

# GOTO文でジャンプする

ゴーツー GOTO

## 使用例

### ① GOTO 10

行番号10にとんで、プログラムの実行を続ける。

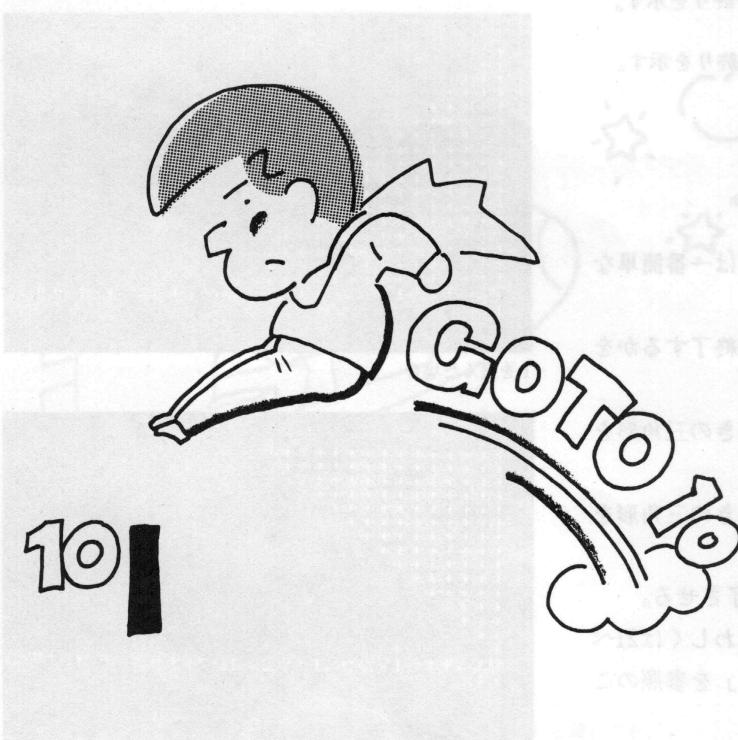
## 一般型式

**GOTO 行番号**

## 説明

●指定された行番号に無条件にとんで、プログラムの実行を続ける。

プログラムは行番号順に実行されるが、GOTO文はその流れを変える命令である。



①

```
LIST REM ** GOTO ノ サンフロ1 ***
10 PRINT "GUNDAM"
20 GOTO 50
30 PRINT "BASIC"
40 END
Ready
RUN
GUNDAM
Ready
```

②

```
LIST REM ** GOTO ノ サンフロ2 ***
10 REM **
20 INPUT K
30 INPUT K
40 PRINT "コウケイ="; K
50 GOTO 30
60 END
Ready
RUN
コウケイ= 574
コウケイ= 652
コウケイ= 1120
コウケイ= 1128
```

行番号30で入力した数字をどんどん加えて合計を画面に出す。行番号60のGOTO文で行番号にとばしているので、何回でも数字を入力できる。くり返しを止めるには、**SHIFT** + **STOP**キーを押さなければならない。

# 判断はおまかせ IF文

イフ ゼン  
IF…THEN

## 使用例

### ① IF A=0 THEN 50

Aが0の場合、行番号50にとぶ。  
0でない場合は次の行番号に進む。

### ② IF A=B THEN PRINT "ヒトシイ"

AとBが等しい場合、THENの後のPRINT文を実行する。等しくない場合は、PRINT文は実行しない。

一般型式 IF 条件文 THEN 命令文または行番号

## 説明

- 指定した条件にあれば、THENの後の命令文を実行するか、または指定した行番号へとばす。

条件にあわなければ、次の行番号へ進む。

- THENの後にIF…THEN文を書くこともできる。

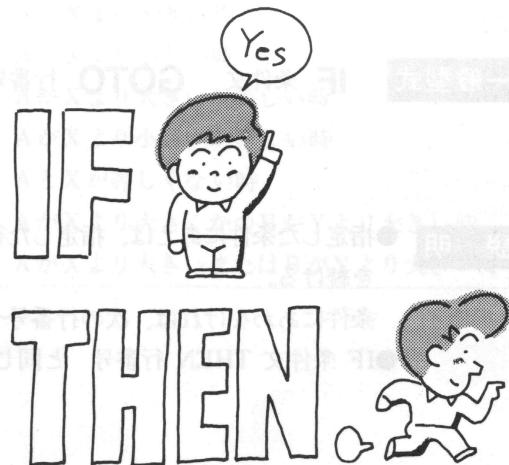
(例) IF A=B THEN IF B=C THEN PRINT  
A, B, C

これは、「A=B=Cの場合にA、B、Cをプリントする」ということだ。

- THENの後に複数の命令文を指定するときは、:で命令文をつなげばよい。

(例) IF A=B THEN PRINT "ヒトシイ":GOTO  
30

は、「AとBが等しい場合、"ヒトシイ"と画面に出し、行番号30にとぶ」ということだ。



```
① LIST
10 REM *** IF THEN ノサンプル1 ***
20 N=0
30 PRINT "*** GUNDAM ***"
40 N=N+1
50 IF N<10 THEN 30
60 END
Ready
RUN
*** GUNDAM ***
Ready
```

行番号50のIF文で、Nが10より小さい場合、行番号30のPRINT文にとぶように指定しているので、\*\*\*GUNDAM\*\*\*が、画面に10行出る。

```
② LIST
10 REM ** カズベヲアテル **
20 PRINT CHR$(6)カズベヲアテナサイ":N
30 INPUT "1カタ) カズベヲアテナサイ":N
40 IF N=6 THEN 100
50 IF N>6 THEN 80
60 PRINT "ティサキルヨ"
70 GOTO 30
80 PRINT "オオキスキルヨ"
90 GOTO 30
100 PRINT "アタリ!!!"
110 END
Ready
```

行番号40のIF…THENでは、入れた数字が6に等しい(アタリ)の場合、行番号100へとばし、行番号50のIF…THENでは、入れた数字が6より大きい場合、行番号80へとばしている。  
入れた数字が6より小さい場合は、行番号60へ進む。

注1. IF…THENのくり返しは、何回でも使えます。しかし、使い方によってはプログラムがわかりにくくなるので気をつけましょう。  
注2. :の使い方は、21ページの「:(コロン)で区切ってマルチステートメント」を見てください。

# 判断してジャンプする

イフ ゴー··· GOTO

## 使用例

**IF X=0 GOTO 90** Xが0ならば、行番号90にとんで実行を続ける。

一般型式 **IF 条件文 GOTO 行番号**

**説明** ●指定した条件にあれば、指定した行番号にとんで実行を続ける。

条件にあわなければ、次の行番号へ進む。

●IF 条件文 THEN 行番号と同じ意味をもつ。



```
① LIST
10 REM ** IF GOTO のサンプル1 ***
20 INPUT A$*
30 IF A$="END" GOTO 60
40 PRINT A$*
50 GOTO 20
60 END
Ready
```

```
RUN
GUNDAM
GUNDAM
SUPER PERSONAL COMPUTER
SUPER PERSONAL COMPUTER
? ■
```

行番号30のIF···GOTO文で、行番号20で入力された文字がENDなら、行番号60へとぼし、ENDでないなら行番号40へ進んで、入力された文字を画面に出す。

```
② LIST
10 REM ** カスノ コウケイ モトマル *
* 20 G=0
30 I=1
40 INPUT "カスノ モトマル"
50 G=G+I
60 I=I+1
70 IF I<=N GOTO 50
80 PRINT "1から"; N; "までの合計は"; G
90 END
Ready
```

```
カスノ モトマル
1から 10 の合計は 55
Ready
■
```

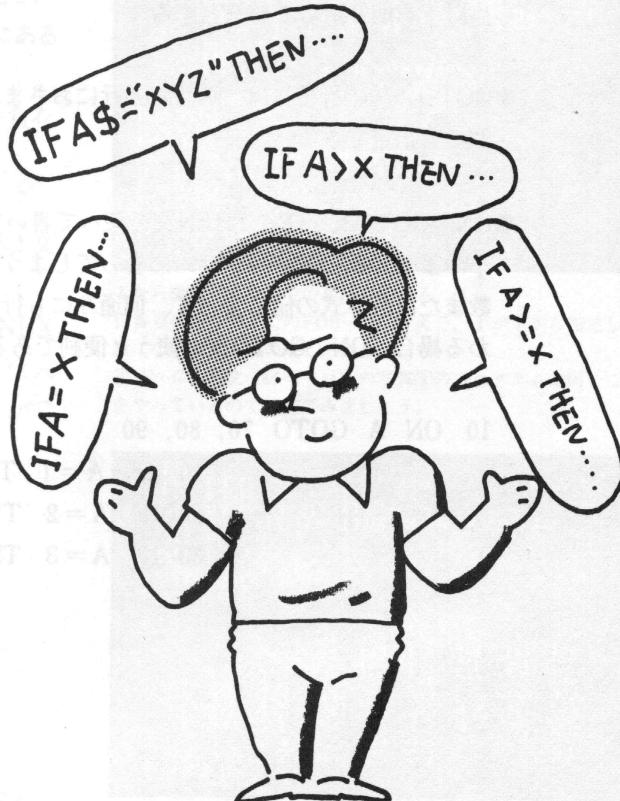
1から指定した数までの合計を求める。  
行番号70のIF···GOTO文で、指定した数をこえていないか判断し、こえていない場合は、足し込みを行なう。

# 条件文で何だろう

判断を行なうIF…THEN文やIF…GOTO文の、IFの後の条件文は、次のような記号を使って与えることができる。

条件文の記号	使 用 例	意 味
=	IF A = X THEN… IF A \$ = "XYZ" THEN…	AとXが等しい時 A\$がXYZの時
>	IF A > X THEN…	AがXより大きい時
<	IF A < X THEN…	AがXより小さい時
>=	IF A >= X THEN…	AがXより大きいか等しい時
<=	IF A <= X THEN…	AがXより小さいか等しい時
<>	IF A <> X THEN…	AとXが等しくない時
*	IF (A > X) * (B > Y) THEN…	AがXより大きくかつBがYより大きい時
+	IF (A > X) + (B > Y) THEN…	AがXより大きいまたはBがYより大きい時

- =は数字や文字・記号が等しいかどうかを
- >, <, >=, <=は、数字や文字・記号の大小関係を
- <>は、数字や文字・記号が等しくないかどうかを
- \*は、2つの条件が両方ともあってるかどうかを
- +は、2つの条件のうち、どちらかがあつてあるかどうかを調べる記号である。



④数字と同じように、文字や記号にも大小関係があります。この場合、アスキーコード表(158ページ)の10進数コードの大小で比較します。  
たとえば、アルファベットは、"A" < "B" < "C" < … のようになります。

# 豊富な行き先

オン ゴーツー  
ON…GOTO

## 使用例

### ON A GOTO 70, 80, 90

変数Aの値が1なら行番号70、2なら行番号80、3なら行番号90へとぶ。1、2、3以外なら次の行番号に進む。

## 一般型式

**ON** 数値変数または計算式

**GOTO** 行番号, 行番号, 行番号, .....

## 説明

●ONの後の数値変数または計算式の値によって、その値と同じ順番に指定した行番号にとぶ。

値と同じ順番に行番号の指定がない場合は次の行番号に進む。

●数値変数または計算式の値が整数でない場合は、小数点以下を切り捨てて整数とする。

●GOTOの後の行番号は、画面の3行におさまれば、何個でも指定できる。

●ON…GOTO文をIF…THEN文を使って書いてみると、下のように数行のプログラムになってしまふ。数値変数または計算式の値によって、何通りにも行き先がかかる場合、ON…GOTO文を使うと便利である。

10 ON A GOTO 70, 80, 90

→ {  
 10 IF A=1 THEN 70  
 20 IF A=2 THEN 80  
 30 IF A=3 THEN 90
 }

```
LIST
10 REM ** ON GOTO ジサンブル ***
20 INPUT "1 カラ 4 マテノカヌ" ライ
30 ON N GOTO 40,60,80,100
40 PRINT "TOKYO"
50 GOTO 20
60 PRINT "NEWYORK"
70 GOTO 20
80 PRINT "PARIS"
90 GOTO 20
100 END
Ready
RUN
1 カラ 4 マテノカヌ ライレナサイ:3
PARIS
1 カラ 4 マテノカヌ ライレナサイ:2
NEWYORK
1 カラ 4 マテノカヌ ライレナサイ:1
TOKYO
1 カラ 4 マテノカヌ ライレナサイ:■
```

行番号20で入力された数字が1か2か3によって、行番号30のON…GOTO文で、それぞれ、行番号40、60、80のPRINT文を実行する。4を入れた場合は行番号100へとび、実行が終わる。

# プログラムの繰り返し—FOR NEXT

フロー

ネクスト

## 使用例

① 10 FOR I=1 TO 10

20 A=A+I

30 NEXT I

変数Iの値を1から1ずつふやし、  
10をこえるまで、行番号10から30  
までの実行を繰り返す。

② 10 FOR J=10 TO 1 STEP -2

20 PRINT J

30 NEXT J

変数Jの値を10から2ずつ減らし、  
1より小さくなるまで、行番号10  
から30までの実行を繰り返す。

一般型式    **FOR** 数値変数=はじめの値    **TO** おわりの値  
              **STEP** 変化の値

こここの間に繰り返されるプログラムを入れる。

**NEXT** 数値変数

## 説明

●変数が「はじめの値」から「変化の値」ずつ変化し、「おわりの値」をこえるまで、FOR…NEXTの間にあるプログラムの実行を繰り返す。

「変化の値」は正ならば変数の値は増えて行き、負ならば減って行く。

●STEP以下を省略すると、STEP 1として実行する。

●NEXTの後の変数は省略できるが、FOR文とNEXT文の数は同じでなくてはならない。

●FOR…NEXT文の内側に、もうひとつのFOR…NEXT文をいれることができる。このことをネスティング(入れ子)と呼ぶ。ネスティングは10個まで作ることができます。

(例)この例ではネスティングは2個である。

10 FOR I=1 TO 10 —————

20 PRINT I

30 FOR J=1 TO 5 —————

40 PRINT I×J

50 NEXT J —————

60 NEXT I —————

①

\*注意 RX-78は浮動小数点を使用しているため、STEPを小さくすると、多小の誤差を生じことがあります。

```
①
LIST REM *** カズノ コウケイ ノ モトブル ***
* 20 G=0
30 INPUT "カズノ ノ イレテクタサイ:";N
40 FOR I=1 TO N
50 G=G+I
60 NEXT I
70 PRINT "1 カラ";N;" マテノワ="
;G
80 END
Ready
RUN
カズノ イレテクタサイ:10
1 カラ 10 マテノワ= 55
Ready
```

1から指定した数までの合計を求める。

行番号40から60までのFOR…NEXT文で、Iが1から指定した数になるまで、数の足し込みを行なう。

②IF…GOTO文(44ページ)の写真②のプログラムで同じことをやっているので比べてみましょう。

```
②
LIST REM *** 22 J ヒヨウ ノ ツクリ ***
20 FOR I=1 TO 9
30 FOR J=1 TO 9
40 PRINT USING "###"; I * J;
50 NEXT J
60 PRINT
70 NEXT I
80 END
Ready
RUN
1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
Ready
```

行番号20から70までのFOR…NEXT文でIを1から9まで、行番号30から50までのFOR…NEXT文でJを1から9まで動かし、行番号40のPRINT USING文でIとJの積を画面に出している。行番号40のPRINT USING文は、I×Jが1×1から9×9まで81回実行される。

## STEP 2 サンプルプログラム(まとめ)

```

10 REM ** エン シカク ヲ カク ***
20 PRINT CHR$(6) : COLOR 4
30 INPUT "エンヨウ ナラ 1, シカクヨウ ナラ 2, オフリ ナラ
3 ヲ オシナサイ"; A
40 ON A GOTO 50, 150, 250
50 INPUT "ト ウシエンノ カスハ?"; N
60 INPUT "モヨウ ヲ スラス カイスウハ?"; K
70 PRINT CHR$(6)
80 FOR I=1 TO K
90 FOR J=1 TO N
100 CIRCLE I*10, 90, J*10
110 NEXT J
120 NEXT I
130 FOR I=1 TO 1500 : NEXT I
140 GOTO 20
150 INPUT "ショウテン ヲ キメヨ [X<=191, Y<=183];
"; X, Y
160 INPUT "モヨウ ヲ スラス カイスウハ?"; K
170 PRINT CHR$(6)
180 FOR I=1 TO K
190 IF I*(X+5)>191 THEN 230
200 IF I*(Y+5)>183 THEN 230
210 BOX I*X, I*Y, I*(X+5), I*(Y+5)
220 NEXT I
230 FOR I=1 TO 1500 : NEXT I
240 GOTO 20
250 END

```

### [解説]

緑色の円や四角を描いてみよう。

円を描くか、四角を描くか、描かずに終了するかを指定する。

- 円を指定すると → 行番号50から120までで、指定した数の同心円を指定した回数だけずらしながら円を描く。

- 四角を指定すると → 行番号150から220までで、指定した位置に左上の頂点がある四角を、指定した回数だけずらしながら描く、画面からはみ出した時は、描くのをやめる。

- 終了を指定すると → 行番号250にとんでプログラムを終了させる。

行番号130、230のFOR…NEXT文は、描いた図形をすぐ消さないように、時間をとるためのものである。これからも、こうしたテクニックはたくさん使うので覚えておこう。

\* サンプルプログラム中に出てくる〔 〕はキーボード上で（ ）と同じです。

### 注釈文

画面の内容をすべて消す。カラーコードを4にする

円なら1、四角なら2、終了なら3をAに指定する。  
Aが、1なら行番号50、2なら行番号150、3なら行番号250へとぶ。

同心円の個数をNに指定する。

円を横にずらす回数をKに指定する。

画面の内容をすべて消す。

Iが1からKの間、行番号120までを繰り返す。

Jが1からNの間、行番号110までを繰り返す。

点(I×10, 90)を中心とする、半径J×10の円を描く。

行番号90の繰り返し命令の終りを示す。

行番号80の繰り返し命令の終りを示す。

Iが1から1500の間、この行番号を繰り返す。

行番号20へとぶ

四角の左上の頂点の位置をX、Yに指定する。

四角をずらす回数をKに指定する。

画面の内容をすべて消す

Iが1からKの間、行番号220までを繰り返す。

I×(X+5)が191をこえた場合、行番号230にとぶ。

I×(Y+5)が183をこえた場合、行番号230にとぶ。

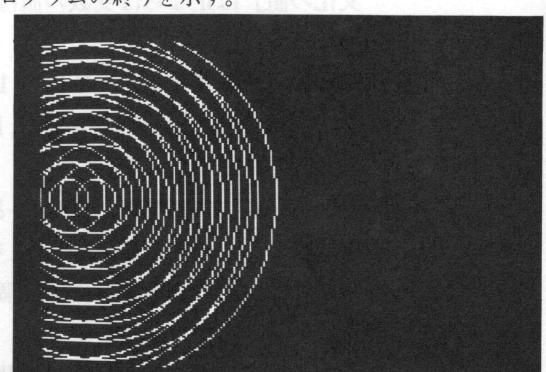
左上の頂点が、(I×X, I×Y)、右下の頂点が(I×(X+5), I×(Y+5))の四角を描く。

行番号180の繰り返し命令の終りを示す。

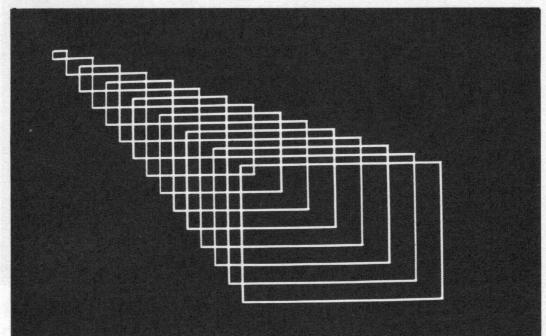
Iが1から1500の間、この行番号を繰り返す。

行番号20へとぶ。

プログラムの終りを示す。



円を選んだ場合



四角を選んだ場合

# STEP 3 RX-78で数字を料理

RX-78では、数学に出てくるサイン、コサインなどの色々な関数が使える。

STEP3は、関数のうち、「数値関数」と呼ばれる数字を扱う関数の説明を行なう。数値関数を使えば、複雑な計算も簡単な命令文でできてしまう。

## STEP 3 サンプルプログラム

```

10 REM ** アンサンノレンシュウ ***
20 PRINT CHR$(6) : CT=0 : I=0 : D=0
30 IF CT>200 THEN 150
40 I=I+1
50 A=INT(RND(1)*100)
60 B=INT(RND(1)*100)
70 ANS=A+B
80 PRINT A;"+";B
90 INPUT "コタエ？";C
100 IF ANS=C THEN 130
110 PRINT "ハズレ";"コタエハ";ANS
120 GOTO 30
130 PRINT "コメイサン"
140 D=D+1 : GOTO 30
150 PRINT "セイカイ";D
160 PRINT "マチカイ";I-D
170 END

```

注釈文  
 画面の内容をすべて消す。CT、I、Dを0とする。  
 CTが200より大きい場合、行番号150にとぶ。  
 Iに1を加える。  
 Aの値を乱数を使って決める。  
 Bの値を乱数を使って決める。  
 ANS=A+Bとする。  
 たし算 A + B の式を画面に出す。  
 たし算の答をCに入れる。  
 ANSとCが等しい場合、行番号130にとぶ。  
 答が違う場合、正解を画面に出す。  
 行番号30にとぶ。  
 "御明算"と画面に出す。  
 Dに1を加える。行番号30にとぶ。  
 正解の回数を画面に出す。  
 まちがいの回数を画面に出す。  
 プログラムの終りを示す。

### 【解説】

RX-78で暗算の練習をしよう。次々に出てくるたし算の問題に答えをいれ、答が違う場合は正解を教えてくれる。

時間に制限があり、時間切れになつたら、正解の回数と間違った回数が画面に出てくる。



```

66+ 38
コタエ ?104
375+ 44
コタエ ?119
コメイサン
13+ 6
コタエ ?13
コメイサン
21+ 93
コタエ ?113
ハズレ コタエハ 114
45+ 53
コタエ ?98
コメイサン
セイカイ 4
マチカイ 1
Ready
■

```

\* サンプルプログラム中に出てくる〔 〕はキーボード上で（ ）と同じです。

# でたらめな数、それが乱数

**RND**

使用例

- |                   |   |
|-------------------|---|
| <b>① A=RND(1)</b> | 0.00000001~0.99999999の数の中から、毎回違う数字を選び、数値変数Aに入れる。    |
| <b>② B=RND(0)</b> | 0.00000001~0.99999999の数の中から、いつも決まった数字を選び、数値変数Bに入れる。 |

一般型式 RND (整数)

**説明** ●0.00000001~0.99999999までの数字のうちから、  
RX-78が数字を選びだす関数である。選んだ数字を  
乱数と呼ぶ。  
(注)

- カッコ内が正の整数（1、2、3、4、…）の場合、RND関数を使うたびに、次々に数字が変わっていく。

- カッコ内が0または負の整数(-1、-2、-3、-4、...)の場合、何度RND関数を使っても、ある特定の数字が出てくる。



```

10 REM ** RND サンプル ***
20 PRINT CHR$(6)
30 PRINT "-- RND(1) --"
40 FOR I=1 TO 5
50 PRINT RND(1)
60 NEXT I
70 PRINT "-- RND(0) --"
80 FOR J=1 TO 5
90 PRINT RND(0)
100 NEXT J
110 PRINT "-- RND(-1) --"
120 FOR K=1 TO 5
130 PRINT RND(-1)
140 NEXT K
150 END

```

行番号50、90、130でRND関数の値を画面に出している。  
RND(1)の場合は、毎回違う数字になり、RND(0)、RND(-1)  
の場合は、毎回同じ数字になるのがわかる。

⑭完全な乱数ではなく、電源を入れた時点、あるいは **RESET** キーを押した時点で初期値に戻る擬似乱数です。

# D 三角関数 SIN, COS, TAN, ATN

## 使用例

- ① **A=SIN(π/2)** 数値変数Aに  $\sin\frac{\pi}{2}$  の値を入れる。
- ② **B=COS(π/2)** 数値変数Bに  $\cos\frac{\pi}{2}$  を入れる。
- ③ **C=TAN(π/3)** 数値変数Cに  $\tan\frac{\pi}{3}$  を入れる。
- ④ **D=ATN(1)** 数値変数Dに  $\arctan 1$  の値を入れる。

## 一般型式

**SIN** (角度)  
**COS** (角度)  
**TAN** (角度)  
**ATN** (数値)

## 説明 SIN, COS, TAN

- 三角関数  $\sin$ (正弦)、 $\cos$ (余弦)、 $\tan$ (正接) とそれぞれ同じである。
- 角度は、“ラジアン”という単位を使って指定する。ラジアンは半径を基準にした単位で、半径と同じ長さの円弧の中心角が1ラジアンになる。角度を度からラジアンに変換するには、 $\pi/180$ をかけなければならない。例えば、 $180^\circ = \pi$ ラジアンである。

## ATN

- $\tan$ (正接)の逆関数  $\arctan$ のことである。結果は、やはりラジアンで出る。したがって、度の単位で結果がほしいときには、 $180/\pi$ をかける。
- ( )に入れる角度(ラジアン)は数値変数、計算式、関数でもよい。

```

10 REM ** SIN COS TAN **
20 PRINT CHR$(6)
30 INPUT "ラジアン テキスト イレル A=π"; A
40 B=π/A : D=180/A
50 QS=SIN(B)
60 QC=COS(B)
70 IF A=2 THEN PRINT "TAN(π/2) のコタエ カテ" : GOTO 110
80 QT=TAN(B)
90 PRINT
100 PRINT : PRINT "TAN("; D; ")="; QT
110 PRINT : PRINT "COS("; D; ")="; QC
120 PRINT : PRINT "SIN("; D; ")="; QS
130 PRINT : PRINT : INPUT "モウイチヤル 1, ハリヤル 2"; C
140 IF C=1 THEN 20
150 IF C=2 THEN END
160 GOTO 130

```

行番号50~80のSIN, COS, TANに与える角度を、行番号30のINPUT文で指定する。

このとき、角度をラジアンで与えるように注意してほしい。

```

ラジアン テキスト イレル A=π/3
TAN( 60° )= 1.7320508
COS( 60° )= 0.5
SIN( 60° )=-0.8660254
モウイチヤル 1, ハリヤル 2 ■

```

# 絶対値、整数、符号—ABS, INT, SGN

## 使用例

- ① **A=ABS(-3)** 数値変数Aに-3の絶対値3を入れる。
- ① **B=INT(3.87)** 数値変数Bに3.87を越えない最大の整数3を入れる。
- ① **C=SGN(-0.5)** -0.5は負の数であるので、数値変数Cに-1を入れる。

## 一般型式

**ABS** (数値)  
**INT** (数値)  
**SGN** (数値)

## 説明

### ABS

- 絶対値を求める関数である。
- 絶対値とは、符号をとった数の値のことである。

(例) ABS(-5) → 5

ABS(0) → 0

ABS(3) → 3

### INT

- カッコ内の数値を、その値を越えない最大の整数にする関数である。特に負の数のときに注意しよう。

(例) INT(0.65) → 0

INT(36.7) → 36

INT(-4.57) → -5

### SGN

- カッコ内の数値の符号を求める関数である。
- 正の数であれば1、0であれば0、負であれば-1になる。

(例) SGN(-5) → -1

SGN(0) → 0

SGN(3.5) → 1

```

LIST REM ** ABS, INT, SGN ***
20 PRINT TAB(13); "ABS" TAB(5)
30 FOR I=1 TO 4
40 READ X
50 XA=ABS(X)
60 XI=INT(X)
70 XS=SGN(X)
80 PRINT X; XA, XI, XS
90 NEXT I
100 DATA A=-5.6,B=0.2
110 DATA C=-3.6,D=0.7
Ready RUN
A=      5.6    ABS   5.6    1
B=      0.2    INT   0      0
C=     -3.6    SGN  -4     -1
D=      0.7
Ready

```

行番号40のREADでA、B、C、Dの数値を読み、行番号60、70で、ABS、INT、SGNに与えた。  
実行結果を見て、それぞれの働きを確かめておこう。

# 特殊関数

平方根  
対数  $e^x$

# SQR, EXP, LOG, LN

## 使用例

- |             |                              |
|-------------|------------------------------|
| ① A=SQR(5)  | 数値変数Aに $\sqrt{5}$ の値を入れる。    |
| ① B=EXP(6)  | 数値変数Bに $e^6$ の値を入れる。         |
| ① C=LOG(15) | 数値変数Cに $\log_{10}15$ の値を入れる。 |
| ① D=LN(7)   | 数値変数Dに $\log_e7$ の値を入れる。     |

## 一般型式

**SQR** (0 または正の数値)  
**EXP** (数値)  
**LOG** (正の数値)  
**LN** (正の数値)

## 説明 SQR

- 平方根を求める関数である。
- 数値は0 または正の数でなければならない。

## EXP

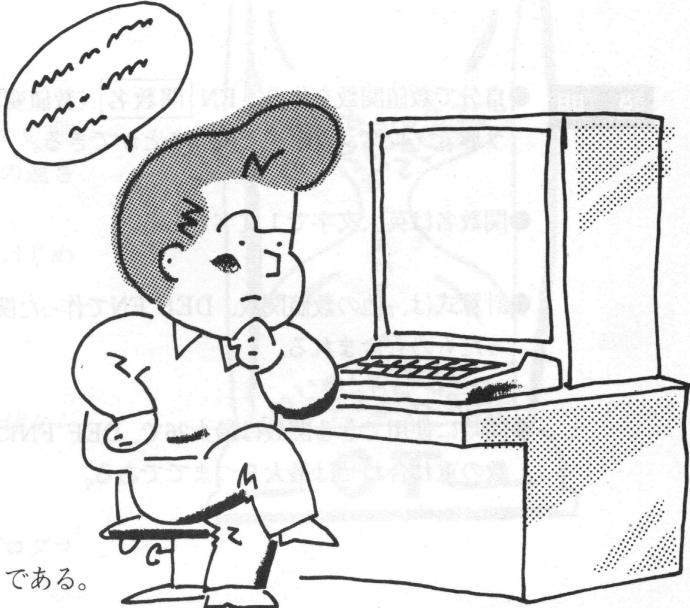
- $e$  のべき乗を求める関数である。
- $e$  とは自然対数( $\log_e$ )の底で、 $e = 2.7182818$  である。

## LOG

- 常用対数( $\log_{10}$ )を求める関数である。
- 数値は正の数でなければならない。

## LN

- 自然対数( $\log_e$ )を求める関数である。
- 数値は正の数でなければならない。



```

1000 REM ** SQR, EXP, LOG, LN ***
1010 A=20 : B=15 : C=10 : D=25
1020 E=EXP(A)
1030 F=LOG(B)
1040 G=LN(C)
1050 H=SQR(D)
1060 PRINT "SQR<";A;">=";H
1070 PRINT "EXP<";B;">=";E
1080 PRINT "LOG<";C;">=";F
1090 PRINT "LN<";D;">=";G
1100 END
Ready

RUN
SQR< 20>= 4.472136
EXP< 15>= 3269017.4
LOG< 10>= 1
LN< 25>= 3.2188758
Ready

```

行番号30~60でいろいろな関数を使って計算を行なった。行番号20のA、B、C、Dの値をかえて、いろいろやってみよう。ただし、Aは0 または正の数、C、Dは正の数でなければならない。

# 自分でつくる数値関数

DEF FN

## 使用例

**DEF FNA(X)=X↑2-X**

FNA(X) を  $X^2 - X$  行なう関数  
としてプログラムの中で使う。

B=INT(3.87)

一般型式 **DEF FN** 関数名 (数値変数) = 計算式

**説明** ●自分で数値関数を作り、FN [関数名] (数値変数) という形でプログラムの中で使うことができる。

- 関数名は英大文字で1文字である。
- 計算式は、他の数値関数、DEF FNで作った関数を使ったものも含まれる。
- 同時に使用できる関数は最大26で、DEF FNによる関数の重ね合わせは最大5つまでである。

```

10 REM ** DEF FN サンプル ***
20 PRINT CHR$(6)
30 DEF FNA(X)=INT(X/3)
40 DEF FNB(X)=X-3*FNA(X)
50 FOR X=11 TO 29
60 A=FNA(X)
70 B=FNB(X)
80 PRINT X;" /3=";A;" アマリ";B
90 NEXT X
100 END

```

行番号30、40で関数を作り、行番号60、70で使っている。FNA、FNBはそれぞれ、数を3で割った答、余りを求める関数である。

1	1	3	3	2	2	1	1
1	3	3	2	2	1	1	2
1	4	4	3	3	2	2	1
1	5	5	4	4	3	3	2
1	6	6	5	5	4	4	3
1	7	7	6	6	5	5	4
1	8	8	7	7	6	6	5
1	9	9	8	8	7	7	6
2	0	0	9	9	8	8	7
2	1	1					
2	2	2					
2	3	3					
2	4	4					
2	5	5					
2	6	6					
2	7	7					
2	8	8					
2	9	9					
3	0	0					
3	1	1					
3	2	2					
3	3	3					
3	4	4					
3	5	5					
3	6	6					
3	7	7					
3	8	8					
3	9	9					
4	0	0					
4	1	1					
4	2	2					
4	3	3					
4	4	4					
4	5	5					
4	6	6					
4	7	7					
4	8	8					
4	9	9					
5	0	0					
5	1	1					
5	2	2					
5	3	3					
5	4	4					
5	5	5					
5	6	6					
5	7	7					
5	8	8					
5	9	9					
6	0	0					
6	1	1					
6	2	2					
6	3	3					
6	4	4					
6	5	5					
6	6	6					
6	7	7					
6	8	8					
6	9	9					
7	0	0					
7	1	1					
7	2	2					
7	3	3					
7	4	4					
7	5	5					
7	6	6					
7	7	7					
7	8	8					
7	9	9					
8	0	0					
8	1	1					
8	2	2					
8	3	3					
8	4	4					
8	5	5					
8	6	6					
8	7	7					
8	8	8					
8	9	9					
9	0	0					
9	1	1					
9	2	2					
9	3	3					
9	4	4					
9	5	5					
9	6	6					
9	7	7					
9	8	8					
9	9	9					

\* サンプルプログラム中で出てくる〔〕はキーボード上で〔〕と同じです。

# N RX-78の中の砂時計

CT

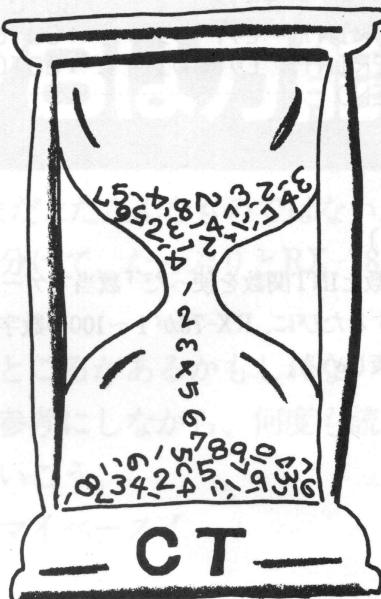
使用例

- ① PRINT CT 現在のカウンタの値を画面にだす。
  - ② CT=0 カウンタCTに0を入れる。

一般型式 CT

說明

- RX-78 に電源を入れたり、リセットスイッチを押したと同時に 0 から 65535 までの数字が一定の速さ(0.133721秒ごと)でカウントされ CT にはいる。
  - CT が 65535 まできたら、ふたたび 0 からかぞえはじめ る。
  - CT = 50 と数値を入れることができ、CT はその値から 65535 になるまでカウントする。
  - ゲームなどのような、制限時間を決めてやるプログラムなどに使える。



```
10 REM ** CT サンプル ***
20 PRINT CHR$(6)
30 CT=0
40 PRINT "*";
50 IF CT<10 THEN 40
60 PRINT ".";
70 IF CT<20 THEN 60
80 PRINT "?";
90 IF CT<30 THEN 80
100 END
```

行番号50、70、90のIF文でCTの値を調べ、条件にあてはまれば  
それぞれ行番号40、60、80のPRINT文をくり返す。

※ サンプルプログラム中に出てくる〔 〕はキーボード上で（ ）と同じです

### STEP 3 サンプルプログラム(まとめ)

```

10 REM ** カスアテ クイズ **
20 PRINT CHR$(6) .....
30 A=INT(RND(1)*100)+1 .....
40 N=N+1 :PRINT N;"カイメ",
50 INPUT "イクリ テ ショウカ ? ";B
60 IF A=B THEN 90 .....
70 IF A<B THEN 100 .....
80 IF A>B THEN 110 .....
90 PRINT "オオアタリ ♪ ";GOTO 120 .....
100 PRINT "モット シタ ↓ ";GOTO 40 .....
110 PRINT "モット ウエ ↑ ";GOTO 40 .....
120 END .....

```

#### 注釈文

画面に出てる内容をすべて消す。  
1~100までの乱数を発生させ、Aに代入する。  
回数を画面に表示する。  
「いくつでしょうか？」と画面に出し、数を入力する。  
AとBが等しい時、行番号90へとぶ。  
AがBより小さい時、行番号100へとぶ  
AがBより大きい時、行番号110へとぶ  
「おお当たり！」と画面へ表示し、行番号120へとぶ。  
「もっと下」と画面へ表示し、行番号40へとぶ。  
「もっと上」と画面へ表示し、行番号40へとぶ  
プログラムの終りを示す。

#### [解説]

RND関数とINT関数を使った「数当てゲーム」である。プログラムを実行するたびに、RX-78が1~100の数字を選んでくれるので、何回でも楽しめる。

```

1カイメ ? ショウカ ?50
イクリエ ! ショウカ ?10
2カイメ ? ショウカ ?70
イクリテ ? ショウカ ?65
モットシタ ? ショウカ ?60
3カイメ ? ショウカ ?65
イクリ ? ショウカ ?60
モットシタ ?
4カイメ ? ショウカ ?60
イクリテ !
オオアタリ!
Ready

```

