

I/O

ています。

⑦BYE

モニタへジャンプします。

コールド・スタート	1300H
ホット・スタート	132AH

エディタ

エディタは、カーソル・エディットができる簡単なエディタです。画面上にリストが残っていれば、いつでも修正可能です。

メモリ内イメージは、シャープのTEXTエディタと同じく、CRで区切られた文字列です。カセットの記録フォーマットも同じなので、シャープのテキスト・エディタでソースを作って、コンパイルしてもかまいません。

コンパイラからエディタへジャンプします。

EDIT	CR
# ■	

これによりエディタ・モードを示すために#を出力して、入力待ちになります。

エディタ・コマンドは以下の8個です。

&...NEW D...DELETE
I...INSERT !...COMPILER
B...BREAK R...READ FILE
L...LIST W...WRITE FILE

①Insert

Iコマンドは、テキストをインサートするときに使います。このコマンドは常にテキスト・エンドへインサートします。

[SHIFT] + [BREAK] を押すことによって、コマンド・レベルへ戻ります。

#I	CR
■	

②Break

Bコマンドは、行間に1行インサートしたいときに使います。

Bn	■
----	---

nで指定した番号の行は1行下にさがり、その上にインサートされます。

#Bn	CR
Strings	CR
# ■	

nは、LISTしたときにエディタが付ける番号です。

③List.

Lコマンドはインサートしたテキストをリストします。次の3つの型があります。

#L	CR
#L, n	CR
#L, P	CR

●L

テキストの先頭からCRTに表示します。 [SHIFT]

[BREAK] によって途中で止めることができます。

●L, n

nはLISTしたときに、エディタが付ける番号で、n番以降のリストを行ないます。LISTを止めるのはLと同じです。

●L, P

プリンタにLISTを出力します。ただし、n番以降のLIST出力はできません。止める場合はLと同じです。

LISTの画面は次のようになっています。

#L	CR
0 : Strings	
1 : Strings	
2 : Strings	
:	
# ■	

左側の番号がエディタの付けた番号で、Stringsはインサートしたテキストです。番号とテキストの間に『:』が入っています。

④Delete

Dコマンドは、1行デリートします。

#Dn	CR
-----	----

nで指定したテキストが抹消されます。

nはエディタの付けた番号です。

⑤Compile

!コマンドはエディタ・モードを抜け、コンパイラ・モードへ移る場合に使います。

#!	CR
OK.	
■	

⑥New

&コマンドはテキスト・エリアをクリアし、初期化します。

#&	CR
# ■	

⑦Read, Write

R, Wコマンドはカセットに対しての入出力用です。これによってテキスト・レベルでプログラムの保存ができます。

●R

テキストを読み込みます。

#R [FILE NAME]	CR
----------------	----

ファイル名は省略可能です。

●W

メモリにあるテキストをカセットに落します。

#W [FILE NAME]	CR
----------------	----

ファイル名は省略可能です。

⑧カーソル移動キー

画面にリストが残っていれば、BASICと同じようにカーソルを動かして修正が可能です。ただし、Iコマンド実行中はできません。

カーソル・エディットするときは、エディタが付けた番号と“:”を消さないでください。これがなくなるとエディタはどのテキストを修正したかがわからなくなり、コマンド待ちの状態へ戻ります。

[CR] はどこで押してもかまいませんが、その行の下に

テキストがある場合には、 ":" の後にカーソルがなければ、 # を出力してカーソルが点滅し、入力待ちになります。このエディタは # がないとコマンドを受け付けられません。

FORTRAN-MZBの文法

1) プログラムの構成

ソース・プログラムは図 1 のような構成になります。宣言文は省略可能です。END 文はプログラムの最後に必要です。コンパイラはこれをみて、コンパイルを終了します。

各々の文は 1 行（1 行は 80 字以内）に書かれ、継続はできません。文の 1 行目に空白以外の文字があると、その文は注釈文になります。

文番号は必要な文にだけ、1 ~ 65,535 の任意の数を割り当てるだけで良く、BASIC のようにすべての文に順に割り当てる必要はありません。

図 1 ソース・プログラムの構成



2) 主プログラムと副プログラム

主プログラムと副プログラム（サブルーチン）はプログラム単位として区別されるのではなく、両者は 1 つのプログラムの中にあります。変数名は両者で共用されます。

3) 文の要素

①定数

a) 整数型定数

-32,767 ~ +32,767 の 10 進数、あるいは \$0000 ~ \$FFF の 16 進数

【例】 123, -65, \$0F, \$A12

b) 実数型定数

$1.67 \times 10^{-19} \sim 1.67 \times 10^{18}$ の任意の数で有効桁数は 7 桁

【例】 1.0, -1.23, 1.5E+12, -0.2E-5

c) 文字型定数

その文字の ASCII コードが数値になる。

【例】 #A, #B

d) π

π は 3.14159 と書くのと同じことを意味する。

②変数

変数名には 1 ~ 4 文字の一連の英数字を使います。最初の文字は必ず英字であることが必要です。変数名の最初の文字が I, J, K, L, M, N のいずれかである場合は整数型の変数名、それ以外の英字である場合は実数型の変数名として区別されます。予約語は変数名として使われません。

③配列

配列は DIMENSION 文で宣言します。配列名の付け方は変数の場合と同じで、やはり整数型と実数型に区別されます。

【例】 DIMENSION A(M) ただし、 $1 \leq M \leq 2,047$

DIMENSION K(M, N) ただし、 $1 \leq M \leq 255$
 $1 \leq N \leq 255$
 $1 \leq (M \times N) \leq 2,047$

④配列の添字

添字は、整定数、整変数、整数型算術式のいずれでもかまいません。

4) 算術式

算術演算子には、

1) \wedge (ベキ乗)

2) *, / (乗算、除算)

3) +, - (加算、減算)

の 5 種類があります。1), 2), 3) の順に優先度が高く、1 つの算術式の中に多種類が使われた場合には、優先度の高いものから順に実行され、同一順位の演算子は左から右に実行されます。

ベキ乗には $J \wedge I$, $A \wedge I$, $A \wedge B$ の 3 種類の型があります。

J \wedge I 型	J, I ともに整数型で、結果も整数型です。 $I \geq 0$ です。I = 0 のとき、結果は 1 です。
A \wedge I 型	A は実数型、I は整数型 (I は必ず整数型変数で、"2", "3" のような数を使うと次の A \uparrow B 型になる) で結果は実数型です。 $-128 \leq I \leq 127$ に制限されます。
A \wedge B 型	A, B ともに実数型で、結果も実数型です。ただし、 $A > 0$ 。

$A \wedge I$ 形は A を I 回加算する方法、 $A \wedge B$ 型は自然対数を使って計算する方法がとられるので、通常は $A \wedge I$ 型の方が格段に速く計算できます。上にも書きましたが、 $A \wedge I$ で計算を行ないたいときは、必ず I に整数型を使わなければなりません。

5) 代入文

一般形は、

$$v = al$$

で、右辺での演算結果を左辺の変数に代入します。左辺の v は変数、配列、メモリ関数、I/O 関数のいずれかです。

変数名および配列名が整数形の場合には右辺の演算は整数型演算、実数型の場合には右辺は実数型演算として処理されます。

6) 制御文

①GOTO文

GOTO x

x は文番号。

②IF文

一般形は次のとおりです。

IF (al) x, y, z

() の中に書かれた算術式 al の演算結果の負、零、正に基づいて、それぞれ、文番号 x, y, z にジャンプします。

算術式は "(" の直後の文字が、A, B, …, H, O, P, …, Z の場合だけ実数型演算として処理され、それ以外の場合には整数型演算とみなされます。また、() の中に論理関数が使われた場合は整数型演算と同じです。

x, y, z のいずれも省略可能です。省略された場合に

I/O

は次の文にジャンプします。

③DO文

一般形は、

DO *x v = n, m, l*

x はループの末端文の文番号, *v* は整変数です。*n* は開始値, *m* はループの終了値, *l* は増分のパラメータです。*l*, *m*, *n* は整定数, 整変数, 整数型算術式が可能です。*l* が 1 の場合は省略できます。

l ≥ 1 , *m* $\geq n$ ≥ 1

- a) DO文は 6 重までネスティングが可能です。
- b) DO文の範囲から外へ飛び出せるが, 外からは飛び込めません。

④CONTINUE文

この文はプログラム中, どこにあっても何も実行しませんが, DOの末端文, GOTO, IFなどの飛び先としても使えます。

⑤CALL文

この文は指定文番号を, 副プログラムの先頭とみなしてコールします。変数は主プログラムと共にします。

⑥RETURN文

CALL文で呼ばれた副プログラムの最後に書き, この文が実行されると CALL文の次の文へ戻ります。

⑦PAUSE文

この文を実行するとプログラムはそこで一時停止します。この状態でキーを押すと, 次の文から実行を再開します。

例 PAUSE *n*

n はプログラマーが任意に付ける数字です。この文が実行されると『PAUSE *n*』が出力され, 一時停止します。*n* はどこの PAUSE 文かを知らせるための番号であるため, 任意に付けてかまいません。必要ななれば *n* は省略可能です。

また, 停止した状態では, [BREAK] キーが受け付けられます。

⑧STOP文

この文はプログラムの実行を停止させるときに使います。STOP 文はプログラム中どこに入れてもかまいません。

例 STOP *n*

n の関係は PAUSE 文と同じです。

⑨BREAK文

[BREAK] キーのチェックのための文です。[BREAK] キーが押されなければ, 何も実行しません。この文は末端文としても使用可能である。

⑩USR文

この文はマシン語とリンクするときに使用します。

例 USR (exp)

exp で指定されるアドレスをサブルーチン・コールします。各レジスタ (AF, BC, DE, HL) はプログラム中で決めることができ, RETURN したときのレジスタへの受け渡しはメモリを介して行なうので, MEM 関数によってセットします。

各レジスタのアドレスは以下の通り。

AF → 12F0
BC → 12F2
DE → 12F4
HL → 12F6

ただし, AF', BC', DE', HL', IX, IY, SP を変えることはできません。

⑪END行

この行はソースの終わりを示すのでコンパイラはこの行を見つけるとコンパイルを停止します。この文には文番号を付けることができません。

7) 入出力文

①SETG, RESG文

320×200 の疑似グラフィックのセット, リセット文です。

例 SETG (expl, exp2)
RESG (expl, exp2)

expl, exp2 で指定される位置にオペレートします。

expl は横方向で 0~319 まで, exp2 は 0~199 までの範囲です。expl ~ 2 は定数, 変数, 式です。

②CHM

VRAM (キャラクタRAM) をクリアします。

③GRCLR

G-RAM 1 (グラフィックRAM 1) をクリアします。

④GRON

G-RAM 1 を CRT に出力します。

⑤GROFF

G-RAM 1 の CRT 出力をやめます。

⑥READ

一般形は,

READ (*v1 fmt, v2 fmt, ..., vn fmt*)

で *v* は変数名, 配列要素で, *fmt* はその形式を表わします。*fmt* には次のようなものがあります。

<i>v.</i>	10進整数入力
<i>v. I</i>	10進 //
<i>v. B</i>	16進 //
<i>v. E</i>	10進実数入力
<i>v. A1</i>	1 文字入力
<i>v. A2</i>	2 文字入力
“Strings”	“ ”の中を出力
	改行

10進整数入力のときは “I” が省略できます。*v* は実数型変数名です。実数型入力のときは, E 型式, F 型式のどちらでもかまいません。

ここで, E 型式というのは,

1.0E+2, -2.30E-3

のように “E” を使って表わす数, F 型式というのは,

1.0, -2.3, 0.123

のように “E” を使わないで表わす数の表現型式のことです。

⑦WRITE

一般形は,

WRITE (*exp1 fmt, exp2 fmt, ..., expn fmt*)

で, *exp* は定数, 変数名, 式などです。

<i>exp</i>	10進左詰め整数表示
<i>exp. In</i>	n 行の 10 進右詰め整数表示
<i>exp. E</i>	E 型式による実数表示
<i>exp. B2</i>	16 進 2 行表示
<i>exp. B4</i>	16 進 4 行表示

exp. X	expだけスペース表示
exp. A1	1文字出力
exp. A2	2文字出力
exp. V	カーソル・バーチカル・セット
exp. H	カーソル・ホリゾンタル・セット
"strings"	文字列出力
/	改行

expは定数、変数、式などです。

なお、0.A1でCRTをセット、1.A1でプリンタをセットすることができます。

各要素は「、」で区切ることができますが、／(改行)だけは区切る必要がありません。

8) 組み込み関数

組み込み関数は以下のとおりです。

1) MEM(exp)	12) COS(fxp)
2) GET	13) TAN(fxp)
3) IOC(exp)	14) EXP(fxp)
4) LOW(exp)	15) ALOG(fxp)
5) MOD(exp1, exp2)	16) ATAN(fxp)
6) IRND(exp)	17) FLOAT(fxp)
7) IABS(exp)	18) IFIX(fxp)
8) ISIGN(exp1, exp2)	19) IOR(exp1, exp2)
9) ABS(fxp)	20) IAND(exp1, exp2)
10) SQRT(fxp)	21) IXOR(exp1, exp2)
11) SIN(fxp)	

expは整数型の定数、変数、算術式を、fxpは実数型のそれらを表わしています。

①MEM

MEM関数はexpで指定されたメモリをアクセスします。

```
v = MEM(exp)
MEM(exp) = exp1
```

MEMを右辺に置くか左辺に置くかによって機能が変わります。前者はPEEK的、後者はPOKE的機能を持ちます。

POKE機能の場合、exp1は0~255まで使用でき、それを越える場合には上位8ビットは無視されます。

②GET

キーからリアルタイム入力を行ないます。GET実行中、キー入力があれば対応するASCIIコードの値を持ちます。入力がなければ0を返します。

```
例 v = GET
      v = GET + exp
```

③IOC

I/Oポートをアクセスします。

```
v = IOC(exp)
IOC(exp) = exp1
```

使用方法はMEMと同じです。

④LOW

```
例 LOW(exp)
```

expの下位バイトを取った値を持ちます。expは-32,767~32,767まで、この関数が持つのは0~255までです。

⑤MOD

MODは除算時の余りを与えます。

```
例 MOD(exp, exp1)
```

余りはexp/exp1のそれです。

⑥IRND

この関数は乱数を与えます。

```
例 IRND(exp)
```

乱数は0~exp-1までの数です。expの上限は32,767までです。

⑦IABS

```
例 IABS(exp)
```

この関数はexpの絶対値を与えます。

⑧ISIGN

```
例 ISIGN(exp1, exp2)
```

exp1にexp2の持つ符号を与えます。この関数が持つ値はexp2の符号の付いたexp1である。

⑨絶対値

```
ABS(fxp)
```

fxpは定数型の定数、変数、あるいは算術式（以下、同じ）で、fxpの絶対値を与えます。

⑩平方根

```
SQRT(fxp)
```

fxpの平方根を与えます。

⑪, ⑫, ⑬ 三角関数

```
SIN(fxp)
COS(fxp)
TAN(fxp)
```

fxpの単位はラジアンで、それぞれ正弦、余弦、正接の値を与えます($180^\circ = \pi$ ラジアン)。

⑭ 指数関数

```
EXP(fxp)
```

指数関数値 e^{fxp} を与えます。

⑮ 自然対数

```
ALOG(fxp)
```

自然対数 $\log e(fxp)$ を与えます。

⑯ 逆正接

```
ATAN(fxp)
```

逆正接 $\tan^{-1}(fxp)$ を与えます。

⑰ 実数変換

```
FLOAT(exp)
```

整数型から実数型への変換用関数で、整数expの実数値を与えます。

【例】FLOAT(1) は 0.100000E+1

FLOAT(-2) は -0.200000E+1

⑱ 整数変換

```
IFIX(fxp)
```

実数型から整数型への変換用関数で、実数fxpに対して、その整数值を与えます。

【例】IFIX(1.2345) は 1

IFIX(-1.23E+1) は -12


```

13: DO 10 I=1,N
14: XR(I)=FLOAT(I-1)
15: XI(I)=0
16: WRITE(1,I,I4,XR(I),E,XI(I),E)
17: 11 IY=GET
18: BREAK
19: IF(IY)11,,11
20: 10 CONTINUE
21:C
22: CALL 5000
23: IPS=0
24: CALL 5100
25: CALL 1000
26: IPS=1
27: CALL 5100
28: CALL 1000
29: STOP
30:C
31: 1000 WRITE(1," I",6,X,"REAL")
32: WRITE(7,X,"IMAGINARY")
33: DO 1020 I=1,N
34: WRITE(1,I-1,I5,XR(I),E,XI(I),E)
35: 1001 IY=GET
36: BREAK
37: IF(IY)1001,,1001
38: 1020 CONTINUE
39: READ(IY,A1)
40: RETURN
41:C COS-TABLE
42: 5000 N2=N/2
43: P=2.0*PI/FLOAT(N)
44: WRITE("//COS-TABLE MAKING")
45: DO 5010 I=1,N/4+1
46: C(I)=COS(P*I/FLOAT(I-1))
47: IF(-I+1),5011,5011
48: C(N-I+2)=C(I)
49: C(N2+I)=-C(I)
50: 5011 C(N2+I)=-C(I)
51: C(N2-I+2)=-C(I)
52: 5010 CONTINUE
53: WRITE("//COS-TABLE COMPLETED")
54: RETURN
55:C FFT-SUBROUTINE
56:C IF IPS=0, THEN FFT
57:C IF IPS=1, THEN INV-FFT
58:C N MUST BE GREATER EQUAL 4
59: 5100 WRITE("//FFT WORKING")
60: N2=N/2
61: N4=N/4
62: ISW=1-2*IPS
63: ID=1
64: IE=N
65: 5180 CONTINUE
66:C WRITE("// IE=",IE,I4)
67: DO 5110 IA=1,ID
68: IB=(IA-1)*IE
69: DO 5110 J=1,IE/2
70: M=-(J-1)*ID*ISW
71: K=IB+J
72: L=K+IE/2
73: BREAK
74: 5120 IF(M),5140,5130
75: M=M+N
76: GOTO 5120
77: 5130 IF(M-N)5140,
78: M=M-N
79: GOTO 5130
80: 5140 V=C(M+1)
81: NN=N4-M
82: IF(NN),5150,5150
83: W=C(NN+N+1)
84: GOTO 5160
85: 5150 W=C(NN+1)
86: 5160 A=XR(K)-XR(L)
87: B=XI(K)-XI(L)
88: XR(K)=XR(K)+XR(L)
89: XI(K)=XI(K)+XI(L)
90: IF(M),5170,
91: IF(M-N4),5171,
92: XR(L)=A*B-W
93: XI(L)=A*B+W
94: GOTO 5110
95: 5170 XR(L)=A
96: XI(L)=B
97: GOTO 5110
98: 5171 XR(L)=-B
99: XI(L)=A
100: 5110 CONTINUE
101: ID=ID*2
102: IE=IE/2
103: IF(IE-1),,5180
104:C
105: WRITE("// BIT REVERSAL")
106: N1=N/256
107: N2=N-N1*256
108: MEM($4CF0)=N2
109: MEM($4CF1)=N1
110: DO 5190 I=1,N
111: IO=I-1
112: I1=IO/256
113: I2=IO-I1*256
114: MEM($4CF2)=I2
115: MEM($4CF3)=I1
116: 99 $ML 2A,FO,4C,ED,5B,F2,4C,AF,47,CB,3A
117: $ML CB,1B,CB,10,17,CB,3C,CB,1D,CB,45
118: $ML 2B,F1,21,F4,4C,70,23,77
119: J=MEM($4CF5)*256+MEM($4CF4)
120: IF(IO-J),5190,5190
121: AR=XR(I)
122: AI=XI(I)
123: XR(I)=XR(J+1)
124: XI(I)=XI(J+1)
125: XR(J+1)=AR
126: XI(J+1)=AI
127: 5190 CONTINUE
128: 5191 IF(IPS-1),5200,
129: WRITE("//FFT FINISHED")
130: RETURN
131: 5200 DO 5210 I=1,N
132: XR(I)=XR(I)/FLOAT(N)
133: XI(I)=XI(I)/FLOAT(N)
134: 5210 CONTINUE
135: WRITE("// INV-FFT FINISHED")
136: RETURN
137: END

```

2) リサージュ図形

グラフィック・コマンドを使ってリサージュ図形を描かせるプログラムです。

0:C	GRAPHIC TEST	11:	X=SIN(A/180.0)
1:C	-- リサージュ --	12:	Y=SIN(A/45.0)
2:C		13:	X=X*100.0
3:C		14:	Y=Y*100.0
4:	GRCLR	15:	L1=IFIX(X)
5:	L=160	16:	M1=IFIX(Y)
6:	M=100	17:	L2=L+L1
7:	CHM	18:	M2=M+M1
8:	GRON	19:	SETG(L2,M2)
9:	DO 10 I=1,1200,2	20:	10 CONTINUE
10:	A=FLOAT(I)	21:	END

参考文献

- 1)野沢,竹部,中本:“FORM”,I/O,’80年5月号~9月号
- 2)秋野実:“FORTRAN-MZ”,I/O,1年1月号