LAMPIRAN

```
program parser (input,output);
{Penulis pertama kali : N. Wirth, E.T.H CH-8092  Zurich, 1.3.76}
{Program ini ditulis dan dimodifikasi dari program Pascal S karya
                R.E. Berry
                Departement of Computer Studies
                of Lancaster
                (ditulis dengan menggunakan UCSD Pascal yang
                digunakan pada Data General Nova, Apple, dan
                Western Digital Microengine machine}
{Modifikasi oleh penulis adalah dengan menggunakan Turbo Pascal
                versi 5.5 Pada komputer yang kompatibel dengan
                IBM}

uses crt,dos;
const
   nwk     = 26  ; alng    = 10  ;  llng    = 80;
   psal    = 15  ; emax    = 322 ;  emin    = -292;
   kmax    = 15  ; tmax    = 100 ;  bmax    = 20;
   amax    = 30  ;
   c2max   = 20  ; csmax   = 30  ;  cmax    = 800;
   lmax    = 7   ; smax    = 600 ;  ermax   = 59;
   omax    = 63  ; xmax    = 32767; nmax    = 32767;
   lineleng = 132;     linelimit = 200;   stacksize = 1450;
type
   symbol = (intcon,realcon,charcon,stringcon,notsy,plus,minus,
             times,idiv,imod,rdiv,andsy,orsy,eql,neq,gtr,geq,
             lss,leq,lparent,rparent,lbrack,rbrack,comma,semicolon,
             period,colon,becomes,constsy,typesy,varsy,funcsy,
             procsy,nul,arraysy,recordsy,programsy,ident,beginsy,
             ifsy,casesy,repeatsy,whilesy,forsy,endsy,elsesy,
             untilsy,ofsy,blank,dosy,tosy,downtosy,inputsy,
             outputsy,thensy,defsy);
   index  = -xmax..+xmax;
   alfa   =  packed array [1..alng] of char;
   obyect =  (konstant,vvariable,typei,prozedure,funktion);
   types  =  (notype,ints,reals,bools,chars,arrays,recor);
   symset =  set of symbol;    typset =  set of types;
   beta      = packed array[1..psal] of char;
   item   =  record
                typ : types; ref : index
             end;
   order  =  packed record
                f : -omax..+omax;  x : -lmax..+lmax;
                y : -nmax..+nmax
             end;
var
   ch     : char;   rnum   : real;   i,j    : integer;
   inum   : integer;   sleng  : integer;  nb :integer;
   cc     : integer;   lc     : integer;   ll     : integer;
   errpos, n: integer;
   t,a,b,sx,c1,c2 : integer;   iflag,oflag,skipflag,stackdump,
   prtables  : boolean;
   sy     : symbol;   errs   : set of 0..ermax;
   id     : alfa;   progname : alfa;   stantyps : typset;
   consbegsys ,typebegsys,blockbegsys,
   facbegsys,statbegsys : symset;
   ksy:array[1..nwk] of symbol;
```

```
    line       : array [1..llng] of char;
    objectcode : array [1..5] of char;
    typescode  : array [1..7] of char;
    key        : array [1..nwk] of alfa;
    sps        : array [char] of symbol;
    display    : array [0..lmax] of integer;
    tab        : array [0..tmax] of
                 packed record
                   name : alfa;          link   : index;
                   obj  : obyect;        typ    : types;
                   ref  : index;         normal : boolean;
                   lev  : 0..lmax;       adr    : integer;
                 end;
    atab       : array [1..amax] of
                 packed record
                   inxtyp,eltyp                : types;
                   elref,low,high,elsize,size : index
                 end;
    btab       : array [1..bmax] of
                 packed record
                   last,lastpar,psize,vsize : index
                 end;
    stab       : packed array [0..smax] of char;
    rconst     : array [1..c2max] of real;
    code       : array[0..cmax] of order;
    psin,psout,prr,prd : text;   inf,outf : string;
    penampungfile : array[1..20000] of byte;
procedure  errormsg;
var   k    : integer;
      msg : array [0..ermax] of beta;
begin
  msg[ 0] := 'tak terdefinisi'; msg[ 1] := 'definisi ganda ';
  msg[ 2] := 'kurang ident.  '; msg[ 3] := 'kurang "program';
  msg[ 4] := 'kurang )       '; msg[ 5] := 'kurang :       ';
  msg[ 6] := 'salah sintak   '; msg[ 7] := 'ident, var     ';
  msg[ 8] := 'kurang of      '; msg[ 9] := 'kurang (       ';
  msg[10] := 'ident, array   '; msg[11] :='kurang [       ';
  msg[12] := 'kurang ]       '; msg[13] := '..            ';
  msg[14] := ';              '; msg[15] := 'tipe fungsi    ';
  msg[16] := 'kurang =       '; msg[17] :='boolean         ';
  msg[18] := 'tipe konfar    '; msg[19] := 'kurang "tipe"  ';
  msg[20] := 'parameter prog.'; msg[21] := 'terlalu besar  ';
  msg[22] := 'kurang .       '; msg[23] :='tipe (pilih)    ';
  msg[24] := 'karakter       '; msg[25] := 'ident. konstan ';
  msg[26] := 'tipe indeks    '; msg[27] := 'bilangan indeks';
  msg[28] := 'tak ada array  '; msg[29] :='tipe identifier';
  msg[30] := 'tipe tak terdef'; msg[31] := 'tak ada record ';
  msg[32] := 'tipe boolean   '; msg[33] := 'tipe aritmetika';
  msg[34] := 'integer        '; msg[35] :='tipe-tipe       ';
  msg[36] := 'tipe parametere'; msg[37] := 'ident.variabel ';
  msg[38] := 'string         '; msg[39] := 'angka parameter';
  msg[40] := 'angka real     '; msg[41] :='tipe            ';
  msg[42] := 'tipe real      '; msg[43] := 'integer        ';
  msg[44] := 'konstan var.   '; msg[45] := 'var.,.prosedur ';
  msg[46] := 'tipe (:=)      '; msg[47] :='tipe (pilih)    ';
  msg[48] := 'tipe           '; msg[49] := 'berlebih       ';
  msg[50] := 'konstanta      '; msg[51] := ':=             ';
  msg[52] := 'maka           '; msg[53] :='sampai          ';
```

```
    msg[54] := 'lalu              '; msg[55] := 'ke / turun      ';
    msg[56] := 'kurang (          '; msg[57] := 'kurang )        ';
    msg[58] := 'faktor            '; msg[59] :='kurang definisi  ';
    writeln;writeln('_____________________ ');
    writeln('  no.kslh     pesan kesalahan ');
    k := 0;
    while errs <> [] do
    begin
       while not (k in errs) do k := k + 1;
       writeln(k:4,'     ',msg[k]);
       errs := errs - [k];
    end
end;

procedure endskip;
begin
    while errpos < cc do
    begin
      write('-'); errpos := errpos +1 ;
    end;
    skipflag := false
end;
procedure nextch;
begin
if cc = ll then
begin
    if eof(psin) then
    begin
       writeln;writeln(' program tak lengkap');errormsg;
       halt;
    end;
    if errpos <> 0 then
    begin  if skipflag then endskip; writeln; errpos := 0   end;
    nb:=nb+1;
    write('  ',nb:4,' '); ll := 0; cc := 0;
    while not eoln(psin) do
    begin
       ll := ll + 1;
       read(psin,ch);
       write(ch);
       line[ll] := ch
    end;
      ll:=ll+1;readln(psin,ch); writeln ;ch:=' ';line[ll] := ch
    end;
    cc:=cc+1;
    ch:=line[cc];
end;
procedure error(n : integer);
begin
    if errpos = 0 then write(' ****');
    if cc > errpos then
    begin
       write(' ': cc-errpos,'^', n : 2);
       errpos := cc + 3;errs:=errs+[n]
    end
end;
procedure fatal(n : integer);
var msg : array [1..7] of alfa;
```

```
begin
   write; errormsg;
   msg[ 1] := 'identifier'; msg[ 2] := 'prosedure ';
   msg[ 3] := 'real      '; msg[ 4] := 'array     ';
   msg[ 5] := 'level     '; msg[ 6] := 'kode      ';
   msg[ 7] := 'string    ';
   writeln(psout,'tabel kompiler untuk ',msg[n],' terlalu kecil');
   halt; end;
procedure insymbol;
label 1,2,3;
var i,j,k,e : integer;
  procedure readscale;
  var s,sign : integer;
  begin
     nextch; sign := 1 ; s := 0;
     if ch = '+' then nextch
     else if ch = '-' then
     begin
        nextch; sign := -1
     end;
     if not ((ch >= '0') and ( ch <= '9')) then error(40)  else
     repeat
        s := 10 * s + ord(ch) - ord('0');
        nextch
     until not (( ch >='0') and ( ch <= '9'));
     e := s * sign + e
  end;

  procedure adjustscale;
  var  s : integer; d,t :real;
  begin
     if k+e > emax then error(21) else
     if k+e < emin then
     rnum := 0 else
     begin
        s := abs(e); t := 1.0 ; d := 10.0;
        repeat
           while not odd(s) do
           begin  s := s div 2 ; d := sqr(d)   end;
           s := s-1 ; t := d * t
        until s = 0;
        if e >= 0  then rnum := rnum * t  else rnum := rnum/t
     end
  end;
  procedure options;
     procedure switch(var b : boolean);
     begin
        b := ch = '+';
        if not b
           then if not (ch = '-') then
           begin
              while ( ch <> '*') and ( ch <> ',') do nextch;
           end
           else nextch
        else nextch
     end;
begin
     repeat
```

```
            nextch; if ch <> '*' then
            begin
               if ch = 't' then  begin   nextch; switch(prtables)
               end else
               if ch = 's' then
               begin
                  nextch; switch(stackdump)
               end;
            end
         until ch <> ',';
      end;
   begin
   1 : while ch = ' ' do
       nextch;
       case ch of
       'a','b','c','d','e','f','g','h','i','j','k','l','m',
       'n','o','p','q','r','s','t','u','v','w','x','y','z' :
       begin
          k := 0 ; id := '          ';
            repeat
              if k < alng then
              begin
                k := k + 1 ; id[k] := ch
              end ;
              nextch;
            until not (ch in ['a'..'z']) or ( ch in ['0'..'9']);
            i := 1 ; j := nwk ;
            repeat
              k := ( i + j ) div 2; if id <= key[k] then j := k - 1;
              if id >= key[k] then i := k + 1
            until i > j ;
            if i - 1 > j then sy := ksy[k] else sy := ident
       end;
        '0','1', '2','3','4','5','6','7','8','9' :
              begin
                 k := 0 ; inum := 0 ; sy := intcon;
                 repeat
                    inum := inum + ord(ch)-ord('0'); k:=k+1; nextch
                 until not (( ch >= '0') and (ch <= '9'));
                 if (k > kmax) or (inum > nmax) then
                 begin
                    error(21) ; inum := 0 ; k := 0
                 end ;
                 if ch = '.' then
                 begin
                    nextch;
                    if ch = '.' then ch := ':' else
                    begin
                       sy := realcon ; rnum := inum ; e := 0;
                       while ( ch >= '0') and ( ch <= '9') do
                       begin
                          e := e - 1 ;
                          rnum := 10.0*rnum+(ord(ch)-ord('0'));
                          nextch
                       end;
                       if e = 0 then error(40);
                       if ch = 'e' then readscale;if e <> 0 then
                       adjustscale
```

```
                end
              end  else
              if ch = 'e' then
              begin
                sy := realcon ; rnum := inum ; e := 0 ;
                readscale;
                if e <> 0 then adjustscale
              end;
            end;
        ':' : begin
              nextch;
              if ch = '=' then
              begin
                sy := becomes; nextch
              end
              else sy := colon
            end;
        '<' : begin
              nextch;
              if ch = '=' then
              begin
                sy := leq ; nextch
              end
              else sy := lss
            end;
        '>' : begin
              nextch;
              if ch = '=' then
              begin
                sy := geq ; nextch
              end
              else sy := gtr
            end;
        '.' : begin
              nextch; if ch = '.' then begin sy := colon ; nextch
              end else sy := period
            end;
        '''' : begin
               k := 0;
    2 :        nextch; if ch = ''' then begin nextch;
               if ch <> '''' then goto 3 end;
               if sx + k = smax then fatal(7);
               stab[sx + k] := ch ; k := k + 1;
               if cc = l then
               begin
                 k := 0
               end else goto 2;
    3 :        if k = 1 then
               begin
                 sy := charcon; inum := ord(stab[sx])
               end
               else
               if k = 0 then
               begin
                 error(38); sy := charcon; inum := 0
               end
               else
               begin
```

```
                    sy := stringcon; inum := sx;
                    sleng := k ; sx := sx + k
                  end
               end;
          '(' : begin
                  nextch;
                  if ch <> '*' then sy := lparent
                  else
                  begin nextch; if ch = '$' then options;
                     repeat
                        while ch <> '*' do nextch;
                        nextch
                     until ch = ')';
                     nextch; goto 1
                  end
               end;
        '+','-','*','/',')','=',',','[',']',':','#','&','{','}' :
               begin
                  sy := sps[ch]; nextch
               end;
      '$','"','@','?','^','!' :
       begin error(24); nextch; goto 1 end ;
       end;
  end;
  procedure enter ( x0 : alfa ; x1 : obyect; x2 : types ;
                    x3 : integer);
  begin
     t := t + 1;
     with tab[t] do
     begin
        name := x0 ; link := t - 1 ; obj := x1;
        typ  := x2;  ref := 0 ; normal := true;
        lev := 0 ; adr := x3
     end
  end;
  procedure enterarray( tp : types ; l,h : integer);
  begin
     if l>h then error(27);
     if (abs(l) > xmax) or (abs(h) > xmax) then
     begin error(27) ; l := 0 ; h := 0; end;
     if a = amax then fatal(4) else
     begin
        a := a + 1;with atab[a] do
        begin
           inxtyp := tp ; low := l ; high := h
        end
     end
  end;
  procedure enterblock;
  begin
     if b = bmax then fatal(2)
     else
     begin
        b := b + 1; btab[b].last := 0 ; btab[b].lastpar := 0
     end
  end;
  procedure enterreal( x : real);
  begin
```

```
    if c2 = c2max - 1  then fatal(3) else
    begin
       rconst[c2 +1] := x; c1 := 1; while rconst[c1] <> x
       do c1 := c1 + 1;
       if  c1 > c2 then c2 := c1
    end
end;
procedure emit(fct : integer);
begin
    if lc = cmax then fatal(6);
    code[lc].f := fct ; lc := lc + 1
end;
procedure emit1( fct ,b : integer);
begin
    if lc = cmax then fatal(6);
    with code[lc] do
    begin
       f := fct ; y := b
    end;
    lc := lc + 1
end;
procedure emit2( fct ,a,b : integer);
begin
    if lc = cmax then fatal(6);
    with code[lc] do
    begin
       f := fct; x := a ; y := b
    end;
    lc := lc + 1
end;
procedure block (fsys : symset; isfun : boolean;
                 level : integer);
type  conrec = record case tp : types of
               ints,chars,bools : (i : integer);
               reals            : (r : real)
               end;
var dx : integer; prt : integer; prb : integer; x : integer;
 procedure skip(fsys : symset ; n : integer);
 begin
    error(n); skipflag := true;
    while not (sy in fsys) do insymbol;
    if skipflag then endskip
 end;
 procedure test(s1,s2 : symset; n : integer);
 begin
    if not (sy in s1) then skip(s1 + s2, n)
 end;
 procedure testsemicolon;
 begin
    if sy = semicolon then insymbol else
    begin
       error(14); if sy in [comma, colon] then insymbol end;
    test([ident] + blockbegsys, fsys, 0)
 end;
 procedure enter( id : alfa ; k : obyect);
 var  j,l : integer;
 begin
    if t = tmax then fatal(1) else
```

```
    begin
       tab[0].name := id;
       j := btab[display[level]].last; l := j;
       while tab[j].name <> id do j := tab[j].link;
       if j <> 0 then error(1) else
       begin
          t := t + 1;
          with tab[t] do
          begin
             name := id; link := l; obj := k; typ := notype;
             ref := 0; lev :=level; adr := 0
          end ;
          btab[display[level]].last := t
       end
    end
end;
function loc( id : alfa) : integer;
var  i,j : integer;
begin
    i := level; tab[0].name := id;
    repeat
       j := btab[display[i]].last;
       while tab[j].name <> id do j := tab[j].link; i := i - 1;
    until ( i < 0 ) or ( j <> 0 );
    if j = 0 then error(0);
    loc := j
end;
procedure entervariable;
begin
    if sy = ident then
    begin
       enter(id, vvariable); insymbol
    end
    else error(2)
end;
procedure constant(fsys : symset; var c: conrec);
var  x, sign : integer;
begin
   c.tp := notype;c.i := 0; test(consbegsys,fsys,50);
    if sy in consbegsys then
    begin
       if sy = charcon then
       begin
          c.tp := chars; c.i := inum; insymbol
       end
       else
       begin
          sign := 1;
          if sy in [plus,minus] then
          begin
             if sy = minus then sign := -1; insymbol
          end;
          if sy = ident then
          begin
             x := loc(id);
             if x <> 0 then if tab[x].obj <> konstant
             then error(25) else
             begin
```

```
                    c.tp := tab[x].typ;
                    if c.tp = reals then
                    c.r := sign * rconst[tab[x].adr]
                    else c.i := sign * tab[x].adr
                end;
                insymbol
            end
            else if sy = intcon then
            begin
               c.tp := ints; c.i := sign * inum; insymbol
            end
            else if sy = realcon then
            begin
               c.tp := reals; c.r := sign * rnum; insymbol
            end
            else skip(fsys,50)
         end;
         test(fsys, [],6)
      end
   end;
   procedure typ(fsys : symset ; var tp : types;
             var rf,sz : integer);
   var   eltp    : types;
         elrf, x : integer;
         elsz,offset, t0 ,t1 : integer;
   procedure arraytyp(var aref,arsz : integer);
   var eltp : types; low,high : conrec; elrf,elsz : integer;
   begin
      constant([colon,rbrack,rparent,ofsy] + fsys, low);
      if low.tp = reals then
      begin
         error(27); low.tp := ints; low.i := 0
      end;
      if sy = colon then insymbol else error(13);
      constant([rbrack,comma,rparent,ofsy],high);
      if high.tp <> low.tp then
      begin
         error(27); high.i := low.i
      end;
      enterarray(low.tp, low.i, high.i);
      aref := a;
      if sy = comma then
      begin
         insymbol; eltp := arrays; arraytyp(elrf,elsz)
      end else
      begin
         if sy = rbrack then insymbol else
         begin
            error(12); if sy = rparent then insymbol
         end;
         if sy = ofsy then insymbol else error(8);
         typ(fsys,eltp, elrf, elsz)
      end;
      with atab[aref] do
      begin
         arsz := ( high - low + 1 ) * elsz; size := arsz;
         eltyp := eltp; elref := elrf; elsize := elsz
      end;
```

```
  end;
begin
   tp := notype; rf := 0; sz := 0; test( typebegsys, fsys, 10);
   if sy in typebegsys then
   begin
      if sy = ident then
      begin
         x := loc(id); if x <> 0 then
         with tab[x] do
            if obj <> type1 then error(29) else
            begin
               tp := typ; rf := ref; sz := adr;
               if tp = notype then error(30)
            end;
         insymbol
      end else
      if sy = arraysy then
      begin
         insymbol; if sy = lbrack then insymbol else
         begin  error(11); if sy = lparent then insymbol end;
         tp := arrays; arraytyp(rf,sz)
      end else
      begin
         insymbol; enterblock; tp := recor ; rf := b;
         if level = lmax then fatal(5);
         level := level + 1; display[level] := b; offset := 0;
         while not (sy in fsys-[semicolon,comma,ident]+[endsy]) do
         begin
          if sy = ident then
          begin
            t0 := t; entervariable;
            while sy = comma do
            begin
              insymbol; entervariable
            end;
            if sy = colon then insymbol else error(5);
            t1 := t;
            typ(fsys+[semicolon,endsy,comma,ident],
            eltp,elrf,elsz);
            while t0 < t1 do
            begin
               t0 := t0 + 1;  with tab[t0] do
               begin
                 typ := eltp; ref := elrf; normal := true;
                 adr := offset; offset := offset + elsz
               end
               end
            end;
            if sy <> endsy then
            begin
               if sy = semicolon then insymbol else
               begin
                  error(14);
                  if sy = comma then insymbol
               end;
               test([ident,endsy,semicolon],fsys,6)
            end
         end;
```

```
            btab[rf].vsize := offset; sz := offset;
            btab[rf].psize := 0;insymbol; level := level - 1
         end;
         test(fsys,[],6)
      end
   end;
   procedure parameterlist;
   var   tp : types; valpar : boolean; rf, sz, x, t0 : integer;
   begin
      insymbol; tp := notype; rf := 0; sz := 0;
      test([ident,varsy], fsys + [rparent], 7);
      while sy in [ ident, varsy ] do
      begin
         if sy <> varsy then valpar := true else
         begin
            insymbol; valpar := false
         end;
         t0 := t; entervariable;
         while sy = comma do
         begin
            insymbol; entervariable;
         end;
         if sy = colon then
         begin
            insymbol;
            if sy <> ident then error(2) else
            begin
               x := loc(id); insymbol;
               if x <> 0 then with tab[x] do
                  if obj <> typei then error(29) else
                  begin
                     tp := typ; rf := ref;
                     if valpar then sz := adr else sz := 1
                  end;
              end ;
              test([semicolon,rparent],[comma,ident] + fsys, 14)
           end else error(5);
           while t0 < t do
           begin
              t0 := t0 + 1;
              with tab[t0] do
              begin
                 typ := tp; ref := rf; adr := dx; lev := level;
                 normal := valpar; dx := dx + sz
              end
           end;
           if sy <> rparent then
           begin
              if sy = semicolon then insymbol else
              begin
                 error(14);
                 if sy = comma then insymbol
              end;
              test([ident,varsy],[rparent] + fsys, 6)
           end
        end;
        if sy = rparent then
        begin
```

```
            insymbol; test([semicolon,colon],fsys, 6);
          end else error(4)
      end;
    procedure constdec;
    var c : conrec;
    begin
       insymbol; test([ident],blockbegsys, 2);
       while sy = ident do
       begin
          enter(id, konstant); insymbol;  if sy = eql
          then insymbol else
          begin  error(16); if sy = becomes then insymbol end;
          constant([semicolon,comma,ident] + fsys, c);
          tab[t].typ := c.tp; tab[t].ref := 0;
          if c.tp = reals then
          begin
             enterreal(c.r); tab[t].adr := c1
          end else tab[t].adr := c.i;
          testsemicolon
       end
    end;
    procedure typedeclaration;
    var tp : types; rf, sz, t1 : integer;
    begin
       insymbol; test([ident],blockbegsys, 2);
       if sy <> ident then begin error(25);insymbol;end;
       while sy = ident do
       begin
          enter(id, type1); t1 := t; insymbol;
          if sy = eql then insymbol else
          begin
             error(16);
             if sy = becomes then insymbol
          end;
          typ([semicolon,comma,ident] + fsys,tp,rf,sz);
          with tab[t1] do
          begin
             typ := tp; ref := rf; adr := sz
          end;
          testsemicolon
       end
    end;
    procedure variabledeclaration;
    var tp : types;  t0, t1, rf, sz : integer;
    begin
       insymbol;if sy <> ident then begin error(37); insymbol end;
       while sy = ident do
       begin
          t0 := t; entervariable;
          while sy = comma do.
          begin
             insymbol; entervariable;
          end;
          if sy = colon then insymbol else error(5);
          t1 := t; typ([semicolon,comma,ident] + fsys, tp, rf, sz);
          while t0 < t1 do
          begin
             t0 := t0 + 1;
```

```
            with tab[t0] do
            begin
               typ := tp; ref := rf; lev := level; adr := dx;
               normal := true; dx := dx + sz
            end
         end;
         testsemicolon
      end
   end;
   procedure procdeclaration;
   var  isfun : boolean;
   begin
      isfun := sy = funcsy; insymbol;
      if sy <> ident then
      begin
         error(2); id := '          ' end;
      if isfun then enter(id, funktion) else enter( id, prozedure);
      tab[t].normal := true; insymbol;
      block([semicolon] + fsys, isfun, level + 1);
      if sy = semicolon then insymbol else error(14);
      emit(32 + ord(isfun))
   end;
   procedure statement(fsys : symset);
   var  i : integer;  x : item;
      procedure expression(fsys : symset; var x : item); forward;
      procedure selector(fsys : symset; var v : item);
      var x : item;  a,j : integer;
      begin
         repeat
            if sy = period then
            begin
               insymbol; if sy <> ident then error(2) else
               begin
                  if v.typ <> recor then error(31) else
                  begin
                     j := btab[v.ref].last; tab[0].name := id;
                     while tab[j].name <> id do  j := tab[j].link;
                     if j = 0 then error(0); v.typ := tab[j].typ;
                     v.ref := tab[j].ref;
                     a := tab[j].adr;
                     if a <> 0 then emit1(9,a)
                  end ;
                  insymbol
               end
            end else
            begin
               if sy <> lbrack then error(11);
               repeat
                  insymbol; expression(fsys + [comma,rbrack], x);
                  if v.typ <> arrays then error(28) else
                  begin
                     a := v.ref;
                     if atab[a].inxtyp <> x.typ then error(26) else
                     if atab[a].elsize = 1 then
                     emit1(20,a) else emit1(21,a);
                     v.typ := atab[a].eltyp; v.ref := atab[a].elref
                  end
               until sy <> comma;
```

```
            if sy = rbrack then insymbol else
            begin
               error(12);
               if sy = rparent then insymbol
            end
         end
      until not (sy in [lbrack, lparent, period]);
      test(fsys,[],6)
   end;
   procedure call( fsys : symset; i : integer);
   var x : item;   lastp , cp, k : integer;
   begin
      emit1(18,i); lastp := btab[tab[i].ref].lastpar; cp := i;
      if sy = lparent then
      begin
         repeat
            insymbol; if cp >= lastp then error(39) else
            begin
               cp := cp + 1; if tab[cp].normal then
               begin
                  expression(fsys + [comma,colon,rparent], x);
                  if x.typ = tab[cp].typ then
                  begin
                     if x.ref<> tab[cp].ref then error(36)
                     else if x.typ = arrays
                     then emit1(22, atab[x.ref].size)
                     else if x.typ = recor
                     then emit1(22,btab[x.ref].vsize)
               end else
               if (x.typ = ints) and ( tab[cp].typ = reals) then
               emit1(26,0)
               else if x.typ <> notype then error(36);
            end else
            begin
               if sy <> ident then error(2) else
               begin
                  k := loc(id); insymbol;
                  if k <> 0 then
                  begin
                     if tab[k].obj <> vvariable then error(37);
                     x.typ := tab[k].typ; x.ref := tab[k].ref;
                     if tab[k].normal
                     then emit2(0, tab[k].lev, tab[k].adr)
                     else emit2(1, tab[k].lev, tab[k].adr);
                     if sy in [ lbrack, lparent, period] then
                     selector( fsys+[comma, colon, rparent], x);
                     if (x.typ <> tab[cp].typ) or
                     (x.ref <> tab[cp].ref) then error(36)
                  end  end  end
            end;
            test([comma, rparent], fsys, 6)
         until sy <> comma;
         if sy = rparent then insymbol else error(4)
      end;
      if cp <  lastp then error(39);
      emit1(19, btab[tab[i].ref].psize-1);
      if tab[i].lev < level then emit2(3, tab[i].lev, level)
   end;
```

```
function resulttype( a,b : types) : types;
begin
   if ( a > reals) or (b > reals) then
   begin
      error(33); resulttype := notype
   end
   else if ( a = notype) or (b = notype) then
   resulttype := notype
   else if a = ints then if b= ints then
   resulttype := ints  else
      begin
         resulttype := reals; emit1(26,1)
      end
   else
   begin
      resulttype := reals;
      if b = ints then emit1(26,0)
   end
end;
procedure expression ( fsys : symset; var x : item);
var   y : item; op : symbol;
   procedure simpleexpression(fsys : symset; var x : item);
   var y : item; op : symbol;
      procedure term( fsys : symset; var x : item);
      var   y : item; op : symbol;  ts : typset;
         procedure factor(fsys : symset ; var x : item);
         var  i,f : integer;
            procedure standfct(n : integer);
            var  ts : typset;
            begin
               if sy = lparent then insymbol else error(9) ;
               if n < 17 then
               begin
                  expression(fsys + [rparent], x);
                  case n of
                  0,2 : begin
                           ts := [ints,reals];
                           tab[i].typ := x.typ;
                           if x.typ = reals then n := n + 1
                        end;
                  4,5 : ts := [ints];
                  6   : ts := [ints, bools, chars];
                  7,8 : begin
                           ts := [ints, bools, chars];
                           tab[i].typ := x.typ
                        end;
                  9,10,11,12,13,14,15,16 :
                        begin
                           ts := [ints, reals];
                           if x.typ = ints then emit1(26,0)
                        end;
                  end;
                  if x.typ in ts then emit1(8,n) else
                  if x.typ <> notype then error(48);
               end else
               begin
                  if sy <> ident then error(2) else
                  if id <> 'input     ' then error(0) else
```

```
            insymbol;
            emit1(8,n);
         end;
         x.typ := tab[i].typ;
         if sy = rparent then insymbol else error(4)
      end;
   begin
      x.typ := notype; x.ref := 0;
      test(facbegsys, fsys, 58);
      while sy in facbegsys do
      begin
         if sy = ident then
         begin
            i := loc(id); insymbol;
            with tab[i] do
            case obj of
            konstant   : begin
                            x.typ := typ; x.ref := 0;
                            if x.typ = reals then
                            emit1(25, adr) else
                            emit1(24,adr)
                         end;
            vvariable  : begin
                            x.typ := typ; x.ref := ref;
                            if sy in [lbrack, lparent,
                            period] then
                            begin
                               if normal then f := 0
                               else f := 1;
                               emit2(f, lev, adr);
                               selector(fsys, x);
                               if x.typ in stantyps
                               then emit(34)
                            end else
                            begin
                               if x.typ in stantyps
                               then if normal then
                               f := 1 else f := 2
                               else if normal then
                               f := 0 else f := 1;
                               emit2(f, lev, adr)
                            end
                         end;
            typei, prozedure : error(44);
            funktion   : begin
                            x.typ := typ;
                            if lev <> 0 then
                            call(fsys,i) else
                            standfct(adr)
                         end  end
            end else if
            sy in [charcon,intcon,realcon] then
            begin
               if sy = realcon then
               begin
                  x.typ := reals;
                  enterreal(rnum); emit1(25,c1)
               end  else
```

```
                begin
                   if sy = charcon then
                   x.typ := chars
                   else x.typ := ints;
                   emit1(24,inum)
                end;
                x.ref := 0 ; insymbol
             end
             else if sy = lparent then
             begin
                insymbol;
                expression(fsys + [rparent], x);
                if sy = rparent then
                insymbol else error(4)
             end
             else if sy = notsy then
             begin
                insymbol; factor(fsys,x);
                if x.typ = bools then emit(35) else
                if x.typ <> notype then error(32)
             end;
             test(fsys, facbegsys, 6)
          end
       end;
begin
   factor(fsys + [times,rdiv,idiv,imod,andsy], x);
   while sy in [ times, rdiv, idiv, imod, andsy] do
   begin
      op := sy; insymbol;
      factor(fsys + [times,rdiv,idiv,imod,andsy],y);
      if op = times then
      begin
         x.typ := resulttype(x.typ, y.typ);
         case x.typ of
         notype : ;
         ints   : emit(57);
         reals : emit(60);
         end
      end else
      if op = rdiv then
      begin
         if x.typ = ints then
         begin
            emit1(26,1); x.typ := reals
         end;
         if y.typ = ints then
         begin
            emit1(26,0); y.typ := reals
         end;
         if (x.typ = reals) and (y.typ = reals)
         then emit(61)
         else
         begin
            if (x.typ <> notype) and (y.typ <> notype)
            then error(33);
            x.typ := notype
         end
      end else
```

```
            if op = andsy then
            begin
               if (x.typ = bools) and (y.typ = bools) then
               emit(56) else
               begin
                  if (x.typ <> notype)
                  and (y.typ <> notype) then error(32);
                  x.typ := notype
               end
            end else
            begin
               if (x.typ = ints) and (y.typ = ints) then
               if op = idiv then emit(58) else emit(59)
               else
               begin
                  if (x.typ <> notype) and (y.typ <> notype)
                  then error(34);
                  x.typ := notype
               end
            end
         end;
      begin  {simple expression}
      if sy in [ plus, minus] then
      begin
         op := sy; insymbol; term(fsys + [plus, minus], x);
         if x.typ > reals then error(33) else
         if op = minus then emit(36)
      end else term(fsys+[plus, minus, orsy], x);
      while sy in [ plus, minus, orsy] do
      begin
         op := sy; insymbol; term(fsys + [plus,minus,orsy],y);
         if op = orsy then
         begin
            if (x.typ = bools) and (y.typ = bools) then
            emit(51) else
            begin
               if (x.typ <> notype) and ( y.typ <> notype) then
               error(32); x.typ := notype
            end
         end else
         begin
            x.typ := resulttype(x.typ, y.typ);
            case x.typ of
            notype : ;
            ints   : if op = plus then emit(52) else emit(53);
            reals : if op = plus then emit(54) else emit(55)
            end
         end
      end
   end;
   begin {expression}
   simpleexpression(fsys + [eql, neq, lss, leq, gtr, geq], x);
   if sy in  [eql, neq, lss, leq, gtr, geq] then
   begin
      op := sy; insymbol; simpleexpression(fsys, y);
      if (x.typ in [notype, ints, bools, chars])
      and ( x.typ = y.typ) then
```

```
        case op of
        eql : emit(45);      neq : emit(46);
        lss : emit(47);      leq : emit(48);
        gtr : emit(49);      geq : emit(50);
      end else
      begin
        if x.typ = ints then
        begin
          x.typ := reals; emit1(26,1)
        end else
        if y.typ = ints then
        begin
          y.typ := reals; emit1(26,0)
        end;
        if (x.typ = reals) and (y.typ = reals) then
        case op of
         eql : emit(39);       neq : emit(40);
         lss : emit(41);       leq : emit(42);
         gtr : emit(43);       geq : emit(44);
        end  else error(35)
      end;
      x.typ := bools
      end
  end;
procedure assignment (lv, ad : integer);
var x,y : item;
    f   : integer;
begin
   x.typ := tab[i].typ; x.ref := tab[i].ref;
   if tab[i].normal then f := 0 else f := 1;
   emit2(f, lv, ad);
   if sy in [lbrack, lparent, period] then
   selector([becomes, eql] + fsys, x);
   if sy = becomes then insymbol else
   begin
      error(51);
      if sy = eql then insymbol
   end;
   expression(fsys, y);
   if x.typ = y.typ then
   if x.typ in stantyps then emit(38) else
   if x.ref <> y.ref then error(46)
   else if x.typ = arrays then emit1(23,atab[x.ref].size) else
   emit1(23, btab[x.ref].vsize)
   else if (x.typ = reals) and (y.typ = ints) then
   begin
   emit1(26,0); emit(38)
   end else
   if (x.typ <> notype) and (y.typ <> notype) then error(46)
end;
procedure compoundstatement;
begin
   insymbol; statement([semicolon, endsy] + fsys);
   while sy in [semicolon] + statbegsys do
   begin
      if sy = semicolon then insymbol else error(14);
      statement([semicolon, endsy] + fsys)
   end;
```

```
    if sy = endsy then insymbol else error(57)
  end;
  procedure ifstatement;
  var  x : item;  lc1, lc2 : integer;
  begin
    insymbol; expression(fsys+[thensy,dosy], x);
    if not (x.typ in [bools, notype]) then error(17);
    lc1 := lc; emit(11);
    if sy = thensy then insymbol else
    begin
      error(52);
      if sy = dosy then insymbol
    end;
    statement(fsys +[elsesy]);
    if sy = elsesy then
    begin
      insymbol; lc2 := lc; emit(10); code[lc1].y := lc;
      statement(fsys); code[lc2].y := lc
    end
    else code[lc1].y := lc
  end;
  procedure casestatement;
  var    x : item ; i, j, k, lc1  : integer;
         casetab : array [1..csmax] of
           packed record
           val, lc : index
           end;
         exittab : array [1..csmax] of integer;
    procedure caselabel;
    var    lab : conrec;  k : integer;
    begin
      constant(fsys + [comma,colon], lab);
      if lab.tp <> x.typ then error(47)
      else if i = csmax then fatal(6) else
      begin
        i := i + 1; k := 0; casetab[i].val := lab.i;
        casetab[i].lc := lc;
        repeat
          k := k + 1
        until casetab[k].val = lab.i;
        if k < i then error(1);
      end
    end;
    procedure onecase;
    begin
      if sy in consbegsys then
      begin
        caselabel;
        while sy = comma do
        begin
          insymbol; caselabel
        end;
        if sy = colon then insymbol else error(5);
        statement([semicolon,endsy] + fsys); j := j + 1;
        exittab[j] := lc; emit(10)
      end
    end;
  begin
```

```
      insymbol; i := 0 ; j := 0;
      expression(fsys + [ofsy,comma,colon], x);
      if not (x.typ in [ints,bools, chars, notype]) then error(23);
      lc1 := lc; emit(12);
      if sy = ofsy then insymbol else error(8);
      onecase;
      while sy = semicolon do
      begin
         insymbol; onecase
      end;
      code[lc].y := lc;
      for k := 1 to i do begin emit1(13,casetab[k].val);
      emit1(13,casetab[k].lc)  end;
      emit1(10,0); for k:=i to j do
      code[exittab[k]].y := lc;
      if sy = endsy then insymbol else error(57)
   end;
   procedure repeatstatement;
   var   x : item;   lc1 : integer;
   begin
      lc1 := lc; insymbol; statement([semicolon,untilsy] + fsys);
      while sy in [semicolon] + statbegsys do
      begin
         if sy = semicolon then insymbol else error(14);
         statement([semicolon, untilsy] + fsys)
      end;
      if sy = untilsy then
      begin
         insymbol; expression(fsys, x);
         if not (x.typ in [bools, notype]) then error(17);
         emit1(11,lc1)
      end else error(53)
   end;
   procedure whilestatement;
   var  x : item;  lc1, lc2 : integer;
   begin
      insymbol; lc1 := lc; expression(fsys + [dosy], x);
      if not (x.typ in [bools, notype]) then error(17);
      lc2 := lc; emit(11);
      if sy = dosy then insymbol else error(54);
      statement(fsys); emit1(10,lc1); code[lc2].y := lc
   end;
   procedure forstatement;
   var   cvt : types;  x : item; i,f,lc1,lc2 : integer;
   begin
      insymbol;
      if sy = ident then
      begin
         i := loc(id); insymbol;
         if i = 0 then cvt := ints
         else if tab[i].obj = vvariable then
         begin
            cvt := tab[i].typ;
            if not tab[i].normal then error(37) else
            emit2(0, tab[i].lev, tab[i].adr);
            if not (cvt in [notype, ints, bools, chars]) then
            error(18)
         end else
```

```
            begin
               error(37); cvt := ints
            end
         end else skip([becomes, tosy, downtosy, dosy] + fsys, 2);
         if sy = becomes then
         begin
            insymbol; expression([tosy, downtosy, dosy] + fsys, x);
            if x.typ <> cvt then error(19);
         end else skip([tosy, downtosy, dosy] + fsys, 51);
         f := 14;
         if sy in [tosy, downtosy] then
         begin
            if sy = downtosy then f := 16; insymbol;
            expression([dosy] + fsys, x);
            if x.typ <> cvt then error(19)
         end else skip([dosy]+ fsys, 55);
         lc1 := lc; emit(f);
         if sy = dosy then insymbol else error(54);
         lc2 := lc; statement(fsys); emit1(f + 1, lc2);
         code[lc1].y := lc
      end;
      procedure standproc(n : integer);
      var   i, f : integer;  x,y : item;
      begin
         case n of
          1,2  : begin
                    if not iflag then
                    begin
                       error(20); iflag := true
                    end;
                    if sy = lparent then
                    begin
                       repeat
                          insymbol;
                          if sy <> ident then error(2) else
                          begin
                             i := loc(id); insymbol;
                             if i <> 0 then
                             if tab[i].obj <> vvariable then
                             error(37) else
                             begin
                                x.typ := tab[i].typ;
                                x.ref := tab[i].ref;
                                if tab[i].normal then f := 0
                                else f := 1;
                                emit2(f, tab[i].lev, tab[i].adr);
                                if sy in [lbrack, lparent, period]
                                then selector(fsys+[comma,rparent], x);
                                if x.typ in [ints, reals, chars, notype]
                                then emit1(27,ord(x.typ)) else error(41)
                             end
                          end;
                          test([comma,rparent], fsys, 6);
                       until sy <> comma;
                       if sy = rparent then insymbol else error(4)
                    end;
                    if n = 2 then emit(62)
                 end;
```

```
      3,4  : begin
                if sy = lparent then
                begin
                   repeat
                      insymbol;
                      if sy = stringcon then
                      begin
                         emit1(24,sleng); emit1(28,inum); insymbol
                      end else
                      begin
                         expression(fsys+[comma,colon,rparent],x);
                         if not (x.typ in stantyps) then error(41);
                         if sy = colon then
                         begin
                          insymbol;
                          expression(fsys+[comma,colon,rparent],y);
                          if y.typ <> ints then error(43);
                          if sy = colon then
                          begin
                            if x.typ <> reals then error(42);
                            insymbol;
                            expression(fsys+[comma, rparent], y);
                            if y.typ <> ints then error(43);
                            emit(37)
                            end else emit1(30, ord(x.typ))
                          end else emit1(29, ord(x.typ))
                      end
                   until sy <> comma;
                   if sy = rparent then insymbol else error(4)
                end;
                if n = 4 then emit(63)
             end;
        end
  end;
  begin
     if sy in statbegsys + [ident] then
     case sy of
     ident      : begin
                  i := loc(id); insymbol; if i <> 0 then
                  case tab[i].obj of
                  konstant,
                  typei     : error(45);
                  vvariable : assignment(tab[i].lev, tab[i].adr);
                  prozedure : if tab[i].lev <> 0  then
                                call(fsys,i) else
                                standproc(tab[i].adr);
                  funktion  : if tab[i].ref = display[level] then
                                assignment(tab[i].lev +1, 0) else
                                error(45)
                  end
                  end;
     beginsy    : compoundstatement ;
     ifsy       : ifstatement;
     casesy     : casestatement;
     whilesy    : whilestatement;
     repeatsy   : repeatstatement;
     forsy      : forstatement;
     end;
```

```
    test(fsys, [], 14)
  end;
  begin     {block}
    dx := 5 ; prt := t;
    if level > lmax then fatal(5);
    test([lparent,colon,semicolon],fsys,14);
    enterblock; prb := b; display[level] := b;
    tab[prt].typ := notype; tab[prt].ref := prb;
    if (sy = lparent) and (level > 1) then parameterlist;
    btab[prb].lastpar := t; btab[prb].psize := dx;
    if isfun then  if sy = colon then
    begin
      insymbol;
      if sy = ident then
      begin
        x := loc(id); insymbol; if x <> 0 then
        if tab[x].obj <> typel then error(29) else
        if tab[x].typ in stantyps then
        tab[prt].typ := tab[x].typ else error(15)
      end else skip([semicolon] + fsys, 2)
    end else error(5);
    if sy = semicolon then insymbol else error(14);
    if sy <> defsy then begin error(59);insymbol end
    else insymbol;
    repeat
      if sy = constsy then constdec;
      if sy = typesy then typedeclaration;
      if sy = varsy then variabledeclaration;
      btab[prb].vsize := dx;
      while sy in [procsy,funcsy] do procdeclaration;
    until sy in statbegsys;
    test([beginsy], blockbegsys + statbegsys, 56);
    tab[prt].adr := lc;
    insymbol;
    statement([semicolon,endsy] + fsys);
    while sy in [semicolon] + statbegsys do
    begin
      if sy = semicolon then insymbol else error(14);
      statement([semicolon,endsy] + fsys)
    end;
    if sy = endsy then insymbol else error(57);
    test(fsys + [period],[], 6)
  end;
  procedure setup;
  begin
    key[ 1] := 'atau      ';  key[ 2] := 'dan       ';
    key[ 3] := 'dari      ';  key[ 4] := 'definisi  ';
    key[ 5] := 'div       ';  key[ 6] := 'fungsi    ';
    key[ 7] := 'jika      ';  key[ 8] := 'kala      ';
    key[ 9] := 'ke        ';  key[10] := 'konstanta ';
    key[11] := 'lain      ';  key[12] := 'lalu      ';
    key[13] := 'maka      ';  key[14] := 'mod       ';
    key[15] := 'pilih     ';  key[16] := 'program   ';
    key[17] := 'rekord    ';  key[18] := 'sampai    ';
    key[19] := 'subprogram';  key[20] := 'susunan   ';
    key[21] := 'tidak     ';  key[22] := 'tipe      ';
    key[23] := 'turun     ';  key[24] := 'ulang     ';
    key[25] := 'untuk     ';  key[26] := 'variabel  ';
```

```
    ksy[ 1] := orsy         ; ksy[ 2] := andsy;
    ksy[ 3] := ofsy         ; ksy[ 4] := defsy;
    ksy[ 5] := idiv         ; ksy[ 6] := funcsy;
    ksy[ 7] := ifsy         ; ksy[ 8] := whilesy;
    ksy[ 9] := tosy         ; ksy[10] := constsy;
    ksy[11] := elsesy       ; ksy[12] := dosy;
    ksy[13] := thensy       ; ksy[14] := imod;
    ksy[15] := casesy       ; ksy[16] := programsy;
    ksy[17] := recordsy     ; ksy[18] := untilsy;
    ksy[19] := procsy;      ; ksy[20] := arraysy;
    ksy[21] := notsy        ; ksy[22] := typesy;
    ksy[23] := downtosy     ; ksy[24] := repeatsy;
    ksy[25] := forsy ;      ; ksy[26] := varsy
    sps['+'] := plus;         sps['-'] := minus;
    sps['*'] := times;        sps['/'] := rdiv;
    sps['('] := lparent;      sps[')'] := rparent;
    sps['='] := eql   ;       sps[','] := comma;
    sps['['] := lbrack;       sps[']'] := rbrack;
    sps['"'] := neq;          sps['&'] := andsy;
    sps[';'] := semicolon;    sps[' '] := blank;
    sps['#'] := neq;      ;   sps['{'] := beginsy;
    sps['}'] := endsy;
end;
procedure enterids;
begin
   enter('            ',vvariable, notype, 0);
   enter('salah       ',konstant, bools, 0);
   enter('benar       ',konstant, bools, 1);
   enter('real        ',typel, reals, 1);
   enter('karakter    ',typel, chars, 1);
   enter('boolean     ',typel, bools, 1);
   enter('integer     ',typel, ints, 1);
   enter('abs         ',funktion, reals, 0);
   enter('akar        ',funktion, reals, 2);
   enter('ganjil      ',funktion, bools, 4);
   enter('krt         ',funktion, chars, 5);
   enter('ord         ',funktion, ints, 6);
   enter('sblm        ',funktion, chars, 7);
   enter('ssdh        ',funktion, chars, 8);
   enter('round       ',funktion, ints, 9);
   enter('trunc       ',funktion, ints, 10);
   enter('sin         ',funktion, reals, 11);
   enter('cos         ',funktion, reals, 12);
   enter('exp         ',funktion, reals, 13);
   enter('ln          ',funktion, reals, 14);
   enter('sqrt        ',funktion, reals, 15);
   enter('arctan      ',funktion, reals, 16);
   enter('eof         ',funktion, bools, 17);
   enter('eoln        ',funktion, bools, 18);
   enter('baca        ',prozedure, notype, 1);
   enter('bacabrs     ',prozedure, notype, 2);
   enter('tulis       ',prozedure, notype, 3);
   enter('tulisbrs    ',prozedure, notype, 4);
   enter('            ',prozedure, notype, 0);
end;

begin
clrscr;writeln;writeln;writeln;
```

```
setup; clrscr;
consbegsys :=[plus,minus,intcon,realcon,charcon,ident];
typebegsys:=[ident,arraysy,recordsy];
blockbegsys:=[constsy,typesy,varsy,procsy,funcsy,beginsy];
facbegsys:=[intcon,realcon,charcon,ident,lparent,notsy];
statbegsys:=[beginsy,ifsy,whilesy,repeatsy,forsy,casesy];
stantyps:=[notype,ints,reals,bools,chars];
lc:=0; ll:=0; cc:=0; ch:=' '; errpos:=0; errs:=[];
writeln; writeln; nb:= 0;
write('file input asal ?'); readln(inf);assign(psin,inf);
settextbuf(psin,penampungfile);reset(psin);
writeln;
insymbol; t:=-1; a:=0; b:=1; sx:=0; c2:=0; display[0]:=1;
iflag:=true; oflag:=true;
skipflag:=false; prtables:=false; stackdump:=false;
if sy<>programsy then error(3) else
begin
   insymbol;
   if sy<>ident then error(2) else
   begin
      progname := id; insymbol;
      if not oflag then error(20)
   end
end;
enterids;with btab[1] do begin last:=t; lastpar:=1;
psize:=0; vsize:= 0 end;
block(blockbegsys+statbegsys,false,1); if sy<>period then
error(22); emit(31);
if errs=[] then writeln; writeln;writeln;
writeln('kompilasi dengan error'); writeln; errormsg;
writeln; close(psin);
end.
```