

変数も配列もすべてまっさら——CLEAR

使用例

CLEAR

変数、配列要素の内容を 0、空(NUL)
(NULL) にする。

一般型式

CLEAR

- 数値をとる変数および配列要素の内容をすべて 0、ストリングをとる変数および配列の内容をすべて空(NULL) にする。
- プログラムを途中から実行させたとき、変数の内容はそのまま残っているので、複数のプログラムをいっぺんに記憶している場合、それぞれのプログラムの先頭で変数の内容を 0 および(NULL) にする。

STEP 3 サンプルプログラム

```

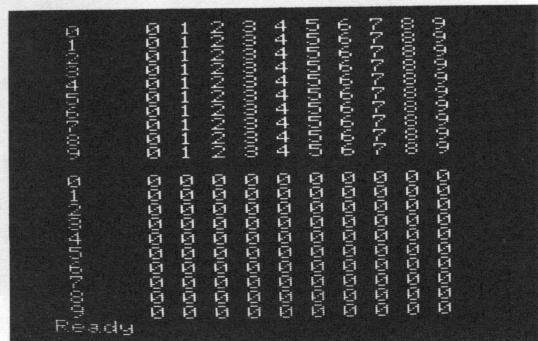
10 REM ** CLEAR サンプル **
20 PRINT CHR$(6)
30 DIM A(9,9)
40 FOR I=0 TO 9
50 FOR J=0 TO 9
60 A(I,J)=J
70 NEXT J,I
80 FOR K=0 TO 9
90 PRINT K;" ";
100 FOR L=0 TO 9
110 PRINT A(K,L);
120 NEXT L :PRINT
130 NEXT K
140 IF B=1 THEN 170
150 PRINT :CLEAR :DIM A(9,9) :B=1
160 GOTO 80
170 END

```

行番号40~70で配列Aに0~9までの数字を入れ、行番号100~130で、配列Aの内容を確かめている。

配列Aには確かに数字が入っていたのだが、行番号150のCLEAR命令によって配列Aの中身はすべて0になっている。

* 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 サンプルプログラム(まとめ)

```
10 REM ** セイセキ ラヨリ **  
20 PRINT CHR$(6)  
30 DIM A$(2) :DIM B$(4) :DIM C(4,2)  
40 FOR I=0 TO 2  
50 READ A$(I)  
60 NEXT I  
70 FOR J=0 TO 3  
80 INPUT "ナミ ?"; B$(J)  
90 NEXT J :B$(4)="ハイキン"  
  
100 FOR K=0 TO 3  
110 PRINT B$(K); " ノ テンスウ"  
120 FOR L=0 TO 1  
130 PRINT A$(L)  
140 INPUT C(K,L)  
150 C(K,2)=C(K,2)+C(K,L)  
160 C(4,L)=C(4,L)+C(K,L)  
170 C(4,2)=C(4,2)+C(K,L)  
180 NEXT L,K  
190 FOR M=0 TO 2  
200 C(4,M)=C(4,M)/4  
210 NEXT M  
220 REM ** カーメン ニ タス **  
230 PRINT CHR$(6)  
240 FOR N=0 TO 2  
250 D=D+7  
260 PRINT TAB(D);A$(N)  
270 NEXT N :PRINT :PRINT  
280 FOR O=0 TO 4  
290 PRINT B$(O)  
300 FOR P=0 TO 2  
310 E=E+7  
320 PRINT TAB(E);C(O,P)  
330 NEXT P  
340 PRINT :PRINT :E=0  
350 NEXT O  
360 END  
370 DATA ココ, サンスウ, コウケイ
```

注釈文
画面にでている内容をすべて消す。
配列宣言
I が 0 から 2 の間、行番号 60 までを繰り返す。
科目名のデータを読む。
行番号 40 のくり返し命令の終りを示す。
J が 0 から 3 の間、行番号 90 までを繰り返す。
名前を入力する。
行番号 70 の繰り返し命令の終りを示す。文字型配列 B \$ を "ハイキン" とする。
K が 0 から 3 の間、行番号 180 までを繰り返す。
名前と "ノ テンスウ" を画面に出す。
L が 0 から 1 の間、行番号 180 までを繰り返す。
科目名を画面にだす。
その名前のその科目名の点数を入力する。
名前別の合計点を計算する。
科目別の合計点を計算する。
2 科目 4 人の合計点を計算する。
行番号 100 と行番号 120 の繰り返し命令の終りを示す。
M が 0 から 2 の間、行番号 210 までを繰り返す。
科目別の平均点を計算する。
行番号 190 の繰り返し命令の終りを示す。
注釈文
画面にでている内容をすべて消す。
N が 0 から 2 の間、行番号 270 までを繰り返す。
画面にだす科目名の位置を計算する。
画面に科目名をだす。
行番号 240 の繰り返し命令の終りを示す。画面を 2 行分あける。
O が 0 から 4 の間、行番号 350 までを繰り返す。
名前とハイキンを画面にだす。
P が 0 から 2 の間、行番号 330 までを繰り返す。
画面にだす点数の位置を計算する。
画面に点数をだす。
行番号 300 の繰り返し命令の終りを示す。
画面を 2 行分あける。次に画面にだす位置を設定する。
行番号 280 の繰り返し命令の終りを示す。
プログラムの終りを示す。
科目名のデータ

〔解説〕

行番号 40~60 では、配列 A \$ に科目を読み込み、行番号 70~90 では、B \$ に名前を読み込んでいる。行番号 100~180 では、配列 C に点数を読み込んでいる。

また、行番号 150~170 は配列 C にそれぞれの合計をたし込んでおり、行番号 200 では、平均を求めている。

行番号 220 以降は配列の中身を画面に書き出しているのであるが、見やすく整理するために、すこし複雑になっている。

変数の変化をよく追いかけて、どの配列要素を使用しているかを正しくつかんでほしい。

	ココ	サンスウ	コウケイ
アキオ	79	84	163
セイコ	63	47	110
ナオコ	89	73	162
モモエ	46	88	126
ハイキン	69.25	71	140.25
Ready			

STEP 4 サブルーチンは指名打者

同じプログラムの中で、何度もくりかえし同じ作業が出てくることがある。それをいちいち打ち込んでいたら、大変な労力と時間のムダ使いである。

そんなとき、「ワンパターン」という感じでいつも登場してくれる助っ人、そうピンチヒッターがいたらどうだろう。同じことをやるのだったら、もうおまかせしちゃうというわけだ。

STEP 4 サンプルプログラム

```

10 REM ** セイクラへ フロク ラム ***
20 PRINT CHR$(6)-
30 DIM A$(9) :DIM A(9)-
40 FOR I=0 TO 9 -
50 READ A$(I),A(I)-
60 GOSUB 130 -
70 NEXT I -
80 FOR J=0 TO 9 -
90 PRINT J+1;"パン ";A$(J);A(J);"cm"-
100 PRINT -
110 NEXT J -
120 END -
130 REM ** ナラヘ カ工 サフルーチン ***
140 FOR K=0 TO 8 -
150 FOR L=K+1 TO 9 -
160 IF A(K)>=A(L) THEN 190 -
170 A=A(K) :A(K)=A(L) :A(L)=A -
180 A$=A$(K) :A$(K)=A$(L) :A$(L)=A$ -
190 NEXT L,K -
200 RETURN -
210 REM ** ナマエト シンチョウ ***
220 DATA ミユキ,152,マコト,168,テツオ,175,シノブ,1
54,ヨシル,170 -
230 DATA トモコ,157,マサミ,162,リツコ,156,ノリコ,1
1,タエコ,151 -

```

[解説]

行番号50で配列A\$、Aに読み込んでいる名前と身長を、行番号60で行番号130~200のサブルーチンで身長順に並べかえ、配列に入れなおしている。

このプログラムで使っている並べかえ（ソート：sort）をやるサブルーチンなどは、いろんなところで使えそうだ。これは大きい順に並ぶようにしてあるけど、小さい順に並べかえるには、どうやったらよいのだろうか。考えてみよう。

```

(63も)(アドバタイジング)
18 REM ブルーベリー
25 PRINT CHR$(6)-
38 DIM A$(2) :DIM B$(2)-
45 FOR I=0 TO 2 -
52 READ A$(I) -
58 NEXT I -
65 FOR J=0 TO 2 -
72 INPUT B$(J) -
85 NEXT J -
92 PRINT "ブルーベリー"
109 FOR K=0 TO 2 -
116 INPUT C$(K) -
123 NEXT K -
130 PRINT C$(0),C$(1),C$(2)
147 FOR L=0 TO 2 -
154 INPUT D$(L) -
161 NEXT L -
168 PRINT D$(0),D$(1),D$(2)
185 FOR M=0 TO 2 -
192 INPUT E$(M) -
200 NEXT M -
217 PRINT E$(0),E$(1),E$(2)
234 FOR N=0 TO 2 -
241 INPUT F$(N) -
248 NEXT N -
255 PRINT F$(0),F$(1),F$(2)
272 FOR O=0 TO 2 -
279 INPUT G$(O) -
286 NEXT O -
293 PRINT G$(0),G$(1),G$(2)
310 FOR P=0 TO 2 -
317 INPUT H$(P) -
324 NEXT P -
331 PRINT H$(0),H$(1),H$(2)
348 FOR Q=0 TO 2 -
355 INPUT I$(Q) -
362 NEXT Q -
369 PRINT I$(0),I$(1),I$(2)
386 FOR R=0 TO 2 -
393 INPUT J$(R) -
399 NEXT R -
406 PRINT J$(0),J$(1),J$(2)
423 FOR S=0 TO 2 -
430 INPUT K$(S) -
437 NEXT S -
444 PRINT K$(0),K$(1),K$(2)
461 FOR T=0 TO 2 -
468 INPUT L$(T) -
475 NEXT T -
482 PRINT L$(0),L$(1),L$(2)
499 FOR U=0 TO 2 -
506 INPUT M$(U) -
513 NEXT U -
520 PRINT M$(0),M$(1),M$(2)
537 FOR V=0 TO 2 -
544 INPUT N$(V) -
551 NEXT V -
558 PRINT N$(0),N$(1),N$(2)
575 FOR W=0 TO 2 -
582 INPUT O$(W) -
589 NEXT W -
596 PRINT O$(0),O$(1),O$(2)
613 FOR X=0 TO 2 -
620 INPUT P$(X) -
627 NEXT X -
634 PRINT P$(0),P$(1),P$(2)
651 FOR Y=0 TO 2 -
658 INPUT Q$(Y) -
665 NEXT Y -
672 PRINT Q$(0),Q$(1),Q$(2)
689 FOR Z=0 TO 2 -
696 INPUT R$(Z) -
703 NEXT Z -
710 PRINT R$(0),R$(1),R$(2)
727 FOR AA=0 TO 2 -
734 INPUT BB$(AA) -
741 NEXT AA -
748 PRINT BB$(0),BB$(1),BB$(2)
765 FOR CC=0 TO 2 -
772 INPUT DD$(CC) -
779 NEXT CC -
786 PRINT DD$(0),DD$(1),DD$(2)
793 FOR EE=0 TO 2 -
799 INPUT FF$(EE) -
806 NEXT EE -
813 PRINT FF$(0),FF$(1),FF$(2)
830 FOR GG=0 TO 2 -
837 INPUT HH$(GG) -
844 NEXT GG -
851 PRINT HH$(0),HH$(1),HH$(2)
868 FOR II=0 TO 2 -
875 INPUT JJ$(II) -
882 NEXT II -
889 PRINT JJ$(0),JJ$(1),JJ$(2)
896 FOR KK=0 TO 2 -
903 INPUT LL$(KK) -
910 NEXT KK -
917 PRINT LL$(0),LL$(1),LL$(2)
924 FOR MM=0 TO 2 -
931 INPUT NN$(MM) -
938 NEXT MM -
945 PRINT NN$(0),NN$(1),NN$(2)
952 FOR OO=0 TO 2 -
959 INPUT PP$(OO) -
966 NEXT OO -
973 PRINT PP$(0),PP$(1),PP$(2)
980 FOR QQ=0 TO 2 -
987 INPUT RR$(QQ) -
994 NEXT QQ -
999 PRINT RR$(0),RR$(1),RR$(2)

```

注釈文

画面に出ている内容をすべて消す。

配列A\$と、Aを宣言する。

Iの値を0から9の間、行番号40~70を繰り返す。

データを読み込む。

サブルーチン130へジャンプする。

行番号40の繰り返しの終りを示す。

Jの値を0から9の間、行番号80~110を繰り返す。

J+1の内容、「パン」、A\$(J)、A(J)の内容、「cm」と画面に出す。

行をかえる。

行番号80の繰り返しの終りを示す。

プログラムの終りを示す。

注釈文（サブルーチン）

Kの値を0から8の間、行番号140~190を繰り返す。

Lの値をK+1から9の間、行番号150~190を繰り返す。

A(K)≥A(L)なら、行番号190へとぶ。

AにA(K)、A(K)にA(L)、A(L)にAを代入する。

A\$にA\$(K)、A\$(K)にA\$(L)、A\$(L)にA\$を代入する。

行番号140の繰り返しの終りを示す。

行番号60のGOSUB命令の次へリターンする。

注釈文

名前と身長のデータ。

名前と身長のデータ。

1 パン テツオ 175cm
2 パン ヨシル 170cm
3 パン マコト 168cm
4 パン マサミ 162cm
5 パン ノリコ 161cm
6 パン トモコ 157cm
7 パン リツコ 156cm
8 パン シノブ 154cm
9 パン ミユキ 152cm
10 パン タエコ 151cm

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。

サブルーチンについて

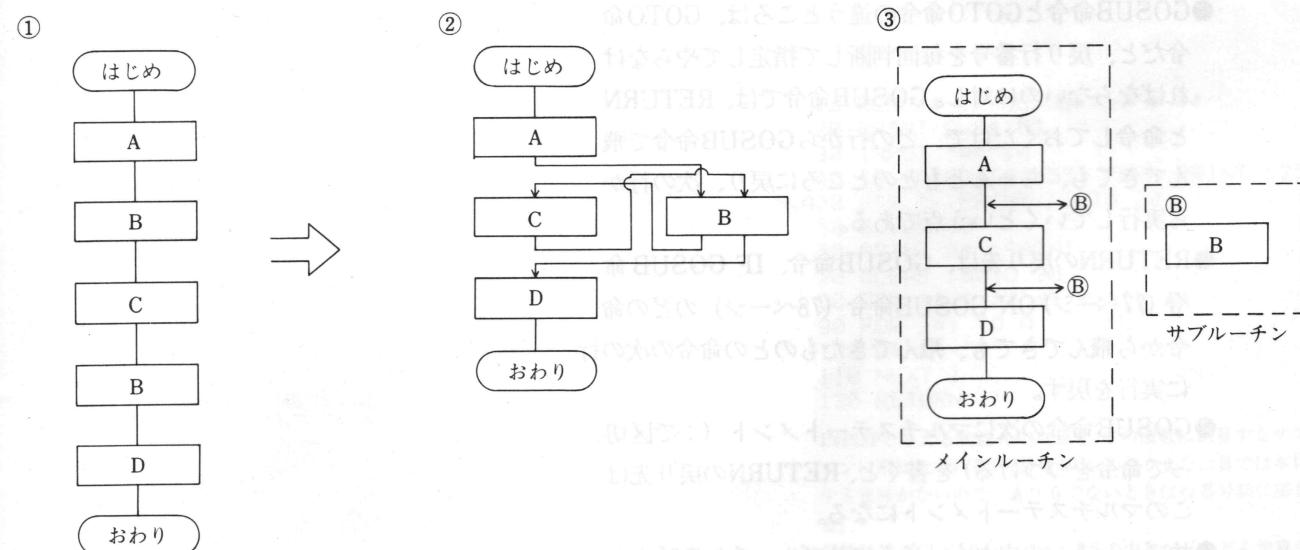
プログラムの中で、何度も出てくるパターンというのを1ヶ所にまとめてしまうという発想を具体的にしてみよう。

下の図で、プログラムの流れを考えてみる。①の流れを見るとわかるように、Bという処理が2箇所に組み込まれている。この部分がムダなところだ。

そこで、このプログラムの組み立てを②のようにしてみた。ムダな部分がなくなっていることがわかる。矢印がからみあっていて、わかりにくいくらいのプログラムになってしまったように見えるが、③のように描いてしまうと非常にスッキリする。

このように、プログラムには流れを決定する幹の部分と、その流れにくついている葉のような部分がある。そして、幹の部分にあたる部分をメインルーチン、葉にあたる部分をサブルーチンと呼んでいる。

プログラムをこのように、メインルーチン、サブルーチンに整理すると、プログラムが見やすくなるばかりでなく、修正が簡単に行なえるといい点がある。



行ってきます——GOSUB…RETURN

使用例

① GOSUB 100

行番号 100 から始まるサブルーチンに実行を移す。

② RETURN

サブルーチンに実行を移す命令を行なった GOSUB 命令の次の行に実行を戻す。

一般型式

**GOSUB サブルーチン先頭行番号
RETURN**

説明

- GOSUB 文で行番号を指定すると、その行番号に実行が移り、RETURN を実行するともとのところに戻り次の行からプログラムを続行していく。
- GOSUB 命令と GOTO 命令の違うところは、GOTO 命令だと、戻り行番号を毎回判断して指定してやらなければならぬのに対し、GOSUB 命令では、RETURN と命令しておくだけで、どの行から GOSUB 命令で飛んでも、ちゃんととのところに戻り、次の行から実行していくという点である。
- RETURN の戻り先は、GOSUB 命令、IF GOSUB 命令 (77 ページ) ON GOSUB 命令 (78 ページ) のどの命令から飛んでも、飛んできたものとの命令の次の行に実行を戻す。
- GOSUB 命令の後にマルチステートメント (: で区切って命令をつづける) を書くと、RETURN の戻り先はこのマルチステートメントになる。
- サブルーチンの中から、さらにサブルーチンを呼ぶことができる。このネスティングは最大 15 まで可能である。

④ サブルーチンは END (あるいは STOP) よりも後ろに書かなければ、そのままサブルーチンを実行してしまい、RETURN を実行した時点で *Error 14 になる。

```
①
LIST REM ** GOSUBサンプル **
20 PRINT "CHR$(5)" : B=1 : C=-5
30 A=10 : B=1 : C=1
40 GOSUB 120
50 PRINT "A*B=";A
60 PRINT "B*C=";B
70 A=A+1 : B=B+1
80 GOSUB 120
90 PRINT "(A*B)+B=";A
100 PRINT "(B*C)/C=";B
110 END
120 A1=A+B : B1=B/C
130 RETURN
Ready
```

行番号 120 の数式を行番号 40, 80 で 2 度使っている。この程度の式だったら、直接書いてもよさそうだが、めんどうな式になると間違いも多くなる。それに、式の変更が 1 箇所ですむので、修正のことも考えると、はるかに合理的であるのがわかる。

行つてきます IF...GOSUB (RETURN)

使用例

IF A=10 GOSUB 500

変数Aの値が10であったならば、
行番号500からのサブルーチンに
実行を移す。

一般型式 **IF** 条件文 **GOSUB** サブルーチン先頭
 行番号

説明 ●IF文の使用方法とまったく同じで、条件文に合えばCOSUP命令を実行する。

- GOSUBの実行は、GOSUBだけの命令と同じである(76ページ)。

```
10 REM ** IF GOSUB サンプル **
20 PRINT CHR$(6)
30 INPUT "A="; A
40 IF (A<0)+(A>255) THEN PRINT "255>=A,
A>0 テ'ス" :PRINT :GOTO 30
50 GOSUB 80
60 PRINT A$ :PRINT
70 CLEAR :GOTO 30
80 REM ** Aノカヌ'タ'ケキヲタ'ス **
90 FOR I=1 TO A
100 A$=A$+"*"
110 NEXT I
120 RETURN
```

INPUTで与える変数Aの数値を、＊の個数に換算するサブルーチンに飛ばしている(行番号50)。Aが0または負では＊に換算する意味がないので、A > 0でないときは行番号30に戻している。

行番号70のCLEARがないと、A＄の中身がどんどん加算され常に合計の個数が入っていることになってしまう。逆にこれを利用すれば、入力された数、合計、平均などというのも簡単に求められるだろう。

```

H=3
+**+
H=10
*****+
H=25
*****+
H=17
*****+
H=11

```

* 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。

どちらへ? ON...GOSUB(RETURN)

使用例

ON A GOSUB 100, 200, 300

変数Aの値が1のときは行番号100、
2のときは行番号200、3のときは
行番号300のそれぞれのサブルー
チンにとぶ。変数Aの値が0以下、
あるいは3より大きいとき、次の
行へ実行をつづける。

一般型式

ON 变数 GOSUB サブルーチン先頭行

番号, サブルーチン先頭行番号, ……

説明

- 変数は0または正の整数をとる数値変数であり(つまり、0、1、2、3、…), その数字によってGOSUB命令のとび先行番号を決定する。
- 変数の値が1のときは1番目の行番号へ、2のときは2番目の行番号へ…, と行き先が決まり、0以下もしくは行き先の個数より大きい数値のときは、次の行へと実行を進める。
- GOSUB命令のあとに指定できる行番号の個数は表示画面の3行を越えない範囲ならいくつでもよい。
- GOSUB命令の実行は、GOSUBだけの命令と同じである(76ページ)。

```
10 REM ** ON GOSUB サンプル **
20 PRINT CHR$(6)
30 B1=0 :B2=10
40 DIM A$(10) :DIM B$(10)
50 FOR I=0 TO 10
60 INPUT "ナマエ ?"; A$(I)
70 INPUT "オトコ-1 オンナ-2 "; B$
80 ON A GOSUB 150,180
90 NEXT I
100 PRINT CHR$(6)
110 FOR J=0 TO 10
120 PRINT B$(J)
130 NEXT J
140 END
150 REM ** オトコ-1 **
160 B$(B1)="オトコ "+A$(I)
170 B1=B1+1 :RETURN
180 REM ** オンナ-2 **
190 B$(B2)="オンナ "+A$(I)
200 B2=B2-1 :RETURN
```

行番号150～170のサブルーチンと行番号180～200のサブルーチンに、男か女かによって行番号80で振り分けている。
男は配列B\$(0)、B\$(1)、…という方向から、女は配列B\$(10)、B\$(9)、…という逆方向から入れている点に注意しよう。

オトコ	イトウ マサヨシ
オトコ	ツツラヤ ヒデアキ
オトコ	カガハワ セイシ
オトコ	ヤノココウイチ
オトコ	フジハタ タロウ
オトコ	ハシモト シンシ
オンナ	コハニヤシマキ
オンナ	オオダアキミコ
オンナ	タクミヨウコ
オンナ	イシイ クニコ
オンナ	マツミヤ チヅコ

Ready

STEP 4 サンプルプログラム(まとめ)

```

10 REM ** ニシ ホウテイシキ ノ カイ ***
20 PRINT CHR$(6)-
30 PRINT "ニシ ホウテイシキ ax^2+bx+c=0 ノ カイ"
40 INPUT "a=";A : INPUT "b=";B : INPUT "c"
   ;C
50 IF A=0 GOSUB 130 : GOTO 30
60 D=B^2-4*A*C-
70 IF D<0 GOSUB 160 : GOTO 30
80 X1=(-B+SQR(D))/2/A-
90 X2=(-B-SQR(D))/2/A-
100 PRINT " X=";X1;
110 IF D>0 THEN PRINT ", X=";X2
120 GOTO 30
130 IF B=0 THEN PRINT "フティ !": RETURN
140 PRINT " X=";-C/B-
150 RETURN
160 R=-B/2/A : I=SQR(ABS(D))/2/A-
170 PRINT " X=";R;"+";I;" !": PRINT "
X=";R;" -";I;" !"
180 RETURN

```

注釈文

画面に出てる内容をすべて消す。

「2次方程式、 $ax^2+bx+c=0$ の解」と画面に出す。

A、B、Cを入力する。

Aの値が0なら、サブルーチン130へジャンプする。行番号30へとぶ。

Dの値を計算する。

Dが0より小さい時、サブルーチン160へジャンプする。行番号30へとぶ。

X1の値を計算する。

X2の値を計算する。

「X=」と、X1の内容を画面に出す。

Dが0より大きい時「X=」とX2の内容を画面に出す。行番号30へとぶ。

B=0なら「フティ」と画面に出し、行番号50のGOSOBの次の命令へリターンする。

「X=」と、-C/Bの計算結果を画面に出す。

行番号50のGOSOBの次の命令へリターンする。

Rの値と、Iの値を計算する。

「X=」と、Rの内容、「+」とIの内容、「!」と「X=」とRの内容、「-」とIの内容、「!」を画面に出す。

行番号70のGOSOBの次の命令へリターンする。

〔解説〕

二次方程式、 $ax^2+bx+c=0$ の解を、係数 a、b、cの場合わけによって求めている。

a=0の場合、行番号130、140のサブルーチン。実数解の場合、行番号80~120。虚数解の場合、行番号160~180のサブルーチンで求めている。

二次方程式というのは中学で学習する内容だから、習っていない人は、それまでのお楽しみというわけだ。プログラムの意味がわからなくても、がっかりすることはない。

```

ニシ ホウテイシキ ax^2+bx+c=0 ノ カイ
a=1
b=5
c=6
ニシ ホウテイシキ ax^2+bx+c=0 ノ カイ
a=1
b=2
c=3
X=-1+ 1.4142136!
ニシ ホウテイシキ ax^2+bx+c=0 ノ カイ
a=1
b=9
c=3
X= 8.6533119 ; X= 0.34668807
ニシ ホウテイシキ ax^2+bx+c=0 ノ カイ
a=■

```

※ 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。
※ べき乗「^」はキーボード上では↑になります。

STEP 5 言葉で遊べ〈文字関数〉

変数には文字変数と数値変数があって、今までそれらを使っていろいろなプログラムを作ってきた。しかし、数値変数には数値関数というのがあっていろいろと便利なのだけれど、文字変数はなにかと不便をしていないだろうか。

最近よくOA(オフィスオートメーション)がどうのこうのというテレビのコマーシャルで、ワードプロセッサ(ワープロ)というのを耳にするが、あれは文字を処理するというコンピュータだ。

RX-78でも文字をあれこれといじくれたら便利にちがいない。「文書処理」なんてカタイこと言わなくとも、ゲームなんかでも使ってみたいもんだ。

実はRX-78にもそんな能力がある。それは文字関数というやつだ。ワープロとまではいかなくとも、十分君の期待に応えてくれるはずだ。

STEP 5 サンプルプログラム

```
10 REM ** オオモシア コモシア **
20 PRINT CHR$(6) -----
30 INPUT "アルファベット ノ オオモシア ヲ イレル "; A$ -----
40 L=LEN(A$) -----
50 FOR J=1' TO L -----
60 B$=MID$(A$, J, 1) -----
70 C=ASC(B$)+32 -----
80 PRINT CHR$(C); -----
90 NEXT J -----
100 END -----
```

注释文

画面に出ている内容をすべて消す。

「アルファベットの大文字を入れる」と画面に出し、A \$ を入力する。

A \$のストリングの長さをLに代入する。

Jの値が1からLの間、行番号50～90を繰り返す。

A \$ のストリ

に代入する。

B\$に入っている文字をアスキーコード

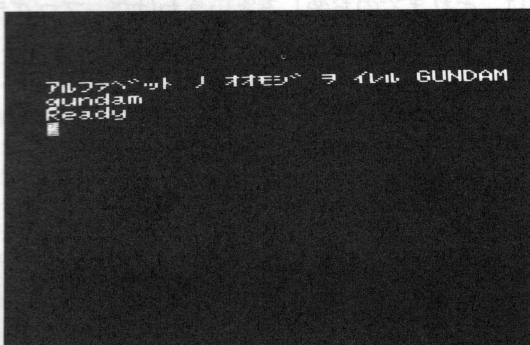
えCに代入する。

アスキーコードに対応する
行番号50の繰り返しの終

[解 説]

英大文字を英小文字にかえるというプログラムだ。プログラムとしては短いのだが、はじめて見る命令がいくつある。

このプログラムでは、ストリングの長さを求めたり、ストリングから文字を1文字ずつとり出したりしている。また、文字をアスキーコードに直すというような、ちょっと難しいこともやっているのだ。



* 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。

文字関数——LEFT\$,RIGHT\$,MID\$

使用例

① **AL\$=LEFT\$(A\$, 5)**

文字変数A \$に入っているストリ
ングの、左から5文字をとりだす。

② **BR\$=RIGHT\$(B\$, 4)**

文字変数B \$に入っているストリ
ングの、右から4文字をとりだす。

③ **CM\$=MID\$(C\$, 3, 5)**

文字変数C \$に入っているストリ
ングの、左から3文字目から5文
字をとりだす。

一般型式

LEFT\$ (文字変数, 文字数)

RIGHT\$ (文字変数, 文字数)

MID\$ (文字変数, 何文字目, 文字数)

説明

- 文字変数に入っているストリングから、好きなところを、好きな文字数分だけとりだす文字関数である。
- とりだす文字数が、ストリングの文字数より多い時は、
→ストリングをそのまま表示。
- とりだす文字数が0の時は、→何も表示しない。
- とりだす文字数が-(マイナス)の時は、
→※Error 3

LEFT\$

- ストリングの左から指定文字数だけとりだす。

(例) A\$ = " ABCDEFG " → B\$ = " HIJK " のとき

① LEFT\$ (A\$, 3) → " ABC "

② LEFT\$ (A\$ + B\$, 9) → " ABCDEFGHI "

RIGHT\$

- ストリングの右から指定文字数だけとりだす。

(例) A\$ = " ABCDEFG ", B\$ = " HIJK " のとき

① RIGHT\$ (A\$, 3) → " EFG "

② RIGHT\$ (A\$ + B\$, 3) → " IJK "

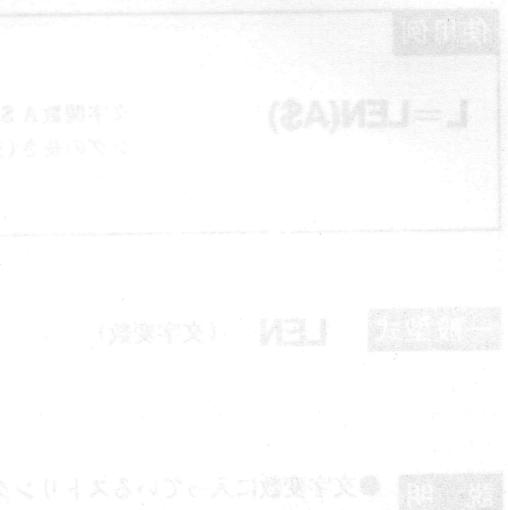
MID\$

- ストリングの指定文字数目から指定文字数だけとりだす。

(例) A\$ = " ABCDEFG ", B\$ = " HIJK "

① MID\$ (A\$, 3, 4) → " CDEF "

② MID\$ (A\$ + B\$, 5, 4) → " EFGH "



```
LIST REM *** モード カンパル 1 ***
20 N$="ミコキ 12カッ20ニチ ハソコン"
30 N$=LEFT$(N$,3)
40 B$=MID$(N$,5,5)
50 H$=RIGHT$(N$,5)
60 PRINT "ナマエ-----";H$
70 PRINT "タンショウヒ-----";B$
80 PRINT "ジュミ-----";N$
90 END
Ready
RUN
ナマエ-----ミコキ
タンショウヒ-----12カッ20ニチ
ジュミ-----ハソコン
Ready
```

行番号20でA \$に入れたストリングを、行番号30~50の文字
数で部分的にN \$、B \$、H \$に入れ、行番号60~80で画面に
だした。

STRING の長さは?

LEN

使用例

L=LEN(A\$)

文字関数 A\$ に入っているストリ
ングの長さ(文字数)をもとめる。

一般型式

LEN (文字変数)

説明 ● 文字変数に入っているストリングの長さ(文字数)を
もとめる。

(例) A\$ = "ABCDEFG", B\$ = "HIJK" のとき

① LEN (A\$) → 7

② LEN (A\$+B\$) → 11

STEP 3

```
10 REM モンスター
20 PRINT CHR$(6)
30 INPUT "名前を入力"
40 L=LEN(N$)
50 FOR I=1 TO L
60 B$=MID$(N$,I,1)
70 C$=ASC(B$)-32
80 PRINT CHR$(C$)
90 NEXT I
100 END
```

```
10 REM ** キャンプル 2 **
20 PRINT CHR$(6)
30 INPUT "アナタノナミ?"; N$
40 L=LEN(N$)
50 GOSUB 100
60 PRINT N$
70 GOSUB 100
80 END
90 REM ** +ヲカクサフルーチン**
100 FOR I=1 TO L
110 PRINT "+";
120 NEXT I
130 PRINT
140 RETURN
```

画面にだす名前の長さに合わせて + をだすために、行番号40で
名前の長さをもとめた。

+は行番号100~130のサブルーチンでだすのだが、行番号110を
何回実行するかを、Lつまり名前の長さで決めている。

④などを好きな個数だけだすには、もっといい方法がある。
それは、STRING\$ (83ページ) を使うのである。

どんなふうにプログラムを修正すればよいか考えてみよう。

```
アナタノナミ? RX-78 GUNDAM
+++++
RX-78 GUNDAM
+++++
Ready
```

同じ文字なら—SPACES,STRINGS

使用例

- ① **PRINT SPACE\$(5)**
空白（スペース）5個のストリングを作る。
- ② **ST\$=STRING\$("＊", 5)**
＊印5個のストリングを作る。

一般型式 **SPACE\$** (空白の個数)
STRING\$ (ストリング、個数)

説明

- 文字変数や文字関数において、空白（スペース）や同じ記号の繰り返しというのは重要な役割をもっている。SPACE\$は空白を、STRING\$は記号や文字を、指定した個数だけ作る関数である。

SPACE\$

- 指定した個数だけの空白（スペース）のストリングを作る。
- ストリングの長さをそろえたり、画面に文字を出すときに右づめにしたりするときによく使う。

STRING\$

- ストリングで与えた記号や文字を、指定個数だけ連続したストリングを作る。
- (例) ① STRING\$ ("＊", 5) → "＊＊＊＊＊"

```

10 REM ** モンクサンプル 3 ***
20 PRINT CHR$(6)
30 L$=STRING$("-", 12)
40 PRINT TAB(8);L$
50 FOR I=1 TO 10
60 READ A$,B$
70 SP=12-LEN(A$+B$)
80 PRINT TAB(8);A$;SPACE$(SP);B$
90 PRINT TAB(8);L$
100 NEXT I
110 END
120 DATA オオシケ, ミユキ, タキヲ, マコト, ナカサワ, テツオ,
アラマキ, トモコ, オノ, ヨウコ
130 DATA ハシモト, ノフュキ, コントウ, マサオ, アラキ, タツヒコ,
クリヤマ, ハリヒコ, タナカ, ハシメ

```

READ文で読み込んだ名前を、行番号70で長さを調べて、画面に出すときに左右のはしがそろのように、苗字と名前の間を空白で埋めた。

また、名前と名前の間の横棒は、行番号30の STRING\$ で作ったストリングである。

オオシケ	ミユキ
タキヲ	マコト
ナカサワ	テツオ
アラマキ	トモコ
オノ	ヨウコ
ハシモト	ノフュキ
コントウ	マサオ
アラキ	タツヒコ
クリヤマ	ハリヒコ
タナカ	ハシメ

Ready

アスキーコード表を使え—CHRS\$, ASC

使用例

① **PRINT CHR\$(71)** (10進)アスキーコードで71の文字、つまりGの文字をとり出し、画面にだす。

① **PRINT ASC("G")** Gの文字の(10進)アスキーコード
つまり71を取り出し、画面にだす。

一般型式 **CHR\$** (アスキーコード)
ASC (ストリング)

說明

- 文字から（10進）アスキーコードに、またはその逆を行なう関数である。
 - RX-78 は、文字に番号をつけて記憶しているのだが、その番号がアスキーコードと呼ばれるものだ。コードの表わし方には10進数とか16進数とかいうのがあるのだが、CHR\$、ASCでは、10進数が使える。（CHR\$は16進数も使えます。）文字とアスキーコードとの対応表は158ページにある。

CHR \$

- アスキーコードから、そのコードに対応している文字をとり出す関数である。
 - PRINT CHR\$(変数) という形でCHR\$を表示したとき、アスキーコードが0~15のコードだった場合、コードに対応する文字が画面に出るのではなく、特殊な働きをする。

ASC

- 文字に対応する ASCII コードをとりだす関数である。
 - ストリングは文字変数で与えてもよいが、1 文字より長いストリングの場合は、先頭の文字についてしか実行しない。

```

LIST
100 REM *** エンカンズウ サンフル 5 ***
110 PRINT CHR$(6)
120 FOR I=16 TO 255
130 PRINT CHR$(I);
140 NEXT I
150 PRINT CHR$(13)
160 INPUT "トランゼ?"; A$
170 L=LEN(A$)
180 PRINT A$; "=";
190 FOR J=1 TO L
200 B$=MID$(A$,J,1)
210 PRINT ASC(B$);
220 NEXT J
230 END
Readme

```

行番号20、行番号60のPRINT CHR\$(変数)命令は特殊な使いかたである。特に行番号60のほうのPRINT CHR\$(13)の改行という働きについては、よく考えておこう。

行番号40のPRINT CHR\$の使い方は、普通のPRINT命令と同じだ。つまり、アスキーコードの16~255に対応する文字が順に画面に出てくるわけだ。

行番号70～130はASC関数を使って、文字をアスキーコードにかえるというプログラムになっている。ASC関数では、ストリングの先頭の1文字しか実行しないので、ストリングの長さを行番号80で求め、行番号110で1文字ずつとり出している。

STEP 5 サンプルプログラム(まとめ)

```

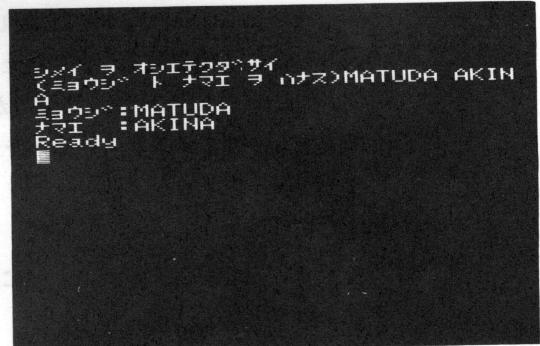
10 REM ** ナマエ ノ セイリ ***
20 PRINT CHR$(6) : A=1
30 PRINT "シメイ ヲ オシエテクタサイ"
40 INPUT "(ミヨウシ" ト ナマエ ヲ ハナス]" ; A$
50 L=LEN(A$)
60 B$=MID$(A$,A,1)
70 IF B$="" THEN 100
80 A=A+1
90 GOTO 60
100 B=L-A
110 PRINT "ミヨウシ：" ; LEFT$(A$,A)
120 PRINT "ナマエ：" ; RIGHT$(A$,B)
130 END

```

注釈文
 画面に出ている内容をすべて消す。Aを初期設定する。
 「氏名を教えてください」と画面に出す。
 「苗字と名前を離す」と画面に出し、A\$を入力する。
 A\$のストリングの長さを求める。
 A\$のストリングの第A文字目から1文字取り出し、B\$へ代入する。
 B\$が空白なら行番号100へとぶ。
 Aを1増やす。
 行番号60へとぶ。
 Bの値を計算する。
 「苗字」と、A\$のストリングの左からA文字まで画面に出す。
 「名前」と、A\$のストリングの右からB文字まで画面に出す。
 プログラムの終りを示す。

〔解説〕

行番号40でRX-78に入力した君の氏名の、苗字と名前の区切りを行番号50~90で見つけている。
 行番号110、120では、その区切りで氏名を分け、2行にして出している。
 文字関数LEN、MID\$、LEFT\$、RIGHT\$がどんなふうに君の氏名、つまりA\$の中身を分析しているか、よく考えてみよう。



```

シメイ ヲ オシエテクタサイ
(ミヨウシ" ト ナマエ ヲ ハナス)MATUDA AKIN
A
ミヨウシ：MATUDA
ナマエ：AKINA
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。