```
{ Program de "ascundere" a unui fisier text intr-un fisier cu tip. }
program Encode;

uses Crt;

{ Constanta care specifica dimensiunea unui bloc de text. Textul
  din fisierul original va fi preluat in blocuri de lungimea specificata
  aici si fiecare asemenea bloc va fi scris intr-un fisier cu tip. }
const BlockSize = 7;

type
    { Tip de date pentru un bloc de text. Se pastreaza numarul de
      ordine al blocului in fisierul original, dimensiunea blocului
      de text (s-ar putea, daca este vorba de ultimul bloc din
      fisierul original, ca blocul sa fie mai scurt) si blocul de text
      efectiv. }
    TBlock = record
           index : Integer;
           size : byte;
           data : string[BlockSize];
    end;

    { Un vector in care se poate pastra o lista de blocuri. }
    TBlockList = array[1..1000] of TBlock;

var
   { Aici pastram toate blocurile de text din fisierul original. }
   blocuri : TBlockList;

   { Aici memoram cate blocuri s-au citit din fisierul original. }
   nr_blocuri : Integer;

   { Fisierul original il deschidem ca fisier de tip caracter. }
   f_in : file of char;

   { Fisierul de iesire il deschidem ca fisier de tipul definit mai sus. }
   f_out : file of TBlock;

   { Variabile folosite la amestecarea blocurilor de text, la
     interschimbarea perechilor de blocuri. }
   a,b:Integer;
   temp : TBlock;

   { Variabila folosita la iteratii. }
   i : Integer;


begin
     ClrScr;

     { Deschidem fisierul original pentru citire. }
     Assign(f_in, 'original.pas');
     Reset(f_in);

     { De curiozitate afisam dimensiunea ocupata de fisier pe disc. }
     WriteLn('Spatiu ocupat pe disc: ', FileSize(f_in));

     { Calculam cate blocuri de text ar fi necesare pentru a pastra
       continutul fisierului. }
     WriteLn('Nr. blocuri de date calculat: ', FileSize(f_in)/BlockSize:0:2);
```

```
      { Atat timp cat nu s-a atins sfarsitul fisierului de intrare,
        citim blocuri din fisier. }
      nr_blocuri := 0;
      while not(eof(f_in)) do
            begin
                 { Daca nu s-a atins sfarsit de fisier, inseamna ca
                   mai avem date pentru cel putin inca un bloc de text.
                   Ca urmare incrementam numarul de blocuri. }
                 nr_blocuri := nr_blocuri+1;

                 { Cu blocul curent executam urmatorii pasi:
                   * Ii initializam numarul de ordine;
                   * Ii initializam dimensiunea cu zero;
                   * Cat timp mai avem date in fisier citim caractere
                     in blocul de text (corespunzator actualizam si
                     dimensiunea blocului). Ne oprim cu citirea cand
                     se umple blocul de text (sau cand nu mai sunt date
                     in fisier). }
                 with blocuri[nr_blocuri] do
                      begin
                           index := nr_blocuri;
                           size := 0;
                           while not(eof(f_in)) and (size<BlockSize) do
                                 begin
                                      size:= size+1;
                                      read(f_in, data[size]);
                                 end;
                      end;
            end;

      { Am incheiat citirea din fisier, deci inchidem fisierul. }
      Close(f_in);

      { Afisam numarul de blocuri utilizate. Ar trebui sa obtinem
        aceeasi valoare ca si cea calculata mai sus. }
      WriteLn('Nr. blocuri real: ', nr_blocuri);

      { Amestecam blocurile de date intre ele. Pentru aceasta alegem
        perechi de blocuri si le interschimbam. }
      Randomize;
      for i:= 1 to 1000 do
          begin
               a := Random(nr_blocuri) + 1;
               b := Random(nr_blocuri) + 1;

               temp := blocuri[a];
               blocuri[a] := blocuri[b];
               blocuri[b] := temp;
          end;

      { Scriem blocurile de date amestecate in fisierul de iesire.
        Pentru asta deschidem fisierul de iesire in mod scriere is
        scriem pe rand fiecare bloc, dupa care inchidem fisierul. }
      Assign(f_out, 'encode.dat');
      Rewrite(f_out);
      for i:= 1 to nr_blocuri do
          Write(f_out, blocuri[i]);
      Close(f_out);
end.
```