文字処理のためのサブ・ルーチン

西端幸雄

最近のパーソナル・コンピューターの発達は、目ざましいもので、その機能は、ここ数年の間、急速に向上し続けている。特に、日本語処理、処理速度、内部記憶容量、ディスクの記憶容量などは、数倍にもなっている。さらに、ベーシック言語においては、文字処理のためのコマンドが豊富にもなってきた。こうしたことから、国語学や国文学の領域の研究者にもかなり使いやすくなったと言えよう。まさに、パーソナル・コンピューター（以下、パソコンと略す。）の名が示すように、使う者が身近に置いて、自在に使いこなせるという、大型コンピューターにはない利点が鮮明に打ち出されてきた。そのため、国語学や国文学の研究者の中にも、パソコンを使って、作業をする方がふえてきたようである。

ところが、パソコンのマニュアルや市販されているベーシック言語の解説書には、文字処理のためのコマンドの説明が十分になされているものが非常に少ないのが、現状のようである。そのために、僅かな処理のためのプログラムを作成するのに、試行錯誤を繰り返し、貴重な時間を浪費するということもあろう。

そこで、小稿では、これまで私が作成したプログラムのうち、ベーシック言語で文字処理や語彙処理を行う際に有用なものを紹介することにした。専門家の方からご覧になれば、非常に稚拙なプログラミングとご批判を受けることと思うが、その点、ご教示をいただければ、幸いである。

１．使用機種

今回用いたパソコンは、ＦＵＪＩＴＳＵ ＭＩＣＲＯ 16β（以下、ＦＭ16βと略す。）とＮＥＣのＰＣ9801ＶＸ２（以下、ＰＣ9801と略す。）である。両者の機能の主な点を摘記すると、以下のようになる。

	ＦＭ16β	ＰＣ9801
ＣＰＵ	80186	V30,80286
内部記憶容量	512KB	640KB
ディスク容量	1MB（2台）	1MB（2台）
日本語入力	VJE-α	NECDIC

なお、ＰＣ9801に登載されているＣＰＵの内、80286のＣＰＵモードで使用すると、ＦＭ16βの80186ＣＰＵやV30ＣＰＵよりも２～３倍の速度で作動するため、大量のデータを処理するのに適しているといえよう。

使用するベーシック言語は、ＦＭ16β用のＦ－ＢＡＳＩＣ86 Ｖ3.1とＰＣ9801用のＮ88－日本語ＢＡＳＩＣ（86）3.0である。

また、ＤＯＳ（Disk Operation System）としては、ＭＳ－ＤＯＳを用いた。このＭＳ－ＤＯＳ上でベーシック言語を用いる利点は、プログラム・ファイルやデータ・ファイルが異機種間でも容易に移植できるという点である。

いま、その方法を簡単に記すと、

１．ドライブＡに、当該のパソコン用ディスケットを入れ、ＭＳ－ＤＯＳを立ち上げる。

２．ドライブＢに、コピー元の異機種用ディスケットを入れる。

３．Ａ＞というプロンプトの後に、下記のように命令を与える。

Ａ＞ＣＯＰＹ　Ｂ：＜ファイル名．拡張子＞　Ａ：

これで、ＭＳ－ＤＯＳ上で作成されたファイルは、コピーできるが、プログラム・ファイルの場合だけ、コピーの前後に絶対に行わなければならないことがある。まず、コピーを行う前に、当該のプログラム・ファイルをアスキー形式でセーブしておく必要がある。（ＳＡＶＥ　”プログラム名．拡張子”，Ａ）また、コピーを行った後、ＭＳ－ＤＯＳのラインエディタ（ＥＤＬＩＮ．ＣＯＭ）で、１行

目だけを削除しなければならない。これは、１行目に不用な情報を持った行が入っているからである。その実際の方法は、下記の通りである。

１．Ａ＞プロンプトの後に、下記のように命令を与える。（ただし、この場合、ドライブＡに入れるディスケット内にＥＤＬＩＮ．ＣＯＭというファイルがコピーされていなければならない。）

Ａ＞ＥＤＬＩＮ　＜ファイル名．拡張子＞

２．＊というプロンプトが表示されれば、１Ｄ（１行目削除の意）を指示する。

３．再び＊プロンプトが表示されれば、Ｅ（ＥＮＤの略）を指示する。

２．両機種の言語差

上に述べた方法により、プログラム・ファイルを相互に移植したとしても、そのプログラムを異機種においてすぐに使うことはできない。パソコンにも我々人間の世界と同じ様な方言が各機種に存在する。その方言部分を当該の機種の表現形式に手直ししなければならない。

今回ＦＭ16βとＰＣ9801の２機種を使ってみて、両機種のベーシック言語間で文字処理機能に大きな差があることが分かった。今、その文字処理機能だけを問題とするなら、ＰＣ9801のそれは非常に貧弱だと思われる。問題となる点を、以下に列挙する。

① 配列変数内のデータを検索するＳＥＡＲＣＨ関数が整数型のデータしか扱えない。そのため、文字型データを検索するには、ディスク・アクセスの方法を原則的には採らざるを得ない。よって、検索に要する時間がかかることになる。対策としては、ＲＡＭディスクを装備して、データをＲＡＭディスクから読み込むようにするという方法がある。（ＦＭ16βは、文字型も扱うことができる。）＜ディスク検索のためのサブ・ルーチンとして、プログラム１を参照＞

② 両機種共に、全角文字を半角文字にするＫＡＣＮＶＳ関数と、逆に半角文字を全角文字にするＡＫＣＮＶＳ関数を備えている。ところが、ＰＣ9801の場合、ＡＫＣＮＶＳ関数で濁点のついた半角文字を全角文字に直すと、濁点が１文字扱いにされてしまい、濁点付きの全角文字に直らない。

（例　ﾀﾞ-------->ﾀﾞ　　正しくは、ﾀﾞ-------->ダ）

また、逆にＫＡＣＮＶＳを用いて、濁点つき全角文字を半角文字に直そうとすると、エラーが生じてしまう。

そのため、濁点の付いた半角文字と全角文字を双方向に変換しようとする場合、プログラム上で非常に手間な手続きを経なければ、正しく濁点がついた文字が得られない。（ＦＭ16βは、双方向共に正しく変換することができる。）＜正しく変換するためのサブ・ルーチンとして、プログラム２を参照＞

③ 全角文字の左右から幾文字かを取り出すＫＬＥＦＴ＄，ＫＲＩＧＨＴ＄関数が備わっていないために、ＫＭＩＤ＄関数しか使えず、手間な手続きを経なければならないし、プログラムも自ずと長くなってしまう。特に、右から幾文字かを取り出すときは、以下のように面倒である。

（例）

```
10 A$=KMID$(B$,KLEN(B$)-2,3)
```

cf.

```
10 A$=KRIGHT$(B$,3)
```

これは、３文字以上の長さを持つ文字変数Ｂ＄の右から３文字を文字変数Ａ＄に代入するという場合である。

（ＦＭ16βにはＫＬＥＦＴ＄，ＫＲＩＧＨＴ＄共に備わっている。）

④ 全角文字を扱う場合、よく用いるのがＪＩＳコードに変換するという法であるが、このＪＩＳコードを求める関数がＪＩＳ＄と文字型になっているために数値型のＪＩＳコードを求めるには、手間な手続きを経なければならない。（ＦＭ16βの場合、下記のように、数値変数としてＪＩＳコードを返してくるので、手続きが簡単である。）

（例）
```
10 A=VAL("&H"+JIS$(B$))
```
cf.
```
10 A=JIS(B$)
```

⑤　上の④で述べた場合の逆で、ＪＩＳコードから全角文字を求めるのも、ＫＮＪ＄関数が文字型のＪＩＳコード（16進数）しか受け付けないので、同様に手間な手続きを経なければならない。（ＦＭ16βの場合、10進数、16進数に関わらず、数値型のＪＩＳコードを変換する。）

（例）
```
10 A$=KNJ$(HEX$(B))
```
cf.
```
10 A$=KNJ$(B)
```

以上に掲げた他にも、細かく見れば、両機種の間にはいくつかの差異が存在するが、その点については、以下、サブ・ルーチンを説明する際に必要に応じて指摘することにする。

３．プログラム１（ディスク検索）

パソコンを用いて、語彙処理をする場合、データをディスケットに登録しておき、以後、必要に応じて、随時そのデータを呼び出すことが頻繁に行われる。ところが、このデータ検索を要領よく短時間で行わなければ、処理作業全体が非常に長くなることもある

下記のプログラムは、その検索作業をできる限り短時間で行うものである。データ量が数千例とかなり多くても、ディスク・ドライブを通した場合、１語を検索するのに２～３秒以内で行う。また、ＲＡＭディスクを用いれば、１秒以内で検索する。

このプログラムを使用する前に、１～８文字の語句データをそれぞれ語句の長さ毎に、ＤＩＣ１．ＳＯＲ～ＤＩＣ８．ＳＯＲというランダム・ファイルに、５０音順に並べ替え、登録されていることが絶対条件である。（並べ替えのプログラムは、プログラム３を参照）

今、210～230行に用いている各変数と数値は、下記の通りとする。各々の値は、必要に応じて、替えて用いることができる。（以下、ランダム・ファイルを用いている場合、同じ設定とする。）

DL:語句データ長（語句・読み・漢字・品詞）注①
DN:１レコード内のデータ数
FI:１レコードのフィールド幅

なお、同音異義語が幾つか登録されている場合、先頭の語彙を検索する手続きは、省略してある。

```
100 CLS:GDAT$="":INPUT "検索したい語句　＝　",GDAT$
110 IF GDAT$="" THEN BEEP:GOTO 100 ELSE GOSUB 200
120 CLS:PRINT "そのファイルはありません":GOTO 150
130 CLS:PRINT "その語句は登録されていません":GOTO 150
140 PRINT:PRINT "検索語句は　＝　"+D$
150 PRINT:YN$="":INPUT "続けますか　＜Ｙ／Ｎ＞　",YN$
160 IF YN$="" THEN BEEP:GOTO 150 ELSE IF INSTR("Yy",YN$)>0 THEN 10
0 ELSE END
200 REM ＊＊　パラメータ付加　＊＊
210 LEG=KLEN(GDAT$):NO$="":NO$=RIGHT$(STR$(LEG),1):IF LEG=1 THEN D
L=14:DN=18:FI=252 ELSE IF LEG=2 THEN DL=22:DN=11:FI=242 ELSE IF LE
G=3 THEN DL=28:DN=9:FI=252
220 IF LEG=4 THEN DL=32:DN=7:FI=224 ELSE IF LEG=5 THEN DL=36:DN=7:
:FI=252
230 IF LEG=6 THEN DL=40:DN=6:FI=240 ELSE IF LEG=7 THEN DL=42:DN=6:
:FI=252 ELSE IF LEG=8 THEN DL=48:DN=5:FI=240
```

```
280 REM ＊＊＊　ディスク検索　＊＊＊
290 OPEN "C:DIC"+NO$+".SOR" AS #1:FIELD #1,FI AS TA$:TR=LOF(1):IF
TR=0 THEN CLOSE #1:RETURN 120 ELSE GET #1,TR
300 FOR I=1 TO DN:IF ASC(MID$(TA$,(I-1)*DL+1,DL))=32 THEN 320
310 NEXT
320 LF=1:RG=((TR-1)*DN)+(I-1)
330 IF RG-LF<0 THEN CLOSE #1:RETURN 130
340 DUMII=INT((RG+LF)/2):DUMTR=INT(DUMII/DN+(DN-1)/DN):DUMJJ=DUMII
-(DUMTR-1)*DN:GET #1,DUMTR:DUMGKB$="":DUMGKB$=MID$(TA$,(DUMJJ-1)*D
L+1,DL):D$="":D$=KMID$(DUMGKB$,1,LEG)
350 IF D$=GDAT$ THEN CLOSE #1:RETURN 140 ELSE IF GDAT$<D$ THEN RG=
DUMII-1:GOTO 330 ELSE IF GDAT$>D$ THEN LF=DUMII+1:GOTO 330
```

４．プログラム２（文字変換）

ここでは、文字変換を行うためのプログラムをいくつか掲げることにする。

まず、先に２の②で述べたように、ＰＣ9801では、ＡＫＣＮＶ＄を用いて濁点の付いた半角文字を全角文字に直そうとした時、濁点を１文字扱いにしてしまうという問題を解決するためには、以下のような手続きが必要となる。

```
10 REM ＊＊　濁点付き文字の処理　＊＊
20 CLS:DIM TW$(10)
30 DW$="":INPUT "濁点付き半角文字　＝　",DW$
40 IF DW$="" THEN BEEP:GOTO 30
50 FOR I=1 TO LEN(DW$):TW$(I)=MID$(DW$,I,1):IF ASC(TW$(I))=32 THEN
 80
60 TA$="":TA$=AKCNV$(TW$(I)):TW$(I)="";TW$(I)=TA$:IF TW$(I)="ﾞ" T
HEN TW$(I-1)=KNJ$(HEX$(VAL("&H"+JIS$(TW$(I-1)))+&H1)):ZE$=KMID$(ZE
$,1,I-2):TW$(I)=TW$(I-1)
70 ZE$=ZE$+TW$(I):NEXT
80 PRINT:PRINT "濁点付き全角文字　＝　"+DW$
```

１０文字までの半角文字を50行目で配列変数TW$()の中に取り込む。そして、60行目でＡＫＣＮＶ＄を用いて、全角文字に直すのであるが、その際、濁点であれば、１文字前の文字を濁点付き全角文字に直し、文字変数DW$にはその濁点付き全角文字の１文字前までの文字を代入し、それに濁点付き全角文字をつなぐ。

ところで、上記のプログラムでは、半角の片仮名を全角の片仮名に直すことしかできない。しかし、実際の文字処理や語彙処理を行う場合には、平仮名を用いることの方が多いと思われる。そこで、上記のプログラムで得られた全角の片仮名を平仮名に直すプログラムを以下に掲げる。（上記のプログラムと連続しているものとする。）

```
90 FOR I=1 TO KLEN(DW$):SE$="":KN$="":KN$=KMID$(DW$,I,1):IF ASC(KN
$)=32 THEN 120 ELSE IF KN$="ｨ" THEN SE$="ぁ":GOTO 110 ELSE IF KN$
="ｪ" THEN SE$="ぇ":GOTO 110
100 SE$=KNJ$(HEX$(VAL("&H"+JIS$(KN$))-&H100))
110 GDAT$=GDAT$+SE$:NEXT
120 PRINT:PRINT "平仮名　＝　"+GDAT$
```

90行目で分かるように、半角の片仮名として「ヰ・ヱ」を用いることができないので、「ヰ・ヱ」を小文字の「ｨ・ｪ」で代用している。

今、全角の片仮名は平仮名のＪＩＳコードより、それぞれ対応する文字の間で、100（16進数）大きい。ということは、片仮名のＪＩＳコードから100（16進数）を引けば、平仮名のＪＩＳコードが求められる。それをプログラム化したのが100行目である。

以上に掲げてきたのは、半角文字を平仮名に直すまでのプログラムであったが、実際の文字処理や語彙処理では、その逆も重要である。特に、国語学や国文学で用いるデータは、普通平仮名や漢字で表記されることの方が多い。そこで、平仮名を半角の片仮名に直すプログラムを以下に掲げる。（上記のプログラムと連続しているものとする。）

```
130 FOR I=1 TO KLEN(GDAT$):KA$="":HI$="":HI$=KMID$(GDAT$,I,1):IF A
SC(HI$)=32 THEN 160 ELSE IF HI$="ゐ" THEN KA$="ｨ":GOTO 150 ELSE I
F HI$="ゑ" THEN KA$="ｪ":GOTO 150
140 KA$=KNJ$(HEX$(VAL("&H"+JIS$(HI$))+&H100))
150 KATA$=KATA$+KA$:NEXT
160 PRINT:PRINT "片仮名　＝　"+KATA$
170 FOR I=1 TO KLEN(KATA$):MO$="":MO$=KMID$(KATA$,I,1):IF ASC(MO$)
=32 THEN 220
180 DM$="";JMO=VAL("&H"+JIS$(MO$)):IF JMO>=9516 AND JMO=<9538 AND
JMO MOD 2=0 THEN 200 ELSE IF JMO>=9541 AND JMO=<9545 AND JMO MOD 2
=1 THEN 200 ELSE IF JMO>=9552 AND JMO=<9564 AND JMO MOD 3=0 THEN 2
00
190 DM$=KACNV$(MO$):GOTO 210
200 DM$=KACNV$(KNJ$(HEX$(JMO-1)))+"ﾞ"
210 HANK$=HANK$+DM$:NEXT
220 PRINT:PRINT "半角の片仮名　＝　"+HANK$
```

180行目で、数値変数JMOに当該の全角文字（片仮名）のコード（10進数）を代入し、そのコードが濁点付きかどうかを調べている。例えば、「ガ」（9516）から「ヂ」（9538）までの濁点付き片仮名は、9516から9538までの偶数のコードで表される。よって、除算の余りを求める演算子ＭＯＤを用いて、結果が余りなし（0）の場合、当該の全角文字を濁点付きと判断できる。

そして、清音を表す片仮名のコードは、濁点付きより1小さいので、数値変数JMOから1を引いて、200行目のように、ＫＡＣＮＶ＄で変換した文字に半角の濁点を付ければよいことになる。

５．プログラム３（並べ替え）

文字処理や語彙処理を行うとき、ある基準に従って、文字や語彙を並べ替える事がよく行われる。しかし、この並べ替えの作業を人間の手で行うとなると、時間も要するし、誤りを犯す可能性も高くなる。その点、その作業をパソコンによって行うと、かなり短時間に、正確に行うことができる。ただし、そこには、パソコン側の機械的な制約もあり、一度に大量のデータを処理することは難しい。そこで、ここでは、1500位のデータを一気に並べ替える方法と、データ量に関係なく並べ替える方法とを掲げることにする。

まず、1500位のデータを一気に並べ替えるプログラムを以下に掲げる。このプログラムは、大量のデータを一度に配列変数の中に格納するため、パソコン内部の文字列データ領域の大きさと１データの長さによって、一度に処理できるデータ量が制約を受ける。よって、先に、1500位というデータ量を示したが、それはあくまで目安でしかない。

並べ替えの方法には、色々のものがあるが、以下に掲げるのは、シェルソート法によるものである。

```
10 REM ＊＊　シェルソート　　＊＊
20 CLS:DIM G$(1000)
30 CO=1000:DA=CO
40 DA=INT(DA/2):IF DA<1 THEN 110
50 DD=CO-DA
60 FOR M=1 TO DD:N=M
```

```
70 IF G$(N)=<G$(N+DA) THEN 90
80 SWAP G$(N),G$(N+DA):N=N-DA:IF N>=1 THEN 70
90 NEXT
100 GOTO 40
110 FOR I=1 TO CO:PRINT G$(I):NEXT
```

10行目に示したように、データの上限を1000としてある。また、配列変数G$()にはすでにデータが格納されているものとする。このプログラムによって、1000位のデータを並べ替えたとしても、ＰＣ9801のV30モードで２分以内、80286モードでは１分以内に作業を終える。非常に単純なプログラムで、これだけの早さで結果が得られるのはよいのだが、先にも述べたように、扱えるデータ量に制約があるため、大量のデータの処理には適しない。

そこで、次のプログラムは、作業時間がかなりかかっても、データ量の制約を受けない、一般にソート・マージと呼ばれる方法を用いたものである。これは、一定量ずつのデータを先頭から順番に並べ替えて（ソート）、ディスケットに登録しておき、その後、登録したデータ間で比較を行いつつ、さらに並べ替えを行う（マージ）ものである。この方法の場合、特に、後半のマージ部分でかなりの時間を要する。

```
10 REM ＊＊　ソート・マージ　＊＊
20 CONSOLE ,,,1:FI=252:DN=18:DL=14
30 LEG=1:SK$="":SK$=RIGHT$(STR$(LEG),1)
40 CLS:COLOR 4:LOCATE 20,0:PRINT "ファイル ＫＡＮＤＩＣ "+SK$" .ＤＩＣ　の　ソート":COLOR 7:PRINT
50 DIM II(100),ML(100),RC(100)
60 REM ＊＊　データの読み出し　＊＊
70 OPEN "B:DIC1.DIC" AS #1:FIELD #1,FI AS K$:TRD=LOF(1):GET #1,TRD
80 FOR A=1 TO DN:IF ASC(MID$(K$,(A-1)*DL+1,DL))=32 THEN 100
90 NEXT
100 KES=(TRD-1)*DN+(A-1):BL=100:NN=0
110 REM ＊＊　100データずつソート：登録　＊＊
120 OPEN "B:WORKFILE" AS #2:FIELD #2,FI AS KK$:K1=1:K2=BL
130 IF KES<=K2 THEN K2=KES
140 GOSUB 400
150 NN=NN+1:RC(NN)=K2-K1+1:IF K2<KES THEN K1=K1+BL:K2=K2+BL:GOTO 130
160 CLOSE
170 KILL "B:DIC1.DIC"
180 REM ＊＊　ワークファイル・オープン　＊＊
190 OPEN "B:WORKFILE" AS #1:FIELD #1,FI AS DK$:OPEN "B:DIC1.SOR" AS #2:FIELD #2,FI AS X$:L=0
200 REM ＊＊　マージする第１データの読み込み　＊＊
210 DIM B$(100)
220 FOR I=1 TO NN:II(I)=1:II=BL*(I-1)+1:KRD=INT(II/DN+(DN-1)/DN):KJJ=II-(KRD-1)*DN:GET #1,KRD:B$(I)=MID$(DK$,(KJJ-1)*DL+1,DL):NEXT
230 REM ＊＊　読み込みデータ中の最小データ　＊＊
240 MB=0:FOR I=1 TO NN:IF ML(I)>0 THEN 270 ELSE IF MB=0 AND ML(I)=0 THEN SI=I:MB=1:GOTO 270
250 DA$="":DUM$="":DA$=B$(I):DUM$=B$(SI):IF DA$<DUM$ THEN OP=1 ELSE IF DA$>DUM$ THEN OP=2 ELSE OP=1
260 IF OP=1 THEN SI=I
270 NEXT
280 REM ＊＊　最小データの登録　＊＊
290 IF ASC(B$(SI))=32 THEN 320
300 L=L+1:KRD=INT(L/DN+(DN-1)/DN):KJJ=L-(KRD-1)*DN:LSET X$=LEFT$(X
```

```
$,(KJJ-1)*DL)+B$(SI):PUT #2,KRD
310 REM ＊＊　次のデータの読み込み　＊＊
320 II(SI)=II(SI)+1:IF II(SI)>RC(SI) THEN ML(SI)=1:GOTO 330 ELSE K
RD=INT((BL*(SI-1)+II(SI))/DN+(DN-1)/DN):KJJ=(BL*(SI-1)+II(SI))-(KR
D-1)*DN:GET #1,KRD:B$(SI)="":B$(SI)=MID$(K$,(KJJ-1)*DL+1,DL):GOTO
230
330 REM ＊＊　最終データのチェック　＊＊
340 FOR I=1 TO NN:IF ML(I)=0 THEN 230
350 NEXT
360 REM ＊＊　ソート・マージ終了　＊＊
370 CLOSE
380 KILL "B:WORKFILE"
390 END
400 REM ＊＊　ソート・ルーチン　＊＊
410 DIM B$(100)
420 FOR K=1 TO K2-K1+1:KRD=INT((K1+K-1)/DN+(DN-1)/DN):KJJ=(K1+K-1)
-(KRD-1)*DN:GET #1,KRD:IF ASC(MID$(K$,(KJJ-1)*DL+1,DL))=32 THEN 44
0
430 B$(K)=MID$(K$,(KJJ-1)*DL+1,DL)
440 NEXT K
450 D=K-1
460 D=INT(D/2):IF D<1 THEN 530
470 DD=(K-1)-D
480 FOR N=1 TO DD:J=N
490 IF B$(J)=<B$(J+D) THEN 510
500 SWAP B$(J),B$(J+D):J=J-D:IF J>=1 THEN 490
510 NEXT
520 GOTO 460
530 FOR K=1 TO K2-K1+1:KRD=INT((K1+K-1)/DN+(DN-1)/DN):KJJ=(K1+K-1)
-(KRD-1)*DN:LSET KK$=LEFT$(KK$,(KJJ-1)*DL)+B$(K):PUT #2,KRD:NEXT
540 ERASE B$:RETURN
```

以上、２つの並べ替えプログラムは、全角の平仮名・片仮名については、正しく５０音順に並べ替えを行う。しかし、漢字については、ＪＩＳコードによって処理されるため、５０音順に正しくは並べ替えられない。また、半角の片仮名については、濁点・半濁点が１文字扱いとなるため、例えば、「バ」は、「ハ」の後に順序よく並ばないこともあるし、さらに、「ヲ」は、「ア」よりアスキーコードが小さいため、正しく「ワ」の後に並ばず、「ア」より前に並ぶという不都合も生じる。そのための対策としては、並べ替えプログラム内に、そうした文字を処理する部分を加えるか、先に掲げたプログラム２によって全角文字に直すかの方法が考えられる。

６．おわりに

こうした報告が国語学や国文学関係の雑誌に馴染むものかどうか迷いつつも、いくつかのプログラムを掲げてきた。冒頭でも断わったように、誠に稚拙なプログラミングで用を達しないかも知れないが、国語学や国文学の研究者の間にもワードプロセッサーやパソコンが浸透しだした現在、手持ちのプログラムやデータを相互に提供しあうことによって、お互いの研究をより発展させ、活性化させることも必要ではないだろうか。そうした意味で、これまでに作成したプログラムのうち、いくつかをここに敢えて掲げた次第である。至らない点、ご不明な点があれば、是非ともご教示をいただければ幸いである。

なお、本誌23号でも若干ふれたが、旺文社刊『古語辞典』に掲載されている見出し語の内、主として、近世語を除いた約３万語（見出し語・漢字・品詞）のデータとその使用プログラムを登録したディスケットを提供する用意がある。ご希

望の方は、ディスケットの種類（２ＨＤ・２ＤＤ）、使用ＤＯＳの種類（ＭＳ－ＤＯＳ・ＦＭ16βの場合、ＣＰ／Ｍ86も含む）を明記の上、ディスケット代及び送料（3,000円）を、本学国文学科西端宛にご送付を願います。

○

末尾ながら、本学教授木村三四吾先生のご退休を惜しむと共に、名誉教授の称号をお受けになったことを心からお喜び申し上げる。また、先生の今後のご健康とご活躍を祈念するものである。

○

注①　語句データ内の構造は、以下のように設定した。使用データによって、適宜変更できる。なお、DN（１レコード内のデータ数）以外の各数値の単位は、バイト。全角文字数は、各数値を２で割る。

	語句	読み	漢字	品詞	DL	DN	FI
DIC1.SOR	2	8	2	2	14	18	252
DIC2.SOR	4	12	4	2	22	11	242
DIC3.SOR	6	14	6	2	28	9	252
DIC4.SOR	8	14	8	2	32	7	224
DIC5.SOR	10	14	10	2	36	7	252
DIC6.SOR	12	14	12	2	40	6	240
DIC7.SOR	14	14	12	2	42	6	252
DIC8.SOR	16	16	14	2	48	5	240