תאריך הדפסה 4/10/99

```
unit chessu;

{Chess project}

{Main unit}

INTERFACE
uses crt;
type
    player  = (w,b,e);
    level   = (pp,mm);

    tabe    =   record
                sol         : char;
                playersol   : player;
                end;


     poin   = ^rec;
     poin2  = ^rec2;

      rec   =   record
                playerturn : player;
                value      : shortint;
                move_type  : char;
                themove    : string[4];
                son        : poin;
                next       : poin;
                end;

      rec2  =   record
                themove    : string[4];
                next       : poin2;
                end;

    tabel   =   array [1..8,'a'..'h'] of tabe;
    shortst =   string[4];

var
treeplay : poin;
a2,a1 : poin;
allmove  : poin;
p:pointer;
f:integer;
ta:tabel;
c:char;
down,p1:player;
f1:text;
s1:string[20];
s2:shortst;

procedure do_action_on_tabel ( move : shortst; var ta1 : tabel );
function  the_best_next_move (playr,down : player; ta1:tabel) : shortst;
procedure ins_bord ( down : player ; var ta1 : tabel ); {inst then borad }

IMPLEMENTATION

function get_sol_value(soln:char) : integer;
var
i:integer;
```

```
begin
case soln of
'p' : i:=1;
'r' : i:=2;
'n' : i:=3;
'b' : i:=4;
'q' : i:=5;
'k' : i:=6;
end; {case}
get_sol_value:=i;
end;

function get_real_nu (i:shortint) : shortint;
begin
case i of
       1 : i:=8;
       2 : i:=7;
       3 : i:=6;
       4 : i:=5;
       5 : i:=4;
       6 : i:=3;
       7 : i:=2;
       8 : i:=1;
       end;{case}
get_real_nu := i;
end;
function get_letter (i:shortint) : char;
var
c:char;
begin
case i of
1 : c:='a';
2 : c:='b';
3 : c:='c';
4 : c:='d';
5 : c:='e';
6 : c:='f';
7 : c:='g';
8 : c:='h';
end;{case}
get_letter:=c;
end;{get_number}
function get_number (c:char) : integer;
var
i:shortint;
begin
case c of
'a' : i:=1;
'b' : i:=2;
'c' : i:=3;
'd' : i:=4;
'e' : i:=5;
'f' : i:=6;
'g' : i:=7;
'h' : i:=8;
end;{case}
get_number:=i;
end;{get_number}
```

```
procedure what_is_it (mikom:shortst; ta1:tabel; var soln:char ;var  solp:player
);
var
i,x : integer;
c   : char;
begin

val(mikom[2],i,x);
i:=get_real_nu(i);
c:=mikom[1];

soln:=ta1[i,c].sol;
solp:=ta1[i,c].playersol;

end;{what_is_it}

procedure get_all_moves(mikom :shortst ; ta1 : tabel ; down : player ;var mo_p :
poin);
var
p_o_f,h_p,p:poin;
op1,d,mo1,op,i,y,x : integer;
y1,x1,soln1,soln,c:char;
solp1,solp:player;
st2,st1,st : shortst;
fin,ok :boolean;
mo:string[8];
begin

val(mikom[2],i,x);    { i = ‡…□„ □’  „˜…™„ □€ □□–□ }
i:=get_real_nu(i);    { ‡™ ‡…□ ‰”□ …‹˜’ □€ „˜‰†‡□„ „‰——□…”□ ‘□‹□ }
c:=mikom[1];          { c =  ˜…ˆ„ □…–‰□ □€ □□–□ }
soln := ta1[i,c].sol; { soln = C □’ I □–□™□ □’ □–‰□„ □‰‰‡„ □™ }
solp := ta1[i,c].playersol;  { solp = □‰‰‡„ □‰‰™ ‰□□ }
p_o_f:=nil;
h_p:=nil;

case soln of
'p' : begin  { †…†□ □…‹‰ □‰‰‡„ ˜™€ □…□…–□„ □‹ }

      for  op:=1 to 4 do  { „†…†□ □…‰–”…€ 4 ™‰ □‰‰‡□ }
      begin
      ok:=false;
      case op of
      1 : begin    { „ˆ□□ …€ „□’□□ ƒ‡€ ƒ’– „†…†□ 1 „˜–□ }
           y:=i;
            case solp of
              w : y:=y-1;
              b : y:=y+1;
            end; {case}
           if ((y<=8) and (y>=1)) then
           begin
           y:=get_real_nu(y);
           st:=c;
           str(y,st1);
           st:=st+st1;
           what_is_it(st,ta1,soln1,solp1);
           if solp1=e then ok:=true;
           end;
          end;{1}

      2 : begin  { „ˆ□□ …€ „□’□□ □€□™ ƒ– „†…†□ 2 „˜–□ }
```

```
     y:=i;
      case solp of
        w : y:=y-1;
        b : y:=y+1;
      end; { case}
     if ((y<=8) and (y>=1)) then
     begin
     x:=get_number(c);
     x:=x-1;
     if x>=1 then begin
                  st:=get_letter(x);
                  y:=get_real_nu(y);
                  str(y,st1);
                  st:=st+st1;
                  what_is_it(st,ta1,soln1,solp1);
                  if solp1<>solp then ok:=true;
                  if solp1=e then ok:=false;
                  end; {if}
     end;
     end; {2}
3 : begin  { „ˆ□□ …€ „□’□□ □‰□‰ ƒ– „†…†□ 2 „˜–□ }
     y:=i;
      case solp of
        w : y:=y-1;
        b : y:=y+1;
      end; { case}
     if ((y<=8) and (y>=1)) then
     begin
     x:=get_number(c);
     x:=x+1;
     if x<=8 then begin
                  st:=get_letter(x);
                  y:=get_real_nu(y);
                  str(y,st1);
                  st:=st+st1;
                  what_is_it(st,ta1,soln1,solp1);
                  if solp1<>solp then ok:=true;
                  if solp1=e then ok:=false;
                  end; {if}
end;
end; {3}

4 : begin
     y:=i;
     if (y=2) or (y=7) then
     begin



      case solp of
        w : y:=y-2;
        b : y:=y+2;
      end; {case}

     y:=get_real_nu(y);
     st:=c;
     str(y,st1);
     st:=st+st1;
     st2:=st;
     what_is_it(st,ta1,soln1,solp1);
     if (solp1=e) then ok:=true;
```

```
           y:=i;
            case solp of
              w : y:=y-1;
              b : y:=y+1;
            end; {case}
           y:=get_real_nu(y);
           st:=c;
           str(y,st1);
           st:=st+st1;
           what_is_it(st,ta1,soln1,solp1);
           if (solp1<>e) then ok:=false;
           st:=st2;




          end;




          end;{4}



      end; {case}

      if ok then begin
                     if p_o_f<>nil then
                     begin
                        new(p);
                        p^.next:=nil;
                        h_p^.next:=p;
                        h_p:=p;
                     end  else
                     begin
                        new(p_o_f);
                        p_o_f^.next:=nil;
                        h_p:=p_o_f;
                     end;
                     h_p^.themove:=st;
                     end;
          end; {for}
      end;  { □‰‰‡ □™ □…‰…˜™″€ }

'n' : begin  { †…†□ □…‹‰ '…'„ ˜™€ □…‰…˜™″€„ □‹ }
      for op:=1 to 8 do
      begin
      ok:=false;
      y:=i;
      x:=get_number(c);
      case op of
      1 : begin y:=y+2; x:=x+1; end;
      2 : begin y:=y+2; x:=x-1; end;
      3 : begin y:=y-2; x:=x+1; end;
      4 : begin y:=y-2; x:=x-1; end;
      5 : begin y:=y+1; x:=x+2; end;
      6 : begin y:=y-1; x:=x+2; end;
      7 : begin y:=y+1; x:=x-2; end;
      8 : begin y:=y-1; x:=x-2; end;
      end; {case}
      if ((y>=1) and (y<=8) and (x>=1) and (x<=8)) then
         begin
          st:=get_letter(x);
          y:=get_real_nu(y);
          str(y,st1);
          st:=st+st1;
          what_is_it(st,ta1,soln1,solp1);
          if solp1<>solp then ok:=true;
          if ok then begin
                     if p_o_f<>nil then
                     begin
                        new(p);
                        p^.next:=nil;
                        h_p^.next:=p;
                        h_p:=p;
                     end  else
                     begin
                        new(p_o_f);
                        p_o_f^.next:=nil;
                        h_p:=p_o_f;
                     end; {else}
                     h_p^.themove:=st;
                     end;  {if}
         end;{if}


         end;{for}
      end;{  `…`□ □…‰…˜™″€„ □<}

'b','r','q','k' :
       begin
       case soln of
       'b' : mo := '08080808';
       'r' : mo := '80808080';
       'q' : mo := '88888888';
       'k' : mo := '11111111';
       end; {case}
       for op:=1 to 8 do    { □…‰< „†‰€□ }
             begin
             fin:=false;
             ok:=false;
             val(mo[op],mo1,d);
             y:=i;
             y:=get_real_nu(y);
             x:=get_number(c);
             if mo1>0 then
             begin
             for op1:=1 to mo1 do          {ƒ…'–□ „□<}
                     begin
                       if not fin then
                         begin
                       case op of
                       1 : y:=y-1;
                       2 : begin y:=y-1; x:=x+1; end;
                       3 : x:=x+1;
                       4 : begin y:=y+1; x:=x+1; end;
                       5 : y:=y+1;
                       6 : begin y:=y+1; x:=x-1; end;
                       7 : x:=x-1;
                       8 : begin y:=y-1; x:=x-1; end;
                       end; {case}
            if ((y>=1) and (y<=8) and (x>=1) and (x<=8)) then
               begin
                 st:=get_letter(x);
{                 y:=get_real_nu(y);}
                 str(y,st1);
                 st:=st+st1;
                 what_is_it(st,ta1,soln1,solp1);
{                 writeln(st,' ',y,' ',mo1,' ',op1,' ',op);}
                 if solp1<>solp then begin ok:=true; fin:=true; end;
                 if solp1=solp  then begin ok:=false ; fin:=true; end;
                 if solp1=e then begin ok:=true; fin := false; end;

                     if ok then begin
                     if p_o_f<>nil then
                     begin
                        new(p);
                        p^.next:=nil;
                        h_p^.next:=p;
                        h_p:=p;
                     end  else
                     begin
                        new(p_o_f);
                        p_o_f^.next:=nil;
                        h_p:=p_o_f;
                     end; {else}
                     h_p^.themove:=st;
                      end;  {if}
                    end; {if}
                  end; {if}
                end; {for}
              end; {if}
             end; {for}

       end; {b,r,q,k}

end;  { case } {□‰□‰‰‡„ □‹ □…‰–"…€}
mo_p:=p_o_f;
end;
procedure get_all_sol_moves(ta1 : tabel ; down,player_m : player ;var mo_p :
poin);
var
p_o_f,h_p:poin;
p,p1   :poin;
i,y:integer;
soln,c:char;
solp:player;
s1,s,st:shortst;
begin
p:=nil;
p1:=nil;
p_o_f:=nil;
h_p:=nil;
for i:=1 to 8 do
              for c:='a' to 'h' do
                                begin
                                str(i,s);
                                st:='';
                                st:=c+s;
                                what_is_it(st,ta1,soln,solp);
                                if (solp<>e) and (solp=player_m) then
                                begin
                           {     p1:=nil;}
                                get_all_moves(st,tal,down,p1);
                                while p1<>nil do
                              begin
                                 if p_o_f<>nil then
                                 begin
                                  new(p);
                                  p^.next:=nil;
                                  h_p^.next:=p;
                                  h_p:=p;
                                 end  else
                                 begin
                                  new(p_o_f);
                                  p_o_f^.next:=nil;
                                  h_p:=p_o_f;
                                 end; {else}
                                 s1:=st+p1^.themove;
                                 h_p^.themove:=s1;
                                 h_p^.value:=0;
                                 p1:=p1^.next;
                              end; {while}

              end; {for}

                            end; {if}
mo_p:=p_o_f;
end;



procedure how_can_die ( tal : tabel ; down ,player_m : player ; var mo_p : poin
);
var
i,y : integer;
c :char;
p_o_f,h_p:poin;
p,p1,po  :poin;
soln:char;
solp:player;
s4,s1,s,st:shortst;
begin

p:=nil;
p1:=nil;
p_o_f:=nil;
h_p:=nil;

if player_m = w then get_all_sol_moves(tal,down,b,po)
                else get_all_sol_moves(tal,down,w,po);
{p1:=p;}
for i:=1 to 8 do
              for c:='a' to 'h' do
                                begin
                                str(i,s);
                                st:='';
                                st:=c+s;
                                what_is_it(st,tal,soln,solp);
                                if (solp<>e) and (solp=player_m) then
                                begin
                           {     p1:=nil;}
```

```
                                          p1:=po;
                                          while p1<>nil do
                                        begin
                                           s1:=p1^.themove[3]+p1^.themove[4];
                                           s4:=p1^.themove[1]+p1^.themove[2];
                                           if s1=st then
                                           begin
                                           h_p^.themove:=soln+s1;
                                           if p_o_f<>nil then
                                           begin
                                            new(p);
                                            p^.next:=nil;
                                            h_p^.next:=p;
                                            h_p:=p;
                                           end  else
                                           begin
                                            new(p_o_f);
                                            p_o_f^.next:=nil;
                                            h_p:=p_o_f;
                                           end; {else}
                                           h_p^.themove:=s1+s4;
                                           end; {if}
                                           p1^.son:=nil;
                                           p1:=p1^.next

                                        end; {while}
                end; {for}

                                      end; {if}
mo_p:=p_o_f;
end;
procedure do_action ( move : shortst; ta1 : tabel ; var ta2 : tabel );
var
c,soln:char;
solp:player;
s1,s2:shortst;
i,x:integer;
begin
for i:=1 to 8 do begin
                 for c:='a' to 'h' do begin
                                      ta2[i,c].sol:=' ';
                                      ta2[i,c].playersol:=e;
                                      ta2[i,c].sol:=ta1[i,c].sol;
                                      ta2[i,c].playersol:=ta1[i,c].playersol;
                                      end;
                 end;
s1:=move[1]+move[2];
s2:=move[3]+move[4];

what_is_it(s1,ta2,soln,solp);

val(s1[2],i,x);
c:=s1[1];
i:=get_real_nu(i);
ta2[i,c].sol:=' ';
ta2[i,c].playersol:=e;
val(s2[2],i,x);
c:=s2[1];
i:=get_real_nu(i);
ta2[i,c].sol:=soln;
ta2[i,c].playersol:=solp;
```

```
end;

procedure do_action_on_tabel ( move : shortst; var ta1 : tabel );
var
ta2:tabel;
c,soln:char;
solp:player;
s1,s2:shortst;
i,x:integer;
begin
for i:=1 to 8 do begin
                 for c:='a' to 'h' do begin
                                      ta2[i,c].sol:=' ';
                                      ta2[i,c].playersol:=e;
                                      ta2[i,c].sol:=ta1[i,c].sol;
                                      ta2[i,c].playersol:=ta1[i,c].playersol;
                                      end;
                 end;
s1:=move[1]+move[2];
s2:=move[3]+move[4];

what_is_it(s1,ta2,soln,solp);

val(s1[2],i,x);
c:=s1[1];
i:=get_real_nu(i);
ta2[i,c].sol:=' ';
ta2[i,c].playersol:=e;
val(s2[2],i,x);
c:=s2[1];
i:=get_real_nu(i);
ta2[i,c].sol:=soln;
ta2[i,c].playersol:=solp;

for i:=1 to 8 do begin
                 for c:='a' to 'h' do begin
                                      ta1[i,c].sol:=' ';
                                      ta1[i,c].playersol:=e;
                                      ta1[i,c].sol:=ta2[i,c].sol;
                                      ta1[i,c].playersol:=ta2[i,c].playersol;
                                      end;
                 end;

end;

procedure make_tree (turnp , down : player ; ta1 : tabel ; var mo_p : poin );
var
p1,p2,p3 :poin;
p4,p5,p6 :poin;
x,i,y : integer;
s1,s2,s3 : shortst;
ta2:tabel;
emp,turnp1:player;
c:char;
begin
get_all_sol_moves(ta1,down,turnp,p1);
p2:=p1;
while p1<>nil do begin
                 do_action(p1^.themove,ta1,ta2);
                 if turnp=w then turnp1:=b else turnp1:=w;
                 {for enmy moves}
```

```
                  get_all_sol_moves(ta2,down,turnp1,p3);
                  p1^.son:=p3;
                  p1:=p1^.next;
                  end; {while}
mo_p:=p2;

end;{make_tree}

procedure ins_bord ( down : player ; var ta1 : tabel ); {inst then borad }
var
i,y :integer;
c   :char;
begin
for i:=1 to 8 do
for c:='a' to 'h' do ta1[i,c].playersol:=w;
for c:='a' to 'h' do begin
                  ta1[7,c].sol := 'p';
                  ta1[2,c].sol := 'p';
                  end;{FOR}

ta1[1,'a'].sol := 'r'; ta1[8,'a'].sol := 'r';
ta1[1,'b'].sol := 'n'; ta1[8,'b'].sol := 'n';
ta1[1,'c'].sol := 'b'; ta1[8,'c'].sol := 'b';
ta1[1,'f'].sol := 'b'; ta1[8,'f'].sol := 'b';
ta1[1,'g'].sol := 'n'; ta1[8,'g'].sol := 'n';
ta1[1,'h'].sol := 'r'; ta1[8,'h'].sol := 'r';


case down of
w  : begin
     for i:=1 to 2 do
                  for c:='a' to 'h' do
                                    ta1[i,c].playersol := b;

     ta1[1,'d'].sol := 'k'; ta1[8,'d'].sol := 'k';
     ta1[1,'e'].sol := 'q'; ta1[8,'e'].sol := 'q';
     end;{w}



b  : begin
     for i:=7 to 8 do
                  for c:='a' to 'h' do
                                    ta1[i,c].playersol := b;

     ta1[1,'d'].sol := 'q'; ta1[8,'d'].sol := 'q';
     ta1[1,'e'].sol := 'k'; ta1[8,'e'].sol := 'k';
     end;{b}


end;{CASE}

for i:=3 to 6 do
              for c:='a' to 'h' do
                                begin
                                ta1[i,c].sol       :=' ';
                                ta1[i,c].playersol :=e;
                                end;{FOR}


end; {INS_BORD}
```

```
procedure the_best_attec ( play:player; ta1 : tabel ; var allmove : poin );
var
p,p1:poin;
val1,val : integer;
st,st1:shortst;
pl : player;
sl : char;
ok : boolean;
begin
p:=allmove;

while allmove<>nil do begin
                    st:=allmove^.themove[1]+allmove^.themove[2];
                    what_is_it(st,ta1,sl,pl);
                    val1:=get_sol_value(sl);

                    st:=allmove^.themove[3]+allmove^.themove[4];
                    what_is_it(st,ta1,sl,pl);
                    if (pl<>play) and (pl<>e) then
                    begin
                    val:=get_sol_value(sl);
                    if allmove^.son<>nil then
                            begin
                            ok:=false;

                            p1:=allmove;
                            allmove:=allmove^.son;
                            while allmove<>nil do begin

st1:=allmove^.themove[3]+allmove^.themove[4];
                                                  if ((st1=st) and (not(ok)))
 then begin

      ok:=true;

      val:=val-val1;

      end;
                                                  allmove:=allmove^.next;

                                                  end;

                            allmove:=p1;
                            allmove^.value:=val;
                            allmove^.move_type:='a';{atteck}

                            end;
                    end;
                    allmove:=allmove^.next;

                    end;


allmove:=p;
end; {procedure }

procedure my_sol_can_die (down,play : player; ta1 : tabel ; var p : poin);
var

p2,p1:poin;
```

```
st,st1:shortst;
sl:char;
pl:player;
begin
p2:=nil;
p1:=nil;
if play= w then make_tree(b,down,ta1,p2) else make_tree(w,down,ta1,p2);
p1:=p2;
while p2<>nil do begin
                 st:=p2^.themove[3]+p2^.themove[4];
                 what_is_it(st,ta1,sl,pl);
                 if pl=play then p2^.move_type:='d';
                 p2:=p2^.next;
                 end;

p2:=p1;
while p2<>nil do begin
st:=p2^.themove[3]+p2^.themove[4];
what_is_it(st,ta1,sl,pl);
if pl<>play then p2^.move_type:=' ';
   p2:=p2^.next;
                 end;
p2:=p1;


p:=p2;

end;

function the_best_next_move (playr,down : player; ta1:tabel) : shortst;
var
nnpp1,nnpp,tm2,tm1,tm,nm:poin;
x,i,y:integer;
c,c1:char;
st,st1,st2,st3:shortst;
wp,tp,np:player;
ta3,ta2:tabel;

begin
st:='';
tm:=nil;nm:=nil;
x:=0;
tp:=playr;
if tp = w then np:= b else np:=w;
make_tree(tp,down,ta1,tm);
make_tree(np,down,ta1,nm);
tm1:=tm;
while tm<>nil do begin

                 if tm^.son<>nil then begin
                                      st:=tm^.themove[3]+tm^.themove[4];
                                      what_is_it (st,ta1,c,wp);
                                      if wp=e then
                                      begin
                                      tm^.move_type:='n' {not good};
                                      tm2:=tm;
                                      tm:=tm^.son;
                                      x:=0;
                                      while tm<>nil do begin

st1:=tm^.themove[3]+tm^.themove[4];
```

```
                                                                if st=st1 then x:=1;
                                                                tm:=tm^.next;
                                                                end;
                                               tm:=tm2;
                                               end;
                          if x=1 then begin
                                      st:=tm^.themove[1]+tm^.themove[2];
                                      what_is_it (st,ta1,c,wp);
                                      x:=get_sol_value(c);
                                      tm^.value:=tm^.value-x;
                                      end
                                      else
                                      tm^.move_type:='s' {simple};


                          end;
                          tm:=tm^.next;
                          end;

         tm:=tm1;




         the_best_attec(np,ta1,nm);
         while nm<>nil do begin
                          if nm^.move_type='a' then if nm^.value>=x then begin
                                                                         st:=nm^.themove;
                                                                         x:=nm^.value;
                                                                         end;
                          nm:=nm^.next;
                          end;
         {st = the best next player attec}

         st1:=st[3]+st[4];
         what_is_it (st1,ta1,c,wp);
         x:=get_sol_value(c);
         {x= the value of the sol how wil die}

         if st<>'' then begin
         {if somebody can die}
                        tm1:=tm;
                        the_best_attec(tp,ta1,tm);
                        while tm<>nil do begin
                                         st1:=st[1]+st[2];
                                         st3:=tm^.themove[3]+tm^.themove[4];
                                         if tm^.move_type='a' then
                                                                if st1=st3 then
                                                                tm^.value:=tm^.value+x;
                                         if ((tm^.move_type ='s') and (tm^.value=0)) then
                                         begin
                                         st1:=st[3]+st[4];
                                         do_action(st,ta1,ta2);
                                         do_action(tm^.themove,ta2,ta3);
                                         get_all_sol_moves(ta3,down,tp,nnpp);

                                         nnpp1:=nnpp;
                                         while nnpp<>nil do
                                                       begin

         st3:=nnpp^.themove[3]+nnpp^.themove[4];
```

```
                                                    if st3=st1 then
                                                    tm^.value:=tm^.value+x;
                                                    nnpp:=nnpp^.next;
                                                    end;
                                       end;



                                       tm:=tm^.next;
                                       end;



                       end {if};


tm:=tm1;
tm1:=tm;
x:=-30;
while tm<>nil do begin
                 if tm^.value>=x then begin
                                      x:=tm^.value;
                                      st:=tm^.themove;
                                      end;
                 tm:=tm^.next;
                 end;

tm:=tm1;




{stam stam stam stam}
{tra la la ala la ala al a}
x:=0;
while tm<>nil do begin
                 if tm^.move_type='a' then if tm^.value>=x then begin
                                                                st:=tm^.themove;
                                                                x:=tm^.value;
                                                                end;
{                 writeln(tm^.themove,' ',tm^.value);
                 delay(100);}
                 tm:=tm^.next;
                 end;
{end of stam}

the_best_next_move:=st;
end; {procedure}


end.
```