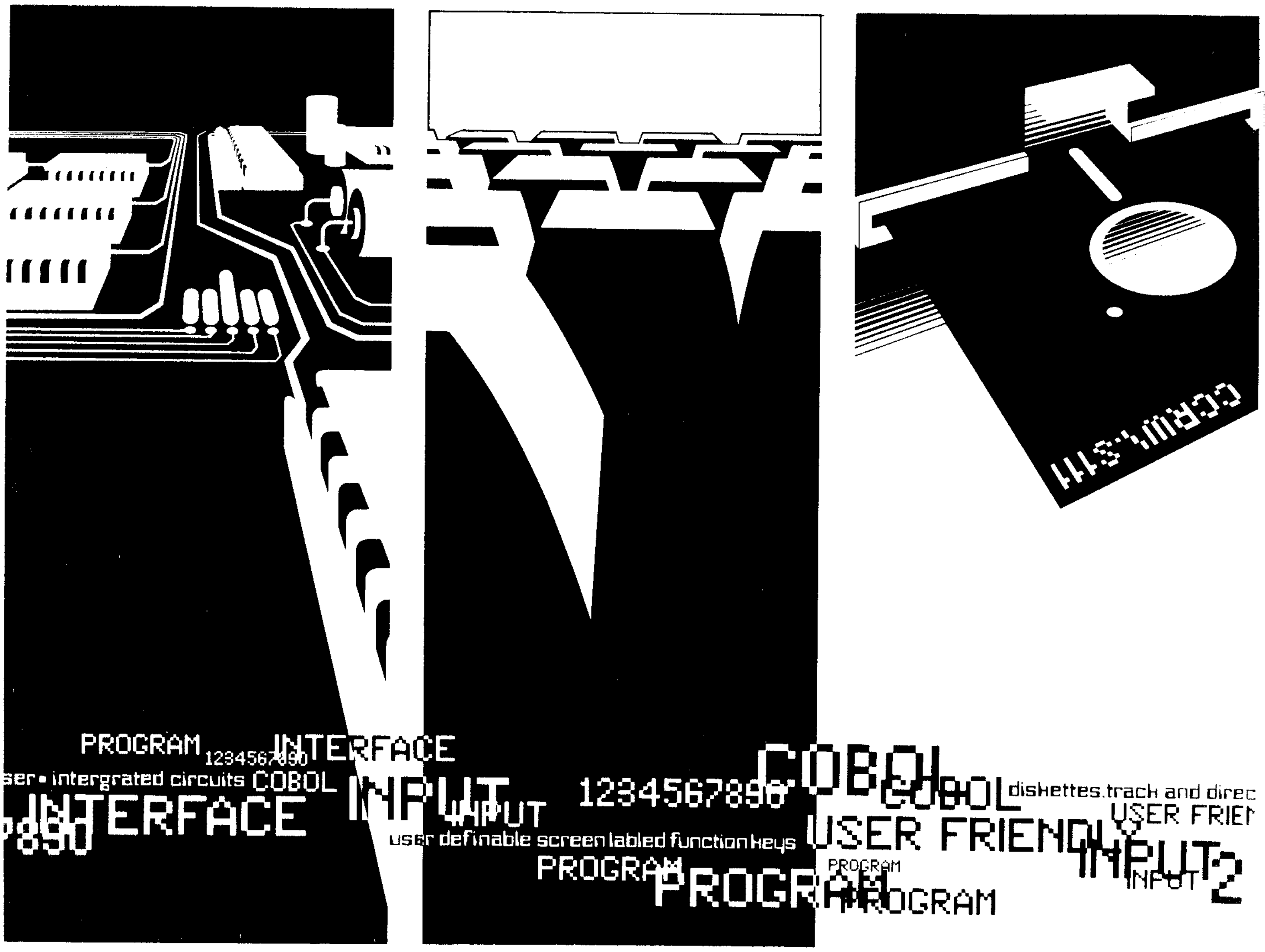
# RESEARCH IN WORD PROCESSING NEWSLETTER

South Dakota School of Mines and Technology Rapid City, SD 57701 (605) 394-2481
Volume 5 Number 7 October 1987

***WordCrunching***
**TEXTUAL DATABASES**

***page 15***


PROGRAM
INTERFACE
COBOL
INPUT
1234567890
USER FRIENDLY
user definable screen labled function keys
PROGRAM
diskettes.track and direc

RESEARCH IN

**WORD PROCESSING**

NEWSLETTER

***Editors***
Bradford A. Morgan
James M. Schwartz

***Contributing Editors***
Bryan Pfaffenberger
University of Virginia
Tom Carney
University of Windsor

***Production Manager***
Dan Dorland

***Art Director***
Francis Jacobs

***Typesetting/Graphics***
Brenda Boyer

***Photography***
Dave Adams

***Printing***
Dick Beshara

***Circulation***
Wendy Painter

***Accounting Manager***
Sondra Wagner

***Research in Word Processing Newsletter.*** Volume 5, Number 7. Copyright © 1987 by the South Dakota School of Mines and Technology. All rights reserved. ISSN: 0748-5484. Also indexed by **ERIC, INSPEC, COMPUTER LITERATURE INDEX,** and **SOFTWARE REVIEWS ON FILE.** The *Research in Word Processing Newsletter* is published 9 times a year (September through May) by the South Dakota School of Mines and Technology, Rapid City, South Dakota 57701-3995; (605)394-2481. Postage paid at Rapid City, South Dakota. **SUBSCRIPTION:** $18 per year (U.S.); $24 per year (Canada); $27 per year (foreign). Address all subscription inquiries and manuscripts to the Editors, *Research in Word Processing Newsletter*, SDSM&T, 501 E. St. Joseph, Rapid City, SD 57701-3995. Please allow 4 to 6 weeks for subscription processing. **POSTMASTER:** Send address changes to RWPN, South Dakota School of Mines and Technology, 501 East St. Joseph, Rapid City, South Dakota 57701-3995.

# A Computer Program for Word Processing

***Eric Johnson***

Readers may use this article in three different ways. First, by following a discussion about how a relatively simple computer program can be modified to perform increasingly sophisticated tasks, those who have little interest in learning to program can nevertheless gain insights into how computers deal with text. Second, the article should be useful for those who do want to learn to program in SNOBOL4. Third, since the finished program is provided near the end of the article, a free piece of software is available to anyone who wants to take the time to enter it.

I have used the SNOBOL4 programming language because it is simply the most powerful and easiest computer language to use for processing of natural language text. The programs and segments have been tested on microcomputer using a SNOBOL4+ compiler available from Catspaw, Inc., P.O. Box 1123, Salida, Colorado 81201; there are other good compilers, but they may require minor changes in the code.

## WORD-FREQUENCY PROGRAM

Following is a SNOBOL4 program which produces a word-frequency list for a text. I will refer to it as PGM1.

```
        LTRS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
+       "abcdefghijklmnopqrstuvwxyz'-0123456789"
        WORD.PATTERN = BREAK(LTRS)  SPAN(LTRS)  .  WORD
        INDEX = TABLE( )
READ    LINE = INPUT                                         :F(OUT)
NEXTW   LINE    WORD.PATTERN =                               :F(READ)
        INDEX<WORD> = INDEX<WORD> + 1                        :(NEXTW)
OUT     J = SORT(INDEX,1)
PRINT   S = S + 1
        OUTPUT = J<S,1>  '    '  J<S,2>                      :S(PRINT)
END
```

**PGM 1**

PGM1 works in this way. The first three lines define what is to be considered a word; the first line and the second line (which is actually a continuation of the first line) list the possible constituent parts of a word (the LTRS): any combination of the twenty-six lower-case letters, the twenty-six capital letters, the apostrophe, the hyphen, and the ten numbers. The third line builds a word pattern (called WORD.PATTERN); it specifies that the computer should assign to a variable named WORD any span of the LTRS, and it should discard anything else. The fourth line establishes a table called INDEX which will be used later to hold the words as they are discovered and the count of each. Line five, which starts with the label READ, is the real beginning of the program; it reads in a line of the text to be counted. Line six removes one word after another from the text line by using the WORD.PATTERN created earlier; when there are no further words in the line, the fifth line of the program is executed again to read the next line. The counting of words is done by line seven: in the table called INDEX, words and their count are stored.

SNOBOL4 allows words to be used as subscripts for a table; thus if the first word encountered is "A", one is added to the Aish element of INDEX. When there is no further input to process, the program converts the table INDEX into the array called J, and at the same time sorts the words in it in alphabetical order (line eight). Each word and its corresponding count are output in line ten. When the end of the array is reached, the program drops to the END label and stops.

Using the title and first paragraph of this article as the input text, a portion of the output from this program is as follows:

```
A   1
Computer   1
Eric   1
First   1
Johnson   1
Processing   1
Program   1
Readers   1
SNOBOL4   1
Second   1
Third   1
Word   1
a   3
about   1
anyone   1
article   3
available   1
be   2
by   1
can   2
```

Two improvements are obviously needed. First, it would look neater and be easier to read the output if the numbers were in a straight column. This is easy to do. The last line in the program before the END statement simply outputs the two parts of the array called J, with three spaces between them. We can replace it with the following to line up the output:

```
  OUTPUT = RPAD(J<S,1>,19) LPAD(J<S,2>,2) :S(PRINT)
```

The function RPAD puts spaces on the right of the word to pad it to nineteen characters (I am assuming that no word in the text is longer than eighteen letters), and LPAD puts a space on the left of the number to pad it to two characters (I am assuming that no number is longer than two digits).

A second improvement is essential to the accuracy of the counts. If a word happens to be capitalized, the elves inside the computer do not recognize it as the same word in lower-case letters. For example, the indefinite article, A, occurs four times in my input text; it is listed as "A" once and as "a" the other three times. If we replace all lower-case letters with capitals as soon as a line of text is read in, then the counts will be accurate. The following in the proper places will do the replacement:

```
  UP = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  LOW = "abcdefghijklmnopqrstuvwzyz"
  LINE = REPLACE(LINE,LOW,UP)
```

PGM2, following, is PGM1 with both improvements made:

```
          UP = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
          LOW = "abcdefghijklmnopqrstuvwxyz"
          LTRS = UP "'-0123456789"
          WORD.PATTERN = BREAK(LTRS)  SPAN(LTRS)  .  WORD
          INDEX = TABLE( )
READ      LINE = INPUT                                   :F(OUT)
          LINE = REPLACE(LINE,LOW,UP)
```

```
NEXTW     LINE    WORD.PATTERN =                                  :F(READ)
          INDEX<WORD> = INDEX<WORD> + 1                           :(NEXTW)
OUT       J = SORT(INDEX,1)
PRINT     S = S + 1
          OUTPUT = RPAD(J<S,1>,19) LPAD(J<S,2>,2)                 :S(PRINT)
END
```

Line one defines upper-case letters (called UP), and line two defines lower-case letters (called LOW). When a line of text is read (line six of PGM2), line seven of the program replaces any instances of lower-case letters in the line with the corresponding upper-case letters. LTRS has been redefined (line three) as any combination of upper-case letters, the apostrophe, the hyphen, and the numbers. A part of the output of PGM2 looks like this:

```
A              4
ABOUT          1
ANYONE         1
ARTICLE        3
AVAILABLE      1
BE             2
BY             1
CAN            2
COMPUTER       2
COMPUTERS      1
DEAL           1
DIFFERENT      1
DISCUSSION     1
DO             1
END            1
ENTER          1
ERIC           1
FINISHED       1
FIRST          1
FOLLOWING      1
FOR            2
```

**AN INDEX PROGRAM**

With a few changes in PGM2, we can transform it from a word-frequency program to an index program to list the words by page number. Of course, we will have to prepare the text file so that it lists the page number at the start of each page, and the lines listing the page numbers will be identified with an asterisk in column one thus:

```
*                                                                  Page 1
```

Following is PGM3, an index program.

```
          UP = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
          LOW = "abcdefghijklmnopqrstuvwxyz"
          LTRS = UP  " '-0123456789"
          WORD.PATTERN = BREAK(LTRS)  SPAN(LTRS)  .  WORD
          PAGE.DES = "0123456789"
          INDEX = TABLE( )
READ      LINE = INPUT                                            :F(OUT)
          LINE = REPLACE(LINE,LOW,UP)
          LINE "*" POS(1)                                         :F(NEXTW)
          LINE "PAGE  " SPAN(PAGE.DES)  .  PGNO                   :(READ)
NEXTW     LINE WORD.PATTERN =                                     :F(READ)
          INDEX<WORD> = INDEX<WORD> "    "   PGNO ","             :(NEXTW)
OUT       J = SORT(INDEX,1)
PRINT     S = S + 1
          OUTPUT = RPAD(J<S,1>,19) LPAD(J<S,2>,2)                 :S(PRINT)
END
```

Except for the addition of line five, which is a definition of what may be the actual page number or designation (PAGE.DES = "0123456789"), the first eight lines of PGM3 are identical with those of PGM2. Line nine checks for an asterisk in column one. If it is found, line ten searches the remainder of the line for the page number (the number following "PAGE ") and stores it as PGNO. A line of text that does not start with an asterisk has each word extracted from it by PGM3's line eleven (which begins with the label NEXTW). Each word is then stored in INDEX with its page number: the current value of PGNO. A space is added before the number and a comma after it. The last three lines convert INDEX to an array and sort it and print it exactly like the last three lines of PGM2.

Following is a portion of the output produced when this article is used as input and each paragraph is numbered Page 101, Page 102, etc.

```
FIRST               102, 106, 106, 106, 106, 106, 107, 113, 116, 132,
FIT                 132,
FIVE                131,
FIVE-AND-ONE-QUARTER-INCH 135,
FOLLOW              121,
FOLLOWED            127,
FOLLOWING           102, 105, 107, 108, 109, 113, 114, 123, 124, 128, 133,
135,
FOLLOWS             106,

   .
   .
   .

TABLE               106, 106, 106, 106, 115, 118, 132,
TAKE                102, 122,
TASKS               102,
TELL                119, 122,
TELLING             130, 132,
TEN                 106, 113,
TEST                116, 116, 131,
TEST2               131,
TESTED              103,
TESTP               116,
TESTPG              131,
TEXT                102, 103, 105, 106, 106, 106, 107, 108, 108, 110, 112,
113, 121, 125, 127, 134, 134, 135,
THAN                107, 107, 121, 131,
THANK               122,
THAT                106, 107, 107, 112, 113, 115, 121, 122, 124, 124, 130,
130, 131, 132, 133, 133, 134, 134, 134, 134, 134, 135, 135,
THE                 102, 102, 102, 102, 102, 103, 103, 103, 103, 106, 106,
106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106,
106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106,
106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 106, 107, 107, 107, 107,
107, 107, 107, 107, 107, 107, 107, 107, 107, 107, 107, 108, 108, 108, 108,
108, 108, 108, 108, 108, 108, 108, 110, 110, 110, 110, 110, 110, 111, 112,
112, 112, 112, 112, 112, 113, 113, 113, 113, 113, 113, 113, 113, 113, 113,
113, 113, 114, 115, 115, 115, 115, 115, 115, 115, 116, 116, 116, 116, 117,
117, 117, 118, 118, 118, 119, 119, 119, 119, 119, 120, 120, 120, 120, 120,
121, 121, 121, 121, 121, 121, 121, 121, 121, 122, 122, 122, 122, 122, 122,
123, 124, 124, 124, 124, 125, 125, 125, 125, 125, 125, 125, 125, 125, 125,
125, 126, 126, 126, 127, 127, 127, 128, 128, 128, 128, 129, 129, 129, 129,
129, 129, 130, 130, 130, 130, 130, 130, 130, 131, 131, 131, 131, 131, 131,
131, 131, 131, 131, 131, 132, 132, 132, 132, 132, 132, 132, 132, 132, 132,
132, 132, 132, 132, 132, 132, 132, 132, 132, 132, 133, 134, 134, 134, 134,
134, 134, 134, 134, 134, 134, 134, 135, 135, 135, 135, 136, 136, 136, 137,
137, 137, 137,
THEIR               106,
```

As always, the output contains exactly what the program told the computer to do. Since the word "table" occurs four times on page 106, the number "106" is listed four times. Perhaps that is what we want. However, do we really want to list the page number for every occurrence of "the"? In an index, a page number is usually listed only once regardless of whether a word occurs once or many times on a page.

In order to list a page number only once, we should test to see if the number is already stored before we add it again. To do this, first we construct the pattern we will test for (a space, the page number, and a comma) and assign the pattern to TESTP:

```
TESTP = " " PGNO ","
```

Next, using a powerful feature called pattern matching for which SNOBOL4 is famous, INDEX is checked to see if the page number is there; if so, the program will go on to the next word, if not, it is added to INDEX:

```
INDEX<WORD> TESTP          :S(NEXTW)
```

When these lines are included in the program, the page numbers for words like "table" and "the" will be listed only once.

However, do we want the page references for "the" to appear in the index at all? Chances are we want the index to ignore articles, prepositions, and similar function words. Well, the elves will not know which words are to be ignored unless we somehow tell them.

We can fill a file with the words we do not want to index. Then at the start of the program, read in these words and store them. Next, continue as before, but before adding the page number for a word, check to see if it is on the list of words to not index.

Also, looking back at the output of PGM3, we might notice the the format could be neater. When the list of page numbers is longer than a single line, it should continue under the other numbers, not in column one under the words. Also, it is odd to see a comma follow the last number. Moreover, something should be done for very long words encountered in the text that run into the listing of page numbers (such as "five-and-one-quarter-inch"); words over about eighteen letters are almost always hyphenated nonce words that can be ignored in an index.

Since SNOBOL4 programs sometimes take a while to run, especially if the data files are large, it would be nice to have messages appear on the screen to tell the user that the program is progressing nicely, thank you. If the output is sent to a file, a final message on the screen should identify the file.

All of these improvements, and a few others, I have incorporated into the following program, which I call BITZER. (I like to name programs after literary characters, usually bunglers from Dickens' novels; Bitzer is a well-crammed pupil in *Hard Times*. PGM1, PGM2, and PGM3 are too short to deserve a name.)

The following lines of code can be entered into a file (which should have the extension .SNO) using any word processor that produces ASCII output. BITZER then can be used to produce an index—provided you have a SNOBOL4+ compiler and have created the necessary files (in this listing, PAGES.DAT contains the document with page numbers to be indexed; NOWORDS.DAT contains a list of words that will not be indexed).

```
*     Part 1: Initialization
*
              INFILE = "PAGES.DAT"
              INWORDS = "NOWORDS.DAT"
              OUTFILE = "BOUT.DEX"
*
              INPUT(.INPUT,1,R,INFILE)
              INPUT(.IN,2,R,INWORDS)
              OUTPUT(.OUTPUT,3,W,OUTFILE)
              OUTPUT(.TOTERM,4,W,"CON")
              &ANCHOR = 0
              &TRIM = 1
              UP =  "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
              LOW = "abcdefghijklmnopqrstuvwxyz"
              NUMBERS = "0123456789"
              LTRS = UP  NUMBERS  "'-><"
```

```
            SPECIAL = "@" | "*" | "#"
            PAGE.DES = UP  NUMBERS " "
            WPAT = BREAK(LTRS) SPAN(LTRS) . WORD
            INDEX = TABLE()
            NOTABLE = TABLE()
            SP = DUPL(" ",20)
*
            DEFINE('CLEAR(MAX)N')                    :(START)
CLEAR       N = 0
SCROLL      TOTERM = LT(N,MAX)                       :F(RETURN)
            N = N + 1                                :(SCROLL)
*
*   Part 2: Read in words not indexed
*
START       CLEAR(25)
            TOTERM = SP "Bitzer is indexing the text."
            TOTERM =
            TOTERM = SP "Please do not interrupt."
            CLEAR(10)
*
GETWDS      ULWORDS = IN                             :F(CONV.NO)
            WORDS = REPLACE(ULWORDS,LOW,UP)
GETW        WORDS    WPAT   =                        :F(GETWDS)
            NOTABLE<WORD> = 1                        :(GETW)
*
CONV.NO     NOARRAY = CONVERT(NOTABLE,'ARRAY')       :S(READ)
            TOTERM = "Error converting NOTABLE; no no.words?" :(END)
*
*   Part 3: Main Processing
*
READ        INLINE = INPUT                           :F(PRINT)
            LINE = REPLACE(INLINE,LOW,UP)
            LINE SPECIAL POS(1)                      :F(NEXTW)
            LINE 'PAGE ' SPAN(PAGE.DES) . PGNO       :(READ)
*
NEXTW       LINE WPAT =                              :F(RESET)
            &ANCHOR = 1
            NUM = 1
TEST        NO.WORD = NOARRAY<NUM,1>                 :F(ENDTEST)
            WORD NO.WORD  RPOS(0)                    :S(NEXTW)
            NUM = NUM + 1                            :(TEST)
ENDTEST     &ANCHOR = 0                              :(TEST2)
RESET       &ANCHOR = 0                              :(READ)
*
TEST2       WORD  ANY(UP)                            :F(NEXTW)
            GT(SIZE(WORD),18)                        :F(TESTPG)
            TOTERM = 'Word over 18 letters: "'
+           WORD    '"; it is ignored.'
            TOTERM =                                 :(NEXTW)
*
TESTPG      TESTP =  " " PGNO  ","
            INDEX<WORD>  TESTP                       :S(NEXTW)
            INDEX<WORD> = INDEX<WORD> " " PGNO "," :(NEXTW)
*
*   Part 4: Output
*
PRINT       CLEAR(25)
            TOTERM = SP "Bitzer is alphabetizing the index."
            TOTERM =
              TOTERM = SP "Please do not interrupt."
              CLEAR(10)
              INDEXA = SORT(INDEX)                     :S(SORTED)
              OUTPUT = 'THERE IS NOTHING IN TABLE!'    :(END)
*
SORTED        OUTPUT = 'WORD                PAGE NUMBERS'
              OUTPUT = ' '
PRINTW        S = S + 1
              C1 = INDEXA<S,1>                         :F(LAST)
              C2 = INDEXA<S,2>
              LT(SIZE(C2),59)                          :F(FORMAT)
REMOVE        C2 RTAB(1)  . WO
              OUTPUT = RPAD(C1,19)  WO                 :(PRINTW)
*
FORMAT        TN = 57
              C2 LEN(TN)  . PART  =
FROMT         PART  TAB(TN - 1)  LEN(1)  . C
              IDENT(C,',')                             :F(TRADE)
              PART  LEN(TN) . NEWPART
              OUTPUT = RPAD(C1,19)  NEWPART
              C1 = DUPL(' ',19)
              LT(SIZE(C2),59)                          :S(REMOVE)F(FORMAT)
TRADE         TN = TN - 1
              TEMP.PART = PART
              TEMP.PART TAB(TN)  LEN(1)  . D  =
              C2 = D C2                                :(FROMT)
*
LAST          CLEAR(25)
              TOTERM = SP "Bitzer is finished."
              TOTERM =
              TOTERM = SP "An index of " INFILE
              TOTERM = SP "excluding the words in " INWORDS
              TOTERM = SP "will be found in " OUTFILE
              CLEAR(9)
END
```

## HOW BITZER WORKS

Part 1 of BITZER.SNO initializes all the variables and sets the values needed in the program. INFILE is assigned the name of the file to be indexed; INWORDS should be set for the file of words to not index; the programs's output will be sent to the file assigned to OUTFILE. Any of these assignments may contain drive letters and path names if they are needed by your microcomputer (eg., INFILE = "C:/DATA/TEXT/PAGES.DAT"). The next four lines make internal assignments for input and output; in addition to the three files just mentioned, output is also sent to the screen when TOTERM is used.

When &ANCHOR is set equal to 1, pattern matching will succeed only if the two words match letter for letter starting at letter one. Thus, set at 1, "get" will not match with "together." When &ANCHOR is set to 0, the match can succeed at any point; thus "get" is found in "together." &ANCHOR is set at 1 and at 0 at different points in the program.

When &TRIM is set to 1, trailing spaces at the end of an input line are discarded. A line of text read into BITZER may contain, say, thirty letters followed by fifty spaces. It makes little sense to try to find words among the fifty spaces. If &TRIM is set at 1, the fifty trailing spaces are discarded upon input.

SPECIAL is the holder of the special character which when found in column one will signal a line containing a page number; SPECIAL is set to any one of three characters: @ or * or #. PAGE.DES, the page number or designation, is set to contain any combination of letters (defined as UP) or NUMBERS or space. Thus a page number may be any of the following: 12, 13A, 14 A. SP is defined as twenty spaces; it is used later to start messages in column twenty-one.

The last four lines of code in Part 1 construct a function to clear the screen by sending blank lines to it; since the screen contains twenty-five lines, CLEAR(25) will clear it; after a message is printed, CLEAR(10) can be used to clear about half the screen and thus push the message up to about the center.

Part 2 of the program clears the screen, prints a message telling Bitzer's progress, reads in the words that will not be indexed, and stores them. The process of reading the file of words not to index and storing them is almost identical to that used later to read the file to be indexed. If something goes wrong, a message is printed, and the program ends.

Part 3 is the main processing of BITZER. It proceeds much like PGM3, but it makes additional checks. The five lines starting with the label TEST check each word against those read in to not be indexed. The line with the label TEST2 discards any "words" that are simply strings of numbers; "SNOBOL4" will be indexed, but "1234" will not. The next four lines ignore words with more than 18 letters; a message is sent to the screen if a such word is ignored. The last three lines (starting with the label TESTPG) check whether the most recently discovered page number for a word is already in the index; if so, it is not added again.

Part 4 handles output. It first clears the screen and sends a new message. The index is converted from a table to an array, and the array is sorted alphabetically. Should this operation fail—probably because there is nothing in the input file—an appropriate message is sent to the output file. The two lines starting with the label SORTED print a heading. The next eighteen lines format and print the output index. The length of the string of page numbers is checked, and, if it is too long to fit on the line, it is broken into parts that are stacked neatly starting in column twenty-one. Regardless of the length of the string of page numbers, the comma at the end of the last number is removed. Last, the screen is cleared again and a final message telling the disposition of the files is printed.

Following is a selection of the output from BITZER.

```
WORD                PAGE NUMBERS

IDENTITY            3
IF                  1, 2, 3, 4, 10, 101, 202, 707, 708, 709, 710, 711, 1022,
                    1023, 1024, 1025, 1026, 1027, 1028
IMPORTANT           101
INDEX               1028, 1028A
INSTANCE            3
INTERESTED          202
INTERESTING         10
ISOLATES            4
JANE                4
JOHNSON             0, 1, 2, 3, 4, 101, 202, 707, 708, 709, 710, 711, 1022,
                    1023, 1024, 1025, 1026, 1027, 1028, 12345, 12346, 12347
LAST                202
LIBERAL             1
LIST                2 B
LISTENERS           1
LITERACY            1
LITERATURE          1, 2 B
```

It can be noticed that BITZER can keep track of page numbers or designations that contain letters and spaces as well as numbers.

Although there may be no improvements that are needed in the nature and format of the output, there is an addition that would help the program run much more quickly. As it stands, the computer must check each of the words of the text against each of the words on the list that are not to be indexed. If that list were put in alphabetical order, code could be written to check words of the text against the list more rapidly. I leave that to the interested reader to add to a new version of BITZER.SNO.

I enjoy programming and sometimes even consider it an end in itself (something like solving riddles). However, I realize that many do not feel that way. In any case, following the construction of a program like BITZER should give insights into how the elves in a computer deal with text. Readers who do want to learn to program in SNOBOL4 should read one or both of the books mentioned in the next paragraph. I will send a copy of BITZER to anyone who sends me a five-and-one-quarter-inch disk; it will run only on MS-DOS microcomputers.

**FOR MORE INFORMATION**

Several introductory books about SNOBOL4 have been written. Susan Hockey's *SNOBOL Programming for the Humanities* (Oxford University Press) and *A SNOBOL4 Primer* by Ralph and Madge Griswold (Prentice-Hall) are recommended. The two- hundred-and-fifty-page book of instruction which accompanies the Catspaw SNOBOL4 + compiler is very useful.

***Eric Johnson*** is Head of the Division of Liberal Arts at Dakota State College, Madison, SD 57042, and is the director of ICEBOL, the International Conference on Applications of SNOBOL and SPITBOL. His article, "SNOBOL4 Programming as Word Processing" appeared in the December, 1986, issue of RWPN.

## PostScript Newsletter Available

The merging of font-enriched text and picture-communicating graphics continues to extend the options of writers producing manuscripts with computers. A new sense of audience seems to be evolving, one in which the writer is becoming more involved with the visual impact of his ideas on the reader. Form and content, subject and structure—these familiar relationships are being reapplied to new interdisciplinary perspectives which ask writers to think more about printing, art, and other aspects of production. Communication is reader-centered, more and more concerned with the whole visual package housing what the reader experiences.

A newsletter which both describes and demonstrates this approach is *Colophon*, featuring news about the PostScript page-description language and about PostScript-related products. *Colophon* is produced entirely with PostScript software and hardware. Text is created with MacWrite on an Apple Macintosh Plus. Artwork is done with Adobe Illustrator. Pages are composed with Aldus PageMaker 2.0 on the Macintosh. Camera-ready film masters are produced on a Linotronic 300 typesetter. The entire process—from original text and images to final film for printing— requires no pasteup.

A no-cost subscription to *Colophon* can be arranged by contacting Liz Bond, Editor, *Colophon*, Adobe Systems Incorporated, 1870 Embarcadero Road, Palo Alto, CA 94303.

## NCTE Convention in Los Angeles

The 1987 Annual Convention of the National Council of Teachers of English is scheduled for November 20-25 in Los Angeles, California. Concurrent sessions will include ■ New Uses for Computers in Teaching Writing ■ Computers in the Composition Classroom ■ Integrating Technology and Human Resources: Word Processing and Peer Partnering in the Composition Classroom, and ■ Connecting via Computer Networks: Applications to Language Arts. In addition, one-day workshops will focus on ■ Creating a Computer-Supported Writing Process, and ■ Writing and Thinking in an Electronic World. Contact NCTE, 1111 Kenyon Road, Urbana, IL 61801.

## Positions Available in Computational Linguistics

Mead Data Central would like to see the resumes of computational linguists who would like to be applied researchers, programmers, or managers on projects involving ■ Natural-language research and advanced parsers/compilers ■ Knowledge representation schemes ■ parallel processing and large-scale transaction systems, and ■ Large-scale database architectures for textual data. Researchers have access to multi-billion character full-text databases to allow "real world" tests of processing research language. Contact A. J. Davis, Mead Data Central, P.O. Box 933, 9393 Springboro Pike, Dayton, OH 45401.

# Bibliography Update

*Bradford A. Morgan*

Abercrombie, John R. "Graphics Display of Foreign Scripts: The Graphics Project at the University of Pennsylvania Has Developed Routines for Generating Exotic Fonts on Microcomputers, As Well As a Full-screen Editor for Textual Research, for Correcting Optically Scanned Texts, and for Entering Exotic Scripts into Existing ASCII Files." *Perspectives in Computing.* 7:1 (Spring 1987), pp. 43-51.

Alderman, Eric. "The Original Gets Better: *PageMaker 2.0.*" *Macworld.* 4:9 (September 1987), pp. 150-152.

Anderson, Clifford W. et al. "Computer Assisted Enhancement of Emotional Tone Accuracy in Translated Narrative." *Literary & Linguistic Computing.* 2:1 (1987), pp. 1-6.

Baldrica, Christine. "Random Notes: *Memorandum* Note-making Utility." *Publish!* 2:7 (August 1987), pp. 91-92.

Beddow, Michael. "*Microsoft Word Version 3.0* for the IBM PC." *Literary & Linguistic Computing.* 2:1 (1987), pp. 34-36.

Beinhorn, George. "Up from *WordStar*: When You Upgrade to *WordPerfect* or *Microsoft Word*, You Can Take the Best of *WordStar* with You." *PC World.* 5:9 (September 1987), pp. 240-245.

Bell, Lydia V. "The Design and Implementation of Online Specifications." *STC Proceedings: 34th International Technical Communication Conference.* (Denver, CO: May 10-13, 1987). Washington, DC: Society for Technical Communication, 1987, pp. ATA/129-ATA/132.

Blodgett, Ralph. "Laser-Perfect: *WordPerfect 4.1* and the LaserJet Plus Went Through Some Rocky Times Together. *WordPerfect 4.2* Makes a Cozier Match." *PC World.* 5:9 (September 1987), pp. 158-163.

Bove, Tony and Cheryl Rhodes. "Are We Publishing Yet? Desktop Publishing Is Nibbling at the Edges of the Commercial Publishing World." *Publish!* 2:7 (August 1987), p. 101.

Brink, Daniel. "MLA's Endorsement of Nota Bene: A Dissent." *Scope.* 4:4 (July-August 1986), p. 38.

Caernarven-Smith, Patricia. "Computers and Composition: Does This Line Count?" *Technical Communication.* 34:3 (August 1987), pp. 165-167, 183. (contrasts page-oriented and galley-oriented word processors)

Callewaert, W. M. "Manuscripts, Archetypes and the Computer: A Report." *Literary & Linguistic Computing.* 2:1 (1987), pp. 44-46.

Cichanowski, Gerald W. "Croak Proof Kermit: This Tip Is One You'll Want To Save for Quick Reference the Next Time You Need To Transfer a File." *DEC Professional.* 6:8 (August 1987), pp. 66-67. (PC/VAX text file transfers)

Coyne, Luann J. et al. "Change Management System (CMS): A Computerized Editing Database for a Multivolume User Manual." *STC Proceedings: 34th International Technical Communication Conference.* (Denver, CO: May 10-13, 1987). Washington, DC: Society for Technical Communication, 1987, pp. ATA/133-ATA/136.

Cummings, Steve. "A Pair of 'Operating Environments' Help Turn Page in Desktop Publishing." *PC Week.* 4:34 (August 25, 1987), pp. 113, 118.

————. "File exchange a Key Desktop-Publishing Issue." *PC Week.* 4:34 (August 25, 1987), pp. 124, 126.

Dansereau, Mary E. "Creating an Online Index." *STC Proceedings: 34th International Technical Communication Conference.* (Denver, CO: May 10-13, 1987). Washington, DC: Society for Technical Communication, 1987, pp. ATA/105-ATA/107.

Day, A. Colin. "Linguistic Word Processing." *Literary & Linguistic Computing*. 2:1 (1987), pp. 40-43.

DiNucci, Darcy. "Preserving the Old Virtues: *Xywrite III* Word Processor." *Publish!* 2:7 (August 1987), pp. 89-90.

Donovan, Ronald John. "The Electronic Writer: The Answer Man. Our Columnist Dips into the Mailbag and Conducts Some Serious Business. And Some Not-so-serious Business." *Writer's Digest*. (September 1987), pp. 57-60.

Dunn, Nancy E., ed. "Insights on *Microsoft Word 3.0*: Savvy Readers Share Their Best Tricks and Tips for the New Version of *Word*." Macworld. 4:9 (September 1987), pp. 215-217, 220-224.

Eckhardt, Robert C. "Face to Fonts: Upscale Characters for the Mac. With PostScript Outline Fonts, the Look Is Elegant and Size Is No Object." *Publish!* 2:7 (August 1987), pp. 74, 77-80.

Felici, James. "Face to Fonts: Typecasting for the PC—Bit Parts: Until They Become More Versatile, Fonts for the PC Will Play a Limited Role." *Publish!* 2:7 (August 1987), pp. 75, 81-85.

Fenno, Charles R. "Interactive Online Editing: A Review of Current Techniques." *STC Proceedings: 34th International Technical Communication Conference*. (Denver, CO: May 10-13, 1987). Washington, DC: Society for Technical Communication, 1987, pp. WE/55-WE/58.

Frahmann, Dennis. "The Interpress Page and Document Description Language." *The PostScript Language Journal*. 1:2 (June 1987), pp. 28-36.

Gabaldon, Diana. "*Nota Bene* Word Processor Caters to Academic Community: Footnoting, Indexing, Style Manual Reference among Features." *InfoWorld*. 9:32 (August 10, 1987), pp. 53, 55.

Ganeshsundaram, P. C. "Processing of Japanese Kanji on a Microcomputer." *Computers and the Humanities*. 21:3 (July-September 1987), pp. 157-167.

Goldstein, Richard. "*PC TeX, Version 1.50f* and *Micro TeX, Version 1.5A1*: Public Domain Code Basis for Powerful Typsetting Package." *InfoWorld*. 9:33 (August 17, 1987), pp. 54-55.

Graham, Gordon J. "Common Sense Steps in Selecting a Desktop Publishing System." *STC Proceedings: 34th International Technical Communication Conference*. (Denver, CO: May 10-13, 1987). Washington, DC: Society for Technical Communication, 1987, pp. ATA/93-ATA/96.

Gruman, Galen. "*Ventura Publisher 1.1*: Xerox Desktop Publisher Now Clear Leader in DOS Market." *InfoWorld*. 9:32 (August 10, 1987), pp. 55, 58-60.

Heid, Jim. "The Desktop Publishing Shopper: The Key to Publishing Productivity Is Finding a Program That Meets Your Needs." *Macworld*. 4:9 (September 1987), pp. 124-131.

Hull, Glynda et al. "Computer Detection of Errors in Natural Language Texts: Some Research on Pattern-Matching." *Computers and the Humanities*. 21:2 (April-June 1987), pp. 103-118.

Hutchinson, Michelle I. "Choosing a Desktop Publishing System: Finding the Pearl in the Oyster." *STC Proceedings: 34th International Technical Communication Conference*. (Denver, CO: May 10-13, 1987). Washington, DC: Society for Technical Communication, 1987, pp. ATA/88-ATA/92.

Jantz, Richard. "Are You the *DeskSet* Type? Out of the Print Shop Comes *DeskSet*, a Rough-and-tumble Desktop Publishing Package That Promises Rewards for Those in Search of True Typesetting." *PC World*. 5:8 (August 1987), pp. 240-249.

Kaferly, D.H.A. "Working with *Nota Bene (Version 1)*." *Literary & Linguistic Computing*. 2:1 (1987), pp. 37-39.

Kidding, Noah. "Learning to Write with a Macintosh, or Saving the Soul with a New Machine." *Small Computers in Libraries*. 6:4 (April 1986), pp. 27-32.

Kircz, Joost G. and Jan Bleeker. "The Use of Relational Databases for Electronic and Conventional Scientific Publishing." *Journal of Information Science Principles & Practice*. 13:2 (1987), pp. 75-89.

Kohl, Louise. "Tools of the Trade: Doug Clapp's *Word Tools* Is Finally Here. Do You Still Want It?" *MacUser*. 3:9 (September 1987), pp. 136-143. (Review of punctuation and style checker)

Kollmeier, Harold H. and Kathleen Henderson Staudt. "Composition Students Online: Database Searching in the Undergraduate Research Paper Course." *Computers and the Humanities*. 21:3 (July-September 1987), pp. 147-155.

Krull, Robert. "Learning to Use Electronic Publishing." *STC Proceedings: 34th International Technical Communication Conference*. (Denver, CO: May 10-13, 1987). Washington, DC: Society for Technical Communication, 1987, pp. ATA/34-ATA/37.

Levine, Martin. "Beyond Records and Fields: Full-text Database Management Systems That Offer Fast Access to Mountains of On-line Information Are Gaining in Popularity." *Digital Review*. 4:16 (August 24, 1987), pp. 61-66.

Marcus, Jeffrey and Steven Brown. "*Pagemaker* Made Over: The New Version of *Pagemaker* for the Macintosh Is Smoother and Sleeker Than Ever." *Publish!* 2:7 (August 1987), pp. 62-67, 73.

Marcus, Judith H. "Implementation and Evaluation of Online Text Formatting." *STC Proceedings: 34th International Technical Communication Conference*. (Denver, CO: May 10-13, 1987). Washington, DC: Society for Technical Communication, 1987, pp. ATA/120-ATA/123.

McColly, William B. "Style and Structure in the Middle English *Cleanness*." *Computers and the Humanities*. 21:3 (July-September 1987), pp. 169-176.

McElroy, Mark. "The Case for Transferring Word Processing to PCs." *Journal of Information Systems Management*. 4 (Summer 1987), pp. 47-53.

Mealand, David L. "Word Processing in Greek Using *Vuwriter Arts*: A Test Case for Foreign Language Word Processing." *Literary & Linguistic Computing*. 2:1 (1987), pp. 30-33.

Mullins, Carolyn J. and Maureen Faust. "Buyers' Guide: Multilingual Word-Processing Programs: How Buyers Evaluate Multilingual Packages. What Buyers Want—Standard Support, Features, Upgrades Plus User-Friendliness—Is Not Easy To Get." *PC Week*. 4:33 (August 18, 1987), pp. 61-63, 66-69, 72, 75-77.

————. "Programs Must Be Multicultural To Work Effectively." *PC Week*. 4:33 (August 18, 1987), pp. 61, 63-64.

Nace, Ted et al. "101 Greatest Tips: Printing & Typesetting." *Publish!* 2:8 (September 1987), pp. 60-64.

"A New Start with Computers in Language Instruction." *Scope*. 5:1 (January-February 1987), pp. 1, 9.

Rardin, Kevin. "Nonstop Pages: *Xpress* Is a Quick Route to High-volume Page Design on the Mac, but Expect a Few Detours along the Way." *Publish!* 2:7 (August 1987), pp.68-73.

Rich, Joan. "Author, Author, Author! Can Groups Edit Better on the Computer? Do Many Hands Make Write Work? It Depends on the Group—and the Application." *PC World*. 5:8 (August 1987), pp. 250-256.

Rosenbaum, Daniel J. "*Wordstar* Plays Catch Up: *Wordstar Professional, Release 4* Word Processor." *Publish!* 2:7 (August 1987), pp. 87-88.

Rothman, David H. *XyWrite Made Easier: Taming the World's Fastest Word Processor*. Blue Ridge Summit, PA: TAB Books, 1987.

Sakhuja, Sanjay et al. "101 Greatest Tips: Text & Typographics." *Publish!* 2:8 (September 1987), pp. 44-49

Schanze, Helmut. "Writing, Literacy and Word-Processing: Changes in the Concept of Literature in the Framework of New Media." *Literary & Linguistic Computing*. 2:1 (1987), pp. 24-29.

Schnelling, Heiner. "Digitizing Shakespeare: Perspectives of Digital Optical Recording of Renaissance Editions in University Libraries." *Literary & Linguistic Computing*. 2:1 (1987), pp. 13-18.

Seiter, Charles. "Letters from Cell A17: Three Add-in Word Processors Let Monkish 1-2-3 Number Crunchers Write Epistles without Ever Leaving Their Cells." *PC World*. 5:9 (September 1987), pp. 192-199.

Sell, William. "By the Numbers: Converting Mathematical Formulas to Glossaries." *Access 87*. 5:1 (September 1987), p. 26.

Semple, Marlene C. "The Electronic Blue Pencil: Editing Online Information." *STC Proceedings: 34th International Technical Communication Conference*. (Denver, CO: May 10-13, 1987). Washington, DC: Society for Technical Communication, 1987, pp. ATA/140-ATA/142.

Smith, Charles R. "How the Writer's Workbench Came to Colorado State University." *Scope*. 5:3 (May-June 1987), pp. 30-31.

Stewart, Charles O. and Daniel J. Rosenbaum. *WordPerfect Tips, Tricks, and Traps*. Indianapolis, IN: Que, 1987.

Strauss, Ken and Averil. "PostScript and *FinalWord II*." *The PostScript Language Journal*. 1:2 (June 1987), pp. 42-45.

Strickland, Dorothy S. et al. "The Computer As Tool: Word Processing." *Using Computers in the Teaching of Reading*. New York: Columbia University Teachers College Press, 1987, pp. 12-30.

Stringer, Gary A. and William R. Vilberg. "The Donne Variorum Textual Collation Program." *Computers and the Humanities*. 21:2 (April-June 1987), pp. 83-89.

Suen, Ching Y. "Character Recognition by Computer and Applications." *Handbook of Pattern Recognition and Image Processing*. Tzay Y. Young and King-sun Fu, eds. Orlando, FL: Academic Press, 1986, pp. 569-586.

Terrizzi, Carol et al. "101 Greatest Tips: Page Makeup & Design." *Publish!* 2:8 (September 1987), pp. 56-59.

Wallia, C. J. "An Artificial Intelligence Program for Prewriting." *Technical Communication*. 34:3 (August 1987), pp. 175-176. (sample dialog from *Thoughtline* software)

Watzman, Suzanne and Marla Schay. "Page Makeover: Open Up the Design of a Technical Data Sheet To Let the Information Through." *Publish!* 2:7 (August 1987), pp. 60-61.

Weiner, E. Judith. "Computational Considerations for the Processing of Explanatory Literal Analogies and Expressive Metaphors." *Computers and the Humanities*. 21:2 (April-June 1987), pp. 91-101.

Wiggins, Robert R. "On the *Xpress* Track: If You Need Desktop Publishing Power, Check Out Quark *Xpress*. It's A Top-of-the-Line Product." *MacUser*. 3:9 (September 1987), pp. 92-100.

Wimberly, Lee W. "Artificial Intelligence in Software Documentation: What Does It Mean?" *STC Proceedings: 34th International Technical Communication Conference*. (Denver, CO: May 10-13, 1987). Washington, DC: Society for Technical Communication, 1987, pp. ATA/84-ATA87.

## THE PROFESSIONAL WRITER'S WORKSTATION

# Mainframe Text Analysis Journeys to Micros

***Bryan Pfaffenberger***

**Program:** *WordCruncher*

**Publisher:** Electronic Text Corporation, 5600 North University Avenue, Provo, UT 84604, (801) 226-0616.

**Category:** Text Analysis Program

**Requires:** IBM PC, XT, AT, or 100% PC-compatible, 512K RAM (640K recommended), DOS 2.1, (DOS 3.2 recommended), and two disk drives (hard disk recommended).

**Summary:** A superb text indexing, retrieval, and concordance-generation program that deserves a market far beyond the bounds of literary scholarship.

Literary scholarship, one supposes, is exclusively concerned with the life of the mind, lofty values, and penetrating insights—until you find out what really goes into text analysis. To produce a key-word-in-context concordance of a 250,000 word literary work by hand, for example, would require what can only be termed an inordinate affection for minute detail, the kind of affection required to produce a garage-sized structure using millions of toothpicks.

Small wonder indeed that, thirty years ago, a few literary scholars began making the lonely journey across the gulf dividing the Two Cultures. Annotated manuscripts in hand, they appeared before bewildered Directors of Academic Computing Centers, explaining their needs as best they could to their astonished audience: they wanted software that could index a text, handle foreign language characters, and provide fast and flexible access to key words throughout the text (and in context, please!).

As if that were not enough, the software should be able to generate word frequency lists, concordances, and indexes. It should, in short, do a very great deal of the grunt work automatically. I wasn't there, but I can imagine the programmer's eyes lighting up ("It CAN be done!")—although in many cases it was the literary scholars themselves who became the programmers. In any case, it wasn't long before software of this ilk (e.g., the *Oxford Concordance Program*) saw the light of the control panel and, housed on big reels of tape, was being bandied about between computing centers.

It may seem like a rather small event in the overall scheme of things, but mainframe concordance-generation software is now making its migration to microcomputers. Oxford, for instance, has just released *micro-OCP*, the IBM Personal Computer version of the redoubtable *Oxford Concordance Program*, which has concorded many a famed work in its day. The justly famed *Brigham Young Concordance Program* is now available in a remarkable microcomputer version, again for the IBM PC, called *WordCruncher*.

Attention spans being what they are (short), let me insist straightaway that every academic computing center, English department, anthropology department, and interested scholar ought to have *WordCruncher*—several copies, in fact. What might not be so obvious, however, is that the program has value far beyond its original milieu (the literary world), and I should like to belabor this point: virtually anyone who is obliged to work on a more or less routine basis with large amounts of text ought to have it, too.

*WordCruncher* comes in two parts, called IndexEtc and View Etc. IndexEtc is used to index a manuscript for retrieval purposes. Using a word processor (*WordPerfect*, ideally, since *WordCruncher* is designed to make use of some *WordPerfect* features), you begin by preparing the DOS text file for processing: you add codes that tell the program where page and paragraph breaks occur in the original, paper version of the manuscript. (You needn't conform to the page/paragraph notation; the text could be differentiated as one would code an outline, with up to three levels of subordination.) Then IndexEtc goes to work automatically, generating a word frequency list as it indexes the file. IndexEtc can handle foreign languages with ease. For a very large text file, IndexEtc may require some time to complete its job, but only one indexing pass is required per file.

Once the index is complete, ViewEtc provides ready access to the indexed text. Because ViewEtc is searching the index created by IndexEtc rather than the text itself, retrieval operations are very fast. The user can retrieve information using words or phrases, lists of words, two or more words in a defined context, substrings (suffixes or stems), or any logical combination of these options. Once retrieved, the text surrounding the targeted words or phrases is displayed in windows on the screen. If desired, the user can see (and page through) a whole-screen display of the original document. Additional commands generate book-length, key-word-in-context concordances and indexes. Program output can be printed or captured to a *WordPerfect Library* clipboard.

*WordCruncher*'s functions, obviously, parallel those of automatic indexing storage-and-retrieval programs such as *Zylndex* to some extent, but such programs are intended to do a rather different job, and the difference should be made clear. *Zylndex* helps users cope with a multitude of text files, files that have proliferated throughout one's hard disk and whose contents have been forgotten. So that you can tell quickly where some text is that you want, a program such as *Zylndex* provides very fast access to the text stored in a multitude of files, but offers relatively primitive retrieval and review functions.

*WordCruncher*, in contrast, is designed to concentrate on just one text file at a time (or, if it is very large, several related text files that the program links together). And it provides a highly sophisticated and diverse battery of procedures to retrieve and review the indexed text. In short, *WordCruncher* enables close analysis of an authoritative text, rather than superficial and rapid scanning of dozens of text files.

It should be obvious from this description that many people, not just literary scholars, will find *WordCruncher* of value. An appendix to the almost-excellent manual (more on that anon) suggests *WordCruncher* strategies for indexing and retrieving bureaucratic memos, policy handbooks, research notes, genealogical data, and even protein formulas. Virtually anyone who works with a bulky, authoritative text, one that must be subjected to close analysis, will find *WordCruncher* of value. Coupled with extensive manual-based tutorials, the program's highly graphic and visual interface makes it reasonably approachable.

Be forewarned, though. *WordCruncher* is a huge, complex program; it comes on five disks, and it will require 1.2 megabytes of hard disk space. It's just barely operable on a dual-floppy system (one has to swap program disks at many program junctures). All this complexity naturally translates into cognitive overload as one attempts to grasp the program and put it to work. The manual, to be sure, is a model of excellence in contemporary technical writing: it uses illustrations very well and breaks up the space on the page to aid the eye. Where it falls down, however, is in providing a sufficiently lucid overview of what program functions do and why one would use them. Such an overview for the entire program was added as an afterthought, perhaps in response to criticism; yet the tutorial and reference sections of the manual need them too.

Not everyone who uses *WordCruncher*, after all, will have a training in literary scholarship, and many program operations will seem mysterious to someone who has never been introduced to such arcana. But this criticism is minor (and the flaws will, I am sure, be remedied soon enough). *WordCruncher* deserves no less than unstinting praise: it opens a new chapter in the increasing utility of microcomputers for those whose primary interests are in any way related to the creation, analysis, and retrieval of text.

Contributing Editor **Bryan Pfaffenberger's** books include *The Scholar's Personal Computing Handbook* (Little, Brown, 1986), *Personal Computer Applications* (Little, Brown, 1987), *Business Communication in the Personal Computer Age* (Richard D. Irwin, 1987), and *Microcomputer Applications in Qualitative Research* (Sage University Papers in Qualitative Research, forthcoming).

## Manuscript Submissions Welcome

The *Newsletter* welcomes article submissions that pertain to word-processing, text-analysis, and research applications in professional writing situations. Also, hardware and software reviews are encouraged, but please contact Dr. Jim Schwartz, Hardware/Software Review Editor, **before** submitting them (call Jim at 605-394-1246). Manuscripts may be submitted either as hard copy or on 5¼" diskettes using *XEROX Ventura Publisher, WordStar, WordPerfect*, DCA, or standard ASCII code. If submitting disks, please make sure they are formatted either in MS-DOS, PC-DOS, or a popular CP/M format (Kaypro, Zenith, etc.) The Editors reserve the right to edit manuscripts, if necessary. If you want your manuscript or diskette returned, please send enough postage to cover the return along with a self-addressed envelope. Address all correspondence to the Editors, *Research in Word Processing Newsletter*, South Dakota School of Mines and Technology, 501 E. St. Joseph, Rapid City, SD 57701-3995. Jim Schwartz may also be reached on *CompuServe* (70177,1154).

# RWPN Back Issues

**Vol. 5 No. 6 [Sept. 1987], 20 pp.**
Building Text Filters in *Turbo Pascal*; News & Notes; Bibliography Update; On the Meaning of the Term "Desktop Publishing"

**Vol. 5 No. 5 [May 1987], 44 pp.**
From Word Processing to Desktop Publishing and CD ROM: A Five-Year Bibliographic Perspective on the Impact of Computers on Writing and Research

**Vol. 5 No. 4 [Apr. 1987], 16 pp.**
"Desktop Publishing": Some Semantic Quibblings; Bibliography Update; News & Notes; The Impact of Word Processing on Authors and Their Books

**Vol. 5 No. 3 [Mar. 1987], 18 pp.**
Software Review: *PC-WRITE 2.7*; Bibliography Update; Evaluating Student Papers with a Word Processor: A Progress Report

**Vol. 5 No. 2 [Feb. 1987], 16 pp.**
Cyrillic Word Processing; 1987 CCCC Convention Program; Bibliography Update; Shareware Integration

**Vol. 5 No. 1 [Jan. 1987], 18 pp.**
Desktop Publishing and the Writer: An Outline of the Future; Footnote to "The Global Microcomputer"; Bibliography Update; The Productivity Chimera

**Vol. 4 No. 9 [Dec. 1986], 15 pp.**
A Customized *Apple Writer* Startup Program; Bibliography Update; SNOBOL4 Programming as Word Processing; 1986 Software Review Index

**Vol. 4 No. 8 [Nov. 1986], 18 pp.**
Multi-Lingual Word-Processing with the Macintosh; Bibliography Update; Memory Resident Thesaurus Programs

**Vol. 4 No. 7 [Oct. 1986], 18 pp.**
*FinalWord II*: Word Processing for a College Writing Program; Word Processing as a Tool for Revision; Bibliography Update

**Vol. 4 No. 6 [Sep. 1986], 18 pp.**
Desktop Publishing That Anyone Can Do; Software Review: *AppleWriter II*; Computer Projected Thinking in the Classroom; Bibliography Update; Electronic Outlining Comes of Age

**Vol. 4 No. 5 [May 1986], 23 pp.**
Word Processing, Electronic Research, and Desktop Publishing: Annual Cumulative Bibliography

**Vol. 4 No. 4 [Apr. 1986], 31 pp.**
Personal Publishing on Microcomputers; Writing-Instruction Software with *HBJ Writer*; Software Review: *WordPerfect 4.1*; Bibliography Update; Scholar's Software Library: *Rightwriter 2.0*

**Vol. 4 No. 3 [Mar. 1986] 15 pp.**
More on Low-Cost Word Processing; Classroom Computers & Job Seeking Strategies; Bibliography Update; Scholar's Software Library: *Fancy Font 2*

**Vol. 4 No. 2 [Feb. 1986], 21 pp.**
A Typology of Word-Processing Programs; News & Notes; Bibliography Update; Addendum to *Microsoft Word* (Macintosh)

**Vol. 4 No. 1 [Jan. 1986], 17 pp.**
Diagrammatic Writing: "Larger Vision" Software; 1985 Software Review Index; Review: *Nota Bene*; Bibliography Update; *TELECOMMUTER*: Laptop to PC Link

**Vol. 3 No. 9 [Dec. 1985], 21 pp.**
The English Department Microlab: An Endangered Species; Software For Text Analysis & Writing Instruction; Bibliography Update; Software Review: *Microsoft Word* (Macintosh)

**Vol. 3 No. 8 [Nov. 1985], 13 pp.**
Effects of Word Processing on the Correctness of Student Writing; Review: *Quintilian Analysis*; Bibliography Update; Review: *NoteBook II*

**Vol. 3 No. 7 [Oct. 1985], 17 pp.**
Introducing Word Processing to Students; Review: *Readability;* A Cautious View of Computers in Teaching Writing; Bibliography Update; Review: *SAMNA* +

**Vol. 3 No. 6 [Sep. 1985], 15 pp.**
Word Processing on a Budget; Bibliography Update; Scholar's Software Library: *ProofWriter;* Software Review: *MacWrite* 4.5

**Vol. 3 No. 5 [May 1985], 19 pp.**
Word Processing, Writing, Literature, and Linguistics: Annual Cumulative Bibliography

**Vol. 3 No. 4 [Apr. 1985], 15 pp.**
Computers, Word Processing, and the Teaching of Writing; Scholar's Software Library: *Framework;* Word-Processing Errors in Technical Writing; Bibliography Update; *ZYIndex*: State-Of-The-Art Text Management

**Vol 3 No. 3 [Mar. 1985], 11 pp.**
Micros, Minis, and Writing: A Critical Survey; Bibliography Update; Review: *Microsoft Word* (IBM)

**Vol. 3 No. 2 [Feb. 1985], 11 pp.**
Setting Up a Word-Processing Microlab; Bibliography Update; Scholar's Software Library: *ASCII;* Software Review: *WordStar* 3.3

**Vol. 3 No. 1 [Jan. 1985], 11 pp.**
A Scholar's Typology of Database Management Software; Bibliography Update; *Textra* Extra; *WordStar* Tips and Tricks

**Vol. 2 No. 9 [Dec. 1984], 11 pp.**
Teaching News Writing With a Computer; Database Management for Teachers and Researchers III; Bibliography Update; Software Review: *WordMARC*

**Vol. 2 No. 8 [Nov. 1984], 11 pp.**
A Meaning-Based Thesaurus from Old English to the Present; National Science Foundation Research Awards; Database Management for Teachers and Researchers II; Bibliography Update; Software Review: *SuperWriter*

**Vol. 2 No. 7 [Oct. 1984], 11 pp.**
Information Overload and the Library Research Paper; A Computerized *Oxford English Dictionary;* Database Management for Teachers and Researchers I; Bibliography Update; Software Review: *Textplus*

**Vol. 2 No. 6 [Sep. 1984], 11 pp.**
Evaluating Student Papers with a Word Processor; Bibliography Update; Software Review: *Textra*

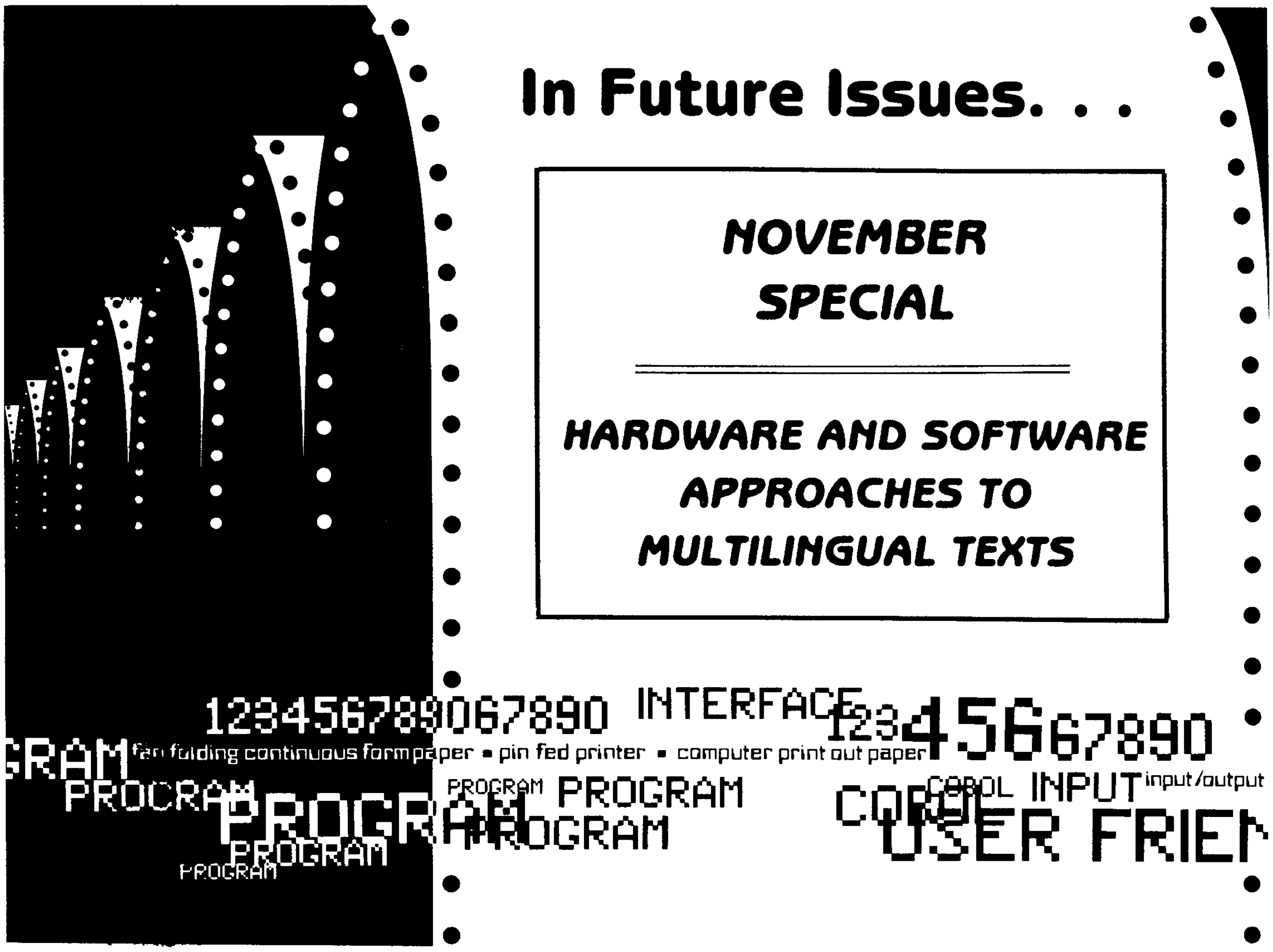
**Vol. 2 No. 4 [Apr. 1984], 9 pp.**
The Future of Word Processing in Academic Writing Programs; Bibliography Update; Software Review: *EasyWriter II*

**BACK ISSUE PRICES:** Volume 5—$4.00 each; Volume 4—$3.00 each; Volumes 3 and 2—$2.00 each

Please specify volume and issue numbers on a separate sheet of paper. Make checks payable to *RWPN* and send to Editors, *Research in Word Processing Newsletter,* South Dakota School of Mines & Technology, 501 E. St. Joseph, Rapid City, SD 57701-3995. Please allow 4 to 6 weeks for order processing and delivery.

## Call for Papers: Scotland

"Technology, Communications, and the Humanities" is the title of a conference to be held August 18-21, 1988, in Edinburgh, Scotland, and sponsored by the Institute for Advanced Studies in the Humanities. The main themes will be technology and decision-making, technology and the acquisition of knowledge, technology and creative design, and technology and daily life. Paper proposals are being accepted. Contact the Director, Institute for Advanced Studies in the Humanities, University of Edinburgh, Hope Park Square, Edinburgh EH8 9NW, Scotland.


12345678890067890 INTERFACE 1234567890
fan folding continuous form paper • pin fed printer • computer print out paper
PROGRAM
COBOL INPUT input/output
USER FRIE

## In Future Issues. . .

**NOVEMBER SPECIAL**

**HARDWARE AND SOFTWARE APPROACHES TO MULTILINGUAL TEXTS**