Jonas Blonskis
Vytautas Bukšnaitis
Renata Burbaitė

ŠIUOLAIKIŠKAS ŽVILGSNIS Į PROGRAMAVIMO PAGRINDUS

Informacinių technologijų vadovėlis
pasirenkamajam kursui IX–X klasėse

Turinys

Skaityk ne tam, kad prieštarautum ir neigtum, ne tam, kad iškart patikėtum, ir ne tam, kad turėtum apie ką kalbėti, bet tam, kad mąstytum ir svarstytum.
F. Bekonas

Įvadas

Žavėdamiesi kompiuterio galimybėmis dažniausiai pamirštame, kad kompiuteris yra tik žmogaus sukurtos programos vykdytojas. Norint kompiuterį efektyviai taikyti praktinėje veikloje vien žinių apie kompiuterio darbo principus ir mokėjimo dirbti taikomąja programine įranga nebepakanka: pasitaiko problemų, kurių sprendimui pačiam naudotojui reikia sugalvoti sprendimo kelią (būdą) ir parašyti programą. Šiuolaikinės taikomosios programos (pvz., skaičiuoklė, tekstų rengyklė ir kt.) turi savo programavimo priemones, kurios leidžia išplėsti jų galimybes, pritaikyti jas konkrečiai veiklos sričiai ar užduočiai spręsti. Programavimo žinios, įgūdžiai ir gebėjimas kurti nesudėtingas programas yra labai naudingi bet kuriam kompiuterio naudotojui.

Išmokti programuoti galima tik pačiam kuriant programas. Todėl mokymąsi siūlome pradėti nuo praktikos darbų. Knygos skyriuje „Praktikos darbai“ rasite penkiolikos praktikos darbų scenarijus. Jie visi pateikiami FreePascal 0.6.4a versijoje. Kiekvienas scenarijus aprašo, kaip žingsnis po žingsnio kuriama programa. Labai svarbu įprasti, parašius nors ir keletą programos eilučių, skirtų kokiam nors veiksmui atlikti, patikrinti programos darbą. Tik įsitikinus, kad programa dirba ir ji dirba teisingai, galima ją rašyti toliau. Scenarijuose vartojamos sąvokos ir terminai glaustai paaiškinti knygos skyriuje „Pagrindinės struktūrinio programavimo sąvokos“. Jų mokytis mintinai nėra tikslo, tačiau suvokti jų prasmę būtina, norint sėkmingai dirbti su kitais knygos skyriais, kita literatūra.

Kiekvieno praktikos darbo pradžioje rasite sąrašą žinių ir gebėjimų, kuriuos įgysite atliekant tą praktikos darbą. Čia tai pat pateikiamos nuorodos į skyrių „Pagrindinės Paskalio kalbos konstrukcijos“ ir „Kai kurie praktikos darbuose naudojami algoritmai“ skyrelius. Skyriuje „Pagrindinės Paskalio kalbos konstrukcijos“ pristatomos pagrindinės Paskalio programavimo kalbos priemonės ir konstrukcijos. Tai žinyno pobūdžio informacija. Skyriuje „Kai kurie praktikos darbuose naudojami algoritmai“ aprašomi klasikiniai algoritmai, naudojami daugeliui įvairaus tipo praktikos užduočių spręsti. Šių skyrių informacija bus naudinga tiems, kas norės pasitikslinti ar pagilinti žinias ir įgūdžius atliekant konkretų praktikos darbą.

Kiekvienas scenarijaus žingsnis – tai tam tikra veikiančios programos versija. Toks scenarijų pateikimo būdas leis kiekvienam mokiniui dirbti individualiai, mokytojui konsultuojant ar padedant, jei reikia. Atlikus kiekvieną darbo žingsnį turime dirbančią, bet dar nebaigtą programą. Darbą galima tęsti kitą pamoką arba namuose nuo sekančio žingsnio.

Kiekvieno praktikos darbo pabaigoje gausu klausimų ir užduočių, kurie padės kiekvienam įsivertinti savo žinias ir įgūdžius. Jie nėra privalomi, tačiau siūlome juos panagrinėti ir atlikti. Jei kurie nors klausimai ar užduotys yra per sunkūs, nepraleiskite jų, pasistenkite juos įveikti išnagrinėję teorinę medžiagą, konsultuodamiesi su mokytoju ar su klasės draugais.

Pirmieji praktikos darbų scenarijai yra pakankamai detalūs, vėliau, programavimo įgūdžiams ir žinioms tvirtėjant, scenarijai nėra tokie išsamūs, paliekama daugiau laisvės saviraiškai. Pirmieji šeši praktikos darbai skirti įvadui į programavimą. Juos privalu atlikti. Kiti darbai skirti gautoms žinioms ir įgūdžiams tobulinti programuojant sudėtingesnes užduotis, todėl nebūtina jų visų atlikti. Praktikos darbų ciklas baigiamas pažintimi su Free Pascal grafikos priemonėmis. Šie darbai taip pat nėra privalomi ir skirti tiems, kurie nori išmokti gautus rezultatus ekrane pateikti vaizdžiai.

Mokiniams, jau turintiems pradinius programavimo įgūdžius, šalia praktikos darbų siūlome individualiai atlikti užduotis, pateiktas skyrelyje „Užduotys savarankiškam darbui“. Šiems mokiniams taip pat bus naudingi scenarijų papildymai ir teorinių skyrelių medžiaga, pažymėti žodžiu „Smalsiems“.

Besidomintiems programavimu knygos gale pateikiamas sąrašas nuorodų, kurios leis rasti internete daugiau informacijos rūpimais klausimais.

Tikimės, kad autorių siūlomas programavimo pagrindų mokymosi kelias bus jums įdomus, suprantamas ir naudingas.

Sėkmės!

Knygos autoriai

1. Pagrindinės struktūrinio programavimo sąvokos

1.1. Algoritmas, jo vykdymas, savybės

Kiekvienas iš mūsų kasdieninėje veikloje susiduriame su įvairiomis taisyklėmis, instrukcijomis, receptais, nurodymais. Pavyzdžiui, tai naudojimosi įvairiais įrenginiais, baldų surinkimo instrukcijos, patiekalų receptai, rali šturmano legenda ir pan.

Štai pavyzdys. Draugas kviečia jus į svečius pasidalyti atostogų įspūdžiais. Jis sako: „išėjęs iš namų pasuk į dešinę, paėjėk iki artimiausios autobuso stotelės, įlipk į autobusą Nr. 5, pavažiuok 3 stoteles, išlipk „Žvaigždžių“ stotelėje. Ten tave pasitiksiu“.

Panašios aprašymo taisyklės sudaromos ir matematikos, fizikos, chemijos uždaviniams spręsti. Naudodamiesi jomis nesunkiai išsprendžiame vieno ar kito tipo uždavinius.

Mokydamiesi gimtąją ir užsienio kalbas išmokstame pagrindines taisykles, kurias taikydami, sėkmingai įveikiame gramatikos subtilybes.

Ebu Abdullah Muhammed bin Musa al-Chwarizmi (apie 780-850 m.)

Pateikti pavyzdžiai yra skirtingo pobūdžio, tačiau turi ir bendrų bruožų:

- juos galima aprašyti atskirais aiškiais nurodymais, ką reikia daryti;
- jie turi tam tikrus pradinius duomenis (pvz.: patiekalui pagaminti reikalingi produktai, spintos sudedamosios dalys ir t. t.);
- gaunamas tam tikras rezultatas (pagaminamas patiekalas, sukonstruojama spinta ir t. t.).

Išvardyti bruožai apibūdina algoritmo (lot. *algorithmus* – pagal Vidurinės Azijos matematiko al-Chwarizmi pavardės lotynišką formą Algorithmi) sąvoką. Šiuo atveju veiksmus, algoritmu vadovaujantis, atlieka žmogus.

Algoritmu vadinami aiškūs vienareikšmiai nurodymai (sakiniai), kaip turint tam tikrus pradinius duomenis gauti reikiamus rezultatus.

Algoritmo sąvoka yra viena iš pagrindinių matematikos ir informatikos sąvokų. Pirmieji algoritmai apibūdino veiksmus, atliekamus su dešimtainės skaičiavimo sistemos skaičiais. Vėliau algoritmo sąvoka pradėta vartoti apibūdinant veiksmų seką, kurią reikia atlikti norint išspręsti uždavinį. Šioje knygoje nagrinėjami tik matematinio pobūdžio algoritmai.

Pradiniai duomenys – tai iš anksto žinomos reikšmės (paprasčiausiu atveju skaičiai), būtini veiksmams atlikti. Pavyzdžiui, norint apskaičiuoti stačiakampio plotą, būtina žinoti jo ilgį ir plotį.

Rezultatai – tai skaičiavimų metu gautos reikšmės.

Tarpiniai rezultatai – tai skaičiavimų metu gautos reikšmės, kurios naudojamos tolesniuose veiksmuose. Jie taip pat reikalingi pasitikrinti, ar pilnai parašyta programa, ar jos dalis, veikia gerai. Programos naudotojui jie nėra pateikiami.

Kiekvienas algoritmas skirtas tam, kad jį kas nors vykdytų. Algoritme aprašomi veiksmai yra skirti **vykdytojui**. Žodžio „algoritmas“ reikšmė artima žodžių „receptas“, „metodas“, „būdas“ reikšmei. Kiekvienam algoritmui būdingos tokios savybės:

- **Diskretiškumas**. Algoritmas suskaidomas į baigtinių veiksmų (žingsnių) seką. Tik atlikus vieną veiksmą galima pereiti prie kito.
- **Aiškumas**. Visi algoritmu aprašomi veiksmai turi būti suprantami vienareikšmiškai bet kuriuo naudotojo. Algoritmas yra kuriamas „nemąstančiam“ vykdytojui. Kad suprasti, ką reiškia ši sąvoka, prisiminkime rašytojo V. Petkevičiaus pasakos vaikams „Sieksnis, Sprindžio vaikas“ vieną epizodą. Tėvai, išleisdami sūnų į mokyklą, sako: „Eidamas dairykis. Neskubėk lėkti per kelią, palauk, kol mašina pravažiuos, kad po ratais nepakliūtum“. Visą dieną Sieksnio iš mokyklos nesulaukęs tėvas išėjo jo ieškoti ir priėjęs kryžkelę pamatė ilgšį pagriovy stovintį ir garsiai žliumbiantį. Tėvo paklaustas, kas atsitiko, Sieksnis atrėžė: „Sakei, kad neskubėčiau, palaukčiau, kol mašina pravažiuos. Aš visą dieną laukiu, o jos kaip nėr, taip nėr“. Ir į mokyklą tą dieną Sieksnis taip ir nenuėjo. Kitą

dieną tėvas sūnų pats per kryžkelę pervedė ir paleido: „Žiūrėk man, eik ir nesidairyk!“. Sieksnis, tiesiai eidamas, pirmiausiai malūną priėjo ir visą dieną malūnininkui talkino. Trečią dieną tėvas pats sūnų į mokyklą nutempė.

Šiame epizode Sieksnį galime laikyti nemąstančiu algoritmo vykdytoju, kuris tiesiogiai (nemąstydamas ir neanalizuodamas) vykdė tėvo nurodymus.

- **Rezultatyvumas**. Atlikus baigtinį algoritmo veiksmų (žingsnių) skaičių turi būti gaunamas rezultatas. Vienas iš galimų rezultatų gali būti ir toks, kai uždavinys neturi sprendinių.
- **Baigtumas.** Rezultatą gauname įvykdę baigtinį algoritmo komandų skaičių.
- **Universalumas**. Naudojant algoritmą galima spręsti visus to tipo uždavinius, t. y. kiekvienam pradinių duomenų rinkiniui gaunamas teisingas rezultatas.

Yra daug uždavinių, kuriems spręsti nėra tikslių algoritmų. Pavyzdžiui, reikia pervežti mašinomis daug įvairaus dydžio tuščių dėžių, kurias galima dėti vienas į kitas. Mašinų su kroviniu skaičius turi būti kuo mažesnis. Net žmogus, kuris pasižymi galimybe intuityviai spręsti tokią užduotį, ne iš karto gaus geriausią rezultatą. Jeigu dėžių daug, o į mašinas telpa nevienodas dėžių skaičius, tuomet neįmanoma perrinkti visų galimų sprendinių ir tenka pasitenkinti kuriuo nors geresniu rezultatu. Tokio tipo uždaviniams spręsti kuriami algoritmai, vadinami **euristiniais**. Taikant euristinius algoritmus tinkamu rezultatu laikomas tas, kuris tenkina žmogaus poreikius.

Kiekvienas algoritmas gali būti aprašomas skirtingais būdais:

- Algoritmo užrašymas **žodžiais** taikomas tuomet, kai norima labai aiškiai nurodyti atliekamus veiksmus. Užrašant algoritmą šiuo būdu komandos gali būti numeruojamos, arba veiksmai aprašomi kaip pasakojimas.
- Algoritmo užrašymas **grafine forma** yra daug vaizdesnis už žodinį aprašymą. Labiausiai paplitę du grafiniai algoritmų vaizdavimo būdai: **simbolinės** (**blokinės**) **schemos** ir **struktūrogramos**.

  Simbolinėse schemose ir struktūrogramose naudojami grafiniai simboliai apibrėžia tam tikro tipo komandą. Simbolinėse schemose grafinius simbolius jungiančios linijos rodo, kokia tvarka atliekami veiksmai. Sutarta, kad linijos eina iš viršaus į apačią ir iš kairės į dešinę. Visais kitais atvejais linijos gale braižoma rodyklė, nurodanti tolesnių veiksmų kryptį. Struktūrogramoje komandų vykdymo tvarka nusakoma pačiais grafiniais simboliais.
- Algoritmai gali būti užrašomi **pseudokodu**. Šis būdas skirtas žmogui. Pseudokode vartojami žodžiai, artimi natūraliai kalbai. Pseudokodas vartojamas algoritmams užrašyti ten, kur aiškinama algoritmo esmė – vadovėliuose, straipsniuose, nes juo algoritmą galima užrašyti trumpiau ir suprantamiau.

Pavyzdžiui, skaičiaus `a` kėlimo `n`-tuoju laipsniu algoritmas užrašytas:

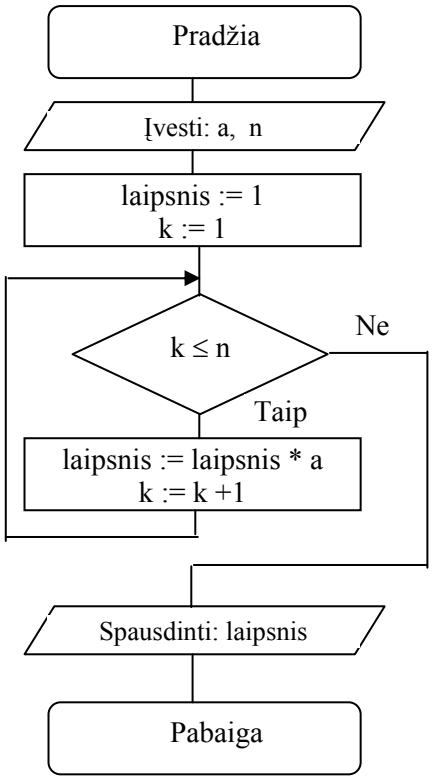
- žodžiu

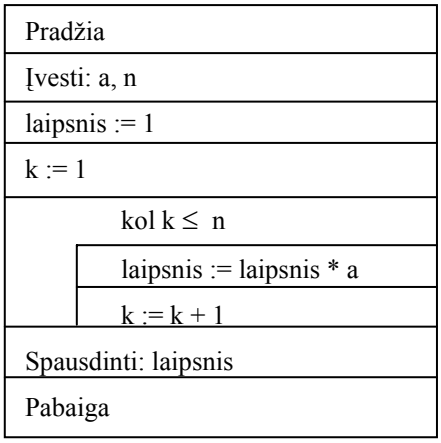
1. Sužinoti `a` ir `n` reikšmes.
2. Skaičiuoti `kartai = 1`.
3. Skaičiuoti `laipsnis = 1`.
4. Jeigu `kartai <= n`, tuomet vykdyti 5 žingsnį, kitaip – 8 žingsnis.
5. Skaičiuoti `laipsnis = laipsnis × a.`
6. Skaičiuoti `kartai = kartai + 1.`
7. Pereiti prie 4 žingsnio.
8. Rezultatas yra `laipsnis` reikšmė.

- pseudokodu

```
įvesti: a, n;
laipsnis := 1;
kartoti n kartų
   laipsnis := laipsnis × a;
spausdinti: laipsnis.
```

- grafine forma (simboline schema ir struktūrograma)





Daugelyje straipsnių ir mokomųjų knygų algoritmams užrašyti naudojamas pseudo Paskalio programavimo kalba, kuri nuo tikrosios skiriasi labai nedaug. Taip atsitiko, nes Paskalio programavimo kalba buvo kuriama kaip galint panašesnė į šnekamąsias kalbas, tikintis, kad pradedantiems mokytis programuoti bus lengviau.

1.2. Programa, programavimo kalba, struktūrinis programavimas

Algoritmas, užrašytas kuria nors programavimo kalba, yra vadinamas **programa**.

Programavimo kalba – skirta algoritmams užrašyti. Programavimo kalba artima šnekamajai, nes ji skirta programuotojui. Programavimo kalbos abėcėlė skirta kalbos konstrukcijoms sudaryti: vardams, aprašams, sakiniams. Kalba, kaip ir šnekamoji, turi savo sintaksę ir semantiką. Programavimo kalbos kiekvienos konstrukcijos, kiekvieno žodžio, kiekvieno sakinio prasmė vienareikšmė ir baigtinė. Programavimo kalbų yra labai daug. Nuo pirmosios programavimo kalbos, sukurtos 1954 m., iki dabar yra suskaičiuojama per 2500 kalbų.

Šioje knygoje programoms kurti naudosime Paskalio programavimo kalbą, kuri artima anglų kalbai. Tačiau norime pabrėžti, kad pati programavimo kalba yra tik priemonė programavimo įgūdžiams susidaryti, bet ne mokymosi tikslas. Stengiamės naudoti tas konkrečios kalbos konstrukcijas, kurios yra bendros ar analogiškos kitose programavimo kalbose.

Programavimas – tai procesas, kuris apima šiuos etapus:

- užduoties parengimą,
- užduoties analizę,
- užduoties skaidymą į dalis,
- sprendimo metodų parinkimą bei sukūrimą,
- duomenų struktūrų parinkimą bei sukūrimą,
- algoritmų sudarymą,

- programos teksto rašymą, derinimą, testavimą.

Struktūrinio programavimo, dar kartais vadinamo procedūriniu, pradžia yra laikoma XX amžiaus aštuntojo dešimtmečio pabaiga. Struktūrinio programavimo esmė labai paprasta: skaldyk ir valdyk. Kiekvienas uždavinys išskaidomas į smulkesnes dalis, kurios programuojamos kaip atskiri uždaviniai. Kiekviena tų dalių skirta vienai griežtai apibrėžtai veiksmų sekai atlikti.

Programavimo terpė (aplinka) – tai aparatinių ir programinių priemonių visuma, skirta naujų programinių priemonių kūrimui. Paprasčiausios programavimo terpės, pavyzdžiui, Free Pascal kalbos, turi teksto rengyklę programų tekstų (ir duomenų) failų kūrimui ir taisymui, kompiliatorių, programos redaktorių, priemones programai derinti ir vykdyti. Programų tekstus galima rašyti ir kitomis priemonėmis, pavyzdžiui NotePad, tačiau terpė turi papildomų funkcijų, kurių nėra kitose tekstų rengyklėse. **Kompiliatorius,** tai programa, kuri verčia parašytą programos tekstą į kompiuteriui suprantamą kalbą. Jeigu randa rašybos (sintaksinių) klaidų (angl. Error), tai vertimą nutraukia nurodydamas rastas klaidas.

1.3. Programos struktūra

Programos struktūrą (sandarą) nulemia programavimo kalba. Vienos kalbos yra griežtos struktūros, kitose yra leistinos alternatyvos. Programos struktūrai turi įtakos ir programavimo technologija.

Vadovėlyje pateiktos programos parašytos Paskalio programavimo kalba, laikantis struktūrinio programavimo technologijos reikalavimų. Programos tekstas visada skirstomas į dvi dalis:

```
program Vardas;
aprašymai
begin
 veiksmai
end.
```

- naudojamų priemonių (konstantų, kintamųjų, naujų tipų, naudojamų bibliotekų) aprašymas;
- veiksmų sakiniai.

1.4. Programavimo stilius ir kultūra

Kaip rašytojams būdingas skirtingas literatūrinis stilius, taip ir programų autoriai turi individualų programavimo stilių. Kiekvienas programuotojas ar programuotojų grupė turi savitą požiūrį į programavimą, mėgstamas programavimo kalbas, naudoja skirtingai programavimo priemones. Programuotojo individualybę išreiškiančius savitumus įprasta vadinti **programavimo stiliumi**. Šios knygos autoriai laikosi tokių esminių programavimo stiliaus principų:

- kintamųjų aprašai grupuojami pagal paskirtį. Jei programoje yra procedūrų ar funkcijų, pagrindinės programos kintamieji aprašomi prieš programos **`begin`**;
- paprastų kintamųjų (saugančių skaičių, simbolį ar loginę reikšmę) vardai pradedami mažąja raide;
- Paskalio kalbos baziniai žodžiai (**`program`**, **`var`**, **`begin`**, **`end`** ir pan.) visada rašomi mažosiomis raidėmis. Šie žodžiai vartojami tik pagal Paskalio kalboje jiems suteiktą prasmę, t. y. nepervardijami;
- kiekvienas žodis programos, procedūros ir funkcijos varde pradedamas didžiąja raide (pvz.: `ReadLn`, `WriteLn`);
- kiekvienas programos sakinys baigiamas kabliataškiu.
- jei programoje yra kuriama procedūra ar funkcija, ji skirta vienam griežtai apibrėžtam veiksmui atlikti (pvz.: duomenims įvesti, vidurkiams apskaičiuoti, rezultatui išvesti).

Programuojant sudėtingą užduotį, labai dažnai jos sprendimas padalijamas į atskiras dalis, kurias realizuoja skirtingi programuotojai. Todėl kiekvienam iš jų svarbu parašyti programą tokiu būdu, kad ją nesudėtingai suprastų kiti, be to visas užduotis nesudėtinga būtų susieti, pataisyti arba panaudoti kitiems uždaviniams spręsti. Todėl programos tekstas turi būti aiškus, vaizdus, lengvai skaitomas ir suprantamas bet kuriam naudotojui. Tekstas lengviau suprantamas, kai jis patogiai išdėstytas lape, eilutės neperkrautos informacija, yra įtraukos, tarpai, komentarai.

Esminės programos teksto išdėstymo taisyklės:

- programos dalims išskirti paliekamos tuščios eilutės arba komentarų eilutės, sudarytos tik iš minuso ar kitokių simbolių;

- pavaldumui išryškinti daromos įtraukos (pradedama rašyti toliau nuo krašto):
  - kartojimo sakiniuose,
  - sakinių blokuose **begin** ... **end**;
- skaitomumui pagerinti sąlygos sakinyje ir sakinių bloke lygiuojami vertikaliai šie žodžiai:
  - **then**
    ...
    else
  - **begin**
    ...
    end

Kad programa būtų aiški, lengvai ir greitai suvokiama bet kuriam naudotojui programos tekstas turi atitikti loginę algoritmo struktūrą ir veiksmų hierarchiją. Rašant programą, reikėtų laikytis tokių **programavimo kultūros** taisyklių:

- algoritmą reikia parinkti tokį, kad jis kuo geriau tiktų duotam uždaviniui spręsti, būtų aiškus, trumpas ir logiškai pagrįstas;
- tekstą rašyti laikantis bendrų rašybos taisyklių. Tarp žodžių ir po skyrybos ženklo palikti bent vieną tarpą. Prieš „:“ ir „:=“ ženklus reikėtų palikti tarpą;
- kintamųjų, konstantų, tipų, procedūrų, funkcijų vardai turi atspindėti juos atitinkančių objektų prasmę, tačiau neturi būti labai ilgi;
- komentarai turi susieti uždavinio aprašymą su algoritmu;
- jei programoje naudojamos savarankiškos algoritmo dalys, tai jos turi būti išreiškiamos kuo mažiau tarpusavyje susijusiomis procedūromis bei funkcijomis.

Tekstą tikslinga rašyti taip, kad atskiros dalys pagal prasmę būtų nesunkiai atpažįstamos ir suvokiamos. Tam reikia jį išdėstyti lape lyg kokį piešinį, kur reikia paliekant didesnes įtraukas.

Pavyzdžiui, šitaip parašytą tekstą sunku skaityti:

```
var pirmas,antras,trecias:real;
begin
Read(pirmas,antras,trecias);
WriteLn(pirmas,antras,trecias);
...
end.
```

Patartume jį rašyti taip:

```
var   pirmas, antras, trecias : real;
begin
   Read(pirmas, antras, trecias);
   WriteLn(pirmas, antras, trecias);
   ...
end.
```

Dažnai programos tekstą padeda suprasti komentarai, kurie skirti programos naudotojui (programuotojui) ir visai neturi įtakos programos vykdytojui (kompiuteriu). Komentarai neturi trukdyti skaityti programą, t. y. jie turi būti trumpi, taikliai papildyti programą, bet jos neužgožti. Komentarai gali sudaryti apie 20 % visos programos apimties.

Rašant komentarus, rekomenduojama:

- programos pradžioje po antrašte (arba prieš ją) nurodyti programos autorių, paskirtį, paskutinio taisymo datą, versijos numerį, taikymo sritis, užduoties sprendimo būdą, programinius apribojimus;
- kintamųjų aprašuose nurodyti jų paskirtį;
- prieš ciklus, procedūras, funkcijas, sakinių blokus nurodyti jų paskirtį;
- paaiškinti neaiškios paskirties veiksmus, pavyzdžiui, sąlygos sakinyje komentuoti atskirai `then` ir `else` dalis;

- jei komentarai užima atskirą eilutę, nuo programos teksto juos galima atskirti tuščiomis eilutėmis.

Pavyzdžiui, šį programos fragmentą sunku skaityti:

```
var  ugis1, ugis2: real; // mokinių ūgiai
     metai1, metai2: integer; // mokinių gimimo metai
     amzius1, amzius2: integer;// mokinių amžiai
     svoris1, svoris2: real; // mokinių svoriai
```

Kad programa būtų aiškesnė ir ją būtų lengviau skaityti, aprašant kintamuosius reikėtų juos grupuoti pagal prasmę ir lyginti pagal pozicijas, o komentarus pateikti darant vienodas įtraukas. Anksčiau pateiktą programos fragmentą siūlome rašyti taip:

```
var                 // Pirmojo mokinio duomenims:
     ugis1  : real;         // ūgiui,
     metai1 : integer;      // gimimo metams,
     amzius1: integer;      // amžiui,
     svoris1: real;         // svoriui.
                    // Antrojo mokinio duomenims:
     ugis2  : real;         // ūgiui,
     metai2 : integer;      // gimimo metams,
     amzius2: integer;      // amžiui,
     svoris2: real;         // svoriui.
```

Taisyklingai rašydami, prarasime šiek tiek laiko, tačiau laimėsime kur kas daugiau – mažiau padarysime klaidų, greičiau jas pastebėsime.

1.5. Dialogas tarp programos ir naudotojo

Tarp naudotojo ir kompiuterio dažnai vykdomas ***dialogas***. Pavyzdžiui, įvedant duomenis klaviatūra, būtina išvesti į ekraną pranešimą, kad kompiuteris laukia duomenų įvedimo, paaiškinti, ką naudotojas turi įvesti ir, jeigu reikia, nurodyti įvedimo formą ir eilės tvarką. Pavyzdžiui, toks programos fragmentas įpareigoja laukti, kol įvesime vieną skaičių (tačiau neaišku, realųjį ar sveikąjį):

```
...
WriteLn('Įveskite apskritimo spindulį');
ReadLn(R);
...
```

Dialogo tekstas, pranešimai, nurodymai turi būti lakoniški, informatyvūs ir vienareikšmiškai suprantami. Pranešimai neturi būti susiję su programos kintamųjų vardais, jeigu jie nėra informatyvūs, pavyzdžiui:

```
...
WriteLn('Įveskite a reikšmę: ');
ReadLn(a);
...

...
WriteLn('Įveskite prekės kainą: ');
ReadLn(a);
...
```

Pirmajame dialoge neaišku, kas yra `a`. Antrajame dialoge aišku, jog reikia įvesti prekės kainą, nebūtina žinoti, kad kaina programoje turi vardą `a`.

Projektuojant programos struktūrą, reikėtų atskirti dialogo ir skaičiavimų dalis, nes jų skirtingi tikslai. Dialogas skirtas bendravimui su programos naudotoju, o skaičiavimai apdoroja duomenis. Dialogo tikslas – gauti tam tikrą informaciją. Jis gali būti dažnai keičiamas, pertvarkomas, papildomas. Tai gali būti susiję ne tik su kalbos tobulinimu, bet ir su techninių galimybių pasikeitimu.

1.6. Bendras programos kūrimo planas

1. ***Pradinių duomenų ir būsimų rezultatų analizė***. Išsiaiškinama, kiek yra pradinių duomenų, kiek bus rezultatų, kokie jų tipai, kokia tvarka juos pateikti.

2. ***Uždavinio sprendimo idėja*** – tai mintis (sumanymas), kaip spręsti uždavinį. Idėjos teisingumu galima įsitikinti modeliuojant programos veiksmus su įvairiais kontroliniais duomenimis. Žinoma, pirmiausia reikia patikrinti idėją su sąlygoje pateiktais duomenimis siekiant gauti rezultatus, sutampančius su sąlygoje pateiktais rezultatų pavyzdžiais.
   Jeigu modeliuojant negaunama reikiamų rezultatų arba paaiškėja, kad sprendimo kelias pernelyg sudėtingas, neracionalus, tai ieškoma kito būdo uždaviniui spręsti. Būtų gerai, kad tik įsitikinus, jog sugalvotas sprendimo būdas teisingas, būtų pradėta rašyti programa.
3. ***Duomenų struktūrų parinkimas pradiniams duomenims ir rezultatams įsiminti.*** Nuo šio žingsnio iš esmės priklauso programos apimtis, struktūra, algoritmas ir jo įgyvendinimas.
4. ***Algoritmo sukūrimas*** – tai uždavinio sprendimo idėjos įgyvendinimas. Dažniausiai tam pritaikomi žinomų matematinių uždavinių sprendimo būdai naudojant konkrečias duomenų struktūras. Algoritmo sudėtingumas paprastai priklauso nuo pasirinktų duomenų struktūrų. **Duomenų struktūros**, tai kompiuterio atmintinės paskirstymo būdas duomenims saugoti. Jeigu šiame etape paaiškėja, kad algoritmas pernelyg sudėtingas, tuomet reikia keisti duomenų struktūras arba net uždavinio sprendimo būdą. Jei metodo nerandama, tuomet tenka spręsti užduotį euristiškai.
5. ***Programos rašymas*** – tai sukurto algoritmo užrašymas konkrečia programavimo kalba. Programa rašoma struktūriniu principu (nedidelėmis dalimis). Pageidautina kiekvieną dalį pirmiausia atskirai suderinti, t. y. įsitikinti, kad jau turima programos dalis duoda reikiamus rezultatus, dirba teisingai. Tai leidžia programą rašyti žingsnis po žingsnio. Kai derinamo teksto dalys nedidelės lengviau ir greičiau gaunama galutinė programa.
6. **Testavimas.** Parašyta programa testuojama, kad įsitikintume, kad norimas rezultatas pasiekiamas visiems galimiems teisingiems pradinių duomenų rinkiniams. Programuotojas turėtų sukurti kuo įvairesnių pradinių duomenų rinkinių, kurie tikrintų įprastas bei ribines situacijas, sprendimo efektyvumą atmintinės kiekio ir vykdymo laiko požiūriu. Jei pradinių duomenų rinkiniai turi būti dideli, gali tekti parašyti programą, generuojančią pradinių duomenų rinkinius. Atkreipiame dėmesį, kad tai būtinas programos rašymo etapas, net ir per olimpiadas.

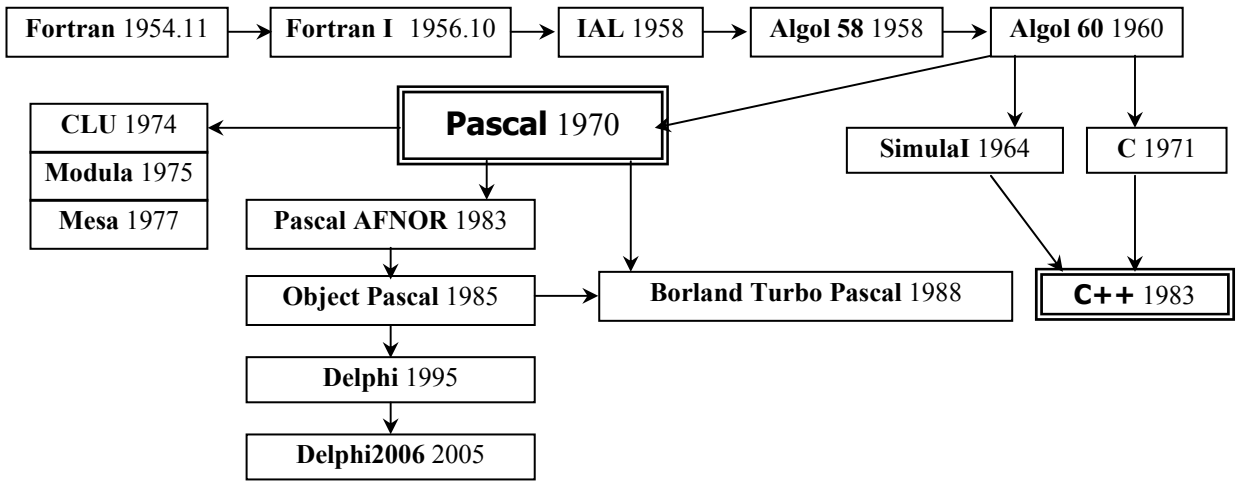
Šie etapai nėra atliekami nuosekliai. Kiekvienu momentu gali prireikti sugrįžti vienu ar keliais žingsniais atgal. Nepatartina skubėti, nes vėliau gali tekti rašyti programą iš naujo.

Programos naudotojui būtinas aprašymas, kaip naudotis programa. Aprašyme paprastai pateikiama tokia informacija:

- programos paskirtis, jos panaudojimui būtina aparatinė ir programinė įranga, diegimas (jeigu būtina);
- programos naudojimo instrukcija, kuri aprašo programos vykdymo galimas situacijas;
- pradinių duomenų pateikimo forma, kiek ir kokių duomenų reikia įvesti;
- rezultatų paaiškinimas, jų susiejimas su pradiniais duomenimis.

1.7. Trumpa Paskalio ir C++ programavimo kalbų atsiradimo istorija

Algoritmus galima užrašyti įvairiomis programavimo kalbomis. Programavimo kalbų sukurta labai daug ir atsiranda vėl naujų. Daugelio programavimo kalbų pagrindinės konstrukcijos yra panašios. Schemoje pateikta populiarių mokymuisi skirtų Pascal ir C++ programavimo kalbų supaprastinta trumpa atsiradimo istorija. Čia paryškinti dviguba linija blokeliai rodo dabar vartojamų mokymui Paskalio ir C++ programavimo kalbų sukūrimo datos. Nuo to laiko tos kalbos buvo modifikuojamos ir papildomos naujomis galimybėmis. Plačiau apie kalbas ir jų istorija galite rasti internetiniame puslapyje, kuris pateiktas knygos pabaigoje.


Fortran 1954.11
Fortran I 1956.10
IAL 1958
Algol 58 1958
Algol 60 1960
CLU 1974
Modula 1975
Mesa 1977
Pascal 1970
SimulaI 1964
C 1971
Pascal AFNOR 1983
Object Pascal 1985
Borland Turbo Pascal 1988
C++ 1983
Delphi 1995
Delphi2006 2005

2. Praktikos darbai

2.1. Įvadas į Free Pascal terpę

Atlikdami šį darbą susipažinsite su terpės Free Pascal darbo aplinka:

- sukursite darbo katalogą ir programos failą;
- pakeisite programos pavadinimą ir ją įrašysite darbo kataloge;
- sukompiliuosite ir įvykdysite paprasčiausią programą;
- programą redaguosite ir papildysite;
- išmoksite išvesti informaciją į ekraną naudodami rašymo procedūras `Write` ir `WriteLn`.

Nuorodos į Paskalio kalbos žinyną (psl.)		Nuorodos į algoritmų žinyną (psl.)	
3.1. Kintamasis, kintamojo reikšmė	94	4.1. Tiesinis algoritmas	103
3.2. Priskyrimo sakinys	94		
3.3. Duomenų įvedimas klaviatūra	95		
3.4. Rezultatų (duomenų) išvedimas ekrane	96		

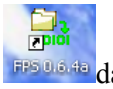
Tai pažintinis praktikos darbas. Atlikdami šį darbą susipažinsite su terpės `Free Pascal` darbo aplinka: meniu, programų kūrimo principais: redagavimu, kompiliavimu, vykdymu, sužinosite, kokie failai sukuriami ir kaip juos išsaugoti.
Atliekant šį praktikos darbą svarbu ne apskaičiuoti kokį nors konkretų rezultatą ar sukurti nurodytą programą, o tiesiog pajusti darbo `Free Pascal` aplinkoje ypatumus. Dirbdami galite atlikti ir daugiau veiksmų, negu čia parašyta. Gal ne visi veiksmai bus sėkmingi, tačiau įgysite patirties, kuri bus reikalinga atliekant kitus darbus.
Kiekviena `Free Pascal` programa sukuria keletą failų. Todėl prieš pradedant darbą patariame sukurti atskirą katalogą kiekvieno darbo failams saugoti.

Pirmas žingsnis. Darbo katalogo sukūrimas.

- Įprastomis `Windows` sistemos priemonėmis kuriame nors savo kompiuterio diske sukurkite bendrą katalogą, kuriame saugosite visus savo `Free Pascal` darbus. Katalogą galite pavadinti savo pavarde, pvz.: `Pavardenis`.
- Šio katalogo viduje sukurkite katalogą `Darbas1`, skirtą mūsų pirmajam (įvadiniam) darbui.

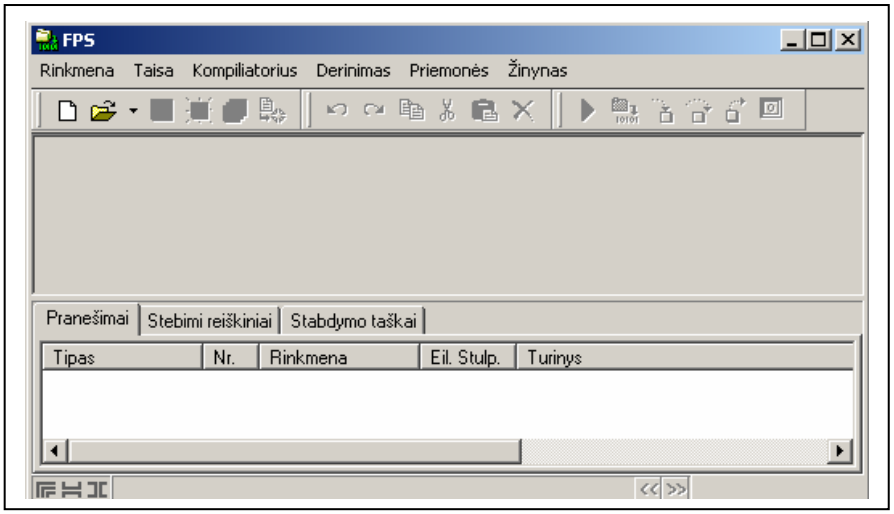
Antras žingsnis. `Free Pascal` terpės paleidimas.

- Norint pradėti dirbti, reikia, kad kompiuteryje būtų įdiegta kuri nors `Free Pascal` versija. Visi šios knygos pavyzdžiai sukurti naudojant `Free Pascal 0.6.4a` versiją.
- `Pradžios` meniu pasirinkite komandą `Visos programos`, suraskite ir pasirinkite `FPS 0.6.4a`.

arba nuorodą 

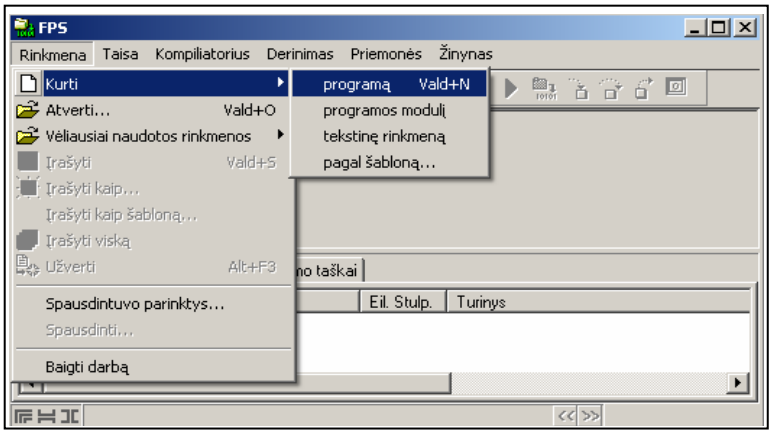
darbalaukyje.

- Sėkmingai įvykdę komandas, patenkate į `Free Pascal` terpę (toliau `FPS`). Tai Windows tipo langas su pagrindiniu meniu ir priemonių juostomis, kaip tekstų rengyklėje ar skaičiuoklėje. Jei programos langas užima ne visą ekraną, o tik jo dalį, darbo langą galima padidinti įprastomis priemonėmis.

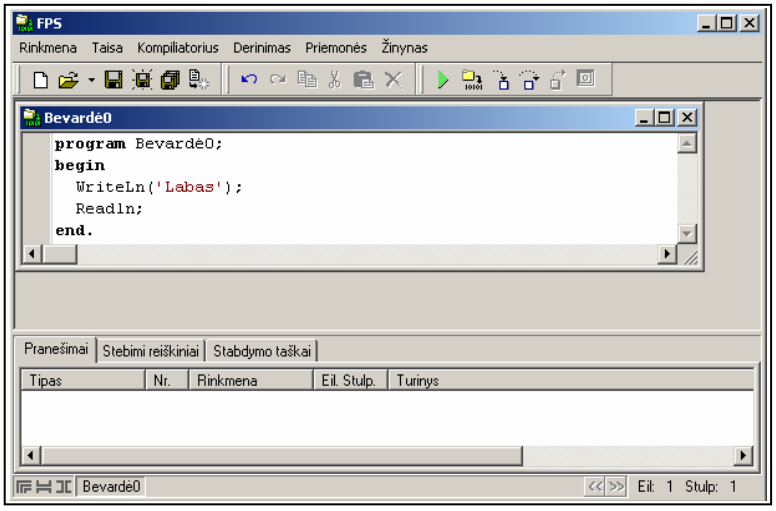


Trečias žingsnis. Programos failo kūrimas.

- Pagrindiniame meniu parinkite komandas: `Rinkmena→Kurti→programą`. Naują programos failą taip pat galima sukurti pasirinkus priemonių juostos mygtuką  arba sparčiųjų klavišų kombinaciją `Ctrl+N`.

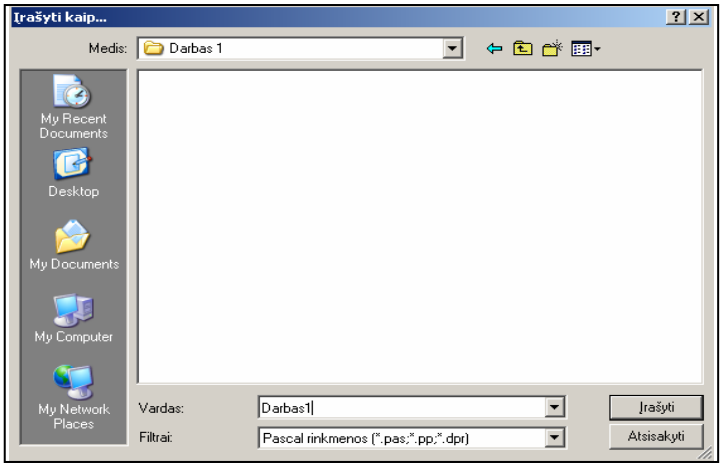


- `FPS` aplinkoje atsiras kuriamos programos rėmai. `FPS` redaktoriaus lange (`Bevardė0`) bus rašomas programos tekstas.

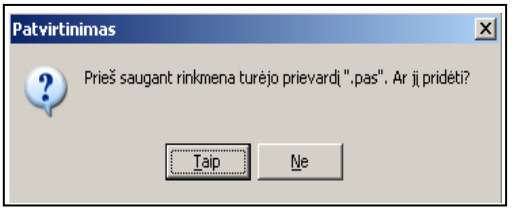


Ketvirtas žingsnis. Programos vardo pakeitimas ir įrašymas kataloge `Darbas1`.

- `FPS` redaktoriaus lange, po bazinio žodžio **`program`**, vietoj žodžio `Bevardė0` įrašykite žodį `Darbas1`. Toks bus Jūsų pirmosios programos vardas.
- Toliau pagrindiniame meniu parinkite komandas:
  `Rinkmena→Įrašyti kaip...`
  Programą įrašyti galima priemonių juostos mygtukais  arba sparčiųjų klavišų deriniu `Ctrl+S`.
- Pasirinkę norimą būdą, atsivėrusiame dialogo lange `Įrašyti kaip...`
  laukelyje `Vardas` vietoj vardo `Bevardė0` įrašykite failo, kuriame bus saugoma sukurta programa, vardą `Darbas1`, o laukelyje `Medis` nurodykite anksčiau sukurtą katalogą `Darbas1` ir paspauskite mygtuką `Įrašyti`. Programa bus įrašyta kompiuterio standžiajame diske.

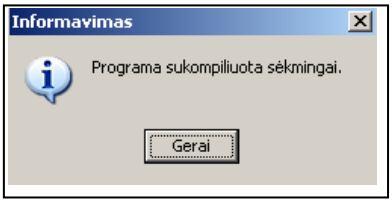


Jeigu užrašydami failo vardą nenurodėte prievardžio `.pas`, sistema paklausia, ar jį užrašyti. Žinoma, reikia atsakyti – `Taip`, nes prievardis parodo, kad faile išsaugota Pascal programa.



Penktas žingsnis. Programos kompiliavimas[1].

- Tai galima atlikti vienu iš trijų būdų:
  - pagrindinio meniu komandomis: `Kompiliatorius→Kompiliuoti`;
  - programos priemonių juostos mygtuku  ;
  - sparčiųjų klavišų deriniu `Ctrl+F9`.

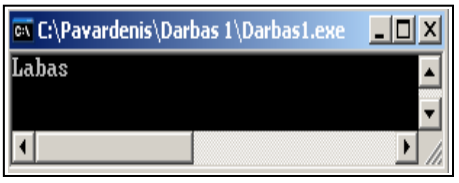


Jeigu programoje nebuvo sintaksės klaidų, ekrane pasirodys informacinis pranešimas. Paspaudę mygtuką `Gerai` patvirtiname, kad susipažinome su informacija apie sėkmingai sukompiliuotą programą.

- Išbandykite visus tris programos kompiliavimo būdus ir pasirinkite sau tinkamiausią.

Tačiau norint pamatyti programos darbo rezultatus ją dar reikia įvykdyti.

Šeštas žingsnis. Programos vykdymas.



- Tai galima atlikti vienu iš trijų būdų:
  - pagrindinio meniu komandomis `Derinimas→Vykdyti`;
  - programos priemonių juostos mygtuku  ;
  - paspaudžiant klaviatūros klavišą `F9`.

Jeigu programoje nebuvo vykdymo klaidų, ekrane pasirodys juodos spalvos programos naudotojo langas su baltos spalvos tekstu: `Labas`.

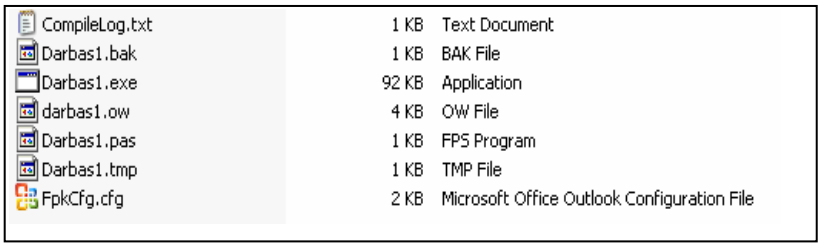
---

[1] Programavimo kalba parašytos programos vertimas į kompiuterinę kalbą.

Kaip ir ankstesniame žingsnyje, kuris programos vykdymo būdas jums pasirodys geriausias, tą toliau ir naudokite.

Rezultatų langą galima uždaryti paspaudžiant klaviatūros klavišą `Enter` arba gerai jums žinomą langų valdymo mygtuką  . Neuždarę rezultatų lango negalėsite redaguoti programos.

Įgiję pakankamai patyrimo norėdami paspartinti darbą `FPS` aplinkoje penktą žingsnį galite ir praleisti. Prieš vykdant programą, jeigu ji buvo redaguota (keista ar papildyta), ji bus iš naujo sukompiliuota ir įvykdyta, jei ne – tik įvykdyta.



Išskleiskite darbo pradžioje sukurtą katalogą `Darbas1` ir pažiūrėkite, kiek ir kokių failų buvo sukurta. Svarbiausias failas yra `Darbas1.pas`, nes jame yra programos tekstas. Šį failą reikia saugoti. Iš šio failo sukuriami kiti tokio paties pavadinimo failai `Darbas1` su skirtingais prievardžiais, bei kiti pagalbiniai failai.

Septintas žingsnis. Pakeiskite ankstesnę programą taip, kad ją įvykdžius, šalia žodžio `Labas` būtų užrašytas ir Jūsų vardas.

- `FPS` programos tekste eilutę `WriteLn('Labas');` pakeiskite tokia:
  `WriteLn('Labas. Mano vardas Vytautas!');`
- Atlikus pakeitimą programa bus tokia:

```
program Darbas1;
begin
  WriteLn('Labas. Mano vardas Vytautas!');
  ReadLn;
end.
```

ir įvykdykite programą, t. y. pakartokite penktą ir šeštą žingsnius, arba tik šeštą žingsnį. Įvykdę programą ekrane turėtumėte matyti:

```
Labas. Mano vardas Vytautas!
```

Aštuntas žingsnis. Pakeiskite ankstesnę programą taip:

```
program Darbas1;
begin
  WriteLn('********************************');
  WriteLn('* Labas. Mano vardas Vytautas! *');
  WriteLn('********************************');
  ReadLn;
end.
```

- Įvykdę programą ekrane turėtumėte matyti:

```
********************************
* Labas. Mano vardas Vytautas! *
********************************
```

- Kaip pastebite, kiekvienas `WriteLn` sakinys, vykdant programą ekrane užrašo po vieną tarp apostrofų užrašytą eilutę.
- Programoje `WriteLn` pakeiskite į `Write`.
  `Write('********************************');`

```
Write('* Labas. Mano vardas Vytautas! *');
Write('********************************');
```

- Įvykdę programą ekrane matote:

```
******************************** Labas. Mano vardas Vytautas! ****************
***************
```

Pastebėjote, kad vietoj `WriteLn` parašius procedūrą `Write`, užpildoma pirma eilutė (eilutėje telpa 80 simbolių), po to pradedama pildyti antra eilutė.

- Papildykite programą taip, kad prieš vardą ir po vardu būtų spausdinamas iš žvaigždučių sudarytas ornamentas:

```
  *   *   *   *   *   *   *   *
 *** *** *** *** *** *** *** ***
  *   *   *   *   *   *   *   *
* Labas. Mano vardas Vytautas! *
  *   *   *   *   *   *   *   *
 *** *** *** *** *** *** *** ***
  *   *   *   *   *   *   *   *
```

```
Black = 0;
Blue = 1;
Green = 2;
Cyan = 3;
Red = 4;
Magenta = 5;
Brown = 6;
LightGray = 7;
DarkGray = 8;
LightBlue = 9;
LightGreen = 10;
LightCyan = 11;
LightRed = 12;
LightMagenta = 13;
Yellow = 14;
White = 15;
Blink = 128;
```

Devintas žingsnis. Spalvų keitimas.

- Sukurkite programą:

```
program Darbas1;
  uses Crt;
begin
  TextBackground (Red);
  TextColor (Yellow);
  WriteLn('Labas. Mano vardas Vytautas!');
  ReadLn;
end.
```

Norėdami pakeisti rezultatų ekrano spalvas, turėjome prisijungti biblioteką `Crt`. Ji prijungiama sakiniu **`uses`** `Crt;` Sakinys rašomas antroje programos eilutėje. Procedūra `TextBackground` nurodo, kokia bus teksto fono spalva, o `TextColor` – kokia bus teksto spalva.

- Įvykdę programą ekrane turėtumėte matyti geltonos spalvos tekstą raudoname fone.
- Pakeiskite teksto fono spalvą į žalią (`Green`), o teksto spalvą į mėlyną (blue).
- Norėdami, kad skirtingose eilutėse užrašytas tekstas ir/ar jo fonas būtų skirtingos spalvos, spalvas būtinai aprašykite prieš `WriteLn` eilutę. Pvz.:

```
program Darbas1;
  uses Crt;
begin
  TextBackground (Green);
  TextColor(Blue);
  WriteLn('Labas. Mano vardas Vytautas!');
  TextBackground (Red);
  TextColor(Yellow);
  WriteLn('Viso gero.');
  ReadLn;
end.
```

- Įvykdykite šią programą. Tekstas „`'Labas. Mano vardas Vytautas!`“ bus mėlynos spalvos žaliame fone, o tekstas „`Viso gero.`“ Bus geltonos spalvos raudoname fone.

Dešimtas žingsnis. Darbo su `FPS` pabaiga.

- Tai galima atlikti pagrindinio meniu komandomis: `Rinkmena→Baigti darbą` (angl. `File→Exit`) arba paspaudžiant `FPS` lango uždarymo mygtuką  dešiniajame viršutiniame darbo lango kampe.

Rašydami programą ir baigdami darbą `FPS` aplinkoje nepamirškite programą įrašyti standžiajame kompiuterio diske. Tai galima atlikti vienu iš trijų būdų:

- pagrindinio meniu komandomis: `Rinkmena→Įrašyti;`
- programos priemonių juostos mygtuku  ;
- sparčiųjų klavišų deriniu `Ctrl+S`.

Klausimai

1. Ką pirmiausiai reikia susikurti prieš pradedant rašyti programą?
2. Kaip iškviečiama `Free Pascal` aplinka?
3. Kaip pradedama kurti nauja programa?
4. Kurioje vietoje rašomas programos vardas?
5. Kokius veiksmus reikia atlikti norint įrašyti programą kompiuterio standžiajame diske pirmą kartą?
6. Kokiais būdais programą galima sukompiliuoti?
7. Kokiais būdais programą galima įvykdyti?
8. Kokiais būdais programą galima įrašyti standžiajame diske?
9. Ar galima programą vykdyti jos nesukompiliavus? Paaiškinkite.
10. Kuo procedūra `WriteLn` skiriasi nuo procedūros `Write`?

Užduotys

1. Parašykite programą, kuri ekrane (arba faile) iš skirtingų spalvų žvaigždučių nupieštų Jūsų vardo ir pavardės pirmųjų raidžių derinį.
2. Parašykite programą, kuri ekrane nupieštų jūsų sugalvotą ornamentą.

Smalsiems

- Norėdami darbo rezultatus matyti ne kompiuterio ekrane, o tekstiniame faile, turėsite:
  - programą papildyti failo kintamuoju,
  - prisijungti failą ir jį susieti su failo kintamuoju,
  - parengti failą įrašymui,
  - įrašyti į failą norimą tekstą,
  - uždaryti tekstinį failą.
- Išnagrinėkite programą ir pabandykite patys ją sukurti ir paleisti vykdyti:

```
program Darbas1;
  var Fr : text;                  // Tekstinio failo kintamasis
begin
  Assign(Fr, 'Darbas1.txt');   // Tekstinis failas Darbas1.txt susiejamas su failo kintamuoju
  Rewrite(Fr);                    // Failas parengiamas įrašymui
  WriteLn(Fr, '*********************************');
  WriteLn(Fr, '* Labas. Mano vardas Vytautas! *');
  WriteLn(Fr, '*********************************');
  Close(Fr);                     // Failas uždaromas
  Readln;
end.
```

- Įvykdę programą matysite tuščią juodą ekraną, nes visa informacija bus saugoma tekstiniame faile `Darbas1.txt`, kuris yra sukuriamas tame pačiame kataloge, kuriame saugomas ir programos failas.

- Matome, kad rašant procedūrą `WriteLn`, pirmiausiai užrašomas failo kintamasis `Fr`, po to tarp apostrofų rašomas tekstas, kurį norėsime matyti įrašytą tekstiniame faile.
- Savo sukurtą programą papildykite, kad tekstas ir ornamentas, kurie buvo išvedami į ekraną, būtų spausdinami į tekstinį failą.

2.2. Išlaidos kambario remontui

Atlikdami šį darbą išsiaiškinsite, kaip kuriama paprastus skaičiavimus atliekanti programa:

- išmoksite aprašyti sveikojo ir realiojo tipo kintamuosius,
- išsiaiškinsite, kaip perskaitomos ir įsimenamos kintamųjų reikšmės,
- suprasite, kaip užrašomi ir atliekami įvairūs skaičiavimai,
- pritaikysite rašymo procedūras `Write` ir `WriteLn` rezultatų išvedimui į ekraną.

Nuorodos į Paskalio kalbos žinyną (psl.)		Nuorodos į algoritmų žinyną (psl.)	
3.1. Kintamasis, kintamojo reikšmė	94	4.1. Tiesinis algoritmas	103
3.2. Priskyrimo sakinys	94		
3.3. Duomenų įvedimas klaviatūra	95		
3.4. Rezultatų (duomenų) išvedimas ekrane	96		

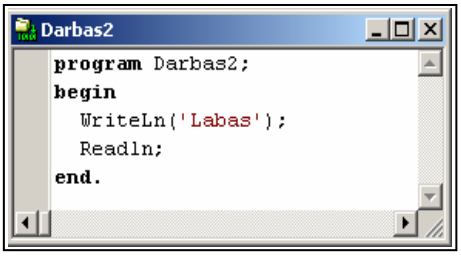
Užduotis. Žinomi kambario išmatavimai (metrais) – `ilgis` ir `plotis`. Abu dydžiai yra sveikieji skaičiai. Reikia apskaičiuoti, kokią pinigų sumą `psuma` kainuos iškloti kambario grindis plytelėmis, jei plytelių vieno kvadratinio metro kaina yra `m2kaina` litų. Plytelių reikia pirkti 5 % daugiau galimiems nuostoliams padengti.

Uždavinio sprendimo algoritmas:

- skaičiuojamas kambario `plotas`;
- skaičiuojama, kokią pinigų sumą `psuma` kainuos plytelės.

Aišku, kintamiesiems galima sugalvoti ir kitokius vardus, tačiau geriausia, kai kintamųjų vardai atspindi jų paskirtį. Tuomet jų nereikia komentuoti.

Pirmas žingsnis. Pasiruošimas.



- Kaip ir pirmajame darbe `Darbas1` atlikite veiksmus:
  - sukurkite katalogą programos failams saugoti: `Darbas2`;
  - iškvieskite `FPS` terpę;
  - sukurkite programos failą;
  - suteikite programai vardą `Darbas2`;
  - išsaugokite failą sukurtame kataloge `Darbas2` vardu `Darbas2.pas`.

Antras žingsnis. Bandomasis programos kompiliavimas ir vykdymas.

- Sukompiliuokite ir įvykdykite programą.

Kaip ir pirmajame darbe, programa tik pasisveikins, t. y. parašys ekrane žodį `Labas`, tačiau jokių skaičiavimų neatliks.

Trečias žingsnis. Programos pradinių duomenų aprašymas ir įvedimas.

- Programos pradžioje aprašykite kambario išmatavimus saugančius sveikojo tipo kintamuosius `ilgis` ir `plotis`.
- Pakeiskite sakinį `WriteLn('Labas');` nauju sakiniu
  `WriteLn('Programa darbą pradėjo.');`
- Parašykite kintamojo `ilgis` reikšmės įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį `Write` ir reikšmės skaitymo sakinį `Readln`.
- Išsaugokite ir įvykdykite programą.

```
program Darbas2;
```

```
var ilgis, plotis : integer;    // Kambario išmatavimai
begin
  WriteLn('Programa darbą pradėjo.');
  Write('Įveskite kambario ilgį: '); ReadLn(ilgis);
  ReadLn;
end.
```

- Paleidę programą, klaviatūra surinkę skaičių `5` ir paspaudę `Enter` klavišą, ekrane matysite:

```
Programa darbą pradėjo.
Įveskite kambario ilgį: 5
```

- Parašykite kintamojo `plotis` reikšmės įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį `Write` ir reikšmės skaitymo sakinį `ReadLn`. Galite kopijuoti kintamojo `ilgis` reikšmės įvedimo sakinius ir vietoje kintamojo vardo `ilgis` parašyti `plotis`.
- Išsaugokite ir įvykdykite programą.

```
program Darbas2;
  var ilgis, plotis : integer;
begin
  WriteLn('Programa darbą pradėjo.');
  Write('Įveskite kambario ilgį: ');  ReadLn(ilgis);
  Write('Įveskite kambario plotį: '); ReadLn(plotis);
  Readln;
end.
```

- Paleidę vykdyti programą, klaviatūra surinkę skaičių `5` ir skaičių `4`, bei paspaudę po kiekvieno iš jų `Enter` klavišą, ekrane matysite:

```
Programa darbą pradėjo.
Įveskite kambario ilgį: 5
Įveskite kambario plotį: 4
```

Ketvirtas žingsnis. Papildomas programos kintamųjų, skirtų rezultatui – kambario plotui – saugoti, aprašymas. Rezultatų skaičiavimas ir rodymas ekrane.

- Papildykite programą nauju sveikojo tipo kintamuoju `plotas`.
- Užrašykite ploto skaičiavimo sakinį: `plotas := ilgis * plotis;`
- Ekrane spausdinkite apskaičiuotą ploto reikšmę.
- Programos pabaigoje prieš sakinį `ReadLn;` užrašykite sakinį
  `WriteLn('Programa darbą baigė.');`
- Išsaugokite ir įvykdykite programą.

```
program Darbas2;
  var ilgis, plotis : integer;
             plotas : integer;
begin
  WriteLn('Programa darbą pradėjo.');
  Write('Įveskite kambario ilgį: ');  ReadLn(ilgis);
  Write('Įveskite kambario plotį: '); ReadLn(plotis);
  plotas := ilgis * plotis;
  WriteLn('Kambario plotas: ', plotas);
  WriteLn('Programa darbą baigė.');
  ReadLn;
end.
```

- Paleidę programą, klaviatūra surinkę skaičių 5 ir skaičių 4, bei paspaudę po kiekvieno iš jų `Enter` klavišą, ekrane matysite:

```
Programa darbą pradėjo.
Įveskite kambario ilgį: 5
Įveskite kambario plotį: 4
Kambario plotas: 20
Programa darbą baigė.
```

Penktas žingsnis. Žinodami kambario plotą apskaičiuosime, kiek kainuos kambarį iškloti plytelėmis. Plytelių reikia pirkti 5 % daugiau galimiems nuostoliams padengti. Dabar reikia apskaičiuoti, kiek jos kainuos, kai žinoma vieno m^2 kaina. Pinigų suma, skirta kambario remontui, gali būti skaičiuojama užrašius priskyrimo sakinius:

```
psuma := plotas * m2kaina + 0.05 * plotas * m2 kaina;
```

arba

```
psuma := 1.05 * plotas * m2kaina;
```

- Papildykite programą tokiais Paskalio kalbos sakiniais:
  - kintamųjų `m2kaina` ir `psuma` aprašymu:

```
m2kaina : real;  // Plytelių 1 kvadratinio metro kaina
psuma   : real;  // Pinigų suma
```

Kaip pastebėjote, šie kintamieji turi būti realaus tipo (`real`), nes plytelių kainą sudaro dvi dalys: sveikoji (litai) ir trupmeninė (centai).

  - plytelių kainos skaičiavimais:

```
Write('Įveskite plytelių 1 kvadr. metro kainą: '); ReadLn(m2kaina);
psuma := 1.05 * plotas * m2kaina;
WriteLn('Pinigų suma, kurią reikia sumokėti: ', psuma:6:2);
```

- Išsaugokite ir įvykdykite programą su pateiktais pradiniais duomenimis:

```
Programa darbą pradėjo.
Įveskite kambario ilgį: 5
Įveskite kambario plotį: 4
Kambario plotas: 20
Įveskite plytelių 1 kvadr. metro kainą: 45.50
Pinigų suma, kurią reikia sumokėti: 955.50
Programa darbą baigė.
```

Klausimai

1. Koks bazinis Paskalio programavimo kalbos žodis yra naudojamas sveikojo tipo kintamiesiems aprašyti?
2. Koks bazinis Paskalio programavimo kalbos žodis yra naudojamas realiojo tipo kintamiesiems aprašyti?
3. Aprašykite du sveikojo tipo kintamuosius: mokinio amžius (metai) ir masė (kilogramai).
4. Aprašykite realiojo tipo kintamąjį: mokinio ūgis (metrai).
5. Ką reiškia raktinis žodis **`var`**, kam jis naudojamas ir kurioje programos vietoje jis rašomas?
6. Kokie du sakiniai naudojami kintamojo reikšmei įvesti? Užrašykite pavyzdį kintamojo `greitis` reikšmei įvesti.
7. Ar galima vienu sakiniu `ReadLn` įvesti kelias kintamųjų reikšmes? Jeigu galima, užrašykite pavyzdį.
8. Kam programos pabaigoje reikalingas sakinys `ReadLn;`?

Užduotys

1. Parašykite programą, skaičiuojančią, kiek lapų popieriaus `k` reikės visos klasės mokinių konspektams nukopijuoti, jei žinomas klasės mokinių skaičius `n` ir kiek lapų `m` sudaro konspektą. *Pasitikrinkite: įvedę n = 20 ir m = 10, turėtumėte gauti k = 200.*
2. Laikrodis rodo `x` valandų ir `y` minučių. Parašykite programą, kuri apskaičiuotų kiek minučių `m` ir kiek sekundžių `s` prabėgo nuo vidurnakčio. Programoje naudokite sveikųjų skaičių tipą `longint` (dideliems sveikiesiems skaičiams).
   Pasitikrinkite: įvedę x = 3 ir y = 5, turėtumėte gauti: m = 185, s = 11100.
3. Šiandien Tautvydas švenčia gimtadienį. Jam sukanka `a` metų. Parašykite programą, kuri apskaičiuotų kiek mėnesių `men`, dienų `d` ir valandų `v` Tautvydas jau pragyveno šiame pasaulyje. Programoje naudokite sveikųjų skaičių tipą `longint`. Spręsdami uždavinį laikykite, kad metai turi 365 dienas. *Pasitikrinkite: įvedę a = 16 turėtumėte gauti: men = 192, d = 5840, v = 140160.*
4. Parašykite programą, skaičiuojančią, kiek knygų `k` vidutiniškai per metus perskaito vienas mokyklos bibliotekos lankytojas, jei bibliotekininkė pateikė vidutinį per vieną mėnesį perskaitytų knygų skaičių `v` ir vidutinį bibliotekos lankytojų skaičių `n` per metus. Spręsdami uždavinį naudokite standartinę Paskalio funkciją `Round`, kuri suapvalins knygų skaičių iki sveikojo skaičiaus. *Pasitikrinkite: įvedę v = 120, n = 800, turėtumėte gauti k = 2.*
5. Parašykite programą, skaičiuojančią, kiek keleivių `k` vidutiniškai važiuoja viename traukinio vagone į Vilnių, jei žinomas traukinio keleivių skaičius `n`, keleivių, vykstančių ne į Vilnių skaičius `nv` ir vagonų skaičius `v`. Spręsdami uždavinį naudokite standartinę Paskalio funkciją `Round`, kuri suapvalins keleivių skaičių iki sveikojo skaičiaus. *Pasitikrinkite: įvedę n = 100, nv = 20 ir v = 4, turėtumėte gauti k = 20.*
6. Parašykite programą, skaičiuojančią stačiakampio, kurio viršutinio kairiojo taško (x1; y1) ir apatinio dešiniojo taško (x2; y2) koordinatės yra sveikieji skaičiai įvedami klaviatūra, plotą s ir perimetrą p. Stačiakampio kraštinės lygiagrečios koordinačių ašims. *Pasitikrinkite: kai x1 = 0, y1 = 5, x2 = 4, y2 = 0, turi būti spausdinama: Stačiakampio plotas s = 20 kvadr. vnt., stačiakampio perimetras p = 18 vnt.*
7. Taškai A (x1;y1) ir B (x2; y2) yra atkarpos galai. Jų koordinatės įvedamos klaviatūra. Atkarpa AB yra skritulio skersmuo. Parašykite programą, skaičiuojančią skritulio plotą s ir jo centro koordinates xc ir yc. *Pasitikrinkite: kai x1 = 0, y1 = 0, x2 = 0, y2 = 4, turi būti spausdinama: Skritulio plotas s = 12.57 kvadr. vnt., skritulio centro koordinatės: xc = 0, yc = 2.*
8. Parašykite programą, kuri sukeistų kintamųjų `x` ir `y` reikšmes vietomis. Rašant programą negalima naudoti papildomų kintamųjų. *Pasitikrinkite: kai x = 0, y = 5, turi būti spausdinama: Sukeitę x ir y reikšmes vietomis, gauname: x = 5, y = 0.*
9. Žinomas laikas, išreikštas sveikaisiais skaičiais: m, s (čia m – minutės ir $0 < m \le 59$, s – sekundės ir $0 < s \le 59$, be to $m \le s$). Parašykite programą, kuri kompiuterio ekrane spausdintų, kokį kampą ms (skaičiuojant pagal laikrodžio rodyklę) sudaro minučių ir sekundžių rodyklės ir kiek mažiausiai laiko minutėmis tm ir sekundėmis ts turi praeiti, kol laikrodžio minučių ir sekundžių rodyklės sutaps. *Pasitikrinkite: kai m = 12, s = 15, turi būti spausdinama: Kampas tarp minučių ir sekundžių rodyklių ms = 18 laipsnių. Kad rodyklės sutaptų, turi praeiti tm = 0 minučių (-ės) ir ts = 58 sekundžių (-ės).*

2.3. Grąžos atidavimas mažiausiu banknotų ir monetų skaičiumi

Atlikdami šį darbą išsiaiškinsite sveikųjų skaičių dalmens sveikosios dalies ir liekanos skaičiavimą:

- išmoksite tinkamai užrašyti sveikųjų skaičių dalybos operacijas;
- įtvirtinsite kintamųjų aprašymo, tinkamo pradinių duomenų įvedimo ir rezultatų pateikimo įgūdžius.

Nuorodos į Paskalio kalbos žinyną (psl.)		Nuorodos į algoritmų žinyną (psl.)	
3.1. Kintamasis, kintamojo reikšmė	94	4.1. Tiesinis algoritmas	103
3.2. Priskyrimo sakinys	94		
3.3. Duomenų įvedimas klaviatūra	95		
3.4. Rezultatų (duomenų) išvedimas ekrane	96		

Užduotis. Parduotuvėje pardavėja nori pirkėjui grąžą `g` Lt (g – sveikasis skaičius) atiduoti mažiausiu 1, 2, 5 Lt nominalo monetų ir 10, 20, 50 ir 100 Lt banknotų skaičiumi. Reikia apskaičiuoti, kiek kokio nominalo monetų ir banknotų pardavėja turės atiduoti pirkėjui.

Uždavinio sprendimo algoritmas:

- pirmiausiai imame didžiausio nominalo banknotą (100 Lt) ir grąžą `g` daliname iš 100 ir skaičiuojame sveikąją dalmens dalį. Gautas rezultatas yra 100 Lt nominalo banknotų skaičius `k100`;
- norėdami apskaičiuoti, kokia pinigų suma liko nepanaudota, galime grąžą `g` dalinti iš 100 ir apskaičiuoti dalmens liekaną. Tai bus nauja grąža `g`. Likusi grąžos g dalis gali būti skaičiuojama ir kitaip: `g := g - k100 * 100;`
- pirmuosius du veiksmus kartojame su visų nominalų monetomis ir banknotais.

Pavyzdžiui, jei pardavėja pirkėjui turi atiduoti `g = 75 Lt` grąžą, tai jai reikės vieno 50 Lt, vieno 20 Lt banknotų ir 5 Lt monetos.

Pirmas žingsnis. Pasiruošimas.

- Kaip ir antrajame darbe `Darbas2` atlikite veiksmus:
  - sukurkite katalogą programos failams saugoti: `Darbas3`;
  - iškvieskite `FPS` terpę;
  - sukurkite programos failą;
  - suteikite programai vardą `Darbas3`;
  - išsaugokite failą sukurtame kataloge `Darbas3` vardu `Darbas3.pas`.

Antras žingsnis. Programos kintamųjų – pradinių duomenų aprašymas ir įvedimas.

- Programos pradžioje aprašykite sveikojo tipo kintamąjį `g`, kuris reiškia pirkėjo grąžą.
- Parašykite kintamojo `g` reikšmės įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį `Write` ir reikšmės skaitymo sakinį `ReadLn`.
- Išsaugokite ir įvykdykite programą.

```
program Darbas3;
  var g : integer;
begin
  Write('Įveskite pirkėjo grąžą: '); ReadLn(g);
  Readln;
end.
```

Paleidę programą, klaviatūra surinkę skaičių `75` ir paspaudę klavišą `Enter`, ekrane matysite:

```
Įveskite pirkėjo grąžą: 75
```

Trečias žingsnis. Papildomas programos kintamųjų, skirtų rezultatams – kiekvieno nominalo banknotų ar monetų skaičiui – sąrašas. Rezultatų skaičiavimas ir rodymas ekrane.

- Papildykite programą naujais sveikojo tipo kintamaisiais `k100, k50, k20, k10, k5, k2` ir `k1`, skirtais kiekvieno nominalo banknotų ar monetų skaičiui saugoti.
- Užrašykite priskyrimo sakinį, skaičiuojantį, kiek reikės 100 Lt nominalo banknotų `k100` grąžai atiduoti:

```
k100 := g div 100;
```

- Užrašykite priskyrimo sakinį, skaičiuojantį, kokia pinigų suma `g` liks, atidavus `k100` 100 Lt banknotų:

```
g := g mod 100;
```

arba

```
g := g - k100 * 100;
```

- Toliau rašome priskyrimo sakinius, skaičiuojančius 50, 20, 10, 5, 2 ir 1 Lt nominalo banknotų ar monetų skaičių.

```
program Darbas3;
var g : integer;
    k100, k50, k20, k10, k5, k2, k1 : integer;
begin
  Write('Įveskite pirkėjo grąžą: '); ReadLn(g);
  k100 := g div 100; g := g mod 100;
  k50  := g div 50;  g := g mod 50;
  k20  := g div 20;  g := g mod 20;
  k10  := g div 10;  g := g mod 10;
  k5   := g div 5;   g := g mod 5;
  k2   := g div 2;   g := g mod 2;
  k1   := g;
  ReadLn;
end.
```

- Programos pabaigoje prieš sakinį `ReadLn` parašykite rezultatų spausdinimo ekrane sakinius:

```
  WriteLn('Pardavėja grąžą atiduos taip:');
  WriteLn('-----------------------------');
  WriteLn('100 Lt ------ ', k100);
  WriteLn(' 50 Lt ------ ', k50);
  WriteLn(' 20 Lt ------ ', k20);
  WriteLn(' 10 Lt ------ ', k10);
  WriteLn('  5 Lt ------ ', k5);
  WriteLn('  2 Lt ------ ', k2);
  WriteLn('  1 Lt ------ ', k1);
  WriteLn('-----------------------------');
```

- Išsaugokite ir įvykdykite programą.

Paleidę programą, klaviatūra surinkę skaičių `75`, ekrane matysite:

```
Pardavėja grąžą atiduos taip:
-----------------------------
100 Lt ------ 0
 50 Lt ------ 1
 20 Lt ------ 1
 10 Lt ------ 0
  5 Lt ------ 1
  2 Lt ------ 0
  1 Lt ------ 0
-----------------------------
```

Užduotys

1. Apskaičiuokite reiškinių reikšmes:
   a. `17 mod 3 + 7 div 3 * 2 =`
   b. `17 mod (3 + 7) div 3 * 2 =`
   c. `17 mod (3 + 7 div 3) *2 =`
   d. `1 + 49 mod 5 + 6 div 3 =`
   e. `(1 + 49) mod 5 - 6 div 3 =`
   f. `1 + 49 mod (5 + 7) div 3 =`
   g. `19 mod 8 + 7 div 3 =`
   h. `19 mod (8 + 7 div 3) =`
2. Nurodykite šių reiškinių rezultatų tipus:
   a. `123 + 45 + 5678`
   b. `49 + (12.5 - 1) * 3`
   c. `73/2 * 2`
   d. `72 div 2 + 1.05`
   e. `100 div 5 + 700 - 25/5`
3. Nuo metų pradžios praėjo d dienų. Parašykite programą, skaičiuojančią, kiek savaičių s praėjo nuo metų pradžios. *Pasitikrinkite: kai d = 15, turi būti spausdinama: Nuo metų pradžios praėjo s = 2 savaičių (-ės).*
4. Miesto informatikos olimpiadoje dalyvavo `n` devintokų. Mokytoja nupirko `m` saldainių „Nomeda“ ir išdalino mokiniams po lygiai. Padalinus saldainių neliko arba liko mažiau, negu yra mokinių. Po kiek saldainių `s` gavo kiekvienas mokinys ir kiek saldainių `k` liko mokytojai? Parašykite programą šiam uždaviniui spręsti. *Pasitikrinkite: kai n = 7 ir m = 23, tai kiekvienas mokinys gavo po s = 3 saldainius, o mokytojai liko k = 2 saldainiai.*
5. Švęsdamas septintąjį gimtadienį Andrius gavo `n` balionų. Su draugais nusprendė balionus paleisti į dangų. Tačiau juos pučiant `k` balionų sprogo. Likusius balionus Andrius pasidalino su `d` draugais po lygiai. Jeigu dalinant balionus visiems po lygiai jų liko, tai juos pasiėmė Andrius. Po kiek balionų `m` gavo kiekvienas draugas ir kiek balionų `a` teko Andriui? Parašykite programą šiam uždaviniui

spręsti. *Pasitikrinkite: kai n = 77, d = 7 ir k = 3, tai kiekvienas draugas gavo po m = 9 balionus, o Andriui teko a = 14 balionų.*

6. Lėktuvas pakilo iš aerouosto, kai buvo A valandų ir B minučių. Lėktuvas ore praleido C minučių. Parašykite programą, kuri nustatytų, kiek bus valandų V ir minučių M, kai lėktuvas nusileis. Atkreipkite dėmesį, kad C reikšmė gali būti didelė ir lėktuvas gali leistis ne tą pačią parą. Parašykite programą šiam uždaviniui spręsti. *Pasitikrinkite: kai A = 23, B = 55, C =14, tai lėktuvas leisis V = 0 valandų ir M = 9 minutės.*
7. Nubrauktas triženklio skaičiaus x antrasis skaitmuo. Kai likusiam dviženkliam skaičiui iš kairės buvo prirašytas nubrauktasis skaitmuo, tuomet buvo gautas skaičius n ($10 < n \leq 999$, be to skaičiaus n dešimčių skaitmuo nelygus nuliui). Parašykite programą, kuri apskaičiuotų, kokia buvo x reikšmė, kai n reikšmė įvedama klaviatūra. *Pasitikrinkite: kai n = 135, turi būti spausdinama: Triženklis skaičius x = 315.*

Smalsiems

- Norėdami sutrumpinti sukurtą grąžos skaičiavimo programą, pasikartojančius veiksmus – banknotų ar monetų kiekio, neatiduotos grąžos likučio skaičiavimą, kiekvieno nominalo banknotų ar monetų kiekio spausdinimą – įkelsime į savarankišką programos dalį – procedūrą.
- Iš pagrindinės programos į procedūrą kreipsimės su skirtingų nominalų pinigais.
- Procedūros tekstas rašomas programos pradžioje po programos antrašte. Labai patogu nuo pagrindinės programos procedūrą atskirti komentaru, pvz., brūkšnelių ar žvaigždučių eilute.
- Užrašykite programos antraštę ir komentarą:

```
program Darbas3;
//-------------------------------------------------------
```

- Rašome procedūrą `Graza`. Procedūros antraštėje skliausteliuose aprašysime tris sveikojo tipo kintamuosius: kintamasis `k` – kokio nominalo banknotų ar monetų skaičius skaičiuojamas, `kx` – kiek `k` nominalo pinigų yra, `g` – kokia grąža dar liko neatiduota. Kintamieji `kx` ir `g` rašomi su žodeliu **`var`**, nes į pagrindinę programą turi būti grąžinamos apskaičiuotos jų reikšmės. Aprašant kintamuosius būtina nurodyti jų tipą. Procedūros visuje parašykite priskyrimo sakinius `kx` ir `g` reikšmėms skaičiuoti ir rašymo sakinį, skirtą `k` nominalo pinigų skaičiaus spausdinimui.

```
procedure Graza(k : integer; var kx : integer; var g : integer);
begin
  kx := g div k; g := g mod k;
  WriteLn(k : 3, ' Lt ------ ', kx);
end;
```

- Po procedūra parašykite skiriamąjį komentarą (brūkšnelių eilę), aprašykite pagrindinės programos kintamuosius:

```
//-------------------------------------------------------
var g : integer;
    k100, k50, k20, k10, k5, k2, k1 : integer;
```

- Pagrindinėje programoje parašykite pradinių duomenų įvedimo sakinius ir du pirmuosius rezultatų spausdinimo sakinius:

```
begin
  Write('Įveskite pirkėjo grąžą: '); ReadLn(g);
  WriteLn('Pardavėja grąžą atiduos taip:');
  WriteLn('----------------------------');
```

- Toliau rašysime kreipinius į procedūrą `Graza`:

```
  Graza(100, k100, g);
  Graza(50, k50, g);
```

```
  Graza(20, k20, g);
  Graza(10, k10, g);
  Graza(5, k5, g);
  Graza(2, k2, g);
  Graza(1, k1, g);
```

- Parašykite paskutinį rašymo sakinį: `WriteLn ('----------------------------');`
- Pasitikrinkite, ar teisingai sukūrėte programą:

```
program Darbas3;
//-----------------------------------------------------
procedure Graza(k : integer; var kx : integer; var g : integer);
begin
  kx := g div k; g := g mod k;
  WriteLn(k : 3, ' Lt ------ ', kx);
end;
//-----------------------------------------------------
var g : integer;
    k100, k50, k20, k10, k5, k2, k1 : integer;
begin
  Write('Įveskite pirkėjo grąžą: '); ReadLn(g);
  WriteLn('Pardavėja grąžą atiduos taip:');
  WriteLn('----------------------------');
  Graza(100, k100, g);
  Graza(50, k50, g);
  Graza(20, k20, g);
  Graza(10, k10, g);
  Graza(5, k5, g);
  Graza(2, k2, g);
  Graza(1, k1, g);
  WriteLn('----------------------------');
  ReadLn;
end.
```

- Įvykdykite programą, įvesdami `75`. Jei viską atlikote teisingai, ekrane turėtumėte matyti:

```
Pardavėja grąžą atiduos taip:
----------------------------
100 Lt ------ 0
 50 Lt ------ 1
 20 Lt ------ 1
 10 Lt ------ 0
  5 Lt ------ 1
  2 Lt ------ 0
  1 Lt ------ 0
----------------------------
```

- Tikriausiai pastebėjote, kad naudojant procedūrą programa tapo paprastesnė: nebereikia kartoti tų pačių sakinių po kelis kartus, daug lengviau surasti ir ištaisyti klaidas.
- Savarankiškos programos dalys labai palengvina uždavinių sprendimą, leidžia išskaidyti uždavinį dalimis, atskiras dalis gali kurti ne vienas žmogus.

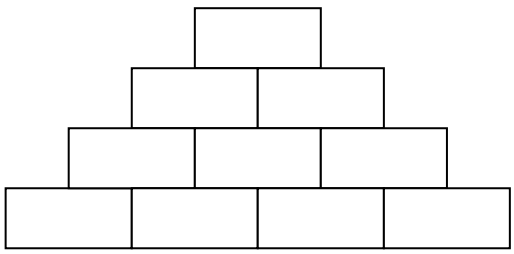
2.4. Piramidė

Atlikdami šį darbą išsiaiškinsite, kaip kuriama ciklinius skaičiavimus atliekanti programa, kai iš anksto nežinoma, kiek kartų reikės kartoti veiksmus:

- išmoksite užrašyti nežinomo kartojimų skaičiaus ciklo antraštę,
- išsiaiškinsite, kaip užrašomi cikle atliekami veiksmai,
- suprasite, kaip skaičiuojama suma ir kiekis,
- išmoksite skaičiavimų rezultatus spausdinti lentelėje.

Nuorodos į Paskalio kalbos žinyną (psl.)		Nuorodos į algoritmų žinyną (psl.)	
3.1. Kintamasis, kintamojo reikšmė	94	4.1. Tiesinis algoritmas	103
3.2. Priskyrimo sakinys	94	4.2. Ciklinis algoritmas	103
3.3. Duomenų įvedimas klaviatūra	95		
3.4. Rezultatų (duomenų) išvedimas ekrane	96		
3.5. Ciklo sakinys while	96		

Užduotis. Iš plytų galima pastatyti vienos plytos pločio taisyklingą piramidę, kurios viršūnėje yra viena plyta, o šonuose – pusės plytos ilgio laipteliai. Parašykite programą, skaičiuojančią, kokio ilgio `a` bus piramidės pagrindas ir kelių plytų `k` aukščio bus piramidė, jei pradinis duomuo yra piramidės statybai skirtų plytų skaičius `p`.



Uždavinio sprendimo algoritmas:

Paveikslėlyje vaizduojamos piramidės pagrindą `a` sudaro `4` plytos, piramidės aukštis `k = 4` plytos, piramidei pastatyti reikėjo `p = 10` plytų. Spręsdami uždavinį žinosime bendrą plytų skaičių `p`, o `a` ir `k` reikšmes turėsime apskaičiuoti.

- Skaičiuosime piramidės statybai panaudotų plytų skaičių `s`, kuris gaunamas prie jau buvusios `s` reikšmės pridedant naujos eilės plytų skaičių `a`.
- Nauja `a` reikšmė gaunama didinant esamą `a` reikšmę vienetu, nes kiekvienoje naujoje eilėje yra viena plyta daugiau negu prieš tai buvusioje.
- Nauja `k` reikšmė gaunama senąją `k` reikšmę didinant vienetu, nes pridėjus naują eilę, eilių skaičius padidėja viena nauja eile.
- Pradinė `s` reikšmė yra lygi nuliui. Skaičiuojamą `s` reikšmę lyginsime su piramidės statybai skirtų plytų skaičiumi `p`. Kai `s` ir `p` reikšmės susilygins, veiksmai bus baigiami.

Lentelėje pavaizduota, kaip keičiasi kintamųjų `a, k` ir `s` reikšmės, atliekant veiksmus, kol `s` reikšmė mažesnė už `p` reikšmę.

Pradinės kintamųjų reikšmės: `a = 0, k = 0, s = 0, p = 10`.

Ciklo žingsniai	Sąlyga: `s < p`	`a = a + 1`	`k = k + 1`	`s = s + a`
1	`0 < 10`	`a = 0 + 1 = 1`	`k = 0 + 1 = 1`	`s = 0 + 1 = 1`
2	`1 < 10`	`a = 1 + 1 = 2`	`k = 1 + 1 = 2`	`s = 1 + 2 = 3`
3	`3 < 10`	`a = 2 + 1 = 3`	`k = 2 + 1 = 3`	`s = 3 + 3 = 6`
4	`6 < 10`	`a = 3 + 1 = 4`	`k = 3 + 1 = 4`	`s = 6 + 4 = 10`
5	`10 < 10`	Veiksmai neatliekami, nes netenkinama sąlyga `s < p`		

Pirmas žingsnis. Pasiruošimas. Sukurkite katalogą `Darbas4`, skirtą programos failams saugoti, iškvieskite `FPS` terpę, sukurkite programos failą ir suteikę programai vardą `Darbas4` įrašykite failą sukurtame kataloge `Darbas4` vardu `Darbas4.pas`.

Antras žingsnis. Programos pradinių duomenų aprašymas ir įvedimas.

- Programos pradžioje aprašykite programoje naudojamus kintamuosius: `p` - piramidei statyti skirtų plytų skaičius, `a` - piramidės pagrindo ilgis, `k` - piramidės aukštis, `s` - piramidės statybai panaudotų plytų skaičius:

```
var p : integer;  // Piramidės statybai skirtų plytų skaičius
    s : integer;  // Piramidės statybai panaudotų plytų skaičius
    a : integer;  // Piramidės pagrindo ilgis
    k : integer;  // Piramidės aukštis
```

- Parašykite kintamojo `p` reikšmės įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį `Write` ir reikšmės skaitymo sakinį `Readln`.

```
program Darbas4;
  var p : integer;  // Piramidės statybai skirtų plytų skaičius
      s : integer;  // Piramidės statybai panaudotų plytų skaičius
      a : integer;  // Piramidės pagrindo ilgis
      k : integer;  // Piramidės aukštis
begin
  Write('Kiek plytų skirta piramidės statybai?: '); ReadLn(p);
  ReadLn;
end.
```

- Išsaugokite ir įvykdykite programą.
- Paleidę programą, klaviatūra surinkę skaičių `10` ir paspaudę klavišą Enter, ekrane matysite:

```
Kiek plytų skirta piramidės statybai?: 10
```

Trečias žingsnis. Suteikiamos kintamiesiems `a, k` ir `s` pradinės reikšmės.

- Papildykite programą trimis priskyrimo sakiniais:
  `a := 0; k := 0; s := 0;`
- Papildykite programą sakiniais, skirtais `a, k` ir `s` reikšmių spausdinimui:

```
program Darbas4;
  var p : integer;  // Piramidės statybai skirtų plytų skaičius
      s : integer;  // Piramidės statybai panaudotų plytų skaičius
      a : integer;  // Piramidės pagrindo ilgis
      k : integer;  // Piramidės aukštis
begin
  Write('Kiek plytų skirta piramidės statybai?: '); ReadLn(p);
  a := 0; k := 0; s := 0;
  WriteLn('Piramidės pagrindo ilgis: ', a);
  WriteLn('Piramidės aukštis: ', k);
  WriteLn('Piramidės statybai panaudotų plytų skaičius: ', s);
  ReadLn;
end.
```

- Išsaugokite ir įvykdykite programą.
- Paleidę programą, klaviatūra surinkę skaičių `10` ir paspaudę klavišą Enter, ekrane matysite:

```
Kiek plytų skirta piramidės statybai?: 10
Piramidės pagrindo ilgis: 0
Piramidės aukštis: 0
Piramidės statybai panaudotų plytų skaičius: 0
```

Ketvirtas žingsnis. Ciklo sakinio antraštė. Veiksmai cikle. Kadangi ciklo sakinyje bus kartojami keli veiksmai, jie turi būti rašomi tarp bazinių žodžių `begin` ir `end`. Toks sudėtinis sakinys baigiamas kabliataškiu.

- Po kintamųjų `a`, `k` ir `s` pradinių reikšmių aprašymu užrašykite ciklo sakinį:

```
while s < p do
   begin
     a := a + 1;
     k := k + 1;
     s := s + a;
   end;
```

- Papildyta programa bus tokia:

```
program Darbas4;
  var p : integer;  // Piramidės statybai skirtų plytų skaičius
      s : integer;  // Piramidės statybai panaudotų plytų skaičius
      a : integer;  // Piramidės pagrindo ilgis
      k : integer;  // Piramidės aukštis
begin
  Write('Kiek plytų skirta piramidės statybai?: '); ReadLn(p);
  a := 0; k := 0; s := 0;
  while s < p do
      begin
        a := a + 1;
        k := k + 1;
        s := s + a;
      end;
  WriteLn('Piramidės pagrindo ilgis: ', a);
  WriteLn('Piramidės aukštis: ', k);
  WriteLn('Piramidės statybai panaudotų plytų skaičius: ', s);
  ReadLn;
end.
```

- Išsaugokite ir įvykdykite programą.
- Paleidę programą, klaviatūra surinkę skaičių `10` ir paspaudę klavišą Enter, ekrane matysite:

```
Kiek plytų skirta piramidės statybai?: 10
Piramidės pagrindo ilgis: 4
Piramidės aukštis: 4
Piramidės statybai panaudotų plytų skaičius: 10
```

Penktas žingsnis. Skaičiavimų tarpinių reikšmių spausdinimas. Norėdami įsitikinti, kad programa skaičiuoja teisingai, galite spausdinti ne tik galutinius, bet ir tarpinius skaičiavimų rezultatus.

- Sudėtinį sakinį, nurodantį, kokie veiksmai bus atliekami ciklo viduje papildykite kintamųjų `a, k` ir `s` reikšmių spausdinimo sakiniais ir sakiniu, atskiriančiu vieną kartojimą nuo kito žvaigždutėmis:

```
while s < p do
   begin
     a := a + 1;
     k := k + 1;
     s := s + a;
     WriteLn('a = ', a);
     WriteLn('k = ', k);
     WriteLn('s = ', s);
     WriteLn('**********');
   end;
```

- Išsaugokite ir įvykdykite programą.
- Paleidę programą, klaviatūra surinkę skaičių `10` ir paspaudę klavišą Enter, ekrane matysite:

```
Kiek plytų skirta piramidės statybai?: 10
a = 1
k = 1
s = 1
**********
a = 2
k = 2
s = 3
**********
a = 3
k = 3
s = 6
**********
a = 4
k = 4
s = 10
**********
Piramidės pagrindo ilgis: 4
Piramidės aukštis: 4
Piramidės statybai panaudotų plytų skaičius: 10
```

Šeštas žingsnis. Skaičiavimų rezultatų spausdinimas lentelėje.

- Norėdami programos skaičiavimų rezultatus spausdinti lentelėje, papildysime programą `WriteLn` sakiniais:

```
program Darbas4;
  var p : integer;   // Piramidės statybai skirtų plytų skaičius
      s : integer;   // Piramidės statybai panaudotų plytų skaičius
      a : integer;   // Piramidės pagrindo ilgis
      k : integer;   // Piramidės aukštis
begin
  WriteLn('Programa, skaičiuojanti piramidės pagrindo ilgį, aukštį');
  WriteLn(' ir statybai panaudotų plytų skaičių');
  WriteLn('************************************************');
  Write('Kiek plytų skirta piramidės statybai?: '); ReadLn(p);
  a := 0; k := 0; s := 0;
  WriteLn('------------------------------------------------');
```

```
  WriteLn('Pagrindo ilgis   Aukštis  Panaudotų plytų skaičius');
  WriteLn('-------------------------------------------------');
   while s < p do
   begin
     a := a + 1;
     k := k + 1;
     s := s + a;
     WriteLn(a : 5, k : 15, s : 20);
   end;
  WriteLn('-------------------------------------------------');
  WriteLn('Piramidės pagrindo ilgis: ', a);
  WriteLn('Piramidės aukštis: ', k);
  WriteLn('Piramidės statybai panaudotų plytų skaičius: ', s);
  ReadLn;
end.
```

- Sakiniai apibrėžiantys programos paskirtį:

```
WriteLn('Programa, skaičiuojanti piramidės pagrindo ilgį, aukštį');
WriteLn(' ir statybai panaudotų plytų skaičių');
WriteLn('*************************************************');
```

- Sakiniai formuojantys lentelės antraštinę eilutę:

```
WriteLn('-------------------------------------------------');
WriteLn('Pagrindo ilgis   Aukštis  Panaudotų plytų skaičius');
WriteLn('-------------------------------------------------');
```

- Sakinys, spausdinantis ciklo viduje apskaičiuotas kintamųjų `a`, `k` ir `s` reikšmes:

```
WriteLn(a : 5, k : 15, s : 20);
```

Šiame sakinyje spausdinamos formatuotos kintamųjų reikšmės: kintamajam a skirtos 5 pozicijos, kintamajam `k` – 15 pozicijų, kintamajam `s` – 20 pozicijų. Pavyzdžiui, jei kintamojo a reikšmė yra lygi 1, tai ši reikšmė spausdinama 5 pozicijoje, o keturios pozicijos iš kairės paliekamos tuščios. Analogiškai spausdinamos ir kitų kintamųjų reikšmės.

- Sakinys, formuojantis lentelės paskutinę eilutę:

```
WriteLn('-------------------------------------------------');
```

- Išsaugokite ir įvykdykite programą.
- Paleidę programą, klaviatūra surinkę skaičių `10` ir paspaudę klavišą Enter, ekrane matysite:

```
Programa, skaičiuojanti piramidės pagrindo ilgį, aukštį
 ir statybai panaudotų plytų skaičių
*************************************************
-------------------------------------------------
Pagrindo ilgis   Aukštis  Panaudotų plytų skaičius
-------------------------------------------------
    1              1                   1
    2              2                   3
    3              3                   6
    4              4                  10
-------------------------------------------------
Piramidės pagrindo ilgis: 4
Piramidės aukštis: 4
Piramidės statybai panaudotų plytų skaičius: 10
```

Septintas žingsnis. Tikrinimas, ar programa visada pateikia teisingus rezultatus.

- Paleidę vykdyti programą įveskite `p` reikšmę, lygią `8`. Įvykdę programą gaunate tą patį rezultatą, kaip ir įvedę `p` reikšmę, lygią `10`. Pastebime, kad rezultatas yra neteisingas, nes piramidės statybai panaudota daugiau plytų, negu skirta.
- Pakeiskite ciklo antraštėje sąlygą `s < p` salyga `p - s >= a + 1` ir įvykdykite programą su `p` reikšme, lygia `8`. Ekrane turėtumėte matyti:

```
Programa, skaičiuojanti piramidės pagrindo ilgį, aukštį
 ir statybai panaudotų plytų skaičių
******************************************************
------------------------------------------------
Pagrindo ilgis   Aukštis  Panaudotų plytų skaičius
------------------------------------------------
    1              1                   1
    2              2                   3
    3              3                   6
------------------------------------------------
Piramidės pagrindo ilgis: 3
Piramidės aukštis: 3
Piramidės statybai panaudotų plytų skaičius: 6
```

- Patikslinus ciklo antraštės sąlygą, programa pateikia teisingus rezultatus su visomis kintamojo p reikšmėmis.

Klausimai

1. Jeigu plytų liko nepanaudotų, tai kiek? Papildykite programą skaičiavimais, kiek liko plytų.
2. Kaip vykdomas kartojimas ciklo sakinyje `while`?
3. Ką daryti, kad ciklas nebūtų begalinis?
4. Kaip užrašyti ciklo sakinį, kai norima kartoti daugiau sakinių?
5. Kurie iš žemiau pateiktų ciklo sakinių, kai a = -3, yra begaliniai ir kodėl?

a)
```
while a < 0 do
   begin
      WriteLn ('argumentas neigiamas');
      a := a + 1;
   end;
```

b)
```
while a < 0 do;
   begin
     WriteLn('argumentas neigiamas');
     a := a + 1;
   end;
```

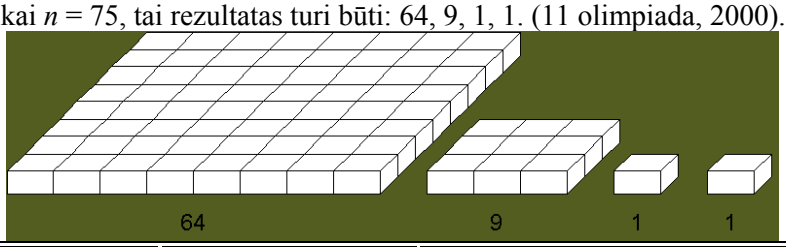
c)
```
while a < 0 do
   WriteLn('argumentas neigiamas');
  a := a + 1;
```

d)
```
while a >= 0 do
    WriteLn('argumentas neigiamas');
  a := a + 1;
```

Užduotys

1. Martynas labai mėgsta saldainius. Mamos slėptuvėje berniukas rado `m` saldainių. Pirmą dieną jis suvalgė 1 saldainį, antrą – 2, trečią – 3 ir kiekvieną tolesnę dieną suvalgydavo vienu saldainiu daugiau negu prieš tai buvusią. Per kelias dienas `d` Martynas suvalgys visus saldainius? Paskutinei dienai gali likti mažiau saldainių. *Pasitikrinkite: kai m = 11, turime gauti tokį rezultatą: d = 5.*
2. **PALŪKANOS.** Pristigo žmogus pinigų ir nuėjo pasiskolinti jų iš prekybininko. Tas sutiko paskolinti, bet paprašė grąžinti juos kitą mėnesį šitaip: pirmąją mėnesio dieną – 1 litą, antrąją – du litus, trečiąją – keturis litus, t. y. **kiekvieną** dieną du kartus daugiau negu prieš tai buvusią. Tą dieną, kai skola galės būti padengta, reikės atiduoti ir visą tos dienos normą. Tai, kas bus atiduota viršaus, ir bus prekybininko palūkanos.
   Užduotis. Parašykite programą, kuri suskaičiuotų, kiek palūkanų litais gaus prekybininkas už paskolintus n litų. (13 olimpiada, 2002).
3. Bankas už indėlius moka `p` procentų palūkanų per metus. Metų gale palūkanos pridedamos prie indėlio. Jei indėlininkas pinigų nė kiek neišima, palūkanos skaičiuojamos nuo vis didesnės sumos. Parašykite programą, kuri apskaičiuotų, per kiek metų `t`, pradinis indėlis `ind` pasieks sumą `s`. *Pasitikrinkite: kai p = 5, ind = 1000, s = 1200, rezultatas t = 4.*
4. Turime kompiuterį, kuris nemoka skaičiuoti natūraliųjų skaičių dalmens sveikosios dalies ir liekanos (nėra DIV ir MOD dalybos operacijų). Parašykite programą, kuri apskaičiuotų skaičių `n` ir `m` dalmens sveikąją dalį `dalmuo` ir liekaną `liekana`. *Pasitikrinkite: kai n = 14, m = 3, turime gauti tokius rezultatus: dalmuo = 4, liekana = 2.*
5. Paskalyje nėra funkcijos, skirtos realiojo skaičiaus `x` kėlimui sveikuoju laipsniu `k`. Parašykite programą šiam uždaviniui spręsti. Rezultatą `y` spausdinkite dviejų ženklų po kablelio tikslumu. *Pasitikrinkite: kai x = 5, k = 2, rezultatas y = 25.00.*
6. **PLYTELIŲ DĖLIOJIMAS.** Iš *n* kvadratinių plytelių reikia sudėlioti vienos plytelės storio kvadratų eilę. Pirmiausia sudedamas didžiausias galimas kvadratas. Iš likusių plytelių – vėl didžiausias ir t. t.

Užduotis. Parašykite programą, kuri išskaidytų duotą plytelių skaičių į dalis, reikalingas kiekvieno kvadrato statybai. Pavyzdžiui, kai *n* = 75, tai rezultatas turi būti: 64, 9, 1, 1. (11 olimpiada, 2000).



Testo nr.	*Pradinis duomuo*	*Rezultatai*	*Paaiškinimai*
1	`3`	`1 1 1`	Kvadratai – atskiros plytelės
2	`35`	`25 9 1`	Trys atsitiktinai parinkti didėjantys testai
3	`89`	`81 4 4`	
4	`149`	`144 4 1`	
5	`225`	`225`	Pradinis duomuo yra kvadratas
6	`32767`	`32761 4 1 1`	Maksimali pradinio duomens reikšmė

Smalsiems

Norėdami, kad piramidės statybos rezultatai būtų išsaugoti tekstiniame faile, programoje `Darbas3.pas` atlikite pakeitimus:

```
program Darbas4;
```

```
   var p : integer;           // Piramidės statybai skirtų plytų skaičius
       s : integer;           //  Piramidės statybai panaudotų plytų skaičius
       a : integer;           // Piramidės pagrindo ilgis
       k : integer;           // Piramidės aukštis
      Fr : text;              //  Tekstinio failo kintamasis
begin
  Assign(Fr, 'Darbas3.txt');  //  Failo kintamasis Fr susiejamas su failo vardu Darbas4.txt
  Rewrite(Fr);                //  Failas parengiamas įrašymui
  WriteLn(Fr, 'Programa, skaičiuojanti piramidės pagrindo ilgį, aukštį');
  WriteLn(Fr, ' ir statybai panaudotų plytų skaičių');
  WriteLn(Fr, '*************************************************');
  Write('Kiek plytų skirta piramidės statybai?: '); ReadLn(p);
  a := 0; k := 0; s := 0;
  WriteLn(Fr, '-------------------------------------------------');
  WriteLn(Fr, 'Pagrindo ilgis   Aukštis  Panaudotų plytų skaičius');
  WriteLn(Fr, '-------------------------------------------------');
  while p - s >= a + 1  do
   begin
     a := a + 1;
     k := k + 1;
     s := s + a;
     WriteLn(Fr, a : 5, k : 15, s : 20);
   end;
  WriteLn(Fr, '-------------------------------------------------');
  WriteLn(Fr, 'Piramidės pagrindo ilgis: ', a);
  WriteLn(Fr, 'Piramidės aukštis: ', k);
  WriteLn(Fr, 'Piramidės statybai panaudotų plytų skaičius: ', s);
  Close(Fr);          // Įrašius visus rezultatus failas uždaromas
  ReadLn;
end.
```

2.5. Elektros laidininko varžos skaičiavimas

Atlikdami šį darbą išsiaiškinsite, kaip kuriama ciklinius skaičiavimus atliekanti programa, kai žinoma, kiek kartų bus kartojami veiksmai:

- išmoksite užrašyti žinomo kartojimų skaičiaus ciklo antraštę,
- išsiaiškinsite, kaip užrašomi cikle atliekami veiksmai,
- prisiminsite, kaip skaičiuojama suma,
- išmoksite tinkamai išvesti apskaičiuotą realiojo tipo rezultatą.

Nuorodos į Paskalio kalbos žinyną (psl.)		Nuorodos į algoritmų žinyną (psl.)	
3.1. Kintamasis, kintamojo reikšmė	94	4.1. Tiesinis algoritmas	103
3.2. Priskyrimo sakinys	94	4.2. Ciklinis algoritmas	103
3.3. Duomenų įvedimas klaviatūra	95		
3.4. Rezultatų (duomenų) išvedimas ekrane	96		
3.5. Ciklo sakinys while	96		

Užduotis. Elektros laidininką sudaro `n` nuosekliai sujungtų laidininkų, kurių varžos yra `r1, r2, ..., rn` omų. Reikia apskaičiuoti grandinės varžą `r`.

Uždavinio sprendimo algoritmas:

Tarkime, kad elektros grandinę sudaro `n = 4` nuosekliai sujungti laidininkai, kurių varžos yra `r1 = 2, r2 = 4, r3 = 1, r4 = 4`. Grandinės varža skaičiuojama sumuojant visų laidininkų varžas. Žinodami laidininkų skaičių `n` ir kiekvieno laidininko varžą `rlaid`, grandinės varžą `r` galime rasti pagal pagal algoritmą, kurio veiksmai kartojami `n` kartų. Prieš pradedant vykdyti veiksmus, būtina žinoti `n` reikšmę, o kintamojo `r` pradinė reikšmė yra lygi nuliui. Atliekami tokie pasikartojaantys skaičiavimai:

- įvedama laidininko varža `rlaid`;
- skaičiuojama grandinės varža `r = r + rlaid;`

Lentelėje pavaizduota, kaip atliekami veiksmai:

Ciklas vykdomas i-tąjį kartą	Įvedama i-tojo laidininko varža	Skaičiuojama grandinės varža `r = r + rlaid;`
1	2	`r = 0 + 2 = 2`
2	4	`r = 2 + 4 = 6`
3	1	`r = 6 + 1 = 7`
4	4	`r = 7 + 4 = 11`

Skaičiavimų rezultatas: `r = 11.`

Pirmas žingsnis. Pasiruošimas. Sukurkite katalogą `Darbas5`, skirtą programos failams saugoti, iškvieskite `FPS` terpę, sukurkite programos failą ir suteikę programai vardą `Darbas5` įrašykite failą sukurtame kataloge `Darbas5` vardu `Darbas5.pas`.

Antras žingsnis. Programos pradinių duomenų aprašymas ir įvedimas.

- Programos pradžioje aprašykite programoje naudojamus kintamuosius: `n` - laidininkų skaičius, `rlaid` - laidininko varža, `r` - grandinės varža, `i` - žinomo kartojimų skaičiaus ciklo parametras.

```
var     n : integer;   // Laidininkų skaičius
        i : integer;   // Žinomo kartojimų skaičiaus ciklo parametras
    rlaid : real;      // Laidininko varža
        r : real;      // Elektros grandinės varža
```

- Parašykite kintamojo `n` reikšmės įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį `Write` ir reikšmės skaitymo sakinį `ReadLn`.

```
program Darbas5;
var     n : integer;   // Laidininkų skaičius
```

```
        i : integer;   // Žinomo kartojimų skaičiaus ciklo parametras
    rlaid : real;      // Laidininko varža
        r : real;      // Elektros grandinės varža
begin
  Write('Kiek laidininkų yra elektros grandinėje?: '); ReadLn(n);
  ReadLn;
end.
```

- Išsaugokite ir įvykdykite programą.
- Paleidę programą, klaviatūra surinkę skaičių `4` ir paspaudę klavišą Enter, ekrane matysite:

```
Kiek laidininkų yra elektros grandinėje?: 4
```

Trečias žingsnis. Aprašoma pradinė kintamųjo `r` reikšmė.

- Papildykite programą priskyrimo sakiniu:

```
r := 0;
```

- Papildykite programą sakiniu, skirtu `r` reikšmės spausdinimui:

```
program Darbas5;
var     n : integer;   // Laidininkų skaičius
        i : integer;   // Žinomo kartojimų skaičiaus ciklo parametras
    rlaid : real;      // Laidininko varža
        r : real;      // Elektros grandinės varža
begin
  Write('Kiek laidininkų yra elektros grandinėje?: '); ReadLn(n);
  r := 0;
  WriteLn('Elektros grandinės varža: ', r : 6 : 2);
  ReadLn;
end.
```

- Išsaugokite ir įvykdykite programą.
- Paleidę programą, klaviatūra surinkę skaičių `4` ir paspaudę klavišą Enter, ekrane matysite:

```
Kiek laidininkų yra elektros grandinėje?: 4
Elektros grandinės varža:   0.00
```

- Sakinyje `WriteLn('Elektros grandinės varža: ', r : 6 : 2);` apskaičiuota elektros grandinės varža spausdinama skiriant reikšmei 6 pozicijas, iš kurių dvi paskutinės pozicijos skiriamos trupmeninei daliai. Pradedama spausdinti iš dešinės į kairę. Taškui, kuris atskiria trupmeninę dalį nuo sveikosios, taip pat skiriama viena pozicija.

Ketvirtas žingsnis. Ciklo sakinio antraštė. Veiksmai cikle. Kadangi ciklo sakinyje bus atliekami keli veiksmai, jie turi būti rašomi tarp bazinių žodžių `begin` ir `end`. Toks sudėtinis sakinys baigiamas kabliataškiu.

- Po kintamojo `r` pradinės reikšmės aprašymo užrašykite ciklo sakinį:

```
for i := 1 to n do
   begin
     Write('Įveskite laidininko varžą: '); ReadLn(rlaid);
     r := r + rlaid;
   end;
```

- Papildyta programa bus tokia:

```
program Darbas5;
var     n : integer;   // Laidininkų skaičius
        i : integer;   // Žinomo kartojimų skaičiaus ciklo parametras
    rlaid : real;      // Laidininko varža
        r : real;      // Elektros grandinės varža
begin
    Write('Kiek laidininkų yra elektros grandinėje?: '); ReadLn(n);
    r := 0;
    for i := 1 to n do
       begin
          Write('Įveskite laidininko varžą: '); ReadLn (rlaid);
          r := r + rlaid;
       end;
    WriteLn('Elektros grandinės varža: ', r : 6 : 2);
    ReadLn;
end.
```

- Išsaugokite ir įvykdykite programą.
- Paleidę programą, klaviatūra surinkę skaičių `4` ir paspaudę klavišą Enter, ekrane matysite:

```
Kiek laidininkų yra elektros grandinėje?: 4
Įveskite laidininko varžą:
```

- Iš eilės įveskite kiekvieno laidininko varžas: `2, 4, 1, 4`. Po kiekvieno įvedimo paspauskite `Enter` klavišą. Įvedę varžas ekrane matysite:

```
Kiek laidininkų yra elektros grandinėje?: 4
Įveskite laidininko varžą: 2
Įveskite laidininko varžą: 4
Įveskite laidininko varžą: 1
Įveskite laidininko varžą: 4
Elektros grandinės varža:  11.00
```

Užduotys

1. Elektros grandinę sudaro `n` lygiagrečiai sujungtų laidininkų, kurių varžos yra `r1, r2, ..., rn` omų. Parašykite programą, skaičiuojančią grandinės varžą `r`. *Pasitikrinkite, kai n = 4, o r1 = 2, r2 = 4, r3 = 1, r4 = 4, turi būti spausdinama: Elektros grandinės varža r = 0.50 omų.*
2. Klasėje mokosi `n` mokinių. Kiekvieno mokinio ūgis yra `u1, u2, ..., un` centimetrų. Parašykite programą, skaičiuojančią vidutinį klasės mokinių ūgį `uvid`. *Pasitikrinkite, kai n = 5, o u1 = 179, u2 = 180, u3 = 178, u4 = 179, u5 = 175, turi būti spausdinama: Vidutinis klasės mokinio ūgis uvid = 178.20 cm.*
3. Šachmatų išradėjas iš valdovo paprašė tokio atlygio: ant pirmojo šachmatų lentos langelio padėk vieną grūdą, ant antrojo – du, ant trečiojo – keturis ir t.t. vis dvigubink, kol pasibaigs langeliai. Valdovas tik nusijuokė ir paliepė atseikėti grūdų. Kiek grūdų gaus šachmatų išradėjas? Šachmatų lentoje yra 64 langeliai. Parašykite programą šiam uždaviniui spręsti. Uždaviniui spręsti panaudokite `real` duomenų tipą rezultatao reikšmei saugoti.

Smalsiems

Jei pradinių duomenų yra labai daug, tai jų įvedimas klaviatūra reikalauja daug laiko, padarius klaidą tenka viską kartoti iš naujo. Daug patogiau būtų pradinius duomenis perskaityti iš tekstinio failo. Papildysime elektros grandinės varžos skaičiavimo programą duomenų skaitymu iš tekstinio failo.

- Tekstinis failas sukuriamas komandomis `Rinkmena → Kurti → tekstinę rinkmeną`. Atsivėrusio lango pirmoje eilutėje įrašykite skaičių 4, kuris reiškia laidininkų skaičių, antroje eilutėje įrašykite keturių laidininkų varžas, vieną nuo kitos atskirdami tarpais: 2 4 1 4. Ekrane turėtumėte matyti:

```
4
2 4 1 4
```

- Sukurtą failą įrašykite kataloge `Darbas5` pavadindami jį `Darbas5.txt`.
- Papildykite ir pakeiskite sukurtą programą `Darbas5.pas`:

```
program Darbas5;
  var  n : integer;              // Laidininkų skaičius
       i : integer;              // Žinomo kartojimų skaičiaus ciklo parametras
   rlaid : real;                 // Laidininko varža
       r : real;                 // Elektros grandinės varža
      Fd : text;                 // Pradinių duomenų failo kintamasis
begin
  Assign(Fd, 'Darbas4.txt');     // Pradinių duomenų failo kintamasis susiejamas su failo vardu
  Reset(Fd);                     // Tekstinis failas parengiamas skaitymui
  ReadLn(Fd, n);                 // Perskaitomas pirmoje failo eilutėje esantis skaičius. Tai n reikšmė
  r := 0;
  for i := 1 to n do
    begin
     Read(Fd, rlaid);    // Skaitomi tolesni skaičiai. Skaitymas baigiamas, kai perskaitoma n reikšmių
     r := r + rlaid;
    end;
  Close(Fd);                          // Pradinių duomenų failas uždaromas
  WriteLn('Elektros grandinės varža: ', r : 6 : 2);
  ReadLn;
end.
```

- Įvykdę programą ekrane matysite:

```
Elektros grandinės varža:  11.00
```

- Kaip pastebėjote, skaitymas iš failo yra labai panašus į duomenų įvedimą klaviatūra. Skaitant duomenis iš failo nebereikia rašyti sakinių, kuriais prašoma įvesti pradines kintamųjų reikšmes.
- Jei pirmoje failo eilutėje įrašytas laidininkų skaičius n, o likusiose n eilučių laidininkų varžos po vieną eilutėje, tuomet pradinių duomenų failas bus toks:

```
4
2
4
1
4
```

- Norint, kad laidininkų varžų reikšmės būtų perskaitytos tinkamai, sakinį `Read(Fd, rlaid);` užtenka pakeisti sakiniu `ReadLn(Fd, rlaid);` Sakinys `Read` skaito reikšmes iki eilutės pabaigos, o `ReadLn` perskaito kreipinyje nurodytą reikšmių kiekį ir persikelia į kitą eilutę.

2.6. Funkcijos apibrėžimo srities tyrimas

Atlikdami šį darbą išmoksite tinkamai užrašyti sąlygos tikrinimo sakinį.

Nuorodos į Paskalio kalbos žinyną (psl.)		Nuorodos į algoritmų žinyną (psl.)	
3.1. Kintamasis, kintamojo reikšmė	94	4.1. Tiesinis algoritmas	103
3.2. Priskyrimo sakinys	94	4.2. Ciklinis algoritmas	103
3.3. Duomenų įvedimas klaviatūra	95	4.3. Šakotas skaičiavimas	104
3.4. Rezultatų (duomenų) išvedimas ekrane	96		
3.5. Ciklo sakinys while	96		
3.7. Sąlygos sakinys if	97		
3.8. Funkcijų sąrašas	99		

Užduotis. Apskaičiuoti funkcijos $y = \frac{m+3}{\sqrt{m^2-100}}$ reikšmę, kai $m_p \le m \le m_g$ ir kinta žingsniu m_z. Čia m_p, m_g, m_z – realūs skaičiai.

Komentaras. Atkreipkite dėmesį, kad trupmenos vardiklis negali būti lygus nuliui, o pošaknis neigiamas.

Pasiruošimas. Sukurkite katalogą `Darbas6` programos failams saugoti, atverkite FPS terpę ir sukurkite programos failą `Darbas6.pas`.

Pirmas žingsnis. Pradinių duomenų įvedimas klaviatūra.

- Pradiniai duomenys:
  `mp` – funkcijos argumento m pradinė reikšmė;
  `mg` – funkcijos argumento m galinė (paskutinė) reikšmė;
  `mz` – funkcijos argumento m kitimo žingsnis.
- Parašykite dialogo sakinius, skirtus kintamųjų `mp`, `mg`, `mz` reikšmių įvedimui klaviatūra.

```
program Darbas6;
  var mp, mg, mz : real;
begin
  Write('Įveskite argumento pradinę reikšmę:  '); ReadLn (mp);
  Write('Įveskite argumento galinę reikšmę:   '); ReadLn (mg);
  Write('Įveskite argumento žingsnio reikšmę: '); ReadLn (mz);
  ReadLn;
end.
```

- Išsaugokite ir įvykdykite programą su tokiais pradiniais duomenimis: `mp = -50, mg = 50, mz = 10`. Ekrane turėtų būti rodomi tokie rezultatai:

```
Įveskite argumento pradinę reikšmę: -50
Įveskite argumento galinę reikšmę:  50
Įveskite argumento žingsnio reikšmę: 10
```

Antras žingsnis. Funkcijos reikšmių skaičiavimas ir spausdinimas ekrane. Papildykite programą sakiniais:

```
  WriteLn ('Funkcijos reikšmių skaičiavimas');
  WriteLn ('mp = ', mp:5:2, ' mg = ', mg:5:2, ' mz = ', mz:5:2);
  m := mp;
  while m <= mg do
    begin
      if m * m - 100 > 0        // Tikrinama sąlyga, ar m priklauso funkcijos apibrėžimo sričiai
                                // Jei sąlyga tenkinama, skaičiuojama funkcijos reikšmė
        then y := (m + 3) / (Sqrt (m * m - 100));
```

```
    WriteLn(m:7:2, y:12:3);
    m := m + mz;
  end;
```

- Išsaugokite ir įvykdykite programą su tokiais pradiniais duomenimis: `mp = -50, mg = 50, mz = 10.` Ekrane turėtų būti rodomi tokie rezultatai:

```
Įveskite argumento pradinę reikšmę: -50
Įveskite argumento galinę reikšmę: 50
Įveskite argumento žingsnio reikšmę: 10
Funkcijos reikšmių skaičiavimas
mp = -50.00 mg = 50.00 mz = 10.00
 -50.00      -0.959
 -40.00      -0.955
 -30.00      -0.955
 -20.00      -0.981
 -10.00      -0.981
   0.00      -0.981
  10.00      -0.981
  20.00       1.328
  30.00       1.167
  40.00       1.110
  50.00       1.082
```

Trečias žingsnis. Rezultatų analizė.

- Išnagrinėkite gautus rezultatus. Kai `m = -40.00` ir `m = -30.00`, funkcijos y reikšmės yra lygios `-0.955`. Patikrinkite skaičiuotuvu, skaičiuodami `0,00001` tikslumu. Kai `m = -40.00`, turėtumėte gauti `y = 0.95534`, kai `m = -30.00`, `y = 0.95459`. Rašydami rezultatų išvedimą į ekraną pasirinkome rodyti tris skaitmenis trupmeninėje dalyje, todėl ir gavome sutampančius rezultatus.
- Pakeiskite spausdinimo sakinį taip, kad rezultatas į ekraną būtų išvedamas su `5` skaitmenimis trupmeninėje dalyje. Jei viską atlikote teisingai, įvykdę programą ekrane turėtumėte matyti rezultatus:

```
Įveskite argumento pradinę reikšmę: -50
Įveskite argumento galinę reikšmę: 50
Įveskite argumento žingsnio reikšmę: 10
Funkcijos reikšmių skaičiavimas
mp = -50.00 mg = 50.00 mz = 10.00
 -50.00      -0.95938
 -40.00      -0.95534
 -30.00      -0.95459
 -20.00      -0.98150
 -10.00      -0.98150
   0.00      -0.98150
  10.00      -0.98150
  20.00       1.32791
  30.00       1.16673
  40.00       1.11026
  50.00       1.08186
```

- Patikslinę rezultatus pastebime, kad esant argumento m reikšmėms `-20.00, -10.00, 0.00` ir `10.00`, y reikšmė yra lygi `-0.98150`. Tokius rezultatus gavome todėl, kad argumento reikšmės `-10.00, 0.00` ir `10.00` netenkina sąlygos `m * m - 100 > 0`. Funkcijos reikšmės spausdinamos įvykdžius sąlygos sakinį. Jeigu sąlyga netenkinama, tuomet spausdinama paskutinė apskaičiuota `y` reikšmė.

Ketvirtas žingsnis. Tinkamas skaičiavimo rezultatų pateikimas.

- Norėdami gauti teisingus rezultatus visoms argumento `m` reikšmėms, funkcijos reikšmių spausdinimą įkelsime į sąlygos sakinyje atliekamus veiksmus. Jei argumento `m` reikšmė nepriklauso funkcijos apibrėžimo sričiai, tuomet vietoj funkcijos reikšmės spausdinsime septynis žvaigždučių (`*`) simbolius. Sąlygos sakinio šakas papildykite rezultatų spausdinimo sakiniais:

```
if m * m - 100 > 0
        then
          begin
            y := (m + 3) / (Sqrt (m * m - 100));
            WriteLn(m:7:2, y:12:5);
          end
        else WriteLn(m:7:2,'*******':12);
```

- Išsaugokite ir įvykdykite programą. Jei viską atlikote teisingai, įvykdę programą ekrane turėtumėte matyti rezultatus:

```
Įveskite argumento pradinę reikšmę: -50
Įveskite argumento galinę reikšmę: 50
Įveskite argumento žingsnio reikšmę: 10
Funkcijos reikšmių skaičiavimas
mp = -50.00 mg = 50.00 mz = 10.00
 -50.00      -0.95938
 -40.00      -0.95534
 -30.00      -0.95459
 -20.00      -0.98150
 -10.00       *******
   0.00       *******
  10.00       *******
  20.00       1.32791
  30.00       1.16673
  40.00       1.11026
  50.00       1.08186
```

Penktas žingsnis. Skaičiavimo rezultatų pateikimas lentelėje.

- Papildykite programą taip, kad argumento ir funkcijos reikšmės būtų spausdinamos lentelėje.

```
Funkcijos reikšmės nuo -50.00 iki 50.00
-----------------------
    m           y
-----------------------
 -50.00    -0.95938
 -40.00    -0.95534
 -30.00    -0.95459
 -20.00    -0.98150
 -10.00     *******
   0.00     *******
  10.00     *******
  20.00     1.32791
  30.00     1.16673
  40.00     1.11026
  50.00     1.08186
-----------------------
```

Klausimai

1. Kokios bus sveikojo tipo kintamųjų x ir y reikšmės atlikus sakinių seką?

a)
```
x := 5;
if x > 4
   then y := x + 3
   else y := x - 3;
```

b)
```
x := 3;
if x <> 3
   then y := x + 3;
x := x + 2;
y := x + 2;
```

c)
```
x := 6;
if x <= 8
   then
      begin
         x := x + 2;
         y := x + 3;
      end
   else y := x - 3;
```

d)
```
x := 2;
if x < 0
   then y := x - 3
   else
      begin
         x := x + 2;
         y := x + 3;
      end;
```

e)
```
x := 1;
if x > 0
   then
     begin
       y := x - 3;
       x := x + 2;
     end
   else
      begin
         x := x + 2;
         y := x + 3;
      end;
```

f)
```
x := 1;
if x = 0
   then
     begin
       y := x - 3;
       x := x + 2;
     end
   else
      begin
         x := x + 2;
         y := x + 3;
      end;
```

2. Kas bus rodoma ekrane atlikus programos fragmentą? Kokią sąlygą tikrina sąlygos sakinys?

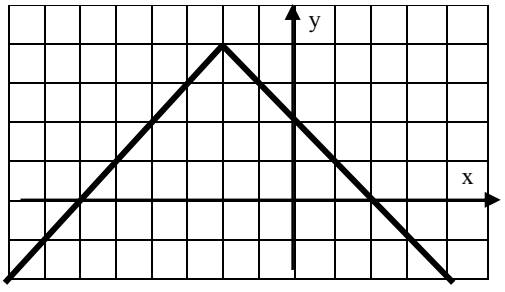
```
for x := 10 to 15 do
   if x mod 2 <> 0
      then WriteLn (x);
```

4. Funkcijos reikšmių skaičiavimui užrašytas sąlygos sakinys. Nubraižykite funkcijos grafiką.

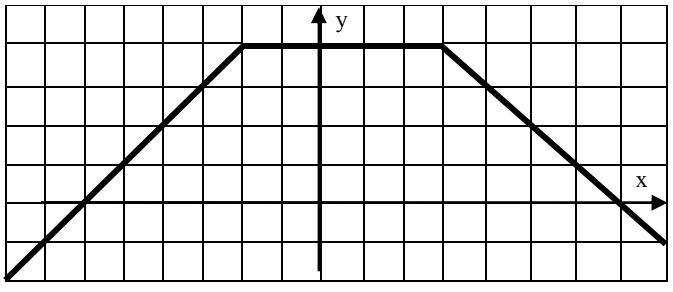
```
if x < 5
   then y := x + 3
   else y := x - 2;
```

7. Remdamiesi funkcijų grafikais užrašykite, kaip skaičiuojamos funkcijų y = f (x) reikšmės. 1 langelis atitinka 1 vienetą.

a)



b)



Atlikdami užduotį užrašykite matematines funkcijų išraiškas.

Užduotys

1. Šviesoforas veikia pagal tokį algoritmą: kiekvienos valandos pradžioje tris minutes dega žalia šviesa, po to dvi minutes – raudona, po to vėl tris minutes žalia ir t.t. Žinoma, kiek minučių t (t – sveikasis skaičius) praėjo nuo valandos pradžios. Parašykite programą, kuri nustatytų, kokia šviesoforo spalva dega. *Pasitikrinkite: kai t = 12, tuomet spausdinama: Dega žalia šviesa. Kai t = 13, tuomet*

spausdinama: Dega žalia šviesa, tuoj degs raudona. Kai t = 5, tuomet spausdinama: Dega raudona šviesa, tuoj degs žalia.

2. Geležinkelio stotys A, B ir C yra n-tajame, m-tajame ir p-tajame geležinkelio ruožo kilometruose. Parašykite programą, kuri surastų, tarp kurių stočių atstumas yra mažiausias. Stotys nebūtinai įvardytos abėcėlės tvarka, pavyzdžiui, po stoties A gali sekti stotis C. *Pasitikrinkite: kai n = 3, m = 8, p = 15, turi būti spausdinama: Atstumas mažiausias tarp A ir B stočių. Kai n = 3, m = 9, p = 15, turi būti spausdinama: Atstumai mažiausi tarp AB ir BC stočių. Kai n = 3, m = 15, p = 9, turi būti spausdinama: Atstumai mažiausi tarp AC ir BC stočių.*
3. Osvaldas nori savaitę slidinėti viename iš trijų kurortų. Kurorte A slidinėjimo sezonas prasideda lapkričio, o baigiasi balandžio mėnesį, bet dėl lavinų pavojaus visą sausio mėnesį slidinėti negalima. Kurorte B slidinėti galima nuo gruodžio pradžios iki kovo pabaigos, tačiau vasario 1-15 dienomis jame vyksta varžybos. Kurorte C slidininkai laukiami nuo lapkričio pradžios iki gegužės pabaigos. Poilsio kaina kiekviename kurorte, įvertinant ir kelionės išlaidas, atitinkamai yra k1, k2, k3 litų. Žinodami atostogų pradžios datą (mėnesį m ir dieną d) nustatykite, ar Osvaldas galės atostogauti bent viename kurorte ir jeigu taip, tai kurį kurortą jam rinktis, kad išleistų mažiausiai pinigų. *Pasitikrinkite: kai m = 2, d = 5, k1 = 500, k2 = 520, k3 = 499, turi būti spausdinama: Osvaldas galės slidinėti C kurorte. Jam reikės 499 Lt.*
4. Pateikiamas dviejų natūraliųjų skaičių a ir b didžiausio bendrojo daliklio (DBD) paieškos algoritmo, dar vadinamo Euklido[2] algoritmu, žodinis aprašymas. Parašykite programą šiam uždaviniui spręsti.
   Euklido algoritmas:
   1. Pradiniai duomenys – natūralieji skaičiai a ir b.
   2. Jei skaičiai yra lygūs, tai bet kuris iš jų yra DBD ir veiksmai toliau neatliekami, jei ne – atliekami tolesni veiksmai.
   3. Nustatoma, kuris skaičius yra didesnis.
   4. Didesniojo skaičiaus reikšme tampa didesniojo ir mažesniojo skaičių skirtumas.
   5. Algoritmo veiksmai kartojami pradedant 2.

[2] Euklidas – senovės graikų matematikas, dabar žinomas kaip „geometrijos tėvas“.

2.7. Kvadratinės lygties sprendinių skaičiavimas

Atlikdami šį darbą įtvirtinsite sąlygos sakinio užrašymo įgūdžius, išmoksite sukurti patogesnę naudotojo sąsają.

Nuorodos į Paskalio kalbos žinyną (psl.)		Nuorodos į algoritmų žinyną (psl.)	
3.3. Duomenų įvedimas klaviatūra		4.1. Tiesinis algoritmas	103
3.4. Rezultatų (duomenų) išvedimas ekrane		4.3. Šakotas skaičiavimas	104
3.7. Sąlygos sakinys if	97		

Užduotis. Rasti kvadratinės lygties $ax^2 + bx + c = 0$ sprendinius. Čia a, b, c – skaičiai, nelygūs nuliui.

Algoritmas. Kvadratinės lygtys sprendžiamos taip:

- Skaičiuojamas diskriminantas: $d = b^2 - 4ac$.
- Tikrinama, ar lygtis turi sprendinių ir jei taip, tai jie skaičiuojami:
  - jei $d < 0$, kvadratinė lygtis neturi realių sprendinių;
  - jei $d = 0$, tuomet kvadratinė lygtis turi vieną sprendinį $x = \frac{-b}{2a}$;
  - jei $d > 0$, tuomet kvadratinė lygtis turi du sprendinius: $x_1 = \frac{-b-\sqrt{d}}{2a}$ ir $x_1 = \frac{-b+\sqrt{d}}{2a}$.

Pasiruošimas. Sukurkite katalogą programos failams saugoti, atverkite FPS terpę ir sukurkite programos failą Darbas7.pas.

Pirmas žingsnis. Pradinių duomenų įvedimas klaviatūra.

- Pradiniai duomenys:

a, b, c – kvadratinės lygties koeficientai.

Rašydami programą, skirtą kvadratinių lygčių sprendinių skaičiavimui, sukursime patogesnę naudotojo sąsają. Prisijungsime biblioteką Crt ir pasinaudosime jos procedūromis:

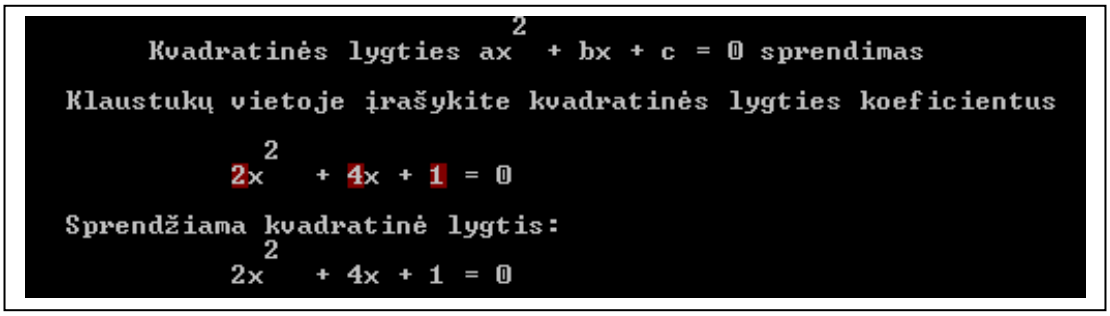
- GotoXY (x, y) – žymeklio padėtis ekrane fiksuojama taške, kurio koordinatės yra (x; y). Šią procedūrą naudosime ekrane įvesdami pradinius duomenis, užrašydami sprendžiamą kvadratinę lygtį, išvesdami lygties sprendinius.
- TextBackground (spalva) – pasirenkama norima teksto fono spalva. Šią procedūrą naudosime įvesdami kvadratinės lygties koeficientus – jie bus įvedami kitos spalvos fone.

- Parašykite sakinius pradinių reikšmių įvedimui klaviatūra.

```
program Darbas7;
  uses Crt;
  var a, b, c, d : integer;  // Lygties koeficientai ir diskriminantas
          x1, x2 : real;     // Lygties šaknys
begin
   GotoXY(10, 2);            // Fiksuojama žymeklio padėtis
   Write('Kvadratinės lygties ax');
   GotoXY(32, 1);      Write('2');
   GotoXY(33, 2);      WriteLn(' + bx + c = 0 sprendimas');
   GotoXY(5, 4);
   WriteLn('Klaustukų vietoje įrašykite kvadratinės lygties koeficientus');
   GotoXY(15, 7);      Write('?x');
   GotoXY(17, 6);      Write('2');
   GotoXY(19, 7);      WriteLn(' + ?x + ? = 0');
   TextBackground (Red);  // Lygties koeficientų vietoje teksto fonas raudonos spalvos
   GotoXy(15, 7);     Read(a);
   GotoXY(22, 7);     Read(b);
```

```
   GotoXY(27, 7);      ReadLn(c);
   TextBackground (Black); // Atstatomas pradinis teksto fonas
   GotoXY(5, 9);       WriteLn('Sprendžiama kvadratinė lygtis:');
   GotoXY(15, 11);     Write(a, 'x');
   GotoXY(17, 10);     Write('2');
   GotoXY(19, 11);     WriteLn(' + ', b,'x + ', c, ' = 0');
   ReadLn;
end.
```

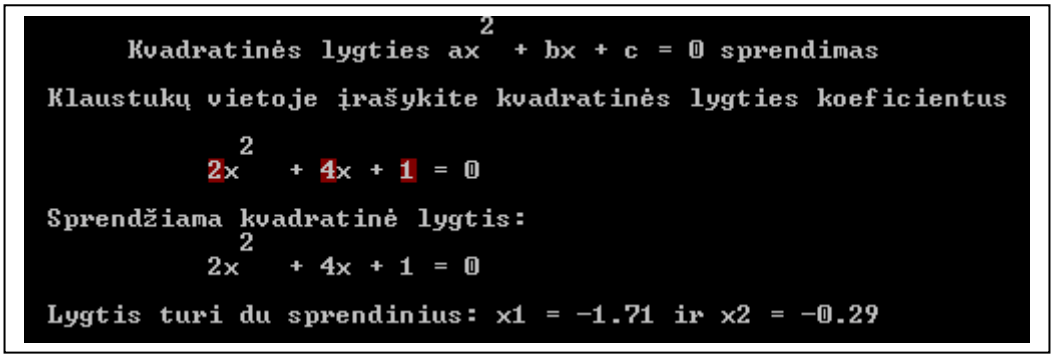
- Išsaugokite ir įvykdykite programą su koeficientų reikšmėmis: `a = 2, b = 4, c = 1`. Ekrane turėtumėte matyti:



Antras žingsnis. Kvadratinės lygties sprendimas ir sprendinių spausdinimas ekrane.

```
   d := b * b - 4 * a * c;
      if d < 0
        then WriteLn('Lygtis neturi realių sprendinių')
         else if d = 0
                then
                  begin
                    x1 := -b / (2 * a);
                    WriteLn('Lygtis turi vieną sprendinį: x = ', x1:0:2);
                  end
                else
                  begin
                    x1 := (-b - Sqrt (d)) / (2 * a);
                    x2 := (-b + Sqrt (d)) / (2 * a);
                    WriteLn('Lygtis turi du sprendinius: x1 = ', x1:0:2,
                            ' ir x2 = ', x2:0:2);
                  end;
```

- Išsaugokite ir įvykdykite programą su koeficientų reikšmėmis: `a = 2, b = 4, c = 1`. Ekrane turėtumėte matyti:



Trečias žingsnis. Programos darbo tikrinimas su įvairiais pradinių duomenų rinkiniais.

- Išspręskite kvadratines lygtis:
  - $x^2 + 14x + 49 = 0;$
  - $x^2 + 12x + 36 = 0;$
  - $x^2 - 8x - 9 = 0;$
  - $x^2 - 6x + 8 = 0;$
  - $x^2 - 3x + 2 = 0;$
  - $x^2 - 5x + 6 = 0;$
  - $x^2 - x + 2 = 0;$
  - $-x^2 + 4x + 1 = 0;$
  - $-5x^2 + 9x - 2 = 0.$

Užduotis.

Pakeiskite programą, kad ji spręstų kvadratines lygtis su realiojo tipo koeficientais. Atkreipkite dėmesį į tai, kad reikės pataisyti žymeklio padėtį ekrane įvedant pradinius duomenis ir spausdinant sprendžiamą kvadratinę lygtį. Laikykite, kad koeficientui įvesti skiriamos 7 pozicijos.

2.8. Vampyro skaičiai

Atlikdami šį darbą susipažinsite su sveikųjų skaičių skaidymu skaitmenimis, naudojant operatorius `div` ir `mod`. Sužinosite kaip kombinatorikos (matematikos šaka) elementus galima taikyti skaičių teorijoje rašant programas. Sužinosite kas yra skaičiaus faktorialas.

Nuorodos į Paskalio kalbos žinyną (psl.)		Nuorodos į algoritmų žinyną (psl.)	
3.2. Priskyrimo sakinys	94	4.2. Ciklinis algoritmas	103
3.3. Duomenų įvedimas klaviatūra	95	4.3. Šakotas skaičiavimas	104
3.4. Rezultatų (duomenų) išvedimas ekrane	96		
3.6. Ciklo sakinys for	97		
3.7. Sąlygos sakinys if	97		

Užduotis. Tarkime, kad turime sveikąjį skaičių `sk` sudarytą iš `n` (`n` - lyginis skaičius) skaitmenų. Jeigu skaičius `sk` turi 2 daugiklius, kiekvienas iš kurių yra sudarytas iš `n/2` skaitmenų, sukonstruotus iš skaičiaus `sk` skaitmenų (visi skaitmenys panaudojami tik po 1 kartą), jis yra vadinamas vampyro skaičiumi, o iš jo skaitmenų sukonstruoti daugikliai – „vampyro iltimis“. Parašykite programą, kuri surastų ir ekrane atspausdintų visus 4-ženklius vampyro skaičius ir jų iltis.

Algoritmas. Prieš pradedant nagrinėti šią užduotį pateiksime keletą 4-ženklių vampyro skaičių ir jų ilčių pavyzdžių:

1260 = 21 * 60
1395 = 15 * 93
1827 = 21 * 87

Kaip pastebite šie 4-ženkliai skaičiai yra išskaidyti į dviejų dviženklių skaičių skaičių sandaugą. Be to šie abu skaičiai turi visus pradinio skaičiaus skaitmenis.

Norint pradėti spręsti šią užduotį pirmiausia reikia mokėti atskirti skaičiaus skaitmenis. Tai padaryti galima naudojant sveikų skaičių dalybos operatorius `div` ir `mod`. Primename, kad pirmasis iš jų leidžia gauti sveikąją dalį, o antrasis dalybos liekaną. Pavyzdžiui, turime keturženklį skaičių `sk = 1234`. Pirmąjį skaičiaus skaitmenį pažymėkime `a`, antrąjį – `b`, trečiąjį – `c`, ketvirtąjį – `d` simboliais, t. y. `sk = abcd`. Dabar parašykime formules, pagal kurias galima rasti šiuos skaitmenis:

```
a =  sk div 1000
b = (sk div 100) mod 10
c = (sk div 10) mod 10
d =  sk mod 10
```

Perkelkime tai kas buvo užrašyta į programą.

Pasiruošimas. Sukurkite katalogą programos failams saugoti, atverkite FPS terpę ir sukurkite programos failą `Darbas8.pas`.
Pradžioje programą rašysime tik vienam 4-ženklio skaičiaus skaidymui atskirais skaitmenimis.

Pirmas žingsnis. Aprašomi kintamieji, skirti pradinių duomenų reikšmėms saugoti. Užrašomi keturi priskyrimo sakiniai pagal aukščiau užrašytas formules. Ekrane spausdinamas gautas rezultatas.

```
program Darbas8;
var
  sk,                      // Keturženklis skaičius
  a, b, c, d : integer;    // Skaičiaus sk (abcd) skaitmenys
begin
  WriteLn('Programa pradėjo darbą');
  sk := 1234;
  a :=  sk div 1000;
```

```
  b := (sk div 100) mod 10;
  c := (sk div 10) mod 10;
  d :=  sk mod 10;
  WriteLn(sk , ' skaitmenys: ', a, ' ', b, ' ', c, ' ', d);
  WriteLn('Programa darbą baigė');
  ReadLn;
end.
```

Paleidę programą, ekrane matysite:

```
Programa pradėjo darbą
1234 skaitmenys: 1 2 3 4
Programa darbą baigė
```

Antras žingsnis. Pirmojo 4-ženklių skaičių dešimtuko skaidymas atskirais skaitmenimis ir rodymas ekrane.

Pakeiskite priskyrimo sakinį `sk := 1234;` ciklo sakiniu **`for`** `sk := 1000` **`to`** `1009` **`do`**

- Užrašykite operatorinius skliaustus `begin .. end.`
- Išsaugokite ir įvykdykite programą.

```
program Darbas8;
var
  sk,                     // Keturženklis skaičius
  a, b, c, d : integer;   // Skaičiaus sk (abcd) skaitmenys
begin
  WriteLn('Programa pradėjo darbą');
  for sk := 1000 to 1009 do
    begin
      a :=  sk div 1000;
      b := (sk div 100) mod 10;
      c := (sk div 10) mod 10;
      d :=  sk mod 10;
      WriteLn(sk , ' skaitmenys: ', a, ' ', b, ' ', c, ' ', d);
    end;
  WriteLn('Programa darbą baigė');
  ReadLn;
end.
```

Paleidę programą, ekrane matysite:

```
Programa pradėjo darbą
1000 skaitmenys: 1 0 0 0
1001 skaitmenys: 1 0 0 1
1002 skaitmenys: 1 0 0 2
1003 skaitmenys: 1 0 0 3
1004 skaitmenys: 1 0 0 4
1005 skaitmenys: 1 0 0 5
1006 skaitmenys: 1 0 0 6
1007 skaitmenys: 1 0 0 7
1008 skaitmenys: 1 0 0 8
1009 skaitmenys: 1 0 0 9
Programa darbą baigė
```

- Išbandykite programą su visais 4-ženkliais skaičiais: `1000..9999`.

Trečias žingsnis. Grįžkime prie užduoties – vampyro skaičių ir jų ilčių. Išskaidžius 4-ženklį skaičių `sk` skaitmenimis `a`, `b`, `c`, `d`, reikia skaitmenis sujungti po du. Galimi šeši keturių skaitmenų sujungimo po du variantai (junginiai):

`ab`, `ac`, `ad`, `bc`, `bd`, `cd`.

Čia galima atpažinti vieną iš kombinatorikos uždavinių – derinių sudarymą. Deriniai kombinatorikoje – baigtinės objektų aibės, turinčios n elementų, junginiai iš k ($1 \le k \le n$) elementų. Derinių skaičius žymimas C_n^k ir randamas pagal formulę:

$$C_n^k = \frac{n(n-1)\cdot...\cdot(n-(k-1))}{k!}$$. Užrašas k! – reiškia skaičiaus faktorialą ir yra lygus skaičių k, k-1, k-2,..., 1 sandaugai, t. y. k! = k·(k-1)·(k-2)·...·1.

Mūsų atveju derinių skaičius $C_4^2 = \frac{4\cdot 3}{2!} = \frac{4\cdot 3}{2\cdot 1} = 6$. Čia 4-ženklio skaičiaus `sk` skaitmenų kiekis n = 4, o skaitmenų kiekis junginyje k = 2. Kaip pastebite apskaičiuotas derinių skaičius 6 atitinka aukščiau užrašytiems 6 junginiams. Kadangi, pagal užduoties sąlygą, gautus junginius reikia grupuoti po du, todėl gaunasi trys poros junginių:

`ab` ir `cd`
`ac` ir `bd`
`ad` ir `bc`

- Programoje pakeiskite sakinį

```
WriteLn(sk , ' skaitmenys: ', a, ' ', b, ' ', c, ' ', d);
```

sekančiais sakiniais:

```
WriteLn(sk , ' junginiu poros:');
WriteLn(' ':20, a,  b, '  ', c, d);
WriteLn(' ':20, a,  c, '  ', b, d);
WriteLn(' ':20, a,  d, '  ', b, c);
```

```
program Darbas7;
var
  sk,                       // Keturženklis skaičius
  a, b, c, d : integer;     // Skaičiaus sk (abcd) skaitmenys
begin
  WriteLn('Programa pradėjo darbą');
  for sk := 1000 to 1009 do
    begin
      a :=  sk div 1000;
      b := (sk div 100) mod 10;
      c := (sk div 10) mod 10;
      d :=  sk mod 10;
      WriteLn(sk , ' junginiu poros:');
      WriteLn(' ':20, a,  b, '  ', c, d);
      WriteLn(' ':20, a,  c, '  ', b, d);
      WriteLn(' ':20, a,  d, '  ', b, c);
    end;
  WriteLn('Programa darbą baigė.');
  Readln;
end.
```

Programą galima patikrinti, pavyzdžiui, su tokiomis ciklo `for` pradžios ir pabaigos reikšmėmis: `1234`.

Paleidę programą, ekrane matysite:

```
Programa pradėjo darbą
1234 junginių poros:
                    12 34
                    13 24
                    14 23
Programa darbą baigė
```

- Išbandykite programą su visais 4-ženkliais skaičiais: 1000..9999.

Ketvirtas žingsnis. Kiekvienai junginių porai (viso 3 poros), sudarykime visas galimas skaitmenų sukeitimo kombinacijas. Kadangi kiekvieną pora sudaro po du skaitmenis, tai kiekvienai porai sudarysime po 4 (2*2) kombinacijas (žiūr. lentelę).

Pirma junginių pora	Antra junginių pora	Trečia junginių pora
ab cd	ac bd	ad bc
ba cd	ca bd	da bc
ab dc	ac db	ad cb
ba dc	ca db	da cb

- Pertvarkykite ankstesnę programą, kiekvieną junginių poros spausdinimo sakinį, pakeisdami keturiais kombinacijų spausdinimo sakiniais. Taigi, programoje vietoje 3-jų spausdinimo sakinių atsiras 12-ka spausdinimo sakinių.

```
program Darbas8;
var
  sk,                     // Keturženklis skaičius
  a, b, c, d : integer;   // Skaičiaus sk (abcd) skaitmenys
begin
  WriteLn('Programa pradėjo darbą');
  for sk := 1234 to 1234 do
    begin
      a :=  sk div 1000;
      b := (sk div 100) mod 10;
      c := (sk div 10) mod 10;
      d :=  sk mod 10;
      WriteLn(sk , ' junginiu poros:');
      WriteLn(' ':20, a, b, ' ', c, d);
      WriteLn(' ':20, b, a, ' ', c, d);
      WriteLn(' ':20, a, b, ' ', d, c);
      WriteLn(' ':20, b, a, ' ', d, c);
      WriteLn;
      WriteLn(' ':20, a, c, ' ', b, d);
      WriteLn(' ':20, c, a, ' ', b, d);
      WriteLn(' ':20, a, c, ' ', d, b);
      WriteLn(' ':20, c, a, ' ', d, b);
      WriteLn;
      WriteLn(' ':20, a, d, ' ', b, c);
      WriteLn(' ':20, d, a, ' ', b, c);
      WriteLn(' ':20, a, d, ' ', c, b);
```

```
      WriteLn(' ':20, d, a, ' ', c, b);
    end;
  WriteLn('Programa darbą baigė');
  ReadLn;
end.
```

- Išsaugokite ir įvykdykite programą. Rezultatas turėtų būti toks:

```
Programa pradėjo darbą
1234 junginių poros:
                    12 34
                    21 34
                    12 43
                    21 43

                    13 24
                    31 24
                    13 42
                    31 42

                    14 23
                    41 23
                    14 32
                    41 32
Programa darbą baigė
```

Penktas žingsnis. Visas aukščiau gautas junginių kombinacijas pavertę dviženkliais skaičiais (pvz.: `a*10+b`), jas sudauginę ir palyginę su pradiniu skaičiumi `sk` galime nustatyti ar skaičius `sk` yra vampyras. Jeigu skaičius yra vampyras bus spausdinamas skaičius ir jo „iltys“.

- Pertvarkykite ankstesnę programą, kiekvieną spausdinimo sakinį (viso 12 sakinių), atitinkamai pakeisdami, Pavyzdžiui, pirmąjį spausdinimo sakinį

  ```
  WriteLn(' ':20, a, b, ' ', c, d);
  ```

  pakeiskite tokiu sakiniu

  ```
  if sk = (a*10+b)*(c*10+d) then
    WriteLn('Vampyras ', sk, ' = ', a, b, ' * ', c, d);
  ```

- Atinkamai pakeiskite ir likusius vienuolika sakinių.
- Pašalinkite iš programos sakinį `WriteLn(sk , ' junginiu poros:');`
- Užrašykite ciklo `for` pradinę reikšmę 1000, o galinę reikšmę 9999.
- Išsaugokite ir įvykdykite programą.

```
program Darbas8;
var
  sk,                     // Keturženklis skaičius
  a, b, c, d : integer;   // Skaičiaus sk (abcd) skaitmenys
begin
  WriteLn('Programa pradėjo darbą');
  for sk := 1000 to 9999 do
    begin
      a :=  sk div 1000;
```

```
      b := (sk div 100) mod 10;
      c := (sk div 10) mod 10;
      d :=  sk mod 10;
      if sk = (a*10+b)*(c*10+d) then
        WriteLn('  Vampyras ', sk, ' = ', a, b, ' * ', c, d);
      if sk = (b*10+a)*(c*10+d) then
        WriteLn('  Vampyras ', sk, ' = ', b, a, ' * ', c, d);
      if sk = (a*10+b)*(d*10+c) then
        WriteLn('  Vampyras ', sk, ' = ', a, b, ' * ', d, c);
      if sk = (b*10+a)*(d*10+c) then
        WriteLn('  Vampyras ', sk, ' = ', b, a, ' * ', d, c);

      if sk = (a*10+c)*(b*10+d) then
        WriteLn('  Vampyras ', sk, ' = ', a, c, ' * ', b, d);
      if sk = (c*10+a)*(b*10+d) then
        WriteLn('  Vampyras ', sk, ' = ', c, a, ' * ', b, d);
      if sk = (a*10+c)*(d*10+b) then
        WriteLn('  Vampyras ', sk, ' = ', a, c, ' * ', d, b);
      if sk = (c*10+a)*(d*10+b) then
        WriteLn('  Vampyras ', sk, ' = ', c, a, ' * ', d, b);

      if sk = (a*10+d)*(b*10+c) then
        WriteLn('  Vampyras ', sk, ' = ', a, d, ' * ', b, c);
      if sk = (d*10+a)*(b*10+c) then
        WriteLn('  Vampyras ', sk, ' = ', d, a, ' * ', b, c);
      if sk = (a*10+d)*(c*10+b) then
        WriteLn('  Vampyras ', sk, ' = ', a, d, ' * ', c, b);
      if sk = (d*10+a)*(c*10+b) then
        WriteLn('  Vampyras ', sk, ' = ', d, a, ' * ', c, b);
    end;
  WriteLn('Programa darbą baigė');
  ReadLn;
end.
```

Įvykdę programą, ekrane matysite:

```
Programa pradėjo darbą
  Vampyras 1260 = 21 * 60
  Vampyras 1395 = 15 * 93
  Vampyras 1435 = 41 * 35
  Vampyras 1530 = 51 * 30
  Vampyras 1827 = 21 * 87
  Vampyras 2187 = 27 * 81
  Vampyras 6880 = 86 * 80
  Vampyras 6880 = 86 * 80
Programa darbą baigė
```

Klausimai

1. Išvardinkite kokius žinote sveikųjų skaičių dalybos operatorius Paskalio kalboje.
2. Koks bus rezultatas `a :=  sk div 1000;`, kur `sk` 4-ženklis sveikas skaičius?
3. Koks bus rezultatas `b := (sk div 100) mod 10;`, kur `sk` 4-ženklis sveikas skaičius?
4. Koks bus rezultatas `c := (sk div 10) mod 10;`, kur `sk` 4-ženklis sveikas skaičius?
5. Koks bus rezultatas `d :=  sk mod 10;`, kur `sk` 4-ženklis sveikas skaičius?

6. Ką gauname bet kokį sveiką skaičių padalinę iš 10 (div 10)?
7. Ką gauname bet kokį sveiką skaičių padalinę iš 10 (mod 10)?

Užduotys

1. Parašykite programą, kuri rastų iš kelių skaitmenų sudarytas duotas sveikasis skaičius `sk`.
2. Parašykite programą, kuri nustatytų, ar duotas sveikasis skaičius `sk` lyginis, ar nelyginis.
3. Parašykite programą, kuri nustatytų, ar duotas sveikasis skaičius `sk` yra palidromas (vienodai skaitomas iš abiejų galų).
4. Parašykite programą, kuri nustatytų, ar duotas sveikasis skaičius `sk` sudarytas vien iš lyginių, vien nelyginių skaitmenų, ar ir iš tokių ir tokių.
5. Parašykite programą, kuri apskaičiuotų sveikojo skaičiaus `sk` faktorialą.
6. Parašykite programą, kuri apskaičiuotų, kiek skirtingų penketukų galima sudaryti iš 10 krepšininkų. Rašydami programą pasinaudokite derinių formule.

Smalsiems

Pirmas papildomas žingsnis. Programa bus aiškesnė ir suprantamesnė, jeigu ją struktūrizuosime, panaudojant procedūrą.

- Sudarykite procedūrą, kuri nustatytų, ar skaičius yra vampyras. Procedūrą pavadinkite tuo pačiu vardu `Vampyras`. Procedūra turės penkis parametrus: 4-ženklį skaičių `sk` ir keturis jo skaitmenis `a`, `b`, `c`, `d`. Jeigu skaičius `sk` bus vampyras, tai procedūra atspausdins skaičių ir jo „iltis“. Programoje į procedūrą `Vampyras` reikės kreiptis 12 kartų su atitinkamais argumentais, t. y. kiekvienas programos `if` sakinys bus pakeistas kreipiniu į šią procedūrą.

```
program Darbas8;

procedure Vampyras(sk, a, b, c, d : integer);
begin
  if sk = (a*10+b)*(c*10+d) then
    WriteLn('  Vampyras ', sk, ' = ', a, b, ' * ', c, d);
end;

var
  sk,                     // Keturženklis skaičius
  a, b, c, d : integer;   // Skaičiaus sk (abcd) skaitmenys
begin
  WriteLn('Programa pradėjo darbą');
  for sk := 1000 to 9999 do
    begin
      a :=  sk div 1000;
      b := (sk div 100) mod 10;
      c := (sk div 10) mod 10;
      d :=  sk mod 10;
      Vampyras(sk, a, b, c, d);
      Vampyras(sk, b, a, c, d);
      Vampyras(sk, a, b, d, c);
      Vampyras(sk, b, a, d, c);

      Vampyras(sk, a, c, b, d);
      Vampyras(sk, c, a, b, d);
      Vampyras(sk, a, c, d, b);
      Vampyras(sk, c, a, d, b);
```

```
      Vampyras(sk, a, d, b, c);
      Vampyras(sk, d, a, b, c);
      Vampyras(sk, a, d, c, b);
      Vampyras(sk, d, a, c, b);
    end;
  WriteLn('Programa darbą baigė.');
  Readln;
end.
```

- Išsaugokite ir įvykdykite programą. Ekrane matysite tokį patį rezultatą, kaip ir ankstesnėje programos versijoje

Antras papildomas žingsnis. Norint aukščiau parašytą programą padaryti dar universalesnę, reikėtų rezultatus pateikti ne ekrane, o rezultatų faile.

- Norint rezultatus pateikti ne ekrane, o rezultatų faile reikia:
  - programą papildyti failo kintamuoju `F`,
  - sugalvoti rezultatų failo vardą, pvz.: `'Vampyrai.txt'`,
  - prisijungti failą ir jį susieti su failo kintamuoju,
  - parengti failą įrašymui,
  - papildyti procedūros `Vampyras` parametrų sąrašą dar vienu kintamuoju (pvz.: `var F : text;`), užrašant jį pirmoje vietoje,
  - procedūroje `Vampyras`, spausdinimo sakinį `WriteLn`, papildykite failo kintamuoju `F`,
  - visuose 12-oje kreipinių į procedūrą `Vampyras` pradžioje užrašyti argumentą `F`,
  - uždaryti tekstinį failą.

Įvykdžius programą ekrane turėtų matytis tik dvi eilutės:

```
Programa pradėjo darbą
Programa darbą baigė
```

Apskaičiuoti rezultatai bus rezultatų faile. Atverkite jį pasinaudodami meniu komanda `Open...`.

2.9. Trys lazdos

Atlikdami šį darbą susipažinsite su loginiais kintamaisiais ir loginiais reiškiniais, loginėmis operacijomis IR, ARBA. Išmoksite loginius reiškinius panaudoti ciklo ir sąlygos sakiniuose.

Nuorodos į Paskalio kalbos žinyną (psl.)		Nuorodos į algoritmų žinyną (psl.)	
3.7. Sąlygos sakinys if	94	4.2. Ciklinis algoritmas	103
3.8. Funkcijų sąrašas	94	4.3. Šakotas skaičiavimas	104

Užduotis. Duota n rinkinių tam tikro ilgio lazdų. Kiekvieną rinkinį sudaro 3 lazdos. Lazdų ilgiai a, b ir c matuojami decimetrais (sveikieji skaičiai). Nustatyti, ar galima iš šių lazdų sudėti trikampį. Jeigu galima, tai kokį: statųjį, lygiakraštį, lygiašonį ar įvairiakraštį. Jeigu negalima, tai užrašyti atitinkamą tekstą.

Algoritmas. Sprendžiant šią užduotį galime įsivaizduoti, kad trys lazdos geometrijoje atitinka tris atitinkamo ilgio atkarpas a, b ir c. Prisiminkite, kad ne visuomet iš trijų atkarpų galima sudėti trikampį. Būtent, trikampio sudėti negalima, jeigu yra bent viena pora atkarpų, kurios ilgių suma yra mažesnė arba lygi už likusiąją trečiąją atkarpą. Arba, galima pasakyti ir taip, kad iš trijų atkarpų trikampį galima sudėti tuomet, kai visų galimų (arba bet kurių) atkarpų porų ilgių suma yra didesnė už trečiąją atkarpą, t. y.

$$(a + b > c) \text{ ir } (a + c > b) \text{ ir } (c + b > a).$$

Pasiruošimas. Sukurkite katalogą programos failams saugoti, atverkite FPS terpę ir sukurkite programos failą `Darbas9.pas`.

Pradžioje programą rašysime tik vienam lazdų rinkiniui. Kadangi vieną rinkinį sudaro trys lazdos, tai lazdų ilgius įvesime vienu `ReadLn` sakiniu ir nustatysime, ar galime sudaryti trikampį, ar ne.

Pirmas žingsnis. Aprašomi kintamieji, skirti pradinių duomenų reikšmėms saugoti, įvedami duomenys.

- Programos pradžioje aprašykite trijų lazdų ilgius saugančius sveikojo tipo kintamuosius `a`, `b`, ir `c`.
- Pakeiskite sakinį `WriteLn('Labas');` nauju sakiniu
  `WriteLn('Programa pradėjo darbą');`
- Parašykite kintamųjų `a`, `b`, ir `c` reikšmių įvedimo klaviatūra sakinius: pranešimo, kokias reikšmes ir kokia eilės tvarka įvesti, sakinį `Write` ir reikšmių skaitymo sakinį `Readln`.
- Programos pabaigoje, prieš sakinį `ReadLn;` užrašykite tokį sakinį:
  `WriteLn('Programa darbą baigė');`
- Išsaugokite ir įvykdykite programą.

```
program Darbas9;
var
  a, b, c : integer;     // Trijų lazdų ilgiai
begin
  WriteLn('Programa pradėjo darbą');
  Write('Įveskite trijų lazdų ilgius: '); ReadLn(a, b, c);
  WriteLn('Programa darbą baigė');
  ReadLn;
end.
```

Paleidę programą, klaviatūra surinkę skaičius `30 50 40` ir paspaudę klavišą `Enter`, ekrane matysite:

```
Programa pradėjo darbą
Įveskite trijų lazdų ilgius: 30 50 40
Programa darbą baigė
```

Antras žingsnis. Pirmojo rezultato – ar iš trijų lazdų rinkinio galima sudėti trikampį – skaičiavimas ir rodymas ekrane.

- Papildykite programą sakiniu `WriteLn`, kuris ekrane spausdintų pradinius duomenis `a`, `b`, `c`.
- Papildykite programą sąlygos sakiniu **`if`** su darbo aprašymo pradžioje pateikto loginio reiškinio skaičiavimu – žodžius „ir“ pakeičiant į **`AND`**.
- Išsaugokite ir įvykdykite programą.

```
program Darbas9;
var
  a, b, c : integer;      // Trijų lazdų ilgiai
begin
  WriteLn('Programa pradėjo darbą');
  Write('Įveskite trijų lazdų ilgius: '); ReadLn(a, b, c);
  Write('Lazdos: ', a:2, ' ', b:2, ' ', c:2);
  if (a + b > c) AND (a + c > b) AND (b + c > a) // Ar trikampis?
    then  WriteLn(' - galima sudaryti trikampį')
    else  WriteLn(' - trikampio sudaryti negalima');
  WriteLn('Programa darbą baigė.');
  ReadLn;
end.
```

Paleidę programą, klaviatūra surinkę skaičius `30 50 40`, bei gale paspaudę klavišą `Enter` , ekrane matysite:

```
Programa pradėjo darbą
Įveskite trijų lazdų ilgius: 30 50 40
Lazdos: 30 50 40 - galima sudaryti trikampį
Programa darbą baigė
```

Dar kartą paleidę programą ir klaviatūra surinkę skaičius `10 50 40`, bei gale paspaudę klavišą `Enter`, ekrane matysite:

```
Programa pradėjo darbą
Įveskite trijų lazdų ilgius: 30 50 40
Lazdos: 10 50 40 - trikampio sudaryti negalima
Programa darbą baigė
```

Trečias žingsnis. Jeigu trikampį galima sudaryti, tai reikia nustatyti, kokio tipo trikampį galima sudaryti.

Statųjį trikampį galima sudaryti, jeigu duotiems dydžiams `a`, `b` ir `c` galioja reiškinys

$(a^2 + b^2 = c^2)$ arba $(a^2 + c^2 = b^2)$ arba $(c^2 + b^2 = a^2)$.

Lygiakraštį trikampį galima sudaryti, jeigu duotiems dydžiams `a`, `b` ir `c` galioja reiškinys

`(a = b)` ir `(b = c)`.

Lygiašonį trikampį galima sudaryti, jeigu duotiems dydžiams `a`, `b` ir `c` galioja reiškinys

`(a = b)` arba `(b = c)` arba `(a = c)`.

Jeigu nei vienas aukščiau užrašytas reiškinys negalioja, tai trikampis yra įvairiakraštis.

- Programoje reikia pakeisti sąlygos sakinio šakoje `then` esantį sakinį

```
WriteLn(' - galima sudaryti trikampį')
```

kitu sudėtiniu salygos sakiniu, kuris analizuos, kokio tipo trikampį galima sudaryti iš trijų duotų lazdų rinkinio.

```
program Darbas9;
var
  a, b, c : integer;     // Trijų lazdų ilgiai
begin
  WriteLn('Programa pradėjo darbą');
  Write('Įveskite trijų lazdų ilgius: '); ReadLn(a, b, c);
  Write('Lazdos: ', a:2, ' ', b:2, ' ', c:2);
  if (a + b > c) AND (a + c > b) AND (b + c > a)  // Ar trikampis?
  then if (a*a + b*b = c*c) OR (a*a + c*c = b*b) OR (b*b + c*c = a*a)
       then WriteLn(' - galima sudaryti statųjį trikampį')
       else if (a = b) AND (b = c)
            then WriteLn(' - galima sudaryti lygiakraštį trikampį')
            else  if (a = b) OR (b = c) OR (a = c)
                  then  WriteLn(' - galima sudaryti lygiašonį trikampį')
                  else  WriteLn(' - trikampio sudaryti negalima');
  WriteLn('Programa darbą baigė');
  ReadLn;
end.
```

Programą reikia patikrinti su penkiais duomenų rinkiniais, pavyzdžiui su tokiais:

`30 50 40` (Statusis trikampis)
`50 50 50` (Lygiakraštis trikampis)
`40 50 40` (Lygiašonis trikampis)
`40 50 60` (Įvairiakraštis trikampis)
`10 50 40` (Trikampis nesusidaro)

Pirmuoju atveju, surinkę duomenis ir paspaudę klavišą `Enter`, ekrane matysite:

```
Programa pradėjo darbą
Įveskite trijų lazdų ilgius: 30 50 40
Lazdos: 30 50 40 – galima sudaryti statųjį trikampį
Programa darbą baigė
```

Analogiški rezultatai turėtų gautis ir likusiais atvejais.

Ketvirtas žingsnis. Pritaikykime gautą programą keliems duomenų – trijų lazdų – rinkiniams. Panaudokime `for` ciklą. Sąlygoje nurodyta, kad skaičiavimus reikia atlikti su `n` duomenų rinkinių.

- Papildykite programą dviem sveikojo tipo kintamaisiais `n` ir `i`. Pirmasis bus skirtas rinkinių skaičiui saugoti, o antrasis rodys, su kuriuo rinkiniu atliekami skaičiavimai.
- Įrašykite dialogo sakinius `n` reikšmės įvedimui.
- Apgaubkite duomenų įvedimo ir salygos sakinius ciklu `for`. Tam reikia naudoti operatorinius skliaustus `begin .. end`, nes cikle bus atliekamas ne vienas , o keli sakiniai.
- Pastumkite cikle `for` tarp `begin` ir `end` esantį tekstą į dešinę. Tai atlikus programos tekstas taps vaizdesnis, o tuo pačiu ir suprantamesnis.
- Įterpkite į duomenų įvedimo dialogo sakinį `WriteLn` ciklo kintamąjį `i`.

```
program Darbas9;
var
  a, b, c : integer;     // Trijų lazdų ilgiai
  n,                     // Lazdų rinkinių skaičius
  i  : integer;          // Ciklo kintamasis
```

```
begin
  WriteLn('Programa pradėjo darbą');
  Write('Iveskite kiek lazdų rinkiniu bus: '); ReadLn(n);
  for i := 1 to n do
  begin
   Write('Iveskite trijų lazdų ', i, '-ąjį rinkinį: '); ReadLn(a, b, c);
   Write('Lazdos: ', a:2, ' ', b:2, ' ', c:2);
   if (a + b > c) AND (a + c > b) AND (b + c > a)  // Ar trikampis?
   then if (a*a + b*b = c*c) OR (a*a + c*c = b*b) OR (b*b + c*c = a*a)
        then WriteLn(' - galima sudaryti statųjį trikampį')
        else if (a = b) AND (b = c)
             then WriteLn(' - galima sudaryti lygiakraštį trikampį')
             else  if (a = b) OR (b = c) OR (a = c)
                   then  WriteLn(' - galima sudaryti lygiašonį trikampį')
                   else  WriteLn(' - trikampio sudaryti negalima');
  end;
  WriteLn('Programa darbą baigė.');
  ReadLn;
end.
```

- Išsaugokite ir įvykdykite programą su ankstesniais pradiniais duomenimis. `n` reikšmę nurodykite `5`. Rezultatas turėtų būti toks:

```
Programa pradėjo darbą
Įveskite kiek rinkinių lazdų bus: 5
Įveskite trijų lazdų 1-ąjį rinkinį: 30 50 40
Lazdos: 30 50 40 - galima sudaryti statųjį trikampį
Įveskite trijų lazdų 2-ąjį rinkinį: 50 50 50
Lazdos: 50 50 50 - galima sudaryti lygiakraštį trikampį
Įveskite trijų lazdų 3-ąjį rinkinį: 40 50 40
Lazdos: 40 50 40 - galima sudaryti lygiašonį trikampį
Įveskite trijų lazdų 4-ąjį rinkinį: 40 50 60
Lazdos: 40 50 60 - galima sudaryti įvairiakraštį trikampį
Įveskite trijų lazdų 5-ąjį rinkinį: 10 50 40
Lazdos: 10 50 40 - trikampio sudaryti negalima
Programa darbą baigė
```

Klausimai

1. Išvardinkite kokius žinote loginius operatorius ir kaip jie užrašomi Paskalio kalboje.
2. Kaip suprantate loginį reiškinį? Paaiškinkite.
3. Kokias reikšmes gali įgauti loginis reiškinys?
4. Kokie operatoriai dažniausiai naudojami aprašant loginį reiškinį?
5. Kokias reikšmes gali įgauti loginis reiškinys?

Užduotys

1. Nubraižykite ir užpildykite loginio operatoriaus IR veiksmų lentelę.
2. Nubraižykite ir užpildykite oginio operatoriaus ARBA veiksmų lentelę.
3. Nubraižykite ir užpildykite loginio operatoriaus NE veiksmų lentelę.
4. Užrašykite dialogo sakinius taško (xt, yt) koordinatėms įvesti.

5. Duotos stačiakampio priešingų kampų koordinatės: kairiojo viršutinio kampo (x1, y1) ir dešinio apatinio kampo (x3, y3). Užrašykite loginius reiškinius:
   - Taško priklausomybei stačiakampio viduje nustatyti,
   - Taško priklausomybei stačiakampio išorėje nustatyti,
   - Taško priklausomybei stačiakampio apatinėje kraštinėje nustatyti,
   - Taško priklausomybei stačiakampio viršutinėje kraštinėje nustatyti,
   - Taško priklausomybei stačiakampio kairėje kraštinėje nustatyti,
   - Taško priklausomybei stačiakampio dešinėje kraštinėje nustatyti.
6. Papildykite programą taip, kad ji rastų ir ekrane parodytų gautų trikampių plotus. Naudokite Herono formulę trikampio plotui apskaičiuoti.
7. Trikampis vienu metu gali būti ir dviejų tipų: statusis ir lygiašonis, lygiašonis ir lygiakraštis. Pakeiskite programą taip, kad ji tai įvertintų.
8. Panaudokite programoje skirtingas spalvas, skirtingiems trikampių tipams ekrane pažymėti.

Smalsiems

Pirmas papildomas žingsnis. Norint aukščiau parašytą programą padaryti universalesnę, reikėtų duomenis programai įvesti ne klaviatūra, bet imti iš iš anksto paruošto duomenų failo. Lygiai taip pat rezultatus reikėtų pateikti ne ekrane, o rezultatų faile.

- Norėdami duomenis imti iš failo pirmiausia reikėtų susikurti duomenų failą, pvz.: `'Lazdos.txt'`:

```
5
30 50 40
50 50 50
40 50 40
40 50 60
10 50 40
```

Pirmoje failo eilutėje užrašytas lazdų rinkinių skaičius n, kitose eilutėse po tris skaičius surašyti lazdų rinkiniai.

- Ką reikia padaryti:
  - programą papildyti failo kintamuoju,
  - prisijungti failą ir jį susieti su failo kintamuoju,
  - parengti failą skaitymui,
  - papildyti `n` ir `a`, `b`, `c` ivedimo sakinius failo kintamuoju,
  - pašalinti sekančius sakinius kaip nereikalingus
    ```
    Write('Iveskite kiek lazdų rinkiniu bus: ');
    Write('Iveskite trijų lazdų ', i, '-ąjį rinkinį: ');
    ```
  - uždaryti tekstinį failą.
- Išsaugokite ir įvykdykite programą.
- Palyginkite gautus rezultatus su ankstesniais rezultatais. Jeigu viską teisingai atlikote, tai jie turėtų sutapti.
- Panašius veiksmus reikėtų atlikti, norint rezultatus pateikti ne ekrane, o rezultatų faile:
  - programą papildyti dar vienu failo kintamuoju,
  - sugalvoti rezultatų failo vardą, pvz.: `'Rezultatai.txt'`,
  - prisijungti failą ir jį susieti su failo kintamuoju,
  - parengti failą įrašymui,
  - papildyti išvedimo sakinius failo kintamuoju,

Įvykdžius programą ekrane turėtų matytis tik dvi eilutės:

```
Programa pradėjo darbą
Programa darbą baigė
```

Apskaičiuoti rezultatai bus rezultatų faile. Atverkite jį pasinaudodami meniu komanda `Open...`.

Antras papildomas žingsnis. Programa bus aiškesnė ir suprantamesnė, jeigu ją struktūrizuosime, panaudojant funkcijas. Be to žinome, kad trikampis gali turėti kelias savybes, pavyzdžiui būti status ir lygiašonis. Todėl reikėtų patikrinti kiekvienam trikampiui atskirai, kokias savybes jis tenkina.

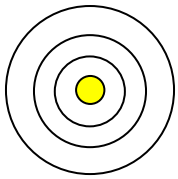
```
program Darbas9;
var
  a, b, c : integer;     // Trijų lazdų ilgiai
  n,                     // Lazdų rinkinių skaičius
  i  : integer;          // Ciklo kintamasis
//-----------------------------------------------
function ArTrikampis: boolean;
begin
   ArTrikampis := (a + b > c) AND (a + c > b) AND (b + c > a);
end;
//-----------------------------------------------
function ArStatus: boolean;
begin
  ArStatus := (a*a + b*b = c*c) OR (a*a + c*c = b*b) OR (b*b + c*c = a*a)
end;
//-----------------------------------------------
function ArLygiakrastis: boolean;
begin
  ArLygiakrastis := (a = b) AND (b = c)
end;
//-----------------------------------------------
function ArLygiasonis: boolean;
begin
  ArLygiasonis := (a = b) OR (b = c) or (a = c)
end;
//-----------------------------------------------
function ArIvairiakrastis: boolean;
begin
    ArIvairiakrastis := (a <> b) AND (a <> c) AND (b <> c);
end;
//-----------------------------------------------
var fd, fr : text;
begin
  WriteLn('Programa pradėjo darbą');
  Assign(fr, 'Rezultatai.txt'); Rewrite(fr);
  Assign( fd, 'Lazdos.txt'); Reset(fd);
  ReadLn(fd, n);
  for i := 1 to n do
  begin
    ReadLn(fd, a, b, c);
    WriteLn(fr, 'Lazda: ', a:2, ' ', b:2, ' ', c:2);
    if ArTrikampis // Ar trikampis?
    then  begin
```

```
      if ArStatus then
                  WriteLn(fr, ' - galima sudaryti statųjį trikampį');
      if ArLygiakrastis then
                  WriteLn(fr, ' - galima sudaryti lygiakraštį trikampį');
      if ArLygiasonis then
                  WriteLn(fr, ' - galima sudaryti lygiašonį trikampį');
      if ArIvairiakrastis then
                  WriteLn(fr, ' - galima sudaryti įvairiakraštį trikampį');
     end
     else  WriteLn(fr, ' - trikampio sudaryti negalima');
   end;
  Close(fr);
  WriteLn('Programa darbą baigė.');
  Readln;
end.
```

- Išsaugokite ir įvykdykite programą. Rezultatų faile matysite:

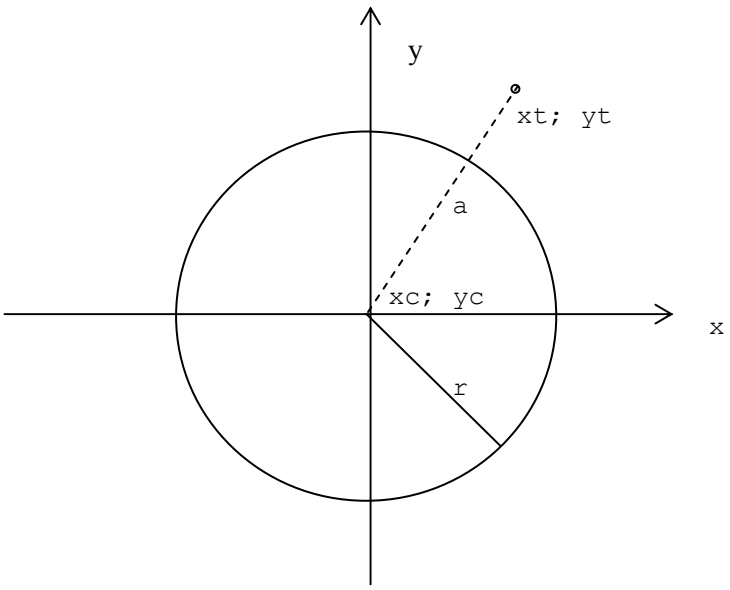
```
Lazda: 30 50 40
 - galima sudaryti statųjį trikampį
 - galima sudaryti įvairiakraštį trikampį
Lazda: 50 50 50
 - galima sudaryti lygiakraštį trikampį
 - galima sudaryti lygiašonį trikampį
Lazda: 40 50 40
 - galima sudaryti lygiašonį trikampį
Lazda: 40 50 60
 - galima sudaryti įvairiakraštį trikampį
Lazda: 10 50 40
 - trikampio sudaryti negalima
```

2.10. Taikinys



Atlikdami šį darbą įtvirtinsite sąlygos sakinio užrašymo įgūdžius. Sužinosite, kaip nustatyti, ar du realūs skaičiai lygūs.
Smalsiems siūlomas kitas programos variantas. Čia taikinio duomenys skaitomi iš tekstinio failo, todėl pakartotinai vykdant programą, nereikia klaviatūra suvedinėti daug pradinių duomenų. Jie surašomi tekstiniame faile ir saugomi tol, kol nepakeičiami naujais. Taip pat čia jūs susipažinsite, kaip, panaudojant procedūras, galima supaprastinti programos struktūrą. Atskirus veiksmus užrašius procedūromis, kiekviena jų tampa lengviau skaitoma, pagrindinė programa tampa veiksmų nuoseklumo valdymo priemone.

Užduotis. Vyksta šaudymo iš lanko varžybos. Taikinys pritvirtintas prie lentos, turinčios elektroninius daviklius. Lentos apatinio kairiojo kampo koordinatės yra (0; 0). Taikinio centras pažymėtas juodu skrituliu. Aplink skritulį yra nubrėžti dar trys apskritimai. Strėlei pataikius į skirtingas taikinio vietas yra skiriamas skirtingas taškų skaičius. Reikia parašyti programą, kuri, priklausomai nuo strėlės pataikymo vietos, skirtų sportininkui taškus. Yra žinomi taikinio duomenys: taikinio centro koordinatės (xc; yc), apskritimų spinduliai, strėlės pataikymo vietos koordinatės (x; y) ir taškų vertės. Jeigu strėlė pataikė ant apskritimo linijos, tuomet sportininkas gauna pusę taškų, kurie būtų skiriami, jeigu taškas būtų apskritimo viduje.
Algoritmas. Jei taškas nuo apskritimo centro yra atstumu a < r, tuomet taškas bus apskritimo viduje, jei a = r – taškas bus ant apskritimo, jei a > r – taškas bus apskritimo išorėje. Bet kurio taško koordinatės (x, y) yra realieji skaičiai. Nustatyti, ar du realieji skaičiai yra lygūs, neįmanoma, nes kiekvienas jų bendru atveju po kablelio turi daug skaitmenų. Kompiuteryje realieji skaičiai saugomi tam tikru tikslumu, todėl skaičiai laikomi lygiais tuomet, kai skirtumas tarp jų yra nedidelis. Yra laikoma, pavyzdžiui, kad du realieji skaičiai yra lygūs, jeigu pirmi penki skaitmenys po kablelio yra vienodi. Daugeliu atvejų atliekant skaičiavimus su realiaisiais skaičiais būtina žinoti, kokiu tikslumu reikia skaičiuoti.
Pavaizduokime situaciją grafiškai:



Atstumas `ats` tarp dviejų taškų skaičiuojamas pagal formulę:

$$ats = \sqrt{(xc - xt)^2 + (yc - yt)^2}$$

Pasiruošimas. Sukurkite katalogą programos failams saugoti, atverkite FPS terpę ir sukurkite programos failą `Darbas10.pas`. Uždavinį suskaidysime į dalis:

- pradinių duomenų skaitymas;
- atstumo a skaičiavimas;

- tikrinimas, kur yra taškas;

Pirmas žingsnis. Pradinių duomenų skaitymas. Aprašomi kintamieji pradiniams duomenims skaityti ir užrašomi duomenų skaitymo sakiniai.

```
program Darbas10;
var     xc, yc : real;              // Taikinio centro koordinatės
        r1, r2, r3, r4 : real;      // Apskritimų spinduliai didėjimo tvarka
        t1, t2, t3, t4 : integer;   // Skiriami taškai atitinkamai pagal apskritimus
        x, y : real;                // Strėlės koordinatės
begin
                                    // Duomenų skaitymas
 Write ('Taikinio centro koordinatė xc =  ');     ReadLn (xc);
 Write ('Taikinio centro koordinatė yc =  ');     ReadLn (yc);
 Write ('Pirmojo apskritimo spindulys r1 = ');    ReadLn (r1);
 Write ('Pirmojo apskritimo taškai    t1 = ');    ReadLn (t1);
 Write ('Antrojo apskritimo spindulys r2 = ');    ReadLn (r2);
 Write ('Antrojo apskritimo taškai    t2 = ');    ReadLn (t2);
 Write ('Trečiojo apskritimo spindulys r3 =  ');  ReadLn (r3);
 Write ('Trečiojo apskritimo taškai    t3 =  ');  ReadLn (t3);
 Write ('Ketvirtojo apskritimo spindulys r4 = '); ReadLn (r4);
 Write ('Ketvirtojo apskritimo taškai    t4 = '); ReadLn (t4);
 Write ('Strėlės koordinatė x = '); ReadLn (x);
 Write ('Strėlės koordinatė y = '); ReadLn (y);
 ReadLn;
end.
```

- Išsaugokite programą ir patikrinkite kaip veikia.

Antras žingsnis. Strėlės padėčiai nagrinėti reikalingas jos pataikymo į taikinį taško atstumas iki apskritimo centro. Jam skaičiuoti parašykite priskyrimo sakinį

```
ats := Sqrt (Sqr (xc - x) + Sqr (yc - y));
```

- Papildykite programos kintamųjų sąrašą nauju realiojo tipo kintamuoju `ats`.
- Parašykite sakinį, kuris į ekraną išveda apskaičiuotą atstumo `ats` reikšmę.
- Išsaugokite programą ir patikrinkite, kaip ji veikia. Įvedę pasirinktus duomenis ekrane matysime, pavyzdžiui, tokias eilutes:

```
Taikinio centro koordinatė xc =  60
Taikinio centro koordinatė yc =  80
Pirmojo apskritimo spindulys r1 = 10
Pirmojo apskritimo taškai    t1 = 10
Antrojo apskritimo spindulys r2 = 15
Antrojo apskritimo taškai    t2 = 8
Trečiojo apskritimo spindulys r3 =  20
Trečiojo apskritimo taškai    t3 =  6
Ketvirtojo apskritimo spindulys r4 = 30
Ketvirtojo apskritimo taškai    t4 = 2
Strėlės koordinatė x = 55
Strėlės koordinatė y = 45
Pataikymo taško atstumas iki taikinio centro: 35.35534
```

Trečias žingsnis. Išvados, kur yra taškas nurodyto apskritimo atžvilgiu, formulavimui reikia žinoti, kokiu tikslumu atlikti apskritimo spindulio ir taško atstumo iki apskritimo centro palyginimą. Tam reikia ivesti skaičiavimų tikslumo reikšmę, kuriai saugoti reikia aprašyti realaus tipo kintamąjį, kurį galime pavadinti `tiks`.

- Papildykite programą skaičiavimų tikslumo reikšmės įvedimo sakiniais.
- Išsaugokite programą ir patikrinkite, kaip ji dirba. Įvedus pasirinktus duomenis ekrane matysime, pavyzdžiui, tokias eilutes:

```
Taikinio centro koordinatė xc =  60
Taikinio centro koordinatė yc =  80
Pirmojo apskritimo spindulys r1 = 10
Pirmojo apskritimo taškai    t1 = 10
Antrojo apskritimo spindulys r2 = 15
Antrojo apskritimo taškai    t2 = 8
Trečiojo apskritimo spindulys r3 =  20
Trečiojo apskritimo taškai    t3 =  6
Ketvirtojo apskritimo spindulys r4 = 30
Ketvirtojo apskritimo taškai    t4 = 2
Strėlės koordinatė x = 55
Strėlės koordinatė y = 45
Pataikymo taško atstumas iki taikinio centro: 35.35534
Skaičiavimų tikslumas: 0.01
```

Ketvirtas žingsnis. Papildykite programą nauju sveikojo tipo kintamuoju `tsk`, skirtu taškams įsiminti. Programoje parašykite sąlygos sakinį, kuris patikrina taško padėtį apskritimų atžvilgiu ir skaičiuoja taškus:

```
if Abs(ats - r1) < tiks  then tsk := t1 div 2
 else if  ats < r1 then tsk := t1
      else if Abs(ats - r2) < tiks then tsk := t2 div 2
           else if ats < r2 then tsk := t2
                else if Abs(ats - r3) < tiks then tsk := t3  div 2
                     else if ats < r3 then tsk := t3
                          else if Abs(ats - r4) < tiks then tsk := t4 div 2
                               else if ats < r4 then tsk := t4
                                    else tsk := 0;
```

Sąlygos sakinyje pirmiausiai reikia patikrinti, ar taškas yra ant apskritimo. Jeigu duotu tikslumu gauname atsakymą, kad jis yra ne ant apskritimo, tuomet galima tikrinti, ar viduje, ar išorėje. Tokia seka reikalinga tam, kad išskirtume atvejį, ar skaičiai lygūs. Tikrinti pradedame nuo vidinio apskritimo.

- Papildykite programą apskaičiuotų taškų reikšmei spausdinti.
- Patikrinkite, ar programa veikia teisingai. Tam reikia paruošti tiek duomenų testinių variantų, kiek yra taškų skaičiavimo variantų: kiekvienam apskritimui, kai taškas yra jo viduje, kai ant apskritimo ir kai jo išorėje. Jeigu programos pateikiami rezultatai sutampa su testų rezultatais, tuomet galime teigti, kad programa skaičiuoja teisingai. Norint įsitikinti, ar gerai skaičiuojamas atstumas, testiniuose duomenų variantuose reikia apskaičiuoti taško atstumus iki apskritimo centro. Tikslinga patikrinti, ar programa gerai skaičiuoja, kai apskritimas ir taškas yra skirtingose stačiakampės koordinačių plokštumos vietose.

Klausimai

1. Kodėl du realiuosius skaičius reikia lyginti tam tikru tikslumu?
2. Kodėl lyginant du realiuosius skaičius iš pradžių reikia patikrinti, ar jie lygūs, o po to nustatyti pirmas didesnis už antrą, ar mažesnis?

3. Jeigu vietoje apskritimo būtų stačiakampis, kurio kraštinės yra lygiagrečios koordinačių ašims, tai kaip reikėtų parašyti patikrinimą, ar taškas yra viduje, ar išorėje, ar ant kurios nors kraštinės?

Užduotys

1. Papildykite programą, kuri skaičiuotų taškus, kai sportininkas šovė iš lanko kelis kartus. Tam reikia strėlės pataikymo koordinates, atstumo skaičiavimą ir taškų skaičiavimą atlikti cikle. Papildomu kintamuoju reikia saugoti, kiek kartų sportininkas šovė.
2. Ne visuomet taikiniuose yra apskritimai. Pakeiskite programą, kai vietoje apskritimų turime stačiakampius, kurių kraštinės lygiagrečios koordinačių ašims.
3. Parašykite programą, kuri nustatytų, ar du apskritimai kertasi, jeigu yra žinomi jų spinduliai ir centrų koordinatės.

Smalsiems

Pirmas žingsnis. Pradinių duomenų skaitymas.

- Pradiniai duomenys surašomi tekstiniame faile. Pavyzdžiui faile `TaikinysDuom.txt` taip:

TaikinysDuom.txt	Paaiškinimai
60 80	Taikinio centro koordinatės xc ir yc
10 10	Pirmojo apskritimo spindulys ir skiriami taškai
15 8	Antrojo apskritimo spindulys ir skiriami taškai
20 6	Trečiojo apskritimo spindulys ir skiriami taškai
30 2	Ketvirtojo apskritimo spindulys ir skiriami taškai
0.01	Skaičiavimų tikslumas

Tai pastovūs duomenys. Derinant programą nereikės kiekvieną kartą suvedinėti taikinio duomenų. Jeigu bus skaičiuojami taškai ne vienam šūviui, tai tereikės tik strėlės pataikymo koordinates įvesti. Jeigu bus kitas taikinys, tuomet duomenų faile turėsite duomenis surašyti iš naujo.

- Aprašomi pradinių duomenų kintamieji.

```
program Darbas10;
var    xc, yc : real;              // Taikinio centro koordinatės
       r1, r2, r3, r4 : real;      // Apskritimų spinduliai didėjimo tvarka
       t1, t2, t3, t4 : integer;   // Skiriami taškai atitinkamai pagal apskritimus
       tiks : real;                // Skaičiavimų tikslumas
```

- Parašoma procedūra duomenims iš failo skaityti ir kreipinys į skaitymo procedūrą pagrindinėje programoje. Procedūros tekstas programoje turi būti prieš pagrindinės programos pradžią, kurią žymi žodis `begin`.

```
//-------------------------------------------------
procedure SkaitytiDuomenis;
  var Fd : text;
begin
  Assign(Fd, 'TaikinysDuom.txt');  Reset(Fd);
  ReadLn(Fd, xc, yc);
  ReadLn(Fd, r1, t1);
  ReadLn(Fd, r2, t2);
  ReadLn(Fd, r3, t3);
  ReadLn(Fd, r4, t4);
  ReadLn(Fd, tiks);
  Close(Fd);
```

```
end;
//--------- Pagrindinė programa (manau, kad reiktų akcentuoti)
begin
  SkaitytiDuomenis;          // Duomenų skaitymas
  ReadLn;
end.
```

- Išsaugokite programą ir patikrinkite kaip veikia. Jeigu jokių klaidų nėra, tuomet galite įsitikinti, ar duomenys iš failo perskaityti teisingai. Tam galite parašyti kitą procedūrą:

```
//-----------------------------------------------
procedure RodytiDuomenis;
begin
   WriteLn('xc = ', xc:5:2, ' yc = ', yc:5:2);
   WriteLn('r1 = ', r1:5:2, ' t1 = ', t2:5);
   WriteLn('r2 = ', r1:5:2, ' t2 = ', t2:5);
   WriteLn('r3 = ', r1:5:2, ' t3 = ', t2:5);
   WriteLn('r4 = ', r1:5:2, ' t4 = ', t2:5);
   WriteLn('Tikslumas: ', tiks:8:5);
end;
//-----------------------------------------------
```

Procedūra `RodytiDuomenis` rašoma toliau po duomenų skaitymo procedūros, tačiau prieš pagrindinę programą.

- Pagrindinėje programoje parašykite kreipinį į procedūrą: `RodytiDuomenis;`
- Išsaugokite programą ir patikrinkite kaip veikia. Jeigu jokių klaidų nėra, tuomet ekrane matysite pradinius duomenis

```
xc= 60.00 yc = 80.00
r1= 10.00 t1 =    10
r2= 15.00 t2       8
r3= 20.00 t3=      6
r4= 30.00 t4=      2
Tikslumas: 0.01000
```

Antras žingsnis. Strėlės padėčiai nagrinėti reikalingas jos pataikymo į taikinį taško atstumas iki apskritimo centro. Jam skaičiuoti toliau pagrindinėje programoje parašykite priskyrimo sakinį:

```
ats := Sqrt (Sqr (xc - x) + Sqr (yc - y));
```

- Papildykite programos kintamųjų sąrašą nauju realiojo tipo kintamuoju `ats`.
- Parašykite sakinį, kuris į ekraną išveda apskaičiuotą atstumo `ats` reikšmę.
- Išsaugokite programą ir patikrinkite, kaip ji veikia. Įvedę pasirinktus duomenis ekrane matysime, pavyzdžiui, tokias eilutes:

```
xc= 60.00 yc = 80.00
r1= 10.00 t1 =    10
r2= 15.00 t2       8
r3= 20.00 t3=      6
r4= 30.00 t4=      2
Tikslumas: 0.01000
Strėlės koordinatė x = 75
Strėlės koordinatė y = 65
Pataikymo taško atstumas iki taikinio centro:  21.21320
```

Trečias žingsnis. Taškų skaičiavimas.

- Papildykite programą nauju sveikojo tipo kintamuoju `tsk`, skirtu taškams įsiminti. Parašykite procedūrą, kurioje parašykite sąlygos sakinį, kuris patikrina taško padėtį apskritimų atžvilgiu ir skaičiuoja taškus:

```
//---------------------------------------------------------------------
procedure Taskai;
begin
 if Abs(ats - r1) < tiks  then tsk := t1 div 2
 else if  ats < r1 then tsk := t1
      else if Abs(ats - r2) < tiks then tsk := t2 div 2
           else if ats < r2 then tsk := t2
                else if Abs(ats - r3) < tiks then tsk := t3  div 2
                     else if ats < r3 then tsk := t3
                          else if Abs(ats - r4) < tiks then tsk := t4 div 2
                               else if ats < r4 then tsk := t4
                                    else tsk := 0;
end;
//---------------------------------------------------------------------
```

Sąlygos sakinyje pirmiausia reikia patikrinti, ar taškas yra ant apskritimo. Jeigu duotu tikslumu gauname atsakymą, kad jis yra ne ant apskritimo, tuomet galima patikrinti, ar viduje, ar išorėje. Tokia seka reikalinga tam, kad išskirtume atvejį, ar skaičiai lygūs. Tikrinti pradedame nuo vidinio apskritimo.

- Papildykite programą kintamaisiais x ir y, skirtais strėlės pataikymo į taikinį koordinatėms saugoti.
- Papildykite programą sakiniais, skirtais strėlės pataikymo į taikinį koordinačių reikšmių įvedimui klaviatūra:

```
Write ('Strėlės koordinatė x = '); ReadLn (x);
Write ('Strėlės koordinatė y = '); ReadLn (y);
```

- Papildykite programą kreipiniu į procedūrą suskaičiuotų taškų reikšmei spausdinti. Pagrindinė programa bus tokia:

```
begin
 SkaitytiDuomenis;                              // Duomenų skaitymas
 RodytiDuomenis;                                // Duomenys rodomi ekrane
 Write ('Strėlės koordinatė x = ');  ReadLn (x);
 Write ('Strėlės koordinatė y = ');  ReadLn (y);
 ats := Sqrt (Sqr (xc - x) + Sqr (yc - y));  // Atstumo skaičiavimas
 WriteLn('Pataikymo taško atstumas iki taikinio centro: ', ats:10:5);
 Taskai;                                        // Taškų skaičiavimas
 WriteLn('Gauti taškai: ', tsk);
 ReadLn;
end.
```

- Išsaugokite programą ir patikrinkite kaip ji veikia.
- Patikrinkite, ar programa veikia teisingai. Tam reikia paruošti tiek duomenų testinių variantų, kiek yra taškų skaičiavimo variantų: kiekvienam apskritimui, kai taškas yra jo viduje, kai ant apskritimo ir kai jo išorėje. Jeigu programos pateikiami rezultatai sutampa su testų rezultatais, tuomet galime teigti, kad programa skaičiuoja teisingai. Norint įsitikinti, ar gerai skaičiuojamas atstumas, testiniuose duomenų variantuose reikia apskaičiuoti taško atstumus iki apskritimo centro. Tikslinga patikrinti, ar programa gerai skaičiuoja, kai apskritimas ir taškas yra skirtingose stačiakampės koordinačių plokštumos vietose.

Programos skaičiavimų rezultatų ekrane pavyzdys:

```
xc= 60.00 yc = 80.00
r1= 10.00 t1 =    10
r2= 15.00 t2       8
r3= 20.00 t3=      6
r4= 30.00 t4=      2
Tikslumas: 0.01000
Strėlės koordinatė x = 75
Strėlės koordinatė y = 65
Pataikymo taško atstumas iki taikinio centro:  21.21320
Gauti taškai:  2
```

Užduotys.

1. Išnagrinėkite pirmojo programos varianto klausimus ir atlikite pateiktas užduotis.
2. Taškų skaičiavimo sąlygos sakinys sudėtingas. Jeigu taikinyje būtų daugiau apskritimų, jis taptų dar sudėtingesnis). Pagalvokite, kaip galima būtų supaprastinti taškų skaičiavimą.

2.11. Elektros grandinės varžos skaičiavimas

Ši užduotis yra paimta iš 2006 m. informacinių technologijų valstybinio egzamino. Atlikdami šį darbą susipažinsite su ciklas cikle struktūra. Išbandysite skaitymo iš failo ir rašymo į failą veiksmus. Prisiminsite sumavimo algoritmą.

Iš fizikos kurso žinome, kad lygiagrečiai sujungtų laidininkų bendra varža skaičiuojama pagal formulę $\frac{1}{R}=\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}+...+\frac{1}{R_n}$, čia R – lygiagrečiai sujungtų laidininkų varža, R_1, R_2, …, R_n – atskirų laidininkų varžos.
Nuosekliai sujungtų laidininkų bendra varža skaičiuojama pagal formulę $R = R_1 + R_2 + \ldots + R_n$, čia R – nuosekliai sujungtų laidininkų bendra varža, R_1, R_2, …, R_n – atskirų laidininkų varžos.

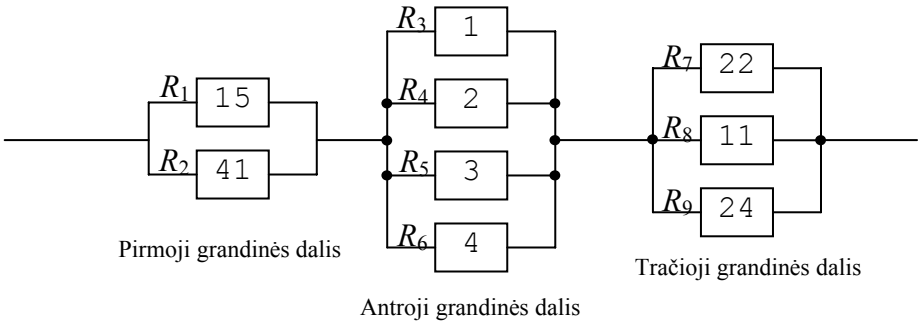
Užduotis. Parašykite programą, kuri apskaičiuotų grandinės bendrą varžą, kai grandinę sudaro viena ar daugiau nuosekliai sujungtų grandinės dalių; kiekviena grandinės dalis sudaryta iš dviejų ar daugiau lygiagrečiai sujungtų žinomos varžos laidininkų.

Programa turi skaityti duomenis iš tekstinio `Duomenys.txt` failo. Pirmoje failo eilutėje įrašytas nuosekliai sujungtų grandinės dalių skaičius (ne daugiau kaip 100). Po to atskirose eilutėse surašyti grandinę sudarančių dalių duomenys: lygiagrečiai sujungtų laidininkų skaičius (ne daugiau kaip 50) ir jų varžų reikšmės.
Rezultatą – apskaičiuotą grandinės bendrą varžą – programa turi įrašyti į `Rezultatai.txt` failą dviejų ženklų po kablelio tikslumu.

Pavyzdys

Duomenų failas	Duomenų paaiškinimas	Rezultatų failas

`3`
`2 15 41`
`4 1 2 3 4`
`3 22 11 24`
	Nuosekliai sujungtų grandinės dalių skaičius:
	pirmoji grandinės dalis (dvi varžų R_1 ir R_2 reikšmės),
	antroji grandinės dalis (keturios varžų R_3, R_4, R_5 ir R_6 reikšmės),
	trečioji grandinės dalis (trys varžų R_7, R_8 ir R_9 reikšmės)
		`17.08`



Skaičiavimas:

$L_1=\frac{1}{R_1}+\frac{1}{R_2}$; $L_2=\frac{1}{R_3}+\frac{1}{R_4}+\frac{1}{R_5}+\frac{1}{R_6}$; $L_3=\frac{1}{R_7}+\frac{1}{R_8}+\frac{1}{R_9}$ – grandinės dalių laidumų skaičiavimas,

$R=\frac{1}{L_1}+\frac{1}{L_2}+\frac{1}{L_3}$ – grandinės bendra varža.

Algoritmas. Kaip pastebite, pirmiausia reikia apskaičiuoti atskirų grandinės dalių varžas, pagal aukščiau duotą formulę ir po to jas susumuoti. Šiose dalyse laidininkai sujungti lygiagrečiai. Pavyzdyje pavaizduotos trys tokios dalys. Yra žinoma kiek kiekvienoje dalyje yra laidininkų. Pradžioje įsivaizduokite, kad grandinę sudaro tik viena dalis, sudaryta iš k lygiagrečiai sujungtų laidininkų. Tegul šių laidininkų varžos surašytos tekstiniame faile `Duomenys.txt` vienoje eilutėje tokia seka: pirmasis skaičius `k` eilutėje nurodo laidininkų skaičių, o toliau surašyta k laidininkų varžos. Duomenų failo `Duomenys.txt` pavyzdys:

```
2 15 41
```

Duomenims iš failo nuskaityti reikia naudoti ciklą `for`. Cikle reikia atlikti atskirų laidininkų varžų sumavimą. Nepamirškite prieš ciklą varžų sumai priskirti reikšmę lygią 0.

Pasiruošimas. Sukurkite katalogą programos failams saugoti, atverkite FPS terpę ir sukurkite programos failą `Darbas11.pas`.

Pirmas žingsnis. Vienos dalies laidininkų varžos skaičiavimas. Tai nesudėtingas ciklinis sumos kaupimo algoritmas. Toliau atlikite sekančius veiksmus.

- Sukurkite tekstinį duomenų failą `Duomenys.txt` ir į jį įrašykite aukščiau pateikto pavyzdžio vieną eilutę.
- Programos pradžioje aprašykite programoje naudojamus kintamuosius.
- Paruoškite duomenų failą įvedimui.
- Sumos kaupimo kintamajam `L` priskirkite reikšmę lygią 0.
- Įveskite duomenis: pirmiausia įveskite laidininkų skaičių k, o po to naudodami ciklą `for` įveskte ir kartu sumuokite laidininkų varžas.
- Uždarykite duomenų failą.
- Atspausdinkite gautą rezultatą.
- Išsaugokite ir įvykdykite programą.

```
program Darbas11;
const CDuom = 'Duomenys.txt';
//-----------------------------------------------------------------------------
var k,                  // Vienos dalies varžų skaičius
    j : integer;        // Ciklo kintamasis
    L,                  // Vienos dalies laidumas
    rj : real;          // Vieno laidininko varža
    F : text;           // Failo kintamasis
begin
  Assign(F, CDuom); Reset(F);
  L := 0;
  Read(F, k);
  for j := 1 to k do
    begin
      Read(F, rj);      // Skaitoma varža
      L := L + 1 / rj;  // Laidumų sumavimas
    end;
    ReadLn(F);          // Vienos dalies varžų sąrašo pabaiga
  Close(F);
  WriteLn('Laidininkų varža: ', 1/L:8:2);
  ReadLn;
end.
```

Paleidę programą, ekrane matysite:

```
Laidininkų varža    10.98
```

Išbandykite šią programą dar du kartus. Pirmąjį kartą į duomenų failą `Duomenys.txt` įrašykite

```
4 1 2 3 4
```

o antrąjį kartą

```
3 22 11 24
```

Pirmuoju atveju rezultatas – laidininkų varža – turėtų būti `0.48`, o antruoju – `5.62`.

Susumavus visus programos bandymus su pradiniais duomenimis (žiūr. schemą) rezultatas turėtų būti toks:

`10.98 + 0.48 + 5.62 = 17.08.`

Kaip pastebėjote šis rezultatas pilnai sutampa su pavyzdžio rezultatu. Tačiau akivaizdu, kad toks darbas su programa neteikia malonumo, nes atliekant skaičiavimus kiekvienai grandinės daliai reikia keisti.duomenų failo turinį ir vėl iš naujo paleidinėti programą. O be to dar gali prisireikti kalkuliatoriaus susumuojant visos grandinės dalių varžas. Reikia ieškoti išeities iš šios situacijos.

Antras žingsnis. Visų nuosekliai sujungtų grandinės dalių bendrai varžai rasti programoje reikalingas dar vienas ciklas `for`, kuris turi apgaubti ankstesnįjį ciklą `for`. Šiam ciklui atlikti, kaip ir ankstesnio ciklo atveju, reikalingi analogiški paruošiamieji veiksmai.

- Pakeiskite duomenų failo `Duomenys.txt` turinį tokiu, koks buvo nurodytas darbo pradžioje
- Papildykite programos kintamųjų aprašus.
- Visos grandinės sumos kaupimo kintamajam `R` priskirkite reikšmę lygią 0.
- Įveskite grandinės dalių skaičių n.
- Užrašykite ciklą `for` kartu su operatoriniais skliaustais `begin..end`, kurie apgaubs ankstenįjį (vidinį) ciklą `for`.
- Vidiniam ciklui priklausančius sakinius pastumkite į dešinę. Tai atlikus programa tampa vaizdesnė ir tuo pačiu suprantamesnė.
- Prieš pasibaigiant šiam ciklui užrašykite sumavimo veiksmą R := R + 1/L, kuris sumuos atskirų grandinės dalių (sujungtų lygiarečiai) varžas, suskaičiuotas vidiniame cikle.
- Pakeiskite rezultato spausdinimo sakinį tokiu

  `WriteLn('Visos grandinės laidininkų varža: ', R:8:2);`
- Išsaugokite ir įvykdykite programą.

```
program Darbas11;
const CDuom = 'Duomenys.txt';
//---------------------------------------------------------------------------
var n,                  // Grandinę sudarančių dalių skaičius
    k,                  // Vienos dalies varžų skaičius
    i, j : integer;     // Ciklų kintamieji
    L,                  // Vienos dalies laidumas
    R,                  // Visos grandinė varža
    rj : real;          // Vieno laidininko varža
    F : text;           // Failo kintamasis
begin
  Assign(F, CDuom); Reset(F);
  R := 0;
```

```
  ReadLn(F, n);
  for i := 1 to n do
    begin
      L := 0;
      Read(F, k);
      for j := 1 to k do
        begin
          Read(F, rj);       // Skaitoma varža
          L := L + 1 / rj;   // Laidumų sumavimas
        end;
      ReadLn(F);             // Vienos dalies varžų sąrašo pabaiga
      R := R + 1 / L;
    end;
  Close(F);
  WriteLn('Visos grandinės laidininkų varža: ', R:8:2);
  ReadLn;
end.
```

Paleidę programą, ekrane matysite:

```
Visos grandinės laidininkų varža    17.08
```

Kaip matote rezultatas sutapo su tuo rezultatu, kuris buvo gautas ankstesnę programą leidžiant tris kartus. Be to šiuo atveju nereikėjo kaitalioti duomenų failo turinio. Na ir galiausiai nereikėjo atlikti atskirų grandinės varžų sumavimo rankomis. Dar kartą pažvelkite į programą. Neįtikėtina, ją sudaro tik 30 eilučių!

Trečias žingsnis. Grįžkime prie užduoties – programa gautą rezultatą (tik skaičių `17.08`) turi įrašyti į rezultatų failą `Rezultatai.txt`.

- Pakeiskite rezultato spausdinimo sakinį

```
WriteLn('Visos grandinės laidininkų varža: ', R:8:2);
```

tokiais sakiniais

```
Assign(F, 'Rezultatai.txt'); Rewrite(F);
WriteLn(F, R:8:2);
Close(F);
```

- Išsaugokite ir įvykdykite programą.

Atvėrę rezultatų failą su komanda `Atverti...` pamatysite skaičių `17.08`. O tai reiškia, kad pagal užduotį parašyta programa atlieka visus skaičiavimus teisingai ir rezultatus pateikia ten kur ir reikia, t. y. rezultatų faile. Aišku, rezultatas nėra vaizdus, trūksta paaiškinamojo teksto, tačiau informacinių technologijų valstybinio egzamino programos testuojamos automatiškai, todėl čia svarbus yra tik pats programos skaičiavimų rezultatas.

Klausimai

1. Kokius paruošiamuosius veiksmus reikia atlikti sumavimo algoritme?
2. Kokio tipo kintamieji naudojami užrašant Paskalio kalba sumavimo algoritmą?
3. Kada galima naudoti apskaičiuotą sumos reikšmę programoje?
4. Paaiškinkite, kuomet reikia naudoti ciklas cikle struktūrą.

Užduotys

1. Tekstiniame faile atskirose eilutėse yra surašyta Jūsų klasės mokinių metiniai įvertinimų pažymiai (viso 10 pažymių). Parašykite programą, kuri apskaičiuotų kiekvieno mokinio pažymių vidurkį ir jį atspausdintų rezultatų faile tokiu pavidalu: mokinio numeris, vidurkis.
2. Tekstiniame faile atskirose eilutėse yra surašyta miesto troleibusų maršrutų atstumai (metrais) tarp visų maršruto stotelių. Eilučių pradžioje užrašytas atitinkamo maršruto stotelių skaičius. Parašykite programą, kuri apskaičiuotų kiekvieno troleibusų maršruto ilgį.

Smalsiems

Pirmas papildomas žingsnis. Programa bus aiškesnė ir suprantamesnė, jeigu ją struktūrizuosime, panaudojant procedūrą.

- Sudarykite procedūrą, kuri apskaičiuotų vienos grandinės dalies laidininkų, sujungtų lygiarečiai varžą, t. y. vidinę ciko cikle dalį apiforminkite procedūra. Procedūrą pavadinkite `RastiRGrandies`. Procedūrai reikia perduoti atidaryto duomenų failo kintamąjį `F`, būtinai nurodant prieš jį raktinį žodelį `var`. Nepamirškite pagrindinėje programoje, likusiame cikle įdėti kreipinį į šią procedūrą.

```
program Darbas11;
const CDuom = 'Duomenys.txt';
//---------------------------------------------------------------------------
// Vienos grandies varža
procedure RastiRGrandies (var F : text; var R : real);
var k,                      // Vienos dalies varžų skaičius
    j : integer;            // Ciklo kintamasis
    rj,                     // Vieno laidininko varža
    L : real;               // Vienos dalies laidumas
begin
  L := 0;
  Read(F, k);
  for j := 1 to k do
    begin
      Read(F, rj);          // Skaitoma varža
      L := L + 1 / rj;      // Laidumų sumavimas
    end;
  ReadLn(F);                // Vienos dalies varžų sąrašo pabaiga
  R := 1 / L;
end;
//---------------------------------------------------------------------------
var n,                    // Grandinę sudarančių dalių skaičius
    i : integer;          // Ciklo kintamasis
    GR,                   // Vienos dalies varža
    R : real;             // Visos grandinė varža
    F : text;             // Failo kintamasis
begin
  Assign(F, CDuom); Reset(F);
  R := 0;
  ReadLn(F, n);
  for i := 1 to n do
    begin
      RastiRGrandies (F, GR);
      R := R + Gr;
    end;
  Close(F);
  WriteLn('Visos grandinės laidininkų varža: ', R:8:2);
```

```
  ReadLn;
end.
```

- Išsaugokite ir įvykdykite programą. Ekrane matysite tokį patį rezultatą, kaip ir ankstesnėje programos versijoje

Antras papildomas žingsnis. Pabandykite aukščiau sukurtą procedūrą pertvarkyti į funkciją. Tam reikia atlikti labai nedidelius pakeitimus aukščiau užrašytoje programoje.

- Kaip procedūrą pertvarkyti į funkciją:
  - procedūros antraštę pakeisti tokia
    ```
    function RGrandies(var F : text) : real;
    ```
  - procedūroje vietoj sakinio `R := 1 / L;` užrašykite sakinį `RGrandies :=  1 / L;`
  - pagrindinėje programoje vietoje cikle `for` esančių operatorinių skliaustų (4 eilutės) užrašykite sakinį
    ```
    R := R + RGrandies(F);
    ```
- Išsaugokite ir įvykdykite programą. Ekrane matysite tokį patį rezultatą, kaip ir ankstesnėje programos versijoje

2.12. Grafikos pradmenys

Atlikdami šį darbą susipažinsite su Paskalio kalbos struktūromis, skirtomis grafiniam ekrano režimui valdyti ir išmoksite kurti paprastus grafinius objektus.

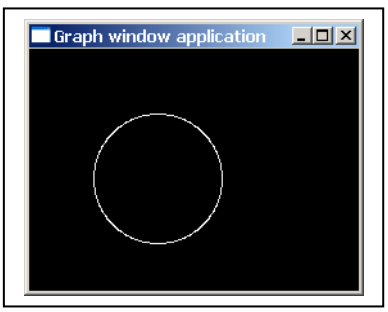
`Free Pascal` terpėje grafiniam ekrano režimui valdyti skirtos struktūros yra saugomos bibliotekoje `Graph`. Prijungus ir aktyvizavus biblioteką `Graph`, sukuriamas grafinis langas, kuriame koordinačių pradžia (0; 0) yra kairysis viršutinis ekrano taškas. x ašis nukreipta į dešinę, y – žemyn. Be to, koordinatės yra tik sveikos ir teigiamos.

Atlikdami šį darbą išbandysite bibliotekų `Graph` ir `Crt` galimybes.

Pirmas žingsnis. Bibliotekos `Graph`, kurioje yra visos reikalingos standartinės procedūros, prijungimas, apskritimo braižymas.

- Po programos antrašte įterpkite sakinį `uses Graph;` šiuo sakiniu prijungiama grafikos biblioteka.
- Biblioteka `Graph` aktyvizuojama pagrindinėje programoje, užrašant sakinį:
  `Initgraph (gd, gm, '');` Čia `gd` ir `gm` – sveikojo tipo kintamieji, kurie aprašomi prieš pagrindinę programą. Kreiptis į bibliotekos standartines procedūras galima iš bet kurios programos vietos ar programoje esančios procedūros.
- Programoje parašykite kreipinį į standartinę procedūrą `Circle`, kad būtų braižomas apskritimas, kurio centras yra taške `(100; 100)`, o spindulys – `50`:

```
program Darbas12;
   uses Graph;
   var gd, gm : integer;
begin
   Initgraph (gd, gm,'');
   Circle (100, 100, 50);
   ReadLn;
end.
```



- Išsaugokite ir įvykdykite sukurtą programą. Jei viską atlikote teisingai, ekrane turėtumėte matyti vaizdą:

Apskritimo braižymui panaudota bibliotekos `Graph` procedūra `Circle (x, y, r)` – braižomas `r` spindulio apskritimas, kurio centras yra taške `(x; y)`. Čia `x, y` ir `r` yra sveikieji teigiami skaičiai.

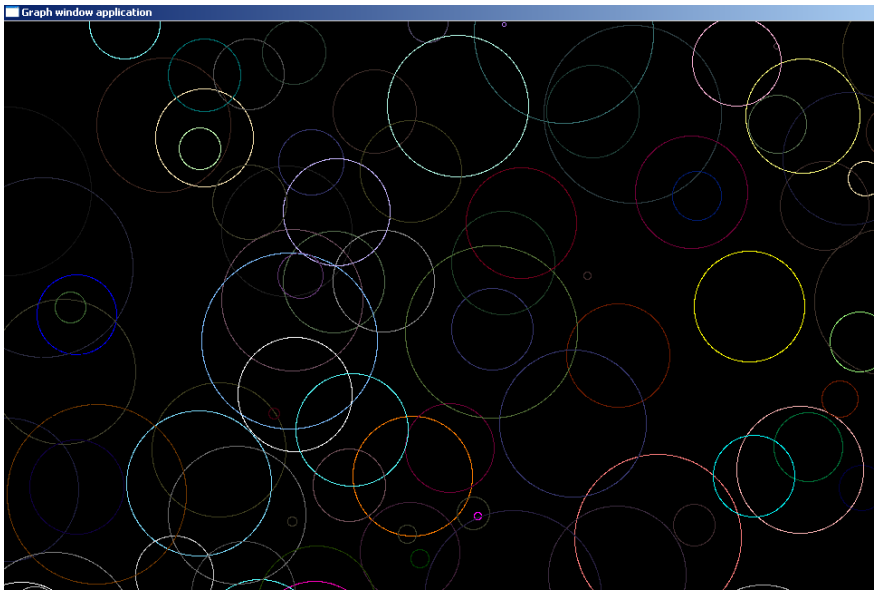
- Grafinis langas uždaromas paspaudus pele lango viršuje dešinėje esantį paskutinį mygtuką  . Tuomet sugrįžtama į tekstinį programos darbo langą.

Antras žingsnis. Daugybės skirtingų spalvų ir spindulių apskritimų braižymas.

- Norėdami grafiniame ekrane matyti daug skirtingų apskritimų, panaudosime standartinę Paskalio funkciją `Random (n)`, kuri generuoja sveikuosius atsitiktinius skaičius, priklausančius intervalui `[0; n-1]`. Prieš panaudojant šią funkciją rekomenduojama iškviesti procedūrą `Randomize`, kuri susieja funkcijos `Random` generuojamos sekos pradžią su kompiuterio laikrodžio parodymu.
- Sukurtą programą papildykite naujais kintamaisiais ir kreipiniais į procedūras ir funkcijas:

```
program Darbas12;
   uses Graph;
   var gd, gm : integer;
      x, y, r : integer;    // Apskritimo centro (x; y) ir spindulio dydžiai
            s : integer;    // Apskritimo spalva
            i : integer;    // Ciklo kintamasis
begin
   Initgraph (gd, gm,'');
   Randomize;               // Kreipinys į procedūrą Randomize, susiejančią funkcijos Random
```

```
                                  // Generuojamą seką su kompiuterio laikrodžio parodymu
   for i := 1 to 100 do           // Ciklas bus vykdomas 100 kartų (bus braižomas 100 apskritimų)
     begin
        x := Random (1000);       // Atsitiktinai generuojama apskritimo centro x koordinatė
        y := Random (800);        // Atsitiktinai generuojama apskritimo centro y koordinatė
        r := Random (100);        // Atsitiktinai generuojamas apskritimo spindulio ilgis
        s := Random (256);        // Atsitiktinai generuojama apskritimo spalva
        SetColor (s);             // Nurodoma apskritimo linijos spalva
        Circle (x, y, r);         // Braižomas r spindulio apskritimas, kurio centras yra taške (x; y)
     end;
   ReadLn;
end.
```



- Išsaugokite ir įvykdykite sukurtą programą. Jei viską atlikote teisingai, ekrane turėtumėte matyti panašų vaizdą:
- Išbandykite, kaip veikia programa su atsitiktinai sugeneruotomis reikšmėmis iš kito intervalo.

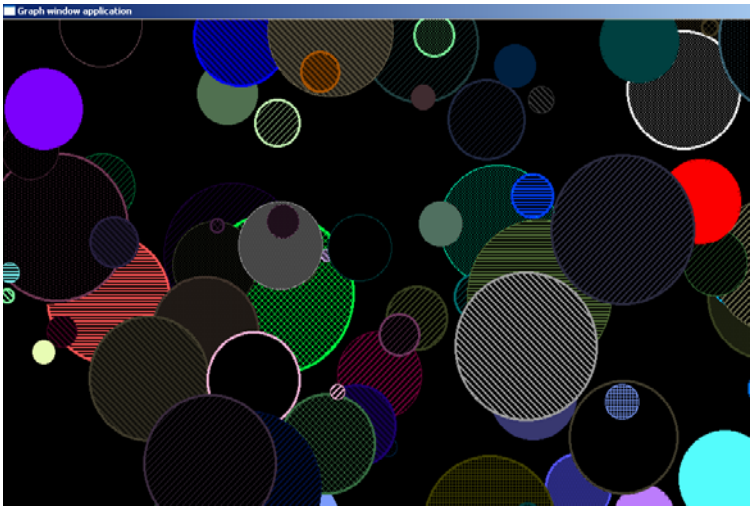
Trečias žingsnis. Apskritimo linijos storio ir užpildo spalvos bei rašto keitimas.

- Sukurtą programą papildykite naujais kintamaisiais ir kreipiniais į procedūras ir funkcijas:

```
program Darbas12;
   uses Graph;
   var gd, gm : integer;
      x, y, r : integer;
        s, sr : integer;      // sr - užpildo rašto spalva
            i : integer;
       storis : integer;      // Kontūro linijos storis
        tipas : integer;      // Kontūro linijos tipas
      rastas : integer;       // Užpildo raštas
begin
   Initgraph (gd, gm,'');
   Randomize;
   for i := 1 to 100 do
     begin
       x := Random (1000);
       y := Random (800);
       r := Random (100);
       s := Random (256);
       sr := Random (256);
       SetColor (s);
       storis := Random (4);        // Atsitiktinai generuojamas kontūro linijos storis
```

```
        tipas := Random (5);        // Atsitiktinai generuojamas kontūro linijos tipas
        SetLineStyle (tipas, tipas, storis);   // Nurodomos linijų charakteristikos
        SetColor (sr);              // Atsitiktinai generuojama rašto spalva
        rastas := Random (13);      // Atsitiktinai generuojamas rašto tipas
        SetFillStyle (rastas, sr);// Nurodomas raštas ir spalva, kurie bus naudojami užpildymui
        FillEllipse (x, y, r, r); // Braižoma užpildyta elipsė, kurios centro koordinatės yra (x; y), o
                                  // abu spinduliai vienodi ir lygūs r (taip gaunamas apskritimas,
                                  // kuris yra atskiras elipsės atvejis).
    end;
  ReadLn;
end.
```

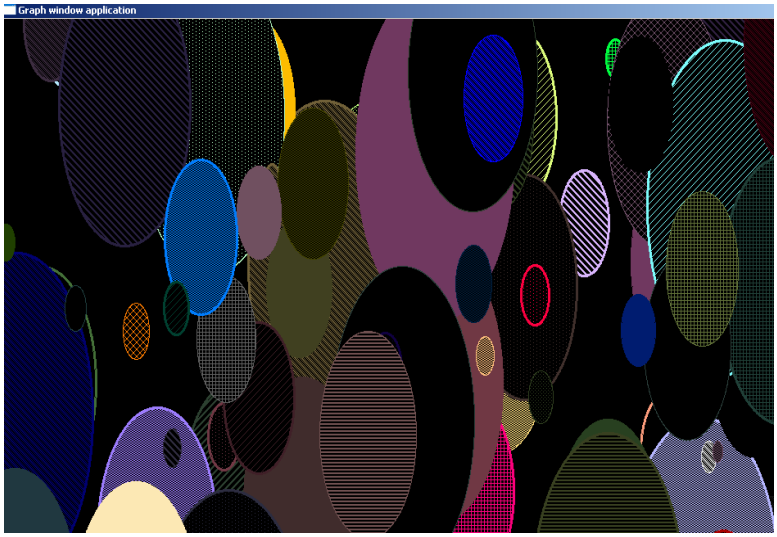
- Išsaugokite ir įvykdykite sukurtą programą. Jei viską atlikote teisingai, ekrane turėtumėte matyti panašų vaizdą:



- Išbandykite, kaip veikia programa parinkus kitas elipsės spindulių reikšmes. Pavyzdžiui, pakeitę kreipinį į procedūrą `FillEllipse` tokiu:

```
FillEllipse (x, y, r, 2 * r);
```

ekrane matysite vaizdą, panašų į pateiktą:



Ketvirtas žingsnis. Judesio programavimas.

- Norėdami suprogramuoti judesį pasinaudosime galimybe „apgauti“ žmogaus akį: rodant keletą ar keliolika paveikslėlių, tam tikru laiko intervalu. Laiko tarpas aprašomas milisekundėmis t. Norimą laiko tarpą aprašo procedūra `Delay (t)`. Ši procedūra yra bibliotekoje `Crt`.
- Trečiame žingsnyje sukurtą programą papildykite prijungdami biblioteką `Crt: uses Graph, Crt;`
- Programą cikle papildykite kreipiniu į procedūrą `Delay`: `Delay (500);`
- Išsaugokite ir įvykdykite sukurtą programą. Kreipinyje į procedūrą `Delay` pakeiskite laiko tarpo reikšmę ir stebėkite situaciją. Pastebėjote, kad tarpas tarp apskritimų pasirodymo keičiasi.

Penktas žingsnis. Garsai.

- Norėdami papildyti sukurtą programą garsais, pasinaudosime bibliotekos `Crt` procedūra `Sound (garso dažnis)`.
- Prieš kreipinį į procedūrą `Delay` įrašykite kreipinį į procedūrą `Sound`: `Sound (500);`
- Išsaugokite ir įvykdykite sukurtą programą. Pastebėjote, kad parodant kiekvieną naują apskritimą, pasigirsta garsas. Kreipinyje į procedūrą `Sound` pakeiskite dažnio reikšmę ir stebėkite situaciją. Ištirkite, kokio dažnio garsai erzina klausą, kokio dažnio garsų jau nebegirdite.

Išsamiai visos bibliotekų `Graph` ir `Crt` procedūros ir funkcijos yra aprašytos `FPS` terpės žinyne, daug pavyzdžių atrasite internete.

Užduotys

1. Pakeiskite antrame žingsnyje sukurtą programą taip, kad grafiniame ekrane būtų rodomi ne apskritimai, o stačiakampiai. Stačiakampis piešiamas naudojant bibliotekos `Graph` procedūrą

   ```
   Rectangle (x1, y1, x2, y2).
   ```

   Čia `x1, y1` – stačiakampio kairiojo viršutinio kampo koordinatės, `x2, y2` – stačiakampio dešiniojo apatinio kampo koordinatės. Stačiakampio koordinatės turi būti generuojamos atsitiktinai.
2. Pakeiskite trečiame žingsnyje sukurtą programą taip, kad grafiniame ekrane būtų rodomi įvairiais raštais užpildyti įvairaus dydžio stačiakampiai. Užpildytas stačiakampis piešiamas naudojant bibliotekos `Graph` procedūrą

   ```
   Bar (x1, y1, x2, y2).
   ```

   Čia `x1, y1` – stačiakampio kairiojo viršutinio kampo koordinatės, `x2, y2` – stačiakampio dešiniojo apatinio kampo koordinatės. Stačiakampio koordinatės turi būti generuojamos atsitiktinai. Programą papildykite kreipiniais į procedūras `Delay` ir `Sound`.

2.13. Reklaminiai užrašai grafiniame ekrane

Atlikdami šį darbą išmoksite panaudoti grafikos galimybes reklaminių užrašų ekrane kūrimui.

Pirmas žingsnis. Bibliotekų `Graph` ir `Crt` prijungimas, ekrano fono užpildymas atsitiktinai generuojamais skirtingų spalvų taškais.

- Sukurkite programą:

```
program Darbas13;
   uses Graph, Crt;
   var gd, gm : integer;
         x, y : integer;    // Taško koordinatėms
            s : integer;    // Taško spalvai
            i : integer;
begin
   Initgraph (gd, gm,'');
   Randomize;
   for i := 1 to 10000 do
     begin
       x := Random (1000);
       y := Random (800);
       s := Random (256);
       PutPixel (x, y, s);
     end;
   ReadLn;
end.
```

Šioje programoje naudojama bibliotekos `Graph` procedūra `PutPixel(x, y, spalva)`, kuri ekrane vaizduoja atsitiktinai sugeneruotos spalvos tašką, kurio koordinatės `(x; y)` yra sveikieji skaičiai, sugeneruoti panaudojant standartinę Paskalio funkciją `Random`.

- Išsaugokite ir įvykdykite sukurtą programą. Jei viską atlikote teisingai, ekrane turėtumėte matyti daug įvairiaspalvių taškų.

Antras žingsnis. Programos papildymas reklaminiu užrašu.

- Norėdami programą papildyti reklaminiu užrašu, panaudosime bibliotekos `Graph` procedūrą
- `OutTextXY (x, y, tekstas)`, čia x ir y – taško, nuo kurio pradedamas rašyti tekstas, koordinatės, `tekstas` – viengubose kabutėse rašomas tekstas. Jei procedūrą užrašysime `OutText (tekstas)`, tuomet tekstas bus rašomas nuo nuo grafinio žymeklio.
- Programoje prieš ciklo sakinį įterpkite kreipinį į procedūrą `OutText`:

```
OutTextXY (20, 20, 'Programuojame Paskaliu');
```

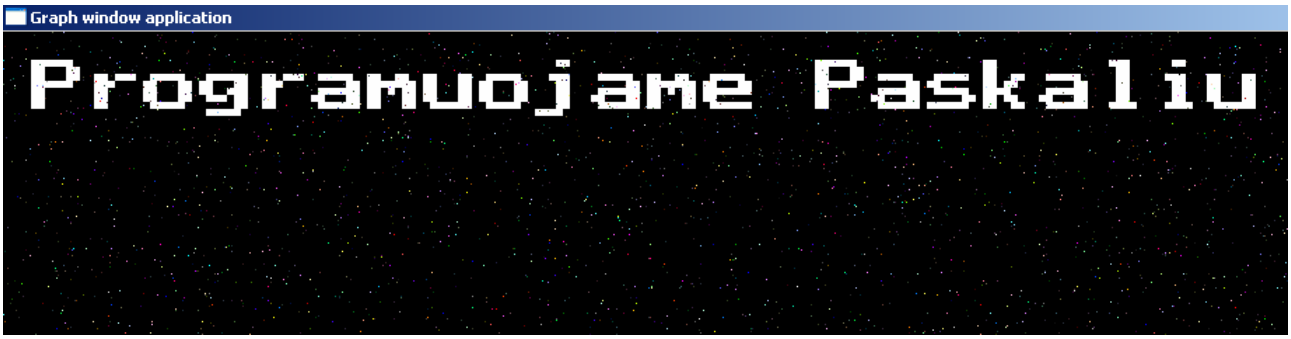
- Išsaugokite ir įvykdykite sukurtą programą. Jei viską atlikote teisingai, ekrane turėtumėte matyti daug įvairiaspalvių taškų ir užrašą „Programuojame Paskaliu“.

Trečias žingsnis. Reklaminio užrašo teksto šrifto, krypties, dydžio keitimas.

- Norėdami programoje nustatyti reklaminio užrašo teksto parametrus, eilute panaudosime bibliotekos `Graph` procedūrą `SetTextStyle (sriftas, kryptis, dydis)`. Parametras `sriftas` leidžia pasirinkti vieną iš 5 šriftų. parametras įgyja reikšmes nuo 0 iki 4. Parametras `kryptis` nurodo rašymo kryptį: jei parametras `kryptis` lygus nuliui, tekstas rašomas horizontaliai, jei 1 – vertikaliai. Parametras `dydis` gali būti skaičius nuo 1 iki 10.
- Programoje prieš reklaminės eilutės rašymo sakinį įterpkite kreipinį į procedūrą `SetTextStyle`:

```
SetTextStyle (4, 0, 5);
```

- Išsaugokite ir įvykdykite sukurtą programą. Jei viską atlikote teisingai, ekrane turėtumėte matyti panašų vaizdą:



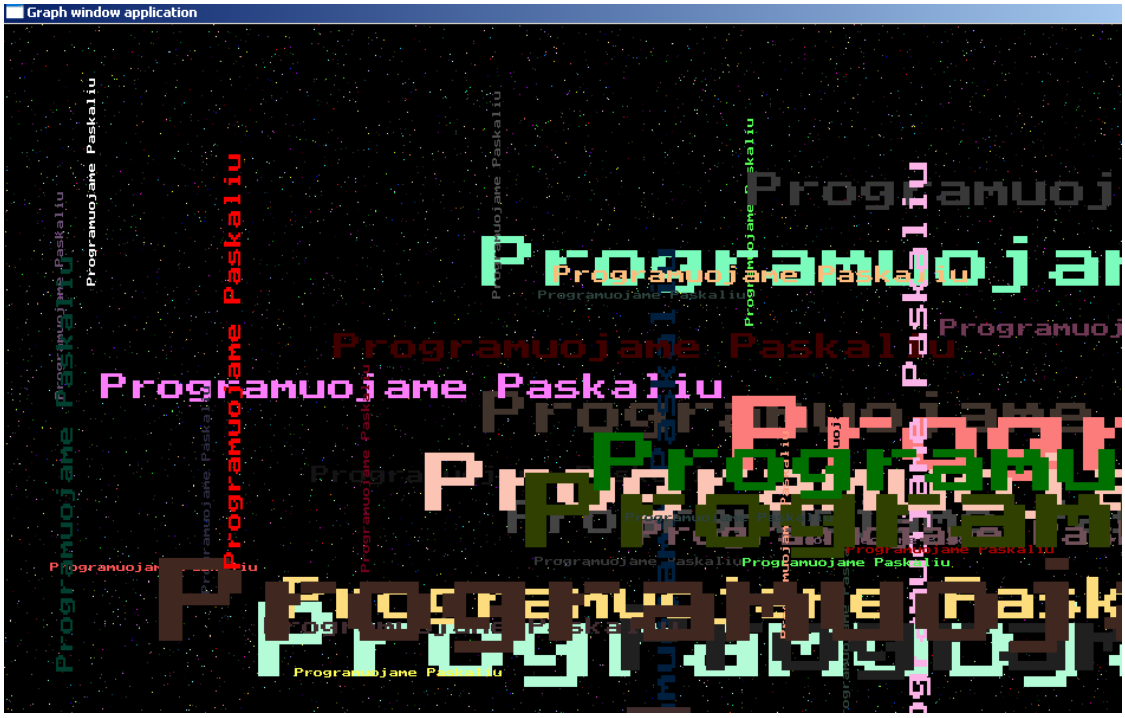
- Kreipinyje į procedūrą `SetTextStyle` pakeiskite parametrų reikšmes ir stebėkite situaciją. Ištirkite, kaip keičiasi reklaminio užrašo tekstas keičiant parametrų reikšmes.

Ketvirtas žingsnis. Reklaminio užrašo teksto spalvos, dydžio, krypties, vietos ekrane keitimas.

- Norėdami pakeisti reklaminio užrašo vietą ekrane vietoj procedūros `OutTextXY` parašykite procedūrą `OutText`, kurią įdėkite į ciklą:

```
program Darbas13;
   uses Graph;
   var gd, gm : integer;
         x, y : integer;    // Taško koordinatėms
            s : integer;    // Taško spalvai
            i : integer;
       xe, ye : integer;    // Teksto koordinatės
    sr, kr, d : integer;    // Teksto formatai
           sp : integer;    // Teksto spalva
begin
   Initgraph (gd, gm,'');
   Randomize;
   for i := 1 to 10000 do
     begin
       x := Random (1000);
       y := Random (800);
       s := Random (256);
       PutPixel (x, y, s);
     end;
   for i := 1 to 100 do
     begin
          xe := Random (1000);
          ye := Random (800);
          sr := Random (5);
          kr := Random (2);
          d := Random (11);
          sp := Random (256);
          SetColor (sp);
          SetTextStyle (sr, kr, d);
          OutTextXY (xe, ye, 'Programuojame Paskaliu');
          Delay (1000);
     end;
   ReadLn;
end.
```

- Išsaugokite ir įvykdykite sukurtą programą. Jei viską atlikote teisingai, ekrane turėtumėte matyti panašų vaizdą. Skirtingais laiko tarpais skirtingose ekrano vietose atsiranda skirtingos spalvos, dydžio ir krypties užrašai.



Užduotis

Sukurkite užrašą ekrane, kuris reklamuotų įdomią knygą. Kurdami užrašą panaudokite įvairias bibliotekų `Graph` ir `Crt` procedūras ir funkcijas, suteikite judesį, papildykite garsais.

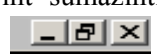
2.14. Judesys.

Paprasčiausio judesio imitacija gaunama tą patį paveikslą perpiešiant kitoje ekrano vietoje. Vieną judesio žingsnį sudaro trys veiksmai: paveikslo piešimas, pauzė ir paveikslo valymas. Nepertraukiamo judesio vaizdas priklauso nuo paveikslo dydžio, piešimo būdo ir pauzės ilgio. Sudėtingesnius paveikslus tikslinga vieną kartą nupiešti ir išsaugoti atmintinėje santykinėmis koordinatėmis. Vėliau toks paveikslas rodomas skirtingose ekrano vietose. Čia demonstruojamas paprasčiausias perpiešimo būdas. Sudėtingesnius būdus ir daugiau informacijos galite rasti literatūroje. Norint jais pasinaudoti, reikia žinoti sudėtingesnes duomenų struktūras ir algoritmus.

Pirmas žingsnis. Grafinio ekrano aktyvizacija.

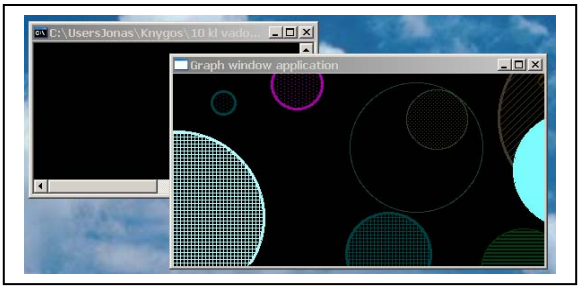
- Po programos antrašte įterpkite sakinį `uses Graph`; šiuo sakiniu prijungiama grafikos biblioteka.
- Parašykite grafinio ekrano aktyvizacijos procedūrą ir ją išbandykite.

```
program Darbas14;
uses Crt, Graph;
//---------------------------------------------------------------------------
// Aktyvizuojamas grafinis ekranas. Grąžina ekrano plotį dx ir aukštį dy
procedure Ekranas( var dx, dy : integer);
var gd, gm : integer;
begin
  gd := D8bit;                      // Grafinė tvarkyklė
  gm := m640x480;                   // Tvarkyklės modifikacija
  InitGraph(gd,gm,'');              // Grafinio lango aktyvizavimas
  if GraphResult <> grOk then       // Aktyvizacijos patikra
    begin
     WriteLn('640x480x256 ekranas nedirba!');
     Halt(1)
    end;
  dx := GetMaxX;       dy := GetMaxY; // Taškų skaičius (ekrano plotis ir aukštis)
end;
//---------------------------------------------------------------------------
var
 dx, dy : integer;
 begin
 Ekranas(dx, dy);          // Grafinio ekrano aktyvizacija
 WriteLn(dx, ' ', dy);     // Tekstiniame ekrane spausdinamas grafinio ekrano plotis ir aukštis
 ReadLn;
 CloseGraph;
end.
```

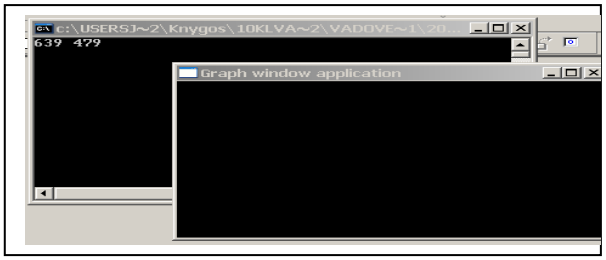
Ankstesniuose darbuose grafinio ekrano dydis buvo parenkamas pagal nutylėjimą ir sutapo su kompiuterio ekranu. Buvo parenkama standartinė grafinio ekrano tvarkyklė. Norint sumažinti ekraną, reikėjo pele paspausti vidurinį ekrano viršuje dešinėje esantį mygtuką: 

Sumažinus langą, po juo rasime tekstinį programos langą. Jį galima aktyvizuoti pele. Taip pat galime padaryti pele perjungus kompiuterio ekrano apačioje paspaudus langus perjungiančius mygtukus:

Sumažinus grafinį ekraną, galite parinkti jo dydį taip, kad būtų matomi abu langai, pavyzdžiui programoje `Darbas12`, galime gauti tokį langų išdėstymą ekrane:



Aktyvizuojant grafinį langą, galima parinkti grafinio lango tvarkyklę ir jos modifikaciją. Grafinio ekrano tvarkyklės priklauso nuo kompiuterio konstrukcinių savybių, todėl skirtinguose kompiuteriuose gali būti skirtingos tvarkyklės. Galimų tvarkyklių sąrašą galima surasti Free Pascal žinyne arba tam skirtoje literatūroje. Šio darbo grafinis ekranas gaunamas iš karto mažesnis.



Antras žingsnis. Skritulių piešimas.

➢ Parašome procedūrą, kuri ekrane atsitiktinai piešia nurodytą skaičių spalvotų skritulių ir ją išbandome. Gauname vaizdą, panašų gautam `Darbas12` programoje.

```
program Darbas14;
uses Crt, Graph;
//-----------------------------------------------------------------------------
// Ekrane piešiami n skritulių. Ekrano plotis dx ir aukštis dy
procedure Skrituliai( n : byte; dx, dy : integer);
 var x, y, r : integer; // Skritulio koordinatės ir spindulys
     i : integer;
 begin
  Randomize;
  for i := 1 to n do  begin
    SetFillStyle( Random(11), Random(15) );          // Parenkama užpildymo raštas
    SetColor(Random(11) ); SetLineStyle( 0, 0, 1 ); // Kontūro spalva ir linija
    x := Random( dx - 30 ) + 20;
    y := Random( dy - 30 ) + 20;
    r := Random( 30 ) + 5;
    FillEllipse( x, y, r,  r  );
   end;
 end;
//=============================================================
var
 dx, dy : integer;
 begin
 Ekranas(dx, dy);           // Grafinio ekrano aktyvizacija
 WriteLn(dx, ' ', dy);      // Tekstiniame ekrane spausinamas grafinio ekrano plotis ir aukštis
 Skrituliai( 25, dx, dy ); // Ekrane piešiami 25 skrituliai
 ReadLn;
 CloseGraph;
end.
```

Trečias žingsnis. Saulutės piešimas.

➢ Nupiešiame saulutę. Tam sukuriama tokia procedūra.

```
program Darbas14;
uses Crt, Graph;
//-----------------------------------------------------------------------------
procedure Skrituliai( n : byte; dx, dy : integer);
//-----------------------------------------------------------------------------
// Piešiama saulutė, kurios koordinatės ekrane (x, y), o spindulys r
procedure Saulute(   x, y, r : integer);
var t, t1, t2, t3, t4, t5 : word;
begin
       // Skaičiuojamos saulutės elementų santykinės reikšmės, priklausomai nuo pateiktų per parametrus dydžių
  t  := r div 4;  t1 := r div 5;  t2 := r div 7;
  t3 := r div 8;  t4 := r div 2;  t5 := Trunc( r / Sqrt( 2));

  SetColor( Black );      SetFillStyle( 1, Yellow );
  FillEllipse( x, y, r, r );            //  Geltonas skritulys

  SetFillStyle( 1, White );             // Akys
  FillEllipse( x-t, y-t4, t2, t1 );    FillEllipse( x+t, y-t4, t2, t1 );
  SetFillStyle( 1, Black );
  FillEllipse( x-t, y-t4+3, t3, t3 );  FillEllipse( x+t, y-t4+3, t3, t3 );

  Arc( x, y+t-2, 180, 360, r-t );       // Šypsena
  Arc( x, y-t,   210, 340, r-2 );
  SetFillStyle( 1, Red );              FloodFill( x, y-t+r, Black );

  SetColor( red );                     // Nosytė
  SetLineStyle( 0, 0, 1 );             Arc( x, y+2, 30, 150, t-2 );
end;
//=============================================================================
var
   dx, dy : integer;
   x, y : integer;
begin
 Ekranas(dx, dy);                  // Grafinio ekrano aktyvizacija
                                   // Tekstiniame ekrane spausinamas grafinio ekrano plotis ir aukštis
 WriteLn(dx, ' ', dy);
 Skrituliai( 25, dx, dy );         // Ekrane piešiami 15 skritulių
 x := dx div 2;   y := dy div 2;   // Saulutė bus piešiama ekrano centre
 Saulute( x, y, r );               // Saulutė ekrano centre
 ReadLn;
 CloseGraph;
end.
```

Išbandykite programą, nupiešdami saulutę ekrano centre. Saulutės spindulys nurodomas konstanta, kuri parašoma programos pradžioje `const r = 40;`

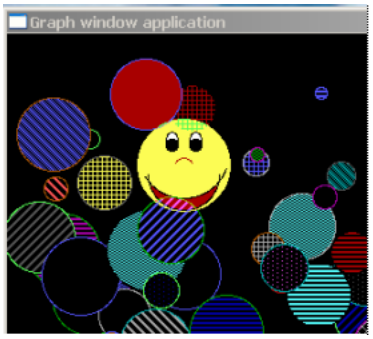
Ketvirtas žingsnis. Saulutės judesys.

Saulutė judės ekrane. Kai pasieks ekrano ribą, bus ekrane nupiešiami papildomi nauji skrituliai ir saulutė pakeis savo judėjimo kryptį. Kryptis parenkama atsitiktinė.

Kadangi saulutę sudaro daug elementų, tai ją kiekvieną kartą naujai perpiešti kitoje vietoje netikslinga. Labai sulėtės judesys. Nupieštą saulutės paveikslą užrašysime atmintinėje. Nereiks kiekvieną kartą kitoje ekrano naujai piešti. Pakaks iš atminties turimą vaizdą padėti kitoje ekrano vietoje. Aišku, ankstesnės vietos saulutės vaizdą reikia valyti.

```
program Darbas14;
uses Crt, Graph;
//---------------------------------------------------------------------------
procedure Skrituliai( n : byte; dx, dy : integer);
//---------------------------------------------------------------------------
procedure Saulute(   x, y, r : integer);
//---------------------------------------------------------------------------
// Generuojama nauja judesio kryptis: koordinačių pokyčio reikšmės x ir y
 procedure Eiti( var x, y : integer );
 var al : integer;
 begin
   repeat   al := Random( 360 );
            x  := Trunc( Cos( al / 180 * Pi )*20);
            y  := Trunc( Sin( al / 180 * Pi )*20);
   until ( x <> 0 ) or ( y <> 0 );
 end;
//---------------------------------------------------------------------
// Saulutės judesys ekrane
procedure Judesys(k, x, y, dx, dy : integer; S : pointer);
var  zx, zy : integer;                    // Saulutės koordinačių pokytis
begin
 PutImage( x-k, y-k, S^, 1 );             // Saulutė perkeliama į ekraną
  repeat
   Eiti( zx, zy );                        // Saulutes judesio zingsniai
   Skrituliai( 25, dx, dy );
    while  ( x < dx-k ) AND ( x > k ) AND // Judesys iki ekrano krašto
           ( y < dy-k ) AND ( y > k ) AND
         NOT KeyPressed   do begin
       PutImage( x-k, y-k, S^, 1 );       // Saulutė perkelima į ekraną
       Delay( 10 );
       PutImage( x-k, y-k, S^, 1 );       // Saulutės valymas senoje vietoje
       x := x + zx;  y := y + zy;         // Naujo vietos koordinatės
     end;
    x := x - zx;   y := y - zy;           // Sugrįžtama į ekraną
 until KeyPressed;
end;
//=====================================================================
var
   dx, dy : integer;
    x,  y : integer;
        k : integer;
        S : pointer;      // Rodyklė į atmintinės vietą, kurioje bus saugomas paveikslas
begin
  Ekranas(dx, dy);         // Grafinio ekrano aktyvizacija
  WriteLn(dx, ' ', dy);    // Tekstiniame ekrane spausdinamas grafinio ekrano plotis ir aukštis
  k := 2*r;                // Saulutės skersmuo
  x := dx div 2;   y := dy div 2; // Ekrano centras
  GetMem( S,  ImageSize( 0, 0, 2*k, 2*k )); // Atminties išskyrimas saulutei saugoti
  Saulute( x, y, r );                       // Piešiama saulutė
  GetImage( x-k, y-k, x+k, y+k, S^ );       // Saulutė užrašoma atmintinėje
  Judesys(k, x, y, dx, dy, S);              // Saulutės judesio ekrane imitacija
```

```
  ReadLn;
  CloseGraph;
end.
```



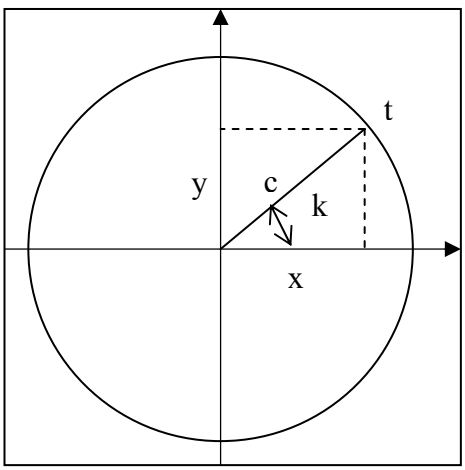
Judesio valdymo ciklas begalinis. Darbas nutraukiamas, kai paspaudžiamas bet koks klavišas klaviatūroje. Tačiau funkcija `KeyPressed` dirba tik tekstiniame ekrane. Taigi, reikia pele pereiti į tekstinį ekraną. Perėjus, darbas grafiniame ekrane nenutrūksta.

2.15. Laikrodis.

Užduotis. Reikia sukurti laikrodį, veikiantį grafiniame režime. Laikrodžio fone išdėstyti valandų skaičius nuo 1 iki 12. Laikrodis privalo turėti tris rodykles: valandoms, minutėms ir sekundėms rodyti. Rodyklės turi judėti.

Sprendžiant šį uždavinį bus reikalingi du dalykai: trigonometrijos žinios ir ekrano grafinis režimas.

Pirmiausia prisiminkime trigonometriją ir ją susiekime su laikrodžiu. Paprasčiausio laikrodžio paviršiuje (dažniausiai apskritimu) yra surašytos valandų reikšmės nuo 1 iki 12. Kaip žinote apskritimas turi 360 laipsnių. Taigi, valandos yra išdėstytos kas 30 laipsnių (360/12), minutės ir sekundės kas 6 laipsniai. Paprasčiausias laikrodis turi tris rodykles valandoms, minutėms ir sekundėms rodyti. Kiekviena rodyklė yra tam tikro ilgio, storio ir spalvos. Šios rodyklės sukasi ratu ir rodo laiką (1 pav.).



1 pav. Grafinis laikrodžio vaizdas

Žinant apskritimo spindulį `c` bei kampą `k` (radianais) taško (laiko) `t` koordinates `x` ir `y` galima apskaičiuoti taikant formules:

`sin(k) = y / c` ir

`cos(k) = x / c`;

Iš šių formulių išreiškiame **`x = c cos(k)`** ir **`y = c sin(k)`**.

Jeigu žinome kampą `k` laipsniais, tai jį galima išreikšti radianais: $\mathbf{k_{rad} = k_{laips}\ (\pi\ /\ 180)}$.

Paskutines tris formules pritaikysime savo užduočiai įvykdyti, t. y. laikrodžio rodyklių pozicijai keisti.

Laiko reikšmes galima gauti panaudojant procedūrą `GetTime`, kuri grąžina kompiuterio sisteminį laiką: valandas, minutes, sekundes ir šimtąsias sekundės dalis. Ši procedūra imama iš bibliotekos Dos.

Kompiuterio ekrane visi vaizduojami elementai sudaromi iš taškų. Tekstiniame ekrano darbo režime visos išvedimo ekraną priemonės dirba fiksuoto dydžio simboliais, kurių forma yra saugoma Free Paskalio kalbos failuose. Šiuo atveju ekrane galima formuoti pranešimus, naudojant tik simbolius, esančius konkrečiame simbolių rinkinyje. Free Paskalio bibliotekoje `Graph` yra priemonių rinkinys darbui su kompiuterio ekranu grafiniame režime. Programuotojas gali pats formuoti vaizdus iš taškų. Norint dirbti grafiniame režime, reikia kompiuterio ekraną paruošti. Tam naudojamos specialios priemonės.

```
procedure InitGraph(var graphDriver : integer;
                    var   graphMode : integer;
                          driverPath : string);
```

Ši procedūra paruošia kompiuterio ekraną darbui grafiniu režimu, t. y. sukuria grafinį langą **Graph window application**.
`graphDriver` kintamasis, kurio reikšmė paprogramei nurodo, kokia bus ekrano grafikos tvarkyklė.
`graphMode` konstanta, kuri nusako ekrano formatą: taškų skaičių, spalvas ir jų kiekį, grafinių puslapių kiekį. Nenurodžius reikšmės, parenkama standartinė pagal nutylėjimą.
`driverPath` kelias į katalogą, kur saugomas grafikos tvarkyklės failas**.**
`grOK` standartinė konstanta, kurios reikšmė žymi, kad grafikos režimas įjungtas teisingai.
`function` `GraphResult : integer` – funkcija, kurios rezultatas yra reikšmė, rodanti, ar sėkmingai buvo įjungtas grafikos režimas. Galimų nesėkmių priežastį rodo funkcijos grąžinama reikšmė.
`procedure` `CloseGraph;` išjungia grafikos režimą. Grįžtama į tekstinį darbo režimą.

```
function GetMaxX : integer;
function GetMaxY : integer;
```

Šios funkcijos grąžina grafinio ekrano taškų kiekį horizontalia ir vertikalia kryptimis. Tai galimos didžiausios `X` ir `Y` koordinačių reikšmės. Atskaitos taškas yra kairysis viršutinis ekrano kampas, kurio koordinatės – (0, 0).
Dabar galima praktiškai palaipsniui pereiti į grafinį ekrano darbo režimą.

Pasiruošimas darbui. Sukurkite katalogą programos failams saugoti, atverkite FPS terpę ir sukurkite programos failą `Darbas15.pas`.

***Pirmas žingsnis*.** Grafinio programos darbo režimo patikrinimas.

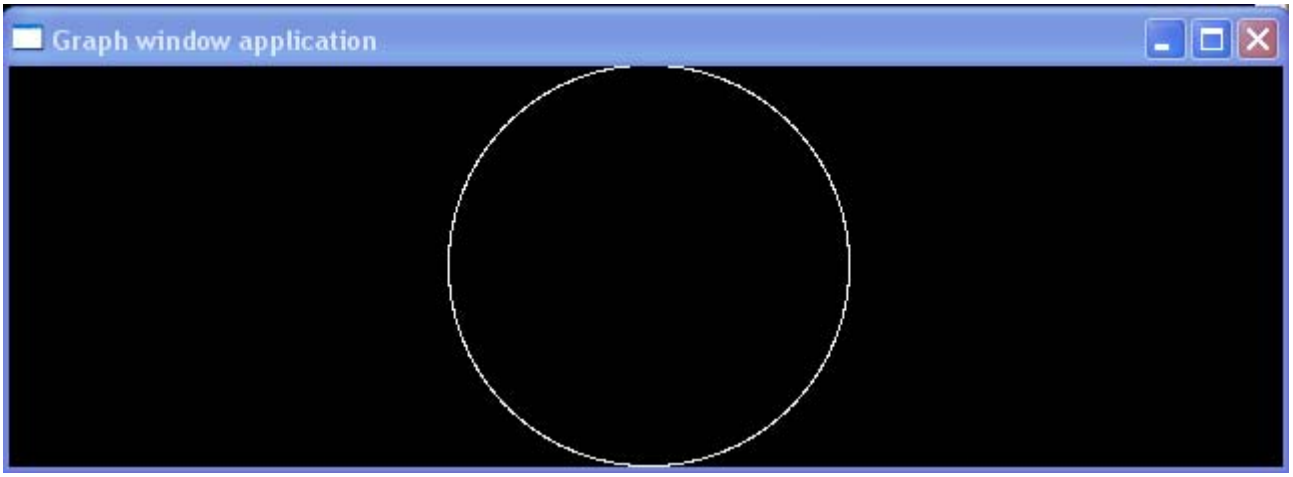
- Užrašykite programos tekstą ir jį įvykdykite.

```
program Darbas15;
uses Crt, Dos, Graph;
var  graphdriver, graphmode, klaida : integer;
//  Programą perveda į grafinį darbo režimą. Jeigu įvyksta klaida, programa stabdoma
procedure GrafinisEkranas;
begin
  graphdriver := vga;
  InitGraph ( graphdriver, graphmode, '' );
  klaida := GraphResult;
  if klaida <> grOk then
    begin
      WriteLn ( 'Klaidos kodas: ', klaida );
      WriteLn ( 'Programa nutraukta' );
      ReadLn;
      Halt (1);
    end
end;
// ------------------------------- Pagrindinė programa ------------------------------------------
var
  mx, my : integer;      //  Grafinio lango išmatavimai
  cx, cy : integer;      //  Grafinio lango centro koordinatės
begin
  GrafinisEkranas;       //  Pereinama į grafinį darbo režimą
  mx := GetmaxX;
  my := GetmaxY;
  cx := mx div 2;
  cy := my div 2;
  Circle(cx, cy, 100);   //  Lango centre piešiamas apskritimas
```

```
  repeat
  until KeyPressed;

  CloseGraph;
  ReadLn;
end.
```

Įvykdę programą, ekrane matysite grafinį langą **Graph window application** su apskritimu lango centre:



Darome išvadą, kad perėjimas į grafinį programos darbo režimą buvo sėkmingas. Darbą galima tęsti toliau.

Antras žingsnis. Laikrodžio valandų – skaičių, nuo 1 iki 12 – surašymas grafiniame ekrane

- Papildykite ankstesnę programą sekančiomis trimis paprogramėmis ir ją įvykdykite.

```
              // Grąžina laikrodžio rodyklės koordinatę x; laikas - laiko reikšmė; ilgis - rodyklės ilgis
function X_Koord( laikas : word; ilgis : integer ) : integer;
              // Grąžina laikrodžio rodyklės koordinatę y; laikas - laiko reikšmė; ilgis - rodyklės ilgis
function Y_Koord( laikas : word; ilgis : integer ) : integer;
              // Nubraižomas laikrodžio fonas, t. y. aplink išdėstoma 12 skaičių (valandų)
              // mx, my - grafinio lango dydis; cx, cy - grafinio lango centro koordinates
procedure Fonas ( mx, my, cx, cy : integer );
```

Pagrindinėje programoje vietoj eilutės

```
  Circle(cx, cy, 100);     // Lango centre piešiamas apskritimas
```

įdėkite kreipinį į procedūrą `Fonas`

```
  Fonas ( mx, my, cx, cy ); // Nubraižomas laikrodžio fonas
```

```
program Darbas15;
uses Crt, Dos, Graph;
var  graphdriver, graphmode, klaida : integer;
// Programą perveda į grafinį darbo režimą. Jeigu įvyksta klaida, programa stabdoma
procedure GrafinisEkranas;
begin
  graphdriver := vga;
  InitGraph ( graphdriver, graphmode, '' );
  klaida := GraphResult;
```

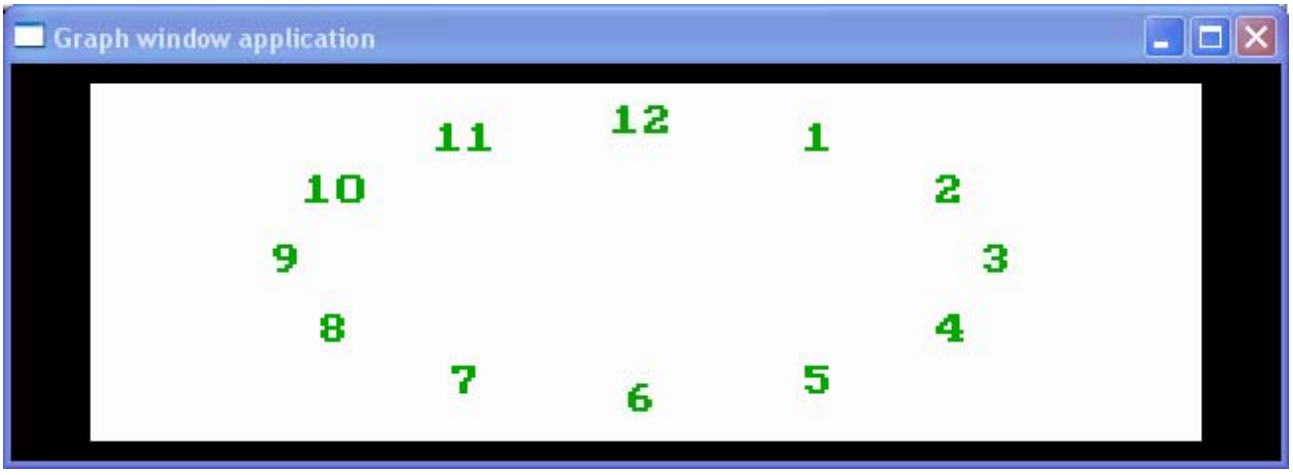
```
  if klaida <> grOk then
    begin
      WriteLn ( 'Klaidos kodas: ', klaida );
      WriteLn ( 'Programa nutraukta.' );
      ReadLn;
      Halt (1);
    end
end;
                // Grąžina laikrodžio rodyklės koordinatę x; laikas - laiko reikšmė; ilgis - rodyklės ilgis
function X_Koord( laikas : word; ilgis : integer ) : integer;
var  k : real;
     x : integer;
begin
  if laikas > 360 then
     laikas := laikas - 360;
  k := laikas * Pi / 180;
  x := - Round ( ilgis * cos( k ) * 639 / 250 );
  X_Koord := x;
end;
               // Grąžina laikrodžio rodyklės koordinatę y; laikas - laiko reikšmė; ilgis - rodyklės ilgis
function Y_Koord( laikas : word; ilgis : integer ) : integer;
var  k : real;
     y : integer;
begin
     if laikas > 360 then
        laikas := laikas - 360;
     k := laikas * Pi / 180;
     y := - Round ( ilgis * sin( k ) );
     Y_Koord := y;
end;
              // Nubraižomas laikrodžio fonas, t. y. aplink išdėstoma 12 skaičių (valandų)
              // mx, my - grafinio lango dydis; cx, cy - grafinio lango centro koordinates
procedure Fonas ( mx, my, cx, cy : integer );
var x, y : integer;
    i  : integer;
    sk : string;
begin
  Bar ( 40, 10, mx - 40, my - 10 );
  for i := 1 to 12 do
    begin
      Str ( i, sk );
      SetTextStyle ( 0, 0, 2 );
      x := X_Koord ( i * 30 + 90, 70 );
      y := Y_Koord ( i * 30 + 90, 70 );
      MoveTo ( cx + x, cy + y );
      SetTextJustify ( 1, 1 );
      SetColor ( Green );
      OutText ( sk );
    end
end;
// ------------------------------- Pagrindinė programa ----------------------------------------
var
  mx, my : integer;        // Grafinio lango išmatavimai
  cx, cy : integer;        // Grafinio lango centro koordinatės
```

```
begin
  GrafinisEkranas;          //  Pereinama į grafinį darbo režimą
  mx := GetmaxX;
  my := GetmaxY;
  cx := mx div 2;
  cy := my div 2;
//  Circle(cx, cy, 100);   //  Lango centre piešiamas apskritimas
  Fonas ( mx, my, cx, cy ); //  Nubraižomas laikrodžio fonas

  repeat
  until KeyPressed;

  CloseGraph;
  ReadLn;
end.
```

Įvykdę programą ekrane matysite grafinį langą **Graph window application** su laikrodžio fonu:



Trečias žingsnis. Trys laikrodžio rodyklės ir jų judėjimas ratu.

- Papildykite ankstesnę programą dar viena procedūra

```
//  Brėžia liniją nukreiptą iš centro (taskas (cx, cy)) į atitinkamą tašką, kuri nusako laikas ir ilgis
//  spalva - linijos spalva; storis - linijos storis
procedure Rodykle( cx, cy : integer; spalva : integer;
                   laikas : word; ilgis, storis : integer );
```

- Pagrindinėje programoje ciklą `repeat until` papildykite reikalingais skaičiavimais, t. y. sisteminio kompiuterio laiko skaitymo procedūra `GetTime` bei trimis kreipiniais į procedūrą `Rodykle`
- Išbandykite programą.

```
program Darbas15;
uses Crt, Dos, Graph;
var  graphdriver, graphmode, klaida : integer;
//  Programą perveda į grafinį darbo režimą. Jeigu įvyksta klaida, programa stabdoma
procedure GrafinisEkranas;
begin
  graphdriver := vga;
  InitGraph ( graphdriver, graphmode, '' );
```

```
  klaida := GraphResult;
  if klaida <> grOk then
    begin
      WriteLn ( 'Klaidos kodas: ', klaida );
      WriteLn ( 'Programa nutraukta' );
      ReadLn;
      Halt (1);
    end
end;
```

// Grąžina laikrodžio rodyklės koordinatę x; laikas - laiko reikšmė; ilgis - rodyklės ilgis

```
function X_Koord( laikas : word; ilgis : integer ) : integer;
var  k : real;
     x : integer;
begin
  if laikas > 360 then
     laikas := laikas - 360;
  k := laikas * Pi / 180;
  x := - Round ( ilgis * cos( k ) * 639 / 250 );
  X_Koord := x;
end;
```

// Grąžina laikrodžio rodyklės koordinatę y; laikas - laiko reikšmė; ilgis - rodyklės ilgis

```
function Y_Koord( laikas : word; ilgis : integer ) : integer;
var  k : real;
     y : integer;
begin
     if laikas > 360 then
        laikas := laikas - 360;
     k := laikas * Pi / 180;
     y := - Round ( ilgis * sin( k ) );
     Y_Koord := y;
end;
```

// Nubraižomas laikrodžio fonas, t. y. aplink išdėstoma 12 skaičių (valandų)
// mx, my - grafinio lango dydis; cx, cy - grafinio lango centro koordinates

```
procedure Fonas ( mx, my, cx, cy : integer );
var x, y : integer;
    i  : integer;
    sk : string;
begin
  Bar ( 40, 10, mx - 40, my - 10 );
  for i := 1 to 12 do
    begin
      Str ( i, sk );
      SetTextStyle ( 0, 0, 2 );
      x := X_Koord ( i * 30 + 90, 70 );
      y := Y_Koord ( i * 30 + 90, 70 );
      MoveTo ( cx + x, cy + y );
      SetTextJustify ( 1, 1 );
      SetColor ( Green );
      OutText ( sk );
    end
end;
```

// Brėžia liniją nukreiptą iš centro (taskas (cx, cy)) į atitinkamą tašką, kuri nusako laikas ir ilgis
// spalva - linijos spalva; storis - linijos storis

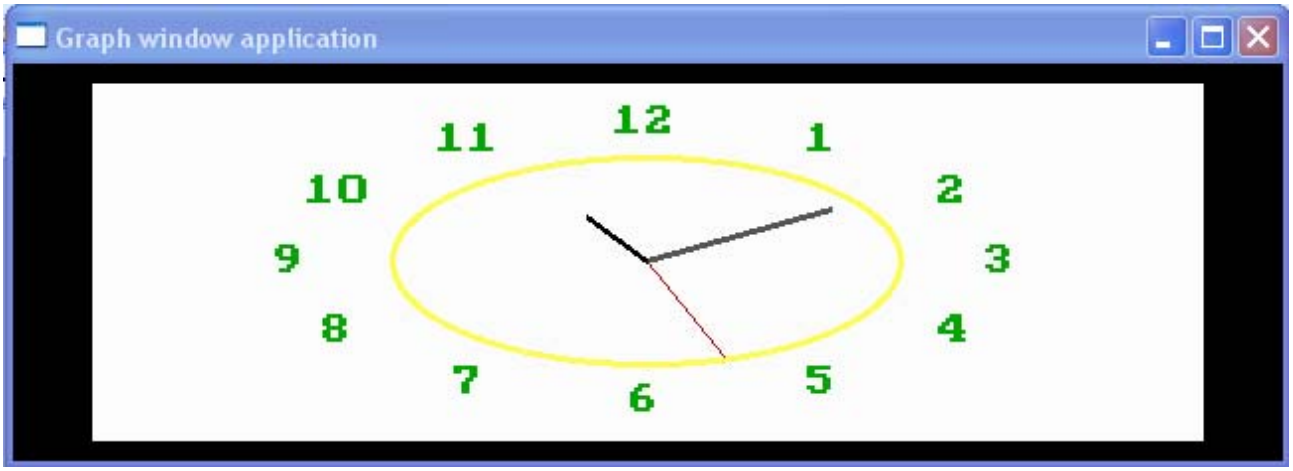
```
procedure Rodykle( cx, cy : integer; spalva : integer;
```

```
                  laikas : word; ilgis, storis : integer );
begin
  MoveTo ( cx, cy );
  SetLineStyle ( 0, 0, storis );
  SetColor ( spalva );
  LineRel ( X_Koord ( laikas, ilgis ), Y_Koord ( laikas, ilgis ) )
end;
// ------------------------------- Pagrindinė programa ----------------------------------------
var
  mx, my : integer;        // Grafinio lango išmatavimai
  cx, cy : integer;        // Grafinio lango centro koordinatės
  val, min, sek, sek1, ss : word;
begin
  GrafinisEkranas;         // Pereinama į grafinį darbo režimą
  mx := GetmaxX;
  my := GetmaxY;
  cx := mx div 2;
  cy := my div 2;
//  Circle(cx, cy, 100);  // Lango centre piešiamas apskritimas
  Fonas ( mx, my, cx, cy ); // Nubraižomas laikrodžio fonas

  sek  := 0;
  sek1 := 0;
  repeat
    while sek = sek1 do
      GetTime ( val, min, sek1, ss );
    sek := sek1;
    SetColor ( Yellow );
    FillEllipse ( cx, cy, Round ( 50 * 639 / 250 ), 52 );
    Rodykle ( cx, cy, red, sek * 6 + 90, 50, 1 );
    Rodykle ( cx, cy, 8, min * 6 + 90, 45, 3 );
    Rodykle ( cx, cy, Black, val * 30 + 90 + Round((min * 6 + 90) / 60), 25, 5);
    Sound ( 100 );
    Delay ( 50 );
    NoSound;
  until KeyPressed;

  CloseGraph;
  ReadLn;
end.
```

Įvykdę programą ekrane matysite grafinį langą **Graph window application** su laikrodžio fonu ir rodyklėmis:



Ketvirtas žingsnis. Papildomi programos pertvarkymai.

- Pakeiskite laikrodžio fono ir skaičių spalvas kitomis.
- Pakeiskite laikrodžio skleidžiamo garso dažnį ir jo ilgumą.
- Laikrodžio fone padarykite sekundžių ir minučių atžymas.

3. Pagrindinės Paskalio kalbos konstrukcijos

Šveicarų mokslininkas Niklausas Virtas (*Niklaus Emil Wirth*), sukūręs programavimo kalbą Pascal
Gimė 1934 m.

Vadovavo kuriant programavimo kalbas: Euler, Algo W, Pascal, Modula, Modula-2, Oberon. Sukūrė paprastą programavimo kalbą PL/0, skirtą kompiuterio projektavimui iliustruoti.
Daug dėmesio skyrė programavimo mokymo metodikai. Jo 1975 m. parašyta knyga "Algoritmai + Duomenų struktūros = Programos" buvo plačiai pripažinta visame pasaulyje ir labai vertinga dar šiuo metu.
Paskalio programavimo kalba buvo sukurta išskirtinai programavimo mokymui, tačiau ji ir po šiai dienai sėkmingai naudojama net programuotojų profesonalų. Modula serijos programavimo kalbos buvo sukurtos Paskalio kalbos pagrindu ir skirtos mokymui, tačiau jos nepasižymėjo populiarumu.

3.1. Kintamasis, kintamojo reikšmė

Kintamieji skirti duomenims saugoti. Jų vardai sudaromi naudojant raides ir skaitmenis, tačiau pirmu simboliu turi būti raidė. Kintamieji aprašomi aprašymų srityje, žymint žodžiu `var` (angliško žodžio `variable` santrumpa). Kiekvienam kintamajam reikia nurodyti, kokio tipo duomenis jis saugos. Kintamojo aprašymo struktūra: `kintamojoVardas : duomenųTipas;` Jeigu yra keletas to paties tipo kintamųjų, tuomet galima juos išvardyti viename sakinyje, skiriant vardus kableliu. Pagrindiniai duomenų tipai nurodomi tokiais žodžiais:

- `real` – realiesiems skaičiamas,
- `integer` – sveikiesiems skaičiams,
- `boolean` – lognėms reikšmėms,
- `char` – simbliams.

Pavyzdžiui:

```
var a, b, c, y: real;      // Realaus tipo kintamieji
    i, k, a2  : integer;   // Sveikojo tipo kintamieji
    simb      : char;      // Simbolinio tipo kintamasis
```

Kompiliatorius kiekvienam kintamajam kompiuterio atmintinėje skiria tiek vietos, kiek reikia nurodyto tipo duomenis saugoti. Kintamieji reikšmes (duomenis) gauna duomenų įvedimo (skaitymo) sakiniais arba priskyrimo sakiniais. Kintamieji negali saugoti kito duomenų tipo reikšmių.

3.2. Priskyrimo sakinys

Duomenų apdorojimo veiksmai aprašomi priskyrimo sakiniais, kurių struktūra:

```
kintamojoVardas := Reiškinys;
```

Čia simbolis `:=` žymi priskyrimo operatorių, o `Reiškinys` aprašo, kokius veiksmus, kokia tvarka ir su kokiais argumentais reikia atlikti. Kairiojoje operatoriaus `:=` pusėje įrašytas kintamojo vardas nurodo, kam suteikiama apskaičiuota reiškinio reikšmė. Dažniausiai vartojami aritmetiniai reiškiniai, aprašantys aritmetinius skaičiavimus ir atitinkantys mums įprastų algebrinių reiškinių sąvoką.
Pavyzdžiui, priskyrimo sakinys `y := x * x;` reiškia, kad argumento x reikšmė keliama kvadratu ir rezultatas priskiriamas kintamajam `y`. Priskyrimo sakinio rašymo forma artima lygybės išraiškai, tačiau tarp šių struktūrų yra esminis skirtumas. Lygybėmis nurodoma egzistuojanti priklausomybė, patvirtinamas tam tikras faktas. Tuo tarpu priskyrimo sakiniai aprašo procesus, kuriems būdinga reikšmių kaita laikui bėgant, todėl taisyklingas ir toks sakinys:

```
a := a + 1;
```

Jis nurodo, kad senoji kintamojo `a` reikšmė turi būti padidinta vienetu, o rezultatas vėl pavadintas `a`. Šį veiksmą galima aprašyti lygtimi:

$a_{(nauja)} = a_{(sena)} + 1.$

Priskyrimo sakiniuose, dešinėje priskyrimo operatoriaus pusėje, gali būti nurodyti įvairių tipų reiškiniai. Todėl būtina sekti, kad priskyrimo operatoriaus kairiojoje pusėje nurodyto kintamojo tipas atitiktų reiškinio, esančio dešiniojoje pusėje, reikšmės tipą.
Reiškiniuose galima naudoti tik tokius kintamuosius, kurių reikšmės apibrėžtos anksčiau įrašytuose programos sakiniuose.
Skiriami trys reiškinių tipai:

- Aritmetinis reiškinys, kuriame yra kintamieji ir konstantos jungiami aritmetinių operacijų ženklais ir kurių skaičiavimo rezultatas yra skaičius; reiškinius užrašant naudojami paprastieji skliaustai, gali būti kreipiniai į funkcijas, kurių darbo rezultatas yra skaičius (pvz, `Sin` funkcija).
- Santykio reiškinys. Tai du aritmetiniai reiškiniai, tarp kurių rašomas santykio operacijos ženklas (>, <, <=, >=, <>, =). Tokio reiškinio rezultatas yra loginė reikšmė `TRUE` (tiesa) arba `FALSE` (netiesa).
- Loginis reiškinys. Tai reiškinys, kuriame santykio reiškiniai jungiami loginių operacijų ženklais. Paskalyje jie žymimi žodžiais `AND` (ir), `OR` (arba), `NOT` (ne). Reiškinio rezultatas yra loginė reikšmė `TRUE` arba `FALSE`.

Santykio ir loginiai reiškiniai dažniausiai naudojami ciklo ir sąlygos sakiniuose. Kadangi santykio reiškiniai yra loginių reiškinių dalis, todėl apbendrintai jie vadinami loginiais reiškiniais.

3.3. Duomenų įvedimas klaviatūra

Duomenys kompiuteriuose gali būti įvedami iš įvairių įrenginių ir jų valdymas yra gana sudėtingas dalykas. Čia reikia nurodyti ne tik perduodamus duomenis, bet ir įvairias įrenginių parengimo bei duomenų tvarkymo operacijas. Asmeniniuose kompiuteriuose standartinis įvesties įrenginys yra klaviatūra.
Duomenų įvedamo klaviatūra procedūros:

```
Read (kintamųjųSąrašas);
ReadLn(kintamųjųSąrašas);
```

Duomenų įvedimo procesas programos darbo metu vykdomas dviem etapais: duomenų surinkimu klaviatūra ir duomenų perdavimu kintamiesiems. Iš pradžių įvedimo procedūra stabdo kompiuterį ir laukia, kol klaviatūra renkami duomenys. Jie kaupiami specialiame buferinės atmintinės įtaise ir vaizduojami ekrane. Tuo metu jie gali būti taisomi vaizdinėmis redagavimo priemonėmis. Nuspaudus įvedimo klavišą, kuris žymimas simboliu ↵ arba žodžiu **`Enter`**, buferyje saugomi duomenys paskirstomi procedūros `Read` arba `ReadLn` argumentų sąraše išvardintiems kintamiesiems, kurie nurodo, kiek ir kokio tipo duomenų turi būti įvesta, bei kaip jie turi būti paskirstyti programos kintamiesiems.
Procedūros `Read` ir `ReadLn` skiriasi duomenų atrinkimo iš pagalbinės atmintinės (buferio) būdais. Procedūra `Read` visuomet ima tiek duomenų, kiek jos sąraše yra parametrų, o procedūra `ReadLn` ima visus buferyje esančius duomenis, kurie įvesti prieš paspaudžiant klavišą **`Enter`**. Tokią pastarosios procedūros savybę nurodo ir jos vardas `ReadLn`, kuris sudarytas iš dviejų anglų kalbos žodžių: `Read` (skaityti) ir `Line` (eilutę).
Pavyzdžiui, jei atliekama komanda `Read(a, b);`, tačiau įvesime tik vieną skaičių `25`↵, kintamajam `a` bus suteikta reikšmė `25` ir kompiuteris lauks kol įvesime `b` reikšmę. Jei, vykdant `Read(a, b);`, įvesime tris reikšmes: `25 -5 10`↵, kintamiesiems `a` ir `b` bus paskirstytos pirmosios dvi reikšmės, o trečioji liks buferyje ir lauks naujo skaitymo veiksmo.
Jeigu tokia pati situacija susidarytų vykdant procedūrą `ReadLn`, trečioji reikšmė `10` būtų prarasta, nes ši procedūra atmeta perteklinius duomenis, kurie lieka paskirsčius perskaitytos eilutės duomenis.
Duomenų grupes įvedant vienoje eilutėje, reikšmės atskiriamos tarpo arba tabuliacijos klavišais.
Procedūrą `ReadLn` galima naudoti ir be parametrų. Programa laukia, kol, paspaudus klavišą **`Enter`**, bus įvykdyta ši procedūra. Tai patogu derinant programas, išvedant į ekraną didelius duomenų kiekius ir kitais atvejais.

Aprašant duomenų įvedimą, prieš kreipiantis į procedūrą `ReadLn` (arba `Read`) patariama rašyti išvedimo sakinį, kuris paaiškintų, kokių duomenų programai reikia. Tokia išvedimo ir įvedimo sakinių pora dažnai dar vadinama duomenų užklausa (arba dialogu). Pavyzdžiui, norint įvesti mokinio svorį (kintamasis `sv`) reikėtų programoje rašyti tokią užklausą:

```
Write('Įveskite mokinio svorį: ');     ReadLn(sv);
```

arba

```
WriteLn('Įveskite mokinio svorį');     ReadLn(sv);
```

Ji nurodo, kad kompiuteriui ekrane reikia suformuoti pranešimą **Įveskite mokinio svorį** ir laukti, kol klaviatūra bus surinktas skaičius (svoris).

3.4. Rezultatų (duomenų) išvedimas ekrane

Standartinis asmeninių kompiuterių išvesties įrenginys, į kurį duomenys nukreipiami procedūromis `Write` ir `WriteLn`, yra ekranas. Kreipinių į šias procedūras sintaksė:

```
Write(duomenųSąrašas);
WriteLn(duomenųSąrašas);
```

Išvedamų duomenų sąrašuose galima nurodyti kintamuosius, konstantas ir reiškinius. Be to, gali būti aprašyti ir atskiriems duomenims skirti laukai. Duomenų sąrašo elemento struktūra:

```
Duomuo[:s1[:s2]]
```

Čia laužtinai skliaustai rodo, kad tos dalies gali nebūti. Sveikojo skaičiaus tipo konstanta `s1` elemento apraše nurodo jam skiriamo lauko dydį, o konstanta `s2` rašoma tik realiesiems skaičiams, turintiems sveikąją ir trupmeninę dalis. Tai trupmeninei daliai skirtų simbolių skaičius. Pavyzdžiai:

```
Write(a);
WriteLn(a:5, ' ', b:6:2);
```

Jei išvedant realiuosius skaičius trupmeninei daliai skiriamo lauko dydis nenurodomas, skaičiai vaizduojami, standartine išraiška, kurioje vartojamas daugiklis 10^n. Laipsnio pagrindas žymimas simboliu `E` ir po jo rašomas sveikuoju skaičiumi išreikštas laipsnio rodiklis n. Pavyzdžiui, `1.254E-05` atitinka $1.254 \cdot 10^{-5}$.
Ekrane išvedami duomenys pradedami rašyti nuo žymeklio pozicijos, vadinamos aktyviąja ekrano pozicija. Jei atliekama procedūra `Write`, ekrane surašius duomenis, aktyvioji ekrano pozicija lieka ties paskutiniojo lauko riba. Procedūra `WriteLn` aktyviąją ekrano poziciją visuomet perkelia į naujos eilutės pradžią. Šią savybę galima panaudoti aprašant tuščias ekrano eilutes:

```
WriteLn;
```

Procedūrų `Write` ir `WriteLn` argumentai gali būti ne tik skaičiai ir kintamieji, bet ir bet kokie tekstiniai duomenys, kuriuos pageidaujama matyti ekrane. Tokius duomenis argumentų sąraše būtina rašyti tarp kabučių (apostrofų). Pavyzdžiui, sakinys

```
WriteLn('Sveiki! Jūsų programa pradeda darbą!');
```

išveda ekrane jo argumento nurodomą tekstą:

Sveiki! Jūsų programa pradeda darbą!

Aprašant programos darbo rezultatų išvedimą, labai patogu procedūros `WriteLn` duomenų sąraše nurodyti tiek duomenų reikšmes, tiek juos paaiškinančias simbolių eilutes. Pavyzdžiui, sakinys

```
WriteLn ('Stačiakampio plotas: ', plotas:10:2);
```

nurodo, kad prieš kintamojo `plotas` reikšmę bus įrašomas paaiškinimas **Stačiakampio plotas:**

3.5. Ciklas `while`

Ciklas yra naudojamas pasikartojantiems skaičiavimams atlikti. Ciklo `while` antraštė valdo tik vieno sakinio kartojimą. Jeigu reikia kartoti kelis sakinius, tai jie jungiami į sudėtinį sakinį ir rašomi tarp `begin` ir `end`:

while *Sąlyga* **do** *KartojamasSakinys*;	**while** *Sąlyga* **do** **begin** *KartojamiSakiniai;* **end;**

Sąlyga – tai bet koks loginis reiškinys, *KartojamasSakinys* – bet koks sakinys (taip pat ir ciklo).
Ciklo `while` vykdymo tvarka: Patikrinama *Sąlyga*. Jei jos reikšmė yra `TRUE` (tiesa), tuomet atliekamas *KartojamasSakinys*, priešingu atveju jis praleidžiamas ir atliekami toliau už ciklo esantys sakiniai.
Organizuojant ciklą reikia atkreipti dėmesį į:

- Ciklas bus vykdomas be galo daug kartų (toks ciklas vadinamas „amžinuoju“ ciklu), jeigu nepasirūpinsite, kad atliekant ciklą (*KartojamasSakinys* viduje) *Sąlyga* kada nors įgautų reikšmę `FALSE`.
- Jeigu ciklo viduje reikia įvykdyti ne vieną sakinį, o sakinių grupę, ši grupė turi būti rašoma tarp `begin` ir `end`.
- Jei prieš pradedant ciklą *Sąlyga* yra `FALSE`, *KartojamasSakinys* ar sakinių grupė nevykdomas nė karto.
- Kintamiesiems, sudarantiems *Sąlyga* loginę išraišką, prieš `while` sakinį esančioje programos dalyje turi būti suteiktos pradinės reikšmės, o *KartojamasSakinys* viduje šių kintamųjų reikšmės turi būti keičiamos taip, kad kada nors *Sąlyga* taptų `FALSE`.

3.6. Ciklas `for`

Ciklo `for` antraštė valdo tik vieno sakinio kartojimą. Jeigu reikia kartoti kelis sakinius, tai jie jungiami į sudėtinį sakinį, kuris apjungiamas `begin end` skliaustais. Ciklo parametro reikšmė turi būti diskretinio tipo. Paprasčiausiu atveju – sveikasis skaičius.

Kitimo žingsnis: +1	Kitimo žingsnis: –1
for cp := R1 **to** R2 **do** *KartojamasSakinys;*	**for** cp := R1 **downto** R2 **do** *KartojamasSakinys;*
for cp := R1 **to** R2 **do** **begin** K*artojamiSakiniai;* **end;**	**for** cp := R1 **downto** R2 **do** **begin** *KartojamiSakiniai;* **end;**

Vykdymo tvarka. Formaliai `for` ciklo veiksmus galima aprašyti taip:

1. Ciklo parametro `cp` reikšmei priskiriama reiškinio `R1` reikšmė. `R1` reikšmė apskaičiuojama pradedant ciklą ir naujai neperskaičiuojama.
2. Skaičiuojama reiškinio `R2` reikšmė. Tai bus ciklo parametro paskutinė reikšmė, kuriai esant ciklas dar bus vykdomas. Ji apskaičiuojama pradedant ciklą ir neperskaičiuojama.
3. Jeigu ciklo parametro `cp` reikšmė ne didesnė už apskaičiuotą paskutinę reikšmę, tuomet atliekami ciklo sakiniai. Priešingu atveju ciklas nutraukiamas.
4. Atlikus ciklo sakinius, `cp` reikšmė didinama vienetu ir veiksmai kartojami nuo trečiojo žingsnio.

Jeigu cikle yra `downto` nuoroda, tuomet paskutiniai du žingsniai tokie:

3. Jeigu ciklo parametro `cp` reikšmė ne mažesnė už apskaičiuotą paskutinę reikšmę, tuomet atliekami ciklo sakiniai. Priešingu atveju ciklas nutraukiamas.
4. Atlikus ciklo sakinius, `cp` reikšmė mažinama vienetu ir veiksmai kartojami nuo trečiojo žingsnio.

3.7. Sąlygos sakinys if

Sąlygos sakiniu keičiama nuosekli sakinių atlikimo tvarka. Atliekami veiksmai priklauso nuo *Sąlygos*: jei *Sąlyga* tenkinama, atliekamas po `then` esantis *PirmasSakinys*, jei ne – po `else` esantis *AntrasSakinys*. Jeigu reikia atlikti kelis sakinius, kai *Sąlyga* tenkinama arba netenkinama, tai jie jungiami į sudėtinį sakinį ir rašomi `begin end` skliaustuose.

```
if Sąlyga
    then PirmasSakinys
    else AntrasSakinys;
```

```
if Sąlyga
  then
    begin
      Sakiniai, atliekami kai sąlyga tenkinama
    end
  else
    begin
      Sakiniai, atliekami kai sąlyga netenkinama
    end;
```

Galima vartoti sutrumpintą sąlygos sakinį, kuriame yra tik šaka `then` ir veiksmai atliekami tik tuomet, kai *Sąlyga* tenkinama.

```
if Sąlyga
    then PirmasSakinys;
```

```
if Sąlyga
    then
      begin
        Sakiniai, atliekami kai sąlyga tenkinama
      end;
```

Sąlygos sakinio `then` ir `else` šakose galima užrašyti bet kokius Paskalio kalbos sakinius. Bet kurioje sąlygos sakinio šakoje galima užrašyti dar vieną sąlygos sakinį, pastarojo šakose dar vieną ir t.t. Toks sakinys vadinamas sudėtingu sąlygos sakiniu. Sudėtingų sąlygos sakinių pavyzdžiai:

```
if Sąlyga1
   then if Sąlyga2
            then PirmasSakinys
            else AntrasSakinys
   else TrečiasSakinys;
```

```
if Sąlyga1
   then if Sąlyga2
            then if Sąlyga3
                     then PirmasSakinys
                     else if Sąlyga4
                              then AntrasSakinys;
```

Paprasčiausios sąlygos aprašomos palyginimo reiškiniais, kurių sintaksė:

```
R1 PalyginimoOperatorius R2
```

Čia `R1` ir `R2` - aritmetiniai reiškiniai, o `PalyginimoOperatorius` gali būti žymimas tokiais operatoriais:

=	lygu	**<**	mažiau	**<=**	mažiau arba lygu
<>	nelygu	**>**	daugiau	**>=**	daugiau arba lygu

Jei reiškinių reikšmės tenkina palyginimo sąlygas, tai palyginimo operacijos rezultatui suteikiama loginė reikšmė `TRUE` („tiesa“), o jei ne – `FALSE` („melas“).

Pirmas pavyzdys:

```
if a > 0
   then  WriteLn('Teigiamas')
   else  WriteLn('Neigiamas arba nulis');
```

Antras pavyzdys:

```
if a > 0
```

```
    then WriteLn('Teigiamas')
    else if a = 0
           then WriteLn('Nulis')
           else WriteLn('Neigiamas');
```

Dažnai tenka tikrinti, ar kintamųjų reikšmės priklauso leistinai reikšmių atkarpai. Pavyzdžiui, gali būti skaičiuojama tik teigiamų arba lygių nuliui argumentų kvadratinės šaknys. Todėl tokioje programoje reikėtų tikrinti, ar argumentų reikšmės yra leistinos, t. y., ar pošaknyje nėra neigiamas:

```
if x >= 0
   then WriteLn('Šaknis: ', Sqrt(x):8:3)
   else WriteLn('Neigiamas pošaknis');
```

Sudėtingos sąlygos aprašomos loginiais reiškiniais. Tai reiškiniai, kuriuose naudojami loginiai operatoriai NOT, AND, OR. Jais jungiami palyginimo reiškiniai į sudėtingesnius, kurie vadinami loginiais reiškiniais.

Loginių operatorių teisingumo lentelės:

Kintamųjų reikšmės		Operacijų su loginėmis reikšmėmis rezultatai		
a	b	a **AND** b	a **OR** b	**NOT** a
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

3.8. Knygoje naudojamų ir rekomenduojamų funkcijų sąrašas

`function Abs(X): argumentoTipas;`	Rezultatas yra argumento X absoliutinė reikšmė.
`function ArcTan(X: real): real;`	Skaičiuoja duoto argumento X arktangetą.
`function Cos(X: real): real;`	Skaičiuoja duoto argumento X kosinusą.
`function Exp(X: real): real;`	Skaičiuoja duoto argumento X eksponentę.
`function Ln(X: real): real;`	Skaičiuoja duoto argumento X natūrinį logaritmą.
`function Pi: real;`	Grąžina reikšmę 3.1415926535897932385.
`function Sin(X: real): real;`	Skaičiuoja duoto argumento X sinusą.
`function Sqr(X): argumentoTipas`	Skaičiuoja duoto argumento X kvadratą.
`function Sqrt(X: real): real;`	Skaičiuoja duoto argumento X kvadratinę šaknį.
`function Round(X: real): longint;`	Realų skaičių verčia sveikuoju (`integer`) apvalinant.
`function Trunc(X: real): longint;`	Realų skaičių verčia sveikuoju (`integer`). Trupmeninė dalis atmetama

Smalsiems

3.9. Duomenų įvedimas iš failo

Duomenų įvedimas klaviatūra visuomet organizuojamas dialogo principu: ekrane spausdinami pranešimai, nusakantys kada ir kokius duomenis reikia surinkti klaviatūra. Kai duomenų skaičius didelis, tai labai nepatogus ir ilgai trunkantis procesas, nes norint pakartoti skaičiavimus, reikia duomenis suvedinėti iš naujo. Nepatogumų galima išvengti, jeigu duomenys iš anksto surašomi tekstiniame faile. Tačiau faile nerašomi pranešimai ir paaiškinimai, kokie duomenys ir kokia seka surašyti. Rašant duomenų skaitymo iš failo sakinius reikia iš anksto žinoti, kokia seka surašyti duomenys. Nuo to priklauso, kiek bus kintamųjų ir kokia seka bus skaitomos jų reikšmės iš failo

Norint duomenis perskaityti iš failo, reikia:

- Aprašyti failo kintamąjį, kurio tipas `text`, pavyzdžiui, `Fd : text;`
- Susieti diskinį failą su failo kintamuoju programoje. Tam naudojama procedūra `Assign`. Pavyzdžiui: `Assign(Fd, 'Duom.txt');` Failo vardą galima nurodyti tekstine konstanta, simbolių eilutės tipo kintamuoju, kuris saugo failo vardą.
- Atidaryti failą duomenų skaitymui. Tam panaudojama procedūra `Reset`, pavyzdžiui: `Reset(Fd);`
- Baigus darbą su failu būtina jį uždaryti. Tam panaudojama procedūra `Close`. Pavyzdžiui, `Close(Fd);`

Visų failo kintamųjų vardus siūloma pradėti didžiaja raide `F`. Tai leis programos tekste juos atpažinti be papildomų paaiškinimų. Antruoju simboliu siūloma rašyti radę `d` (duomenys) arba `r` (rezultatai). Tolesni simboliai, jeigu reikia, gali būti parenkami programuotojo nuožiūra.
Duomenims iš failo skaityti naudojamos tos pačios procedūros, kaip ir įvedamiems klaviatūra, tik pirmuoju argumentu rašomas failo kintamasis:

```
Read(Fd, kintamųjųSąrašas);
ReadLn(Fd, kintamųjųSąrašas);
```

Procedūrų darbas niekuo nesiskiria nuo jau nagrinėto duomenų įvedimo klaviatūra. Žinant, kad klaviatūra yra taip pat tekstinis failas, kuris pagal nutylėjimą yra susiejamas su failo kintamuoju `Input`, kurio nebūtina (bet galima) rašyti procedūroje pirmuoju argumentu, panaudojimo skirtumų nelieka.

3.10. Rezultatų (duomenų) išvedimas į failą

Akivaizdu, kad spausdinti skaičiavimų rezultatus ekrane galima, jeigu jų yra nedaug. Juos bus nesunku peržiūrėti ir/arba nusirašyti. Tačiau jeigu rezultatų yra daug, tai žymiai patogiau juos surašyti į tekstinį failą. Taip suformuotus rezultatus lengva panaudoti dokumentuose, kaip duomenis kitose programose, apdoroti kitomis taikomosiomis programomis (pvz., skaičiuokle).
Norint duomenis įrašyti į failą, reikia:

- Apašyti failo kintamąjį, kurio tipas `text`, pavyzdžiui, `Fr : text;`
- Susieti failą su failo kintamuoju. Tam naudojama procedūra `Assign`. Pavyzdžiui: `Assign(Fr, Rez.txt');` Failo vardą galima nurodyti tekstine konstanta arba simbolių eilutės tipo kintamuoju, kuris saugo failo vardą
- Atidaryti failą duomenų rašymui. Tam skirta procedūra `Rewrite`. Jeigu failo nurodytu vardu kataloge nėra, tuomet jis sukuriamas. Svarbu žinoti tai, kad jeigu failas tokiu vardu jau yra, tuomet jame esantys duomenys naikinami (failas išvalomas). Pavyzdžiui: `Rewrite(Fr);`
- Baigus darbą su failu, būtina jį uždaryti. Tam panaudojama procedūra `Close`. Pavyzdžiui, `Close(Fr);` Tai padaryti būtina, nes rašymas į failą vyksta netiesiogiai. Pradžioje rašoma į specialią atmintinės vietą (buferį), o ją užpildžius, informacija siunčiama į failą. Po to buferis valomas ir duomenys rašomi toliau. Procedūra `Close` siunčia į failą buferyje likusią dar neįrašytą informaciją.

Duomenims į failą spausdinti naudojamos tos pačios procedūros, kaip ir spausdinant rezultatus ekrane, tik pirmuoju argumentu rašomas failo kintamasis:

```
Write(Fr, duomenųSąrašas);
WriteLn(Fr, duomenųSąrašas);
```

Procedūrų darbas niekuo nesiskiria nuo jau nagrinėto spausdinimo ekrane. Žinant, kad ekranas yra taip pat tekstinis failas, kuris pagal nutylėjimą yra susiejamas su failo kintamuoju `Output`, kurio nebūtina (bet galima) rašyti procedūroje pirmuoju argumentu, panaudojimo skirtumų nelieka.
Papildymui failas atidaromas procedūra `Assign`. Tuomet esami duomenys nebus naikinami, o failas bus paruošiamas naujų duomenų rašymui failo pabaigoje.

3.11. Procedūros

Procedūromis yra vadinami programos struktūriniai elementai, į kuriuos galima kreiptis daugelį kartų. Jos padeda geriau struktūrizuoti programas, padaro jas lengviau skaitomas ir analizuojamas. Procedūroms yra skiriamos dviejų tipų sintaksinės struktūros: aprašymai ir kreipiniai. Aprašymai nurodo, ką procedūra turi atlikti. Vykdant programą jie praleidžiami. Procedūrose aprašyti veiksmai yra vykdomi tiktai tada, kai į jas kreipiamasi. Procedūrų aprašymai gali būti bet kurioje aprašų vietoje. Procedūrų aprašo sintaksė:

```
procedure Vardas(FormaliųjųParametrųSąrašas);
  LokaliejiAprašai
begin
  ProcedūrosKamienas     // Veiksmai, kuriuos atlieka procedūra
end;
```

`Vardas` parenkamas pagal tas pačias taisykles, kaip kintamųjų vardai.

`FormaliųjųParametrųSąrašas`, tai kintamųjų aprašymas, kurie skirti ryšiui su programa, kuri kviečia procedūrą. Kintamieji, prieš kurių aprašymą nėra žodelio `var`, skirti tik duomenų perdavimui į procedūrą. Jie vadinami parametrai-reikšmės Kintamieji, prieš kurių aprašymą yra žodelis `var`, skirti ne tik duomenų perdavimui į procedūrą, bet ir rezultatų grąžinimui iš procedūros. Jie vadinami parametrais–kintamaisiais. Parametrai skirti informaciniams ryšiams su programa palaikyti. Jų aprašymai galioja tik procedūroje.

`LokaliejiAprašai`, tai kintamųjų, kurie reikalingi procedūros kamiene surašomiems veiksmams, sąrašas. Čia gali būti ir sudėtingesni aprašai. Galioja tik procedūroje.

`ProcedūrosKamienas` aprašo procedūros vykdomus veiksmus.

Į procedūras kreipiamasi jų vardais, už kurių lenktiniuose skliaustuose surašomi faktiniai parametrai (argumentai):

```
Vardas (FaktiniųParametrųSąrašas);
```

Faktiniai parametrai nurodo, kokiais pagrindinės programos duomenimis turi būti keičiami formalūs parametrai. Parametrų keitimo būdas priklauso nuo jų tipo. Formalūs parametrai–kintamieji keičiami kreipinyje nurodytais pagrindinės programos kintamaisiais. Tokie parametrai yra naudingi tada, kai pageidaujama, kad procedūra pakeistų pagrindinės programos kintamųjų reikšmes.

Kai norime apsaugoti pagrindinės programos kintamuosius nuo pakeitimų procedūrų darbo metu, yra vartojami parametrai-reikšmės, kurių aprašuose žodelis `var` nenurodomas. Tada duomenys perduodami tik viena kryptimi – iš besikreipiančios programos vietos į procedūrą. Įvykdžius procedūrą, tokiais faktiniais parametrais nurodomos kintamųjų reikšmės nepakinta. Be to, faktiniais parametrais-reikšmėmis gali būti ne tik kintamieji, bet ir konstantos bei išraiškos.

Būtina žinoti, kad failo kintamieji gali būti tik parametrais-kintamaisiais, pavyzdžiui

```
procedure Spausdinti(var Fr : text);
```

3.12. Funkcijos

Funkcijomis yra vadinami programos struktūriniai elementai, į kuriuos galima kreiptis daugelį kartų. Jos padeda geriau struktūrizuoti programas, padaro jas lengviau skaitomas ir analizuojamas. Funkcijoms yra skiriamos dviejų tipų sintaksinės struktūros: aprašymai ir kreipiniai. Aprašymai nurodo, ką funkcija turi atlikti. Vykdant programą jie praleidžiami. Funkcijose aprašyti veiksmai yra vykdomi tiktai tada, kai į jas kreipiamasi. Funkcijų aprašymai gali būti bet kurioje aprašų vietoje. Funkcijų aprašo sintaksė:

```
function Vardas(FormaliųjųParametrųSąrašas): RezultatoTipas;
  LokaliejiAprašai
begin
  FunkcijosKamienas     // Veiksmai, kuriuos atlieka funkcija
end;
```

`Vardas` parenkamas pagal tas pačias taisykles, kaip kintamųjų vardai.

`FormaliųjųParametrųSąrašas`, tai kintamųjų aprašymas, kurie skirti ryšiui su programa, kuri kviečia funkciją ir sudaromas pagal tas pačias taisykles, kaip ir procedūrose. Parametrus–kintamuosius nerekomenduojama naudoti.

`RezultatoTipas`, tai funkcijoje suskaičiuotos reikšmės, kuri grąžinama funkcijos vardu, tipas. Funkcija skiriasi nuo procedūros tuo, kas jos vardas yra kartu ir skaičiavimų rezultato kintamasis. Funkcijos naudojamos, kai skaičiavimų rezultatas yra viena reikšmė ir ją reikia panaudoti tiesiogiai reiškiniuose (analogiškai, kaip standartines funkcijas `Abs`, `Sin`, `Sqrt` ir pan.).

`LokaliejiAprašai`, tai kintamųjų, kurie reikalingi procedūros kamiene surašytiems veiksmams atlikti, sąrašas. Čia gali būti ir sudėtingesni aprašai. Galioja tik funkcijoje.

`FunkcijosKamienas` aprašo funkcijos vykdomus veiksmus. Jų tarpe turi būti bent vienas sakinys, kuriuo suskaičiuota reikšmė priskiriama funkcijos vardui.

Į funkcijas kreipiamasi reiškiniuose jų vardais, už kurių lenktiniuose skliaustuose surašomi faktiniai parametrai (argumentai), pvz., `y := Vardas(FaktiniųParametrųSąrašas)* c;`

Faktiniams parametrams galioja tos pačios taisyklės, kaip ir procedūrose.

3.13. Knygoje naudojamų ir rekomenduojamų funkcijų ir procedūrų sąrašas

Darbas su failu	
`procedure Assign(var FailoKintamasis; FailoVardas);`	Prijungia išorinį failą.
`procedure Close(var FailoKintamasis);`	Uždaro atidarytą failą.
`procedure Reset(var FailoKintamasis: text);`	Atidaro failą skaitymui.
`procedure Rewrite(var FailoKintamasis: text);`	Atidaro failą rašymui.
`procedure Append(var FailoKintamasis: text);`	Atidaro egzistuojantį failą papildymui.
`function Eof (var FailoKintamasis: text):boolean;`	Analizuoja tekstinio failo pabaigą.
`function Eoln (var FailoKintamasis: text): boolean;`	Analizuoja failo eilutės pabaigą.

Įvairios	
`function Random[(Riba: word)]: SutampaSuKintamojo tipu;`	Grąžina atsitiktinį skaičių, ne didesnį už nurodytą ribą. Jei riba nenurodoma, tai reikšmė bus intervale [0,1].
`procedure Randomize;`	Paruošia darbui atsitiktinių skaičių generatorių su pradine atsitiktine reikšme.
`function Chr(X: byte): char;`	Grąžina nurodytą kodu simbolį.
`function Ord(X): longint;`	Grąžina nurodytos sekos elemento kodą.
`function Round(X: real): longint;`	Realų skaičių verčia sveikuoju (`integer`) apvalinant.
`function Trunc(X: real): longint;`	Realų skaičių verčia sveikuoju (`integer`). Trupmeninė dalis atmetama.
`procedure ClrScr;`	Valo ekraną.
`procedure TextColor(spalva : byte);`	Teksto spalva: spalvos kodas arba konstanta.
`procedure TextBackground(spalva : byte)`	Fono spalva: spalvos kodas arba konstanta
`procedure Window(x1, y1, x2, y2 : integer);`	Aktyvaus lango aprašymas: (x1,y1) kairiojo viršutinio ir (x2,y2) dešinio apatinio lango kampų koordinatės.
`procedure GoToXY (x, y : byte);`	Perkelia žymeklį į naują ekrano vietą (x, y).
`function WhereX: byte;`	Grąžina žymeklio vietos koordinatę x.
`function WhereY: byte;`	Grąžina žymeklio vietos koordinatę y.

Data ir laikas.
`procedure GetDate(metai, mėnuo, diena, savaitėsDiena : word);`

Savaitės dienos reikšmė nulis nurodo sekmadienį. Duomenų tipas word nurodo sveiką teigiamą skaičių.
`procedure GetTime(valanda, minutė, sekundė, sekundėsŠimtojiDalis : word);`

4. Kai kurie praktikos darbuose naudojami algoritmai


Vienas iš struktūrinio programavimo pradininkų olandas E. W. Dijkstra (1930–2002 m.)

1969 m. E. W. Dijkstra straipsnyje „Duomenų struktūros ir algoritmai“ įrodė, kad kiekvieną algoritmą galima aprašyti trimis pagrindinėmis konstrukcijomis:

1. Nuosekliai atliekamais veiksmais. Tokie algoritmai vadinami **tiesiniais**.
2. Cikliniais veiksmais, kai tas pats veiksmas kartojamas daug kartų. Tokie algoritmai vadinami **cikliniais**.
3. Veiksmų pasirinkimo veiksmais. Tokie algoritmai vadinami **šakotaisiais**.

Dijkstros teigimu, jei uždavinį galima išspręsti užrašant algoritmą, tai galima sugalvoti daugybę algoritmų, iš kurių po to išrenkamas pats efektyviausias.

4.1. Tiesiniai algoritmai

Jais vadinami veiksmai, kurie atliekami jų surašymo eilės tvarka.

Pavyzdys 1. Triženklio natūraliojo skaičiaus x skaitmenų sumos s skaičiavimas. Norint išspręsti šį uždavinį, reikia skaičių x išskaidyti skaitmenimis, skaitmenis sudėti ir pateikti skaičiavimų rezultatą.

Algoritmo užrašymas žodžiais:

1. Pradinis duomuo – triženklis natūralusis skaičius x.
2. Atskiriamas pirmasis triženklio skaičiaus skaitmuo a.
3. Atskiriamas antrasis triženklio skaičiaus skaitmuo b.
4. Atskiriamas trečiasis triženklio skaičiaus skaitmuo c.
5. Skaičiuojama triženklio skaičiaus skaitmenų suma s = a + b + c.
6. Pateikiamas gautas skaičiavimų rezultatas.

Algoritmas, užrašytas Paskalio programavimo kalba:

```
program Suma1;
   var x, a, b, c, s: integer;
begin
  WriteLn('Įveskite triženklį natūralųjį skaičių');
  ReadLn (x);
  a := x div 100;
  b := x div 10 mod 10;
  c := x mod 10;
  s := a + b + c;
  WriteLn ('Skaičiaus skaitmenų suma lygi ', s);
end.
```

4.2. Cikliniai algoritmai

Tai veiksmai, kuriuos reikia daug kartų atlikti. Veiksmų kartojimų skaičius nusakomas santykio bei loginiais reiškiniais.

Pavyzdys 2. Skaičiuoti visų natūraliųjų triženklių skaičių skaitmenų sumas. Norint išspręsti šį uždavinį, reikia kiekvieną triženklį skaičių išskaidyti skaitmenimis, skaitmenis sudėti ir pateikti gautą rezultatą.

Algoritmas žodžiais.

1. Pradinis duomuo – pirmas triženklis natūralusis skaičius x = 100.
2. Atskiriamas pirmasis triženklio skaičiaus skaitmuo a.

3. Atskiriamas antrasis triženklio skaičiaus skaitmuo b.
4. Atskiriamas trečiasis triženklio skaičiaus skaitmuo c.
5. Skaičiuojama triženklio skaičiaus skaitmenų suma s = a + b + c.
6. Pateikiamas gautas skaičiavimų rezultatas.
7. x reikšmė padidinama vienetu.
8. 2-7 veiksmai kartojami tol, kol x < 1000.

Uždavinio sprendimas, užrašytas Paskalio programavimo kalba:

```
program Suma2;
   var x, a, b, c, s: integer;
begin
  x := 100;
  while x < 1000 do
     begin
        a := x div 100;
        b := x div 10 mod 10;
        c := x mod 10;
        s := a + b + c;
        WriteLn ('Skaičiaus ',x,' skaitmenų suma lygi ', s);
        x := x + 1;
     end;
end.
```

4.3. Šakotieji skaičiavimai.

Tai veiksmai, kai, priklausomai nuo gautų ankstesnių skaičiavimo rezultatų, reikia vykdyti vienokius arba kitokius veiksmus. Tokie skaičiavimai aprašomi sąlygos sakiniais, kuriuose naudojami santykio bei loginiai reiškiniai.

Pavyzdys 3. Ciklinis algoritmas papildomas šakojimu: turi būti skaičiuojama kiekvieno natūraliojo triženklio nelyginio skaičiaus skaitmenų suma. Norint išspręsti šį uždavinį, reikia kiekvieną triženklį skaičių patikrinti, ar jis nelyginis, jei taip, tuomet išskaidyti skaitmenimis, skaitmenis sudėti ir pateikti gautą rezultatą.

Uždavinio sprendimo algoritmas užrašytas žodžiais:

1. Pradinis duomuo – pirmas triženklis natūralusis skaičius x = 100.
2. Tikrinama sąlyga, ar triženklis skaičius yra nelyginis.
   - 2.1. Jei sąlyga tenkinama, tuomet atliekami veiksmai:
     1. Atskiriamas pirmasis triženklio skaičiaus skaitmuo a.
     2. Atskiriamas antrasis triženklio skaičiaus skaitmuo b.
     3. Atskiriamas trečiasis triženklio skaičiaus skaitmuo c.
     4. Skaičiuojama triženklio skaičiaus skaitmenų suma s = a + b + c.
     5. Pateikiamas gautas skaičiavimų rezultatas.
     6. x reikšmė padidinama vienetu.
   - 2.2. Jei sąlyga netenkinama, x reikšmė padidinama vienetu.
3. 2 veiksmas kartojamas tol, kol x < 1000.

Uždavinio sprendimas Paskalio programavimo kalba:

```
program Suma3;
   var x, a, b, c, s: integer;
begin
  x := 100;
  while x < 1000 do
```

```
     begin
        if x mod 2 <> 0
           then
              begin
                 a := x div 100;
                 b := x div 10 mod 10;
                 c := x mod 10;
                 s := a + b + c;
                 WriteLn ('Skaičiaus ',x,' skaitmenų suma lygi ', s);
                 x := x + 1;
              end
           else x := x + 1;
     end;
end.
```

Veiksmus galima supaprastinti, iškėlus x reikšmės didinimą iš sąlygos tikrinimo:

```
  while x < 1000 do
     begin
        if x mod 2 <> 0
           then
              begin
                 a := x div 100;
                 b := x div 10 mod 10;
                 c := x mod 10;
                 s := a + b + c;
                 WriteLn ('Skaičiaus ',x,' skaitmenų suma lygi ', s);
              end;
         x := x + 1;
   end;
```

Sprendžiant 3 pavyzdžio uždavinį, sąlygos `if x mod 2 <> 0` galima ir netikrinti:

1. Pradinis duomuo – pirmas triženklis natūralusis skaičius x = 101.
2. Atskiriamas pirmasis triženklio skaičiaus skaitmuo a.
3. Atskiriamas antrasis triženklio skaičiaus skaitmuo b.
4. Atskiriamas trečiasis triženklio skaičiaus skaitmuo c.
5. Skaičiuojama triženklio skaičiaus skaitmenų suma s = a + b + c.
6. Pateikiamas gautas skaičiavimų rezultatas.
7. x reikšmė padidinama 2.
8. 2-7 veiksmai kartojami tol, kol x < 1000.

Uždavinio sprendimas Paskalio programavimo kalba:

```
program Suma5;
   var x, a, b, c, s: integer;
begin
  x := 101;
  while x < 1000 do
     begin
        a := x div 100;
        b := x div 10 mod 10;
        c := x mod 10;
        s := a + b + c;
```

```
        WriteLn ('Skaičiaus ',x,' skaitmenų suma lygi ', s);
        x := x + 2;
    end;
end.
```

Mokydamiesi programuoti daug dėmesio skirsime **programavimo kultūrai** – taisyklėms, kurių turi būti laikomasi rašant teisingas ir lengvai skaitomas programas. Programavimo kultūros pagrindiniai požymiai:

- tvarkingai išdėstytas programos tekstas;
- prasmingi programos objektų (kintamųjų, paprogramių, duomenų tipų vardai);
- suprantami ir reikiamose programos vietose parašyti komentarai.

Rašydami programas stenkitės laikytis šių taisyklių:

- Vieno sintaksinio lygmens sakiniai turi būti vienodai atitraukti nuo eilutės pradžios. Žemesnio lygmens sakiniai yra dešiniau negu aukštesnio.
- Jei programoje yra kelios logiškai nepriklausomos dalys, jos viena nuo kitos skiriamos tuščia eilute arba linija, sudaryta iš brūkšnelių, žvaigždučių arba kitokių ženklų.
- Programoje `begin-end`, `then-else` turi būti lygiuojami, jei jie priklauso tai pačiai struktūrai.
- Iš abiejų pusių tarpais turi būti skiriami ženklai `>`, `<`, `<>`, `>=`, `<=`, `=`, `:=`, `+`, `-`, `*`, `/`. Po ženklų **`:`** ir **`,`** paliekamas tarpas.

Smalsiems

4.4. Sumos skaičiavimo algoritmas

Kelių skaičių sumą galime užrašyti taip: `suma = sk1 + sk2 + … + skn`.
Čia `sk1, sk2, … skn` yra skaičiai, kuriuos sudėję gauname sumą `suma`.
Atlikdami veiksmus skaičiuokliu (ar pieštuku popieriuje), visuomet sumuojame du skaičius. Veiksmus galime išskleisti taip:

```
suma2 = sk1 + sk2;       // Pirmųjų dviejų skaičių suma
suma3 = suma2 + sk3;     // Ankstesnio rezultato ir trečio skaičiaus suma
suma4 = suma3 + sk4;     // Ankstesnio rezultato ir ketvirto skaičiaus suma
. . .
suma = suman-1 + skn ;   // Visų n skaičių suma
```

Skaičiuoklio ekrane visuomet matome tik paskutinio veiksmo rezultatą (dalinę sumą). Kiekviena nauja suma gaunama prie jau apskaičiuotos sumos pridedant po vieną dėmenį. Todėl užrašytoje veiksmų sekoje visose eilutėse galima parašyti:

```
suma = sk1 + sk2;    // Pirmųjų dviejų skaičių suma
suma = suma + sk3;   // Trijų skaičių suma
suma = suma + sk4;   // Keturių skaičių suma
. . .
suma = suma + skn;   // Visų n skaičių suma
```

Sumos skaičiavimą galima užrašyti: `suma = suma + skaičius`. Tokiu atveju pradinė `suma` reikšmė turi būti lygi nuliui, nes:

```
suma = 0;            // Pradinė rezultato reikšmė
suma = suma + sk1;   // Pirmojo skaičiaus ir nulio suma
suma = suma + sk2;   // Dviejų skaičių suma
suma = suma + sk3;   // Trijų skaičių suma
. . .
suma = suma + skn;   // Visų n skaičių suma
```

Bendru atveju veiksmus galime išreikšti tokiu algoritmu:

```
Pradžia Skaičių suma
  suma = 0;
  kol yra skaičių
   ciklo pradžia
      Įvesti skaičių sk;
      suma = suma + sk;
  ciklo pabaiga
  Spausdinti (rodyti) rezultatą suma
Pabaiga
```

Ciklo sąlygą „kol yra skaičių“ galime nurodyti labai įvairiai. Paprasčiausias atvejis, kai prieš ciklą yra tiksliai apibrėžiama, kiek bus skaičių, kuriuos reikia sudėti.

Galimas toks programos tekstas:

```
// Skaičių suma
program SkaiciuSuma;
   var suma, sk : integer; // Sumos ir dėmens kintamieji
               n : integer; // Skaičių kiekis
               i : integer; // Ciklo parametras
begin
   Write('Kiek bus skaičių?'); ReadLn(n);
   suma := 0;
   for i := 1 to n do
     begin
        Write('Koks bus ', i, ' skaičius?');  ReadLn(sk);
        suma := suma + sk;
     end;
   WriteLn('Sudėta skaičių: ', n, ' Rezultatas = ', suma);
   WriteLn('Pabaiga');
   ReadLn;
end.
```

```
Kiek bus skaičių? 3
Koks bus 1 skaičius? 12
Koks bus 2 skaičius? 5
Koks bus 3 skaičius? 10
Sudėta skaičių: 3 Rezultatas = 27
Pabaiga
```

Kaip matote, sumos skaičiavimo veiksmai panašūs į kasininkės veiksmus parduotuvėje:

1. Pradedant skaičiuoti, kasos aparato ekrane turi būti rodoma reikšmė nulis.
2. Skenuojant prekes, jų kaina pridedama prie ekrane matomo skaičiaus. Pinigų suma, kurią reikės mokėti, didėja.
3. Nuskenavus paskutinės prekės kainą, ekrane matomas rezultatas.

Procesą galime pavaizduoti tokia schema

```
suma := 0
        Kol yra prekių
     suma := suma + prekesKaina
Rezultatas: suma
```

4.5. Sandaugos skaičiavimo algoritmas

Kelių skaičių sandaugą galime užrašyti taip: `sand = sk1 * sk2 * … * skn`.
Čia `sk1, sk2, … skn` yra skaičiai, kuriuos sudauginus gauname sandaugą `sand`.
Atliekant veiksmus skaičiuokliu (ar pieštuku popieriuje), visuomet dauginame du skaičius. Veiksmus galime išskleisti taip pat, kaip ir skaičių sumavimo atveju. Kiekviena nauja sandauga gaunama jau apskaičiuotą

sandaugą dauginant iš naujo dauginamojo: `sand = sand * skaičius`. Pradinė sandaugos reikšmė turi būti lygi vienetui.
Sandaugos skaičiavimų seka bus tokia:

```
sand = 1;           // Pradinė rezultato reikšmė
sand = sand * sk1; // Pirmojo skaičiaus ir vieneto sandauga
sand = sand * sk2; // Dviejų skaičių sandauga
sand = sand * sk3; // Trijų skaičių sandauga
. . .
sand = sand * skn; // Visų n skaičių sandauga
```

Bendru atveju veiksmus galime išreikšti tokiu pačiu algoritmu, kaip ir sumavimo, tik pradinė sandaugos reikšmė turi būti vienetas, o sudėties operatorius (+) pakeistas į daugybos (*).

Kėlimas laipsniu. Sandaugos skaičiavimo algoritmas panaudojamas skaičiaus kėlimui sveikuoju laipsniu: $R = x^n$.

```
// Skaičiaus kėlimas laipsniu. Panaudojamas sandaugos skaičiavimo algoritmas.
program Laipsnis;
var i : integer;   // Ciklo parametras
    R,             // Sandaugos kintamasis
    x,             // Laipsnio pagrindas
    n : integer;   // Laipsnio rodiklis
begin
  Write('Skaičius, kurį kelsite laipsniu: ');  ReadLn(x);
  Write('Laipsnio rodiklio reikšmė: ');  ReadLn(n);
  R := 1;
  for i := 1 to n do
     R := R * x;
  WriteLn('Rezultatas: ', R);
  WriteLn('Pabaiga');
  Readln;
end.
```

4.6. Kiekio skaičiavimo algoritmas

Kiek yra dviženklių natūraliųjų skaičių, kurie dalinasi iš penkių?

```
program KiekSkaiciu;
   var x : integer;  // Pirmasis dviženklis skaičius, kuris dalinasi iš penkių
       k : integer;  // Dviženklių skaičių, dalių iš 5, kiekis
begin
   x := 10; k := 0;
   while x < 100 do
     begin
       k := k + 1;
       x := x + 5;
     end;
   WriteLn ('Dviženklių skaičių, dalių iš 5, yra ', k);
   ReadLn;
end.
```

Uždavinio sprendimo algoritmas. Pirmasis dviženklis skaičius, kuris dalinasi iš penkių, yra 10. Kiekio pradinė reikšmė yra lygi nuliui. Ciklo antraštėje rašome sąlygą, kad ciklo vykdymas būtų nutrauktas, peržiūrėjus visus dviženklius skaičius. Kadangi į ciklą ateiname su pirmuoju dviženkliu skaičiumi, kuris dalinasi iš 5, tai tokių skaičių kiekis padidėja vienu nauju skaičiumi. Po to x reikšmė didinama 5, tikrinama ciklo sąlyga ir atliekamai veiksmai ciklo viduje.
Veiksmų algoritmas analogiškas sumos skaičiavimui, tik čia bus sumuojamos ne reikšmės, o vienetukai.

Dar vienas pavyzdys: rasti, kiek natūralusis skaičius x turi daliklių k.
Uždavinio sprendimo algoritmas yra klasikinis: sukamas ciklas nuo 1 iki x ir cikle tikrinama, ar natūralusis skaičius x be liekanos dalinasi iš ciklo kintamojo i. Jei taip, tai i yra x daliklis ir daliklių skaičius didinamas vienu nauju dalikliu.

```
program Dalikliai;
      var x, i, k : integer;
begin
   WriteLn ('Įveskite natūralųjį skaičių x');
   ReadLn (x);
   k := 0;
   for i := 1 to x do
     if x mod i = 0
       then k := k + 1;
   WriteLn ('Natūralusis skaičius ', x, ' turi ', k,' daliklių (-ius)');
   ReadLn;
end.
```

4.7. Aritmetinio vidurkio skaičiavimas

Skaičiavimus sudaro dvi atskiros dalys:

1. Skaičių sumos skaičiavimas;
2. Gautos sumos padalinimas iš skaičių kiekio.

Pirmoji dalis jau nagrinėta. Reikia atkreipti dėmesį, kad jeigu iš anksto nežinomas sumuojamų skaičių kiekis, tai reikia sumuojant skaičius kartu skaičiuoti ir kiek tokių skaičių buvo.
Antrą veiksmų dalį sudaro dviejų skaičių dalybos veiksmas: `vidurkis = suma / skaičiųKiekis`.

Ne visuomet toks veiksmas bus teisingas. Tarkime, kad reikia suskaičiuoti teigiamų skaičių aritmetinį vidurkį. O jeigu nei vieno teigiamo skaičiaus nebuvo? Visi skaičiai buvo tik neigiami ir nuliai. Žinome, kad dalyba iš nulio negalima. Taigi, prieš atliekant dalybos veiksmą, būtina įsitikinti, kad galima dalinti.

4.8. Didžiausios (mažiausios) reikšmės paieška

Tai tradiciniai programavimo uždaviniai. Populiariausias jų sprendimo būdas yra toks. Aprašomi du kintamieji įvedamai (`x`) ir didžiausiai (`d`) reikšmėi saugoti. Įvedant pirmąją reikšmę, daroma prielaida, kad ši yra didžiausia (kai ieškoma didžiausios reikšmės):

```
      d := x;
```

Po to, paeiliui skaitomos kitos reikšmės ir lyginamos su `d`. Jei randama didesnė, kintamojo `d` reikšmė keičiama nauja:

```
      if  x > d  then  d := x;
```

Taip patikrinus visą įvedamą duomenų srautą, kintamojo `d` reikšmė bus didžiausia įvesta reikšmė.
Analogiškai ieškoma mažiausia reikšmė. Skirtumas tik palyginimo sąlygoje:

```
      if  x < d  then  d := x;
```

Pateikiame vieną tokio algoritmo aprašymo Paskalio kalba variantą:

```
program Didelis;
 var n, x, d, i : integer;
```

```
begin
   Write('Kiek bus skaičių: ');  ReadLn(n);
   Write('Įveskite pirmą skaičių: ');  ReadLn(d);
   for i := 2 to n do begin
      Write('Įveskite ', i,'- skaičių: ');  ReadLn(x);
      if x > d then d := x;
   end;
   WriteLn(' Didžiausias skaičius: , d);
end;
```

Pirmąją reikšmę laikyti pradine didžiausia (arba mažiausia) reikšme ne visuomet patogu. Ypač tuomet, kai ta reikšmė turi būti apskaičiuojama (pavyzdžiui, ieškant mažiausio teigiamo skaičiaus, kai srauto pradžioje gali būti daug neigiamų skaičių). Tokiu atveju galima pradinei didžiausiai reikšmei priskirti pakankamai mažą reikšmę: tokią, kuri tikrai būtų mažesnė už visas galimas reikšmes, tarp kurių ieškome didžiausios reikšmės. Cikle analizuojamos visos reikšmės. Ieškant mažiausios reikšmės, pradinei reikšmei reikia priskirti pakankamai didelį skaičių, geriausia jį laikyti lygiu standartinei Paskalio konstantai `maxint` (arba `maxlongint`, jei sveikasis tipas – `longint`).

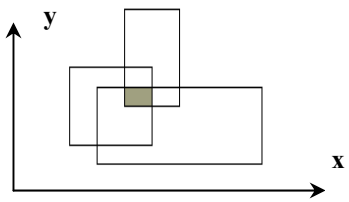
Pavyzdys. Turguje ant prekystalio yra n arbūzų, kurių masės žinomos. Reikia sudaryti programą sunkiausiam arbūzui rasti.

```
program Turgus;
  var  n  : integer;        // Arbūzų skaičius
       m  : real;           // Arbūzo masė
       i  : integer;        // Arbūzo numeris
       s  : real;           // Sunkiausio arbūzo masė
       ns : integer;        // Sunkiausio arbūzo numeris
//-----------------------------------------------------
begin
  Write('Kiek yra arbūzų?  ');  Read(n);
                                    //   Sunkiausio arbūzo paieška
  WriteLn('Arbūzų masių įvedimas');
  s := 0;  ns := 0;
  for i := 1 to n do
    begin
      Write('Įveskite', i:2, '-ojo arbūzo masę: ');  ReadLn(m);
      if m > s then
        begin
          s := m; ns := i;
        end;
    end;
                                  //  Rezultatai
  WriteLn('Sunkiausias arbūzas Nr', ns:4);
  WriteLn('Jo masė:', s:6:2);
  WriteLn('Spauskite ENTER'); ReadLn;
end.
```

5. Užduotys savarankiškam darbui

TRYS SODININKAI. Trys draugai apsigyvenę kaime nusprendė mokytis sodininkauti. Kaime buvo didžiulis sodas, jame augo po vieną vaismedį kiekviename kvadratiniame ploto vienete.

Kiekvienas iš trijų draugų pasirinko stačiakampį sklypą ir nusprendė prižiūrėti jame esančius medžius. Susirinkus draugėn paaiškėjo, kad jų pasirinkti sklypai persidengia, t. y. kai kuriuos vaismedžius prižiūrės ne vienas, o keletas sodininkų.



Užduotis. Parašykite algoritmą, kuris suskaičiuotų, kiek vaismedžių panorėjo prižiūrėti visi trys draugai.

Pradiniai duomenys pateikiami trijose tekstinės bylos SODAS.DAT eilutėse. Į kiekvieną eilutę įrašyta po keturis skaičius, apibūdinančius kiekvieno draugo pasirinktą sklypą: sklypo apatinio kairiojo ir viršutinio dešiniojo kampų koordinatės (pirma koordinatė x, po to – y.). Visos koordinatės sveikieji skaičiai. (10 olimpiada, 1999).

Testo nr.	1	2	3	4	5	6
Pradiniai duomenys	2 3 6 7 4 1 8 5 6 0 10 3	1 5 9 9 5 6 13 12 6 1 9 11	0 4 6 8 0 4 6 8 0 0 6 4	10 8 13 12 2 3 9 7 6 0 13 2	30 30 80 70 10 20 70 90 50 20 100 90	10 30 100 80 60 10 120 60 70 30 100 50
Rezultatas	0	9	0	0	800	600

JAPONŲ KALENDORIUS. Senovės japonų kalendorių sudarė 60 metų ciklas. Visi metai cikle buvo sunumeruoti nuo 1 iki 60 ir suskirstyti poromis, kurių kiekviena turėjo savo spalvą (žalią, raudoną, geltoną, baltą ar juodą). Ciklo metų spalvos buvo paskirstytos šitaip:

- 1, 2, 11, 12, 21, 22, …, 51, 52 metai – žalia spalva;
- 3, 4, 13, 14, 23, 24, …, 53, 54 metai – raudona spalva;
- 5, 6, 15, 16, 25, 26, …, 55, 56 metai – geltona spalva;
- 7, 8, 17, 18, 27, 28, …, 57, 58 metai – balta spalva;
- 9, 10, 19, 20, 29, 30, …, 59, 60 metai – juoda spalva;

Žinoma, kad naujasis 60 metų ciklas prasidėjo 1984–aisiais ir baigsis 2043–iaisiais metais; 1984–ieji ir 1985–ieji buvo žalios spalvos metai, 1986-ieji ir 1987–ieji buvo raudonos spalvos metai, 2043–ieji bus juodos spalvos metai.

Užduotis. Duoti metai m ($1800 \le m \le 2200$). Parašykite programą, kuri nustatytų ir išspausdintų, kokia duotųjų metų spalva.

Testo nr.	*Pradinis duomuo*	*Rezultatas*	*Paaiškinimai*
1	1984	ŽALIA	Paprasčiausias atvejis – 1984-ieji metai
2	2001	BALTA	Einamieji metai
3	1804	ŽALIA	Pirmieji ciklo metai
4	2103	JUODA	Paskutinieji ciklo metai
5	1945	ŽALIA	Žalios spalvos metai
6	2137	RAUDONA	Raudonos spalvos metai
7	1859	GELTONA	Geltonos spalvos metai
8	1970	BALTA	Baltos spalvos metai

9	1942	JUODA	Juodos spalvos metai (baigiasi skaitmeniu 9)
10	1943	JUODA	Juodos spalvos metai (baigiasi nuliu)
11	2200	BALTA	Ribinis atvejis

SKAITMENYS. Laura mokosi rašyti skaičius. Mokytoja jai liepė parašyti visus natūraliuosius skaičius nuo 1 iki N. Įdomu, kiek skaitmenų Laura iš viso parašys savo sąsiuvinyje?

Užduotis. Parašykite programą, kuri rastų Lauros parašytą bendrą skaitmenų skaičių.

Pradiniai duomenys. Pirmoje ir vienintelėje pradinių duomenų failo `skait.in` eilutėje pateiktas paskutinysis Lauros parašytas natūralusis skaičius N ($1 \le N < 10\,000$).

Rezultatas. Rezultatų faile `skait.out` turi būti įrašytas vienas skaičius – bendras Lauros parašytų skaitmenų skaičius. (19 olimpiada, 2008)

Pavyzdžiai

Pradiniai duomenys	***Rezultatas***	***Paaiškinimai***
`10`	`11`	`1 2 3 4 5 6 7 8 9 10` – iš viso 11 skaitmenų
`5`	`5`	`1 2 3 4 5` – iš viso 5 skaitmenys.

ŽIOGAS. Žiogas tupi ant horizontaliai ištemptos virvutės, prie pat kairiojo krašto. Virvutės ilgis *s* sprindžių. Žiogas moka šokti į priekį *a* sprindžių ir atgal *b* sprindžių. Jam reikia patekti ant virvutėje užmegzto mazgo, kuris nutolęs nuo žiogo pradinės padėties per *c* sprindžių (visi sprindžiai vienodo ilgio).

Pradiniai duomenys s, *a*, *b* ir *c* – natūralieji skaičiai, be to, $s > c > a > b > 0$.

Parašykite algoritmą, kuris apskaičiuotų, kiek mažiausiai šuolių turi padaryti žiogas, kad pasiektų mazgą. (8 olimpiada, 1997).

Testo nr.	*Pradiniai duomenys*	*Rezultatas*	*Paaiškinimai*
1	`10 3 2 6`	`2`	Žiogas turi šokti du kartus į priekį
2	`20 3 1 5`	`3`	Žiogas šoks du kartus į priekį ir kartą atgal
3	`20 4 2 5`	`Negalima`	Mazgo pasiekti negalima
4	`9 7 4 8`	`Negalima`	Mazgą galima būtų pasiekti tik tada, jei virvutė būtų begalinio ilgio (tada reiktų 9 šuolių)
5	`100 13 1 27`	`15`	Žiogas šoka dideliais žingsniais į priekį ir mažais atgal

SULTIS GERTI SVEIKA. Kiekvienos savaitės pirmadienio rytą Jonas gauna k centų kišenpinigiams. Vienas butelis sulčių kainuoja *s* centų, o tušti buteliai superkami po *b* centų. Kartą (tai buvo *i*-oji savaitės diena), iš viso turėdamas *n* centų, Jonas nusprendė kasdien (pradedant *i*-ąja diena) pirkti sulčių už visus turimus pinigus. Pinigai, gauti pardavus butelius, bus panaudojami kitą dieną sultims pirkti. Šitaip Jonas darys tol, kol jis įstengs nusipirkti bent vieną sulčių butelį.

Kiek butelių sulčių išgers Jonas?

Parašykite programą šiam uždaviniui spręsti. *Pradiniai duomenys* – natūralieji skaičiai *k*, *s*, *b*, *n*, *i* ($1 <= i <= 7$). Jei nusprendžiama sultis pirkti pirmadienį, tai iš karto gaunami kišenpinigiai (pasipildo turima pinigų suma). Už pirmadienį gautus kišenpinigius sultys perkamos pirmadienį. (9 olimpiada, 1998).

Testo nr.	*Pradiniai duomenys*	*Rezultatas*	*Paaiškinimai*

1	$k = 25$, $s = 10$, $b = 3$, $i = 2$, $n = 15$	1	Pirmąją dieną nusiperkamas tik vienas butelis
2	$k = 24$, $s = 10$, $b = 2$, $i = 1$, $n = 1$	2	Prieš perkant sultis pirmąją dieną, būtina pasiimti kišenpinigius
3	$k = 25$, $s = 9$, $b = 2$, $i = 4$, $n = 23$	3	Pinigų trečiajam buteliui užteks tik tuomet, jei bus parduoti abu buteliai
4	$k = 100$, $s = 10$, $b = 2$, $i = 5$, $n = 500$	74	Didesnis testas, kai buteliai perkami kelias dienas, o tarp dienų yra pirmadienis

DIDMENINIS PIRKIMAS. Žinoma, kad perkant daugiau prekių jų vienetas kainuoja pigiau. Pundelyje yra 12 porų kojinių, dėžėje – 12 pundelių. Pavyzdžiui, kojinių dėžė kainuoja 247 litus, pundelis – 21 lt, pora – 2 lt. Įdomu tai, kad jei mums reikėtų 11 porų kojinių, tai geriau pirkti pundelį ir vienerias kojines kam nors atiduoti.

Užduotis. Pirkėjas nori įsigyti *n* porų kojinių. Sudarykite algoritmą, vadovaudamiesi kuriuo pigiausiai nupirktume kojines. Jei už tą pačią kainą galima nupirkti didesnį ir mažesnį kiekį kojinių, tai perkamas didesnis kiekis. Raskite perkamų dėžių, pundelių ir porų skaičių.

Pradinius duomenis sudaro keturi skaičiai: kojinių porų skaičius *n*, vienos dėžės, vieno pundelio bei vienos poros kaina litais. (10 olimpiada, 1999).

Testo nr.	*Pradiniai duomenys*	*Rezultatai*	*Paaiškinimai*

1	`720 303 32 3`	`Dėžių skaičius: 5`
		`Pundelių skaičius: 0`
		`Porų skaičius: 0`
			n dalus iš 144 (porų skaičiaus dėžėje)

2	`84 210 20 2`	`Dėžių skaičius: 0`
		`Pundelių skaičius: 7`
		`Porų skaičius: 0`
			n dalus iš 12 (porų skaičiaus pundelyje)

3	`196 100 20 5`	`Dėžių skaičius: 2`
		`Pundelių skaičius: 0`
		`Porų skaičius: 0`
			Perkame tik dėžes; jei pirktume kitais dviem būdais (1, 4, 4 arba 1, 5, 0), kaina būtų ta pati, tačiau kojinių kiekis mažesnis

4	`355 292 41 6`	`Dėžių skaičius: 2`
		`Pundelių skaičius: 6`
		`Porų skaičius: 0`
			Perkame tik dėžes ir pundelius

5	`355 291 50 6`	`Dėžių skaičius: 3`
		`Pundelių skaičius: 0`
		`Porų skaičius: 0`
			Perkame tik dėžes

6	`355 292 49 6`	`Dėžių skaičius: 2`
		`Pundelių skaičius: 5`
		`Porų skaičius: 7`
			Perkame ir dėžes, ir pundelius, ir poras

VAIZDO ĮRAŠAI. 240 minučių trukmės vaizdajuostė kainuoja 10 litų 90 centų, 180 minučių trukmės vaizdajuostė kainuoja 9 litus 15 centų. Tomas peržiūrėjo Baltijos televizijos savaitės programą ir panorėjo įsirašyti n laidų. Žinoma kiek valandų ir kiek minučių trunka kiekviena laida. Reikalui esant, bet kuri laida gali būti suskaidyta į dalis ir įrašyta į dvi ar daugiau vaizdajuosčių, o į vieną vaizdajuostę gali būti rašomos kelios laidos. Deja, Tomas neturi nei vienos tuščios vaizdajuostės.

Užduotis. Parašykite algoritmą, kuris suskaičiuotų, kiek mažiausiai pinigų (litų bei centų) reikės išleisti Tomui, norint nusipirkti tiek vaizdajuosčių, kad jų užtektų norimoms laidoms įrašyti. (11 olimpiada, 2000).

Pirmoje kiekvieno duomenų rinkinio eilutėje nurodytas laidų skaičius *n*, o likusiose *n* eilučių – kiekvienos laidos trukmė. Pirma nurodomas valandų, paskui minučių skaičius.

Testo nr.	1	2	3	4	5	6	7	8	9	10	11
Pradiniai duomenys	1 1 10	3 2 00 2 00 0 30	4 0 30 0 20 3 40 1 30	3 2 30 2 30 1 01	5 1 10 1 05 1 00 4 45 1 00	5 1 10 1 05 1 00 4 45 1 05	3 5 30 3 20 2 00	3 5 00 5 00 3 00	3 5 00 5 00 3 05	6 3 00 3 00 3 00 3 00 3 00 0 55	6 8 30 8 30 8 30 8 30 8 30 2 00
Bendra laidų trukmė	1:10	4:30	6:00	6:01	9:00	9:05	10:50	13:00	13:05	15:55	44:30
Valandų skaičius	2	5	6	7	9	10	11	13	14	16	45
Ilgų ir trumpų vaizdajuosčių skaičius	0 1	0 2	0 2	1 1	0 3	1 2	2 1	1 3	2 2	4 0	9 3
Rezultatas	9,15	18,30	18,30	20,05	27,45	29,20	30,95	38,35	40,10	43,60	125,55

VARLIŲ KONCERTAS. Varlių koncertas. Kartą vienoje kūdroje gyveno daug varlių, ir ne bet kokių, o dresiruotų. Kiekviena varlė sugebėdavo iššokti iš vandens ir sukvarksėti kas tiksliai jai būdingą laiko periodą. Pavyzdžiui, varlės kvarkimo periodas lygus 5. Tai reiškia, kad jei varlė sukvarksėjo pirmą minutę, antrą kartą ji kvarksės po penkių minučių, t. y. šeštą minutę, trečią kartą – vienuoliktą minutę ir t. t.

Užduotis. Patekėjus saulei visos varlės iššoko iš vandens ir sukvarksėjo. Sudarykite algoritmą, kuris nustatytų, po kiek valandų ir minučių (<60) įvyks antrasis varlių koncertas, t. y. vienu metu iššoks iš vandens ir sukvarksės visos kūdroje esančios varlės.

Pradiniai duomenys įvedami tokia tvarka. Pirma įvedamas varlių skaičius, po to – kiekvienos varlės kvarkimo periodas. Varlių pasirodymo periodai surašyti iš eilės: p_1, p_2, p_3, p_4, p_5 ir t. t., čia p_1 – pirmosios varlės pasirodymo periodas, p_2 – antrosios ir t. t. ($0 < p_i <= 20$ minučių). Varlių skaičius kūdroje neviršijo 10.

Atkreipiame dėmesį, kad valandų skaičius, po kurio įvyks antrasis koncertas, neviršija *maxlongint*. (11 olimpiada, 2000).

Testo nr.	*Pradiniai duomenys*	*Rezultatai*	*Paaiškinimai*
1	2 8 8	Koncertas įvyks po 0 val. ir 8 min.	Paprasčiausias atvejis: abu periodai sutampa
2	3 2 3 5	Koncertas įvyks po 0 val. ir 30 min.	Periodai neturi bendrų daliklių: kartotinis lygus periodų sandaugai
3	6 12 15 20 10 6 30	Koncertas įvyks po 1 val. ir 0 min.	Periodai turi bendrų daliklių
4	5 18 7 3 2 4	Koncertas įvyks po 4 val. ir 12 min.	Tikrinama, ar laikas teisingai perskaičiuojamas į valandas ir minutes
5	10 11 12 13 14 15 16 17 18 19 20	Koncertas įvyks po 3879876 val. ir 0 min.	Šiuo testu tikrinamas sprendimo efektyvumas

SENAS KALENDORIUS. Metai yra keliamieji, jeigu jie dalūs iš 4 ir nesidalija iš 100, arba jeigu jie dalūs iš 400. Pavyzdžiui:

- 2000 metai buvo keliamieji, nes jie dalūs iš 400;
- 2004 metai keliamieji, nes jie dalūs iš 4 ir nėra dalūs iš 100;
- 1900 metai nėra keliamieji, nes jie dalūs iš 100, bet ne iš 400.

Užduotis. Turime seną kalendorių, kuris buvo išleistas tarp 1900 ir 2004 metų. Parašykite programą, kuri rastų artimiausius būsimus metus (t. y. pradedant 2005-aisiais), kuriems tinka turimas kalendorius.
Pradiniai duomenys – du sveikieji skaičiai – skaitomi iš ekrano. Pirmasis skaičius – tai metai, kuriems buvo išleistas senasis kalendorius. Antrasis skaičius yra iš intervalo [1..7] ir nusako, kuria savaitės diena prasidėjo tie metai.
Rezultatas rašomas ekrane.
Pavyzdžiui, jeigu pradinis duomuo yra `1979 1`, tai rezultatas turėtų būti `2007`. (16 olimpiada, 2005)

DEGTUKAI. Duota *n* degtukų. Parašykite algoritmą, kuris nustatytų, ar iš duotų degtukų galima sudėlioti bent vieną iš šių figūrų: lygiakraštį trikampį, kvadratą ar stačiakampį. Dėliojamai figūrai turi būti panaudoti visi degtukai; be to, degtukų laužyti negalima. (8 olimpiada, 1997).

Testo nr.	*Pradinis duomuo*	*Rezultatas*	*Paaiškinimai*
1	1	Negalima	Per mažai degtukų;
2	2	Negalima	Per mažai degtukų;
3	3	Galima	Paprastas atvejis, kai galima sudėti lygiakraštį trikampį;
4	12	Galima	Galima sudėti visas figūras;
5	15	Galima	Galima sudėti tik trikampį;
6	16	Galima	Galima sudėti kvadratą ir stačiakampį;
7	35	Negalima	Negalima sudėti nei vienos figūros;

NUTRINTI SKAIČIAI. Ant popieriaus lapo buvo užrašyti keturi natūralieji skaičiai: a, b, s, d. Po to du iš jų buvo nutrinti (juos žymėsime nuliais). Reikia atstatyti nutrintuosius skaičius, jeigu žinoma, kad yra likęs bent vienas iš skaičių *a* ir *b* ir kad skaičiai tenkino šitokias lygybes:

$s = a + b$;
$d = a \times b$. (8 olimpiada, 1997).

Testo nr.	*Pradiniai duomenys*	*Rezultatas*	*Paaiškinimai*
1	0 12 0 48	4 12 16 48	Nutrinti skaičiai *a* ir *s*
2	0 5 9 0	4 5 9 20	Nutrinti skaičiai *a* ir *d*
3	3 0 0 39	3 13 16 39	Nutrinti skaičiai *b* ir *s*
4	15 0 105 0	15 90 105 1350	Nutrinti skaičiai *b* ir *d*
5	25 13 0 0	25 13 38 325	Nutrinti skaičiai *s* ir *d*
6	1 0 0 32766	1 32766 32766 32766	Rezultatai – skaičiai, artimi *maxint*;

Literatūra

1. Jonas Blonskis, Kazys Baniulis, Vacius Jusas, Romas Marcinkevičius, Jonas Smolinskas. Programavimas. Vadovėlis, II leidimas. – Kaunas: Technologija, 2000, 378 psl. ISBN 9986 13 673 3.
2. Jonas Blonskis, Vytautas Bukšnaitis, Jonas Smolinskas, Antanas Vidžiūnas. Programavimo praktikumas. Programavimas su Turbo Paskaliu. – Kaunas: Smaltija, 2000, 251 psl. ISBN 9986-965-18-7.
3. Bronislovas Burgis, Antanas Kulikauskas. Kompiuterija.Mokymosi knyga studentams, moksleiviams, entuziastams. – Kaunas: Naujasis lankas, 2000, ISBN 9955 03 026 7
4. Dagienė V., Grigas G. Programavimo pradmenų uždavinynas / Realinio profilio vidurinėms mokykloms. – Vilnius: TEV, 2000, 208 p.
5. Jonas Blonskis, Valentina Dagienė. Programavimo pradmenys. Vadovėlis XI-XII klasėms. – Vilnius: TEV, 2001, 270 psl. ISBN 9955-491-01-9.
6. Dagienė V. Informatikos pradmenys, IX–X kl., II d.: Algoritmai. – Vilnius: TEV, 1998, 104 p.; Pataisytas ir papildytas leidimas, Vilnius: TEV, 2001, 112 p.
7. Valentina Dagienė. Informacinės technologijos IX-X klasėms. 2 dalis. – Vilnius, TEV, 2003, 256 p. ISBN: 9955-491-39-6
8. Valentina Dagienė, Alvida Lozdienė. Informacinės technologijos. Algoritmai: Paskalis. Pratybų sąsiuvinis IX-X klasėms. – Vilnius: TEV, 2004, 64 p. ISBN: 9955-491-83-3
9. Dagienė V., Lozdienė A. Programavimas Paskaliu, – Vilnius: TEV, 2005, 64 p.
10. Valentina Dagienė, Gintautas Grigas, Tatjana Jevsikova. Enciklopedinis kompiuterijos žodynas. – Vilnius: TEV, 2005, 388 p. ISBN 9955-680-19-9. Internetinis žodyno variantas – svetainėje www.likit.lt

Naudingos nuorodos

1) http://ims.mii.lt/asmen/rimga/vu/didaktika/free_pascal/fpc_ir_tp_duomtipiu_skirtumai.htm

R. Laucius. Free Pascal ir Turbo Pascal duomenų tipų skirtumai

2) http://62.80.232.136/informatika/kurybiniai_darbai/vad9_10/

Kompiuterinis "**Informatikos pradmenų vadovėlis**" sukurtas Alytaus Vidzgirio vidurinėje mokykloje ir vėliau modifikuotas Alytaus Likiškėlių vidurinėje mokykloje. Autorius - informatikos vyr. mokytojas Vidas Žemaitis. Idėja kurti šį vadovėlį kilo tada, kai tapo aišku, kad mokykloje truks INFORMATIKOS PRADMENŲ vadovėlių 9 - 10 klasių mokiniams. Vadovėlyje mokomoji medžiaga pateikta pagal Valentinos Dagienės vadovėlius INFORMATIKOS PRADMENYS (I ir II dalis). Kiekvieną skyrių sudaro teorija bei pratimai ir užduotys. Kai kurių skyrių pratimuose ir užduotyse galima rasti testus pagal išeitą temą. Testų užduočių skaičius nuo 10 iki 14, o mokiniui pateikiamos tik 8 atsitiktinai parinktos užduotys.

3) http://vmc.ppf.ktu.lt/saulius/v2/?puslapis=327

Autorius: Sinkevičius Saulius; 2001 m. išleista ***„Turbo Pascal 7.0 trumpai. Žinynas“*** knyga.
Ši knyga gavo grifą, t. y. ji yra rekomenduojama Lietuvos Respublikos švietimo ir mokslo ministerijos.
Recenzavo dr. Jonas Blonskis, dr. Valentina Dagienė, mok. metodininkas Arvydas Verseckas

2000 m. padarytas informatikos puslapis, kuris buvo pastoviai pildomas iki 2002 metų. Vėliau tik po didelės pertraukos vėl atnaujintas. Puslapis pirmą kartą publikuotas 2000-06-25. Šis darbas 2002-03-17 laimėjo pirmąją vietą KTU Informatikos forume, moksleivių sukurtų programų konkurse. Puslapis susilauktų žymiai didesnio mano darbo, jei būtų šioks toks finansavimas.

Darbe panaudotos technologijos: FrontPage, Java Script, Flash, GIF animacija, Java Apletai. 2002 m. atnaujinant panaudota: PHP, MySQL, CSS.)

4) http://www.ivinskis.kursenai.lm.lt/gimnaz/dalykai/files/ppp10/
A. Verseckas. Programavimo pradmenys

5) http://www.ipc.lt/21z/mokymas/mokymopr/el/ek/knyga1.htm
G.Grigas. Programavimas Paskaliu. Elektroninė knyga
Ši knyga yra programavimo pradžiamokslis. Aprašomos pagrindinės Paskalio kalbos konstrukcijas bei jų bei jų panaudojimas algoritmams ir programoms užrašyti. Medžiaga pateikiama taip, kad skaitytojas galėtų ją įsisavinti spręsdamas uždavinius. Pateikiami uždaviniai ir praktikos darbai. Uždaviniai skirti savikontrolei. Knygos pabaigoje pateikiami jų sprendimai (atsakymai). Praktikos darbai – tai nedideli programavimo darbai, skirti programavimo įgūdžiams pagilinti bei darbo su kompiuteriu praktikai.
Knyga turėtų būti naudinga vyresniųjų klasių moksleiviams, informatikos mokytojams ir pirmųjų kursų pedagoginės krypties studentams.

6) http://ims.mii.lt/fps/download/0.6.4/
Iš čia galima parsisiųsti Free Pascal.

7) http://62.80.232.136/informatika/uzduotys/vadoveliai/fps.pdf
Lokalizuota Free Pascal programavimo sistema

8) http://www.taoyue.com/tutorials/pascal/contents.html
Paskalis. Mokomoji medžiaga. Anglų kalba

9) http://www.geocities.com/SiliconValley/Bay/5707/pascal.html
Paskalis. Mokomoji medžiaga. Anglų kalba

10) http://freepascal.org/download.var
Iš čia galima parsisiųsti naujausią Free Pascal versija (Nelokalizuotą)

11) Užklausa internete:
Informacinių technologijų valstybinis brandos egzaminas praktinės užduotys 2006 m.
Pateiktis, kur gerai pristatyti Free Pascal diegimas ir aplinka

12) http://www.levenez.com/lang/
Išsami informacija apie įvairias programavimo kalbas