Nežinomas kodavimas

Pradėkime nuo baito sąvokos. Baitą sudaro aštuoni bitai. Bitas – tai vienetas arba nulis.

Bito nr.	7	6	5	4	3	2	1	0
	0	1	1	0	1	0	0	0

Pavyzdžiui, baitas $01101000_2 = 68_{16} = 104_{10}$; Tai yra dvejetainis, šešioliktainis bei dešimtainis baito užrašai.

Į pradinių duomenų failą žiūrima kaip į ištisinę bitų eilę. Fiziškai failuose bitai saugomi nuo mažiausio iki vyriausio. Aukščiau parodytas baitas faile bus saugomas taip:

Bito nr.	0	1	2	3	4	5	6	7
	0	0	0	1	0	1	1	0

Taigi jei būtų grupuojama po 5 bitus, į pirmąją grupę pakliūtų šie bitai: 0 0 0 1 0.

Sąlygoje pasakyta, kad bitai skirstomi į vienodo ilgio grupes po 7, 6, 5, 4, 3, arba 2 bitus. Taigi pirmiausia reikėtų nustatyti, į kokio ilgio grupes skirstomi bitai. Tam reikėtų susikurti keletą skirtingo ilgio pradinių duomenų bylų.

Pavyzdžiui, susikuriame bylą, kurios ilgis – keturi baitai. Joje bus $4 \times 8 = 32$ bitai. Pagal rezultatų bylos ilgį jau galima daryti išvadas:

Grupės dydis	*Rezultatų .failo ilgis*
2	16 pilnų grupių × 1 baitas = 16 baitų
3	(10 pilnų grupių + 1 nepilna grupė) × 1 baitas = 11 baitų
4	8 pilnos grupės × 1 baitas = 8 baitai
5	(6 pilnos grupės + 1 nepilna grupė) × 1 baitas = 7 baitai
6	(5 pilnos grupės + 1 nepilna grupė) × 1 baitas = 6 baitai
7	(4 pilnos grupės + 1 nepilna grupė) × 1 baitas = 5 baitai

Pasižiūrėję į rezultatų failą galime pasakyti, į kokio didumo grupes skirstomi bitai. Mūsų uždavinio atveju bitų grupės ilgis yra 4.

Jei kiekviena keturių bitų grupė koduojama vienu baitu, reikia išnagrinėti visas galimas keturių bitų grupes. Jų bus šešiolika: 0000, 0001, ... 1111. Reikia susikurti pradinių duomenų failą, kuriame būtų visos įmanomos kombinacijos.

Tokį failą nesunkiai galima sukurti patiems. Yra įvairių būdų tai padaryti. Tam reikėtų panagrinėti simbolių kodus. Dirbdami Paskaliu, tai galite padaryti naudodami funkciją *ord*. Ši funkcija grąžina dešimtainį simbolio kodą. Šešioliktainius simbolių kodus galite sužinoti žiūrėdami bylų turinius su Nortono komanderiu arba DOS'o navigatoriumi.

Panagrinėję pamatysite, kad simbolių ‘0’..., ‘9’ šešioliktainiai kodai yra 30_{16}, ..., 39_{16}. (gausime visas bitų kombinacijas nuo 0000 iki 1001), o simbolių ‘J’, ‘K’, ‘L’, ‘M’, ‘N’, ‘O’ kodai yra $4A_{16}$, ...$4F_{16}$ (gausime visas likusias bitų kombinacijas).

Sudarę tokią duomenų failą, ir jį užkodavę, galime pasakyti, kaip koduojama kiekviena bitų grupė:

Grupė (jos dvejetainis užrašas)	Simbolis iš kodų lentelės, atitinkantis baitą, kuriuo koduojama
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Pavyzdžiui, pradinėje duomenų byloje yra simbolis ‘G’. Jo šešioliktainis kodas yra $47_{16} = 01000111_2$. Pirmą baitų grupę sudarys baitai 0111 (koduojama 7), antrąją – 0100 (koduojama 4). Taigi simbolis ‘G’ bus užkoduotas ‘74’.

Pastebėsime, kad kiekvienas simbolis koduojamas dviem simboliais – apsuktu to simbolio šešioliktainiu kodu (du skaitmenys). Taip gaunama todėl, kad fiziniame lygyje baitai išdėstyti nuo mažiausios iki vyriausios skilties.

Pateikiame programos KOD.EXE pradinį tekstą:

```
program kodavimas_4_bitais;

  const bitai = 4;            { bitų grupės ilgis }
        lent: string[16] = '0123456789ABCDEF';
              { kodavimui naudojami simboliai }
        max = 10000;  { maksimalus rezultatų bylos dydis baitais }

  procedure klaida;
  { nutraukia programos darbą dėl įvedimo arba išvedimo klaidos }
  begin
    writeln ('Įvedimo/išvedimo klaida!');
    halt
  end;  { klaida }

  procedure atidar_bylas (var fskait, fraš: file);
  { patikrina parametrus ir atidaro bylas }
  begin
    writeln;
    if paramcount <> 2  { jei nenurodyti tinkami parametrai }
       then begin
              writeln ('Naudojimas: ');
              writeln(' KOD.EXE  duom.dat rez.dat');
              writeln;
              halt
            end;
    assign(fskait, paramstr(1));
    {$I-}
    reset (fskait, 1);
    {$I+}
    if ioresult <> 0
       then klaida;
    assign(fraš, paramstr(2));
    {$I-}
    rewrite(fraš, 1);
    {$i+}
    if ioresult <> 0
       then klaida
  end;  { atidar_bylas }
```

```
procedure uždar_bylas (var fskait, fraš: file);
begin
  close(fraš);
  close(fskait)
end; { uždar_bylas }

procedure kodavimas (var fskait, fraš: file);
{ užkoduoja duomenis }
  var duom, rez: array [1..max] of byte;
      dydis, { pradinių duomenų bylos dydis baitais }
      rsk,   { rezultatų bylos dydis baitais }
      perskait, i, poslinkis, kaukė: integer;
      bit4: byte; { keturių bitų grupė }
begin
  { inicializuojame pradinių duomenų ir rezultatų masyvus }
  fillchar (duom, sizeof(duom), 0);
  fillchar (rez, sizeof(rez), 0);
  { randame pradinių duomenų kiekį baitais }
  dydis := filesize (fskait);
  { perskaitome pradinius duomenis }
  blockread (fskait, duom, dydis, perskait);
  if perskait <> dydis
     then begin
            uždar_bylas (fskait, fraš);
            klaida
          end;
  i := 1;
  poslinkis := 0;
  rsk := 0;
  kaukė := $0F; { 00001111 }
  while i <= dydis do
    begin
      rsk := rsk + 1;
      { paimam eilinius 4 bitus }
      bit4 := (duom[i] shr poslinkis) and kaukė;
      { juos užkoduojame vienu baitu kurį įrašome į rezultatų masyvą }
      rez[rsk] := ord (lent[bit4 + 1]);
      poslinkis := poslinkis + bitai;
      if poslinkis > 7 { paslenkama arba per 0 arba per 4 bitus }
         then begin
                poslinkis := poslinkis - 8;
                i := i + 1
              end
    end;
  { įrašome užkoduotus duomenis į rezultatų bylą }
  blockwrite (fraš, rez, rsk, dydis);
```

```
    if rsk <> dydis
       then klaida
  end;  { kodavimas }

  var fskait, fraš: file;
begin
  atidar_bylas (fskait, fraš);
  kodavimas (fskait, fraš);
  uždar_bylas (fskait, fraš)
end.
```

Kadangi šis uždavinys buvo skirtas jaunesniems moksleiviams, pateiksime paprastesnį jo sprendimą. Atkoduodami į pradinių duomenų ir į rezultatų failus žiūrime kaip į tekstinius failus. Iš pradinių duomenų failų skaitysime po du simbolius (tai bus du baitai) ir juos iškoduosime.

Du perskaitytieji simboliai bus du šešioliktainiai skaitmenys, pavyzdžiui ‘F4’. Šiuos simbolius sukeisime vietomis (gausime ‘4F’), paversime skaitmenimis ir gautą šešioliktainį skaičių užrašysime dešimtaine sistema. $4F_{16} = 79_{10}$. Tai yra užkoduoto simbolio kodas. Belieka į rezultatų failą išvesti šį simbolį.

```
procedure atkoduoti (var fduom, frez: text);
  var b1, b2, simb: char;
      kodas, sk1, sk2: integer;
begin
  while not eof (fduom) do
    begin
      { perskaitome du baitus, kurias buvo koduojamas vienas baitas }
      read (fduom, b1, b2);
      { šešioliktainius skaitmenis užrašysime dešimtainiais }
      if b1 in ['A'..'Z']
         then sk1 := ord(b1) - ord('A') + 10
         else sk1 := ord(b1) - ord('0');
      if b2 in ['A'..'Z']
         then sk2 := ord(b2) - ord('A') + 10
         else sk2 := ord(b2) - ord('0');
      kodas := sk2 * 16 + sk1;
      simb := chr (kodas);
      write (frez, simb)
    end
end;  { atkoduoti }
```

Šio uždavinio sprendimui patikrinti užtektų vieno didelio testo, kuriame būtų surašyti visi galimi simboliai (t. y. simboliai kurių kodai priklauso intervalui [00, … FF]).