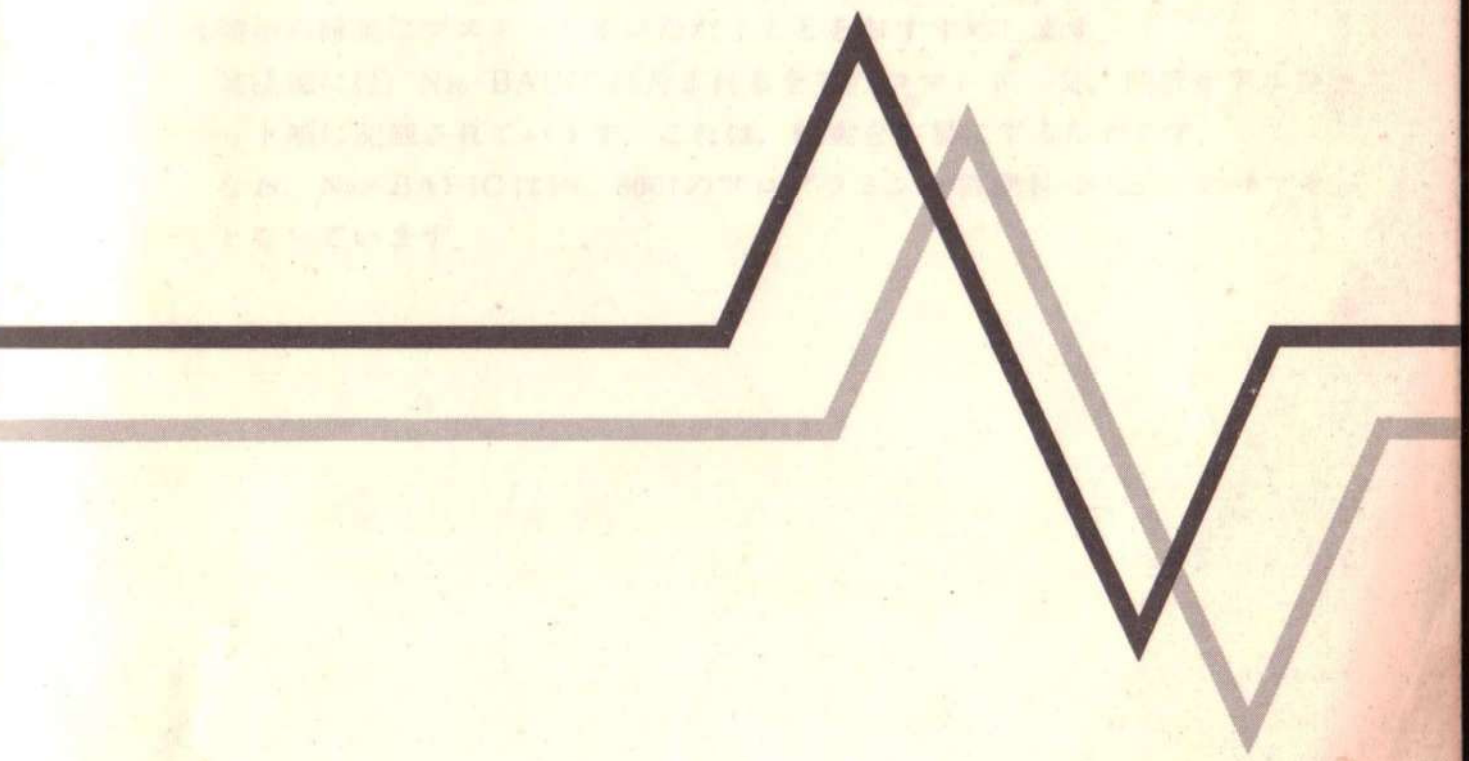
Handheld Personal Computer

PC-2001

N20-BASICマニュアル

NEC

PC-2001
N_{20}-BASICマニュアル



はじめに

本書には，PC-2001のプログラミング言語であるN_{20}-BASICに関する説明がなされています．

PC-2001の取り扱い方法および，オプション機器などに対する解説は「PC-2001取扱説明書」に記載されていますので，そちらを参照してください．

このBASICマニュアルは新しい試みとして，単にBASICのコマンド，文の解説のみでなく初心者から上級者まで使用可能なように，1冊のマニュアルに入門編，文法編の両者を収めて編集されています．

入門編では，原則としてダイレクトモードでコマンド，文を使う場合の例示には RETURN が付加されています．RETURN が付加されていない場合はプログラムの一つの文と考えてください．また，初歩からBASICを学んでいただくために，プログラムの基本から順次内容が高度になっていきますから，基礎から確実にマスターしていただくことをおすすめします．

文法編には，N_{20}-BASICに含まれる全てのコマンド，文，関数がアルファベット順に記載されています．これは，検索を容易にするためです．

なお，N_{20}-BASICはPC-8001のプログラミング言語N-BASICのサブセットとなっています．

PC-2001
N₂₀-BASIC ユーザーズマニュアル

CONTENTS

第1章　入門編

1. 入門編

第2章　文法編

第3章　付録

本機を始め，ほとんどのパーソナル・コンピュータではBASIC言語が最も多く使われています．

BASICは英語のような文章形式で書く言語で，その文章それぞれがコンピュータへの動作指令なのです．

何をどのように処理したいかに応じてこの命令を組み合わせてプログラムを構成すれば，コンピュータに仕事をさせることができます．つまり，プログラムとはデータを処理するための手順を表わしたものなのです．

ところで，BASICで書かれたプログラムはどちらかというと人間にわかりやすくても，コンピュータにはそのままの形ではまったく理解できません．このことはBASICに限らず，その他のプログラミング言語でも同様です．

そこで，コンピュータは与えられたプログラムを自分自身で理解できる言葉(機械語)に翻訳し，それから実行します．

この方法には2通りあります．まず，プログラムを全て機械語に翻訳してから実行する方法です．これを**コンパイラ形式**といいます．もう一つは，プログラムを順番に翻訳しながら実行する方法で，**インタプリタ形式**といいます．

パーソナル・コンピュータに多く使われているBASICは後者によって実行されます．

BASICの特長は，命令や文法がやさしいことと作ったプログラムをすぐ動作させてみることができることなどが挙げられます．

また，会話型になっているので，文法上のルール違反などがあればコンピュータはディスプレイを通じてすぐ指摘してくれますから，誤りを訂正して再び実行することが可能です．

このようにして，作りながら途中で何回も実行してみて気に入らないところを修正したり，誤りを訂正したりしながら次第に完全なものに仕上げていくことができるのもBASICプログラムの大きな特長です．

1. BASICプログラムとは

コンピュータは自分自身で考え，仕事をしてくれるものではありません．全て，人間が指令したことを忠実に実行してくれるだけです．

この指令をプログラムといいます．つまり，プログラムとは私達が人にものを頼むのに書類を書くのと同様に，コンピュータに仕事を頼む文書だと考えることもできます．

コンピュータはプログラムのとおりの仕事しかしません．そのプログラムが，たとえ間違っていたとしても仕事を続け，自分のわからない言葉が出てくれば仕事を途中で放り出します．しかし，運よく全部処理できれば，満足して仕事を終了します．

その結果が万一違っていたとしても，コンピュータの罪ではありません．彼はあなたのいうとおりのことをやったまでで，ほめられることはあっても怒られる筋合いのものではありません．

先ほど，自分にわからない言葉が出てくれば仕事を放り出すと書きましたが，これはプログラムがある一定の規則—文法—で書かれていなければならず，プログラムが文法にちょっとでも違反していた場合に仕事をやめるということです．

人間の世界にも英語，日本語，フランス語など種々の言語があるように，コンピュータの世界にも，その利用目的に応じて種々の言語があります．それらをプログラミング言語といい，たとえば次のようなものがあります．

FORTRAN（フォートラン）
COBOL（コボル）
FORTH（フォース）
BASIC（ベーシック）
PASCAL（パスカル）
PL/I（ピーエル・ワン）
ASSEMBLER（アセンブラ）
Prolog（プロローグ）

2.1 そろばんとメモ—変数について—

人間が計算するときにはそろばんを使ったりしますが，途中経過をメモに残したりもします．

コンピュータでは，計算や処理をCPU(中央演算処理装置)というところで行ない，結果を変数というところにメモしておきます．そして，必要になると変数の内容をみたり，書き換えたりしながら処理を進めていきます．

変数につけられた名前を**変数名**といいます．変数名は255文字以内の英字と数字を組み合わせてつけます．しかし，それぞれの変数は最初の2文字のみで区別されます．また，最初の文字は英字でなければなりませんが，残りの文字は英字または数字のいずれでもよいので，ユーザーが好きな名前の変数を作ることができます．

変数の例

A ， D ， ZZ ， NEC ， PC2001 ， PC

変数の区別は最初の2文字によって行なわれますから，上の例の PC と PC 2001 は同じ変数となってしまいます．

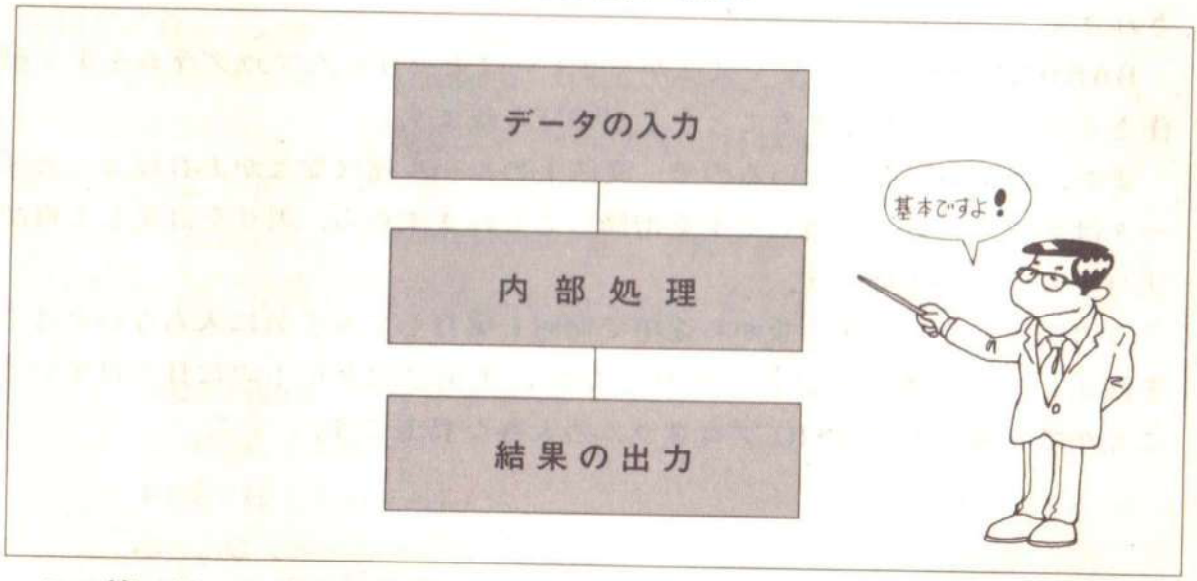
2. プログラムの基本

BASICに限らず，どんなプログラミング言語でも，大きく分けると次の3つの処理しか行ないません．

① データの入力
② 内部処理
③ 結果の出力

普通，この①，②，③の順に処理するようにプログラムを書きます．プログラムを処理していく順を**制御の流れ**，または**処理の流れ**といいますが，基本的な制御の流れを示すと次の図のようになります．

一般的な制御の流れ



この節では，入出力と計算についてのプログラムを書くことにします．

2.2 変数と型

変数には，数値を記憶するための**数値変数**と，文字を記憶するための**文字変数**があります．

また，数値変数の中には整数のみを記憶する整数変数と，小数点つきの数値を記憶する単精度変数があります．

以上のように，変数には3つの形式があり，次のように区別されます．

■**整数変数**：−32768～＋32767までの全ての整数を記憶し，変数名の後に**%**をつけます．

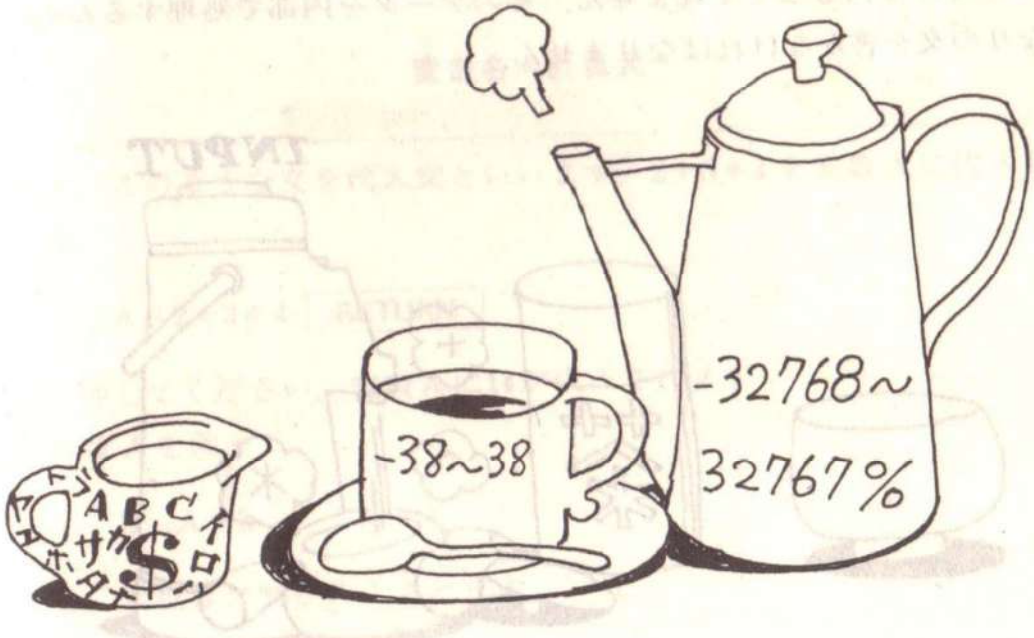
例　**A%** ，**LIMIT%**

■**単精度変数**：有効桁7桁以下で指数部が−38～＋38の範囲にある数値を記憶します．変数名の後に**!**をつけるか，何もつけない場合にこの形式となります．

例　**B!** ，**MINIMUM!** ，**ABC**

■**文字変数**：255文字までの英数字，記号およびカタカナを記憶します．変数名の後に**$**をつけることにより指定します．

例　**C$** ，**MOJI$** ，**NAME$**



また，BASICには処理の都合上すでに予約された単語(PRINT，COSなど)があり，それらと同じか，含んだ変数名をつけることは許されません．

たとえば，

変数名の悪い例

IF ， GOTO ， ENDO ， FORTRAN

などは変数名として使えません．

最初の二つは，予約語IFとGOTOをそのまま使っています．次のENDOは，予約語ENDから始まっています．最後のFORTRANも予約語FORから始まっています．このような予約語を含んだ変数名はつけられません．

予約語については，後で述べます．

2.4 計算をする

内部処理をするための命令は，たくさんあります．ここでは，手始めに計算をさせることにします．計算をするには，変数名や数値と，"＋"，"－"，"＊"，"／"，"へ"を組み合わせて計算式を作ってやります．コンピュータでは，"＊"は掛け算の"×"を，"／"は割り算の"÷"を，"へ"はべき乗を表わします．

たとえば，Aを2倍するには，

2＊A　または　A＊2

とします．そして，これらの計算の順序は数学で行なう順序と同じで，べき乗，乗除算，加減算の順序となります．

A+3＊B

とすると，3＊Bを先に行ない，その結果にAを加えることになります．もちろん，"(" や ")" を使って計算の順序を指定することも，三角関数などの関数を使うこともできます．

つまり，計算式どおりにキーボードから打ち込んでやれば，BASICを使って計算を行なうことができるわけです．

また，計算した結果を適当な変数にしまっておくことを**代入する**といいます．代入するには，

変数名＝計算式

とします．このような文を**代入文**といいます．2+3＊4を変数Aに代入してみましょう．

A＝2+3＊4 RETURN

とキーを押してください．変数Aに14が代入されます．

この変数Aを使って，

B＝A＊2

とすると，変数Bに28が代入されます．

2.3 データを与える—INPUT文—

キーボードやカセットから，変数にデータを与えることを**入力**といいます．入力命令はいくつかありますが，ここではINPUTを使うことにします．INPUTの後に，入力したい変数の変数名をつけると，初めて**INPUT文**という文になります．また，たくさんの変数に入力するときは，INPUT文を一つずつ書く代わりに，変数名を" , "で区切って並べることにより，一つの文ですませることもできます．INPUT文の一般形は，次のとおりです．

INPUT　変数名
INPUT　変数名１，変数名２，……

たとえば，Aという変数に入力するには，

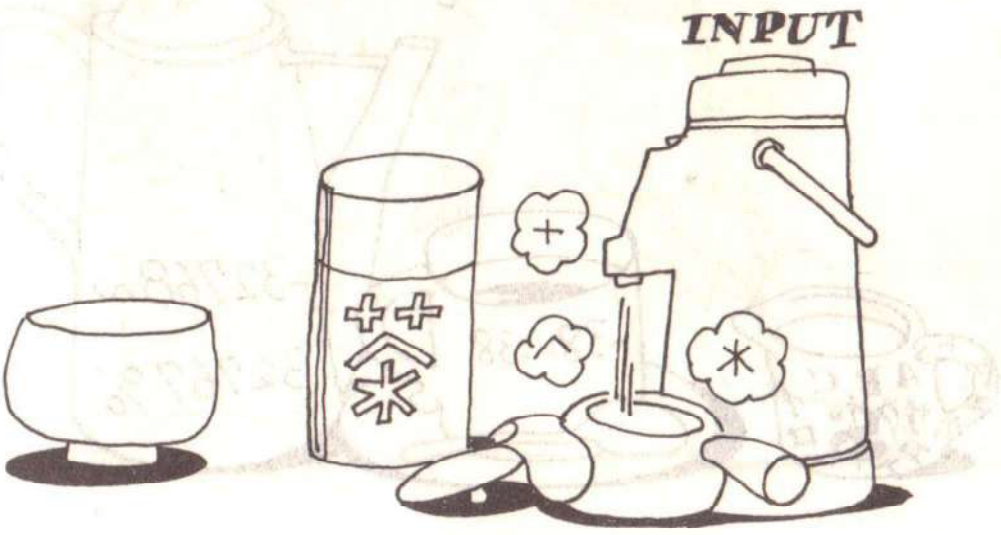
```
INPUT  A
```

AAとBBという変数に入力するには，

```
INPUT  AA , BB
```

と書きます．

さて，データを与えるには，INPUT文を使いますが，それだけではデータを与えるだけで何もしてくれません．そのデータを内部で処理するためには，それなりの文を書かなければなりません．



2.5 結果がなければ意味がない－PRINT文－

データを入力し，処理をさせてもそのままでは私達の欲しい結果を出してくれません．変数の内容や計算式の結果を表示する命令がPRINTです．
INPUT文と同じように，PRINTの後に変数名や式を並べて書くと，その変数の内容や式の値を表示してくれます．一般にPRINT文は次のように書きます．

PRINT　変数名または式
PRINT　変数名または式，変数名または式……
PRINT　変数名または式；変数名または式……

これらの表示方法で，","や";"が入っていますが，それぞれ意味を持っています．

","で区切ると，14文字分ずつの領域に分けながら変数の内容や式の値を表示します．また,";"で区切ると次の値を直前に表示したすぐ後に続けて表示します．

本機は電卓のように計算をすることもできますが，手計算で行なうときは，必ず最初にPRINTをつけます．

また，PRINTと打つのが面倒な場合は，代わりに"?"としても同じ働きをします．

電卓のような手計算を行なってみましょう．電源を入れて，

```
PRINT2+3*4
```

または，

```
?2+3*4
```

と押してください．最後に，

RETURN

と書いてあるキーを押してください．"14"と表示されます．RETURNキーは私達の文章でいう句読点のようなもので，文の終わりをコンピュータに教えてやるためのものだと思ってください．

また，代入文は，

A＝A＋1

というように書くこともできます．少し見慣れない計算式ですが，これは，

AはA＋1である

とは読まずに，

A＋1の結果をAに代入せよ

と読みます．

ところで，計算したり，代入したりするだけではコンピュータは結果を出してくれません．結果を表示するための文が必要です．それが**PRINT文**です．

```
 14                70
PRINT    INPUT    GOTO     LIST     RUN.
```

のように表示されます．また，

PRINT A;B RETURN

とすると，

```
 14  70
PRINT    INPUT    GOTO     LIST     RUN.
```

のように表示されます．

このように，PRINT文の区切りを","または";"にすると表示の形式が変わります．

2.6 簡単なプログラム

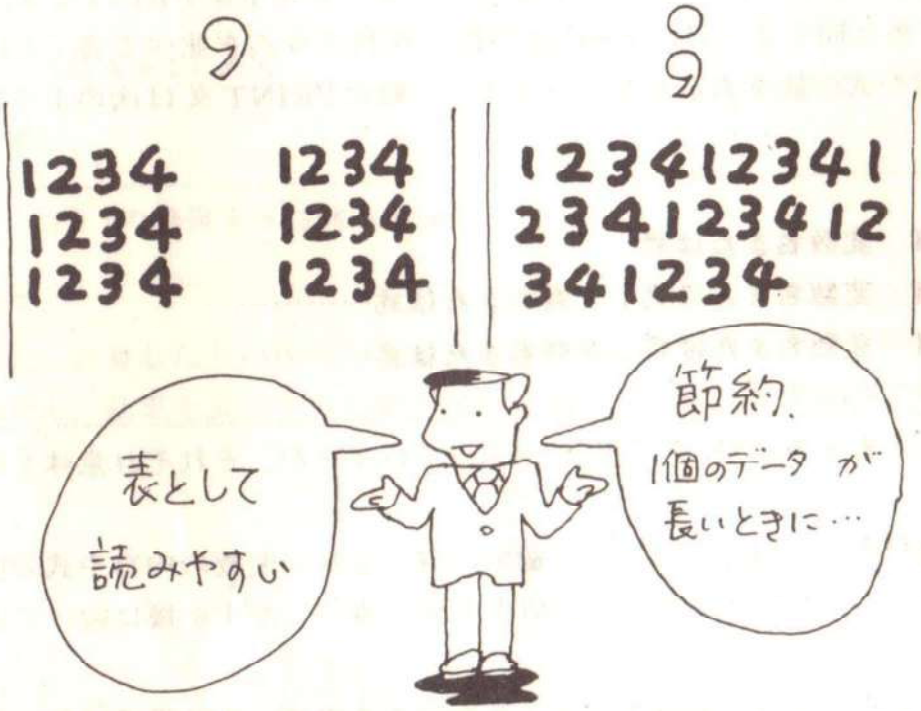
クドクドと書きましたが，実はINPUT文，代入文，PRINT文だけでも立派にプログラムは書けるのです．他の命令は，たくさんあるプログラムを要領よく書くための付録にすぎません．

3つの文を使ってプログラムを書いてみましょう．いよいよあなたもプログラマーの仲間入りです．

例 **利率と預け入れ期間(年)および元金から受け取り額を求めるプログラムを作る．利率はパーセント単位で与え，複利計算とする．**

このままの状態で何かキーを押してもまったく受けつけてくれません．1回表示した後は，必ず RETURN キーを押してください．

これにより，キーを受けつける状態，コマンドレベルに戻ります．



次に，変数の内容を表示させてみましょう．まず，

A＝2＋3＊4 RETURN

として，変数A に2＋3＊4を代入してください．次に，

PRINT A RETURN

としてください．先ほどまでと同じように"14"と表示されます．

RETURN キーを押してから，今度は，

B＝5＊A RETURN

として，変数BにAの5倍を代入してから，

PRINT A，B RETURN

と押してください．

て，これらの変数を使って先ほどの受け取り額を示す式を書いてみると，

```
GOKEI=GANKIN*(1+RIRITSU/100)^KIKAN
```

となります．これで，受け取り額を与える代入文ができました．入力として必要なデータは，利率，預け入れ期間，元金の3つですからINPUT文は，

```
INPUT GANKIN,KIKAN,RIRITSU
```

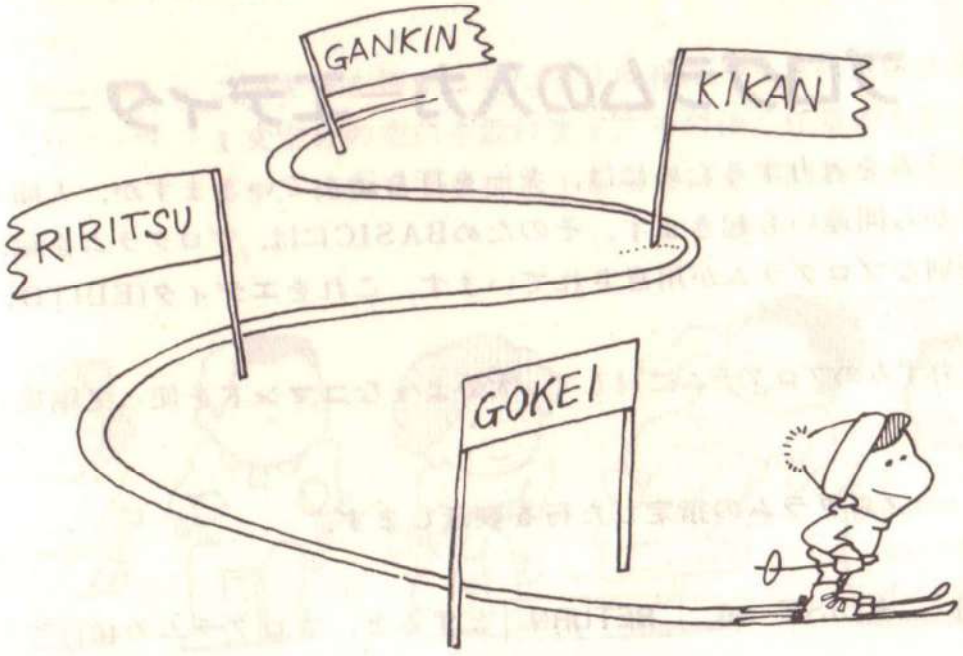
となります．さらに，結果を出力するPRINT文は，受け取り額を示す変数がGOKEIですから，

```
PRINT GOKEI
```

となります．順番に並べてみましょう．処理する順番を考えると，

```
INPUT GANKIN,KIKAN,RIRITSU
GOKEI=GANKIN*(1+RIRITSU/100)^KIKAN
PRINT GOKEI
```

と書けばよいわけです．**「上から下へ向かって処理していく」**という取り決めさえしておけば，必ず仕事をしてくれます．このようにして箇条書きで整理します．



プログラムを作るためには，まず何をどのようにするのかを箇条書きなどにして整理します．この場合，

1) 利率，預け入れ期間，元金の入力
2) 受け取り額の計算
3) 受け取り額の表示

となります．このように箇条書きにすると，INPUT文，代入文，PRINT文の3つを使うことがよくわかると思います．

次に受け取り額を求める計算式を考えましょう．下図を見てください．

元金と受け取り額の関係

現　在	元金
1年後	元金＊(1＋利率)
2年後	元金＊(1＋利率)＊(1＋利率)
3年後	元金＊(1＋利率)＊(1＋利率)＊(1＋利率)

元金は1年後に(1＋利率)倍になります．2年後にはさらに(1＋利率)倍になります．すると，複利計算による受け取り額は，

$$受け取り額 = 元金 * (1 + 利率)^{期間}$$

となります．

ここで変数名を決めます．変数とするのは，**利率**，**預け入れ期間**，**元金**，**受け取り額**の4つがあります．したがって変数名も4つ必要なので，次のようにつけてみます．

RIRITSU ……利率
KIKAN……預け入れ期間
GANKIN……元金
GOKEI……受け取り額

受け取り額をUKETORIとしたいところですが，TOが予約語なのでできません．これらの変数には型指定文字がついていませんから，単精度変数です．そし

ます.

★LIST RETURN とすると，プログラムの最初の行を表示します.

DELETE……**指定した行，または指定した行から行までを消去します.**

例 ★DELETE 40 RETURN とすると，プログラムの40行を消去します.

★DELETE 40－70 RETURN とすると，プログラムの40～70行を消去します.

★DELETE －40 RETURN とすると，プログラムの最初の行から40行までを消去します.

★DELETE 40－ RETURN とすると，プログラムの40行から最後の行までを消去します.

NEW……**プログラムとデータの全てを消去します.**

上に挙げたコマンドはプログラムの各行について作用するものですが，次の各キーを使うと1行のある部分を消去したり，ある部分に挿入したりすることができます.

INS キー…… このキーを押すことにより現在カーソルの位置する場所に1文字分の空白を設けます．その後，任意の文字を挿入します．複数文字を挿入する場合はその回数分キーを押します.

ところで，BASICの処理の順番はどうなっているのでしょうか．文を入力した順でしょうか．ところが，それだけでは大変都合が悪いのです．なぜなら，何かの都合で，文と文との間に文を挿入するためには，最初から全ての文を入力し直さなければならないからです．そのために，BASICでは，文の先頭に番号をつけ，その番号の小さい順から処理していくという取り決めがされています．この番号を**行番号**といい，0～65,529の整数を用いることができます．

これからは行番号のつけられた文を，行ということにします．私たちの作った文に行番号をつけてみると，次のようになります．

```
10 INPUT GANKIN,KIKAN,RIRITSU
20 GOKEI=GANKIN*(1+RIRITSU/100)^KIKAN
30 PRINT GOKEI
40 END
```

40行の**END**は，これでプログラムの実行を終了するという命令で**END文**といいます．

行番号が飛び飛びになっていますね．賢明な方ならおわかりと思いますが，1，2，3，……とつめて行番号をつけると，後で1行目と2行目の間にあらたに行を挿入しようとしても，1と2の間に整数がないので，挿入できません．そんな用心のために，行番号は普通，10おきとか，100おきのように飛び飛びにつけます．

2.7 プログラムの入力－エディタ－

プログラムを入力するためには，キーを打ち込んでいきますが，人間がやることですから間違いも起きます．そのためBASICには，プログラムを編集するための特別なプログラムが用意されています．これを**エディタ**(EDITOR)といいます．

現在入力ずみのプログラムに対して，次のような**コマンド**を使って編集していきます．

LIST……プログラムの指定した行を表示します．

例　★LIST 40 RETURN とすると，プログラムの40行を表示し

さて，前節で作ったプログラムを入力してみましょう．

```
10 INPUT GANKIN,KIKAN,RIRITSU
20 GOKEI=GANKIN*(1+RIRITSU/100)^KIKAN
30 PRINT GOKEI
40 END
```

まず，

NEW RETURN

として，PC-2001 の中を全て消去してください．そのうえで，プログラムを10行から入力してください．

全部の入力が終わったら，

LIST RETURN

とすると，10行を表示します．さらに，▼キーを押していくと，20行，30行と表示が進んでいきます．逆に▲キーを押すと表示が戻っていきます．それにより，プログラムを確認して誤りがあれば前述の方法で修正します．

誤りがなければ，

RUN RETURN

としてください．RUNとは，「プログラムを実行せよ」というコマンドです．RUNすると，すぐに，

? _

と表示されます．これは，入力の要求です．INPUT文により入力する変数の値は何か，と聞いてきているのです．入力するデータは，INPUT 文で指定した順番に対応します．

いま，データを，

元金……１００００（円）
期間……3　　（年）
利率……8　　（%）

DEL キー…… このキーを押すことにより現在カーソルの位置する左側の文字を消去します．複数文字を消去する場合はその回数分キーを押します．

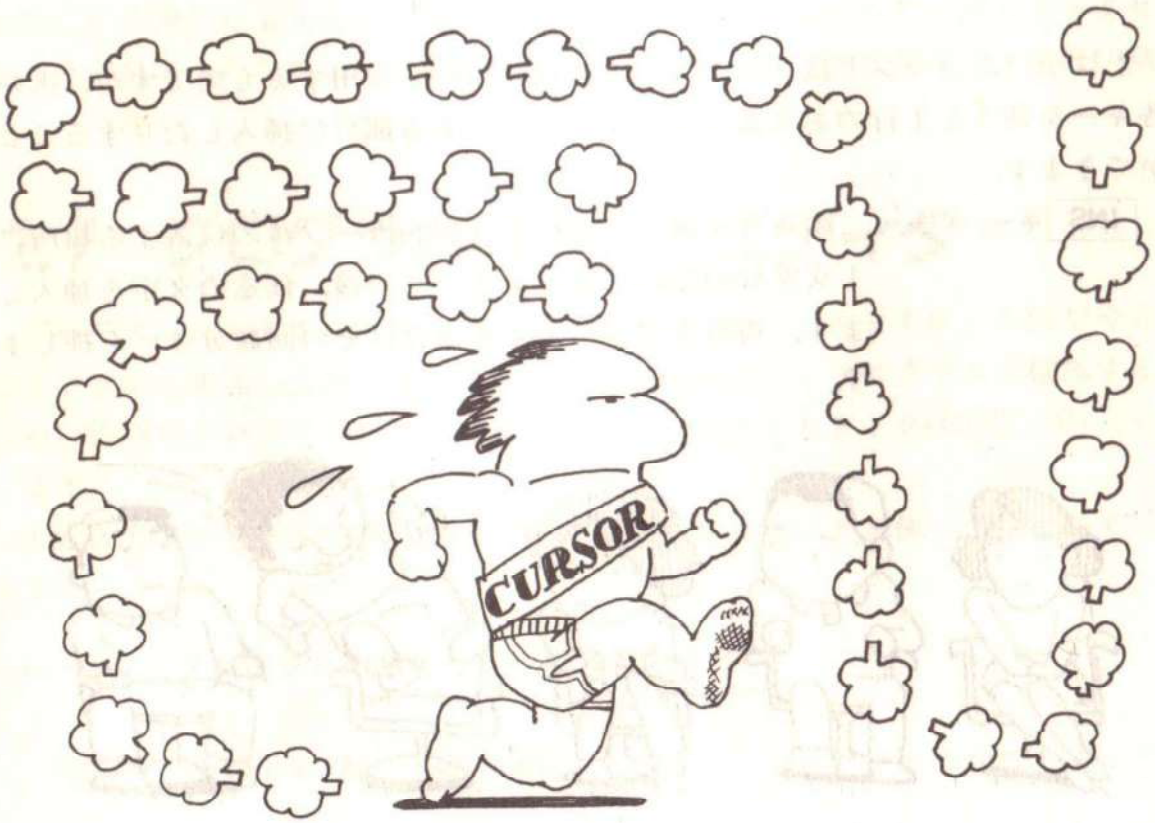
▶ キー …… このキーを押すことによりカーソルが右に1文字分移動します．

◀ キー …… このキーを押すことによりカーソルが左に1文字分移動します．

▲ キー …… 現在表示している行の一つ前の行を表示します．

▼ キー …… 現在表示している行の次の行を表示します．

注）カーソルは，キー入力をする場合に常に出ている下線（アンダーバー）で，入力したキーがその位置に入ることを意味する目印となります．



文による表示は，現在の表示に続けて行なわれます．

これらを使って預金計算プログラムを書き直してみました．入力して，実際に確かめてください．前のものより，はるかに使いよくなっています．

```
10 INPUT "ｶﾞﾝｷﾝ(¥)=";GK
20 INPUT "ｷｶﾝ(ﾈﾝ)=";KI
30 INPUT "ﾘﾘﾂ(%)=";RI
40 GA=GK*(1+RI/100)^KI
50 PRINT "ｳｹﾄﾘｶﾞｸ=";GA
60 END
```

とすると，最初の？に対して，

10000 RETURN

とします．次に，

??_

と表示されます．？が2個並ぶのは2番め以後の入力データであることを示します．元金と同様にして期間，利率を入力すると，受け取り額が表示されます．

```
3        RETURN    ……… 期間の入力
8        RETURN    ……… 利率の入力
12597.1            ……… 受け取り額
```

2.8 入力メッセージ

利率や期間などの入力に"？"だけでなく，メッセージを表示してみましょう．それには，INPUT文を次のようにします．

```
INPUT "メッセージ"; 変数名
```

このようにすると，入力データを入れ終わるまでメッセージが表示されていますから，データ入力を確認しながら行なうことができます．

もちろん，

```
INPUT "メッセージ"; 変数名1，変数名2，…
```

のように使うこともできます．

結果の表示に対してもメッセージをつけることができます．

```
PRINT "メッセージ"; 変数名
```

とします．また，PRINT文の最後に","または";"をつけると，RETURNキーを押す必要なしに実行を継続するようになります．そして，以降のPRINT

ですから，条件は正しいのですが逆に，

A＝0

とすると，

0＞1

は，正しくありません．このように，条件には正しい場合と正しくない場合があります．正しい場合を**真**，正しくない場合を**偽**といいます．

BASICのIF文では，条件が真のときのみ**THEN**より右にある文を実行します．次のIF文は，

IF　A＜3　THEN　B＝B＊2

Aが3より小さい場合に限り，Bを2倍してBに代入します．

また，条件が真のときはTHEN以下の文を実行し，条件が偽のときはELSE以下の文を実行するような形のIF文を書くこともできます．

たとえば，

IF　A＞5　THEN　B＝A　ELSE　B＝A＊2

とすると，Aが5より大きい場合BにAを代入し，Aが5以下の場合BにAの2倍を代入します．

3. 繰り返しと判断

この節では，判断，分岐および判断と分岐を組み合わせた繰り返しについて述べます．

代入文とこれらを組み合わせると，ほとんどの処理をプログラムとして書くことができます．それほどこれらの機能は強力ですが，言い換えれば代入文よりは難しいものです．この点は，ちょうど薬と同じです．**強い薬ほど，副作用も強いもの**です．世の中便利なことばかりではありません．

3.1 もし，～ならば(真と偽)

「もし，雨が降ればナイターは中止である」という言葉を考えてみましょう．これには，雨が降るかどうか，という判断が入っています．BASICでもこのような判断を行なうことができます．それが**IF文**です．今の文章をIF文によって表わしてみると，

IF　雨が降る　THEN　ナイターは中止

となります．この「雨が降る」に相当するものを**条件**といいます．条件とは判断の材料になるものです．条件の例をいくつか挙げてみます．

A＞1　，A＊B＝5　，A－B＜C

などです．ここで使われている "＝" は代入文のときと違い，本当の意味の「等しい」として使われます．ところで，Aは変数ですから，一口に，

A＞1

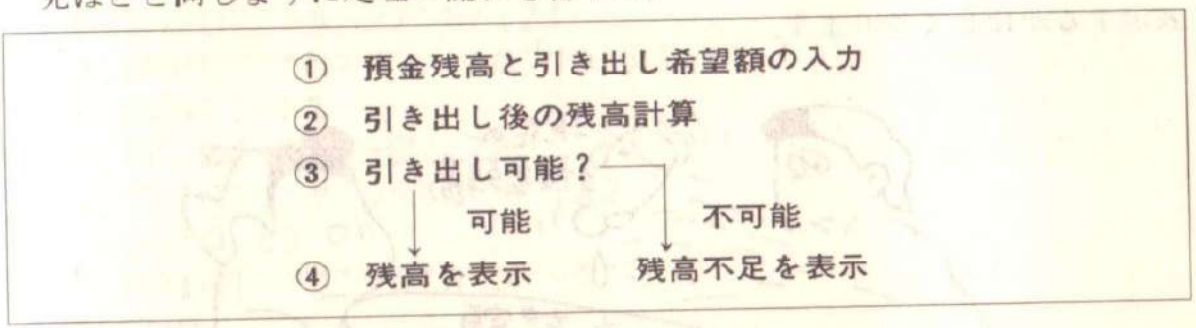
と言っても，Aの値によっては条件が正しくない場合があります．たとえば，

A＝3

とすると，

3＞1

先ほどと同じように処理の流れを書くと，



のような形になります．

このうち，①，②，④はそれぞれ入力，代入，出力ですから，前に作った預金利息計算のプログラムと同じように作ることができます．

変数をそれぞれ，

Z 1……現在の預金残高
K I……引き出し希望額
Z 2……引き出し後の残高

とすると，①については，

```
INPUT "ｻﾞﾝﾀﾞｶ";Z1
INPUT "ﾋｷﾀﾞｼ ｷﾎﾞｳｶﾞｸ";KI
```

となり，②は現在の預金残高から引き出し希望額を引けばよいので，

```
Z2=Z1-KI
```

となります．

さらに④は残高表示が，

```
PRINT "ｻﾞﾝﾀﾞｶ =";Z2
```

残高不足の表示が，

```
PRINT "ｻﾞﾝﾀﾞｶ ﾌﾞｿｸ"
```

となります．

③に入りますが，引き出し可能というのはZ2の値が0~~より大きい~~以上のときで，引き出し不可能の場合は，Z2の値が負の場合です．判断が加わるわけですから，

このように，IF文は判断をすることができます．また，THENおよびELSEの右には，代入文ばかりでなく全ての文を書くことができます．さらに，次のGOTO文と同時に使うことによって，制御の流れをある条件のときだけ変えることもできます．

条件を示す記号にはいくつかありますが，≧などはキーにありませんので，ちょっと違う書き方をします．それらを表にまとめておきます．

条件とその意味

通常の記法	BASICでの記法	意　味
A＝B	A = B	AはBと等しい
A＞B	A > B	AはBより大きい
A≧B	A>=B	AはB以上
A＞B	A < B	AはBより小さい
A≦B	A<=B	AはB以下
A≠B	A<>B	AはBと等しくない

3.2 分岐

プログラム中での制御の流れは，通常行番号の小さい順に行なわれますが，ある処理を飛び越したり，前の方へ戻ったりしたい場合があります．その場合に，GOTO文が使われ，次のように書きます．

GOTO　行番号

GOTO文の機能は，その行番号へ飛ぶ，つまり制御を移すことです．

GOTO　100

とすると，次に100行を実行します．それでは，実際にIF文とGOTO文を使ってみましょう．

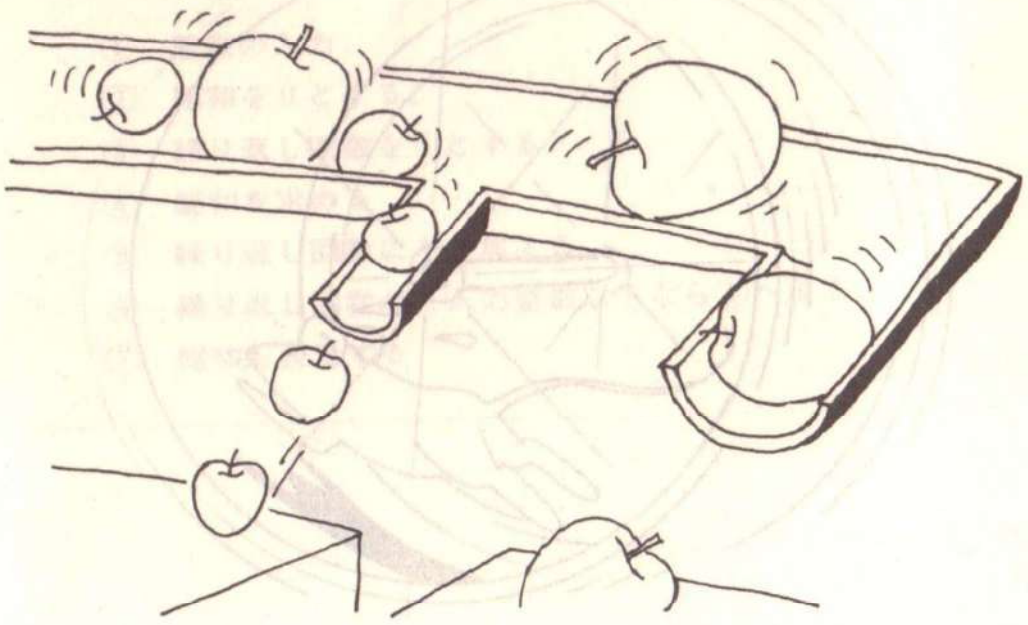
例　預金残高と引き出し希望額を入力すると引き出し可能ならば，引き出し後の残高を，不可能ならば残高不足を表示するプログラムを作る．

```
10 INPUT "ｻﾞﾝﾀﾞｶ";Z1
20 INPUT "ﾋｷﾀﾞｼ ｷﾎﾞｳｶﾞｸ";KI
30 Z2=Z1-KI
40 IF Z2<0 THEN 70
50 PRINT "ｻﾞﾝﾀﾞｶ =";Z2
60 GOTO 80
70 PRINT "ｻﾞﾝﾀﾞｶ ﾌﾞｿｸ"
80 END
```

となります．プログラムを入力して，実行してみてください．

上のプログラムは，GOTO文によって分岐していますが，IF ～ THEN ……ELSE……の形のIF文を使えば，Z2＜0のときはTHEN以下を，そうでないときはELSE以下を実行するように書くこともできます．この場合にGOTO文は必要なく，次のようなプログラムになります．

```
10 INPUT "ｻﾞﾝﾀﾞｶ";Z1
20 INPUT "ﾋｷﾀﾞｼ ｷﾎﾞｳｶﾞｸ";KI
30 Z2=Z1-KI
40 IF Z2<0 THEN PRINT "ｻﾞﾝﾀﾞｶ ﾌﾞｿｸ" ELSE
   PRINT "ｻﾞﾝﾀﾞｶ =";Z2
50 END
```



ここにはIF文を使います．Z2の値によって，残高を表示するか，残高不足を表示するかにしてやります．

さて，紙に処理の流れを書く場合には，左右に分けて書けますが，プログラムは上から下への一方にしか書けませんのでGOTO文を使って処理の流れを変えてやります．

```
INPUT "ｻﾞﾝﾀﾞｶ";Z1
INPUT "ﾋｷﾀﾞｼ ｷﾎﾞｳｶﾞｸ";KI
Z2=Z1-KI
IF Z2<0 THEN
PRINT "ｻﾞﾝﾀﾞｶ =";Z2
GOTO
PRINT "ｻﾞﾝﾀﾞｶ ﾌﾞｿｸ"
END
```

残高不足を表示する行へ飛ぶ

処理終了へ飛ぶ

矢印のように飛ぶようにGOTO文を使います．IF文を使って，ある行へ飛ばしてやるには，

IF　条件　THEN　行番号

としてGOTOを省くこともできます．

さあ，行番号をつけてプログラムを完成させましょう．

何回か処理を行なったのち，終了させるためにはどうするのでしょうか．その場合に，IF文を使います．

処理の回数を数えておいて，IF文によりあらかじめ決められた回数以上になったら，終了するようにしてやります．しばしば使われる方法に次のようなものがあります．ここで，Nは繰り返し回数です．

```
N＝1
繰り返し処理部分
N＝N＋1　←……(Nに1を加えたものをNとする代入文)
IF　N＜＝繰り返し終了数　THEN　繰り返し処理開始行番号
```

最初に繰り返し回数を1にしておきます．繰り返し処理が1回終わったときに，その繰り返し回数に1を加えて，IF文で繰り返し処理を続けるかどうかを判断します．

実際にプログラムを作るとわかりますが，繰り返し処理は，必ずといってよいほど出てきます．それでは，実際にプログラムを作ってみましょう．

例　**1から入力した整数までの整数和を求めるプログラムを作る．**

例によって，処理の流れを書くと次のようになります．

① 整数の入力
② 総和を0とする．
③ 繰り返し回数を1とする．
④ 総和を求める　←
⑤ 繰り返し回数に1を加える
⑥ 繰り返し回数が，入力整数以下なら④へ戻る
⑦ 総和を表示する

3.3 繰り返し処理

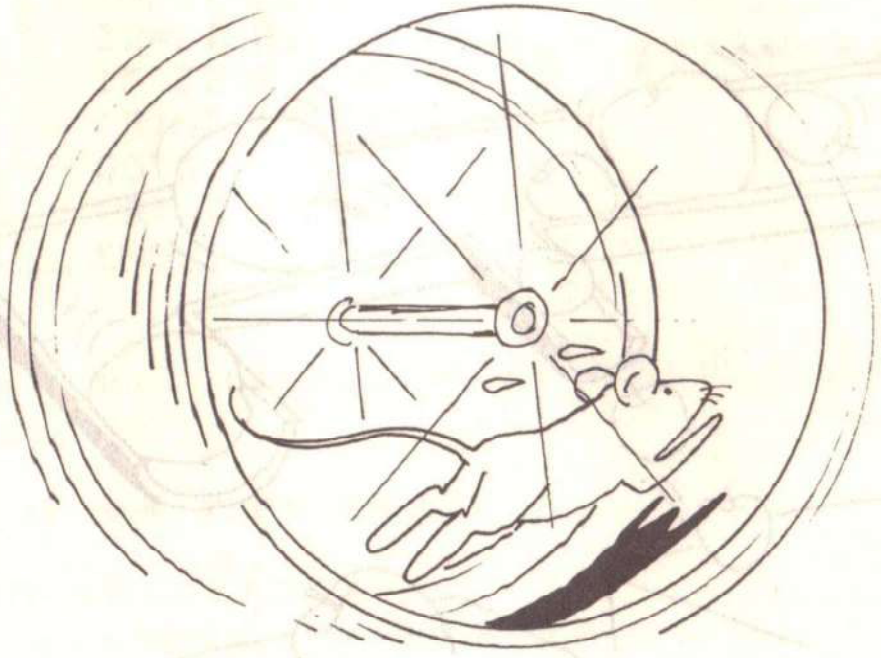
GOTO文によって飛ぶのは，後の行へばかりではありません．前の行へ飛ぶこともでき，そのような使い方をすることで繰り返し処理を行なうことができます．いま作った残高計算プログラムのEND文を

```
GOTO  20
```

としてみましょう．以前は，一度計算を終了するともう一度処理を行なうためには再びRUNしていましたが，このようにGOTO文を使うと結果の表示後，

RETURN

を押すことにより，再び引き出し希望額の入力へ戻り，何回も繰り返し処理できるようになります．しかし，このプログラムは一度実行したら永久に実行を続けます．処理の終了を知らせるENDがないからです．このような処理，すなわち一度実行したら終了しないような処理を**永久ループ**または**無限ループ**といいますが，間違いではありません．そのようなプログラムとしたからです．



となります．

このプログラムでNを10としたときの総和を求めると45になります．これは，

$$1+2+3+4+5+6+7+8+9$$

の結果です．

さらに，30～50行を，

```
30 LP=0
40 LP=LP+1
50 WA=WA+LP
```

とした以下のプログラム，

```
10 INPUT "N=";N
20 WA=0
30 LP=0
40 LP=LP+1
50 WA=WA+LP
60 IF LP<=N THEN 40
70 PRINT "WA=";WA
80 END
```

でNを10として実行すると総和は66となります．これは，1～11の総和です．

このようにIF文を使って繰り返し処理をするときは，繰り返し回数を増加したり，終了の判断をどこで，どのように行なうかに注意が必要です．

また，1～Nまでの総和Sは公式，

$$S=n(n+1)/2$$

によっても求めることができます．

次に変数名を決めましょう．

N……入力する整数
WA……総和
LP……繰り返し回数

と決めると，プログラムは次のようになります．

```
10 INPUT "N=";N
20 WA=0
30 LP=1
40 WA=WA+LP
50 LP=LP+1
60 IF LP<=N THEN 40
70 PRINT "WA=";WA
80 END
```

20行は，総和を0にしておくためのものです．なお，Nを10として計算すると，総和は，

$$1+2+3+4+5+6+7+8+9+10$$

となります．総和を求めるには繰り返した回数を順々に加えていけばよいわけです．

60行にある"＜＝"と"＜"では条件が違うことに注意してください．

たとえば，条件を"＜"にしてみるとプログラムは，

```
10 INPUT "N=";N
20 WA=0
30 LP=1
40 WA=WA+LP
50 LP=LP+1
60 IF LP<N THEN 40
70 PRINT "WA=";WA
80 END
```

```
10 INPUT "N=";N
20 WA=0
30 FOR LP=1 TO N
40 WA=WA+LP
50 NEXT LP
60 PRINT "WA=";WA
70 END
```

となります．30行を，

30 FOR LP=1 TO N STEP 1

とした人もいるでしょう．それも正解ですが FOR文は増分が1の場合に限り，STEP以下を省略してもかまいません．

さて，公式S＝n(n＋1)/2を使えば1～Nの総和は計算できるのに，IF文やFOR～NEXT文によって処理した理由は，このプログラムを多少変更すればいろいろなプログラムを作ることができるからです．

たとえば，1～Nまでの奇数の和は，次のようになります．

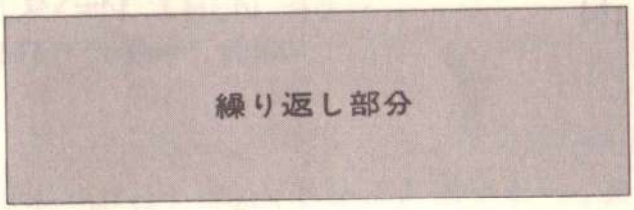
4. 強力な繰り返し機能 —FOR~NEXT文—

FOR～NEXT文は，繰り返し処理に必要な，

```
N＝O
N＝N＋1
IF N＜＝ 繰り返し終了数 THEN 繰り返し処理開始行番号
```

の機能を，1つにまとめたような文です．次のように使います．

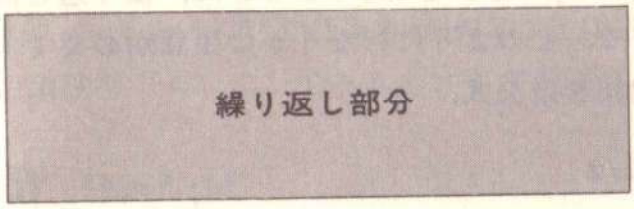
```
FOR 制御変数名＝初期値 TO 終値 STEP 増分
```




```
NEXT 制御変数名
```

これと同じことをIF文とGOTO文を使って書くと，

```
変数名＝初期値
```




```
変数名＝変数名＋増分
IF 変数名＜＝ 終値 THEN 繰り返し開始行番号
```

となります．FOR～NEXT文の方が，はるかにわかりやすく，しかも処理速度も速くなります．

先ほどの「1からNまでの総和を求めるプログラム」をFOR～NEXT文を使うように改めてみましょう．変数名をそのままとすると，

いること，また，制御変数名は異なったものを使用すること．

❺制御変数の値を，ループ内で変更しないこと．

以上の点を詳しく書くと次のようになります．

❶図のように，左側の例は許されますが，右側の例は許されません．

飛び込み禁止

```
100  FOR I=1 TO 10          100  GOTO 150
 :                            :
150  GOTO 200               120  FOR I=1 TO 10
 :                            :
190  NEXT I                 150  A=B*C
200  .....                   :
                            180  NEXT I
```

飛び出す場合——可　　飛び込む場合——不可

❷制御変数に増分を加えるのに増分が0では，いつまでたっても制御変数の値が変わりませんから，ループから抜けられず，永久ループになります．STEPに計算式を使ったときに起こりやすいので注意してください．

❸以下のように，最初からループする条件から外れていても，必ず1回はループ内の処理をするということです．

```
FOR  I=5 TO 3
FOR  J=0 TO 10 STEP -1
```

```
10 INPUT "N=";N
20 WA=0
30 FOR LP=1 TO N STEP 2
40 WA=WA+LP
50 NEXT LP
60 PRINT "WA=";WA
70 END
```

また，1～Nまでの積を計算するプログラムは次のようになります．

```
10 INPUT "N=";N
20 SK=1
30 FOR LP=1 TO N
40 SK=SK*LP
50 NEXT LP
60 PRINT "1 カラ";N;"マデ゛ノ セキ ﾊ";SK
70 END
```

4.1 初期値，終値，増分

PC-2001のFOR～NEXT文では，**初期値，終値，増分**の値として，

−32,768～+32,767

までの整数をとることができます．初期値より終値が小さい場合でも，増分の値を負にすることによって制御変数が，だんだん小さくなるようなループを作ることも，初期値，終値，増分として計算式をあてることもできます．最後に，FOR～NEXT文についての注意をいくつか挙げます．

❶FOR～NEXT文で作るループの中からGOTO文で飛び出してもよいがFOR～NEXT文の外から飛び込むことは禁止される．
❷増分として，0をとることは無意味である．
❸FOR～NEXT文によるループ処理は，必ず1回は行なわれる．
❹いくつかのFOR～NEXT文を使って多重ループを作ってもよいが，内部にあるFOR～NEXT文は，外部のFOR～NEXT文に完全に含まれて

5. マルチステートメント

1行には2つ以上の文を書くこともできます．PC-2001では1行に255文字まで書くことが許されますから，その範囲内でなら，文と文を":"で区切るだけで，つなげて書くことが可能です．これを**マルチステートメント**といいます．

2行以上をマルチステートメントを使って1行にまとめて書くと全体のサイズが若干小さくなりますが，その反面プログラム全体が読みにくくなります．

マルチステートメントの例として，先ほどの1～Nまでの整数和を1行に書くと，

```
10 INPUT "N=";N:WA=0:FOR LP=1 TO N:WA=WA+LP
   :NEXT LP: PRINT "WA=";WA:END
```

となり，非常に見にくくなります．

ですから，マルチステートメントを使うのはプログラムが大きくて少しでもメモリに余裕を持たせたい場合と，IF文の中で2つ以上の処理を行なう場合に限るのが一般に無難です．

IF文の中で使うのは次のような場合です．

```
100 IF A=1 THEN B=B+1:C=C+1 ELSE B=0:C=0
110 ・・・・・・・・・・・
```

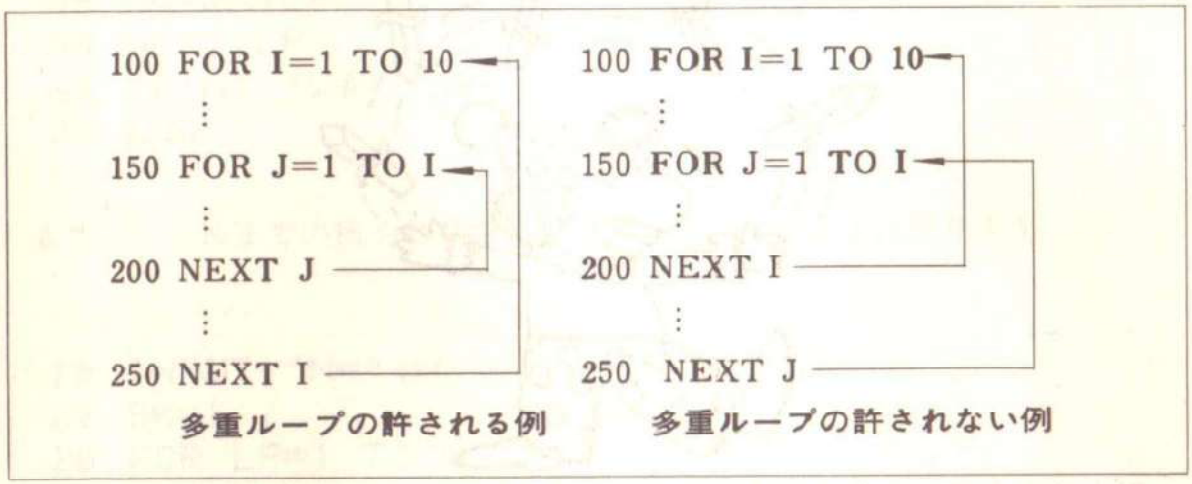
もし，A＝1ならばBとCに1を加え，A＝1でなければBとCを0とするということを行なったものです．

これをマルチステートメントを使わずに書くと，

```
100 IF A=1 THEN 140
110 B=0
120 C=0
130 GOTO 160
140 B=B+1
150 C=C+1
160 ・・・・・・・・・・・
```

❹図のように，ループを作った場合です．左側の例は許されますが，右側の例は許されません．

多重ループが許されない例



上図のように，対になるFOR文とNEXT文を結んでみて，その線が交わるような使い方はできません．

❺制御変数をループ内で変化させると，その変化した値により，ループ終了の判断をするということで，このような使いかたができないという意味ではありません．

6. サブルーチンと共通処理

私達の世界では，一つの仕事を何人かで分担したり，製造業では下請けという方法も使われています．プログラムの世界でも，このような方法を使います．一つのプログラムをいくつかの小さなプログラムの集まりとしたり，同じ処理を行なっている部分を共通の処理として一つにまとめたりします．これらの下請けに使われるプログラムを**サブルーチン**といいます．

また，親にあたるものを**メインルーチン**といいます．

6.1 サブルーチンの使いかた

例　**1～Nまでの整数に対して，$\sqrt{x}$，x^2，x^3の値を求めるプログラムを作る（$\sqrt{x}$を求めるには関数SQR(X)を使う）．**

このプログラムは入力部と出力部をメインルーチンにおき，計算部はサブルーチンにすることにします．$\sqrt{x}$を示す変数をXR，x^2を示す変数をX2，x^3を示す変数をX3とすると，計算部は次のようになります．

```
100 XR=SQR(X)
110 X2=X*X
120 X3=X^3
130 RETURN
```

130行の**RETURN**はサブルーチンからメインルーチンへ戻ることを示します．メインルーチンからサブルーチンを呼び出すには，**GOSUB文**を使います．

GOSUB 行番号

という文は，プログラムの中からその行番号で始まるサブルーチンを実行することを示します．そして，**RETURN文**を実行すると，GOSUB文の次の文を実行します．

のようになり，かえって煩雑となります．

つまり，１行に書いた方がわかりやすい場合には１行にした方がよいということです．

次のプログラムは，１～Ｎまでの和と積を一度に求めるものですが，20行の初期化と40行の和と積を求める部分をマルチステートメントにしたものです．処理の内容が同一，またはそれに近いときは１行にまとめた方が見やすい場合もあります．

```
10 INPUT "N=";N
20 WA=0:SK=1
30 FOR I=1 TO N
40 WA=WA+I:SK=SK*I
50 NEXT I
60 PRINT "WA=";WA;"  SEKI=";SK
70 END
```

サブルーチンは，このように他のルーチンのどこからでも呼ぶことができますから，いろいろなサブルーチンを用意しておいて，必要に応じて使うこともできます．これが，**サブルーチン・ライブラリ**です．

6.3 サブルーチンの利点

サブルーチンは，共通部分を処理させるためだけのものではありません．第一に，プログラムを短かくすることもできます．共通部分が多い場合など，それらを一つのサブルーチンにすることより，はるかに短かいプログラムとすることができます．第二に，サブルーチンは，それぞれ独立した処理を行ないます．このようなサブルーチンは，短かいプログラムですむので，もしプログラムにミスがあったとしても見つけやすくなります．

サブルーチンには，できるだけ１個の機能を持たせるようにすれば，サブルーチン自体も短かくすむし，ミスの入る度合も減ります．１個の機能とは，**半径を与えると面積を返してくる**とか，**縦，横，高さを与えると体積と表面積を返してくる**などという与えるデータと返してくるデータがはっきりしているものをいいます．このように，１個のサブルーチンの処理が明確になっていればその部分だけのテストをするにしても，与えるデータと返してくるデータの関係から，正しいのか，正しくないのかがすぐわかります．正しくない場合には，そのサブルーチンだけの修正を行なえばよいのです．

あるサブルーチンを呼び出す側にとって，そのサブルーチンの中味は問題ではありません．ただ，あるデータを与えたときに，そのサブルーチンが目的とする機能を果たしてさえくれればよいわけですから，１個のブラック・ボックス的な考えをすることができます．

6.4 関数サブプログラム

PC-2001にはsin，cosなどの初等関数を始め，いろいろな関数が内蔵されています．これらを**組み込み関数**といいます．使い方は次のようにします．

A＝SIN(X)…… Xの値のsin値をAに代入します．
PRINT LOG(3)…… $\log_e 3$の値を表示します．

また，PC-2001では，これらの内蔵関数の他にユーザー(使用者)が関数を定

この例のメインルーチンは次のようになります.

```
10 INPUT "N=";N
20 FOR X=1 TO N
30 GOSUB 100
40 PRINT "X=";X;" XR=";XR;" X2=";X2;
   " X3=";X3
50 NEXT X
60 END
```

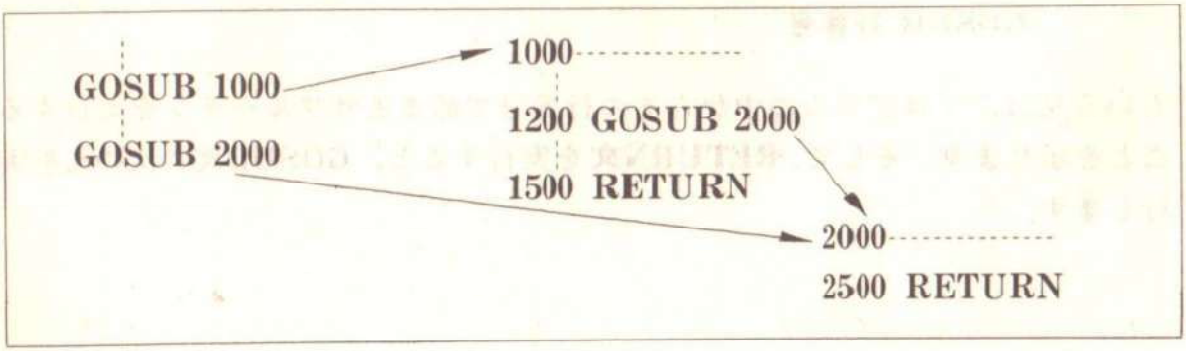
これら2つをまとめたものが次のプログラムです.

```
10 INPUT "N=";N
20 FOR X=1 TO N
30 GOSUB 100
40 PRINT "X=";X;" XR=";XR;" X2=";X2;
   " X3=";X3
50 NEXT X
60 END
100 XR=SQR(X)
110 X2=X*X
120 X3=X^3
130 RETURN
```

6.2 下請け、孫請け、エトセトラ

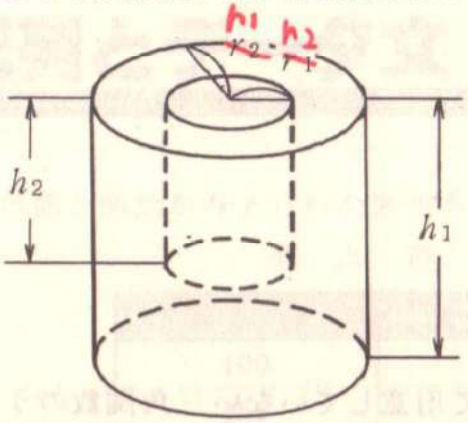
メインルーチンは大会社ですから，下請け会社をいくつ持ってもかまいません．また，**孫請け**という言葉もあるように，下請けの下請けつまり**サブルーチンが，サブルーチンを呼ぶ**ことができます．図に表わすと下のようになります．

サブルーチンと制御の流れ



関数サブプログラムを使ったプログラムを作ってみましょう．

例　**次のような円筒形の体積を求めるプログラムを作る．**



考え方としては，まず全体がつまっているものとして，体積を求めます．次に，穴の空いた部分の体積を求めて最後に引き算をするのが簡単です．

つまり，

❶半径r_1，r_2，高さh_1，h_2を入力する．
❷全体の体積v_1を求める．
❸穴の部分の体積v_2を求める．
❹体積$v_1 - v_2$を表示する．

となります．円柱の体積は，

v＝円周率×(半径)^2×高さ

ですから，これを関数Vとして定義すると，

```
DEF FNV(R,H)=3.14159*R^2*H
```

となります．

これを使ってプログラムを組むと次のようになります．

義することができます．それには，**DEFFN文**を使います．

DEFFN文は，一般的に次のように定義します．

DEF FN 関数名(引数リスト)＝定義式

関数名は，その関数の名前であり変数名と同じようにつけます．引数リストはこの関数を他の文で使うときに実際の変数や定数，式と置き換えられるものです．定義式はこの関数を表わす式です．

たとえば，関数$y=x^2+4x-3$は次のように定義します．

DEF FNY(X)＝X↑2＋4＊X－3

ここでいう，Xはこの関数定義式の中だけで使われますから，他にXという名前の変数が使われていてもかまいません．また，これら引数リストの中にある変数は仮のものですから，名前も任意でかまいません．

さて，この関数を使うためには，

❶A＝FNY(3)

❷PRINT 5＊FNY(2＊3)

のように使います．( )の中の変数や定数または式の結果が関数にXとして渡され，関数計算が行なわれます．

つまり，❶ではAに18が代入されます．また，❷は関数の結果は57ですが，最終的には5倍されて，285が表示されます．このように，関数が演算式中で使われた場合は他の演算よりも先に実行されます．

また，引数が2つ以上ある場合はそれぞれの引数をカンマで区切ります．

たとえば，関数$z=3x+2y$は，

DEF FNZ(X , Y)＝3＊X＋2＊Y

となります．この場合，この関数を呼び出すには，

FNZ(3 , 5＊A)

などのように，変数や定数，式をカンマで区切って並べます．また，関数の宣言はその関数を使う前に行なわなければなりません．

7. 配列とデータ文

次のように，単価と個数が与えられた表があるとしましょう．

売 上 表

単　価	個　数
100	4
150	7
200	10
250	2
300	5
350	3
400	6

この表から金額と合計を計算するプログラムを作ってみます．金額の計算は，

単価×個数＝金額

ですから，単価をTA，個数をKO，金額をKI，合計をGKとすると次のようなプログラムを考えることができます．

```
10 GK=0
20 FOR I=1 TO 7
30 INPUT "ﾀﾝｶ=";TA
40 INPUT "ｺｽｳ=";KO
50 KI=TA*KO
60 GK=GK+KI
70 PRINT "ｷﾝｶﾞｸ=";I;"= ¥";KI
80 NEXT I
90 PRINT "ｺﾞｳｹｲ=";GK
100 END
```

このプログラムは，入力，計算，出力をFOR文の中で行ない，繰り返し処理を行なっています．しかし，この方法には欠点があります．それは，ループ内で，入力，計算，出力を行なっているために，途中で入力ミスがあった場合で

```
10 DEF FNV(R,H)=3.14159*R^2*H
20 INPUT "R1=";R1
30 INPUT "R2=";R2
40 INPUT "H1=";H1
50 INPUT "H2=";H2
60 V1=FNV(R1,H1)
70 V2=FNV(R2,H2)
80 PRINT V1-V2
90 END
```

BASIC が組み込み関数として用意していない三角関数のうち，いくつかは，用意された関数を使って作り出すことができます．以下の数式を参考にしてDEFFN文で定義してください(誤差の範囲に注意が必要です)．

目的とする関数	組み込み関数からの誘導式
セカント	SEC(X)=1/COS(X)
コセカント	CSC(X)=1/SIN(X)
コタンジェント	COT(X)=1/TAN(X)
アークサイン	ARCSIN(X)=ATN(X/SQR(−X*X+1))
アークコサイン	ARCCOS(X)=−ATN(X/SQR(−X*X+1))+1.5708
アークセカント	ARCSEC(X)=ATN(SQR(X*X−1))+(SGN(X)−1)*1.5708
アークコセカント	ARCCSC(X)=ATN(1/SQR(X*X−1))+(SGN(X)−1)*1.5708
アークコタンジェント	ARCCOT(X)=−ATN(X)+1.5708
ハイパーボリック・サイン	SINH(X)=(EXP(X)−EXP(−X))/2
ハイパーボリック・コサイン	COSH(H)=(EXP(X)+EXP(−X))/2
ハイパーボリック・タンジェント	TANH(X)=−EXP(−X)/(EXP(X)+EXP(−X))*2+1
ハイパーボリック・セカント	SECH(X)=2/(EXP(X)+EXP(−X))
ハイパーボリック・コセカント	CSCH(X)=2/(EXP(X)−EXP(−X))
ハイパーボリック・コタンジェント	COTH(X)=EXP(−X)/(EXP(X)−EXP(−X))*2+1
ハイパーボリック・アークサイン	ARCSINH(H)=LOG(X+SQR(X*X+1))
ハイパーボリック・アークコサイン	ARCCOSH(X)=LOG(X+SQR(X*X−1))
ハイパーボリック・アークタンジェント	ARCTANH(X)=LOG((1+X)/(1−X))/2
ハイパーボリック・アークセカント	ARCSECH(X)=LOG((SQR(−X*X+1)+1/X)
ハイパーボリック・アークコセカント	ARCCSCH(X)=LOG((SGN(X)*SQR(X*X+1)+1)/X)
ハイパーボリック・アークコタンジェント	ARCCOTH(X)=LOG((X+1)/(X−1))/2

たとえば，sec (x) の定義は，

```
DEF FNSEC (X)=1/COS (X)
```

となります．

7.1 データの予約－配列－

配列は変数と同じく**配列名**を持ち，各データ(**配列の要素**といいます)は，かっこに入れた**添字**で区別します．配列の指定は，

DIM　配列名(添字の最大数)

とします．これを，**配列の宣言**といいます．たとえば，

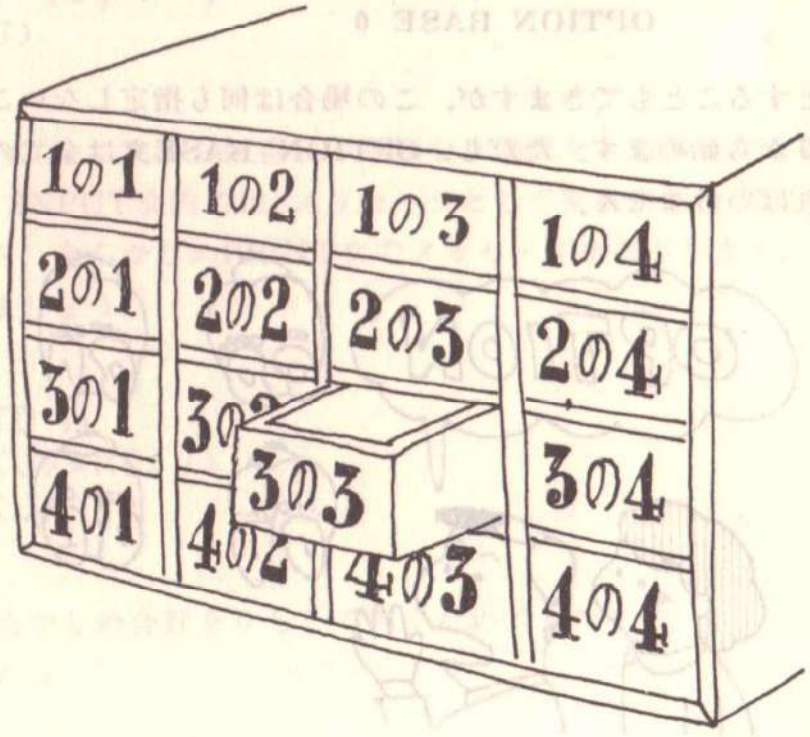
DIM　A(5)

とすると，

A(0)，A(1)，A(2)，A(3)，A(4)，A(5)

の6つの要素を持つ配列Aが宣言されます．なお，これらの配列Aは，単なる変数Aとはまったく違うものです．

配列はちょうど引き出しのたくさんついた箱と考えることができます．0～5までの番号のふられた引き出しにデータを出し入れすることと同じなのです．



も，それを修正するのが面倒であるという点です．

できるなら，このように入力データが比較的多いプログラムでは，入力後そのデータの確認をするステップを入れておきたいものです．

処理の流れでいうと，図のようなプログラムの形が理想的です．そのためには，入力したデータをどこかに保存しなければなりません．そのような場合に**配列**を使います．

理想的なプログラムの流れ

また，配列の添字の最大数を変数で指定することもできます．ただし，配列を宣言する前に，その変数の値が確定していなければなりません．

```
10 INPUT N,M
20 DIM A(N),B(N,M)
30 ..............
```

とすることにより，データ量に応じてNとMの値を入力すれば，適当な大きさの配列をとることができます．

さて，先ほどのプログラムを配列を使って作りましょう．ここでは，データの確認，修正に関しては省略することにします．まず，単価を示す配列をTA，個数を示す配列をKO，金額を示す配列をKI，合計を示す変数をGKとしましょう．入力部は，次のようになります．

```
10 INPUT "N=";N
20 OPTION BASE 1
30 DIM TA(N),KO(N),KI(N)
40 FOR J=1 TO N
50 PRINT "タンカ";J;"=";
60 INPUT TA(J)
70 PRINT "コスウ";J;"=";
80 INPUT KO(J)
90 NEXT J
```

50行と70行のPRINT文は，これから何番目の単価なり個数を入力するかを表示するメッセージです．INPUT文内では，メッセージとして変数の値を表示することはできませんから，あらかじめPRINT文でメッセージを表示します．

次に計算部分を作りましょう．

```
100 GK=0
110 FOR J=1 TO N
120 KI(J)=TA(J)*KO(J)
130 GK=GK+KI(J)
140 NEXT J
```

となります．100行はあらかじめ合計を0としておくためのものです．

最後に結果の出力ですが，

また，

```
DIM  A(2, 3)
```

のように配列を宣言することもできます．この場合には，

```
A(0, 0), A(0, 1), A(0, 2), A(0, 3)
A(1, 0), A(1, 1), A(1, 2), A(1, 3)
A(2, 0), A(2, 1), A(2, 2), A(2, 3)
```

のように，12個の要素を持つ配列Aが宣言されます．添字の数が1個の配列を1次元配列，2つある配列を2次元配列といいます．

BASICの配列は不思議なことに，A(0)のように，添字が0から始まります．0行とか0列というのは数え間違いやすいので，よほどのことがない限り0番目の要素は使わない方がよいでしょう．

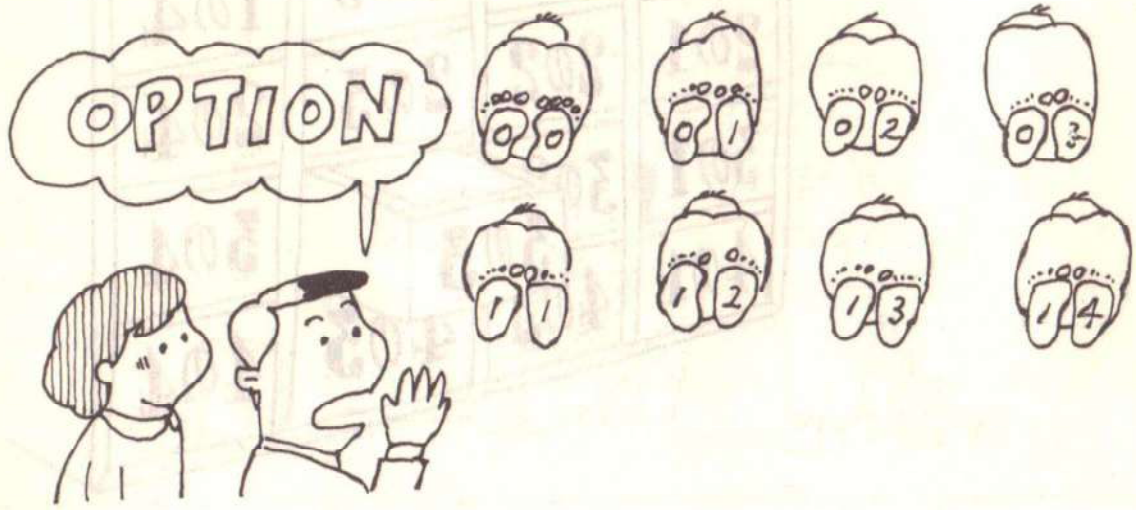
また，PC-2001では配列の添字を1から始めることもできます．それには，配列を宣言する前に，

```
OPTION BASE 1
```

としてやります．また，

```
OPTION BASE 0
```

とすることもできますが，この場合は何も指定しないことと同じになり添字を0から始めます．ただし，OPTION BASE文は全てのDIM文の前に使わなければいけません．



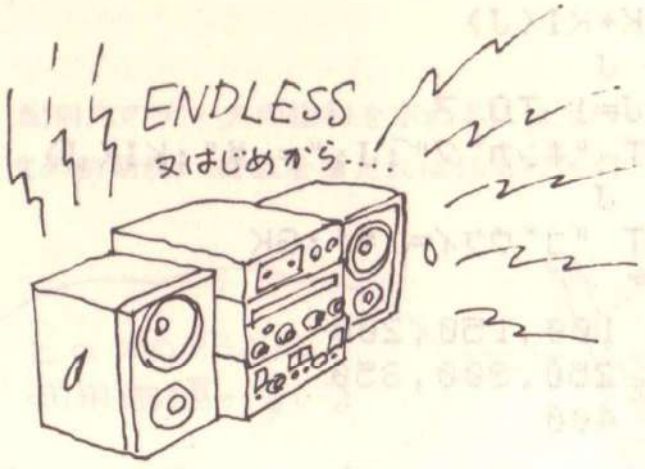
とすると，その行番号のDATA文のデータから再び読み込まれます．例を示しましょう．

```
10 DIM AA(6)
20 READ A,B,C
30 RESTORE 110
40 READ D,E
50 RESTORE
60 FOR I=1 TO 6
70 READ AA(I)
80 NEXT I
90 END
100 DATA 1,2
110 DATA 3,4
120 DATA 5,6
```

とすると，READ文実行後には，

A＝1，B＝2，C＝3
D＝3，E＝4
AA(1)＝1，AA(2)＝2，AA(3)＝3
AA(4)＝4，AA(5)＝5，AA(6)＝6

となります．配列とRESTORE文を組み合わせて，データの配置をよく考えると非常に強力な機能を持ちます．



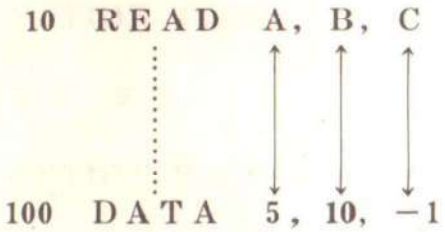
```
150 FOR J=1 TO N
160 PRINT "ｷﾝｶﾞｸ";J;"= ¥";KI(J)
170 NEXT J
180 PRINT "ｺﾞｳｹｲ= ¥";GK
190 END
```

となります.

7.2 データの束を読む −READ文とDATA文−

今の場合は，単価を入力しましたが，もし単価があらかじめ決まっていたとしたら，単価を入力するのはムダなことです．そのような場合に，あらかじめ単価を定義しておくことができます．それには，**READ文**と**DATA文**を使います.

READ文とDATA文はペアにして使いますが，READ文は，あらかじめプログラム内でDATA文により作っておいたデータを読み込むためのものです.



上のように使うと，Aに5，Bに10，Cに−1が読み込まれます．READ文で指定される変数の数，または配列の要素の数とDATA文中のデータとは1対1に対応しますから，データの数が少ないとエラーになります.

また，データはいくつかのDATA文に分けてもかまいません．この場合，データを読む順序は文番号の小さい方からの順となります.

プログラム内で読み込むデータの順序を**RESTORE文**によって変えることができます．RESTORE文には，2種類あります．まず，

RESTORE

として単独に使う場合です．このときは，プログラムの一番初めにあるDATA文のデータから再び読み込まれます．また，

RESTORE　行番号

7.3 配列の応用

配列を使うといろいろな処理を行うことができます．ここでは，配列の応用としてデータの総和と平均を求めるプログラムと並べ換えのプログラムを作ってみましょう．

7.3−1 総和と平均

次のように与えられた成績表から，総合点と平均点を求めるプログラムを作ります．

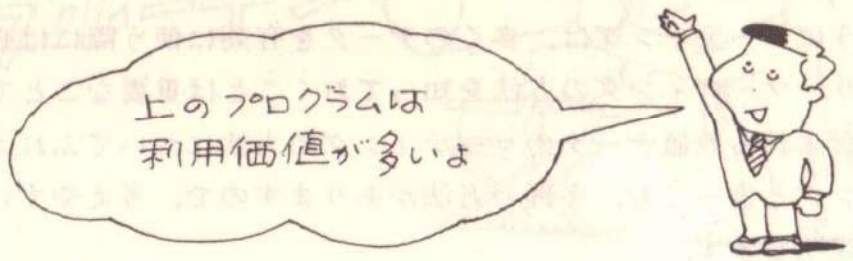
成　績　表

教　科　1	点　数　1
教　科　2	点　数　2
⋮	⋮
教　科　N	点　数　N

各教科の点数が配列ＫＹに入っているものとすると，合計点SOを求めるには，教科数をＮとすると，

```
SO=0
FOR I=1 TO N
SO=SO+KY(I)
NEXT I
```

のようになります．配列内のデータの総和を求めるには上の方法を使えば，全てできます．FOR文の初期値，終値を変えれば任意の区間での総和を求めることもできます．



さて，先ほどのプログラムに使ってみましょう．データは，**表**(53ページ)のみの場合としてあります．

```
10 OPTION BASE 1
20 DIM TA(7),KO(7),KI(7)
30 FOR J=1 TO 7
40 READ TA(J)
50 PRINT "ｺｽｳ";J"=";
60 INPUT KO(J)
70 NEXT J
80 GK=0
90 FOR J=1 TO 7
100 KI(J)=TA(J)*KO(J)
110 GK=GK+KI(J)
120 NEXT J
130 FOR J=1 TO 7
140 PRINT "ｷﾝｶﾞｸ";J;"= ¥";KI(J)
150 NEXT J
160 PRINT "ｺﾞｳｹｲ= ¥";GK
170 END
180 DATA 100,150,200
190 DATA 250,300,350
200 DATA 400
```

単純選択法の考え方は，配列の要素の中から最小値(または最大値)を見つけ出しながら順に並べ換えていくというものです．

この方法を表に表わすと次のようになります．

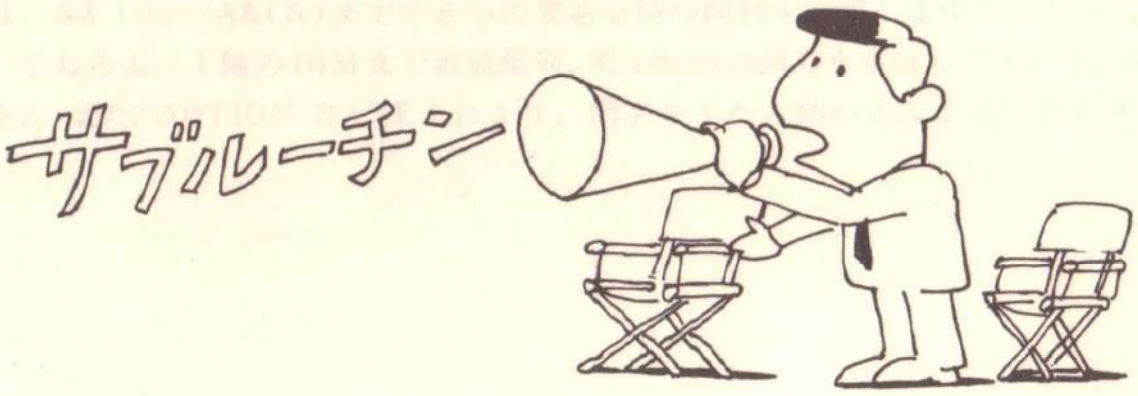
ソーティングの方法

初期状態	25	17	80	12	6	60
1回め	6	17	80	12	25	60
2回め	6	12	80	17	25	60
3回め	6	12	17	80	25	60
4回め	6	12	17	25	80	60
5回め	6	12	17	25	60	80

ソーティングする配列をAとすると，プログラムは次のようになります．

```
200 FOR I=1 TO N-1
210 M=I:AA=A(I)
220 FOR J=I+1 TO N
230 IF A(J)<AA THEN M=J:AA=A(J)
240 NEXT J
250 A(M)=A(I):A(I)=AA
260 NEXT I
270 RETURN
```

このプログラムはサブルーチンとなっていますから，メインルーチンでデータを定義してから呼び出します．



次に，平均点は合計点を教科数で割ればよいわけですから，

```
HE=SO/N
```

となります．

以上の計算部分に入力部，出力部をつけたプログラムは次のようになります．

```
10 INPUT "ｷｮｳｶｽｳ=";N
20 OPTION BASE 1
30 DIM KY(N)
40 FOR I=1 TO N
50 PRINT "ｷｮｳｶ";I;"=";
60 INPUT KY(I)
70 NEXT I
80 SO=0
90 FOR I=1 TO N
100 SO=SO+KY(I)
110 NEXT I
120 HE=SO/N
130 PRINT "ｺﾞｳｹｲ ﾊ";SO;"ﾃﾝ";
140 PRINT "  ﾍｲｷﾝ ﾊ";HE;"ﾃﾝ"
150 END
```

7.3−2　並べ換え

データをアイウエオ順に並べ換えたり，大小順に並べ換えたりすることを**ソーティング**といいます．ソーティングの目的は，データを順序正しく並べ換えて，後の検索を容易にすることにあります．したがって，それはほとんど日常的に行なわれることであり，私達のまわりを見ると，電話帳でも，図書館のカードでも，辞書でも，目次でも全て特定のルールにしたがった順序にデータが並べられています．

このようにソーティングは，多くのデータを有効に使う際には必ず必要な手続きであり，ソーティングの方法を知っておくことは重要なことです．

配列に含まれる数値データのソーティングの方法についてふれます．一口にソーティングと言っても，多種の方法がありますので，考えやすい**単純選択法**について説明します．

8. 文字列処理

前節までは，データとして数値だけを扱ってきましたが，PC-2001では**文字データ**も扱うことができます．

いくつかの文字からなるデータを**文字列**といい，PC-2001では文字列を変数に代入したり，つないだり，分解したりすることができます．文字列を扱う変数を**文字変数**と呼びます．文字変数を表すには，変数名の後に＄をつけ，

```
A$, AA$, XI$
```

のように表わします．これら文字変数には255文字までの文字列を代入することができます．代入するには，文字列を" "でくくり，

```
A$="ABCDE"
B$="1 2 3 4 5 "
```

のようにします．

また，文字変数を配列とすることもできます．宣言は数値配列と同じように，

```
DIM　文字配列名(添字の最大数)
```

とします．たとえば，

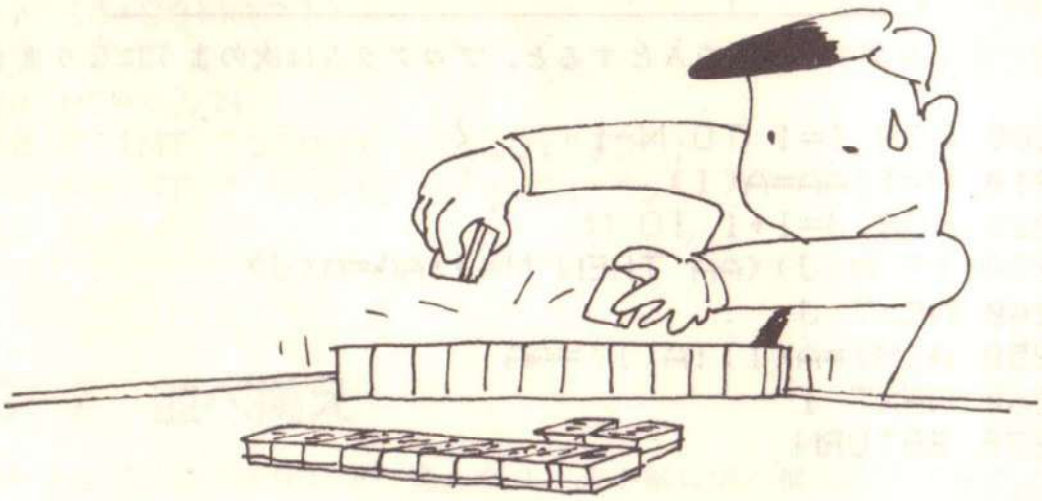
```
DIM　A$(5)
```

は，A$（0）～A$(5)までの６つの要素を持つ配列を宣言します．

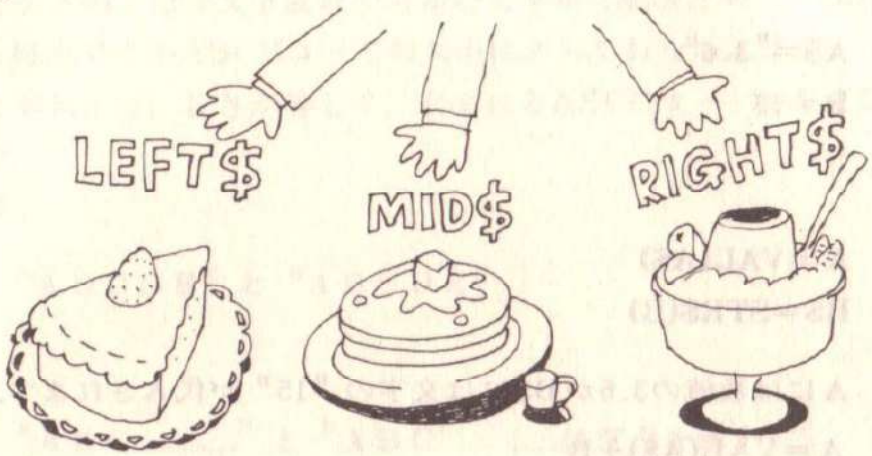
もちろん，１個のDIM文で数値配列，文字配列の両方を宣言してもかまいません．また，OPTION BASE文により，添字を１から始めることもできます．

メインルーチンは，たとえば次のように作ります．

```
10 INPUT "N=";N
20 OPTION BASE 1
30 DIM A(N)
40 FOR I=1 TO N
50 PRINT "DATA NO.";I;"=";
60 INPUT A(I)
70 NEXT I
80 GOSUB 200
90 FOR I=1 TO N
100 PRINT "DATA NO.";I;"=";A(I)
110 NEXT I
120 END
```



```
10 A$="ABCDEFGHIJ"
20 PRINT "LEFT=";LEFT$(A$,5)
30 PRINT "RIGHT=";RIGHT$(A$,5)
40 FOR I=1 TO 6
50 PRINT "MID=";I;"=";MID$(A$,I,5)
60 NEXT I
70 END
```



LEFT$, RIGHT$, MID$は文字数として，文字列の長さ以上を指定したりするなど範囲外の値を指定するとエラーになります．そこで，文字列自身の長さを求める関数も用意されています．

LEN（文字列または文字変数）

とすることにより，文字列や文字変数に含まれる文字数を求めることができます．また，文字列，

"1 2 3"

と，数値の，

1 2 3

は，同じものとして扱うことができません．つまり，

"1"+"2"

の結果は，"3" ではなく "1 2" になります．

そこで，文字データで表わされる数値を計算に使うために数値データに直し

8.1 文字列の編集

数値データが足したり，引いたりなどの演算ができたのと同じように，文字列や文字変数も演算することができます．ただし，**文字列の演算**とは，文字列同士をつないだり，分解したりして編集することをいいます．

PC-2001には，編集するための命令がいくつか用意されています．文字列をつなぐためには＋を使います．

```
A$=B$+C$
```

とすると，B$とC$をつないだ文字列をA$に代入します．たとえば，

```
B$="PC "
C$="2001"
```

の場合には，

```
A$="PC  2001"
```

となります．文字列内では，空白(スペース)は意味を持つことに注意してください．文字列をつないだ結果が255文字より大きくなるとエラーになります．

文字列を分解したり，途中から取り出すためには次の3個の関数を使います．

LEFT$（文字列または文字変数，文字数）
RIGHT$（文字列または文字変数，文字数）
MID$（文字列または文字変数，開始位置，文字数）

LEFT$は，指定した文字列や文字変数の左から文字数分だけを取り出します．RIGHT$は，LEFT$の逆で右から文字数分を取り出してきます．

それに対して，MID$は，取り出し開始位置より文字数分だけを取り出してきます．次のプログラムを実行してみてください．三者の違いがはっきりします．

PRINT CHR$(90) RETURN

とすると，文字Zが表示されます．ASCIIコードと文字との対応は巻末のキャラクタコード表およびキーコード表を参照してください．

PC-2001の内部では，文字はASCIIコードで表わされますから，文字列同士の大小関係や一致しているかの比較をすることもできます．文字列の大小比較は，二つの文字列または文字変数の最初の文字から順次比較し，異なった文字を見つけた時点でそのASCIIコードの大小によって行なわれます．

等しい文字列とは，長さが等しく，含まれるASCIIコードが全て等しい場合を言います．

たとえば，

"ABCDE" と "ABCDE"

は等しく，

"ABC␣␣␣" と "ABC"　（␣は空白を示す）

は等しくありません．文字列の中では空白も意味を持つからです．また，大小

"ABC" と "ABD"

は，2番目の文字までは等しいですが3番目の文字C(ASCIIコード67)とD(ASCIIコード68)ではDの方が大きいため，"ABD"の方が大きくなります．また，

"AB" と "ABC"

では，"ABC" の方が大きくなります．これは，3番目の文字を比較するとき，長さをそろえるため，"AB" のうしろに**ヌルストリング**(何も含まれないという文字でASCIIコード0)をつないでいるからです．

ASCIIコードで比較するため，長さが短かくても，大きいと判断されることもあります．

"X" と "AAA"

では，最初の文字XとAの大小関係から "X" の方が大きくなります．

たり，逆に数値を文字データに変換したりする関数もあります．これが，

VAL（文字列または文字変数）
STR$（数値または数値変数）

です．VALは，文字列を数値に変換し，STR$は逆に数値を文字列に変換します．たとえば，

```
A$="3.6"
B=15
```

のとき，

```
A=VAL(A$)
B$=STR$(B)
```

とすると，Aには数値の3.6がB$には文字の "15" が代入されます．また，

```
A=VAL(A$)+B
```

とすると，Aには3.6と15をたした結果の18.6が代入されます．

8.2 文字列の比較

数値データが比較できたのと同じように，文字列も比較することができます．BASICの内部では，文字はASCII コードと呼ばれるコードで表されています．

たとえば，

```
PRINT  ASC("A")  RETURN
```

としてみてください．65と表示されます．これは，文字AのASCIIコードが65であることを示しています．同様にB，C，……ZのASCIIコードは66，67，……，90となります．

つまり，ASC関数はある文字のASCIIコードを求めるものです．逆に，あるASCIIコードに相当する文字を求める関数もあります．それは，

CHR$(ASCIIコード)

で，たとえば，

このIF文の中のANDは2つの条件の両方を結合したもので，A$(I)が"A"より大きいか等しく，かつA$(I)が"B"より小さいときに真となります．

条件1　かつ　条件2

の真偽を判断することになります．また，ANDの代わりにORを使うと，

条件1　または　条件2

の真偽を判断することになります．たとえば，条件，

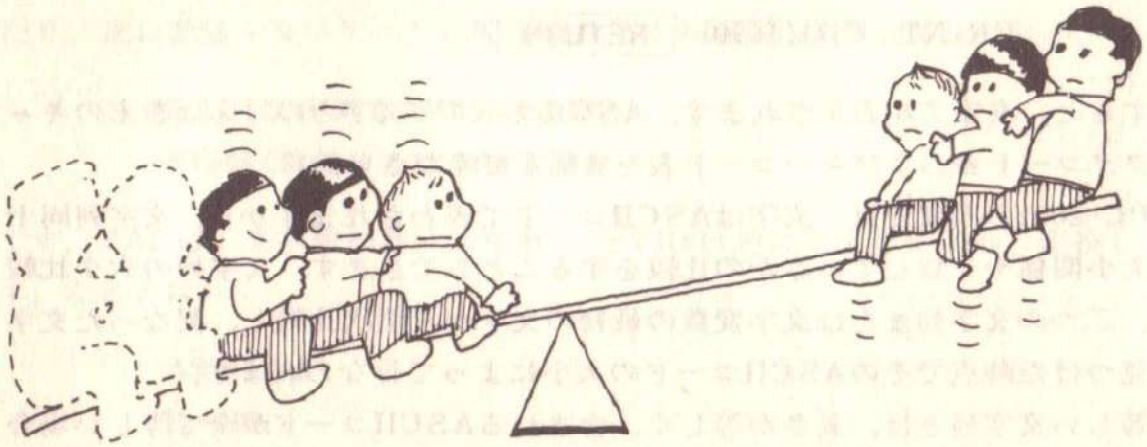
```
A$<"C"  OR  A$>="D"
```

は，A$が"C"より小さいか，A$が"D"より大きいか等しいときに真となります．言い換えれば，A$の最初の文字が"C"でなければ真となります．

もちろん，ANDとORは数値の比較にも使えます．変数Aが1以上9未満であるかを調べるための条件は，

```
A>=1  AND  A<9
```

となります．



文字配列内のデータを比較することにより，ソーティングをすることも可能です．前節のソーティング・プログラムの必要とする変数を文字変数に変更するだけで，次のようになります．

```
200 FOR I=1 TO N-1
210 M=I:AA$=A$(I)
220 FOR J=I+1 TO N
230 IF A$(J)<AA$ THEN M=J:AA$=A$(J)
240 NEXT J
250 A$(M)=A$(I):A$(I)=AA$
260 NEXT I
270 RETURN
```

また，文字配列内から条件を満たすようなデータを見つけ出すこともできます．

```
IF A$(I)>="D" THEN PRINT A$(I)
```

は，最初の文字がD 以降の文字データを見つけます．逆に，

```
IF A$(I)<"F" THEN PRINT A$(I)
```

は，最初の文字がFより小さい文字データを見つけます．さらに，最初の文字がAで始まる文字データを見つけるときは，

```
IF A$(I)>="A" AND A$(I)<"B" THEN PRINT A$(I)
```

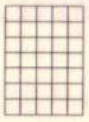
とします．

9. 動きと音を楽しむ

PC－2001では，表示部に漢字や記号を描くことが可能です．表示されている英字や数字をよく見ると，いくつかの点で表わされています．このように，いくつかの点で表わされたパターンを**ドットパターン**といいます．PC-2001に漢字や記号を描くには，ドットパターンを作る必要があります．

9.1 ユーザー定義キャラクタ

一つの文字は，下図の左のような横5ドット縦7ドットのちょうど碁盤の目のような上に書かれています．たとえば，英字のAは右のように表わされます．



このドットパターンで表わすことにより，任意の文字や記号を画面に表示することができます．これらの文字を**ユーザー定義キャラクタ**といい，10種類まで定義することができます．その方法は，縦1列ずつを1組のドットパターンとして，2桁の**16進数**で指定します．

16進数とは，10進数の0～15を1桁として表わす数で，10～15を英字のA～Fで表わします(**下表参照**)．この16進数での指定により，縦1列の各ドットをつけたり消したりします．下表の横に，1，2，4，8と記してありますが，つけたい部分を加え合わせて，16進数に変換します．たとえば，全部つけるために上位桁は，

1＋2＋4＝7 ……16進数でも7

下位桁は，

10進数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16進数	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

これらを使って文字配列内から指定した文字で始まるデータを見つけ出すプログラムは，次のようになります．

サブルーチンになっていますから，文字配列A$とデータの個数Nは呼び出す側で定義します．

```
200 PRINT "サイショ ノ モジ ハ";AA$
210 FOR I=1 TO N
220 IF LEFT$(A$(I),1)=AA$ THEN PRINT A$(I)
230 NEXT
240 RETURN
```

任意の文字，記号を作るためには5バイトのデータをメモリに書き込みますが，たとえば，

を定義するには，

```
POKE &HFCAE, &H7F
POKE &HFCAF, &H49
POKE &HFCB0, &H7F
POKE &HFCB1, &H49
POKE &HFCB2, &H7F
```

とした上で，

```
PRINT CHR$(&HE0)
```

とすると，

を表示します．

ユーザーが定義できるキャラクタコードは，&HE0～&HE9までの10種でドットパターンを書き込むメモリの番地とは次の表のように対応しています．

キャラクタコード	メモリの番地
&HE0	&HFCAE ～ &HFCB2
&HE1	&HFCB3 ～ &HFCB7
&HE2	&HFCB8 ～ &HFCBC
&HE3	&HFCBD ～ &HFCC1
&HE4	&HFCC2 ～ &HFCC6
&HE5	&HFCC7 ～ &HFCCB
&HE6	&HFCCC ～ &HFCD0
&HE7	&HFCD1 ～ &HFCD5
&HE8	&HFCD6 ～ &HFCDA
&HE9	&HFCDB ～ &HFCDF

1+2+4+8=15 ……16進数ではF

として，2桁の16進数

&H7F

で指定します．

このパターンをメモリの&HFCAE番地から5列分書き込みます．そして，

PRINT CHR$(&HE0)

とすることにより，ディスプレイに表示することができます．メモリへの書き込みはPOKE文で行ないます．

POKE &HFCAE, &H7F

とすると&HFCAE番地に&H7Fを書き込みます．16進数2桁で表わされる数値は&H00～&HFF(10進数で0～255)まであり，これらを1バイトのデータといいます．つまり，メモリのある番地へデータを書き込むには，番地を2バイトで指定し，書き込むデータを1バイトで指定します．

逆に，メモリのある番地のデータを読み出すのが，**PEEK関数**です．

PEEK(メモリ番地)

とすると，その番地のデータを与えます．画面に&HFCAE番地の内容を表示するには，

PRINT PEEK(&HFCAE) RETURN

とすると10進数で表示し，

PRINT HEX$(PEEK(&HFCAE)) RETURN

とすると16進数の文字列として表示します．**HEX$**は数値を16進数の文字列として与える関数です．

```
10 FOR I=0 TO 31
20 BEEP I,1
30 NEXT I
40 END
```

とすると全ての音階を0.1秒ずつ鳴らします.

また, 次のプログラムは「ドナウ河のさざなみ」を演奏するものです.

```
10 OPTION BASE 1
20 DIM T(22),L(22)
30 FOR I=1 TO 22
40 READ T(I),L(I)
50 NEXT I
60 FOR J=1 TO 22
70 BEEP T(J),L(J)
80 NEXT J
90 GOTO 60
100 END
110 DATA 12,2,12,12,16,2,17,2,19,12,16,2
120 DATA 12,2,20,12,19,2,17,2,24,15
130 DATA 24,2,25,12,24,2,22,2,24,12,22,2
140 DATA 20,2,19,12,20,2,19,2,17,15
```

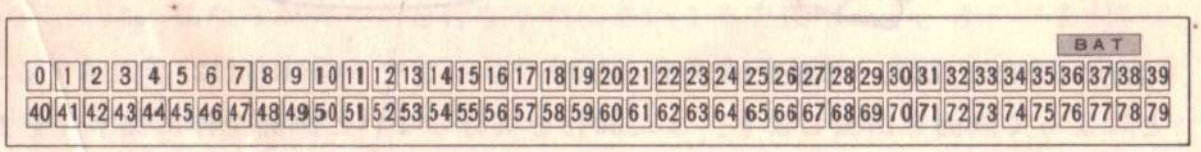
9.3 文字を動かす

ある文字列を左から順に動かしていくには, **LOCATE文**と**WAIT文**, PAUSE**文**またはPRINT文を使います.

LOCATE文は文字列や式の値を表示する位置を定めるものです. LOCATE文は,

LOCATE　0～79の整数

として使いますが, 0～79の整数はディスプレイに対して次の図のように対応します.



次のプログラムはドットパターンデータをメモリに書き込むためのものです．文字数をNで指定するとDATA文からドットパターンデータを取り出しながら&HFCAE番地以降に書き込みます．

```
10 INPUT "N=";N
20 IF N<=0 OR N>10 THEN 10
30 AD=&HFCAE
40 FOR I=0 TO N*5-1
50 READ DT$
60 POKE AD+I,VAL("&H"+DT$)
70 NEXT I
80 END
90 DATA 7F,49,49,49,7F
100 DATA ・・・・・・・・・・・・・・・
110 DATA ・・・・・・・・・・・・・・・
```

9.2 音を出す

PC-2001では，内蔵ブザーにより音を出すことができます．

BEEP 式1，式2

としますが，式1は音階を表わし，式2は音長を表わします．音階は0～31の数値で与えられ，次の表のとおりに決められます．

式1	音階	周波数(Hz)	式1	音階	波数(Hz)	式1	音階	周波数(Hz)	式1	音階	周波数(Hz)
0	無音	—	8	ド	525	16	ソ＃	833	24	ミ	1330
1	ファ	349	9	ド＃	553	17	ラ	880	25	ファ	1389
2	ファ＃	370	10	レ	584	18	ラ＃	933	26	ファ＃	1488
3	ソ	393	11	レ＃	625	19	シ	992	27	ソ	1563
4	ソ＃	414	12	ミ	658	20	ド	1042	28	ソ＃	1645
5	ラ	440	13	ファ	702	21	ド＃	1116	29	ラ	1736
6	ラ＃	466	14	ファ＃	735	22	レ	1179	30	ラ＃	1838
7	シ	492	15	ソ	781	23	レ＃	1250	31	シ	1953

音長は1～255までの整数で指定しますが，音の出ている時間は式2の値÷10秒となります．式2は省略してもかまいませんが，その場合は4が指定されますから音の出ている時間は0.4秒となります．

示します．

PRINT文の最後にセミコロン(;)をつけると実行を停止せずに次の文を実行します．それに対して，PAUSE文はWAIT文で指定しただけ表示し，それを消去してから実行を続けます．WAIT文は，

WAIT 式

のように使い，式÷10秒間の表示を指定できます．

次のプログラムは，文字Aが移動しますが，表示時間は約0.1秒となります．

```
10 CONSOLE 1,0,0
20 WAIT 1
30 FOR I=0 TO 79
40 LOCATE I
50 PAUSE "A"
60 NEXT I
70 END
```

9.4 乱数—RNDとRANDOMIZE—

乱数とは，その名の通り乱れた数，デタラメの数のことです．RNDは関数で，

RND(X)

と使います．

ここで，Xの値として正の値を使うと，同じ系列の次の乱数を与えます．Xの値が負の場合には新しい乱数の系列を与え，Xが0ならば一つ前に発生した乱数を与えます．普通はXの値として正の値を用います．Xを省略して，

RND

としてもかまいませんが，この場合X＝1と同様になります．

乱数はいろいろのゲームに利用できます．たとえば，実際に乱数をとるということは，私たちがゲームを行なうのにさいころを振ったり，カードを引いたりするときです．そのような，1から任意の数までの間のデタラメの数の出るさいころを作ることができます．

たとえば，

LOCATE　5：PRINT　"A"　RETURN

とすると，1行めの6文字めに文字Aを表示します．

LOCATE文で指定する値の最大値は79ですがCONSOLEコマンドで指定された表示方法により，最大値が小さくなります．

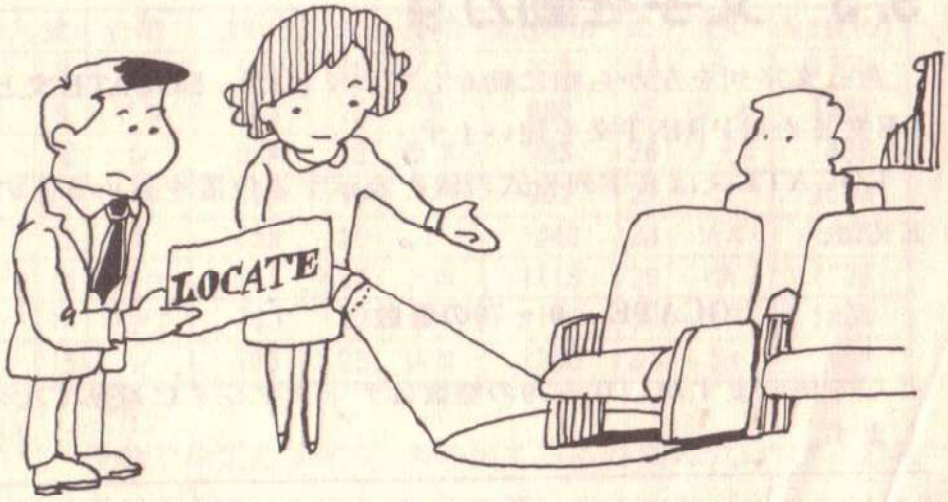
下の行にファンクションキーの内容を表示しているときは，0～39が有効となります．上の行に内部時計，下の行にファンクションキーの内容を表示しているときは，0～31が有効となります．下の行に内部時計を表示しているときは，0～63が有効となります．0～79まで指定できるのは，

CONSOLE　0または1，0，0

としたときです．

次のプログラムは，文字Aがディスプレイの左上隅から右下隅まで移動します．

```
10 CONSOLE 1,0,0
20 FOR I=0 TO 79
30 PRINT CHR$(12);
40 LOCATE I
50 PRINT "A";
60 NEXT I
70 END
```



30行のPRINT文は，画面を消去するためのものです．CHR$(12)を表示することによって画面を消去します．また30行を実行しないと画面全体に文字Aを表

9.5　1文字の読み込み－INKEY$－

入力によく使われるのはINPUT文ですが，PC-2001にはINKEY$というキーボードの押されているキーを文字として与える関数があります．

INKEY$はどのキーも押されていないときには，ヌルキャラクタ(CHR$(0)または""で表わします)を与えます．

```
100 IF INKEY$="" THEN 100
```

とすると，いずれかのキーが押されるまで100行でループします．

INKEY$はゲームなどでキャラクタを動かしたりするのに非常によく使う関数で，RETURNキーを押さなくても文字を1文字入力できるのが特徴です．

次のプログラムは画面を2行にみたて，キーによって文字Aを移動させるもので，よくゲームに使われるプログラムの一部です．キーは，4…左，6…右，8…上，2…下へ移動するのに使います．

```
10 CONSOLE 0,0,0
20 LX=20:LY=0
30 PRINT CHR$(12);
40 LOCATE LY*40+LX:PRINT "A";
50 S$=INKEY$:IF S$="" THEN 50
60 IF S$="4" THEN LX=LX-1
70 IF S$="6" THEN LX=LX+1
80 IF S$="2" AND LY=0 THEN LY=1
90 IF S$="8" AND LY=1 THEN LY=0
100 IF LX<0 THEN LX=0
110 IF LX>39 THEN LX=39
120 GOTO 30
```

PC-2001のRND関数は，電源ONから繰り返すと常に同じ乱数を発生します．これはこれでよいのですが，さいころとしてはちょっと困ります．そこで，実行ごとに違う乱数を発生させるために，**RANDOMIZE 文**があります．

RANDOMIZE　式

としてRND関数を使う前に実行すると，実行ごとに式の値に基づく乱数を発生するようになります．そして，式の値を変えながら実行すると，乱数の系列を実行ごとに変えることができます．また，式を省略すると実行を中断し，

(0－65529) ?

を表示しますから，範囲内の整数を入力するとその値を式の値とした系列の乱数が用意され，実行を続けます．

RND関数が発生する乱数は0以上1未満の乱数です．1～Nの整数の乱数を発生するには，

INT (RND (1) *N) + 1

とします．INTは関数で，

INT (X)

とすると，Xより小さく，かつ最大の整数を与えます．たとえば，

INT (5.5)

は，5となります．

次のプログラムは，さいころを2個表示するもののですが，GOSUB文を増やせばさいころをより多く表示することができます．

```
10 RANDOMIZE
20 GOSUB 100
30 GOSUB 100
40 PRINT
50 GOTO 10
100 PRINT INT(RND(1)¥6)+1;"  ";
110 RETURN
```

■＆ｎ個の空白＆

式のリストとして与えられた文字列の内，2＋ｎ文字が出力されます．文字列が2＋ｎ文字より長い場合は，余分の文字は無視されます．また，2＋ｎ文字に満たない場合は，文字列は左づめに出力され，空いた右側には空白がつめられます．

これにより，文字列をいくつも表示するときに，各文字列の表示桁数を指定できますから，桁数をそろえて，美しく整った表示をすることができます．

```
10 A$="USA"
20 B$="JAPAN"
30 C$="FRANCE"
40 PRINT A$;",";B$;",";C$
50 PRINT USING "& &";A$;B$;C$
60 PRINT USING "&     &";A$;B$;C$
```

この例では，行番号50で先頭より3桁，行番号60では先頭より6桁表示しますから，

```
USA,JAPAN,FRANCE
USAJAPFRA
USA    JAPAN FRANCE
```

となります．この例では，フォーマット文は行番号50，60ともにＡ＄の出力に対するものしかありませんが，Ｂ＄，Ｃ＄と続いている場合，何度でもフォーマット文が先頭から使用されます．

■＃

この記号は各桁の位置を示すのに使います．この指定された桁は常に数字で満たされます．しかし指定した桁数の方が少ない場合には，その領域に右づめで出力され，左側の空いた部分には空白がつめられます．

もし，指定された数値領域の長さよりも出力される数値の桁数の方が長い場合には，数値の前にパーセント記号(％)が出力されます．

```
10 A=123
20 PRINT USING "###";A
30 PRINT USING "######";A
40 PRINT USING "##";A
```

10. 結果をそろえて表示する —PRINT USING—

結果をそろえて表示するには，単にPRINT文を使うだけではうまくいきません．そのような場合に，**PRINT USING文**を使って書式を指定することにより，希望どおりに表示することができます．書式を指定するには，

PRINT USING"＃＃＃";A

のように，USINGの後に，"でくくった，＃．，＋＊＆などの文字を桁数分だけ並べます．これらの文字を**フォーマット文字**といい，フォーマット文字を用いて表示の形式を細かく指定する文を**フォーマット文**といいます．**フォーマット文**に続いて変数や式をセミコロンで区切って書くことができます．最後にセミコロンがあると，実行を継続し以降の表示は現在の表示に続いて行なわれます．セミコロンがないときは RETURN キーが押されるまで実行を停止します．したがって，1行の表示を，いくつかのPRINT文またはPRINT USING文により行なうこともできます．それぞれのフォーマット文字は次のような意味を持ちます．

■！

式のリストとして与えられた文字列の最初の文字だけを出力するように指定します．

たとえば，文字列全体とその省略形としての先頭の一文字との対応を表示するためには，次のようにすることができます．

```
10 A$="JAPAN"
20 PRINT USING "!";A$;
30 PRINT "=";A$
```

これを実行すると，

```
J=JAPAN
```

となります．

これを実行すると，

```
+258   -147
258+   147-
```

となります．

■－

フォーマット文の最後にマイナス記号を書くと，負の数値の後ろにマイナスが表示されます．

```
10 A=369
20 B=-753
30 PRINT USING "######-";A;B
```

これを実行すると，

```
 369    753-
```

となります．プラス記号とは違って，正の数値の後ろには空白がプリントされます．また，マイナス記号を数値の前に書くことはゆるされません．

■＊＊

フォーマット文の最初に書くことにより，数値領域の初めの空白部分にアスタリスク(＊)を出力します．また＊＊は，さらに2桁分の領域を確保します．

```
10 A=2.72
20 PRINT USING "**#.##";A
30 PRINT USING "**#.##";A*10
40 PRINT USING "**#.##";A*100
```

これを実行すると，

```
**2.72
*27.20
272.00
```

これを実行すると，

```
123
  123
%123
```

となります．行番号40では指定した桁数の方が少ないことを意味するため，前にパーセント記号(%)が出力されています．

■．（小数点）

領域の任意の場所に小数点を挿入します．フォーマット文の指定に小数点に続く桁がある場合，それらの桁は必ず出力されます．

```
10 PI=3.14
20 PRINT USING "#.#";PI
30 PRINT USING "#.##";PI
40 PRINT USING "#.#####";PI
```

これを実行すると，

```
3.1
3.14
3.1400
```

となります．行番号40では，小数点以下の桁数より，指定した桁数の方が多いので，その桁のところに"0"が出力されます．

■+

フォーマット文の最初，または最後のプラス記号は，数値の前または後ろにその数値の符号をつけます．

```
10 A=258
20 B=-147
30 PRINT USING "+######";A;B
40 PRINT USING "######+";A;B
```

■,（コンマ）

フォーマット文の小数点の左側にコンマがある場合は，小数点の左側の数値の3桁ごとにコンマが出力されます．指数形式とともに使われた場合は無効となります．

```
10 A=1024
20 B=123.5
30 PRINT USING "##########,";A
40 PRINT USING "##########,";A*1000
50 PRINT USING "#######,.#";B
60 PRINT USING "#######,.#";B*1000
```

これを実行すると，

```
     1,024
1,024,000
     123.5
123,500.0
```

となります．

■^^^^

フォーマット文字#に続いて4つの上向き矢印をおくと，指数形式の書式を指定します．4つの上向き矢印は，E+nnが出力される領域を確保します．

```
10 PI=3.14
20 PRINT USING "##.##^^^^
    ";PI;PI*1000;PI/1000
```

これを実行すると，

```
3.14E+00    3.14E+03    3.14E-03
```

となります．

となります．行番号40では空白部分がないので，アスタリスク(＊)は出力されません．

■￥￥

フォーマット文の最初に書くことにより，出力される数値の直前に円記号を出力します．￥￥は数字2桁分の位置を確保しますが，円記号にはそのうち1桁を使います．指数型式のときは￥￥を使用できません．

```
10 A=1024
20 B=-2048
30 PRINT USING "￥￥######";A
40 PRINT USING "￥￥######";B
```

これを実行すると，

```
 ￥1024
-￥2048
```

となります．

■＊＊￥

フォーマット文の最初に書くことにより，＊＊と￥￥の両方の機能を果たします．上記の条件は全て適用されますが，異なる点は，＊＊￥が数字3桁分の位置を確保し，そのうちの1個は円記号となります．

```
10 A=2.56
20 PRINT USING " **####.## ";A;A*100
30 PRINT USING "￥￥####.## ";A;A*100
40 PRINT USING "**￥###.## ";A;A*100
```

これを実行すると，

```
  ****2.56   **256.00
    ￥2.56   ￥256.00
 ****￥2.56 **￥256.00
```

となります．

11. いろいろなコマンド

コマンドとはプログラムを実行したり，プログラムのリストをとったりするために用意されている命令です．

コマンドは，一般にダイレクトモードで実行し，実行が終わると再びコマンドレベル(コマンド入力を受けつける状態)に戻ります．プログラム中で使えるコマンドもありますが，一部を除き実行後直ちにコマンドレベルに戻ります．

この節では，これらコマンドについて解説します．

11.1 プログラムを実行する－RUN－

プログラムを実行するためのコマンドが**RUNコマンド**です．

RUN RETURN

とすると，プログラムの一番最初の行から実行を開始します．また，

RUN 行番号 RETURN

とすると，その行から実行を開始します．この場合，存在しない行番号を指定するとエラーになります．

11.2 リストをとる－LIST－

プログラムを画面に表示することを**リストをとる**といいます．そのためのコマンドが**LISTコマンド**です．PC-2001の画面には1行のプログラムラインしか表示できませんから，

LIST RETURN

とすると，プログラムの最初の行を表示します．また，

LIST 行番号 RETURN

■任意の文字列の表示

フォーマット文中に，フォーマット文字以外の文字を書くと，その文字はそのまま画面に表示されます．

```
10 A=4096
20 B=2
30 PRINT USING "#####*#=####";A;B;A*B
40 PRINT USING "¥¥#####     #.#     ####
   ^^^^";A;B;A^B
```

これを実行すると，

```
4096*2=8192
 ¥4096     2.0      168E+05
```

となります．

11.4　行番号をつけ直す—RENUM—

プログラムの行番号の並びは，最初10おきにつけていたとしても，途中で何回かの修正の結果，不規則になってしまいます．

RENUMコマンドは行番号の並びを変更して整理するものです．

RENUM RETURN

とすると，プログラムの最初の行の行番号を10として，10おきに行番号をつけ直します．また，

RENUM 行番号$_1$，行番号$_2$，増分

というように使うと，任意の行から，新しい行番号で，任意の間隔で行番号をつけ直すことができます．行番号$_1$で示す行番号は新しい先頭の行番号で，省略することもでき，省略すると10とみなされます．

行番号$_2$で示す行番号はもとのプログラムの行番号で，つけ換えたい先頭の行番号です．

増分は新しくつけ直される行番号の間隔です．省略すると10とみなされます．次のプログラムに対して，

```
10 INPUT "ｷｮｳｶｽｳ=";N
20 OPTION BASE 1
30 DIM KY(N)
40 FOR I=1 TO N
50 PRINT "ｷｮｳｶ";I;"=";
60 INPUT KY(I)
70 NEXT I
80 SO=0
90 FOR I=1 TO N
100 SO=SO+KY(I)
110 NEXT I
120 HE=SO/N
130 PRINT "ｺﾞｳｹｲ ﾊ";SO;"ﾃﾝ";
140 PRINT "   ﾍｲｷﾝ ﾊ";HE;"ﾃﾝ"
150 END
```

とすると，その行を表示します．LISTコマンドは，プログラムを確認したり修正するために表示するのに使います．

LISTコマンドにより表示されている行の上下の行を表示するには，▲と▼のキーを使います．▲キーを押すと1行前の行を表示し，▼キーを押すと次の行を表示します．また，長い行の場合に▶キーを押すと表示し切れない部分を見ることができます．◀キーは▶キーの逆で，戻るときに使います．表示が画面からあふれている場合に，SHIFTキーを押したまま，◀，▶キーを押すと，カーソルの位置は固定されたまま，表示全体が左右に移動します．

11.3 表示面を変える－CONSOLE－

今までは，2行あるディスプレイの下1行には，ファンクションキーの内容が表示されていましたが，**CONSOLEコマンド**を使うと，その表示を消して80文字のディスプレイにしたり，内部時計を表示したりすることができます．

CONSOLEコマンドは，

CONSOLE キー音スイッチ，時計スイッチ，
ファンクションキースイッチ RETURN

として使います．それぞれのスイッチは、0ならばOFF，1ならばONとなり左から順に以下を定めます．

❶キーのクリック音の有無を指定します．
❷内部時計の表示の有無を指定します．
❸ファンクションキーの表示の有無を指定します．

❶1番目のスイッチをONにすると，キーを押したときに音を発生し，OFFにするとキーを押しても音は出なくなります．

❷2番目のスイッチをONにすると，内部時計の値を表示します．

❸3番目のスイッチをONにすると，ファンクションキーの内容を2行目に表示します．この場合，ユーザーは上の行のみが使用できます．たとえば，

```
CONSOLE 0,0,0
```

とすると，全てのスイッチがOFFになります．

とすると，再実行します．ただし，実行を停止したあとで変数の内容を表示したりするのはかまいませんが，プログラムの修正をすると，CONTコマンドでの実行はできなくなります．

11.6 プログラムの消去－NEW－

PC-2001は電源を切っても以前に入力したプログラムやデータを保存しています．そのため，新しいプログラムを入力する場合には，メモリの中味を消去する必要があります．そのコマンドが**NEWコマンド**です．

NEW RETURN

とすると，プログラム，データの全てを消去します．一瞬のうちに消去してしまいますから誤まって大切なプログラムを消さないようにしてください．

11.7 データの消去－CLEAR, ERASE－

NEWコマンドは，プログラムとデータの全てを消去しますが，**CLEARコマンド**を使うとデータのみを消去します．

また，CLEARコマンドを使うと文字領域のサイズを変更できます．

CLEAR 500 RETURN

とすると文字領域として，500バイト用意します．

CLEARコマンドを使わない場合は文字領域に300バイトが割り当てられます．

ERASEは本来は文ですが，ダイレクトモードにして，

ERASE A RETURN

とすると，配列のみを全て消去します． また，**ERASE**のあとに配列名をつけると，その配列のみを消去できます．

11.8 LOCKとUNLOCK

PC-2001では，プログラムを誤って消去しないために，**LOCK機能**があります．

```
RENUM 200,120,5
```
RETURN

を実行すると，

```
10 INPUT "ｷｮｳｶｽｳ=";N
20 OPTION BASE 1
30 DIM KY(N)
40 FOR I=1 TO N
50 PRINT "ｷｮｳｶ";I;"=";
60 INPUT KY(I)
70 NEXT I
80 SO=0
90 FOR I=1 TO N
100 SO=SO+KY(I)
110 NEXT I
200 HE=SO/N
205 PRINT "ｺﾞｳｹｲ ﾊ";SO;"ﾃﾝ";
210 PRINT "  ﾍｲｷﾝ ﾊ";HE;"ﾃﾝ"
215 END
```

となります．

11.5　実行を継続する―CONT―

プログラムの実行を終了する命令はEND文ですが，プログラムの実行を中断する命令に**STOP文**があります．

STOP文は，プログラム中の任意の場所に置くことができ，BASICはSTOP文を実行すると，次のようなメッセージを出して停止します．

BREAK IN *nnnnn*

ここで，*nnnnn*は停止した行の行番号です．また逆にCONTコマンドは停止したプログラムを再実行するためのもので，

CONT RETURN

12. プログラムやデータを保存する

ここではデータやプログラムをカセットテープに記録したり，それをメモリに読み込む方法について説明します．

12.1 プログラムのセーブ,ロード

カセットテープにプログラムやデータを記録することを**セーブ**，逆に読み込むことを**ロード**といいます．プログラムのセーブは，カセットを録音状態にしてから，

CSAVE "ファイル名" RETURN

とします．**ファイル名**とは，記録したプログラム全体につける名前のことで，内容がわかるようなファイル名をつけます．

逆に，プログラムをロードするには，カセットを再生状態にしてから，

CLOAD "ファイル名" RETURN

とします．するとPC-2001は，そのファイル名のプログラムを探してロードします．このとき，テープを巻き戻し，または早送りをして記録されている部分の直前までテープを進めてからロードしてください．

ロード中にエラーが起きたときはボリュームを変えてもう一度ロードします．

12.2 プログラムの照合

プログラムが正しくセーブできたことを確認するために，照合命令があります．せっかく記録したプログラムがロード不可能だったでは泣くに泣けませんから，セーブの後には必ず照合を行なう習慣をつけたいものです．

照合には，

CLOAD? "ファイル名" RETURN

LOCK RETURN

とすると，現在メモリに記憶されているプログラムに対してプログラムの修正や，NEW，CLEAR，CLOAD，ERASE，DELETEコマンドは無効となり，それらを使うとエラーになります．

UNLOCK RETURN

とするとLOCKを解除します．

11.9 行の消去—DELETE—

DELETEコマンドを使うと，指定した行を消去します．このコマンドには書式がいくつかあります．

❶ **DELETE 行番号**

行番号で示す行を消去します(行番号 RETURN とするのと同じです)．

❷ **DELETE 行番号$_1$ー行番号$_2$**

行番号$_1$から行番号$_2$までの行を全て消去します．

❸ **DELETE 行番号ー**

行番号で示す行以降を全て消去します．

❹ **DELETE ー行番号**

行番号で示す行以前の行を全て消去します．

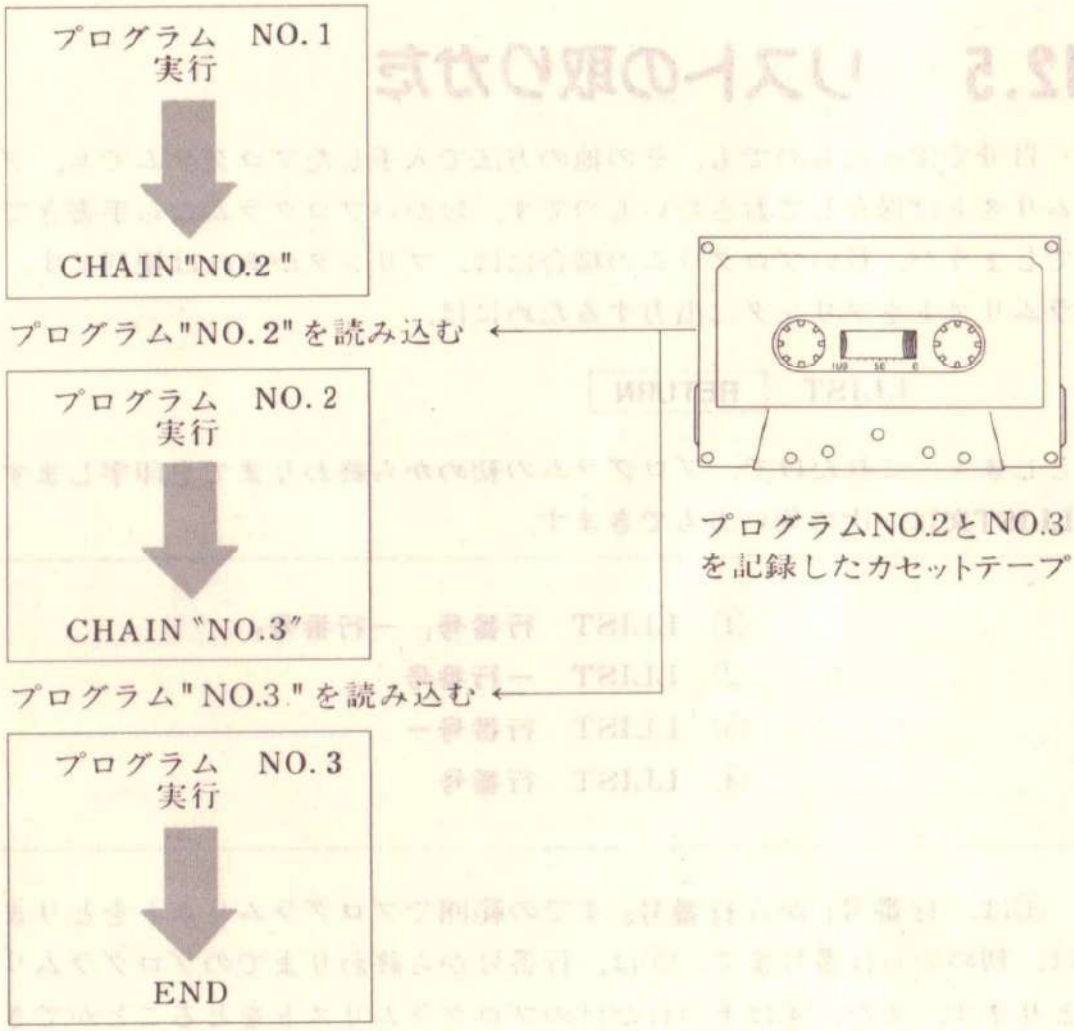
もし，指定した行番号がない場合はエラーになります．カセットレコーダ，プリンタに関するコマンドについては第12節を，RS-232Cインタフェースについてのコマンドは第14節を参照してください．

命令を実行すると，以前のプログラムは消えてしまうということです．メモリにあるプログラムを実行し，カセットのプログラムをチェインして実行し，さらに以前メモリにあったプログラムを実行しようとしてもできません．ただし数値変数は消えません．文字変数は消えることがありますから，必要な文字変数は以前のプログラム中でヌルストリング("")とつなぎます．たとえば，A$をチェインするプログラムに引き継ぐには，

```
A$="ABC"+""
```

とします．

CHAIN 命令でプログラムを次々に実行する様子を図解すると次のようになります．



します．この命令実行によって，不一致が生じた場合にエラーになるので，このときは，再びセーブを行ないます．また不一致がなければプロンプト表示（>）となります．

プログラムをセーブするときに，間違えてCLOAD命令を実行してしまうと，せっかくのプログラムが消えてしまいます．RETURNキーを押す前に，もう一度確かめてください．

12.3 プログラムの結合

メモリにあるプログラムを破壊せずに，その後ろに続けてカセットからプログラムを入力することを，アペンドといいます．よく使うサブルーチンプログラムをカセットにセーブしておけば，プログラムを作るたびにそのサブルーチンをキーボードから入力する必要はありません．

そのときは，

APPEND "ファイル名" RETURN

とすればよいのです．ただし，このときメモリにあるプログラムの最大行番号より，カセットのプログラムの最小行番号の方が大きくなければなりません．

そうでないときは，メモリにあるプログラムの最大行番号をRENUM命令を使って小さくしておきます．

またアペンドしたプログラムをさらにアペンドすることもできます．そのときもアペンドする前にそのつどRENUM命令を実行した方がよいでしょう．

プログラムが大きすぎてメモリに収まり切らないときは，プログラムをいくつかに分けてカセットテープにセーブしておき，それを次々に読み込みながら実行を続けていくことができます．このことをチェインといいます．チェインするためには，プログラム中で，

CHAIN "ファイル名"

または，

CHAIN "ファイル名", 行番号

とします．前者はカセットから入力したプログラムの先頭の行番号から，後者は，指定された行番号から実行します．注意しなければならないことはCHAIN

12.6　結果の印字

PRINT文の代わりにLPRINT文を使うことにより，プリンタに印字することができます．PRINT文とLPRINT文を使うことにより，表示面とプリンタのどちらへでも自由に表示を行なうことができますから，たとえば入力メッセージを表示面に表示し，計算結果だけをプリンタに印字したりすることもできます．

またLPRINT USING文も使えます．これについては，PRINT USING 文を参照してください．

12.4　データのセーブ,ロード

データをセーブ，ロードすることも可能です．セーブするためには，

PRINT　＃－１，変数名１，変数名２，……

とします．すると指定された変数の内容をセーブします．データをロードするには，

INPUT　＃－１，変数名１，変数名２，……

とします．このとき，変数の型および個数は，PRINT　＃－１で指定したものと同じでなければなりません．

12.5　リストの取りかた

自分で作ったものでも，その他の方法で入手したプログラムでも，プログラムリストは保存しておきたいものです．短かいプログラムなら手書きでもよいでしょうが，長いプログラムの場合には，プリンタがあれば便利です．プログラムリストをプリンタに出力するためには，

LLIST RETURN

とします．これだけで，プログラムの初めから終わりまでを印字します．この**LLIST文**は，次の使い方もできます．

① LLIST　行番号$_1$ －行番号$_2$
② LLIST　－行番号
③ LLIST　行番号－
④ LLIST　行番号

①は，行番号$_1$ から行番号$_2$ までの範囲でプログラムリストをとります．②は，初めから行番号まで，③は，行番号から終わりまでのプログラムリストをとります．また，④はその行だけのプログラムリストをとることができます．

次のプログラムは，１次方程式$ax+b=c$と２次方程式$ax^2+bx+c=0$の実数解を求めるものです．

```
10 INPUT "1ｼﾞ=1, 2ｼﾞ=2 ";N
20 ON N GOSUB 100,200
30 IF N=1 OR N=2 THEN 10
40 WAIT 10:PAUSE "ﾃﾞｷﾏｾﾝ !!":GOTO 10
50 END
100 INPUT "aX+b=c   a,b,c";A,B,C
110 PRINT "ｶｲ =";(C-B)/A
120 RETURN
200 INPUT "aX^2+bX+c=0   a,b,c";A,B,C
210 D=B^2-4*A*C
220 IF D<0 THEN PRINT "ﾌｸｿｶｲ ﾃﾞｽ":RETURN
230 X1=(-B+SQR(D))/(2*A)
240 X2=(-B-SQR(D))/(2*A)
250 PRINT "ｶｲ ﾊ ";X1;"ﾄ ";X2;"ﾃﾞｽ"
260 RETURN
```

13.2　コメントをつける－REM－

プログラムが長くなると，プログラムのそれぞれの部分が何をやっているのかがわかりにくくなります．また，自分で作ったプログラムでも何日もすると何をするプログラムなのかわからなくなることもあります．

このようなときにプログラムに**コメント**をつけておくと，後の理解の助けになります．コメントは，

```
100 REM コメント
```

または，

```
100 ' コメント
```

のように書きます．

13. その他の命令

13.1 ON～GOTO文とON～GOSUB文

GOTO文やGOSUB文は定められた文番号への分岐を行ないましたが，**ON～GOTO文**と**ON～GOSUB文**は，いくつかの行番号のうちの一つに分岐します．

```
ON A GOTO 100, 200, 300
```

とするとAの値が1ならば100行へ，2ならば200行へ，3ならば300行へ分岐します．また，Aの値が0または3より大きい場合は，分岐せずに次の行へ実行が移ります．

行番号の並びはいくつあってもかまいません．ON～GOSUB文も同じように書かれ，ON～GOTO文と違う点は分岐ではなく，サブルーチンを呼び出す点です．

次のプログラムは，Nを入力した後，何を求めるかを入力すると，Nの平方根，2乗，3乗のいずれかを表示します．

```
10 INPUT "N=";N
20 INPUT "1=ﾙｰﾄ, 2=2ｼﾞｮｳ, 3=3ｼﾞｮｳ";M
30 ON M GOTO 60,70,80
40 WAIT 10:PAUSE "Data Error!!
50 GOTO 10
60 PRINT SQR(N):GOTO 10
70 PRINT N*N:GOTO 10
80 PRINT N^3:GOTO 10
90 END
```

```
530 M=VAL(MID$(DT$,4,2))
540 D=VAL(RIGHT$(DT$,2))
550 RETURN
```

500行からサブルーチンを呼び出すとYに年，Mに月，Dに日が数値としてセットされます．

2つのタイマーは両方とも電源を切っても電池のある限り作動しますが，初期設定をするには，

```
TIME$="HH：MM：SS"
DATE$="YY/MM/DD"
```

のように文字列で指定します．

13.4 ファンクションキー —KEY,KEYLIST—

ファンクションキーは1回押すと定義した文字が入力されます．ファンクションキーは10個あり，[f・1]～[f・5]はそのままで，[SHIFT]キーを押しながら[f・6]～[f・10]は[f・1]～[f・5]を押すことにより入力できます．あらかじめシステムが定義するファンクションキーは，

[f・1]	PRINT	[f・6]	CSAVE
[f・2]	INPUT	[f・7]	KEY
[f・3]	GOTO	[f・8]	CLOAD
[f・4]	LIST	[f・9]	CONSOLE
[f・5]	RUN [RETURN]	[f・10]	CONT [RETURN]

となっています．

KEY LIST [RETURN]

とするとファンクションキーの内容を全て表示します．

内容は一度に表示しきれませんから，右側に隠れている部分は[SHIFT]キーと[▶]キーを押して表示させます．

13.3 TIME$とDATE$

PC-2001は内部にタイマーを持っていますから，日付けや時刻を表示したり，プログラムの一部に組み込むこともできます．

PRINT TIME$ RETURN

とすると，

HH：MM：SS

の形の文字列として時分秒を表示します．HHは時間，MMは分，SSは秒をそれぞれ2桁で表わします．

時分秒を数値として取り出すには，次のようにします．

```
1000 ' ｼﾞｺｸ ヲ ﾄﾘﾀﾞｽ
1010 TI$=TIME$
1020 H=VAL(TI$)
1030 M=VAL(MID$(TI$,4,2))
1040 S=VAL(RIGHT$(TI$,2))
1050 RETURN
```

1000行からのサブルーチンを呼び出すことにより，H，M，Sにそれぞれ時分秒が数値としてセットされます．

日付けを表示するには，

PRINT DATE$ RETURN

とすると，

YY/MM/DD

の形の文字列として年月日を表示します．YYは年，MMは月，DDは日をそれぞれ2桁で表わします．

年月日を数値として取り出すには次のようにします．

```
500 ' ﾋﾂﾞｹ ヲ ﾄﾘﾀﾞｽ
510 DT$=DATE$
520 Y=VAL(DT$)
```

DEFINT　文字範囲 …… 整数型宣言
DEFSNG　文字範囲 …… 単精度型宣言
DEFSTR　文字範囲 …… 文字型宣言

として使います．これらを型宣言文字といいます．また，

DEFINT　A，C－P，X－Z

のように間を開けながら宣言することもできます．

変数の型宣言は通常は，型宣言文字で行ないますが，同じ型の変数がたくさんある場合に，型宣言文を使うと便利です．

RETURN キーの入力により，コマンドレベルに戻ります．ファンクションキーを定義するには，

KEY キー番号，"文字列" RETURN

とします．文字列としては，キーボードから打ち込めるものはそのまま，それ以外はCHR$とASCIIコードを使って定義します．その場合，画面上ではピリオド(．)で表示されます．f・1 に，

LLIST RETURN

を定義するには，

KEY1, "LLIST"+CHR$(13) RETURN

とします．

13.5 変数の型宣言－DEFINT/SNG/STR－

変数には型があることは前に書きましたが，次のようなものでした．

A% …… 整数型変数
A！ …… 単精度型変数
A …… 〃
A$ …… 文字型変数

これら変数名の後につけられた%，！，$を型宣言文字といいますが，これらを関数名の後につけてユーザー定義の関数を作ることもできます．

変数や関数の型を決めるもう一つの方法が，

DEFINT, DEFSNG, DEFSTR

を使う方法です．これは，変数や関数名の最初の1文字によって型を決めてしまう方法です．たとえば，

DEFINT A－E

とすると，最初の文字がA～Eで始まる変数は全て整数型となります．つまり，

TERM コード，パリティ，ボーレート

コードは，A，a，J，jのいずれかで指定し，Aまたはaを指定したときは7ビットASCIIコード，Jまたはjを指定したときは8ビットJISコードとしてデータが転送されます．8ビットJISコードを指定すると，データの最上位ビットで英数字とカナとを切り換えるモードとなります．7ビットASCIIコードを指定した場合には，英数字とカナの切り換えをビットによって行なうことができません．転送中にSOコード(0EH)がくると，それ以後のデータをカナとみなし，SI(0FH)がくるとそれ以後のデータを英数字とみなしてしまいます．

パリティは0～2の値で指定します．0を指定するとパリティなし，1を指定すると奇数パリティ，2を指定すると偶数パリティとなります．

パリティはデータ転送中の誤りを検出するためのもので，1バイトのデータは7ビットまたは8ビットで転送されますから，1となっているビットの個数が偶数になっているか奇数になっているかによって誤りを検出します．奇数パリティを指定すると全てのデータは1となっているビットの総数が奇数，偶数パリティを指定すると1となっているビットの総数が偶数として転送されます．また，指定パリティに反するデータを受け取るとエラーになります．

ボーレートは，0～5の値で指定します．指定した値とボーレートの関係は下表の通りです．

ボーレート

0	110ボー
1	150ボー
2	300ボー
3	600ボー
4	1200ボー
5	2400ボー

ボーレートは，データの転送速度であり，1秒間に何ビットのデータを送るのかを定めるものです．

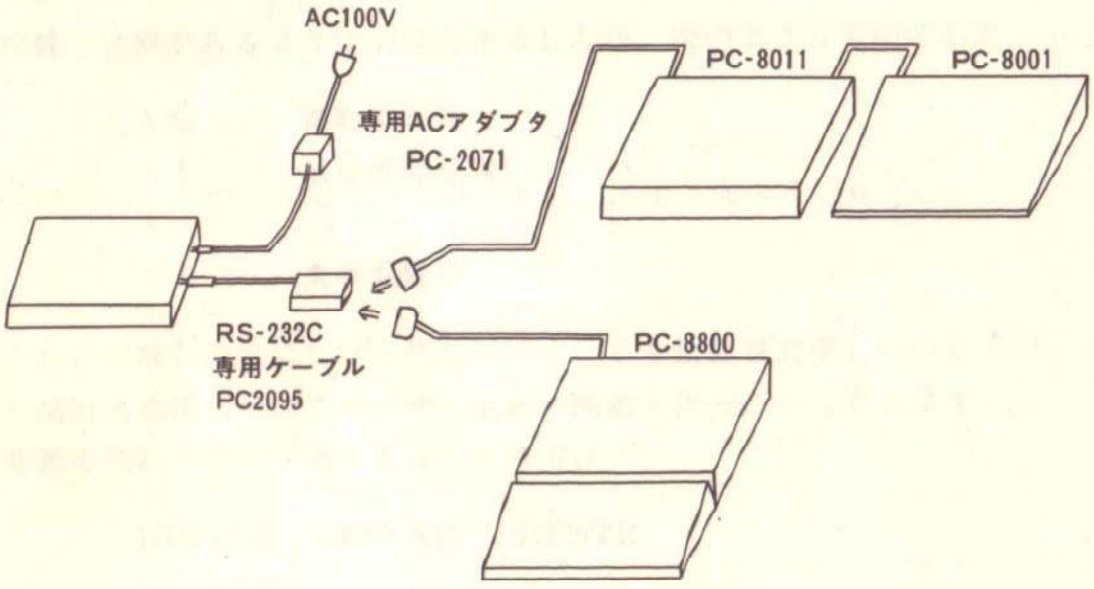
これらのモードは，データを送る側，受ける側の両者で一致していないとデータは正しく転送されません．

14. PC-8000,PC-8800 システムとの接続

PC-2001は，オプションのRS-232C ケーブル(PC-2095)と，ACアダプタ(PC-2071)とを同時に使うことにより，容易に他のRS-232Cインタフェースを持つ機器と接続して，データの転送を行なうことができます．

下図のようにPC-8000，あるいはPC-8800システムと接続することにより，PC-2001とのデータ・コミニュケーションが行なえます．

ここでは，PC-2001とPC-8000システムとの接続について説明しますが，PC-8800システムと接続する場合，および詳細についてはPC-2095の取扱説明書も合わせて参照してください．



14.1 データ転送のための命令

BASICによって，PC-8000との間でデータ転送を行なうためには，TERMコマンド，INPUT%１文，PRINT%１文を使います．

TERMコマンドは，インタフェースのモードを規定するコマンドで，次の形式で使います．

14.2 PC-8000システムとの接続例

PC-2001とPC-8000システムとをRS-232Cケーブル(PC-2095)を用いて接続してください．さらに，PC-2001にACアダプタ(PC-2071)を接続してください．

なお，これらの接続については，電源を切ってから確実に行ないます．詳しくは，各製品の取扱説明書を参照してください．

ここでは例として，PC-8000側から送られたデータを順次表示する例と，PC-2001から送ったデータをPC-8000側で順次表示する例を示します．

サンプルプログラム1

PC-2001からPC-8000システムへの出力

■PC-2001側

```
1000 ' PC-2001 ---> PC-8000 (PC-2001)
1010 '
1020 TERM J,0,4                    8ビット,パリティなし,1200ボー
1030 PRINT #1,DT$                  データ出力
1040 INPUT #1,A$                   確認コードの入力
1050 IF A$<>"#" THEN 1040          確認コードの判定
1060 RETURN
```

このプログラムはサブルーチンになっていますから，PC-8001に送りたい文字データを変数DT$にセットして，このサブルーチンを呼び出します．

■PC-8000側

```
1000 ' PC-2001 ---> PC-8000 (PC-8000)
1010 '
1020 INIT #1,&HCE,&H37
1030 INPUT #1,DT$                  8ビット,ストップビット2,パリティなし
1040 PRINT #1,"#";CHR$(13);        データの入力
1050 PRINT DT$                     確認コードの出力
1060 RETURN                        データの表示
```

PC-2001からRS-232Cインタフェースを通して出力する場合はPRINT%1文を使います.

PRINT%1文は,

PRINT%1, 出力するデータの並び

として使い, 出力するデータの並びは",で区切ります. たとえば, 変数A, B, C, Dの値を出力するならば,

PRINT% 1, A, B, C, D

文字列"PC-2001"と変数A$の値を出力するには,

PRINT% 1, "PC−2001", A$

のようにします.

また, PRINT%1文の最後を上の例のように何もつけないで指定すると, CRコード(13)を付加し, ";"を最後につけて指定すると, CRコードは付加されません.

逆にRS-232Cインタフェースを通して, PC-2001にデータを入力する場合は, INPUT%1文を使います.

INPUT%1文は,

INPUT%1, 変数の並び

として使い, 変数の並びは",で区切ります.

入力されたデータは, 255バイトのバッファに格納され, バッファ内からCRコードまでの文字を入力文字列とみなして, それらを指定された変数に代入します.

そのため, バッファにCRコードがなければ(CRコードが入力されなければ)CRコードが入力されるまで入力待ちとなります.

INPUT%1文による入力データが読み込まれないうちに, 次のデータがRS-232Cインタフェースを通して送られると, 以前に入力したデータがこわされオーバーランエラーが生じます. そのため, INPUT%1文でデータを入力後, 相手側機器にPC-2001から入力終了コードを送り, 相手側もそのコードを受けた後に次のデータを送るようにプログラムを作ってください.

■PC-8000側

```
1000 ' PC-8000 ---> PC-2001 (PC-8000)
1010 '
1020 INIT %1,&HCE,&H37
1030 PRINT %1,DT$;CHR$(13); 8ビット，ストップビット2，パリティなし
1040 INPUT %1,A$            データの出力
1050 IF A$<>"#" THEN 1040   確認コードの入力
1060 RETURN                 確認コードの~~判定~~判定
```

使用方法はサンプルプログラム1と同じです.

このプログラムもサブルーチンになっていますから，適当な形でメインルーチンから呼び出します．

また，PC-2001，PC-8000システムのINPUT%1文によってデータ受信を行なう場合，データが読み込まれないうちに次のデータが受信されたときには，オーバーランエラーになります．ですから，上の例のようにデータ受信終了後，送信側へデータ受信確認コードを送信し，送信側は確認コードを受けつけるまで次のデータを送らないように工夫が必要です．なお，前ページの例ではデータ受信確認コードとして"#"を用いています．

また，PC-8000システムのPRINT%1文は，";"を最後につけないで使った場合には，CRコード(13)およびLFコード(10)を送信します．一方，PC-2001側でのINPUT%1文はCRコードのみを伝送終了コードとみなします．したがってLFコードは受信バッファ上に残ってしまい，次のデータの最初の文字として取り扱われてしまいます．

そこで，PC-8000側のPRINT%1文は，

PRINT%1，〈出力するデータの並び〉;CHR$(13)；

として，データの最後にCRコードを付加するようにして使います．

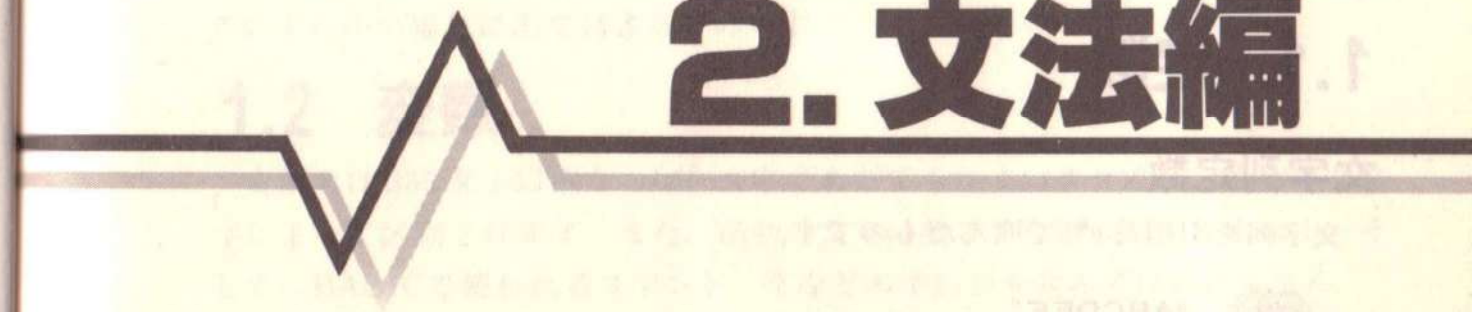
また，ストップビットの数は，2400ボー以下のときは1ビット以上，2400ボーのときは2ビット以上としてください．

サンプルプログラム2

PC-8000システムからPC-2001への出力

■PC-2001側

```
1000 ' PC-8000 ---> PC-2001 (PC-2001)
1010 '
1020 TERM J,0,4          8ビット，パリティなし，1200ボー
1030 INPUT #1,DT$        データ入力
1040 PRINT #1,"#"        確認コードの出力
1050 PRINT DT$           データの表示
1060 RETURN
```



2. 文法編

単精度数値定数とは，

1．有効桁7桁以下の数

2．Eを使った指数形式表現

3．最後に感嘆符(！)を伴なったもの

のいずれかの場合にあてはまるものです．

1.2 変数

変数名は255文字以内ならば何文字であってもかまいませんが，最初の2文字によって区別されます．また，最初の文字は英字でなければなりません．そして，BASICで使われるコマンド，文などの予約語を含んではいけません．

変数は数値または文字列のいずれかを表わします．文字変数の名前は最後に＄をつけます．

数値変数は，整数，単精度のいずれかの形式を持ちます．それぞれの形式は名前の最後に次の型宣言文字をつけることにより指定されます．

％……整数変数

！……単精度変数

もし，型宣言文字が省かれている場合または型宣言文で型宣言されていない場合は，その変数は単精度変数とみなします．

1.3 式と演算

式とは，ある値を得るために定数や変数を演算子で結合したものです．

演算子は与えられた値について，算術または論理演算を実行します．

1.3−1 算術演算子の優先順位

算術演算子には四則演算，べき乗があり，それぞれ演算の優先順位があり，表の上位にあるほど優先順位が高くなります．

演算子	演　　算
^	べき乗
+，−	符号
＊，／	乗算および単精度形式での除算
￥，MOD	整数除算および剰余算
+，−	加算，減算

演算の順序を変える場合は，カッコを使用します．

1. 概要

1.1 定数

文字列定数

文字列を引用符(")で囲んだものです.

例 "ABCDEF"

数値定数

数値定数には次の形式があります.

■整数形式：－32768～＋32767の全ての整数で，コンマや小数点を含まないものです．プラスの符号は省略できます.

■固定小数点形式：正または負の小数点を含む数です.

■浮動小数点形式：指数形式で表現された正または負の数です．指数部の範囲は－38～＋38です.

例 235.98E－7(＝0.000023598)

2359E6(＝2359000000)

■16進形式：&Hを伴なった16進数で，この形式で入力された16進数は10進数で出力されます.

例 &H76

&H32F

■8進形式：&Oまたは&を伴なった8進数で，この形式で入力された8進数は10進数で出力されます.

例 &O347

&1234

1.1－1 単精度の数値定数

数値定数は単精度の浮動小数点数です．7桁までの精度で格納され，6桁までの桁数で表示されます.

2. コマンド,文(ステートメント),関数

この節では，BASICの持つ全てのコマンド，文，関数について説明します．

機 能	そのコマンド，文，関数についての機能を説明します．
例	使い方を例示します．
解 説	そのコマンド，文，関数についての詳しい説明です．

■書式の記法

文またはコマンドの書式を示す場合には，以下の約束が適用されます．

1 アルファベットの大文字で記された項目はそのままの形で入力してください．つまり，CAPSモードで大文字にして入力する必要はなく，小文字のまま入力してかまいません．ただし，引用符で囲まれた文字列のなかで使う場合（文字定数）には大文字と小文字は区別して入力しなければなりません．

2 大カッコ｛ ｝で囲まれた項目は選択を示します．ユーザーはいずれか一つを選択します．

3 かぎカッコ〔 〕で囲まれた項目はオプションです．省略した場合，必要に応じてデフォルト値または以前に指定した値が適用されます．デフォルト値とはBASICがあらかじめ定めた値です．

4 かぎカッコおよび大カッコ以外のコンマ，カッコ，セミコロン，ハイフン，等号などの記号は示された位置に正しく入力してください．

5 省略記号 … の続く項目は，1行の許す長さ以内で任意の回数繰り返すことができます．

1.3-2 関係演算子

関係演算子は2つの値を比較するのに用いられます．比較の結果は**真**(－1)または**偽**(0)になります．

演算子	テストされる関係
=	等しい
<>, ><	等しくない
<	小さい
>	大きい
<=, =<	等しいか小さい
>=, =>	等しいか大きい

1つの式の中で算術演算子と関係演算子が同時に使われた場合は，常に算術演算が初めに実行されます．

1.3-3 論理演算子

ビット操作やブール演算を行なったり，いくつもの関係を調べたりするために論理演算の機能が用意されています．

論理演算子も次のように優先順位があり，表の上位にあるほど優先順位は高くなります．

演算子	演　　算
NOT	否定
AND	論理積
OR	論理和
XOR	排他的論理和

1つの式の中では，論理演算は算術および関係演算のあとで実行されます．

1.3-4 文字列の演算

文字列はプラス記号(＋)によって連結することができます．

ATN（式）　関数

機 能　式の値のアークタンジェント値（$\tan^{-1}x$）を与えます．

例
```
PRINT ATN(3)
 1.24905
```

解 説　角度の単位はラジアンですから，結果の範囲は$-\pi/2 \sim \pi/2$となります．

BEEP 式1〔，式2〕　文

機 能　ブザーを鳴らします．

例
```
BEEP 3,10
```

解 説　式1は音階を表わし，式2は音長を表わします．音階はmod32，音長はmod 256で表わされる負でない整数で，整数値と音階の関係は次の表のようになります．音の長さは式2の値÷10秒でデフォルト値は4です．また，式2が0の場合は，エラーになります．

式1	音階	周波数(Hz)	式1	音階	周波数(Hz)	式1	音階	周波数(Hz)	式1	音階	周波数(Hz)
0	無音	—	8	ド	525	16	ソ＃	833	24	ミ	1330
1	ファ	349	9	ド＃	553	17	ラ	880	25	ファ	1389
2	ファ＃	370	10	レ	584	18	ラ＃	933	26	ファ＃	1488
3	ソ	393	11	レ＃	625	19	シ	992	27	ソ	1563
4	ソ ＃	414	12	ミ	658	20	ド	1042	28	ソ＃	1645
5	ラ	440	13	ファ	702	21	ド＃	1116	29	ラ	1736
6	ラ ＃	466	14	ファ＃	735	22	レ	1179	30	ラ＃	1838
7	シ	492	15	ソ	781	23	レ＃	1250	31	シ	1953

⇨ 76ページ参照

ABS（式）　関数

機 能　式の値の絶対値を与えます.

例
```
PRINT ABS(7*(-5))
 35
```

APPEND "ファイル名"　コマンド

機 能　すでにメモリ内にあるプログラムとカセットテープ上のプログラムとを結合します.

例　APPEND "PROG1"

解 説　メモリ内のプログラムを破壊せずにカセットテープよりロードします.

ただし，追加されるカセットテープ上のプログラムの最初の行番号は，メモリ内のプログラムの最大行番号より大きくなければなりません.

⇨ 96ページ参照

ASC（文字列）　関数

機 能　文字列の最初の文字のASCIIコードを与えます.

例
```
PRINT ASC("TEST")
 84
```

解 説　1文字のASCIIコードを表わす数値が，この関数の値になりますから，値は0～255の整数値です.

文字列として，ヌルストリングを指定するとエラーになります.

⇨ 68ページ参照

CLOAD〔?〕"ファイル名"　コマンド

機能　カセットテープからメモリへプログラムをロードします.

例
```
CLOAD "MAX2"
CLOAD? "MAX2"
```

解説　**ファイル名**はCSAVEによりセーブしたときに指定した文字列の最初の6文字です.

?をつけるとメモリ内のプログラムとテープ上のプログラムファイルとを比較することによりベリファイを行ないます.

このコマンドはプログラム中では使用できません.　⇨ 95ページ参照

CONSOLE〔キークリック音発生スイッチ〕〔,時刻表示スイッチ〕〔,ファンクションキー表示スイッチ〕　コマンド

機能　画面の表示に関する機能を設定します.

例
```
CONSOLE 1,1,0
```

解説　各スイッチは，0のときOFF，0以外のときONとなります.

キークリック音発生スイッチをONにすると，キーが押されたとき，またはエラー発生時に音を発生します.

時刻表示スイッチをONにすると画面の右端に現在のタイマーの値を表示します.

ファンクションキー表示スイッチをONにすると画面の下段に現在のファンクションキーの内容が表示されます.　⇨ 90ページ参照

CHAIN "ファイル名"〔, 行番号〕 コマンド

機能 カセットテープからプログラムをロードし実行します.

例
```
CHAIN "CMT",100
```

解説 指定されたファイル名のプログラムをロードし, 指定された行番号から実行します. 行番号が指定されない場合は最小の行番号から実行します. CHAINコマンド実行前のプログラムは破壊されますが数値変数は破壊されません. ただし, 文字変数は演算が行なわれない場合には, 引き継がれたプログラムでの値は保証されなくなります.

⇨ 96ページ参照

CHR$（式） 関数

機能 ASCIIコードが式の値である文字を与えます. ASCIIコードについては, 159ページのキャラクタコード表を参照してください.

例
```
PRINT CHR$(66)
B
```

解説 通常CHR$は特別な文字を出力するために使います.

⇨ 68ページ参照

CLEAR〔式1〔, 式2〕〕 コマンド

機能 変数の初期化とメモリの領域の設定をします.

例
```
CLEAR 500,61440
```

解説 式1により文字領域の大きさを設定します. 電源投入直後は300に設定されます.

式2によりBASICが使うことのできるメモリの上限を設定します.

⇨ 93ページ参照

DATA 定数〔,定数……〕 文

機能 READ文により読まれる数値および文字定数を格納します.

例
```
100 DATA 10,40,AB
```

解説 DATA文は非実行文でプログラム中のどこに置いてもかまいません. 定数は任意の数値, すなわち固定小数点, 浮動小数点, 整数のいずれか, または文字列です.

文字定数は, それがコンマ, コロン, または前後に意味のある空白を含む場合にのみ引用符 (") で区切ってください. それ以外の場合には, 引用符で区切る必要は特にありません.

READ文によって指定される変数の型は対応するDATA文の定数と一致しなくてはなりません.

⇨ 58ページ参照

DATE$ 関数

機能 内蔵クロックの示す日付けを与えます.

例
```
PRINT DATE$
82/11/10
```

解説 内蔵されているクロックは日付けと時刻を保持しています. 日付けは次のような形の文字列として保持されます.

YY/MM/DD

YYが年, MMが月, DDが日を表わす2桁の数字で, 次のようにセットします.

```
DATE$="82/11/10"
```

⇨ 102ページ参照

CONT　コマンド

機能　プログラムの実行を再開します.

解説　ブレークが発生した時点から実行が再開されます. このコマンドはプログラム中では使用できません.

⇨ 92 ページ参照

COS（式）　関数

機能　式の値のコサイン値(cos x)を与えます.

例

```
10 X=2*COS(0.4)
20 PRINT X
RUN
 1.84212
```

解説　式の値の単位はラジアンです.

CSAVE "ファイル名"　コマンド

機能　メモリ内にあるプログラムをカセットテープ上にセーブします.

例

```
CSAVE "TIMER"
```

解説　ファイル名は6文字以上あってもかまいませんが, 最初の6文字のみがファイル名として使われます.

⇨ 95ページ参照

DEFUSR〔数字〕＝開始番地　文

機能　機械語のサブルーチンの**開始番地**を指定します.

例
```
200 DEFUSR0=24000
210 X=USR0(Y^2/2.89)
```

解説　**数字**は0～9の任意の数で，番地を指定されたUSRルーチンの番号に対応します.

開始番地はUSRルーチンの**開始番地**です．**数字**を省略した場合は0とみなされます.（機械語のサブルーチンを作る場合は，μCOM-87LCユーザーズマニュアル，アプリケーションノートを参照してください.）

DELETE {行番号〔－〔行番号〕〕 / －行番号}　コマンド

機能　プログラムの行を消去します.

例

`DELETE 40`	40行を消去します.
`DELETE 40-100`	40行から100行までを消去します.
`DELETE -40`	最初の行から40行までを消去します.
`DELETE 40-`	40行以降の行をすべて消去します.

解説　**行番号**がないか，あるいは該当する行がなければエラーになります.

⇨ 94ページ参照

DIM 変数名（添字の最大値〔，添字の最大値…〕）　文

機能　配列変数を定義します.

DEF型 文字範囲　文

機 能　変数の型を整数，単精度または文字に指定します．

例
```
10 DEFSTR A-C,S-Z
```

解 説　型はINT，SNG，STRのいずれかで，それぞれ整数，単精度，文字型を示します．

あらたに出てくる変数名の最初の文字が**文字範囲**で指定された文字で始まっている場合，その変数の型をDEF**型**で示すものに指定します．

⇨104ページ参照

DEFFN 関数名〔(引数〔,引数〕…)〕＝関数定義式　文

機 能　ユーザー関数を定義し，名前をつけます．

例
```
100 DEFFNB(X,Y)=X^3/Y^2
120 T=FNB(I,J)
```

解 説　**関数名**は変数名の規則に従いつけます．**引数**は，その関数が呼ばれたときに置き換えられる関数の定義式中の変数に対応します．**関数定義式**は1行に限られます．

関数は，文字列関数でも数値関数でもよく，関数名の型で決まります．この関数を他の文で使うときは，関数名の先頭にFNをつけ（ ）で囲んで**引数**の並びに対応するように式を書きます．

DEFFN文は，それが定義する関数が呼ばれる前に実行されなければなりません．

⇨ 50ページ参照

ERR/ERL　関数

機能　エラー処理ルーチンで，エラーの番号と発生行番号を与えます．

例
```
100 ERR=5 THEN ....
200 ERL=50 THEN ....
```

解説　エラー処理サブルーチンにはいったとき，ERR はエラーの番号を，ERLはエラーの起きた行番号を与えます．ERRおよびERLは通常IF…THEN文で使われ，エラー処理ルーチンの処理の流れを制御します．

EXP（式）　関数

機能　指数関数値を与えます．

例
```
10 A=5
20 PRINT EXP(A-1)
RUN
 54.5982
```

解説　*e*を基数とした指数関数値を与えますが，式の値は 87.3366 未満でなければなりません。もし，EXP 関数がオーバーフローした場合は，エラーになります．

FIX（式）　関数

機能　式の値の整数部分を与えます．

例　100 DIM A(20)

解説　DIM 文なしに**配列変数名**が使われた場合には，**添字の最大値**は10とみなされます．

添字の最小値は0となりますが，OPTION BASE 文により**1**とすることもできます．⇨ 55ページ参照

END　文

機能　プログラムの実行を終了します．

例　100 END

解説　END文を実行するとコマンドレベルに戻ります．また，プログラム中のどこに置いてもかまいません．⇨ 24ページ参照

ERASE 配列変数名〔，配列変数名……〕　文

機能　プログラムから配列を消去します．

例　450 ERASE A,B
　　460 DIM B(99)

解説　ERASE 文によって消去した後，再び同じ配列を宣言したり，使用されていたメモリスペースを他に転用することができます．⇨ 93ページ参照

域，引数が**文字列**の場合は文字領域の未使用バイト数を与えます．

GOSUB 行番号　文

機 能　サブルーチンへ分岐します．

例　100 GOSUB 40

解 説　**行番号**はサブルーチンの最初の行の**行番号**です．一つのサブルーチンが，プログラムの中で何度呼ばれてもかまいませんし，サブルーチンがサブルーチンを呼ぶことも許されます．

⇨ 47ページ参照

GOTO 行番号　文

機 能　指定された行へ分岐します．

例　100 GOTO 10

解 説　**行番号**の行の文が実行文であれば，その文および引き続く文が実行されます．もし，非実行文のときは，**行番号**の行の後の最初の実行文より実行されます．

⇨ 32ページ参照

HEX$（式）　関数

機 能　**式**の値を16進数に変換した文字列として与えます．

例
```
PRINT FIX(58.75)
 58
PRINT FIX(-58.75)
-58
```

解説　同じような機能を持つ関数に INT があります．FIX とINT の主な違いは，Xが負のとき FIX は式の値より大きい整数を，INTは式の値より小さい整数を与えることです．

FOR 変数名＝式1 TO 式2〔STEP　式3〕　文

機能　一連の文を指定回数繰り返し実行します．

例
```
10 FOR I=1 TO K STEP 2
     ⋮
50 NEXT I
```

解説　FOR 文は NEXT 文と対にして使用します．変数名で示す変数を制御変数と呼びます．**式**$_1$はカウンタの初期値，**式**$_2$は終値を示します．また**式**$_3$により，1回のループに対する増分を指定します．STEP以後を省略すると増分は1となります．
⇨ 40ページ参照

FRE(式 / 文字列)　関数

機能　BASIC 領域，文字領域の未使用バイト数を与えます．

例
```
PRINT FRE(0)
 14542
```

解説　FRE 関数に渡す引数はダミーです．引数が**式**の場合は BASIC 領

INKEY$ 関数

機能 キーボードが押されていれば，その文字を与えます．

例
```
10 IF INKEY$="" THEN 10
```

解説 なにも押されていない場合はヌルストリングを与えます．

⇨ 81ページ参照

INPUT〔文字列;〕変数名〔，変数名……〕 文

機能 プログラム実行時に，キーボードから入力することを可能にします．

例
```
10 INPUT "A=";A
```

解説 INPUT文が実行されると，プログラムの実行は停止しプログラムはデータの入力を待っていることを示す疑問符が表示されます．もし，**文字列**がある場合には疑問符の前にその文が表示されます．

入力される各データは，**変数名**で指定した型に一致していなくてはなりません．キー入力バッファサイズは，255バイトです．

⇨ 16, 28ページ参照

INPUT#－1,変数名〔，変数名……〕 文

機能 データをカセットテープから入力します．

例
```
100 INPUT #-1,A$,X
```

解説 指定された変数にカセットテープからデータを入力します．それぞれの変数の並びは，PRINT#－1文で指定したものと，個数，型が一致していなければなりません．入力バッファサイズは，255バイトです．

⇨ 98ページ参照

例

```
10 INPUT X
20 A$=HEX$(X)
30 PRINT X;" IS";A$
RUN
? 32
 32 IS 20
```

解説　式の値は小数部分を切り捨てて整数に変換してから，HEX$の値を求めます．

⇨ 74ページ参照

IF 論理式 THEN { 文 / 行番号 } 〔**ELSE** { 文 / 行番号 }〕
または
IF 論理式 GOTO 行番号　　文

機能　論理式の真偽によって，プログラムの実行の流れに関する判断を行ないます．

例

```
100 IF A=B THEN PRINT"EQUAL"
      ELSE PRINT"NOT EQUAL"
```

解説　論理式の結果が真ならば，THEN 以下の文を実行します．THEN 以下には分岐する行の行番号または実行する文のどちらでも書くことができます．

もし論理式が偽ならば，THEN 以下の文は無視されます．そして ELSE 文があれば，それが実行されますが，ない場合は次の行へ制御が移ります．

⇨ 31，34ページ参照

KEY LIST　文

機能　ファンクションキーの内容を表示します.

解説　KEY LIST は，ファンクションキーの内容を全て表示します.

⇨103ページ参照

LEFT$（文字列，式）　関数

機能　**文字列**の左から，**式**で示す文字数分の**文字列**を与えます.

例

```
10 A$="CMT BASIC"
20 B$=LEFT$(A$,3)
30 PRINT B$
RUN
CMT
```

解説　**式**の値は0～255の範囲でなければなりません．**式**の値が文字列長よりも大きい場合は，**文字列**全体を与えます．もし，**式**の値が0の場合はヌルストリングを与えます.

⇨ 66ページ参照

LEN（文字列）　関数

INPUT% 1，変数名〔，変数名……〕 文

機 能 データをRS-232C ポートから入力します.

例
```
100 INPUT %1,B$,Y
```

解説 それぞれの変数の並びの個数，データの型はRS-232Cポートから入力されるデータの型と一致していなければなりません．ポート入力バッファサイズは，255バイトです.

⇨106ページ参照

INT（式） 関数

機 能 式の値を超えない最大の整数を与えます.

例
```
PRINT INT(99.89)
 99
PRINT INT(-12.11)
-13
```

解 説 式の値が正のときはFIX関数と同じ働きをしますが，負のときは違う働きをします．FIX関数の項も参照してください.

⇨ 80ページ参照

KEY キー番号，文字列 文

機 能 プログラマブル・ファンクションキーを定義します.

例
```
KEY 5,"RETURN"+CHR$(13)
```

解 説 ファンクションキーはシフトモードを加えると合計10個の**文字列**を記憶できます．各ファンクションキーは最大15文字までの**文字列**を定義できます．キーボードから入力できない文字は CHR $ 関数を（+）でつないで使います．**キー番号**は1～10までで，6～10はシフトモードです.

⇨103ページ参照

機 能 メモリ内のプログラムの全部または一部をプリンタに出力します.

例

```
LLIST            プログラム全体を出力します.
LLIST 500        500行のみを出力します.
LLIST 150-       150行から最後の行までを出力します.
LLIST -300       最も小さい行から300行までを出力します.
LLIST 150-300    150行から300行までを出力します.
```

⇨ 98ページ参照

LOCATE 式 文

機 能 ディスプレイ上の任意の位置にカーソルを移動します.

例

```
20 LOCATE 10
```

解 説 式は0～79の値を指定できますが, CONSOLE指定によって次の表ように式の値の範囲が変わります.

⇨ 77ページ参照

式の値	CONSOLE指定
0～31	×,1,1
0～39	×,0,1
0～63	×,1,0
0～79	×,0,0

注) ×は0または1

LOCK/UNLOCK コマンド

機 能 プログラムの変更に対してロックをかけ(LOCK), またはロックをはずします(UNLOCK).

解 説 LOCKを実行すると, プログラム編集, NEW, CLEAR, ERASE, CLOAD, DELETEコマンドは無効となりエラーがでます. また, UNLOCKを実行すると, LOCKで無効となったコマンドが全て有効となりプログラムの編集が可能になります.

⇨ 93ページ参照

機 能　文字列の文字長を与えます.

例
```
10 X$="ABC DEF"
20 PRINT LEN(X$)
RUN
 7
```

解 説　プリントされない文字および空白も数えます.　⇨ 67ページ参照

〔LET〕変数名＝式　文

機 能　式で表わされる値を変数に代入します.

例　100 LET D=12

解 説　キーワードLETはなくてもかまいません. つまり式の値を変数に代入するには等号だけでよいのです.

LIST〔行番号〕　コマンド

機 能　メモリ内にあるプログラムを表示します.

例　LIST 100

解 説　**行番号**により指定された行を表示します. **行番号**が省略された場合は, 最も小さい**行番号**の行を表示します.　⇨ 89ページ参照

LLIST { 〔行番号〕〔－行番号〕 / 行番号－ }　コマンド

MID$（文字列，式1〔，式2〕）　関数

機能　**文字列**の中から任意の長さの**文字列**を与えます．

例

```
10 A$="ABCDEFG"
20 PRINT MID$(A$,3,4)
RUN
CDEF
```

解説　**文字列**の初めから**式1**文字目に始まって**式2**文字の長さの**文字列**を与えます．

式1と**式2**の範囲は0～255でなければなりません．また，**式2**を省略した場合や**文字列**の**式1**番目の文字から右の文字数が**式2**より小さい場合は，**文字列**の**式1**番目より右全ての**文字列**を与えます．**文字列**の文字数が**式1**より小さければヌルストリングを与えます．

⇨ 66ページ参照

MOTOR〔スイッチ〕　文

機能　カセットテープレコーダのモーターのON，OFFを制御します．

例

```
MOTOR 1
```

解説　**スイッチ**を0にすればモーターはOFFとなり，0以外の値にすればONとなります．また，**スイッチ**を指定しない場合，モーターがONならばOFFに，OFFならばONにします．

NEW　コマンド

LOG（式）　関数

機能　式の値の自然対数（$\log_e x$）を与えます．

例
```
PRINT LOG(45/7)
 1.86075
```

解説　式の値は0より大きくなければなりません．

LPRINT〔式〔{ , / ; / 空白 }〔式〕〕…〕　文

機能　プリンタにデータを印字します．

例
```
LPRINT A,B,C
```

解説　出力がプリンタに行くことを除くと，PRINT文と同じです．PRINT文の項を参照してください．ただし，実行は継続します．　⇨ 99ページ参照

LPRINT USING 文字列；式〔；式〕　文

機能　プリンタにデータを書式に従い出力します．

例
```
10 LPRINT USING "####";A;B
```

解説　出力がプリンタに行くことを除くと，PRINT USING文と同じです．PRINT USING文の項を参照してください．ただし，実行は継続します．

⇨ 99ページ参照

ON 式 GOSUB 行番号〔,行番号…〕 文

機能 式の値により，いくつかのサブルーチンの内の一つに分岐します．

例
```
ON L GOSUB 200,300,400,500
```

解説 各行番号はサブルーチンの最初の行の**行番号**です．分岐の方法は，ON GOTO文と同じです．
⇨100ページ参照

ON 式 GOTO 行番号〔,行番号…〕 文

機能 式の値により，いくつかの行の一つに分岐します．

例
```
100 ON L GOTO 150,200,250
```

解説 式の値がリストのどの**行番号**の文に分岐するかを決定します．たとえば，式の値が3であれば，**行番号**の内3番目のものに分岐します．

式の値が0または**行番号**の組の個数より大きい場合には，次の行へ制御が移ります．
⇨100ページ参照

ON ERROR GOTO 行番号 文

機能 エラートラップを可能にし，エラー処理ルーチンの最初の行を指定します．

例
```
10 ON ERROR GOTO 1000
```

解説 エラートラップ機能が可能になると，エラーが検出されたときに指定されたエラー処理ルーチンへ制御が移ります．エラートラップ機能を禁止するにはON ERROR GOTO 0を実行してください．

機 能　メモリ内のプログラムを消去し，全ての変数を消去します．

解 説　NEWコマンドは，コマンドレベルにあるときに新しいプログラムを入力するまえに実行します．

⇨ 93ページ参照

NEXT〔変数名〔，変数名…〕〕　文

機 能　FOR文による繰り返しの終端を示します．

例
```
30 FOR I=1 TO N
  :
100 NEXT I
```

解 説　**変数名**で示す変数は，直前にあるFOR文の制御変数と対になっている必要があります．

変数名が省略された場合，NEXT文は直前のFOR文に対応します．

⇨ 40ページ参照

OCT$（式）　関数

機 能　**式**の値を8進数に変換した文字列として与えます．

例
```
PRINT OCT$(24)
 30
```

解 説　**式**の値の小数部分を切り捨てて整数に変換してから，OCT$の値を求めます．

例　A=PEEK(&HFCAE)

解説　式の値は0～65535までの範囲になければなりませんが，負の場合には式＋65536とみなします．

結果は0～255までの値です．

⇨ 74ページ参照

POKE 式1，式2　文

機能　メモリにデータを書き込みます．

例　10 POKE &HFCAE,&HFF

解説　式1はデータを書き込むメモリアドレスで，0～65535(16進数で0～&HFFFF)の整数値です．

式2は書き込むデータで，0～255(16進数で0～&ＨＦＦ)の整数値です．不容易にPOKE文を使うとプログラム内容を破壊することがありますから注意してください．

⇨ 74ページ参照

PRINT〔式〔{，／；／空白}〔式〕〕…〕　文

機能　画面に情報を出力します．

例　20 PRINT X,A$,X^5

解説　式，コンマ，セミコロンを省略した場合は，単に停止します．式が含まれるときは，画面にそれらの式の値が表示されます．

それらの項目の表示される位置は，各項目の式のリスト内の区切り方により決められます．BASICは，1行を14文字ずつの領域に分けます．もし，式をコ

OPTION BASE { 0 / 1 }　文

機能　配列の添字の最小値を宣言します.

例
```
10 OPTION BASE 1
20 DIM A(10)
```

解説　OPTION BASE 1が実行されたとき，配列の添字の最初は1となります．OPTION BASE文がないときまたは0を指定したときは，配列の添字の最小値は0です.

この文は全てのDIM文より前に実行されなければなりません.

⇨ 56ページ参照

PAUSE〔USING 文字列；〕式〔；式…〕　文

機能　式のリストをWAIT文により示された時間だけ表示します.

例
```
10 WAIT 10
20 PAUSE "ABC"
```

解説　WAIT文により指定された式÷10秒間，式のリストを表示します．PAUSE文ではUSING節により，式のリストの書式を指定することができます.

USING節はPRINT USINGの項を参照してください.　⇨ 77ページ参照

PEEK（式）　関数

機能　式の値で示すメモリアドレスの内容を与えます.

.（小数点）…… 領域の任意の場所に小数点を挿入します．フォーマット文の指定に小数点に続く桁がある場合，それらの桁は必ず出力されます．

＋……フォーマット文の最初，または最後のプラス記号は，数値の前または後ろにその数値の符合をつけます．

－……フォーマット文の最後にマイナス記号を書くと，負の数値の後ろにマイナスが表示されます．

＊＊……フォーマット文の最初に書くことにより，数値領域の初めの空白部分にアスタリスク(＊)を出力します．また＊＊は，さらに2桁分の領域を確保します．

￥￥……フォーマット文の最初に書くことにより，出力される数値の直前に円記号を出力します．￥￥は数字2桁分の位置を確保しますが，円記号にはそのうち1桁を使います．指数型式のときは￥￥を使用することはできません．

＊＊￥……フォーマット文の最初に書くことにより，＊＊と￥￥の両方の機能を果たします．上記条件は全て適用されますが，異なる点は＊＊￥は数字3桁分の位置を確保し，そのうちの1個は円記号となります．

，（コンマ）……フォーマット文の小数点の左側にコンマがある場合は，小数点の左側の数値の3桁ごとにコンマが出力されます．指数形式とともに使われた場合は無効となります．

＾＾＾＾……桁指定文字＃の後方に4つの上向き矢印を置くと，指数形式の書式を指定します．4つの上向き矢印は，E＋*nn*が出力される領域を確保します．

もし，指定された数値領域の長さよりも出力される数値の桁数の方が長い場合には，数値の前にパーセント記号（％）が出力されます．

ンマで区切ると，次の値はその次の領域の先頭から表示します．またセミコロンを使うと，次の値は最後の値のすぐ後から表示されます．

もし，式の最後がコンマ，またはセミコロンで終わっている場合は，次のPRINT文による表示は上に述べた間隔で始まります．式のリストの最後がコンマまたはセミコロンのいずれでも終わっていない場合には，RETURN キーが押されるまで停止します．もし，画面から表示があふれた場合には，▶ キーを使ってあふれた分を表示することができます．

⇨ 19ページ参照

PRINT USING 文字列；式〔；式〕 文

機 能　式を指定した書式で出力します．

例　
```
40 PRINT USING"&  &";A$;B$
```

解 説　式は，セミコロンで区切ります．文字列はフォーマット文で次に示すようなフォーマット文字で構成されます．これらのフォーマット文字により，出力される式の書式が決められます．

■文字領域　！……式として与えられた文字列の最初の文字だけを出力するように指定します．

＆ｎ個の空白＆……式として与えられた文字列の内，２＋ｎ文字が出力されます．文字列が２＋ｎ文字より長い場合は，余分の文字は無視されます．また，２＋ｎ文字に満たない場合は，文字列は左づめに出力され，空いた右側には空白がつめられます．

■数値領域　＃……各桁の位置を示すのに使います．指定された桁は常に数字で満たされますが，指定した桁数より出力する数値の桁数の方が，小さい場合には，その領域に右づめで出力され，左側の空いた部分には空白がつめられます．

この文が実行されないとRND関数はいつも同じ乱数列を発生します．実行のたびに乱数列を変えたいときは，プログラムの先頭にRANDOMIZE文を置き，実行ごとに式の値を変えるか，入力します．

⇨ 79ページ参照

READ 変数名〔，変数名…〕 文

機 能 DATA 文より値を読み，変数に割り当てます．

例

```
100 READ A,B$,C$
```

解 説 READ 文はいつでも DATA 文と組み合わせて使います．READ 文はDATA文のデータを1対1対応の方法で変数に割り当てていきます．

READ 文の変数は数値変数でも文字変数でもかまいませんが，読み出された値の型とは一致しなければなりません．

⇨ 58ページ参照

REM 注釈 文

機 能 プログラムに注釈(コメント)を入れます．

例

```
10 REM TEST PROGRAM
```

解 説 REM 文は実行はされませんが，プログラムリストをとると入力した内容がそのままリストされます．キーワードREMの代わりにアポストロフィ(')を使うことができます．

⇨101ページ参照

フォーマット文の前または後ろにフォーマット文字以外を書いた場合には，そのキャラクタが表示されます。 ⇨ 82ページ参照

PRINT#−1，式，〔，式…〕 文

機能 データをカセットテープに出力します。

例
```
110 PRINT#-1,A%,B$
```

解説 INPUT#−1文も参照してください。 ⇨ 98ページ参照

PRINT% 1，式〔，式…〕〔；〕 文

機能 RS-232Cポートにデータを出力します。

例
```
10 PRINT%1,A$,B$
```

解説 文の最後にセミコロンをつけない場合には，CRコードを付加し，セミコロンをつけた場合は，式の並びだけを出力します。INPUT%文も参照してください。 106ページ参照

RANDOMIZE〔式〕 文

機能 乱数列を新しく発生させます。

例
```
10 RANDOMIZE 1000
```

解説 式が省略された場合は，実行が中断されます。(0—65529)？が表示され，入力待ちとなります。

再開します.

NEXT をつけるとエラー原因となった文のすぐ次の文から実行を再開します.

行番号を指定するとその行から実行を再開します.

RETURN 文

機 能 サブルーチンから復帰します.

例
```
100 RETURN
```

解 説 サブルーチンの RETURN 文を実行すると, BASICは直前に実行された GOSUB 文の次の文へ戻ります. 一つのサブルーチンに複数の RETURN 文があってもかまいません.

⇨ 47ページ参照

RIGHT$（文字列，式） 関数

機 能 **文字列**の右から**式**の値が示す文字数分の**文字列**を与えます.

例
```
10 A$="??BASIC"
20 PRINT RIGHT$(A$,5)
RUN
BASIC
```

解 説 **式**の値が文字列長より大きい場合には**文字列**全体を与え，**式**の値が0の場合はヌルストリングを与えます.

⇨ 66ページ参照

RENUM〔〔新番号〕〔,〔旧番号〕〔,増分〕〕〕　コマンド

機能　プログラムの行番号をつけなおします.

例　RENUM 100,10,50

解説　**新番号**は，新しくつける行番号の最初の番号で，デフォルト値は10です．**旧番号**は番号のつけ替えをはじめる，現在のプログラムの行番号です．デフォルト値はプログラムの最初の番号です．**増分**は新しく付ける行番号の増分で，デフォルト値は10です．

⇨ 91ページ参照

RESTORE〔行番号〕　文

機能　DATA 文を初め（または，指定した部分）から読めるようにします.

例　100 RESOTRE 50

解説　**行番号**によって，指定した行の DATA 文から READ 文は読み始めます．**行番号**を省略すると READ 文はプログラムの最初の DATA 文から読み始めます.

⇨ 58ページ参照

RESUME { 〔0〕 / NEXT / 行番号 }　文

機能　エラー回復処理終了後，プログラムの実行を再開します.

例　1000 RESUME 100

解説　RESUME，RESUME 0とするとエラー原因となった文から実行を

SIN（式）　関数

機能　式の値のサイン値(sin$x$)を与えます.

例
```
10 X=2*SIN(0.8)
20 PRINT X
RUN
 1.43471
```

解説　式の値の単位はラジアンです.

SQR（式）　関数

機能　式の値の平方根を与えます.

例
```
PRINT SQR(X)
 3.16228
```

解説　式の値が負であってはなりません.

⇨ 47ページ参照

STOP　文

機能　プログラムの実行を中断します.

例
```
30 STOP
```

解説　STOP文はプログラムのどこで使ってもかまいません. STOP文を実行すると BREAK IN nnnnn というメッセージが表示されます.

また, CONTコマンドによりプログラムの実行を再開します.

RND〔(式)〕 関数

機 能 0と1の間の乱数を与えます.

例
```
20 PRINT INT(RND(1)*100)
```

解 説 式が負の場合は，新しい乱数の系列を作ります．式が正なら，同じ乱数の系列の次の乱数を与えます．式が0なら1つ前に発生した乱数を与えます．普通は正の数を使います．(式)を省略したときは式＝1と設定します.

⇨ 79ページ参照

RUN〔行番号〕 コマンド

機 能 メモリ内のプログラムを実行します.

解 説 **行番号**を指定すると，その行から実行します．指定のない場合には，最も小さい**行番号**の行から実行します.

⇨ 89ページ参照

SGN（式） 関数

機 能 **式**の値が正の場合は1を，**式**の値が0の場合は0を，**式**の値が負の場合は－1を与えます.

例
```
ON SGN(X)+2 GOTO 100,200,300
```

解 説 この例では，Xが負なら100行へ，0なら200行へ，正なら300行へ分岐します.

解 説　式の値の単位はラジアンです．TAN関数がオーバーフローした場合はエラーになります．

TERM コード，パリティ，ボーレート　コマンド

機 能　INPUT%1，PRINT%1におけるデータ転送を規定します．

例　TERM J,1,2

解 説　**コード**はASCIIコードとJISコードとのスイッチになり，ASCIIコードの場合はAまたはa，JISコードの場合はJまたはjで指定します．

パリティは0～2で指定し，0ではnonパリティ，1ではoddパリティ，2ではevenパリティモードとなります．

ボーレートは0～5で指定し，それぞれ110，150，300，600，1200，2400ボーを示します．

⇨106ページ参照

TIME$　関数

機 能　内蔵クロックの時刻を与えます．

例
```
PRINT TIME$
15:20:36
```

解 説　時刻は次のような形の文字列となります．

HH：MM：SS

HHは時，MMは分，SSは秒を2桁で表します．

TIME$は次のようにセットすることができます．

92ページ参照

STR$（式）　関数

機能　式の値を表わす文字列を与えます.

例
```
PRINT STR$(15+3)+"SEC"
 18SEC
```

解説　VAL関数の項も参照してください.　68ページ参照

TAB（式）　関数

機能　式で示す値の位置まで画面またはプリンタに空白を表示します.

例
```
PRINT TAB(5);"ABC"
     ABC
```

解説　式の値の範囲は0～255です.現在の表示位置が式の値を超えていれば，次の行の式の値が示す位置まで空白を表示します．また，TABはPRINTまたはLPRINT文中でしか使えません.

TAN（式）　関数

機能　式の値のタンジェント値（tan$x$）を与えます.

例
```
PRINT TAN(1.5)
 14.101
```

機 能　文字列を数値に変換して与えます.

例
```
PRINT VAL("12.3")
 12.3
```

解 説　文字列の最初の文字が＋，－，&，または数字以外であれば，VALの値は0となります.

⇨ 68ページ参照

VARPTR（変数名）　関数

機 能　変数名で指定されるデータの最初のバイトのメモリ番地を与えます.

例
```
10 B=VARPTR(A)
```

解 説　VARPTRを使う前に，その変数に値が代入されていなければなりません.

VARPTRの値は－32768～32767の値をとりますが，負の値の場合に実際のメモリ番地は65536を加えた値となります.

変数名には，任意の型を選ぶことができます.

WAIT 式　文

機 能　PAUSE文の表示する時間の設定をします.

例
```
100 WAIT A*2
110 PAUSE B$
```

解 説　PAUSE文は，WAIT文で指定された式÷10秒間の表示を行ないます.

⇨ 77ページ参照

```
TIME$="12:34:56"
34
56"
```

⇨102ページ参照

TRON/TROFF　文

機能　プログラムの実行の状態を追跡します．

解説　プログラムのデバックの助けとして，TRON文を実行するとトレースフラグが立ち，実行するプログラムの行番号をプリントします．それぞれの行番号はカギカッコに囲んで表示します．トレースフラグはTROFF文（またはNEWコマンド)を実行するとリセットされます．

USR〔番号〕(式)　関数

機能　式の値を持ってユーザーの機械語ルーチンを呼び出します．

例
```
40 B=T*SIN(Y)
50 C=USR0(B/2)
60 D=USR1(B/3)
```

解説　番号は0～9の整数で，DEFUSR文で定義した番号に対応します．番号を省略した場合は0とみなします．

VAL（文字列）　関数

3. 付　録

13	代入文などで式の左右の型が一致していない(数値と文字など).
14	CLEARなどで指定した文字変数用メモリエリアが足りなくなった.
15	ストリング(引用符で囲まれた文字列) が255文字を超えている.
16	文字式が複雑すぎる(カッコ内のネスティングレベルが多すぎるなど).
17	CONT コマンドを入力しても続行できない(ポインタがすでに破壊されている).
18	参照されたユーザー関数の定義(DEF文による)がなされていない.
19	エラー処理後, プログラムの実行を再開することができない.
20	エラーとRESUME が対応していない(エラーがないのにRESUME した).
21	メッセージの定義されていないエラー.
22	ステートメント中に必要とされるパラメータなどが定義されていない.
23	BASICプログラムを入力する際に1行の長さが有効範囲を超えている.

1. エラーメッセージ表

エラー番号	意味
1	FOR～NEXTが正しく対応していない(NEXTが多すぎる).
2	文法が間違っている. プログラム中に規定のステートメント以外のものがある.
3	GOSUB～RETURN が正しく対応していない(RETURN 文だけがある).
4	READ 文で読まれるべきデータが DATA 文の中に用意されていない.
5	ステートメントの機能の呼び方が間違っている.
6	入力された数値や演算結果が許容される範囲をはずれている.
7	メモリ内容が足りなくなった（プログラムが長すぎる, 配列が大きすぎるなど).
8	必要とされるプログラム行（GOTO の飛び先など）が定義されていない.
9	配列変数の添字が規定の範囲内にない.
10	同じ配列を再定義している.
11	0 による除算が実行された.
12	ダイレクト・ステートメントとして使えないコマンドが入力された.

2. キャラクタコード表

上位／下位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			空白	0	@	P		p				―	タ	ミ	U0	
1			!	1	A	Q	a	q			。	ア	チ	ム	U1	
2			"	2	B	R	b	r			「	イ	ツ	メ	U2	
3			#	3	C	S	c	s			」	ウ	テ	モ	U3	
4			$	4	D	T	d	t			、	エ	ト	ヤ	U4	
5			%	5	E	U	e	u			・	オ	ナ	ユ	U5	
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ	U6	
7			'	7	G	W	g	w			ァ	キ	ヌ	ラ	U7	
8			(	8	H	X	h	x			ィ	ク	ネ	リ	U8	
9			)	9	I	Y	i	y			ゥ	ケ	ノ	ル	U9	
A			*	:	J	Z	j	z			ェ	コ	ハ	レ		
B			+	;	K	[	k				ォ	サ	ヒ	ロ		
C	CL		,	<	L	¥	l				ャ	シ	フ	ワ		
D	CR		−	=	M	]	m				ュ	ス	ヘ	ン		
E			.	>	N	^	n				ョ	セ	ホ	゛		
F			/	?	O	_	o				ッ	ソ	マ	゜		

注）U0からU9はユーザーの定義キャラクタ

0CHにあるCLは画面消去

0DHにあるCRは RETURN の意味

24	カセットテープレコーダからの入力が正しく行なわれない(テープの読み誤りなど).
25	ファイル上にあるデータの形式が間違っている.
26	ロックされているときにプログラムの変更を行なおうとした.
27	コミュニケーションバッファがオーバーフローした.
28	オーバーランエラーが起きた.
29	パリティエラーが起きた.
30	ストップビットがなかった.

4. ステートメント機能対応表

○……互換性あり　×……互換性なし　▲……修正すれば使用可

キーワード	PC-2001	PC-8001	キーワード	PC-2001	PC-8001
ABS	○	○	HEX$	○	○
AND	○	○	IF	○	○
APPEND	○	×	INKEY$	○	○
ASC	○	○	INPUT	○	○
ATN	○	○	INT	○	○
BEEP	○	▲	KEY	○	○
CHAIN	○	×	LEFT$	○	○
CHR$	○	○	LEN	○	○
CLEAR	○	○	LET	○	○
CLOAD	○	○	LIST	○	▲
CONSOLE	○	▲	LLIST	○	○
CONT	○	○	LOCATE	○	▲
COS	○	○	LOCK	○	×
CSAVE	○	○	LOG	○	○
DATA	○	○	LPRINT	○	○
DATE$	○	○	MID$	○	▲
DEFFN	○	○	MOD	○	○
DEFINT	○	○	MOTOR	○	○
DEFSNG	○	○	NEW	○	○
DEFSTR	○	○	NEXT	○	○
DEFUSR	○	○	NOT	○	○
DELETE	○	▲	OCT$	○	○
DIM	○	○	ON	○	○
ELSE	○	○	OPTION BASE	○	×
END	○	○	OR	○	○
ERASE	○	○	PAUSE	○	×
ERL	○	○	PEEK	○	○
ERR	○	○	POKE	○	○
ERROR	○	○	PRINT	○	○
EXP	○	○	RANDOMIZE	○	×
FIX	○	○	READ	○	○
FOR	○	○	REM	○	○
FRE	○	○	RENUM	○	○
GOSUB	○	○	RESTORE	○	○
GOTO	○	○	RESUME	○	○

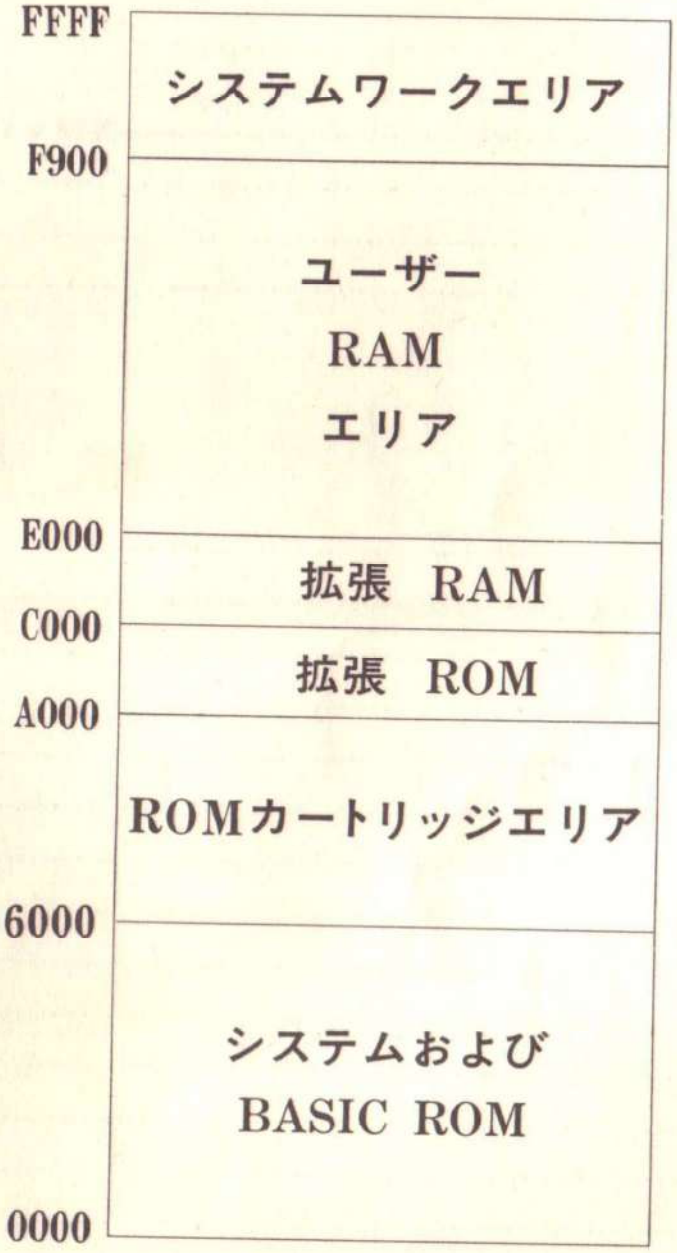
3. キーコード表

上位/下位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	CTL P	SP	0	@	P		p				ー	タ	ミ	U0	F1
1	CTL A	CTL Q	!	1	A	Q	a	q			。	ア	チ	ム	U1	F2
2	CTL B	CTL R	"	2	B	R	b	r			「	イ	ツ	メ	U2	F3
3	CTL C	CTL S	#	3	C	S	c	s			」	ウ	テ	モ	U3	F4
4	CTL D	CTL T	$	4	D	T	d	t			、	エ	ト	ヤ	U4	F5
5	CTL E	CTL U	%	5	E	U	e	u			・	オ	ナ	ユ	U5	F6
6	CTL F	CTL V	&	6	F	V	f	v			ヲ	カ	ニ	ヨ	U6	F7
7	CTL G	CTL W	'	7	G	W	g	w			ァ	キ	ヌ	ラ	U7	F8
8	CTL H	CTL X	(	8	H	X	h	x			ィ	ク	ネ	リ	U8	F9
9	CTL I	CTL Y	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	U9	F10
A	CTL J	CTL Z	*	:	J	Z	j	z			ェ	コ	ハ	レ		STOP
B	CTL K	CTL [	+	;	K	[	k				ォ	サ	ヒ	ロ		カナ
C	CTL L	▶	,	<	L	¥	l				ャ	シ	フ	ワ		CAP
D	CTL M	◀	−	=	M	]	m				ュ	ス	ヘ	ン		
E	CTL N	▲	.	>	N	^	n				ョ	セ	ホ	゛		
F	CTL O	▼	/	?	O	—	o	DEL			ッ	ソ	マ	゜		

■特殊なキーコード

CTL + ▶ ····· 01H
CTL + ◀ ····· 02H
CLR ····· 0CH
RETURN ····· 0DH
SHIFT + ▶ ····· 18H
SHIFT + ◀ ····· 19H
INS ····· 12H
▶ ····· 1CH
◀ ····· 1DH
▲ ····· 1EH
▼ ····· 1FH
DEL ····· 7FH
CAPS カナ ····· FBH
SHIFT + CAPS カナ ····· FCH
CTL + D ····· (04H)
デリートキャラクタ
CTL + E ····· (05H)
ラインイレーズ
CTL + P ····· (10H)
プリンタハードコピー

5. メモリマップ



ただし1000H～2000Hは空きエリアとなっています.

キーワード	PC-2001	PC-8001	キーワード	PC-2001	PC-8001
RETURN	○	○	UNLOCK	○	×
RIGHT$	○	○	USING	○	○
RND	○	○	USR	○	○
RUN	○	○	VAL	○	○
SGN	○	○	VARPTR	○	▲
SIN	○	○	WAIT	○	×
SQR	○	○	XOR	○	○
STEP	○	○	=	○	○
STOP	○	○	+	○	○
STR$	○	○	−	○	○
TAB	○	○	*	○	○
TAN	○	○	/	○	○
TERM	○	×	>	○	○
THEN	○	○	<	○	○
TIME$	○	○	^	○	○
TO	○	○	¥	○	○
TROFF	○	○		○	○
TRON	○	○			

INDEX

あ

か

な

は

さ

た

A

B

C

D

ま

や

ら

J

K

L

M

E

F

G

H

I

S

T

U

N

O

P

R