

5

第5章

RX-78は万能

RX-78のBS-BASICはまだまだこんなものではない。この章では5つのSTEPに分けて、たっぷりとRX-78のBASICを満載してある。

少しばかり難しく感じるところがあるかもしれないが、サンプルプログラムを参考にしながら、何度も読み返して一歩ずつ理解していこう。

ともかく、マイペース、マイペース！



STEP 1 思った場所にワープする

PRINT命令を使うと画面に数字や文字がだせるが、画面のどこにだすかということが自由に決められない。広い画面がもったいない。しかしご安心、RX-78にはそんな君の悩みがよく分っている。STEP 1で紹介する命令文を使うと、まるでカーソルがワープするかのように、画面を自由に飛びまわるのだ。

STEP 1 サンプルプログラム

```
10 REM ** ワープ* セヨ **-----  
20 PRINT CHR$(6) -----  
30 PRINT TAB(10); "START!!" -----  
40 FOR I=1 TO 30 -----  
50 FOR J=1 TO 2000 :NEXT J -----  
60 PRINT CHR$(6) -----  
70 X=RND(1)*20 :Y=RND(1)*20 -----  
80 CURSOR X, Y -----  
90 PRINT "GUNDAM" -----  
100 NEXT I -----  
110 END -----
```

注釈文

画面にでている内容をすべて消す。

画面の左はこれから10文字分とばして、START!!とだす。
Iが1から30の間、行番号100までの命令文を繰り返す。

Jが1から2000の間この行番号の命令文を繰り返す。

画面にでている内容をすべて消す。

画面のよこ方向、たて方向の位置を乱数を使って決める。
カーソルを行番号70で決めた位置にとばす。

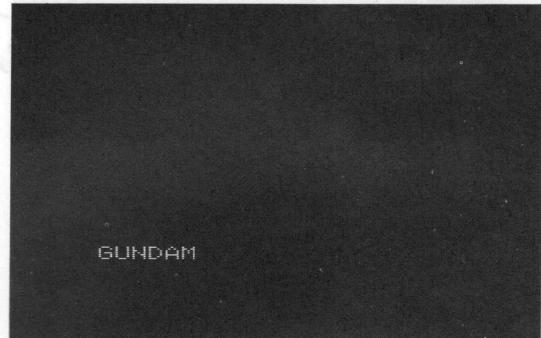
「GUNDAM」と画面にだす。

行番号40の繰り返し命令の終りを示す。

プログラムの終りを示す。

〔解説〕

画面のいろんな位置に文字を出してみよう。このプログラムを動かすと、まず画面のまん中にSTART!!とでて、次にGUNDAMという文字が画面のあちこちにでたり消えたりする。このカギを握っているのは、行番号30のTABと行番号80のCURSORという2つの命令だ。君もこうした命令を覚えて、画面を自由に使いこなそう。



④行番号50、70ではマルチステートメントを使っています。くわしくは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。

* 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。

同じ行ならとばそう

タブ

TAB

使用例

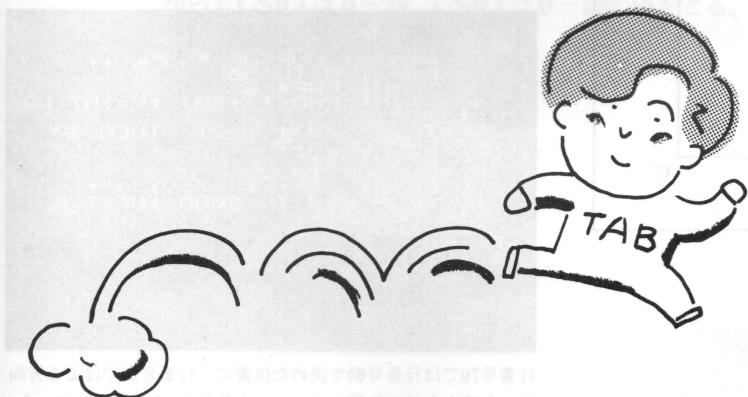
```
PRINT TAB(10); " A "
```

画面よこ方向、左はしから10文字
分とばして、Aという文字を出す。

一般型式 TAB (画面よこ方向のとばす文字数)

説明 ●画面の左はしからかぞえて何文字分とばして文字をだすか、つまりカーソルを左はしから何文字進めてから文字をだすかをコントロールする。

●TABの後に指定できる数字は0~255であるが、画面の1行は最大30文字なので、30以上の数字を指定すると、30で割ったあまり分だけ文字をとばす。



④TABでとばした位置にすでに文字がでていた場合、1行下に文字をだします。たとえば、

```
10 PRINT "AAAAAA";
20 PRINT TAB(3); "B"
```

というプログラムを動かすと、結果は A A A B A とはならず、A A A A A となります。

B

```
LIST REM ** TAB **
20 PRINT CHR$(5)
30 FOR I=0 TO 20
40 PRINT TAB(I); "A";
50 NEXT I
70 END
Ready
```

*を行をかえ、しかも右に1文字ずつずらしながら21個画面にだす。

自由自在にカーソル移動—CURSOR

カーソル

使用例**CURSOR 3, 7**

カーソルをよこ方向3、たて方向7(7行目)の位置に移動する。

一般型式 **CURSOR** よこ方向のカーソル位置、たて方向のカーソル位置

説明 ●画面のどの位置にカーソルをもってくるかを、指定する。

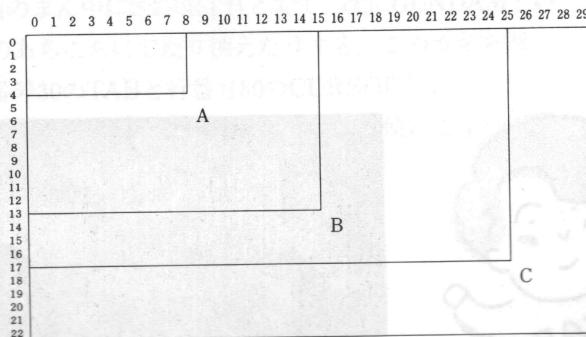
●カーソル位置は、よこ方向が0~29の30文字分、たて方向が0~22の23行分指定できる。

たとえば、下の図で、CURSOR 8, 4はAの位置

CURSOR 15, 13はBの位置

CURSOR 25, 17はCの位置

になる。



```

LIST REM ** CURSOR サンプル ***
20 FOR I=0 TO 50
30 PRINT CHR$(6)
40 X=INT(RND(1)*30) : Y=INT(RND(1)*30)
50 IF (X>29)+(Y>21) THEN 40
60 CURSOR X,Y
70 PRINT "*"
80 CURSOR 10,22
90 PRINT "(";X;"";Y;")"
100 FOR B=0 TO 2000 :NEXT B
110 NEXT I
120 END
Ready

```

行番号70では行番号60で決めた位置に、行番号80ではよこ方向10、たて方向22の位置にカーソルを移動し、文字をだしている。

(12, 8)

今、カーソルはどこ?—CSR_X, CSR_Y

使用例

① PRINT CSRX

現在カーソルが、画面上どこ方向のどの位置にあるかを知らせてくれる（システム）変数である。

① PRINT CSRY

現在カーソルが、画面上たて方向のどの位置にあるか（何行目にあるか）を知らせてくれる（システム）変数である。

一般型式

CSR
CSRY

說明

- CSRX、CSRYは、RX-78が持っている変数（システム変数という）で、それぞれ、カーソルのよこ方向位置、たて方向の位置が常に入っている。
 - プログラムを実行中、カーソルの位置が知りたい場合がある。そんなときにCSRXとCSRYを使うといい。
 - RX-78によって表示されている画面は、よこ方向が0～29の30文字、たて方向が0～22の23行である。したがってCSRXは0～29、CSRYは0～22の値をとる。

```

10 REM ** CSR X,CSR Y サンプル ***
20 PRINT CHR$(6)
30 FOR I=1 TO 20
40 A=INT(RND(1)*21)
50 FOR J=0 TO A
60 PRINT ".";
70 NEXT J
80 PRINT "*";CSRX-1;",";CSRY
90 NEXT I
100 END

```

行番号60でだす・の数を行番号40で決めている。行番号80で＊を画面にだし、その位置をCSRXとCSRЫでもとめた。
CSRХ-1としているのは、CSRХつまりカーソルのよこ方向の位置が、＊の次になっているからだ。

```

* 5, 1      * 15, 2
             * 14, 3
             * 18, 4
* 4, 5
* 5, 6
* 5, 7      * 16, 8
* 1, 9      * 10, 10
             * 8, 12      * 20, 11
             * 4, 14      * 13, 13
             * 20, 15
             * 19, 16
             * 20, 17
* 8, 18      * 14, 19
             * 11, 20

```

Ready

④CSRX=10, CSRY=15というように数値を入れることはできない。しかし、CURSOR 10, 15とすれば、それぞれに数字を入れたことになる。

* 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 1 サンプルプログラム(まとめ)

```

10 REM ** ワオ！ カンタム **
20 PRINT CHR$(6)-----
30 FOR X=5 TO 20 STEP 5-----
40 FOR Y=2 TO 18 STEP 2-----
50 CURSOR X,Y -----
60 PRINT "ワオ！" -----
70 FOR I=0 TO 500:NEXT I -----
80 NEXT Y,X -----
90 END -----

```

注釈文

画面に出ている内容をすべて消す。

Xの値を5から20の間を5づつ増やし、行番号30～80までを繰り返す。

Yの値を2から18の間を2づつ増やし、行番号40～80までを繰り返す。

カーソルの位置を決める。

ワオ！を画面に出す。

Iを0から500まで繰り返す(時間をかせぐ)。

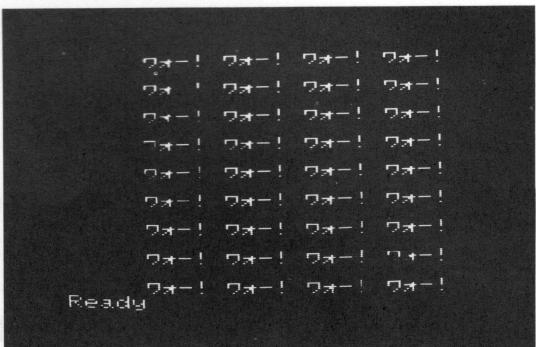
行番号30、40の繰り返しの終りを示す。

プログラムの終りを示す。

[解説]

ワオ！という文字を画面に順に出すプログラムだが、出てくる順番に注目してほしい。行番号50のCURSOR命令でなくてはできない芸当だ。

結果だけを見て、これと同じようにワオ！を並べるのに、TABを使ってやるとどうなるだろうか。考えてみよう。



＊ 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。

行番号50ではカーソルを左に移動する命令で、行番号40では右に移動する命令でカーソルの位置を決めて、各カーソル位置に文字をなして、各

STEP 2

情報は有効に

プログラムで使う変数に値を与える方法として、今までに、INPUT文なんていうのが出てきた。しかし、変数にまとめて値を与えていた場合、いちいちRX-78の問いかけに答えているのはめんどうだ。STEP2に出てくる命令文を使えば、まとめて値を与えておいて、次々に利用したり、何回も再利用したりできる。つまり情報を有効に使うことができるようになるわけだ。

STEP 2 サンプルプログラム

```

10 REM ** シンショウ ノ フンフ ト ハイキン **
20 K1=0 :K2=0 :K3=0 :K4=0 :K5=0 :G=0 :P
RINT CHR$(6) -----
30 READ N -----
40 FOR I=1 TO N -----
50 READ A -----
60 IF A<140 THEN K1=K1+1 -----
70 IF (A)>=140)*(A<150) THEN K2=K2+1 -----
80 IF (A)>=150)*(A<160) THEN K3=K3+1 -----
90 IF (A)>=160)*(A<170) THEN K4=K4+1 -----
100 IF A>=170 THEN K5=K5+1 -----
110 NEXT I -----
120 PRINT " シンショウ      ニンズウ "
130 PRINT "140ミマン      ";K1 -----
140 PRINT "140-150      ";K2 -----
150 PRINT "150-160      ";K3 -----
160 PRINT "160-170      ";K4 -----
170 PRINT "170イショウ      ";K5 -----
180 PRINT -----
190 PRINT "ハイキンチ ヲ タスナラハ CONT トイレテ ツ
タサイ" -----
200 STOP -----
210 RESTORE 300 -----
220 FOR I=1 TO N -----
230 READ A -----
240 G=G+A -----
250 NEXT I -----
260 H=G/N -----
270 PRINT "ハイキン = ";H -----
280 END -----
290 DATA 40 -----
300 DATA 148,152,157,135,142,150,160,14
1,162,147 -----
310 DATA 158,155,167,144,168,142,151,15
4,160,153 -----
320 DATA 160,152,153,148,147,159,161,15
5,165,150 -----
330 DATA 169,144,142,151,156,150,162,15
9,157,146

```

注釈文

各種変数に0をいれる。画面に出ている内容をすべて消す。
データ(人数)を読み込む。

Iが1から行番号30で読み込んだ数字までの間、行番号110までの命令文を繰り返す。

データを読み込む。

読み込んだデータが140未満の場合、K1に1を加える。
読み込んだデータが140以上150未満の場合、K2に1を加える。

読み込んだデータが150以上160未満の場合、K3に1を加える。

読み込んだデータが160以上170未満の場合、K4に1を加える。

読み込んだデータが170以上の場合、K5に1を加える。
行番号40の繰り返し命令の終りを示す。

タイトルを画面に出す。

} 身長の範囲毎の人数を画面に出す。

画面の1行分をあける。

平均値を求める方法を画面に出す。

プログラムの実行を止める。

読み込むデータを行番号300のDATA文の先頭にもどす。
Iが1から行番号30で読み込んだ数字までの間、行番号250までの命令文を繰り返す。

データを読み込む。

読み込んだデータを合計に加える。

行番号220の繰り返し命令の終りを示す。

身長の合計を人数で割って、平均身長を求める。

平均身長を画面に出す。

プログラムの終りを示す。

人数のデータ

} 身長のデータ

* 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。

[解説] サンプルプログラム(まとめ) ① 教科書で学んだことを上に記入する手順を教えるので、自分で

ある中学校の1クラスの身長の分布と平均身長を求めてみよう。

行番号290のDATA文にクラスの人数が、行番号300～330のDATA文にクラス全員の身長が入っている。

まず、行番号30から170までの命令文で身長の分布を求め、プログラムの実行が一度止まる。ここで、CONT [RETURN]といれてやると、行番号210から270までの命令文で平均身長を求めてくれる。

```

シニチヨウ ニンスツウ
140-150 11
150-160 18
160-170 18
170-180 0
180-190 0
190-200 0

ハイキンチ ラ タスナラハ CONT ト イレテ クタ
サ1
*STOP IN 200
Ready

```

```

シニチヨウ ニンスツウ
140-150 1
150-160 11
160-170 18
170-180 0
180-190 0
190-200 0

ハイキンチ ラ タスナラハ CONT ト イレテ クタ
サイ
*STOP IN 200
B03d4
CONT
ハイキン = 153.3
Ready

```

データを読む

リード

データ

READ DATA

使用例

① 10 READ A,B,C
100 DATA 25,-36,417

変数A、B、Cにそれぞれデータ
25、-36、417を読み込む。

② 20 READ K\$,K,L\$,L
200 DATA ホッカイドウ,552
210 DATA キュウシュウ,1393

変数K\$、K、L\$、Lにそれぞれ、
ホッカイドウ、552、キュウシュウ、
1393を読み込む。

① 10 READ A,B,C
20 RESTORE
30 READ D,E
40 DATA 25,-36,417

一般型式 **READ** 変数, 変数, ……
 DATA データ, データ, ……

説明 ●READ文の後の変数に、DATA文のデータ（値）を読み込む。

●変数とその変数に読み込まれるデータの順番は同じである。

●変数の個数が、データの個数より多い場合はエラーになる。

●変数とその変数に読み込まれるデータのタイプは一致していなければならない。つまり、数値変数には数字、文字変数には文字を与えなければならない。



```
LIST REM *** コウケイト ハイキン ラートブル
** G=0
30 READ N
40 FOR I=1 TO N
50 READ A
60 G=G+A
70 NEXT I
80 H=G/N
90 PRINT
100 PRINT "コウケイ = ";G
110 PRINT "ハイキン = ";H
120 END
130 DATA 29
140 DATA 52,54,98,81,77,56,18
,93,15,27
150 DATA 83,65,32,68,33,31,17
,43,51,92
Ready
```

行番号30のREAD文で、行番号130のDATA文からデータの個数を読み込み、その個数回行番号50のREAD文を繰り返し、行番号140、150のデータを次々と読み込んでいる。

データの再利用

リストア

RESTORE

使用例

① 10 READ A,B,C

20 RESTORE

30 READ D,E

40 DATA 3,5,-4,7,1

変数A、B、Cにそれぞれ3、5、-4
がはいり、行番号20のRESTORE
文で、次に読み込むデータをDATA
文の最初にもどし、変数D、Eに
それぞれ3、5がはいる。

② 10 READ A,B,C

20 RESTORE 50

30 READ D,E

40 DATA 3,5

50 DATA -4,7,1

変数A、B、Cにそれぞれ3、5、-4
がはいり、行番号20のRESTORE
文で、次に読み込むデータを行番
号50のDATA文の最初にもどし、
変数D、Eにそれぞれ-4、7が
はいる。

一般型式

RESTORE

行番号

説明

●次に読み込むデータを指定した行番号のDATA文の先頭のデータにもどす。

行番号を指定しない場合は、最初のDATA文の先頭のデータにもどす。

```

10 REM *** シンショウ ノ ハイキン トノ サ ***
* 20 G=0 :PRINT CHR$(<6>)
30 READ N
40 FOR I=1 TO N
50 READ A :G=G+A
60 NEXT I
70 H=G/N
80 RESTORE 160
90 PRINT "シンショウ ハンサクハイキン トノ サ"
 100 FOR I=1 TO N
110 READ A
120 PRINT A:TAB(15):A-H
130 NEXT I
140 END
150 DATA 28
150 DATA 148,152,157,125,142,
150,180,141,162,142
170 DATA 158,155,167,149,168,
142,151,154,160,155
Ready

```

行番号30のREAD文で、行番号150のDATA文よりデータの個数を読み込み、行番号50のREAD文で、個数分のデータを行番号160、170のDATA文より次々と読み込む。

行番号80のRESTORE文で、行番号160のDATA文の先頭にもどり、行番号110のREAD文で、次々とデータを読み込む。

シンショウ ハンサクハイキン トノ サ	-4.15
148	-0.14999999
152	4.85
157	-22.15
125	-10.15
142	-2.15
150	7.85
180	-11.15
141	2.85
151	2.85
142	-5.15
158	5.85
155	2.85
167	14.85
142	-3.15
149	15.85
160	-10.15
142	-1.15
151	1.85
154	2.85
155	2.85

GET

待ったなしのキー入力

使用例

① GET A

キーボードから数値変数Aに対して、1けたの数字をいれる。

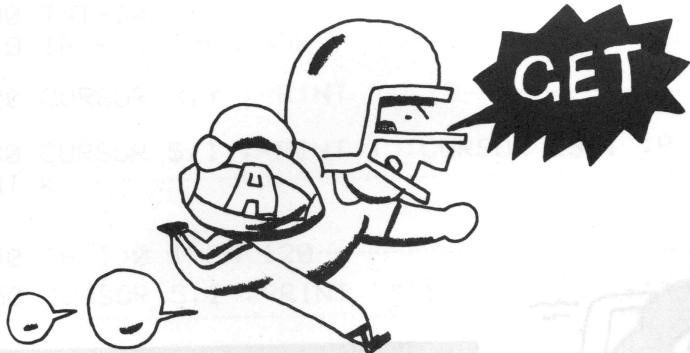
② GET A\$

キーボードから文字変数A\$に対して文字を1字いれる。
キーが押されていないとA\$はからっぽ(NULL)になる。

一般型式 GET 変数

説明 ●キーボードから、変数に1けたのデータをいれる。

●GET文を実行している瞬間に、キーが押されているかいないかで、変数の値が決まる。



④キーを押したままにしても、次のGET文を実行したときには、そのキーの値は、いません。

```
LIST REM *** ニュウヨウ シタ モシバ ノ インシ
*** PRINT CHR$(6)
30 GET A$
40 IF A$="" THEN 30
50 PRINT A$
60 END
Ready
```

キーが押されていると、その文字を画面に出し、押されていないと、押されるまで行番号30のGET文をくり返す。

```
LIST REM *** ミチヲ ワケル ***
20 X=14 : Y=12 : PRINT CHR$(6)
30 CURSOR X,Y : PRINT "*"
40 GET N
50 IF N=1 THEN X=X+1 : GOTO 10
60 IF N=2 THEN Y=Y+1 : GOTO 10
70 IF N=3 THEN X=X-1 : GOTO 10
80 IF N=4 THEN Y=Y-1 : GOTO 10
90 GOTO 40
100 IF (X>29)+(Y>22)+(X<0)+(Y<0) THEN 120
110 CURSOR X,Y : PRINT "*" : GO TO 40
120 END
Ready
```

行番号40のGET文で、キーボードからいれた数字を読み込む。最初のカーソル位置を(14, 12)とし、読み込んだ数字が1なら右、2なら下、3なら左、4なら上へカーソルを進め、*を画面に出す。*を出す位置が画面をはみ出したら、プログラムの実行は終了する。

```
*****
* *
* *
* *
* *
* * *****
* * * ***
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
*****
```

一旦停止と再発車 STOP, CONT

ストップ

コント

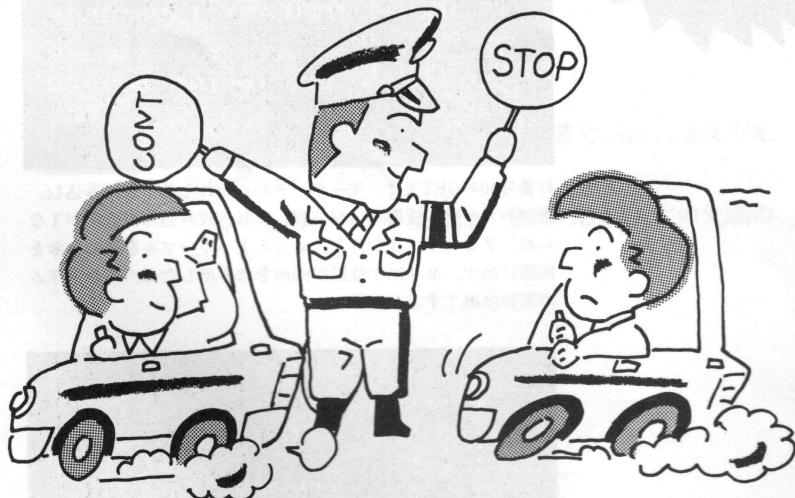
使用例

- ① **STOP** プログラムの実行を停止する。
- ① **CONT** **RETURN** 止めたプログラムの実行を続ける。

一般型式

STOP
CONT **RETURN**

- 説明**
- STOP文でプログラムは実行をやめる。ここで、CONTとキーボードから打ち**RETURN**を押すと、止まった所から実行を続ける。
 - プログラムがEND文や、**SHIFT** + **STOP**キーで止まった場合も、CONT **RETURN**で、続きを実行できる。
 - プログラムが止まった後、行番号を使ったプログラムの修正を行なうと、CONT **RETURN**といれても、続きを実行することはできない。
 - 実行中エラーがあり、Ready状態にもどってしまったときは、CONT **RETURN**は使えない。



- ④プログラムを実行させても、結果がうまく出ないとき、STOP文を何か所かに入れ、止まつたときの変数の値を、ダイレクトモードのPRINT文で、画面に出て確かめることができます。

```

LIST REM ** ヘンズウ ノ ナイヨウ ノ カクニン
*20 READ A,B,C
30 X=A+B-B*C
40 STOP
50 Y=(A+B)/X
60 PRINT X,Y
70 END
80 DATA 3,5,3
Ready
RUN
**STOP IN 40
Ready
PRINT X
0
Ready
CONT
*Error 2 in 50
Ready

```

行番号40のSTOP文でプログラムを止めて、ダイレクトモードのPRINT文でXの値を確かめてみる。Xの値が0の場合は行番号50でエラーになってしまう。0でない場合は、CONT **RETURN**で、続きを実行できる。

STEP 2 サンプルプログラム(まとめ)

```

10 REM ** モグラ タタキ ***
20 PRINT CHR$(6) : T=500 : K=0

30 FOR I=1 TO 3
40 FOR J=1 TO 3
50 CIRCLE 45+48*(J-1), 40+60*(I-1), 20

60 NEXT J
70 NEXT I
80 CURSOR 0,1 : PRINT "モチテン："

90 CURSOR 5,1 : PRINT T
100 CURSOR 22,0 : PRINT "！"

110 CURSOR 13,1 : PRINT "タイム オーバー：" ; K ; "カイ"

120 REM ** モグラ の シュツケン ***
130 CURSOR 0,0
140 FOR I=1 TO 22 : PRINT " " ; NEXT I

150 P=10*RND(1) : Q=10*RND(1)
160 IF P<=3 THEN X=5
170 IF (P>3)*(P<7) THEN X=13
180 IF P>=7 THEN X=21
190 IF Q<=3 THEN Y=5
200 IF (Q>3)*(Q<7) THEN Y=12
210 IF Q>=7 THEN Y=20
220 N=INT(10*RND(1))
230 CURSOR X,Y : PRINT N

240 REM ** モグラ ッ タタク ***
250 CT=0
260 GET A
270 CURSOR CT*2,0 : PRINT "→→"
280 IF CT*2>19 THEN K=K+1 : GOTO 310

290 IF A<>N THEN 260
300 T=T+100 : GOTO 320
310 IF K>5 THEN 350
320 CURSOR X,Y : PRINT "   "
330 CURSOR 5,1 : PRINT T : CURSOR 23,1 : PRINT K

340 IF T>0 THEN 120
350 CURSOR 5,2 : PRINT "サンネット シタ。オワリテス"
360 END

```

注釈文
画面に出てる内容をすべて消す。変数に初期値を代入する。
Iが1から3の間、行番号70までの命令文を繰り返す。
Jが1から3の間、行番号60までの命令文を繰り返す。
(45+48×(J-1)、40+60×(I-1))を中心とする、半径20の円を描く。
行番号40の繰り返し命令の終りを示す。
行番号30の繰り返し命令の終りを示す。
(0, 1)の位置にカーソルを移動し、"モチテン："と画面に出す。
(5, 1)の位置にカーソルを移動し、持ち点を画面に出す。
(22, 0)の位置にカーソルを移動し、"！"を画面に出す。
(13, 1)の位置にカーソルを移動し、タイムオーバーの回数を画面に出す。

注釈文
(0, 0)の位置にカーソルを移動する。
Iが1から22の間、1つのスペースを繰り返し画面に出す。
変数P、Qの値を乱数を使って決める。
Pが3以下なら、X=5とする。
Pが3より大きく、7より小さければ、X=13とする。
Pが7以上なら、X=21とする。
Qが3以下なら、Y=5とする。
Qが3より大きく、7より小さければ、Y=12とする。
Qが7以上なら、Y=20とする。
変数Nの値を乱数を使って決める。
(X, Y)の位置にカーソルを移動し、Nの値を画面に出す。

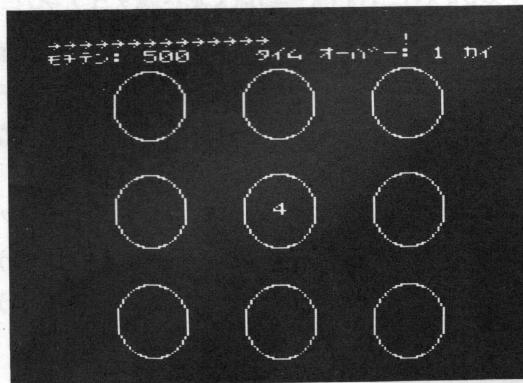
注釈文
変数CTに0をいれる。
変数Aの値をキーボードから読み込む。
(CT×2, 0)の位置にカーソルを移動し、→→と画面に出す。
CT×2が19より大きい場合、タイムオーバーの回数に1を加えて、行番号310の命令文にとぶ。
AとNが等しくない場合、行番号260の命令文にとぶ。
持ち点に100を加える。行番号320の命令文にとぶ。
タイムオーバーの回数が5回をこえた場合、行番号350にとぶ。
(X, Y)の位置にカーソルを移動し、スペースを3個画面に出す。

注釈文
(5, 1)の位置にカーソルを移動し、持ち点を画面に出す。
(23, 1)の位置にカーソルを移動し、タイムオーバーの回数を画面に出す。
持ち点が0より多ければ、行番号120にとぶ。
(5, 2)の位置にカーソルを移動し、終りのメッセージを画面に出す。
プログラムの終りを示す。

〔解説〕

いくつかの穴からあらわれる数字をモグラにたとえて、出てきた数字と同じキーをたたいて消すゲームである。
画面に9個の穴を描いて、1つの穴から、1けたの数字が出てくる。
出てきた数字のキーをGET文でいれるわけだが、制限時間こえてはいけない。こえた回数が5回をこえるとゲームオーバーになる。
うまく行った場合は、得点がふえて、何回でもゲームをすることができる。

* 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 RX-78の中に街を持つ

プログラムであつかうデータの個数がすごくたくさんになると、やたら変数だらけになって、何がなんだかわからなくなるとか、プログラムに同じようなところが何度もでてくるとか、どうにも不便な思いをしたことがないだろうか。

そのような時、配列を使うとプログラムが簡単でスマートになる。

配列とはデータのマンションのようなもの

配列というのは、Aマンション、Bマンションというマンションのようなもの、AマンションのA-1号室にはXさん。A-5号室にはYさんという感じでデータを入れておくことができる。

さらに大きなマンションということで、Aマンション3階のA-301号室などと、たくさんのデータをしまっておくことができる。

配列では、まず配列の大きさというのを決め(宣言して)、RX-78にそれだけの大きさの配列を準備してもらう。そしてあとは、君が自由にデータのだし入れをしてやるという手順になる。

それはちょうど、まずマンションを建ててから、住人が住めるようになるというのと同じだ。

STEP 3 サンプルプログラム

```
10 REM ** ハイレツ サンプル **
20 PRINT CHR$(6) ..... 注釈文
30 DIM A$(9) ..... 配列 A$ を宣言する。
40 FOR I=0 TO 9 ..... I の値を 0 から 9 の間、行番号 40~60 を繰り返す。
50 READ A$(I) ..... データを読み込む。
60 NEXT I ..... 行番号 40 の繰り返しの終りを示す。
70 FOR J=0 TO 4 ..... J の値を 0 から 4 の間、行番号 70~100 を繰り返す。
80 PRINT A$(J); ..... A $(0) ~ A $(4) を 5 コ画面に出す。
90 PRINT TAB(10);A$(J+5) ..... A $(5) ~ A $(9) を 5 コ画面左から 11 番目の所に出す。
100 PRINT :NEXT J ..... 行番号 70 の繰り返しの終りを示す。
110 END ..... プログラムの終りを示す。
120 DATA ニッポン,アメリカ,ソビエト,オーストラリア,イタリア,
カナダ,ト・イツ,イギリス,フランス,イントル ..... 行番号 50 によって読み込まれるデータ。
```

【解説】

国名をREAD～DATAで配列A\$に読み込んで(行番号30～60)、それを画面に出すという単純なプログラムだが、画面での国名の並びに注目してほしい。もしこれを配列を使わずにやろうとするとどうなるだろうか、まず2倍の長さのプログラムになってしまうだろう。

たったこれだけのことをやるのに、こうもちがうのだから、読み込んだ数値で計算をしたり、画面への出し方をくふうする事などを考えると、いかに配列が便利だという事が分ると思う。

ニッポン	カナダ
アメリカ	ト・イツ
ソビエト	イギリス
オーストラリア	フランス
イタリア	イントル
Reads	

* 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。

配列は便利な格納庫

DIM

使用例

① DIM A(10)

A(0)～A(10)の合計11個の配列要素を準備する(宣言する)。

② DIM B(2, 2)

B(0, 0), B(0, 1), B(0, 2)
B(1, 0), B(1, 1), B(1, 2)
B(2, 0), B(2, 1), B(2, 2)の合計9個の配列要素を準備する(宣言する)。

一般型式

DIM 配列名 (数値) 1次元配列

DIM 配列名 (数値, 数値) 2次元配列

説明

- 配列要素という、変数に番号をつけたものの集まりを、指定個数ぶんRX-78に用意する(宣言するという)。
- 配列名は変数と同じように、英大文字ではじまり、英大文字と数字がつかえる。やはり変数と同じで、先頭2文字しか判別せず、ストリングをあつかう場合は\$マークをつける。
- ひとつの配列名について使用できる配列要素は、最大256個であり、宣言していない配列を使用しようとしたり、宣言してある配列でも配列要素として準備されていないものを使用しようとしたりするとエラーになる。ただし、宣言した配列を使用しなくてもエラーにはならない。(注)

1次元配列

- 配列で DIM A(5)と宣言すると、
A(0)、A(1)、A(2)、A(3)、A(4)、A(5)
合計6個の配列要素が使用可能になる。それぞれの配列要素は、ふつうの変数のように使うことができる。

2次元配列

- 配列で DIM A(2, 4)と宣言すると、
A(0,0)、A(0,1)、A(0,2)、A(0,3)、A(0,4)
A(1,0)、A(1,1)、A(1,2)、A(1,3)、A(1,4)
A(2,0)、A(2,1)、A(2,2)、A(2,3)、A(2,4)
合計15(3×5)個の配列要素が使用可能になる。それぞれの配列要素は、ふつうの変数と同じように使うことができる。

④配列要素は0番から始まることに注意しよう。

※ 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。

※ プログラムの容量によって、使用できる配列要素の個数が異なりますのでご注意ください。

```
10 REM ** DIM サンプル **
20 PRINT CHR$(6)
30 DIM A$(10,4) :A=177 :B=215
40 FOR I=0 TO 6
50 FOR J=0 TO 4
60 A$(I,J)=CHR$(A)
70 A=A+1
80 NEXT J,I
90 FOR K=0 TO 4
100 A$(8,K)=CHR$(B)
110 B=B+1
120 NEXT K
130 A$(7,0)="ヤ" :A$(7,2)="ユ"
140 A$(7,4)="ヨ" :A$(9,0)="ワ" :A$(10,0)
="ン"
150 FOR L=0 TO 10
160 FOR M=0 TO 4
170 PRINT A$(L,M); " ";
180 NEXT M :PRINT
190 NEXT L
200 END
```

アスキーコード表から五十音をぬき出して、行番号60で配列A\$に入れている。ヤユヨワンの5音は並びが不規則なので、行番号に130～140で特別に入れている。

