```
PROGRAM TPLOT_SIZE_EVALUATION;
{This program takes data from vegetative assessments of the LTSP IRS
 preliminary site survey and uses adjacent veg assessment squares to
 build conglomerate squares of various sizes. Data from the smaller
 squares is used to determine the results obtained with varying square
 size. These results are output to a data file for later analysis with
 statistical computer packages. The variables output are square size,
 vegetative cover, and number of species.}

CONST
        input_file = 'c:\pfsl\psurvey\alldata\cleanc3.dat';
        location = 'W';
        dim_file = 'c:\pfsl\psurvey\alldata\dims.dat';
        coutput_file = 'c:\pfsl\psurvey\alldata\psiz3fwt.dat';
        outsim_file = 'c:\pfsl\psurvey\alldata\sims3fwt.dat';
        site = 'F';         {Code for the site whose data will be used.}
        soutput_file = 'c:\pfsl\psurvey\alldata\nspc3fwt.dat';
VAR
        blank,
        location_in,
        site_in                 : CHAR;
        g,
        nmbr_of_sims,
        plot_in,
        plot_max,
        plot_min,
        plot_offset,
        rep,
        total_squares,
        total_nspecies,
        total_nrows             : INTEGER;
        row_width,
        sim_size                : REAL;
        species                 : ARRAY[1..90] OF STRING[5];
        size,
        max_x,
        max_y,
        min_x,
        min_y                   : ARRAY[1..330] OF REAL;
        cover                   : ARRAY[1..90] OF REAL;
        plot                    : ARRAY[1..120] OF INTEGER;
        square                  : ARRAY[1..120] OF STRING[2];
        x,
        y                               : ARRAY[1..120] OF REAL;
        infile,
        coutfile,
        dimfile,
        outsim,
        soutfile                : TEXT;

PROCEDURE PLOT_NUMBERS;
{This procedure determines which plot numbers in the input data
 correspond to 9x9 transect plots for the site and location
 indicated by "site" and "location".}

VAR
        irs_site,
        location_type : CHAR;

BEGIN {PLOT_NUMBERS}
```

```
        irs_site := site;

        plot_offset := 0;
        plot_min :=  1;
        plot_max := 12;

        CASE irs_site OF
        'N': plot_offset := 500;
        'S': plot_offset := 400;
        'Y': plot_offset := 300;
        'F': plot_offset := 200;
        'U': plot_offset := 100;
        'W': plot_offset :=   0;
        END;

        location_type := location;

        CASE location_type OF
        'W': plot_offset := plot_offset + 0;
        'C': plot_offset := plot_offset + 50;
        END;

        plot_max := plot_max + plot_offset;
        plot_min := plot_min + plot_offset;

        WRITELN('max: ',plot_max,'  min: ',plot_min);

END;{PLOT_NUMBERS}


PROCEDURE PLOT_LOCATIONS;
{This procedure assigns the (x,y) coordinates of the center point of
 each veg assessment square used in the 4 central 9x9m plots on the
 site being examined. These coordinates are later used to determine
 which squares must be combined to yield conglomerate squares of the
 proper size in procedure PLOT_SIMULATION.}

VAR
        az,
        i,
        j,
        cycle_of_3,
        last_cycle_value,
        n_of_plots,
        n_plots_read,
        n_of_squares            : INTEGER;
        nmbr                            : ARRAY [1..25] OF INTEGER;
        transect_loc            : REAL;
        dis                                     : ARRAY [1..25] OF REAL;
        digit1,
        digit2                          : CHAR;
        pt_type                         : STRING[2];
        location_found          : BOOLEAN;

BEGIN {PLOT_LOCATIONS}

        FOR i := 1 TO 120 DO
                BEGIN
                        plot[i] := 0;
                        square[i] := ' ';
```

```
                x[i] := 0;
                y[i] := 0;
        END;

{This next section reads the distances recorded in the "dims.dat" file
 for location of plots on transects. The plot number is stored in array
 "nmbr" and the distance is stored in the corresponding location in
 array "dis".}

FOR j := 1 TO 25 DO
        BEGIN
                nmbr[j] := 0;
                dis[j] := 0;
        END;

ASSIGN(dimfile,dim_file);
RESET(dimfile);

j := 1;
WHILE NOT EOF(dimfile) DO
        BEGIN
                READLN(dimfile,nmbr[j],blank,pt_type,az,dis[j]);
                IF (nmbr[j]>=plot_min) AND
                   (nmbr[j]<=plot_max) THEN j := j + 1
                ELSE
                        BEGIN
                                nmbr[j] := 0;
                                dis[j] := 0.0;
                        END
        END;

n_plots_read := j-1;

CLOSE(dimfile);

{End of plot location reading. Begin calculating coordinates for squares}

digit1 := '1';
digit2 := 'C';
last_cycle_value := 1;
n_of_plots := 12;  {Number of potential transect plots at one location}
n_of_squares := 9 * n_of_plots; {No of potential squares at one location}

FOR i := 1 TO n_of_squares DO
        BEGIN

                {The next statement assigns a plot number for each potential
                 square.}

                plot[i] := plot_offset + ((i+8) div 9);

                {This next short section determines square ID codes for all
                 squares which can potentially be found at the location
being
                 used. The first letter of the code is "digit1", and the
second
                 is "digit2", these are combined and stored as a complete
                 square ID in array "square" towards the end of this
subprogram.}
```

```
                        cycle_of_3 := (i+2) div 3;
                        IF (cycle_of_3 > last_cycle_value) THEN
                                BEGIN
                                        CASE digit1 OF
                                        '1': digit1 := '2';
                                        '2': digit1 := '3';
                                        '3': digit1 := '1';
                                        END;
                                        last_cycle_value := cycle_of_3;
                                END;

                        CASE digit2 OF
                                'A': digit2 := 'B';
                                'B': digit2 := 'C';
                                'C': digit2 := 'A';
                        END;

                        {This section assigns (x,y) coordinates in meters. Plot
center
                         is used as the origin, and the transect with plots 1-4 is
                         used as the negative x axis.}

                        j := 0;
                        location_found := FALSE;
                        WHILE (NOT location_found) AND (j<=n_plots_read) DO
                                BEGIN
                                        j := j + 1;
                                        IF nmbr[j]=plot[i] THEN
                                                BEGIN
                                                        location_found := TRUE;
                                                        transect_loc := dis[j];
                                                END;
                                END;

            CASE plot[i]-plot_offset OF
                        1..4:BEGIN     {plots 1-4 on horiz left transect}

                                        CASE digit1 OF
                                                '1':x[i] := -2.5 * row_width -
transect_loc;
                                                '2':x[i] := -1.5 * row_width -
transect_loc;
                                                '3':x[i] := -0.5 * row_width -
transect_loc;
                                        END;

                                        CASE digit2 OF
                                                'A': y[i] := -1.0 * row_width;
                                                'B': y[i] :=  0.0 * row_width;
                                                'C': y[i] :=  1.0 * row_width;
                                        END;

                                END;

                        5..6:BEGIN

                                        CASE digit1 OF
                                                '1': y[i] :=  2.5 * row_width +
transect_loc;
                                                '2': y[i] :=  1.5 * row_width +
```

```
transect_loc;
                                                '3': y[i] :=  0.5 * row_width +
transect_loc;
                                        END;

                                        CASE digit2 OF
                                                'A': x[i] := -1.0 * row_width;
                                                'B': x[i] :=  0.0 * row_width;
                                                'C': x[i] :=  1.0 * row_width;
                                        END;

                                END;

                        7..8:BEGIN

                                        CASE digit1 OF
                                                '1': y[i] := -2.5 * row_width  -
transect_loc;
                                                '2': y[i] := -1.5 * row_width  -
transect_loc;
                                                '3': y[i] := -0.5 * row_width  -
transect_loc;
                                        END;

                                        CASE digit2 OF
                                                'A': x[i] :=  1.0 * row_width;
                                                'B': x[i] :=  0.0 * row_width;
                                                'C': x[i] := -1.0 * row_width;
                                        END;


                                END;

                        9..12:BEGIN

                                        CASE digit1 OF
                                                '1':x[i] :=  2.5 * row_width +
transect_loc;
                                                '2':x[i] :=  1.5 * row_width +
transect_loc;
                                                '3':x[i] :=  0.5 * row_width +
transect_loc;
                                        END;

                                        CASE digit2 OF
                                                'A': y[i] :=  1.0 * row_width;
                                                'B': y[i] :=  0.0 * row_width;
                                                'C': y[i] := -1.0 * row_width;
                                        END;


                                END;
                        END;

                        square[i] := digit1 + digit2;
                        WRITELN(digit1,' ',digit2,' ',square[i]);

                END;
```

```
        WRITELN('Square coordinates');
        FOR i := 1 TO 108 DO
                WRITELN(plot[i]:3,square[i]:3,x[i]:5:1,y[i]:5:1);

END; {PLOT_LOCATIONS}

PROCEDURE PLOT_SIMULATION(ssize:REAL;VAR nsims:INTEGER);
{This procedure determines, for a given previously chosen conglomerate
 square size, which (x,y) coordinates define the lower left and upper
 right limits of squares of that size. All possible squares of the
 chosen size are generated using the data from the central group of 4
 9x9m plots used on a prospective IRS site.}

VAR
        i,
        j,
        x_positions,
        y_positions,
        total_positions : INTEGER;
        outsim                  : TEXT;

BEGIN {PLOT_SIMULATION}


        x_positions := 1 + total_nrows -g; {Number of positions for plot

simulations of size g*row_width
                                                                                in a
9x9m plot}

        y_positions := x_positions;          {For a 9x9 rectangle plot}

        {Total number of potential positions for plot simulations.}

        total_positions := 9 * 12;  {Try all possible locations. Let later
                                                                parts of the
program weed out those sims
                                                                which don't fall
entirely within 9x9s}

        {For a plot simulation of size 'ssize', this next loop determines
         minimum (x,y) and maximum (x,y) coordinates of plot simulation squares.
         Simulation number is
         determined sequentially for all plot simulation sizes. The calculations
         are stored in arrays sim_nmbr,size,min_x,min_y,max_x, max_y, and are
         writen to file 'outsim_file'.}

        WRITELN('Plot simulation begun');
        FOR i := 1 TO total_positions DO
                BEGIN
                        j := nsims + i;
                        min_x[j] := x[i] - 1.5;
                        min_y[j] := y[i] - 1.5;
                        max_x[j] := x[i] - 1.5 + ssize;
                        max_y[j] := y[i] - 1.5 + ssize;
                        size[j] := ssize;

                        ASSIGN(outsim,outsim_file);
                        APPEND(outsim);
                        WRITELN(outsim,j:3,min_x[j]:6:1,min_y[j]:6:1,
```

```
                                        max_x[j]:6:1,max_y[j]:6:1);
                        CLOSE(outsim);

                END;

        nsims := nsims + total_positions;
        WRITELN('Total no. of simulations = ',nsims);

END;{PLOT_SIMULATION}

PROCEDURE DATA_EVALUATION(nsims:INTEGER);
{This procedure combines data from the component squares determined to
 be appropriate using the results of the PLOT_SIMULATION procedure and
 outputs the resulting data to a data file. The name of the output data
 file is indicated by the value of the pascal constant "output_file",
 which is defined in the CONST block of the main program
 (PLOT_SIZE_EVALUATION).}

VAR
        h,
        i,
        n_potential_squares,
        n_squares_in            : INTEGER;
        weight  : REAL;
        include,
        data_found,
        simulation_complete     : BOOLEAN;

PROCEDURE SPECIES_INVENTORY(plot_nmbr:INTEGER;sqid:STRING;VAR sq_used:BOOLEAN);
{This procedure searches the data file indicated by the value of the
 pascal constant "input_file" for all species occuring on veg squares
 indicated by procedure DATA_EVALUATION as containing data contributing
 to the present plot simulation in DATA_EVALUATION. Cover data from an
 appropriate square is weighted by the proportion of the area the square
 occupies in the simulated plot. The weighted sum of a species' cover
 values from all included squares is used as the cover value of the
 simulated plot. The subprogram returns sq_used value TRUE if data is
 found for the veg square being tallied. }

VAR
        j,
        k,
        rep_in           : INTEGER;
        cover_in,
        wt                       : REAL;
        square_in        : STRING[2];
        species_in   : STRING[5];
        done,
        match,
        sp_already_used  : BOOLEAN;

BEGIN {SPECIES_INVENTORY}

        ASSIGN(infile,input_file);
        RESET(infile);

        wt := 1 / ((size[g]/row_width)*(size[g]/row_width));

        sp_already_used := FALSE;
        k := 0;
```

```
        WHILE (NOT EOF(infile)) DO
                BEGIN
                  done := FALSE;
                  READLN(infile,plot_in,blank,square_in,rep_in,blank,
                                 species_in,cover_in);
                  match := (plot_nmbr = plot_in) AND
                                   (sqid = square_in) AND
                                   (rep_in = 1);
                  k := k + 1;
                  IF (species_in[5] IN ['X','A','B','1','2'])
                         THEN species_in[5] := ' ';

                  IF match THEN sq_used := TRUE;

                  IF match AND (total_nspecies >= 1) THEN
                         BEGIN
                                 j := 0;
                                 REPEAT
                                         j := j + 1;
                                         IF (species_in = species[j]) THEN
                                                BEGIN
                                                   cover[j] := cover[j] + cover_in *
wt;
                                                   sp_already_used := TRUE;
                                                END;
                                         done := sp_already_used OR
                                                        (j = total_nspecies);
                                 UNTIL done;
                         END;
                  IF done AND (NOT sp_already_used) THEN
                         BEGIN
                                  total_nspecies := total_nspecies + 1;
                                  species[total_nspecies] := species_in;
                                  cover[total_nspecies] := cover[total_nspecies] +

cover_in * wt;
                                  WRITELN(species_in);
                         END;
                  IF match AND (total_nspecies = 0) THEN
                         BEGIN
                                  total_nspecies := 1;
                                  species[1] := species_in;
                                  cover[1] := cover[1] + cover_in * wt;
                                  WRITELN(species_in);
                         END;
                END;

   CLOSE(infile);

   WRITELN('Nspecies after assessment of plot ',plot_nmbr:3,' square ',
                   sqid,':', total_nspecies);

END;{SPECIES_INVENTORY}

BEGIN {DATA_EVALUATION}

        WRITELN('Simulation ',g:4,', (',size[g]:3:0,'m square)');

        n_potential_squares := round((size[g]/row_width)*(size[g]/row_width));
        n_squares_in := 0;
```

```
        data_found := FALSE;
        simulation_complete := FALSE;

    total_nspecies := 0;
        FOR h := 1 TO 90 DO
                BEGIN
                        species[h] := '';
                        cover[h] := 0;
                END;

        h := 0;
        WHILE (h<108) AND (NOT simulation_complete) DO
                BEGIN
                        h := h + 1;
                        include := (min_x[g] < x[h]) AND
                                           (min_y[g] < y[h]) AND
                                           (max_x[g] > x[h]) AND
                                           (max_y[g] > y[h]);
                        IF include THEN
                        BEGIN

                                SPECIES_INVENTORY(plot[h],square[h],data_found);

                                IF data_found THEN n_squares_in := n_squares_in + 1;

                                {This boolean tests whether all squares that SHOULD
be in
                                 a plot simulation have contributed.}

                                simulation_complete := (n_squares_in =
n_potential_squares);
                        END;
        END;

        {If simulation_complete is still false, this means that data isn't
         available for all squares that SHOULD have contributed. Estimates
         are written to output files only if all the squares are available.}

        IF simulation_complete THEN
        BEGIN

                ASSIGN(coutfile,coutput_file);
                APPEND(coutfile);

        FOR h := 1 TO total_nspecies DO
                        WRITELN(coutfile,site,' ',location,' trnsct',g:4,
                                        size[g]:4:0,species[h]:6,cover[h]:7:3);

                CLOSE(coutfile);

                ASSIGN(soutfile,soutput_file);
                APPEND(soutfile);
                WRITELN(soutfile,site,' ',location,' trnsct',g:4,size[g]:4:0,
                                total_nspecies:5);
                CLOSE(soutfile);

        END;

    WRITELN('Nspecies for simulation ',g:3,' = ',total_nspecies);
```

```
END; {DATA_EVALUATION}

BEGIN {Main body of program}

        ASSIGN(coutfile,coutput_file);
        REWRITE(coutfile);
        CLOSE(coutfile);

        ASSIGN(soutfile,soutput_file);
        REWRITE(soutfile);
        CLOSE(soutfile);

        ASSIGN(outsim,outsim_file);
        REWRITE(outsim);
        CLOSE(outsim);

        nmbr_of_sims := 0;
        total_nrows := 3;
        row_width := 3.0;

        PLOT_NUMBERS;
        PLOT_LOCATIONS;

        FOR g := 1 TO total_nrows DO
                BEGIN
                        sim_size := g * row_width;
                        PLOT_SIMULATION(sim_size,nmbr_of_sims);
                        WRITELN('Nmbr_of_sims = ',nmbr_of_sims);
                END;

        FOR g := 1 TO nmbr_of_sims DO
                DATA_EVALUATION(nmbr_of_sims);

END. {Main program}
```