Atspėk skaičių

Tai pirmasis dialoginis uždavinys, pateiktas Lietuvos Informatikos olimpiadose. Jo sprendimas nėra sudėtingas. Svarbu, kad programa tinkamai pasinaudotu pateiktu moduliu.

Sugalvotasis skaičius gali būti bet koks keturženklis skaičius iš intervalo [1000; 9999], kurio visi skaitmenys skirtingi. Taigi, susidarome visų keturženklių skaičių kurių skaitmenys skirtingi sąrašą. Sąraše saugosime galimus sprendinius.

Pirmąjį spėjamą skaičių parenkame atsitiktinai – bet kurį skaičių iš sąrašo. Tarkime, kad tai buvo 4521. Gaunamas atsakymas: atspėti du skaičiai, iš jų vienas yra savo vietoje. Pasinaudokime šia informacija. Sąraše palikime tik tuos skaičius, kurių lygiai du skaitmenys sutampa su spėtuoju skaičiumi ir iš dviejų tik vienas yra toje pačioj vietoje, kaip ir spėtajame skaičiuje. Pavyzdžiui, skaičiai 4302, 1589 bus palikti sąraše, o skaičiai 4891, 1284 pašalinti.

Toliau vėl spėjama atsitiktinai. Imamas bet kuris sąraše likęs skaičius, ir vėl pasinaudojama pateiktu atsakymu. Veiksmai kartojami, kol sąraše lieka tik vienas skaičius.

Spėjant tokiu būdu bet kokį skaičių įmanoma atspėti per 7 ar mažiau kartų. Todėl laikoma, kad atsakymas gautas, jei atspėta per 7 ar mažiau kartų.

```
program atspėk_skaičių;
  uses žaidimas;
  type sąrašas = ^elem;
       elem = record
                ska: 1000..9999;
                tolesnis: sąrašas
              end;

  function skirtingi (ket: integer): boolean;
  { ar keturženklio skaičiaus ket skaitmenys yra skirtingi }
    var aibė: set of char;
        elem_sk: integer;
        c: char;
        eil: string;
  begin
    str (ket, eil);   { skaičius paverčiamas eilute }
    aibė := [eil[1]] + [eil[2]] + [eil[3]] + [eil[4]];
    elem_sk := 0;
    for c := '0' to '9' do
      if c in aibė
```

```
        then elem_sk := elem_sk + 1;
  skirtingi := elem_sk = 4;
end;  { skirtingi }

procedure sąrašo_sudarymas (var s: sąrašas);
  var p, p1: sąrašas;
      ket: integer;
begin
  new (p);
  p^.ska := 1023;  { mažiausias keturženklis skaičius, kurio }
                   { visi skaitmenys skirtingi }
  p1 := p;
  for ket := 1024 to 9999 do
    if skirtingi (ket)
        then begin     { įtraukiame skaičių į sąrašą }
               new (p^.tolesnis);
               p := p^.tolesnis;
               p^.ska := ket
             end;
  p^.tolesnis := nil;
  s := p1
end;  { sąrašo_sudarymas }

function vienodi (sk1, sk2: integer): integer;
{ suskaičiuoja, kiek skaičiai sk1 ir sk2 turi vienodų skaitmenų }
  var aibė: set of 0..9;
      kiek: integer;
begin
  aibė := [];
  while sk1 <> 0 do  { pirmojo skaičiaus skaitmenis įrašome }
    begin            { į aibę }
      aibė := aibė + [sk1 mod 10];
      sk1 := sk1 div 10;
    end;
  kiek := 0;
  while sk2 <> 0 do     { tikriname kiek antrojo skaičiaus skaitmenų }
    begin               { yra aibėje }
      if sk2 mod 10 in aibė
         then kiek := kiek + 1;
      sk2 := sk2 div 10
    end;
  vienodi := kiek
end;   { vienodi }
```

```
  function vietoje (sk1, sk2: integer): integer;
  { suskaičiuoja, kiek skaičiai sk1 ir sk2 turi vienodų skaitmenų }
  { esančių tose pačiose pozicijose }
    var eil1, eil2: string[4];
        i, kiek: integer;
  begin
    str (sk1, eil1);
    str (sk2, eil2);
    kiek := 0;
    for i := 1 to 4 do
      if eil1[i] = eil2[i]
         then kiek := kiek + 1;
    vietoje := kiek
  end; { vietoje }

  procedure spėti (s: sąrašas);
    var sk, atsp, vietoj: integer;
        p: sąrašas;
  begin
    while s <> nil do
      begin { kol sąraše daugiau nei vienas elementas }
        sk := s^.ska;
        ėjimas (sk, atsp, vietoj);
        p := s;   { peržiūrėsime sąrašą }
        while p^.tolesnis <> nil do
          begin
            if (vienodi(sk, p^.tolesnis^.ska) <> atsp) or
               (vietoje(sk, p^.tolesnis^.ska) <> vietoj)
               then { pašalinamas elementas }
                    p^.tolesnis := p^.tolesnis^.tolesnis
               else p := p^.tolesnis
          end;
        s := s^.tolesnis; { pašalinamas pirmasis sąrašo elementas }
      end;                { nes jį jau spėjome }
    ėjimas (sk, atsp, vietoj);
  end; { spėti }

  var s: sąrašas;
begin
  sąrašo_sudarymas (s);
  spėti (s);
end.
```

Rodyklės tipas yra sudėtinga duomenų struktūra. Dėl to pailgėjo ir pati programa. Juk keturženklius skaičius buvo galima saugoti ir loginiame masyve. Rodyklės tipą pasirinkome nes šis tipas sudaro daug sunkumų sprendžiantiems: norėjome pateikti dar vieną algoritmo, kuriame vartojami šie duomenų tipai, pavyzdį. O šiame uždavinyje jį vartoti logiška: jei tik skaičius nėra tinkamas, jis pašalinamas iš sąrašo ir daugiau nenagrinėjamas.

Pastaba. Sąlygoje pateikėme demonstracinį modulį ŽAIDIMAS. Testavimui vartotas modulis iš esmės nesiskyrė nuo demonstracinio. Jame buvo įdėta papildoma apsauga nuo bandančių nesąžiningai spręsti uždavinį.