Jonas Blonskis
Vytautas Bukšnaitis
Renata Burbaitė

# Šiuolaikiškas žvilgsnis į programavimo pagrindus C++

Pasirenkamasis informacinių technologijų kursas IX–X klasėms

Jonas Blonskis, Vytautas Bukšnaitis, Renata Burbaitė

# Šiuolaikiškas žvilgsnis į programavimo pagrindus C++

Informacinių technologijų pasirenkamasis kursas
IX–X klasėms

TEV

## 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, kurioms spręsti naudotojas turi atrasti sprendimo būdą ir parašyti programą. Šiuolaikinės taikomosios programos (pvz., skaičiuoklė, rašyklė ir kt.) turi savo programavimo priemones, kurios leidžia išplėsti jų galimybes, pritaikyti šias programas konkrečiai veiklos sričiai ar užduočiai spręsti. Bet kuriam kompiuterio naudotojui yra labai svarbios programavimo žinios, įgūdžiai ir gebėjimas kurti nesudėtingas programas.

Išmokti programuoti galima tik pačiam kuriant programas. Todėl mokymąsi siūlome pradėti nuo praktikos darbų. Knygos skyriuje *Praktikos darbai* rasite vienuolika praktikos darbų. Jie visi pateikiami naudojantis *C++* programavimo kalba. Kiekviename darbe aprašoma, kaip žingsnis po žingsnio kuriama programa. Parašius nors ir keletą programos eilučių, labai svarbu patikrinti programos darbą. Tik įsitikinus, kad programa dirba gerai, galima ją rašyti toliau. Vartojamos sąvokos ir terminai glaustai paaiškinti knygos skyriuje *Pagrindinės struktūrinio programavimo sąvokos*. Jų mokytis atmintinai nereikia, tačiau, norint sėkmingai dirbti, būtina suvokti sąvokų prasmę.

Kiekvieno praktikos darbo pradžioje rasite sąrašą žinių ir gebėjimų, kuriuos įgysite atlikdami praktikos darbą. Čia pateikiamos ir nuorodos į skyrius *C++ kalbos žinynas* bei *Algoritmų žinynas*. *C++ kalbos žinyne* pristatomos *C++* programavimo kalbos pagrindinės priemonės ir konstrukcijos. *Algoritmų žinyne* aprašomi klasikiniai algoritmai, naudojami įvairaus tipo praktikos užduotims spręsti. Šių skyrių informacija bus naudinga tiems, kurie norės pasitikslinti ar pagilinti žinias atlikdami konkretų praktikos darbą.

Kiekvienas praktikos darbo žingsnis – tai tam tikra veikiančios programos versija. Todėl mokiniai gali dirbti individualiai, prireikus – pasikonsultuoti su mokytoju. Kiekvieno atlikto žingsnio rezultatas – veikianti, bet dar nebaigta programa. Darbą galima tęsti kitą pamoką arba namuose.

Praktikos darbų pabaigoje gausu klausimų ir užduočių, kurios padės kiekvienam įsivertinti žinias ir įgūdžius. Tai neprivaloma, tačiau siūlome jas pasinagrinėti ir atlikti. Jei kurios nors užduotys yra per sunkios, nepraleiskite jų, pasistenkite jas įveikti išnagrinėję teorinę medžiagą, konsultuodamiesi su mokytoju ar klasės draugais.

Pirmieji šeši praktikos darbai aprašomi gana detaliai. Jie skirti programavimo pradmenų mokymui, todėl yra privalomi. Kitų praktikos darbų tikslas – susisteminti įgytas žinias ir patobulinti programavimo įgūdžius. Šių praktikos darbų aprašymai nėra tokie išsamūs, paliekama daugiau laisvės saviraiškai, todėl nebūtina jų visų atlikti.

Mokiniams, jau turintiems pradinių programavimo įgūdžių, šalia praktikos darbų siūlome individualiai atlikti užduotis, pateiktas skyriuje *Savarankiško darbo užduotys*. Šiems mokiniams taip pat bus naudingi darbų aprašymų papildymai, pažymėti žodžiu *Smalsiems*.

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

Sėkmės!

*Knygos autoriai*

# 1 PAGRINDINĖS STRUKTŪRINIO PROGRAMAVIMO SĄVOKOS

## Algoritmas, jo vykdymas, savybės

Kasdienėje veikloje kiekvienas susiduriame su įvairiomis taisyklėmis, nurodymais, pavyzdžiui: naudojimosi įvairiais įrenginiais ar baldų surinkimo instrukcijomis, patiekalų receptais ir pan.

Yra ir kitokių nurodymų. Pavyzdžiui, 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.“

Panašiai taisyklės sudaromos ir matematikos, fizikos, chemijos uždaviniams spręsti. Naudodamiesi jomis, nesunkiai išsprendžiame vienokio ar kitokio tipo uždavinius. Mokydamiesi gimtąją ir užsienio kalbas, išmokstame pagrindines taisykles ir jas taikydami sėkmingai įveikiame gramatikos subtilybes.

Iš pateiktų pavyzdžių matyti, kad taisyklės yra skirtingo pobūdžio, tačiau turi ir bendrų bruožų:

- jas galima aprašyti atskirais aiškiais nurodymais, ką reikia daryti;
- yra pradiniai duomenys (pavyzdžiui: patiekalui pagaminti reikalingi produktai, spintos sudedamosios dalys ir kt.);
- gaunamas tam tikras rezultatas (pagaminamas patiekalas, sukonstruojama spinta ir kt.).

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

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

***Algoritmù*** vadinami aiškūs vienareikšmiai nurodymai (sakiniai), kaip turint tam tikrus pradinius duomenis galima 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 matematinio pobūdžio algoritmai. Pateikiame pagrindines jų sąvokas.

***Pradìniai dúomenys*** – tai iš anksto žinomos reikšmės (paprasčiausiu atveju – skaičiai), būtinos veiksmams atlikti. Pavyzdžiui, norint apskaičiuoti stačiakampio plotą, būtina žinoti jo ilgį ir plotį.

***Rezultãtai*** – tai reikšmės, gautos atlikus visus skaičiavimus.

***Tárpiniai rezultãtai*** – tai apskaičiuotos reikšmės, kurios naudojamos tolesniems veiksmams atlikti. Tarpiniai rezultatai padeda programuotojui pasitikrinti, ar parašyta visa programa, ar tik jos dalis, ar programos dalys veikia gerai.

Algoritmu aprašomi veiksmai yra skirti ***výkdytojui***.

Kiekvienam algoritmui būdingos tokios savybės:

- ***Diskretùmas***. Algoritmas suskaidomas į baigtinę žingsnių seką. Tik atlikus vieno žingsnio veiksmus galima pereiti prie kito žingsnio.
- ***Aiškùmas***. Visus algoritmu aprašomus veiksmus bet kuris vykdytojas turi suprasti vienareikšmiškai. Šioje knygoje pateikiami algoritmai skirti kompiuteriui – *nemąstančiam vykdytojui*. Kad suprastume, 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, pirmiausia 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, t. y. nemąstydamas ir neanalizuodamas, vykdė tėvo nurodymus.
- ***Rezultatyvùmas***. Atlikus baigtinį skaičių algoritmo veiksmų, gaunamas rezultatas. Vienas iš galimų rezultatų – uždavinys sprendinių neturi.
- ***Baigtùmas***. Rezultatas gaunamas įvykdžius baigtinį skaičių algoritmo veiksmų.
- ***Universalùmas***. Naudojant tą patį algoritmą, sprendžiami visi to tipo uždaviniai, t. y. kiekvienam pradinių duomenų rinkiniui gaunamas teisingas rezultatas.

Yra daug uždavinių, kuriems spręsti nėra tikslių algoritmų. Pavyzdžiui, reikia mašinomis gabenti daug įvairaus dydžio tuščių dėžių, kurias galima dėti vienas į kitas. Mašinų su kroviniu skaičius turi būti kuo mažesnis. Net žmogus, galintis intuityviai spręsti tokį uždavinį, ne iš karto gaus geriausią rezultatą. Jeigu dėžių daug, o į mašinas telpa nevienodas jų skaičius, tuomet neįmanoma perrinkti visų galimų sprendimo variantų ir tenka pasirinkti vieną kurį nors geresnį rezultatą. Tokio tipo uždaviniams spręsti kuriami algoritmai vadinami ***eurìstiniais***.

Algoritmai gali būti pateikiami skirtingais būdais:

- Užrašomi ***žõdžiais***. Šis būdas naudojamas, kai norima labai aiškiai nurodyti atliekamus veiksmus. Užrašomos komandos gali būti numeruojamos arba veiksmai aprašomi kaip pasakojimas.
- Vaizduojami ***grãfiškai*** – dažniausiai ***simbòlinėmis*** (***blòkinėmis***) ***schèmomis*** arba ***struktūrogrãmomis***.

  Vartojami grafiniai simboliai apibrėžia tam tikro tipo veiksmą. Simbolinėse schemose grafinius simbolius jungiančios linijos rodo, kokia tvarka tie veiksmai atliekami. 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 veiksmų vykdymo tvarka nusakoma grafiniais simboliais.
- Užrašomi ***pseudokodù***. Vartojami žodžiai, artimi natūraliai kalbai. Pseudokodu patogu užrašyti algoritmus, kai norima trumpiau ir suprantamiau atskleisti jų esmę – vadovėliuose, straipsniuose.

**Pavyzdys**

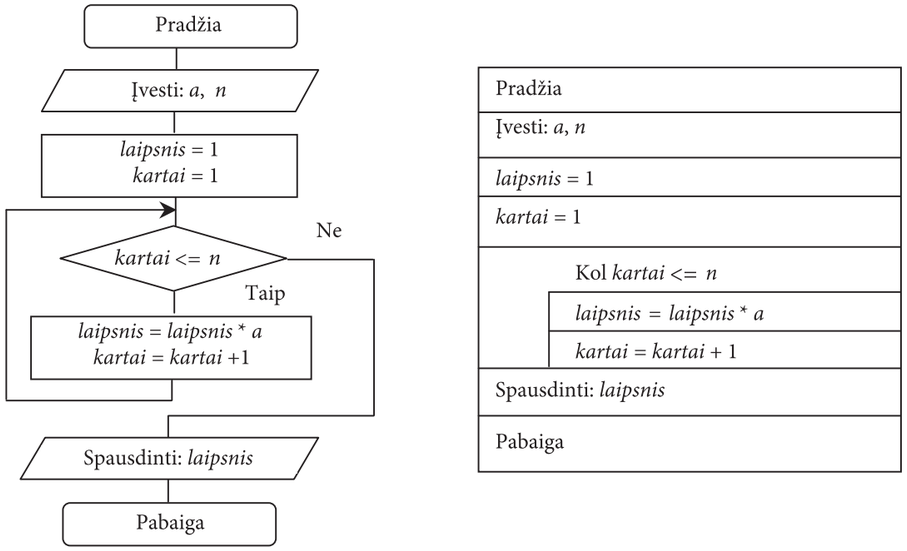
Kompiuteryje skaičių *a* keliant *n*-uoju laipsniu skaičiuojama sandauga (tarkime, $2^3 = 2 * 2 * 2$).

Skaičiaus *a* kėlimo *n*-uoju laipsniu algoritmas gali būti užrašytas:

- Žodžiu:
  1. Sužinoti *a* ir *n* reikšmes.
  2. Skaičiuoti *kartai* = 1.
  3. Skaičiuoti *laipsnis* = 1.
  4. Jeigu *kartai* <= *n*, tuomet vykdyti 5 žingsnį, priešingu atveju – pereiti prie 8 žingsnio.
  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*.

- Grafiškai (simboline schema ir struktūrograma):

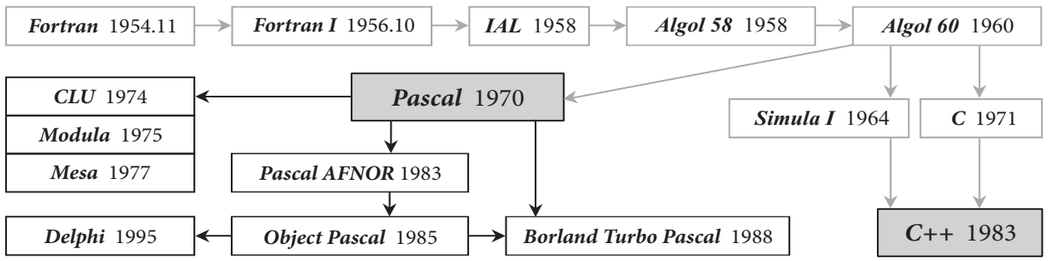



| Pradžia |
|---|
| Įvesti: *a*, *n* |
| *laipsnis* = 1 |
| *kartai* = 1 |
| Kol *kartai* <= *n* |
| *laipsnis* = *laipsnis* * *a* |
| *kartai* = *kartai* + 1 |
| Spausdinti: *laipsnis* |
| Pabaiga |

## Programa, programavimo kalba, struktūrinis programavimas

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

***Programãvimo kalbà***, kaip ir šnekamoji, turi savo sintaksę ir semantiką. Programavimo kalbos kiekvienos konstrukcijos, kiekvieno žodžio, kiekvieno sakinio prasmė vienareikšmė. Programavimo kalbų yra labai daug. Nuo 1954 m., kai buvo sukurta pirmoji programavimo kalba, iki dabar yra suskaičiuojama per 2500 kalbų. Schemoje pateikta šiuo metu populiarių mokymuisi skirtų *Pascal* ir *C++* programavimo kalbų trumpa atsiradimo istorijos schema. Daugiau informacijos apie programavimo kalbas ir jų istoriją galite rasti internete (*www.levenez.com/lang*).




Šioje knygoje programoms kurti vartosime C++ programavimo kalbą. Tačiau norime pabrėžti, kad pati programavimo kalba yra tik priemonė programavimo įgūdžiams įgyti, bet ne mokymosi tikslas. Stengiamės naudoti tas kalbos konstrukcijas, kurios yra bendros ar analogiškos kitų programavimo kalbų konstrukcijoms.

***Programãvimas*** – tai procesas, kuris apima šiuos etapus:

- užduoties analizės;
- užduoties skaidymo į dalis;
- sprendimo metodų parinkimo ir sukūrimo;
- kintamųjų parinkimo;
- algoritmų sudarymo;
- programos teksto rašymo, derinimo ir testavimo.



Yra keletas programavimo technologijų. Viena jų – ***struktūrinis programãvimas*** (kartais vadinamas ***procedūriniu programãvimu***). Struktūrinio programavimo pradžia yra laikoma XX a. 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.

***Programãvimo aplinkà*** – tai aparatinės ir programinės įrangos priemonių visuma, skirta naujoms programinėms priemonėms kurti. Paprasčiausios programavimo aplinkos, pavyzdžiui, *CodeBlocks*, turi rašyklę programų tekstams kurti ir taisyti, kompiliatorių, priemones programai derinti ir vykdyti.

***Kompiliãtorius*** – tai programa, kuri verčia parašytą programos tekstą į kompiuteriui suprantamą kalbą. Jeigu randa rašybos (sintaksės) klaidų, tai vertimą nutraukia ir nurodo rastas klaidas.

## Programos struktūra

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

Vadovėlyje pateiktos programos parašytos C++ programavimo kalba, laikantis struktūrinio programavimo technologijos reikalavimų. Programos tekstas susideda iš dviejų arba trijų dalių:

- naudojamų priemonių (bibliotekų, funkcijų prototipų, konstantų, kintamųjų, naujų tipų) aprašymo;
- pagrindinės funkcijos **`main()`** kintamųjų ir veiksmų sakinių;
- naudojamų funkcijų, jeigu jos yra, tekstų.

| priemonių aprašymas |
|---|
| funkcijų prototipai |
| `int main()`<br>`{`<br>kintamieji<br>veiksmai<br>`return 0;`<br>`}` |
| funkcijų tekstai |

## Programavimo stilius ir kultūra

Kaip ir rašytojams, programų autoriams būdingas individualus ***programãvimo stìlius***. Kiekvieno programuotojo ar programuotojų grupės požiūris į programavimą ir programavimo kalbas yra savitas. Visi jie skirtingai naudoja programavimo priemones. Šios knygos autoriai laikosi tokių esminių programavimo stiliaus principų:

- kintamųjų aprašai grupuojami pagal paskirtį ir aprašomi pagrindinės funkcijos **`main()`** pradžioje;
- paprastų kintamųjų (skirtų skaičiams, simboliams ar loginėms reikšmėms atmintyje laikyti) vardai pradedami mažąja raide;
- jei programoje yra kuriama funkcija, ji skirta vienam griežtai apibrėžtam veiksmui atlikti (pavyzdžiui: duomenims įvesti, vidurkiui apskaičiuoti, rezultatui išvesti).

Programuojant sudėtingą užduotį, labai dažnai jos sprendimas padalijamas į atskiras dalis, kurias realizuoja skirtingi programuotojai. Kiekvienas iš jų turi parašyti savo dalį taip, kad ją lengvai suprastų kiti. Svarbu, kad visas užduoties dalis būtų galima susieti, pataisyti arba panaudoti kitiems uždaviniams spręsti. Todėl programos tekstas turi būti aiškus, vaizdus, lengvai skaitomas ir suprantamas bet kuriam naudotojui: su įtraukomis, tarpais ir komentarais. Be to, programos tekstas turi atitikti loginę algoritmo struktūrą ir veiksmų hierarchiją.

Toliau aptarsime pagrindines programos kūrimo ir jos teksto pateikimo taisykles.

Rašant programą, reikėtų laikytis tokių ***programãvimo kultūros*** taisyklių:

- algoritmas turi geriausiai tikti uždaviniui spręsti, būti aiškus, trumpas ir logiškai pagrįstas;
- kintamųjų, konstantų, tipų, funkcijų vardai turi atitikti aprašomų objektų prasmę, tačiau neturi būti ilgi;
- programa turi būti rašoma su komentarais.

Esminės programos teksto dėstymo taisyklės:

- programos dalims išskirti paliekamos tuščios eilutės arba rašomi komentarai, sudaryti tik iš minuso ar kitokių ženklų;
- pavaldumui išryškinti daromos įtraukos (pradedama rašyti toliau nuo krašto) ciklo ir sąlyginiuose sakiniuose;
- skaitomumui pagerinti sąlyginiuose ir sudėtiniuose sakiniuose vertikaliai lygiuojami šie žodžiai: **`if`**, **`else`** ir **`{`**, **`}`**.
- programos tekstas rašomas iš abiejų pusių tarpais atskiriant šiuos ženklus:
  `>`, `<`, `!=`, `>=`, `<=`, `=`, `+`, `-`, `*`, `/`, `+=`, `-=`, `*=`, `/=`, `>>`, `<<`.

Tekstą reikėtų rašyti taip, kad atskiros dalys pagal prasmę būtų nesunkiai atpažįstamos ir suvokiamos. Jas reikėtų išdėstyti lape lyg kokį piešinį.

Pavyzdžiui, tekstą

```
...
double pirmas,antras,trecias;
cin>>pirmas>>antras>>trecias;
cout<<pirmas<<antras<<trecias;
...
```

sunku skaityti, todėl patartume jį rašyti taip:

```
...
double pirmas, antras, trecias;
cin >> pirmas >> antras >> trecias;
cout << pirmas << antras << trecias;
...
```

Programos tekstą padeda suprasti komentarai. Jie skirti programuotojui ir visai neturi įtakos programos vykdytojui (kompiuteriui). Komentarai turi būti trumpi, taikliai papildyti programą ir jos neužgožti. Rašant komentarus, siūloma:

- programos pradžioje po antrašte (arba prieš ją) nurodyti programos autorių, paskirtį, paskutinio taisymo datą, versijos numerį, užduoties sprendimo būdą, programos apribojimus;
- kintamųjų aprašuose nurodyti jų paskirtį;
- prieš sąlyginius sakinius, ciklus, funkcijas, sakinių blokus **`{`** ir **`}`** nurodyti jų paskirtį;
- lygiuoti juos vertikaliai.

Pavyzdžiui, programos fragmentą

```
double ugis1, ugis2; // mokinių ūgis
int metai1, metai2; // mokinių gimimo metai
int amzius1, amzius2; // mokinių amžius
double svoris1, svoris2; // mokinių svoris
```

sunku skaityti. Siūlome kintamuosius grupuoti pagal prasmę, juos ir komentarus vienodai lygiuoti:

```
// Pirmo mokinio duomenys:
double ugis1;       // ūgis
int    metai1;      // gimimo metai
int    amzius1;     // amžius
double svoris1;     // svoris
// Antro mokinio duomenys:
double ugis2;       // ūgis
int    metai2;      // gimimo metai
int    amzius2;     // amžius
double svoris2;     // svoris
```

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



## Programos ir jos naudotojo dialogas

Tarp naudotojo ir kompiuterio dažnai vykdomas ***dialògas***. Įvedant duomenis klaviatūra, būtina išvesti į ekraną pranešimą, ką naudotojas turi įvesti, ir, jeigu reikia, nurodyti įvedimo formą bei eilės tvarką. Pavyzdžiui, programos fragmentas

```
...
cout << "Apskritimo spindulys: ";
cin >> R;
...
```

įpareigoja kompiuterį laukti, kol įvesime vieną skaičių. Pranešime dar vertėtų nurodyti, kokį skaičių įvesti (realųjį ar sveikąjį).

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

```
...
cout << "a = ";
cin >> a;
...

...
cout << "Knygos kaina: ";
cin >> a;
...
```

Iš pirmojo dialogo neaišku, kas yra `a`, iš antrojo – aišku, kad reikia įvesti prekės kainą, nebūtina žinoti, kad kaina programoje turi vardą `a`.

## Pavyzdinis programos kūrimo planas

1. ***Pradìnių duomenų ir būsimų rezultãtų anãlizė***. Išsiaiškinama, kiek yra pradinių duomenų, kiek bus rezultatų, kokie jų tipai, kokia tvarka juos pateikti.
2. ***Ùždavinio sprendìmo idéja*** – tai mintis (sumanymas), kaip spręsti uždavinį. Idėjos teisingumu galima įsitikinti modeliuojant programos veiksmus su įvairiais duomenimis. Programą reikėtų pradėti rašyti tik įsitikinus, kad sugalvotas sprendimo būdas yra teisingas.
3. ***Kintamųjų parinkìmas pradìniams duomenìms ir rezultãtams laikýti***. Nuo jų parinkimo priklauso programos apimtis, struktūra, algoritmas ir jo įgyvendinimas.
4. ***Algorìtmo sukūrìmas*** – tai uždavinio sprendimo idėjos įgyvendinimas. Dažniausiai taikomi žinomų matematinių uždavinių sprendimo būdai naudojant pasirinktus kintamuosius.
5. ***Prográmos rãšymas*** – tai sukurto algoritmo pateikimas pasirinkta programavimo kalba. Programa rašoma laikantis struktūrinio programavimo principų (nedidelėmis dalimis). Patariama įsitikinti, kad kiekviena programos dalis dirba teisingai. Tai leidžia programą rašyti žingsnis po žingsnio. Kai derinamo teksto dalys nedidelės, galutinė programa gaunama lengviau ir greičiau.
6. ***Testãvimas***. Parašyta programa testuojama. Taip įsitikinama, kad norimas rezultatas pasiekiamas esant visiems galimiems teisingiems pradinių duomenų rinkiniams. Programuotojas turėtų sukurti kuo įvairesnių pradinių duomenų rinkinių, kad būtų galima patikrinti įprastas situacijas ir ribinius atvejus, sprendimo efektyvumą naudojamos atminties ir vykdymo laiko požiūriu.

# 2 PRAKTIKOS DARBAI

## 2.1. Pažintis su *CodeBlocks* aplinka

Atlikdami šį darbą, susipažinsite su *CodeBlocks* aplinka:

- sukursite darbo katalogą ir programos failą;
- pakeisite programos pavadinimą, programą įrašysite į darbo katalogą;
- sukompiliuosite ir įvykdysite paprasčiausią programą;
- išmoksite programą taisyti;
- išmoksite išvesti duomenis į ekraną naudodami išvesties srautą `wcout`.

| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.5. Rezultatų (duomenų) išvedimas į ekraną<br>3.13. Knygoje naudojamų įterpiamų failų sąrašas | 4.1. Tiesiniai algoritmai |

Atliekant šį praktikos darbą, svarbu ne apskaičiuoti kokį nors konkretų rezultatą ar sukurti nurodytą programą, o tiesiog pajusti darbo *CodeBlocks* aplinkoje ypatumus, įgyti patirties, kurios prireiks atliekant kitus darbus. Dirbdami galite atlikti ir daugiau veiksmų, negu čia parašyta.

Kiekviena *CodeBlocks* programa sukuria keletą failų. Todėl, prieš pradėdami darbą, sukurkite atskirą katalogą kiekvieno praktikos darbo failams laikyti.

### 1 Darbo katalogo kūrimas

- Kurioje nors laikmenoje, naudodamiesi įprastomis *Windows* sistemos priemonėmis, sukurkite bendrą katalogą, kuriame laikysite visus savo *CodeBlocks* darbus. Katalogą galite pavadinti savo pavarde, pavyzdžiui, *Pavardenis*.
- Šiame kataloge sukurkite pakatalogį *Darbas1*, skirtą pirmajam darbui.

### 2 *CodeBlocks* paleidimas

Norint pradėti dirbti, reikia, kad kompiuteryje būtų įdiegta kuri nors *CodeBlocks* versija. Visi šios knygos pavyzdžiai sukurti naudojantis *CodeBlocks 10.05* versija. Ją galima rasti internete (*http://prdownload.berlios.de/codeblocks/codeblocks-10.05mingw-setup.exe*). Detaliau, kaip įdiegti ir paruošti aplinką darbui, aprašyta 5 skyriuje *Aplinka CodeBlocks*.

*CodeBlocks* galima paleisti:

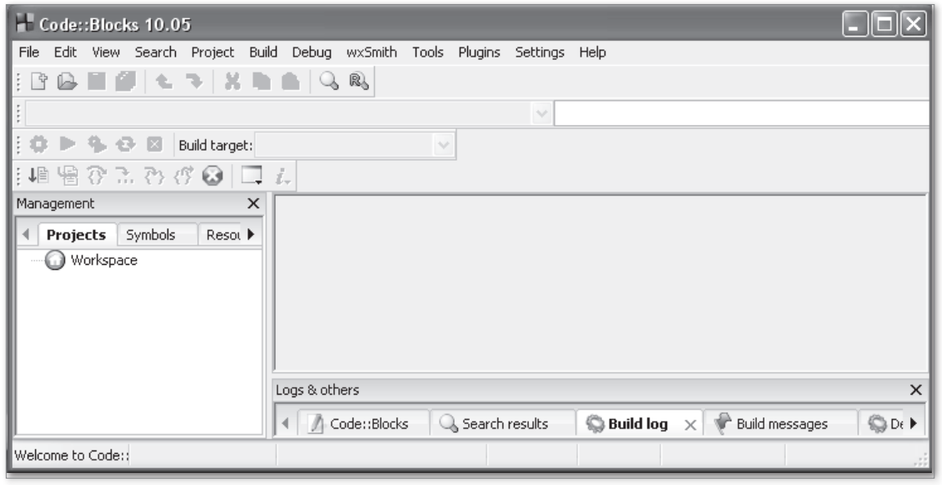
- pasirinkus pradžios meniu komandas: *Visos programos* → *CodeBlocks*:


- dukart spragtelėjus darbalaukyje šaukinį 

.

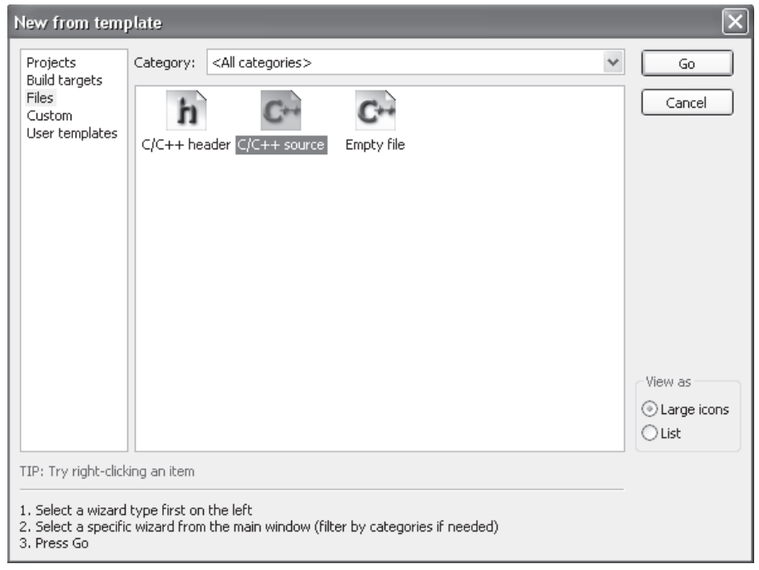


Sėkmingai įvykdę nurodymus, pateksite į *CodeBlocks* langą, kuris valdomas (padidinamas, sumažinamas, pernešamas ar užveriamas) įprastomis operacinės sistemos *Windows* priemonėmis.

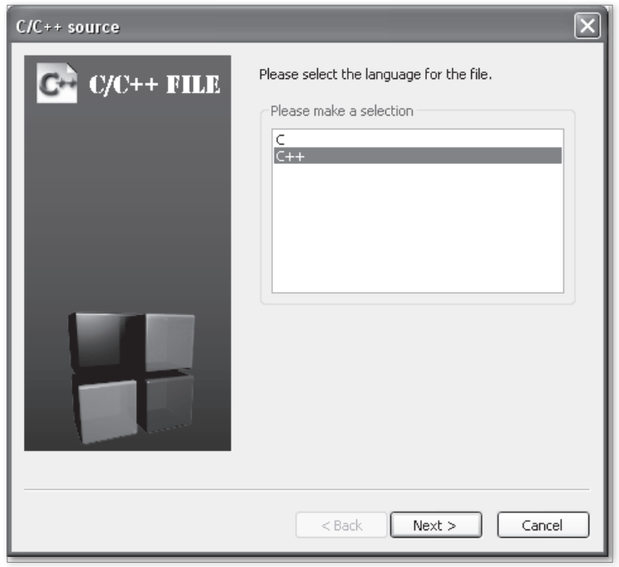



### 3 Programos failo kūrimas

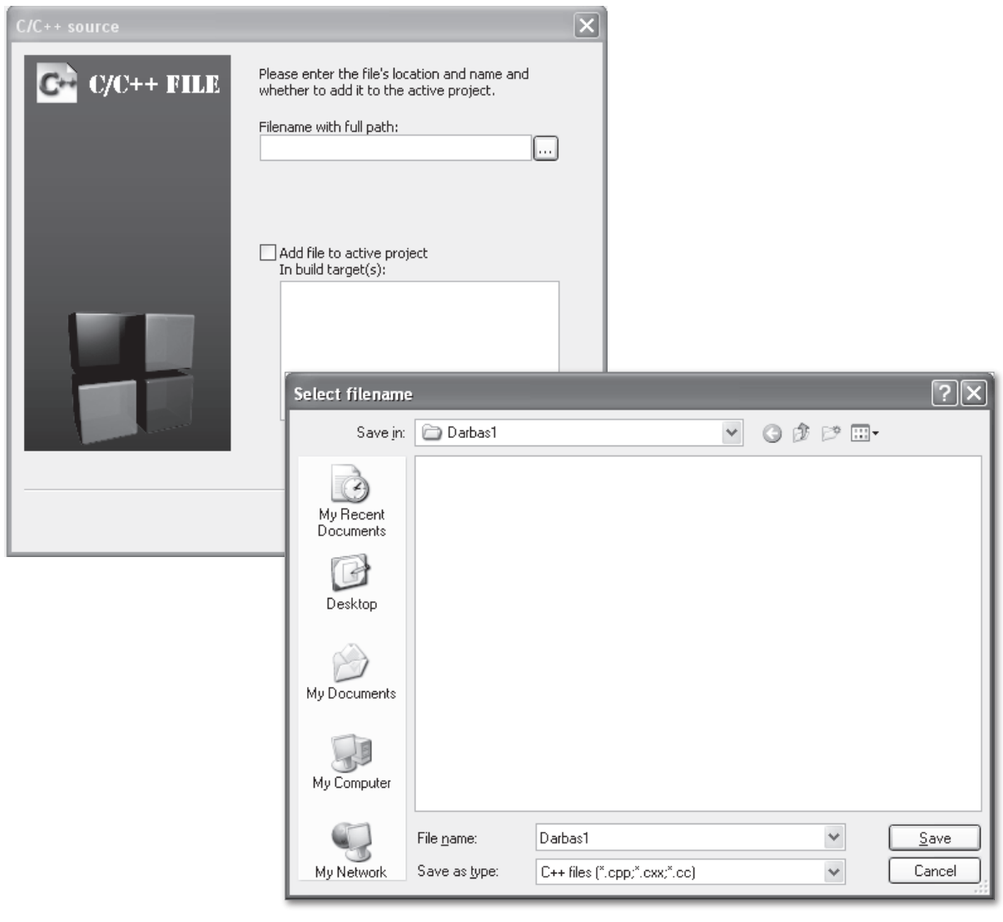
- Pagrindiniame meniu pasirinkite komandas *File* → *New* →*File...* ir atsivėrusiame dialogo lange pažymėkite *C/C++ source*. Patvirtinkite pasirinkimą spragtelėdami mygtuką *Go*.



➢ Atsivėrusiame dialogo lange pasirinkite programavimo kalbą *C++* ir spragtelėkite mygtuką *Next*.

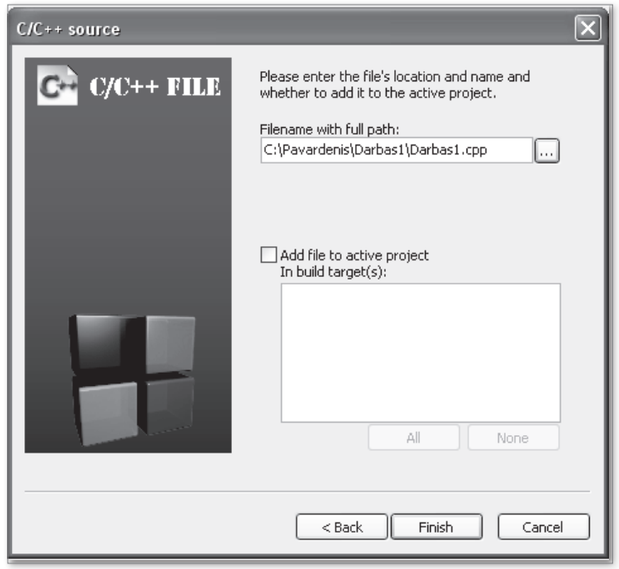



➢ Pasirinkite, kur įrašyti failą: spragtelėję mygtuką ..., atsivėrusiame dialogo lange nurodykite katalogą *Darbas1*, o laukelyje *File name* įrašykite failo, kuriame bus sukurta programa, vardą *Darbas1*. Spragtelėkite mygtuką *Save*.






Dialogo lange matysite visą programos failo adresą.




Spragtelėjus mygtuką *Finish*, aplinkoje *CodeBlocks* atsivers kuriamos programos šablonas, kurį galėsite redaguoti *CodeBlocks* rengykle.

Naują programos failą taip pat galima sukurti pasirinkus priemonių juostos mygtuką arba sparčiaisiais klavišais *Ctrl* + *Shift* + *N*.

## Programos šablonas

Panašiems tam tikro tipo dokumentams (pvz., C++ programoms) rengti yra patogu naudotis šablonu (pavyzdžiu). Jame įrašoma informacija, kuri to tipo dokumentuose visada yra tokia pati. Be to, jame gali būti nurodoma, kokia informacija dokumente turi būti keičiama. Skyriuje *Aplinka CodeBlocks* paaiškinta, kaip galima sukurti C++ programų šabloną.

Mūsų sukurtame C++ programų šablone įrašyti sakiniai, reikalingi paprasčiausiai veikiančiai programai. Panagrinėkime šį šabloną.

*Darbas1.cpp

```
1   // Vieta programos vardui įrašyti
2   #include <fcntl.h>
3   #include <io.h>
4   #include <iostream>
5   using namespace std;
6   int main ()
7   {
8     _setmode (_fileno(stdout), _O_U16TEXT);
9     wcout << L"Labas" << endl;
10    return 0;
11  }
```

➢ Pirmoje eilutėje įrašytas komentaras, kuris neturi jokios įtakos programos darbui.

```
// Vieta programos vardui įrašyti
```

Komentaro tekstą programuotojas gali keisti, užrašydamas jo vietoje programos vardą, trumpai apibūdindamas kuriamos programos paskirtį. Jeigu reikalinga, šį komentarą galima praplėsti iki keleto eilučių.

Programos pradžioje surašytos instrukcijos *parengiamajai doroklei* (angl. preprocessor), kurios žymimos simboliu `#`. Įterpimo instrukcijomis `include` nurodoma, kokių failų tekstai turi būti įterpti instrukcijų pažymėtose

vietose pirminio apdorojimo metu. Įterpiamų failų vardai rašomi tarp simbolių < >. Šablone įrašyti pirmajame darbe naudojami failai.

| Įterpiamas failas | Paaiškinimas |
|---|---|
| `iostream` | Duomenų įvedimo klaviatūra ir išvedimo į ekraną priemonės |
| `fcntl.h` | Priemonės lietuviškiems rašmenims išvesti į ekraną |
| `io.h` | Priemonės lietuviškiems rašmenims išvesti į ekraną |

- Sakinys
  ```
  using namespace std;
  ```
  rašomas visada, jei programoje įterpiamas bent vienas antraštinis failas (pvz., `iostream`).
- Toliau rašoma programos pagrindinės funkcijos antraštė
  ```
  int main ()
  ```
- Pagrindinės funkcijos kamienas (veiksmų sritis) pradedamas ženklu `{`, baigiamas – ženklu `}`.
- Sakinys
  ```
  _setmode (_fileno(stdout), _O_U16TEXT);
  ```
  reikalingas tam, kad ekrane būtų teisingai rodomi lietuviški rašmenys su diakritiniais ženklais.
- Sakinys
  ```
  wcout << L"Labas" << endl;
  ```
  į ekraną išveda žodį Labas.

  Tekstui su lietuviškais rašmenimis išvesti į ekraną reikia naudoti išvedimo srauto modifikaciją `wcout`. Prieš simbolių eilutes, kuriose yra lietuviškų rašmenų, reikia parašyti didžiąją raidę `L`. Kitais atvejais raidės `L` rašyti nebūtina.
- Sakinys
  ```
  return 0;
  ```
  nurodo programai baigti funkcijos `main()` darbą.

## 5 Programos vardo pakeitimas. Programos failo įrašymas į katalogą *Darbas1*

➢ Pirmojoje eilutėje įrašykite programos vardą `Darbas1`.

```
// Darbas1
#include <fcntl.h>
#include <io.h>
```

➢ Toliau pasirinkite vieną iš išvardytų pagrindinio meniu komandų:
- *File → Save File;*
- *File → Save File As...;*
- *File → Save all Files;*
- *File → Save everything.*

Programą įrašyti taip pat galima priemonių juostos mygtukais arba sparčiaisiais klavišais *Ctrl* + *S*, *Ctrl* + *Shift* + *S* arba *Alt* + *Shift* + *S*.

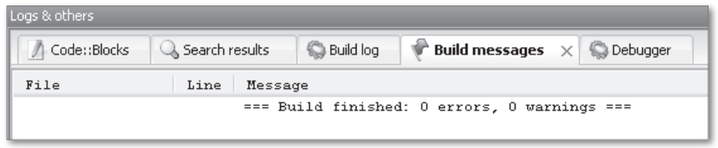
Kurdami programą, kartkartėmis įrašykite ją į laikmeną. Tai galite atlikti vienu iš šių būdų:
- pagrindinio meniu komandomis: *File → Save File*;
- programos priemonių juostos mygtuku ;
- sparčiaisiais klavišais *Ctrl* + *S*.



## 6 Programos kompiliavimas

Tai galima atlikti vienu iš trijų būdų:
- pagrindinio meniu komandomis *Build → Build* arba *Build → Compile current file*;
- programos priemonių juostos mygtuku ;
- sparčiaisiais klavišais *Ctrl* + *F9* arba *Ctrl* + *Shift* + *F9*.

Jeigu programoje nebuvo sintaksės klaidų, apatinėje darbo lango dalyje bus rodomas pranešimas apie sėkmingai sukompiliuotą programą.



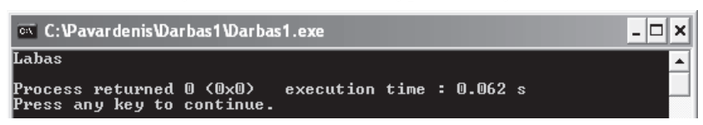

➢ Išbandykite visus tris programos kompiliavimo būdus ir pasirinkite tinkamiausią.

Tačiau, norint pamatyti programos darbo rezultatus, ją dar reikia įvykdyti.

## 7 Programos vykdymas

Tai galima atlikti vienu iš trijų būdų:
- pagrindinio meniu komandomis *Build → Run*;
- programos priemonių juostos mygtuku ;
- funkciniu klavišu *Ctrl* + *F10*.

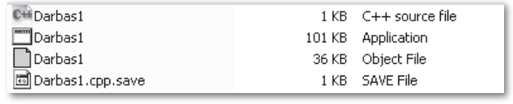
Jeigu programoje nebuvo kompiliavimo klaidų, ekrane atsivers juodos spalvos programos rezultatų langas, kuriame išvysite žodį `Labas` ir informacinį pranešimą.




Norint taisyti programą, pirmiausia reikia užverti rezultatų langą. Tai galima padaryti spustelėjus klaviatūros klavišą *Enter* arba spragtelėjus lango užvėrimo mygtuką .

Įgijus patirties ir norint paspartinti darbą *CodeBlocks* aplinkoje, penktą žingsnį *Programos kompiliavimas* galima praleisti. Prieš vykdant programą, kuri buvo taisyta, ji iš naujo sukompiliuojama. Tam patogu naudotis pagrindinio meniu komandomis *Build → Build and Run* arba funkciniu klavišu *F9*.

➢ Išskleiskite darbo pradžioje sukurtą katalogą *Darbas1* ir pažiūrėkite, kiek ir kokių failų sukurta. Svarbiausias failas yra *Darbas1.cpp* (*C++ source file*), nes jame yra programos tekstas. Šį failą reikia saugoti. Iš šio failo sukuriami kiti to paties pavadinimo failai *Darbas1* su skirtingais prievardžiais bei kiti pagalbiniai failai.




## 8 Teksto išvedimas į ekraną

➢ Norėdami, kad ekrane šalia žodžio `Labas` būtų užrašytas ir jūsų vardas, programos teksto eilutę
```
wcout << L"Labas" << endl;
```
pakeiskite tokia:
```
wcout << L"Labas. Mano vardas Ąžuolas!" << endl;
```

Atlikus pakeitimą, programa bus tokia:

```
// Darbas1
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  wcout << L"Labas. Mano vardas Ąžuolas!" << endl;
  return 0;
}
```

➢ Įvykdykite programą, t. y. pakartokite penktą ir šeštą žingsnius arba tik šeštą žingsnį. Ekrane turėtumėte matyti:

```
Labas. Mano vardas Ąžuolas!
```

## 9 Teksto ir ornamento išvedimas į ekraną

➢ Pakeiskite ankstesnę programą taip:

```
// Darbas1
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  wcout << L"*******************************" << endl;
  wcout << L"* Labas. Mano vardas Ąžuolas! *" << endl;
  wcout << L"*******************************" << endl;
  return 0;
}
```

➢ Įvykdykite programą. Ekrane turėtumėte matyti:

```
*******************************
* Labas. Mano vardas Ąžuolas! *
*******************************
```

Kaip pastebite, vykdant programą, kiekvienu `wcout` sakiniu, kuris baigiasi manipuliatoriumi `endl`, ekrane atskirose eilutėse išvedamas tekstas, nurodytas tarp paprastųjų kabučių. Manipuliatorius `endl` perkelia žymeklį į naują eilutę.

➢ Programoje pašalinkite `<< endl`.

```
  wcout << L"*******************************";
  wcout << L"* Labas. Mano vardas Ąžuolas! *";
  wcout << L"*******************************";
```

➢ Įvykdykite programą. Ekrane matysite:

```
******************************* Labas. Mano vardas Ąžuolas! ****************
***************
```



Pastebėjote, kad be manipuliatoriaus `endl`, užpildoma pirma eilutė (eilutėje telpa 80 ženklų), po to pradedama pildyti antroji.

➢ Pakeiskite programą taip, kad gautumėte tokį vaizdą:

```
  *   *   *   *   *   *   *   *
 *** *** *** *** *** *** *** ***
  *   *   *   *   *   *   *   *
* Labas. Mano vardas Ąžuolas! *
  *   *   *   *   *   *   *   *
 *** *** *** *** *** *** *** ***
  *   *   *   *   *   *   *   *
```

## 10 Darbo su *CodeBlocks* pabaiga

Darbą galima baigti pagrindinio meniu komandomis *File* → *Quit*, sparčiaisiais klavišais *Ctrl* + *Q* arba spragtelėjus *CodeBlocks* lango užvėrimo mygtuką.

➢ Įrašykite programą ir baikite darbą su *CodeBlocks*.

## Klausimai

1. Ką pirmiausia patartina susikurti prieš pradedant rašyti programą?
2. Kaip paleidžiama *CodeBlocks*?
3. Kaip sukuriama nauja programa?
4. Kurioje vietoje rašomas programos vardas?
5. Kokius veiksmus reikia atlikti norint įrašyti programą į laikmeną?
6. Kokiais būdais programą galima sukompiliuoti?
7. Kokiais būdais programą galima įvykdyti?
8. Kokį veiksmą atlieka manipuliatorius `endl`?

## Užduotys

1. Parašykite programą, kuri ekrane iš žvaigždučių nupieštų jūsų inicialus.
2. Parašykite programą, kuri ekrane nupieštų jūsų sugalvotą ornamentą.

## SMALSIEMS

➢ Norėdami darbo rezultatus matyti ne kompiuterio ekrane, o tekstiniame faile, turėsite:
- parengti failą įrašymui;
- įrašyti į failą norimą tekstą;
- užverti tekstinį failą.

**Nuorodos į C++ kalbos žinyną**

3.1. Kintamasis, kintamojo reikšmė
3.11. Rezultatų (duomenų) išvedimas į failą
3.13. Knygoje naudojamų įterpiamų failų sąrašas

➢ Išnagrinėkite ir parašykite programą, kuri į tekstinį failą įrašo lietuvių kalbos abėcėlės didžiąsias ir mažąsias raides su diakritiniais ženklais.

➢ Programos pradžioje įterpkite failą `fstream`, kuriame yra duomenų skaitymo iš failo ir išvedimo į failą priemonės.

```
// Darbas1
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <fstream>
using namespace std;
int main ()
{
    _setmode (_fileno(stdout), _O_U16TEXT);
    ofstream fr ("Darbas1.txt"); // failas parengiamas duomenims įrašyti
    fr << "ĄČĘĖĮŠŲŪŽ" << endl;
    fr << "ąčęėįšųūž" << endl;
    fr.close();                  // failas užveriamas
    return 0;
}
```

➢ Įvykdykite programą. Matysite juodą ekraną, nes rezultatai įrašyti į tekstinį failą *Darbas1.txt*. Jis sukurtas tame pačiame kataloge, kuriame yra ir programos failas *Darbas1.cpp*. Failą *Darbas1.txt* galima atverti meniu komandomis *File* → *Open* arba sparčiaisiais klavišais *Ctrl* + *O*.

Failas *Darbas1.txt*

```
ĄČĘĖĮŠŲŪŽ
ąčęėįšųūž
```

➢ Savo sukurtą programą papildykite, kad tekstas ir ornamentas, kurie buvo išvedami į ekraną, būtų išvedami į tekstinį failą.

Failas *Darbas1.txt*

```
  *   *   *   *   *   *   *   *
 *** *** *** *** *** *** *** ***
  *   *   *   *   *   *   *   *
* Labas. Mano vardas Ąžuolas! *
  *   *   *   *   *   *   *   *
 *** *** *** *** *** *** *** ***
  *   *   *   *   *   *   *   *
```



# 2.2. Kambario remonto išlaidos

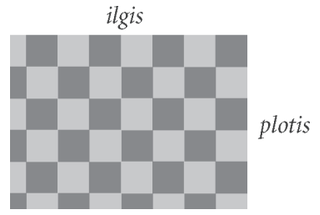
Atlikdami šį darbą, išsiaiškinsite, kaip kuriama paprastus skaičiavimus atliekanti programa:

- išmoksite aprašyti sveikojo ir realiojo tipo kintamuosius;
- išsiaiškinsite, kaip iš įvedimo srauto `cin` perskaitomos ir įsimenamos kintamųjų reikšmės;
- suprasite, kaip užrašomi ir atliekami įvairūs skaičiavimai;
- pritaikysite išvedimo srautą `wcout` rezultatams išvesti į ekraną.

| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.1. Kintamasis, kintamojo reikšmė<br>3.2. Priskyrimo sakinys<br>3.4. Duomenų įvedimas klaviatūra<br>3.5. Rezultatų (duomenų) išvedimas į ekraną<br>3.13. Knygoje naudojamų įterpiamų failų sąrašas | 4.1. Tiesiniai algoritmai |

## Užduotis

Žinomi kambario matmenys (metrais) – *ilgis* ir *plotis*. Abu dydžiai yra sveikieji skaičiai. Reikia apskaičiuoti, kokią pinigų sumą *psuma* reikės mokėti už plyteles, skirtas kambario grindims iškloti, jei plytelių vieno kvadratinio metro kaina yra *m2kaina* litų. Plytelių reikia pirkti 5 proc. daugiau galimiems nuostoliams padengti.




## Algoritmas

Užduotis sprendžiama taip:

- apskaičiuojamas kambario plotas;
- apskaičiuojama pinigų suma, kurią reikės mokėti už plyteles.

### 1 Pasiruošimas

➢ Kaip ir pirmajame darbe, atlikite tokius veiksmus:

- sukurkite katalogą *Darbas2* programos failams laikyti;
- paleiskite *CodeBlocks*;
- sukurkite programos failą;
- suteikite programai vardą `Darbas2`;
- įrašykite failą į sukurtą katalogą *Darbas2* vardu *Darbas2.cpp*.

*Darbas2.cpp

```
1  // Vieta programos vardui įrašyti
2  #include <fcntl.h>
3  #include <io.h>
4  #include <iostream>
5  using namespace std;
6  int main ()
7  {
8      _setmode (_fileno(stdout), _O_U16TEXT);
9      wcout << L"Labas" << endl;
10     return 0;
11 }
```

### 2 Bandomasis programos kompiliavimas ir vykdymas

➢ Sukompiliuokite ir įvykdykite programą.

Kaip ir pirmajame darbe, programa tik pasisveikins, t. y. išves į ekraną žodį `Labas`, tačiau jokių skaičiavimų neatliks.

## 3 Kintamųjų, skirtų pradiniams duomenims atmintyje laikyti, aprašymas ir jų reikšmių įvedimas

- Programos pradžioje aprašykite sveikojo tipo `int` kintamuosius `ilgis` ir `plotis` kambario matmenims atmintyje laikyti.
- Pakeiskite sakinį
  ```
  wcout << L"Labas" << endl;
  ```
  nauju sakiniu
  ```
  wcout << L"Programa darbą pradėjo." << endl;
  ```
- Parašykite kintamojo `ilgis` reikšmės įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį (`wcout`) ir reikšmės skaitymo sakinį (`cin`).

```
// Darbas 2
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int ilgis, plotis;  // kambario matmenys
  wcout << L"Programa darbą pradėjo." << endl;
  wcout << L"Įveskite kambario ilgį: ";  cin >> ilgis;
  return 0;
}
```

- Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `5` ir spustelėkite klavišą *Enter*. Ekrane matysite:

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

- Parašykite kintamojo `plotis` reikšmės įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį (`wcout`) ir reikšmės skaitymo sakinį (`cin`). Galite kopijuoti kintamojo `ilgis` reikšmės įvedimo sakinius ir vietoj kintamojo vardo `ilgis` parašyti `plotis`.

```
// Darbas 2
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int ilgis, plotis;  // kambario matmenys
  wcout << L"Programa darbą pradėjo." << endl;
  wcout << L"Įveskite kambario ilgį: ";  cin >> ilgis;
  wcout << L"Įveskite kambario plotį: "; cin >> plotis;
  return 0;
}
```

- Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `5` ir skaičių `4` bei spustelėkite po kiekvieno iš jų klavišą *Enter*. Ekrane matysite:

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



## 4 Kintamųjų, skirtų rezultatui atmintyje laikyti, aprašymas. Rezultatų skaičiavimas ir išvedimas į ekraną

- Papildykite programą nauju sveikojo tipo kintamuoju `plotas`.
- Užrašykite ploto skaičiavimo sakinį:
  ```
  plotas = ilgis * plotis;
  ```
- Išveskite į ekraną apskaičiuotą ploto reikšmę.
- Programos pabaigoje prieš sakinį
  ```
  return 0;
  ```
  užrašykite sakinį
  ```
  wcout << L"Programa darbą baigė.";
  ```

```
// Darbas 2
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int ilgis, plotis;  // kambario matmenys
  int plotas;         // kambario grindų plotas
  wcout << L"Programa darbą pradėjo." << endl;
  wcout << L"Įveskite kambario ilgį: ";  cin >> ilgis;
  wcout << L"Įveskite kambario plotį: "; cin >> plotis;
  plotas = ilgis * plotis;
  wcout << L"Kambario grindų plotas: "  << plotas << endl;
  wcout << L"Programa darbą baigė.";
  return 0;
}
```

Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių 5 ir spustelėkite klavišą *Enter*. Po to klaviatūra įveskite skaičių 4 ir spustelėkite klavišą *Enter*. Ekrane matysite:

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

## 5 Kambario remonto išlaidų skaičiavimas

Dabar reikia apskaičiuoti, kiek kainuos plytelės, kai žinoma vieno $m^2$ kaina `m2kaina`. Žinome, kad plytelių reikia pirkti 5 proc. daugiau galimiems nuostoliams padengti. Pinigų sumą `psuma`, reikalingą kambario remontui, galima apskaičiuoti priskyrimo sakiniais:

```
psuma = (plotas * m2kaina) + (0.05 * plotas * m2kaina);
```

arba

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

- Papildykite programą tokiais sakiniais:
  - realiojo tipo (`double`) kintamųjų `m2kaina` ir `psuma` aprašymo:

```
double m2kaina;        // plytelių 1 kvadratinio metro kaina
double psuma;          // pinigų suma
```

- plytelių vieno kvadratinio metro kainos įvedimo klaviatūra:

```
wcout << L"Įveskite plytelių 1 kvadratinio metro kainą: ";
cin >> m2kaina;
```

- plytelių kainos skaičiavimo:

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

- rezultatų išvedimo į ekraną:

```
wcout << L"Pinigų suma, kurią reikia sumokėti: " << psuma << endl;
```

Kaip pastebėjote, kintamasis `psuma` turi būti realiojo tipo (`double`), nes plytelių kainą sudaro dvi dalys: sveikoji (litai) ir trupmeninė (centai).

➢ Įrašykite ir įvykdykite programą su pateiktais pradiniais duomenimis. Ekrane matysite:

```
Programa darbą pradėjo.
Įveskite kambario ilgį: 5
Įveskite kambario plotį: 4
Kambario grindų plotas: 20
Įveskite plytelių 1 kvadratinio metro kainą: 45.50
Pinigų suma, kurią reikia sumokėti: 955.5
Programa darbą baigė.
```

## Klausimai

1. Koks bazinis C++ programavimo kalbos žodis yra vartojamas sveikojo tipo kintamiesiems aprašyti?
2. Koks bazinis C++ programavimo kalbos žodis yra vartojamas realiojo tipo kintamiesiems aprašyti?
3. Aprašykite du sveikojo tipo kintamuosius, kurie nurodo mokinio amžių (`metai`) ir masę (`kilogramai`).
4. Kokie du sakiniai vartojami kintamojo reikšmei įvesti? Užrašykite pavyzdį kintamojo `greitis` reikšmei įvesti.

## Užduotys

1. Parašykite programą, kuri apskaičiuotų, kiek popieriaus lapų *k* reikės norint nukopijuoti konspektą visos klasės mokiniams. Žinoma, kad klasėje yra *n* mokinių ir konspektą sudaro *m* lapų.

   *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.

   *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 gyvena šiame pasaulyje. Tarkime, kad metai turi 365 dienas.

   *Pasitikrinkite*. Įvedę $a = 16$, turėtumėte gauti: $men = 192$, $d = 5840$, $v = 140160$.

4. Parašykite programą, kuri apskaičiuotų, kiek knygų *k* vidutiniškai per metus perskaito vienas mokyklos bibliotekos lankytojas. Žinomas vidutiniškai per vieną mėnesį perskaitytų knygų skaičius *v* ir vidutiniškai per metus apsilankiusiųjų bibliotekoje skaičius *n*.

   *Pasitikrinkite*. Įvedę $v = 120$, $n = 800$, turėtumėte gauti $k = 2$.



5. Parašykite programą, kuri apskaičiuotų, kiek vidutiniškai keleivių *k* važiuoja į Vilnių viename traukinio vagone, jei žinomas traukinio keleivių skaičius *n*, keleivių, vykstančių ne į Vilnių, skaičius *m* ir vagonų skaičius *v*.

   *Pasitikrinkite*. Įvedę $n = 100$, $m = 20$ ir $v = 4$, turėtumėte gauti $k = 20$.

6. Parašykite programą, kuri apskaičiuotų stačiakampio, kurio viršutinio kairiojo taško (*x1*; *y1*) ir apatinio dešiniojo taško (*x2*; *y2*) koordinatės yra sveikieji skaičiai, plotą *s* ir perimetrą *p*. Nurodytų taškų koordinatės įvedamos klaviatūra. Stačiakampio kraštinės lygiagrečios su koordinačių ašimis.

   *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. Tarakonas yra vienas greičiausių gyvūnų. Jo greitis yra *g* kilometrų per valandą. Apskaičiuokite, kiek centimetrų *c* tarakonas nubėga per sekundę.

   *Pasitikrinkite*. Kai $g = 1.08$, turi būti spausdinama:

   ```
   c = 30 cm
   ```

8. Vienas garsus Lietuvos pramogų pasaulio atstovas per kito garsaus pramogų atstovo vestuves klaidingai informavo policiją apie užminuotą pokylio vietą. Teismas paskyrė sumokėti *k* tūkstančių litų baudą. Kaltininkas baudą sumokėjo 1 cento monetomis. Kiek kilogramų *m* monetų buvo nuvežta į banką, jei viena 1 cento moneta sveria 0,83 gramo?

   *Pasitikrinkite*. Kai $k = 15000$, turi būti spausdinama:

   ```
   m = 1245 kg
   ```

## 2.3. Grąžos atidavimas

Atlikdami šį darbą, išsiaiškinsite, kaip skaičiuojama sveikųjų skaičių ***dalmeñs sveikóji dalìs*** ir ***dalmeñs liẽkana***:

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

| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.1. Kintamasis, kintamojo reikšmė<br>3.2. Priskyrimo sakinys<br>3.4. Duomenų įvedimas klaviatūra<br>3.5. Rezultatų (duomenų) išvedimas į ekraną<br>3.9. Knygoje naudojamų ir / ar rekomenduojamų matematinių funkcijų sąrašas | 4.1. Tiesiniai algoritmai |

### Užduotis

Parduotuvėje pardavėja grąžą *g* Lt (*g* – sveikasis skaičius) pirkėjui nori atiduoti 100, 50, 20, 10 Lt nominalo banknotais ir 5, 2, 1 Lt nominalo monetomis jų nominalų mažėjimo tvarka. Reikia apskaičiuoti, kiek kokio nominalo banknotų ir monetų pardavėja turės atiduoti pirkėjui. Pavyzdžiui, jei pardavėja pirkėjui turi atiduoti *g* = 75 Lt grąžą, tai jai reikės vieno 50 Lt, vieno 20 Lt banknotų ir 5 Lt monetos.

### Algoritmas

Užduotis sprendžiama taip:

- Pirmiausia imamas didžiausio nominalo banknotas (100 Lt), grąža *g* dalijama iš 100 ir imama sveikoji dalmens dalis. Gautas rezultatas yra 100 Lt nominalo banknotų skaičius *k100*.
- Apskaičiuojama, kokia pinigų suma liko neatiduota. Grąža *g* dalijama iš 100 ir imama dalmens liekana. Tai yra nauja neatiduota grąža *g*. Ji gali būti skaičiuojama ir kitaip: *g* = *g* – *k100* * 100.
- Veiksmai kartojami su visų nominalų banknotais ir monetomis.

### 1 Pasiruošimas

- Atlikite veiksmus:
  - sukurkite katalogą *Darbas3* programos failams laikyti;
  - paleiskite *CodeBlocks*;
  - sukurkite programos failą;
  - suteikite programai vardą `Darbas3`;
  - įrašykite failą į katalogą *Darbas3* vardu *Darbas3.cpp*.

### 2 Kintamojo, skirto pradiniam duomeniui atmintyje laikyti, aprašymas ir jo reikšmės įvedimas

- Aprašykite sveikojo tipo kintamąjį `g`, kuris reiškia pirkėjo grąžą litais.
- Parašykite kintamojo `g` reikšmės įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį (`wcout`) ir reikšmės skaitymo sakinį (`cin`).



```
// Darbas3
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int g;
  wcout << L"Įveskite pirkėjo grąžą: "; cin >> g;
  return 0;
}
```

- Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `75` ir spustelėkite klavišą *Enter*. Ekrane matysite:

```
Įveskite pirkėjo grąžą: 75
```

### 3 Kintamųjų, skirtų rezultatams, t. y. kiekvieno nominalo banknotų ar monetų skaičiui, atmintyje laikyti, aprašymas. Rezultatų skaičiavimas ir išvedimas į ekraną

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

```
  k100 = g / 100;
```

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

```
  g = g % 100;
```

arba

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

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

```
// Darbas3
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int g;
  int k100, k50, k20, k10, k5, k2, k1;
  wcout << L"Įveskite pirkėjo grąžą: "; cin >> g;
  k100 = g / 100; g = g % 100;
  k50  = g / 50;  g = g % 50;
  k20  = g / 20;  g = g % 20;
  k10  = g / 10;  g = g % 10;
  k5   = g / 5;   g = g % 5;
  k2   = g / 2;   g = g % 2;
  k1   = g;
  return 0;
}
```

➢ Programos pabaigoje prieš sakinį `return 0;` parašykite rezultatų išvedimo į ekraną sakinius:

```
  wcout << L"Pardavėja grąžą atiduos taip:" << endl;
  wcout << L"------------------------------" << endl;
  wcout << L"100 Lt ------> " << k100 << endl;
  wcout << L" 50 Lt ------> " << k50 << endl;
  wcout << L" 20 Lt ------> " << k20 << endl;
  wcout << L" 10 Lt ------> " << k10 << endl;
  wcout << L"  5 Lt ------> " << k5 << endl;
  wcout << L"  2 Lt ------> " << k2 << endl;
  wcout << L"  1 Lt ------> " << k1 << endl;
  wcout << L"------------------------------" << endl;
```

➢ Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `75` ir spustelėkite klavišą *Enter*. Ekrane turėtumėte matyti:

```
Įveskite pirkėjo grąžą: 75
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. Nuo metų pradžios praėjo *d* dienų. Parašykite programą, kuri apskaičiuotų, kiek savaičių *s* praėjo nuo metų pradžios.

   *Pasitikrinkite.* Kai $d = 15$, turi būti spausdinama: `Nuo metų pradžios praėjo s = 2 savaitės.`

2. Miesto informatikos olimpiadoje dalyvavo *n* devintokų. Mokytoja nupirko *m* saldainių „Nomeda“ ($m \geq n$) ir išdalijo mokiniams po lygiai. 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.

3. Andrius septintojo gimtadienio proga gavo *n* balionų. Su draugais nusprendė balionus paleisti į dangų. Dalis pučiamų balionų *k* sprogo. Likusius balionus Andrius pasidalijo su *d* draugais po lygiai. Jeigu po dalybų dar liko balionų, 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 $a = 11$ balionų.

4. Lėktuvas pakilo iš oro uosto, 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.* Jei $a = 23$, $b = 55$, $c = 14$, tai lėktuvas leisis, kai bus $v = 0$ valandų ir $m = 9$ minutės.

5. Nubrauktas triženklio skaičiaus *x* antrasis skaitmuo. Prie likusio dviženklio skaičiaus iš kairės prirašius nubrauktąjį skaitmenį, 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

| Nuorodos į C++ kalbos žinyną |
|---|
| 3.5. Rezultatų (duomenų) išvedimas į ekraną<br>3.12. Funkcijos<br>3.13. Knygoje naudojamų įterpiamų failų sąrašas |

Norint sutrumpinti sukurtą grąžos skaičiavimo programą, reikėtų:

- kartojamus veiksmus (banknotų, monetų kiekio ir neatiduotos grąžos likučio skaičiavimo, kiekvieno nominalo banknotų ir monetų kiekio spausdinimo) įkelti į savarankišką programos dalį, t. y. į funkciją;
- iš funkcijos `main()` į funkciją kreiptis esant skirtingiems pinigų nominalams;
- programos pradžioje parašyti funkcijos prototipą;
- funkcijos tekstą parašyti programos gale. Labai patogu funkciją nuo pagrindinės funkcijos `main()` atskirti komentaru, pavyzdžiui, brūkšnelių ar žvaigždučių eilute.

➢ Parašykite funkciją `Graza`. Jos antraštėje skliaustuose nurodykite du sveikojo tipo kintamuosius: `k` – kokio nominalo banknotų ar monetų skaičius skaičiuojamas, `g` – kokia grąža dar liko neatiduota.

Aprašant kintamuosius funkcijos antraštėje, būtina nurodyti jų tipą. Kintamasis `g` rašomas su ženklu `&`, nes į funkciją `main()` turi būti grąžinama apskaičiuota jo reikšmė. Kintamasis `k` aprašomas be ženklo `&`, nes jo reikšmės iš funkcijos į funkciją `main()` grąžinti nereikia.

Funkcijos viduje parašykite priskyrimo sakinius `kx` ir `g` reikšmėms skaičiuoti ir sakinį, skirtą `k` nominalo banknotų ar monetų skaičiui išvesti.

```
// Funkcija, skaičiuojanti pirkėjo grąžą g, kai nominalas k
// Į ekraną išvedama nominalo reikšmė ir to nominalo banknotų skaičius
void Graza(int k, int & g)
{
  int kx;
  kx = g / k; g = g % k;
  wcout << setw(3) << k << L" Lt ------>" << kx << endl;
};
```

➢ Programos pradžioje įterpkite failą `iomanip`, kuriame yra duomenų išvedimo į failų srautus (ekranas, failas) priemonės.

➢ Programos pradžioje parašykite funkcijos prototipą:

```
// Darbas3
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
//-------------------------------------------
void Graza(int k, int & g);  // funkcijos prototipas
//-------------------------------------------
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
```

➢ Aprašykite funkcijos `main ()` kintamuosius:

```
  int g;
  int k100, k50, k20, k10, k5, k2, k1;
```

➢ Funkcijoje `main()` parašykite pradinių duomenų įvedimo sakinius ir du pirmuosius rezultatų išvedimo sakinius:

```
wcout << L"Įveskite pirkėjo grąžą: "; cin >> g;
wcout << L"Pardavėja grąžą atiduos taip:" << endl;
wcout << L"----------------------------" << endl;
```

➢ Nurodykite kreipinį į funkciją `Graza`:

```
Graza(100, g);
```

➢ Toliau nurodykite likusius kreipinius į funkciją `Graza`:

```
Graza(50, g);
Graza(20, g);
Graza(10, g);
Graza(5, g);
Graza(2, g);
Graza(1, g);
```

Lentelėje parodyta, kaip keičiasi kintamųjų reikšmės.

| Kreipinys į funkciją | Banknotų ar monetų nominalo reikšmė | Į funkciją perduodama neatiduotos grąžos `g` reikšmė | Į funkciją `main()` grąžinama nauja neatiduotos grąžos `g` reikšmė |
|---|---|---|---|
| `Graza(100, g);` | 100 | `g = 75` | `g = 75` |
| `Graza(50, g);` | 50 | `g = 75` | `g = 25` |
| `Graza(20, g);` | 20 | `g = 25` | `g = 5` |
| `Graza(10, g);` | 10 | `g = 5` | `g = 5` |
| `Graza(5, g);` | 5 | `g = 5` | `g = 0` |
| `Graza(2, g);` | 2 | `g = 0` | `g = 0` |
| `Graza(1, g);` | 1 | `g = 0` | `g = 0` |

Parašykite paskutinį rašymo sakinį:

```
wcout << L"----------------------------" << endl;
```



➢ Pasitikrinkite, ar teisingai sukūrėte programą:

```
// Darbas3
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
//-----------------------------------------
void Graza(int k, int & g);  // funkcijos prototipas
//-----------------------------------------
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int g;
  int k100, k50, k20, k10, k5, k2, k1;
  wcout << L"Įveskite pirkėjo grąžą: ";  cin >> g;
  wcout << L"Pardavėja grąžą atiduos taip:" << endl;
  wcout << L"----------------------------" << endl;
  Graza(100, g);
  Graza(50, g);
  Graza(20, g);
  Graza(10, g);
  Graza(5, g);
  Graza(2, g);
  Graza(1, g);
  wcout << L"----------------------------" << endl;
  return 0;
}
//-----------------------------------------
// Funkcija, skaičiuojanti pirkėjo grąžą g, kai nominalas k
// Į ekraną išvedama nominalo reikšmė ir to nominalo banknotų skaičius
void Graza(int k, int & g)
{
  int kx;
  kx = g / k; g = g % k;
  wcout << setw(3) << k << L" Lt ------>" << kx << endl;
}
//-----------------------------------------
```

➢ Įrašykite ir įvykdykite programą. Įveskite skaičių `75`. Ekrane matysite:

```
Įveskite pirkėjo grąžą: 75
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 funkciją programa tapo paprastesnė: nebereikia kartoti tų pačių sakinių po kelis kartus, daug lengviau galima rasti ir ištaisyti klaidas.

Savarankiškos programos dalys palengvina uždavinių sprendimą, leidžia išskaidyti uždavinį dalimis. Be to, atskiras dalis gali kurti ne vienas žmogus.

# 2.4. Kvadratinės lygties sprendinių skaičiavimas

Atlikdami šį darbą, išmoksite tinkamai užrašyti sąlyginį sakinį.

| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.4. Duomenų įvedimas klaviatūra<br>3.5. Rezultatų (duomenų) išvedimas į ekraną<br>3.8. Sąlyginis sakinys **`if`**<br>3.9. Knygoje naudojamų ir / ar rekomenduojamų matematinių funkcijų sąrašas | 4.1. Tiesiniai algoritmai<br>4.3. Šakotieji skaičiavimai |

## Užduotis

Reikia rasti kvadratinės lygties $ax^2 + bx + c = 0$ sprendinius; čia $a$, $b$, $c$ – sveikieji skaičiai, nelygūs nuliui.

## Algoritmas

Kvadratinės lygtys sprendžiamos taip:

- Skaičiuojamas diskriminantas $d = b^2 - 4ac$.
- Tikrinama, ar lygtis turi sprendinių:
  - 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 $x_2 = \frac{-b+\sqrt{d}}{2a}$.

### 1 Pasiruošimas

➢ Sukurkite katalogą *Darbas4*, skirtą programos failams laikyti. Paleiskite *CodeBlocks*. Sukurkite programos failą *Darbas4.cpp*, įrašykite jį į katalogą *Darbas4*. Suteikite programai vardą `Darbas4`.

### 2 Pradinių duomenų įvedimas klaviatūra

Pradiniai duomenys: `a`, `b`, `c` – kvadratinės lygties koeficientai.

➢ Programos pradžioje įterpkite failą `iomanip`, kuriame yra duomenų išvedimo į failų srautus (ekranas, failas) priemonės ir failą `cmath`, kuriame yra matematinių funkcijų rinkinys.

➢ Parašykite sakinius pradinėms kintamųjų `a`, `b`, `c` reikšmėms įvesti klaviatūra:

```
// Darbas4
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int a, b, c;          // lygties koeficientai
  double d;             // diskriminantas
  double x1, x2;        // lygties šaknys
  wcout << L"Įveskite kvadratinės lygties koeficientus a, b ir c: ";
  cin >> a >> b >> c;
  wcout << L"a = " << a << L" b = " << b << L" c = " << c << endl;
  return 0;
}
```



➢ Įrašykite programą. Ją įvykdykite, pasirinkę tokias koeficientų reikšmes: `a = 2, b = 4, c = 1`. Ekrane turėtumėte matyti:

```
Įveskite kvadratinės lygties koeficientus a, b ir c: 2 4 1
a = 2 b = 4 c = 1
```

### 3 Kvadratinės lygties sprendimas ir sprendinių pateikimas ekrane

➢ Papildykite programą sakiniais, kurie skirti diskriminantui skaičiuoti, kvadratinės lygties sprendiniams rasti ir jiems išvesti į ekraną.

```
// Skaičiavimai ir rezultatų išvedimas

d = b * b - 4 * a * c;
if (d < 0) wcout << L"Lygtis neturi realių sprendinių" << endl;
else if (d == 0) {
       x1 = -b / (2 * a);
       wcout << L"Lygtis turi vieną sprendinį: x = " << x1 << endl;
     }
     else {
       x1 = (-b - sqrt(d)) / (2 * a);
       x2 = (-b + sqrt(d)) / (2 * a);
       wcout << L"Lygtis turi du sprendinius: x1 = "
             << setw(6) << fixed << setprecision(2) << x1 << L" ir x2 = "
             << setw(6) << fixed << setprecision(2) << x2 << endl;
     }
```

➢ Įrašykite ir įvykdykite programą, kai koeficientų reikšmės yra tokios: `a = 2, b = 4, c = 1`. Ekrane turėtumėte matyti:

```
Įveskite kvadratinės lygties koeficientus a, b ir c: 2 4 1
a = 2 b = 4 c = 1
Lygtis turi du sprendinius: x1 = -1.71 ir x2 = -0.29
```

### 4 Programos darbo tikrinimas, esant įvairiems pradinių duomenų rinkiniams

➢ Išspręskite kvadratines lygtis:

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

## Klausimai

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

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

b)
```
x = 3;
if (x != 3) y = x + 3;
x = x + 2;
y = x + 2;
```

c)
```
x = 6;
if (x <= 8) {
  x = x + 2;
  y = x + 3;
}
else y = x - 3;
```

d)
```
x = 2;
if (x < 0) y = x - 3;
else {
  x = x + 2;
  y = x + 3;
}
```

e)
```
x = 1;
if (x > 0) {
  y = x - 3;
  x = x + 2;
}
else {
  x = x + 2;
  y = x + 3;
}
```

f)
```
x = 1;
if (x == 0) {
  y = x - 3;
  x = x + 2;
}
else {
  x = x + 2;
  y = x + 3;
}
```

2. Funkcijos reikšmėms skaičiuoti užrašytas sąlyginis sakinys:

a)
```
if (x < 5) y = x + 3;
  else y = x - 2;
```

b)
```
y = 9;
if (x != 5);
else y = 5 * x + 3;
wcout << y << endl;
```

Kokia bus `y` reikšmė, kai `x` reikšmė lygi: a) 2? b) 5? c) 7?

## Užduotys

1. Pakeiskite programą `Darbas4`, kad ji spręstų kvadratines lygtis su realiojo tipo koeficientais.

   *Pasitikrinkite.* Kai $a = 3.05$, $b = -2.15$ ir $c = -7.5$, tai lygties sprendiniai yra: $x1 = -1.25$, $x2 = 1.96$.

2. Pirmosios olimpinės žaidynės įvyko 1896 metais ir toliau organizuojamos kas ketveri metai. Jei žaidynės neįvyksta, tie metai vis tiek laikomi olimpiniais, o žaidynėms skiriamas eilės numeris. Parašykite programą, kuri surastų $m$-ųjų metų olimpinių žaidynių numerį $n$. Jei metai neolimpiniai, turi būti spausdinama „Metai neolimpiniai“.

   *Pasitikrinkite.* Kai $m = 1904$, turi būti spausdinama: $n = 3$. Kai m = 2005, turi būti spausdinama:
   `Metai neolimpiniai.`

3. Ūkininkas nusprendė virve pažymėti stačiakampį plotą, kuriame sodins ankstyvąsias bulves. Virvės ilgis lygus $m$ metrų (sveikasis skaičius). Kokį didžiausią plotą $s$ galės pažymėti ūkininkas? Rezultatą pateikite sveikuoju skaičiumi (gali likti nepanaudotas virvės galas).

   *Pasitikrinkite.* Kai $m = 22$, turi būti spausdinama: $s = 30$. Kai $m = 21$, turi būti spausdinama: $s = 25$.

4. Šviesoforas veikia pagal tokį algoritmą: kiekvienos valandos pirmąsias 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 šviesa dega.

   *Pasitikrinkite.* Kai $t = 12$, turi būti spausdinama: `Dega žalia šviesa`. Kai $t = 13$, turi būti spausdinama: `Dega žalia šviesa, tuoj užsidegs raudona`. Kai $t = 5$, turi būti spausdinama:
   `Dega raudona šviesa, tuoj užsidegs žalia.`



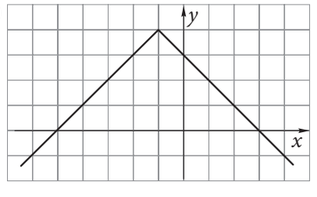
5. Geležinkelio stotys *A*, *B* ir *C* yra *n*-ajame, *m*-ajame ir *p*-ajame geležinkelio ruožo kilometruose. Parašykite programą, kuri surastų, tarp kurių stočių atstumas yra mažiausias. Stotys nebūtinai įvardytos abėcėlės tvarka, pavyzdžiui, po stoties *A* gali sekti stotis *C*.

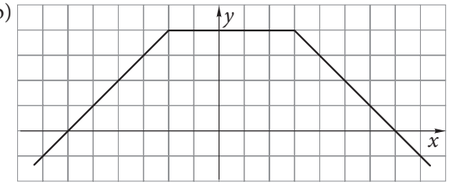
   *Pasitikrinkite.* Kai $n = 3$, $m = 8$, $p = 15$, turi būti spausdinama:
   `Atstumas mažiausias tarp A ir B stočių.`
   Kai $n = 3$, $m = 9$, $p = 15$, turi būti spausdinama:
   `Atstumai mažiausi tarp A ir B bei B ir C stočių.`
   Kai $n = 3$, $m = 15$, $p = 9$, turi būti spausdinama:
   `Atstumai mažiausi tarp A ir C bei B ir C stočių.`

6. 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 čia vyksta varžybos. Kurorte *C* slidininkai laukiami nuo lapkričio pradžios iki gegužės pabaigos. Poilsio kaina kiekviename kurorte, įtraukus 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. 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 kurorte C. Jam reikės 499 Lt.`

7. Remdamiesi funkcijų grafikais, užrašykite, kaip skaičiuojamos tiesinių funkcijų reikšmės (1 langelis atitinka 1 vienetą).

a)



b)


# 2.5. Elektros laidininkų varžos skaičiavimas

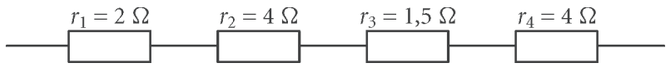
Atlikdami šį darbą, išsiaiškinsite, kaip kuriama ciklinius skaičiavimus atliekanti programa, kai žinoma, kiek kartų bus kartojami veiksmai:

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

| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.1. Kintamasis, kintamojo reikšmė<br>3.2. Priskyrimo sakinys<br>3.4. Duomenų įvedimas klaviatūra<br>3.5. Rezultatų (duomenų) išvedimas į ekraną<br>3.7. Ciklo sakinys **`for`** | 4.1. Tiesiniai algoritmai<br>4.2. Cikliniai algoritmai<br>4.4. Sumos skaičiavimo algoritmas |

## Užduotis

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




## Algoritmas

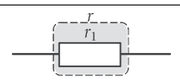
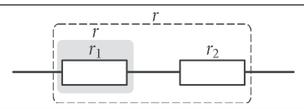
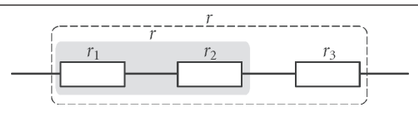
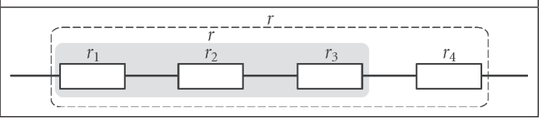
Tarkime, kad elektros grandinę sudaro $n = 4$ nuosekliai sujungti laidininkai, kurių varžos yra $r1 = 2$, $r2 = 4$, $r3 = 1{,}5$, $r4 = 4$. Bendroji grandinės varža skaičiuojama sumuojant visų laidininkų varžas.

Žinant laidininkų skaičių *n* ir kiekvieno laidininko varžą *rlaid*, grandinės varžą *r* galima rasti pagal algoritmą, kurio veiksmai kartojami *n* kartų:

- įvedama laidininko varža *rlaid*;
- skaičiuojama grandinės varža $r = r + rlaid$.

Prieš pradedant vykdyti veiksmus, būtina žinoti *n* reikšmę, o kintamojo *r* pradinė reikšmė turi būti lygi nuliui.

Lentelėje pavaizduota, kaip atliekami veiksmai.

| Ciklas vykdomas *i*-ąjį kartą | Įvedama *i*-ojo laidininko varža | Skaičiuojama grandinės varža $r = r + rlaid$ | Iliustracija |
|---|---|---|---|
| 1 | 2 | $r = 0 + 2 = 2$ |  |
| 2 | 4 | $r = 2 + 4 = 6$ |  |
| 3 | 1,5 | $r = 6 + 1{,}5 = 7{,}5$ |  |
| 4 | 4 | $r = 7{,}5 + 4 = 11{,}5$ |  |

Skaičiavimų rezultatas: $r = 11{,}5$.



## 1 Pasiruošimas

- Sukurkite katalogą *Darbas5*, skirtą programos failams laikyti. Paleiskite *CodeBlocks*. Sukurkite programos failą ir, suteikę programai vardą `Darbas5`, įrašykite failą vardu *Darbas5.cpp* į katalogą *Darbas5*.

## 2 Programos pradinių duomenų aprašymas ir įvedimas

- Programos pradžioje aprašykite naudojamus kintamuosius (laidininkų skaičių `n`, laidininko varžą `rlaid`, grandinės varžą `r`, žinomo kartojimų skaičiaus ciklo kintamąjį `i`).
- Parašykite kintamojo `n` reikšmės įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį (`wcout`) ir reikšmės skaitymo sakinį (`cin`).

```
// Darbas5
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int n;           // laidininkų skaičius
  double rlaid;    // laidininko varža
  double r;        // elektros grandinės varža
  int i;           // žinomo kartojimų skaičiaus ciklo kintamasis
  wcout << L"Kiek laidininkų yra elektros grandinėje? ";  cin >> n;
  return 0;
}
```

## 3 Pradinės kintamojo `r` reikšmės aprašymas ir rezultato išvedimas į ekraną

- Papildykite programą priskyrimo sakiniu, skirtu kintamojo `r` pradinei reikšmei aprašyti.
- Papildykite programą sakiniu, skirtu `r` reikšmei išvesti.

```
// Darbas5
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int n;           // laidininkų skaičius
  double rlaid;    // laidininko varža
  double r;        // elektros grandinės varža
  int i;           // žinomo kartojimų skaičiaus ciklo kintamasis
  wcout << L"Kiek laidininkų yra elektros grandinėje? ";  cin >> n;
  r = 0;
  wcout << L"Elektros grandinės varža: "
        <<  setw(6) << fixed << setprecision(2) << r << endl;
  return 0;
}
```

Sakinyje

```
wcout << L"Elektros grandinės varža: "
      <<  setw(6) << fixed << setprecision(2) << r << endl;
```

nurodoma, kad apskaičiuotai elektros grandinės varžos reikšmei *r* išvesti į ekraną skiriamos 6 pozicijos, iš kurių 2 – trupmeninei daliai. Manipuliatorius `fixed` rodo, kad trupmeninė dalis visuomet turi būti iš dviejų skaitmenų. Taškui, kuris atskiria trupmeninę dalį nuo sveikosios, taip pat skiriama viena pozicija.

➢ Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `4` ir spustelėkite klavišą *Enter*. Ekrane matysite:

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

## Ciklo sakinio antraštė. Veiksmai cikle

Ciklo sakinyje atliekami keli veiksmai, todėl jie turi būti rašomi tarp skliaustų **{** ir **}**.

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

```
for (i = 1; i <= n; i = i + 1) {
  wcout  << L"Įveskite laidininko varžą: ";  cin >> rlaid;
  r = r + rlaid;
}
```

Papildyta programa bus tokia:

```
// Darbas5
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int n;          // laidininkų skaičius
  double rlaid;   // laidininko varža
  double r;       // elektros grandinės varža
  int i;          // žinomo kartojimų skaičiaus ciklo kintamasis
  wcout << L"Kiek laidininkų yra elektros grandinėje? ";  cin >> n;
  r = 0;
  for (i = 1; i <= n; i = i + 1) {
    wcout  << L"Įveskite laidininko varžą: ";  cin >> rlaid;
    r = r + rlaid;
  }
  wcout << L"Elektros grandinės varža: "
        << setw(6) << fixed << setprecision(2) << r << endl;
  return 0;
}
```

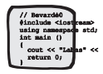
➢ Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `4` ir spustelėkite klavišą *Enter*. Ekrane matysite:

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

➢ Iš eilės įveskite laidininkų varžas: `2`, `4`, `1.5`, `4`. Po kiekvienos jų spustelėkite klavišą *Enter*. Ekrane matysite:



```
Kiek laidininkų yra elektros grandinėje? 4
Įveskite laidininko varžą 2
Įveskite laidininko varžą 4
Įveskite laidininko varžą 1.5
Įveskite laidininko varžą 4
Elektros grandinės varža: 11.50
```



## Užduotys

1. Elektros grandinę sudaro *n* lygiagrečiai sujungtų laidininkų, kurių varžos yra *r1*, *r2*, ..., *rn* omų. Parašykite programą, kuri apskaičiuotų 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. Jų ūgiai atitinkamai yra *u1*, *u2*, ..., *un* centimetrų. Parašykite programą, kuri apskaičiuotų 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 mokinių ūgis uvid = 178.20 cm.`
3. Keliamieji metai turi 366 dienas, paprastieji – 365. Keliamaisiais vadinami metai, kurie be liekanos dalijasi iš 4. Šimtmečių metai keliamaisiais laikomi tuomet, kai jie be liekanos dalijasi iš 400. Parašykite programą, kuri ekrane parodytų keliamuosius metus laikotarpio, kuris prasideda *m*-aisiais, o baigiasi *n*-aisiais metais.
   *Pasitikrinkite.* Kai *m* = 1898, o n = 1910, turi būti spausdinama:
   `Keliamieji metai yra 1904, 1908.`
4. Šachmatų išradėjas iš valdovo paprašė tokio atlygio: ant pirmo šachmatų lentos langelio padėti vieną grūdą, ant antrojo – du, ant trečiojo – keturis ir t. t., vis dvigubinant, 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. Rezultato reikšmei atmintyje laikyti panaudokite `double` duomenų tipą.
   *Pasitikrinkite.* Ekrane turi būti spausdinama:
   `Šachmatų išradėjas gaus 18446744073709550000 grūdų.`
5. Autobusų parko administracija nusprendė keleiviams, kurių bilietų numeriai laimingi, dovanoti kelionę už pusę kainos. Autobuso bilietas laikomas laimingu, jei jo pirmųjų trijų skaitmenų trejetas sutampa su paskutinių trijų skaitmenų trejetu (pvz., laimingas bilietas, kurio numeris yra 234234). Autobusų parko administracija nutarė bilietus sunumeruoti nuo *m*-ojo iki *n*-ojo šešiaženklio skaičiaus. Parašykite programą, kuri apskaičiuotų, kiek keleivių *k* įsigis laimingus bilietus.
   *Pasitikrinkite.* Kai *m* = 170849, o *n* = 189965, turi būti spausdinama:
   `Laimingus bilietus įsigijo k = 19 keleivių.`
6. Architektas suprojektavo salę, kurioje bus *n* eilių. Pirmoje eilėje stovės *k* kėdžių, o kiekvienoje kitoje eilėje – 2 kėdėmis daugiau, negu prieš tai buvusioje. Parašykite programą, kuri apskaičiuotų, kiek iš viso kėdžių (*s*) reikia užsakyti, kad architekto sumanymas būtų įgyvendintas.
   *Pasitikrinkite.* Kai *n* = 3, o *k* = 8, turi būti spausdinama: `s = 30 kėdžių.`

## SMALSIEMS

| Nuorodos į C++ kalbos žinyną |
|---|
| 3.10. Duomenų įvedimas iš failo |

Jei pradinių duomenų yra labai daug, tai jų įvedimas klaviatūra reikalauja nemažai laiko, juolab kad padarius klaidą tenka duomenis įvesti iš naujo. Daug patogiau pradinius duomenis įvesti iš tekstinio failo. Papildysime elektros grandinės varžos skaičiavimo programą – duomenis įvesime iš tekstinio failo.

➢ Tekstinis failas sukuriamas komandomis: *File* → *New* → *Empty file*. Atsivėrusio lango pirmoje eilutėje įrašykite skaičių `4`, kuris reiškia laidininkų skaičių, antroje eilutėje įrašykite keturių laidininkų varžas, vieną nuo kitos atskirdami tarpais: `2 4 1.5 4`. Ekrane turėtumėte matyti:

```
4
2 4 1.5 4
```

➢ Sukurtą failą pavadinkite *Darbas5.txt* ir įrašykite į katalogą *Darbas5*.

➢ Pakeiskite sukurtą programą *Darbas5.cpp*:

```
// Darbas5
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int n;           // laidininkų skaičius
  double rlaid;    // laidininko varža
  double r;        // elektros grandinės varža
  int i;           // žinomo kartojimų skaičiaus ciklo kintamasis
  ifstream fd("Darbas5.txt");   // paruošiamas failas duomenų skaitymui
  fd >> n;         // įvedama pirmoje failo eilutėje esanti skaičiaus n reikšmė
  r = 0;
  // Įvedami kiti skaičiai. Skaityti baigiama, kai įvedama n reikšmių
  for (i = 1; i <= n; i = i + 1) {
    fd >> rlaid;
    r = r + rlaid;
  }
  fd.close();      // duomenų failas užveriamas
  wcout << L"Elektros grandinės varža: "
        << setw(6) << fixed << setprecision(2) <<r << endl;
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą. Ekrane matysite:

```
Elektros grandinės varža:  11.50
```

Kaip pastebėjote, duomenys iš failo įvedami panašiai kaip ir klaviatūra. Tik 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
1.5
4
```



# 2.6. Siena

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

- išmoksite užrašyti nežinomo kartojimų skaičiaus ciklo antraštę;
- išsiaiškinsite, kaip užrašomi cikle atliekami veiksmai;
- prisiminsite, kaip skaičiuojama suma;
- suprasite, kaip skaičiuojamas kiekis;
- išmoksite skaičiavimų rezultatus pateikti lentele.

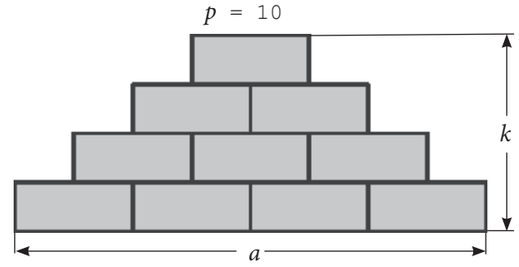
| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.1. Kintamasis, kintamojo reikšmė<br>3.2. Priskyrimo sakinys<br>3.4. Duomenų įvedimas klaviatūra<br>3.5. Rezultatų (duomenų) išvedimas į ekraną<br>3.6. Ciklo sakinys **`while`** | 4.1. Tiesiniai algoritmai<br>4.2. Cikliniai algoritmai<br>4.4. Sumos skaičiavimo algoritmas<br>4.6. Kiekio skaičiavimo algoritmas |

## Užduotis

Iš plytų galima pastatyti vienos plytos storio taisyklingą sieną, kurios viršūnėje yra viena plyta, o šonuose – pusės plytos ilgio laipteliai. Reikia parašyti programą, kuri apskaičiuotų, kelių plytų *a* bus sienos pagrindas ir kelių plytų aukščio *k* bus siena, jei žinomas sienos statybai skirtų plytų skaičius *p*.

## Algoritmas

Paveiksle pavaizduotos sienos pagrindą *a* sudaro `4` plytos, sienos aukštis *k* = 4 plytos, sienai pastatyti reikėjo *p* = 10 plytų.

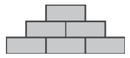
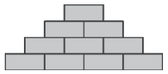



Žinome bendrą plytų skaičių *p*, o *a* ir *k* reikšmes turime apskaičiuoti. Sienai pastatyti panaudotą plytų skaičių paprasčiau skaičiuoti iš viršaus į apačią.

Skaičiavimai kartojami, kol statyboms panaudotų plytų skaičius *s* mažesnis už statyboms skirtų plytų skaičių *p*:

- Plytų skaičius eilėje *a* gaunamas didinant buvusią *a* reikšmę vienetu, nes kiekvienoje naujoje eilėje yra viena plyta daugiau negu prieš tai buvusioje.
- Sienos aukščio *k* reikšmė gaunama didinant buvusią *k* reikšmę vienetu, nes pradedamas skaičiuoti plytų skaičius naujoje eilėje.
- Sienos statybai panaudotų plytų skaičius *s* gaunamas prie jau buvusios *s* reikšmės pridedant naujos eilės plytų skaičių *a*.

Lentelėje pavaizduota, kaip keičiasi kintamųjų *a*, *k* ir *s* reikšmės, atliekant veiksmus, kol *s* reikšmė mažesnė už *p* reikšmę. Pradinės kintamųjų reikšmės: $a = 0$, $k = 0$, $s = 0$, $p = 10$.

| Eilė | Ar statyboms panaudotų plytų skaičius mažesnis už sienai skirtų plytų skaičių | | Plytų skaičius eilėje *a* | Sienos aukštis *k* | Statyboms panaudotų plytų skaičius *s* | |
|---|---|---|---|---|---|---|
| Ciklo žingsniai | Sąlyga: $s < p$ | | $a = a + 1$ | $k = k + 1$ | $s = s + a$ | Paveikslas |
| 1 | $0 < 10$ | TAIP | $a = 0 + 1 = 1$ | $k = 0 + 1 = 1$ | $s = 0 + 1 = 1$ |  |
| 2 | $1 < 10$ | TAIP | $a = 1 + 1 = 2$ | $k = 1 + 1 = 2$ | $s = 1 + 2 = 3$ |  |
| 3 | $3 < 10$ | TAIP | $a = 2 + 1 = 3$ | $k = 2 + 1 = 3$ | $s = 3 + 3 = 6$ |  |
| 4 | $6 < 10$ | TAIP | $a = 3 + 1 = 4$ | $k = 3 + 1 = 4$ | $s = 6 + 4 = 10$ |  |
| 5 | $10 < 10$ | NE | Veiksmai neatliekami, nes netenkinama sąlyga $s < p$ | | | |

## Pasiruošimas

➢ Sukurkite katalogą *Darbas6*, skirtą programos failams laikyti, paleiskite *CodeBlocks*, sukurkite programos failą ir, suteikę programai vardą `Darbas6`, įrašykite failą vardu *Darbas6.cpp* į katalogą *Darbas6*.

## Programos pradinių duomenų aprašymas ir įvedimas

➢ Programos pradžioje aprašykite naudojamus kintamuosius (sienai statyti skirtų plytų skaičių `p`, sienos pagrindo ilgį `a`, sienos aukštį `k`, sienos statybai panaudotų plytų skaičių `s`).

➢ Parašykite kintamojo `p` reikšmės įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį (`wcout`) ir reikšmės skaitymo sakinį (`cin`).

```
// Darbas6
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int p,     // sienos statybai skirtų plytų skaičius
      s,     // sienos statybai panaudotų plytų skaičius
      a,     // sienos pagrindo ilgis
      k;     // sienos aukštis
  wcout << L"Kiek plytų skirta sienos statybai? "; cin >> p;
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `10` ir spustelėkite klavišą *Enter*. Ekrane matysite:

```
Kiek plytų skirta sienos statybai? 10
```



## Pradinių reikšmių kintamiesiems `a`, `k` ir `s` priskyrimas ir jų išvedimas į ekraną

➢ Papildykite programą trimis priskyrimo sakiniais:

```
a = 0; k = 0; s = 0;
```

ir sakiniais, skirtais kintamųjų `a`, `k` ir `s` reikšmėms išvesti į ekraną:

```
// Darbas6
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);

  int p,     // sienos statybai skirtų plytų skaičius
      s,     // sienos statybai panaudotų plytų skaičius
      a,     // sienos pagrindo ilgis
      k;     // sienos aukštis

  wcout << L"Kiek plytų skirta sienos statybai? "; cin >> p;
  a = 0; k = 0; s = 0;
  wcout << L"Sienos pagrindo ilgis: " << a << endl;
  wcout << L"Sienos aukštis: " << k << endl;
  wcout << L"Sienos statybai panaudotų plytų skaičius: " << s << endl;
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `10` ir spustelėkite klavišą *Enter*. Ekrane matysite:

```
Kiek plytų skirta sienos statybai? 10
Sienos pagrindo ilgis: 0
Sienos aukštis: 0
Sienos statybai panaudotų plytų skaičius: 0
```

## Ciklo sakinio antraštė. Veiksmai cikle

Cikle atliekami keli veiksmai, todėl ciklo sakinyje jie turi būti rašomi tarp skliaustų `{` ir `}`.

➢ Priskyrę kintamiesiems `a`, `k` ir `s` pradines reikšmes, užrašykite ciklo sakinį:

```
while (s < p) {
  a = a + 1;
  k = k + 1;
  s = s + a;
}
```

Papildyta programa bus tokia:

```
// Darbas6
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int p,     // sienos statybai skirtų plytų skaičius
      s,     // sienos statybai panaudotų plytų skaičius
      a,     // sienos pagrindo ilgis
      k;     // sienos aukštis
  wcout << L"Kiek plytų skirta sienos statybai? "; cin >> p;
  a = 0; k = 0; s = 0;
  while (s < p) {
    a = a + 1;
    k = k + 1;
    s = s + a;
  }
  wcout << L"Sienos pagrindo ilgis: " << a << endl;
  wcout << L"Sienos aukštis: " << k << endl;
  wcout << L"Sienos statybai panaudotų plytų skaičius: " << s << endl;
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `10` ir spustelėkite klavišą *Enter*. Ekrane matysite:

```
Kiek plytų skirta sienos statybai? 10
Sienos pagrindo ilgis: 4
Sienos aukštis: 4
Sienos statybai panaudotų plytų skaičius: 10
```

## 5 Skaičiavimų tarpinių rezultatų išvedimas

Norėdami įsitikinti, kad programa skaičiuoja teisingai, galite išvesti ne tik galutinius, bet ir tarpinius rezultatus.

➢ Sudėtinį sakinį, nurodantį, kokie veiksmai bus atliekami ciklo viduje, papildykite kintamųjų `a`, `k` ir `s` reikšmių išvedimo sakiniais ir sakiniu, atskiriančiu vieną skaičiavimą nuo kito žvaigždutėmis.

```
  while (s < p) {
    a = a + 1;
    k = k + 1;
    s = s + a;
    wcout << L"a = " << a << endl;
    wcout << L"k = " << k << endl;
    wcout << L"s = " << s << endl;
    wcout << L"**********" << endl;
  }
```



➢ Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `10` ir spustelėkite klavišą *Enter*. Ekrane matysite:

```
Kiek plytų skirta sienos 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
**********
Sienos pagrindo ilgis: 4
Sienos aukštis: 4
Sienos statybai panaudotų plytų skaičius: 10
```

## 6 Skaičiavimų rezultatų pateikimas lentele

➢ Norėdami programos skaičiavimų rezultatus pateikti lentele, programą papildykite:

➢ Sakiniais, apibrėžiančiais programos paskirtį:

```
wcout << L"Programa, skaičiuojanti sienos pagrindo ilgį, aukštį" << endl;
wcout << L" ir statybai panaudotų plytų skaičių" << endl;
wcout << L" " << endl;
```

➢ Sakiniais, formuojančiais antraštinę lentelės eilutę:

```
wcout << L"-----------------------------------------------------" << endl;
wcout << L"Pagrindo ilgis   Aukštis  Panaudotų plytų skaičius" << endl;
wcout << L"-----------------------------------------------------" << endl;
```

➢ Sakiniu, išvedančiu ciklo viduje apskaičiuotas kintamųjų `a`, `k` ir `s` reikšmes:

```
wcout << L"   " << a << L"          " << k << L"            " << s << endl;
```

➢ Sakiniu, formuojančiu paskutinę lentelės eilutę:

```
wcout << L"-----------------------------------------------------" << endl;
```

Papildyta programa bus tokia:

```
// Darbas6
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int p,     // sienos statybai skirtų plytų skaičius
      s,     // sienos statybai panaudotų plytų skaičius
      a,     // sienos pagrindo ilgis
      k;     // sienos aukštis

  wcout << L"Programa, skaičiuojanti sienos pagrindo ilgį, aukštį" << endl;
  wcout << L" ir statybai panaudotų plytų skaičių" << endl;
  wcout << L" " << endl;
  wcout << L"Kiek plytų skirta sienos statybai? "; cin >> p;
  a = 0; k = 0; s = 0;
  wcout << L"-------------------------------------------------" << endl;
  wcout << L"Pagrindo ilgis   Aukštis  Panaudotų plytų skaičius" << endl;
  wcout << L"-------------------------------------------------" << endl;
  while (s < p) {
    a = a + 1;
    k = k + 1;
    s = s + a;
    wcout << L"    " << a << L"             " << k << L"             " << s
          << endl;
  }
  wcout << L"-------------------------------------------------" << endl;
  wcout << L"Sienos pagrindo ilgis: " << a << endl;
  wcout << L"Sienos aukštis: " << k << endl;
  wcout << L"Sienos statybai panaudotų plytų skaičius: " << s << endl;
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių `10` ir spustelėkite klavišą *Enter*. Ekrane matysite:

```
Programa, skaičiuojanti sienos pagrindo ilgį, aukštį
 ir statybai panaudotų plytų skaičių

Kiek plytų skirta sienos statybai? 10
-------------------------------------------------
Pagrindo ilgis   Aukštis  Panaudotų plytų skaičius
-------------------------------------------------
    1             1                  1
    2             2                  3
    3             3                  6
    4             4                 10
-------------------------------------------------
Sienos pagrindo ilgis: 4
Sienos aukštis: 4
Sienos statybai panaudotų plytų skaičius: 10
```

## 7 Tikrinimas, ar programa visada pateikia teisingus rezultatus

➢ Vykdydami programą, įveskite `p` reikšmę, lygią `8`. Įvykdę programą, gausite tą patį rezultatą, kaip ir įvedę `p` reikšmę, lygią `10`. Rezultatas yra neteisingas, nes sienos statybai panaudota daugiau plytų, negu skirta.

*Išvada*. Reikia tikrinti, ar likusių plytų užteks naujai eilei.



➢ Ciklo antraštėje sąlygą `s < p` pakeiskite sąlyga `a + 1 <= p - s` ir įvykdykite programą esant `p` reikšmei, lygiai `8`. Ekrane turėtumėte matyti:

```
Programa, skaičiuojanti sienos pagrindo ilgį, aukštį
 ir statybai panaudotų plytų skaičių

Kiek plytų skirta sienos statybai? 8
-------------------------------------------------
Pagrindo ilgis   Aukštis  Panaudotų plytų skaičius
-------------------------------------------------
    1             1                  1
    2             2                  3
    3             3                  6
-------------------------------------------------
Sienos pagrindo ilgis: 3
Sienos aukštis: 3
Sienos statybai panaudotų plytų skaičius: 6
```

Patikslinus ciklo antraštės sąlygą, programa pateikia teisingus rezultatus esant visoms kintamojo `p` reikšmėms.

➢ Papildykite programą skaičiavimais, kiek liko nepanaudotų plytų.

➢ Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičių 8 ir spustelėkite klavišą *Enter*. Ekrane turėtumėte matyti:

```
Nepanaudotų plytų skaičius: 2
```

## Klausimai

1. Kaip vykdomas ciklo sakinys **`while`**?
2. Kaip užrašyti ciklo sakinį, kai norima atlikti daugiau veiksmų?
3. Ką daryti, kad ciklas nebūtų begalinis?
4. Kurie iš nurodytų ciklo sakinių, kai `a = -3`, yra begaliniai ir kodėl?

a)
```
while (a < 0){
    wcout << L"argumentas neigiamas" << endl;
    a = a + 1;
}
```

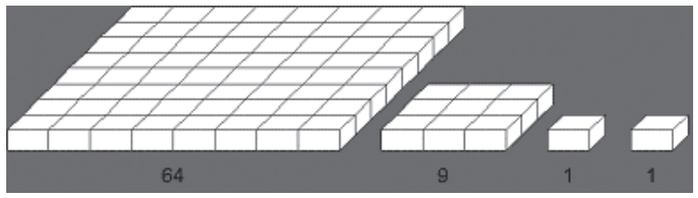
b)
```
while (a < 0);{
    wcout << L"argumentas neigiamas" << endl;
    a = a + 1;
}
```

c)
```
while (a < 0)
    wcout << L"argumentas neigiamas" << endl;
a = a + 1;
```

d)
```
while (a >= 0)
    wcout << L"argumentas neigiamas" << endl;
a = a + 1;
```

## Užduotys

1. Martynas labai mėgsta saldainius. Mamos slėptuvėje berniukas rado *m* saldainių. Pirmą dieną jis suvalgė 1 saldainį, antrąją – 2, trečiąją – 3. Kiekvieną kitą dieną jis suvalgydavo vienu saldainiu daugiau negu prieš tai buvusią dieną. Per kelias dienas *d* Martynas suvalgys visus saldainius? Paskutinei dienai gali likti mažiau saldainių.
   *Pasitikrinkite*. Kai $m = 11$, turėtumėte gauti $d = 5$.
2. 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 $p = 5$, $ind = 1000$, $s = 1200$, turėtumėte gauti $t = 4$.
3. Turime kompiuterį, kuris nemoka apskaičiuoti natūraliųjų skaičių dalmens sveikosios dalies ir liekanos (nėra `/` ir `%` dalybos operacijų). Parašykite programą, kuri apskaičiuotų skaičių *n* ir *m* dalmens sveikąją dalį *dalmuo* ir liekaną *liekana*.
   *Pasitikrinkite.* Kai $n = 14$, $m = 3$, turėtumėte gauti: *dalmuo* = 4, *liekana* = 2.
4. Pristigo žmogus pinigų ir nuėjo pasiskolinti jų iš kaimyno. Tas sutiko paskolinti, bet paprašė grąžinti juos kitą mėnesį tokia tvarka: pirmą mėnesio dieną – 1 litą, antrąją – 2 litus, trečiąją – 4 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 daugiau, ir bus kaimyno *palūkanos*. Parašykite programą, kuri apskaičiuotų, kiek palūkanų litais gaus kaimynas už paskolintus *n* litų.
   *Pasitikrinkite*. Kai $n = 11$, turėtumėte gauti *palūkanos* = 4. (XIII olimpiada, 2002)
5. Iš *n* kvadratinių plytelių reikia sudėlioti vienos plytelės storio kvadratus: pirmiausia sudėti didžiausią galimą kvadratą, iš likusių plytelių – vėl didžiausią ir t. t. Parašykite programą, kuri išskaidytų nurodytą plytelių skaičių į dalis, reikalingas kiekvieno kvadrato statybai.
   *Pasitikrinkite.* Kai $n = 75$, turėtumėte gauti: 64, 9, 1, 1.




(XI olimpiada, 2000)



## SMALSIEMS

| Nuorodos į C++ kalbos žinyną |
|---|
| 3.11. Rezultatų (duomenų) išvedimas į failą |

➢ Norėdami, kad sienos statybos rezultatai būtų išvedami į tekstinį failą, atlikite programos *Darbas6.cpp* pakeitimus:

```
// Darbas6
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <fstream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int p,     // sienos statybai skirtų plytų skaičius
      s,     // sienos statybai panaudotų plytų skaičius
      a,     // sienos pagrindo ilgis
      k;     // sienos aukštis
  ofstream fr("Darbas6.txt");
  fr << "Programa, skaičiuojanti sienos pagrindo ilgį, aukštį" << endl;
  fr << " ir statybai panaudotų plytų skaičių" << endl;
  fr << endl;
  wcout << L"Kiek plytų skirta sienos statybai? "; cin >> p;
  a = 0; k = 0; s = 0;
  fr << "-------------------------------------------------" << endl;
  fr << "Pagrindo ilgis   Aukštis  Panaudotų plytų skaičius" << endl;
  fr << "-------------------------------------------------" << endl;
  while (a + 1 <= p - s) {
    a = a + 1;
    k = k + 1;
    s = s + a;
    fr << "    " << a <<"             " << k << "                  " << s << endl;
  }
  fr << "-------------------------------------------------" << endl;
  fr << "Sienos pagrindo ilgis: " << a << endl;
  fr << "Sienos aukštis: " << k << endl;
  fr << "Sienos statybai panaudotų plytų skaičius: " << s << endl;
  fr.close();
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą. Atverkite rezultatų failą ir peržiūrėkite rezultatus. Jie turėtų būti tokie patys, kaip ir rezultatai, išvesti į ekraną.

# 2.7. Funkcijos apibrėžimo srities tyrimas

Atlikdami šį darbą, įtvirtinsite sąlyginio ir nežinomo kartojimų skaičiaus ciklo sakinių užrašymo įgūdžius.

| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.1. Kintamasis, kintamojo reikšmė<br>3.2. Priskyrimo sakinys<br>3.4. Duomenų įvedimas klaviatūra<br>3.5. Rezultatų (duomenų) išvedimas į ekraną<br>3.6. Ciklo sakinys **`while`**<br>3.8. Sąlyginis sakinys **`if`**<br>3.9. Knygoje naudojamų ir / ar rekomenduojamų matematinių funkcijų sąrašas | 4.1. Tiesiniai algoritmai<br>4.2. Cikliniai algoritmai<br>4.3. Šakotieji skaičiavimai |

## Užduotis

Reikia apskaičiuoti funkcijos $y = \dfrac{m+3}{\sqrt{m^2-100}}$ reikšmę, kai $mp \le m \le mg$ ir kinta žingsniu $mz$; čia $mp$, $mg$, $mz$ – realieji skaičiai.

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

### 1 Pasiruošimas

➢ Sukurkite katalogą *Darbas7* programos failams laikyti. Paleiskite *CodeBlocks*. Sukurkite programos failą *Darbas7.cpp,* įrašykite jį į katalogą *Darbas7.* Programai suteikite vardą `Darbas7`.

### 2 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` reikšmėms įvesti klaviatūra.

```
// Darbas7
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  double mp, mg, mz, m, y;
  wcout << L"Įveskite argumento pradinę reikšmę: ";         cin >> mp;
  wcout << L"Įveskite argumento galinę reikšmę:   ";        cin >> mg;
  wcout << L"Įveskite argumento kitimo žingsnio reikšmę: "; cin >> mz;
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą, kai pradiniai duomenys yra tokie: `mp = -50, mg = 50, mz = 10.` Ekrane matysite:



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

### 3 Funkcijos reikšmių skaičiavimas ir išvedimas į ekraną

➢ Papildykite programą tokiais sakiniais:

```
  wcout << L"Funkcijos reikšmių skaičiavimas" << endl;
  wcout << L" mp = " << setw(5) << fixed << setprecision(2) << mp
        << L" mg = " << setw(5) << fixed << setprecision(2) << mg
        << L" mz = " << setw(5) << fixed << setprecision(2) << mz << endl;
  m = mp;
  while (m <= mg){
    if (m * m - 100 > 0)     // tikrinama sąlyga, ar m priklauso apibrėžimo sričiai
                             // jei sąlyga tenkinama, skaičiuojama funkcijos reikšmė
      y = (m + 3) / (sqrt(m * m - 100));
      wcout << setw(7)  << fixed << setprecision(2) << m
            << setw(12) << fixed << setprecision(5) << y << endl;
      m = m + mz;
  }
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą, kai pradiniai duomenys yra tokie: `mp = -50, mg = 50, mz = 10.` Ekrane turėtumėte matyti:

```
Įveskite argumento pradinę reikšmę: -50
Įveskite argumento galinę reikšmę: 50
Įveskite argumento kitimo 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
```

### 4 Rezultatų analizė

➢ Išnagrinėkite gautus rezultatus.

Kai `m = -40.00` ir `m = -30.00`, funkcijos `y` reikšmės yra lygios `-0.955`. Esant toms pačioms argumento reikšmėms funkcijos reikšmes apskaičiuokite skaičiuotuvu. Turėtumėte gauti atitinkamai `y = 0.95534` ir `y = 0.95459`. Palyginkite programos pateiktus rezultatus su apskaičiuotais skaičiuotuvu.

*Išvada.* Rašydami rezultatų išvedimo į ekraną sakinius, pasirinkome trupmeninėje dalyje rodyti tris skaitmenis, todėl ir gavome sutampančius rezultatus.

➢ Pakeiskite rašymo sakinį taip, kad išvedamo į ekraną rezultato trupmeninėje dalyje būtų `5` skaitmenys.

➢ Įvykdykite programą. Ekrane turėtumėte matyti:

```
Įveskite argumento pradinę reikšmę: -50
Įveskite argumento galinę reikšmę: 50
Įveskite argumento kitimo 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 ta pati ir lygi `-0.98150`. Tokius rezultatus gavome todėl, kad argumento reikšmės `-10.00`, `0.00` ir `10.00` netenkina sąlygos `m * m - 100 > 0` ir į ekraną išvedama paskutinė apskaičiuota `y` reikšmė.

*Išvada*. Reikia taisyti programą.

## 5 Teisingų rezultatų visoms argumento `m` reikšmėms pateikimas

➢ Funkcijos reikšmių išvedimo sakinį įkelkite į sąlyginį sakinį. Jei argumento `m` reikšmė nepriklauso funkcijos apibrėžimo sričiai, tuomet vietoj funkcijos reikšmės išveskite septynias žvaigždutes (`*`).

```
if (m * m - 100 > 0) {    // tikrinama sąlyga, ar m priklauso apibrėžimo sričiai
                          // jei sąlyga tenkinama, skaičiuojama funkcijos reikšmė
  y = (m + 3) / (sqrt(m * m - 100));
  wcout << setw(7)  << fixed << setprecision(2) << m
        << setw(12) << fixed << setprecision(5) << y << endl;
}
else wcout << setw(7)  << fixed << setprecision(2) << m
           << setw(12) << fixed << setprecision(0) << L"*******" << endl;
```

➢ Įrašykite ir įvykdykite programą. Jei viską atlikote teisingai, ekrane turėtumėte matyti:

```
Įveskite argumento pradinę reikšmę: -50
Įveskite argumento galinę reikšmę: 50
Įveskite argumento kitimo 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
```



## 6 Skaičiavimo rezultatų pateikimas lentele

➢ Papildykite programą taip, kad argumento ir funkcijos reikšmės būtų spausdinamos lentele:

```
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
-----------------------
```

## Užduotys

1. Klasėje yra *n* mokinių. Jų informacinių technologijų savarankiško darbo pažymiai yra *p1*, *p2*, ..., *pn*. Parašykite programą, kuri apskaičiuotų, kelių mokinių *k* darbai buvo įvertinti 9 ir 10.

   *Pasitikrinkite*. Kai *n* = 7, *p1* = 8, *p2* = 9, p3 = 8, *p4* = 9, *p5* = 10, *p6* = 9, *p7* = 10, turi būti spausdinama: `Devintukus ir dešimtukus gavo k = 5 mokiniai.`

2. Parduotuvėje žmogus pirko *n* rūšių prekių. Jų kainos yra *k1*, *k2*, ..., *kn* litų. Parašykite programą, kuri apskaičiuotų, kiek yra prekių *k*, kurių kaina ne didesnė kaip *m* litų, ir kokią pinigų sumą *s* reikės mokėti už šias prekes.

   *Pasitikrinkite*. Kai *n* = 5, *m* = 14, *k1* = 12, *k2* = 6, *k3* = 19, *k4* = 16, *k5* = 2, turi būti spausdinama:
   `Prekių, kurių kaina ne didesnė kaip 14 litų, yra k = 3.`
   `Už jas reikės mokėti s = 20 litų.`

3. Klasėje mokosi *n* mokinių. Jų ūgiai yra *u1*, *u2*, ..., *un* centimetrų. Merginų ūgis žymimas sveikaisiais teigiamais, vaikinų – sveikaisiais neigiamais skaičiais. Parašykite programą, kuri apskaičiuotų, koks yra vidutinis klasės vaikinų *vidvaikinu* ir vidutinis klasės merginų *vidmerginu* ūgis.

   *Pasitikrinkite*. Kai *n* = 7, *u1* = 168, *u2* = –179, *u3* = –178, *u4* = –189, *u5* = 170, *u6* = 169, *u7* = –180, turi būti spausdinama: `vidvaikinu = 181.5, vidmerginu = 169.0.`

4. Varžybose dalyvavo *n* daugiakovininkų. Sportininkai surinko *t1*, *t2*, ..., *tn* taškų (sveikieji skaičiai). Parašykite programą, kuri apskaičiuotų didžiausią *tmax* ir mažiausią *tmin* sportininkų surinktų taškų skaičių.

   *Pasitikrinkite*. Kai *n* = 5, *t1* = 1682, *u2* = 1794, *u3* = 1787, *u4* = 1891, *u5* = 1710, turi būti spausdinama: `tmax = 1891, tmin = 1682.`

5. Pateikiamas dviejų natūraliųjų skaičių *a* ir *b* didžiausiojo bendrojo daliklio (*DBD*) paieškos, vadinamojo *Euklido*, algoritmo 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 nuo 2-ojo žingsnio.

   *Pasitikrinkite*. Kai *a* = 14, *b* = 4, turi būti spausdinama: `DBD = 2`.

# 2.8. Trys lazdos

Atlikdami šį darbą, susipažinsite su loginiais kintamaisiais ir loginiais reiškiniais, loginėmis operacijomis *ARBA* (`||`), *IR* (`&&`). Išmoksite loginius reiškinius naudoti ciklo ir sąlyginiuose sakiniuose.

| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.7. Ciklo sakinys **`for`**<br>3.8. Sąlyginis sakinys **`if`** | 4.2. Cikliniai algoritmai<br>4.3. Šakotieji skaičiavimai |

## Užduotis

Yra *n* tam tikro ilgio lazdų rinkinių. Kiekvieną rinkinį sudaro 3 lazdos. Lazdų ilgiai *a*, *b* ir *c* matuojami decimetrais (sveikaisiais skaičiais). Ar galima iš šių lazdų sudaryti trikampį? Jeigu galima, tai kokį: lygiakraštį, lygiašonį ar įvairiakraštį? Jeigu negalima, reikia išvesti pranešimą, kad trikampio sudaryti negalima.

## Algoritmas

Sprendžiant šią užduotį, galima įsivaizduoti, kad trys lazdos geometrijoje atitinka tris atitinkamo ilgio atkarpas *a*, *b* ir *c*. Prisiminkite, kad ne visuomet iš trijų atkarpų galima sudaryti trikampį. Iš trijų atkarpų trikampį galima sudėlioti tuomet, kai bet kurių atkarpų porų ilgių suma yra didesnė už trečiosios atkarpos ilgį, t. y. $(a + b > c)$ ir $(a + c > b)$ ir $(c + b > a)$ .

### 1 Pasiruošimas

- Sukurkite katalogą programos failams laikyti. Paleiskite *CodeBlocks*. Sukurkite programos failą *Darbas8.cpp,* įrašykite jį į katalogą *Darbas8.* Programai suteikite vardą `Darbas8`.

*Pastaba*. Iš pradžių rašysime programą, skirtą tik vienam lazdų rinkiniui. Vieną rinkinį sudaro trys lazdos, todėl lazdų ilgius įvesime vienu `cin` sakiniu ir nustatysime, ar galima iš lazdų sudaryti trikampį, ar ne.

### 2 Kintamųjų aprašymas ir pradinių duomenų įvedimas

- Programos pradžioje aprašykite trijų lazdų ilgius nusakančius sveikojo tipo kintamuosius `a`, `b`, ir `c`.
- Parašykite kintamųjų `a`, `b`, ir `c` reikšmių įvedimo klaviatūra sakinius: pranešimo, kokias reikšmes ir kokia eilės tvarka įvesti, sakinį (`wcout`) ir reikšmių skaitymo sakinį (`cin`).

```
// Darbas8
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int a, b, c;                  // trijų lazdų ilgiai
  wcout << L"Įveskite trijų lazdų ilgius: ";
  cin >> a >> b >> c;
  return 0;
}
```

- Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičius `30 50 40` ir spustelėkite klavišą *Enter*. Ekrane matysite:

```
Įveskite trijų lazdų ilgius: 30 50 40
```



### 3 Tikrinimas, ar iš trijų lazdų rinkinio galima sudaryti trikampį. Rezultato išvedimas į ekraną

- Papildykite programą sakiniu `wcout`, kuris į ekraną išvestų pradinius duomenis `a`, `b`, `c`.
- Papildykite programą sąlyginiu sakiniu **`if`**, skirtu patikrinti, ar iš nurodytų lazdų galima sudaryti trikampį.

```
// Darbas8
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int a, b, c;                  // trijų lazdų ilgiai
  wcout << L"Įveskite trijų lazdų ilgius: ";
  cin >> a >> b >> c;
  wcout << L"Lazdos: " <<  setw(2) << fixed << a << L" "
        <<  setw(2) << fixed << b << L" " <<  setw(2) << fixed << c;
  if ((a + b > c) && (a + c > b) && (b + c > a)) // ar trikampis?
    wcout << L" - trikampį sudaryti galima" << endl;
  else wcout << L" - trikampio sudaryti negalima" << endl;
  return 0;
}
```

- Įrašykite ir įvykdykite programą. Klaviatūra įveskite skaičius `30 50 40` ir spustelėkite klavišą *Enter*. Ekrane matysite:

```
Įveskite trijų lazdų ilgius: 30 50 40
Lazdos: 30 50 40 - trikampį sudaryti galima
```

- Dar kartą įvykdykite programą. Klaviatūra įveskite skaičius `10 50 40` ir spustelėkite klavišą *Enter*. Ekrane matysite:

```
Įveskite trijų lazdų ilgius: 10 50 40
Lazdos: 10 50 40 - trikampio sudaryti negalima
```

### 4 Tikrinimas, koks yra trikampis, kai jį galima sudaryti

*Lygiakraštį* trikampį galima sudaryti, jeigu dydžiai `a`, `b` ir `c` tenkina sąlygą

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

*Lygiašonį* trikampį galima sudaryti, jeigu dydžiai `a`, `b` ir `c` tenkina sąlygą

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

Jeigu nė viena užrašyta sąlyga netenkinama, tai trikampis yra *įvairiakraštis*.

- Pakeiskite sąlyginio sakinio išvedimo į ekraną sakinį

  `wcout << L" - trikampį sudaryti galima";`

  kitu sudėtiniu sąlyginiu sakiniu, kuris analizuotų, kokio tipo trikampį galima sudaryti iš trijų lazdų rinkinio.

```
// Darbas8
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int a, b, c;                // trijų lazdų ilgiai
  wcout << L"Įveskite trijų lazdų ilgius: ";
  cin >> a >> b >> c;
  wcout << L"Lazdos: " <<  setw(2) << fixed << a << L" "
        <<  setw(2) << fixed << b << L" " <<  setw(2) << fixed << c;
  if ((a + b > c) && (a + c > b) && (b + c > a)) // ar trikampis?
    if ((a == b) && (b == c))
      wcout << L" - galima sudaryti lygiakraštį trikampį" << endl;
    else if ((a == b) || (b == c) || (a == c))
           wcout << L" - galima sudaryti lygiašonį trikampį" << endl;
         else wcout << L" - galima sudaryti įvairiakraštį trikampį" << endl;
  else wcout << L" - trikampio sudaryti negalima" << endl;
  return 0;
}
```

➢ Patikrinkite programą, esant šiems duomenų rinkiniams:

`50 50 50` (Lygiakraštis trikampis)
`40 50 40` (Lygiašonis trikampis)
`40 50 60` (Įvairiakraštis trikampis)
`10 50 40` (Trikampio sudaryti negalima)

Pirmuoju atveju, įvedę duomenis ir spustelėję klavišą *Enter*, ekrane matysite:

```
Įveskite trijų lazdų ilgius: 50 50 50
Lazdos: 50 50 50 - galima sudaryti lygiakraštį trikampį
```

## Gautos programos pritaikymas dideliam skaičiui duomenų rinkinių

Žinome, kad pradinių duomenų rinkinių yra `n`. Jiems analizuoti panaudokime ciklą **`for`**.

➢ Papildykite programą dviem sveikojo tipo kintamaisiais `n` ir `i`. Pirmasis bus skirtas rinkinių skaičiui atmintyje laikyti, o antrasis rodys, su kuriuo rinkiniu atliekami skaičiavimai.

➢ Parašykite dialogo sakinius `n` reikšmei įvesti.

➢ Įkelkite duomenų įvedimo ir sąlyginį sakinius į ciklą **`for`**. Cikle bus vykdomi keli sakiniai, todėl juos apgaubkite **`{`** ir **`}`**.

➢ Įterpkite į duomenų įvedimo dialogo sakinį `wcout` ciklo kintamąjį `i`.



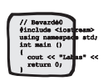
```
// Darbas8
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int a, b, c;                // trijų lazdų ilgiai
  int       n,                // lazdų rinkinių skaičius
            i;                // ciklo kintamasis
  wcout << L"Įveskite, kiek lazdų rinkinių bus: "; cin >> n;
  for (i = 1; i <= n; i = i + 1){
    wcout << L"Įveskite trijų lazdų " << i << L"-ąjį rinkinį: ";
    cin >> a >> b >> c;
    wcout << L"Lazdos: " <<  setw(2) << fixed << a << L" "
          <<  setw(2) << fixed << b << L" " <<  setw(2) << fixed << c;
    if ((a + b > c) && (a + c > b) && (b + c > a)) // ar trikampis?
      if ((a == b) && (b == c))
        wcout << L" - galima sudaryti lygiakraštį trikampį" << endl;
      else if ((a == b) || (b == c) || (a == c))
             wcout << L" - galima sudaryti lygiašonį trikampį" << endl;
           else wcout << L" - galima sudaryti įvairiakraštį trikampį" << endl;
    else wcout << L" - trikampio sudaryti negalima" << endl;
  }
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą su ankstesniais pradiniais duomenimis, `n` reikšmę nurodykite `4`. Ekrane turėtumėte matyti:

```
Įveskite, kiek lazdų rinkinių bus: 4
Įveskite trijų lazdų 1-ąjį rinkinį: 50 50 50
Lazdos: 50 50 50 — galima sudaryti lygiakraštį trikampį
Įveskite trijų lazdų 2-ąjį rinkinį: 40 50 40
Lazdos: 40 50 40 — galima sudaryti lygiašonį trikampį
Įveskite trijų lazdų 3-ąjį rinkinį: 40 50 60
Lazdos: 40 50 60 — galima sudaryti įvairiakraštį trikampį
Įveskite trijų lazdų 4-ąjį rinkinį: 10 50 40
Lazdos: 10 50 40 — trikampio sudaryti negalima
```

## Klausimai

1. Kokias žinote logines operacijas ir kaip jos užrašomos C++ programavimo kalba?
2. Kaip suprantate loginį reiškinį? Paaiškinkite.
3. Kokios operacijos dažniausiai naudojamos aprašant loginį reiškinį?
4. Kokias reikšmes gali įgyti loginis reiškinys?



## Užduotys

1 Trikampis vienu metu gali būti dviejų tipų: statusis ir lygiašonis, lygiašonis ir lygiakraštis. Pakeiskite programą taip, kad ji tai nustatytų.

*Pasitikrinkite.*

Kai $a = 50$, $b = 40$, $c = 30$, turi būti spausdinama: `Trikampis statusis ir įvairiakraštis.`

Kai $a = 50$, $b = 50$, $c = 50$, turi būti spausdinama: `Trikampis lygiakraštis ir lygiašonis.`

2. Papildykite programą taip, kad ji apskaičiuotų ir į ekraną išvestų gautų trikampių plotus. Naudokitės *Herono* formule trikampio plotui apskaičiuoti.

   *Pasitikrinkite*.

   Kai $a = 50$, $b = 50$, $c = 50$, turi būti spausdinama: `s = 1083`.

   Kai $a = 40$, $b = 50$, $c = 40$, turi būti spausdinama: `s = 781`.

   Kai $a = 40$, $b = 50$, $c = 60$, turi būti spausdinama: `s = 992`.

   Kai $a = 10$, $b = 50$, $c = 40$, turi būti spausdinama: `Trikampio sudaryti negalima`.

3. Yra žinomos stačiakampio, kurio kraštinės lygiagrečios su koordinačių ašimis, priešingų kampų (kairiojo viršutinio ir dešiniojo apatinio) koordinatės (*x1*; *y1*) ir (*x2*; *y2*). Parašykite programą, kuri nustatytų, ar taškas (*xt*; *yt*) yra:
   - viduje;
   - išorėje;
   - apatinėje kraštinėje;
   - viršutinėje kraštinėje;
   - kairiojoje kraštinėje;
   - dešiniojoje kraštinėje.

   *Pasitikrinkite*.

   Kai $x1 = 1$, $y1 = 5$, $x2 = 10$, $y2 = 1$, $xt = 4$, $yt = 3$, turi būti spausdinama:
   `Taškas yra stačiakampio viduje`.

   Kai $x1 = 1$, $y1 = 5$, $x2 = 10$, $y2 = 1$, $xt = 0$, $yt = 0$, turi būti spausdinama:
   `Taškas yra stačiakampio išorėje`.

   Kai $x1 = 1$, $y1 = 5$, $x2 = 10$, $y2 = 1$, $xt = 3$, $yt = 1$, turi būti spausdinama:
   `Taškas yra apatinėje kraštinėje`.

   Kai $x1 = 1$, $y1 = 5$, $x2 = 10$, $y2 = 1$, $xt = 5$, $yt = 5$, turi būti spausdinama:
   `Taškas yra viršutinėje kraštinėje`.

   Kai $x1 = 1$, $y1 = 5$, $x2 = 10$, $y2 = 1$, $xt = 1$, $yt = 3$, turi būti spausdinama:
   `Taškas yra kairiojoje kraštinėje`.

   Kai $x1 = 1$, $y1 = 5$, $x2 = 10$, $y2 = 1$, $xt = 10$, $yt = 4$, turi būti spausdinama:
   `Taškas yra dešiniojoje kraštinėje`.

## SMALSIEMS

| Nuorodos į C++ kalbos žinyną |
|---|
| 3.10. Duomenų įvedimas iš failo<br>3.11. Rezultatų (duomenų) išvedimas į failą<br>3.12. Funkcijos |

### 1 Duomenų įvedimas iš failo ir rezultatų išvedimas į failą

- Susikurkite pradinių duomenų failą, pavyzdžiui, *Lazdos.txt*:

```
4
50 50 50
40 50 40
40 50 60
10 50 40
```

Čia pirmoje eilutėje įrašytas lazdų rinkinių skaičius `n`, kitose eilutėse po tris skaičius surašyti lazdų rinkiniai.



- Atlikite tokius programos pakeitimus:
  - programos pradžioje įterpkite failą `fstream`.
  - parenkite failą *Lazdos.txt* duomenims skaityti;
  - įveskite kintamojo `n` reikšmę iš failo;
  - įveskite kintamųjų `a`, `b`, `c` reikšmes iš failo;
  - pašalinkite nereikalingus sakinius:

```
#include <fcntl.h>
#include <io.h>
#include <iostream>
wcout << L"Įveskite, kiek lazdų rinkinių bus: ";
wcout << L"Įveskite trijų lazdų " << i << L"-ąjį rinkinį: ";
```

  - užverkite failą *Lazdos.txt*.
- Įrašykite ir įvykdykite programą.
- Palyginkite gautus rezultatus su ankstesniais. Jeigu viską atlikote teisingai, tai rezultatai turėtų sutapti.
- Panašius veiksmus atlikite rezultatams išvesti ne į ekraną, o į rezultatų failą:
  - sugalvokite rezultatų failo vardą, pavyzdžiui, *Rezultatai.txt*;
  - parenkite failą *Rezultatai.txt* duomenims įrašyti;
  - parašykite išvedimo į failą sakinius.
  - užverkite failą *Rezultatai.txt*.
- Įrašykite ir įvykdykite programą.

Apskaičiuoti rezultatai bus rezultatų faile *Rezultatai.txt*. Jį galima atverti meniu komandomis: *File* → *Open*.

### 2 Programos struktūrizavimas naudojant funkcijas

- Parašykite funkcijas, kurios patikrintų, ar iš 3 lazdų rinkinio galima sudaryti trikampį, ir koks jis yra.

```
// Darbas8
#include <iomanip>
#include <fstream>
using namespace std;
//-----------------------------------------------------------------------------
bool ArTrikampis(int a, int b, int c);
bool ArLygiakrastis(int a, int b, int c);
bool ArLygiasonis(int a, int b, int c);
bool ArIvairiakrastis(int a, int b, int c);
//-----------------------------------------------------------------------------
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int a, b, c,                    // trijų lazdų ilgiai
            n,                    // lazdų rinkinių skaičius
            i;                    // ciklo kintamasis
  ifstream fd("Lazdos.txt");
  ofstream fr("Rezultatai.txt");
  fd >> n;
  for (i = 1 ; i <= n; i = i + 1){
    fd >> a >> b >> c;
    fr << "Lazda: " <<  setw(2) << fixed << a << setw(2) << fixed
       << " " << b << setw(2) << fixed << " " << c << endl;
    if (ArTrikampis(a, b, c)){    // ar trikampis?
      if (ArLygiakrastis(a, b, c))
        fr << " - galima sudaryti lygiakraštį trikampį" << endl;
      if (ArLygiasonis(a, b, c))
        fr << " - galima sudaryti lygiašonį trikampį" << endl;
      if (ArIvairiakrastis(a, b, c))
        fr << " - galima sudaryti įvairikraštį trikampį" << endl;
    }
    else fr << " - trikampio sudaryti negalima" << endl;
  }
```

```
  fd.close();
  fr.close();
  return 0;
}
//---------------------------------------------------------------------------
bool ArTrikampis(int a, int b, int c)
{
  bool t;
  t = (a + b > c) && (a + c > b) && (b + c > a);
  return t;
}
//---------------------------------------------------------------------------
bool ArLygiakrastis(int a, int b, int c)
{
  bool t;
  t = ((a == b) && (b == c));
  return t;
}//---------------------------------------------------------------------------
bool ArLygiasonis(int a, int b, int c)
{
  bool t;
  t = (a == b) || (b == c) || (a == c);
  return t;
}
//---------------------------------------------------------------------------
bool ArIvairiakrastis(int a, int b, int c)
{
  bool t;
  t = (a != b) && (a != c) && (b != c);
  return t;
}
//---------------------------------------------------------------------------
```

➢ Įrašykite ir įvykdykite programą. Rezultatų faile turėtumėte matyti:

```
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
```



# 2.9. Vampyro skaičiai

Atlikdami šį darbą, išmoksite natūraliuosius skaičius skaidyti skaitmenimis, naudodamiesi dalybos operatoriais **/** ir **%**. Sužinosite, kokie skaičiai vadinami *vampyro* skaičiais, išmoksite juos rasti.

| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.4. Duomenų įvedimas klaviatūra<br>3.5. Rezultatų (duomenų) išvedimas į ekraną<br>3.7. Ciklo sakinys **for**<br>3.8. Sąlyginis sakinys **if** | 4.2. Cikliniai algoritmai<br>4.3. Šakotieji skaičiavimai |

## Užduotis

Tarkime, kad turime sveikąjį skaičių *sk*, sudarytą iš *n* (*n* – lyginis skaičius) skaitmenų. Jeigu skaičius *sk* turi 2 daugiklius, iš kurių kiekvienas yra sudarytas iš *n*/2 skaitmenų ir skaičiaus *sk* skaitmenų (visi skaitmenys naudojami tik po 1 kartą), tai jis yra vadinamas *vampỹro skaĩčiumi*, o iš jo skaitmenų sudaryti daugikliai – *vampỹro skaĩčiaus ỉltimis*. Reikia parašyti programą, kuri surastų ir išvestų į ekraną visus keturženklius vampyro skaičius bei jų „iltis“.

## Algoritmas

Prieš pradėdami nagrinėti šią užduotį, pateiksime keletą keturženklių vampyro skaičių ir jo „ilčių“ pavyzdžių:

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

Kaip pastebite, šie keturženkliai skaičiai yra lygūs dviejų dviženklių skaičių sandaugai. Be to, kiekvieno skaičiaus abu dauginamieji turi visus pradinio skaičiaus skaitmenis.

Norint pradėti spręsti šią užduotį, pirmiausia reikia mokėti atskirti skaičiaus skaitmenis. Tai padaryti galima naudojant sveikųjų skaičių dalybos operacijas / ir %. Primename, kad / nurodo dalybos sveikąją dalį, o % – dalybos sveikąją liekaną. Pavyzdžiui, turime keturženklį skaičių *sk* = 1234. Pirmąjį skaičiaus skaitmenį pavadinkime *a*, antrąjį – *b*, trečiąjį – *c*, ketvirtąjį – *d*, t. y. *sk* = *a* * 1000 + *b* * 100 + *c* * 10 + *d*. Dabar parašykime formules, pagal kurias galima rasti šiuos skaitmenis:

*a* = *sk* / 1000
*b* = (*sk* / 100) % 10
*c* = (*sk* / 10) % 10
*d* = *sk* % 10

Atskyrus keturženklio skaičiaus *sk* skaitmenis *a*, *b*, *c*, *d*, juos reikia sujungti po du, t. y. paversti dviženkliais skaičiais *sk1* ir *sk2*. Toliau šiuos dviženklius skaičius reikia sudauginti ir gautą rezultatą lyginti su pradiniu keturženkliu skaičiumi. Jeigu gauta dviženklių skaičių *sk1* ir *sk2* sandauga lygi keturženkliam skaičiui *sk*, vadinasi, keturženklis skaičius *sk* yra vampyras, o dviženkliai skaičiai *sk1* ir *sk2* – jo „iltys“. Šiuos veiksmus detaliau panagrinėsime ir pritaikysime programoje.

### Pasiruošimas

➢ Sukurkite katalogą programos failams laikyti. Paleiskite *CodeBlocks*. Sukurkite programos failą *Darbas9.cpp,* įrašykite jį į katalogą *Darbas9*. Programai suteikite vardą `Darbas9`.

*Pastaba*. Iš pradžių parašysime programą, kuri išskaidytų skaitmenimis tik vieną keturženklį skaičių.

## 2 Keturženklio skaičiaus išskaidymas skaitmenimis

➢ Pagal nurodytas formules parašykite keturis priskyrimo sakinius:

```
// Darbas9
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int sk,                          // keturženklis skaičius
      a, b, c, d;                  // skaičiaus sk skaitmenys
  sk = 1234;
  a = sk / 1000;
  b = (sk / 100) % 10;
  c = (sk / 10) % 10;
  d = sk % 10;
  wcout << L"Skaičiaus " << sk << L"skaitmenys: "
        << a << L" " << b << L" " << c << L" " << d << endl;
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą. Ekrane matysite:

```
Skaičiaus 1234 skaitmenys: 1 2 3 4
```

## 3 Pirmojo keturženklių skaičių dešimtuko skaidymas atskirais skaitmenimis ir išvedimas į ekraną

➢ Pakeiskite priskyrimo sakinį

```
sk = 1234;
```

ciklo sakiniu, kuriame skaičiai nuo `1000` iki `1009` išskaidomi skaitmenimis.

```
// Darbas9
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int sk,                          // keturženklis skaičius
      a, b, c, d;                  // skaičiaus sk skaitmenys
  for (sk = 1000; sk <= 1009; sk = sk + 1) {
    a = sk / 1000;
    b = (sk / 100) % 10;
    c = (sk / 10) % 10;
    d = sk % 10;
    wcout << L"Skaičiaus " << sk << L"skaitmenys: "
          << a << L" " << b << L" " << c << L" " << d << endl;
  }
  return 0;
}
```



➢ Įrašykite ir įvykdykite programą. Ekrane matysite:

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

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

## 4 Išskaidytų skaitmenų jungimas po du

Išskaidžius keturženklį skaičių `sk` skaitmenimis `a`, `b`, `c`, `d`, juos reikia sujungti po du. Galimi šeši keturių skaitmenų jungimo po du variantai (junginiai):

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

*Pastaba*. Skaitmenų tvarka junginyje ir junginių eiliškumas yra visiškai nesvarbūs.

Pagal užduoties sąlygą gautus junginius reikia grupuoti po du taip, kad visi skaitmenys būtų naudojami po vieną kartą. Yra trys tokių junginių poros:

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

➢ Pakeiskite programos sakinį

```
wcout << L"Skaičiaus " << sk << L"skaitmenys: "
      << a << L" " << b << L" " << c << L" " << d << endl;
```

šiais sakiniais:

```
wcout << L"Skaičiaus " << sk << L"junginių poros: "  << endl;
wcout << a << b << L" " << c << d << endl;
wcout << a << c << L" " << b << d << endl;
wcout << a << d << L" " << b << c << endl;
```

➢ Įrašykite ir įvykdykite programą. Ją patikrinkite ciklo **`for`** pradžios ir pabaigos reikšmes pasirinkę, pavyzdžiui, `1234`. Ekrane matysite:

```
Skaičiaus 1234 junginių poros:
12 34
13 24
14 23
```

➢ Išbandykite programą su visais keturženkliais skaičiais: `1000..9999`.

## 5 Derinių sudarymas

Junginį sudaro du skaitmenys, todėl kiekvienai junginių porai galima sudaryti po 4 (2 * 2) derinius. Deriniai sudaromi sukeičiant junginyje skaitmenis vietomis. Lentelėje pateikiami visų trijų anksčiau sudarytų junginių porų deriniai:

| Pirmos junginių poros `ab` ir `cd` deriniai | Antros junginių poros `ac` ir `bd` deriniai | Trečios junginių poros `ad` ir `bc` deriniai |
|---|---|---|
| ab cd<br>ba cd<br>ab dc<br>ba dc | ac bd<br>ca bd<br>ac db<br>ca db | ad bc<br>da bc<br>ad cb<br>da cb |

➢ Pertvarkykite ankstesnę programą, kiekvieną junginių poros išvedimo sakinį pakeisdami keturiais derinių išvedimo sakiniais. Taigi programoje vietoj trijų atsiras dvylika išvedimo sakinių.

```
// Darbas9
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int sk,                              // keturženklis skaičius
      a, b, c, d;                      // skaičiaus sk skaitmenys
  for (sk = 1000; sk <= 1009; sk = sk + 1) {
    a = sk / 1000;
    b = (sk / 100) % 10;
    c = (sk / 10) % 10;
    d = sk % 10;
    wcout << L"Skaičiaus " << sk << L"junginių poros: "  << endl;
    wcout << a << b << L" " << c << d << endl;
    wcout << b << a << L" " << c << d << endl;
    wcout << a << b << L" " << d << c << endl;
    wcout << b << a << L" " << d << c << endl;
    wcout << endl;
    wcout << a << c << L" " << b << d << endl;
    wcout << c << a << L" " << b << d << endl;
    wcout << a << c << L" " << d << b << endl;
    wcout << c << a << L" " << d << b << endl;
    wcout << endl;
    wcout << a << d << L" " << b << c << endl;
    wcout << d << a << L" " << b << c << endl;
    wcout << a << d << L" " << c << b << endl;
    wcout << d << a << L" " << c << b << endl;
  }
  return 0;
}
```



➢ Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti:

```
Skaičiaus 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
```

## 6 Tikrinimas, ar skaičius yra vampyro skaičius

Visus gautus junginių derinius pavertus dviženkliais skaičiais (pavyzdžiui, `a * 10 + b` ir `c * 10 + d`), juos sudauginus ir palyginus su pradiniu skaičiumi `sk`, galima nustatyti, ar `sk` yra vampyro skaičius. Jeigu taip, tai į ekraną išvedamas skaičius ir jo „iltys“.

➢ Pašalinkite iš programos sakinį:

```
  wcout << L"Skaičiaus " << sk << L"junginių poros: "  << endl;
```

➢ Pakeiskite ankstesnėje programoje 12 išvedimo sakinių. Pavyzdžiui, pirmąjį išvedimo sakinį

```
  wcout << a << b << L" " << c << d << endl;
```

pakeiskite tokiu:

```
    if (sk == (a * 10 + b) * (c * 10 + d))
      wcout << sk << L" = " << a << b << L" * " << c << d << endl;
```

Nurodykite ciklo **`for`** pradinę reikšmę `1000`, o galutinę reikšmę – `9999`.

```
// Darbas9
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int sk,                              // keturženklis skaičius
      a, b, c, d;                      // skaičiaus sk skaitmenys
  wcout << L"Vampyro skaičiai: " << endl;
  for (sk = 1000; sk <= 9999; sk = sk + 1) {
    a = sk / 1000;
    b = (sk / 100) % 10;
    c = (sk / 10) % 10;
    d = sk % 10;
    if (sk == (a * 10 + b) * (c * 10 + d))
      wcout << sk << L" = " << a << b << L" * " << c << d << endl;
    if (sk == (b * 10 + a) * (c * 10 + d))
      wcout << sk << L" = " << b << a << L" * " << c << d << endl;
    if (sk == (a * 10 + b) * (d * 10 + c))
      wcout << sk << L" = " << a << b << L" * " << d << c << endl;
```

```
    if (sk == (b * 10 + a) * (d * 10 + c))
      wcout << sk << L" = " << b << a << L" * " << d << c << endl;
    if (sk == (a * 10 + c) * (b * 10 + d))
      wcout << sk << L" = " << a << c << L" * " << b << d << endl;
    if (sk == (c * 10 + a) * (b * 10 + d))
      wcout << sk << L" = " << c << a << L" * " << b << d << endl;
    if (sk == (a * 10 + c) * (d * 10 + b))
      wcout << sk << L" = " << a << c << L" * " << d << b << endl;
    if (sk == (c * 10 + a) * (d * 10 + b))
      wcout << sk << L" = " << c << a << L" * " << d << b << endl;
    if (sk == (a * 10 + d) * (b * 10 + c))
      wcout << sk << L" = " << a << d << L" * " << b << c << endl;
    if (sk == (d * 10 + a) * (b * 10 + c))
      wcout << sk << L" = " << d << a << L" * " << b << c << endl;
    if (sk == (a * 10 + d) * (c * 10 + b))
      wcout << sk << L" = " << a << d << L" * " << c << b << endl;
    if (sk == (d * 10 + a) * (c * 10 + b))
      wcout << sk << L" = " << d << a << L" * " << c << b << endl;
  }
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą. Ekrane matysite:

```
Vampyro skaičiai:
1260 = 21 * 60
1395 = 15 * 93
1435 = 41 * 35
1530 = 51 * 30
1827 = 21 * 87
2187 = 27 * 81
6880 = 86 * 80
6880 = 86 * 80
```

## Klausimai

1. Tarkime, kad `sk` yra keturženklis sveikasis skaičius. Kokie yra nurodytų priskyrimo sakinių rezultatai?
   a) `a := sk / 1000;`
   b) `b := (sk / 100) % 10;`
   c) `c := (sk / 10) % 10;`
   d) `d := sk % 10;`
2. Ką gausite bet kokį sveikąjį skaičių padaliję iš 10 ir paėmę dalybos sveikąją dalį?
3. Ką gausite bet kokį sveikąjį skaičių padaliję iš 10 ir paėmę dalybos liekaną?

## Užduotys

1. Parašykite programą, kuri nustatytų, iš kelių skaitmenų *k* sudarytas sveikasis skaičius *sk*.
   *Pasitikrinkite*. Kai *sk* = 12345, tai *k* = 5. Kai *sk* = 4, tai *k* = 1.
2. Parašykite programą, kuri nustatytų, ar sveikasis skaičius *sk* lyginis, ar nelyginis.
   *Pasitikrinkite*. Kai *sk* = 123, turi būti spausdinama: `Skaičius nelyginis`.
3. Parašykite programą, kuri nustatytų, ar sveikasis skaičius *sk* yra *palindròmas* (vienodai skaitomas iš abiejų pusių).
   *Pasitikrinkite*. Kai *sk* = 1221, turi būti spausdinama: `Skaičius yra palindromas`.
4. Parašykite programą, kuri nustatytų, ar sveikasis skaičius *sk* sudarytas vien iš lyginių, ar vien iš nelyginių skaitmenų, ar iš lyginių ir nelyginių.
   *Pasitikrinkite*. Kai *sk* = 1234, turi būti spausdinama:
   `Skaičiuje yra ir lyginių, ir nelyginių skaitmenų.`



5. Parašykite programą, kuri apskaičiuotų sveikojo skaičiaus *sk faktoriālą f*.
   *Pasitikrinkite*. Kai *sk* = 5, tai *f* = 120.

## SMALSIEMS

| Nuorodos į C++ kalbos žinyną |
|---|
| 3.11. Rezultatų (duomenų) išvedimas į failą<br>3.12. Funkcijos |

### 1 Programos struktūrizavimas naudojant funkciją

➢ Sudarykite funkciją, kuri nustatytų, ar skaičius yra vampyro skaičius. Funkciją pavadinkite vardu `Vampyras`. Funkcija turės penkis parametrus: keturženklį skaičių `sk` ir keturis jo skaitmenis `a`, `b`, `c`, `d`. Jeigu `sk` bus vampyro skaičius, tai funkcija išves šį skaičių ir jo „iltis“. Programoje į funkciją reikės kreiptis 12 kartų nurodant tinkamus argumentus. Todėl programoje kiekvieną **`if`** sakinį pakeiskite kreipiniu į funkciją `Vampyras`.

```
// Darbas9
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
//---------------------------------------------------------------------------
void Vampyras(int sk, int a, int b, int c, int d);
//---------------------------------------------------------------------------
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int sk,                          //  keturženklis skaičius
      a, b, c, d;                  //  skaičiaus sk skaitmenys
  wcout << L"Vampyro skaičiai: ";
  for (sk = 1000; sk <= 9999; sk = sk + 1) {
    a = sk / 1000;
    b = (sk / 100) % 10;
    c = (sk / 10)  % 10;
    d = sk % 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);
  }
  return 0;
}
//---------------------------------------------------------------------------
void Vampyras(int sk, int a, int b, int c, int d)
{
  if (sk == (a * 10 + b) * ( c * 10 + d))
    wcout << sk << L" = " << a << b << L" * " << c << d << endl;
}
//---------------------------------------------------------------------------
```

➢ Įrašykite ir įvykdykite programą. Ekrane matysite tokį pat rezultatą, kaip ir ankstesnės programos versijos.

### 2 Programos rezultatų išvedimas į rezultatų failą

➢ Atlikite tokius programos pakeitimus:

- perkelkite į programą antraštinį failą `fstream`;
- sugalvokite rezultatų failo vardą, pavyzdžiui, *Vampyrai.txt*;
- parenkite failą įrašymui;
- papildykite funkcijos `Vampyras` parametrų sąrašą dar vienu kintamuoju (pavyzdžiui, `ofstream & fr,`), užrašydami jį pirmiausia;
- išvedimo sakinį `wcout << L"Vampyro skaičiai: ";` pakeiskite sakiniu
  `fr << "Vampyro skaičiai: ";`
- funkcijoje `Vampyras` išvedimo sakinį `wcout` pakeiskite sakiniu
  `fr << sk << " = " << a << b << " * " << c << d << endl;`
- užverkite tekstinį failą;
- parašykite sakinį
  `wcout << L"Rezultatai yra faile Vampyrai.txt" << endl;.`

➢ Įrašykite ir įvykdykite programą. Ekrane turėtų būti tik tokia eilutė:

```
Rezultatai yra faile Vampyrai.txt
```

Apskaičiuoti rezultatai bus rezultatų faile. Jį atverti galima naudojantis meniu komandomis: *File* → *Open...*

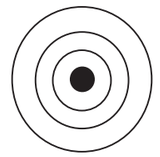


# 2.10. Taikinys

Atlikdami šį darbą, įtvirtinsite sąlyginio sakinio užrašymo įgūdžius. Sužinosite, kaip nustatyti, ar du realieji skaičiai lygūs.

| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.4. Duomenų įvedimas klaviatūra<br>3.5. Rezultatų (duomenų) išvedimas į ekraną<br>3.8. Sąlyginis sakinys **`if`**<br>3.9. Knygoje naudojamų ir / ar rekomenduojamų matematinių funkcijų sąrašas | 4.3. Šakotieji skaičiavimai |

## Užduotis

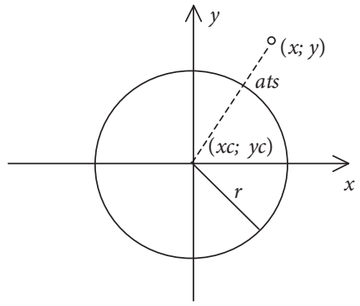


Vyksta šaudymo iš lanko varžybos. Taikinys pritvirtintas prie lentos, turinčios elektroninius daviklius. Lentos apatinio kairiojo kampo koordinatės yra (0; 0). Taikinio centras pažymėtas juodu skrituliu. Aplink jį yra nubrėžti dar trys apskritimai. Strėlei pataikius į skirtingas taikinio vietas, yra skiriamas skirtingas skaičius taškų. Reikia parašyti programą, kuri, atsižvelgdama į strėlės pataikymo vietą, skirtų sportininkui taškus. Yra žinomi taikinio duomenys: taikinio centro koordinatės (*xc*; *yc*), apskritimų spinduliai, strėlės pataikymo taško koordinatės (*x*; *y*) ir taškų vertės. Jeigu strėlė pataikė į apskritimo liniją, tuomet sportininkas gauna pusę taškų, kurie būtų skiriami, jeigu strėlė pataikytų į apskritimo vidų.

## Algoritmas

Jei strėlės pataikymo taškas nuo apskritimo centro nutolęs atstumu *ats* < *r* (čia *r* – apskritimo spindulys), tuomet pataikymo taškas yra apskritimo viduje, jei *ats* = *r* – pataikymo taškas yra ant apskritimo, jei *ats* > *r* – pataikymo taškas yra apskritimo išorėje.

Pavaizduokime situaciją grafiškai:




Atstumas *ats* tarp dviejų taškų skaičiuojamas pagal formulę

$$ats = \sqrt{(xc-x)^2 + (yc-y)^2}$$

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. Į kompiuterio atmintinę realieji skaičiai įrašomi tam tikru tikslumu, todėl skaičiai laikomi lygiais tuomet, kai skirtumas tarp jų yra nedidelis. Pavyzdžiui, du realieji skaičiai yra lygūs, jeigu pirmieji penki skaitmenys po kablelio yra vienodi. Daugeliu atvejų atliekant skaičiavimus su realiaisiais skaičiais būtina žinoti, kokiu tikslumu reikia skaičiuoti.

### 1 Pasiruošimas

➢ Sukurkite katalogą programos failams laikyti, paleiskite *CodeBlocks* ir sukurkite programos failą *Darbas10.cpp.*

*Pastaba*. Uždavinį suskaidysime į dalis:

- pradinių duomenų skaitymas;
- atstumo `ats` skaičiavimas;
- tikrinimas, kur pataikė strėlė.

## 2 Pradinių duomenų skaitymas

➢ Aprašykite kintamuosius pradiniams duomenims atmintyje laikyti ir užrašykite pradinių duomenų skaitymo sakinius.

```
// Darbas10
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  double         xc, yc,           // taikinio centro koordinatės
         r1, r2, r3, r4;           // taikinio apskritimų spinduliai didėjimo tvarka
  int    t1, t2, t3, t4;           // skiriami taškai atsižvelgiant į apskritimus
  double           x, y;           // strėlės pataikymo taško koordinatės
  // Duomenų skaitymas
  wcout << L"Taikinio centro koordinatė      xc = "; cin >> xc;
  wcout << L"Taikinio centro koordinatė      yc = "; cin >> yc;
  wcout << L"Pirmojo apskritimo spindulys    r1 = "; cin >> r1;
  wcout << L"Pirmojo apskritimo taškai       t1 = "; cin >> t1;
  wcout << L"Antrojo apskritimo spindulys    r2 = "; cin >> r2;
  wcout << L"Antrojo apskritimo taškai       t2 = "; cin >> t2;
  wcout << L"Trečiojo apskritimo spindulys   r3 = "; cin >> r3;
  wcout << L"Trečiojo apskritimo taškai      t3 = "; cin >> t3;
  wcout << L"Ketvirtojo apskritimo spindulys r4 = "; cin >> r4;
  wcout << L"Ketvirtojo apskritimo taškai    t4 = "; cin >> t4;
  wcout << L"Strėlės koordinatė               x = "; cin >> x;
  wcout << L"Strėlės koordinatė               y = "; cin >> y;
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą.

## 3 Atstumo nuo pataikymo taško iki apskritimo centro skaičiavimas

Norint nagrinėti strėlės padėtį, reikia žinoti jos pataikymo taško atstumą iki apskritimo centro.

➢ Papildykite programos kintamųjų sąrašą nauju realiojo tipo kintamuoju `ats`.

➢ Parašykite priskyrimo sakinį atstumui skaičiuoti:

```
ats = sqrt((xc - x) * (xc - x) + (yc - y) * (yc - y));
```

➢ Parašykite sakinį, kuris į ekraną išvestų apskaičiuotą atstumo `ats` reikšmę penkių skaitmenų po kablelio tikslumu.



➢ Įrašykite ir įvykdykite programą. Patikrinkite, kaip veikia programa:

```
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
```

## 4 Skaičiavimų tikslumo įvedimas

Norėdami sužinoti, kur pataikė strėlė nurodyto apskritimo atžvilgiu, turime palyginti pataikymo taško atstumą iki apskritimo centro su nagrinėjamo apskritimo spinduliu. Tam reikia pasirinkti skaičiavimų tikslumą.

➢ Aprašykite realiojo tipo kintamąjį `tiks`, skirtą skaičiavimų tikslumo reikšmei laikyti. Papildykite programą kintamojo `tiks` reikšmės įvedimo sakiniais.

➢ Įrašykite ir įvykdykite programą. Patikrinkite, kaip veikia programa:

```
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: 0.00001
```

## 5 Pataikymo taško padėties nustatymas

➢ Papildykite programą nauju sveikojo tipo kintamuoju `tsk`, skirtu sportininko gautiems taškams įsiminti. Parašykite sąlyginį sakinį, kuris nustatytų pataikymo taško padėtį apskritimų atžvilgiu ir skaičiuotų taškus.

Pirmiausia reikia patikrinti, ar pataikymo taškas yra ant apskritimo. Jeigu nurodytu tikslumu gauname atsakymą, kad jis yra ne ant apskritimo, tuomet galima tikrinti, ar jis viduje, ar išorėje. Tokia seka reikalinga tam, kad išsiaiškintume, ar skaičiai lygūs. Tikrinti pradedame nuo vidinio apskritimo:

```
if (fabs(ats - r1) < tiks)          //  ar skirtumas mažesnis už tikslumo reikšmę
  tsk = t1 / 2;                     //  ant apskritimo linijos
else if (ats < r1)                  //  apskritimo viduje
       tsk = t1;
     else if (fabs(ats - r2) < tiks)
            tsk = t2 / 2;
          else if (ats < r2)
                 tsk = t2;
               else if (fabs(ats - r3) < tiks)
                      tsk = t3 / 2;
                    else if (ats < r3)
                           tsk = t3;
                         else if (fabs(ats - r4) < tiks)
                                tsk = t4 / 2;
                              else if (ats < r4)
                                     tsk = t4;
                                   else tsk = 0;
```

- Papildykite programą `Darbas10` sakiniais, skirtais apskaičiuotų taškų reikšmei išvesti į ekraną.
- Įrašykite ir įvykdykite programą. Patikrinkite, ar ji veikia teisingai. Paruoškite tiek testinių duomenų variantų, kiek yra sportininko surinktų taškų skaičiavimo variantų: kiekvienam apskritimui, kai pataikymo taškas yra jo viduje, ant apskritimo ir jo išorėje. Jeigu programos pateikiami rezultatai sutampa su testų rezultatais, tuomet galime teigti, kad programa skaičiuoja teisingai. Tikslinga patikrinti, ar programa gerai skaičiuoja, kai apskritimas ir pataikymo taškas yra skirtingose stačiakampės koordinačių plokštumos vietose.

  *Pasitikrinkite.* Kai *xc* = 60, *yc* = 80, *r1* = 10, *t1* = 10, *r2* = 15, *t2* = 8, *r3* = 20, *t3* = 6, *r4* = 30, *t4* = 2, *tiks* = 0.00001, *x* = 75, *y* = 65, turi būti spausdinama: `tsk = 2`.

## 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, ar pirmasis didesnis už antrąjį, ar mažesnis?

## Užduotis

1. Papildykite programą, kad ji skaičiuotų taškus, kai sportininkas šovė iš lanko kelis kartus. Tam reikia strėlės pataikymo taško koordinačių, atstumo ir taškų skaičiavimus atlikti cikle. Be to, reikia aprašyti papildomą kintamąjį sportininko šūvių skaičiui atmintyje laikyti.

   *Pasitikrinkite*. Tarkime, kad *xc* = 60, *yc* = 80, *r1* = 10, *t1* = 10, *r2* = 15, *t2* = 8, *r3* = 20, *t3* = 6, *r4* = 30, *t4* = 2, *tiks* = 0.00001, *n* = 4. Tuomet, kai:
   - *x* = 75, *y* = 65, tai *tsk* = 2;
   - *x* = 65, *y* = 75, tai *tsk* = 10;
   - *x* = 70, *y* = 85, tai *tsk* = 8.

## SMALSIEMS

| Nuorodos į C++ kalbos žinyną |
|---|
| 3.10. Duomenų įvedimas iš failo<br>3.12. Funkcijos |



## Pradinių duomenų skaitymas iš failo

- Pradinius duomenis surašykite į tekstinį failą, pavyzdžiui, *TaikinysDuom.txt,* taip:

| Failas *TaikinysDuom.txt* | Paaiškinimai |
|---|---|
| `60  80`<br>`10  10`<br>`15  8`<br>`20  6`<br>`30  2`<br>`0.00001` | Taikinio centro koordinatės yra `xc` ir `yc`<br>Pirmojo apskritimo spindulys ir skiriami taškai<br>Antrojo apskritimo spindulys ir skiriami taškai<br>Trečiojo apskritimo spindulys ir skiriami taškai<br>Ketvirtojo apskritimo spindulys ir skiriami taškai<br>Skaičiavimų tikslumas |

Tai pastovūs duomenys. Derinant programą, nereikės kiekvieną kartą suvedinėti taikinio duomenų. Jeigu bus skaičiuojami ne vieno šūvio taškai, tai reikės įvesti tik strėlės pataikymo taško koordinates. Jeigu bus kitas taikinys, teks į failą duomenis surašyti iš naujo.

- Programos pradžioje įterpkite failą `fstream`.
- Aprašykite pradinių duomenų globaliuosius kintamuosius.

```
// Darbas10
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
#include <cmath>
#include <fstream>
using namespace std;
//---------------------------------------------------------------------------
//  Globalieji kintamieji
  double           xc, yc,            //  taikinio centro koordinatės
              r1, r2, r3, r4;         //  taikinio apskritimų spinduliai didėjimo tvarka
  int         t1, t2, t3, t4;         //  skiriami taškai atsižvelgiant į apskritimus
  double             x, y,            //  strėlės pataikymo taško koordinatės
                     tiks;            //  skaičiavimų tikslumas
```

- Papildykite programą funkcija duomenims iš failo skaityti, funkcijos prototipu ir kreipiniu į ją. Funkcijos tekstas turi būti po funkcija `main()`.

```
void SkaitytiDuomenis();              //  funkcijos prototipas
//---------------------------------------------------------------------------
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  SkaitytiDuomenis();                 //  duomenų skaitymas
  return 0;
}
//---------------------------------------------------------------------------
void SkaitytiDuomenis()
{
  ifstream fd("TaikinysDuom.txt");
  fd >> xc >> yc;
  fd >> r1 >> t1;
  fd >> r2 >> t2;
  fd >> r3 >> t3;
  fd >> r4 >> t4;
  fd >> tiks;
  fd.close();
}
//---------------------------------------------------------------------------
```

➢ Įrašykite ir įvykdykite programą. Patikrinkite, kaip ji veikia. Jeigu jokių klaidų nėra, tuomet galite įsitikinti, ar duomenys iš failo nuskaityti teisingai. Tam galite parašyti kitą funkciją:

```
void RodytiDuomenis()
{
  wcout << L"xc = " << setw(5) << fixed << setprecision(2) << xc
        << L"yc = " << setw(5) << fixed << setprecision(2) << yc << endl;
  wcout << L"r1 = " << setw(5) << fixed << setprecision(2) << r1
        << L"t1 = " << setw(5) << fixed << setprecision(2) << t1 << endl;
  wcout << L"r2 = " << setw(5) << fixed << setprecision(2) << r2
        << L"t2 = " << setw(5) << fixed << setprecision(2) << t2 << endl;
  wcout << L"r3 = " << setw(5) << fixed << setprecision(2) << r3
        << L"t3 = " << setw(5) << fixed << setprecision(2) << t3 << endl;
  wcout << L"r4 = " << setw(5) << fixed << setprecision(2) << r4
        << L"t4 = " << setw(5) << fixed << setprecision(2) << t4 << endl;
  wcout << L"Tikslumas: " << setw(8) << fixed << setprecision(5) << tiks << endl;
}
//---------------------------------------------------------------------------
```

Funkcija `RodytiDuomenis` rašoma po duomenų skaitymo funkcijos.

➢ Funkcijoje `main ()` parašykite kreipinį į funkciją `RodytiDuomenis();`.

➢ Įrašykite ir įvykdykite programą. Patikrinkite, kaip ji veikia. Jeigu jokių klaidų nėra, tuomet ekrane matysite iš failo nuskaitytus 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: 0.00001
```

## Atstumo nuo strėlės pataikymo taško iki apskritimo centro skaičiavimas

➢ Papildykite programos kintamųjų sąrašą nauju realiojo tipo globaliuoju kintamuoju `ats`.

➢ Parašykite funkcijoje `main()` priskyrimo sakinį atstumui skaičiuoti:

```
  ats = sqrt((xc - x) * (xc - x) + (yc - y) * (yc - y));
```

➢ Parašykite sakinį, kuris į ekraną išvestų apskaičiuotą atstumo `ats` reikšmę.

➢ Papildykite programą dialogo sakiniais, skirtais strėlės pataikymo į taikinį koordinačių reikšmėms įvesti klaviatūra.

➢ Įrašykite ir įvykdykite programą. 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: 0.00001
Strėlės koordinatė x = 75
Strėlės koordinatė y = 65
Pataikymo taško atstumas iki taikinio centro:  21.21320
```



## Sportininko surinktų taškų skaičiavimas

➢ Papildykite programą nauju sveikojo tipo kintamuoju `tsk`, skirtu taškams įsiminti. Parašykite funkciją, kurioje sąlyginis sakinys patikrintų taško padėtį apskritimų atžvilgiu ir skaičiuotų taškus:

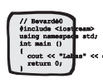
```
//---------------------------------------------------------------------------
void Taskai()
{
  if (fabs(ats - r1) < tiks)    // ar skirtumas mažesnis už tikslumo reikšmę
    tsk = t1 / 2;               // ant apskritimo linijos
  else if (ats < r1)            // apskritimo viduje
         tsk = t1;
       else if (fabs(ats - r2) < tiks)
              tsk = t2 / 2;
            else if (ats < r2)
                   tsk = t2;
                 else if (fabs(ats - r3) < tiks)
                        tsk = t3 / 2;
                      else if (ats < r3)
                             tsk = t3;
                           else if (fabs(ats - r4) < tiks)
                                  tsk = t4 / 2;
                                else if (ats < r4)
                                       tsk = t4;
                                     else tsk = 0;
}
//---------------------------------------------------------------------------
```

➢ Papildykite programą kreipiniu į funkciją apskaičiuotų taškų reikšmei išvesti. Funkcija `main()` bus tokia:

```
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  SkaitytiDuomenis();           // duomenų skaitymas
  RodytiDuomenis();             // duomenų rodymas
  wcout << L"Strėlės koordinatė x = "; cin >> x;
  wcout << L"Strėlės koordinatė y = "; cin >> y;
  ats = sqrt ((xc - x) * (xc - x) + (yc - y) * (yc - y));
  wcout << L"Pataikymo taško atstumas iki taikinio centro: "
        << setw(10) << fixed << setprecision(5) << ats << endl;
  Taskai();
  wcout << L"Gauti taškai: " << tsk << endl;
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą. Patikrinkite, kaip ji veikia. 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: 0.00001
Strėlės koordinatė x = 75
Strėlės koordinatė y = 65
Pataikymo taško atstumas iki taikinio centro:  21.21320
Gauti taškai:  2
```




## Užduotis

Taškų skaičiavimo sąlyginis sakinys yra sudėtinis. Jeigu taikinyje būtų daugiau apskritimų, sąlyginiame sakinyje reikėtų nurodyti dar daugiau sąlygų. Pagalvokite, kaip galima būtų supaprastinti taškų skaičiavimą.

# 2.11. Elektros grandinės varžos skaičiavimas

Atlikdami šį darbą, susipažinsite su ciklo cikle struktūra, prisiminsite sumavimo algoritmą.

| Nuorodos į C++ kalbos žinyną | Nuorodos į algoritmų žinyną |
|---|---|
| 3.7. Ciklo sakinys **`for`** | 4.2. Cikliniai algoritmai<br>4.4. Sumos skaičiavimo algoritmas |

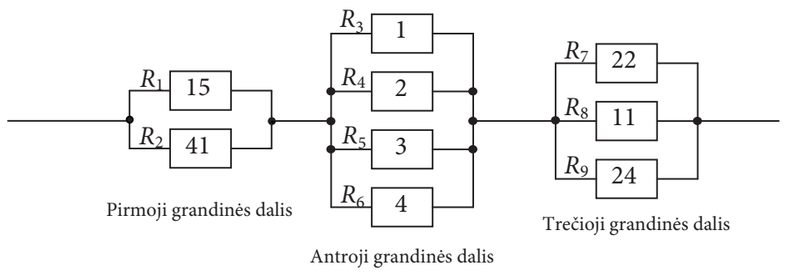
## Užduotis

Reikia parašyti programą, kuri apskaičiuotų bendrąją grandinės varžą, kai grandinę sudaro viena ar daugiau nuosekliai sujungtų grandinės dalių. Kiekviena grandinės dalis sudaryta iš dviejų ar daugiau lygiagrečiai sujungtų žinomos varžos laidininkų. Žinomas nuosekliai sujungtų grandinės dalių skaičius *n*, kiekvienos dalies lygiagrečiai sujungtų laidininkų skaičius *k* ir laidininkų varžų reikšmės *rj*.

Programa turi apskaičiuoti bendrąją grandinės varžą *R* ir ją išvesti į ekraną dviejų ženklų po kablelio tikslumu.

## Algoritmas

Tarkime, turime trijų nuosekliai sujungtų dalių ($n = 3$) grandinę:




Pirmos grandinės dalies varžų skaičius $k = 2$, varžų reikšmės yra 15 Ω ir 41 Ω. Antros grandinės dalies varžų skaičius $k = 4$, varžų reikšmės yra 1 Ω, 2 Ω, 3 Ω ir 4 Ω. Trečios grandinės dalies varžų skaičius $k = 3$, varžų reikšmės yra 22 Ω, 11 Ω ir 24 Ω.

Skaičiavimams naudojamos formulės:

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

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

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

Nuosekliai sujungtų laidininkų bendroji varža skaičiuojama pagal formulę $R = R_1 + R_2 + \ldots + R_n$, čia $R_1, R_2, R_3, \ldots, R_n$ – atskirų laidininkų varžos.

Pirmiausia reikia apskaičiuoti atskirų grandinės dalių, kuriose laidininkai sujungti lygiagrečiai, varžas. Jas susumavus, gaunama bendroji grandinės varža.

Iš pradžių apskaičiuokime vienos dalies, sudarytos iš *k* lygiagrečiai sujungtų laidininkų, varžą. Duomenys įvedami klaviatūra. Pirmiausia nurodomas laidininkų skaičius *k*, toliau įvedamos laidininkų varžos. Duomenims nuskaityti reikia naudoti ciklą **`for`**. Cikle reikia sumuoti atskirų laidininkų varžas. Nepamirškime prieš ciklą varžų sumai priskirti reikšmę, lygią 0.



### 1 Pasiruošimas

- Sukurkite katalogą programos failams laikyti. Paleiskite *CodeBlocks*. Sukurkite programos failą *Darbas11.cpp*.

### 2 Vienos dalies laidininkų varžos skaičiavimas

- Atlikite nurodytus veiksmus:
  - programos pradžioje aprašykite naudojamus kintamuosius;
  - sumos kaupimo kintamajam `L` priskirkite reikšmę, lygią 0;
  - įveskite laidininkų skaičių `k`;
  - naudodami ciklą **`for`**, įveskite ir sumuokite laidininkų varžas;
  - gautą rezultatą išveskite į ekraną.

```
// Darbas11
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int       k,            // vienos grandinės dalies varžų skaičius
            j;            // ciklo kintamasis
  double    L,            // vienos grandinės dalies laidumas
           rj;            // vieno laidininko varža
  L = 0;
  wcout << L"Laidininkų skaičius grandinės dalyje "; cin >> k;
  for (j = 1; j <= k; j = j + 1){
    wcout << L"Laidininko varža "; cin >> rj;
    L = L + 1 / rj;       // laidumų sumavimas
  }
  wcout << L"Laidininkų varža "
        << setw(6) << fixed << setprecision(2) << 1/L << endl;
  return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane matysite:

```
Laidininkų skaičius grandinės dalyje 2
Laidininko varža 15
Laidininko varža 41
Laidininkų varža 10.98
```

- Įvykdykite šią programą dar du kartus: apskaičiuokite antros ir trečios grandinės dalių varžas. Antros grandinės dalies varžą turėtumėte gauti lygią `0.48`, trečiosios – `5.62`.

### 3 Visų grandinės dalių varžos skaičiavimas

Norint apskaičiuoti visų nuosekliai sujungtų grandinės dalių bendrąją varžą, programoje reikia dar vieno ciklo **`for`**. Jis turi apgaubti ankstesnįjį ciklą **`for`**.

- Atlikite paruošiamuosius veiksmus:
  - papildykite programos kintamųjų aprašus ciklo kintamuoju `i`, nuosekliai sujungtų grandinės dalių skaičiumi `n` ir visos grandinės bendrosios varžos kintamuoju `R`;

- visos grandinės bendrosios varžos kintamajam `R` priskirkite reikšmę, lygią 0;
- įveskite grandinės dalių skaičių `n`;
- užrašykite ciklą **`for`** kartu su sakinių bloku **`{`** ir **`}`**, kuris apgaubs ankstesnįjį (vidinį) ciklą **`for`**;
- vidiniam ciklui priklausančius sakinius pastumkite į dešinę – programa taps vaizdesnė ir suprantamesnė;
- prieš baigdami išorinį ciklą užrašykite sakinį `R = R + 1/L;`, kuris susumuos atskirų lygiagrečiai sujungtų grandinės dalių varžas, apskaičiuotas vidiniame cikle;
- rezultato išvedimo sakinyje nurodykite kintamojo `R` išvedimo formatą.

```
// Darbas11
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int       n,                  // grandinės dalių skaičius
            k,                  // vienos grandinės dalies varžų skaičius
          j, i;                 // ciklo kintamieji
  double    L,                  // vienos grandinės dalies laidumas
           rj,                  // vieno laidininko varža
            R;                  // bendroji grandinės varža
  R = 0;
  wcout << L"Kelios dalys sudaro elektros grandinę? "; cin >> n;
  for (i = 1; i <= n; i = i + 1) {
    L = 0;
    wcout << L"Laidininkų skaičius grandinės dalyje "; cin >> k;
    for (j = 1; j <= k; j = j + 1){
      wcout << L"Laidininko varža "; cin >> rj;
      L = L + 1 / rj;          // laidumų sumavimas
    }
    R = R + 1 / L;
  }
  wcout << L"Bendroji grandinės varža "
        << setw(6) << fixed << setprecision(2) << R << endl;
  return 0;
}
```

➢ Įrašykite ir įvykdykite programą. Įveskite pradinius duomenis ir pasitikrinkite, ar gavote teisingą rezultatą:

```
Kelios dalys sudaro elektros grandinę? 3
Laidininkų skaičius grandinės dalyje 2
Laidininko varža 15
Laidininko varža 41
Laidininkų skaičius grandinės dalyje 4
Laidininko varža 1
Laidininko varža 2
Laidininko varža 3
Laidininko varža 4
Laidininkų skaičius grandinės dalyje 3
Laidininko varža 22
Laidininko varža 11
Laidininko varža 24
Bendroji grandinės varža 17.08
```



## Klausimai

1. Kokius paruošiamuosius darbus reikia atlikti prieš skaičiuojant sumą?
2. Kokio tipo kintamieji naudojami sumos reikšmei laikyti atmintinėje?
3. Kur programoje reikėtų įrašyti sumos reikšmės išvedimo sakinį?
4. Kada naudojama ciklo cikle struktūra?

## SMALSIEMS

| Nuorodos į C++ kalbos žinyną |
|---|
| 3.10. Duomenų įvedimas iš failo<br>3.11. Rezultatų (duomenų) išvedimas į failą<br>3.12. Funkcijos |

### 1 Pradinių duomenų įvedimas iš failo *Duomenys.txt*

➢ Sukurkite tekstinį failą *Duomenys.txt* ir į jį įrašykite pradinius duomenis:

| Failas *Duomenys.txt* | Paaiškinimai |
|---|---|
| `3`<br>`2 15 41`<br>`4 1 2 3 4`<br>`3 22 11 24` | Nuosekliai sujungtų grandinės dalių skaičius<br>Pirmos grandinės dalies varžų skaičius ir $R_1$ bei $R_2$ reikšmės<br>Antros grandinės dalies varžų skaičius ir $R_3$, $R_4$, $R_5$, $R_6$ reikšmės<br>Trečios grandinės dalies varžų skaičius ir $R_7$, $R_8$, $R_9$ reikšmės |

➢ Papildykite programą pradinių duomenų failo kintamuoju `fd`.

➢ Paruoškite failą duomenims skaityti.

➢ Ištrinkite dialogo sakinius:

```
  wcout << L"Kelios dalys sudaro elektros grandinę? ";
  wcout << L"Laidininkų skaičius grandinės dalyje ";
  wcout << L"Laidininko varža ";
```

➢ Pakeiskite rašymo sakinius.

➢ Baigę skaityti, užverkite pradinių duomenų failą.

```
// Darbas11
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int       n,                  // grandinės dalių skaičius
            k,                  // vienos grandinės dalies varžų skaičius
          j, i;                 // ciklo kintamieji
  double    L,                  // vienos grandinės dalies laidumas
           rj,                  // vieno laidininko varža
            R;                  // bendroji grandinės varža
  R = 0;
  ifstream fd("Duomenys.txt");
  fd >> n;
```

```
  for (i = 1; i <= n; i = i + 1) {
    L = 0;
    fd >> k;
    for (j = 1; j <= k; j = j + 1) {
      fd >> rj;
      L = L + 1 / rj;        // laidumų sumavimas
    }
    R = R + 1 / L;
  }
  fd.close();
  wcout << L"Bendroji grandinės varža "
        << setw(6) << fixed << setprecision(2) << R << endl;
  return 0;
}
```

Įrašykite ir įvykdykite programą. Jei viską atlikote teisingai, ekrane turėtumėte matyti:

```
Bendroji grandinės varža 17.08
```

## 2 Rezultato įrašymas į failą *Rezultatai.txt*

➢ Pakeiskite rezultato išvedimo sakinį

```
  wcout << L"Bendroji grandinės varža "
        << setw(6) << fixed << setprecision(2) << R << endl;
```

tokiais sakiniais:

```
  ofstream fr("Rezultatai.txt");
  fr << setw(6) << fixed << setprecision(2) << R <<endl;
  fr.close();
```

➢ Įrašykite ir įvykdykite programą.

Atvėrę rezultatų failą komanda *Open...*, pamatysite skaičių `17.08`. Vadinasi, pagal užduotį parašyta programa atlieka visus skaičiavimus teisingai ir rezultatus pateikia ten, kur ir reikia, t. y. rezultatų faile.

## 3 Programos struktūrizavimas naudojant funkciją su parametrais–nuorodomis

➢ Sudarykite funkciją, kuri apskaičiuotų vienos grandinės dalies lygiagrečiai sujungtų laidininkų varžą, t. y. vidinę ciklo cikle dalį aprašykite funkcija. Ją pavadinkite `RastiRGrandies()`. Funkcijai reikia pateikti atverto duomenų srauto kintamąjį `fd`. Prieš jį būtina parašyti `&`. Nepamirškite likusiame pagrindinės funkcijos `main()` cikle parašyti kreipinį į šią funkciją.

```
// Darbas11
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;
//-----------------------------------------------------------------------------
void RastiRGrandies(ifstream & fd, double & R);
//-----------------------------------------------------------------------------
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  int       n,             // grandinės dalių skaičius
            i;             // ciklo kintamasis
  double    gr,            // vienos grandies varža
            R;             // bendroji grandinės varža
```



```
  ifstream fd("Duomenys.txt");
  fd >> n;
  R = 0;
  for (i = 1; i <= n; i++){
    RastiRGrandies(fd, gr);
    R = R + gr;
  }
  fd.close();
  wcout << L"Bendroji grandinės varža "
        << setw(6) << fixed << setprecision(2) << R << endl;
  return 0;
}
//-----------------------------------------------------------------------------
// Apskaičiuoja lygiagrečiai sujungtų laidininkų varžą
// fd – duomenų srauto vardas
// R – apskaičiuota varža
void RastiRGrandies(ifstream & fd, double & R)
{
  int
            k,             // vienos grandinės dalies varžų skaičius
            j;             // ciklo kintamasis
  double    L,             // vienos grandinės dalies laidumas
           rj;             // vieno laidininko varža
  L = 0;
  fd >> k;
  for (j = 1; j <= k; j = j + 1){
    fd >> rj;
    L = L + 1 / rj;        // laidumų sumavimas
  }
  R = 1 / L;
}
```

➢ Įrašykite ir įvykdykite programą. Ekrane matysite tokį pat rezultatą, kaip ir ankstesnės programos versijos.

## 4 Programos struktūrizavimas naudojant funkciją, grąžinančią reikšmę per funkcijos vardą

➢ Pabandykite sukurtą funkciją `RastiRGrandies()` pakeisti funkcija `RGrandies()`:

```
double RGrandies(ifstream & fd)
{
  int
            k,             // vienos grandinės dalies varžų skaičius
            j;             // ciklo kintamasis
  double    L,             // vienos grandinės dalies laidumas
           rj,             // vieno laidininko varža
            R;
  L = 0;
  fd >> k;
  for (j = 1; j <= k; j = j + 1){
    fd >> rj;
    L = L + 1 / rj;        // laidumų sumavimas
  }
  R = 1 / L;
  return R;
}
```

➢ pagrindinės funkcijos `main()` cikle **`for`** įrašykite sakinį

```
R = R + RGrandies(fd);
```

➢ Įrašykite ir įvykdykite programą. Ekrane matysite tokį pat rezultatą, kaip ir ankstesnės programos versijos.

## Užduotys

1. Tekstinio failo *Mokiniai.txt* pirmoje eilutėje įrašytas klasės mokinių skaičius *n*. Tolesnėse eilutėse surašyti kiekvieno mokinio metiniai pažymiai (po 5 pažymius). Vieno mokinio pažymiai pateikti vienoje eilutėje ir vienas nuo kito atskirti tarpais. Parašykite programą, kuri apskaičiuotų kiekvieno mokinio pažymių aritmetinį vidurkį dviejų ženklų po kablelio tikslumu ir į rezultatų failą *Vidurkiai.txt* įrašytų mokinių numerius bei pažymių vidurkius.

   *Pasitikrinkite*. Pradinių duomenų ir rezultatų tekstiniuose failuose turėtų būti įrašyta:

| *Mokiniai.txt* | *Vidurkiai.txt* |
|---|---|
| 3<br>7 8 9 7 8<br>5 7 9 5 7<br>6 8 7 9 7 | 1 7.80<br>2 6.60<br>3 7.40 |

2. Tekstinio failo *Troleibusai.txt* pirmoje eilutėje įrašytas miesto troleibusų skaičius *n*. Kitose eilutėse įrašyta informacija apie kiekvieną troleibusą: pirmasis skaičius rodo, keliose stotelėse troleibusas sustoja, o tolesni – atstumus (metrais) tarp gretimų troleibuso maršruto stotelių. Parašykite programą, kuri apskaičiuotų ir į rezultatų failą *Marsrutai.txt* įrašytų kiekvieno troleibuso maršruto ilgį.

   *Pasitikrinkite*. Pradinių duomenų ir rezultatų tekstiniuose failuose turėtų būti įrašyta:

| *Troleibusai.txt* | *Marsrutai.txt* |
|---|---|
| 3<br>3 200 500<br>5 200 200 150 300<br>4 150 200 150 | 700<br>850<br>500 |



# 3 C++ KALBOS ŽINYNAS

Danų mokslininkas Bjornas Stroustrupas (*Bjarne Stroustrup*), sukūręs struktūrinio programavimo kalbą *C++*.
Gimė 1950 m.

Programavimo kalbos *C++* pirmtakas buvo programavimo kalba *C*, kurią 1972 m. sukūrė Kenas Tompsonas (*Ken Thompson*) ir Denis Ričis (*Dennis M. Ritchie*). Iš pradžių *C* kalba buvo tik sisteminio programavimo darbinis instrumentas ir nepretendavo į platųjį pritaikymą. Vėliau dėl šios kalbos visapusiškumo, ji tapo populiaria taikomojo ir sisteminio programavimo kalba. *C* kalba – tai procedūrinio programavimo kalba. Ji yra nepriklausoma nuo kompiuterio techninių charakteristikų ir operacinės sistemos, patogi, raiški, lanksti, turinti paprastą sintaksę.

1983 m. Bjornas Stroustrupas (*Bjarne Stroustrup*) sukūrė *C++* programavimo kalbą, skirtą objektinio programavimo technologijai. *C++* kalba sudaryta taip, kad *C* kalba išliktų jos poaibiu. Todėl daugelis programų, parašytų *C* kalba, yra ir *C++* kalbos programomis. Nuo 1990 m. *C++* kalba yra viena populiariausių programavimo kalbų.

*C++* kalba neturi versijų. 1998 m. patvirtintas *C++* kalbos ANSI/ISO standartas. Jis apibrėžia kalbos pagrindus ir jos standartines bibliotekas. 2003 m. patvirtintas kitas kalbos standartas – Standard *C++*.

## 3.1. Kintamasis, kintamojo reikšmė

Programos atlieka veiksmus su duomenimis (pvz.: sveikaisiais, realiaisiais skaičiais; simboliais; simbolių eilutėmis). Duomenys gali būti pastovūs (pvz., gimimo data) ir kintami (pvz., amžius). Duomenys, kurie atliekant programą nekinta, vadinami ***konstàntomis***.

***Kintamíeji*** skirti duomenims, kurių reikšmės keičiasi atliekant programą, atmintyje laikyti. Jų ***vardaĩ*** sudaromi iš raidžių ir skaitmenų, tačiau pirmas ženklas turi būti raidė. Parenkant kintamųjų vardus reikia stengtis, kad jie pažymėtų duomenų paskirtį. Pavyzdžiui, jei skaičiuojame trijų skaičių sumą, tai juos atitinkančius kintamuosius patogu žymėti `a`, `b`, `c`, o sumą – vardu `suma`. Kintamieji gali būti aprašomi bet kurioje programos vietoje, bet prieš juos panaudojant veiksmuose. Dažniausiai jie aprašomi funkcijos `main()` pradžioje. Kiekvienam kintamajam reikia nurodyti, kokio tipo duomenis jis turi laikyti atmintyje. Duomenų tipas apibrėžia tam tikrą aibę reikšmių ir veiksmų, kuriuos galima atlikti su tomis reikšmėmis.

Lentelėje pateikiami dažniausiai naudojami duomenų tipai:

| Tipo pavadinimas | Galimų reikšmių intervalas |
|---|---|
| `int` | –2147483648 .. 2147483647 |
| `double` | $-2.23\times10^{-308}$ .. $1.79\times10^{308}$ |
| `char` | Vienas simbolis, pavyzdžiui: `'A'`, `'8'`, `'+'` arba skaičius –128 .. 127 |
| `bool` | true (1), false (0) |

Kintamieji aprašomi taip:

```
duomenųTipas kintamojoVardas;
```

Kintamojo `duomenųTipą` nustato programuotojas. Jeigu yra keletas to paties tipo kintamųjų, tuomet juos galima išvardyti viename sakinyje, atskiriant vardus kableliu.

Kintamųjų aprašymo pavyzdys:

```
double a, b, c, y;   // realiojo tipo kintamieji
int i, k, a2;        // sveikojo tipo kintamieji
char simb;           // simbolinio tipo kintamasis
```

Vykdant programą, kiekvienam kintamajam kompiuterio atmintyje skiriama tiek vietos, kiek reikia nurodyto tipo duomenims laikyti. Kintamieji atmintyje gali laikyti tik nurodyto tipo duomenų ***réikšmes***. Kintamieji reikšmes (duomenis) gauna duomenų ***įvedìmo*** (skaitymo) ***sakiniaĩs***, kai kintamiesiems reikšmės suteikiamos imant jas iš išorinių įrenginių (klaviatūros, diskinių ir kitų ilgalaikės atminties įrenginių), arba ***priskyrìmo sakiniaĩs***, kai kintamiesiems reikšmės suteikiamos programos tekste. Pavyzdžiui, kintamiesiems reikšmes galima priskirti jų aprašymo sakiniuose taip:

```
duomenųTipas kintamojoVardas = pradinėReikšmė;
```

Pavyzdžiui:

```
double pi = 3.1415;
char a = 'A';
int k = 5 / 3;
bool status = false;
int k = 5, b = 15;
```

Kintamojo galiojimo pradžia – jo aprašymo vieta. Kintamasis galioja tame programos bloke, kuriame jis yra aprašytas. Bloką sudaro programavimo kalbos sakinių seka, parašyta tarp riestinių skliaustų `{}`. Jis skiriasi nuo sudėtinio sakinio tuo, kad jame yra ir sakiniai, ir kintamųjų aprašai. Dažniausiai blokas naudojamas funkcijoms užrašyti.

***Sudėtinis sakinys*** – tai tarp riestinių skliaustų `{ }` parašyta sakinių seka.

Sudėtinio sakinio pavyzdys:

```
{
   n = n + 1;
   suma = suma + n;
}
```

Aprašant kintamuosius, reikia prisiminti:

- Kintamieji, aprašyti prieš pagrindinę funkciją `main()`, vadinami ***globaliaisiais***. Jie galioja visoje programoje.
- Kintamieji, aprašyti funkcijoje, vadinami ***lokaliaisiais***. Jie galioja tik toje funkcijoje, išėjus už funkcijos ribų, šių kintamųjų reikšmės neišsaugomos.
- Jei kintamasis tokiu pačiu vardu aprašytas prieš pagrindinę funkciją `main()` ir funkcijoje, pirmenybė suteikiama lokaliajam, t. y. funkcijoje globalusis kintamasis negalioja.

## 3.2. Priskyrimo sakinys

***Priskyrimo sakinys*** naudojamas, kai kintamajam reikia suteikti reikšmę programos tekste. Priskyrimo sakinio struktūra tokia:

```
kintamojoVardas = Reiškinys;
```

Čia simbolis = žymi priskyrimo operaciją, o `Reiškinys` nurodo, kokius veiksmus, kokia tvarka ir su kokiais argumentais reikia atlikti. Kairiojoje priskyrimo operacijos pusėje įrašytas kintamojo vardas nurodo, kam atmintyje suteikiama apskaičiuota reiškinio reikšmė. Pavyzdžiui, priskyrimo sakinys `y = x * x;` reiškia, kad argumento `x` reikšmė keliama kvadratu ir rezultatas priskiriamas kintamajam `y`.

Būtina, kad priskyrimo operacijos kairėje pusėje nurodyto kintamojo tipas atitiktų reiškinio, esančio dešiniojoje pusėje, reikšmės tipą.

Reiškiniuose galima naudoti tik tokius kintamuosius, kurių reikšmės apibrėžtos anksčiau įrašytuose programos sakiniuose.

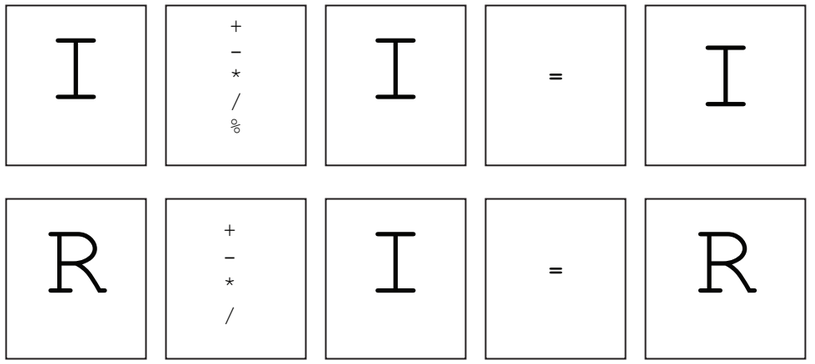
Skiriami trys reiškinių tipai:

- ***Aritmėtinis***. Tai reiškinys, kuriame kintamieji ir konstantos jungiamos aritmetinių operacijų ženklais (+, –, *, /, sveikųjų skaičių dalmens liekana `%`). Tokio reiškinio skaičiavimo rezultatas yra skaičius.
  Pavyzdžiui:
  ```
  a = 5; b = 7; c = 9;
  s = a * b – c;
  ```



Rezultato tipas priklauso nuo jį sudarančių argumentų tipų ir operacijų, atliekamų su argumentais. Jei visi argumentai yra sveikojo tipo, tai visų operacijų, atliekamų su šiais argumentais, rezultatas yra sveikojo tipo. Tuomet dalybos operacijos (ji žymima `/`) rezultatas visada yra sveikoji gautos dalybos dalis.

Schemoje pavaizduoti dviejų tipų argumentai, su jais atliekamos operacijos ir rezultatas (`I` žymi sveikąjį duomenų tipą, `R` – realųjį).




- ***Sántykio***. Tai du aritmetiniai reiškiniai, tarp kurių rašomas santykio operacijos ženklas (>, <, <=, >=, !=, ==). Tokio reiškinio rezultatas yra loginė reikšmė `true` (tiesa) arba `false` (netiesa). Pavyzdžiui,
  ```
  k = 5 != 7 – 2;
  ```
  Atlikus šį priskyrimo sakinį kintamojo `k` reikšmė lygi `false`, nes reiškinys `5 ≠ 5` yra neteisingas.
- ***Lòginis***. Tai reiškinys, kuriame naudojami loginių operacijų ženklai. C++ jie žymimi ženklų grupėmis `&&` (IR), `||` (ARBA), `!` (NE). Tokio reiškinio rezultatas yra loginė reikšmė `true` arba `false`.

  Pavyzdžiui:

  ```
  a = true; b = false;
  a = !a || b;
  ```

  Pateikiame loginių reiškinių teisingumo lentelę.

| Kintamųjų reikšmės | | Loginių reiškinių rezultatai | | |
|---|---|---|---|---|
| a | b | a && b | a \|\| b | ! a |
| true | true | true | true | false |
| true | false | false | true | false |
| false | true | false | true | true |
| false | false | false | false | true |

Priskyrimo sakinio rašymo forma tokia pati, kaip ir lygybės išraiškos, tačiau šios struktūros prasmė iš esmės skiriasi. Lygybėmis patvirtinamas tam tikras faktas. Tuo tarpu priskyrimo sakiniai aprašo procesus, kuriems būdinga reikšmių kaita, 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

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

Pavyzdžiui, atlikus sakinių seką

```
a = 5;
a = a + 1;
```

kintamojo `a` reikšmė lygi `6`.

## 3.3. Įvedimo ir išvedimo srauto samprata

Kalboje C++ įvedimo / išvedimo sistema yra susijusi su srauto sąvoka. ***Įvedimo / išvedimo srautas*** – tai loginis įrenginys, kuris išveda ir priima naudotojo informaciją. Įvedimo / išvedimo sistema susieja srautą su fiziniu įrenginiu (klaviatūra, ekranu, diskų įtaisais). Visi įvedimo / išvedimo srautai veikia vienodai nepriklausomai nuo fizinių įrenginių charakteristikų.

Paleidus C++ programą, automatiškai atidaromi keturi srautai:

`cin` – įvedimas (klaviatūra);
`cout` – išvedimas (ekranas);
`cerr` – pranešimai apie klaidas (ekranas);
`clog` – buferizuota `cerr` versija (ekranas).

Paprastai šie srautai yra naudojami darbui su klaviatūra ir ekranu.

Darbo priemonės su įvedimo / išvedimo srautais yra saugomos faile `iostream`. Šiame faile apibrėžti *klasių* rinkiniai, kurie leidžia atlikti įvedimo / išvedimo operacijas.

***Klasė*** – tai darinys, kuriame saugomi kintamieji ir ***metodai*** – funkcijos, taikomos darbui su klasės kintamųjų reikšmėmis.

Antraštinio failo `iostream` priemonės perkeliamos į programą sakiniu:

```
#include <iostream>
```

Duomenims nuskaityti iš srauto naudojamas operatorius >>, išvesti į srautą – operatorius <<.

## 3.4. Duomenų įvedimas klaviatūra

Duomenims įvesti iš standartinio įvedimo srauto `cin`, paprastai susieto su klaviatūra, ir / arba iš srauto, susieto su failu, dažniausiai naudojamas operatorius >>. Asmeniniuose kompiuteriuose standartinis įvedimo įtaisas yra klaviatūra.

Duomenų įvedimo procesas vykdomas dviem etapais: pirmiausia duomenys įvedami klaviatūra, po to jie priskiriami kintamiesiems. Iš pradžių įvedimo operatorius >> stabdo programos darbą ir laukia, kol klaviatūra įvedami duomenys. Duomenys kaupiami specialioje atmintyje (buferyje) ir vaizduojami ekrane. Įvedimo metu jie gali būti taisomi įprastinėmis teksto redagavimo priemonėmis. Spustelėjus įvedimo klavišą `Enter` (↵), buferyje laikomi duomenys paskirstomi įvedimo srauto `cin` sąraše išvardytiems kintamiesiems, kurie nurodo, kiek kokio tipo duomenų reikia įvesti ir kokia tvarka juos paskirstyti programos kintamiesiems.

Pavyzdžiui, jei buvo sakinys `cin >> a >> b;`, tačiau įvestas tik vienas skaičius `25`↵, kintamajam `a` bus suteikta reikšmė `25` ir kompiuteris lauks, kol bus įvesta kintamojo `b` reikšmė. Jei, vykdant sakinį `cin >> a >> b;`, įvestos trys reikšmės `25 -5 10`↵, tai kintamiesiems `a` ir `b` bus paskirstytos pirmosios dvi reikšmės, o trečioji liks buferyje ir lauks naujo skaitymo veiksmo. Įvedant duomenis viena eilute, reikšmės atskiriamos tarpais.

Prieš kreipiantis į įvedimo srautą `cin`, patariama rašyti išvedimo sakinį, kuris paaiškintų, kokių duomenų reikia programai. 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:

```
cout << "Įveskite mokinio svorį: ";  cin >> sv;
```

```
Įveskite mokinio svorį: 40
```

Ši užklausa rodo, kad kompiuteris ekrane turi suformuoti pranešimą `Įveskite mokinio svorį` ir laukti, kol klaviatūra bus įvestas skaičius (svoris).

## 3.5. Rezultatų (duomenų) išvedimas į ekraną

Duomenims išvesti į standartinį išvedimo srautą `cout`, paprastai susietą su ekranu, ir / arba į srautą, susietą su failu, dažniausiai naudojamas operatorius <<.

Išvedamų duomenų sąrašuose galima nurodyti kintamuosius, konstantas, reiškinius ir tekstinius duomenis.



Paaiškinimus būtina rašyti tarp kabučių. Pavyzdžiui, užrašius sakinį

```
cout << "Sveiki! Jūsų programa pradeda darbą!" << endl;
```

į ekraną išvedamas toks tekstas:

```
Sveiki! Jūsų programa pradeda darbą!
```

Antrasis argumentas `endl` yra konstanta, kurios reikšmė yra simbolis `'\n'` – perėjimas į naują eilutę. Šį simbolį galima įterpti į išvedamą tekstą. Pavyzdžiui, sakinys

```
cout << "Sveiki! Jūsų programa pradeda darbą!\n";
```

ekrane pateiks tą patį rezultatą:

```
Sveiki! Jūsų programa pradeda darbą!
```

Išvedant duomenis, galima nurodyti jų išvedimo formatą. Tam naudojami ***manipuliatoriai*** – specialūs nurodymai, kurie įterpiami į išvedimo srautą. Manipuliatoriai yra dviejų tipų – su argumentais ir be jų. Norint pasinaudoti manipuliatoriais su argumentais, būtina perkelti į programą priemones, kurios yra antraštiniame faile `iomanip`, tokiu sakiniu:

```
#include <iomanip>
```

Lentelėje pateiktas dažniausiai naudojamų manipuliatorių sąrašas.

| Manipuliatorius | Paskirtis |
|---|---|
| `left` | Išvedant duomenis, jie lygiuojami pagal kairįjį kraštą |
| `right` | Išvedant duomenis, jie lygiuojami pagal dešinįjį kraštą |
| `endl` | Į srautą nusiunčia eilučių skyriklio simbolį `'\n'` ir išvalo buferį |
| `setw(int n)` | Nustato išvedamų duomenų lauko plotį `n` |
| `setprecision(int n)` | Nustato realiojo skaičiaus išvedamų skaitmenų skaičių `n`.<br>Jeigu skaičius sveikojoje dalyje turi daugiau skaitmenų, tuomet jis vaizduojamas standartine išraiška, kurioje yra daugiklis $10^n$. Laipsnio pagrindas žymimas simboliu `e` ir po jo rašomas laipsnio rodiklis. Pavyzdžiui, užrašas `3.1416e+006` atitinka skaičių $3.14 \cdot 10^6$.<br>Jeigu prieš manipuliatorių `setprecision` rašomas manipuliatorius `fixed`, tuomet `n` nustato, kiek realiojo skaičiaus trupmeninės dalies skaitmenų reikia išvesti, įskaitant ir nulius skaičiaus gale, pavyzdžiui, `3141592653.400` |

Išvedimo sraute galima nurodyti, kiek pozicijų ekrane skirti atskiriems duomenims. Nurodymai rašomi prieš išvedamus duomenis. Pavyzdžiui:

```
int a = 45;
double b = 123.258;
cout << a;
cout << setw(5) << a << " "
     << fixed << setprecision(2) << b;
```

Ekrane matysime:

```
45␣␣␣45␣1.26␣␣␣␣␣␣␣␣␣
```

Realaus skaičiaus trupmeninei daliai išvesti nurodytų pozicijų skaičius per mažas, todėl buvo išvesti tik du skaitmenys po kablelio, paskutinis – suapvalintas.

## 3.6. Ciklo sakinys `while`

***Cìklas*** yra naudojamas pasikartojantiems veiksmams atlikti. Labai dažnai veiksmai kartojami tol, kol tenkinama nurodyta sąlyga. Tokiais atvejais naudojamas ciklo sakinys **`while`**. Jeigu reikia kartoti kelis sakinius, tai jie rašomi tarp `{` ir `}`:

| | |
|---|---|
| **`while`** (*Sąlyga*)<br>  *KartojamasSakinys*`;` | **`while`** (*Sąlyga*) `{`<br>  *KartojamiSakiniai*`;`<br>`};` |

***Sąlyga*** – tai bet koks loginis ar santykio reiškinys, *KartojamasSakinys* – bet koks sakinys (taip pat ir ciklo).

Vykdant sakinį **`while`** pirmiausia patikrinama *Sąlyga*: jei jos reikšmė yra `true`, tuomet atliekamas *KartojamasSakinys*, priešingu atveju jis praleidžiamas ir atliekami toliau už ciklo esantys sakiniai. Jei kartoti reikia ne vieną sakinį, rašomas sudėtinis sakinys `{ }`.

Štai keletas ciklo **`while`** pavyzdžių.

### 1 pavyzdys

```
x = 10;
while (x < 15)
  x = x + 1;
```

| Ciklo žingsniai | Sąlyga: $x < 15$ | | Kintamojo $x$ reikšmė: $x = x + 1$ |
|---|---|---|---|
| 1 | 10 < 15 | true | $x = 10 + 1 = 11$ |
| 2 | 11 < 15 | true | $x = 11 + 1 = 12$ |
| 3 | 12 < 15 | true | $x = 12 + 1 = 13$ |
| 4 | 13 < 15 | true | $x = 13 + 1 = 14$ |
| 5 | 14 < 15 | true | $x = 14 + 1 = 15$ |
| 6 | 15 < 15 | false | Veiksmai neatliekami, nes netenkinama sąlyga |

Ciklas atlikamas 5 kartus. Atlikus veiksmus, kintamojo $x$ reikšmė lygi 15.

### 2 pavyzdys

```
x = 10; y = 14;
while (x <= y) {
  x = x + 1;
  y = y - 1;
}
```

| Ciklo žingsniai | Sąlyga: $x <= y$ | | Kintamojo $x$ reikšmė: $x = x + 1$ | Kintamojo $y$ reikšmė: $y = y - 1$ |
|---|---|---|---|---|
| 1 | 10 <= 14 | true | $x = 10 + 1 = 11$ | $y = 14 - 1 = 13$ |
| 2 | 11 <= 13 | true | $x = 11 + 1 = 12$ | $y = 13 - 1 = 12$ |
| 3 | 12 <= 12 | true | $x = 12 + 1 = 13$ | $y = 12 - 1 = 11$ |
| 4 | 13 <= 11 | false | Veiksmai neatliekami, nes netenkinama sąlyga | |

Ciklas atliekamas 3 kartus. Atlikus veiksmus, kintamojo $x$ reikšmė lygi 13, kintamojo $y$ reikšmė yra 11.



### 3 pavyzdys

```
x = 5;
while (x > 10)          // 5 > 10 (sąlyga netenkinama, veiksmai cikle neatliekami)
  x = x + 3;
x = x * 2 + 3;          // x = 5 * 2 + 3 = 13.
```

Ciklas neatliekamas nė karto. Atliekamas už ciklo esantis sakinys, kintamojo $x$ reikšmė lygi 13.

### 4 pavyzdys

```
x = 7; y = 8;
while (x <= y)
  x = x - 2;
```

| Ciklo žingsniai | Sąlyga: $x <= y$ | | Kintamojo $x$ reikšmė: $x = x + 1$ |
|---|---|---|---|
| 1 | 7 <= 8 | true | $x = 7 - 2 = 5$ |
| 2 | 5 <= 8 | true | $x = 5 - 2 = 3$ |
| 3 | 3 <= 8 | true | $x = 3 - 2 = 1$ |
| 4 | 1 <= 8 | true | $x = 1 - 2 = -1$ |
| 5 | Sąlyga visada bus teisinga | | |

Ciklas atliekamas be galo daug kartų. Tai ***amžinàsis cìklas***.

Naudojant nežinomo kartojimų skaičiaus ciklą **`while`**, reikia atkreipti dėmesį į šiuos dalykus:

- jeigu ciklo viduje reikia įvykdyti kelis sakinius, tai jie turi būti rašomi tarp `{` ir `}`;
- jei prieš pradedant ciklą *Sąlyga* yra netenkinama (`false`), *KartojamasSakinys* (ar sakinių grupė) nevykdomas nė karto;
- kintamiesiems, kurie yra sąlygos reiškinyje, prieš sakinį **`while`** esančioje programos dalyje turi būti suteiktos pradinės reikšmės. Ciklo viduje šių kintamųjų reikšmės turi būti keičiamos taip, kad kada nors *Sąlyga* taptų `false`. Kitu atveju ciklas bus vykdomas be galo daug kartų.

## 3.7. Ciklo sakinys `for`

Kai iš anksto žinoma, kiek kartų reikia kartoti veiksmus, naudojamas žinomo kartojimų skaičiaus ciklas **`for`**. Ciklo **`for`** antraštė valdo tik vieno sakinio kartojimą. Jeigu reikia kartoti kelis sakinius, tai jie rašomi tarp `{` ir `}`.

Ciklo **`for`** sintaksė yra tokia:

| | |
|---|---|
| **`for`** (*R1*`;` *Sąlyga*`;` *R2*)<br>  *KartojamasSakinys*`;` | **`for`** (*R1*`;` *Sąlyga*`;` *R2*) `{`<br>  *KartojamiSakiniai*`;`<br>`}` |

*R1* – priskyrimo sakinys, *Sąlyga* – bet koks santykio arba loginis reiškinys, *R2* – priskyrimo sakinys, *KartojamasSakinys* (*KartojamiSakiniai*) – bet koks sakinys (sakiniai).

Formaliai ciklo **`for`** veiksmus galima aprašyti taip:

1. Vykdomas sakinys *R1*. Paprastai jis naudojamas ***ciklo kintamojo*** (kintamasis, kuris vartojamas ciklo pabaigai apibrėžti) pradinei reikšmei priskirti. Sakinys *R1* įvykdomas tik pirmą kartą.
2. Patikrinama *Sąlyga*. Jei jos reikšmė yra `true`, tuomet atliekamas *KartojamasSakinys* (arba *KartojamiSakiniai*). Priešingu atveju ciklas nutraukiamas.
3. Vykdomas sakinys *R2*. Jis nurodo, kaip turi būti apskaičiuojama ciklo kintamojo reikšmė. Veiksmai kartojami nuo 2 žingsnio.

Naudojant žinomo kartojimų skaičiaus ciklą, reikia atkreipti dėmesį į šiuos dalykus:

- jeigu ciklo viduje reikia įvykdyti kelis sakinius, tai jie turi būti rašomi tarp `{` ir `}`;
- jei prieš pradedant ciklą *Sąlyga* yra netenkinama (`false`), *KartojamasSakinys* (ar sakinių grupė) nevykdomas nė karto;
- kintamiesiems, kurie yra sąlygos reiškinyje, prieš patikrinant *Sąlygą* pirmą kartą turi būti suteiktos pradinės reikšmės. Jos gali būti nurodomos prieš ciklą arba sakinyje *R1* priskyrimo sakiniu, kurie atskiriami kableliu. Ciklo viduje kintamųjų, kurie yra sąlygos reiškinyje, reikšmės turi būti keičiamos taip, kad kada nors *Sąlyga* taptų `false`. Kitu atveju ciklas bus vykdomas be galo daug kartų. Toks ciklas vadinamas ***amžinúoju ciklu***.

Pateikiame ciklo **`for`** naudojimo pavyzdžių.

**1 pavyzdys**

```
m = 10; n = 12;
for (i = m; i <= n; i = i + 1)
  cout << i << endl;
```

| Ciklo žingsniai | Kintamojo *i* reikšmė | Sąlyga: *i* <= 12 | | Ekrane matysite |
|---|---|---|---|---|
| 1 | 10 | 10 <= 12 | true | `10` |
| 2 | 11 | 11 <= 12 | true | `11` |
| 3 | 12 | 12 <= 12 | true | `12` |
| 4 | 13 | 13 <= 12 | false | |

Ciklas atliekamas 3 kartus.

**2 pavyzdys**

```
m = 10; n = 12; k = 5;
for (i = m; i <= n; i = i + 1) {
  k = k + 2;
  cout << i << "  " << k << endl;
}
```

| Ciklo žingsniai | Kintamojo *i* reikšmė | Sąlyga: *i* <= 12 | | Kintamojo *k* reikšmė | Ekrane matysite |
|---|---|---|---|---|---|
| 1 | 10 | 10 <= 12 | true | 7 | `10  7` |
| 2 | 11 | 11 <= 12 | true | 9 | `11  9` |
| 3 | 12 | 12 <= 12 | true | 11 | `12 11` |
| | | 13 <= 12 | false | | |

Ciklas atliekamas 3 kartus.

**3 pavyzdys**

```
m = 12; n = 10;
for (i = m; i <= n; i = i + 1)
  cout << i << endl;
cout << i << endl;
```

| Ciklo žingsniai | Kintamojo *i* reikšmė | Sąlyga: *i* <= 10 | | Ekrane matysite |
|---|---|---|---|---|
| | 12 | 12 <= 10 | false | `12` |

Ciklas neatliekamas, nes pradinė ciklo kintamojo reikšmė yra didesnė už galutinę. Atliekamas sakinys, esantis už ciklo.



**4 pavyzdys**

```
m = 12; n = 10;
for (i = m; i >= n; i = i - 1)
  cout << i << endl;
```

| Ciklo žingsniai | Kintamojo *i* reikšmė | Sąlyga: *i* >= 10 | | Ekrane matysite |
|---|---|---|---|---|
| 1 | 12 | 12 >= 10 | true | `12` |
| 2 | 11 | 11 >= 10 | true | `11` |
| 3 | 10 | 10 >= 10 | true | `10` |
| | 9 | 9 >= 10 | false | |

Ciklas atliekamas 3 kartus.

**5 pavyzdys**

```
for (i = 0; ; i = i + 1)
  cout << i;
```

Ciklas yra amžinasis (begalinis), nes nėra *Sąlygos*.

## 3.8. Sąlyginis sakinys `if`

***Sąlyginiù sãkiniu*** programoje keičiama nuosekli sakinių atlikimo tvarka: jei *Sąlyga* tenkinama, atliekamas *PirmasSakinys*, jei ne – po **`else`** esantis *AntrasSakinys*. Jeigu reikia atlikti kelis sakinius, kai *Sąlyga* tenkinama arba netenkinama, tai jie rašomi tarp `{` ir `}`. *Sąlyga* – bet koks santykio arba loginis reiškinys.

| | |
|---|---|
| **`if (`***Sąlyga***`)`** *PirmasSakinys*`;`<br>**`else`** *AntrasSakinys*`;` | **`if (`***Sąlyga***`) {`**<br>*Sakiniai, kurie atliekami, kai Sąlyga tenkinama*`;`<br>**`}`**<br>**`else {`**<br>*Sakiniai, kurie atliekami, kai Sąlyga netenkinama*`;`<br>**`}`** |

*Sąlyga* – tai bet koks loginis ar santykio reiškinys, *PirmasSakinys*, *AntrasSakinys* – bet koks sakinys.

Pateikiame keletą sąlyginio sakinio pavyzdžių.

| Pavyzdžiai | Sąlyga | | Ekrane matysite |
|---|---|---|---|
| `x = 4;`<br>**`if`** `(x < 9) y = x + 3;`<br>**`else`** `y = x + 5;`<br>`cout << y;` | 4 < 9 | true | `7` |
| `x = 14;`<br>**`if`** `(x < 9) y = x + 3;`<br>**`else`** `y = x + 5;`<br>`cout << y;` | 14 < 9 | false | `19` |
| `x = 4;`<br>**`if`** `(x < 9) {`<br>`  x = x + 2;`<br>`  y = x * 3;`<br>`}`<br>**`else`** `{`<br>`  x = x - 5;`<br>`  y = x * 5;`<br>`}`<br>`cout << x << "  " << y;` | 4 < 9 | true | `6 18` |

| Pavyzdžiai | Sąlyga | | Ekrane matysite |
|---|---|---|---|
| `x = 14;`<br>`if (x < 9) {`<br>`  x = x + 2;`<br>`  y = x * 3;`<br>`}`<br>`else {`<br>`  x = x – 5;`<br>`  y = x * 5;`<br>`}`<br>`cout << x << "  " << y;` | 14 < 9 | false | `9 45` |

Galima rašyti sutrumpintą sąlyginį sakinį, kuriame yra tik *PirmasSakinys* ir veiksmai atliekami tik tuomet, kai *Sąlyga* tenkinama.

| | |
|---|---|
| **if (***Sąlyga***)** *PirmasSakinys;* | **if (***Sąlyga***) {**<br>*Sakiniai, kurie atliekami, kai Sąlyga tenkinama;*<br>**}** |

Štai keletas tokių sąlyginių sakinių pavyzdžių.

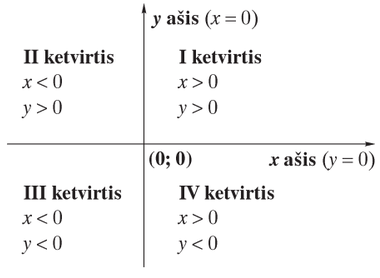
| Pavyzdžiai | Sąlyga | | Ekrane matysite |
|---|---|---|---|
| `x = 4; y = 14;`<br>`if (x < 9) y = x + 3;`<br>`cout << y;` | 4 < 9 | true | `7` |
| `x = 14; y = 14;`<br>`if (x < 9) y = x + 3;`<br>`cout << y;` | 14 < 9 | false | `14` |

Sąlyginio sakinio šakose galima užrašyti bet kokius C++ kalbos sakinius. Bet kurioje sąlyginio sakinio šakoje galima užrašyti dar vieną sąlyginį sakinį, pastarojo šakose – dar po vieną ir t. t. Toks sakinys vadinamas ***sudėtingu sąlyginiù sakiniu***.

| | |
|---|---|
| **if (***Sąlyga1***)**<br>**if (***Sąlyga2***)**<br>*PirmasSakinys;*<br>**else** *AntrasSakinys;*<br>**else** *TrečiasSakinys;* | **if (***Sąlyga1***)**<br>**if (***Sąlyga2***)**<br>**if (***Sąlyga3***)**<br>*PirmasSakinys;*<br>**else if (***Sąlyga4***)** *AntrasSakinys;* |

Pavyzdžiui, norėdami patikrinti, kuriame koordinačių plokštumos ketvirtyje (arba koordinačių ašyje) yra taškas (`x`; `y`), galime užrašyti tokį sąlyginį sakinį:

```
if ((x > 0) && (y > 0)) cout << "I ketvirtis " << endl;
else  if ((x < 0) && (y > 0)) cout << "II ketvirtis " << endl;
      else  if ((x < 0) && (y < 0)) cout << "III ketvirtis " << endl;
            else  if ((x > 0) && (y < 0)) cout << "IV ketvirtis " << endl;
                  else  if ((x == 0) && (y != 0)) cout << "Y ašis " << endl;
                        else  if ((x != 0) && (y == 0)) cout << "X ašis " << endl;
                              else cout << "Koordinačių pradžia " << endl
```






## 3.9. Knygoje naudojamų ir / ar rekomenduojamų matematinių funkcijų sąrašas

| Funkcija | Paskirtis |
|---|---|
| `int` **`abs`**`(int x);` | Grąžina sveikojo skaičiaus `x` absoliutųjį didumą. Esant teigiamam ar neigiamam funkcijos parametrui, rezultatas yra teigiamas.<br>Pavyzdys:<br>`int x = -10;`<br>`cout << "Sveikojo skaičiaus " << x`<br>`  << " modulis yra " << abs(x) << endl;`<br>Ekrane matysite:<br>`Sveikojo skaičiaus -10 modulis yra 10` |
| `double` **`fabs`**`(double x);` | Grąžina realiojo skaičiaus `x` absoliutųjį didumą.<br>Pavyzdys:<br>`double x = -15.5;`<br>`cout << "Realiojo skaičiaus " << x`<br>`  << " modulis yra " << fabs(x) << endl;`<br>Ekrane matysite:<br>`Realiojo skaičiaus -15.5 modulis yra 15.5` |
| `double` **`pow`**`(double x, double y);` | Skaičių `x` kelia laipsniu `y`. (Jei y yra `0.5`, tai funkcija skaičiuoja neneigiamo argumento `x` kvadratinę šaknį.)<br>Pavyzdys:<br>`int x = 15, y = 3;`<br>`cout << "Skaičius " << x`<br>`  << " pakeltas laipsniu " << y`<br>`  << ": " << pow(x, y) << endl;`<br>Ekrane matysite:<br>`Skaičius 15 pakeltas laipsniu 3: 3325` |
| `double` **`sqrt`**`(double x);` | Skaičiuoja neneigiamo argumento `x` kvadratinę šaknį.<br>Pavyzdys:<br>`double x = 15.5;`<br>`cout << "Kvadratinė šaknis iš realiojo skaičiaus "`<br>`  << x << " yra "`<br>`  << sqrt(x) << endl;`<br>Ekrane matysite:<br>`Kvadratinė šaknis iš realiojo skaičiaus 15.5 yra 3.937` |
| `int` **`random`**`(int Riba);` | Grąžina atsitiktinį sveikąjį skaičių iš intervalo `[0; Riba)`.<br>Pavyzdys:<br>`cout << "Atsitiktinis skaičius intervale 0-99: "`<br>`  << random(100) << endl;`<br>Ekrane matysite:<br>`Atsitiktinis skaičius intervale 0-99: 84` |
| `int` **`rand`**`(void);` | Grąžina atsitiktinį sveikąjį skaičių iš intervalo `[0; RAND_MAX]`. Konstanta `RAND_MAX` dažniausiai lygi 32767. Norint pasinaudoti konstanta `RAND_MAX`, reikia prie programos prijungti failą `stdlib.h` sakiniu `#include <stdlib.h>`<br>Pavyzdys:<br>`cout << "Atsitiktinis skaičius: " <<`<br>`  << rand() << endl;`<br>Ekrane matysite:<br>`Atsitiktinis skaičius: 130` |

| | |
|---|---|
| `void` **`randomize`**`(void);` | Priskiria pradinę reikšmę atsitiktinių skaičių generatoriui.<br>Funkcijos **`random`**`()` ir **`rand`**`()` visada generavimą pradeda nuo tos pačios reikšmės ir skaičius generuoja tokia pačia seka. Tačiau, jei prieš kreipimąsi į šias funkcijas, kreipiamasi į funkciją **`randomize`**`()`, tuomet funkcijos **`random`**`()` ir **`rand`**`()` kiekvieną kartą generuos skirtingų skaičių sekas.<br>(Jei atsitiktiniai skaičiai generuojami atliekame cikle, tuomet kreiptis į funkciją **`randomize`**`()` reikia prieš ciklą.)<br>Pavyzdys:<br>`randomize();`<br>`cout << "Funkcijos random() pateiktas skaičius: "`<br>`  << random(1000) << endl;`<br>`  << "Funkcijos rand() pateiktas skaičius: "`<br>`  << rand() << endl;`<br>Ekrane matysite:<br>`Funkcijos random() pateiktas skaičius: 540`<br>`Funkcijos rand() pateiktas skaičius: 2599` |
| `double` **`cos`**`(double x);` | Skaičiuoja argumento `x`, pateikiamo radianais, kosinusą.<br>Pavyzdys:<br>`double x = 0;`<br>`cout << "Kosinusas " << x << " yra "`<br>`  << cos(x) << endl;`<br>Ekrane matysite:<br>`Kosinusas 0 yra 1` |
| `double` **`sin`**`(double x);` | Skaičiuoja argumento `x`, pateikiamo radianais, sinusą.<br>Pavyzdys:<br>`double x = 0;`<br>`cout << "Sinusas " << x << " yra "`<br>`  << sin(x) << endl;`<br>Ekrane matysite:<br>`Sinusas 0 yra 0` |
| `double` **`tan`**`(double x);` | Skaičiuoja argumento `x`, pateikiamo radianais, tangentą.<br>Pavyzdys:<br>`double x = 0;`<br>`cout << "Tangentas " << x << " yra "`<br>`  << tan(x) << endl;`<br>Ekrane matysite:<br>`Tangentas 0 yra 0` |

## 3.10. Duomenų įvedimas iš failo

Kai pradinių duomenų daug, juos įvedinėti klaviatūra nėra patogu. Be to, norint skaičiavimus pakartoti, reikia duomenis įvesti iš naujo. Nepatogumų galima išvengti, duomenis iš anksto įrašant į tekstinį failą. Tačiau į failą nerašomi pranešimai ir paaiškinimai, kokie duomenys ir kokia eilės tvarka pateikiami. Rašant duomenų skaitymo iš failo sakinius, reikia iš anksto žinoti, kokia eilės tvarka faile surašyti duomenys. Nuo to priklauso, kiek bus kintamųjų ir kokia eilės tvarka jų reikšmės bus skaitomos iš failo.

Norint duomenis nuskaityti iš failo, reikia:

- Aprašyti įvedimo iš srauto `ifstream` kintamąjį, pavyzdžiui, `ifstream fd;`.
- Programoje susieti kintamąjį su tekstiniu failu, pavyzdžiui, `fd.open("Duom.txt").`

  Galima aprašyti įvedimo iš srauto `ifstream` kintamąjį ir susieti jį su tekstiniu failu vienu sakiniu, pavyzdžiui, `ifstream fd("Duom.txt");`
- Baigus darbą, failą būtina užverti, pavyzdžiui, `fd.close();`.

Visų įvedimo iš srauto duomenų failų kintamųjų vardus rekomenduojame pradėti raidėmis `fd` (raidė `f` reiškia failą, `d` – duomenis). Tuomet programos tekste juos galima atpažinti be papildomų paaiškinimų.



Duomenims iš failo skaityti naudojamas tas pats operatorius >>, kaip ir įvedamiems klaviatūra, tik srauto vardas `cin` yra keičiamas įvedimo iš srauto, susijusio su duomenų failu, kintamojo vardu, pavyzdžiui,

```
fd >> x;
```

Įvedimo ir išvedimo srautų, susijusių su duomenų ir rezultatų failais, priemonės aprašytos antraštiniame faile `fstream`. Jis perkeliamas į programą tokiu sakiniu:

```
#include <fstream>
```

**Pavyzdys.** Iš tekstinio failo *Duomenys.txt* nuskaitomi du sveikieji skaičiai ir į ekraną išvedama nuskaitytų skaičių suma.

| *Duomenys.txt* |
|---|
| `15 134` |

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
  int a, b, s;
  ifstream fd("Duomenys.txt");
  fd >> a >> b;
  s = a + b;
  fd.close();
  cout << s << endl;
  return 0;
}
```

## 3.11. Rezultatų (duomenų) išvedimas į failą

Akivaizdu, kad skaičiavimų rezultatus į ekraną galima išvesti, jeigu jų yra nedaug. Tuomet nesunku juos peržiūrėti ir / arba nusirašyti. Tačiau jeigu rezultatų yra daug, tai kur kas patogiau juos išvesti į tekstinį failą. Tada duomenis lengva panaudoti dokumentuose, apdoroti kitomis taikomosiomis programomis (pvz., skaičiuokle).

Norint duomenis įrašyti į failą, reikia:

- Aprašyti išvedimo į srautą `ofstream` kintamąjį, pavyzdžiui, `ofstream fd;`.
- Programoje susieti failo kintamąjį su tekstiniu failu, pavyzdžiui, `fr.open("Duom.txt").`

  Galima aprašyti įvedimo į srautą ifstream kintamąjį ir susieti jį su tekstiniu failu vienu sakiniu, pavyzdžiui, `ofstream fr("Rezultatas.txt");`

  Vykdant programą, tuo atveju, jeigu darbiniame kataloge yra tekstinis failas nurodytu vardu, tai jame esantys duomenys pašalinami. Priešingu atveju sukuriamas naujas failas.

  Norint jau egzistuojantį rezultatų failą papildyti, jis atveriamas, pavyzdžiui, taip:

  `ofstream fr("Rezultatas.txt", ios::app);`

  Tuomet esami duomenys išlieka, o failas paruošiamas naujiems duomenims rašyti failo pabaigoje.
- Baigus darbą, failą būtina užverti, pavyzdžiui, `fr.close();`.

Visų išvedimo į failų srautus kintamųjų vardus rekomenduojame pradėti raidėmis `fr` (raidė `f` reiškia failą, `r` – rezultatą). Tuomet programos tekste juos galima atpažinti be papildomų paaiškinimų.

Duomenims į failą rašyti naudojamas tas pats operatorius <<, kaip ir įvedamiems klaviatūra, tik srauto vardas `cout` yra keičiamas išvedimo į srauto, susijusio su duomenų failu, kintamojo vardu, pavyzdžiui,

```
fr << x;
```

Įvedimo ir išvedimo srautų, susijusių su duomenų ir rezultatų failais, priemonės aprašytos antraštiniame faile `fstream`. Jis perkeliamas į programą tokiu sakiniu:

```
#include <fstream>
```

**Pavyzdys.** Į tekstinį failą Rezultatas.txt išvedama dviejų sveikųjų skaičių, įvestų klaviatūra, suma.

```
#include <fstream>
#include <iostream>
using namespace std;
int main ()
{
  int a, b, s;
  cin >> a >> b;
  s = a + b;
  ofstream fr("Rezultatas.txt");
  fr << s << endl;
  fr.close();
  return 0;
}
```

Klaviatūra įvedę skaičius 49 ir 100, rezultatų faile *Rezultatai.txt* matysite:

| *Rezultatas.txt* |
|---|
| 149 |

## 3.12. Funkcijos

***Fùnkcijomis*** yra vadinamos programos konstrukcijos, kurios atlieka savarankiškus veiksmus. Vykdant programą, į funkcijas galima kreiptis daug kartų. Funkcijos padeda struktūrizuoti programas. Programas su funkcijomis lengviau skaityti ir analizuoti.

Naujai kuriamų funkcijų aprašymą galima skaidyti į dvi dalis: *prototipą* (išankstinį aprašą) ir realizavimo aprašymą.

Funkcijos prototipas nurodo naudotojui, kokia tvarka duomenys perduodami funkcijai ir kaip gaunami rezultatai. Funkcijos prototipas rašomas prieš funkciją `main()` ir baigiamas kabliataškiu. Prototipo struktūra tokia:

```
rezultatoTipas funkcijosVardas(formaliiejiParametrai);
```

Funkcijos prototipo pavyzdys: `int Suma(int a, int b);`

Paprastai funkcijos realizavimo aprašas (funkcijos *antraštė* ir *veiksmai*, kuriuos ji turi atlikti) rašomas už `main()` funkcijos. Jei funkcijos realizavimo aprašas pateikiamas virš `main()`, tuomet nereikia rašyti funkcijos prototipo.

Funkcijos aprašas atrodo taip:

```
rezultatoTipas funkcijosVardas(formaliejiParametrai)
{
  funkcijosKamienas
}
```

Funkcijos antraštėje pirmiausiai nurodomas grąžinamos reikšmės tipas. Jei funkcija jokios reikšmės negrąžina, vietoj `rezultatoTipas` nurodomas bazinis žodis `void`. Toliau rašomas `funkcijosVardas`, kuris parenkamas pagal tas pačias taisykles, kaip ir kintamųjų vardai. Po to skliaustuose išvardijami parametrai. Jei funkcija neturi parametrų, tuomet rašomi tušti skliaustai.

`FormaliejiParametrai` skirti funkcijos duomenims susieti su programos duomenimis. Parametrai apibrėžiami pagal tas pačias taisykles, kaip ir kintamieji: nurodant jų tipą ir vardą. Vienas nuo kito parametrai atskiriami kableliais.

`FunkcijosKamiene` aprašomi veiksmai, kuriuos turi atlikti funkcija. Jei funkcija skirta kokiai nors reikšmei grąžinti, tai tarp funkcijos veiksmų turi būti bent vienas sakinys, kuriuo apskaičiuota reikšmė grąžinama į programą. Funkcijos kamiene gali būti aprašomi kintamieji, reikalingi funkcijos veiksmams atlikti. Jie galioja tik funkcijoje.

Reikšmei grąžinti per funkcijos vardą naudojamas sakinys `return`. Sakinio `return` sintaksė:

```
return Reiškinys;
```



`Reiškinys` – tai bet koks reiškinys, kurio reikšmė grąžinama atlikus funkciją. Grąžinamos reikšmės tipas ir funkcijos antraštėje nurodytas rezultato tipas turi būti tarpusavyje suderinami. Sakinys `return` ne tik grąžina nurodytą reikšmę, bet ir nutraukia funkcijos darbą. Sakinys `return;` tiesiog nutraukia funkcijos darbą.

Į funkcijas kreipiamasi jų vardais, už kurių skliaustuose pateikiami faktiniai parametrai (argumentai). Jei funkcija grąžina reikšmę, į ją kreipiamasi reiškiniuose, pavyzdžiui,

```
y = funkcijosVardas(faktiniaiParametrai) * c;.
```

Jei funkcija jokios reikšmės negrąžina, į ją kreipiamasi taip:

```
funkcijosVardas(faktiniaiParametrai);
```

`FaktiniaiParametrai` nurodo tuos duomenis, su kuriais bus atliekami funkcijos veiksmai.

Kreipinyje į funkciją faktinių parametrų paprastai būna tiek pat ir tokio pat tipo, kaip nurodyta funkcijos antraštėje. Parametrai rašomi tokia pat eilės tvarka, kaip nurodyta funkcijos antraštėje, ir tarpusavyje atskiriami kableliais. Jei funkcija parametrų neturi, kreipinyje nurodomi tušti skliaustai.

**Pavyzdys**

```
// Stačiakampio plotas
#include <iostream>
using namespace std;
//-----------------------------------------------------
int Plotas(int a, int b);           // funkcijos Plotas prototipas
//-----------------------------------------------------
int main ()
{
  int x = 5, y = 4, s;
  s = Plotas(x, y);                 // kreipinys į funkciją Plotas
  cout << "Plotas = " << s << endl;
  return 0;
}
//-----------------------------------------------------
// Skaičiuoja stačiakampio, kurio kraštinės a ir b, plotą
int Plotas(int a, int b)            // funkcijos antraštė
{
  return a * b;                     // grąžinamas apskaičiuotas stačiakampio plotas
}
```

Įvykdę programą, ekrane matysite:

```
Stačiakampio plotas 20
```

Jei funkcija turi grąžinti keletą reikšmių, tuomet naudojami *parametrai–nuorodos*. Prieš juos funkcijos antraštėje rašomas ženklas `&`:

```
rezultatoTipas funkcijosVardas(tipas & vardas1, tipas & vardas2);
```

Tuo atveju, kai į funkciją kreipiamasi, perduodant jai parametrus–reikšmes, funkcija sukuria naujus tokių pačių tipų kintamuosius, kaip ir perduodami parametrai, ir šiems kintamiesiems priskiria parametrų reikšmes. Vadinasi, funkcija dirba su parametrų reikšmių kopijomis, bet ne su pačiais parametrais. Toks mechanizmas yra patogus, kai funkcijai nereikia keisti parametrų reikšmių.

Tuo atveju, kai į funkciją kreipiamasi, perduodant jai parametrus–nuorodas, ji gauna ne kintamųjų reikšmes, o nuorodas į kintamuosius (kintamųjų adresai). Vadinasi, funkcija tiesiogiai naudoja perduodamus kintamuosius. Štai pavyzdys, kuriame funkcija `Sukeisti()`, naudodama parametrų vardus `pirmas` ir `antras`, faktiškai naudojasi kintamaisiais `x` ir `y`.

```
// Dviejų kintamųjų reikšmių sukeitimas
#include <iostream>
using namespace std;
//-------------------------------------------------
void Sukeisti(int & pirmas, int & antras);
//-------------------------------------------------
int main ()
{
  int x = 11, y = 25;
  Sukeisti(x, y);
  cout << " Kintamasis x po keitimo " << x << endl;
  cout << " Kintamasis y po keitimo " << y << endl;
  return 0;
}
//-------------------------------------------------
void Sukeisti(int & pirmas, int & antras)
{
  int papildomas = pirmas;
      pirmas = antras;
      antras = papildomas;
}
```

Įvykdę programą ekrane matysite:

```
Kintamasis x po keitimo 25
Kintamasis y po keitimo 11
```

## 3.13. Knygoje naudojamų įterpiamų failų sąrašas

Programos pradžioje rašomos instrukcijos *parengiamajai doroklei* (angl. preprocessor), kokių failų tekstai turi būti įterpiami į programą pirminio apdorojimo metu. Įterpimo instrukcijos pradžioje rašoma `#include`, toliau tarp simbolių `< >` nurodomi įterpiamų failų vardai. Pavyzdžiui, antraštinio failo `iostream` priemonės perkeliamos į programą sakiniu: `#include <iostream>`.

Lentelėje pateikiami tik praktikos darbuose naudojami antraštiniai failai.

| Įterpiamas failas | Paaiškinimas |
|---|---|
| `iostream` | Duomenų įvedimo klaviatūra ir išvedimo į ekraną priemonės |
| `fstream` | Duomenų skaitymo iš failo ir išvedimo į failą priemonės |
| `iomanip` | Duomenų išvedimo į failų srautus (ekranas, failas) priemonės |
| `cmath` | Matematinių funkcijų rinkinys |
| `fcntl.h` | Priemonės lietuviškiems rašmenims ekrane išvesti į ekraną |
| `io.h` | Priemonės lietuviškiems rašmenims ekrane išvesti į ekraną |



# 4 ALGORITMŲ ŽINYNAS

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

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

E. W. Dijkstros teigimu, jei uždaviniui spręsti galima parašyti algoritmą, tai galima sugalvoti daugybę algoritmų ir po to iš jų išsirinkti patį efektyviausią.

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

Toliau aptarsime tik kai kuriuos praktikos darbuose naudojamus algoritmus ir jų C++ programavimo kalba užrašytus tekstus (kodus, fragmentus). Visus pateiktus kodus galima išbandyti, įkeliant juos į `main()` arba į savo sukurtą funkciją.

Norint išvesti į ekraną tekstą su lietuviškais rašmenimis, reikia parašyti sakinį:

```
_setmode (_fileno(stdout), _O_U16TEXT);
```

Tekstui išvesti į ekraną naudojama išvedimo srauto modifikacija `wcout`. Prieš simbolių eilutes, kuriose yra lietuviškų rašmenų, reikia parašyti didžiąją raidę `L`. Pavyzdžiui:

```
wcout << L"Skaičių suma yra  ";.
```

Šios priemonės tinka naudojant aplinką *Microsoft Visual C++* arba aplinką *CodeBlocks*, kuri diegiant buvo susieta su aplinka *Microsoft Visual C++*.

Žinyne visoms simbolių eilutėms išvesti į ekraną naudojamas srautas `cout`.

## 4.1. Tiesiniai algoritmai

Tai algoritmai, kai veiksmai atliekami jų surašymo eilės tvarka.

**1 pavyzdys**. *Triženklio natūraliojo skaičiaus x skaitmenų sumos s skaičiavimas.*

Norint išspręsti šį uždavinį, reikia skaičių $x$ išskaidyti skaitmenimis, skaitmenis sudėti ir pateikti skaičiavimų rezultatą.

Algoritmas, užrašytas 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 C++ programavimo kalba:

```
...
int x, a, b, c, s;
cout << "Įveskite triženklį natūralųjį skaičių ";
cin >> x;
a = x / 100;
b = x / 10 % 10;
c = x % 10;
s = a + b + c;
cout << "Skaičiaus " << x << " skaitmenų suma lygi " << s;
...
```

```
Skaičiaus 158 skaitmenų suma lygi 14
```

## 4.2. Cikliniai algoritmai

Tai algoritmai, kai veiksmus programoje reikia atlikti daug kartų. Veiksmų kartojimų skaičius nusakomas santykio ir loginiais reiškiniais arba konkrečiu skaičiumi.

**2 pavyzdys**. *Visų natūraliųjų triženklių skaičių skaitmenų sumų skaičiavimas.*

Norint išspręsti šį uždavinį, reikia kiekvieną triženklį skaičių išskaidyti skaitmenimis, skaitmenis sudėti ir pateikti gautą rezultatą (skaičių, jo skaitmenis ir jų sumą).

Algoritmas, užrašytas žodžiais:

1. Pradinis duomuo – pirmas triženklis natūralusis skaičius $x = 100$.
2. Kartojami 3–8 veiksmai tol, kol $x < 1000$.
3. Atskiriamas pirmasis triženklio skaičiaus skaitmuo $a$.
4. Atskiriamas antrasis triženklio skaičiaus skaitmuo $b$.
5. Atskiriamas trečiasis triženklio skaičiaus skaitmuo $c$.
6. Skaičiuojama triženklio skaičiaus skaitmenų suma $s = a + b + c$.
7. Pateikiamas gautas skaičiavimų rezultatas.
8. Reikšmė $x$ padidinama vienetu.

Algoritmas, užrašytas C++ programavimo kalba:

```
...
int x, a, b, c, s;
x = 100;
while (x < 1000) {
  a = x / 100;
  b = x / 10 % 10;
  c = x % 10;
  s = a + b + c;
  cout << "Skaičiaus " << x << " skaitmenų suma lygi " << s << endl;
  x = x + 1;
}
...
```

## 4.3. Šakotieji skaičiavimai

Tai algoritmai, kai veiksmus reikia vykdyti atsižvelgiant į gautus ankstesnių skaičiavimų rezultatus. Tokie skaičiavimai aprašomi sąlyginiais sakiniais, kuriuose naudojami santykio ir loginiai reiškiniai.

**3 pavyzdys**. *Kiekvieno natūraliojo triženklio nelyginio skaičiaus skaitmenų sumos skaičiavimas.*

Norint išspręsti šį uždavinį, reikia kiekvieną triženklį skaičių patikrinti, ar jis nelyginis. Jei taip, tuomet reikia jį išskaidyti skaitmenimis, juos sudėti ir pateikti gautą rezultatą.



Algoritmas, užrašytas žodžiais:

1. Pradinis duomuo – pirmas triženklis natūralusis skaičius $x = 100$.
2. Trečias veiksmas kartojamas tol, kol $x < 1000$.
3. Tikrinama sąlyga, ar triženklis skaičius yra nelyginis.
   - 3.1. Jei sąlyga tenkinama, tuomet atliekami veiksmai:
     - 3.1.1. Atskiriamas pirmasis triženklio skaičiaus skaitmuo $a$.
     - 3.1.2. Atskiriamas antrasis triženklio skaičiaus skaitmuo $b$.
     - 3.1.3. Atskiriamas trečiasis triženklio skaičiaus skaitmuo $c$.
     - 3.1.4. Skaičiuojama triženklio skaičiaus skaitmenų suma $s = a + b + c$.
     - 3.1.5. Pateikiamas gautas skaičiavimų rezultatas.
     - 3.1.6. Reikšmė $x$ padidinama vienetu.
   - 3.2. Jei sąlyga netenkinama, reikšmė $x$ padidinama vienetu.

Algoritmas, užrašytas C++ programavimo kalba:

```
...
int x, a, b, c, s;
x = 100;
while (x < 1000) {
  if (x % 2 != 0) {
    a = x / 100;
    b = x / 10 % 10;
    c = x % 10;
    s = a + b + c;
    cout << "Skaičiaus " << x << " skaitmenų suma lygi " << s << endl;
    x = x + 1;
  }
  else x = x + 1;
}
...
```

Veiksmus galima supaprastinti, iškėlus priskyrimo sakinį `x = x + 1;` iš sąlygos tikrinimo:

```
...
while (x < 1000) {
  if (x % 2 != 0) {
    a = x / 100;
    b = x / 10 % 10;
    c = x % 10;
    s = a + b + c;
    cout << "Skaičiaus " << x << " skaitmenų suma lygi " << s << endl;
  }
  x = x + 1;
}
...
```

Sprendžiant 3 pavyzdžio uždavinį, sąlygos `if (x % 2 != 0)` galima ir netikrinti (nelieka šakojimo veiksmo).

Algoritmas, užrašytas žodžiais:

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

Algoritmas, užrašytas C++ programavimo kalba:

```
...
int x, a, b, c, s;
x = 101;
while (x < 1000) {
  a = x / 100;
  b = x / 10 % 10;
  c = x % 10;
  s = a + b + c;
  cout << "Skaičiaus " << x << " skaitmenų suma lygi " << s << endl;
  x = x + 2;
}
...
```

## 4.4. Sumos skaičiavimo algoritmas

Kelių skaičių sumą galima užrašyti taip: `suma = sk1 + sk2 + … + skn`.

Čia `sk1, sk2, …, skn` yra skaičiai, kuriuos sudėjus gaunama suma `suma`.

Atliekant veiksmus skaičiuotuvu (ar pieštuku popieriuje), visuomet sumuojami du skaičiai. Veiksmus galima 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
```

Skaičiuotuvo ekrane visuomet matyti tik paskutinio veiksmo rezultatas (dalinė suma). Kiekviena nauja suma gaunama prie jau apskaičiuotos sumos pridedant po vieną dėmenį. Todėl nagrinėjamų veiksmų seką galima užrašyti taip:

```
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 skaičiavimo formulę galima užrašyti ir taip: `suma = suma + skaičius`. Šiuo atveju pradinė `suma` reikšmė turi būti lygi nuliui:

```
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 algoritmą galima užrašyti pseudokodu taip:

**Pradžia** Skaičių suma
    *suma* = 0;
    kol yra skaičių
        ciklo pradžia
            Įvesti skaičių *sk*;
            *suma* = *suma* + *sk*;
        ciklo pabaiga
    Išvesti rezultatą *suma*
**Pabaiga**



Ciklo sąlygą „kol yra skaičių“ galima nurodyti labai įvairiai. Paprasčiausiu atveju prieš ciklą tiksliai apibrėžiama, kiek bus skaičių, kuriuos reikia sudėti.

Galimas toks algoritmo užrašymas C++ programavimo kalba:

```
// Skaičių suma
...
  int suma, sk; // suma ir dėmuo
  int n;        // skaičių kiekis
  int i;        // ciklo kintamasis
  cout << "Kiek bus skaičių?  ";
  cin >> n;
  suma = 0;
  for (i = 1; i <= n; i = i + 1) {
    cout << "Koks bus " << i << " skaičius? ";
    cin >> sk;
    suma = suma + sk;
  }
  cout << "Įvestų skaičių suma = " << suma << endl;
  ...
```

```
Kiek bus skaičių? 3
Koks bus 1 skaičius? 12
Koks bus 2 skaičius? 5
Koks bus 3 skaičius? 10
Įvestų skaičių suma = 27
```

Kaip matote, sumos skaičiavimo veiksmai panašūs į tuos, kurie atliekami atsiskaitant už prekes parduotuvėje:

- Pradedant skaičiuoti, kasos aparato ekrane rodoma reikšmė lygi nuliui.
- Nuskaitant brūkšninio kodo skaitytuvu prekių kodus, jų kaina pridedama prie ekrane matomo skaičiaus. Pinigų suma, kurią reikės mokėti, didėja.
- Nuskaičius paskutinės prekės kodą, ekrane parodomas rezultatas.

## 4.5. Sandaugos skaičiavimo algoritmas

Kelių skaičių sandaugą galima užrašyti taip: `sand = sk1 * sk2 * … * skn`.

Čia `sk1, sk2, …, skn` yra skaičiai, kuriuos sudauginus, gaunama sandauga `sand`.

Atliekant veiksmus skaičiuotuvu (ar pieštuku popieriuje), visuomet dauginami du skaičiai. Veiksmus galima išskleisti taip pat, kaip ir sumuojant skaičius. Kiekviena nauja sandauga gaunama jau apskaičiuotą sandaugą dauginant iš naujo dauginamojo: `sand = sand * skaičius`. Pradinė sandaugos reikšmė turi būti lygi vienetui.

Sandaugos skaičiavimų seka yra 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 galima aprašyti tokiu pat algoritmu, kaip ir skaičiuojant sumą, tik pradinė sandaugos reikšmė turi būti lygi vienetui, o sudėties operacijas (+) reikia pakeisti daugybos operacijomis (*).

**4 pavyzdys**. *Skaičiaus kėlimas sveikuoju laipsniu* ($r = x^n$).

Algoritmas, užrašytas C++ programavimo kalba:

```
// Skaičiaus kėlimas laipsniu. Taikomas sandaugos skaičiavimo algoritmas
...
  int r,  // sandauga
      x,  // laipsnio pagrindas
      n;  // laipsnio rodiklis
  int i;  // ciklo kintamasis
  cout << "Skaičius, kurį kelsite laipsniu: ";
  cin >> x;
  cout << "Laipsnio rodiklio reikšmė: ";
  cin >> n;
  r = 1;
  for (i = 1; i <= n; i = i + 1)
    r = r * x;
  cout << "Rezultatas: " << r << endl;
  ...
```

```
Skaičius, kurį kelsite laipsniu: 2
Laipsnio rodiklio reikšmė: 3
Rezultatas: 8
```

## 4.6. Kiekio skaičiavimo algoritmas

**5 pavyzdys.** *Dviženklių natūraliųjų skaičių, kurie dalijasi iš penkių, kiekio skaičiavimas.*

Pirmasis dviženklis skaičius, kuris dalijasi iš penkių, yra `10`. Kiekio pradinė reikšmė yra lygi nuliui. Ciklo antraštėje rašoma sąlyga, kad ciklas būtų nutrauktas, peržiūrėjus visus dviženklius skaičius. Ciklas pradedamas pirmuoju dviženkliu skaičiumi, kuris dalijasi iš `5`, todėl tokių skaičių kiekis padidėja vienetu. Po to *x* reikšmė didinama `5`, tikrinama ciklo sąlyga ir atliekami veiksmai ciklo viduje.

Veiksmų algoritmas analogiškas sumos skaičiavimo algoritmui, tik čia sumuojamos ne reikšmės, o vienetukai.

Algoritmas, užrašytas C++ programavimo kalba:

```
  ...
  int x,  // pirmasis dviženklis skaičius, kuris dalijasi iš penkių
      k;  // dviženklių skaičių, dalių iš 5, kiekis
  x = 10;
  k = 0;
  while (x < 100) {
    k = k + 1;
    x = x + 5;
  }
  cout << "Dviženklių skaičių, dalių iš 5, yra " << k << endl;
  ...
```

```
Dviženklių skaičių, dalių iš 5, yra 18
```



**6 pavyzdys**. *Natūraliojo skaičiaus x daliklių kiekio k skaičiavimas.*

Uždavinio sprendimo algoritmas yra klasikinis: ciklas vykdomas nuo 1 iki *x*. Tikrinama, ar natūralusis skaičius *x* be liekanos dalijasi iš ciklo kintamojo *i*. Jei taip, tai *i* yra *x* daliklis ir daliklių kiekis *k* didinamas vienetu.

Algoritmas, užrašytas C++ programavimo kalba:

```
  ...
  int x, i, k;
  cout << "Skaičius x = ";
  cin >> x;
  k = 0;
  for (i = 1; i <= x; i = i + 1)
    if (x % i == 0)
      k = k + 1;
  cout << " turi " << k << " daliklių(-ius) " << endl;
  ...
```

```
Skaičius x = 6 turi 4 daliklių(-ius)
```

## 4.7. Aritmetinio vidurkio skaičiavimas

Aritmetinis vidurkis skaičiuojamas dviem etapais:

1) apskaičiuojama skaičių suma;
2) gauta suma padalijama iš skaičių kiekio.

Sumos skaičiavimo algoritmą jau išnagrinėjome. Reikia atkreipti dėmesį, kad jeigu iš anksto nežinoma, kiek skaičių sumuojama, tai sumuojant reikia kartu skaičiuoti, kiek tokių skaičių buvo.

Antrasis etapas – tai dviejų skaičių (gautos sumos ir skaičių kiekio) dalyba:

```
vidurkis = suma / skaičiųKiekis.
```

Žinome, kad dalyba iš nulio negalima. Todėl, prieš atliekant dalybos veiksmą (pvz., kai reikia apskaičiuoti teigiamų skaičių aritmetinį vidurkį), būtina įsitikinti, kad galima dalyti (kad įvesti ne vien neigiami skaičiai ir nuliai).

## 4.8. Didžiausios (mažiausios) reikšmės paieška

Tai tradiciniai programavimo uždaviniai. Populiariausias jų sprendimo būdas yra toks. Ieškant didžiausios reikšmės aprašomi du kintamieji: įvedamai *x* ir didžiausiai *d* reikšmėms atmintyje laikyti. Įvedant pirmąją reikšmę, daroma prielaida, kad ši yra didžiausia:

```
d = x;
```

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

```
if (x > d) d = x;
```

Taip patikrinus visą įvedamą duomenų srautą, kintamojo *d* reikšmė bus didžiausia įvesta reikšmė.

Analogiškai ieškoma mažiausios reikšmės. Skiriasi tik palyginimo sąlyga:

```
if (x < d) d = x;
```

Algoritmas, užrašytas C++ programavimo kalba:

```
  ...
  int n, x, d, i;
  cout << "Kiek bus skaičių: ";
  cin >> n;
  cout << "Koks 1 skaičius: ";
  cin >> d;
  for (i = 2; i <= n; i = i + 1) {
    cout << "Koks " << i << " skaičius: ";
    cin >> x;
    if (x > d) d = x;
  }
  cout << "Didžiausias skaičius: " << d;
  ...
```

```
Kiek bus skaičių: 5
Koks 1 skaičius: 7
Koks 2 skaičius: 9
Koks 3 skaičius: 12
Koks 4 skaičius: 8
Koks 5 skaičius: 6
Didžiausias skaičius: 12
```

Pirmąją reikšmę laikyti pradine didžiausia (arba mažiausia) reikšme ne visuomet patogu. Ypač tada, kai ta reikšmė turi būti apskaičiuojama (pavyzdžiui, ieškant mažiausio teigiamo skaičiaus, kai sekos pradžioje gali būti daug neigiamų skaičių). Tokiu atveju pradinei didžiausiai reikšmei galima priskirti 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 tokią, kuri tikrai būtų didesnė už visas galimas reikšmes, tarp kurių ieškome mažiausios reikšmės. Geriausiai tam tinka standartinė C++ konstanta `RAND_MAX`, kuri yra faile `stdlib.h`.

**7 pavyzdys**. *Sunkiausio arbūzo paieška.*

Tarkime, ant prekystalio yra *n* arbūzų, kurių masės žinomos.

Algoritmas, užrašytas C++ programavimo kalba:

```
...
int n;        // arbūzų skaičius
double m;     // arbūzo masė
int i;        // arbūzo numeris
double s;     // sunkiausio arbūzo masė
int ns;       // sunkiausio arbūzo numeris
cout << "Kiek yra arbūzų?  ";
cin >> n;
// Sunkiausio arbūzo paieška
cout << "Arbūzų masių įvedimas";
s = 0;  ns = 0;
for (i = 1; i <= n; i = i + 1) {
  cout << "Įveskite" << i << " arbūzo masę: ";
  cin >> m;
  if (m > s) {
    s = m; ns = i;
  }
}
// Rezultatai
cout << "Sunkiausias arbūzas Nr. " << ns << endl;
cout << "Jo masė: " << s << endl;
...
```

```
Kiek yra arbūzų? 5
Arbūzų masių įvedimas
Įveskite 1 arbūzo masę: 3.9
Įveskite 2 arbūzo masę: 3.5
Įveskite 3 arbūzo masę: 4.7
Įveskite 4 arbūzo masę: 3.9
Įveskite 5 arbūzo masę: 4.2
Sunkiausias arbūzas Nr. 3
Jo masė: 4.7
```



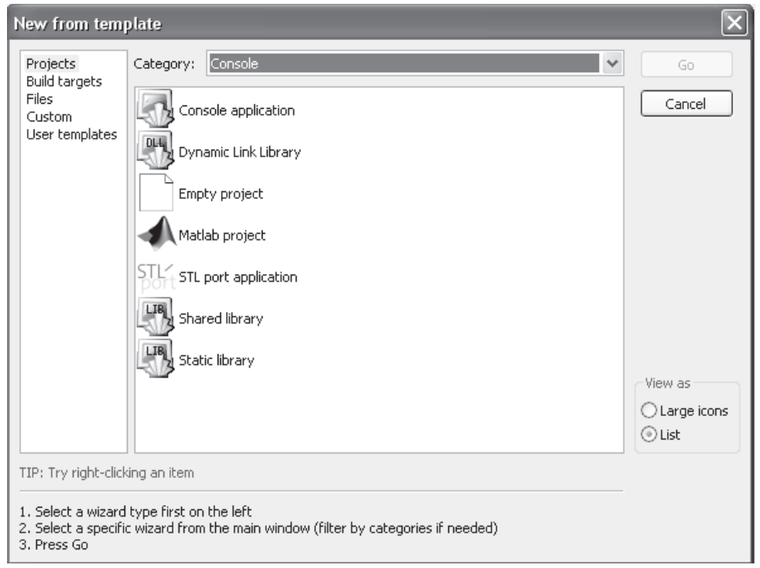
# 5 APLINKA CodeBlocks

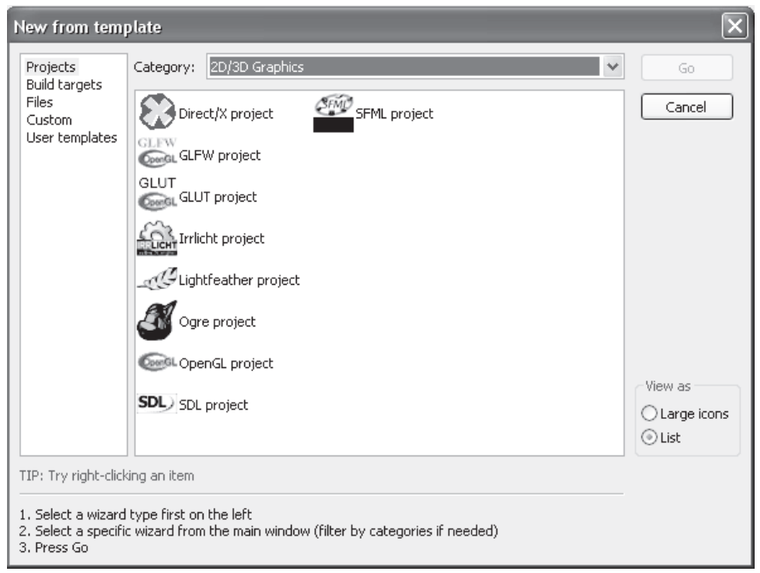
## 5.1. Pagrindiniai *CodeBlocks* bruožai

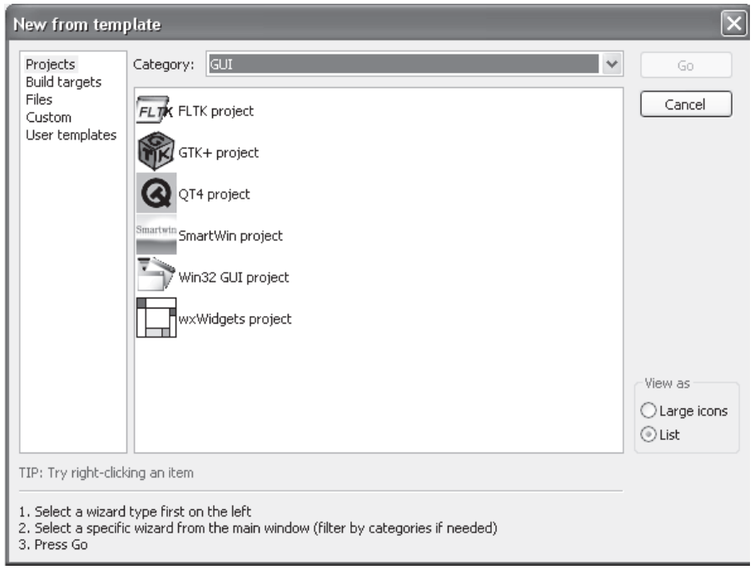
*CodeBlocks* – populiari, patogi, nemokama, atvirojo kodo, nuolat atnaujinama ir tobulinama programavimo aplinka, skirta programuoti *C* / *C++* kalbomis operacinėse sistemose *Windows*, *Linux*, *Mac OS X*.

*CodeBlocks* turi patogią rengyklę, kuri išskiria programos fragmentus, automatiškai pildo programos kodą (pvz., užrašius vieną skliaustą, automatiškai įrašomas kitas skliaustas).

Naudojantis šia aplinka galima kurti įvairius projektus:



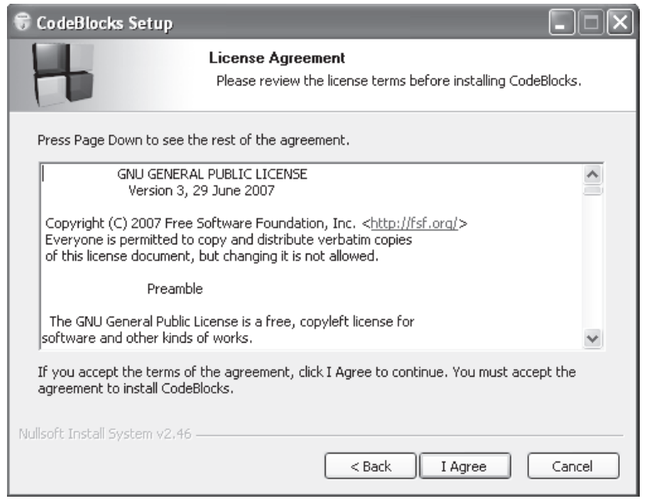





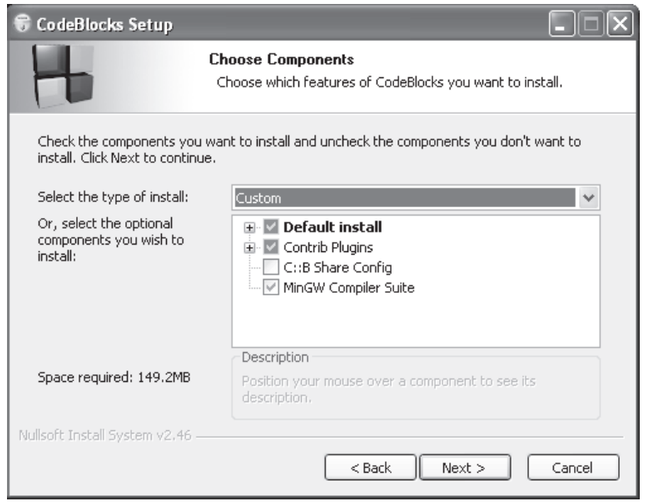

## 5.2. *CodeBlocks* įdiegimas

➢ Paleiskite diegimo failą. Pasirodžius informacinio pranešimo langui spragtelėkite mygtuką *Next*.

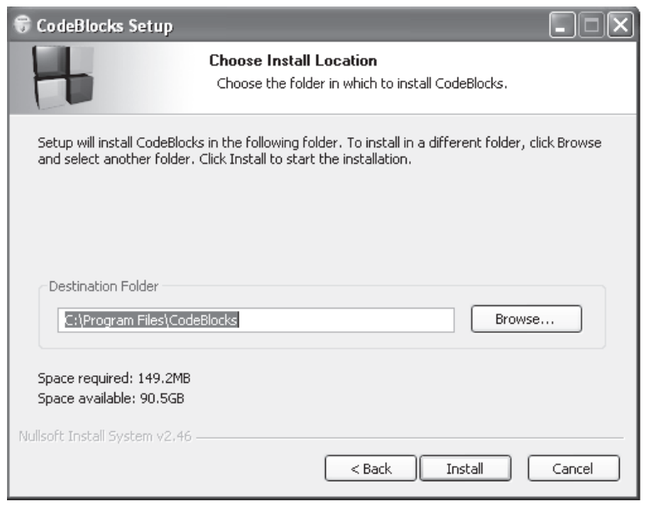

➢ Spragtelėkite mygtuką *I Agree* patvirtinti, kad sutinkame su licencijos reikalavimais.



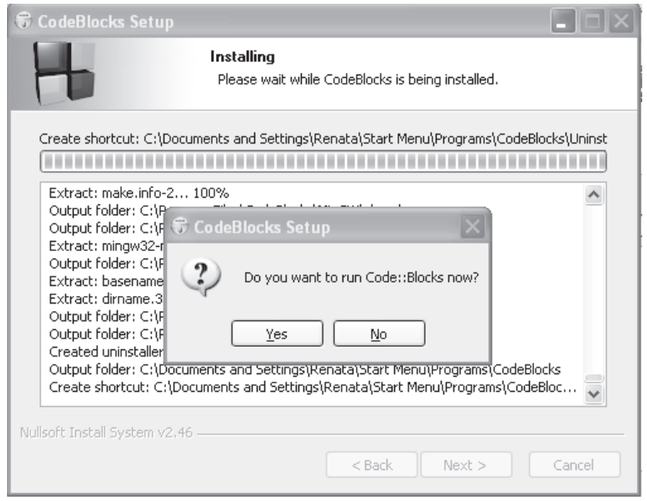

➢ Pasirinkite diegimo tipą ir spragtelėkite mygtuką *Next*:



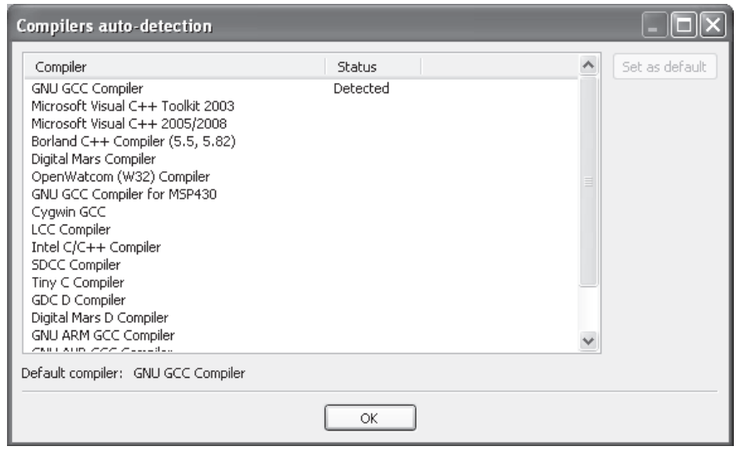

➢ Pasirinkite, kur įdiegti aplinką *CodeBlocks* ir spragtelėkite mygtuką *Install*.



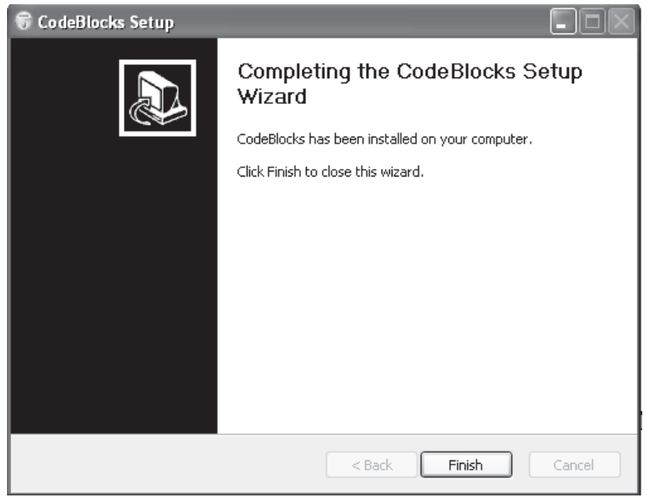
➢ Įdiegus programą, siūloma ją paleisti. Spragtelėkite mygtuką *Yes*.




➢ Pateikiamas aplinkos *CodeBlocks* palaikomų kompiliatorių sąrašas. Numatytasis yra GNU GCC kompiliatorius. Spragtelėkite mygtuką *OK*.

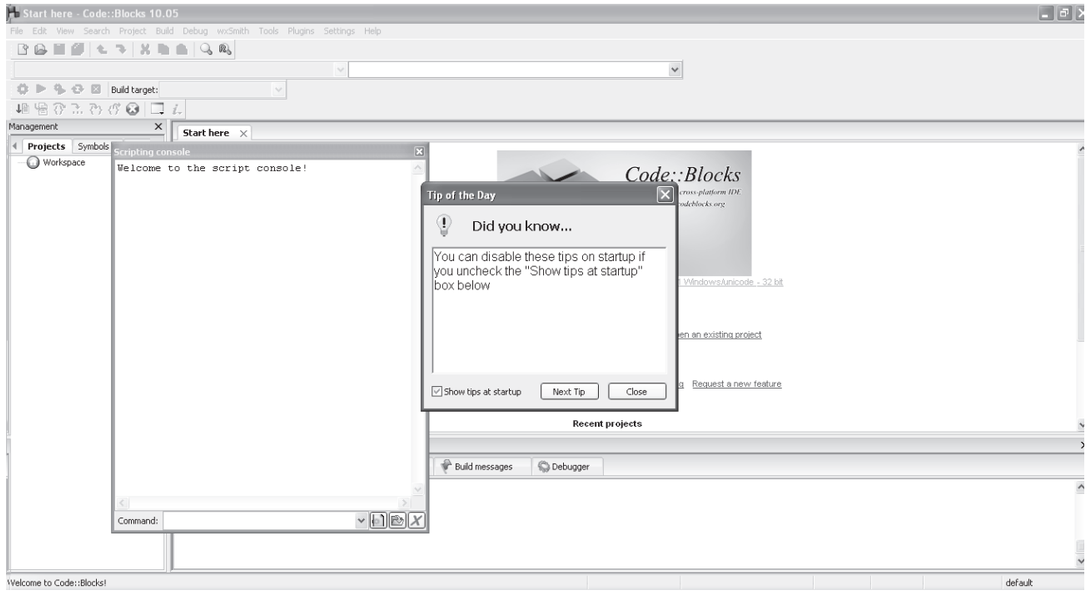



➢ Pranešama, kad diegimas sėkmingai baigtas. Spragtelėkite mygtuką *Finish*.

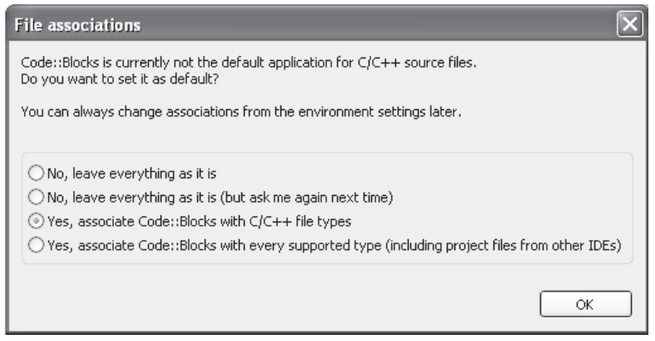




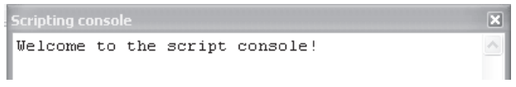

➢ Atsiveria *CodeBlocks* aplinka. Pasirenkame, kad daugiau nebūtų rodomas informacinis pranešimas. Dialogo lange *Tip of the Day* panaikinamas parinkties *Show tips on startup* žymėjimas ir spragtelimas mygtukas *Close*.




➢ Pasirenkama, kad aplinka *CodeBlocks* būtų numatytoji atveriant *C / C++* failus.

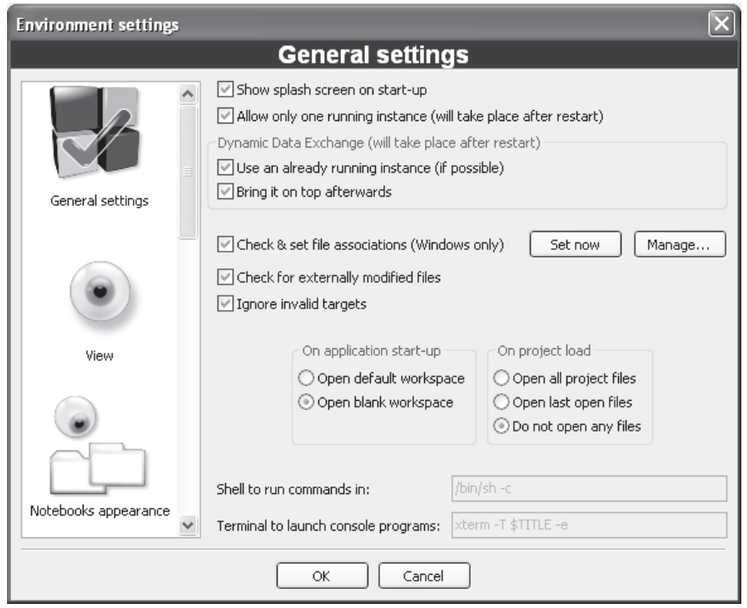



➢ Spragtelimas *Scripting Console* lango užvėrimo mygtukas .

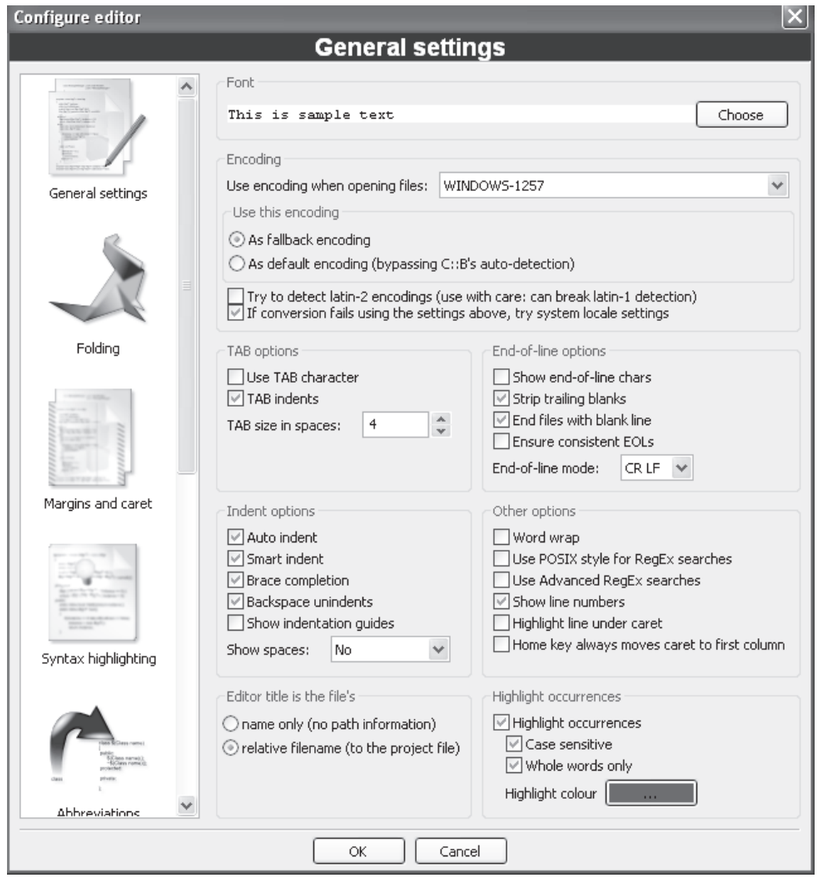



## 5.3. *CodeBlocks* konfigūravimas

➢ Nurodykite pagrindinius aplinkos *CodeBlocks* nustatymus. Pasirinkite komandas: *Settings* → *Environment*... Srityje *On Project load* pažymėkite parinktį *Do not open any files*, kitas parinktis patariama pasilikti numatytąsias.




➢ Pasirinkite rengyklės nustatymus. Pasirinkite komandas: *Settings* → *Editor*... Pasirinkite tokias rengyklės parinktis:






## 5.4. Programos šablono parengimas

Programuojant labai patogu naudotis programos šablonu. Norint sukurti šabloną reikia pasirinkti komandas: *Settings* → *Editor* → *Default Code*. Į *Default Code* dialogo langą įrašykite pateikiamą kodą ir spragtelėkite mygtuką OK.

```
// Vieta programos vardui įrašyti
#include <fcntl.h>
#include <io.h>
#include <iostream>
using namespace std;
int main ()
{
  _setmode (_fileno(stdout), _O_U16TEXT);
  wcout << L"Labas." << endl;
  return 0;
}
```

Pasirinkus naują programos failą, jame jau turėsite ši programos kodo fragmentą.

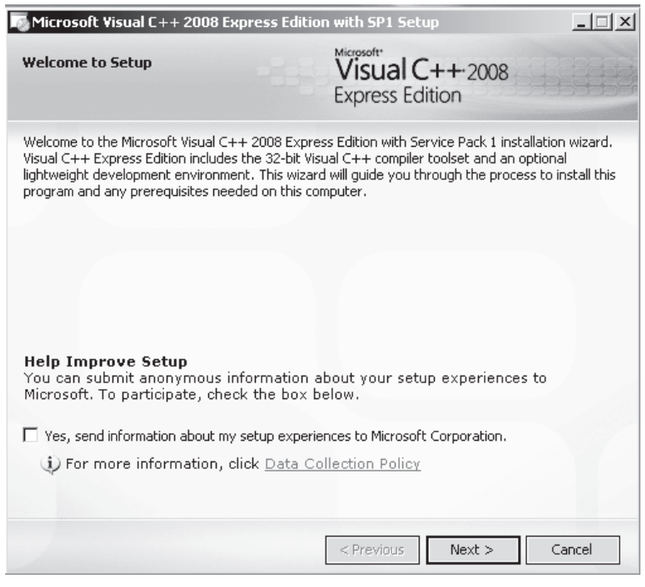
## 5.5. Numatytojo kompiliatoriaus pasirinkimas ir įdiegimas

*CodeBlocks* aplinką sukonfigūruokite taip, kad numatytasis kompiliatorius būtų *Microsoft Visual C++ 2005/2008*. Šis kompiliatorius pasirinktas dėl to, kad juo naudojantis į ekraną galima paprasčiau išvesti lietuviškus rašmenis.

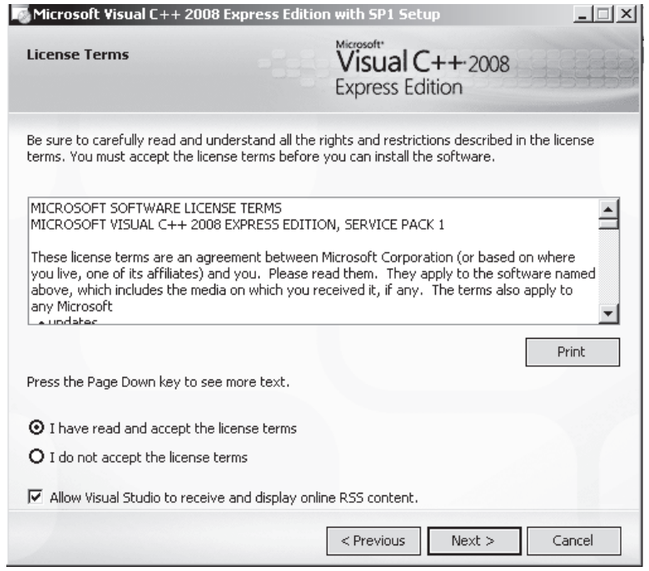
➢ Parsisiųskite *Microsoft Visual C++ 2008 Express Edition*. Jį galima rasti adresu *http://www.brothersoft.com/visual-c++-download-65282.html* ir naudoti mokymui nemokamai.

➢ Įdiekite *Microsoft Visual C++ 2008 Express Edition*.

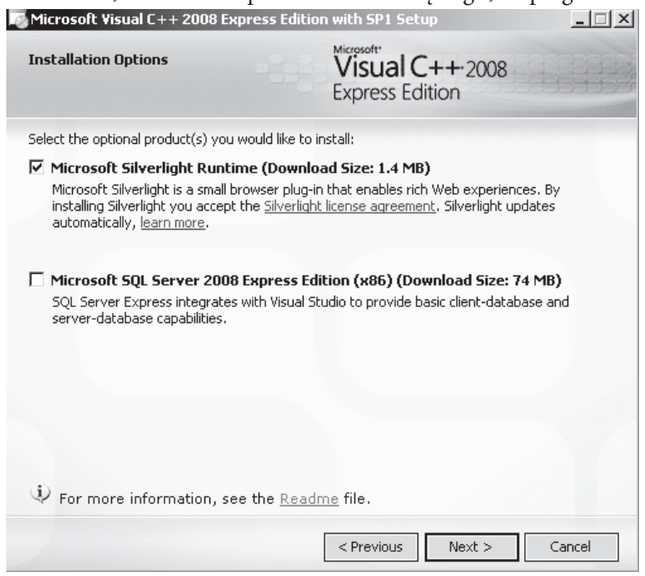
➢ Paleiskite vykdomąjį failą *vcsetup.exe*. Atsivėrusiame lange spragtelėkite mygtuką *Next*.



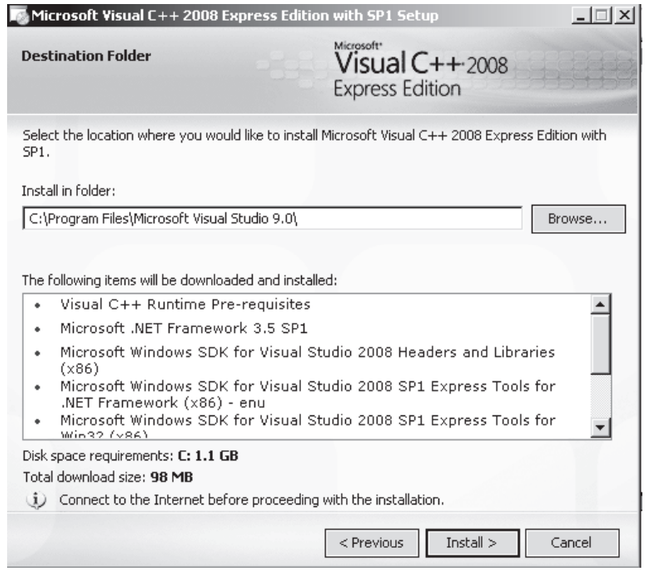
- Pasirinkite parinktį *I have read and accept the license terms* patvirtinti, kad sutinkate su licencijos reikalavimais, ir spragtelėkite mygtuką *Next*.



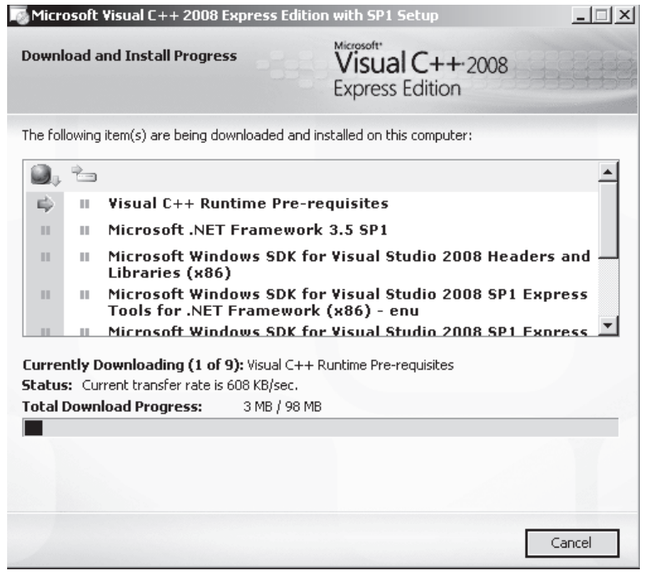

- Pasirinkite, kuriuos komponentus norėsite įdiegti, ir spragtelėkite mygtuką *Next*:




- Pasirinkite, kur norėsite įdiegti, ir spragtelėkite mygtuką *Install*:

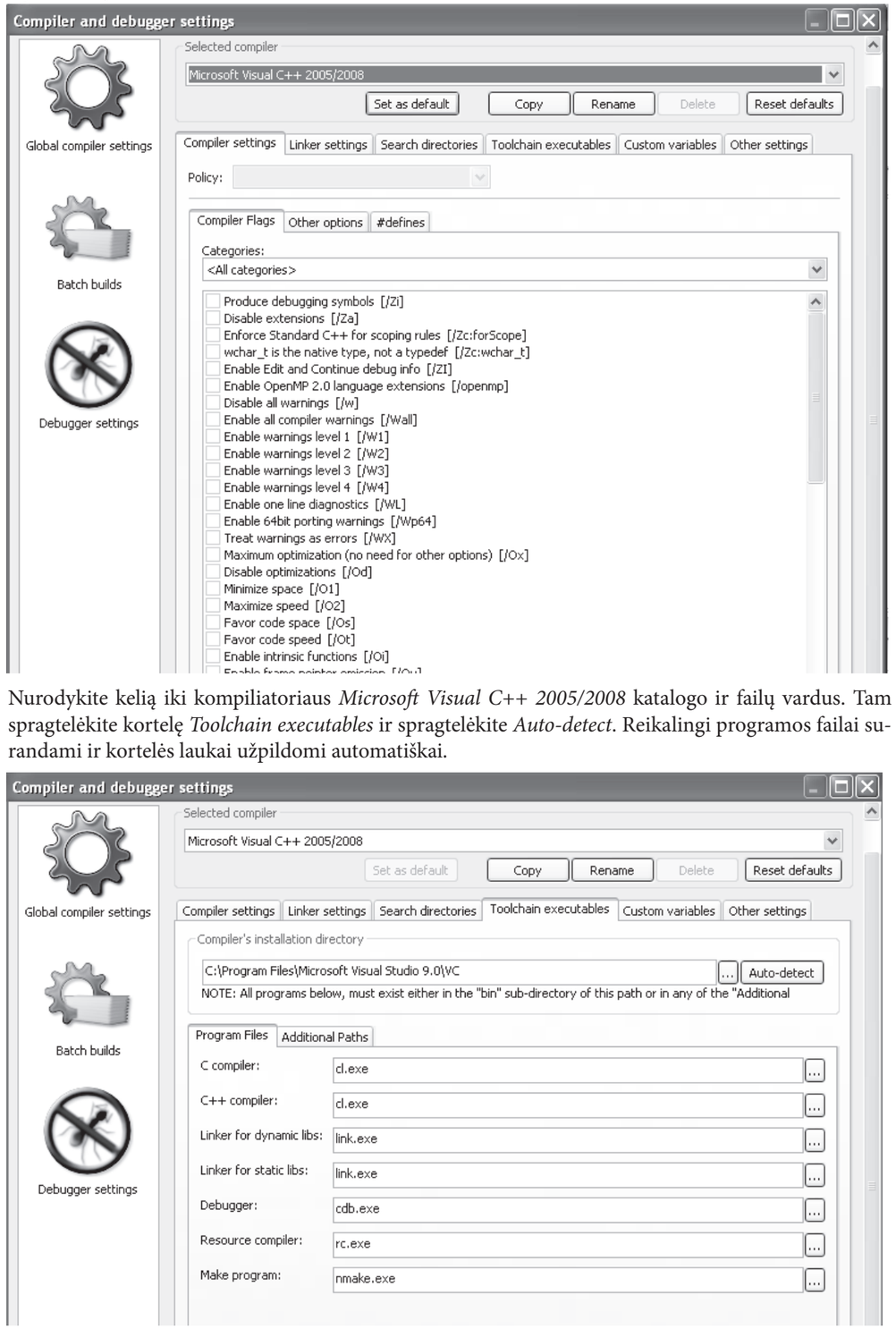



- Iš interneto automatiškai parsiunčiami ir į pasirinktą katalogą įdiegiami reikalingi failai:




- Baigus įdiegti programą, reikia perkrauti kompiuterį tam, kad programa veiktų tinkamai.

➢ Įdiegę *Microsoft Visual C++ 2008 Express Edition*, aplinkoje *CodeBlocks* pasirinkite numatytuoju kompiliatorių *Microsoft Visual C++ 2005/2008*. Pasirinkite komandas: *Settings* → *Compiler* and *debugger*. Atsivėrusiame lange pasirinkite kompiliatorių *Microsoft Visual C++ 2005/2008* ir spragtelėkite mygtuką *Set as default*.

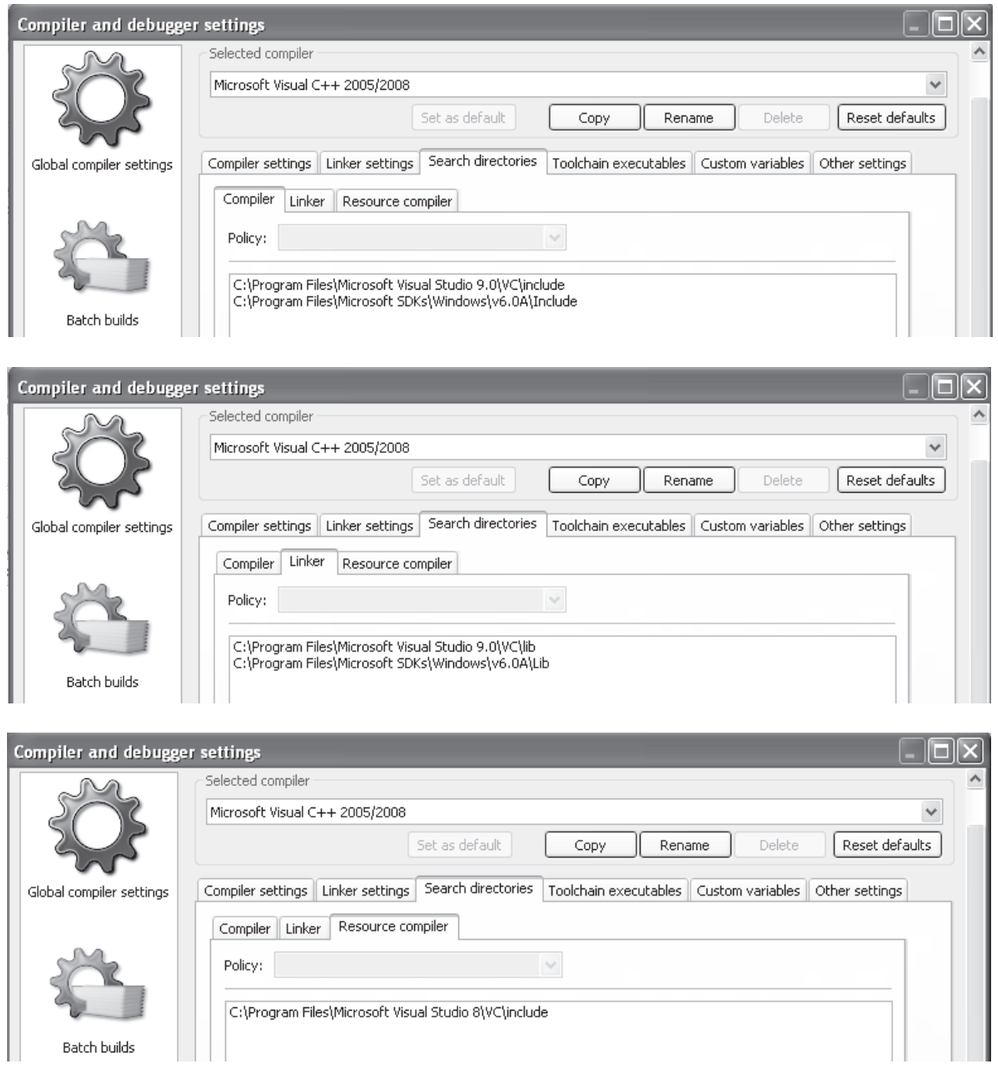



➢ Nurodykite kelią iki kompiliatoriaus *Microsoft Visual C++ 2005/2008* katalogo ir failų vardus. Tam spragtelėkite kortelę *Toolchain executables* ir spragtelėkite *Auto-detect*. Reikalingi programos failai surandami ir kortelės laukai užpildomi automatiškai.






➢ Spragtelėjus kortelę *Search directories*, reikia įkelti nuorodas į *Compiler*, *Linker* ir *Resource compiler* laukus. Laukeliai užpildomi spragtelėjus mygtuką *Add*, esantį lango apačioje, ir pasirinkus norimą katalogą mygtuku *Browse*:



# 6 SAVARANKIŠKO DARBO UŽDUOTYS

**1. Nutrinti skaičiai**. Ant popieriaus lapo užrašyti keturi natūralieji skaičiai: *a*, *b*, *s*, *d*. Po to du iš jų buvo nutrinti (juos žymėsime nuliais). Reikia atkurti nutrintuosius skaičius, jeigu žinoma, kad yra likęs bent vienas iš skaičių *a* ir *b* ir kad skaičiai tenkino šitokias lygybes:

$s = a + b$;
$d = a * b$.

*Pavyzdžiai:*

| Testo Nr. | Pradiniai duomenys | Rezultatai | 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* |

(VIII olimpiada, 1997)

**2. Degtukai**. Yra *n* degtukų. Parašykite programą, kuri nustatytų, ar iš tų degtukų galima sudėti bent vieną iš šių figūrų: lygiakraštį trikampį, kvadratą ar stačiakampį. Dėliojamai figūrai turi būti panaudoti visi degtukai; be to, degtukų laužyti negalima.

*Pavyzdžiai*:

| 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 nė vienos figūros |

(VIII olimpiada, 1997)

**3. 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 taip:

- 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.* Žinomi metai *m* ($1800 \le m \le 2200$). Parašykite programą, kuri nustatytų ir išspausdintų, kokia tų metų spalva.



*Pavyzdžiai*:

| 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` | Paskutiniai 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 |

(XIII olimpiada, 2002)

**4. 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 apskaičiuotų bendrą Lauros parašytų skaitmenų skaičių.

*Pradiniai duomenys*. Pirmoje ir vienintelėje pradinių duomenų failo *skait.in* eilutėje pateiktas paskutinis 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.

*Pavyzdžiai*:

| Pradinis duomuo | 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 |

(XIX olimpiada, 2008)

**5. Žiogas**. Žiogas tupi ant horizontaliai ištemptos virvutės, kairiajame gale. Virvutės ilgis yra *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).

*Užduotis*. Parašykite programą, kuri apskaičiuotų, kiek mažiausiai šuolių turi padaryti žiogas, kad pasiektų mazgą.

*Pradiniai duomenys* – natūralieji skaičiai *s*, *a*, *b*, *c* ($s > c > a > b > 0$).

*Pavyzdžiai*:

| 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 reikėtų 9 šuolių) |
| 5 | 100 13 1 27 | `15` | Žiogas šoka dideliais žingsniais į priekį ir mažais atgal |

(VIII olimpiada, 1997)

**6. Sultis gerti sveika**. Kiekvienos savaitės pirmadienio rytą Jonas gauna *k* centų kišenpinigių. 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?

*Užduotis*. Parašykite programą šiam uždaviniui spręsti.

*Pradiniai duomenys* – natūralieji skaičiai $k$, $s$, $b$, $n$, $i$ ($1 \le i \le 7$). Jei nusprendžiama sultis pirkti pirmadienį, tai iš karto gaunami kišenpinigiai (pasipildo turima pinigų suma). Už pirmadienį gautus kišenpinigius sultys perkamos pirmadienį.

*Pavyzdžiai*:

| 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 pirmadienį, 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 ir viena diena yra pirmadienis |

(IX olimpiada, 1998)

**7. Vaizdo įrašai**. 240 minučių trukmės vaizdajuostė kainuoja 10 Lt ir 90 ct, 180 minučių trukmės vaizdajuostė kainuoja 9 Lt ir 15 ct. 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 galima įrašyti kelias laidas. Deja, Tomas neturi nė vienos tuščios vaizdajuostės.

*Užduotis.* Parašykite programą, kuri apskaičiuotų, kiek mažiausiai pinigų (litų ir centų) Tomui reikės išleisti vaizdajuostėms nusipirkti, kad jų užtektų norimoms laidoms įrašyti.

*Pradiniai duomenys.* Pirmoje kiekvieno duomenų rinkinio eilutėje nurodytas laidų skaičius $n$, o likusiose $n$ eilučių – kiekvienos laidos trukmė. Pirma nurodomas valandų, paskui minučių skaičius.

*Pavyzdžiai*:

| Testo Nr. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Pradiniai duomenys | 1<br>1 10 | 3<br>2 00<br>2 00<br>0 30 | 4<br>0 30<br>0 20<br>3 40<br>1 30 | 3<br>2 30<br>2 30<br>1 01 | 5<br>1 10<br>1 05<br>1 00<br>4 45<br>1 00 | 5<br>1 10<br>1 05<br>1 00<br>4 45<br>1 05 | 3<br>5 30<br>3 20<br>2 00 | 3<br>5 00<br>5 00<br>3 00 | 3<br>5 00<br>5 00<br>3 05 | 6<br>3 00<br>3 00<br>3 00<br>3 00<br>3 00<br>0 55 | 6<br>8 30<br>8 30<br>8 30<br>8 30<br>8 30<br>2 00 |
| Rezultatas | `9,15` | `18,30` | `18,30` | `20,05` | `27,45` | `29,20` | `30,95` | `38,35` | `40,10` | `43,60` | `125,55` |
| Paaiškinimai | | | | | | | | | | | |
| 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 |

(XI olimpiada, 2000)

**8. Didmeninis pirkimas**. Žinoma, kad, perkant daugiau prekių, jų vienetas kainuoja pigiau. Ryšulyje yra 12 porų kojinių, dėžėje – 12 ryšulių. Pavyzdžiui, kojinių dėžė kainuoja 247 Lt, ryšulys – 21 Lt, pora – 2 Lt. Įdomu tai, kad jei mums reikėtų 11 porų kojinių, tai geriau pirkti ryšulį ir vienas kojines kam nors atiduoti.

*Užduotis*. Pirkėjas nori įsigyti $n$ porų kojinių. Parašykite programą, kuria vadovaudamiesi pigiausiai nupirktume kojines. Jei už tą pačią kainą galima nupirkti didesnį ir mažesnį kiekį kojinių, tai perkamas didesnis kiekis. Apskaičiuokite perkamų dėžių, ryšulių ir porų skaičių.



*Pradinius duomenis* sudaro keturi skaičiai: kojinių porų skaičius $n$, vienos dėžės, vieno ryšulio ir vienos poros kaina litais.

*Pavyzdžiai:*

| Testo Nr. | Pradiniai duomenys | Rezultatai | Paaiškinimai |
|---|---|---|---|
| 1 | 720 303 32 3 | `Dėžių skaičius: 5`<br>`Ryšulių skaičius: 0`<br>`Porų skaičius: 0` | $n$ dalus iš 144 (porų skaičiaus dėžėje) |
| 2 | 84 210 20 2 | `Dėžių skaičius: 0`<br>`Ryšulių skaičius: 7`<br>`Porų skaičius: 0` | $n$ dalus iš 12 (porų skaičiaus ryšulyje) |
| 3 | 196 100 20 5 | `Dėžių skaičius: 2`<br>`Ryšulių skaičius: 0`<br>`Porų skaičius: 0` | Perkamos tik dėžės; perkant dėžių, ryšulių ir porų atitinkamai 1, 4, 4 arba 1, 5, 0, kaina būtų ta pati, tačiau kojinių kiekis būtų mažesnis |
| 4 | 355 292 41 6 | `Dėžių skaičius: 2`<br>`Ryšulių skaičius: 6`<br>`Porų skaičius: 0` | Perkamos tik dėžės ir ryšuliai |
| 5 | 355 291 50 6 | `Dėžių skaičius: 3`<br>`Ryšulių skaičius: 0`<br>`Porų skaičius: 0` | Perkamos tik dėžės |
| 6 | 355 292 49 6 | `Dėžių skaičius: 2`<br>`Ryšulių skaičius: 5`<br>`Porų skaičius: 7` | Perkamos ir dėžės, ir ryšuliai, ir poros |

(X olimpiada, 1999)

**9. Varlių koncertas**. Kartą vienoje kūdroje gyveno daug varlių, ir ne bet kokių, o dresuotų. Kiekviena varlė sugebėdavo iššokti iš vandens ir sukvarksėti jai būdingais tiksliai pasikartojančiais laiko momentais. Pavyzdžiui, tarkime, kad varlės kvarksėjimo periodas lygus 5. Vadinasi, 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*. 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 $p_1$ – pirmos varlės pasirodymo periodas, $p_2$ – antros ir t. t. ($0 < p_i \le 20$ minučių). Varlių skaičius kūdroje neviršijo 10.

Atkreipiame dėmesį, kad skaičius valandų, po kurių įvyks antrasis koncertas, neviršija `long long`.

*Pavyzdžiai*:

| Testo Nr. | Pradiniai duomenys | Rezultatai | Paaiškinimai |
|---|---|---|---|
| 1 | 2<br>8 8 | `Koncertas įvyks po`<br>`0 val. ir 8 min.` | Paprasčiausias atvejis: abu periodai sutampa |
| 2 | 3<br>2 3 5 | `Koncertas įvyks po`<br>`0 val. ir 30 min.` | Periodai neturi bendrų daliklių: kartotinis lygus periodų sandaugai |
| 3 | 6<br>12 15 20 10 6 30 | `Koncertas įvyks po`<br>`1 val. ir 0 min.` | Periodai turi bendrų daliklių |
| 4 | 5<br>18 7 3 2 4 | `Koncertas įvyks po`<br>`4 val. ir 12 min.` | Tikrinama, ar laikas teisingai perskaičiuojamas į valandas ir minutes |
| 5 | 10<br>11 12 13 14 15 16 17 18 19 20 | `Koncertas įvyks po`<br>`3879876 val. ir 0 min.` | Šiuo testu tikrinamas sprendimo efektyvumas |

(XI olimpiada, 2000)

**10. Senas kalendorius.** Metai yra keliamieji, jeigu jie dalūs iš 4 ir nesidalija iš 100 arba jeigu jie dalūs iš 400. *Pavyzdžiui*:

- 2000-ieji metai keliamieji, nes jie dalūs iš 400;
- 2004-ieji metai keliamieji, nes jie dalūs iš 4 ir nėra dalūs iš 100;
- 1900-ieji metai nėra keliamieji, nes jie dalūs iš 100, bet nėra dalūs 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, įvedami klaviatūra. Pirmas skaičius – tai metai, kuriems buvo išleistas senasis kalendorius. Antrasis skaičius yra iš intervalo [1..7] ir nusako, kuria savaitės diena prasidėjo tie metai.

*Rezultatas* išvedamas į ekraną.

*Pavyzdžiai*:

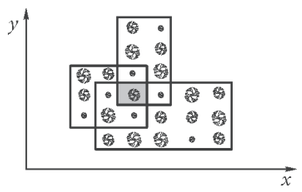
| Testo Nr. | Pradiniai duomenys | Rezultatas |
|---|---|---|
| 1 | 1979 1 | 2007 |
| 2 | 1941 3 | 2014 |
| 3 | 1964 3 | 2020 |
| 4 | 1993 5 | 2010 |
| 5 | 2002 2 | 2013 |

(XVI olimpiada, 2005)

**11. Trys sodininkai**. Trys draugai, apsigyvenę kaime, nusprendė mokytis sodininkauti. Kaime buvo didžiulis sodas, kurio kiekviename vienetiniame plotelyje augo po vieną vaismedį.

Kiekvienas iš trijų draugų pasirinko stačiakampį sklypą ir nusprendė prižiūrėti jame esančius medžius. Susirinkus draugėn paaiškėjo, kad jų pasirinkti sklypai įsiterpia vienas į kitą, t. y. kai kuriuos vaismedžius prižiūrės ne vienas, o keletas sodininkų.

*Užduotis*. Parašykite programą, kuri apskaičiuotų, kiek vaismedžių panorėjo prižiūrėti visi trys draugai.




*Pradiniai duomenys* pateikiami trijose tekstinio failo *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.

*Pavyzdžiai*:

| Testo Nr. | Pradiniai duomenys | Rezultatas |
|---|---|---|
| 1 | 2 3 6 7<br>4 1 8 5<br>6 0 10 3 | 0 |
| 2 | 1 5 9 9<br>5 6 13 12<br>6 1 9 11 | 9 |
| 3 | 0 4 6 8<br>0 4 6 8<br>0 0 6 4 | 0 |
| 4 | 30 30 80 70<br>10 20 70 90<br>50 20 100 90 | 800 |

(X olimpiada, 1999)



## Rekomenduojama literatūra


1. J. Blonskis, V. Bukšnaitis, V. Jusas, R. Marcinkevičius, D. Rubliauskas. *Programavimas C++*. Vadovėlis. KTU leidykla *Technologija*, Kaunas, 2005.
2. J. Blonskis, V. Bukšnaitis, V. Jusas, R. Marcinkevičius, A. Misevičius, S. Turskienė. *Programavimo kalba C++. Mokomoji knyga*. Smaltijos leidykla, Kaunas, 2008.
3. J. Blonskis, V. Bukšnaitis, J. Končienė, D. Rubliauskas. *C++ praktikumas*. KTU leidykla *Technologija*, Kaunas, 2001.
4. J. Blonskis, V. Bukšnaitis, V. Jusas, R. Marcinkevičius, A. Misevičius. *C++ Builder*. Mokomoji knyga. Smaltijos leidykla, Kaunas, 2005.
5. A. Vidžiūnas. *C++ ir objektinis programavimas. Programuotojo vadovas*. Smaltijos leidykla, Kaunas, 2008.
6. H. M. Deitel, P. J. Deitel. *C++ How to Program*. Prentice Hall, 2001.
7. V. Dagienė, G. Grigas, T. Jevsikova, *Enciklopedinis kompiuterijos žodynas*. II papildytas leidimas. TEV, Vilnius, 2008. Žodyno svetainė: *www.likit.lt/term/enciklo.html*.