Nežinomas kodavimas

Šio uždavinio sprendimas primena to paties uždavinio versijos jaunesniems sprendimą. Todėl prieš pradėdami nagrinėti šį sprendimą išanalizuokite to uždavinio sprendimą.

Įvykdykime KOD.EXE su savo sukurta pradinių duomenų byla. Pasižiūrėję į rezultatų bylą pamatysime į kokio didumo grupes skirstomi bitai. Šiuo atveju bitų grupės ilgis yra 6.

Kaip ir ankstesnio uždavinio atveju, reikia išnagrinėti visas galimas šešių bitų grupes. Jų bus šešiasdešimt keturios (2^6): 000000, 000001, ... 111111. Susikuriame pradinių duomenų bylą, kurioje būtų visos įmanomos kombinacijos.

Šį kartą bylą sukurti kiek sunkiau: keturios bitų grupės sujungtos kartu sudarys tris baitus. Bus geriausia, jei parašysime programą, kuriančią tokią bylą:

```
program kodavimo_sistemos_nustatymas;
{ programa sukuria bylą, kurioje eilės tvarka išdėstyti visi }
{ galimi šešių bitų grupės variantai: nuo 000000 iki 111111 }
  procedure bitų_grupės (var f: file);
    var kiek, num, i: integer;
        baitas1, baitas2, baitas3: byte;
        rezultatas: array [1..48] of byte;
  begin
    i := 0;    { bitų grupė }
    num := 1;  { nagrinėjamo baito numeris }
    while i < 64 do
      begin
        { sudarysime keturias bitų grupes, kurios užims tris baitus }
        baitas1 := byte(i + (i + 1) shl 6);
        baitas2 := byte((i + 1) shl 2 shr 4 + (i + 2) shl 4);
        baitas3 := byte((i + 2) shr 4 + (i + 3) shl 2);
        rezultatas[num]   := baitas1;
        rezultatas[num+1] := baitas2;
        rezultatas[num+2] := baitas3;
        i := i + 4;
        num := num + 3;
      end;
    { 64 bitų grupės po 6 bitus sudarys 48 baitus }
    blockwrite (f, rezultatas, 48, kiek);
  end; { bitų_grupės }
```

```
  var f: file;
begin
  writeln ('Ši programa sukuria bylą TEST.DAT,
            kurioje eilės tvarka');
  writeln ('surašytos 64 bitų grupės (po 6 bitus):
            nuo 000000 iki 111111');
  assign (f, 'TEST.DAT');
  rewrite (f, 1);
  bitų_grupės (f);
  close (f);
end.
```

Sukūrę bylą TEST.DAT ir ją užkodavę programa KOD.EXE, pamatysime, kad baitų grupės koduojamos tokia simbolių eilute:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz+-

T. y. bitų grupė „000000“ koduojama simboliu „0“, bitų grupė „000001“ koduojama simboliu „1“ ir t. t. Bitų grupė „111111“ koduojama simboliu „-“.

Žinodami kokiu būdu užkoduojama galime parašyti dekodavimo programą. Prieš tai panagrinėkime tokį pavyzdį. Sakykime, kad užkoduoti duomenys pavaizduoti tokia simbolių eilute: B96.

Paaiškinsime, kaip jie dekoduojami. Užkoduotą tekstą sudaro trys baitai. Randame visų trijų simbolių numerius kodavimui naudojamų simbolių eilutėje: simbolio „B“ numeris yra 12, simbolio „9“ – 10, ir simbolio „6“ – 7.

Simbolio numeris, užrašytas dvejetaine sistema, turėtų sutapti su tuo simbolių užkoduota šešių bitų grupe. Tačiau pirmoji koduojama bitų grupė yra sudaryta vien iš nulių„000000“, o simboliai eilutėje numeruojami nuo 1. Taigi, norint rasti šešių bitų grupę, kuri buvo užkoduota, reikia iš simbolio numerio atimti vienetą. Todėl nagrinėsime skaičius 11_{10}, 9_{10}, 6_{10}.

Šiuos skaičius užrašome dvejetaine sistema:

$11_{10} = 00001011_2$;
$9_{10} = 00001001_2$;
$6_{10}.= 00000110_2$;

Žinome, kad fiziškai kompiuteryje baitai laikomi nuo jauniausio iki vyriausio. Imame po šešis jaunesnius kiekvieno baito bitus ir juos surašome vienas po kito. Rašome nuo jauniausio iki vyriausio: 110100100100011000

Atskiriame pirmuosius aštuonis bitus (11010010) juos perrašome nuo vyriausio iki jauniausio: $01001011_2 = 4B_{16} = 75_{10}$. ASCII kodų lentelėje tai yra simbolio „K“ kodas. Imame antrąjį baitą: $01100010_2 = 62_{16} = 98_{10}$. ASCII kodų lentelėje tai yra simbolio „b“ kodas. Likę du nuliniai bitai „00“ baito nebesudaro, taigi į juos dėmesio nebekreipiame. Kadangi užkoduojamuose duomenyse bitų skaičius nesidalijo iš 6, teko prirašyti du papildomus nulinius bitus.

Atkodavę duomenis, gavome „Kb“.

Pateikiame programos KOD.EXE pradinį tekstą:

```
program kodavimas_6_bitais;

  const
   BITAI = 6;            { bitų grupės ilgis }
   LENT: string[64] = { kodavimui naudojami simboliai }
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
+-';
   MAX = 10000;  { maksimalus rezultatų bylos dydis baitais }

  procedure klaida;
  { nutraukia programos darbą dėl įvedimo/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 }
      i, { nagrinėjamo baito numeris }
      perskait, poslinkis, kaukė: integer;
      bit6: byte; { šešių 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 := $3F; { 00111111 }
  while i <= dydis do
    begin
      rsk := rsk + 1;
      { paimame eilinius 6 bitus }
      if i = dydis
```

```
            then bit6 := (duom[i] shr poslinkis) and kaukė
            else bit6 := (duom[i] shr poslinkis +
                          duom[i+1] shl (8-poslinkis))
                          and kaukė;
         { juos užkoduojame vienu baitu kurį įrašome į rezultatų masyvą }
         rez[rsk] := ord (lent[bit6 + 1]);
         poslinkis := poslinkis + BITAI;
         if poslinkis > 7
            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.
```

Žemiau pateikiame procedūrą, atkoduojančią duomenis. Pagrindinė programos dalis bei procedūra *klaida* (į ją kreipiasi procedūra *dekodavimas*) yra analogiškos aukščiau pateiktoje programoje esančiai pagrindinei programos daliai bei procedūrai *klaida*.

```
const BITAI = 6;            { bitų grupės ilgis }
      MAX = 10000;  { maksimalus užkoduotos bylos dydis baitais }
      LENT: string[64] = { kodavimui naudojami simboliai }
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz+
-';

procedure dekodavimas (var fskait: text; var fraš: file);
{ dekoduoja duomenis }
  var rez: array [1..MAX] of byte;
      num, nr, poslinkis, dydis: integer;
      simb: char;
  function numeris (c: char): integer;
  { randa simbolio c numerį lentelėje LENT }
    var rasta: boolean;
        i: integer;
```

```
  begin
    i := 0; rasta := false;
    while (i < 64) and not rasta do
      begin
        i := i + 1;
        rasta := c = LENT[i];
      end;
    if rasta
       then numeris := i
       else numeris := 0
  end; { numeris }
begin
  { inicializuojame rezultatų masyvą }
  fillchar (rez, sizeof(rez), 0);
  poslinkis := 0; { poslinkis baito pradžios atžvilgiu }
  nr := 1; { baito numeris atkoduotame tekste }
  while not eof (fskait) do
    begin
      { iš užkoduotos bylos perskaitome simbolį }
      read (fskait, simb);
      { randame jo numerį simbolių eilutėje lent }
      num := numeris (simb);
      { jei tokio numerio nerasta }
      if num = 0
         then klaida;
      { dekoduojame baitą }
      rez[nr] := byte(rez[nr] + (num-1) shl poslinkis);
      if poslinkis > 2
         then rez[nr+1] := byte((num-1) shr (8 - poslinkis));
      { perskaičiuojame poslinkį }
      poslinkis := poslinkis + BITAI;
      if poslinkis > 7
         then begin
                poslinkis := poslinkis - 8;
                nr := nr + 1
              end;
    end;
  { sumažiname nr reikšmę: arba ji buvo padidinta be reikalo }
  { arba paskutinysis baitas užpildytas užkoduojant papildomai }
  { prirašytais nuliais }
  nr := nr - 1;
  { įrašome dekoduotus duomenis į rezultatų bylą }
  blockwrite (fraš, rez, nr, dydis);
  if nr <> dydis
     then klaida;
end; { kodavimas }
```