のプチコンでは、AをBで割ったあまりを計算するのにA%Bと書いてた時代があったんじゃが、プチコン 3 号では、A　MOD　Bで計算するんじゃよ。

サンプルプログラムで、入力も表示も「%」になっているのは、前に出たプチコンでのサンプルプログラムを使い回したまま、計算の部分だけを修正したからなんだよ。

（インテリ君……！　確かにその通りなんじゃが、今日のキミはとってもシビアなのじゃ……！）

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　MOD　B」と書いて求める「MOD」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 INPUT 命令で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）
0～133 までの種類のサウンドが鳴ります。

スマイルツール

画面の 

SMILE ボタンを押すと起動します。

サウンドの確認をはじめ、いろいろな機能があります。

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？

```
@MUGEN
PRINT"くりかえし"
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（フォー）～NEXT（ネクスト）命令なんだ。
まずこのプログラムを見てごらん。

```
01 FOR A=1 TO 10
02 PRINT"くりかえし"
03 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。

ちゃんとくりかえしを 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。

```
01 FOR A=1 TO 10
02 PRINT"くりかえし";A
03 NEXT
```

これをするとこうなるね。

```
OK
RUN
くりかえし 1
くりかえし 2
くりかえし 3
くりかえし 4
くりかえし 5
くりかえし 6
くりかえし 7
くりかえし 8
くりかえし 9
くりかえし 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！

```
01 FOR A=1 TO 10
02 PRINT"くりかえし"
03 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。

```
01 APPLE=56
02 PRINT"りんご";APPLE;"こ"
```

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"！

```
りんご56こ
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。

```
01 DIM APPLE[1]
02 APPLE[0]=56
03 PRINT"りんご";APPLE[0];"こ"
```

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。

```
りんご56こ
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！

```
01 DIM APPLE[2]
02 APPLE[0]=56
03 APPLE[1]=32
04 PRINT"りんご" ;APPLE[0];"こ"
05 PRINT"りんご" ;APPLE[1];"こ"
```

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……

```
02 APPLE0=56
03 APPLE1=32
04 PRINT"りんご";APPLE0;"こ"
05 PRINT"りんご";APPLE1;"こ"
```

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？

```
DIM APPLE[32]
APPLE[1]=56
APPLE[2]=32
    :
    :
APPLE[31]=45
PRINT"1にちめ りんご";APPLE[1];"こ"
PRINT"2にちめ りんご";APPLE[2];"こ"
    :
    :
PRINT"31にちめ りんご";APPLE[31];"こ"
```

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！

```
APPLE[1]=56
APPLE[2]=32
    :
    :
APPLE[31]=45
PRINT"1にちめ りんご";APPLE[1];"こ"
PRINT"2にちめ りんご";APPLE[2];"こ"
    :
    :
PRINT"31にちめ りんご";APPLE[31];"こ"
```

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！

```
DIM APPLE[32]
APPLE[1]=56
APPLE[2]=32
    :
    :
APPLE[31]=45
FOR DAY=1 TO 31
PRINT DAY;"にちめ りんご";APPLE[DAY];"こ"
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;"にちめ　りんご";APPLE[1];"こ"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;"にちめ　りんご";APPLE[2];"こ"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"なんにちめを　しらべますか";DAY
PRINT　DAY;"にちめ　りんご";APPLE[DAY];"こ"
```

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

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。

```
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」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！

```
APPLE[31]=1
Undefined valiable in 0:1
OK
```

というエラーが出るんだよね。

ヤロー！　こうすりゃいいんだろ！　負けたキブンだぜ！

```
DIM APPLE[32]
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？

ところがプチコンが用意できる変数の数は無限ではないんだ。数値の変数だけならだいたい 100 万個ぐらいまでなら大丈夫だけど、文字列の変数だと入れる文字列が長いほど使える量が減っていったりするし、あとになって足りなくなったら困るだろ？

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……

```
DIM APPLE[32]
APPLE[1]=56
APPLE[2]=32
    :
    :
APPLE[31]=45
FOR DAY=1 TO 31
PRINT DAY;"にちめ りんご";APPLE[DAY];"こ"
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！

```
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
```

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ぜ！

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、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。

```
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……

```
READ  APPLE[1],APPLE[2],APPLE[3]
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。

```
FOR DAY=1 TO 31
READ APPLE[DAY]
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。

```
DIM APPLE[32]
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
```

```
FOR DAY=1 TO 31
READ APPLE[DAY]
NEXT
FOR DAY=1 TO 31
PRINT DAY;"にち りんご";APPLE[DAY];"こ"
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。

```
DIM APPLE[32]
FOR DAY=1 TO 31
READ APPLE[DAY]
PRINT DAY;"にち りんご";APPLE[DAY];"こ"
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。

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。

付録　デザインを変えよう!!

プチコン 3 号・BIG には、編集モードのレイアウトが可能です。

どうするのかな?

トップメニューの「option」をタップして、「編集色」で変更できるんじゃ。

「TEXT」=文字の色

「NUMERIC」=数値の色

「STRING」=文字列の色

「STATEMENT」=

「KEYWORD」=キーワード

「COMMENT」=REM 文の色

「LABEL」=ラベルの色

「BACK」=背景

と分けられるのじゃ。

よし、全部白にするぜー!!

…!?

全部白で見えねーぞ!!

あちゃ…

みんなもやってみてね!!

お問い合わせについて

本書に関するお問い合わせは、プチバース又はプチコンまとめ Wiki にてお願いします。

ご質問につきましては、明確にお願いします。

非公式初心者講座第 2 巻

2018/09/08　初版

著者　takumi

発行者　takumi

発行所　プチコンまとめ Wiki