```
{ Program za POLIFAZNO SORTIRANJE kot je zapisan v knjigi :          }
{ Niklaus Wirth: Racunalnisko programiranje-2.del, (str. 172-174).   }
{ Pozor, program je zapisan v STANDARDNEM Pascalu!                   }

Program Polysort(output);   { polyphase sort with n tapes n}

  Const n=6;   { number of tapes }
  Type item=record
         key:integer;
       end;
       tape = file of item;
       tapeno = 1..n;
  Var leng,rand:integer; { used to generate file }
      eot:boolean;
      buf:item;
      f0:tape; { f0 is input tape with random numbers }
      f:array[1..n] of tape;

  Procedure List(var f:tape; n:tapeno);
  var z:integer;
  begin
    z:=0;
    writeln('TAPE ',n:2);
    while not eof(f) do
    begin
      read(f,buf);
      write(output,buf.key:5);
      z:=z+1;
      if z = 25 then
      begin
         writeln(output);
         z:=0;
      end;
    end;
    if z<>0 then
      writeln(output);
    reset(f);
  end; { List }

  Procedure PolyphaseSort;

  var i,j,mx,tn:tapeno;
      k,level:integer;
      a,d: array[tapeno] of integer;
        { a[j] = ideal number of runs on tape j }
        { d[j] = number of dummy runs on tape j }
      dn,x,min,z:integer;
      last: array[tapeno] of integer;
        { last[j] = key of tail item on tape j }
      t,ta: array[tapeno] of tapeno;
        { mappings of tape numbers }
```

```
   procedure selecttape;

   var i:tapeno;
       z:integer;

   begin
     if d[j]<d[j+1] then j:= j+1
     else
     begin
       if d[j]=0 then
       begin
         level:=level+1;
         z:=a[1];
         for i:= 1 to n-1 do
         begin
           d[i]:=z+a[i+1]-a[i];
           a[i]:=z+a[i+1];
         end;
       end;
       j:=1;
     end;
     d[j]:=d[j]-1;
   end;

   procedure copyrun;

   begin  { copy one run from f0 to tape j }
     repeat
       read(f0,buf);
       write(f[j],buf);
     until eof(f0) or (buf.key>f0^.key);
     last[j]:=buf.key;
   end;

begin     { distribute initial runs }
   for i := 1 to n-1 do
   begin
     a[i]:=1;
     d[i]:=1;
     rewrite(f[i]);
   end;
   level:=1;
   j:=1;
   a[n]:=0;
   d[n]:=0;
   repeat
     selecttape;
     copyrun
   until eof(f0) or (j=n-1);
```

```
while not eof(f0) do
begin
  selecttape;
  if last[j]<=f0^.key then
  begin  { continue old run }
    copyrun;
    if eof(f0) then d[j]:=d[j]+1
    else copyrun
  end
  else copyrun;
end;
for i:=1 to n-1 do
  reset(f[i]);
for i:=1 to n do
  t[i]:=i;
repeat
  { merge from t[1]…t[n-1] to t[n] }
  z:=a[n-1];
  d[n]:=0;
  rewrite(f[t[n]]);
  repeat
    k:=0;   { merge one run }
    for i:=1 to n-1 do
      if d[i]>0 then
        d[i]:=d[i]-1
      else
      begin
        k:=k+1;
        ta[k]:=t[i];
      end
    if k=0 then
      d[n]:=d[n]+1
    else
    begin  { merge one real run from t[1]…t[k] }
      repeat
        i:=1; mx:=1;
        min:=f[ta[1]]^.key;
        while i<k do
        begin
          i:=i+1;
          x:=f[ta[i]]^.key;
          if x<min then
          begin
            min:=x;
            mx:=i;
          end;
        end;
        { ta[mx] contains minimal element; move it to t[n] }
        read(f[ta[mx]],buf);
        eot:=eof(f[ta[mx]]);
        write(f[t[n]],buf);
```

```
          if buf.key>f[ta[mx]]^.key or eot then
          begin  { drop this tape }
            ta[mx]:=ta[k];
            k:=k-1
          end
        until k=0;
      end;
      z:=z-1;
    until z=0;
    reset(f[t[n]]);
    List(f[t[n]],t[n]);
    { rotate tapes }
    tn:=t[n];
    dn:=d[n];
    z:=a[n-1];
    for i:=n downto 2 do
    begin
      t[i]:=t[i-1];
      d[i]:=d[i-1];
      a[i]:=a[i-1]-z;
    end
    t[1]:=tn;
    d[1]:=dn;
    a[1]:=z;
    { sorted output is on t[1] }
    List(f[t[1]],t[1]);
    Level:=level-1;
  until level=0;
end; { PolyphaseSort }

Begin
  { generate random file }
  leng:=200; rand:=7789;
  repeat
    rand:=(131071*rand) mod 2147483647;
    buf.key:= rand div 2147484;
    write(f0,buf);
    leng:=leng-1;
  until leng=0;
  reset(f0);
  List(f0,1);
  PolyphaseSort;
End.
```