Paper 91-25

Let's Play a Game: A SAS® Program for Creating a Word Seach Matrix

Robert S. Matthews, University of Alabama at Birmingham, Birmingham, AL

ABSTRACT

This paper describes a process for inserting a list of words into a matrix of random letters. The output produces two tables. The first table is a display of the matrix after the words in the list are inserted, but before random letters are inserted into the remaining cells in the matrix. The second table is the final matrix after empty cells are filled with random letters. The program has two levels of difficulty and highlights a number of techniques for working with two-dimensional arrays. It is applicable to all versions and platforms of The SAS® System.

INTRODUCTION

A word search matrix has a number of potential uses; it can be both educational as well as entertaining. The primary goal in designing a program to produce such a matrix was that it be flexible enough to handle different size matrices, word lists, and skill levels. A flowchart, in Appendix I, describes the sequence of events for implementing the program. The program source code is listed in Appendix II.

DESIGN

The design of this program involves three primary steps.

1. Input a list of words
2. Determine where to place each word in the matrix
3. Print the final matrix to the screen or other output device

The first step is to obtain from the user a list of words to be inserted into the matrix. This can be done in a variety of different ways, such as reading the words from a text file, creating a data entry screen in SAS, or placing the words directly in the program code and reading them via a CARDS statement.

Step two involves placing each word into the output matrix. This is the most complicated part of the entire process. The flowchart, in Appendix I, is extremely helpful in visualizing how a word is inserted into the output matrix. In brief, these are the steps involved.

1. Determine the starting row and column
2. Pick a random direction
3. Determine the ending row and column
4. Check to see if the ending row and column are within the matrix boundaries
5. Check to see if an existing word in the matrix is being overwritten
6. After all words are placed into the matrix, empty cells are filled with random letters

The last step is to print the final matrix to the output device. The user also has the choice of printing an "answer key".

CONCLUSION

Designing a program to produce a word search matrix for my children to use in their schoolwork seemed pretty simple until I starting writing the code to actually implement it. It was an interesting programming challenge and I hope that others can adapt the program for their own uses.

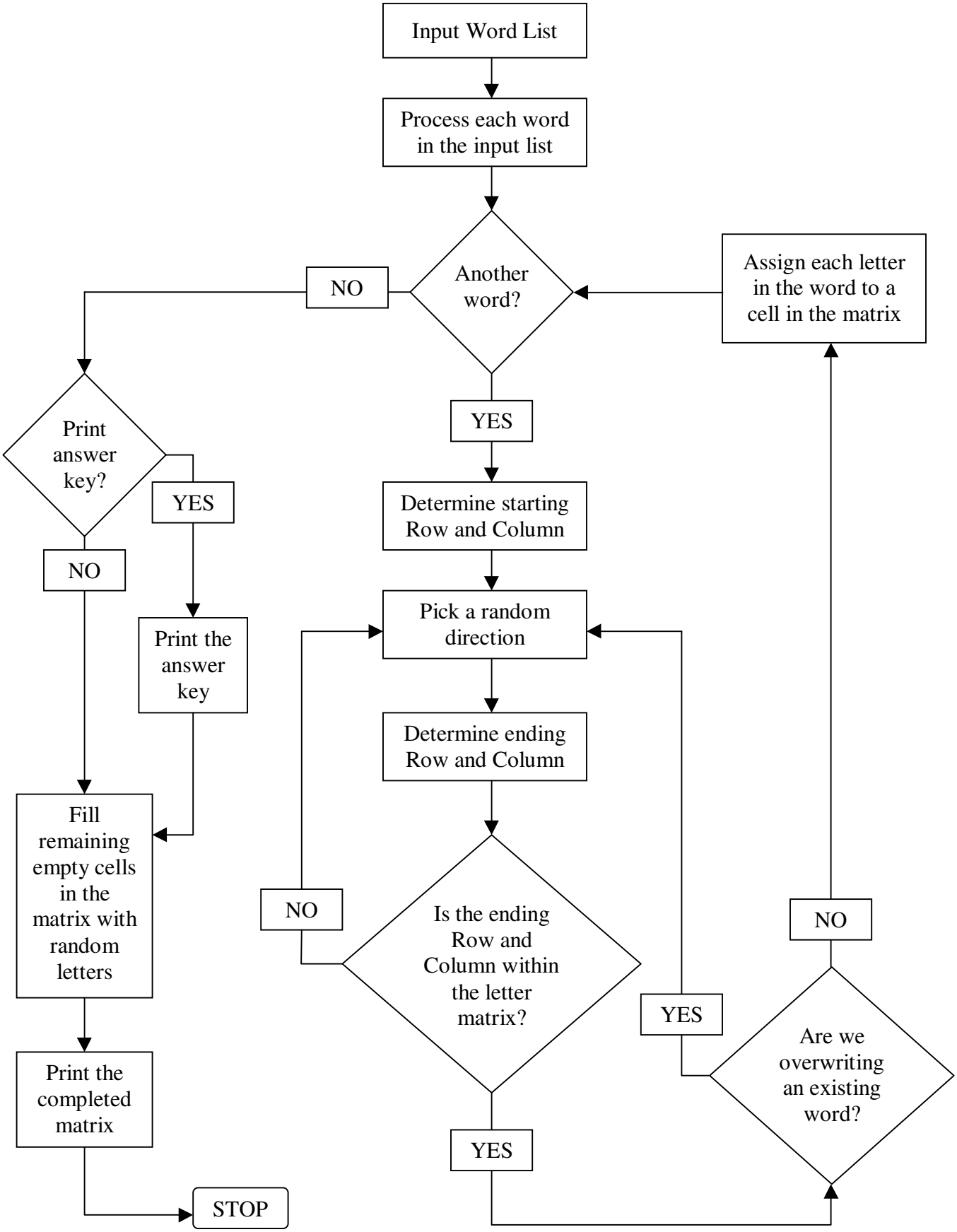
ADDITIONAL INFORMATION

For more documentation about the word search program, contact Robert Matthews at one of the following addresses:

University of Alabama at Birmingham
Department of Epidemiology
1665 University Blvd. Room 445
Birmingham, AL 35294-0022
E-mail: rsm@uab.edu

APPENDIX I

Word Search flowchart



APPENDIX II

Program source code

```
* Word Search   v.1.5  Robert Matthews 5/99 ;

* If you make improvements on this code I would
  appreciate a copy sent to rsm@uab.edu;

title '* * *   WORD SEARCH   * * * ';
title3 ' ';
options nodate nonumber ps=42 ls=80 pageno=1;

data wordlst ;
  length wrd $20;
  array w{*} $20 word1-word30;
  retain count 1 word1-word30;
  input wrd @@;
  w{count} = upcase(wrd);
  count+1;
  drop wrd;
cards;
flat heaven fridge lift torch christian
robert cinema bobby icelolly crisps
;
run;

data wordlist;
  set wordlst nobs=n;
  count=count-1;
  if _n_ = n ;
run;

/* Alternate method for entering words;
data getwords;
  length word $20;

  window words
    #5 @5 'Please enter the spelling words'
    #7 @10 'Word: ' word;

  display words;
  if word='' then
    stop;
  else
    word = upcase(word);
run;

proc transpose data=getwords out=newlist(drop=_name_) prefix=word;
var word; run;

data wordlist;
  set newlist end=last;
  retain count 0;
  array w word1-word30;
  do over w;
    w=upcase(w);
    if last and w ne '' then count+1;
  end;

  if last;
run;        */

%let row=12; %let col=20;
%let totlcell = %eval(&row * &col);

data matrix;
  set wordlist;

  skill = 2;      * 1 - Easy   2 - Hard;

  array m{&row,&col} $1 m1-m&totlcell;

  array w{*} word1-word30;
  array d{*} d1-d8;

  alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

  do i = 1 to count;

    wlength = length(w{i});
    r_rnd = round(ranuni(0) * &row,1);
    if r_rnd = 0 then r_rnd = 1;

    c_rnd = round(ranuni(0) * &col,1);
    if c_rnd = 0 then c_rnd = 1;

    valid = 0; lcount=0;

    do until (valid);

      * Possible directions for a word to go;
      * 1-N  2-NE  3-E  4-SE  5-S  6-SW  7-W  8-NW;

here:   d_rnd = round(ranuni(0) * 8 + 1,1);
      if skill=1 and d_rnd not in (3,5) then goto here;

      if d_rnd = 0 then d_rnd = 1;
      select (d_rnd);
        when (1) do; er = r_rnd-wlength+1;
                     ec = c_rnd;
                     r_offset = -1;
                     c_offset = 0 ;
                   end;
        when (2) do; er = r_rnd-wlength+1;
                     ec = c_rnd+wlength-1;
                     r_offset = -1;
                     c_offset = 1 ;
                   end;
        when (3) do;
                     er = r_rnd;
                     ec = c_rnd+wlength-1;
                     r_offset = 0;
                     c_offset = 1;
                   end;
        when (4) do;
                     er = r_rnd+wlength-1;
                     ec = c_rnd+wlength-1;
                     r_offset = 1;
                     c_offset = 1;
                   end;
        when (5) do;
                     er = r_rnd+wlength-1;
                     ec = c_rnd;
                     r_offset = 1;
                     c_offset = 0;
                   end;
        when (6) do;
                     er = r_rnd+wlength-1;
                     ec = c_rnd-wlength+1;
                     r_offset = 1;
                     c_offset = -1;
                   end;
        when (7) do;
                     er = r_rnd;
                     ec = c_rnd-wlength+1;
                     r_offset = 0;
                     c_offset = -1;
                   end;
        when (8) do;
                     er = r_rnd-wlength+1;
                     ec = c_rnd-wlength+1;
                     r_offset = -1;
                     c_offset = -1;
                   end;
        otherwise;
      end;
```

```
    if er > 0 and ec > 0 and
      er <= &row and ec <= &col then valid = 1;

    if valid then
      do;
        * Check to make sure we aren't
          overwriting an existing letter;
        cr = r_rnd; cc = c_rnd;
        do j = 1 to wlength;
          if m{cr,cc} ne '' and
            m{cr,cc} ne substr(w{i},j,1) then
              valid=0;
          cr = cr+r_offset;
          cc = cc+c_offset;
        end;
      end;

    lcount+1;
    if lcount > 20 then
      do;
        r_rnd = round(ranuni(0) * &row,1);
        if r_rnd = 0 then r_rnd = 1;

        c_rnd = round(ranuni(0) * &col,1);
        if c_rnd = 0 then c_rnd = 1;
        lcount=1;
      end;
  end; * DO UNTIL;

  * Insert each letter into the
    appropriate cell in the matrix;

  cr = r_rnd; cc = c_rnd;
  do j = 1 to wlength;
    m{cr,cc} = substr(w{i},j,1);
    cr = cr+r_offset;
    cc = cc+c_offset;
  end;
end;

link print; * print "BEFORE" matrix (answer key);

* fill in empty cells with random letters;
  do r=1 to &row;
    do c=1 to &col;
      if m{r,c} = '' then
        do;
          m_rnd = int(ranuni(0) * 26 + 1);
          m{r,c} = substr(alphabet, m_rnd, 1);
        end;
    end;
  end;

  put _page_ @;
  link print ;  * print "AFTER" matrix;
return;

print:
  * print matrix;
  file print;
  wcount=1;

  put '- Word List ' 28*'-'
      '  Word Search Matrix  ' 17*'-' /;

  do r=1 to &row;
    do c=1 to &col;
      if c = 1 then
        do;
          if wcount <= count then
            do;
              put @3 w{wcount} @;
              wcount+1;
            end;

          put  @20 '| ' @;
        end;

      put m{r,c} ' ' @;
    end;
    put @20 '|';
  end;
  put 79*'-';

 * put // 'Additional text can be placed on these lines. ' /
        'It will be printed below each table.          ';
run;
```