*tp

18264

MICRO CDS ISIS PASCAL PROGRAMS

FOR THE PETRO DATABASE SUPPORTING

5r

```
       Program SELFOR(f: string) [menu];
       {     This program selects one of the display formats                   }
       {     A moving arrow and highlighting are used as markers               }
       {     The program is invoked from the menu EXGEN, option F              }

                            { **** by M.Muraszkiewicz, Dec.4,    1989 **** }

       var    sop, lin, col, hi, wi, le                :real;
              str, fornm                               :string;

       Function SELECT(stlin,stcol,high,wide,len :real;str :string): real;
           {  This function allows for selecting an option by means    }
           {  of an arrow and highlighting.                            }
           {    Input paramaters:  stlin, stcol, high, wide, len       }
           {    Output parameters: SELECT, str                         }
       var sc, i, k :real;
       begin
       i:= 0; k:= 0; str:= '';
       repeat
       CURSOR(stlin+1+i,stcol+wide);
       writeln('<----');                                   { draw  new arrow
}
       ATTR(' ',0,26,80,1);                                { hide a cursor
}
       sc:= KBDKEY(str);
       if sc = 80 then i:= i+1;                            { down arrow was met
}
       if sc = 72 then i:= i-1;                            { up arrow was met
}
       if i > high-3 then i:= 0;                           { skip to the top
}
       if i < 0 then i:=high-3;                            { skip to the bottom
}
       if sc=72 or sc=80 then
          begin
          CLEARBOX(stlin+1+k,stcol+wide,1,5,0);            { remove old arrow
}
          CHATTR(0,stlin+1+k,stcol+1,len);                 { remove old hlight
}
          CHATTR(2,stlin+1+i,stcol+1,len);                 { put new highlight
}
          end;
       k:= i;
       until sc=28 or str='x' or str='X';                  { ENTER or X was met
}
       SELECT:= k;
       end;

       Procedure INDEX_BOX(stlin,stcol,high,wide,len :real);
       {  Draws a box and fills it out with indexes names  }
       begin
```

```
BOX(stlin,stcol,high,wide,1);
ATTR(' ',2,stlin+1,stcol+1,len);                    { 2 - hlight  }
CURSOR(stlin+1,stcol+1); write('Display Enterprises');
CURSOR(stlin+2,stcol+1); write('Display Products');
CURSOR(stlin+3,stcol+1); write('Display Processes');
CURSOR(stlin+high+1,stcol+7); write('X - Exit');
end;
```

```
Procedure MESSAGE;
begin
CURSOR(3,30); write('Display Formats');
CURSOR(22,1); write('Select your format using down and/or up arrows');
CURSOR(23,1); write('Confirm you choice by pressing ENTER');
CURSOR(24,1); write('Strike X to exit');
end;
                      { -------- Body of Program -------- }
begin
CLEAR;                                                { clear screen }
MESSAGE;                                              { display msgs }
CURSOR(4,3);

   { ---------- Display and pick up a format name ----------- }
   lin:= 8; col:= 24; hi:= 5; wi:= 25; le:= wi-2;
   INDEX_BOX(lin,col,hi,wi,le); CURSOR(lin-1,col+1); {draw a box + opt
}
   sop:= SELECT(lin,col,hi,wi,le,str);                   { pick up by <---
}
   if str <> 'x' and str <> 'X' then
      begin                                              { X - Exit not me
 }
      { --------- formats --------- }
      if sop = 0 then fornm:='ENTRP';
      if sop = 1 then fornm:='PROD';
      if sop = 2 then fornm:='PROC';
      GETFMT('@'!fornm);
      end;
   f:= ' ';                                               { return to menu
}
end.

Program SELWSH(w: string) [menu];
{     This program selects a worksheet from PETRO and PRODUCT        }
{     A moving arrow and highlighting are used as markers            }
{     The program is invoked from the menu EXE1, option W            }

                         { **** by M.Muraszkiewicz, Nov. 15, 1989 **** }

var    sop, d, lin, col, hi, wi, le          :real;
       str, sheet                            :string;

Function SELECT(stlin,stcol,high,wide,len :real;str :string): real;
    {  This function allows for selecting an option by means     }
    {  of an arrow and highlighting.                             }
    {    Input parameters:  stlin, stcol, high, wide, len        }
    {    Output parameters: SELECT, str                          }
var sc, i, k :real;
begin
i:= 0; k:= 0; str:= '';
```

```
repeat
```

```
        CURSOR(stlin+1+i,stcol+wide);
        writeln('<----');                                    { draw  new arrow
}
        ATTR(' ',0,26,80,1);                                 { hide a cursor
}
        sc:= KBDKEY(str);
        if sc = 80 then i:= i+1;                             { down arrow was met
}
        if sc = 72 then i:= i-1;                             { up arrow was met
}
        if i > high-3 then i:= 0;                            { skip to the top
}
        if i < 0 then i:=high-3;                             { skip to the bottom
}
        if sc=72 or sc=80 then
           begin
           CLEARBOX(stlin+1+k,stcol+wide,1,5,0);             { remove old arrow
}
           CHATTR(0,stlin+1+k,stcol+1,len);                  { remove old hlight
}
           CHATTR(2,stlin+1+i,stcol+1,len);                  { put new highlight
}
           end;
        k:= i;
        until sc=28 or str='x' or str='X';                   { ENTER or X was met
}
        SELECT:= k;
        end;

        Procedure INDEX_BOX(stlin,stcol,high,wide,len :real);
        {  Draws a box and fills it out with worksheets names   }
        begin
        BOX(stlin,stcol,high,wide,1);
        ATTR(' ',2,stlin+1,stcol+1,len);                     { 2 - hlight  }
        CURSOR(stlin+1,stcol+1); write('Worksheet for Enterprise');
        CURSOR(stlin+2,stcol+1); write('Worksheet for Product');
        CURSOR(stlin+3,stcol+1); write('Worksheet for Process');
        CURSOR(stlin+high+1,stcol+7); write('X - Exit');
        end;

        Procedure MESSAGE;
        begin
        CURSOR(3,24); write('Data Entry Worksheets Available');
        CURSOR(22,1); write('Select your worksheet using down and/or up arrows
');
        CURSOR(23,1); write('Confirm you choice by pressing ENTER');
        CURSOR(24,1); write('Strike X to exit');
        end;
                        { ------- Body of Program - ------ }
        begin
        CLEAR;                                              { clear screen }
        MESSAGE:                                          { display msgs }
        CURSOR(4,3);

           { ---------- Display and pick up a worksheet ----------- }
           lin:= 8: col:= 27; hi:= 5; wi:= 27; le:= wi-2;
           INDEX BOX(lin,col,hi,wi,le); CURSOR(lin-1,col+1); {draw a box + opt
}
           sop:= SELECT(lin,col,hi,wi,le,str);               { pick up by <---
}
           if str <> 'x' and str <> 'X' then
           begin                                              { X - Exit not me
 }
           { --------- worksheets --------- }
           if sop = 0 then sheet:='ENTRP';
           if sop = 1 then sheet:='PROD';
           if sop = 2 then sheet:='PROC';
           d:= WORKSHEET(sheet);
           end;
        w:= ' ';                                               { return to menu
}
     end.


     Program SELIND(s: string) [menu];
     {     This program selects one of the PRTRO Indexes for printing    }
     {     A moving arrow and highlighting are used as markers           }
     {     The program is invoked from the menu EXPRT, option S          }

                           { **** by M.Muraszkiewicz, Dec. 7, 1989 **** }

     var    sop1, sop2, lin, col, hi, wi, le, t  :real;
            str, sheet                           :string;

     Function SELECT(stlin,stcol,high,wide,len,tg :real;str :string): real;
         {  This function allows for selecting an option by means    }
         {  of an arrow and highlighting.                            }
         {    Input paramaters:  stlin, stcol, high, wide, len, tg  }
         {    Output parameters: SELECT, str                        }
     var sc, i, k :real;
     begin
     i:= 0; k:= 0; str:= '';
     REPEAT
     CURSOR(stlin+1+i,stcol+wide);
     writeln('<----');                                   { draw  new arrow
}
     ATTR(' ',0,26,80,1);                                { hide a cursor
}
     sc:= KBDKEY(str);
     if sc = 80 and tg = 0 then i:= i+1;
     if sc = 80 and tg = 1 then begin                    { down arrow was met
}
                         i:= i+1;
                         if i=2 then i:=3;
                         if i=4 then i:=5;
                         if i=6 then i:=7;
                         end,
     if sc = 72 and tg = 0 then i:= i-1;
     if sc = 72 and tg = 1 then begin                    { up arrow was met
}
                         i:= i-1;
                         if i=4 then i:=3;
if i=2 then i:=1;
if i < 0 then i:=5;
end;
```

```
        if i > high-3 then i:= 0;                                  { skip to the top
}
        if i < 0 then i:= high-3;                                  { skip to the bottom
}
        if sc=72 or sc=80 then
           begin
           CLEARBOX(stlin+1+k,stcol+wide,1,5,0);                   { remove old arrow
}
           CHATTR(0,stlin+1+k,stcol+1,len);                        { remove old hlight
}
           CHATTR(2,stlin+1+i,stcol+1,len);                        { put new highlight
}
           end;
        k:= i;
        until sc=28 or str='x' or str='X';                         { ENTER or X was met
}
        SELECT:= k;
        end;

        Procedure INDEX_BOX1(stlin,stcol,high,wide,len :real);
        {  Draws a box and fills it out with indexes names  }
        begin
        BOX(stlin,stcol,high,wide,1);
        ATTR(' ',2,stlin+1,stcol+1,len);                           { 2 - hlight  }
        CURSOR(stlin+1,stcol+1); write('Whole database');
        CURSOR(stlin+2,stcol+1); write('Hits of standard queries');
        CURSOR(stlin+3,stcol+1); write('Hits of predefined queries');
        CURSOR(stlin+high+1,stcol+7); write('X - Exit');
        end;

        Procedure INDEX_BOX2(stlin,stcol,high,wide,len :real);
        {  Draws a box and fills it out with indexes names  }
        begin
        BOX(stlin,stcol,high,wide,1);
        ATTR(' ',2,stlin+1,stcol+1,len);                           { 2 - hlight  }
        CURSOR(stlin+1,stcol+1); write('Enterprises');
        CURSOR(stlin+2,stcol+1); write('Products');
        CURSOR(stlin+3,stcol+1); write('Processes');
        CURSOR(stlin+high+1,stcol+7); write('X - Exit');
        end;

        Procedure INDEX_BOX3(stlin,stcol,high,wide,len :real);
        {  Draws a box and fills it out with answers to queries  }
        begin
        BOX(stlin.stcol,high,wide,1);
        ATTR(' ',2,stlin+1,stcol+1,len);                           { 2 - hlight  }
        CURSOR(stlin+1,stcol+1);
        write('Manufacturer(s) of a given product');
        CURSOR(stlin+2,stcol+1);
        write('Materials for manufacturing the product');
        CURSOR(stlin+3,stcol+1);
```

```
write('  - and manufacturer(s) of these materials?');
CURSOR(stlin+4,stcol+1);
write('Products manufactured from a given material');
```

```
       CURSOR(stlin+5,stcol+1);
       write('  - other materials needed for the product and the
       manufacturers');
       CURSOR(stlin+6,stcol+1);
       write('Licensor(s) of a given process');
       CURSOR(stlin+7,stcol+1);
       write('  - description and level of the process');
       CURSOR(stlin+high+1,stcol+27); write('X - Exit');
       end;

       Procedure MESSAGE;
       begin
       CURSOR(2,24); write('PETRO Indexes - Printing');
       CURSOR(22,1); write('Select your index using down and/or up arrows');
       CURSOR(23,1); write('Confirm you choice by pressing ENTER');
       CURSOR(24,1); write('Strike X to exit');
       end;
                       { -------- Body of Program -------- }
       begin
       CLEAR;                                            { clear screen }
       MESSAGE;                                          { display msgs }
       CURSOR(5,3);

              { --------- Hits or Whole database ? ----------- }
       write('Do you want to generate indexes from hits or whole database?');
       lin:= 8; col:= 24; hi:= 5; wi:= 28; le:= wi-2;
       t:= 0;    { simple SELECT }
       s:= ' ';                                          { return to print men
}
       INDEX_BOX1(lin,col,hi,wi,le);                     { draw a box + option
}
       sop1:= SELECT(lin,col,hi,wi,le,t,str);            { pick up by <----
}

       CLEAR;
       if str <> 'x' and str <> 'X' then
          begin                                           { X - Exit not met }
          { ---------- Display and pick up Query answer types ----------- }
          MESSAGE;
          CASE sop1 OF
          0: begin                                      { whole database }
             lin:= 8; col:= 24; hi:= 5; wi:= 22; le:= wi-2;
             CURSOR(lin-2,col+1); write('WHOLE DATABASE');
             INDEX_BOX2(lin,col,hi,wi,le);
             sop2:= SELECT(lin,col,hi,wi,le,t,str);  { pick up by <--- }
             if str <> 'x' and str <> 'X' then
               begin                                    { X - Exit not met }
             { ----------- worksheets ----------------- }
               if sop2 = 0 then sheet:='eyd0';      { enterprise }
               if sop2 = 1 then sheet:='eyd1';      { products   }
               if sop2 = 2 then sheet:='ey2';       { processes  }
```

```
         end;
        end;  { whole database }
   1: begin                                            { hits-standard }
       lin:= 8; col:= 24; hi:= 5; wi:= 22; le:= wi-2;
       INDEX_BOX2(lin,col,hi,wi,le);
       CURSOR(lin-2,col+1); write('HITS OF STANDARD QUESTIONS');
       sop2:= SELECT(lin,col,hi,wi,le,t,str);  { pick up by <--- }
       if str <> 'x' and str <> 'X' then
         begin                                         { X - Exit not met }
       { ----------- worksheets ----------------- }
         if sop2 = 0 then sheet:='eyh0';          { enterprise }
         if sop2 = 1 then sheet:='eyh1';          { products   }
         if sop2 = 2 then sheet:='eyh2';          { processes  }
         end;
        end;  { hits-standard }
   2: begin                                            { hits-predefined }
       lin:= 8; col:= 6; hi:= 9; wi:= 67; le:= wi-2; t:= 1;
       INDEX_BOX3(lin,col,hi,wi,le);
       CURSOR(lin-2,col+1); write('HITS OF PREDEFINED QUESTIONS');
       sop2:= SELECT(lin,col,hi,wi,le,t,str); { pick up by <--- }
       if str <> 'x' and str <> 'X' then
         begin                                         { X - Exit not met }
       { ----------- worksheets ----------------- }
         if sop2 = 0 then sheet:='eyq0';
         if sop2 = 1 then sheet:='eyq1';
         if sop2 = 3 then sheet:='eyq3';
         if sop2 = 5 then sheet:='eyq5';
         end;
        end;  { hit-predefined }
   end;  { CASE }
   if str <> 'x' and str <> 'X' then
     begin
     AUTOTYPE(sheet);
     CLEARMSG; CURSOR(22,21); write(sheet);
     s:= '.S';                                { return & call a worksheet }
     end;
   end;  { if }                               { return to menu }
end.


Program QUERY(f: string) [menu];
{     This program displays standard queries and process them          }
{     The program is invoked from the menu ?????, option ?             }

                          { **** by M.Muraszkiewicz, Nov. 30, 1989 **** }


var    sop, lin, col, hi, wi, le, d          :real;
```

```
         str, fornm  inline, term             :string;

   Function SELECT(stlin,stcol,high,wide,len :real;str :string): real;
       {  This function allows for selecting an option by means    }
       {  of an arrow and highlighting.                            }
       {     Input paramaters:  stlin, stcol, high, wide, len      }
       {     Output parameters: SELECT, str                        }
   var sc, i, k :real;
   begin
   i:= 0; k:= 0; str:= '';
   repeat
   CURSOR(stlin+1+i,stcol+wide);
   writeln('<----');                                { draw  new arrow
}
   ATTR(' ',0,26,80,1);                             { hide a cursor
}
   sc:= KBDKEY(str);
   if sc = 80 then begin                            { down arrow was met
}
                   i:= i+1;
                   if i=2 then i:=3;
                   if i=4 then i:=5;
                   if i=6 then i:=7;
                   end;
   if sc = 72 then begin                            { up arrow was met
}
                   i:= i-1;
                   if i=4 then i:=3;
                   if i=2 then i:=1;
                   if i < 0 then i:=5;
                   end;
   if i > high-3 then i:= 0;                        { skip to the top
}
   if i < 0 then i:= high-3;                        { skip to the bottom
}
   if sc=72 or sc=80 then
      begin
      CLEARBOX(stlin+1+k,stcol+wide,1,5,0);         { remove old arrow
}
      CHATTR(0,stlin+1+k,stcol+1,len);              { remove old hlight
}
      CHATTR(2,stlin+1+i,stcol+1,len);              { put new highlight
}
      end;
   k:= i; {if i=3 then i:=2;}
   until sc=28 or str='x' or str='X';               { ENTER or X was met
}
   SELECT:= k;
   end;

   Procedure INDEX_BOX(stlin,stcol,high,wide,len :real);
```

```
{  Draws a box and fills it out with query skeletons  }
begin
BOX(stlin,stcol,high,wide,1);
ATTR(' ',2,stlin+1,stcol+1,len);                     { 2 - hlight  }
CURSOR(stlin+1,stcol+1);
write('Who is manufacturing the product?');
CURSOR(stlin+2,stcol+1);
```

```
write('What are the materials for manufacturing the product?');
CURSOR(stlin+3,stcol+1);
write('  - who is manufacturing these materials?');
CURSOR(stlin+4,stcol+1);
write('What can be manufactured from a given material?');
CURSOR(stlin+5,stcol+1);
write('  - which are other materials and where are they
available?');
CURSOR(stlin+6,stcol+1);
write('Who is the licensor of the process?');
CURSOR(stlin+7,stcol+1);
write('  - what is the description and level of the process?');
CURSOR(stlin+high+1,stcol+27); write('X - Exit');
end;

Procedure MESSAGE;
begin
CURSOR(3,35); write('QUERIES');
CURSOR(22,1); write('Select your query using down and/or up arrows');
CURSOR(23,1); write('Confirm you choice by pressing ENTER');
CURSOR(24,1); write('Strike X to exit');
end;

Procedure DELFIELD(fldtag: real);
{   Deletes the field number fldtag }
var nocfld,fldnr,j,d  :real;
begin
fldnr:= FIELDN(fldtag,1);        { get a field number    }
IF fldnr <> 0 THEN
   begin                         { delete all occ in 997 }
   nocfld:= NOCC(fldtag);
   FOR j:=1 TO nocfld DO
     begin
     fldnr:= FIELDN(fldtag,1);
     d:= FLDDEL(fldnr);
     end;
   end;
end;

Procedure LOOKO(prf,line: string);
{   Searches for a set of hits. Puts value of the line          }
{   to the dummy field 998 for a reason required by the         }
{   display format ENTRPO.PFT                                   }

var    setnum,nrr,nfm,i,d                              :real;
       n998                                            :real;
       str,upline                                      :string;
begin
```

```
        upline:= line; UC(upline);  { convert to uper case }
        str:= prf!line;
        setnum:= SEARCH(str);        { get set number of the query }
        nrr:= SETPOS(setnum,0);      { give number of hits }
        FOR i:=1 to nrr DO           { process hit-by-hit  }
          begin
          nfm:= SETPOS(setnum,i);
          d:= RECORD(nfm);           { get a record         }
          DELFIELD(998);             { delete 998           }
          d:= FLDADD(998,1,line);    { set up 998           }
          UPDATE;                    { give record back     }
          end; { end of hit-by-hit loop }
end;

Procedure LOOK5(prf,line: string);
{   Searches for a set of hits. Set value of the fiels 120^s    }
{   to the dummy fields 997, 998 and 999 (repeatable) for       }
{   reasons required by the display format ENTRP5.PFT           }

var     setnum,nrr,nfm,i,j,d,dd                     :real;
        n120,n997,noc120                            :real;
        str,f120,f120up,upline                      :string;
begin
        upline:= line; UC(upline);  { convert to uper case }
        str:= prf!line;
        setnum:= SEARCH(str);        { get set number of the query }
        nrr:= SETPOS(setnum,0);      { give number of hits }
        FOR i:=1 to nrr DO           { process hit-by-hit  }
        begin
          nfm:= SETPOS(setnum,i);
          d:= RECORD(nfm);             { get a record }

 { -- transfer 120 to 997 while converting to upper case -- }
          DELFIELD(997);
          noc120:= NOCC(120);
          FOR j:=1 TO noc120 DO
            begin                  { transfer and convert }
            n120:= FIELDN(120,j);
            n997:=FIELDN(997,j); f120:= FIELD(n120);
            f120up:= f120; UC(f120up); dd:= FLDADD(997,1,f120up);
            end;

{ -- set up 998 and 999 (upper case) with PROCESS NAME -- }
          DELFIELD(998);
          DELFIELD(999);
          dd:= FLDADD(998,1,line);     { put line to 998 }
          IF d = 0 THEN                { put line to 999 }
                 begin
```

```
                      FOR j:=1 TO noc120+1 DO
                        begin
                        dd:= FLDADD(999,1,upline);
                        end;
                      end;
             UPDATE;
           end; { end of hit-by-hit loop }
      end;

                    { -------- Body of Program -------- }
      begin
      CLEAR;                                              { clear screen }
      MESSAGE;                                            { display msgs }
      CURSOR(4,3);
      OPEN('petro');
         { ---------- Display and pick up a query ----------- }
         lin:= 8; col:= 6; hi:= 9; wi:= 67; le:= wi-2;
         INDEX_BOX(lin,col,hi,wi,le); CURSOR(lin-1,col+1); {draw a box + opt.
}
         sop:= SELECT(lin,col,hi,wi,le,str);                  { pick up by <---
}
         if str <> 'x' and str <> 'X' then
            begin                                             { X - Exit not met
 }
         { ---------- Now we process queries ---------------- }
          CLEAR;
          CURSOR(2,8);
          write('ATT! Usage OF $ for query truncation is NOT allowed !!!');
          CURSOR(4,4);
          write('YOUR QUERY WAS:');
          CASE sop OF                                    { select a query "processor"
 }
          0: begin
             CURSOR(6,4); write('Who is manufacturing the product?');
             CURSOR(8,4); write('Please enter the product name:');
             CURSOR(10,4); readln(inline);
             LOOKO('PROD=',inline); GETFMT('@ENTRPO');
             end;
          1: begin
             CURSOR(6,4);
             write('What are the materials for manufacturing the product?');
             CURSOR(7,4);
             write('  - who is manufacturing these materials?');
             CURSOR(9,4); write('Please enter the product name:');
             CURSOR(11,4); write ('NOT IMPLEMENTED !!!');
             { readln(inline); }
             end;
          3: begin
             CURSOR(6,4);
             write('What can be manufactured from a given material?');
             CURSOR(7,4);
```

```
write('  - which are other materials and where are ');
```

```
        write('they available?');
        CURSOR(9,4); write('Please enter the material name:');
        CURSOR(11,4); write('NOT IMPLEMENTED !!!');
        { readln(inline); }
        end;
     5: begin
        CURSOR(6,4);
        write('Who is the licensor of the process?');
        CURSOR(7,4);
        write('  - what is the description and level of the process?');
        CURSOR(9,4); write('Please enter the process name:');
        CURSOR(11,4); readln(inline);
        LOOK5('PSL=',inline); GETFMT('@ENTRP5');
        end;
     end; { CASE }
  end; { end of processing queries }
  d:= WORKSHEET('ENTRP');
  f:= ' '; { return to menu }
end.

Program chem(option: string) {menu};

var dt: array[1..15] of real;    {displayed tags array}
    doc: array[1..15] of real;   {displayed occurences array}
    dmfn: array[1..15] of real;  {displayed mfns array}

    mfnstack: array[1..30] of real;   {mfn stack - only for COF}
    dld: array[1..30] of real;  {displayed levels indicators: 0}
                               {0 - last node taken on this level;
    1 - otherwise}
    tagstack: array[1..30] of real;   {tags occurences stack -
    only for COF}
    stackptr: real;
    tag: array[1..10] of real;   { tag of relation }
    maxt: real;                  { max no. of tags (upper bound
                                   of tag) }
    tst:  string;
    maxl: real;                  { max no. of lines (upper bound
                                   of dt,doc) }
    rel,invrel: string;          { Relation indicators }

    pgno :real;                 {current page number}
    maxpg:real;                 {last page number}

    it,io: real;                 { current tag/occ }
    nl: real;                    { lines on this page }
    cl: real;                    { current line }
    term: string;                { current term }
    q: string;                        { query }
    dbname: string;                   { current data base }
    cmfn: real;                      { current mfn to be put on stack }
    mfn: real;                      { top mfn }
    s,action,ft: string;
    i,k,kl,lq,rc: real;



PROCEDURE HELP(H:STRING);
{------------------------------------------------------------------}
{ Display help screen - H is an extension of HELP file             }
{------------------------------------------------------------------}

var s,c: string;
    i: real;
begin
savescr(1);
assign('INP','C:SISETROETDATAELP.';H);
c:=' ';
while (c<>'X') and (not EOF(INP)) do
begin
I:=0;
clear;
cursor(1,1);
REPEAT
   READLN(INP,S);
   i:=i+1;
   WRITELN(S);
UNTIL  (EOF(INP)) or (i>22);
c:=inkey;
uc(c);
end;
page(1);
end;

Function FUC(s: string): string;

{------------------------------------------------------------------}
{ Converts string s to upper case }
{------------------------------------------------------------------}

var us: string;
begin
us:=s; uc(us);
fuc:=us;
end;
```

```
Procedure ERRMSG(t: string);

{-------------------------------------------------------------------}
{ Display error message t and pause                                 }
{-------------------------------------------------------------------}

var s: string;
begin
clearmsg; writeln(chr(7),t);
write('Press ENTER to continue'); s:=inkey;
end;

Procedure DISPLT;

{------------------------------------------------------------------}
{ Display top term box }
{------------------------------------------------------------------}

begin
cleardata;
box(1,1,3,32,2); clearbox(2,2,1,30,2);
cursor(2,2); write(fuc(term));
if action='S' then
   begin
   box(1,74,3,7,1);
   cursor(1,76); write('MFN'); cursor(2,75); write(mfn:5);
   end;
savescr(1);
end;


{-------------STACK begin---------------------------------------}
Procedure STACK(FUNC: STRING);
VAR f:string;
begin
f:=func;
case f of
'ADD': begin
                  {   writeln('now adding');   test}
          stackptr:=stackptr + 1;
                      {writeln('ptr=',stackptr); test}
          mfnstack[stackptr]:=cmfn;
          tagstack[stackptr]:=1;
       end;
'DEL': begin
{          writeln('now deleting');      test;
```

```
                {          tst:=inkey;}    {test}
            mfnstack[stackptr]:=0;
            stackptr:=stackptr - 1;
         end;
end;
{writeln('top on stack: ','mfn=', mfnstack[stackptr],
             ' occ=', tagstack[stackptr],' stackptr=',stackptr);}
end;  {of stack}


{-----------GETNXT    begin-----------------------------------}
{------it takes the next linked descriptor (only COF is
taken into account)}
{---------------------- i.e. tag 06    -------------------------}
FUNCTION GETNXT: STRING;
VAR f,uf:string;
    maxocc: real;               { max no. of occurences for 06 tag) }
    rc1,rc2: real;

PROCEDURE FAIL;
BEGIN
rc2:=1;
io:=io+1;
{rebuild the cmfn}
cmfn:=mfnstack[stackptr];
rc1:=record(cmfn);
end;    {of FAIL within getnxt}


FUNCTION NXTOCC: STRING;
var f, uf: string;
BEGIN
if io < maxocc then dld[stackptr]:= 1
 else dld[stackptr]:= 0;  {see nextocc}
repeat    {for the worst case when CHEM is unconsistent}
                 { gets the first proper field 06 }
          {   if io < maxocc then dld[stackptr]:= 1
 else dld[stackptr]:= 0;}
          f:= field(fieldn(it,io)); {this will possibly
 be getnext value}
              {  writeln('nxtocc=',f,'it=',it,'io=',io); test }
          tagstack[stackptr]:=tagstack[stackptr] + 1;
          uf:=fuc(f);
          rc2:=find(uf);
          if rc2=0 then
             begin
              if nxtpost<0
                 then fail                 {RC2:=1 AND io:=io+1;}
```

```
                    else begin
                         cmfn:=posting('MFN'); {this is the case when
 CHEM is OK}
                         rc2:=record(cmfn);
                         if rc2=0 then  stack('ADD')
                             else fail;     {begin rc2:=1; io:=io+1 end}
                         end;
                 end
                   else fail {begin rc2:=1; io:=io+1;
 end i.e.unsuccessfull FIND}
until  (rc2=0) or (io > maxocc);
                 { writeln('rc2=',rc,'uf=',uf); test   }
     if rc2=0 then nxtocc:=uf else nxtocc:='';

END;      {NXTXOCC within getnxt}


{--------------------- UPSTCK ------------------------------}
FUNCTION UPSTCK: STRING;
var f,uf: string;
    rc1: real;
BEGIN
repeat
     STACK('DEL');
     cmfn:=MFNSTACK[stackptr];
     rc1:=record(cmfn);
     io:=tagstack[stackptr];
     maxocc:=NOCC(it);
     f:=nxtocc;
until (stackptr = 0) or f<> '';
if f <> '' then uf:= fuc(f) else uf:='';
UPSTCK:=uf;
END; {of upstck}


begin       {body of getnxt}
it:=6;
cmfn:=mfnstack[stackptr];
rc1:=record(cmfn);
       { --if rc1 <> 0 then begin - cant happen}
io:=tagstack[stackptr];
maxocc:=NOCC(it);
          {writeln('getnxt ' ,'cmfn=',cmfn,'rc1=',rc1,'io=',
 io,'maxocc=',maxocc
          {tst:=inkey;    {test}
  if maxocc >0 and io <=maxocc then
                          {there are some occurences of 06 }
                          {and not last has been taken till now}
```

```
        begin

        f:=nxtocc;
        if f <> '' then  getnxt:=f
                   else  getnxt:=UPSTCK;
        end
     else    getnxt:=upstck;

                  {writeln('getnext=',f); test}
end;    { of GETNXT }


{------------DSPL begin------------------------------------}
PROCEDURE DSPL(n:real;term: STRING);

{-------------------------------------------------------------}
{ -----------displays A line nl for the term  t                }
{------------xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx-------------}

var i,no, lev: real;
    t,line, undr, bar, cnt, angle :string;
begin
                      {writeln('next line');}
                      {tst:=inkey;}
undr:='_  ';
bar:='  3  ';
cnt:='  C';
angle:='  @';
lev:=stackptr; no:=n;
t:=term;

{writeln('level=',lev)}
line:=undr;
if lev>1 then
         begin
           for  i:=1 to lev-1 do
            begin
             if i < lev-1 then
                 if dld[i] = 0 then line:=line!'    '
                                else line:=line!bar
                          else
                 if dld[i] = 0 then line:=line!angle
                                else line:=line!cnt
             end; {of for}
           end; {of if lev}
line:= line!t;
cursor(no+4,1);
writeln(line);
```

```
end;   {of dspl}


{-------------BLDPAGE begin--------------------------}
FUNCTION BLDPAGE(term: STRING):REAL;

{-----------------------------------------------------}
{ BUILDS AND DISPLAYS A PAGE FOR A SELECTED term      }
{-------------xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}

var rc,fn: real;
    t:string;
    i:real;        {test}
begin
nl:=1;
t:=fuc(term);
rc:=find(t);
bldpage:=rc;

if rc=0 then
   if nxtpost<0
      then bldpage:=1     {no postings available for the term}
      else begin
           cmfn:=posting('MFN'); {mfn keeps top record ?}
            {tst}
           rc:=record(cmfn);
           bldpage:=rc;
           if rc=0 then
              begin
                stack('ADD');

                dmfn[nl]:=mfnstack[stackptr];
                dt[nl]:=1;     {displayed tag is 01}
                doc[nl]:=1;    {occurence 1}
                       {writeln('term=',t);
                       writeln('cmfn=',cmfn,'rc=',rc);
                       writeln('nl=',nl,'maxl=',maxl,
'stackptr=',stackptr);
                       tst:=inkey;  ==========================}
                while (nl<= maxl) and (stackptr > 0) do
                begin
                             {writeln('in loop');}
                 dspl(nl,t);
         {test   for i:=1 to stackptr do
                      begin
                        writeln('i=',i,'dld[i]=',dld[i]);
                      end;
                             tst:=inkey; test}
```

```
                    nl:=nl+1;
                         {  dt[nl]:=it; doc[nl]:=io; io:=io+1;}
                    t:=getnxt;
                    dmfn[nl]:=mfnstack[stackptr];
                    dt[nl]:=1;      {displayed tag is 01}
                    doc[nl]:=1;     {occurence 1}
                              {          writeln('in loop getnxt=',t);}
                   end;
                   if stackptr =0 then nl:=nl-1; {RHR}
                   {          writeln('now end of bldpage',
                   'nl=',nl,'stackptr=',stackptr); } {    test}
                 end;
              end; {in case when nxtpost is positive}

END; {of bldpage}
{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}


{-------------BLD-tree end-------------------------------------}


Procedure DISPLAY(t,o: real);

{------------------------------------------------------------}
{ Display term relations starting from tag[t], occurrence o }
{------------------------------------------------------------}

var rc,fn: real;

begin
nl:=0;
if t=1
   then begin
        displt;
        it:=1; io:=1;
        end
   else begin
        clearbox(5,1,15,80,0);
        it:=t; io:=o;
        end;
while (it<=maxt) and (nl<=maxl) do
  begin
  repeat
    fn:=fieldn(tag[it],io);
    if fn=0 then begin it:=it+1; io:=1; end;
  until (fn>0) or (it>maxt);
  if fn>0 then
     begin
```

```
      nl:=nl+1; dt[nl]:=it; doc[nl]:=io; io:=io+1;
      cursor(nl+4,1);
      write('  ',substr(rel,(it-1)*3+1,3),' ',field(fn));
      end;
   end;
end;


Function DECIDE(l: real): string;

{-------------------------------------------------------------}
{ Read action code (<CR>,B,F and P are handled here;
 othere codes returned   }
{-------------------------------------------------------------}

var s, fld: string;
    tag,occ,sc: real;
begin
cl:=1;
if nl>0 then
  begin
  clearmsg;
  writeln('Y Next  B[ack]   F[irst]     P[age]   S[elect]
 T[erm select]  Q[uery
  write  ('?[display query]  A[dd link]  E[dit]   D[elete]
  C[reate node]  X[exi
  repeat
  if cl<1 then cl:=1;
  if cl>nl then cl:=nl;
  cursor(cl+4,1);
  sc:=kbdkey(s); uc(s);
  if s=chr(13) then s:=' ';
  case s of
  ' ': if cl>=nl then cl:=1 else cl:=cl+1;
  'B': cl:=cl-1;
  'E': begin displt; clearmsg;
             rc:=record(mfn);
             write('Enter/edit description of the chemical');
             cursor(10,1); writeln('Explanation field: ');
             fld:=field(fieldn(2,1));  {take the field 02 -
 description }
             sc:=edit(fld,255,11,1,200,1,'_');
             {fld to be modified in the current record}
             sc:=flddel(fieldn(2,1));
             sc:=fldadd(2,1,fld);
             update;
             sc:=record(mfn);
             display(1,1); cl:=1;clearmsg; {repeat the lines}
```

```
              writeln('Y Next  B[ack]   F[irst]     P[age]
  S[elect]  T[erm sele
              write  ('?[display query]  A[dd link]  E[dit]
 D[elete]  C[reate n

         end;
  'F': begin display(1,1); cl:=1; end;
  'P': begin
       display(dt[nl],doc[nl]);
       cl:=1;
       end;
  end;
  until position('?ACDLMQSTX',s,1)>0;
  end;
decide:=s;
if s='S' then begin
            mfn:=dmfn[cl];
            tag:=dt[cl];              {?}
            occ:=doc[cl];             {?}
            sc:=record(mfn);
            term:=field(fieldn(dt[cl],doc[cl]));
    {                writeln('cl=',cl,  vvvvvvvvvvvvvvvvvv}
          end;
end;


Function FINDTRM1(term: string): real;

{------------------------------------------------------------}
{ Search and display selected term                           }
{      Return 0 if term exists (action contains a valid action code)}
{             1 if term does not exist (action is not set)       }
{------------------------------------------------------------}

var rc: real;
    t,k: string;
begin
stackptr:=0;   {rhrtest}
t:=fuc(term);
rc:=find(t);
findtrm1:=rc;
if rc=0 then
   if nxtpost<0
      then findtrm1:=1
      else begin
           mfn:=posting('MFN');
           rc:=record(mfn);
           findtrm1:=rc;
```

```
            if rc=0 then
               begin
               pgno:=1; maxpg:=1;    { replaces  display(1.1);    }
               displt;
               findtrml:=bldpage(t);
                                    {  writeln('now mfns ');  }
               action:=decide(0);
               end;
            end;
end;

{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}


{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
Function FLDUC(k: real): string;

{-----------------------------------------------------------------}
{ Returns k-th field of record converted to upper case      }
{-----------------------------------------------------------------}

var f: string;
begin
f:=field(k); uc(f);
flduc:=f;
end;


Function CHKREL(t: string): real;

{-----------------------------------------------------------------}
{ Check if a relation already exists                         }
{-----------------------------------------------------------------}

var i,n: real;
begin
n:=nfields; i:=1;
while (i<=n) and (flduc(i)<>t) do i:=i+1;
if i>n then chkrel:=0
       else chkrel:=i;
end;


Procedure UPDINVF;

{-----------------------------------------------------------------}
{ Update inverted file (screen is clear because FST is displayed)}
{-----------------------------------------------------------------}
```

```
begin
cleardata;
updif;
end;


Procedure CREATERM;

{-----------------------------------------------------------------}
{ Create new thesaurus term                                       }
{-----------------------------------------------------------------}

var tuc: string;
    rc,np: real;
begin
term:=''; clearmsg;
displt;
clearmsg; write('Enter new term');
rc:=edit(term,30,2,2,30,1,' ');
if term<>'' then
   begin
   tuc:=term; uc(tuc); rc:=find(tuc); np:=-1;
   if rc=0 then np:=nxtpost;
   if (rc=0) and (np>0)
      then errmsg('Term already exists')
      else begin
           mfn:=newrec;
           rc:=fldadd(tag[1],1,term);
           update; updinvf;
           action:='S';
           end;
   end
   else action:='T';
end;


Procedure ADDREL;

{-----------------------------------------------------------------}
{ Add new relation to a term                                      }
{-----------------------------------------------------------------}

var r,rt,rtu: string;
    rc,i,rtag: real;

Function ADDIT: real;
var tt,ir: string;
```

```
    n,k: real;
    relmfn: real;

Procedure RELADD;
var rc: real;
begin
n:=nocc(rtag); k:=1;
while (k<=n) and (flduc(fieldn(rtag,k))<rtu) do k:=k+1;
rc:=fldadd(rtag,k+1,rt); update;
end;

begin
if (find(rtu)<>0) and (substr(r,1,3)<>'TXT')
   then begin
        addit:=1;
        errmsg('Related term does not exist');
        end
   else
if (chkrel(rtu)<>0) and (substr(r,1,3)<>'TXT')
   then begin
        addit:=1;
        errmsg('Relation already exists');
        end
   else
begin
rtag:=tag[(rtag-1)/3+1];
reladd;
ir:=substr(invrel,(rtag-1)*3+1,3);
if ir<>'   ' then
   begin
   k:=nxtpost; relmfn:=posting('MFN');
   rtag:=tag[(position(rel,ir,1)-1)/3+1];
   rt:=field(fieldn(tag[1],1)); rtu:=rt; uc(rtu);
   k:=record(relmfn);
   reladd;
   end;
k:=record(mfn);
addit:=0;
end;
end;

begin
rc:=record(mfn);
box(18,10,3,5,1); box(18,14,3,52,1);
cursor(19,1); write('Relation');
r:=''; rt:='';
repeat
clearbox(19,15,1,50,1);
```

```
clearmsg; write('Enter link code: ');
for i:=2 to maxt do write(substr(rel,(i-1)*3+1,3),'   ');
cursor(23,1);
writeln('For HELP enter H in the relation field and press ENTER');
REPEAT
  clearbox(19,11,1,3,1);
  r:='';
  rc:=edit(r,3,19,11,3,1,' '); uc(r);
  IF r='H' then help('REL');
UNTIL r <> 'H';
rtag:=position(rel,r,1);
if rtag=0 then write(chr(7));
until (r='') or (rtag>0);
repeat
i:=0;
if rtag>0 then
   begin
   clearmsg;
   rc:=edit(rt,30,19,16,30,1,' '); rtu:=rt; uc(rtu);
   if rtu<>'' then i:=addit;
   end;
until i=0;
action:='S';
end;


Procedure DELREL;

{-------------------------------------------------------------}
{ Delete a relation                                          }
{-------------------------------------------------------------}

var rtag,rc,k,relmfn: real;
    rt,rtu,ir: string;
begin
rc:=record(mfn);
rtag:=fieldn(dt[cl],doc[cl]);
rt:=field(rtag); rtu:=rt; uc(rtu);
rc:=flddel(rtag);
update;
ir:=substr(invrel,(dt[cl]-1)*3+1,3);
if ir<>'   ' then
   begin
   rc:=find(rtu);
   if rc=0 then
      begin
      k:=nxtpost;
      if k>=0 then
```

```
         begin
         relmfn:=posting('MFN');
         rtag:=tag[(position(rel,ir,1)-1)/3+1];
         rt:=field(fieldn(tag[1],1));
         rtu:=rt; uc(rtu);
         rc:=record(relmfn);
         if rc=0 then
            begin
            k:=chkrel(rtu);
            if k>0 then
               begin
               rc:=flddel(k);
               update;
               end;
            end;
         end;
      end;
   end;
k:=record(mfn);
action:='S';
end;


Procedure DELTRM;

{---------------------------------------------------------------}
{ Delete a thesaurus term                                      }
{---------------------------------------------------------------}

begin
if nfields>1
   then begin
        errmsg('Cannot delete term with relations. Delete
all relations first.')
        action:='S';
        end
   else begin
        rc:=flddel(1);
        update; updinvf;
        action:='T';
        end;
end;


Procedure SHOWDICT;

{---------------------------------------------------------------}
{ List dictionary                                              }
```

```
{-----------------------------------------------------------------}

var i,ii,k,sc: real;
    tp: array[1..16] of real;
    ts: array[1..16] of real;
    pg,ft: string;

begin
ft:=term;
repeat
pg:=''; i:=1; sc:=find(ft);
repeat
tp[i]:=size(pg)+1; ts[i]:=size(ft);
pg:=pg¦ft;
ft:=nxtterm; i:=i+1;
until (i=17) or (ft='');
i:=i-1;
for k:=1 to i do
    begin cursor(k+4,5); writeln('_ ',substr(pg,tp[k],ts[k])); end;
k:=1;
repeat
ii:=k;
chattr(1,k+4,5,30); term:=substr(pg,tp[k],ts[k]);
sc:=kbdkey(action); uc(action);
if action=chr(13) then k:=k+1 else
if action='B' then if k>1 then k:=k-1;
chattr(0,ii+4,5,30);
until (position('CPSTX', action,1)>0) or (k>i);
page(1);
until (position('CSTX',action,1)>0) or (term='');
end;



{------------------------- Body of program CHEM ---------------}

begin

maxt:=9;                               { Number of defined relations }
rel:=   '        USEUF POFCOFEQUPRTPRS';    { Name of relations }
invrel:='        UF USECCFPOFEQUPRSPRT'; { Name of inverse relation }
for i:=1 to maxt do tag[i]:=i;             { Tag of relation }
stackptr:=0;
maxl:=15; q:='';
dbname:=dbn; { save currently selected data base }
if dbname<>'CHEM' then open('CHEM');
clear;
box(11,22,3,22,3);
```

```
cursor(12,24);
writeln('Trees of chemicals');
cursor(24,1);
writeln('Press any key');
action:=inkey;
if maxmfn=1 then action:='C' else action:='T';

repeat

case action of

'T': { Term selection }

     begin
     clearmsg;
     write('Select chemical name');
     term:=''; displt;
     cursor(2,2); readln(term);
     if term='' then action:='X' else
     if (substr(term,size(term),1)='$') or (findtrml(term)<>0)
        then action:='L';
     end;

'L': { List of thesaurus terms }

     begin
     uc(term);
     rc:=find(term);
     page(1);
     clearmsg;
     writeln('Y [Next]           B[previous]          P[age]
     S[elect]');
     write  ('C[reate node]    T[erm select]      X[exit]');
     savescr(1);
     showdict;
     if term='' then action:='L';
     end;

'S': { Display term relations }

     begin
     rc:=findtrml(term);
     WRITELN('RC=',RC);
     TST:= INKEY;
     if rc<>0 then action:='L';
     end;

'A': { Add a relation }
```

```
     addrel;

'C': { Create a new term }

     createrm;

'D': { Delete a term or a relation }

     if cl=1 then deltrm else delrel;

'Q': { Select term for searching }

     begin
     s:=field(fieldn(tag[dt[cl]],doc[cl]));
     if size(s)+size(q)+3>255
        then begin
             write('');
             action:='?';
             end
        else begin
             if q<>'' then q:=q¦' + ';
             q:=q¦s;
             action:=decide(cl+1);
             end;
     end;

'?': { Display current query }

     begin
     savescr(2);
     box(16,8,6,66,2); clearbox(17,9,4,64,1);
     cursor(17,9); lq:=size(q);
     if lq=0 then write('No chemicals currently selected for
 querying') else
        begin
        k:=1; kl:=17;
        repeat
        if lq>64 then i:=64 else i:=lq;
        writeln(substr(q,k,i));
        k:=k+i; lq:=lq-i;
        kl:=kl+1; cursor(kl,9);
        until lq=0;
        end;
     clearmsg; write('Press any key to continue');
     s:=inkey;
     page(2);
     action:=decide(cl+1);
     end;

  end;
  until action='X';

  if dbname<>'CHEM' then
     begin
     open(dbname);
     if size(q)>0 then
        begin
        clear;
        clearmsg; write('Edit search expresssion or press Enter');
        rc:=edit(q,254,2,1,254,0,' ');
        if size(q)>0 then rc:=search(q);
        end;
     end;
  option :=' ';  {rhr}
  end.


  Program chem(option: string) [menu];

  var dt: array[1..15] of real;     {displayed tags array}
      doc: array[1..15] of real;    {displayed occurences array}
      dmfn: array[1..15] of real;   {displayed mfns array}
      mfnstack: array[1..30] of real;   {mfn stack - only for COF}
      tagstack: array[1..30] of real;   {tags occurences stack -
   only for COF}
      stackptr: real;
      tag: array[1..10] of real;    { tag of relation }

      maxt: real;                { max no. of tags (upper bound of tag) }
      maxl: real;            { max no. of lines (upper bound of dt,doc) }
      rel,fullname,invrel: string;          { Relation indicators }

      c_page:real;                  {current page number}
      n_page:real;                  {last page number}

      it,io: real;                  { current tag/occ }
      nl: real;                     { lines on this page }
      cl: real;                     { current line }
      term: string;                 { current term }
      q: string;                    { query }
      dbname: string;               { current data base }
      mfn: real;                    { current mfn (in THES data base) }
      s,action,ft: string;
      i,k,kl,lq,rc: real;
```

```
PROCEDURE HELP(H:STRING);
{-----------------------------------------------------------}
{ Display help screen - H is an extension of HELP file      }
{-----------------------------------------------------------}

var s,c: string;
    i: real;
begin
savescr(1);
assign('INP','C:SISETROETDATAELP.'!H);
c:=' ';
while (c<>'X') and (not EOF(INP)) do
begin
I:=0;
clear;
cursor(1,1);
REPEAT
   READLN(INP,S);
   i:=i+1;
   WRITELN(S);
UNTIL  (EOF(INP)) or (i>23);
c:=inkey;
uc(c);
end;
page(1);
end;


Function FUC(s: string): string;

{-----------------------------------------------------------}
{ Converts string s to upper case }
{-----------------------------------------------------------}

var us: string;
begin
us:=s; uc(us);
fuc:=us;
end;


Procedure ERRMSG(t: string);

{-----------------------------------------------------------}
```

```
{ Display error message t and pause                                }
{------------------------------------------------------------------}

var s: string;
begin
clearmsg; writeln(chr(7),t);
write('Press ENTER to continue'); s:=inkey;
end;

Procedure DISPLT;

{------------------------------------------------------------------}
{ Display top term box }
{------------------------------------------------------------------}

begin
cleardata;
box(1,1,3,32,2); clearbox(2,2,1,30,2);
cursor(2,2); write(fuc(term));
if action='S' then
   begin
   box(1,74,3,7,1);
   cursor(1,76); write('MFN'); cursor(2,75); write(mfn:5);
   end;
savescr(1);
end;


Procedure DISPLAY(t,o: real);

{------------------------------------------------------------------}
{ Display term relations starting from tag[t], occurrence o }
{------------------------------------------------------------------}

var rc,fn: real;

begin
nl:=0;
if t=1
   then begin
        displt;
        it:=1; io:=1;
        end
   else begin
        clearbox(5,1,15,80,0);
        it:=t; io:=o;
        end;
```

```
while (it<=maxt) and (nl<=maxl) do
  begin
  repeat
    fn:=fieldn(tag[it],io);
    if fn=0 then begin it:=it+1; io:=1; end;
  until (fn>0) or (it>maxt);
  if fn>0 then
     begin
     nl:=nl+1; dt[nl]:=it; doc[nl]:=io; io:=io+1;
     cursor(nl+4,1);
     write('_ ',substr(rel,(it-1)*3+1,3),' ',field(fn));
     end;
  end;
end;
{==========================new display  =================}
Procedure DISPLY1(t,o: real);

{-------------------------------------------------------------}
{ Display term relations starting from tag[t], occurrence o }
{-------------------------------------------------------------}

var rc,fn,dl,nol,i: real;
    l,title:string;
begin

nl:=0;
if t=1
   then begin
        displt;
        it:=3; io:=1;
        end
   else begin
        clearbox(5,1,15,49,0);
        it:=t; io:=o;
        end;


getfmt('V02');
rc:=format(25);
nol:=lines;
rc:=nxtline(l);
if (nol > 0) and (l <> '' )
              then  begin
                box(4,52,nol+3,28,2);
                cursor(5,53);
                for i:=1 to nol do
                 begin
                   writeln(l);
```

```
                    rc:=nxtline(1);
                    cursor(5+i,53);
                  end;
               end;
while (it<=maxt) and (nl<=maxi) do
  begin
  repeat
    fn:=fieldn(tag[it],io);
    if fn=0 then begin it:=it+1; io:=1; end;
  until (fn>0) or (it>maxt);  {takes the first occurence
                                   of a field with the tag it}
  if fn>0 then
     begin
     nl:=nl+1; dt[nl]:=it; doc[nl]:=io;
     cursor(nl+4,1);
     if io=1 then title:=substr(fullname,(it-3)*11+1,11)!' _ '
             else  title:='              ';
     io:=io+1;

        {      write('_ ',substr(rel,(it-1)*3+1,3),' ',field(fn));
      write(title,field(fn));
     end;
  end;
end;

{==========================new display  =================}

Function DECIDE(l: real): string;

{--------------------------------------------------------------}
{ Read action code (<CR>,B,F and P are handled here; othere codes retu
ned   }
{--------------------------------------------------------------}

var s, fld: string;
    sc: real;
begin
cl:=1;
if nl>0 then
  begin
  clearmsg;
  writeln('Y Next  B[ack]   F[irst]     P[age]   S[elect]
  T[erm select]  Q[uery
  write  ('?[display query]  A[dd link]  E[dit]   D[elete]
  C[reate node]  X[exi
  repeat
  if cl<1 then cl:=1;
  if cl>nl then cl:=nl;
{  cursor(cl+4,13);}
```

```
  cursor(cl+4,1);
  sc:=kbdkey(s); uc(s);
  if s=chr(13) then s:=' ';
  case s of
  ' ': if cl>=nl then cl:=1 else cl:=cl+1;
  'B': cl:=cl-1;
  'E': begin displt; clearmsg;
             write('Enter/edit description of the chemical');
             cursor(10,1); writeln('Explanation field: ');
         fld:=field(fieldn(2,1));  {take the field 02 - description }
             sc:=edit(fld,255,11,1,200,1,'_');
             {fld to be modified in the current record}
             sc:=flddel(fieldn(2,1));
             sc:=fldadd(2,1,fld);
             update;
             sc:=record(mfn);
             display(1,1); cl:=1;clearmsg; {repeat the lines}
             writeln('Y Next  B[ack]   F[irst]    P[age]
 S[elect]  T[erm sele
             write  ('?[display query]  A[dd link]  E[dit]
 D[elete]  C[reate n

         end;
  'F': begin display(1,1); cl:=1; end;
  'P': begin
       display(dt[nl],doc[nl]);
       cl:=1;
       end;
  end;
  until position('?ACDLMQSTX',s,1)>0;
  end;
decide:=s;
if s='S' then term:=field(fieldn(tag[dt[cl]],doc[cl]));
end;


Function FINDTERM(term: string): real;

{-------------------------------------------------------------}
{ Search and display selected term                           }
{      Return 0 if term exists (action contains a valid action code)}
{              1 if term does not exist (action is not set)  }
{-------------------------------------------------------------}

var rc: real;
    t: string;
begin
t:=fuc(term);
```