４．ＢＡＳＩＣ言語

コンピュータで使用する言語として最も一般的なものはＢＡＳＩＣである。卒業研究ではＢＡＳＩＣは後で説明するアセンブラやＤＳＰのプログラムで使用するデータテーブル（Data Table）と呼ばれる演算のためのデータを作成するために利用している。ここでは，データテーブル作成に必要なコマンドを説明するが，他のコマンドは必要があれば「リファレンスマニュアル」を参照されたい。

利用するＢＡＳＩＣは「N88-日本語 BASIC(86)Ver.6.0（MS-DOS 版）」であり，これは MS-DOS 上（MS は Microsoft の略）で動作する BASIC インタプリタシステム（Interpreter System）である。コンピュータ上で実行する言語は最終的にマシン語（Machine Language）とよばれる２進数データに変換（翻訳）されて実行されるが，インタプリタとはプログラムを１行１行翻訳して実行するシステムのことである。

ＤＯＳからＢＡＳＩＣを起動するには「N88BASIC.EXE」プログラムを実行するが，本システムでは「N88」という名前のバッチファイル（Batch File：一連のＤＯＳコマンドを記述したファイル）を作成しているので，カレントドライブをＡとして

```
A:\>N88
```

と入力する。ＢＡＳＩＣ起動後，右のようなメッセージが表示され「ダイレクトモード」と呼ばれる入力待ち状態となる。ダイレクトモードではＢＡＳＩＣの命令を行番号を付けずにそのまま入力し「ﾘﾀｰﾝ」キーを押すことで，直ちにその命令が実行される。このモードでは入力された命令はメモリに格納されないので，ディスクには保存できない。

```
NEC N-88 BASIC(86) version 6.0
Copyright (C) 1984,88 by NEC Corporation
XXXXXX Bytes free
OK
■
```

一方，行番号を付けて命令を入力する場合を「プログラムモード」といい，順次，行番号の小さい方からメモリへ格納される。行番号中の命令は「ﾘﾀｰﾝ」キーを押すことでメモリに格納される。作成したプログラムの実行は「RUN」命令により１行１行実行され，プログラム実行中にエラーが発生，あるいは終了した時にはダイレクトモードに戻る。ダイレクトモードでもプログラムモードでも同等に全ての命令が使用でき，また，一行に複数の命令を「：（コロン）」で区切って記述するマルチステートメントが可能である。

以下，通常使用している基本命令について説明するが，詳細はマニュアルを参照。

［変数］
変数名は頭が英字で始まる英数字であり，予約語（命令）以外の名前。
文字列を表す変数には最後に「＄」を付ける。
１６進数には頭に「＆Ｈ」を付ける。
ジャンプなど，飛び先の行番号の代わりにラベル（＊を付けた英数字文字列）を利用できる。

［画面表示とキーボード入力］

BEEP [<ｽｲｯﾁ>]

ｽﾋﾟｰｶを鳴らす。０：止める，１：鳴らす，省略時は一定期間鳴らす。

CLS [<機能>]

画面のｸﾘｱ。機能は１(ﾃﾞﾌｫﾙﾄ)：ﾃｷｽﾄ画面，２：ｸﾞﾗﾌｨｯｸ画面，３：ﾃｷｽﾄとｸﾞﾗﾌｨｯｸ画面

COLOR [<ﾌｧﾝｸｼｮﾝｺｰﾄﾞ>][,<BGC>][,<BC>][,<FGC>][,<PM>]

ﾃﾞｨｽﾌﾟﾚｲ表示の色を指定する。通常，ｶﾗｰﾓｰﾄﾞで「COLOR 5」等と設定。
色は０：黒，１：青，２：赤，３：紫，４：緑，５：水色，６：黄，７：白

CONSOLE [<ｽｸﾛｰﾙ開始行>][,<ｽｸﾛｰﾙ行数>][,<ﾌｧﾝｸｼｮﾝ表示ｽｲｯﾁ>][,<ｶﾗｰ／白黒ｽｲｯﾁ>][,<ｷｬﾗｸﾀﾓｰﾄﾞ>]

ｽｸﾛｰﾙとは表示が上に移動することである。通常「CONSOLE 0,25,0,1」と設定。

INPUT [<ﾌﾟﾛﾝﾌﾟﾄ文>|;または,|]<変数>[,<変数>・・・・]

ｷｰﾎﾞｰﾄﾞから入力されたﾃﾞｰﾀを変数に代入する。ﾌﾟﾛﾝﾌﾟﾄとは表示するﾒｯｾｰｼﾞ。「；」時はメッセージ表示後，「？」とﾌﾞﾗﾝｸを表示。「，」時は何も表示しない。

LOCATE [<X>][,<Y>][,<ｶｰｿﾙｽｲｯﾁ>]

ｶｰｿﾙを指定位置へ移動する。通常「LOCATE 2,1」等と設定。

PRINT [<式>][|;または,|<式>・・][|;または,|]

画面にデータを表示する。「PRINT」は「?」でもよい。例えば「? “DATA=”;DD」等

PRINT USING <書式制御文字列>;<式>[|;または,|<式>・・][|;または,|]

書式を指定して画面にデータを表示する。例えば「PRINT USING “###.##”;DD」（小数部２桁，「PRINT “DATA=”;USING “###”;DD」（「DATA=」を表示して整数３桁）等

SCREEN [<画面ﾓｰﾄﾞ>][,<画面ｽｲｯﾁ>][,<ｱｸﾃｨﾌﾞﾍﾟｰｼﾞ>][,<ﾃﾞｨｽﾌﾟﾚｲﾍﾟｰｼﾞ>]

通常「SCREEN 3」とし，「高分解ｶﾗｰﾓｰﾄﾞ 640×400ﾄﾞｯﾄ」に設定。

WIDTH <桁数> [,<行数>]

画面の桁，行数を設定する。通常は「WIDTH 80,25」に設定。

［一般命令］

END

プログラムの終了を宣言する。なくてもよい。

FOR <変数名>=<初期値> TO <終値>[STEP <増分>]　～　NEXT [<変数名>][,<変数名>]

FOR～NEXT 間の命令を FOR 中で指定した条件に従って繰り返し実行する。

GOTO <行番号>

＜行番号＞へジャンプする。＜行番号＞はラベルでもよい。

IF <論理式> THEN [|<文>または<行番号>|] [ELSE |<文>または<行番号>|]

＜論理式＞の条件に従って実行を制御する。＜行番号＞はラベルでもよい。

［算術・文字列操作］

ABS (<数式>)

＜数式＞の絶対値を得る。

COS (<数式>)

＜数式＞の余弦値を得る。＜数式＞の単位はラジアン。

HEX$ (<数式>)

＜数式＞の値を１６進数に変換し，その文字列を得る。

INT (<数式>)

＜数式＞の値を越えない最大の整数を得る。

RIGHT$ (<文字列>,<式>)

＜文字列＞の右側（最後）から＜式＞で指定した桁数の文字列を取り出す。

SIN (<数式>)

＜数式＞の正弦値を得る。＜数式＞の単位はラジアン。

［コマンド］

FILES [<ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ>]

ﾃﾞｨｽｸ内容を表示する。ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀを省略した場合には N88BASIC を起動した時のｶﾚﾝﾄﾞ

ﾄﾞﾗｲﾌﾞ内容が表示される。別のﾄﾞﾗｲﾌﾞ内容を表示する時は，例えば，「FILES “B:¥BFILE”」等と指定する（ﾀﾞﾌﾞﾙｸｳｫｰﾃｰｼｮﾝ「“」で囲む）。

LOAD <ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ>[,R]

ﾃﾞｨｽｸ上のﾌﾟﾛｸﾞﾗﾑをﾒﾓﾘにﾛｰﾄﾞする。R ｵﾌﾟｼｮﾝ指定時は，ﾛｰﾄﾞ後実行する。
例えば，「LOAD “B:¥BFILE¥PROG.BAS”」等と指定。「.BAS」は省略可能。

RENUM [<新行番号>][,<旧行番号>][,<増分>]

ﾌﾟﾛｸﾞﾗﾑの行番号を付け直す。

RUN [<行番号>]

ﾌﾟﾛｸﾞﾗﾑを実行する。＜行番号＞省略時は，最初の行から実行する。

SAVE <ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ>[,|A または P|]

ﾒﾓﾘ上のﾌﾟﾛｸﾞﾗﾑをﾃﾞｨｽｸに保存する。A ｵﾌﾟｼｮﾝ指定時はｱｽｷｰ形式で保存（この場合には，DOS 上のｴﾃﾞｨﾀで編集可能），P ｵﾌﾟｼｮﾝ指定時はﾌﾟﾛﾃｸﾄ形式で保存，ｵﾌﾟｼｮﾝ省略時はﾊﾞｲﾅﾘ形式で保存。例えば，「SAVE “B:¥BFILE¥PROG.BAS”,A」等と指定。「.BAS」は省略不可。

SYSTEM

N88BASIC を終了し，DOS に制御を戻す（ﾒﾓﾘ上のﾌﾟﾛｸﾞﾗﾑは消去される）。

［メモリ操作関係］

BLOAD <ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ>[,<ﾛｰﾄﾞｱﾄﾞﾚｽ>][,R]

<ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ>で指定されたﾃﾞｨｽｸ上の機械語ﾌｧｲﾙを＜ﾛｰﾄﾞｱﾄﾞﾚｽ＞（ｵﾌｾｯﾄ）からﾛｰﾄﾞする（物理アドレスは直前に実行された「DEF SEG」に依存する）。Ｒオプション指定時はﾛｰﾄﾞ後その先頭ｱﾄﾞﾚｽから実行を開始する（この場合には機械語ﾌｧｲﾙは実行可能なﾌｧｲﾙでなければならない）。

BSAVE <ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ>,<開始ｱﾄﾞﾚｽ>,<長さ>

ﾒﾓﾘ上の指定範囲の内容を<ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ>で指定されたﾃﾞｨｽｸ上に機械語ﾌｧｲﾙとして保存する（物理アドレスは直前に実行された「DEF SEG」に依存する）。

DEF SEG=<ｾｸﾞﾒﾝﾄﾍﾞｰｽ>

＜ｾｸﾞﾒﾝﾄﾍﾞｰｽ＞の値（CPU 内の「DS」）を設定する。

PEEK(ｱﾄﾞﾚｽ)

指定されたｱﾄﾞﾚｽ（ｵﾌｾｯﾄ）のﾒﾓﾘから１ﾊﾞｲﾄ（８ﾋﾞｯﾄ）ﾃﾞｰﾀを読み出す。

POKE <ｱﾄﾞﾚｽ>,<式>

指定されたｱﾄﾞﾚｽ（ｵﾌｾｯﾄ）のﾒﾓﾘ上に１ﾊﾞｲﾄ（８ﾋﾞｯﾄ）ﾃﾞｰﾀを書き込む。

例として，３章で述べたテキストＶＲＡＭをアクセスしてみよう。画面をクリアしてカーソルを１０行目付近に移動し，ダイレクトモードで次のように入力する。

```
DEF SEG=&HA000:POKE &H0,&H41
```

画面の左上に「A」が表示される。これは「DEF SEG」によりセグメントを「A000H」に設定し，「POKE」によってオフセットアドレス「0000H」番地に「41H」のデータを書き込んでいる。すなわち，物理アドレスは「A0000H」であり，これはテキストＶＲＡＭの画面左上に対応したアドレスである。書き込むデータはキャラクタコードであり，「41H」は英大文字の「A」である。

では，次に画面左上に「０」を入力し，「ﾘﾀｰﾝ」キーを押さずにカーソルを１０行目付近に移動し，ダイレクトモードで次のように入力する。

```
DEF SEG=&HA000:DD=PEEK(&H0):PRINT DD
```

結果は，次の行に「48」と表示される。POKE とは逆に PEEK はオフセットアドレス「0000H」番地からデータを取り込む。そのデータは変数「DD」（名前は何でもよい）に入れられ，「PRINT」命令によって表示される。表示データは１０進数で，これは１６進

数では「30H」，すなわち，キャラクタコードの「０」である。なお，「PRINT」で１６進数表示したい時には「HEX$」命令を使用し，次のように記述する。

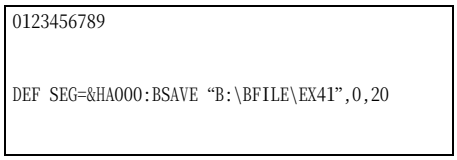
```
DEF SEG=&HA000:DD=PEEK(&H0):PRINT HEX$(DD)
```

あるいは

```
DEF SEG=&HA000:DD$=HEX$(PEEK(&H0)):PRINT DD$
```

なお，後の場合には「HEX$」によって文字列に変換されるので，変数「DD」は「DD$」のように文字列変数としなければならない（変数「DD」と「DD$」は全く別の変数である）。

次に，右図のように画面左上に「０～９」を入力し，「ﾘﾀｰﾝ」キーを押さずにカーソルを下の行に移動し，ダイレクトモードで次のように入力する。




```
DEF SEG=&HA000:BSAVE “B:\BFILE\EX41”,0,20
```

これはテキストＶＲＡＭ上の物理アドレス「A0000H～A0014H」の２０ﾊﾞｲﾄのデータをディスク「B:¥BFILE¥」内にファイル名「EX41」で保存するということを行う。ファイルが保存されているかどうか「FILES」命令で確かめてみよう。次に，画面をクリアし，１０行目付近に次のように入力する。

```
DEF SEG=&HA000:BLOAD “B:\BFILE\EX41”,0
```

先ほどの「０～９」までの数字が同じ場所に表示される。なお，＜ﾛｰﾄﾞｱﾄﾞﾚｽ＞が「０」の時は省略可能である。

グラフィックＶＲＡＭに対しても同じことができ，次のように順次操作してみよう。

```
DEF SEG=&HA800:FOR I=0 TO 9:POKE I,&HFF:NEXT I
```

画面左上に青色の線が１０バイト（８０ドット）表示される。

```
BSAVE “B:\BFILE\EX42”,0,10
```

データを保存する。クリア命令「CLS 3」を実行し，次の命令を入力する。

```
BLOAD “B:\BFILE\EX42”
```

再び，画面左上に青色の線が１０バイト（８０ドット）表示される。セグメントの値は「DEF SEG」命令が実行されない限り変わらないので，毎回定義する必要はない。

［ディスクファイル制御］

CLOSE [[#]<ﾌｧｲﾙ番号>][,[#]<ﾌｧｲﾙ番号>･･･]

OPEN 文によって開かれたﾌｧｲﾙ番号のﾃﾞｨｽｸ上ﾌｧｲﾙを閉じる。<ﾌｧｲﾙ番号>省略時は全てのフ

ファイルを閉じる。

INPUT #<ﾌｧｲﾙ番号>,<変数名>[,<変数名>･･･]

OPEN 文によって開かれたﾌｧｲﾙ番号のﾃﾞｨｽｸ上ﾌｧｲﾙ（ｼｰｹﾝｼｬﾙﾌｧｲﾙ）からﾃﾞｰﾀを読み込む。

OPEN <ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ>[FOR <入出力ﾓｰﾄﾞ>][<共用ﾓｰﾄﾞ>] AS [#1]<ﾌｧｲﾙ番号>

<ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ>で指定されたﾌｧｲﾙに対して入出力をｵｰﾌﾟﾝし，それにﾌｧｲﾙ番号を割り当てる。通常はｼｰｹﾝｼｬﾙﾌｧｲﾙで OPEN するので，例えば

「OPEN “ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ“ FOR INPUT AS #1」-------ﾌｧｲﾙから入力の場合

「OPEN “ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ“ FOR OUTPUT AS #1」-------ﾌｧｲﾙへ出力の場合

のように定義する。

PRINT #<ﾌｧｲﾙ番号>,<式>[|,または;|<式>･･･][|,または;|]

OPEN 文によって開かれた出力ﾓｰﾄﾞのﾌｧｲﾙ番号のﾃﾞｨｽｸ上ﾌｧｲﾙ（ｼｰｹﾝｼｬﾙﾌｧｲﾙ）へ，＜式＞で指定した文字列や数値などのﾃﾞｰﾀを書き出す。

簡単な例として，１～１０の２乗を求め，その結果をシーケンシャルデータに書き込みおよび読み出しを行うプログラムを作成する。まず，右のようなプログラムを作成し，これを「EX43.BAS」のファイル名で自分のディスク（ドライブB）のサブディレクトリ「BFILE」内にアスキーセーブする（バイナリセーブでもよいが，後のプログラム編集で使用するため）。これを実行すると，結果は同じディレクトリ内に「EX43.DAT」のファイル名で保存され，その内容は右のようになっている。

（ﾌｧｲﾙ名：EX43.BAS）

```
100 OPEN "B:\BFILE\EX43.DAT" FOR OUTPUT AS #1
110 FOR I=1 TO 10
120 X=I*I
130 PRINT #1,X
140 NEXT I
150 CLOSE
```

（ﾌｧｲﾙ名：EX43.DAT）

```
 1
 4
 9
 16
 25
 36
 49
 64
 81
 100
```

次に「EX43.DAT」からデータを読み込み，値を表示させるために右のプログラムを作成し，同一ディレクトリ内に「EX44.BAS」としてアスキーセーブする。これを実行すると，先ほどの２乗結果が表示される。

（ﾌｧｲﾙ名：EX44.BAS）

```
100 OPEN "B:\BFILE\EX43.DAT" FOR INPUT AS #1
110 FOR I=1 TO 10
120 INPUT #1,X
130 PRINT X
140 NEXT I
150 CLOSE
```

では，「EX43.DAT」の１３０行目（１００行目のファイル名も変更）を少し変えて下のような「EX45.BAS」を作成，結果のデータファイルを作成すると，そのデータファイルの内容は右のようになる。すなわち，「PRINT」文の変数の後に「；」

（ﾌｧｲﾙ名：EX45.BAS）

```
100 OPEN "B:\BFILE\EX45.DAT" FOR OUTPUT AS #1
110 FOR I=1 TO 10
120 X=I*I
130 PRINT #1,X;
140 NEXT I
150 CLOSE
```

（ﾌｧｲﾙ名：EX45.DAT）

```
 1  4  9  16  25  36  49  64  81  100
```

を付けることにより，データは「ブランク」で区切られた連続データとなる。これを先の「EX44.BAS」で読み込んでも（１００行目のファイル名を変えて）「EX43.DAT」の場

合と同じである。すなわち，どちらの形式でデータファイルを作成してもよいことになる。

一方，これから作成するアセンブラやＤＳＰのプログラムでは「データテーブル」と呼ばれるデータを必要とし，これらのデータのファイル形式では全てのデータが連続し，１６進数表示のアスキー形式でなければならない。このようなデータを作成するために右のようなプログラムを作成する（ファイル名は「EX46.BAS」）。これにより作成されるデータファイル「EX46.DAT」は右のようであり，４文字づつ区切って読むと１６進数で前と同じデータとなっている。なお，このような形式で作成したデータファイルはＢＡＳＩＣでは読み込めない。

（ﾌｧｲﾙ名：EX46.BAS）

```
100 OPEN "B:\BFILE\EX46.DAT" FOR OUTPUT AS #1
110 FOR I=1 TO 10
120 X=I*I
130 DD$=RIGHT$("0000"+HEX$(X),4)
140 PRINT #1,DD$;
150 NEXT I
160 CLOSE
```

（ﾌｧｲﾙ名：EX46.DAT）

```
0001000400090010001900240034004000510064
```

ここまでは例として正の数（整数）のデータを作成したが，負の数（整数）ではどうなるだろうか。「EX43.BAS」のようにそのまま数値でデータファイルを作成する場合には，データは例えば「－１０」のように保存される。しかし，１６進数のアスキー形式では様子が異なる。下に示すプログラム「EX47.BAS」（０～－１０の値を保存）を作成した場合には，１６ビット（４桁）２の補数による正負表現となっている「EX47.DAT」。すなわち，ＢＡＳＩＣで扱う整数は１６ビット正負表現によってＣＰＵで処理されている。

（ﾌｧｲﾙ名：EX47.BAS）

```
100 OPEN "B:\BFILE\EX47.DAT" FOR OUTPUT AS #1
110 FOR I=0 TO -10 STEP -1
120 DD$=RIGHT$("000"+HEX$(I),4)
130 PRINT #1,DD$;
140 NEXT I
150 CLOSE
```

（ﾌｧｲﾙ名：EX47.DAT）

```
0000FFFFFFFEFFFDFFFCFFFBFFFAFFF9FFF8FFF7FFF6
```

なお，ＢＡＳＩＣでの実数は浮動小数点表現（Floating-Point Representation）であるが，これはアセンブラプログラムでは用いないので，説明は省略する。