Dataset Tahkuse_2004_2014

Technical Supplement

H. Tammet
University of Tartu, Estonia

The present document includes algorithms of two programs:

- Tahkuse_make_years (pages 1–7)
- Textractor11 (pages 8–10).

The program *Tahkuse_make_years* was compiled for conversion of the 203 MB data source *Tahkuse_2002_2014_master_10a.xls* into 11 yearly files presented in the dataset *Tahkuse_2004_2014*. The program was launched in the common folder with the source and the conversion was automatically accomplished during about 15 minutes.

The program f *Textractor11* is a tool for the user of the dataset. The exe-file must be launched in the same folder, where are located all yearly data files. The guide of *Textractor11* is given in the document *Tahkuse_data_2004_2014.doc.* The program is self-explaining and issues a text file including the guide when launched first time.

Both programs are pure number crunchers written in Pascal without graphical procedures. Thus the code may be compiled in different implementations of Pascal and easily translated into different languages including C, Fortran and Python. Original exe-files were compiled as console applications in the environment of Delphi 6 or Lazarus 6. The program code includes few specific introductory lines, which should be changed when using a different compiler.

program Tahkuse_make_years;

```
{$APPTYPE CONSOLE}
uses sysutils, math, dateutils;
// processing time on i5 is about 15 minutes

const nhours = 96432;  // = 11 * 365 + 3 = 4018 days in 2004..2014
        skip = 105121; // 2 years = 1 + 2*365*24*6 rows
       n_dia = 10;
       n_mob = 9;
npercentiles = 7;
  percentile : array [1..npercentiles] of single =
               (0, 1, 10, 50, 90, 99, 100); // 0 ==> min, 100 ==> max
    dataname = 'Tahkuse_2002_2014_master_10a.xls';
      header : array [1..114] of string = ('year', 'month', 'day', 'hour',
 'weekday', 'p:mb', 'T:C', 'RH%', 'v:m/s', 'vmax:m/s', 'v:deg', 'Tbox:C',
 'Tattic:C', 'Tchimney:C', 'rad:W/m2', 'NO2:ug/m3', 'flow:lpm', 'Rn:Bq/m3',
 'Rn-s:Bq/m3', 'd@4.2', 'd@7.5', 'd@13', 'd@24', 'd@42', 'd@75', 'd@133',
 'd@237', 'd@422', 'd@750', 'd@1334', 'd@2371', 'd@4217', 'd@7500', 'd:defect',
 'Z+2.77', 'Z+2.22', 'Z+1.77', 'Z+1.41', 'Z+1.13', 'Z+0.88', 'Z+0.70', 'Z+0.56',
 'Z+0.44', 'Z+0.35', 'Z+0.28', 'Z+0.213', 'Z+0.107', 'Z+0.051', 'Z+0.024',
 'Z+0.015', 'Z+0.0068', 'Z+0.0031', 'Z+0.0014', 'Z+0.0007', 'sum+0.3', 'Z-2.77',
 'Z-2.22', 'Z-1.77', 'Z-1.41', 'Z-1.13', 'Z-0.88', 'Z-0.70', 'Z-0.56', 'Z-0.44',
 'Z-0.35', 'Z-0.28', 'Z-0.213', 'Z-0.107', 'Z-0.051', 'Z-0.024', 'Z-0.015',
 'Z-0.0068', 'Z-0.0031', 'Z-0.0014', 'Z-0.0007', 'sum-0.3', 'Z1noise',
 'Z2noise','n+', 'Z+', 'fS/m+', 'n-', 'Z-', 'fS/m-', 'n&', 'Z&','fS/m&','q-eas',
 'q-ion1', 'q-ion2', 'q-eas-st', 'q-ion-st', 'q-ion-sym', 'q-low', 'a', 'b',
 'p', 'd0', 'dev%', 'c+', 'g+', 'k+', 'Z0+', 'dev+%', 'c-', 'g-', 'k-', 'Z0-',
 'dev-%', 'c&', 'g&', 'k&', 'Z0&', 'dev&%');
         dec : array [1..114] of integer = (0,0,0,0,0,1,1,1,1,1,0,1,1,1,0,
         2,2,1,1,0,0,0,0,0,0,0,1,1,2,3,3,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,2,
         0,3,2,0,3,2,0,0,0,0,0,0,0,0,3,3,1,1,2,3,3,3,1,2,3,3,3,1,2,3,3,3,1);
type d_vector = array [1..n_dia] of single;
type z_vector = array [1..n_mob] of single;

// Global variables
var y : array [1..nhours] of single;
 data : array [1..nhours, 1..114] of single; // hourly averages
```

```
procedure readmaster; // read the Tahkuse master data at first into 6 ten-minute
                      // table x, then calculate hourly averages and noise
                      // parameters for quality indexing and fill table "data"
                      // NB: –99 (exatly!) is the code of missing value
   var i, j, h,
     n, n1, n2 : integer;
   s, ss, u, v : single;
             x : array [1..6, 1..78] of single; // [decaminute, #var]
             f : text;
   begin
   assign (f, dataname); reset (f);
   for h := 1 to skip do readln (f); // skip 2 years = 1 + 2*365*24*6 rows
   for h := 1 to nhours do begin     // read and preprocess data hour by hour
      for j := 1 to 6 do for i := 1 to 78 do read (f, x [j, i]); // decaminutes
      for i := 1 to 4 do data [h, i] := x [1, i]; // year month day hour
      data [h, 5] := dayoftheweek (encodedate (round (x [1, 1]),
                           round (x [1, 2]), round (x [1, 3])));
                           // ISO 8601 compliant Monday = 1, Sunday = 7
      for i := 6 to 78 do begin // measured variables from p:mb to Z2noise
         // estimate hourly average
         n1 := 0; n2 := 0; s := 0;
         for j := 1 to 3 do if x [j, i] <> -99 then begin // first 3 decaminutes
            n1 := n1 + 1; s := s + x [j, i];
         end;
         for j := 4 to 6 do if x [j, i] <> -99 then begin // last 3 decaminutes
            n2 := n2 + 1; s := s + x [j, i];
         end;
         if (n1 = 0) or (n2 = 0) or (n1 + n2 < 3) then data [h, i] := -99
         else data [h, i] := s / (n1 + n2); // hourly average
      end;
      // data [h, 88..93] are noise measures for estimating of q-parameters
      data [h, 88] := data [h, 34]; // eas defect
      data [h, 89] := data [h, 77]; // Z1noise
      data [h, 90] := data [h, 78]; // Z2noise

      s := 0; ss := 0; n := 0; // begin eas dispersion
      for j := 1 to 6 do for i := 22 to 27 do begin
         v := x [j, i]; // 13..237 nm through the hour
         if v <> -99 then begin n := n + 1; s := s + v; ss := ss + v * v end;
      end;
      if n < 36 then data [h, 91] := -99
      else data [h, 91] := ss / n - sqr (s / n); // done eas dispersion

      s := 0; ss := 0; n := 0; // begin ion dispersion
      for j := 1 to 6 do for i := 48 to 51 do begin
         v := x [j, i]; // mobility +0.005..0.015 through the hour
         if v <> -99 then begin n := n + 1; s := s + v; ss := ss + v * v end;
         v := x [j, i + 21]; // mobility -0.005..0.015 through the hour
         if v <> -99 then begin n := n + 1; s := s + v; ss := ss + v * v end;
      end;
      if n < 48 then data [h, 92] := -99
      else data [h, 92] := ss / n - sqr (s / n); // done ion dispersion

      ss := 0; n := 0; // begin ion asymmetry
      for j := 1 to 6 do begin
         v := x [j, 55]; u := x [j, 76];
         if (v <> -99) and (u <> -99) then begin
         n := n + 1; ss := ss + sqr (v - u) end;
      end;
      if n <> 6 then data [h, 93] := -99 else data [h, 93] := ss; // done
   end; // of h := 1 to nhours
   close (f);
   end; // of readmaster : data 1..78 & unsorted 91..93 is collected
```

```
procedure sortq (q : integer);
   // uses global nhours & data and replaces data [h, q] values with rankindices
   // index = 0 corresponds to missing mmeasurement,
   // index = 1..1000 is the per-mille-position in the sorted list of quality,
   // where 1000 corresponds to the highest quality
   var h, j, k,
        n, gap : integer;
             t : single;
          done : boolean;
             x : array [1..nhours] of single;
             y : array [1..nhours] of integer;

   begin
   n := 0; // number of determined values
   for h := 1 to nhours do x [h] := data [h, q];
   for h := 1 to nhours do if x [h] = -99 then x [h] := 1e9 else n := n + 1;
   for h := 1 to nhours do y [h] := h;  // number in initial table
   gap := nhours;
   repeat
      gap := gap div 3 + 1;
      repeat
         done := true;
         for h := 1 + gap to nhours do begin
            j := h - gap;
            if x [j] > x [h] then begin
               t := x [h];
               x [h] := x [j];
               x [j] := t;
               k := y [h];
               y [h] := y [j];
               y [j] := k;
               done := false;
            end;
         end;
      until done;
   until gap = 1;  // h = number in the sorted table, y [h] in the original
   for h := 1 to nhours do data [y [h], q] := 1000 - trunc (1000 * h / n);
   for h := 1 to nhours do if data [h, q] < 0 then data [h, q] := 0;
   end; // of sortq


function d_parameters (dia, nlgd : d_vector; // diameters and dN/dlgd
                       var a, b, p, d0, dev : single) : integer; //  = n27
         // calculates parameters of two-power model for size distribution
   var    lnnlgd, aa : d_vector; // aa = -ln (approx / a)
     i, m, j, k, n27 : integer;
       h, y, yy, yyy,
           u, v, lna,
            ppb, pmb, // p + b and p - b
         bb, pp, dd : single;
   begin
   for i := 1 to n_dia do lnnlgd [i] := ln (nlgd [i]);
   ppb := 4.5; pmb := 3.5; d0 := 180; // initial approximation
   y := 1e33; h := 0.2; n27 := 0;
   repeat
      repeat yyy := y;
         for i := -1 to 1 do for j := -1 to 1 do for k := -1 to 1 do begin
            bb := pmb + i * h;
            if bb > 10 then bb := 10;
            pp := ppb + j * h;
            if pp < 0.1 then pp := 0.1;
            if pp > 20 then pp := 20;
            dd := d0 + k * h * 50;
            if dd < 10 then dd := 10;
            if dd > 10000 then dd := 10000;
            u := 0; v := 0;
            p := (pp + bb) / 2; b := (pp - bb) / 2;
            for m := 1 to n_dia do begin
               aa [m] := ln ((power (dia [m] / dd, -b) +
                              power (dia [m] / dd, p)));
               u := u + lnnlgd [m] + aa [m];
            end;
```

```
            lna := u / n_dia; a := exp (lna);
            for m := 1 to n_dia do v := v + sqr (lna - aa [m] - lnnlgd [m]);
            yy := sqrt (v / n_dia) / ln (10);
            if yy < y then begin
               pmb := bb; ppb := pp; d0 := dd; y := yy
            end;
         end; // of 27 tests
         n27 := n27 + 1;
      until (y = yyy) or (n27 > 1999); // dead end
      h := 0.5 * h;
   until (h < 1e-5) or (n27 > 1999);
   dev := y; // standard deviation of lg (nlgd)
   result := n27;
   end;

function z_parameters (mob, nlgz : z_vector; // mobilities and dN/dlgZ
                       var a, b, p, z0, dev : single) : integer; // n27
         // calculates parameters of two-power model for mobility distribution

   var    lnf, xx : z_vector; // aa = -ln (approx / a)
    h, y, yy, yyy,
    u, bb, pp, zz : single;
      i, m , j, k,
          i0, n27 : integer;
   begin
   u := 1e9; i0 := 1;
   for i := 1 to n_mob do begin
      lnf [i] := ln (nlgz [i]);
      if lnf [i] < u then begin
         u := lnf [i];
         i0 := i;
      end;
   end;
   if (i0 = 1) or (i0 = n_mob) then begin
      a := -99; b := -99; p := -99; z0 := -99; dev := -99; result := 0;
   exit end;
   // initial approx of b, p, z0
   b := 2 * (lnf [1] - lnf [i0]) / (ln (mob [i0] / mob [1]));
   p := 2 * (lnf [n_mob] - lnf [i0]) / (ln (mob [n_mob] / mob [i0]));
   z0 := mob [i0];
   y := 1e33; h := 0.1; n27 := 0;
   repeat
      repeat yyy := y;
         for i := -1 to 1 do for j := -1 to 1 do for k := -1 to 1 do begin
            // scan test points bb, pp, zz around b, p, z0
            bb := b + i * h; if bb > 10 then bb := 10;
            pp := p + j * h; if pp < 0.1 then pp := 0.1;
                             if pp > 21 then pp := 21;
            zz := z0 + k * h / 20;
            if zz < mob [1] then zz := mob [1];
            if zz > mob [n_mob] then zz := mob [n_mob];
            // estimate xx = two-power-root at a = 1, bb, pp, and zz
            for m := 1 to n_mob do xx [m] :=
               sqrt (power (mob [m] / zz, -bb) +  power (mob [m] / zz, pp));
            // estimate a and justify xx to approx of nlgz
            u := 0; for m := 1 to n_mob do u := u + ln (nlgz [m] / xx [m]);
            a := exp (u / n_mob);
            for m := 1 to n_mob do xx [m] := a * xx [m];
            // estimate error as root-mean-square of log10 (approx / measured)
            u := 0; for m := 1 to n_mob do u := u + sqr (log10 (xx [m] / nlgz [m]));
            yy := sqrt (u / n_mob);
            if yy < y then begin // accept best aproximation
               b := bb; p := pp; z0 := zz; y := yy
            end;
         end; // of 27 ijk tests
         n27 := n27 + 1;
      until (y = yyy) or (n27 > 999); // dead end or difficult problem
      h := 0.6 * h; // 0.6 is empiric optimum
   until (h < 1e-6) or (n27 > 999);
   dev := y; // root-mean-square of log10 (approx / measured)
   result := n27;
   end;
```

```
procedure process_eas (h : integer); // h = hour index in data
       // aerosol measurements carried out bu means of EAS
var j, bad : integer;
   a, b, p,
   d0, dev : single;
     d, fd : d_vector;

   begin
   for j := 1 to n_dia do  d [j] := power (10, (j + 2.5) / 4);
   for j := 1 to n_dia do fd [j] := data [h, 20 + j];
   bad := 0;
   for j := 1 to n_dia do
      if fd [j] < 0.01 then begin fd [j] := 0.01; bad := bad + 1; end;
   if bad < 2 then d_parameters (d, fd, a, b, p, d0, dev)
   else begin a := -99; b := -99; p := -99; d0 := -99; dev := -99; end;
   data [h, 95] := a;
   data [h, 96] := b;
   data [h, 97] := p;
   data [h, 98] := d0;
   if dev <> -99 then dev := 100 * dev; // %
   data [h, 99] := dev;
   end;

procedure process_ions (h, polarity : integer); // h = hour index in data
// measurements of ion distribution for 1 = positive, 2 = negative or 3 = both
const
   mob : z_vector =
   (0.00141, 0.00307, 0.00667, 0.0148, 0.0507, 0.107, 0.213, 0.556, 0.696);
   pos : array [1..9] of integer = (53, 52, 51, 50, 48, 47, 46, 42, 41);
   Zfr : array [1..8] of single =
                 (0.556, 0.696, 0.88, 1.127, 1.413, 1.773, 2.22, 2.775);
   dlgZ = 0.098;
   jcl : array [1..8] of integer = (42, 41, 40, 39, 38, 37, 36, 35);

var i, k, bad : integer;
      a, b, p,
      z0, dev,
     sumn, fS,
         Zave : single;
           fd : z_vector;
         nfr : array [1..8] of single;
         row : array [1..78] of single;
   begin
   for i := 1 to 76 do row [i] := data [h, i]; // positive ions
   if polarity = 2 then for i := 35 to 55 do row [i] := row [i + 21]; // negative
   if polarity = 3 then for i := 35 to 55 do row [i] := row [i] + row [i + 21];
   // calculate two-power approximation
   for i := 1 to n_mob do fd [i] := row [pos [i]];
   bad := 0;
   for i := 1 to n_mob do if fd [i] < 2 then begin
      bad := bad + 1;
      fd [i] := 1;
   end;
   if bad < 2 then Z_parameters (mob, fd, a, b, p, z0, dev)
   else begin a := -99; b := -99; p := -99; z0 := -99; dev := -99; end;
   // correct directly measured small ions at Z0 = 0.3
   if data [h, 55] < 5 then data [h, 55] := -99;
   if data [h, 76] < 5 then data [h, 76] := -99;
   // process small ions
   for i := 1 to 8 do nfr [i] := row [jcl [i]] * dlgZ;
   bad := 0;
   for i := 1 to 8 do if nfr [i] < 0.5 then begin
      bad := bad + 1;
      nfr [i] := 0.5;
   end;
   if bad < 2 then begin
      sumn := 0; Zave := 0;
      for i := 1 to 8 do sumn := sumn + nfr [i];
      for i := 1 to 8 do Zave := Zave + Zfr [i] * nfr [i] / sumn;
      fS := 1.6e-2 * Zave * sumn;
   end
```

```
   else begin
      sumn := -99; Zave := -99; fS := -99; // missing code
   end;
   k := 75 + 3 * polarity;
   data [h, k + 1] := sumn;
   data [h, k + 2] := Zave;
   data [h, k + 3] := fs;
   k := 94 + 5 * polarity;
   data [h, k + 1] := a;
   data [h, k + 2] := b;
   data [h, k + 3] := p;
   data [h, k + 4] := z0;
   if dev <> -99 then dev := 100 * dev; // deviation will be presented in percents
   data [h, k + 5] := dev;
   end;

Procedure sort_y (n : integer); // sort first n elements of global array y
   var i, j, gap : integer;
               t : single;
            done : boolean;
   begin
   gap := n;
   repeat
      gap := gap div 3 + 1;
      repeat
         done := true;
         for i := 1 + gap to n do begin
            j := i - gap;
            if y [j] > y [i] then begin
               t := y [i];
               y [i] := y [j];
               y [j] := t;
               done := false;
            end;
         end;
      until done;
   until gap = 1;
   end;

function cleanprint (x : single; h : integer) : string;
   var s : string;
   begin
   str (x:0:h, s);
   if h > 0 then begin // cut tail zeroes
      while s [length (s)] = '0' do s := copy (s, 1, length (s) - 1);
      if s [length (s)] = '.' then  s := copy (s, 1, length (s) - 1);
   end;
   result := s;
   end;

procedure basic_stat;
   var h, i, k, n : integer;
    sum1, sum2, r : single;
          n_known : array [1..114] of integer;
             stat : array [1..114, 1..npercentiles + 2] of single;
                g : text;
   begin
   for i := 1 to 114 do begin
      for h := 1 to nhours do y [h] := data [h, i];
      for h := 1 to nhours do if y [h] = -99 then y [h] := 1e34;
      if (i > 87) and (i < 95) then // q-parameters
      for h := 1 to nhours do if y [h] = 0 then y [h] := 1e34;
      sort_y (nhours);
      n := nhours;
      while (n > 0) and (y [n] > 1e33) do n := n - 1;
      n_known [i] := n;
      if n < 3 then for k := 1 to npercentiles + 2 do stat [i, k] := 0
      else begin
         for k := 1 to npercentiles do stat [i, k] :=
            y [1 + round (percentile [k] * (n - 1) / 100)];
         sum1 := 0; sum2 := 0;
         for h := 1 to n do begin
```

```
            sum1 := sum1 + y [h];
            sum2 := sum2 + y [h] * y [h];
         end;
         r := sum1 / n;
         stat [i, npercentiles + 1] := r;
         stat [i, npercentiles + 2] := sqrt (abs ((sum2 - r * sum1) / (n - 1)));
      end;
   end;
   assign  (g, 'Tahkuse_2002_2014_basic_statistics.xls'); rewrite (g);
   writeln (g, 'variable'#9'min'#9'p01%'#9'p10%'#9'p50%'#9'p90%',
                        #9'p99%'#9'max'#9'ave'#9'std'#9'missing%');
   for i := 1 to 114 do begin
      write (g, i, ') ', header [i]);
      for k := 1 to npercentiles + 2 do
         write (g, #9, cleanprint (stat [i, k], dec [i]));
      writeln (g, #9, cleanprint (100 * (nhours - n_known [i]) / nhours, 1));
   end;
   close (g);
   end;

procedure yeartables;
   var h, i, year, oldyear : integer;
   s : string;
   g : text;
   begin
   oldyear := 0;
   for h := 1 to nhours do begin
      year := round (data [h, 1]);
      if year > oldyear then begin // make new file
         if h > 1 then close (g);
         str (year, s);
         assign (g, 'Tahkuse_' + s + '_data.xls'); rewrite (g);
         for i := 1 to 113 do write (g, header [i], #9);
         writeln (g, header [114]);
         oldyear := year;
      end;
      for i := 1 to 114 do begin
         write (g, cleanprint ( data [h, i] , dec [i]));
         if i = 114 then writeln (g) else write (g, #9);
      end;
   end;
   close (g);
   end;

var h, k : integer;
    q : single;

begin
writeln; write (' Reading Tahkuse master data...');
readmaster;
writeln; write (' making quality rank-indices...');
for k := 88 to 93 do sortq (k);
for h := 1 to nhours do begin // determine q_low
   q := 1000;
   for k := 88 to 93 do if data [h, k] < q then q := data [h, k];
   data [h, 94] := q;
end;
writeln; writeln (' two-power fitting, remained measurement hours:');
write (nhours:6, ', ');
for h := 1 to nhours do  begin
   process_eas (h);
   process_ions (h, 1);
   process_ions (h, 2);
   process_ions (h, 3);
   k := nhours - h; if k mod 1000 = 0 then write (k:6, ', ');
end;
writeln; write (' delivering results...');
basic_stat;
yeartables;
writeln; writeln (' done, press ENTER!');
readln;
end.
```

```
program Textractor11;
{$R extractor.res}
{$mode delphi}{$H+}
uses sysutils, dateutils;

const         maxvars = 333;

var            n_vars : integer;
  firstdate, lastdate,
    filename, missing : string;
            delimiter : char;
           var_number : array [1..maxvars] of integer; // values 1..maxvars
                fopen : boolean;
                    f : text;

procedure failure (x : string);
begin
   if fopen then close (f);
   writeln ('Error in the ini-file: ' + x);
   writeln ('Press ENTER, make corrections and try again!');
   readln; halt;
end;

function doy (yyyymmdd : string) : integer;
   var y, m, d : word;
             k : integer;
             t : tdatetime;
   begin
   val (copy (yyyymmdd, 1, 4), y, k); if k <> 0 then failure (yyyymmdd);
   val (copy (yyyymmdd, 5, 2), m, k); if k <> 0 then failure (yyyymmdd);
   val (copy (yyyymmdd, 7, 2), d, k); if k <> 0 then failure (yyyymmdd);
   if not tryencodedate (y, m, d, t) then failure (yyyymmdd);
   result := dayoftheyear (t);
   end;

procedure make_ini;
var g : text;
begin
  assign  (g, 'Textractor11.ini'); rewrite (g);
  writeln (g, 'A pattern of Textractor11.ini:');
  writeln (g);
  writeln (g, 'The extraction of a table is controlled with phrases in the 6 control lines.');
  writeln (g, 'Everything else in the control file has no effect on the extraction process.');
  writeln (g, 'NB: Edit and save the control file before running the Textractor11!');
  writeln (g);
  writeln (g, '============================== CONTROL LINES
==============================');
  writeln (g, 'First day to extract YYYYMMDD: 20090301');
  writeln (g, ' Last day to extract YYYYMMDD: 20101031');
  writeln (g, '   List of columns to extract: 1-4, 33-20, 34');
  writeln (g, 'Delimiter in the output table: (           )');
  writeln (g, '  Missing entry in the output: (?)');
  writeln (g, '      Name of the output file: Tahkuse test extract.xl');
  writeln (g, '============================== EXPLANATIONS
==============================');
  writeln (g);
  writeln (g, 'The variables to be extracted should be indicated with numbers of columns in');
  writeln (g, 'the data files. The numbers should be delimited with commas or hyphens. A pair');
  writeln (g, 'of hyphen-limited numbers denotes a range from the first to the second number');
  writeln (g, 'in the similar way as in the list of pages when printing a document in MS Word.');
  writeln (g, 'An extra option is possibility to define an inverted range like 33-20 in the');
  writeln (g, 'example above. Here the variables will be extracted in order of 33, 32, .. 20.');
  writeln (g);
  writeln (g, 'Delimiters and missing entry codes can include invisible symbols and must be');
  writeln (g, 'written between parentheses. E.g. the output delimiter above is the invisible');
  writeln (g, 'symbol of tabulator. The parentheses are used here only as a frame and do not');
```

```
  writeln (g, 'belong to the code. A delimiter should be presented with one symbol while');
  writeln (g, 'a missing entry code may consist of several symbols, one symbol or nothing.');
  writeln (g, 'In the last case a delimiter will immediately follow the preceding delimiter.');
  writeln (g);
  writeln (g, 'NB: the cells of a table are interpreted and compared as the text. ');
  writeln (g, 'Thus the cells containing 999 and 999.0 are considered not equal.');
  writeln (g);
  writeln (g, 'NB: the files Textractor11.exe, Textractor11.ini and all 11 Tahkuse data files');
  writeln (g, '(Tahkuse_2004.xls ... Tahkuse_2014.xls) must be located in the common folder. ');
  writeln (g);
  writeln (g, 'NOTES ADDED BY THE USER:');
  writeln (g);
  close   (g);
  writeln;
  writeln (' NB: the control file "Textractor11.ini" not found and');
  writeln (' a new example control file is automatically created.');
  writeln (' Press ENTER, edit this file and try again!');
  readln; halt;
end;

procedure readini; // uses global f
                   // delivers global firstdate, lastdate, n_vars,
                   // var_number [i..n_vars], delimiter, missing
var s, u, a, b, ab, varlist : string;
        i, p, k, q, x, v, w : integer;

begin
   if not fileexists ('Textractor11.ini') then make_ini;
   assign (f, 'Textractor11.ini'); reset (f); fopen := true;
   repeat readln (f, s) until eof (f) or (copy (s, 1, 3) = '===');
   if eof (f) then failure ('no === line in the ini-file');
   readln (f, s); firstdate := trim (copy (s, pos (':', s) + 1, 999));
   if length (firstdate) <>  8 then failure (s);
   if (firstdate < '20040101') or (firstdate > '20141231') then failure (s);
   readln (f, s); lastdate := trim (copy (s, pos (':', s) + 1, 999));
   if length (lastdate) <> 8 then failure (s);
   if (lastdate < firstdate) or (lastdate > '20141231') then failure (s);
   readln (f, s); u := copy (s, pos (':', s) + 1, 999);
   p := length (u); varlist := '';
   for i := 1 to p do if u [i] in ['0'..'9', ',', '-']
      then varlist := varlist + u [i];
   if varlist = '' then failure (s);
   readln (f, s); u := trim (copy (s, pos (':', s) + 1, 999));
   if length (u) <> 3 then failure (s);
   delimiter := u [2];
   readln (f, s); u := trim (copy (s, pos (':', s) + 1, 999));
   if length (u) < 2 then failure (s);
   missing := copy (u, 2, length (u) - 2);
   readln (f, s); filename := trim (copy (s, pos (':', s) + 1, 999));
   if length (filename) < 1 then failure (s);
   if fileexists (filename) then
      failure ('this name is used, change name or delete the old file!');
   close (f); fopen := false;
   s := varlist + ','; n_vars := 0;
   repeat
      p := pos (',', s);
      if p > 1 then begin
         ab := copy (s, 1, p - 1);  // found between commas
         s := copy (s, p + 1, 999); // residue
         q := pos ('-', ab);
         if q = 0 then begin // process a single number
            n_vars := n_vars + 1;
            val (ab, x, k);
            if k <> 0 then failure ('illegal number in the variable list');
            var_number [n_vars] := x;
         end
         else begin // process a range of numbers
            a := copy (ab, 1, q - 1);   // first
            b := copy (ab, q + 1, 999); // last
```

```
            val (a, v, k);
            if k <> 0 then failure ('illegal subrange in the variable list');
            val (b, w, k);
            if k <> 0 then failure ('illegal subrange in the variable list');
            if w < v  then for i := v downto w do begin
               n_vars := n_vars + 1;
               var_number [n_vars] := i;
            end else for i := v to w do begin
               n_vars := n_vars + 1;
               var_number [n_vars] := i;
            end;
         end;
      end;
   until p < 2;
end; // of readini

function transform (row : string) : string;
   var i, n, u : integer;
             x : char;
          s, y : string;
          cell : array [1..maxvars] of string;
   begin
   s := #9 + row; u := length (s); n := 0;
   for i := 1 to u do begin
      x := s [i];
      if x = #9 then begin
         n := n + 1; cell [n] := ''
      end
      else cell [n] := cell [n] + x;
   end;
   y := '';
   for i := 1 to n_vars do begin
      if i > 1 then y := y + delimiter;
      s := cell [var_number [i]];
      if s = '-99' then s := missing;
      y := y + s;
   end;
   result := y;
   end;

var       nn, a, b, k,
           hour, year,
  firstyear, lastyear : integer;
  s, header, yearname : string;
                   g : text;

begin
readini;
write ('Running, please wait a little...');
nn := 0;
val (copy (firstdate, 1, 4), firstyear, k);
val (copy (lastdate, 1, 4), lastyear, k);
assign (g, filename); rewrite (g);
for year := firstyear to lastyear do begin
   str (year, yearname);
   assign (f, 'Tahkuse_' + yearname + '_data.xls'); reset (f);
   readln (f, header);
   if year = firstyear then writeln (g, transform (header));
   a := 1; b := 365 * 24; if year mod 4 = 0 then b := b + 24;
   if year = firstyear then a := 24 * doy (firstdate) - 23;
   if year =  lastyear then b := 24 * doy (lastdate);
   if a > 1 then for hour := 1 to a - 1 do readln (f, s);
   for hour := a to b do begin
      readln (f, s);
      writeln (g, transform (s));
      nn := nn + 1;
   end;
   close (f);
end;
close (g);
write ('Extracted ', nn, ' hours of data, press ENTER!'); readln;
end.
```