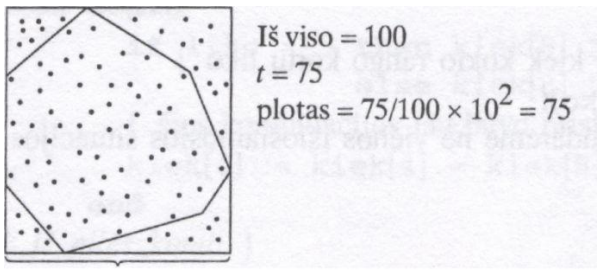
Didelė valstybė

Tai vienas įdomesnių ir sunkesnių šios olimpiados uždavinių. Prieš nagrinėdami galimus sprendimus paaiškinsime sąvoką „penkių skaitmenų tikslumu“. Uždavinio autoriams buvo svarbūs pirmieji penki skaitmenys, net jei jie ir nėra reikšminiai. Sakykime, kad tikrasis daugiakampio plotas yra 1 458 963,12. Galimas atsakymas yra 1 459 000,00. Jei daugiakampio plotas lygus 0,0005487, tai galimas atsakymas -0,0005. Sąlygoje nepasakyta, ar reikia suapvalinti gautą atsakymą, ar ne. Kadangi ploto ieškoma keleto skaitmenų tikslumu, suapvalinti būtų logiška.

Pereikime prie galimų uždavinio sprendimo būdų.

Monte Karlo metodas. Imame atsitiktinius kvadratui priklausančius taškus ir tikriname, kurie jų priklauso valstybei. Tarkime, kad iš viso išbandėme *iš_viso* taškų ir iš jų valstybei priklausė *t* taškų (žr pav.):

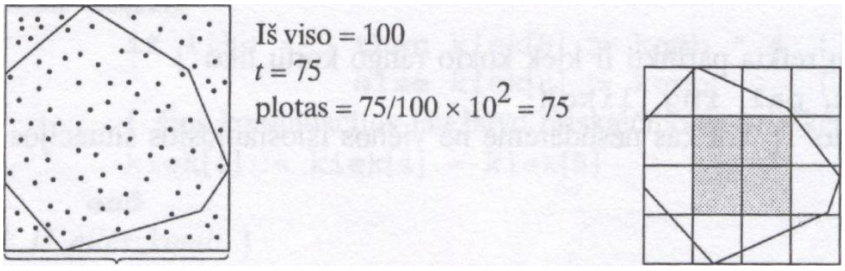


Galime pasakyti, kad valstybė sudaro *t*/*iš_viso* procentų kvadrato ploto, t.y. valstybės plotas lygus $(t \times n^2)$/*iš_viso*. Kuo daugiau taškų parinksime, tuo tikslesnis bus rezultatas.

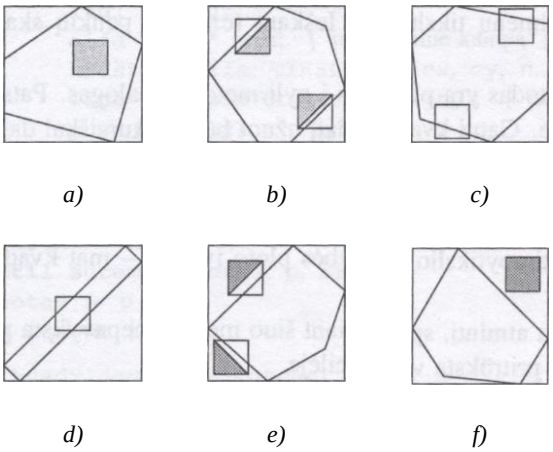
Kad metodas būtų tikslus, kvadratui priklausantys taškai turi būti generuojami *atsitiktinai.* Deja, Paskalio kalbos priemonės atsitiktiniams skaičiams generuoti (procedūra *randomize* bei funkcija *random)* nėra labai gerai padarytos.

Sprendžiant tokiu būdu nelabai aišku kada pasiekiamas norimas tikslumas – kada sustoti. Galima atsitiktinius taškus generuoti tol, kol baigsis laiko limitas tikintis, kad laiko užteks. Sprendžiant šiuo būdu buvo gaunama santykinė paklaida iki 5%.

Dalijimo į vienodo dydžio kvadratėlius metodas. Šį kartą skaičiuosime ne tik valstybės plotą, bet ir valstybei nepriklausantį kvadrato plotą. Imame duotąjį kvadratą ir padalijame jį mažesnius kvadratėlius (žr. pav.):



Tuomet ištiriame kiekvieną kvadratėlį: suskaičiuojame, kelios jo viršūnės priklauso valstybei (žr. pav.):



Jei priklauso visos keturios (a), kvadratėlio plotą pridedame prie valstybės ploto. Jei trys (b) - pusę kvadratėlio ploto pridedame prie valstybės ploto. Jei priklauso tik viena viršūnė (e), pusę kvadratėlio ploto pridedame prie valstybei nepriklausančio kvadrato ploto, jei nepriklauso nė viena (f), visą kvadratėlio plotą pridedame prie valstybei nepriklausančio kvadrato ploto

Žinodami kvadrato plotą bei tiksliai suskaičiuotą valstybei priklausančią ir nepriklausančią ploto dalį, galime pasakyti, kokiu tikslumu apskaičiavome plotą. Jei tikslumas nepakankamas, tai pradinį kvadratą dalijame į dar didesnį kvadratėlių skaičių ir skaičiuojame iš naujo.

Vienas didesnių šio metodo trūkumų yra tas, kad nagrinėjami tik fiksuoto dydžio kvadratėliai. Jei kvadrato kraštinė padalijama į k dalių, tuomet visiems kvadratėliams patikrinti prireiks $(k + 1)^2$ kreipinių į funkciją *valstybė.*

Pagal šį algoritmą parašyta programa skaičiuodavo dviejų bei trijų skaitmenų tikslumu. Gauta paklaida svyravo nuo 0,5% iki 0,7%. Bandant skaičiuoti tiksliau būdavo viršijami laiko limitai.

Grįžimo arba paieškos į gylį metodas. Pradinis kvadratas dalijamas į keturis mažesnius kvadratus. Tuomet patikrinama, kelios kiekvieno šių kvadratų viršūnės priklauso valstybei. Jei priklauso visos keturios arba nepriklauso nė viena, tai tų kvadratų plotai arba pridedami prie valstybės ploto, arba prie valstybei nepriklausančio ploto.

Visi likę kvadratai vėl dalijami į keturias dalis ir skaičiuojama, kelios gautų kvadratėlių viršūnės priklauso valstybei.

Atkreipkite dėmesį, kad pirma būtina padalyti pradinį kvadratą į keturis kvadratėlius ir tik po to skaičiuoti, kelios viršūnės priklauso valstybei. Mat valstybei gali nepriklausyti nė viena pradinio kvadrato viršūnė, tačiau tai nereiškia, kad kvadrato viduje nėra valstybei priklausančių taškų. O štai visiems mažesniems kvadratėliams šį taisyklė galioja: jei nė viena viršūnė nepriklauso valstybei, tai ir nė vienas kvadratėlio taškas jai nepriklauso.

Šis metodas ištaisytų praeito metodo trūkumą: jei radome didelį kvadratą, kurio visos viršūnės priklauso valstybei, tai jo plotą pridedame prie valstybės ploto ir daugiau to kvadrato nebenagrinėjame.

Tačiau kyla problema: kaip nustatyti kada užbaigti skaičiavimus. Gali būti, kad tas ketvirtadalis pradinio kvadrato, kurį šiuo momentu rekursiškai nagrinėjame tesudaro tūkstantąją valstybės ploto dalį. Galime kitu būdu (pavyzdžiui, Monte Karlo arba dalijimo į vienodo dydžio kvadratėlius metodu) apytiksliai įvertinti valstybės plotą. Vertinant apytiksliai, už tenka rasti plotą vieno skaitmens tikslumu. Tai neužims daug procesoriaus laiko. O tuomet galima apskaičiuoti tikslumą. Jį žinodami, galėsime pasakyti, kada užbaigti rekursiją.

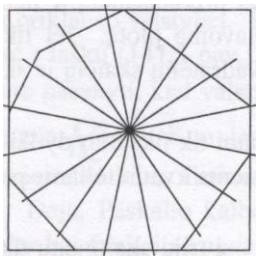
Spręsdami uždavinį šiuo metodu, gaudavome paklaidą nuo 0,005% iki 0,02%. Būdavo skaičiuojama trijų bei keturių skaitmenų tikslumu. Ieškant rezultato penkių skaitmenų tikslumu, viršijami laiko limitai.

Paieškos į plotį metodas. Šis metodas yra paieškos į gylį metodo analogas. Gauti kvadratėliai, užuot buvę rekursiškai dalijami į mažesnius kvadratėlius, įrašomi į eilę ir dalijami į keturias dalis tada, kai pasiekia eilės pradžią. Laiko sąnaudomis šis metodas nesiskiria nuo paieškos į gylį. Tačiau sprendžiant šiuo metodu nebereikia kitu būdu ieškoti apytikslio valstybės ploto įverčio – mat kvadratėliai tikrinami jų plotų mažėjimo tvarka.

Deja, net ir labai taupiai vartojant atmintį, sprendžiant šiuo metodu nepavyksta pasiekti didesnio tikslumo, nes labai greitai pritrūksta vietos eilėje.

Dalijimo į trikampius metodas. Žinodami, kad valstybė - tai iškilasis daugiakampis, taip pat kad tas daugiakampis liečia visas keturias kvadrato kraštines, galime teigti, kad kvadrato vidurio taškas priklauso valstybei. Iš šio taško leidžiame spindulius į visas puses ir dvejetainės paieškos metodu randame taškus, kur tie spinduliai kerta valstybės sieną. Valstybę padalijame į trikampius. Jos plotas lygus tų trikampių plotų sumai.

Kuo mažesnis yra kampas tarp spindulių, tuo tikslesnis rezultatas.



```
program didelė_valstybė;
  uses terrainc;
  const TIKSLUMAS = 0.000001; { šiuo tikslumu ieškome taško, kuriame }
                              { spindulys kerta valstybės sieną }
      SKAIT = 5;              { tiek skaitmenų tikslumu reikia rasti plotą }
  function plotas (kvadx, kvady, n: integer): real;
  { randa valstybės plotą }
    var senas, p, cx, cy, alfa, ilg2, ilgi, min, x,
        y: real;
        i, kiek: integer;
begin
  p := 0; {kol kas plotas lygus 0 }
  cx := kvadx + n/2; cy := kvady + n/2; { kvadrato vidurio taškas }
  kiek := 1440; { pasirenkame pradinį padalijimų skaičių }
  repeat
    senas := p; p := 0;
    min := 2 * PI / kiek;
    alfa := 0;
    { randa tašką, kuriame spindulys, išvestas iš kvadrato centro kampu alfa, kerta daugiakampį }
    kraštas (alfa, TIKSLUMAS, cx, cy, n, x, y);
```

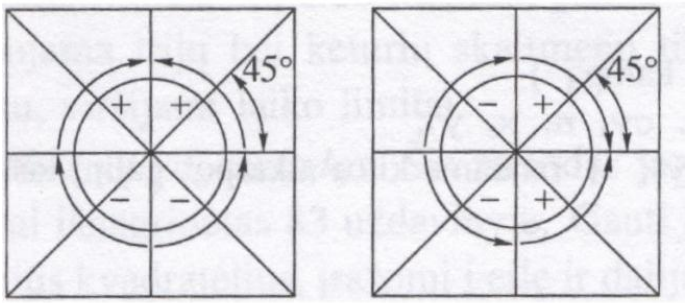
```
    { rasime atkarpos, išvestos iš kvadrato centro j tašką (x, y), ilgį }
    ilg2 := ilgis (x - cx, y - cy);
    for i := 1 to kiek do begin
      ilgi := ilg2;
      alfa := i * min; { padidiname kampą }
      kraštas (alfa, TIKSLUMAS, cx, cy, n, x, y);
      ilg2 := ilgis (x - cx, y - cy); { randame kitos atkarpos galinį tašką }
      p := p + ilgi * ilg2
    end;
    p := p * 1/2 * sin (min);
    kiek := kiek * 2
  until sutampa (senas, p, SKAIT);
  plotas := p
end; { plotas }

  var kvadx, kvady, n: integer;
begin
  dydis (kvadx, kvady, n);
  spausdinti (plotas (kvadx, kvady, n), SKAIT)
end.
```

Siek tiek paaiškinsime programą. Ploto tikslumas priklauso nuo dviejų parametrų; nuo konstantos TIKSLUMAS ir nuo to, į kelias dalis padalijamas skritulys (t. y. nuo *kiek* reikšmės). Derinant programą paaiškėjo, kad tikslumas nuo minėtos konstantos labai mažai priklauso (nebent pasirinktumėte labai didelę konstantą). Mes pasirinkome mažesnę, tačiau ją prilyginus 0,1, tikslumo pakaktų. Taigi valstybės ploto artinius skaičiuojame didindami trikampių, į kuriuos dalijama valstybė, skaičių.

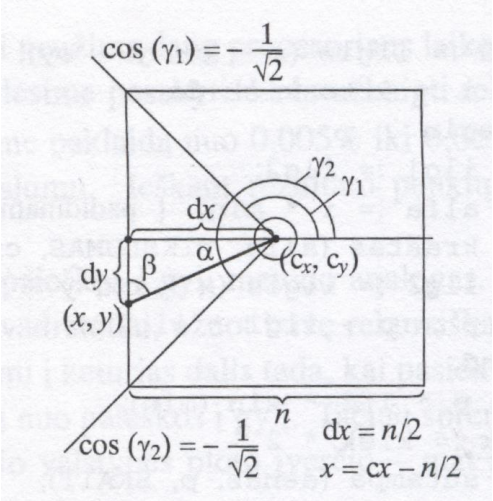
Paaiškinsime trūkstamas funkcijas ir procedūras. Funkcija *ilgis* apskaičiuoja atkarpos ilgį, remiantis Pitagoro teorema.

```
function ilgis (a, b: real): real;
{ randa stačiojo trikampio įžambinės ilgį, kai žinomi jos statiniai }
begin
ilgis := sqrt(a * a + b * b)
end; { ilgis }
```

Norint rasti tašką, kuriame iš kvadrato centro išvestas spindulys kerta daugiakampį, p *ima reikia rasti tašką, kuriame tas* pats spindulys kerta kvadrato kraštinę. Pastarąjį tašką randa procedūra *galas.* Pirmiausia ji nustato, kurią kvadrato kraštinę kirs spindulys. Iš algebros žinome, kad sin(45°) = sin(90 + 45°) = cos(45°) = cos(-45°) = 1/√2. *Taip* pat žinome, kuriuose apskritimo ketvirčiuose sinusas ir kosinusas yra teigiamas, kuriuose -neigiamas (žr. a-b pav.), kuriuose jis didėja, kuriuose - mažėja.



Sakykime, kad spindulys išvestas iš kvadrato centro kampu α. Jei suskaičiavę šio kampo kosinusą gauname, kad kosinusas neigiamas ir mažesnis už 1/√2, tai galime teigti, kad spindulys kirs kairiąją kvadrato kraštinę (žr. pav. aukščiau). Taigi taško, kuriame spindulys kirs kvadratą, koordinatė *x* lygi (*cx* – n/2), čia *cx* – kvadrato apatinio kairiojo kampo koordinatė *x*, *n* – kvadrato kraštinės ilgis. Ieškomo taško koordinatė *y* lygi (*cy–n/2*×tg(α) (žr. pav.).




```
procedure galas (alfa, cx, cy: real; n: integer;  { kvadrato kraštinė }
                 var gx, gy: real);
{ randa tašką, kuriame spindulys, išvestas iš kvadrato centro (c*, cy) kampu a, kerta kraštinę }
  var sinus, cosin, pi_4: real;
begin
  sinus := sin(alfa); cosin := cos(alfa);
  pi_4 := 1 / sqrt(2);  { sin(45) = cos (45), kampas matuojamas laipsniais }
```

```
  if cosin <= -pi_4  { ieškomas taškas yra kairiojoje kvadrato kraštinėje }
     then begin
            gx := cx - n/2;
            gy := cy - n/2 * sinus/cosin
          end
    else if cosin >= pi_4 then  { taškas dešiniojoje kraštinėje }
         begin
            gx := cx + n/2;
            gy := cy + n/2 * sinus/cosin
         end
    else if sinus <= - pi_4 then  { taškas apatinėje kraštinėje }
         begin
            gy := cy - n/2;
            gx := cx - n/2 * cosin/sinus
         end
    else begin  { taškas viršutinėje kraštinėje }
            gy := cy + n/2;
            gx := cx + n/2 * cosin/sinus
         end
end;  { galas }
```

Taško, kuriame spindulys, išvestas iš kvadrato centro kampu α, kerta valstybę, ieškome dvejetainės paieškos metodu.

```
procedure kraštas (alfa, epsilon, cx, cy: real; n: integer;
                    { kvadrato kraštinė } var x, y: real);
{ randa tašką, kuriame spindulys, išvestas iš kvadrato centro (cx, xy) kampu α, kerta daugiakampį )
  var gx, gy, vx, vy: real;
begin
  x := cx; y := cy;  { spindulio pradžia bus kvadrato centre; ji priklauso valstybei }
  galas (alfa, cx, cy, n, gx, gy);  { randame tašką, kuriame spindulys kerta kraštinę }
  if valstybė (gx, gy)
  then begin  { jei atkarpos galas priklauso valstybei }
          X := gx; y := gy
       end
  else repeat  { priešingu atveju ieškome reikiamo taško }
         vx := (x + gx)/2;
         vy := (y + gy)/2;  { randame atkarpos vidurio tašką }
```

```
        if valstybė (vx, vy)
           then begin { jei vidurio taškas priklauso valstybei }
                  x := vx;
                  y := vy
                end
        else begin { jei nepriklauso }
                  gx := vx;
                  gy := vy
             end
     until ilgis (gx - x, gy - y) < epsilon
end; { kraštas }
```

Funkcija *sutampa* ir procedūra *spausdinti* yra panašios: abi tikrina realiojo skaičiaus skaitmenis. Ar suapvalinus du duotus realiuosius skaičius sutampa pirmieji penki skaitmenys, tikrinama taip. Kablelis abiejuose skaičiuose perkeliamas į kairę pusę taip, kad pirmasis šių skaičių taptų mažesnis už 1. Pavyzdžiui, jei reikia palyginti skaičius 1456,279 ir 1456,300, abu skaičiai padalijami iš 104. Gauname 0,1456279 ir 0,1456300. Tuomet abu skaičiai padauginami iš 105 (nes reikia patikrinti, ar sutampa penki skaitmenys). Gauname 14562,79 ir 14563,00. Tuomet jie *suapvalinami* iki sveikųjų skaičių ir palyginami. Mūsų atveju gauname: 14563 = 14563.

```
function sutampa (senas, naujas: real; ska: integer): boolean;
{ patikrina, ar suapvalinus sutampa duotų skaičių pirmieji ska skaitmenų }
{ 0 <= senas <= naujas }
  var i, daug, laipsn: longint;
begin
  daug := 1;
  while naujas > daug do
     daug := daug * 10;
     laipsn := 1;
     for i := 1 to ska do
        laipsn := laipsn * 10;
        { skaičius pertvarkome taip, kad skaičiaus naujas sveikąją dalį sudarytų ska skaitmenų }
        naujas := naujas / daug * laipsn;
        senas := senas / daug * laipsn;
        sutampa := round (senas) = round (naujas)
end; { sutampa }
```

Norėdami atspausdinti skaičių, pirmiausia suskaičiuojame, kiek skaitmenų sudaro jo sveikąją dalį, o pagal tai - kiek skaitmenų turi būti spausdinama po kablelio. Jei po kablelio nereikia spausdinti nė vieno skaitmens, tuomet suapvalinama ir atspausdinama skaičiaus

sveikoji dalis. Jei reikia, kreipinyje į procedūrą *writeln* nurodomas skaitmenų po kablelio skaičius.

```
procedure spausdinti (sk: real; tiksi: integer);
{ spausdina plotą nurodytu tikslumu }
  var kiek, po, i, skaičius: longint;
      tarp: real;
begin
  kiek := 1; tarp := sk;
  while tarp > 10 do { suskaičiuojame, kiek skaitmenų sudaro sveikoji dalis }
    begin
      tarp := tarp / 10;
      kiek := kiek + 1
    end;
  po := tiksi - kiek; { tiek skaitmenų reikia atspausdinti po kablelio }
  if po = 0    { sveikoji dalis turi tiksi skaitmenų }
    then writeln (round (sk))
  else if po > 0
    then writeln (sk:l:po)
  else begin { reikia apvalinti sveikąją dalį }
    for i := 1 to kiek do
      sk := sk/10;
    skaičius := round (sk);
    for i := 1 to kiek do
      skaičius := skaičius * 10;
    writeln (skaičius)
  end
end; { spausdinti }
```

Deja, net ir šis algoritmas nepateikia atsakymo reikiamu tikslumu. Jei atsakymai dideli, jis randa plotą reikiamu tikslumu. Jei atsakymas mažesnis skaičius (tenka ieškoti trečio ar ketvirto skaitmens po kablelio), algoritmas veikia labai ilgai arba gauna didelę paklaidą. Šeštasis testas (kurio neįveikia šis algoritmas) olimpiados metu nebuvo pateiktas.

Pastaba. Vertinant paklaidą, kurią gauna aukščiau pateikti sprendimo metodai, buvo testuojama tik su tokiais pradiniais duomenimis, kai daugiakampio plotas yra ne mažesnis kaip 100 (t. y. reikia ieškoti ne daugiau kaip dviejų skaitmenų po kablelio). Vertinant tokiu pačiu būdu, santykinė paskutiniojo algoritmo paklaida lygi nuliui. Pateikiame modulio TERRAINC.PAS pradinį tekstą.

```
unit terrainc;

interface

  procedure dydis (var kordx, kordy, dydis: integer);
  function valstybė (kordx, kordy: real): boolean;

implementation

  uses dos;
  const MAX = 100;  { maksimalus daugiakampio viršūnių skaičius }
  ype taškas = record
                    x, y: real;
                end;
  viršūnės = array [1..MAX] of taškas;  { žemiau surašyti globalieji kintamieji }
  var x, y, n,   { kvadrato apatinio kairiojo kampo koordinatės ir kvadrato dydis }
      vsk: integer;  { daugiakampio viršūnių skaičius }
      v: viršūnės;  { jo koordinatės }
      p: real;   { tikslus daugiakampio plotas }
      kviet: longint;
      viskas: boolean;  { ar įvyko klaida }

procedure daugiakampio_plotas (vsk: integer; v. viršūnės;
                               var p: real);
{ randa iškilojo daugiakampio plotą }
  var x, y, xl, yl, x2, y2, per, a, b, c: real;
      i: integer:
begin
  p := 0;
  x := v[l].x; y := v[l].y;
  for i := 2 to vsk-1 do begin
    { atskiriame trikampį nuo daugiakampio }
    xl := v[i].x;    yl := v[i].y;
    x2 := v[i+l].x;  y2 := v[i+l].y;
    { apskaičiuojame trikampio kraštines }
    a := sqrt (sqr (x - xl) + sqr (y - yl));
    b := sqrt (sqr (x - x2) + sqr (y - y2));
    c := sqrt (sqr (xl - x2) + sqr (yl - y2));
    per := (a + b + c)/2;  { perimetro pusė }
    p := p + sqrt (per * (per - a) * (per - b) * (per - c))  { plotas }
  end
end;  { daugiakampio_plotas }
```

```
procedure dydis (var kordx, kordy, dydis: integer);
begin
  kordx := x; kordy := y;
  dydis := n
end; { dydis }

function valstybė (kordx, kordy: real): boolean;

  function dešinė_pri (t, tl, t2: taškas): boolean;
  { tikrina, ar taškas t yra dešinėje pusėje (arba priklauso) tiesės, }
  { išvestos per taškust t1 ir t2 tiesės kryptis iš t1 į t2 }
 begin
  dešinė_pri := (t.x - tl.x) * (t2.y - tl.y) -
                (t.y - tl.y) * (t2 .x - 11.x) >= 0
end; { dešinė_pri }

var t: taškas;
    gerai: boolean;
    i: integer;
begin
  kviet := kviet + 1; { padidiname iškvietimų skaičių }
  t.x := kordx; t.y := kordy;
  gerai := true;
  for i := 1 to vsk-1 do
    gerai := gerai and dešinė_pri (t, v[i], v[i + l]);
    valstybė := gerai and dešinė_pri (t, v[vsk], v[l])
end; { valstybė }

procedure spausdinti (sk: real; tiksi: integer);
{ spausdina plotą (<= maxlongint) nurodytu tikslumu }
  var kiek, po, i, skaičius: longint;
  tarp: real;
begin
  kiek := 1; tarp := sk;
  while tarp > 10 do { suskaičiuojame, kiek skaitmenų sudaro sveikąją dalį }
    begin
      tarp := tarp / 10;
      kiek := kiek + 1
    end;
  po := tiksi - kiek; { tiek skaitmenų reikia atspausdinti po kablelio }
```

```
  if po = 0 { sveikoji dalis turi tiksi skaitmenų }
     then writeln (round (sk))
     else if po > 0
             then writeln (sk:l:po)
             else begin { reikia apvalinti sveikąją dalį }
                     for i := 1 to kiek do
                       sk := sk/10;
                     skaičius := round (sk);
                     for i := 1 to kiek do
                       skaičius := skaičius * 10;
                     writeln (skaičius)
                  end
end; { spausdinti }

procedure pabaiga; far;
  var rez: real;
begin
  writeln;
  writeln ('Iškvietimų skaičius: ', kviet);
  write ('Įveskite jūsų programos rastą plotą: ');
  readln (rez);
  writeln ('Santykinė paklaida procentais: ',
           abs(rez-p)/p*100:10:5)
end; { pabaiga }

var i, num: integer;
    byla, numeris: string;
    f: text;
begin
  writeln;
  writeln (' +------------------------------------ ++')',
  writeln;
  writeln ('Šiam uždaviniui parengti testai sunumeruoti nuo 0 iki 6 ');
  write ('Kurį testą norėsite vykdyti? ');
  readln (num);
  writeln;
  str (num, numeris); { perskaitomi pradiniai duomenys }
  byla := 'VALST'+ numeris + '.DAT';
  assign (f, byla); reset (f);
  readln (f, x, y, n);
  readln (f, vsk);
```

```
  for i := 1 to vsk do
    readln (f, v[i].x, v[i].y);
  close (f);
  daugiakampio_plotas (vsk, v, p); { randamas tikrasis daugiakampio plotas }
  writeln ('Tikrasis valstybės plotas lygus: ', p: 1: 5);
  write ('Suapvalintas valstybės plotas lygus: ');
  spausdinti (p, 5);
  kviet := 0;  { dar neįvykdytas nė vienas iškvietimas }
  write ('Jūsų apskaičiuotas plotas lygus: ');
  exitproc := @pabaiga
end.
```