

```

PROGRAM PLOT_ENTRY;
{Used to prompt, check, and write to files data obtained during the IRS
 initial survey.}
USES
    Crt,
    Dos;
CONST
    debrisfile = 'debris.dat';
    dimfile = 'dims.dat';
    plotfile = 'plots.dat';
    soilfile = 'soil.dat';
VAR
    debris_table,
    dimension_table,
    plot_table,
    soil_table,
    tempscreen : TEXT;

    az,
    id ,
    id_resp,
    matches,
    newid          : INTEGER;

    dist          : REAL;

    blank,
    focus,
    focus_resp,
    plotlocation,
    plotlocation_resp,
    pt_type,
    site,
    site_resp,
    vegunit       : CHAR;

    datetime     : STRING[19];

    change,
    dataentry,
    end_entry,
    end_program,
    new_plot,
    old_plot,
    some_match,
    valid_ans     : BOOLEAN;

PROCEDURE GETYN(temp:STRING;VAR bvar:BOOLEAN;value:CHAR);
VAR
    tempt          : STRING[2];
    valid2        : BOOLEAN;
BEGIN
    REPEAT
        WINDOW(1,24,60,25);
        GOTOXY(1,2);CLREOL;
        tempt := '';
        WRITE(temp,' (Y,N):');
        READLN(tempt);
        IF (LENGTH(tempt) = 0) THEN tempt := 'Y'
    Page 1

```

```

ELSE
    BEGIN
        tempt := UPCASE(tempt[1]);
        valid2 := (tempt[1] = 'Y') OR
                  (tempt[1] = 'N');
    END;
IF (NOT valid2) THEN
    BEGIN
        SOUND(900);
        DELAY(100);
        NOSOUND;
        WRITE('Invalid answer. ');
    END;
UNTIL valid2;
bvar := (tempt[1] = value);
END; {GETYN}

PROCEDURE GETINTVALUE(text_prompt:STRING;VAR value:INTEGER;min,max:INTEGER);
{This procedure prompts user for value of 'variable' (as defined in
ENTERVEGDATA) and requires entry of a value between min and max.}
VAR
    code : INTEGER;
    temp : STRING;
BEGIN
    WINDOW(1,24,60,25);
    GOTOXY(1,2);CLREOL;
    REPEAT
        valid_ans := FALSE;
        WRITE(text_prompt);
        READLN(temp);
        VAL(temp,value,code);
        IF (code = 0) THEN valid_ans := (value >= min) AND
                                         (value <= max);
        IF (NOT valid_ans) THEN
            BEGIN
                SOUND(900);
                DELAY(100);
                NOSOUND;
                WRITE('Invalid value. ');
            END;
    UNTIL valid_ans;
END; {GETINTVALUE}

```

```

PROCEDURE GETREALVALUE(text_prompt:STRING;VAR value:REAL;min,max:REAL);
{This procedure prompts user for value of 'variable' (as defined in
ENTERVEGDATA) and requires entry of a value between min and max.}
VAR
    code : INTEGER;
    temp : STRING;
BEGIN
    WINDOW(1,24,60,25);
    GOTOXY(1,2);CLREOL;
    REPEAT
        valid_ans := FALSE;
        WRITE(text_prompt);
        READLN(temp);
        VAL(temp,value,code);
        IF (code = 0) THEN valid_ans := (value >= min) AND
                                         (value < max);
        IF (NOT valid_ans) THEN

```

```

                                BEGIN
                                SOUND(900);
                                DELAY(100);
                                NOSOUND;
                                WRITE('Invalid value. ');
                                END;
UNTIL valid_ans;
END; {GETREALVALUE}

PROCEDURE GETDATETIME;
{Obtains system date and time, combines these into one string called
'datetime'.}
VAR
    year,
    month,
    day,
    wkday,
    hour,
    min,
    sec,
    sec100 : WORD;
    tyear : STRING[4];
    tmonth,
    tday,
    thour,
    tmin,
    tsec : STRING[2];
    tempdate : STRING[19];

BEGIN;                                {Main body of GETDATETIME}
    GETDATE(year, month, day, wkday);
    GETTIME(hour, min, sec, sec100);
    STR(year:4, tyear);
    STR(month, tmonth);
    STR(day, tday);
    STR(hour, thour);
    STR(min, tmin);
    STR(sec, tsec);
    datetime := tmonth + '/' + tday + '/' + tyear + ' ' +
                thour + ':' + tmin + ':' + tsec;
END; {GETDATETIME}

PROCEDURE GETSITE;
{This procedure prompts user for site, checks for valid response.
The procedure requires the user to provide a valid response before
passing control.}
VAR
    valid : BOOLEAN;

BEGIN
    site_resp := 'Y';    {Siskiyou site}
    {REPEAT
        WRITE('Willamette, Umatilla, or Forks site? (W,U,F): ');
        READLN(site_resp);
        site_resp := UPCASE(site_resp);
        valid := (site_resp = 'W') OR
                 (site_resp = 'U') OR
                 (site_resp = 'F');
        IF (NOT valid) THEN
            BEGIN
                SOUND(900);

```

```

                                DELAY(100);
                                NOSOUND;
                                WRITE(' Invalid site value.');
```

END;

```

    UNTIL valid;}
END; {GETSITE}

PROCEDURE GETFOCUS;
{This procedure prompts user for plot type, and checks for a valid
 responses. Valid responses are required before control is passed.}

VAR
    valid : BOOLEAN;
BEGIN
    plotlocation_resp := ' ';
    REPEAT
        WRITE('Debris or soil transect? (D,S):');
        READLN(focus_resp);
        focus_resp := UPCASE(focus_resp);
        valid := (focus_resp = 'D') OR
                (focus_resp = 'S');
        IF (NOT valid) THEN
            BEGIN
                SOUND(900);
                DELAY(100);
                NOSOUND;
                WRITE('Invalid response. ');
            END;
        UNTIL valid;
    END; {GETFOCUS}

PROCEDURE SHOWPLOTS;
{Read ID data already entered for measurement plots, seek matches with
 the user-specified site and plot type. Any matches found will be
 listed. The program will also determine the next plot ID number
 to be useable by a new measurement plot. }

VAR
    az2,
    i,
    id2,
    maxid,
    x,
    y                : INTEGER;
    dist2            : REAL;
    pt_type2        : CHAR;
    first_match,
    match            : BOOLEAN;
BEGIN
    ASSIGN(plot_table, plotfile);
    RESET(plot_table);

    i := 0;
    maxid := 0;
    some_match := FALSE;
    WHILE NOT EOF(plot_table) DO
        BEGIN
            READLN(plot_table, id, blank, site, blank, focus, blank,
plotlocation);
```

```

match := (site = site_resp) AND
         (focus = focus_resp) AND
         (plotlocation = plotlocation_resp);
first_match := (NOT some_match) AND
              (match);
IF first_match THEN
BEGIN
    WINDOW(1,16,60,17);CLRSCR;
    CASE focus_resp OF
    'D':BEGIN
        WRITELN('Debris transects on site
',site_resp,':');
        WRITE(' ID azimuth length ',
              ' ID azimuth length');
        END;
    'S':BEGIN
        WRITELN('Soil transects on site
',site_resp,':');
        WRITE(' ID azimuth length ',
              ' ID azimuth length');
        END;
    END;
    WINDOW(1,18,60,23);CLRSCR;
END;
IF match THEN
BEGIN
    ASSIGN(dimension_table,dimfile);
    RESET(dimension_table);
    WHILE (NOT EOF(dimension_table)) DO
        BEGIN
READLN(dimension_table,id2,blank,pt_type2,az2,dist2);
            IF (id2 = id) THEN
                BEGIN
                    pt_type := pt_type2;
                    az := az2;
                    dist := dist2;
                END;
            END;
            i := i + 1;
            x := 1 + 25 * TRUNC((i-1)/25);
            y := i - 6 * TRUNC((i-1)/6);
            GOTOXY(x,y);
            WRITELN(id:3,az:8,dist:8:1);
        END;
    IF (id > maxid) THEN maxid := id;
    some_match := (first_match) OR
                 (some_match);
END;
newid := maxid + 1;

CLOSE(plot_table);

END; {SHOWPLOTS}

PROCEDURE GETACTION;
{This procedure prompts the user to indicate whether the data to be entered
is from an old plot or a new plot. Sets values of change, new_plot,
old_plot, and end_program. }

```

```

BEGIN

    REPEAT

        change := FALSE;
        new_plot := FALSE;
        old_plot := FALSE;
        end_program := FALSE;

        SHOWPLOTS;

        {If other plots of this type were used on this site, the user is
         asked whether one of these old plots is to be used.}
        WINDOW(1,24,60,25);CLRSCR;
        IF some_match THEN
            BEGIN
                GETYN('Use one of these plots? ',old_plot,'Y');
                IF (old_plot) THEN
                    GETYN('Leave plot location info unchanged?
',change,'N')
                ELSE
                    BEGIN
                        GETYN('Initiate a new plot? ',new_plot,'Y');
                        IF (NOT new_plot) THEN
                            GETYN('Terminate program?
',end_program,'Y');
                    END;
                END;
            END

        {If no plots of the type specified have been entered, the program
         will ask if the user wishes to begin a new plot or reenter
         plot information.}
        ELSE
            BEGIN
                WINDOW(1,24,60,25);
                GOTOXY(1,2);CLREOL;
                WRITE('No plots of this type found. ');
                GETYN('Initiate a new plot? ', new_plot,'Y');
                IF (NOT new_plot) THEN
                    GETYN('Terminate program? ', end_program, 'Y')
                END;
            END

        UNTIL old_plot OR new_plot OR end_program;

    END; {GETACTION}

PROCEDURE GETOLDPLOTID;
{This procedure prompts the user for the ID of the old plot to
be used for further data entry, and checks to see that an old
plot matching the user's specifications exists. The procedure
requires a valid ID before releasing control.}

VAR
    tempint,
    code      : INTEGER;
    temptext  : STRING[4];
    validtext,
    valid     : BOOLEAN;
BEGIN

```

```

ASSIGN(plot_table, plotfile);

    valid := FALSE;
WINDOW(1,24,60,25);
GOTOXY(1,2);CLREOL;
REPEAT
    REPEAT
        WRITE('Enter the plot ID: ');
            READLN(temptext);
        VAL(temptext,id_resp,code);
        validtext := (code = 0);
            IF (NOT validtext) THEN
                BEGIN
                    SOUND(900);
                    DELAY(100);
                    NOSOUND;
                    WRITE(' Invalid value.');
```

```

VAR
    plot_matches : INTEGER;
    azimuth      : ARRAY[ptvalues] OF INTEGER;
    distance     : ARRAY[ptvalues] OF REAL;

BEGIN
    {Main body of SHOWLOCATIONS}

    ASSIGN(dimension_table, dimfile);
    RESET(dimension_table);

    azimuth[E] := 0;
    distance[E] := 0;

    plot_matches := 0;
    WHILE (NOT EOF(dimension_table)) DO
        BEGIN
            READLN(dimension_table, id, blank, pt_type, az, dist);
            IF (id = id_resp) THEN
                BEGIN
                    plot_matches := plot_matches + 1;
                    azimuth[E] := az;
                    distance[E] := dist;
                END;
            END;
        END;

    CLOSE(dimension_table);

    WINDOW(1,16,60,16); CLRSCR;

    WRITE('Transect azimuth length(m)');
    WINDOW(1,17,60,23); CLRSCR;
    WRITELN(azimuth[E]:10, distance[E]:17:1);

END; {SHOWLOCATIONS}

PROCEDURE WRITEPLOTINFO;
{Subordinate to ENTERPLOTINFO.}
{This procedure writes appropriate plot information to the plot_table
file.}

BEGIN

    ASSIGN(plot_table, plotfile);
    APPEND(plot_table);

    GETDATETIME;
    WRITELN(plot_table, id_resp:3, site_resp:2, focus_resp:2,
            plotlocation_resp:2, datetime:19);

    CLOSE(plot_table);

END; {WRITEPLOTINFO}

PROCEDURE ENTERDIMINFO(point:CHAR);
{Subordinate to ENTERPLOTINFO.}
{This procedure writes appropriate plot information to the plot_table
file.}

BEGIN

```

```

        GETINTVALUE('Az-C1 to end pt(0..359):',az_resp,0,359);
        GETREALVALUE('Transect length(m):',dist_resp,0,400);

ASSIGN(dimension_table, dimfile);
APPEND(dimension_table);

        GETDATETIME;
        WRITELN(dimension_table,id_resp:3,point:2,az_resp:4,dist_resp:7:2,
                datetime:19);

CLOSE(dimension_table);

END;{ENTERDIMINFO}

BEGIN                                     {Main body of ENTERPLOTINFO}
        first_pass := TRUE;
        REPEAT
                IF change THEN                                {Reentry of corrected location}
                        BEGIN
                                IF first_pass THEN SHOWLOCATIONS;

                                pt_type := 'E';                {Transect}

                                ENTERDIMINFO(pt_type);
                                END;
                                first_pass := FALSE;
                IF new_plot THEN                                {Entry of dimensions for new
plot}
                        BEGIN
                                id_resp := newid;
                                WINDOW(1,24,60,25);
                                GOTOXY(1,2);CLREOL;
                                WRITELN('Plot number ',id_resp,' assigned.');
```

```

        ffdepth,
        fhdepth,
maxtransectlocation,
        tempreal      : REAL;
humus1,
        humus2      : STRING[2];
        species,
        species_resp: STRING[5];
        comment      : STRING[40];
        screenline   : ARRAY[1..8] OF STRING[60];
        valid,
        write_data    : BOOLEAN;

```

```

PROCEDURE GETREP(max:INTEGER;VAR response:INTEGER);
{Subordinate to procedure IDENTIFYDATA,ENTERDATA}
{This procedure prompts the user for repetition number for data to be
  entered, checks for valid response.}

```

```

VAR
  rep  : CHAR;
  code : INTEGER;
  temptext : STRING[1];
BEGIN
  code := 0;
  REPEAT
    WINDOW(1,24,60,25);
    GOTOXY(1,2);CLREOL;
    WRITE('Rep mbr to be corrected:');
    READLN(rep);
    STR(max,temptext);
    valid := (rep > '0') AND
             (rep <= temptext);
    IF valid THEN VAL(rep,response,code);
    IF (code <> 0) OR (NOT valid) THEN
      BEGIN
        SOUND(900);
        DELAY(100);
        NOSOUND;
        WRITE(' Invalid response3. ');
      END;
  UNTIL valid;
END;{GETREP}

```

```

PROCEDURE SAVELINE;
{Saves contents of window(1,17,60,23) to array screenline. Upon entry
  into SAVELINE, y2 should have the row value for the last data line
  written to window2, upon exit y2 will have the value of the next line.}

```

```

VAR
  j4 : INTEGER;
  sid,
  srep,
  sdec,
  sdiam,
  sff,
  sfh      : STRING[20];
  template : STRING[60];
BEGIN
  WINDOW(1,1,80,15);CLRSCR;
  IF (focus_resp = 'D') THEN
    BEGIN

```

```

        STR(measid_resp:6,sid);
        STR(thisrep:4,srep);
        STR(decay:5,sdec);
        STR(diam:8:1,sdiam);
        templine := sid + srep + ' ' + species + sdec + sdiam + ' ' +
                    COPY(comment,1,15);
    END;
    IF (focus_resp = 'S') THEN
    BEGIN
        STR(measid_resp:6,sid);
        STR(thisrep:4,srep);
        STR(ffdepth:7:1,sff);
        STR(fhdepth:7:1,sfh);
        templine := sid + srep + sff + sfh + ' ' +
                    humus1 + ' ' + humus2 + ' ' + COPY(comment,1,25);
    END;
    IF (present_row = last_row + 1) THEN
        screenline[present_row] := templine;
    IF (last_row = 7) AND (present_row = 7) THEN
    BEGIN
        FOR j4 := 2 TO 7 DO screenline[j4-1] := screenline[j4];
        screenline[7] := templine;
    END;
END; {SAVELINE}

```

```

PROCEDURE RETRIEVEWINDOW2;
{Retrieves contents of prior window to (1,17,60,23) from "screenfile".
Upon entry to RETRIEVEWINDOW2, last_row should contain the ROW number
of the last line written to window 2.}

```

```

VAR
    j4 : INTEGER;
    templine : STRING[60];
BEGIN
    WINDOW(1,17,60,23);
    FOR j4 := 1 TO last_row DO
        BEGIN
            CLREOL;
            WRITELN(screenline[j4]);
        END;
    END; {RETRIEVEWINDOW2}

```

```

PROCEDURE GETSPECIES;
{Subprogram in ENTERDATA}
VAR

```

```

    j,
    j2 : WORD;
    temp : STRING[6];
    valid : BOOLEAN;
BEGIN
    j2 := 0;
    REPEAT
        WINDOW(1,24,60,25);
        GOTOXY(1,2);
        WRITE('Species code:');
        READLN(temp);
        j2 := j + 1;
        FOR j := 1 TO LENGTH(temp) DO temp[j] := UPCASE(temp[j]);
        IF (LENGTH(temp) = 4) OR (LENGTH(temp) = 5) THEN
            BEGIN
                j := 0;

```

```

                REPEAT
                    j := j + 1;
                    IF (j <= 4) THEN
valid := (temp[j] >= 'A') AND
                                                (temp[j] <= 'Z')
                ELSE
                    valid := ((temp[5] >= 'A') AND (temp[5] <= 'Z')) OR
                        ((temp[5] >= '0') AND (temp[5] <= '9'));
                    UNTIL (j = LENGTH(temp)) OR (NOT valid);
                END
            ELSE valid := FALSE;
            IF (NOT valid) THEN
                BEGIN
                    SOUND(900);
                    DELAY(100);
                    NOSOUND;
                    WRITE(' Invalid code. ');
                END;
            UNTIL valid;
            IF (LENGTH(temp) = 4) THEN temp := temp + 'X';
            species := copy(temp,1,5);
END; {GETSPECIES}

```

```

PROCEDURE ROLLDICE;
{Subprodecure for ENTERDATA}
{This procedure determines whether a given measurement will be repeated
to evaluate measurement error. The probability of remeasurement is
equivalent to Prob[X=1] for a Bernoulli RV with p=.05.}
VAR
    i4 : INTEGER;
    remeasure : BOOLEAN;
BEGIN
    remeasure := (RANDOM(99) <= 5);
    IF remeasure THEN
        BEGIN
            IF (present_row = 7) THEN
                BEGIN
                    FOR i4 := 2 TO 7 DO screenline[i4-1] := screenline[i4];
                    screenline[7] := 'Mark for repeat measurement';
                END
            ELSE
                BEGIN;
                    present_row := present_row + 1;
                    last_row := last_row + 1;
                    screenline[last_row] := 'Mark for repeat measurement';
                END;
            RETRIEVEWINDOW2;
        END;
    END;
END;

```

```

PROCEDURE IDENTIFYSOILDATA;
{Procedure for ENTERDATA}
{This procedure prompts the user for identification of data. It checks
to see if a data entry is a repeat of an earlier measurment. If it is,
it notifies the user and asks for verification or correct identification.}
VAR
    maxrep : INTEGER;
    correction,
    new,

```

```

    reenter,
    error_measure,
    match      : BOOLEAN;

PROCEDURE CHECKSOILDATA(VAR datafile:TEXT);
{Subordinate to procedure IDENTIFYDATA,ENTERDATA.}
{This procedure reads all data from the user-specified measurement
 unit, determines the maximum repetition number, and the number of
 data entries made for this measurement unit.}
VAR
    i : BYTE;
    ff_data,
    fh_data : ARRAY[1..5] OF REAL;
    h1_data,
    h2_data : ARRAY[1..5] OF STRING[2];
    h3_data: ARRAY[1..5] OF STRING[40];
    near      : BOOLEAN;

BEGIN
    {Main body of CHECKSOILDATA}
    repnmbr := 0;
    maxrep := 0;
    matches := 0;

    RESET(datafile);
    WHILE (NOT EOF(datafile)) DO
    BEGIN
        READLN(datafile,id,measid,repnmbr,ffdepth,fhdepth,
                humus1,humus2,blank,datetime,blank,comment);
        match := (id = id_resp) AND
                (measid = measid_resp);

        IF match THEN
            BEGIN
                matches := matches + 1;
                ff_data[repnmbr] := ffdepth;
                fh_data[repnmbr] := fhdepth;
                h1_data[repnmbr] := humus1;
                h2_data[repnmbr] := humus2;
                h3_data[repnmbr] := comment;
                IF (repnmbr > maxrep) THEN maxrep := repnmbr;

            END;
        END;
    CLOSE(datafile);
    IF (maxrep > 0) THEN
        BEGIN
            WINDOW(1,24,60,25);
            GOTOXY(1,2);CLREOL;
            WRITELN('Data already entered at ',measid_resp,' cm:');
            WINDOW(1,16,60,16);CLRSCR;
            WRITE('Rep    ff      fh  h1 h2 comment');
            WINDOW(1,17,60,23);CLRSCR;
            FOR i := 1 TO maxrep DO
                BEGIN
                    WRITELN(i:2,ff_data[i]:8:1,fh_data[i]:7:1,
                            ' ',h1_data[i],
                            ',h2_data[i],', ' ',
                            COPY(h3_data[i],1,25));

                END;
            END;
        END;
    END; {CHECKSOILDATA}

```

```

BEGIN          {Main body of IDENTIFYSOILDATA begins here}

    REPEAT
reenter := FALSE;
thisrep := 1;
    WINDOW(1,24,60,25);
    GOTOXY(1,2);CLREOL;
    GETREALVALUE('Location on transect(m):',tempreal,0,
                maxtransectlocation);
    measid_resp := ROUND(tempreal*100); {Dist in cm}
    ASSIGN(soil_table, soilfile);
    CHECKSOILDATA(soil_table);

    {If other entries are found for this data entity the user is
    prompted to clarify the type of measurment being recorded.}

    IF (maxrep > 0) THEN
        BEGIN
            error_measure := FALSE;
            correction := FALSE;
            WINDOW(1,24,60,25);CLRSCR;
            WRITELN(maxrep,' reps, ',matches,' entries already entered. ');
            IF (maxrep > 0) THEN
                BEGIN
                    GETYN('Error measurement?
',error_measure,'Y');

                    IF (NOT error_measure) THEN
                        GETYN('Is this a correction?
',correction,'Y');

                        END;

                    {Determine the appropriate rep number for the new data,
                    or ask the user to specify it}

                    IF error_measure THEN thisrep := maxrep + 1;
                    IF correction THEN GETREP(maxrep,thisrep);

                    IF (NOT correction) AND (NOT error_measure) THEN reenter :=
TRUE;
                        END;

                    UNTIL (NOT reenter); {Reenter value of 'TRUE' starts ENTERDATA again}

                END; {IDENTIFYSOILDATA}

PROCEDURE ENTERSOILDATA;
{Suprogram for ENTERDATA}
{This procedure prompts user for entry of data from the user-specified
plot.}

VAR
    i : INTEGER;
BEGIN
    write_data := TRUE;
    REPEAT
        WINDOW(1,16,60,16);CLRSCR;
        WRITE('Loc(cm) rep    ff        fh  h1 h2 comment(1st 25 chars)');
        WINDOW(1,17,60,23);
        IF write_data THEN

```

```

BEGIN
  CLRSCR;
  IF (last_row >= 1) THEN RETRIEVEWINDOW2;
END;
GETREALVALUE('Forest floor depth (cm): ',ffdepth,0,300);
  GOTOXY(1,1);CLREOL;
  WRITE('(' ,measid_resp,'): ',ffdepth:5:1);
  GETREALVALUE('First horizon depth (cm): ',fhdepth,0,300);
  GOTOXY(1,1);CLREOL;
  WRITE('(' ,measid_resp,'): ',ffdepth:5:1,fhdepth:5:1);
WINDOW(1,24,60,25);
GOTOXY(1,2);CLREOL;
REPEAT
  GOTOXY(1,2);CLREOL;
  WRITE('Mull, mor, or moder? (L,R,D):');
  READLN(humus1);
  FOR i := 1 TO 2 DO humus1[i] := UPCASE(humus1[i]);
  valid_ans := (humus1 = 'L') OR
                (humus1 = 'R') OR
                (humus1 = 'D');
  IF (LENGTH(humus1) > 1) THEN
    BEGIN
      WRITE('Response must be 1 char. ');
      valid_ans := FALSE;
    END;
    IF (NOT valid_ans) THEN
      BEGIN
        SOUND(900);
        DELAY(100);
        NOSOUND;
        WRITE('Invalid value.');
      END;
UNTIL valid_ans;
  GOTOXY(1,1);CLREOL;
  WRITE('(' ,measid_resp,'): ',ffdepth:5:1,fhdepth:5:1,humus1:2);

REPEAT
  GOTOXY(1,2);CLREOL;
  WRITE('Xeric, mesic, or hygric? (X,M,H):');
  READLN(humus2);
  FOR i := 1 TO 2 DO humus2[i] := UPCASE(humus2[i]);
  valid_ans := (humus2 = 'X') OR
                (humus2 = 'M') OR
                (humus2 = 'H');
  IF (LENGTH(humus2) > 1) THEN
    BEGIN
      WRITE('Response must be 1 char. ');
      valid_ans := FALSE;
    END;
    IF (NOT valid_ans) THEN
      BEGIN
        SOUND(900);
        DELAY(100);
        NOSOUND;
        WRITE('Invalid value.');
      END;
UNTIL valid_ans;
  GOTOXY(1,2);CLREOL;
  WRITE('(' ,measid_resp,'): ',ffdepth:5:1,fhdepth:5:1,humus1:2,humus2:2);

```

```

GOTOXY(1,2);CLREOL;
      WRITELN(' ',measid_resp,' ) Comment (<=40 chars):');
READLN(comment);

WINDOW(1,17,60,23);
IF (last_row = 7) THEN present_row := 7
ELSE present_row := last_row + 1;
GOTOXY(1,present_row);

WRITE(measid_resp:6,thisrep:4,ffdepth:7:1,fhdepth:7:1,
      ' ',humus1,' ',humus2,' ',COPY(comment,1,25));
CLREOL;

GETYN('Last data line correct? ',write_data,'Y');
IF (NOT write_data) THEN
  BEGIN
    WINDOW(1,24,60,25);
    GOTOXY(1,2);CLREOL;
    WRITE('Reenter information.');
```

{Write soil data}

```

  END;
UNTIL write_data;

ASSIGN(soil_table, soilfile);
APPEND(soil_table);
GETDATETIME;
WRITELN(soil_table,id_resp:3,measid_resp:6,thisrep:2,ffdepth:6:1,
      fhdepth:6:1,humus1:2,humus2:2,datetime:19,' ',comment);
CLOSE(soil_table);
SAVELINE;
last_row := present_row;
IF (thisrep = 1) THEN ROLLDICE;

END; {ENTERSOILDATA}

PROCEDURE ENTERDEBRISDATA;
{Subprocedure for ENTERDATA}
{This procedure prompts user for entry of data from the user-specified
plot.}

PROCEDURE CHECKDEBRISDATA;
{Checks previously entered data to determine whether the present
measurement has been done previously.}
VAR
  i2,
  matches2,
  maxrep2,
  nmbr_near : INTEGER;
  sp_data : ARRAY[1..5] OF STRING[5];
  d_data : ARRAY[1..5] OF INTEGER;
  diam_data :ARRAY[1..5] OF REAL;
  correct2,
  error2,
  done,
  match2,
  near,
  reenter2 : BOOLEAN;
BEGIN
  REPEAT
    nmbr_near := 0;
```

```

matches2 := 0;
maxrep2 := 0;
reenter2 := FALSE;
ASSIGN(debris_table,debrisfile);
RESET(debris_table);
WHILE (NOT EOF(debris_table)) DO
  BEGIN
    READLN(debris_table,id,measid,repnbr,blank,species,decay,
            diam);
    match2 := (id = id_resp) AND
              (measid = measid_resp);
    near := (id = id_resp) AND
            (ABS(measid - measid_resp) < 50);
    IF match2 THEN
      BEGIN
        matches2 := matches2 + 1;
        sp_data[repnbr] := species;
        d_data[repnbr] := decay;
        diam_data[repnbr] := diam;
        IF (repnbr > maxrep2) THEN maxrep2 := repnbr;
      END;
    IF near THEN nmbr_near := nmbr_near + 1;
  END;
CLOSE(debris_table);
IF (maxrep2 >= 1) THEN
  BEGIN
    WINDOW(1,24,60,25);
    GOTOXY(1,2);
    WRITELN('Data already entered..');
    WINDOW(1,16,60,16);CLRSCR;
    WRITE('Loc(cm) rep species decay diam(cm)');
    WINDOW(1,17,60,23);CLRSCR;
    FOR i2 := 1 TO maxrep2 DO
      WRITELN(measid_resp:6,i2:4,'
,sp_data[i2],
            d_data[i2]:5,diam_data[i2]:8:1);
    END;
  IF (maxrep2 = 0) AND (nmbr_near > 0) THEN
    BEGIN
      WINDOW(1,24,60,25);
      GOTOXY(1,2);
      WRITELN('Data entered near this pt..');
      WINDOW(1,16,60,16);CLRSCR;
      WRITE('Loc(cm) rep species decay diam(cm)');
      WINDOW(1,17,60,23);CLRSCR;
      ASSIGN(debris_table,debrisfile);
      RESET(debris_table);
      WHILE (NOT EOF(debris_table)) DO
        BEGIN
          READLN(debris_table,id,measid,repnbr,blank,species,
                  decay,diam);
          IF (id = id_resp) AND (ABS(measid-measid_resp)<50)
            THEN WRITELN(measid:6,repnbr:4,'
,
species,decay:5,diam:8:1);
        END;
      CLOSE(debris_table);
      REPEAT
        WINDOW(1,24,60,25);

```

```

                GOTOXY(1,2);
                GETYN('Nearby data values displayed. Continue?',
                    done,'Y');
            UNTIL done;
        END;
    {If other entries are found for this data entity the user is
     prompted to clarify the type of measurement being recorded.}

    IF (maxrep2 > 0) THEN
        BEGIN
            error2 := FALSE;
            correct2 := FALSE;
            WINDOW(1,24,60,25);
            GOTOXY(1,2);
            WRITELN(maxrep2,' reps, ',matches2,' entries already entered.');
```

IF (maxrep2 > 0) THEN

```

                BEGIN
                    GOTOXY(1,2);
                    GETYN('Error measurement? ',error2,'Y');
                    IF (NOT error2) THEN
                        BEGIN
                            GOTOXY(1,2);
                            GETYN('Is this a correction? ',correct2,'Y');
                        END;
                    END;
                END;

                {Determine the appropriate rep number for the new data,
                 or ask the user to specify if it is not obtainable by
                 computation.}

                IF error2 THEN thisrep := maxrep2 + 1;
                IF correct2 THEN GETREP(maxrep2,thisrep);

                reenter2 := ((NOT correct2) AND (NOT error2));
            END;

            UNTIL (NOT reenter2); {Reenter2 = 'TRUE' starts CHECKDEBRISDATA again}
        END; {CHECKDEBRISDATA}

    BEGIN
        write_data := TRUE;
        REPEAT
            thisrep := 1;
            WINDOW(1,24,60,25); CLRSCR;
            GETREALVALUE('Location on transect(m):',tempreal,0,
                maxtransectlocation);
            measid_resp := ROUND(tempreal*100);           {Dist in cm}
            CHECKDEBRISDATA;
            WINDOW(1,17,60,23);
            IF write_data THEN
                BEGIN
                    CLRSCR;
                    IF (last_row >= 1) THEN RETRIEVEWINDOW2;
                END;
            WINDOW(1,16,60,16); CLRSCR;
            WRITE('Loc(cm) rep species decay diam comment(1st 15 chars)');
            WINDOW(1,24,60,25); CLRSCR;
            GETSPECIES;
            GOTOXY(1,1); CLREOL;
            WRITE('(',measid_resp,') ',species);

```

```

GETINTVALUE('Decay class (1,2,3,4,5):',decay,1,5);
  GOTOXY(1,1);CLREOL;
  WRITE('(',measid_resp,') ',species,decay:2);
GETREALVALUE('Diam at intercept (cm): ',diam,0,400);
  WRITELN('(',measid_resp,') ',species,decay:2,diam:5:1,
' Comment?
(<=40 chars):');
  READLN(comment);
  WINDOW(1,17,60,23);
  IF (last_row = 7) THEN
    BEGIN
      present_row := 7;
    END
  ELSE present_row := last_row + 1;
  GOTOXY(1,present_row);
  WRITE(measid_resp:6,thisrep:4,' ',species,decay:5,
    diam:8:1,' ',COPY(comment,1,25));
  CLREOL;
  WINDOW(1,24,60,25);
  GOTOXY(1,2);CLREOL;
  GETYN('Last line of data correct? ',write_data,'Y');
  IF (NOT write_data) THEN WRITELN('Reenter information.');
```

```

UNTIL write_data;
SAVELINE;
last_row := present_row;

ASSIGN(debris_table, debrisfile);
APPEND(debris_table);
GETDATETIME;
WRITELN(debris_table,id_resp:3,measid_resp:6,thisrep:2,species:6,
    decay:2,diam:5:1,datetime:19,' ',comment);
CLOSE(debris_table);
IF (thisrep = 1) THEN ROLLDICE;
END;{ENTERDEBRISDATA}

BEGIN                                {Main body of ENTERDATA}
  CLRSCR;

  maxtransectlocation := 0;
  ASSIGN(dimension_table,dimfile);
  RESET(dimension_table);
  WHILE (NOT EOF(dimension_table)) DO
    REPEAT
      READLN(dimension_table,id,blank,pt_type,az,dist);
      IF (id = id_resp) THEN
        maxtransectlocation := dist + 0.001;
      UNTIL (maxtransectlocation > 1);
  CLOSE(dimension_table);

  REPEAT
    CASE focus_resp OF
      'S':BEGIN
        WINDOW(1,16,60,25);CLRSCR;
        last_row := 0;
        present_row := 0;
        FOR i3 := 1 TO 8 DO screenline[i3] := '';
        REPEAT
          IDENTIFYSOILDATA;
```

```

                ENTERSOILDATA;
                GETYN('Cont this transect?
',end_entry,'N');
                UNTIL end_entry;
                END;

                'D':BEGIN
                WINDOW(1,17,60,23);CLRSCR;
                last_row := 0;
                present_row := 0;
                FOR i3 := 1 TO 8 DO screenline[i3] := '';
                REPEAT
                        ENTERDEBRISDATA;
                        WINDOW(1,24,60,25);CLRSCR;
                        GETYN('Cont this transect?
',end_entry,'N');
                        UNTIL end_entry;
                        END;

                END;
                UNTIL end_entry;
END; {ENTERDATA}

BEGIN
here.}
                {Main body of program begins

                RANDOMIZE;
                REPEAT
                        CLRSCR;
                        WINDOW(1,16,60,25);CLRSCR;
                        WINDOW(1,16,60,23);
                        GETSITE;
                        GETFOCUS;
                        GETACTION;
                        IF (NOT end_program) THEN

                                BEGIN
                                        IF old_plot THEN GETOLDPLOTID;
                                        IF new_plot OR change THEN ENTERPLOTINFO;
                                        ENTERDATA;
                                        GETYN('Enter data for different transect?
',end_program,'N');
                                END;

                                UNTIL end_program;

                                {Main program ends here.}
END.

```