

Renata Burbaitė  
Jonas Blonskis  
Vytautas Bukšnaitis



# Šiuolaikiškas žvilgsnis į programavimą

# C++

Pasirenkamasis informacinių  
technologijų kursas  
XI–XII klasėms

# TURINYS

IVADAS	5
1. PRAKTIKOS DARBAI	6
1.1. Kartojimas	8
1.2. Duomenų skaitymas iš failo ir rezultatų rašymas į failą	14
1.3. Ciklas cikle	20
1.4. Funkcija, grąžinanti apskaičiuotą reikšmę per funkcijos vardą	27
1.5. Funkcija su parametrais-nuorodomis	36
1.6. Pažintis su masyvu	42
1.7. Didžiausios ir mažiausios reikšmių paieška	51
1.8. Didžiausios ir mažiausios reikšmės vieta	58
1.9. Masyvo elementų paieška ir jų šalinimas iš masyvo	65
1.10. Reikšmių įterpimas į masyvą	74
1.11. Simboliai	82
1.12. Simboliai ir skaičiai	90
1.13. Simbolių eilutės	97
1.14. Pažintis su struktūros duomenų tipu	103
1.15. Paieška nesutvarkytame sąrašė	112
1.16. Duomenų atranka	119
1.17. Duomenų šalinimas ir papildymas	129
2. C++ KALBOS IR DUOMENŲ STRUKTŪRŲ ŽINYNAS	141
2.1. Konstantos	141
2.2. Operatoriai	142
2.3. Kintamasis, rodyklė, adresas	144
2.4. Duomenų skaitymas iš failo	146
2.5. Rezultatų (duomenų) rašymas į failą	147
2.6. Funkcijos	149
2.7. Masyvas	151
2.8. Simbolių eilutė <code>char[]</code>	156
2.9. Simbolių eilutė <code>string</code>	162
2.10. Struktūra	164
2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	169
3. ALGORITMŲ ŽINYNAS	170
Rekomenduojama literatūra	176
Naudingos nuorodos	176

Vedono gimnazijos informacinių technologijų mokytojas ekspertis Tarpinis šaltinis, jo išskaitus pasta-  
kas ir siūlymas buvo labai naudingi.

Tikimės, kad šiuose nuotraukose programavimo pagrindų mokymosi laidas bus jums įdomus, suprantamas ir  
naudingas.

Sėkmės!

Knygos autorius

UDK 004.42(075.3)

Bu287

Valstybinės lietuvių kalbos komisijos 2011-06-20 posėdžio nutarimu Nr. S2-420(6.1)  
vadovėlis atitinka kalbos taisyklingumo reikalavimus

Projekto vadovė *Ieva Mackevič*

Redaktorė *Zita Manstavičienė*

Kompiuterinė grafika: *Edita Tatarinavičiūtė*

Maketavo *Silvestra Markuckienė*

Recenzentai: doc. dr. *Antanas Vidžiūnas*,  
informacinių technologijų mokytoja ekspertė *Tatjana Balvočienė*

Skaitmeninė vadovėlio versija – svetainėje <http://www.vadovėliai.lt>

Vadovėlio interneto svetainė <http://it.vadovėliai.lt>

© Leidykla TEV, Vilnius, 2011

© Renata Burbaitė, 2011

© Jonas Blonskis, 2011

© Vytautas Bukšnaitis, 2011

© Viršelio dail. *Jadvyga Butrimienė*, 2011

ISBN 978-609-433-040-7

Renata Burbaitė, Jonas Blonskis, Vytautas Bukšnaitis

IVADAS	1
<b>I. PRAKTIKOS DARBAI</b>	6
1.1. Kartojimas	8
1.2. Duomenų skaitymas iš failo ir rezultatų rašymas į failą	14
1.3. Ciklas cikle	20
1.4. Funkcija, grafinanti apskaičiuotą reikšmę per funkcijos vardą	27
1.5. Funkcija su parametrais nuorodomis	36
1.6. Pažintis su m	37
1.7. Didžiausios ir mažiausios reikšmės vieta	40
1.8. Didžiausios ir mažiausios reikšmės vieta	40
1.9. Masyvo elementai	47
1.10. Reikšmių lyginimas	47
1.11. Simboliai	58
1.12. Simboliai ir skaičiai	60
1.13. Simbolių elius	78
1.14. Pažintis su	101
1.15. Paieška nes	112
1.16. Duomenų ar	119
1.17. Duomenų šalinimas ir papildymas	121
<b>2. C++ KALBOS IR DUOMENŲ STRUKTŪRŲ ŽINYNAS</b>	131
2.1. Konstantos	131
2.2. Operatoriai	132
2.3. Kintamasis, rodyklė, adresas	144
2.4. Duomenų skaitymas iš failo	146
2.5. Rezultatų (duomenų) rašymas į failą	147
2.6. Funkcijos	149
2.7. Masyvas	151
2.8. Simbolių elius char[]	151
2.9. Simbolių elius string	161
2.10. Struktūra	161
2.11. Knypie naudojamų tarpiausių failų sąrašas	169
<b>3. ALGORITMŲ ŽINYNAS</b>	170
Rekomenduojama literatūra	170
Naudingos nuorodos	171

# Šiuolaikiškas žvilgsnis į programavimą

## C++

Pasirenkamasis informacinių technologijų kursas  
XI–XII klasėms



VILNIUS 2011



Šis knygos įvadas yra pirmasis dokumentas (pvz., C++ programos) su informacija, kuri padės dokumentuoti šį vadą yra ta pati. Be to, gali būti naudinga informacija dokumente turi būti laikoma.

Pirmus žingsnius į programavimą aprašėme knygoje „Šiuolaikiškas žvilgsnis į programavimo pagrindus. C++“. Pasirenkamasis informacinių technologijų kursas IX–X klasėms“. Šioje knygoje tęsiame pažintį su programavimu.

Knygoje rasite septyniolika praktikos darbų, kuriems atlikti reikia sudėtingesnių programavimo priemonių. Praktikos darbai pateikiami C++ programavimo kalba. Darbui naudojama *CodeBlocks* programavimo aplinka.

Pirmieji praktikos darbai skirti ankstesnėse klasėse įgytoms programavimo žinioms pakartoti. Tik antrame ir trečiame praktikos darbuose duomenų šrautai jau siejami ne su ekranu ir klaviatūra, bet su failais.

Ketvirtas ir penktas praktikos darbai supažindina skaitytoją su viena pagrindinių programavimo priemonių – funkcija. Skaidant programos veiksmus į atskiras savarankiškas dedamąsias dalis (funkcijas), programavimo procesas tampa paprastesnis, vieną programą gali rašyti grupė programuotojų. Tokią programą lengviau modifikuoti, papildyti. Esant reikalui, parašytas funkcijas galima pritaikyti kitoms programoms.

Praktinę naudą turi tik tos programos, kurios apdoroja didelius duomenų rinkinius. Šeštas–dešimtas praktikos darbai supažindina su svarbiausia duomenų struktūra – masyvu. Masyve galima laikyti daug to paties tipo reikšmių. Knygoje nuosekliai pateikiami darbo su masyve laikomomis reikšmėmis algoritmai: tie, kurie jau žinomi, ir nauji (reikšmių skaitymas / rašymas; sumos, sandaugos, kiekio, aritmetinio vidurkio skaičiavimas; mažiausios, didžiausios reikšmės paieška; rikiavimas; reikšmių šalinimas, įterpimas).

Visi skaičiai realiaame pasaulyje siejami su pavadinimais. Tai neatskiriama sudėtinė apdorojamų duomenų dalis. Pavadinimai kompiuteryje – tai simbolių eilutės. Vienuoliktame ir dvyliktame praktikos darbuose nagrinėjami simbolių masyvai. Suteikiant papildomų savybių, masyvai tampa simbolių eilutėmis.

Tryliktas–septynioliktas praktikos darbai supažindina su struktūros duomenų tipu, kuris susieja įvairių tipų kintamuosius. Tai įrašo duomenų bazėse analogas. Čia naudojami ankstesniuose darbuose pateikti algoritmai.

Kiekvieno praktikos darbo pradžioje rasite sąrašą žinių ir gebėjimų, kuriuos įgysite atlikdami praktikos darbą. Pateikiamos nuorodos į skyrius *C++ kalbos ir duomenų struktūrų žinynas* bei *Algoritmų žinynas*. *C++ kalbos ir duomenų struktūrų žinyne* apibūdinamos C++ programavimo kalbos pagrindinės priemonės ir konstrukcijos. Čia supažindinama su masyvais, simbolių eilutėmis ir struktūromis. *Algoritmų žinyne* aprašomi klasikiniai algoritmai, naudojami įvairaus tipo praktinėms 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 yra užduočių, kurios padės kiekvienam įsivertinti įgytas žinias ir įgūdžius. Tai neprivaloma, tačiau siūlome šias užduotis pasinagrinėti ir atlikti. Jei kurios nors jų yra per sunkios, nepraleiskite, pasistenkite jas įveikti išnagrinėję teorinę medžiagą, konsultuodamiesi su mokytoju ar klasės draugais.

Nagrinėdami pateikiamus darbus, susipažinsite su programavimo priemonėmis, duomenų tipais (masyvu, struktūra), algoritmais, tačiau programuoti neišmoksitė. Programavimas – kūrybinis procesas. Išmokti programuoti galima tik savarankiškai rašant programas. Stenkitės kurti tokias duomenų struktūras, kurios būtų ne tik patogios duomenims laikyti ir jiems apdoroti, bet kad ir pačios programos būtų racialesnės ir universalesnės. Todėl nebijokite eksperimentuoti taikydami įgytas žinias.

Nuoširdžiai dėkojame Vytauto Didžiojo universiteto informatikos fakulteto doc. dr. *Antanui Vidžiūnui* ir Šilutės Vydūno gimnazijos informacinių technologijų mokytojai ekspertei *Tatjanai Balvočienei*. Jų išsakytos pastabos ir siūlymai buvo labai naudingi.

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

Sėkmės!

*Knygos autoriai*



## Programos struktūra

Vadovėlio praktikos darbuose pateikiamų programų struktūra tokia, kokia yra nurodoma daugelyje šaltinių (žr. dešinėje pateiktą pavyzdį).

Kiekviena struktūrinė programos dalis nuo kitų atskiriama komentaru – brūkšnelių eilute. Jeigu kurios nors dalies kuriamoje programoje nėra, tuomet nereikia rašyti ir komentaro.

Prieš kiekvienos sukurtos funkcijos antraštę nurodoma funkcijos ir jos parametrų (jei jų yra) paskirtis.

### Programos teksto pavyzdys

```
// Trikampių perimetrai
#include <fstream>
using namespace std;
//-----
const char CDfv[] = "Duomenys.txt";
const char CRfv[] = "Rezultatai.txt";
//-----
bool ArTrikampis(int a, int b, int c);
int Perimetras(int a, int b, int c);
//-----
int main()
{
    int n; // atkarpų trejetų skaičius
    int a, b, c; // atkarpų ilgiai
    ifstream fd(CDfv);
    ofstream fr(CRfv);
    fd >> n;
    for (int i = 1; i <= n; i++) {
        fd >> a >> b >> c;
        if (ArTrikampis(a, b, c))
            fr << "Trikampio perimetras: " << Perimetras(a, b, c) << endl;
        else fr << "Trikampio sudaryti negalima";
    }
    fd.close();
    fr.close();
    return 0;
}
//-----
// Tikrinama, ar susidaro trikampis
// Gražina true – jeigu iš atkarpų a, b, c galima sudaryti trikampį ir false – priešingu atveju
bool ArTrikampis(int a, int b, int c)
{
    if ((a + b > c) && (a + c > b) && (b + c > a)) return true;
    else return false;
}
//-----
// Apskaičiuoja ir gražina trikampio, kurio kraštinių ilgių yra a, b, c, perimetrą
int Perimetras(int a, int b, int c)
{
    return a + b + c;
}
```

Programos pavadinimas, užrašomas kaip komentaras
Nurodomi įterpiamieji failai
Aprašomos konstantos
Aprašomi programuotojo sukurti duomenų tipai
Nurodomi funkcijų prototipai
Funkcija main()
Pateikiamos programuotojo sukurtos funkcijos

## Programos šablonas

Rengiant tam tikro tipo dokumentus (pvz., C++ programas), yra patogu naudotis šablonu (pavyzdžiu). Jame pateikiama informacija, kuri panašiuose dokumentuose visada yra ta pati. Be to, gali būti nurodoma, kokia informacija dokumente turi būti keičiama.

Norint sukurti šabloną aplinkoje *CodeBlocks* (iš kur ją galima atsisiųsti ir kaip įdiegti, detalai paašškinta vadovėlyje „Šiuolaikiškas žvilgsnis į programavimo pagrindus. C++“. Pasirenkamasis informacinių technologijų kursas IX–X klasėms“, žr. naudingų nuorodų sąrašą), reikia pasirinkti komandas *Settings* → *Editor* → *Default Code*, po to į dialogo langą įrašyti reikiamą kodą ir spragtelėti mygtuką *OK*.

### C++ programos šablonas

```
// Vieta programos vardui įrašyti
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cmath>
using namespace std;
//-----
int main()
{
    return 0;
}
```

Mūsų sukurtame C++ programų šablone įrašyti sakiniai, reikalingi paprasčiausiai veikiančiam programui. Panairinėkime šį šabloną.

Pirmoje eilutėje yra komentaras:

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

Jis programos darbui jokios įtakos neturi. Komentaro tekstą programuotojas gali keisti: nurodyti programos vardą, trumpai apibūdinti jos paskirtį ir kt. Jei reikia, šį komentarą galima praplėsti iki kelių eilučių.

Programos pradžioje surašytos instrukcijos *parengiamajai doroklei* (angl. *preprocessor*). Jos žymimos simboliu #. Įterpimo instrukcijomis *include* nurodoma, kokių failų tekstai turi būti įterpti pažymėtose vietose pirminio apdorojimo metu. Įterpiamųjų failų vardai rašomi tarp simbolių < >. Šablone pateikti praktikos darbuose naudojami įterpiamieji failai.

Įterpiamasis failas	Paašškimas
<i>iostream</i>	Duomenų įvedimo klaviatūra ir rodymo ekrane priemonės
<i>fstream</i>	Duomenų skaitymo iš failo ir rašymo į failą priemonės
<i>iomanip</i>	Duomenų išvedimo į failų srautus (ekraną, failą) priemonės
<i>cmath</i>	Matematinė funkcijų rinkinys

Sakinys `using namespace std;` rašomas visada, jei į programą įterpiamas bent vienas antraštinis failas (pvz., *iostream*).

Toliau rašoma programos pagrindinės funkcijos antraštė: `int main()`

Pagrindinės funkcijos kamienas (veiksmų sritis) pradedamas ženklų {, baigiamas ženklų }.

Sakinys `return 0;` nurodo programai baigti funkcijos *main()* darbą.

Pasirinkę naują programos failą, jau turėsite šablone įrašytą kodą.

Jei nėra programos ir jos naudotojo dialogo, nenurodyta pradinius duomenis įvesti klaviatūra, rezultatus rodyti ekrane, tuomet, sukompilavę ir įvykdę programą, ekrane turėtumėte matyti informacinį pranešimą, pavyzdžiui, tokį:

```
Process returned 0 (0x0)   execution time : 0.219 s
Press any key to continue.
```

# 1.1. Kartojimas

Atlikdami šį darbą, prisiminsite IX–X klasių kursą:

- ✓ kaip rašomi ciklo ir sąlygos sakiniai;
- ✓ kaip įvedami pradiniai duomenys ir išvedami rezultatai.

## Užduotis

Lenktynės. Petriukas sapnuoja, kad jį vežasi tigras, vilkas ir informatikos mokytojas. Sapno pradžioje Petriukas bėgo labai greitai ir persekiotojus paliko gana toli: tigras atsiliko  $kt$  km, vilkas –  $km, o mokytojas –  $km. Vėliau Petriukas taip pavargo, kad nebegalėjo bėgti greičiau kaip 1 km/val. greičiu. Kas Petriuką pavys, o kas ne, jeigu laikrodis pažadins jį praėjus 8 val. nuo to momento, kai jis pradėjo bėgti lėtai? Tigras vijosi  $km/val., vilkas –  $km/val., o mokytojas –  $km/val. greičiu.$$$$$

## Algoritmas

Užduotis sprendžiama taip:

1. Įvedami duomenys.
2. Skaičiuojami visų besivejančiųjų atstumai iki Petriuko, kol jis pabus.
3. Ekrane parodoma, kurie besivejančiųjų pasivijo Petriuką ir kurie ne.



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

- Aprašykite sveikojo tipo kintamuosius pradiniam duomenims atmintyje laikyti.
- Parašykite kintamųjų reikšmių įvedimo klaviatūra sakinius: pranešimo, kokią reikšmę įvesti, sakinį (cout) ir reikšmės skaitymo sakinį (cin).

```
int main()
{
    int kt,          // kiek kilometrų atsiliko tigras
        kv,          // kiek kilometrų atsiliko vilkas
        km,          // kiek kilometrų atsiliko mokytojas
        gt,          // tigro bėgimo greitis
        gv,          // vilko bėgimo greitis
        gm;          // mokytojo bėgimo greitis
    cout << "Tigras atsiliko: ";    cin >> kt;
    cout << "Vilkas atsiliko: ";    cin >> kv;
    cout << "Mokytojas atsiliko: "; cin >> km;
    cout << "Tigro greitis: ";      cin >> gt;
    cout << "Vilko greitis: ";      cin >> gv;
    cout << "Mokytojo greitis: ";   cin >> gm;
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Klaviatūra įveskite pradinis duomenis, pavyzdžiui, tokius: 49, 79, 127, 7, 5, 25. Įsitikinkite, kad programa dirba teisingai. Ekrane turėtumėte matyti tokį vaizdą:

```
Tigras atsiliko: 49
Vilkas atsiliko: 79
Mokytojas atsiliko: 127
Tigro greitis: 7
Vilko greitis: 5
Mokytojo greitis: 25
```





## 2 Atstumų tarp besivejančiųjų ir Petriuko po 8 valandų skaičiavimas

- Papildykite programą veiksmiais, kuriais būtų skaičiuojami besivejančiųjų atstumai iki Petriuko po 8 valandų.

```
kt = kt - 8 * gt + 8; // tigras nubėgo 8*gt km, o mokinys per tą patį laiką nubėgo 8 km
kv = kv - 8 * gv + 8; // vilkas nubėgo 8*gv km, o mokinys per tą patį laiką nubėgo 8 km
km = km - 8 * gm + 8; // mokytojas nubėgo 8*gm km, o mokinys per tą patį laiką nubėgo 8 km
```

- Įrašykite ir įvykdykite programą. Klaviatūra įveskite duomenis. Įsitikinkite, kad programa dirba.
- Norėdami sužinoti, kokie atstumai Petriuką skirs nuo persekiotojų, kai jis pabus po 8 valandų, programą papildykite tokiu sakiniu:

```
cout << " kt = " << kt << " kv = " << kv << " km = " << km << endl;
```

- Įrašykite ir įvykdykite programą. Ekране turėtumėte matyti, pavyzdžiui, tokį vaizdą:

```
Tigras atsiliko: 49
Vilkas atsiliko: 79
Mokytojas atsiliko: 127
Tigro greitis: 7
Vilko greitis: 5
Mokytojo greitis: 25
kt = 1 kv = 47 km = -65
```

Pažvelgę į programos darbo rezultatus, matome, kad kai Petriukas pabudo, tigrą nuo jo skyrė vienas kilometras, vilką – 47 kilometrai. Mokytojas Petriuką pavijo, pralenkė ir bėgo toliau. Kai Petriukas pabudo, mokytojas jį buvo pralenkęs net 65 km.



## 3 Atstumų, kuriais besivejantieji buvo atsilikę nuo Petriuko kas valandą, skaičiavimas

- Ankstesnių skaičiavimų sakinius apgaubkite ciklo sakiniu, nurodydami kiekvieno besivejančiojo atstumus iki Petriuko skaičiuoti kas valandą:

```
for (int i = 1; i <= 8; i++) {
    kt = kt - gt + 1; // tigras nubėgo gt km, o mokinys per tą patį laiką nubėgo 1 km
    kv = kv - gv + 1; // vilkas nubėgo gv km, o mokinys per tą patį laiką nubėgo 1 km
    km = km - gm + 1; // mokytojas nubėgo gm km, o mokinys per tą patį laiką nubėgo 1 km
}
```

- Įrašykite ir įvykdykite programą. Ekране turėtumėte matyti tokį pat vaizdą, kaip ir po ankstesnio žingsnio.

```
Tigras atsiliko: 49
Vilkas atsiliko: 79
Mokytojas atsiliko: 127
Tigro greitis: 7
Vilko greitis: 5
Mokytojo greitis: 25
kt = 1 kv = 47 km = -65
```

- Pabandykite pildyti lentelę. Patikrinkite, ar programa visais atvejais parodo teisingus rezultatus.



#### 4 Atstumų, kuriais besivejantieji buvo atsilikę nuo Petriuką kas valandą, rodymas ekrane

- Rodymo ekrane sakinį perkeltkite į ciklą. Matysite, kokie atstumai skyrė Petriuką nuo persekiotojų kiekvieną vijimosi valandą.

```
cout << "kt = " << kt << " kv = " << kv << " km = " << km << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti, pavyzdžiui, tokį vaizdą:

```
Tigras atsiliko: 49
Vilkas atsiliko: 79
Mokytojas atsiliko: 127
Tigro greitis: 7
Vilko greitis: 5
Mokytojo greitis: 25
kt = 43 kv = 75 km = 103
kt = 37 kv = 71 km = 79
kt = 31 kv = 67 km = 55
kt = 25 kv = 63 km = 31
kt = 19 kv = 59 km = 7
kt = 13 kv = 55 km = -17
kt = 7 kv = 51 km = -41
kt = 1 kv = 47 km = -65
```



#### 5 Rezultatų, kas pavijo Petriuką, rodymas ekrane

- Iš programos pašalinkite sakinį, kuris parodo ekrane atstumus, skiriančius Petriuką nuo persekiotojų kiekvieną vijimosi valandą.

```
cout << " kt = " << kt << " kv = " << kv << " km = " << km << endl;
```

- Papildykite programą sakiniais, kad ekrane būtų rodomi paaiškinimai, kas pavijo mokinį:

```
if (kt <= 0) cout << "Tigras pavijo" << endl;
else cout << "Tigras nepavijo" << endl;
if (kv <= 0) cout << "Vilkas pavijo" << endl;
else cout << "Vilkas nepavijo" << endl;
if (km <= 0) cout << "Mokytojas pavijo" << endl;
else cout << "Mokytojas nepavijo" << endl;
```

- Įrašykite ir įvykdykite programą. Klaviatūra įveskite pradinis duomenis. Įsitinkite, kad ekrane rodomi pranešimai, kas pavijo Petriuką ir kas ne. Ekrane turėtumėte matyti, pavyzdžiui, tokį vaizdą:

```
Tigras atsiliko: 49
Vilkas atsiliko: 79
Mokytojas atsiliko: 127
Tigro greitis: 7
Vilko greitis: 5
Mokytojo greitis: 25
Tigras nepavijo
Vilkas nepavijo
Mokytojas pavijo
```

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
int main()
{
    int kt,          // kiek kilometrų atsiliko tigras
        kv,          // kiek kilometrų atsiliko vilkas
        km,          // kiek kilometrų atsiliko mokytojas
        gt,          // tigro bėgimo greitis
        gv,          // vilko bėgimo greitis
        gm;          // mokytojo bėgimo greitis
    cout << "Tigras atsiliko: ";    cin >> kt;
    cout << "Vilkas atsiliko: ";    cin >> kv;
    cout << "Mokytojas atsiliko: "; cin >> km;
    cout << "Tigro greitis: ";      cin >> gt;
    cout << "Vilko greitis: ";      cin >> gv;
    cout << "Mokytojo greitis: ";   cin >> gm;
    for (int i = 1; i <= 8; i++) {
        kt = kt - gt + 1; // tigras nubėgo gt km, tačiau mokinyš per tą patį laiką nubėgo 1 km
        kv = kv - gv + 1; // vilkas nubėgo gv km, tačiau mokinyš per tą patį laiką nubėgo 1 km
        km = km - gm + 1; // mokytojas nubėgo gm km, tačiau mokinyš per tą patį laiką nubėgo 1 km
    }
    if (kt <= 0) cout << "Tigras pavijo" << endl;
    else        cout << "Tigras nepavijo" << endl;
    if (kv <= 0) cout << "Vilkas pavijo" << endl;
    else        cout << "Vilkas nepavijo" << endl;
    if (km <= 0) cout << "Mokytojas pavijo" << endl;
    else        cout << "Mokytojas nepavijo" << endl;
    return 0;
}
```



### Programos patikrinimas

Parengus programą, reikia įsitikinti, kad joje nėra klaidų ir norimas rezultatas pasiekiamas esant visiems galimiems teisingiems pradinėms duomenų rinkiniams. Aritmetines klaidas rasti nesunku, nes, parinkus nedidelius skaičius, visuomet galima patiemis apskaičiuoti, kokie bus rezultatai.

- Patikrinkite, ar programa dirba teisingai tuo atveju, kai nagrinėjama, kas Petriuką pavijo, o kas – ne. Kontroliniams duomenų rinkiniams galima pasirušti lentelę, pavyzdžiui, tokią:

Pradiniai duomenys						Rezultatai
kt	kv	km	gt	gv	gm	
49	79	127	15	22	35	Visi pavijo
45	125	125	10	1	1	Tigras pavijo
12	12	100	5	4	7	Tigras pavijo Vilkas pavijo
						Tigras pavijo Mokytojas pavijo
						Vilkas pavijo Mokytojas pavijo
						Vilkas pavijo
						Mokytojas pavijo
						Niekas nepavijo

- Pabaikite pildyti lentelę. Patikrinkite, ar programa visais atvejais parodo teisingus rezultatus.



## Programos papildymas

- Pakeiskite programą taip, kad tuo atveju, kai niekas Petriuko nepaveja, ekrane būtų rodomas pranešimas. Programos pabaigoje parašykite tokį sakinį:

```
if ((kt > 0) && (kv > 0) && (km > 0)) cout << "Niekas nepavijo" << endl;
```

Žinoma, tuomet tikslinga jau esančiuose sąlyginiuose sakiniuose pašalinti `else` dalis.

- Besivejančiųjų užduotis – pagauti mokinį. Jeigu kuris nors iš besivejančiųjų paveja Petriuką, tai Petriukas pabunda ir lenktynės baigiasi. Taigi programa turėtų baigti darbą. Todėl 8 valandų skaičiavimo ciklą reikia pakeisti ciklu, kuriuo skaičiavimai būtų vykdomi tol, kol kas nors Petriuką pavys arba kol nesuskambės žadintuvas. Ciklo antraštė gali būti tokia:

```
while ((i <= 8) && (kt >= 0) && (kv >= 0) && (km >= 0))
```

Žinoma, kad tuomet kintamajam  $i$  pradinė reikšmė turi būti suteikta prieš ciklą, o cikle ji didinama.



## Užduotys

### 1. Gimtadienis

Dvynukai Saulius ir Ramunė nusprendė į gimtadienio šventę pakviesti visus savo draugus. Abu jie turi  $d$  bendrų draugų. Saulius dar nori pakviesti  $s$  savo draugų, Ramunė –  $r$  savo draugių. Gimtadieniui vaikai sutaupė  $g$  litų. Parenkite programą, kuri apskaičiuotų, kokią didžiausią pinigų sumą  $k$  iš savo sutaupyty pinigų vaikai gali skirti kiekvienam svečiui.

*Pasitikrinkite.* Kai  $d = 6$ ,  $s = 3$ ,  $r = 3$ ,  $g = 55$ , ekrane turi būti rodoma:  $k = 4.58$ .

### 2. Kelionė į mokyklą

Petras išėjo iš namų, kai laikrodis rodė  $v1$  valandų ir  $m1$  minučių. Į gimnaziją Petro kelionė trunka  $m2$  minučių. Parenkite programą, kuri ekrane parodytų pranešimą apie tai, ar Petras nepavėluos į pamoką, prasi-dedančią  $v$  valandą ir  $m$  minučių.

*Pasitikrinkite.* Kai  $v1 = 8$ ,  $m1 = 29$ ,  $m2 = 43$ ,  $v = 9$ ,  $m = 5$ , ekrane turi būti rodomas pranešimas:

Petras į pamoką pavėluos.

Kai  $v1 = 8$ ,  $m1 = 29$ ,  $m2 = 23$ ,  $v = 9$ ,  $m = 5$ , ekrane turi būti rodomas pranešimas:

Petras į pamoką nepavėluos.

### 3. Kelionė po Lietuvą

Vienuoliktokai nusprendė pasibaigus mokslo metams aplankyti gražiausias Lietuvos vietas. Kelionės maršrutą jie pradėjo rinktis iš anksto, kad galėtų numatyti kelionės išlaidas. Mokiniai pasirinko maršrutą Panevėžys–Plungė–Panevėžys (320 km). Į kelionę planuoja vykti  $k$  vienuoliktokų. Kelionės trukmė –  $d$  dienų. Kelionei kiekvienas vienuoliktokas gali skirti po  $t$  litų. Maistui per dieną vienam žmogui reikia  $v$  litų. Litras benzino kainuoja  $n$  litų. Degalų sąnaudos sudaro  $b$  litrų šimtui kilometrų. Parenkite programą, kuri ekrane parodytų pranešimą *Vienuoliktokai gali vykti į šią kelionę*, jei numatoma kelionei skirti pinigai padengia kelionės išlaidas, arba *Vienuoliktokai negali vykti į šią kelionę*, jei numatoma kelionei skirti pinigai nepadengia kelionės išlaidų.

*Pasitikrinkite.* Kai  $k = 20$ ,  $t = 100$ ,  $d = 2$ ,  $v = 10$ ,  $n = 4$ ,  $b = 10$ , ekrane turi būti rodomas pranešimas:

Vienuoliktokai gali vykti į šią kelionę.

Kai  $k = 20$ ,  $t = 20$ ,  $d = 2$ ,  $v = 10$ ,  $n = 4$ ,  $b = 10$ , ekrane turi būti rodomas pranešimas:

Vienuoliktokai negali vykti į šią kelionę.

### 4. Slidinėjimas

Slidinėjimo varžybų trasą sudaro  $n$  ratų. Vieno rato ilgis yra  $m$  metrų. Sportininkas pirmą ratą įveikė per  $t1$  sekundžių, antrąjį – per  $t2$  sekundžių ir t. t. Parenkite programą, kuri apskaičiuotų, kokių vidutiniu greičiu  $v$  čiuožė sportininkas ir kiek laiko  $t$  sugaišo, įveikdamas trasą.

*Pasitikrinkite.* Kai  $n = 4$ ,  $m = 500$ ,  $t_1 = 45$ ,  $t_2 = 42$ ,  $t_3 = 39$ ,  $t_4 = 37$ , ekrane turi būti rodomas pranešimas: Sportininko vidutinis greitis  $v = 12.3$  m/s, distancijoje sugaišo 163 sekundes.

## 5. Pietūs

Pirmąjį patiekalą pietums mama gamina  $t_1$  minučių, antrąjį –  $k$  minučių ilgiau negu pirmąjį, trečiąjį –  $k$  minučių ilgiau negu antrąjį ir t. t. Parenkite programą, kuri apskaičiuotų, kiek patiekalų  $n$  suspės pagaminti mama, kol vaikai sugrįš iš lauko, jei pietus virti pradėjo dabar, o vaikai iš lauko grįš po  $t$  minučių. Visi duomenys yra sveikojo tipo.

*Pasitikrinkite.* Kai  $t_1 = 17$ ,  $k = 3$ ,  $t = 65$ , ekrane turi būti rodomas pranešimas:

Mama suspės pagaminti 3 patiekalus.

## 6. Katino vakarienė

Katino šeima vakarienei nutarė pasigauti žuvelių. Meškerioja senis katinas. Į krepšį jis deda ne visas žuvelės, o tik tas, kurios didelės ir skanios. Meškeriojimo duomenys įvedami klaviatūra. Pirmas skaičius parodo, kiek iš viso žuvelių pagavo katinas. Toliau pateikiamas kiekvienos žuvelės svoris ir informacija apie skanumą (1 – skani, 0 – neskani). Žuvelės, kurios sveria mažiau kaip 1, laikomos mažomis. Parenkite programą, kuri apskaičiuotų, kiek žuvelių katinas parnešė į namus ir koks buvo jų svoris.

*Pasitikrinkite.* Tarkime, kad klaviatūra įvedami tokie duomenys:

```
6
1.5 1
2.1 1
0.5 0
3.5 1
0.5 1
4.12 0
```

Tuomet ekrane turi būti rodomas rezultatas: 8 <sup>4-11-22</sup> <sub>4-7-10</sub> *giron*

- Papildykite programą skaičiavimais, kiek didelių ir kiek skanių žuvelių pagavo katinas. Įvedus nurodytus patikrinimo duomenis, rezultatas turėtų būti toks: Didelių žuvelių – 4, skanių – 4.
- Pakeiskite programą taip, kad duomenų įvedimo pabaiga būtų nurodoma dviem nuliais (žuvelės svoris 0, skanumo požymis – 0).
- Pakeiskite programą taip, kad duomenų įvedimas būtų valdomas dialogu: ar dar yra žuvelių (atsakymai T (taip, yra) arba N (ne, nėra)).

## 7. Įdomūs mėnesiai

2010 metų spalio mėnuo buvo ypatingas – jis turėjo 5 penktadienius, 5 šeštadienius ir 5 sekmadienius. Taip atsitinka vieną kartą per 823 metus.

Parenkite programą, kuri nustatytų, kurie mėnesiai ir kuriais nurodyto intervalo metais (pvz., [2005; 2012]) turi tokią savybę. Įvertinkite, kad keliamieji metai turi viena diena daugiau. Tai atsitinka tais metais, kurių skaičius dalijasi iš 4 be liekanos. Tačiau paskutiniai amžiaus metai (dalijasi iš 100 be liekanos) yra keliamieji tik tada, jei be liekanos dalijasi iš 400. Jie nustatomi taip:

```
if (metai % 400 == 0 || (metai % 100 != 0 && metai % 4 == 0))
    keliamieji metai;
else
    nekeliamieji metai;
```

*Pasitikrinkite.* 2010 metais tokius penkių dienų rinkinius turėjo sausis ir spalio, o 2011 metais – tik liepa.

- Papildykite programą taip, kad ji apskaičiuotų, kas kiek metų toks dienų rinkinys pasikartoja sausio mėnesį.
- Papildykite programą taip, kad ji nustatytų, kas kiek metų toks dienų rinkinys pasikartoja esant nurodytam mėnesiui  $m$ .



## 1.2. Duomenų skaitymas iš failo ir rezultatų rašymas į failą

Atlikdami šį darbą:

- ✓ išmoksite skaityti duomenis iš tekstinio failo;
- ✓ prisiminsite ir pritaikysite sumos skaičiavimo algoritmą;
- ✓ išmoksite rezultatus įrašyti į tekstinį failą.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.1. Konstantos 2.2. Operatoriai 2.4. Duomenų skaitymas iš failo 2.5. Rezultatų (duomenų) rašymas į failą 2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	4.4. Sumos skaičiavimo algoritmas (žr. vadovėlio „Šiuolaikiškas žvilgsnis į programavimo pagrindus. C++. Pasirenkamasis informacinių technologijų kursas IX–X klasėms“ algoritmų žinyną)

### Užduotis

**Medelių sodinimas.** Sodininkų bendrijos nariai, norintys rudenį pasisodinti žemaūgių obelaičių, nusprendė, kad surašys, kiek kuriam reikia medelių, po to į medelyną nuvažiuos vienas žmogus ir parveš visus medelius. Reikia apskaičiuoti, kiek medelių iš viso užsakė sodininkų bendrijos nariai, ir gautą rezultatą įrašyti į failą.

Pirmoje pradinių duomenų failo eilutėje įrašytas sodininkų, kurie užsisakė medelių, skaičius. Tolesnėse failo eilutėse įrašyta po vieną sveikąjį skaičių – kiek kiekvienas sodininkas užsisakė medelių.

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Paiškinimai	Rezultatai	Paiškinimai
3	Sodininkų skaičius	19	Iš viso užsakyta medelių
7	Pirmo sodininko užsakymas		
6	Antro sodininko užsakymas		
6	Trečio sodininko užsakymas		

### Algoritmas

Užduotis sprendžiama taip:

1. Nurodoma pradinė sumos (užsakytų medelių skaičiaus) reikšmė. Ji lygi nuliui.
2. Iš pradinių duomenų failo perskaitoma, keli sodininkai užsisakė medelių.
3. Veiksmai kartojami tiek kartų, kiek sodininkų užsisakė medelius:
  - ✓ iš failo perskaitomas kiekvieno sodininko užsakytas medelių skaičius;
  - ✓ sodininko užsakytas medelių skaičius pridamas prie bendros medelių sumos.
4. Į rezultatų failą įrašomas visų sodininkų užsakytų medelių skaičius.



#### Pradinių duomenų failo kūrimas, kintamųjų aprašymas

- Sukurkite tekstinį failą *Duomenys2.txt* ir į jį įrašykite pateiktus pavyzdyje pradinius duomenis.
- Aprašykite eilučių tipo konstantas pradinių duomenų ir rezultatų failų vardams atmintyje laikyti, taip pat sveikojo tipo kintamuosius:

*n* – medelius užsakusių sodininkų skaičių,

*m* – kiekvieno sodininko užsakytų medelių skaičių,

*s* – visų sodininkų užsakytų medelių skaičių.

Konstantų aprašymas pradedamas žodžiu `const`. Po to nurodomas konstantos reikšmės tipas, vardas ir reikšmė:

```
const Tipas Vardas = reikšmė;
```

```
const char CDfv[] = "Duomenys2.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai2.txt"; // rezultatų failo vardas
//-----
int main()
{
    int n, m; // sodininkų skaičius ir kiekvieno sodininko užsakytų medelių skaičius
    int s = 0; // visų sodininkų užsakytų medelių skaičius
    return 0;
}
```

Atkreipkite dėmesį į tai, kaip nurodyta pradinė kintamojo `s` (sumos) reikšmė.

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tik informacinį pranešimą.



## Pirmojo skaičiaus (sodininkų skaičiaus `n`) skaitymas iš failo

- Papildykite programą sakiniu, kuris sukurtų įvesties srautą `fd`, susietą su failu `Duomenys2.txt`:

```
ifstream fd(CDfv);
```

Duomenų srautų valdymo priemonės yra antraštiniame faile `fstream`, kuris įterpiamas į programą sakiniu:

```
#include <fstream>
```

Patikrinkite, ar šis sakinytis yra programoje.

- Parašykite sakinį, kuriuo perskaitomas sodininkų skaičius:

```
fd >> n;
```

- Pasitikrinkite, ar iš failo teisingai perskaityta `n` reikšmė, t. y. parašykite sakinį, kuris ekrane ją parodytų:

```
cout << n << endl;
```

```
int main()
{
    int n, m; // sodininkų skaičius ir kiekvieno sodininko užsakytų medelių skaičius
    int s = 0; // visų sodininkų užsakytų medelių skaičius
    ifstream fd(CDfv);
    fd >> n;
    cout << n << endl;
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti pirmąjį iš failo perskaitytą skaičių:

3

Pradinis duomenys	Pasiskaitoma
3 10 12	Sodininkų skaičius, sodininkas užsakytų medelių skaičius, medelių skaičius per keturis tulpas
12	Pirmo sodininko užsakymas
10	Antra sodininko užsakymas
6	Trečio sodininko užsakymas
8	Ketvirtą sodininko užsakymas
11	Penkto sodininko užsakymas



### Veiksmų kartojimas

- Papildykite programą ciklo sakiniu, kuriuo būtų perskaitomas kiekvieno sodininko užsakytų medelių skaičius ir pridamas prie visos medelių sumos:

```
for (int i = 1; i <= n; i++) {  
    fd >> m; // perskaitomas sodininko užsakytų medelių skaičius  
    s = s + m; // užsakytų medelių suma papildoma nauju skaičiumi  
    cout << m << " " << s << endl; // ekrane parodomi tarpinių skaičiavimų rezultatai  
}
```

- Kai visi duomenys bus perskaityti, srautą užverkite:

```
fd.close();
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti visus iš failo perskaitytus pradinius duomenis ir tarpinius rezultatus:

```
3  
7 7  
6 13  
6 19
```



### Rezultatų rašymas į failą *Rezultatai2.txt*

- Papildykite programą sakiniu, kuris sukurtų išvesties srautą `fr`, susietą su failu *Rezultatai2.txt*:

```
ofstream fr(CRfv);
```

- Parašykite sakinį, kuriuo būtų įrašomas į failą visų sodininkų užsakytų medelių skaičius:

```
fr << s;
```

- Kai visi rezultatai bus surašyti, srautą užverkite:

```
fr.close();
```

- Pašalinkite iš programos sakinius, kurie buvo skirti patikrinti, ar programa dirba teisingai:

```
cout << n << endl;  
cout << m << " " << s << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tik informacinį pranešimą.
- Atverkite rezultatų failą *Rezultatai2.txt*. Jame turėtų būti skaičius 19 – visų sodininkų užsakytų medelių skaičius. Failą galima atverti tekstų rengykle, pavyzdžiui, *Notepad*. Aplinkoje *CodeBlocks* failui atverti naudojama komanda *File* → *Open*.



## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDfv[] = "Duomenys2.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai2.txt"; // rezultatų failo vardas
//-----
int main()
{
    int n, m; // sodininkų skaičius ir kiekvieno sodininko užsakytų medelių skaičius
    int s = 0; // visų sodininkų užsakytų medelių skaičius
    ifstream fd(CDfv);
    fd >> n;
    for (int i = 1; i <= n; i++) {
        fd >> m; // perskaitomas sodininko užsakytas medelių skaičius
        s = s + m; // užsakytų medelių suma papildoma nauju skaičiumi
    }
    fd.close();
    ofstream fr(CRfv);
    fr << s;
    fr.close();
    return 0;
}
```



### Programos patikrinimas

- Sukurkite pradinių duomenų rinkinį, kuriame duomenų būtų per mažai. Pavyzdžiui, nurodyta, kad medelių užsisakė 3 sodininkai, o tolesnėse failo eilutėse įrašyti tik dviejų sodininkų pageidaujami medelių kiekiai. Įvykdę programą, suprasite, kad skaičiuojant medelius prie pirmojo sodininko užsakytų medelių pridėdamas dvigubas antrojo sodininko užsakytų medelių kiekius. Taip atsitinka todėl, kad ciklas vykdomas tris kartus: pirmą kartą perskaitomas pirmojo sodininko užsakytas medelių kiekis, antrą kartą – antrojo sodininko, o vykdant ciklą trečią kartą faile duomenų jau nebėra ir kintamojo  $m$  reikšmė lieka nepakitusi (antrojo sodininko užsakytas medelių skaičius).
- Sukurkite pradinių duomenų rinkinį, kuriame duomenų būtų per daug. Pavyzdžiui, nurodyta, kad medelių užsisakė 3 sodininkai, o tolesnėse failo eilutėse įrašyti penkių sodininkų pageidaujami medelių kiekiai. Įvykdę programą ir peržiūrėję rezultatus, pastebėsite, kad buvo perskaityti ir sumuojami tik trijų sodininkų užsakyti medeliai.

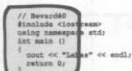


### Programos papildymas

- Papildykite sukurta programą, kad ji skaičiuotų, kiek sodininkų užsakė ne mažiau kaip  $x$  medelių. Reikšmę  $x$  įrašykite į pirmą pradinių duomenų failo eilutę, nuo sodininkų skaičiaus ją atskirdami tarpu. Rezultatų faile turi būti dar viena eilutė, kurioje būtų nurodomas sodininkų, užsisakusių ne mažiau kaip  $x$  medelių, skaičius.
- Papildykite sukurta programą, kad ji skaičiuotų, kelis kartus reikės važiuoti į medelyną, jei į mašinos priekabą telpa  $y$  medelių. Reikšmę  $y$  nurodykite pirmoje pradinių duomenų failo eilutėje, nuo ankstesnių reikšmių ją atskirdami tarpu. Gautą rezultatą įrašykite į rezultatų failą.

Pradiniai duomenys	Paaiškinimai
5 10 12	Sodininkų skaičius, mažiausias užsakomų medelių skaičius, mašinos priekabos talpa
12	Pirmo sodininko užsakymas
10	Antro sodininko užsakymas
6	Trečio sodininko užsakymas
8	Ketvirto sodininko užsakymas
11	Penkto sodininko užsakymas

Rezultatai	Paaiškinimai
47	Iš viso užsakyta medelių
3	Trys sodininkai užsakė ne mažiau kaip numatyta
4	Iš viso mašina važiuos į medelyną 4 kartus



## Užduotys

### 1. Uždarbis

Per pirmą darbo mėnesį žmogus uždirbo  $p1$  litų, per antrąjį –  $p2$  ir t. t. Parenkite programą, kuri apskaičiuotų, kiek pinigų iš viso uždirbo žmogus per  $n$  mėnesių.

Pirmoje pradinių duomenų failo eilutėje įrašytas mėnesių skaičius  $n$ . Tolesnėse  $n$  eilučių įrašyta po vieną realųjį skaičių – žmogaus mėnesio atlyginimas.

Rezultatų faile turi būti įrašyta žmogaus uždirbta per  $n$  mėnesių pinigų suma.

Pradiniai duomenys	Rezultatas
3	1901.00
700.50	
600.25	
600.25	

### 2. Prekyba ledais

Pirmąją prekybos dieną verslininkas pardavė  $k$  porcijų ledų. Prekyba sekėsi gerai – kiekvieną kitą dieną jis parduodavo po  $m$  porcijų ledų daugiau negu prieš tai buvusią. Parenkite programą, kuri apskaičiuotų, kiek porcijų ledų  $vk$  verslininkas pardavė per  $n$  dienų.

Pradinių duomenų faile įrašyti trys atskirti tarpais skaičiai:  $k$  (kiek pirmąją prekybos dieną parduota porcijų),  $m$  (keliomis porcijomis kiekvieną kitą dieną buvo parduodama daugiau) ir  $n$  (kelias dienas buvo prekiauta ledais).

Rezultatų faile turi būti įrašyta, kiek porcijų ledų verslininkas pardavė per  $n$  dienų.

Pradiniai duomenys	Rezultatas
170 30 3	600

### 3. Kačių dresuotojas

Pramušgalvis penkiametis Andrius nusprendė tapti kačių dresuotoju. Andrius mokosi dresuodamas savo katiną Ziną. Pirmą dresavimo dieną Zinas buvo dresuojamas  $t1$  minučių, antrąją –  $t2$  ir t. t. Parenkite programą, kuri apskaičiuotų, kiek minučių buvo dresuojamas Zinas, jei Andrius jį dresavo  $n$  dienų, ir kiek vidutiniškai minučių Zinas buvo dresuojamas per dieną.

Pirmoje pradinių duomenų failo eilutėje įrašytas dienų skaičius  $n$ . Tolesnėse  $n$  eilučių įrašyta po vieną sveikąjį skaičių – kiek minučių buvo dresuojamas Zinas kiekvieną dieną.

Rezultatų faile turi būti įrašyta, kiek minučių buvo dresuojamas Zinas per  $n$  dienų ir kiek vidutiniškai minučių Zinas buvo dresuojamas per dieną.

Pradiniai duomenys	Rezultatai
4	60
15	15
12	
13	
20	

#### 4. Biatlonininko rezultatas

Biatlono varžybose yra  $n$  etapų, kuriuose reikia po  $x$  kartų pataikyti į taikinį. Pirmą etapą varžybų dalyvis įveikė per  $t_1$  minučių ir pataikė  $k_1$  kartų, antrąjį – per  $t_2$  minučių ir pataikė  $k_2$  kartų ir t. t. Už kiekvieną nepataikytą kartą sportininkui pridedama  $b$  baudos minučių. Parenkite programą, kuri apskaičiuotų, per kiek minučių varžybų dalyvis įveikė trasą. Spręsdami uždavinį, turėkite omenyje, kad pradiniai duomenys ir rezultatai yra sveikieji skaičiai.

Pirmoje pradinių duomenų failo eilutėje įrašytas dienų skaičius  $n$ , šuvių skaičius  $x$  ir baudos minučių skaičius  $b$ . Skaičiai vienas nuo kito atskirti tarpais. Tolesnėse  $n$  eilučių įrašyta po du tarpais atskirtus skaičius: laikas, per kurį dalyvis įveikė varžybų etapą, ir kiek kartų jis pataikė į taikinį.

Rezultatų faile turi būti įrašyta, per kiek minučių varžybų dalyvis įveikė trasą.

Pradiniai duomenys	Rezultatas
4 5 3	46
5 3	
3 2	
3 1	
5 4	

#### 5. Kabelis

Gaminant daugiagyslių kabelį, kiekvienas jo laidininkas (varinė viela) dengiamas izoliacijos apvaskalu. Gautos gyslos grupuojamos ir dengiamos kitu izoliacijos apvaskalu. Taip daroma tol, kol visos gyslos (sudarytos iš mažesnių) apvelkamos bendru išoriniu apvaskalu. Kuo daugiau izoliacijos apvaskalų, tuo kabelis geriau apsaugotas nuo išorinių aplinkos poveikių. Tačiau viena atskira gysla negali turėti kelių izoliacijos sluoksnių. Parenkite programą, kuri apskaičiuotų didžiausią galimą apvaskalų kabelyje skaičių.



Pirmoje ir vienintelėje pradinių duomenų failo eilutėje nurodytas laidininkų skaičius  $n$  ( $1 \leq n \leq 109$ ).

Vienintelėje rezultato failo eilutėje turi būti įrašytas didžiausias galimas izoliacijos apvaskalų skaičius.

Pradiniai duomenys	Rezultatas	Paiškinimas
5	9	<p>Paveiksluose parodyti du būdai, kaip galima geriausiai izoluoti kabelius. Pateiktas skersinis kabelio pjūvis: juodi skrituliai – tai laidininkai, o apskritimai – izoliacijos apvaskalai (gyslos).</p>

(XIX olimpiada, 2007)

#### 6. Superfibonačio skaičiai

Tikriausiai esate girdėję apie Fibonačio skaičių seką: 1 1 2 3 5 8 ...

Ji apibrėžiama tokiu būdu:  $F_1 = 1, F_2 = 1, F_n = F_{n-1} + F_{n-2}$ , t. y. kiekvienas narys, pradedant trečiuoju, lygus prieš jį esančių dviejų narių sumai. Panašiai apibrėžkime *superfibonačio* skaičius:  $F_1 = 1, F_2 = 1, F_3 = 1, F_n = F_{n-1} + F_{n-2} + F_{n-3}$ , t. y. pirmieji trys sekos nariai yra vienetai, o kiekvienas tolesnis narys gaunamas sudėjus tris paskutinius sekos narius. Parenkite programą, kuri rastų  $n$ -ąjį *superfibonačio* sekos narį.

Pirmoje ir vienintelėje pradinių duomenų failo eilutėje įrašytas vienas sveikasis skaičius  $n$  ( $1 \leq n < 20$ ) – ieškomo *superfibonačio* sekos nario numeris.

Rezultatų faile turi būti vienas skaičius –  $n$ -asis *superfibonačio* sekos narys.

Pradiniai duomenys	Rezultatas	Paiškinimas
6	9	1, 1, 1, 3, 5, 9

(XIX olimpiada, 2007)



## 1.3. Ciklas cikle

Atlikdami šį darbą:

- ✓ išmoksite naudoti ciklą cikle;
- ✓ įvirtinsite skaitymo iš failo ir rašymo į failą įgūdžius;
- ✓ pritaikysite sumos skaičiavimo algoritmą.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.1. Konstantos 2.2. Operatoriai 2.4. Duomenų skaitymas iš failo 2.5. Rezultatų (duomenų) rašymas į failą 2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	4.4. Sumos skaičiavimo algoritmas (žr. vadovėlio „Šiuolaikiškas žvilgsnis į programavimo pagrindus. C++“. Pasirenkamasis informacinių technologijų kursas IX–X klasėms“ algoritmų žinyną)

### Užduotis

**Programuotojų maratonas.** Gimnazijos programuotojai, rengdamiesi komandinėms varžyboms, surado svetainę <http://projecteuler.net>, kurioje yra daug programavimo uždavinių. Jie nusprendė surengti savaitės trukmės programavimo maratoną. Visi programuotojai sprendė skirtingus uždavinius. Reikia apskaičiuoti, kiek uždavinių išsprendė kiekvienas programuotojas per savaitę, kiek uždavinių iš viso programuotojai išsprendė per savaitę ir kiek uždavinių vidutiniškai išspręsta per dieną.

Pirmoje pradinių duomenų failo eilutėje įrašytas varžybose dalyvavusių programuotojų skaičius  $n$ . Tolesnėse eilutėse surašyti skaičiai tokia tvarka: pirmasis skaičius  $d$  rodo, kiek dienų programuotojas sprendė uždavinius, o likusieji  $d$  skaičių – kiek uždavinių išsprendė kiekvieną dieną.

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Paiškinimai
5	Programuotojų skaičius
5 3 2 3 1 2	Pirmas programuotojas dirbo 5 dienas; jo parašytų kiekvieną dieną programų skaičiai
3 6 2 4	Antras programuotojas dirbo 3 dienas; jo parašytų kiekvieną dieną programų skaičiai
4 2 2 1 2	Trečias programuotojas dirbo 4 dienas; jo parašytų kiekvieną dieną programų skaičiai
3 3 3 3	Ketvirtas programuotojas dirbo 3 dienas; jo parašytų kiekvieną dieną programų skaičiai
2 3 4	Penktas programuotojas dirbo 2 dienas; jo parašytų kiekvieną dieną programų skaičiai
Rezultatai	Paiškinimai
11	Pirmas programuotojas parašė 11 programų
12	Antras programuotojas parašė 12 programų
7	Trečias programuotojas parašė 7 programas
9	Ketvirtas programuotojas parašė 9 programas
7	Penktas programuotojas parašė 7 programas
46	Visi programuotojai parašė 46 programas
2.7	Vidutiniškai per dieną buvo parašyta 2,7 programos

### Algoritmas

Užduotis sprendžiama taip:

1. Nurodomos pradinės išspręstų uždavinių skaičiaus ir dalyvavimo varžybose dienų skaičiaus reikšmės. Jos lygios nuliui.
2. Iš pradinių duomenų failo perskaitoma, keli programuotojai dalyvavo maratone.
3. Veiksmai kartojami tiek kartų, kiek programuotojų dalyvavo maratone:
  - ✓ nurodoma vieno programuotojo išspręstų uždavinių kiekio pradinė reikšmė (ji lygi nuliui);
  - ✓ iš failo perskaitoma, kiek dienų kiekvienas programuotojas sprendė uždavinius;
  - ✓ veiksmai kartojami tiek kartų, kiek dienų programuotojas dirbo;

- iš failo perskaitomas vieną dieną išspręstų uždavinių skaičius;
  - uždavinių skaičius pridedamas prie visos programuotojo išspręstų uždavinių sumos.
- ✓ į rezultatų failą įrašomas programuotojo išspręstų uždavinių skaičius;
  - ✓ prie visų dienų pridedamas programuotojo dirbtų dienų skaičius;
  - ✓ prie visų išspręstų uždavinių pridedamas programuotojo išspręstų uždavinių skaičius.
4. Skaičiuojama, kiek vidutiniškai uždavinių išspręsta per dieną.
  5. Į rezultatų failą įrašomas iš viso išspręstų uždavinių skaičius ir vidutiniškai per dieną išspręstų uždavinių skaičius.

## 1 Pradinių duomenų failo kūrimas, konstantų ir kintamųjų aprašymas

- Sukurkite tekstinį failą *Duomenys3.txt* ir į jį įrašykite pavyzdyje pateiktus pradinius duomenis.
- Aprašykite eilučių tipo konstantas pradinių duomenų ir rezultatų failų vardams atmintyje laikyti:

```
const char CDfv[] = "Duomenys3.txt";
const char CRfv[] = "Rezultatai3.txt";
```

- Aprašykite kintamuosius:

n – kiek programuotojų dalyvavo turnyre,  
 us – kiek iš viso išsprendė uždavinių,  
 ds – kelias dienas sprendė,  
 up – kiek uždavinių išsprendė vienas programuotojas,  
 d – kiek dienų vienas programuotojas dirbo,  
 p – kiek uždavinių per dieną programuotojas išsprendė,  
 uv – kiek vidutiniškai uždavinių išspręsta per dieną.

```
const char CDfv[] = "Duomenys3.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai3.txt"; // rezultatų failo vardas
//-----
int main()
{
    int n; // programuotojų skaičius
    int us = 0; // programuotojų parašytų programų skaičius
    int ds = 0; // visų dienų, kuriomis buvo rašomos programos, skaičius
    int p, d, up; // programuotojo duomenys: per dieną parašytų programų skaičius,
                // dalyvavimo maratone dienų skaičius, iš viso parašytų programų skaičius
    double uv; // vidutiniškai per dieną parašytų programų skaičius
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tik informacinį pranešimą.

## 2 Pirmojo duomens (programuotojų skaičiaus n) skaitymas iš failo

- Papildykite programą sakiniu, kuriuo įvesties srautas *fd* būtų susiejamas su failu *Duomenys3.txt*:

```
ifstream fd(CDfv);
```

- Papildykite programą sakiniu, kuriuo išvesties srautas *fr* būtų susiejamas su failu *Rezultatai3.txt*:

```
ofstream fr(CRfv);
```

- Parašykite sakinį, kuriuo būtų perskaitomas programuotojų, dalyvavusių maratone, skaičius:

```
fd >> n;
```

- Parašykite sakinį, kuriuo ekrane būtų parodoma perskaityta reikšmė:

```
cout << n << endl;
```

```
int main()
{
    int n;           // programuotojų skaičius
    int us = 0;     // programuotojų parašytų programų skaičius
    int ds = 0;     // visų dienų, kuriomis buvo rašomos programos, skaičius
    int p, d, up;   // programuotojo duomenys: per dieną parašytų programų skaičius,
                  // dalyvavimo maratone dienų skaičius, iš viso parašytų programų skaičius
    double uv;     // vidutiniškai per dieną parašytų programų skaičius
    ifstream fd(CDfv);
    ofstream fr(CRFv);
    fd >> n;
    cout << n << endl;
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti programuotojų skaičių:

```
5
```

3

### Vieno programuotojo išspręstų uždavinių skaičiavimas

- Papildykite programą sakiniu, kuris iš failo perskaitytų, kiek dienų  $d$  programuotojas sprendė uždavinius:

```
fd >> d;
```

- Papildykite programą sakiniu, kuris nurodytų programuotojo išspręstų uždavinių skaičiaus pradinę reikšmę:

```
up = 0;
```

- Parašykite ciklo sakinį, kuriuo būtų perskaitomas programuotojo kiekvieną dieną išspręstų uždavinių skaičius ir skaičiuojama programuotojo išspręstų uždavinių suma:

```
for (int j = 1; j <= d; j++) {
    fd >> p;
    up = up + p;
}
```

- Parašykite sakinį, kuris ekrane parodytų apskaičiuotą programuotojo išspręstų uždavinių skaičiaus reikšmę:

```
cout << up << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti programuotojų, dalyvavusių turnyre, ir pirmojo programuotojo parašytų programų skaičių reikšmes:

```
5
11
```



4

### Visų programuotojų išspręstų uždavinių skaičiavimas

- Norėdami apskaičiuoti, kiek iš viso uždavinių išsprendė programuotojai ir kiek dienų jiems prirėkė, papildykite programą dar vienu ciklo sakiniu `for`, kuris apgaubtų ankstesnį ciklo sakinį:

```
for (int i = 1; i <= n; i++) {
    fd >> d;
    up = 0;
    for (int j = 1; j <= d; j++) {
        fd >> p;
        up = up + p;
    }
    fr << up << endl;

    ds = ds + d;
    us = us + up;
}
```

Išoriniame ciklo sakinyje apskaičiuotas vieno programuotojo išspręstų uždavinių skaičius įrašomas į rezultatų failą:

```
fr << up << endl;
```

- Naudodamiesi sumos skaičiavimo algoritmu, apskaičiuokite, kiek iš viso dienų buvo dirbta ir kiek uždavinių išspręsta:

```
ds = ds + d;
us = us + up;
```

- Kai visi duomenys bus perskaityti, srautą užverkite:

```
fd.close();
```

- Įrašykite į rezultatų failą programuotojų iš viso parašytų programų skaičių:

```
fr << us << endl;
```

- Kai visi rezultatai bus surašyti, srautą užverkite:

```
fr.close();
```

- Įrašykite ir įvykdykite programą. Faile matysite, kiek kiekvienas programuotojas parašė programų ir kiek jų iš viso buvo parašyta.



5

### Vidutiniškai per dieną išspręstų uždavinių skaičiaus radimas ir rezultatų rašymas į failą

- Parašykite sakinį, kuriuo būtų apskaičiuojama, kiek vidutiniškai per dieną išspręsta uždavinių:

```
uv = (double) us / ds;
```

- Parašykite sakinį, kuris į rezultatų failą įrašytų vidutiniškai per dieną išspręstų uždavinių skaičių:

```
fr << fixed << setprecision(1) << uv << endl;
```

- Pašalinkite iš programos sakinius, skirtus patikrinti, ar programa dirba teisingai:

```
cout << n << endl;  
cout << up << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tik aplinkos *CodeBlocks* informacinę pranešimą.
- Atverkite rezultatų failą *Rezultatai3.txt*. Jame turėtumėte matyti tokius pat rezultatus, kokie pateikti pavyzdyje.

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDFv[] = "Duomenys3.txt"; // duomenų failo vardas  
const char CRfv[] = "Rezultatai3.txt"; // rezultatų failo vardas  
//-----  
int main()  
{  
    int n; // programuotojų skaičius  
    int us = 0; // programuotojų parašytų programų skaičius  
    int ds = 0; // visų dienų, kuriomis buvo rašomos programos, skaičius  
    int p, d, up; // programuotojo duomenys: per dieną parašytų programų skaičius,  
                // dalyvavimo maratone dienų skaičius, iš viso parašytų programų skaičius  
    double uv; // vidutiniškai per dieną parašytų programų skaičius  
  
    ifstream fd(CDFv);  
    ofstream fr(CRfv);  
    fd >> n;  
    for (int i = 1; i <= n; i++) {  
        fd >> d;  
        up = 0;  
        for (int j = 1; j <= d; j++) {  
            fd >> p;  
            up = up + p;  
        }  
        fr << up << endl;  
        ds = ds + d;  
        us = us + up;  
    }  
    fd.close();  
    fr << us << endl;  
    uv = (double) us / ds;  
    fr << fixed << setprecision(1) << uv << endl;  
    fr.close();  
    return 0;  
}
```



## Programos patikrinimas

- Sukurkite pradinių duomenų rinkinį, kuriame duomenų būtų per mažai:
  - ✓ nurodyta, kad maratone dalyvavo 5 programuotojai, o tolesnėse eilutėse yra tik trijų programuotojų duomenys;
  - ✓ nurodyta, kad programuotojas uždavinius sprendė 5 dienas, o pateikti tik trijų dienų sprendimo rezultatai.



- Sukurkite pradinių duomenų rinkinį, kuriame duomenų būtų per daug:
  - ✓ nurodyta, kad programavimo maratone dalyvavo 5 programuotojai, o tolesnėse eilutėse yra septynių programuotojų duomenys;
  - ✓ nurodyta, kad programuotojas uždavinius sprendė 5 dienas, o pateikti septynių dienų sprendimo rezultatai.



### Programos papildymas

- Papildykite sukurtą programą, kad ji apskaičiuotų, kiek vidutiniškai uždavinių per dieną išsprendžia kiekvienas mokinys. Gautą rezultatą įrašykite į rezultatų failą šalia mokinio išspręstų uždavinių skaičiaus.

Pradiniai duomenys	Rezultatai
5	11 2.2
5 3 2 3 1 2	12 4.0
3 6 2 4	7 1.8
4 2 2 1 2	9 3.0
3 3 3 3	7 3.5
2 3 4	46
	2.7



### Užduotys

#### 1. Aritmetikos užduotys

Pradinukų mokytoja nutarė susikurti užduotis, kurios padėtų patikrinti, kaip vaikai moka dauginti, dalyti, sudėti ir atimti skaičius. Kad būtų greičiau, ji paprašė jaunųjų programuotojų pagalbos. Mokytoja nurodė dviejų sveikųjų skaičių kitimo ribas  $[x1; x2]$  ir pageidavo, kad kiekvienas pradinukas gautų po 4 skirtingas užduotis (sudėties, atimties, daugybos, dalybos be liekanos).

Parenkite programą, kuri iš pradinių duomenų failo perskaitytų 2 sveikuosius skaičius  $x1$  ir  $x2$ , o į rezultatų failą įrašytų visas galimas skirtingas užduotis. Vieną užduočių rinkinį nuo kito atskirkite žvaigždutėmis.

Pradiniai duomenys	Rezultatai
3 10	6 + 3 = 9
	6 - 3 = 3
	6 * 3 = 18
	6 / 3 = 2
	*****
	8 + 4 = 12
	8 - 4 = 4
	8 * 4 = 32
	8 / 4 = 2
	*****
	9 + 3 = 12
	9 - 3 = 6
	9 * 3 = 27
	9 / 3 = 3
	*****
	10 + 5 = 15
	10 - 5 = 5
	10 * 5 = 50
	10 / 5 = 2
	*****

## 2. Pirkiniai

Pirkėjas aplankė  $n$  parduotuvių ir kiekvienoje jų įsigijo po  $m$  prekių. Parenkite programą, kuri apskaičiuotų už kokią pinigų sumą pirkėjas įsigijo prekių kiekvienoje parduotuvėje ir kiek pinigų išleido iš viso.

Pirmoje pradinių duomenų failo eilutėje įrašytas parduotuvių skaičius  $n$  ir prekių skaičius  $m$ . Tolesnėse eilutėse įrašyta po  $m$  realiųjų skaičių – kiekvienos prekės kaina. Vienos parduotuvės prekių kainos surašyto vienoje eilutėje ir viena nuo kitos atskirtos tarpais.

Rezultatų failo pirmose  $n$  eilučių turi būti du tarpais atskirti skaičiai: parduotuvės numeris ir pinigų suma pateikta dviejų ženklų po kablelio tikslumu. Paskutinėje rezultatų failo eilutėje turi būti įrašyta visa pirkėjų išleista pinigų suma dviejų ženklų po kablelio tikslumu.

Pradiniai duomenys	Rezultatai
3 5	1 9.62
1.27 2.92 3.45 1.09 0.89	2 8.89
1.08 2.25 3.75 1.12 0.69	3 8.90
0.98 2.48 3.62 1.10 0.72	27.41

## 3. Temperatūros

Yra žinomi vienos sausio mėnesio savaitės oro temperatūros rodmenys vidurdienį įvairiuose Lietuvos miestuose. Parenkite programą, kuri apskaičiuotų vidutinę savaitės temperatūrą kiekviename mieste.

Pirmoje pradinių duomenų failo eilutėje yra užrašytas Lietuvos miestų, kuriuose buvo fiksuojama temperatūra, skaičius  $n$  ( $1 \leq n \leq 50$ ). Tolesnėse  $n$  eilučių yra po septynis skaičius – kiekvienos dienos oro temperatūra mieste. Rezultatus surašykite į failą po du skaičius kiekvienoje eilutėje: nurodykite miesto eilės numerį ir vidutinę savaitės temperatūrą tame mieste trijų ženklų po kablelio tikslumu.

Pradiniai duomenys	Rezultatai
3	1 -4.429
-5 -7 -5 0 1 -6 -9	2 -3.000
-2 -2 0 1 -4 -7 -7	3 -3.571
-8 -5 -4 -1 0 -2 -5	

## 4. Gucikai, mucikai ir fucikai

Toli visatoje esančioje planetoje egzistuoja gyvybė. Ten gyvena trijų rūšių būtybės: *gucikai*, *mucikai* ir *fucikai*. Skirtingų rūšių būtybės gali turėti nevienodą skaičių kojų, rankų ir akių, o vienos rūšies būtybės turi po vienodą skaičių. Žinoma, kad bet kuri būtybė turi bent vieną ranką, bent vieną koją ir bent vieną akį. Pavyzdžiui, *gucikai* gali turėti po dvi kojas, po dvi rankas ir po tris akis, *mucikai* – po keturias kojas, po vieną ranką ir po vieną akį, *fucikai* – po vieną koją, po penkias rankas ir po tris akis. Žinoma, po kiek kojų, po kiek rankų ir po kiek akių turi *fucikai*, *mucikai* ir *gucikai*. Taip pat žinoma, kiek kojų, rankų ir akių turintis visos planetoje gyvenančios būtybės paėmus kartu. Parenkite programą, kuri apskaičiuotų, kiek planetoje gyvena *gucikų*, *mucikų* ir *fucikų*.

Pirmoje pradinių duomenų failo eilutėje įrašyti trys skaičiai: kiek iš viso planetoje gyvenančios būtybės turi rankų, kojų ir akių. Antroje eilutėje nurodyti vieno *guciko*, trečiojoje – vieno *muciko*, ketvirtojoje – vieno *fuciko* rankų, kojų ir akių skaičiai. Planetoje gyvena ne daugiau kaip 500 kiekvienos rūšies būtybių ir kiekviena būtybė gali turėti ne daugiau kaip 20 rankų, ne daugiau kaip 20 kojų ir ne daugiau kaip 20 akių. Rezultatų faile turi būti įrašyti planetoje gyvenančių *gucikų*, *mucikų* ir *fucikų* skaičiai.

Pradiniai duomenys	Rezultatai
39 19 20	3 8 1
2 2 3	
4 1 1	
1 5 3	

(XVI olimpiada, 200

## 1.4. Funkcija, gražinanti apskaičiuotą reikšmę per funkcijos vardą

Atlikdami šį darbą:

- ✓ susipažinsite su pagrindine programavimo priemone – funkcija;
- ✓ parašysite funkcijas, kurios gražina apskaičiuotas reikšmes per funkcijos vardą;
- ✓ pritaikysite sumos skaičiavimo algoritmą;
- ✓ įtvirtinsite duomenų skaitymo iš failo ir rezultatų rašymo į failą įgūdžius.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.1. Konstantos 2.2. Operatoriai 2.3. Duomenų skaitymas iš failo 2.4. Rezultatų (duomenų) rašymas į failą 2.6. Funkcijos 2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	4.4. Sumos skaičiavimo algoritmas (žr. vadovėlio „Šiuolaikiškas žvilgsnis į programavimo pagrindus. C++“. Pasirenkamasis informacinių technologijų kursas IX–X klasėms“ algoritmų žinyną)

### Užduotis

Mokyklos grindų dangos kaina. Mokykloje keičiama kabinetų grindų danga. Parenkite programą, kuri apskaičiuotų, kokia pinigų suma turi būti skirta naujai grindų dangai. Žinoma, kad kiekvienam kabinetui dangos reikia pirkti 3 proc. daugiau galimiems nuostoliams padengti. Apskaičiuotą pinigų sumą įrašykite į rezultatų failą.

Pirmoje pradinių duomenų failo eilutėje nurodytas kabinetų skaičius. Tolesnėse eilutėse įrašyta po tris realiūs skaičius, atskirtus tarpais: kabineto ilgis, plotis ir vieno kvadratinio metro dangos kaina.

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Paiškinimai	Rezultatas	Paiškinimai
3	Kabinetų skaičius	3034.54	Reikalinga pinigų suma
4 5 54.60	Kabineto ilgis, plotis, dangos m <sup>2</sup> kaina		
3.5 4.5 35.55	Kabineto ilgis, plotis, dangos m <sup>2</sup> kaina		
9.2 3.3 42.63	Kabineto ilgis, plotis, dangos m <sup>2</sup> kaina		

### Algoritmas

Užduotis sprendžiama taip:

1. Nurodoma pinigų sumos, reikalingos visų kabinetų grindų dangai pakeisti, pradinė reikšmė (ji lygi nuliui).
2. Iš duomenų failo nuskaitomas kabinetų skaičius.
3. Veiksmai kartojami tiek kartų, kiek yra kabinetų:
  - ✓ iš duomenų failo perskaitomas kabineto ilgis, plotis ir vieno kvadratinio metro dangos kaina;
  - ✓ skaičiuojamas kabineto plotas;
  - ✓ skaičiuojama, kiek kainuos kabineto grindų danga;
  - ✓ prie visos pinigų sumos pridedama apskaičiuota kabineto grindų dangos kaina.
4. Į rezultatų failą įrašoma apskaičiuota pinigų suma, reikalinga visų kabinetų grindų dangai pakeisti.

### Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Plotas ()	Stačiakampio ploto skaičiavimas
DangosKaina ()	Vieno kabineto grindų dangos kainos skaičiavimas



## Funkcijos kabineto plotui skaičiuoti rašymas

- Prieš pagrindinę funkciją `main()` parašykite funkcijos, skaičiuojančios stačiakampio plotą, prototipą:

```
double Plotas(double a, double b);
```

čia `a` yra stačiakampio ilgis, `b` – plotis.

- Po pagrindine funkcija `main()` parašykite funkciją `Plotas()`:

```
//-----  
// Apskaičiuoja stačiakampio, kurio ilgis a ir plotis b, plotą  
double Plotas(double a, double b)  
{  
    double p;  
    p = a * b;  
    return p;  
}
```

arba

```
//-----  
// Apskaičiuoja stačiakampio, kurio ilgis a ir plotis b, plotą  
double Plotas(double a, double b)  
{  
    return a * b;  
}
```

Šios funkcijos antraštėje yra du realiojo tipo kintamieji – `a` ir `b`. Jie atitinka stačiakampio ilgį ir plotį. Duomenų tipas `double`, parašytas prieš funkcijos vardą, parodo, kad apskaičiuotas plotas taip pat bus realusis skaičius. Funkcija gali būti užrašoma dviem būdais:

- 1) naudojant papildomą kintamąjį `p` ir per funkcijos vardą grąžinant apskaičiuotą kintamojo `p` reikšmę;
- 2) nenaudojant papildomo kintamojo, tik per funkcijos vardą grąžinant stačiakampio ilgio ir pločio sandaugą.

Pasirinkite tą būdą, kuris jums suprantamesnis.

- Patikrinkite, ar teisingai dirba sukurta funkcija: pagrindinėje funkcijoje parašykite, pavyzdžiui, tokį sakinį:

```
cout << Plotas(15, 10) << endl;
```

```
double Plotas(double a, double b);  
//-----  
int main()  
{  
    cout << Plotas(15, 10) << endl;  
    return 0;  
}  
//-----  
// Apskaičiuoja stačiakampio, kurio ilgis a ir plotis b, plotą  
double Plotas(double a, double b)  
{  
    return a * b;  
}
```

- Įrašykite ir įvykdysite programą. Ekrane turėtumėte matyti skaičių 150. Tai plotas stačiakampio, kurio ilgis lygus 15, o plotis – 10.

Parašytą funkciją panaudosime stačiakampio formos kabineto grindų plotui skaičiuoti, kai yra žinomas kabineto ilgis ir plotis.



## Funkcijos, skaičiuojančios kabineto grindų dangos kainą, rašymas

- Prieš pagrindinę funkciją `main()` parašykite funkcijos, skaičiuojančios kabineto grindų dangos kainą, prototipą:

```
double DangosKaina(double p, double kaina);
```

čia `p` yra kabineto grindų plotas, `kaina` – vieno kvadratinio metro grindų dangos kaina.

- Po pagrindine funkcija `main()` parašykite funkciją `DangosKaina()`, skaičiuojančią pinigų sumą, reikalingą kabineto grindų dangai pirkti:

```
//-----  
// Apskaičiuoja kabineto grindų dangos kainą  
// p – kabineto plotas, kaina – dangos ploto vieneto kaina  
double DangosKaina(double p, double kaina)  
{  
    double dk;  
    dk = 1.03 * p * kaina;  
    return dk;  
}
```

arba

```
//-----  
// Apskaičiuoja kabineto grindų dangos kainą  
// p – kabineto plotas, kaina – dangos ploto vieneto kaina  
double DangosKaina(double p, double kaina)  
{  
    return 1.03 * p * kaina;  
}
```

Šios funkcijos antraštėje yra du realiojo tipo kintamieji – `p` ir `kaina`. Jie atitinka kabineto plotą ir vieno kvadratinio metro dangos kainą. Duomenų tipas `double`, parašytas prieš funkcijos vardą, parodo, kad apskaičiuota dangos kaina taip pat bus realusis skaičius. Funkcija gali būti užrašoma dviem būdais:

- 1) naudojant papildomą kintamąjį `dk` ir per funkcijos vardą grąžinant apskaičiuotą kintamojo `dk` reikšmę;
- 2) nenaudojant papildomo kintamojo, tik per funkcijos vardą grąžinant apskaičiuotą dangos kainą.

Pasirinkite tą būdą, kuris jums paprastesnis ir suprantamesnis.

- Patikrinkite, ar teisingai dirba funkcija: pagrindinėje funkcijoje parašykite, pavyzdžiui, tokį sakinį:

```
cout << DangosKaina(12.5, 10) << endl;
```

```
double Plotas(double a, double b);  
double DangosKaina(double p, double kaina);  
//-----  
int main()  
{  
    cout << DangosKaina(12.5, 10) << endl;  
    return 0;  
}  
//-----  
// Apskaičiuoja stačiakampio, kurio ilgis a ir plotis b, plotą  
double Plotas(double a, double b)  
{  
    return a * b;  
}
```

```
//-----
// Apskaičiuoja kabineto grindų dangos kainą
// p – kabineto plotas, kaina – dangos ploto vieneto kaina
double DangosKaina(double p, double kaina)
{
    return 1.03 * p * kaina;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti skaičių 128.75. Tai pinigų suma, reikalinga kabineto grindų dangai pirkti, kai kabineto grindų plotas yra 12,5 m<sup>2</sup>, o dangos vieno m<sup>2</sup> kaina lygi 10.

3

### Pradinių duomenų failo kūrimas, kintamųjų aprašymas

- Sukurkite tekstinį failą *Duomenys4.txt* ir į jį įrašykite pateiktus pavyzdyje pradinius duomenis.
- Aprašykite eilučių tipo konstantas pradinių duomenų ir rezultatų failų vardams atmintyje laikyti, funkcijų prototipus ir kintamuosius:

- n – kabinetų skaičius,
- s – visų kabinetų dangos kaina,
- a – kabineto ilgis,
- b – kabineto plotis,
- k – vieno kvadratinio metro dangos kaina.

```
const char CDFv[] = "Duomenys4.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai4.txt"; // rezultatų failo vardas
//-----
double Plotas(double a, double b);
double DangosKaina(double p, double kaina);
//-----
int main()
{
    int n; // kabinetų skaičius
    double s = 0; // pinigų suma, reikalinga visų kabinetų grindų dangai pirkti
    double a, b, k; // kabineto ilgis, plotis ir grindų dangos vieno kv. metro kaina
    cout << DangosKaina(12.5, 10) << endl;
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tokį pat rezultatą.

4

### Pradinių duomenų skaitymas iš failo

- Pašalinkite iš programos sakinį, kuriuo buvo tikrinama, ar teisingai skaičiuojama kabineto dangos kaina:

```
cout << DangosKaina(12.5, 10) << endl;
```

- Papildykite programą sakiniu, kuriuo įvesties srautas *fd* susiejamas su failu *Duomenys4.txt*:

```
ifstream fd(CDFv);
```

- Parašykite sakinį kabinetų skaičiui perskaityti:

```
fd >> n;
```

- Parašykite sakinį, kuris patikrintų, ar reikšmė perskaityta teisingai:

```
cout << n << endl;
```

- Parašykite ciklo sakinį, kad būtų galima patikrinti, ar vieno kabineto pradiniai duomenys perskaityti teisingai:

```
for (int i = 1; i <= n; i++) {  
    fd >> a >> b >> k; // kabineto ilgis, plotis, dangos vieno kv. metro kaina  
    cout << a << " " << b << " " << k << endl;  
}
```

- Irašykite ir įvykdysite programą. Ekrane turėtumėte matyti:

```
3  
4 5 54.60  
3.5 4.5 35.55  
9.2 3.3 42.63
```



## 5 Kreipinių į funkcijas rašymas

- Pagrindinėje funkcijoje main() aprašykite realiojo tipo kintamąjį p:

```
double p;
```

- Pagrindinės funkcijos main() cikle esantį sakinį

```
cout << a << " " << b << " " << k << endl;
```

pakeiskite tokiais sakiniais:

```
p = Plotas(a, b); // skaičiuojamas kabineto  
// grindų plotas  
cout << fixed << setprecision(2) << p << endl; // rodomas ekrane apskaičiuotas  
// kabineto grindų plotas
```

- Irašykite ir įvykdysite programą. Ekrane turėtumėte matyti kiekvieno kabineto grindų plotą:

```
3  
20.00  
15.75  
30.36
```

- Pagrindinę funkciją main() papildykite realiojo tipo kintamuoju dk (jį galite aprašyti ciklo viduje) ir parašykite sakinį:

```
dk = DangosKaina(p, k); // pinigų suma, reikalinga kabineto grindų dangai pirkti
```

- Sakinį

```
cout << fixed << setprecision(2) << p << endl;
```

pakeiskite tokiu, kad ekrane būtų rodoma kiekvieno kabineto dangos kaina:

```
cout << fixed << setprecision(2) << dk << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti pinigų sumas, reikalingas kiekvieno kabineto grindų dangai pirkti:

```
3
1124.76
576.71
1333.07
```



## 6 Visų kabinetų grindų dangai reikalingos pinigų sumos skaičiavimas

- Norint apskaičiuoti pinigų sumą, kurios prireiks visų kabinetų grindų dangai įsigyti, reikia sudėti pinigų sumas, apskaičiuotas kiekvienam kabinetui. Papildykite pagrindinę funkciją `main()`: po kreipiniais į abi funkcijas parašykite sumos skaičiavimo sakinį:

```
s = s + dk;
```

Tuomet ciklo sakinytis bus toks:

```
for (int i = 1; i <= n; i++) {
    double p, dk;
    fd >> a >> b >> k;
    p = Plotas(a, b);
    dk = DangosKaina(p, k);
    s = s + dk;
}
```

- Sumos skaičiavimą galima užrašyti vienu sakiniu:

```
s = s + DangosKaina(Plotas(a, b), k);
```

arba

```
s += DangosKaina(Plotas(a, b), k);
```

Tuomet ciklo sakinytis taps trumpesnis:

```
for (int i = 1; i <= n; i++) {
    fd >> a >> b >> k;
    s = s + DangosKaina(Plotas(a, b), k);
}
```

arba

```
for (int i = 1; i <= n; i++) {
    fd >> a >> b >> k;
    s += DangosKaina(Plotas(a, b), k);
}
```

- Tarpinių skaičiavimo rezultatų rodymo ekrane sakinius pašalinkite, o už ciklo parašykite tokį sakinį:

```
cout << fixed << setprecision(2) << s << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti apskaičiuotą pinigų sumą, reikalingą visų kabinetų grindų dangai įsigyti:

```
3034.54
```





## Rezultatų rašymas į failą *Rezultatai4.txt*

- Papildykite programą sakiniu, kuriuo išvesties srautas *fr* susiejamas su failu *Rezultatai4.txt*:

```
ofstream fr(CRfv);
```

- Parašykite apskaičiuotos pinigų sumos rašymo į failą sakinį:

```
fr << fixed << setprecision(2) << s << endl;
```

- Kai visi rezultatai bus surašyti, srautą užverkite:

```
fr.close();
```

- Pašalinkite sakinius, kurie skaičiavimo rezultatus rodydavo ekrane.
- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tik informacinį pranešimą.
- Atverkite rezultatų failą *Rezultatai4.txt*. Jame turėtų būti skaičius 3034.54 – pinigų suma, reikalinga grindų dangai įsigyti.

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDFv[] = "Duomenys4.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai4.txt"; // rezultatų failo vardas
//-----
double Plotas(double a, double b);
double DangosKaina(double p, double kaina);
//-----
int main()
{
    int n; // kabinetų skaičius
    double s = 0; // pinigų suma, reikalinga visų kabinetų grindų dangai pirkti
    double a, b, k; // kabineto ilgis, plotis ir grindų dangos vieno kv. metro kaina
    ifstream fd(CDFv);
    fd >> n;
    for (int i = 1; i <= n; i++) {
        fd >> a >> b >> k;
        s += DangosKaina(Plotas(a, b), k);
    }
    fd.close();
    ofstream fr(CRfv);
    fr << fixed << setprecision(2) << s << endl;
    fr.close();
    return 0;
}
```



## Programos patikrinimas

- Patikrinkite, kaip dirba programa, kai:
  - ✓ mokykla turi tik vieną kabinetą;
  - ✓ pradinių duomenų faile kabineto plotis ir ilgis surašomi bet kuria eilės tvarka;
  - ✓ pirmoje failo eilutėje kabinetų skaičius yra 0 (nulis).



## Programos papildymas

Yra žinoma, kad galimiems nuostoliams padengti dangos reikia pirkti daugiau.

- Papildykite programą taip, kad pirmoje pradinių duomenų failo eilutėje būtų pateikiamas visų kabinetų dangos nuostolių koeficientas. Paprastai tai sveikasis skaičius, kuris parodo, kiek procentų dangos reikia pirkti daugiau.
- Jeigu kiekvieno kabineto danga yra skirtinga, tuomet ir dangos nuostoliai nevienodi. Papildykite programą, kad būtų nurodomi kiekvieno kabineto dangos nuostolių procentai. Eilutėse, kuriose pateikiami kabinetų dydžiai ir dangos kaina, atsiras ketvirtas skaičius.

Kiekvienu atveju pasiruoškite kontrolinių duomenų. Iš anksto apskaičiuokite, kokie turėtų būti rezultatai.



## Užduotys

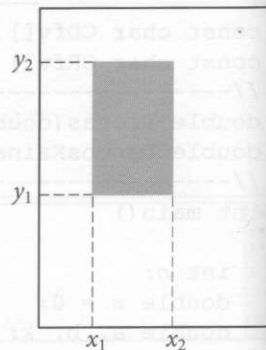
### 1. Siena

Tomas kolekcionuoja plakatus. Vieną dieną jis sugalvojo visus turimus plakatus suklijuoti ant savo kambario sienos. Šiek tiek pamąstęs, jis popieriaus lape sužymėjo tikslias plakatų vietas ant sienos. Tomas nori matyti visą savo kolekciją, todėl stengėsi, kad nė vienas plakatas neuždengtų kitų. Visi plakatai yra stačiakampiai ir pagal Tomo sudarytą planą jų kraštinės turi būti lygiagrečios su sienos šonais.

Tomo mama susirūpinusi – neseniai dažyta siena dabar bus uždengta plakatais. Jai įdomu, kiek sienos liko neuždengta. Parenkite programą, kuri apskaičiuotų nepaslėptos po plakatais sienos dalies plotą. Primename, kad stačiakampio plotas lygus jo kraštinių ilgių sandaugai.

Pradiniai duomenys pateikti faile. Pirmoje failo eilutėje įrašyti du skaičiai – sienos plotis  $p$  ir aukštis  $a$  centimetrais ( $1 \leq p, a \leq 1000$ ). Antroje eilutėje įrašytas vienas sveikasis skaičius  $N$  ( $1 \leq N \leq 1000$ ) – plakatų kiekis. Tolesnėse  $N$  eilučių įrašyta po keturis sveikuosius skaičius  $x_1, y_1, x_2, y_2$  ( $0 \leq x_1 < x_2 \leq p$ ;  $0 \leq y_1 < y_2 \leq a$ ):  $x_1$  – atstumas nuo kairiojo sienos krašto iki kairiosios plakato kraštinės;  $y_1$  – atstumas nuo apatinio sienos krašto iki apatinės plakato kraštinės; atitinkamai  $x_2$  ir  $y_2$  – atstumai iki plakato dešinėsios ir viršutinės kraštinių. Visi atstumai nurodyti centimetrais.

Į rezultatų failą įrašykite vieną skaičių – neuždengtos plakatais sienos dalies plotą kvadratiniais centimetrais.



Pradiniai duomenys	Rezultatas
300 200	45000
2	
50 100 150 150	
200 0 300 100	

(XVIII olimpiada, 2006)

### 2. Dangoraižis

Naujame dangoraižyje yra  $n$  kabinetų, sunumeruotų nuo 1 iki  $n$ . Prie kiekvieno kabineto durų reikia prikalti lenteles su to kabineto numeriu. Ant vienos lentelės užrašytas tik vienas skaitmuo. Jei kabineto numeris nevienženklis skaičius, tai prie durų kalamos kelios lentelės. Parenkite programą, kuri apskaičiuotų, kiek reikės lentelių, norint prikalti numerius prie visų dangoraižio kabinetų durų.

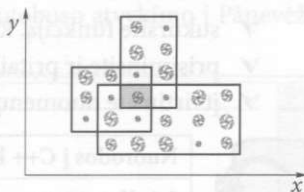
Pradinių duomenų faile įrašyta, kiek kabinetų yra dangoraižyje. Rezultatų faile turi būti įrašytas lentelių skaičius.

Pradinis duomuo	Rezultatas
16	23

### 3. 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. Parenkite programą, kuri apskaičiuotų, kiek vaismedžių panorėjo prižiūrėti visi trys draugai.



Pradiniai duomenys pateikiami trijose duomenų failo 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. Rezultatų faile turi būti įrašytas vienas skaičius – kiek vaismedžių panorėjo prižiūrėti visi trys draugai.

Pradiniai duomenys	Rezultatas
30 30 80 70	800
10 20 70 90	
50 20 100 90	

(X olimpiada, 1999)

### 4. Vandens pilstymas

Turime indų, kuriuose telpa  $V_1, V_2, V_3, \dots, V_n$  litrų vandens (tūriai yra sveikieji skaičiai). Parenkite programą, kuri patikrintų, ar galima šiais indais įpilti tiksliai  $V$  litrų vandens ( $V$  – sveikasis skaičius).

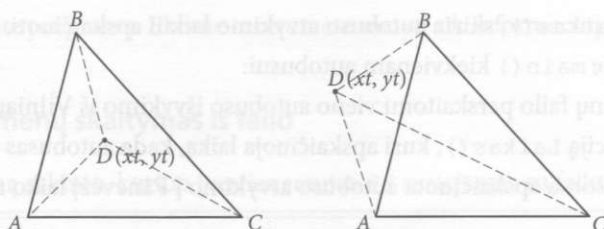
Pirmoje tekstinio failo eilutėje įrašytas indų skaičius  $n$  ( $n < 100$ ). Tolesnėse  $n$  eilučių įrašyta, kiek vandens telpa kiekviename inde. Paskutinėje failo eilutėje įrašytas tūris  $V$ . Į rezultatų failą reikia įrašyti žodį *Taip*, jei galima įpilti  $V$  litrų vandens, arba žodį *Ne*, jei vandens įpilti negalima.

Pradiniai duomenys	Rezultatas
2	Ne
2	
4	
3	

### 5. Trikampis ir taškas

Stačiakampėje koordinatinių plokštumoje nubraižytas trikampis, kurio viršūnių koordinatės yra žinomos. Plokštumoje padėtas taškas, kurio koordinatės yra  $(xt, yt)$ . Parenkite programą, kuri nustatytų, ar taškas yra trikampio viduje, ar jo išorėje. Atstumas tarp dviejų taškų  $(x_1, y_1)$  ir  $(x_2, y_2)$  yra skaičiuojamas pagal formulę  $atst = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ .

Jeigu taškas yra trikampio viduje, tai, tašką sujungus su trikampio viršūnėmis, gautų trikampių plotų suma bus lygi duoto trikampio plotui. Jeigu išorėje, tuomet gautų trikampių plotų suma bus didesnė už duoto trikampio plotą.



Kai taškas yra trikampio viduje, tai  $PlotasABC = PlotasABD + PlotasBCD + PlotasACD$ .

Visi skaičiavimai atliekami tam tikru tikslumu, nes duomenys yra realieji skaičiai (double tipo).

**Pasitikrinkite.** Kai  $A(10, 20)$ ,  $B(20, -5)$ ,  $C(-5, -4)$ ,  $xt = 5$ ,  $yt = 5$ , ekrane turi būti rodomas pranešimas: Taškas yra trikampio viduje.

Kai  $xt = -6$ ,  $yt = 9$ , ekrane turi būti rodomas pranešimas:

Taškas yra trikampio išorėje.



## 1.5. Funkcija su parametrais-nuorodomis

Atlikdami šį darbą:

- ✓ sukursite funkciją, kuri gražina apskaičiuotas kintamųjų reikšmes per parametrus-nuorodas;
- ✓ prisiminsite ir pritaikysite sveikųjų skaičių dalybos taisykles;
- ✓ įtvirtinsite duomenų skaitymo iš failo ir rezultatų rašymo į failą įgūdžius.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną

Nuorodos į algoritmų žinyną

2.1. Konstantos  
2.2. Operatoriai  
2.4. Duomenų skaitymas iš failo  
2.5. Rezultatų (duomenų) rašymas į failą  
2.6. Funkcijos  
2.11. Knygoje naudojamų įterpiamųjų failų sąrašas

### Užduotis

**Autobusų tvarkaraštis.** Maršrutu Vilnius–Panevėžys važiuoja  $n$  autobusų. Autobusų tvarkaraštyje nurodyta autobuso išvykimo iš Vilniaus laikas ir kelionės trukmė. Sukurkite funkciją, kuri apskaičiuotų, kada autobusas turi atvykti į Pānevėžį. Sukurtą funkciją panaudokite visų autobusų atvykimo laikui į Pānevėžį skaičiuoti. Rezultatus surašykite į failą.

Pirmoje pradinių duomenų failo eilutėje įrašytas autobusų skaičius  $n$ . Tolesnėse  $n$  eilučių įrašyta po keturis ta pais atskirtus skaičius – autobuso išvykimo iš Vilniaus laikas (val. ir min.) ir kelionės trukmė (val. ir min.).

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Paaiškinimai
3	Autobusų skaičius
6 10 1 45	Pirmas autobusas išvyksta 6 val. 10 min. ir kelionėje užtrunka 1 val. ir 45 min.
6 30 1 55	Antras autobusas išvyksta 6 val. 30 min. ir kelionėje užtrunka 1 val. ir 55 min.
7 05 1 35	Trečias autobusas išvyksta 7 val. 5 min. ir kelionėje užtrunka 1 val. ir 35 min.
Rezultatai	Paaiškinimai
7 55	Pirmas autobusas atvyksta 7 val. 55 min.
8 25	Antras autobusas atvyksta 8 val. 25 min.
8 40	Trečias autobusas atvyksta 8 val. 40 min.

### Algoritmas

Šios užduoties sprendimas skaidomas į dvi dalis:

1. Sukuriama funkcija `Laikas()`, skirta autobuso atvykimo laikui apskaičiuoti.
2. Pagrindinėje funkcijoje `main()` kiekvienam autobusui:
  - ✓ iš pradinių duomenų failo perskaitomi vieno autobuso išvykimo iš Vilniaus duomenys;
  - ✓ kreipiamasi į funkciją `Laikas()`, kuri apskaičiuoja laiką, kada autobusas atvyks į Pānevėžį;
  - ✓ į rezultatų failą įrašoma apskaičiuota autobuso atvykimo į Pānevėžį laiko reikšmė.



#### Funkcijos, skaičiuojančios autobuso atvykimo laiką, kūrimas

- Prieš pagrindinę funkciją parašykite funkcijos `Laikas()` prototipą:

```
void Laikas(int v1, int m1, int v2, int m2, int & v3, int & m3);
```

čia pirmi du parametrai ( $v_1$  ir  $m_1$ ) atitinka išvykimo laiką – valandas ir minutes. Kiti du parametrai ( $v_2$  ir  $m_2$ ) nurodo, kiek laiko autobusas sugaiš kelionėje. Paskutiniai du parametrai ( $v_3$  ir  $m_3$ ) skirti skaičiavimui, kada autobusas atvyks į paskirties vietą, rezultatams.

- Po pagrindine funkcija parašykite funkciją `Laikas()`, kuri skaičiuoja autobuso atvykimo į Pānevėžį laiką:

```
//-----  
// Apskaičiuoja autobuso atvykimo į paskirties vietą laiką  
// -v1, m1 – išvykimo laikas (val. ir min.)  
// v2, m2 – kelionės trukmė val. ir min.  
// v3, m3 – atvykimo į paskirties vietą laikas (val. ir min.)  
void Laikas(int v1, int m1, int v2, int m2, int & v3, int & m3)  
{  
    v3 = (v1 * 60 + m1 + v2 * 60 + m2) / 60;  
    m3 = (v1 * 60 + m1 + v2 * 60 + m2) % 60;  
}
```

Kintamieji  $v_3$  ir  $m_3$  rašomi su ženklu `&` (adreso operatorius), nes į pagrindinę funkciją `main()` grąžinamos apskaičiuotos jų reikšmės.

- Įrašykite ir įvykdykite programą. Ekране turėtumėte matyti tik informacinį pranešimą.

2

### Pradinių duomenų failo kūrimas, kintamųjų aprašymas

- Sukurkite tekstinį failą `Duomenys5.txt` ir į jį įrašykite pateiktus pavyzdyje pradinius duomenis.
- Aprašykite eilučių tipo konstantas pradinių duomenų ir rezultatų failų vardams atmintyje laikyti ir kintamuosius.

```
const char CDfv[] = "Duomenys5.txt"; // pradinių duomenų failo vardas  
const char CRfv[] = "Rezultatai5.txt"; // rezultatų failo vardas  
//-----  
void Laikas(int v1, int m1, int v2, int m2, int & v3, int & m3);  
//-----  
int main()  
{  
    int n; // autobusų skaičius  
    int v1, m1, // autobuso išvykimo laikas (val. ir min.)  
        v2, m2, // kelionės trukmė (val. ir min. skaičiai)  
        v3, m3; // atvykimo į paskirties vietą laikas (val. ir min.)  
    return 0;  
}
```

- Įrašykite ir įvykdykite programą. Ekране turėtumėte matyti tik informacinį pranešimą.

3

### Pradinių duomenų skaitymas iš failo

- Papildykite programą sakiniu, kuriuo įvesties srautas `fd` susiejamas su failu `Duomenys5.txt`:

```
ifstream fd(CDfv);
```

- Parašykite sakinį, kuris perskaitytų iš failo autobusų skaičiaus reikšmę:

```
fd >> n;
```

- Parašykite sakinį, kuris ekrane parodytų perskaitytą reikšmę:

```
cout << n << endl;
```

- Parašykite ciklo sakinį, kuriuo iš failo būtų skaitomas ir ekrane parodomas kiekvieno autobuso išvyimo laikas ir kelionės trukmė:

```
for (int i = 1; i <= n; i++) {  
    fd >> v1 >> m1 >> v2 >> m2;  
    cout << v1 << " " << m1 << " " << v2 << " " << m2 << endl;  
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti pradinį duomenis:

```
3  
6 10 1 45  
6 30 1 55  
7 5 1 35
```

#### 4 Kreipinys į funkciją Laikas ()

- Ciklo sakinyje parašykite kreipinį į funkciją Laikas ():

```
Laikas(v1, m1, v2, m2, v3, m3);
```

- Sakinį

```
cout << v1 << " " << m1 << " " << v2 << " " << m2 << endl;
```

pakeiskite tokiu, kad ekrane būtų rodoma, kada autobusas atvyks į Pānevėžį:

```
cout << v3 << " " << m3 << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti:

```
3  
7 55  
8 25  
8 40
```

- Pašalinkite sakinį, kuris ekrane parodo autobusų skaičiaus reikšmę.

#### 5 Rezultatų rašymas į failą Rezultatai5.txt

- Papildykite programą sakiniu, kuriuo išvesties srautas `fr` susiejamas su failu `Rezultatai5.txt`:

```
ofstream fr(CRfv);
```

- Sakinį, kuriuo ekrane buvo rodomas autobuso atvykimo į Pānevėžį laikas, pakeiskite tokiu, kad atbuso atvykimo į Pānevėžį laikas būtų rašomas į failą:

```
fr << v3 << " " << m3 << endl;
```

- Kai visi rezultatai bus surašyti, srautą užverkite:

```
fr.close();
```

- Įrašykite ir įvykdysite programą. Ekране turėtumėte matyti tik informacinį pranešimą.
- Atverkite failą *Rezultatai5.txt*. Jame turėtų būti pavyzdyje pateikti rezultatai.

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDfv[] = "Duomenys5.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai5.txt"; // rezultatų failo vardas
//-----
void Laikas(int v1, int m1, int v2, int m2, int & v3, int & m3);
//-----
int main()
{
    int n; // autobusų skaičius
    int v1, m1, // autobuso išvykimo laikas (val. ir min.)
        v2, m2, // kelionės trukmė (val. ir min. skaičiai)
        v3, m3; // atvykimo į paskirties vietą laikas (val. ir min.)
    ifstream fd(CDfv);
    ofstream fr(CRfv);
    fd >> n;
    for (int i = 1; i <= n; i++) {
        fd >> v1 >> m1 >> v2 >> m2;
        Laikas(v1, m1, v2, m2, v3, m3);
        fr << v3 << " " << m3 << endl;
    }
    fd.close ();
    fr.close ();
    return 0;
}
```



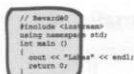
### Programos patikrinimas

- Patikrinkite, kaip dirba programa, kai:
  - ✓ pradinių duomenų faile yra tik vieno autobuso duomenys;
  - ✓ autobusas kelionėje užtrunka kelias minutes (kai išvykimo laikas yra, pavyzdžiui, 15 val. 02 min. ar 15 val. 59 min., o kelionės trukmė – tik 10 min.).



### Programos papildymas

- Patikrinkite, koks bus rezultatas, kai, pavyzdžiui, autobusas išvyksta 23 val. 50 min. ir kelionėje užtrunka 2 val. ir 10 min. Programa apskaičiuos, kad autobusas turi atvykti 26 val. ir 0 min. Taigi laiko skaičiavimo programa dirba gerai tada, kai autobusas išvyksta ir atvyksta tą pačią parą. Tam, kad programa visada nurodytų tinkamą atvykimo laiką, reikia koreguoti funkcijos *Laikas()* valandų skaičiavimo dalį.
- Įsitinkite, ar programa dirba gerai, kai autobusas iš Pānevėžio išvyksta 15 val. 55 min. ir kelionėje į Lūdoną užtrunka 48 val. ir 22 min. Jeigu reikia, tikslinkite programos dalį, kurioje skaičiuojamas atvykimo laikas.



## Užduotys

### 1. Colinė matavimo sistema

Pasaulyje vis dar labai populiarī colinė matavimo sistema: 12 colių sudaro 1 pėdą, 3 pėdos – 1 jarda, 1 colis = 2,54 cm. Parenkite programą, kuri metrinės matavimo sistemos vienetų, t. y. metrų nuo 1 iki 5, paverstų colinės matavimo sistemos vienetais.

Rezultatai			
Metrai	Coliai	Pėdos	Jardai
1	39.37	3.28	1.09
2	78.74	6.56	2.19
3	118.11	9.84	3.28
4	157.48	13.12	4.37
5	196.85	16.40	5.47

### 2. obuolių sultys

Geraū užderėjus obuolių derliui, ūkininkai nusprendė gaminti obuolių sultis. Pagamintas sultis išpilstė į 2 ir 1 litro talpos indus. Kiekvienas ūkininkas pirmiausia užpildė 5, po to – 2 ir po to – 1 litro talpos indus. Parenkite programą, kuri apskaičiuotų, kiek 5, 2 ir 1 litro indų sulčių buvo pagaminta.

Pirmoje pradinių duomenų failo eilutėje įrašytas ūkininkų skaičius  $n$ . Tolesnėse  $n$  eilučių – kiekvieno ūkininko pagamintų obuolių sulčių kiekis litrais. Į rezultatų failą turi būti surašyti kiekvieno ūkininko pagamintų obuolių sulčių 5, 2 ir 1 litro talpos indų skaičiai, vienas nuo kito atskirti tarpais. Kiekvienam ūkininkui skiriama viena eilutė.

Pradiniai duomenys	Rezultatai
3	9 0 0
45	18 1 0
92	6 1 1
33	

### 3. Omo dėsnis

Vienas svarbiausių elektrotechnikos dėsnų yra Omo dėsnis. Jis teigia, kad elektros srovės stipris  $I$  grandinėje yra tiesiog proporcingas įtampai  $U$  grandinės dalies galuose ir atvirkščiai proporcingas varžai  $R$ :

$$I = \frac{U}{R}$$

Pirmoje pradinių duomenų failo eilutėje įrašytas skaičius  $n$ , rodantis, kiek bandymų buvo atlikta. Tolesnėse  $n$  eilučių įrašytos  $I$ ,  $U$  ir  $R$  reikšmės, atskirtos tarpais. Du dydžiai yra žinomi, vienas – nežinomas. Nežinomas dydis faile žymimas 0. Parenkite programą, kuri apskaičiuotų ir išspausdintų lentelę visų dydžių reikšmės vienetų tikslumu.

Pradiniai duomenys	Rezultatai
3	Bandymų rezultatai
5 5 0	-----
0 5 5	I U R
5 0 5	-----
	5 5 1
	1 5 5
	5 25 5
	-----



#### 4. Paprastosios trupmenos

Paprastąją trupmeną sudaro dvi dalys: skaitiklis ir vardiklis. Penktokai mokosi atlikti veiksmus su paprastosiomis trupmenomis: sudėti, atimti, dauginti, dalyti. Rengdama savarankiško darbo užduotis, matematikos mokytoja sukūrė pradinių duomenų failą. Pirmoje jo eilutėje yra savarankiško darbo užduočių skaičius  $n$ . Tolesnėse  $n$  eilučių įrašyta po keturis atskirtus tarpais sveikuosius skaičius: pirmosios trupmenos skaitiklis, pirmosios trupmenos vardiklis, antrosios trupmenos skaitiklis ir antrosios trupmenos vardiklis. Padėkite mokytojai – parenkite programą, kuri apskaičiuotų trupmenų sumą, skirtumą, sandaugą, dalmenį ir juos suprastintų. Rezultatus pateikite lentele tekstiniam failui.

Pradiniai duomenys	Rezultatai					
	T1	T2	Suma	Skirtumas	Sandauga	Dalmuo
3						
4 5 3 4						
3 5 2 9						
7 8 2 3						
	4/5	3/4	31/20	1/20	3/5	16/15
	3/5	2/9	37/45	17/45	2/15	27/10
	7/8	2/3	37/24	5/24	7/12	21/16

#### 5. Internetas

Vilius turi daug įvairių užsiėmimų, todėl jo laikas griežtai suplanuotas. Namuose jis būna tik dalį dienos ir kasdien prie kompiuterio dirba tuo pačiu laiku – nuo  $A_h$  valandų  $A_{min}$  minučių iki  $B_h$  valandų  $B_{min}$  minučių. Lygiai vidurnaktį Vilius eina miegoti, t. y. jis pradeda dirbti kompiuteriu ir baigia tą pačią parą.

Deja, ir per tą trumpą laiką, leidžiamą prie kompiuterio, Vilius ne visada gali pasidžiaugti geru interneto ryšiu. Tuo pačiu paros metu ryšys tai dingsta, tai vėl atsiranda. Be to, interneto teikėjas teigia, kad tai vyksta ne dėl teikėjo kaltės. Norėdamas informuoti klientus, interneto teikėjas viešai paskelbė laiko intervalus, kada stringa interneto ryšys.

Parenkite programą, kuri apskaičiuotų, kiek minučių Vilius gali naudotis internetu be trukdžių.

Pirmoje pradinių duomenų failo eilutėje yra keturi tarpais atskirti sveikieji skaičiai  $A_h, A_{min}, B_h, B_{min}$  ( $0 \leq A_h, B_h < 24; 0 \leq A_{min}, B_{min} < 60$ ), apibūdinantys laiko, kurį Vilius praleidžia prie kompiuterio, intervalą.

Antroje eilutėje nurodytas trukdžių intervalų skaičius  $k$  ( $0 < k < 1440$ ). Toliau kiekvienoje eilutėje įrašyta po keturis sveikuosius skaičius  $A_{k,h}, A_{k,min}, B_{k,h}, B_{k,min}$ , kurie apibūdina  $k$ -ąjį trukdžių intervalą (atitinkamai intervalo pradžios valanda ir minutė, pabaigos valanda ir minutė). Nurodyti intervalai nesikerta, priklauso tai pačiai parai ir surikiuoti pagal intervalų pradžios laiką didėjančiai.

Rezultatų failo eilutėje turi būti įrašytas skaičius, kuris rodo, kiek minučių per dieną Vilius gali naudotis internetu be trukdžių.

Pradiniai duomenys	Rezultatas	Paiškinimai
12 0 12 59 2 12 58 13 0 13 0 20 0	58	Prie kompiuterio Vilius praleidžia 59 minutes. Viliui dirbant kompiuteriu, internetas sutrinka vienai minutei (nuo 12.58 iki 12.59). Kitas trukdis (nuo 13.00 iki 20.00) Viliui neaktualus, nes tuo metu jis nedirba kompiuteriu. Laikas be trukdžių: 59 min – 1 min = 58 min.

(XIX olimpiada, 2007)



## 1.6. Pažintis su masyvu

Atlikdami šį darbą:

- ✓ sužinosite, kas yra masyvas ir kaip jis aprašomas;
- ✓ išmoksite duomenis skaityti iš failo į masyvą, kai žinomas duomenų kiekis;
- ✓ išmoksite masyvo reikšmes rašyti į failą lentele;
- ✓ pritaikysite sumos, kiekio ir vidurkio skaičiavimo algoritmus darbui su masyvo reikšmėmis.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.1. Konstantos	3.1. Sumos skaičiavimo algoritmas
2.2. Operatoriai	3.3. Kiekio skaičiavimo algoritmas
2.4. Duomenų skaitymas iš failo	3.4. Aritmetinio vidurkio skaičiavimo algoritmas
2.5. Rezultatų (duomenų) rašymas į failą	
2.6. Funkcijos	
2.7. Masyvas	
2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	

### Užduotis

**Krituliai.** Lietuvos hidrometeorologijos tarnyba kaupia įvairiose Lietuvos vietovėse vasarą iškritusių kritulių stebėjimų duomenis.

Parenkite programą, kuri pateiktų informaciją apie lietingas dienas:

- ✓ kiek milimetrų kritulių iškrito iš viso;
- ✓ kiek dienų nelijo;
- ✓ kiek milimetrų kritulių vidutiniškai iškrito tomis dienomis, kai lijo.

Pirmoje failo eilutėje yra nurodytas stebėtų dienų skaičius. Antroje eilutėje surašyti kiekvieną stebėtų dienų iškritusių kritulių kiekiai (milimetrais). Jeigu kurią nors dieną nelijo, tai kritulių kiekis tą dieną lygus 0.

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Paaiškinimai
10 10 20 0 0 0 45 25 30 50 25	Stebėtų dienų skaičius Kiekvienos stebėtos dienos kritulių kiekis
Rezultatai	Paaiškinimai
Krituliai (lietus) ----- Diena Kritulių kiekis (mm) ----- 1 10 2 20 3 0 4 0 5 0 6 45 7 25 8 30 9 50 10 25 -----	Pradiniai duomenys
Iš viso iškrito kritulių (mm): 205 Nelijo (dienas): 3 Vidutiniškai kiekvieną lietingą dieną iškrito kritulių (mm): 29	Skaičiavimų rezultatai

## Algoritmas

Užduotis sprendžiama taip:

1. Iš failo į masyvą skaitomi pradiniai duomenys.
2. Masyvo reikšmės rašomos į failą lentele: diena, kritulių kiekis.
3. Atliekami skaičiavimai:
  - ✓ kiek milimetrų kritulių iškrito iš viso;
  - ✓ kiek dienų nelijo;
  - ✓ kiek milimetrų kritulių vidutiniškai iškrito tomis dienomis, kai lijo.
4. Apskaičiuoti rezultatai rašomi į failą.

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Skaityti()	Pradinių duomenų skaitymas iš failo į masyvą
Spausdinti()	Masyvo reikšmių rašymas į failą lentele
KrituliuKiekis()	Kritulių kiekio (sumos) skaičiavimas
DienuSkaicius()	Nelietingų dienų skaičiaus radimas
KrituliuVidurkis()	Lietingomis dienomis iškritusių kritulių vidurkio skaičiavimas



### Pradinių duomenų failo kūrimas, konstantų ir kintamųjų aprašymas

- Sukurkite tekstinį failą *Duomenys6.txt* ir į jį įrašykite pateiktus pavyzdyje pradinius duomenis.
- Prieš pagrindinę funkciją `main()` aprašykite konstantas pradinių duomenų ir rezultatų failų vardams bei masyvo dydžiui atmintyje laikyti.
- Pagrindinėje funkcijoje aprašykite sveikųjų skaičių masyvą `Kr(n)` duomenims – kritulių kiekiams – atmintyje laikyti.

```
const char Cdfv[] = "Duomenys6.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai6.txt"; // rezultatų failo vardas
const int CMax = 100; // masyvo dydis
//-----
// Funkcijų prototipai
//-----
int main()
{
    int Kr[CMax]; int n; // kritulių kiekių masyvas ir jame laikomų reikšmių kiekis
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekране turėtumėte matyti tik informacinį pranešimą.



### Pradinių duomenų skaitymas iš failo į masyvą `Kr(n)`

- Sukurkite funkciją `Skaityti()`, kuri iš duomenų failo skaitytų pradinius duomenis.
- Parašykite funkcijos `Skaityti()` prototipą:

```
void Skaityti(int A[], int &n);
```

čia parametras `A` skirtas sveikųjų skaičių masyvui, o `n` – jame laikomų reikšmių skaičiui įsiminti.

- Parašykite funkcijos `Skaityti()` tekstą:

```
// Skaito duomenis iš failo CDfv į masyvą A(n)
void Skaityti(int A[], int &n)
{
    ifstream fd(CDfv);           // atidaromas įvesties srautas
    fd >> n;                     // perskaitomas masyvo reikšmių kiekis
    for (int i = 0; i < n; i++)
        fd >> A[i];             // perskaitoma i-oji reikšmė
    fd.close();                 // uždaromas įvesties srautas
}
```

- Funkcijoje `main()` parašykite kreipinį į funkciją `Skaityti()` ir perskaitytos kintamojo `n` reikšmės rodymo ekrane sakinius:

```
Skaityti(Kr, n);
cout << n << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane matysite stebėtų dienų skaičių:

```
10
```

Į masyvą `Kr(n)` iš failo buvo perskaityta 10 reikšmių.

- Iš programos pašalinkite eilutę, kad kintamojo `n` reikšmė nebūtų rodoma ekrane.



### Masyvo `Kr(n)` reikšmių (kritulių kiekių) rašymas į failą lentele

- Norint peržiūrėti, kokie duomenys buvo perskaityti iš failo į masyvą, būtina juos įrašyti. Sukurk funkciją `Spausdinti()`, kuri įrašytų lentele masyvo `Kr(n)` reikšmes į rezultatų failą.
- Parašykite funkcijos `Spausdinti()` prototipą:

```
void Spausdinti(int A[], int n);
```

- Parašykite funkcijos `Spausdinti()` tekstą:

```
// Rašo lentele masyvo A(n) reikšmes į failą CRfv
void Spausdinti(int A[], int n)
{
    ofstream fr(CRfv);           // atidaromas išvesties srautas
    fr << "    Krituliai (lietus)    " << endl;
    fr << "-----" << endl;
    fr << "Diena Kritulių kiekis (mm)" << endl;
    fr << "-----" << endl;
    for (int i = 0; i < n; i++)
        fr << setw(4) << i+1 << "          " << setw(3) << A[i] << endl;
    fr << "-----" << endl;
    fr.close();                 // uždaromas išvesties srautas
}
```

- Funkcijoje `main()` parašykite kreipinį į funkciją `Spausdinti()`:

```
Spausdinti(Kr, n);
```

- Įrašykite ir įvykdykite programą.

- Atverkite rezultatų failą *Rezultatai6.txt*. Jame pamatysite kritulių kiekių lentelę:

Krituliai (lietus)	
Diena	Kritulių kiekis (mm)
1	10
2	20
3	0
4	0
5	0
6	45
7	25
8	30
9	50
10	25



4

### Viso kritulių kiekio (sumos) skaičiavimas

- Parašykite funkcijos `KrituliuKiekis()` prototipą:

```
int KrituliuKiekis(int A[], int n);
```

- Parašykite funkcijos `KrituliuKiekis()` tekstą:

```
// Apskaičiuoja ir grąžina masyvo A (n) reikšmių sumą
int KrituliuKiekis(int A[], int n)
{
    int suma = 0;
    for (int i = 0; i < n; i++)
        suma = suma + A[i];
    return suma;
}
```

- Papildykite funkciją `main()` kintamaisiais `kk` ir `fr`:

```
int kk; // visas kritulių kiekis (suma)
ofstream fr; // išvesties srautas
```

- Funkcijoje `main()` atverkite papildyti išvesties srautą `fr`:

```
fr.open(CRfv, ios::app); // atidaro išvesties srautą papildyti
```

- Parašykite kreipinį į funkciją `KrituliuKiekis()`:

```
kk = KrituliuKiekis(Kr, n);
```

- Įrašykite gautą rezultatą į rezultatų failą, o po to užverkite išvesties srautą:

```
fr << "Iš viso iškrito kritulių (mm): " << kk << endl;
fr.close(); // uždaro išvesties srautą
```

- Įrašykite ir įvykdykite programą. Atverkite rezultatų failą *Rezultatai6.txt*. Jame turėtų būti papildoma eilutė:

```
Iš viso iškrito kritulių (mm): 205
```

**5****Nelietingų dienų skaičiavimas**

- Parašykite funkcijos `DienuSkaicius()` prototipą:

```
int DienuSkaicius(int A[], int n);
```

- Parašykite funkcijos `DienuSkaicius()` tekstą:

```
// Apskaičiuoja, kiek masyve A(n) yra nulinių reikšmių, ir jas grąžina
int DienuSkaicius(int A[], int n)
{
    int kiek = 0;
    for (int i = 0; i < n; i++)
        if (A[i] == 0)
            kiek = kiek + 1;
    return kiek;
}
```

- Papildykite funkciją `main()` kintamuoju `kd`:

```
int kd; // kiek dienų nelijo
```

- Funkcijoje `main()` parašykite kreipinį į funkciją `DienuSkaicius()`:

```
kd = DienuSkaicius(Kr, n);
```

- Įrašykite gautą rezultatą į failą:

```
fr << "Nelijo (dienas): " << kd << endl;
```

- Įrašykite ir įvykdykite programą. Atverkite rezultatų failą *Rezultatai6.txt*. Jame turėtų būti papildomai eilutė:

```
Nelijo (dienas): 3
```

**6****Lietingomis dienomis iškritusių kritulių vidurkio skaičiavimas**

- Parašykite funkcijos `KrituliuVidurkis()` prototipą:

```
int KrituliuVidurkis(int A[], int n);
```

- Parašykite funkcijos `KrituliuVidurkis()` tekstą:

```
// Apskaičiuoja masyvo A(n) teigiamųjų reikšmių vidurkį ir jį grąžina
// Jeigu masyve teigiamųjų reikšmių nebuvo, grąžina 0 (nulį)
int KrituliuVidurkis(int A[], int n)
{
    int suma = 0;
    int kiek = 0;
    for (int i = 0; i < n; i++)
        if (A[i] > 0) {
            suma = suma + A[i];
            kiek = kiek + 1;
        }
    if (kiek > 0)
        return suma / kiek;
    return 0;
}
```

- Papildykite funkciją main() kintamuoju vid:

```
int vid; // kritulių vidurkis lietingomis dienomis
```

- Funkcijoje main() parašykite kreipinį į funkciją KrituliuVidurkis():

```
vid = KrituliuVidurkis(Kr, n);
```

- Įrašykite gautą rezultatą į rezultatų failą:

```
fr << "Vidutiniškai kiekvieną lietingą dieną iškrito kritulių (mm): "  
<< vid << endl;
```

- Įrašykite ir įvykdykite programą.

- Atverkite rezultatų failą Rezultatai6.txt. Jame pamatysite visus sukurtos programos rezultatus:

Krituliai (lietus)

-----  
Diena Kritulių kiekis (mm)  
-----

1	10
2	20
3	0
4	0
5	0
6	45
7	25
8	30
9	50
10	25

-----  
Iš viso iškrito kritulių (mm): 205

Nelijo (dienas): 3

Vidutiniškai kiekvieną lietingą dieną iškrito kritulių (mm): 29

## Pagrindinė funkcija

Seکliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDFv[] = "Duomenys6.txt"; // pradinių duomenų failo vardas  
const char CRfv[] = "Rezultatai6.txt"; // rezultatų failo vardas  
const int CMax = 100; // masyvo dydis
```

```
//-----  
void Skaityti(int A[], int & n);  
void Spausdinti(int A[], int n);  
int KrituliuKiekis(int A[], int n);  
int DienuSkaicius(int A[], int n);  
int KrituliuVidurkis(int A[], int n);  
//-----
```

```
int main ()
```

```
{  
    int Kr[CMax]; int n; // kritulių kiekių masyvas  
    int kk; // visas kritulių kiekis (suma)  
    int kd; // kiek dienų nelijo  
    int vid; // kritulių vidurkis lietingomis dienomis  
    ofstream fr; // išvesties srautas
```

```

Skaityti(Kr, n);
Spausdinti(Kr, n);
fr.open(CRfV, ios::app); // atidaro išvesties srautą papildyti
kk = KrituliuKiekis(Kr, n);
fr << "Iš viso iškrito kritulių (mm): " << kk << endl;
kd = DienuSkaicius(Kr, n);
fr << "Nelijo (dienas): " << kd << endl;
vid = KrituliuVidurkis(Kr, n);
fr << "Vidutiniškai kiekvieną lietingą dieną iškrito kritulių (mm): "
<< vid << endl;
fr.close(); // uždaro išvesties srautą
return 0;
}

```



### Programos patikrinimas

- Paruoškite skirtingų duomenų rinkinius ir patikrinkite, ar programa visais atvejais pateikia teisingus rezultatus. Siūlome patikrinti programą tais atvejais, kai:
  - ✓ kiekvieną stebėtą dieną lijo;
  - ✓ nė vieną stebėtą dieną nelijo;
  - ✓ iškritusių kritulių reikšmės pradinių duomenų faile surašytos keliose eilutėse.



### Programos papildymas

- Papildykite programą, kad ji apskaičiuotų:
  - ✓ kiek dienų lijo;
  - ✓ kiek vidutiniškai kritulių iškrito per visas stebėtas dienas.

Kiekvienu atveju pasiruoškite kontrolinių duomenų rinkinius. Iš anksto apskaičiuokite, kokie turėtų būti rezultatai.



### Užduotys

#### 1. Gyventojai

Pirmoje pradinių duomenų failo eilutėje įrašytas natūralusis skaičius  $n$  ( $n < 500$ ). Jis nurodo, kiek gatvėje yra namų. Toliau yra  $n$  eilučių, kurių kiekvienoje yra du tarpais atskirti sveikieji skaičiai: namo numeris ir gyventojų tame name skaičius. Namai, kurių numeriai lyginiai, stovi dešiniojoje, o kurių nelyginiai – kairiojoje gatvės pusėje. Parenkite programą, kuri apskaičiuotų ir į rezultatų failą įrašytų:

- ✓ kiek gyventojų iš viso gyvena gatvėje;
- ✓ kiek gyventojų gyvena kairiojoje gatvės pusėje ir kiek – dešiniojoje;
- ✓ kiek vidutiniškai gyventojų gyvena kiekviename name, esančiame kairiojoje gatvės pusėje, ir kiek – dešiniojoje.



Pradiniai duomenys	Rezultatai	Paiškinimai
10	34	Iš viso gyventojų gatvėje
1 5	20	Iš viso gyventojų kairiojoje gatvės pusėje
2 4	14	Iš viso gyventojų dešiniojoje gatvės pusėje
3 4	4.00	Vidutinis gyventojų skaičius name, esančiame kairiojoje gatvės pusėje
4 5	2.80	Vidutinis gyventojų skaičius name, esančiame dešiniojoje gatvės pusėje
5 6		
6 5		
7 0		
8 0		
9 5		
10 0		

## 2. Ūgiai

Pirmoje pradinių duomenų failo eilutėje nurodytas klasės mokinių skaičius  $n$  ( $n < 30$ ). Tolesnėse  $n$  eilutėse įrašytas kiekvieno klasės mokinio ūgis centimetrais. Merginų ūgiai yra teigiamieji, vaikinių – neigiamieji sveikieji skaičiai. Parenkite programą, kuri apskaičiuotų ir į rezultatų failą įrašytų:

- ✓ klasės mokinių vidutinį ūgį 0,1 cm tikslumu;
- ✓ klasės merginų vidutinį ūgį 0,01 cm tikslumu;
- ✓ klasės vaikinių vidutinį ūgį 0,01 cm tikslumu;
- ✓ pranešimą, ar galima sudaryti iš klasės merginų ir vaikinių krepšinio komandas. Reikalavimai krepšinio komandai: ūgis ne mažesnis negu 175 cm, komandoje turi būti 7 žaidėjai (5 pagrindiniai ir 2 atsarginiai).

Pradiniai duomenys	Rezultatai
13	175.8
-178	172.57
175	179.67
-186	Merginų komandos sudaryti negalima
172	Vaikinų komandos sudaryti negalima
173	
175	
-185	
-180	
-169	
165	
176	
172	
-180	

## 3. Skaitytiniai

Pirmoje pradinių duomenų failo eilutėje nurodytas klasės mokinių skaičius  $n$  ( $n < 30$ ). Tolesnėse  $n$  eilutėse įrašyta po vieną sveikąjį skaičių – kiek kiekvieno mokinio per mokslo metus (10 mėnesių) perskaityta knygų. Parenkite programą, kuri į rezultatų failą įrašytų:

- ✓ kiek iš viso knygų per mokslo metus perskaitė mokiniai;
- ✓ kiek vidutiniškai knygų per mokslo metus perskaitė vienas mokinys (rezultatas apvalinamas iki sveikąjo skaičiaus);
- ✓ kiek vidutiniškai knygų per mėnesį perskaitė vienas mokinys (rezultatas pateikiamas vieno ženklo po kablelio tikslumu).

Pradiniai duomenys	Rezultatai
5	65
12	13
13	1.3
12	
15	
13	

#### 4. Garso signalai

Garso signalas gali būti koduojamas sveikųjų skaičių seka. Šie skaičiai rodo signalo stiprumą periodiniais laiko intervalais. Signalą iškraipantis triukšmas šiek tiek pakeičia tų skaičių reikšmes.

Signalas „išlyginimo“ metu triukšmas pašalinamas tokiu būdu: kiekvienas skaičius keičiamas jo ir dviejų jam gretimų skaičių vidurkiu (vidutinės reikšmės sveikąja dalimi). Pirmas ir paskutinis skaičiai atitinkamai keičiami dviejų pirmųjų arba dviejų paskutinių skaičių vidurkiu.

Faile yra  $n$  ( $0 < n < 50$ ) garso signalus atitinkančių skaičių sekų. Kiekviena seka sudaryta iš  $k$  ( $1 < k < 50$ ) sveikųjų skaičių ir užrašyta vienoje failo eilutėje, skaičiai atskirti tarpais. Pirmoje failo eilutėje yra  $n$  ir  $k$  reikšmės, atskirtos tarpais.

Į rezultatų failą surašykite „išlygintus“ garso signalus atitinkančias skaičių sekas atskiromis eilutėmis.

Pradiniai duomenys	Rezultatai
3 5	5 4 5 5 6
4 7 3 5 8	8 8 8 7 7
8 9 7 8 6	4 5 5 6 6
5 4 6 7 6	

#### 5. Akmenukai

Eilė stovi  $n$  vaikų. Jie sunumeruoti (iš kairės į dešinę) iš eilės nuo 1 iki  $n$ . Kiekvienas vaikas arba turi saujoje akmenukų, arba sauja yra tuščia. Vaikai paėmiami dalija akmenukus stovintiems dešinėje vaikams.

Pirmasis vaikas duoda vieną akmenuką antrajam, po to – vieną akmenuką trečiajam ir t. t. Taip dalydamas akmenukus, jis eina į dešinią eilės galą, po to atgal link savo vietos kiekvienam vaikui vėl duodamas po akmenuką. Dešiniajame eilės gale stovintis ( $n$ -asis) vaikas vis gauna po akmenuką, kai dalijantis akmenukus vaikas eina į priekį ir atgal.

Vaikas nustoja dalyti akmenukus, kai jis prieina savo vietą arba pasibaigia akmenukai. Jeigu vaikui pasibaigia akmenukai dar nepriėjus savo vietos, jis grįžta tuščiomis rankomis. Jeigu vaikas prieina savo vietą turėdamas rankose akmenukų, jis atsistoja ir pasilieka likusius akmenukus.

Toliau analogiškai akmenukus dalija antrasis vaikas, trečiasis ir t. t. Paskutinis ( $n$ -asis) vaikas, stovintis dešiniajame eilės gale, akmenukų nedalija.

Akmenukai dalijami tik dešinėje stovintiems vaikams, todėl antrasis vaikas duoda akmenukų tik trečiajam, ketvirtajam ir t. t., trečiasis – ketvirtajam, penktajam ir t. t.

Parenkite programą, kuri apskaičiuotų, kiek akmenukų turės kiekvienas vaikas, kai baigsis dalijimas.

Pirmoje pradinių duomenų failo eilutėje nurodytas vaikų skaičius ( $3 \leq n \leq 10$ ). Antroje eilutėje įrašyta  $n$  tarpais atskirtų sveikųjų neneigiamųjų skaičių – kiekvieno vaiko turimas akmenukų skaičius.

Rezultatų failo vienoje eilutėje turi būti pateikiami kiekvieno vaiko turimi akmenukų skaičiai, atskirti tarpais.

Pradiniai duomenys	Rezultatai
5	0 1 0 15 16
3 6 1 12 10	

(XV olimpiada, 2005)



## 1.7. Didžiausios ir mažiausios reikšmių paieška

Atlikdami šį darbą:

- ✔ įtvirtinsite masyvo skaitymo ir rašymo į rezultatų failą įgūdžius;
- ✔ pritaikysite didžiausios ir mažiausios reikšmės paieškos masyve algoritmus.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną

Nuorodos į algoritmų žinyną

2.7. Masyvas

3.5. Didžiausios reikšmės paieškos algoritmas

### Užduotis

**Gėlių puokštė.** Gimtadienio proga Jokūbas padovanojo savo draugei Ievai didelę ką tik pražydusių gėlių puokštę. Jis galvojo, kad tol, kol gėlės žydės, jo draugė svajos tik apie jį.

Žinodami, kiek dienų žydi puokštėje esanti kiekvienos rūšies gėlė, parašykite programą, kuri apskaičiuotų:

- ✔ po kelių dienų visos gėlės nuvys;
- ✔ kurių rūšių gėlės reiktų dėti į puokštę, kad ji ilgiausiai nenuvystų (kelios vienodos didžiausios reikšmės).

Pirmoje pradinių duomenų failo eilutėje nurodytas gėlių rūšių skaičius. Kitose eilutėse yra nurodyta kiekvienos rūšies gėlės žydėjimo trukmė (dienomis).

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Rezultatai
7	Puokštės gėlės
12	-----
9	G. Nr.    Ž. laikas
16	-----
4	1        12
16	2        9
16	3        16
8	4        4
	5        16
	6        16
	7        8
	-----
	Visos gėlės nuvys po 16 d.
	Gėlių, kurias reiktų dėti į puokštę, numeriai:
	3 5 6

### Algoritmas

Užduotis sprendžiama taip:

1. Iš failo į masyvą skaitomi duomenys – gėlių žydėjimo trukmė.
2. Masyvo reikšmės rašomos į rezultatų failą lentelė: gėlės numeris, žydėjimo trukmė.
3. Atliekami skaičiavimai:
  - ✔ kiek ilgiausiai dienų žydės gėlės;
  - ✔ kurių rūšių gėlės žydės ilgiausiai (kelios vienodai ilgai žydinčios gėlės).
4. Apskaičiuoti rezultatai rašomi į failą.

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Skaityti()	Pradinių duomenų skaitymas iš failo į masyvą
Spausdinti()	Masyvo reikšmių rašymas į failą lentele
IlgiausiaiZydi()	Ilgiausiai žydinčių gėlių dienų skaičiaus paieška
SpausdintiIlgZydi()	Visų ilgiausiai žydinčių gėlių rūšių paieška ir rašymas į rezultatų failą



### 1 Pradinių duomenų failo kūrimas, konstantų ir kintamųjų aprašymas

- Sukurkite tekstinį failą *Duomenys7.txt* ir į jį įrašykite pateiktus pavyzdyje pradinius duomenis.
- Prieš pagrindinę funkciją `main()` aprašykite konstantas pradinių duomenų ir rezultatų failų vardams bei masyvo dydžiui atmintyje laikyti.
- Pagrindinėje funkcijoje aprašykite sveikųjų skaičių masyvą `P(n)` pradiniais duomenimis – gėlių žydėjimo trukmėms – laikyti.

```
const char CDFv[] = "Duomenys7.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai7.txt"; // rezultatų failo vardas
const int CMax = 50; // masyvo dydis
//-----
// Funkcijų prototipai
//-----
int main ()
{
    int P[CMax]; int n; // gėlių žydėjimo trukmių masyvas
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tik informacinį pranešimą.



### 2 Pradinių duomenų skaitymas iš failo į masyvą `P(n)`

Sukursime funkciją `Skaityti()`, kuri iš duomenų failo į masyvą perskaitytų pradinius duomenis. Tai tokia pat funkcija, kaip ir praeitame darbe, tik pradinių duomenų failas turi būti nurodomas per parametrus. Tuomet funkcija nepriklausys nuo programos kintamųjų ir konstantų.

- Parašykite funkcijos `Skaityti()` prototipą:

```
void Skaityti(const char fv[], int A[], int & n);
```

- Parašykite funkcijos `Skaityti()` tekstą:

```
// Skaito duomenis iš failo fv į masyvą A(n)
void Skaityti(const char fv[], int A[], int & n)
{
    ifstream fd(fv); // atidaromas įvesties srautas
    fd >> n; // perskaitomas masyvo reikšmių kiekis
    for (int i = 0; i < n; i++)
        fd >> A[i]; // perskaitoma i-oji reikšmė
    fd.close(); // uždaromas įvesties srautas
}
```

- Funkcijoje `main()` parašykite kreipinį į funkciją `Skaityti()` ir perskaitytos kintamojo `n` reikšmės rodymo ekrane sakinius:

```
Skaityti(CDfv, P, n);
cout << n << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane matysite puokštę sudarančių gėlių skaičių `n`:

7

Skaičius 7 parodo, kad iš failo buvo perskaitytos 7 reikšmės ir įrašytos į masyvą `P(n)`.

- Iš programos pašalinkite sakinį, kuris parodo kintamojo `n` reikšmę ekrane.



### Masyvo `P(n)` reikšmių (gėlių žydėjimo trukmių) rašymas į rezultatų failą

Sukursime funkciją `Spausdinti()`, kuri masyvo `P(n)` reikšmes surašytų lentele į rezultatų failą. Nuo ankstesniame darbe parašytos funkcijos ji turi skirtis tuo, kad failo vardas nurodomas per parametrus. Tuomet funkcija nepriklausys nuo programos kintamųjų ir konstantų.

- Parašykite funkcijos `Spausdinti()` prototipą:

```
void Spausdinti(const char fv[], int A[], int n);
```

- Parašykite funkcijos `Skaityti()` tekstą:

```
// Surašo masyvo A(n) reikšmes į failą fv lentele
void Spausdinti(const char fv[], int A[], int n)
{
    ofstream fr(fv); // atidaromas išvesties srautas
    fr << " Puokštės gėlės " << endl;
    fr << "-----" << endl;
    fr << " G. Nr. Ž. laikas" << endl;
    fr << "-----" << endl;
    for (int i = 0; i < n; i++)
        fr << setw(4) << i+1 << " " << setw(2) << A[i] << endl;
    fr << "-----" << endl;
    fr.close(); // uždaromas išvesties srautas
}
```

- Funkcijoje `main()` parašykite kreipinį į funkciją `Spausdinti()`:

```
Spausdinti(CRfv, P, n);
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą `Rezultatai7.txt`. Jame pamatysite puokštę sudarančių gėlių sąrašą, pateiktą lentele:

```
Puokštės gėlės
-----
G. Nr.  Ž. laikas
-----
1      12
2      9
3      16
4      4
5      16
6      16
7      8
-----
```

4

## Ilgiausiai žydinčios gėlės žydėjimo trukmės paieška

- Parašykite funkcijos `IlgiausiaiZydi()` prototipą:

```
int IlgiausiaiZydi(int A[], int n);
```

- Parašykite funkcijos `IlgiausiaiZydi()` tekstą:

```
// Randa ir grąžina masyvo A(n) didžiausią reikšmę
int IlgiausiaiZydi(int A[], int n)
{
    int max = A[0];
    for (int i = 1; i < n; i++)
        if (A[i] > max)
            max = A[i];
    return max;
}
```

- Papildykite funkciją `main()` kintamaisiais `max` ir `fr`:

```
int max;           // ilgiausia gėlių žydėjimo trukmė
ofstream fr;      // išvesties srautas
```

- Funkcijoje `main()` atverkite išvesties srautą `fr` papildyti:

```
fr.open(CRfv, ios::app);           // atidaro išvesties srautą papildyti
```

- Parašykite kreipinį į funkciją `IlgiausiaiZydi()`:

```
max = IlgiausiaiZydi(P, n);
```

- Įrašykite ilgiausiai žydinčių gėlių trukmę į rezultatų failą, o po to užverkite išvesties srautą:

```
fr << "Visos gėlės nuvys po " << max << " d." << endl;
fr.close();           // uždaro išvesties srautą
```

- Įrašykite ir įvykdykite programą.

- Atverkite rezultatų failą `Rezultatai7.txt`. Jame turėtumėte matyti papildomą eilutę:

```
Visos gėlės nuvys po 16 d.
```

5

## Visų ilgiausiai žydinčių gėlių rūšių paieška ir rašymas į rezultatų failą

- Parašykite funkcijos `SpausdintiIlgZydi()` prototipą:

```
void SpausdintiIlgZydi(const char fv[], int A[], int n, int sk);
```

- Parašykite funkcijos `SpausdintiIlgZydi()` tekstą:

```
// Surašo masyvo A(n) elementų reikšmių, lygių skaičiui sk, indeksus į failą fv
void SpausdintiIlgZydi(const char fv[], int A[], int n, int sk)
{
    ofstream fr(fv, ios::app);           // atidaro išvesties srautą papildyti
    fr << "Gėlių, kurias reiktų dėti į puokštę, numeriai:" << endl;
    for (int i = 0; i < n; i++)
```

```

    if (A[i] == sk)
        fr << " " << i+1;
    fr << endl;
    fr.close(); // uždaro išvesties srautą
}

```

- Funkcijoje main() parašykite kreipinį į funkciją SpausdintiIlgZydi():

```
SpausdintiIlgZydi(CRfv, P, n, max);
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą Rezultatai7.txt. Jame pamatysite visus sukurtos programos rezultatus:

Puokštės gėlės

G.Nr.	Ž.laikas
1	12
2	9
3	16
4	4
5	16
6	16
7	8

Visos gėlės nuvys po 16 d.  
Gėlių, kurias reiktų dėti į puokštę, numeriai:  
3 5 6

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```

const char CDfv[] = "Duomenys7.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai7.txt"; // rezultatų failo vardas
const int CMax = 50; // masyvo dydis
//-----
void Skaityti(const char fv[], int A[], int & n);
void Spausdinti(const char fv[], int A[], int n);
int IlgiausiaiZydi(int A[], int n);
void SpausdintiIlgZydi(const char fv[], int A[], int n, int sk);
//-----
int main ()
{
    int P[CMax]; int n; // gėlių puokštės žydėjimo trukmės masyvas
    int max; // ilgiausia gėlių žydėjimo trukmė
    ofstream fr; // išvesties srautas

    Skaityti(CDfv, P, n);
    Spausdinti(CRfv, P, n);
    fr.open(CRfv, ios::app); // atidaro išvesties srautą papildyti
    max = IlgiausiaiZydi(P, n);
    fr << "Visos gėlės nuvys po " << max << " d." << endl;
    fr.close(); // uždaro išvesties srautą
    SpausdintiIlgZydi(CRfv, P, n, max);
    return 0;
}

```



## Programos patikrinimas

- Patikrinkite, kaip dirba programa esant tokiems pradinė duomenų rinkiniams:
  - ✓ puokštę sudaro vienos rūšies gėlės;
  - ✓ puokštę sudaro gėlės, kurių žydėjimo trukmės visos vienodos;
  - ✓ puokštę sudaro gėlės, kurių žydėjimo trukmės visos skirtingos.



## Programos papildymas

- Papildykite programą, kad ji apskaičiuotų:
  - ✓ kada gėlių puokštė pradės vysti (trumpiausiai žydinčios gėlės);
  - ✓ puokštėje esančių gėlių, kurios žydi trumpiausiai, rūšių skaičių.
- Į rezultatų failą surašykite rūšis gėlių, kurios žydi trumpiausiai.



## Užduotys

### 1. Arbūzas

Pirkėjas nori turguje nusipirkti vidutinį arbūzą. Visi arbūzai sunumeruoti iš eilės ir yra žinoma kiekvieno jų masė. Parenkite programą, kuri nurodytų reikiamą arbūzą ir jo masę.

Pirmoje pradinė duomenų failo eilutėje įrašytas arbūzų skaičius. Toliau pateikiamos visų arbūzų masės (realieji skaičiai).

Pradiniai duomenys	Rezultatai	Paiškinimai
4 4 3 7 8	1 4.00	Vidutinis arbūzas yra pirmas, nes jo masė yra artimiausia visų arbūzų masių aritmetiniam vidurkiui

### 2. Konteineriai

Transporto įmonė veža krovinius iš Vilniaus į Klaipėdą. Gamintojai prekes krauna į konteinerius. Į vieną konteinerį kraunami tik vieno gamintojo gaminiai. Gali būti ir nepilnų konteinerių. Konkretaus gamintojo prekės į konteinerį kraunamos tol, kol jos pasibaigia arba pripildomas konteineris.

Visų gamintojų gaminiai vienodo dydžio (t. y. į konteinerį telpa vienodas gaminių kiekis).

Parenkite programą, kuri apskaičiuotų, kiek mažiausiai reikia konteinerių visiems gaminiams, ir nustatytų, kiek šie konteineriai bus užpildyti.

Pavyzdžiui, jeigu į konteinerį telpa 6 gaminiai, tai reikės rasti, kiek apskritai reikės konteinerių, keli konteineriai bus užpildyti (6 gaminiai), keliuose bus penki gaminiai, keliuose – keturi, trys, du ir keliuose bus tik vienas gaminy.

Pradiniai duomenys įrašyti tekstiname faile. Pirmoje eilutėje yra du sveikieji skaičiai: gamintojų skaičius  $n$  ( $1 \leq n \leq 1000$ ) ir konteinerio talpa  $g$  ( $1 \leq g \leq 100$ ). Tolesnėse  $n$  eilučių įrašyta, kiek gaminių kiekvienas gamintojas nori nuvežti į Klaipėdą. Tai sveikieji skaičiai nuo 1 iki 1000.

Rezultatai įrašomi į tekstinį failą. Pirmoje eilutėje reikia nurodyti konteinerių, kurių prireiks visiems gaminiams nuvežti, skaičių. Tolesnėse  $g$  eilučių turi būti surašyti konteinerių skaičiai pagal konteineriuose esančių gaminių kiekį mažėjimo tvarka.

Pradiniai duomenys	Rezultatai	Paiškinimai
1 5 26	6 5 0 0 0 1	Yra 1 gamintojas, viename konteineriulyje telpa 5 gaminiai, reikia nuvežti 26 gaminius Tam prireiks 6 konteinerių, iš jų 5 bus visiškai užpildyti (t. y. juose bus po 5 gaminius) ir 1 konteineris bus pustuštis – jame bus tik 1 gaminy



5	4	14	Iš viso reikia 14 konteinerių
15		9	Visiškai užpildyti 9 konteineriai
6		2	Dviejuose konteineriuose bus tik po 2 gaminius
22		2	Dviejuose konteineriuose bus tik po 2 gaminius
3		1	Viename konteineryje bus tik 1 gaminys
1			

(XVIII olimpiada, 2006)

### 3. Gyventojai

Lietuvės gyventojų duomenų registre gyventojų asmens kodo duomenys laikomi tokiu formatu SYYMMDDXXXX. Čia:

S – lytis: 3 (vyras, 1900–1999), 4 (moteris, 1900–1999), 5 (vyras, 2000–), 6 (moteris, 2000–),

YY – gimimo metų paskutiniai du skaitmenys (00, 01, 02, 03, ..., 98, 99),

MM – gimimo mėnesio numeris (01, 02, 03, ..., 11, 12),

DD – gimimo dienos numeris (01, 02, 03, ..., 29, 30, 31),

XXXX – registracijos numeris (0001 ... 9999).

Parenkite programą, kuri rastų vyriausio Lietuvos vyro ir vyriausios moters gimimo datas ir jas parodytų ekrane.

Pirmoje pradinių duomenų failo eilutėje yra asmens kodų kiekis  $n$  ( $1 \leq n \leq 3000$ ). Toliau kiekvienoje eilutėje yra po vieną asmens kodą.

Pradiniai duomenys	Rezultatai
6 36508230023 43210060068 60111300321 40203190010 30501150007 50310241123	Vyriausias vyras gimė: 1905 m. sausio 15 d. Vyriausia moteris gimė: 1902 m. kovo 19 d.
2 33303033333 33303023332	Vyriausias vyras gimė: 1933 m. kovo 2 d. Moterų nėra

### 4. Daržas

Darže vienoje ilgoje lysvėje vienodu atstumu viena nuo kitos buvo pasėtos saulėgražos. Jos užaugo didelės ir gražios. Atėjus garbingam svečiui, daržininkas nutarė padovanoti jam dvi saulėgražų galvas. Tačiau svečias gali rinktis tik tokias dvi galvas, kurių bendra masė didžiausia ir kurios auga greta. Parenkite programą, kuri padėtų pasirinkti dovanojamą porą galvų. Nurodykite rastos poros kiekvienos galvos vietą lysvėje ir masę. Jeigu yra kelios vienodos poros, užtenka pateikti tik vieną. Jeigu tokios poros nėra, tuomet reikia išspausdinti pranešimą *Pora nesurasta*.

Pirmoje pradinių duomenų failo eilutėje įrašytas saulėgražų vietų lysvėje skaičius  $n$  ( $1 \leq n \leq 1000$ ). Toliau keliose eilutėse surašytos saulėgražų galvų masės. Jeigu kurioje nors vietoje saulėgražos nėra (jau padovanojota), tuomet rašomas 0 (nulis).

Pradiniai duomenys	Rezultatai
20 2.5 1 3.1 5 1 4.1 2.2 11 0 0 1 1 5.8 9.4 11 8 2 0 0 0.8	14 9.4 15 11
6 2 0 2 0 0 0 1 0 15	Pora nesurasta



## 1.8. Didžiausios ir mažiausios reikšmės vieta

Atlikdami šį darbą:

- ✓ išmokssite skaityti duomenis iš failo į du skirtingo tipo masyvus;
- ✓ išmokssite dviejų vienodo dydžio masyvų reikšmes rašyti į failą lentelę;
- ✓ pritaikysite didžiausios reikšmės masyve indekso paieškos algoritmą.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną

2.7. Masyvas

Nuorodos į algoritmų žinyną

3.5. Didžiausios reikšmės paieškos algoritmas

### Užduotis

**Ligonis.** Ligonio temperatūra per parą matuojama kas valandą arba kas kelias valandas ir užrašoma į ligonio kortelę. Remdamasis šiais įrašais, gydytojas ligoniui skiria gydymą. Parašykite programą, kuri rastų:

- ✓ kurią valandą temperatūra buvo aukščiausia;
- ✓ kuriomis matavimo valandomis temperatūra buvo artima ( $\pm 0.5$  laipsnio) aukščiausiai temperatūrai.

Pirmoje pradinių duomenų failo eilutėje įrašyta, kiek kartų per parą buvo matuota ligonio temperatūra. Likusiose eilutėse yra po du tarpais atskirti skaičiai: valanda (sveikasis skaičius), kada matuota, ir temperatūra (realusis skaičius).

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Rezultatai
10	Ligonio temperatūra
5 38.5	-----
6 38.1	Valanda    Temperatūra
8 38.6	-----
10 37.2	5            38.5
16 39.5	6            38.1
17 39.7	8            38.6
18 39.7	10           37.2
20 40.1	16           39.5
21 39.5	17           39.7
22 39.8	18           39.7
	20           40.1
	21           39.5
	22           39.8
	-----
	Aukščiausia temperatūra 40.1 buvo 20 val.
	Aukšta temperatūra dar buvo:
	17 val. 39.7
	18 val. 39.7
	20 val. 40.1
	22 val. 39.8

### Algoritmas

Sprendami užduotį, naudosisimės vienu sveikųjų ir vienu realiųjų skaičių masyvu.

Užduotis gali būti sprendžiama taip:

1. Duomenys – valandos ir temperatūros tomis valandomis – skaitomi iš failo į du masyvus.
2. Masyvų elementų reikšmės rašomos į rezultatų failą lentelę: valanda, temperatūra.
3. Atliekami skaičiavimai:
  - ✓ kurią valandą temperatūra buvo aukščiausia;
  - ✓ kuriomis matavimo valandomis temperatūra buvo artima ( $\pm 0.5$  laipsnio) aukščiausiai temperatūrai.
4. Apskaičiuoti rezultatai rašomi į rezultatų failą.

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Skaityti()	Pradinių duomenų skaitymas iš failo į du masyvus
Spausdinti()	Masyvų elementų reikšmių rašymas į failą lentelę
AukščiausiaTemp()	Aukščiausios temperatūros įrašo vietos radimas masyve
SpausdintiAukstaTemp()	Valandų, kada temperatūra buvo artima aukščiausiai, rašymas į failą



### Pradinių duomenų failo kūrimas, konstantų ir kintamųjų aprašymas

- Sukurkite tekstinį failą *Duomenys8.txt* ir į jį įrašykite pateiktus pavyzdys pradinius duomenis.
- Prieš pagrindinę funkciją `main()` aprašykite konstantas pradinių duomenų ir rezultatų failų vardams bei masyvų dydžiui atmintyje laikyti (elementų skaičius abiejuose masyvuose vienodas).
- Pagrindinėje funkcijoje aprašykite du masyvus: sveikųjų skaičių masyvą  $V(n)$  temperatūros matavimo valandoms ir realiųjų skaičių masyvą  $T(n)$  liginio temperatūrai atmintyje laikyti.

```
const char CFv[] = "Duomenys8.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai8.txt"; // rezultatų failo vardas
const int CMax = 24; // masyvų dydis
//-----
// Funkcijų prototipai
//-----
int main ()
{
    int V[CMax]; // temperatūros matavimo valandos
    double T[CMax]; // liginio temperatūros rodmenys
    int n; // temperatūros matavimų skaičius
    return 0;
}
```

- Įrašykite ir įvykdysite programą. Ekrane turėtumėte matyti tik informacinį pranešimą.



### Pradinių duomenų – temperatūros matavimo valandų ir temperatūros rodmenų – skaitymas iš failo į masyvus $V(n)$ ir $T(n)$

Duomenys faile įrašyti poromis, kurios tarpusavyje logiškai susijusios. Todėl skaitant pirmasis skaičius turi būti rašomas į masyvą  $V(n)$ , o antrasis – į masyvą  $T(n)$ . Tuomet abiejų masyvų elementų reikšmės bus logiškai susietos tuo pačiu indeksu: pirmą kartą liginio temperatūra matuota 5 valandą ir buvo 38,5 °C, antrąjį – 6 valandą ir buvo 38,1 °C ir t. t.

- Parašykite funkcijos `Skaityti()` prototipą:

```
void Skaityti(const char fv[], int A[], double B[], int & n);
```

- Parašykite funkcijos `Skaityti()` tekstą:

```
// Skaito duomenis iš failo fv į masyvus A(n) ir B(n)
void Skaityti(const char fv[], int A[], double B[], int & n)
{
    ifstream fd(fv);
    fd >> n;
    for (int i = 0; i < n; i++)
        fd >> A[i] >> B[i];
    fd.close();
}
```

- Funkcijoje `main()` parašykite kreipinį į funkciją `Skaityti()` ir nurodykite perskaitytą kintamojo `n` reikšmę rodyti ekrane:

```
Skaityti(CDfv, V, T, n);
cout << n << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane matysite:

10

Skaicius 10 parodo, kad į masyvus `V(n)` ir `T(n)` iš failo buvo perskaityta po 10 reikšmių.

- Iš programos pašalinkite eilutę, kad kintamojo `n` reikšmė nebūtų rodoma ekrane.

3

### 3 Masvų `V(n)` ir `T(n)` reikšmių rašymas į rezultatų failą lentele

- Sukurkite funkciją `Spausdinti()`, kuri masyvų `V(n)` ir `T(n)` reikšmes rašytų į rezultatų failą lentele.
- Parašykite funkcijos `Spausdinti()` prototipą:

```
void Spausdinti(const char fv[], int A[], double B[], int n);
```

- Parašykite funkcijos `Spausdinti()` tekstą:

```
// Rašo masyvų A(n) ir B(n) reikšmes į failą fv lentele
void Spausdinti(const char fv[], int A[], double B[], int n)
{
    ofstream fr(fv);
    fr << " Ligonio temperatūra " << endl;
    fr << "-----" << endl;
    fr << " Valanda Temperatūra " << endl;
    fr << "-----" << endl;
    for (int i = 0; i < n; i++)
        fr << setw(5) << A[i] << "      "
            << fixed << setw(2) << setprecision(1) << B[i] << endl;
    fr << "-----" << endl;
    fr.close();
}
```

- Funkcijoje `main()` parašykite kreipinį į funkciją `Spausdinti()`:

```
Spausdinti(CRfv, V, T, n);
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą `Rezultatai8.txt`. Jame matysite tokią lentelę:

```
Ligonio temperatūra
-----
Valanda  Temperatūra
-----
    5      38.5
    6      38.1
    8      38.6
   10      37.2
   16      39.5
   17      39.7
   18      39.7
   20      40.1
   21      39.5
   22      39.8
-----
```

4

## Aukščiausios temperatūros reikšmės vietos masyve paieška

- Parašykite funkcijos `AuksciausiaTemp()` prototipą:

```
int AuksciausiaTemp(double B[], int n);
```

- Parašykite funkcijos `AuksciausiaTemp()` tekstą:

```
// Randa masyvo B(n) didžiausios reikšmės indeksą ir jį grąžina
int AuksciausiaTemp(double B[], int n)
{
    double max = B[0];           // didžiausio skaičiaus pradinė reikšmė
    int maxind = 0;             // didžiausio skaičiaus vieta masyve
    for (int i = 1; i < n; i++)
        if (B[i] > max) {       // jeigu rasta didesnė reikšmė, tai
            max = B[i];         // įsimenama ši reikšmė ir
            maxind = i;         // jos vieta masyve
        }
    return maxind;             // grąžinama didžiausios reikšmės vieta masyve
}
```

- Papildykite `main()` funkciją kintamaisiais `indmax` ir `fr`:

```
int indmax;           // aukščiausios temperatūros indeksas
ofstream fr;         // išvesties srautas
```

- Funkcijoje `main()` atverkite išvesties srautą `fr` papildyti:

```
fr.open(CRfv, ios::app); // atidaro išvesties srautą papildyti
```

- Parašykite kreipinį į funkciją `AuksciausiaTemp()`:

```
indmax = AuksciausiaTemp(T, n);
```

- Įrašykite gautą rezultatą, o po to išvesties srautą užverkite:

```
fr << "Aukščiausia temperatūra " << T[indmax]
  << " buvo " << V[indmax] << " val." << endl;
fr.close(); // uždaro išvesties srautą
```

- Įrašykite ir įvykdykite programą.

- Atverkite rezultatų failą *Rezultatai8.txt*. Be anksčiau išvestos lentelės, jame turėtumėte matyti tokią eilutę:

```
Aukščiausia temperatūra 40.1 buvo 20 val.
```

5

## Valandų, kuriomis temperatūra buvo artima ( $\pm 0.5$ laipsnio) aukščiausiai temperatūrai, radimas

- Parašykite funkcijos `SpausdintiAukstaTemp()` prototipą:

```
void SpausdintiAukstaTemp(const char fv[], int A[], double B[], int n,
                          double max);
```

- Parašykite funkcijos SpausdintiAukstaTemp() tekstą:

```
// Pagal masyvo B(n) reikšmes, artimas (+0.5) skaičiui max,
// rašo į failą fv masyvo A(n) reikšmes
void SpausdintiAukstaTemp(const char fv[], int A[], double B[], int n,
                          double max)
{
    ofstream fr(fv, ios::app);
    fr << "Aukšta temperatūra dar buvo:" << endl;
    for (int i = 0; i < n; i++)
        if (fabs(max - B[i]) <= 0.5)
            fr << A[i] << " val. " << B[i] << endl;
    fr.close();
}
```

- Funkcijoje main() užrašykite kreipinį į funkciją SpausdintiAukstaTemp():

```
SpausdintiAukstaTemp(CRfv, V, T, n, T[indmax]);
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą Rezultatai8.txt. Jame turėtumėte matyti visus sukurtos programos rezultatus:

Ligonio temperatūra

-----  
Valanda    Temperatūra  
-----

5	38.5
6	38.1
8	38.6
10	37.2
16	39.5
17	39.7
18	39.7
20	40.1
21	39.5
22	39.8

-----  
Aukščiausia temperatūra 40.1 buvo 20 val.

Aukšta temperatūra dar buvo:

17 val. 39.7  
18 val. 39.7  
20 val. 40.1  
22 val. 39.8

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDfv[] = "Duomenys8.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai8.txt"; // rezultatų failo vardas
const int CMax = 24; // masyvų dydis
//-----
void Skaityti(const char fv[], int A[], double B[], int &n);
void Spausdinti(const char fv[], int A[], double B[], int n);
int AuksciausiaTemp(double B[], int n);
void SpausdintiAukstaTemp(const char fv[], int A[], double B[], int n,
                          double max);
```

```

//-----
int main ()
{
    int V[CMax]; // temperatūros matavimo valandos
    double T[CMax]; // ligonio temperatūros rodmenys
    int n; // temperatūros matavimų skaičius
    int indmax; // aukščiausios temperatūros indeksas
    ofstream fr; // išvesties srautas

    Skaityti(CDfv, V, T, n);
    Spausdinti(CRfv, V, T, n);
    fr.open(CRfv, ios::app); // atidaro išvesties srautą papildyti
    indmax = AuksciausiaTemp(T, n);
    fr << "Aukščiausia temperatūra " << T[indmax]
        << " buvo " << V[indmax] << " val." << endl;
    fr.close(); // uždaro išvesties srautą
    SpausdintiAukstaTemp(CRfv, V, T, n, T[indmax]);
    return 0;
}

```



### Programos patikrinimas

- Patikrinkite, kaip dirba programa, kai:
  - ✓ ligonio temperatūra matuojama vieną kartą per parą;
  - ✓ ligonio temperatūra matuojama visą parą kas valandą;
  - ✓ kiekvieną matavimo valandą temperatūra yra vienoda.



### Programos papildymas

- Papildykite programą, kad ji apskaičiuotų:
  - ✓ kiek kartų temperatūra buvo artima aukščiausiai;
  - ✓ kokia buvo vidutinė ligonio temperatūra per visą matavimų laikotarpį;
  - ✓ kokia buvo vidutinė ligonio temperatūra per nurodytą matavimų laikotarpį.
- Parašykite funkciją, kuri suformuotų du masyvus:  $V_1(m)$  ir  $T_1(m)$ . Į juos surašykite tuos  $V(n)$  ir  $T(n)$  duomenis, kai temperatūra buvo artima aukščiausiai. Išbandykite sukurtą funkciją programoje.



### Užduotys

#### 1. Grybai

Petriukas labai mėgsta grybauti. Kiekvieną kartą, sugrįžęs iš miško, savo dienoraštyje jis užsirašo visų surinktų grybų masę ir rastų baravykų bei raudonviršių skaičių. Parenkite programą ir sužinokite, kuri grybavimo sezono diena buvo derlingiausia, kurią dieną Petriukas surinko daugiausia baravykų ir kurią – daugiausia raudonviršių.

Pirmoje pradinių duomenų failo eilutėje nurodytas grybavimo dienų skaičius  $n$  ( $1 \leq n \leq 100$ ). Kitose  $n$  eilutėse yra po penkis skaičius: mėnuo, diena, parneštų grybų masė, baravykų skaičius ir raudonviršių skaičius.

Pradiniai duomenys	Rezultatai
5 7 4 12.5 5 0 7 15 14.2 0 0 7 16 5.5 8 1 8 13 10.1 15 25 8 25 3.5 0 4	Derlingiausia diena: 7 15 Derlingiausia baravykų diena: 8 13 Derlingiausia raudonviršių diena: 8 13
2 7 4 12.5 5 0 8 25 3.5 0 0	Derlingiausia diena: 7 4 Derlingiausia baravykų diena: 7 4 Derlingiausia raudonviršių diena: nėra

## 2. Žibintai

Parką apšviečia  $n$  ( $1 \leq n \leq 100$ ) žibintų. Kiekvienas jų visą naktį šviečia arba ne. Žinoma, kurie žibintai pirmą naktį švietė, o kurie ne. Kiekvieną kitą naktį žibintų būseną nusakoma tokiomis taisyklėmis:

- ✓ žibintas nešvies, jeigu praeitą naktį abu jo kaimynai švietė;
- ✓ žibintas švies, jeigu praeitą naktį vienas jo kaimynas švietė, o kitas nešvietė;
- ✓ jeigu žibintui negalioja nė viena pirmų dviejų taisyklių, tai jis švies, kai naktis lyginė, ir nešvies, kai naktis nelyginė.

Parenkite programą, kuri nustatytų, kurie žibintai švies ir kurie nešvies, kai praeis  $k$  ( $1 \leq k \leq 100$ ) naktų, ir rastų, kurią naktį švietė daugiausia žibintų ir kurią – mažiausiai. Jeigu yra kelios vienodos naktys, kai švietė daugiausia ar mažiausiai žibintų, tai reikia nurodyti tą naktį, kurios numeris mažesnis.

Pirmoje pradinį duomenų failo eilutėje yra du sveikieji skaičiai: žibintų skaičius  $n$  ir naktų skaičius  $k$ . Antroje eilutėje yra nulių ir vienetų, atskirtų vienu tarpu, seka. Tai kiekvieno žibinto būseną: vienetas reiškia, kad pirmą naktį žibintas švietė, o nulis, kad nešvietė.

Pradiniai duomenys	Rezultatai	Paiškinimas
6 2 0 1 0 1 1 1	1 1 0 1 0 1 Naktis, kai švietė daugiausia: 1 Naktis, kai švietė mažiausiai: 1	0 1 0 1 1 1      Pirmą naktis 1 1 0 1 0 1      Antrą naktis
9 5 0 1 1 1 1 0 1 0 1	0 1 0 1 0 0 0 0 0 Naktis, kai švietė daugiausia: 4 Naktis, kai švietė mažiausiai: 5	0 1 1 1 1 0 1 0 1      Pirmą naktis 1 1 0 0 1 0 1 0 1      Antrą naktis 0 1 1 1 0 0 0 0 0      Trečią naktis 1 1 0 1 1 1 1 1 1      Ketvirtą naktis 0 1 0 1 0 0 0 0 0      Penktą naktis

## 3. Dalelė

Mokslininkas stebi vandens paviršiuje pakibusios kietosios dalelės judėjimą ir kas  $n$  sekundžių užrašo jos koordinatas. Parenkite programą, kuri apskaičiuotų kietosios dalelės mažiausią, vidutinį bei didžiausią greičius ir dalelės nukeliatą atstumą.

Pirmoje pradinį duomenų failo eilutėje įrašytas dalelės koordinatų skaičius  $m$  ( $1 < m \leq 100$ ) ir koordinatų fiksavimo intervalas  $n$  ( $n > 0$ ) sekundėmis. Tolnesnėse eilutėse nurodytos dalelės koordinatės  $x_i$  ir  $y_i$  ( $1 \leq i \leq m$ ) milimetrais ( $x_i$  ir  $y_i$  – realieji skaičiai).

Į rezultatų failą įrašykite kietosios dalelės mažiausią, vidutinį bei didžiausią greičius ir dalelės nukeliatą atstumą.

Pradiniai duomenys	Rezultatai
5 2 1.8 2.7 -0.5 4.3 -1.4 3.2 0.5 1.4 0.7 -0.5	Mažiausias greitis: 0.71 mm/s Vidutinis greitis: 1.09 mm/s Didžiausias greitis: 1.40 mm/s Nukeliatas atstumas: 8.76 mm





## 1.9. Masyvo elementų paieška ir jų šalinimas iš masyvo

Atlikdami šį darbą:

- ✓ sužinosite, kad masyve gali būti laikomi ne tik skaičiai, bet ir simboliai;
- ✓ išmoksite atlikti duomenų paiešką masyve;
- ✓ pritaikysite nurodytos reikšmės šalinimo masyve algoritmą.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną

2.7. Masyvas

Nuorodos į algoritmų žinyną

3.6. Reikšmės šalinimo algoritmas

### Užduotis

Batų parduotuvė. Uždarojoje batų parduotuvėje vyksta likusių batų porų išpardavimas, nuolaidos – iki 75 %. Parduotuvėje apsilankė klasės draugai Mykolas ir Marytė. Mykolas nori nusipirkti 43-io, Marytė – 38-o dydžio batus.

Parašykite programą, kuri:

- ✓ nustatytų, ar parduotuvėje yra nurodyto tipo (vyriški / moteriški) ir dydžio batų;
- ✓ pašalintų iš parduotuvės batų sąrašo nurodyto tipo ir dydžio batus, jeigu jie buvo parduoti.

Pirmoje pradinių duomenų failo eilutėje nurodytas parduotuvėje esančių batų porų skaičius. Kiekvienoje kitoje eilutėje įrašytas batų tipas (simboliais: *v* – vyriški, *m* – moteriški) ir batų poros dydis (sveikasis skaičius), atskirti tarpais.

Pradinių duomenų ir rezultatų failų pavyzdys

Pradiniai duomenys	Rezultatai
11	Batų sąrašas
v 45	
v 44	Tipas Dydis
v 43	-----
m 38	v 45
m 40	v 44
v 43	v 43
m 37	m 38
m 38	m 40
v 40	v 43
m 36	m 37
v 39	m 38
	v 40
	m 36
	v 39
	-----
	Mykolo 43-io dydžio batų indeksas masyve 5
	Marytės 38-o dydžio batų indeksas masyve 6
	Batų sąrašas
	-----
	Tipas Dydis
	-----
	v 45
	v 44
	v 43
	m 38
	m 40
	m 37
	v 40
	m 36
	v 39
	-----

## Algoritmas

Užduotis gali būti sprendžiama taip:

1. Iš failo duomenys – batų tipai ir dydžiai – skaitomi į du masyvus.
2. Masyvų reikšmės rašomos į failą lentele: tipas, dydis.
3. Atliekami veiksmai:
  - ✓ paieška, ar parduotuvėje yra nurodyto tipo ir dydžio batų pora;
  - ✓ jei batų pora yra, tai ji laikoma parduota ir ją reikia pašalinti iš batų sąrašo.
4. Apskaičiuoti rezultatai rašomi į rezultatų failą.

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Skaityti ()	Pradinių duomenų skaitymas iš failo į masyvus
Spausdinti ()	Masyvų reikšmių rašymas į failą lentele
IeskotiBatu ()	Nurodyto tipo ir dydžio batų paieška masyvuose
SalintiBatus ()	Nurodyto tipo ir dydžio batų šalinimas iš masyvų



### 1 Pradinių duomenų failo kūrimas, konstantų ir kintamųjų aprašymas

- Sukurkite tekstinį failą *Duomenys9.txt* ir į jį įrašykite pateiktus pavyzdys pradinius duomenis.
- Prieš pagrindinę funkciją `main()` aprašykite konstantas pradinių duomenų ir rezultatų failų vardams bei masyvų dydžiui atmintyje laikyti.
- Pagrindinėje funkcijoje `main()` aprašykite du masyvus: simbolių masyvą  $T(n)$  batų tipams ir sveikųjų skaičių masyvą  $D(n)$  batų dydžiams atmintyje laikyti.

```
const char CDFv[] = "Duomenys9.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai9.txt"; // rezultatų failo vardas
const int CMax = 100; // masyvo dydis
//-----
// Funkcijų prototipai
//-----
int main()
{
    char T[CMax]; // batų tipai (m – moteriški, v – vyriški)
    int D[CMax]; // batų dydžiai
    int n; // batų skaičius
    return 0;
}
```

- Įrašykite ir įvykdysite programą. Ekране turėtumėte matyti tik informacinį pranešimą.



### 2 Pradinių duomenų skaitymas iš failo į masyvus $T(n)$ ir $D(n)$

Duomenys faile pateikti poromis, kurios tarpusavyje logiškai susijusios. Todėl skaitant šiuos duomenis pirmas duomuo turi būti įrašomas į masyvą  $T(n)$ , o antrasis – į masyvą  $D(n)$ . Abiejų masyvų reikšmės bus logiškai susietos tuo pačiu indeksu.

- Parašykite funkcijos `Skaityti()` prototipą:

```
void Skaityti(const char fv[], char A[], int B[], int &n);
```

- Parašykite funkcijos `Skaityti()` tekstą:

```
// Skaito duomenis iš failo fv į masyvus A(n) ir B(n)
void Skaityti(const char fv[], char A[], int B[], int & n)
{
    ifstream fd(fv);           // atidaromas įvesties srautas
    fd >> n;                   // perskaitomas masyvų reikšmių kiekis
    for (int i = 0; i < n; i++)
        fd >> A[i] >> B[i];
    fd.close();               // uždaromas įvesties srautas
}
```

- Funkcijoje `main()` parašykite kreipinį į funkciją `Skaityti()` ir perskaitytos kintamojo `n` reikšmės rodymo ekrane sakinį:

```
Skaityti(CDfv, T, D, n);
cout << n << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane matysite:

```
11
```

Skaičius 11 parodo, kad į masyvus `T(n)` ir `D(n)` iš failo buvo perskaityta po 11 reikšmių.

- Iš programos pašalinkite eilutę, kad kintamojo `n` reikšmė nebūtų rodoma ekrane.

3

### Rezultatų failo paruošimas (išvalymas)

Norint kiekvieną kartą matyti įvykdytos programos rezultatus, rezultatų failą reikia išvalyti.

- Pagrindinėje funkcijoje aprašykite išvesties srautą `fr`:

```
ofstream fr;           // išvesties srautas
```

- Sukurkite naują rezultatų failą arba išvalykite senus failo `CRfv` rezultatus:

```
fr.open(CRfv); fr.close(); // išvalo (sukuria naują) rezultatų failą
```

4

### Masyvų `T(n)` ir `D(n)` reikšmių rašymas į rezultatų failą lentele

- Sukurkite funkciją `Spausdinti()`, kuri rašytų masyvų `T(n)` ir `D(n)` reikšmes į rezultatų failą lentele.
- Parašykite funkcijos `Spausdinti()` prototipą:

```
void Spausdinti(const char fv[], char A[], int B[], int n);
```

- Parašykite funkcijos `Spausdinti()` tekstą:

```
// Rašo masyvų A(n) ir B(n) reikšmes į failą fv lentele
void Spausdinti(const char fv[], char A[], int B[], int n)
{
    ofstream fr(fv, ios::app);           // atidaro išvesties srautą papildyti
    fr << " Batų sąrašas " << endl;
    fr << "-----" << endl;
    fr << " Tipas Dydis " << endl;
    fr << "-----" << endl;
}
```

```

for (int i = 0; i < n; i++)
    fr << setw(4) << A[i] << "      "
        << setw(2) << B[i] << endl;
fr << "-----" << endl;
fr.close(); // uždaro išvesties srautą
}

```

- Funkcijoje main() parašykite kreipinį į funkciją Spausdinti():

```
Spausdinti(CRFv, T, D, n);
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą Rezultatai9.txt. Jame matysite batų sąrašą:

```

Batų sąrašas
-----
Tipas Dydis
-----
v      45
v      44
v      43
m      38
m      40
v      43
m      37
m      38
v      40
m      36
v      39
-----

```

5

### Nurodyto tipo ir dydžio batų paieška masyvuose T(n) ir D(n)

- Parašykite funkcijos IeskotiBatu() prototipą:

```
int IeskotiBatu(char A[], int B[], int n, char tp, int dd);
```

- Parašykite funkcijos IeskotiBatu() tekstą:

```

// Masyvuose A(n) ir B(n) ieško reikšmių, atitinkamai lygių tp (tipas) ir dd (dydis);
// jeigu reikšmes suranda, grąžina masyvų reikšmių elementų indeksą,
// jeigu tokių reikšmių masyvuose nėra, grąžina -1
int IeskotiBatu(char A[], int B[], int n, char tp, int dd)
{
    int ind = -1;
    for (int i = 0; i < n; i++)
        if ((A[i] == tp) && (B[i] == dd))
            ind = i;
    return ind;
}

```

- Papildykite main() funkciją kintamuoju bind:

```
int bind; // batų paieškos indeksas
```

- Funkcijoje `main()` atverkite išvesties srautą `fr` papildyti, parašykite kreipinį į funkciją `IeskotiBatu()`. Jame nurodykite batų tipą `v` ir dydį `43` (Martyno pageidaujami batai). Patikrinkite, ar tokie batai yra pradinuose duomenyse (masyvuose). Įrašykite gautą rezultatą arba pranešimą, kad pageidaujama batų nėra, ir užverkite išvesties srautą:

```
fr.open(CRfv, ios::app);           // atidaro išvesties srautą papildyti
bind = IeskotiBatu(T, D, n, 'v', 43);
if (bind >= 0)                     // batai surasti
    fr << "Mykolo 43-io dydžio batų indeksas masyve " << bind << endl;
else                                // batų nerasta
    fr << "Parduotuvėje Mykolo 43-io dydžio batų nebuvo" << endl;
fr.close();                         // uždaro išvesties srautą
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą *Rezultatai9.txt*. Jame turėtų būti papildoma eilutė:

```
Mykolo 43-io dydžio batų indeksas masyve 5
```

Atkreipkite dėmesį į tai, kad iš sąrašo esančių vienetų galimų atsakymų nurodoma paskutinė tinkama. Taip yra todėl, kad paieškos ciklas nenutraukiamas radus tinkamą reikšmę, o dirba, kol bus peržiūrėtas visas sąrašas.



## 6 Nupirkto batų poros duomenų šalinimas iš masyvų $T(n)$ ir $D(n)$

- Parašykite funkcijos `SalintiBatus()` prototipą:

```
void SalintiBatus(char A[], int B[], int & n, int ind);
```

- Parašykite funkcijos `SalintiBatus()` tekstą:

```
// Pašalina iš masyvų A(n) ir B(n) elementų, kurių indeksas yra ind, reikšmes
void SalintiBatus(char A[], int B[], int & n, int ind)
{
    for (int i = ind; i < n-1; i++) {
        A[i] = A[i+1];
        B[i] = B[i+1];
    }
    n--;
}
```

- Funkcijos `main()` sakinį `if` papildykite kreipiniu į funkciją `SalintiBatus()`. Kreipinį parašykite toje šakoje, kurioje batai (vieta masyve) buvo surasti:

```
if (bind >= 0) {                    // batai surasti
    fr << "Mykolo 43-io dydžio batų indeksas masyve " << bind << endl;
    SalintiBatus(T, D, n, bind);
}
else                                // batų nerasta
    fr << "Parduotuvėje Mykolo 43-io dydžio batų nebuvo" << endl;
```

- Surašykite masyvų  $T(n)$  ir  $D(n)$  reikšmes į rezultatų failą:

```
Spausdinti(CRfv, T, D, n);
```

- Įrašykite ir įvykdykite programą.

- Atverkite rezultatų failą *Rezultatai9.txt*. Jame turėtumėte matyti dar vieną lentelę. Pasitikrinkite, ar nurodyto indekso batų masyvuose tikrai neliko.

#### Batų sąrašas

Tipas	Dydis
v	45
v	44
v	43
m	38
m	40
m	37
m	38
v	40
m	36
v	39

- Analogiškai atlikite Marytės pageidaujamų 38-o dydžio moteriškų batų paiešką. Suradę pašalinkite iš batų sąrašo. Įvykdykite programą. *Rezultatų faile nebus dar vienos eilutės – Marytės nupirktų batų duomenys bus pašalinti iš masyvų.*

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, programa turėtų būti tokia:

```
const char CDFv[] = "Duomenys9.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai9.txt"; // rezultatų failo vardas
const int CMax = 100; // masyvo dydis
//-----
void Skaityti(const char fv[], char A[], int B[], int & n);
void Spausdinti(const char fv[], char A[], int B[], int n);
int IeskotiBatu(char A[], int B[], int n, char tp, int dd);
void SalintiBatus(char A[], int B[], int & n, int ind);
//-----
int main()
{
    char T[CMax]; // batų tipai (m – moteriški, v – vyriški)
    int D[CMax]; // batų dydžiai
    int n; // batų skaičius
    int bind; // batų paieškos indeksas
    ofstream fr; // išvesties srautas

    Skaityti(CDFv, T, D, n);
    fr.open(CRfv); fr.close(); // išvalo rezultatų failą
    Spausdinti(CRfv, T, D, n);

    fr.open(CRfv, ios::app); // atidaro išvesties srautą
    bind = IeskotiBatu(T, D, n, 'v', 43);
    if (bind >= 0) { // batai surasti
        fr << "Mykolo 43-io dydžio batų indeksas masyve " << bind << endl;
        SalintiBatus(T, D, n, bind);
    }
    else // batų nerasta
        fr << "Parduotuvėje Mykolo 43-io dydžio batų nebuvo" << endl;

    bind = IeskotiBatu(T, D, n, 'm', 38);
    if (bind >= 0) { // batai surasti
        fr << "Marytės 38-o dydžio batų indeksas masyve " << bind << endl;
        SalintiBatus(T, D, n, bind);
    }
}
```

```

else // batų nerasta
fr << "Parduotuvėje Marytės 38-o dydžio batų nebuvo" << endl;
fr.close(); // uždaro išvesties srautą
Spausdinti(CRfv, T, D, n);
return 0;
}

```



### Programos patikrinimas

- Patikrinkite, kaip dirba programa esant tokiems pradinį duomenų rinkiniams:
  - ✓ nebuvo parduota nė viena Martyno ir Marytės pageidaujama batų pora;
  - ✓ parduotuvėje buvo tik viena batų pora ir ji buvo parduota;
  - ✓ parduotuvėje buvo dvi batų poros ir abi jos buvo parduotos.



### Programos papildymas

- Funkcija IeskotiBatu() yra neracionali, nes paieška joje tęsiama, nors masyvuose ieškomi dydžiai ir surasti. Sukurkite kitą paieškos funkciją arba modifikuokite senąją taip, kad paieška būtų nutraukiamą, kai tik ieškomi dydžiai masyvuose aptinkami.
- Kaip pastebėjote, duomenų faile yra pasikartojančių eilučių, pavyzdžiui, v\_43 faile pasikartoja du kartus. Kaip reikėtų pakeisti pradinis duomenis, kad jie faile būtų užrašyti be pasikartojimų, t. y. tik vienoje vietoje? Kaip pakeistų programa, jeigu ji dirbtų su tokiais duomenimis? Pakeiskite pradinis duomenis ir programą.
- Kaip pakeistų programa, jeigu pirkėjai būtų ne du (Mykolas ir Marytė), o daugiau? Sukurkite dialogą pirkėjų pageidaujamiems batams įsigyti.



### Užduotys

#### 1. Prekės

Kindziulis turi  $p$  pinigų ( $1 \leq p \leq 20000$ ). Parduotuvėje prekės sudėliotos į  $n$  lentynų ( $1 \leq n \leq 50$ ). Kiekvienos prekės kaina yra užrašyta ant pakuotės. Kindziulis iš kiekvienos lentynos ima tik vieną kuo brangesnę prekę, už kurią gali sumokėti. Po to pereina prie kitos lentynos. Jeigu nė už vieną tos lentynos prekę Kindziulis negali sumokėti, eina toliau. Parenkite programą, kuri apskaičiuotų, kiek prekių Kindziulis išsirinko ir kiek jam dar liko pinigų.

Pirmoje pradinį duomenų failo eilutėje parašytas turimas pinigų skaičius  $p$  ir lentynų skaičius  $n$ . Tolesnėse eilutėse nurodytas kiekvienos lentynos prekių kiekis ir surašytos jų kainos. Visi skaičiai yra sveikieji.

Pradiniai duomenys	Rezultatai	Paiškinimai
1000 4 3 15 360 8 5 2 1 145 99 300 4 700 600 900 800 2 8 1	3 332	Iš pirmos lentynos perka antrą prekę, kurios kaina 360 Iš antros lentynos perka penktą prekę, kurios kaina 300 Iš trečios lentynos nė vienos prekės negali pirkti, nes pinigų liko tik 340 Iš ketvirtos lentynos perka pirmą prekę, kurios kaina 8
5 3 2 5 6 1 14 3 5 6 8	1 0	Iš pirmos lentynos perka pirmą prekę, kurios kaina 5 Iš antros lentynos nė vienos prekės negali pirkti, nes neliko pinigų Iš trečios lentynos nė vienos prekės negali pirkti, nes neliko pinigų

## 2. Spalvotos kortelės

Yra keturių spalvų kortelės (R – raudonos, G – geltonos, M – mėlynos, J – juodos), ant kurių užrašyti skaičiai nuo 1 iki 13. Pavyzdžiui: M 7 – mėlynos spalvos kortelė, pažymėta skaičiumi 7, R 13 – raudonos spalvos kortelė, pažymėta skaičiumi 13. Visų kortelių yra po du vienetus, t. y. 104 kortelės. Iš atsitiktinai išrinktų kortelių galima sudaryti sekas. Seką gali sudaryti ne mažiau kaip trys vienos spalvos iš eilės einančių didėjančių skaičių kortelės, pavyzdžiui, R 5 R 6 R 7 R 8 R 9 R 10. Parenkite programą, kuri iš nurodytų kortelių sudarytų visas galimas maksimalaus ilgio sekas ir jas parodytų ekrane. Kiekviena kortelė gali būti panaudota tik vieną kartą. Jeigu sekų sudaryti nepavyko, reikia ekrane parodyti pranešimą *Nėra*.

Pirmoje pradinių duomenų failo eilutėje užrašytas kortelių skaičius. Kitose eilutėse išvardytos atsitiktinai išrinktos kortelės po vieną eilutėje.

Pradiniai duomenys	Rezultatai
14	G 1 G 2 G 3
M 9	M 1 M 2 M 3 M 4
G 3	M 9 M 10 M 11
M 1	
M 1	
M 10	
G 2	
M 3	
R 6	
M 2	
G 1	
M 11	
M 4	
J 7	
M 4	

## 3. Varžtai ir veržlės

Vienoje dėžėje yra įvairaus skersmens varžtai. Kitoje dėžėje yra veržlės. Parenkite programą, kuri apskaičiuotų, kokiais varžtais ir kokiomis veržlėmis reikia papildyti dėžes, kad visi varžtai turėtų po veržlę. Jeigu varžtų reikia, pirmoje rezultatų failo eilutėje įrašykite *Reikalingi varžtai*. Kitose eilutėse išvardykite poromis (didėjančiai pagal skersmenį), kiek ir kokio skersmens varžtų reikia. Jeigu varžtų nereikia, pirmoje eilutėje įrašykite *Varžtų nereikia*. Toliau analogiškai įrašykite pranešimus apie veržles (*Reikalingos veržlės*: arba *Veržlių nereikia*).

Pirmoje pradinių duomenų failo eilutėje yra varžtų skaičius  $n$  ( $1 \leq n \leq 100$ ). Antroje eilutėje surašyti varžtų skersmenys (sveikieji skaičiai nuo 2 iki 20). Trečioje eilutėje yra veržlių skaičius  $m$  ( $1 \leq m \leq 100$ ). Ketvirtoje eilutėje yra veržlių skersmenys (sveikieji skaičiai nuo 2 iki 20).

Pradiniai duomenys	Rezultatai	Paaiškinimai
8	Reikalingi varžtai:	3 varžtai, kurių skersmuo 2
8 4 2 8 10 5 4 5	3 2	2 varžtai, kurių skersmuo 4
10	2 4	
2 2 2 5 5 4 4 4 2 4	Reikalingos veržlės:	2 veržlės, kurių skersmuo 8
	2 8	1 veržlė, kurios skersmuo 10
	1 10	

## 4. Kolekcija

Aušra ir Rūta kolekcionuoja lietuviškus pašto ženklus. Pašto ženklų kataloge kiekvienas pašto ženklas turi savo numerį. Tai sveikasis skaičius. Žinoma, kad kolekciijoje yra bent vienas pašto ženklas. Taip pat žinoma, kad kolekciijoje yra ne daugiau kaip 1000 ženklų. Aušra siūlo Rūtai mainytis. Ji gali pasiūlyti tik tuos pašto ženklus, kurių turi daugiau kaip vieną ir kurių neturi Rūta. Negali siūlyti kelių vienodų pašto ženklų. Parenkite programą, kuri Aušrai padėtų atrinkti mainams tinkamus pašto ženklus. Jei Aušra neturi ką pasiūlyti, įrašykite į rezultatų failą pranešimą *Ženklų pasikeisti nėra*.



Pradiniai duomenys surašyti dviejuose failuose. Pirmame failu yra Aušros kolekcijos duomenys, o antrajame – Rūtos. Pirmose failo eilutėse yra kolekcijose esančių pašto ženklų skaičiai, kitose – ženklų kataloginiai numeriai, išdėlioti didėjančiai.

Pradiniai duomenys		Rezultatas
Aušros kolekcija	Rūtos kolekcija	
14	9	2 13
1 1 1 2 2 2 5 6 12 13 13 25 99 648	1 5 6 7 7 14 44 25 127	

### 5. Bakterijos

Vienos rūšies bakterijos išsiskiria tuo, kad didesnės minta mažesnėmis greta jų esančiomis bakterijomis. Ry tą pabunda pirmoji iš eilės bakterija, kuri greta esančias bakterijas valgo tol, kol pažadina didesnę už save ir toji ją suvalgo. Kai bakterija pasisotina (pasiekia dydį  $d$ ), ji pažadina savo kaimynę, bet jos nebesuvalgo ir leidžia pusryčiauti jai. Veiksmai kartojami tol, kol pabunda visos bakterijos.

Pirmoje pradinių duomenų failo eilutėje nurodytas bakterijų skaičius  $n$  ir sočiosios bakterijos dydis  $d$  (sveikasis skaičius). Antroje eilutėje įrašyta  $n$  sveikųjų skaičių – kiekvienos bakterijos dydis.

Pirmoje rezultatų failo eilutėje turi būti įrašyti sveikieji skaičiai – po pusryčių likusių bakterijų dydžiai (gali būti ir truputį persivalgiusių (didesnių už  $d$ ), ir alkanų bakterijų).

Pradiniai duomenys	
10 7 2 1 5 2 7 2 4 3 5 4 2	
Rezultatai	Paiškinimas
8 2 7 9 9 2	Pirmoji bakterija suvalgo antrąją ir tampa pirmąja: 3 5 2 7 2 4 3 5 4 2 Antroji bakterija pabunda, suvalgo pirmąją ir tampa pirmąja: 8 2 7 2 4 3 5 4 2 Pirmoji bakterija soti, ji pažadina antrąją, o antroji – trečiąją. Trečioji bakterija soti, todėl ji pažadina ketvirtąją. Ketvirtoji bakterija alkana, todėl ji pažadina penktąją. Pastaroji suvalgo ketvirtąją ir persikelia į jos vietą: 8 2 7 6 3 5 4 2 Ketvirtoji bakterija dar nesoti, ji suvalgo penktąją ir pažadina jos buvusią kaimynę: 8 2 7 9 5 4 2 Penktoji bakterija suvalgo šeštąją ir pažadina jos kaimynę. Septintoji lieka alkana. 8 2 7 9 9 2

### 6. Kas yra vadas?

Žaidžiantys  $n$  ( $1 \leq n \leq 50$ ) vaikų stovi ratu. Vienas jų pasako skaičių  $k$  ( $2 \leq k < n$ ). Norėdami išsirinkti vadą, vaikai pagal laikrodžio rodyklę skaičiuoja iki  $k$ -ojo vaiko. Tada  $k$ -asis vaikas išeina iš rato, kiti lieka stovėti, o ratas susiglaudžia. Taip skaičiuojama tol, kol lieka vienas vaikas – vadas. Parenkite programą, kuri nurodytų, kas taps vadu, t. y. apskaičiuotų to vaiko numerį  $nr$  rate. Vaikai sunumeruoti pagal laikrodžio rodyklę.

Pradinių duomenų failu nurodytos kintamųjų  $n$  ir  $k$  reikšmės. Rezultatų failu turi būti įrašytas vado numeris  $nr$ .

Pradiniai duomenys	Rezultatai
6 2	5

(XII olimpiada, 2000)



## 1.10. Reikšmių įterpimas į masyvą

Atlikdami šį darbą:

- ✓ sužinosite, kaip vykdoma reikalingų duomenų paieška masyve;
- ✓ išmoksite taikyti nurodytos reikšmės įterpimo į masyvą algoritmą.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną

2.7. Masyvas

Nuorodos į algoritmų žinyną

3.7. Reikšmės įterpimo algoritmas

### Užduotis

Pasikartojantys skaičiai. Sprendžiant įvairius matematinius galvosūkius, dažnai pasitaiko, kad duomenys yra skaičių seka, kurios elementai pasikartoja. Pavyzdžiui, kiekvienas sekos 3 3 3 0 0 11 11 9 13 13 13 7 7 1 1 1 skaičius, išskyrus 9, pasikartoja. Šią seką reikia papildyti nauju skaičiumi  $m$ , tarkime,  $m = 11$ .

Parašykite programą, kuri:

- ✓ nustatytų, ar sekoje yra skaičius  $m$ ;
- ✓ įterptų į seką skaičių  $m$  ir nesuardytų esamos tvarkos: jei toks skaičius sekoje jau yra, tai  $m$  įterptų prieš jį, priešingu atveju – sekos pabaigoje.

Pirmoje failo eilutėje įrašytas sekoje esančių skaičių kiekis. Tolesnėse eilutėse surašyti sekos skaičiai, atskirti tarpais, arba skirtingose eilutėse.

Pradinių duomenų ir rezultatų failų pavyzdys

Pradiniai duomenys	Rezultatai
16	Skaičių seka
3 3 3 0 0	-----
11 11 9	3 3 3
13 13 13	0 0
7 7 1 1 1	11 11
	9
	13 13 13
	7 7
	1 1 1
	-----
	Skaičiaus 11 indeksas masyve 5
	Skaičių seka
	-----
	3 3 3
	0 0
	11 11 11
	9
	13 13 13
	7 7
	1 1 1
	-----

### Algoritmas

Užduotis gali būti sprendžiama taip:

1. Skaičių seka skaitoma iš pradinių duomenų failo į masyvą.
2. Masyvo reikšmės rašomos į rezultatų failą (kiekvienoje eilutėje tik vienodi skaičiai).
3. Atliekami veiksmai:
  - ✓ patikrinama, ar skaičių sekoje yra skaičius  $m$ ;
  - ✓ skaičius  $m$  įterpiamas į skaičių seką jam skirtoje vietoje.
4. Gauti rezultatai rašomi į rezultatų failą.

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Skaityti()	Pradinių duomenų skaitymas iš failo į masyvą
Spausdinti()	Masyvo reikšmių rašymas į failą
IeskotiSkaiciaus()	Skaičiaus paieška masyve
IterptiSkaiciu()	Skaičiaus įterpimas į masyvą



### 1 Pradinių duomenų failo kūrimas, konstantų ir kintamųjų aprašymas

- Sukurkite tekstinį failą *Duomenys10.txt* ir į jį įrašykite pateiktus pavyzdyje pradžinius duomenis.
- Prieš pagrindinę funkciją `main()` aprašykite konstantas pradinių duomenų ir rezultatų failų vardams bei masyvo dydžiui atmintyje laikyti.
- Pagrindinėje funkcijoje aprašykite sveikųjų skaičių masyvą skaičių sekai atmintyje laikyti.

```
const char CDFv[] = "Duomenys10.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai10.txt"; // rezultatų failo vardas
const int CMax = 1000; // masyvo dydis
//-----
// Funkcijų prototipai
//-----
int main()
{
    int S[CMax]; int n; // skaičių sekos masyvas
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tik informacinį pranešimą.



### 2 Pradinių duomenų – skaičių sekos – skaitymas iš failo į masyvą S (n)

- Parašykite funkcijos `Skaityti()` prototipą:

```
void Skaityti(const char fv[], int A[], int &n);
```

- Parašykite funkcijos `Skaityti()` tekstą:

```
// Skaitymo duomenis iš failo fv į masyvą A (n)
void Skaityti(const char fv[], int A[], int &n)
{
    ifstream fd(fv); // atidaromas įvesties srautas
    fd >> n; // perskaitomas masyvo reikšmių kiekis
    for (int i = 0; i < n; i++)
        fd >> A[i];
    fd.close(); // uždaromas įvesties srautas
}
```

- Funkcijoje `main()` parašykite kreipinį į funkciją `Skaityti()` ir perskaitytos kintamojo `n` rodymo išvedimo ekrane sakinius:

```
Skaityti(CDFv, S, n);
cout << n << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane matysite:

16

Skaičius 16 parodo, kad į masyvą  $S(n)$  iš failo buvo perskaityta 16 reikšmių.

- Pašalinkite kintamojo  $n$  reikšmės rodymo ekrane sakinį.

3

### Failo paruošimas (išvalymas) rezultatams įrašyti

Norint kiekvieną kartą matyti įvykdytos programos rezultatus, rezultatų failą reikia išvalyti.

- Pagrindinėje funkcijoje aprašykite išvesties srautą `fr`:

```
ofstream fr; // išvesties srautas
```

- Sukurkite naują rezultatų failą arba išvalykite senus failo `CRfv` rezultatus:

```
fr.open(CRfv); fr.close(); // išvalo (sukuria) rezultatų failą
```

4

### Skaičių sekos masyvą $S(n)$ rašymas į rezultatų failą

- Sukurkite funkciją `Spausdinti()`, kuri surašytų masyvą  $S(n)$  reikšmes į rezultatų failą (kiekvienoje eilutėje tik vienodus skaičius).

- Parašykite funkcijos `Spausdinti()` prototipą:

```
void Spausdinti(const char fv[], int A[], int n);
```

- Parašykite funkcijos `Spausdinti()` tekstą:

```
// Rašo masyvą A(n) reikšmes į failą fv
// Vienoje eilutėje vienodi skaičiai
void Spausdinti(const char fv[], int A[], int n)
{
    ofstream fr(fv, ios::app); // atidaromas išvesties srautas papildyti
    fr << " Skaičių seka " << endl;
    fr << "-----" << endl;
    for (int i = 0; i < n - 1; i++) {
        while ((A[i] == A[i+1]) && (A[i] == A[i+1])) {
            fr << setw(4) << A[i]; // rašomi eilutėje vienodi skaičiai
            i++;
        }
        fr << setw(4) << A[i] << endl;
    }
    fr << "-----" << endl;
    fr.close(); // uždaromas išvesties srautas
}
```

- Funkcijoje `main()` parašykite kreipinį į funkciją `Spausdinti()`:

```
Spausdinti(CRfv, S, n);
```

- Įrašykite ir įvykdykite programą.

- Atverkite rezultatų failą *Rezultatai10.txt*. Jame matysite skaičių seką, suskirstytą į eilutes:

```
Skaičių seka
-----
 3  3  3
 0  0
11 11
 9
13 13 13
 7  7
 1  1  1
-----
```

5

### Skaičiaus $m$ (11) paieška masyve $S(n)$

- Parašykite funkcijos `IeskotiSkaiciaus()` prototipą:

```
int IeskotiSkaiciaus(int A[], int n, int sk);
```

- Parašykite funkcijos `IeskotiSkaiciaus()` tekstą:

```
// Masyve A(n) ieško reikšmės, lygios skaičiui sk
// Jeigu reikšmę suranda, grąžina masyvo elemento indeksą
// Jeigu tokios reikšmės masyve neranda, grąžina -1
int IeskotiSkaiciaus(int A[], int n, int sk)
{
    for (int i = 0; i < n; i++)
        if (A[i] == sk)
            return i;
    return -1;
}
```

- Papildykite funkciją `main()` kintamaisiais  $m$  ir  $mind$ :

```
int m = 11;           // skaičius, kuriuo reikia papildyti skaičių seką
int mind;            // skaičiaus m paieškos indeksas
```

- Funkcijoje `main()` atverkite išvesties srautą `fr` papildyti, parašykite kreipinį į funkciją `IeskotiSkaiciaus()`. Patikrinkite, ar toks skaičius sekoje (masyve  $S(n)$ ) yra. Jei taip, įrašykite gautą rezultatą į failą, jei ne – pranešimą, kad skaičius nebuvo rastas, ir užverkite išvesties srautą:

```
fr.open(CRfV, ios::app);
mind = IeskotiSkaiciaus(S, n, m);
if (mind >= 0) // skaičius m sekoje yra
    fr << "Skaičiaus " << m << " indeksas masyve " << mind << endl;
else // skaičiaus m sekoje nėra
    fr << "Skaičius " << m << " masyve nerastas" << endl;
fr.close();
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą *Rezultatai10.txt*. Jame turėtų būti dar viena eilutė:

```
Skaičiaus 11 indeksas masyve 5
```

6

## Skaičiaus m įterpimas į masyvą S (n)

- Parašykite funkcijos IterptiSkaiciu() prototipą:

```
void IterptiSkaiciu(int A[], int & n, int sk, int ind);
```

- Parašykite funkcijos IterptiSkaiciu() tekstą:

```
// Įterpia skaičių sk į masyvą A(n) prieš elementą, kurio indeksas yra ind
void IterptiSkaiciu(int A[], int & n, int sk, int ind)
{
    for (int i = n - 1; i >= ind; i--)
        A[i+1] = A[i];
    A[ind] = sk;
    n++;
}
```

- Funkcijoje main() abi sąlyginio sakinio if šakas papildykite kreipiniais į funkciją IterptiSkaiciu(): šakoje „Taip“ skaičiaus m įterpimo vieta masyve S(n) yra mind (toks skaičius sekoje jau buvo), šakoje „Ne“ – n (tokio skaičiaus sekoje nebuvo):

```
if (mind >= 0) { // skaičius m sekoje yra
    fr << "Skaičiaus " << m << " indeksas masyve " << mind << endl;
    IterptiSkaiciu(S, n, m, mind);
}
else { // skaičiaus m sekoje nėra
    fr << "Skaičius " << m << " masyve nerastas" << endl;
    IterptiSkaiciu(S, n, m, n);
}
```

- Surašykite masyvo S(n) reikšmes į rezultatų failą:

```
Spausdinti(CRfv, S, n);
```

- Įrašykite ir įvykdykite programą.

- Atverkite rezultatų failą Rezultatai10.txt. Jame turėtumėte matyti skaičių seką, papildytą skaičiumi m (11):

Skaičių seka

```
-----
3 3 3
0 0
11 11 11
9
13 13 13
7 7
1 1 1
-----
```

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDfv[] = "Duomenys10.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai10.txt"; // rezultatų failo vardas
const int CMax = 1000; // masyvo dydis
//-----
```

```

void Skaityti(const char fv[], int A[], int & n);
void Spausdinti(const char fv[], int A[], int n);
int IeskotiSkaiciaus(int A[], int n, int sk);
void IterptiSkaiciu(int A[], int & n, int sk, int ind);
//-----
int main()
{
    int S[CMax]; int n;    // skaičių sekos masyvas
    int m = 11;           // skaičius, kuriuo reikia papildyti skaičių seką
    int mind;             // skaičiaus m paieškos indeksas
    ofstream fr;         // išvesties srautas

    Skaityti(CDfv, S, n);
    fr.open(CRfv); fr.close(); // išvalo rezultatų failą
    Spausdinti(CRfv, S, n);
    fr.open(CRfv, ios::app); // atidaro išvesties srautą papildyti
    mind = IeskotiSkaiciaus(S, n, m);
    if (mind >= 0) { // skaičius m sekoje yra
        fr << "Skaičiaus " << m << " indeksas masyve " << mind << endl;
        IterptiSkaiciu(S, n, m, mind);
    }
    else { // skaičiaus m sekoje nėra
        fr << "Skaičius " << m << " masyve nerastas" << endl;
        IterptiSkaiciu(S, n, m, n);
    }
    fr.close(); // uždaro išvesties srautą
    Spausdinti(CRfv, S, n);
    return 0;
}

```



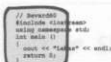
### Programos patikrinimas

- Patikrinkite, kaip dirba programa esant tokiems pradinėms duomenų rinkiniams:
  - ✔ skaičių sekoje yra tik vienas skaičius;
  - ✔ skaičių sekoje nėra skaičių;
  - ✔ skaičių sekoje nėra skaičiaus m;
  - ✔ skaičius m yra skaičių sekos pradžioje;
  - ✔ skaičius m yra skaičių sekos pabaigoje.



### Programos papildymas

- Parenkite funkciją, kuri skaičių seką įrašytų į rezultatų failą lentelė. Joje turėtų būti nurodytas skaičius ir kiek kartų jis sekoje pasikartoja. Išbandykite, kaip dirba programa šiuo atveju.
- Kaip pasikeistų programa, jeigu įterpiamų į seką skaičių būtų ne vienas, o daug? Pakeiskite programą:
  - ✔ sukurkite dialogą įterpiamiems skaičiams įvesti klaviatūra;
  - ✔ įterpiamus skaičius įrašykite į failą, o po to juos įterpkite į skaičių seką;
  - ✔ pašalinkite sekos dalį, kurioje yra skaičiai m.



## Užduotys

### 1. Seka

Sekoje A yra  $n$  ( $n \leq 10$ ) pirminių skaičių, surikiuotų didėjančiai. Sekoje B yra  $m$  ( $m \leq 10$ ) bet kokių sveikųjų teigiamųjų skaičių (skaičiai nesikartoja ir nėra pirminiai). Parenkite programą, kuri visus sekos B skaičius įterptų į pirminių skaičių seką A taip, kad:

- ✓ skaičiai iš sekos B būtų imami iš eilės;
- ✓ skaičius būtų įterpiamas iš karto už tų pirminių skaičių, iš kurių jis dalijasi be liekanos (yra tų skaičių kartotinis);
- ✓ skaičiai, kurie nėra nė vieno duotų pirminių skaičių kartotiniai, į seką A nebūtų įterpiami.

Pirmoje pradinių duomenų failo eilutėje nurodytas sekos A skaičių kiekis, o antroje – šios sekos skaičiai. Trečioje eilutėje įrašytas sekos B skaičių kiekis, o ketvirtoje – šios sekos skaičiai.

Pradiniai duomenys	Rezultatai
3	3 21 9 7 133 14 21 19 133
3 7 19	
5	
4 9 21 14 133	

### 2. Kolekcijos

Dauguma kolekcionuojamų daiktų yra indeksuojami. Pagal indeksus sudaromi katalogai. Kiekvienas kolekcininkas sudaro savo turimų daiktų indeksų sąrašus. Rasa ir Rimas turi kolekcijas, kurių sąrašus sudaro didėjančios dviženkliai ir triženkliai indeksų sekos. Pasitarę jie nusprendė, kad Rasa dovanos Rimui visus daiktus, kurių indeksai yra triženkliai skaičiai, o Rimas Rasai atiduos tuos, kurių indeksai yra dviženkliai skaičiai. Parenkite programą, kuri perkeltų atitinkamus indeksus iš vieno sąrašo į kitą taip, kad sąrašai liktų išdėstyti didėjančiai.

Pradiniai duomenys yra dviejuose failuose, kurių pirmose eilutėse įrašyti kolekcijose esančių daiktų kiekiai, toliau didėjimo tvarka pateikiami daiktų indeksai.

Pradiniai duomenys	Rezultatai
8	Rasos kolekcija:
37 49 58 125 200 545 685 721	37 49 49 52 58 66
5	Rimo kolekcija:
49 52 66 133 200	125 133 200 200 545 685 721

### 3. Monetos

Spauda praneša, kad Estijoje, pakeitus kronas į eurus, labai populiarūs tapo monetų dėklai. Juose kiekvieno nominalo monetai yra numatyta vieta, kurioje telpa tam tikras konkrečios vertės monetų skaičius. Dėklais labai patogu naudotis, nes nereikia ilgai ir nuobodžiai ieškoti reikiamo nominalo monetos. Petriukas labai patenkintas tokiu stebuklingu dėklu, tačiau jis sugaišta daugiau laiko, dėliodamas gautas monetas į joms skirtas vietas. Parenkite programą, kuri gautas monetas „sudėtų“ į dėklą.

Pradiniai duomenys yra dviejuose failuose. Jų pirmose eilutėse nurodyti monetų skaičiai, o antroje – monetų vertės. Pirmame faile išvardytos Petriuko dėklo monetos didėjančiai pagal jų vertę. Antrame faile atsitiktine tvarka surašytos parduotuvėje gautos gražos monetos.

Pradiniai duomenys	Rezultatai
8	1 1 1 2 2 5 5 5 5 10 10 10 20
1 1 2 5 5 5 10 10	
5	
2 1 10 20 5	



#### 4. Krepšininkai

Visi komandos krepšininkai varžybų metu dėvi marškinėlius su skirtingais numeriais, kurie nebūtinai yra nuosekli skaičių seka. Varžybų komentatoriai visuomet gauna žaidžiančiųjų sąrašus, sudarytus didėjimo tvarka pagal jų asmeninius numerius. Varžybų sezono pradžioje komanda pasipildė keliais naujokais, kuriems iš sandėlio išdavė marškinėlius su dar laisvais numeriais. Parenkite programą, kuri papildytų komandos narių sąrašą, skirtą komentatoriams.

Pradiniai duomenys yra dviejuose failuose. Jų pirmose eilutėse nurodyti žaidėjų skaičiai, o antrosios – žaidėjų numeriai. Pirmame failu yra žaidėjų sąrašas, skirtas komentatoriams, antrame failu – naujokų numeriai.

Pradiniai duomenys	Rezultatai
8 1 2 5 6 9 10 12 15	1 2 4 5 6 8 9 10 11 12 15
3 8 11 4	

#### 5. Slėpynės

Vaikai (jų buvo  $n$ ) susirinko kieme ir sutarė pažaisti tradicines slėpynes. Tačiau vos tik sustojo į eilę skaičiuotei, atbėgo dar vienas  $(n + 1)$ -sis vaikas. Vaikai sutiko priimti jį žaisti su sąlyga, kad jis pirmasis liks nežiūrėti ir ieškoti pasislėpusių, tačiau jis privalo dalyvauti skaičiuotėje ir likti paskutinis po skaičiuotės.

Skaičiuotė pradeda nuo pirmo eilėje stovinčio vaiko. Jeigu pasiekiamas eilės galas, tai skaičiuoti pradeda vėl nuo eilės pradžios. Iš eilės išeina vaikas, kuriam tenka skaičiuotės paskutinis žodis. Skaičiuotė iš naujo tęsiama nuo toliau eilėje stovinčio vaiko. Paskutinis likęs vaikas ieškos pasislėpusiųjų.

Parenkite programą, kuri patartų, kurioje eilės vietoje turi atsistoti atbėgęs vaikas, kad po skaičiuotės jis liktų paskutinis.

Pirmoje pradinių duomenų failo eilutėje yra trys sveikieji skaičiai: pradinis vaikų skaičius  $n$  ( $1 < n \leq 25$ ), skaičiuotėje esančių žodžių skaičius  $k$  ( $1 < k \leq 50$ ) ir naujai atbėgusio vaiko numeris  $nr$ .

Antroje eilutėje nurodyti vaikų numeriai tokia tvarka, kokia vaikai sustojo į eilę skaičiuotei prieš atbėgant naujokui. Vaikų numeriai nepasikartoja ir gali būti bet kokie sveikieji teigiamieji skaičiai.

Rezultatų failu turi būti  $n + 1$  skaičių seka – pradinė vaikų numerių seka su reikiamoje vietoje įterptu atbėgusio vaiko numeriu.

Pradiniai duomenys	Rezultatai
5 7 13 12 16 3 2 1	12 16 3 2 13 1
2 4 16 33 44	33 16 44
10 5 235 1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 235 8 9 10

(XVIII olimpiada, 2006)



## 1.11. Simboliai

Atlikdami šį darbą:

- ✓ susipažinsite su simboliu duomenų tipu ir simbolių masyvu;
- ✓ išmoksite skaityti simbolius, juos atpažinti ir palyginti;
- ✓ sužinosite, kaip atpažinti tekste lietuvių kalbos abėcėlės raides.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.4. Duomenų skaitymas iš failo 2.5. Rezultatų (duomenų) rašymas į failą 2.6. Funkcijos 2.7. Masyvas 2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	3.3. Kiekio skaičiavimo algoritmas 3.8. Rikiavimo išrinkimo būdu algoritmas 3.12. Kiti rikiavimo algoritmai

### Užduotis

**Raidžių dažnis tekste.** Kompiuterio klaviatūroje klavišai su simboliais išdėstyti pagal tam tikras taisykles. Dauguma profesionalų dirba visais dešimčia pirštų. Pagrindiniai pirštai yra smiliumi ir didysis. Jiems skirti klavišai, esantys klaviatūros centre. Klaviatūroje simboliai išdėstyti pagal jų statistinį dažnį tam tikros kalbos žodžiuose.

Pradinių duomenų faile yra keleto eilučių tekstas. Parašykite programą, kuri apskaičiuotų raidžių dažnį pateiktame tekste ir į rezultatų failą įrašytų raides, surikiuotas mažėjančiai pagal jų pasikartojimo šiame tekste dažnį.

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys									
Lygumos – tėviškė su žemomis kalvelėmis lyg paminklais, nors nežinia kam, joms tai geriau žinoti, bet nepasako. Aš daug kartų klausiau, ir vėl klausiu, ir kiekvieną kartą jos tyli, šypso tylėdamos ir šnara nesuprantama kalba žolių stiebais, ilgomis saldziomis smilgomis, kai ištrauki ir čiulpi, jų stiebai minkšti ir trapūs, o paskui – jau ne, sumedėję, be skonio, gal tik su žolės skoniu, kurs smagus pažiūrėti, o burnai – jau ne. Šnara ir šnabžda krūmokšniais ir pelkėmis ir retais miškų ežerais, vylingais ir gaivinančiais vandeniomis, gaivinančiais kūną ir kviečiančiais ir atstumiančiais gilybių paslaptimi. Icchokas Meras. Striptizas. Baltos lankos, 2008, 286 p.									
Rezultatai									
i	84	a	64	s	50	n	30	r	28
k	28	l	24	m	23	o	23	t	23
e	22	u	21	p	13	g	12	b	9
v	9	ė	9	š	9	ž	9	y	7
d	6	j	6	č	6	ų	5	ū	4
ą	3	c	2	ę	1	h	1	z	1
į	0	w	0	x	0	q	0	f	0

### Algoritmas

Užduotis gali būti sprendžiama taip:

1. Rastos tekste raidės ir jų pasikartojimo skaičiai laikomi dviejuose masyvuose: simbolių ir skaičių.
2. Apskaičiuojamas kiekvienos raidės, rastos pateiktame tekste, pasikartojimo skaičius.
3. Simbolių masyvas rikiuojamas mažėjančiai, atitinkamai keičiant vietomis ir simbolių pasikartojimo skaičius masyve.
4. Rezultatai rašomi į failą.

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Kiek () Rikiuoti ()	Skaičiavimas, kiek kartų pateiktame tekste pasitaiko parametru nurodytas simbolis Simbolių ir jų pasikartojimo skaičių masyvų rikiavimas

### 1 Pradinių duomenų failo kūrimas, konstantų aprašymas

- Sukurkite tekstinį failą *Duomenys11.txt* ir į jį įrašykite pateiktą pavyzdyje tekstą.
- Prieš pagrindinę funkciją `main()` aprašykite konstantas pradinių duomenų ir rezultatų failų vardams atmintyje laikyti.

```
const char CDfv[] = "Duomenys11.txt"; // pradinių duomenų failo vardas  
const char CRFv[] = "Rezultatai11.txt"; // rezultatų failo vardas
```

### 2 Raidės pasikartojimo nurodytame tekste dažnio skaičiavimas

- Parašykite funkcijos `Kiek()` prototipą:

```
int Kiek(char sim);
```

- Parašykite funkcijos `Kiek()` tekstą:

```
// Apskaičiuoja ir grąžina simbolio sim pasikartojimo pradinių duomenų faile skaičių  
int Kiek(char sim)  
{  
    char s;  
    int k = 0;  
    ifstream fd(CDfv);  
    while (!fd.eof()) {  
        fd.get(s);  
        if (!fd.eof() && (s == sim)) k++;  
    }  
    fd.close();  
    return k;  
}
```

- Pagrindinėje funkcijoje `main()` parašykite kreipinį į funkciją `Kiek()`, kuri apskaičiuotų, kiek kartų pradinių duomenų faile pasitaiko raidė `a`.
- Gautą rezultatą išveskite į rezultatų failą:

```
int main()  
{  
    ofstream fr(CRFv);  
    fr << 'a' << " " << Kiek('a') << endl;  
    fr.close();  
    return 0;  
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tik informacinį pranešimą.
- Atverkite rezultatų failą *Rezultatai11.txt*. Jame matysite, kiek kartų pateiktame tekste pasikartojė raidė `a`:

```
a 63
```



### Didžiųjų raidžių keitimas mažosiomis

Skaičiuojant raidžių pasikartojimo tekste dažnį, reikia sutapatinti didžiąsias abėcėlės raides su mažosiomis. C++ kalbos funkcija `tolower()` nurodytą raidę keičia mažąja, funkcija `toupper()` – didžiąja. Šios funkcijos tinka tik lotynų abėcėlės raidėms.

- Kreipinyje į funkciją `Kiek()` nurodoma mažoji raidė, todėl funkcijoje simbolių palyginimo sakinį pakeiskite tokiu:

```
if (!fd.eof() && (tolower(s) == sim)) k++;
```

- Įrašykite ir įvykdykite programą. Rezultatų faile *Rezultatai11.txt* matysite:

```
a 64
```



### Lietuvių kalbos abėcėlės raidžių pasikartojimo dažnio skaičiavimas

- Norint apskaičiuoti lietuvių kalbos abėcėlės raidžių su diakritiniais ženklais pasikartojimo dažnį pateiktame tekste, reikia parašyti atitinkamą kreipinį į funkciją `Kiek()`, pavyzdžiui,

```
fr << 'a' << " " << Kiek('a') << endl;
```

- Įrašykite ir įvykdykite programą. Rezultatų faile *Rezultatai11.txt* matysite:

```
a 3
```



### Raidžių kodų rašymas į failą

Veiksmai su simboliais keičiami į veiksmus su jų kodais (sveikaisiais skaičiais). Simbolių (ženklų) kodų lentelėje visos raidės nuo 'a' iki 'z' yra surašytos iš eilės.

- Pagrindinėje funkcijoje raidžių 'a' pasikartojimo dažnio skaičiavimo ir rašymo į failą sakinius pakeiskite tokiu ciklo sakiniu:

```
for (char sim = 'a'; sim <= 'z'; sim++)
    fr << sim << " " << (int) sim << endl;
```

- Įrašykite ir įvykdykite programą. Rezultatų faile *Rezultatai11.txt* turėtumėte matyti raidžių kodus:

```
a 97
b 98
c 99
d 100
e 101
f 102
g 103
h 104
i 105
j 106
k 107
l 108
m 109
n 110
o 111
p 112
q 113
r 114
s 115
t 116
```

```

u 117
v 118
w 119
x 120
y 121
z 122

```

- Lietuvių kalbos abėcėlės raidės su diakritiniais ženklais simbolių lentelėje surašytos ne iš eilės, todėl ieškant kiekvienos tokios raidės veiksmus reikia atlikti atskirai. Papildykite pagrindinę funkciją `main()` tokiais sakiniiais:

```

fr << 'a' << " " << (int) 'a' << endl;
fr << 'č' << " " << (int) 'č' << endl;
fr << 'e' << " " << (int) 'e' << endl;
fr << 'ė' << " " << (int) 'ė' << endl;
fr << 'i' << " " << (int) 'i' << endl;
fr << 'š' << " " << (int) 'š' << endl;
fr << 'ų' << " " << (int) 'ų' << endl;
fr << 'ū' << " " << (int) 'ū' << endl;
fr << 'ž' << " " << (int) 'ž' << endl;

```

- Įrašykite ir įvykdykite programą. Rezultatų failo *Rezultatai11.txt* pabaigoje matysite:

```

a -32
č -24
e -26
ė -21
i -31
š -16
ų -8
ū -5
ž -2

```



6

### Visų raidžių pasikartojimo pateiktame tekste dažnio skaičiavimas

Reikalingus pagal užduoties sąlygą skaičiavimus atlikome, tačiau surikiuoti gautų rezultatų negalime, nes jie nėra įrašyti į kompiuterio atmintį. Aišku, galima iš rezultatų failo tuos duomenis skaityti į simbolių ir skaičių masyvus, surikiuoti ir įrašyti į tą patį rezultatų failą. Galima skaičiavimų rezultatų į failą nerašyti, o įrašyti į simbolių ir skaičių masyvus, surikiuoti ir tuomet juos įrašyti į failą. Pasirinkime antrąjį būdą.

- Papildykite pagrindinę funkciją `main()` konstantos, skirtos masyvų dydžiui atmintyje laikyti, ir masyvų kintamųjų aprašymais.
- Pagrindinės funkcijos `main()` veiksmus pakeiskite, kad rezultatai būtų rašomi į masyvus ir iš jų – į rezultatų failą:

```

const char CDFv[] = "Duomenys11.txt"; // pradinių duomenų failo vardas
const char CRFv[] = "Rezultatai11.txt"; // rezultatų failo vardas
const int CMax = 256; // masyvų dydis
//-----
int Kiek(char sim);
//-----
int main()
{
    char S[CMax]; // raidžių masyvas
    int A[CMax]; // raidžių pasikartojimo skaičiai
    int n = 0; // raidžių kiekis
    for (char sim = 'a'; sim <= 'z'; sim++) {
        S[n] = sim; A[n] = Kiek(sim);
        n++;
    }
}

```

```

S[n] = 'a'; A[n] = Kiek('a'); n++;
S[n] = 'č'; A[n] = Kiek('č'); n++;
S[n] = 'ė'; A[n] = Kiek('ė'); n++;
S[n] = 'è'; A[n] = Kiek('è'); n++;
S[n] = 'ì'; A[n] = Kiek('ì'); n++;
S[n] = 'š'; A[n] = Kiek('š'); n++;
S[n] = 'ų'; A[n] = Kiek('ų'); n++;
S[n] = 'ū'; A[n] = Kiek('ū'); n++;
S[n] = 'ž'; A[n] = Kiek('ž'); n++;
ofstream fr(CRfv);
for (int i = 0; i < n; i++) { // rašomi simboliai po penkis eilutėje
    fr << S[i] << " " << setw(2) << A[i] << " ";
    if ((i + 1) % 5 == 0) fr << endl;
}
fr << endl;
fr.close();
return 0;
}

```

- Įrašykite ir įvykdykite programą.
- Rezultatų faile *Rezultatai11.txt* matysite tuos pačius duomenis, kaip ir po ankstesnio žingsnio:

a	64	b	9	c	2	d	6	e	22
f	0	g	12	h	1	i	84	j	6
k	28	l	24	m	23	n	30	o	23
p	13	q	0	r	28	s	50	t	23
u	21	v	9	w	0	x	0	y	7
z	1	ą	3	č	6	ė	1	è	9
ì	0	š	9	ų	5	ū	4	ž	9



## 7 Rezultatų masyvų rikiavimas

- Parašykite funkcijos `Rikiuoti()` prototipą:

```
void Rikiuoti(char S[], int A[], int n);
```

- Parašykite funkcijos `Rikiuoti()` tekstą:

```

// Simbolių masyvas rikiuojamas mažėjančiai pagal simbolių pasikartojimo skaičių
// Kartu rikiuojamas ir simbolių pasikartojimo skaičių masyvas
void Rikiuoti(char S[], int A[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = i + 1; j < n; j++)
            if (A[j] > A[i]) {
                int sk = A[i]; A[i] = A[j]; A[j] = sk;
                char sim = S[i]; S[i] = S[j]; S[j] = sim;
            }
}

```

- Pagrindinėje funkcijoje `main()` parašykite kreipinį į funkciją `Rikiuoti()`, kuri išrikiuotų rezultatų masyvų duomenis mažėjančiai.
- Įrašykite ir įvykdykite programą.

- Rezultatų faile *Rezultatai11.txt* matysite tuos pačius duomenis, kaip ir po ankstesnio žingsnio, tik surikiuotus mažėjančiai pagal raidžių pasikartojimo pateiktame tekste dažnį.

i	84	a	64	s	50	n	30	r	28
k	28	l	24	m	23	o	23	t	23
e	22	u	21	p	13	g	12	b	9
v	9	ė	9	š	9	ž	9	y	7
d	6	j	6	č	6	ų	5	ū	4
a	3	c	2	ę	1	h	1	z	1
į	0	w	0	x	0	q	0	f	0

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDFv[] = "Duomenys11.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai11.txt"; // rezultatų failo vardas
const int CMax = 256; // masyvų dydis
//-----
int Kiek(char sim);
void Rikiuoti(char S[], int A[], int n);
//-----
int main()
{
    char S[CMax]; // raidžių masyvas
    int A[CMax]; // raidžių pasikartojimo skaičiai
    int n = 0; // raidžių kiekis
    for (char sim = 'a'; sim <= 'z'; sim++) {
        S[n] = sim; A[n] = Kiek(sim);
        n++;
    }
    S[n] = 'a'; A[n] = Kiek('a'); n++;
    S[n] = 'č'; A[n] = Kiek('č'); n++;
    S[n] = 'ę'; A[n] = Kiek('ę'); n++;
    S[n] = 'ė'; A[n] = Kiek('ė'); n++;
    S[n] = 'į'; A[n] = Kiek('į'); n++;
    S[n] = 'š'; A[n] = Kiek('š'); n++;
    S[n] = 'ų'; A[n] = Kiek('ų'); n++;
    S[n] = 'ū'; A[n] = Kiek('ū'); n++;
    S[n] = 'ž'; A[n] = Kiek('ž'); n++;
    Rikiuoti(S, A, n);
    ofstream fr(CRfv);
    for (int i = 0; i < n; i++) {
        fr << S[i] << " " << setw(2) << A[i] << " ";
        if ((i + 1) % 5 == 0) fr << endl;
    }
    fr << endl;
    fr.close();
    return 0;
}
```



### Programos patikrinimas

- Programoje skaitomos iš pradinių duomenų failo raidės buvo keičiamos mažosiomis ir skaičiamai buvo atliekami su mažosiomis raidėmis. Nurodykite, kad programa skaičiuotų didžiąsias raides: funkcijos `Kiek()` sakinyje

```
if (!fd.eof() && (tolower(s) == sim)) k++;
```

`tolower(s)` pakeiskite `touper(s)`.

Programos rezultatai turėtų būti tie patys.

- Programa, parašyta C++ programavimo kalba, yra laikoma tekstiniaame faile. Pakeiskite pradinių duomenų failo vardą programos failo vardu. Įsitinkinkite, kad raidžių dažnį programa skaičiuoja teisingai.



## Programos papildymas

- Programa surašo į failą visas raides. Papildykite programą veiksmis, kad būtų rašomos tik tos raidės, kurios tekste aptinkamos bent vieną kartą.
- Atskiru sąrašu pateikite raides, kurių tekste nėra.
- Funkcija `Kiek()` skaito pradinių duomenų failą daug kartų. Pertvarkykite programą taip, kad skaitomi iš pradinių duomenų failo simboliai būtų įrašomi į simbolių masyvą, o funkcija `Kiek()` skaičiuotų, kiek kartų kiekvienas simbolis pasikartoja masyve.
- Pagrindinėje funkcijoje rezultatų rašymo į failą ciklą pakeiskite funkcija ir parašykite kreipinį į ją.
- Apskaičiuokite, kiek kartų pateiktame tekste kartojasi skrybybos ženklai (pvz.: tarpas, taškas, kablelis, brūkšnys).
- Papildykite programą sakiniiais, kad būtų skaičiuojamos pasikartojančios pateiktame tekste didžiosios lietuvių kalbos abėcėlės raidės su diakritiniais ženklais ir rezultatai būtų rašomi į failą.



## Užduotys

### 1. Skaitmenys

Parenkite programą, kuri apskaičiuotų, kiek ir kokių skaitmenų yra pateiktame tekste.

Pradiniai duomenys
Jūs gerai skaičiuojate? Tikrai? Patikrinsim!.. Siūlau keletą visiškai paprastų, didelio dėmesio nereikalaujančių veiksmų... Pasiruošę? Pradedam... Tuščias autobusas išvažiuoja savo įprastu maršrutu. Pirmoje stotelėje įlipa 15 keleivių. Autobusas važiuoja toliau... Kita stotelė. 10 keleivių įlipa, 12 išlipa. Važiuojam toliau... Kita stotelė: 3 keleiviai įlipa, 5 išlipa. Dar viena stotelė: 11 keleivių įlipa, 1 išlipa. Autobusas važiuoja toliau... Kita stotelė. 5 keleiviai įlipa, niekas neišlipa. Autobusas važiuoja toliau... Kita stotelė: niekas neįlipa, 3 išlipa. Autobusas važiuoja toliau... Dar viena stotelė: niekas neįlipa, 5 keleiviai išlipa. Važiuojam toliau... Kita stotelė: 3 keleiviai įlipa, 2 išlipa. Autobusas vėl pajuda ir važiuoja iki paskutinės stotelės. Sustoja. Ar jūs pasiruošę klausimui? Na ką gi, jei jau tokie protingi...Kiek buvo stotelių?
Rezultatai
0 1 1 6 2 2 3 3 4 0 5 4 6 0 7 0 8 0 9 0



## 2. Dvigarsiai

Parenkite programą, kuri apskaičiuotų, kiek pateiktame tekste yra dvigarsių *au*.

Pradiniai duomenys
- Archimedas, maudydamasis vonioje, sušuko: „Eureka“... - Atsiprašau, - nutraukia Petriukas, - ką reiškia šis žodis? - „Suradau“. Ir kaip manai, Petriuk, ką jis surado? - Tikriausiai muilą!
Rezultatai
5

## 3. Komentarai

Parenkite programą, kuri apskaičiuotų, kiek joje yra komentarų, kuriuos žymi du simboliai `"/"`.

Pradiniai duomenys
<pre>// Dviejų skaičių suma #include &lt;fstream&gt; using namespace std; int main() {     int a, b;      // du sveikieji skaičiai     int s;        // a ir b suma     ifstream fd("Duomenys2.txt");     fd &gt;&gt; a &gt;&gt; b; // perskaitomi du sveikieji skaičiai     fd.close();     s = a + b;    // skaičiuojama a ir b suma     ofstream fr("Rezultatai2.txt");     fr &lt;&lt; s;     fr.close();     return 0; }</pre>
Rezultatai
5

## 4. Teksto ženklai

Parenkite programą, kuri apskaičiuotų, kiek pateiktame tekste yra:

- ✓ raidžių;
- ✓ skaitmenų;
- ✓ tarpo simbolių;
- ✓ kitokių simbolių.

Pradiniai duomenys
Kovo 14 diena vadinama pi diena, nes matematikoje pi reikšmė lygi 3,14 (suapvalinta iki 2 skaitmenų po kablelio). Skaičius pi reiškia apskritimo ilgio ir skersmens santykį. Ypatingai kviečiama švęsti 1 val. 59 min. dėl tolimesnių skaičių po kablelio - 3,14159. Beje, 1879 m. kovo 14 d. gimė ir garsus mokslininkas Albertas Enšteinas.
Rezultatai
Raidžių yra 246 Skaitmenų yra 21 Tarpo simbolių yra 51 Kitokių simbolių yra 15



## 1.12. Simboliai ir skaičiai

Atlikdami šį darbą:

- ✓ pakartosite veiksmus su simboliais ir simbolių masyvu;
- ✓ išmoksite aprašyti masyvą, kai jo elementų reikšmės iš anksto žinomos.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.2. Operatoriai 2.4. Duomenų skaitymas iš failo 2.5. Rezultatų (duomenų) rašymas į failą 2.6. Funkcijos 2.7. Masyvas 2.9. Simbolių eilutė string 2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	3.3. Kiekio skaičiavimo algoritmas

### Užduotis

Šachmatai. Gamykloje yra daug robotų, kurie pagal iš anksto parengtą programą gamina šachmatų figūras. Pagamintos figūros patenka į bendrą konteinerį, kurio duomenys automatiškai fiksuojami faile. Parašykite programą, kuri apskaičiuotų, kiek kokių figūrų yra konteineryje, kiek iš jų galima paruošti žaidimo komplektų ir kiek po to liks dar nepanaudotų figūrų.

Žinoma, kad žaidimo komplektą sudaro 16 pėstininkų, 4 bokštai, 4 žirgai, 4 rikiiai 2 valdovės ir 2 karaliai. Pusė komplektui reikalingų figūrų bus vėliau nudažytos juodai, nes robotai gamina tik baltos spalvos figūras.

Pirmoje pradinių duomenų failo eilutėje įrašytas visų konteineryje esančių figūrų skaičius  $n$  ( $1 \leq n \leq 1000$ ). Kitose eilutėse surašyti bet kokia tvarka figūrų tipai, kurie žymimi tokiais simboliais:  $p$  – pėstininkas,  $b$  – bokštas,  $z$  – žirgas,  $r$  – rikiis,  $v$  – valdovė,  $k$  – karalius. Į failą išveskite turimų figūrų skaičius (nurodykite tipą ir skaičių), po to – kiek galima suformuoti žaidimo komplektų ir kiek kokio tipo figūrų lieka. Figūras vardykite tokia eilės tvarka:  $p, b, z, r, v, k$ .

Pradinių duomenų ir rezultatų failų pavyzdys

Pradiniai duomenys	Rezultatai	Paiškinimai
48	p 24	Figūrų skaičius
p p b b z z r	b 7	
r v k v k z	z 4	Komplektų skaičius Liko figūrų
z k p p r r r	r 7	
p p p p p p p	v 3	
b b b b b r	k 3	
r v p p p p	1	
p p p p p p	p 8	
p p p	b 3	
	z 0	
	r 3	
	v 1	
	k 1	

### Algoritmas

Užduotis gali būti sprendžiama taip:

1. Skaitomi pradiniai duomenys.
2. Rašoma į failą, kiek kokio tipo figūrų yra.
3. Skaičiuojama ir į failą rašoma, kiek galima paruošti žaidimo komplektų.
4. Rašoma į failą, kiek kokio tipo figūrų liko.

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
<code>void Duomenys()</code> <code>int Kiek()</code> <code>void Spausdinti()</code> <code>int Komplektai()</code> <code>void Liko()</code>	Pradinių duomenų skaitymas iš failo Skaičiavimas, kiek kartų pateiktame faile pasitaiko nurodytas simbolis Kiekvieno tipo figūrų skaičiaus išvedimas Skaičiavimas, kiek žaidimo komplektų galima sudaryti Skaičiavimas, kiek kokių figūrų liks, kai bus pagaminti žaidimo komplektai



### Pradinių duomenų failo kūrimas

- Sukurkite tekstinį failą *Duomenys12.txt* ir į jį įrašykite pateiktus pavyzdys pradinius duomenis.



### Pradinių duomenų skaitymas

Galima aprašyti šešis kintamuosius pradiniam kiekvieno tipo figūrų kiekiui atmintyje laikyti. Po to, naudojantis ankstesnio praktikos darbo funkcija `Kiek()`, galima apskaičiuoti, kiek kartų tekste pasitaiko simboliai `p`, `b`, `z`, `r`, `v`, `k`. Kiti veiksmai būtų atliekami atskirai su kiekvieno kintamojo reikšme. Tai nėra patogiu.

Patogiau vietoj šešių atskirų kintamųjų naudoti masyvą. Reikia susitarti, kurį masyvo elementą kuriai šachmatų figūrai priskirti. Šioje užduotyje yra nurodyta figūrų kiekių išvedimo tvarka. Jeigu masyve tokia pat tvarka bus laikomi duomenys, tuomet masyvo reikšmių nereikės rikiuoti. Sukurkime tris masyvus:

Indeksas	0	1	2	3	4	5	Paskirtis
Masyvas							
A	<code>p</code>	<code>b</code>	<code>z</code>	<code>r</code>	<code>v</code>	<code>k</code>	Figūrų tipų pavadinimai (raidės)
B	16	4	4	4	2	2	Vienam šachmatų komplektui reikalingos figūros
C	24	7	4	7	3	3	Pradiniai duomenys: kiek kokio tipo figūrų yra

Masyvų A ir B reikšmės yra pastovios, todėl jas galima iš anksto įrašyti į šiuos masyvus. Masyvas C skirtas pradiniam duomenims atmintyje laikyti. Sudarius žaidimų komplektus, šiame masyve bus laikomi nepanaudotų figūrų kiekiai. Pradiniam duomenims iš failo skaityti galima pasinaudoti ankstesnio praktikos darbo funkcija `Kiek()`, tik ją reikia truputį pakeisti.

- Prieš pagrindinę funkciją aprašykite konstantas pradinių duomenų ir rezultatų failų vardams atmintyje laikyti.
- Pagrindinėje funkcijoje aprašykite masyvus A, B ir C.
- Parašykite funkcijų `Duomenys()` ir `Kiek()` prototipus.
- Parašykite duomenų įvedimo funkciją `Duomenys()`.
- Pritaikykite ankstesnio darbo funkciją `Kiek()`.
- Pagrindinėje funkcijoje parašykite kreipinį į funkciją `Duomenys()`.
- Norėdami įsitikinti, kad duomenys įvesti teisingai, pagrindinėje funkcijoje parašykite kontrolinį masyvų A ir C reikšmių rašymo į rezultatų failą sakinį:

```
for (int i = 0; i < 6; i++) // įrašoma figūros tipo raidė ir kiek tokių figūrų yra  
fr << A[i] << ' ' << C[i] << endl;
```

```
const char Cdfv[] = "Duomenys12.txt";  
const char CRfv[] = "Rezultatai12.txt";  
//-----  
void Duomenys(char A[], int C[]);  
int Kiek(char sim);  
//-----
```

```

int main()
{
    char A[6] = {'p', 'b', 'z', 'r', 'v', 'k'};
    int B[6] = {16, 4, 4, 4, 2, 2};
    int C[6];
    Duomenys(A, C);
    ofstream fr(CRfv);
    for (int i = 0; i < 6; i++) // įrašoma figūros tipo raidė ir kiek tokių figūrų yra
        fr << A[i] << ' ' << C[i] << endl;
    fr.close();
    return 0;
}
//-----
// Šachmatų figūrų kiekiai surašomi į masyvą C(6). Masyve A(6) yra laikomi figūrų pavadinimai (raidės)
void Duomenys(char A[], int C[])
{
    for (int i = 0; i < 6; i++)
        C[i] = Kiek(A[i]);
}
//-----
// Apskaičiuoja, kiek kartų nurodytame tekste pasitaiko simbolis sim
int Kiek(char sim)
{
    char s;
    int n, k = 0;
    ifstream fd(CDFv);
    fd >> n;
    for (int i = 0; i < n; i++){
        fd >> s; // skaitant tarpo ir eilutės pabaigos simboliai praleidžiami
        if (s == sim) k++;
    }
    fd.close();
    return k;
}
}

```

- Įrašykite ir įvykdysite programą. Ekrane turėtumėte matyti tik informacinį pranešimą.
- Atverkite rezultatų failą *Rezultatai12.txt*. Jame matysite, kiek kokių figūrų yra:

```

p 24
b 7
z 4
r 7
v 3
k 3

```



### Figūrų sąrašo rašymas

- Parašykite funkcijos `Spausdinti()`, kuri įrašo figūrų sąrašą į failą, prototipą:

```
void Spausdinti(ofstream & fr, char A[], int C[]);
```

- Parašykite funkcijos `Spausdinti()` tekstą:

```

// Įrašo į failą, susietą su srautu fr, šachmatų figūrų pavadinimus ir jų kiekius
// A – figūrų pavadinimai (raidės), C – figūrų kiekiai
void Spausdinti(ofstream & fr, char A[], int C[])
{
    for (int i = 0; i < 6; i++)
        fr << A[i] << ' ' << C[i] << endl;
}

```

- Pagrindinėje funkcijoje main() masyvo rašymo į failą ciklą pakeiskite kreipiniu į funkciją:

```
Spausdinti(fr, A, C);
```

- Įrašykite ir įvykdykite programą. Ekrane matysite tuos pačius rezultatus, kaip ir po ankstesnio žingsnio.



4

### Šachmatų komplektų kiekio skaičiavimas

Yra žinoma, kiek figūrų reikia vienam šachmatų komplektui sudaryti. Tai surašyta masyve B. Taip pat žinoma, kiek kokių figūrų yra pagaminta. Tai nurodyta masyve C. Norint apskaičiuoti, kiek komplektų galima sudaryti, reikia kiekvienos figūros turimą kiekį padalyti iš reikiamo vienam komplektui kiekio. Mažiausias iš gautų šešių skaičių ir bus ieškomų komplektų kiekis.

- Parašykite funkcijos Komplektai() prototipą:

```
int Komplektai(int B[], int C[]);
```

- Parašykite funkcijos Komplektai(), kuri skaičiuotų žaidimo komplektus, tekstą:

```
// Apskaičiuoja ir grąžina, kiek žaidimo komplektų galima sudaryti iš turimų šachmatų figūrų
// B – vieno komplekto figūrų kiekiai, C – turimi figūrų kiekiai
int Komplektai(int B[], int C[])
{
    int i, L = 9999;
    for (int i = 0; i < 6; i++)
        if (C[i] / B[i] < L) L = C[i] / B[i];
    return L;
}
```

- Pagrindinėje funkcijoje parašykite kreipinį į funkciją Komplektai() ir gauto rezultato rašymo į failą sakinį:

```
int k = Komplektai(B, C);
fr << k << endl;
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą Rezultatai12.txt. Jo pabaigoje matysite, kiek komplektų galima sudaryti iš turimų šachmatų figūrų:

```
p 24
b 7
z 4
r 7
v 3
k 3
l
```



5

### Skaičiavimas, kiek liko nepanaudotų figūrų

Norint apskaičiuoti, kiek kokių figūrų liko nepanaudota, reikia iš kiekvienos figūros pradinio kiekio atimti panaudotą visiems žaidimų komplektams šios figūros kiekį.

- Parašykite funkcijos Liko() prototipą:

```
void Liko(int k, int B[], int C[]);
```

- Parašykite funkcijos `Liko()`, skaičiuojančios, kiek kokių šachmatų figūrų liko, tekstą:

```
// Apskaičiuoja, kiek liko nepanaudotų figūrų
// k – komplektų skaičius, B – figūrų kiekiai, reikalingi vienam žaidimo komplektui,
// C – turimi figūrų kiekiai
void Liko(int k, int B[], int C[])
{
    for (int i = 0; i < 6; i++)
        C[i] -= B[i] * k;
}
```

- Pagrindinėje funkcijoje parašykite kreipinį į funkciją ir kreipinį į figūrų sąrašo rašymo funkciją:

```
Liko(k, B, C);
Spausdinti(fr, A, C);
```

- Įrašykite ir įvykdykite programą. Rezultatų faile turėtų būti tokie pat duomenys, kaip ir pateikti pavyzdyje.

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDfv[] = "Duomenys12.txt";
const char CRfv[] = "Rezultatai12.txt";
//-----
void Duomenys(char A[], int B[]);
int Kiek(char sim);
void Spausdinti(ofstream & fr, char A[], int C[]);
int Komplektai(int B[], int C[]);
void Liko(int k, int B[], int C[]);
//-----
int main()
{
    char A[6] = {'p', 'b', 'z', 'r', 'v', 'k'};
    int B[6] = {16, 4, 4, 4, 2, 2};
    int C[6];
    Duomenys(A, C);
    ofstream fr(CRfv);
    Spausdinti(fr, A, C);
    int k = Komplektai(B, C);
    fr << k << endl;
    Liko(k, B, C);
    Spausdinti(fr, A, C);
    fr.close();
    return 0;
}
```



### Programos patikrinimas

- Pirmoje pradinių duomenų failo eilutėje įrašykite skaičių 1 (buvo pagaminta tik viena figūra). Patikrinkite, ar programa pateikia teisingus rezultatus.
- Papildykite duomenų sąrašą didesniu pagamintų figūrų kiekiu, kad būtų galima gauti kelis žaidimų komplektus. Patikrinkite, ar gaunami teisingi rezultatai.



Pirmoje pradinių duomenų failo eilutėje yra sakiny, kurį reikia užšifruoti. Jame žodžiai skiriami vienu tarpo simboliu. Koduojamos tik raidės. Kiti ženklai ir lietuviškos raidės (ąčėįšųžĄČĖĮŠŲŪŽ) nekoduojamos. Eilutės pabaigos simbolis yra '\n'.

Pradiniai duomenys	
RYTO RASA	
Rezultatai	Paiškinimai
•••••	Pakeitus raides ženklais
•••••	Papildomai kiekviename žodyje sukeitus vietomis raides

#### 4. Užšifruotas laiškas

Marytė gavo iš Petriuko šifruotą laišką. Ji žino, kaip laiškas buvo užšifruotas. Parenkite programą, kuri iššifruotų Petriuko Marytei skirtą laišką (laiško šifravimo algoritmas toks pat, kaip ir 3 užduotyje).

Pradiniai duomenys	
•••••	
•••••	
Rezultatai	Paiškinimai
RYTO RASA	Pakeitus raides ženklais
	Papildomai kiekviename žodyje sukeitus vietomis raides

#### 5. Pieštukai

Turime  $n$  ( $1 \leq n \leq 100$ ) spalvotų pieštukų naudojimo istoriją. Pieštukai naudojimo metu buvo drožiami peiliu arba drožtuku. Nudrožtais pieštukais buvo braižomi brėžiniai. Pieštukai galėjo būti ir nulūžę. Naudojimo istorija koduojama raidėmis: D – drožimas drožtuku, P – drožimas peiliu, B – braižymas, L – lūžimas.

Parenkite programą, kuri apskaičiuotų likusį pieštukų ilgį. Skaičiuojant svarbu žinoti, kad:

1. Naudojimo pradžioje visi pieštukai yra nauji, nedrožti ir yra 15 cm ilgio.
2. Drožiant pieštuką peiliu, jis sutrumpėja 10 mm.
3. Drožiant pieštuką drožtuku, jis sutrumpėja 7 mm.
4. Pieštukas gali nulūžti drožiant arba braižant.
5. Jeigu braižant brėžinius pieštukas nenulūžta, tai:
  - a) drožtas peiliu jis sutrumpėja 7 mm;
  - b) drožtas drožtuku jis sutrumpėja 5 mm.
6. Jeigu braižant brėžinius pieštukas nulūžta, tai jis drožiamas peiliu arba drožtuku.

Pirmoje pradinių duomenų failo eilutėje nurodytas pieštukų skaičius  $n$ , kitose  $n$  eilučių yra pieštukų naudojimo istorijos. Eilutės pradžioje nurodyta pieštuko spalva (jai skiriama 20 pozicijų). Toliau eilutėje yra veiksmų, atliekamų su pieštuku, skaičius  $m$  ( $1 \leq m \leq 100$ ) ir patys veiksmai, atskirti tarpo simboliais.

Atskirose rezultatų failo eilutėse turi būti nurodoma kiekvieno pieštuko spalva ir jo ilgis milimetrais naudojimo pabaigoje.

Pradiniai duomenys	Rezultatai
3	Mėlynas 81
Mėlynas 9 P B L D B P L P B	Raudonas 62
Raudonas 15 D B D B D B D L D B D B D B L	Šviesiai žalias 140
Šviesiai žalias 1 P	





## 1.13. Simbolių eilutės

Atlikdami šį darbą:

- ✓ susipažinsite su simbolių eilučių masyvu;
- ✓ išmoksite skaityti simbolių eilutes, kai jas sudaro keli žodžiai;
- ✓ sužinosite, kaip simbolių eilutes galima surikiuoti abėcėliškai.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.1. Operatoriai	3.8. Rikiavimo išrinkimo būdu algoritmas
2.4. Duomenų skaitymas iš failo	3.12. Kiti rikiavimo algoritmai
2.5. Rezultatų (duomenų) rašymas į failą	
2.6. Funkcijos	
2.7. Masyvas	
2.9. Simbolių eilutė <code>string</code>	
2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	

### Užduotis

Lankytinų vietų sąrašas. Sudarytas Lietuvos lankytinų vietų sąrašas. Kiekvienam vietovardžiui faile skirta viena eilutė. Parašykite programą, kuri surikiuotų vietovardžius abėcėliškai.

Pirmoje failo eilutėje įrašytas lankytinų vietų skaičius  $n$  ( $1 \leq n \leq 300$ ). Toliau kiekvienoje eilutėje nurodytas lankytinos vietos pavadinimas.

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Rezultatai
5	-----
Kuršių nerija	Pradiniai duomenys
Trakai	-----
Novaraisčio ornitologinis draustinis	Kuršių nerija
Raigardo slėnis	Trakai
Grūto parkas	Novaraisčio ornitologinis draustinis
	Raigardo slėnis
	Grūto parkas
	-----
	Surikiuoti duomenys
	-----
	Grūto parkas
	Kuršių nerija
	Novaraisčio ornitologinis draustinis
	Raigardo slėnis
	Trakai

### Algoritmas

Užduotis gali būti sprendžiama taip:

1. Pradiniai duomenys skaitomi į simbolių eilučių masyvą.
2. Pradiniai duomenys rašomi į rezultatų failą.
3. Rikiuojamos simbolių eilutės masyve abėcėliškai.
4. Rezultatai rašomi į failą.

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Skaityti()	Pradinių duomenų skaitymas iš failo į simbolių eilučių masyvą
Spausdinti()	Rezultatų rašymas į failą
Rikiuoti()	Simbolių eilučių rikiavimas

### 1 Pradinių duomenų failo kūrimas, konstantų ir kintamųjų aprašymas

- Sukurkite tekstinį failą *Duomenys13.txt* ir į jį įrašykite pateiktus pavyzdyje pradžios duomenis.
- Prieš pagrindinę funkciją `main()` aprašykite konstantas pradinių duomenų ir rezultatų failų vardams bei masyvo dydžiui atmintyje laikyti:

```
const char CDFv[] = "Duomenys13.txt";
const char CRfv[] = "Rezultatai13.txt";
const int CMax = 30;
```

### 2 Funkcijos, skirtos simbolių eilutėms iš failo skaityti į masyvą, rašymas

- Parašykite funkcijos `Skaityti()` prototipą:

```
void Skaityti(string A[], int &n);
```

- Parašykite funkcijos `Skaityti()` tekstą:

```
// Simbolių eilutės skaitomos iš konstanta CDFv nurodyto failo į masyvą A (n)
void Skaityti(string A[], int &n)
{
    ifstream fd(CDFv);
    fd >> n; // perskaitomas eilučių skaičius
    fd.ignore(80, '\n'); // faile pereinama į kitos eilutės pradžią
    for (int i = 0; i < n; i++)
        getline(fd, A[i]); // perskaitomi visi simboliai iki failo eilutės pabaigos ir
                               // pereinama į kitos eilutės pradžią
    fd.close();
}
```

- Papildykite pagrindinę funkciją `main()` simbolių eilučių masyvu ir kreipiniu į funkciją.
- Įsitinkinkite, kad duomenys perskaityti teisingai: pagrindinėje funkcijoje parašykite perskaitytų simbolių eilučių skaičiaus rodymo ekrane sakinį.

```
const char CDFv[] = "Duomenys13.txt";
const char CRfv[] = "Rezultatai13.txt";
const int CMax = 30;
//-----
void Skaityti(string A[], int &n);
//-----
int main()
{
    string A[CMax]; int n;
    Skaityti(A, n);
    cout << "n = " << n << endl; // parodomas ekrane perskaitytų eilučių skaičius
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tokią kintamojo  $n$  reikšmę:

```
n = 5
```

- Iš programos pašalinkite eilutę, kad kintamojo  $n$  reikšmė nebūtų rodoma ekrane.

3

### Pradinių duomenų rašymas į rezultatų failą

- Parašykite funkcijos `Spausdinti()` prototipą:

```
void Spausdinti(string A[], int n, string komentaras);
```

- Papildykite programą funkcijos `Spausdinti()`, skirtos eilučių masyvo  $A(n)$  reikšmėms rašyti į rezultatų failą, tekstu. Skaitant eilučių pabaigos simboliai nebuvo išsaugomi, todėl rašant duomenis, reikia nurodyti `endl`.

```
// Masyve A(n) laikomos eilutės rašomos į failą, nurodytą konstanta CRfv
// komentaras – išvedamų duomenų paaiškinimas
void Spausdinti(string A[], int n, string komentaras)
{
    ofstream fr(CRfv, ios::app); // srautas paruošiamas papildyti
    fr << "-----" << endl;
    fr << komentaras << endl;
    fr << "-----" << endl;
    for (int i = 0; i < n; i++)
        fr << A[i] << endl;
    fr.close();
}
```

- Papildykite pagrindinę funkciją `main()` šiais sakiniais:  
`ofstream fr(CRfv); fr.close();`
- Pagrindinėje funkcijoje `main()` parašykite kreipinį į funkciją `Spausdinti()`:  
`Spausdinti(A, n, "Pradiniai duomenys");`
- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą *Rezultatai13.txt*. Jame turėtumėte matyti:

```
-----
Pradiniai duomenys
-----
Kuršių nerija
Trakai
Novaraisčio ornitologinis draustinis
Raigardo slėnis
Grūdo parkas
```

4

### Simbolių eilučių rikiavimas

- Parašykite funkcijos `Rikiuoti()` prototipą:

```
void Rikiuoti(string A[], int n);
```

- Papildykite programą funkcijos Rikiuoti () tekstu:

```
// Abėcėliškai rikiuoja masyve A(n) laikomas simbolių eilutes
void Rikiuoti(string A[], int n)
{
    string eil;
    for (int i = 0; i < n - 1; i++)
        for (int j = i + 1; j < n; j++)
            // Dvi simbolių eilutes palygina ir, jeigu reikia, sukeičia vietomis
            if (A[j] < A[i]) {
                eil = A[j];
                A[j] = A[i];
                A[i] = eil;
            }
}
```

- Papildykite pagrindinę funkciją kreipiniais į funkciją Rikiuoti () ir Spausdinti ():

```
Rikiuoti(A, n);
Spausdinti(A, n, "Surikiuoti duomenys");
```

- Įrašykite ir įvykdykite programą. Rezultatų faile, be pradinių duomenų, matysite surikiuotų vietovardžių sąrašą.

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDFv[] = "Duomenys13.txt";
const char CRfv[] = "Rezultatai13.txt";
const int CMax = 30;
//-----
void Skaityti(string A[], int &n);
void Spausdinti(string A[], int n, string komentaras);
void Rikiuoti(string A[], int n);
//-----
int main()
{
    ofstream fr(CRfv); fr.close();
    string A[CMax]; int n;
    Skaityti(A, n);
    Spausdinti(A, n, "Pradiniai duomenys");
    Rikiuoti(A, n);
    Spausdinti(A, n, "Surikiuoti duomenys");
    return 0;
}
```



## Programos patikrinimas

- Patikrinkite, kaip dirba programa esant skirtingiems pradinių duomenų rinkiniams, pavyzdžiui, kai:
- ✓ yra tik vienas vietovardis;
  - ✓ vietovardžio pavadinimas yra iš vienos raidės;
  - ✓ yra 30 vietovardžių (ribinis duomenų skaičius);
  - ✓ vietovardžių nedaug, bet visų jų pavadinimai prasideda ta pačia raide; šiuo atveju įsitikinkite, kad eilutės rikiuojamos abėcėliškai.

*Pastaba.* Pradinių duomenų rinkinius galima surašyti į skirtingus failus. Tada, prieš vykdant programą, kaskart pradinių duomenų failo vardą programoje reikia pakeisti nauju. Jeigu norima, kad ir rezultatai

būtų rašomi į skirtingus failus, tai prieš vykdant programą reikia pakeisti rezultatų failo vardo konstantos reikšmę.



## Programos papildymas

- Papildykite programą funkcija, kuri rastų ilgiausią vietovardžio pavadinimą (eilutę). Taikykite didžiausios reikšmės paieškos algoritimą. Sąlyginio sakinio `if` santykio reiškinyje naudokite simbolių eilutės ilgio skaičiavimo funkciją `length()`. Patikrinkite, kaip papildyta programa veikia su visais duomenų rinkiniais.
- Programa turi pateikti rezultatus ir tada, kai failas tuščias, t. y. pirmoje eilutėje įrašytas skaičius 0 (nulis). Ką šiuo atveju matysite rezultatų faile? Papildykite programą reikalingais pranešimais, kurie būtų rašomi į rezultatų failą.



## Užduotys

### 1. Miestai

Parenkite programą, kuri iš miestų sąrašo išrinktų pavadinimus, kuriuos sudaro daugiau kaip vienas žodis. Kiekvienam miesto pavadinimui faile skirta viena eilutė. Žodžiai pavadinime atskirti vienu tarpo simboliu. Prieš miesto pavadinimą ir po jo tarpo simbolių nėra.

Pradiniai duomenys	Rezultatai
5 Druskininkai Kazlų Rūda Pasvalys Kuršėnai Naujoji Akmenė	----- Pradiniai duomenys ----- Druskininkai Kazlų Rūda Pasvalys Kuršėnai Naujoji Akmenė ----- Išrinkti miestai ----- Kazlų Rūda Naujoji Akmenė

### 2. Populiariausia raidė

Parenkite programą, kuri apskaičiuotų, kiek kokių raidžių yra tekste. Tekstą sudaro  $n$  eilučių. Didžiausias ir mažiausias raides skaičiuokite kartu. Kuri raidė dažniausiai pasikartoja nurodytame tekste?

Pradiniai duomenys	Rezultatai
2 Šiandien šviečia saulė. Rytoj tikriausiai lis lietus.	Aa 5 Aa 0 Bb 0 Cc 0 Čč 1 Dd 1 Ee 3 Ee 0 Ėė 1 Ff 0 Gg 0 Hh 0 Ii 10 Įį 0 Yy 1 Jj 1 Kk 1 Ll 3 Mm 0 Nn 2 Oo 1 Pp 0 Rr 2 Ss 4 Šš 2 Tt 3 Uu 3 Uų 0 Ūū 0 Vv 1 Zz 0 Žž 0 Dažniausiai kartojasi raidė Ii.

### 3. Referendumas

Zuikių partija nutarė surengti referendumą dėl lapių ir vilkų teisių apribojimo. Tam reikia surinkti  $n$  miško gyventojų parašų. Pasibaigus parašų rinkimo terminui, visi duomenys buvo perkelti į failą. Pirmoje failo eilutėje įrašytas pasirašiusių skaičius  $n$ . Tolesnėse eilutėse nurodyti kiekvieno pasirašiusiojo duomenys: vardas, pavardė, gyvenamoji vieta. Jie skiriami vienu tarpo simboliu. Parenkite programą, kuri patikrintų, ar pakanka parašų referendumui surengti. Tam reikia suformuoti sąrašą, kuriame nebūtų pasikartojančių (vienodų) eilučių, ir patikrinti, ar naujajame sąrašė yra ne mažiau kaip  $n$  eilučių.

Pradiniai duomenys	Rezultatai
7 Baikštutis Ilgaausis Laukymė3 Drasutis Ilgaausis Pamiškė2 Greitutis Trumpauodegis Laukymė1 Baikštutis Trumpauodegis Laukymė2 Greitutis Ilgaausis Pamiškė1 Drasutis Trumpauodegis Pamiškė3 Greitutis Trumpauodegis Laukymė1	----- Sąrašas ----- Baikštutis Ilgaausis Laukymė3 Drasutis Ilgaausis Pamiškė2 Greitutis Trumpauodegis Laukymė1 Baikštutis Trumpauodegis Laukymė2 Greitutis Ilgaausis Pamiškė1 Drasutis Trumpauodegis Pamiškė3 ----- Išvada ----- Referendumui parašų neužtenka.

#### 4. Mokinių statistika

Parenkite programą, kuri apskaičiuotų, kiek mokykloje yra mergaičių, jeigu žinoma, kad berniukų vardai baigiasi raide s. Pirmoje failo eilutėje nurodytas mokinių skaičius. Tolesnėse eilutėse pateikiamas mokinių sąrašas. Kiekvienoje failo eilutėje įrašyti vieno mokinio pavardė ir vardas, atskirti bent vienu tarpo simboliu.

Pradiniai duomenys	Rezultatai
7 Petraitis Rokas Augė Artūras Mikalauskaitė Aušra Slivka Donatas Stakėnaitė Ieva Skrebė Domas Bruzgaitė Akvilė	3 Mikalauskaitė Aušra Stakėnaitė Ieva Bruzgaitė Akvilė

*Pastaba.* C++ kalboje string tipo eilutes galima analizuoti kaip paprastą masyvą. Pirmasis simbolis yra nulinėje vietoje. Simbolių skaičių eilutėje galima sužinoti naudojantis funkcija `length()`.

#### 5. Kodas

Viename pradinio duomenų faile įrašytas sakiny, kurio pabaigoje yra taškas, o kitame – raidžių kodai. Sakinį sudaro tik žodžiai, atskirti tarpo simboliu. Parenkite programą, kuri užkoduotų ir išspausdintų atskirose eilutėse kiekvieną sakinio žodį.

Pirmoje antroje pradinio duomenų failo eilutėje įrašytas koduojamų raidžių skaičius. Kitose eilutėse abėcėliškai surašytos visos sakinyje esančios raidės ir nurodyti jų kodai, atskirti vienu tarpo simboliu. Kodo ilgis gali būti nuo 1 iki 4 simbolių.

Atskirose rezultatų failo eilutėse turi būti išspausdinti užkoduoti pradinio teksto žodžiai. Kodai vienas nuo kito atskiriami tarpo simboliu.

Pradiniai duomenys	Rezultatai
SVEIKI IR SUDIE.	*** ** * * * * * ** * * _ _ * _ _ ** *** ** _ _ * * * *
8 D _ ** E * I ** K _ * R * _ S *** U ** V *** _	



## 1.14. Pažintis su struktūros duomenų tipu

Atlikdami šį darbą:

- ✓ išmokssite aprašyti struktūros duomenų tipą ir šio tipo kintamuosius;
- ✓ suprasite, kaip skaitomi struktūros duomenų tipo duomenys;
- ✓ pritaikysite sumos skaičiavimo algoritmą;
- ✓ sužinosite, kaip skaičiavimo rezultatai rašomi į masyvą su struktūros tipo duomenimis;
- ✓ išmokssite rašyti rezultatus į failą lentele.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.2. Operatoriai 2.4. Duomenų skaitymas iš failo 2.5. Rezultatų (duomenų) rašymas į failą 2.6. Funkcijos 2.7. Masyvas 2.9. Simbolių eilutė <code>string</code> 2.10. Struktūra 2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	3.1. Sumos skaičiavimo algoritmas

### Užduotis

**Krepšininkai.** Klasė surengė mėtymo į krepšį rungtynes. Per vieną pamoką visi mokiniai metė į krepšį vienodą skaičių kartų iš arti ir toli. Parašykite programą, kuri:

- atrinktų mokinius, kurie surinko ne mažiau kaip 15 taškų;
- apskaičiuotų, kiek vidutiniškai taškų surinko klasės mokinys ir atrinktieji mokiniai.

Pirmoje failo eilutėje yra nurodytas mokinių skaičius  $n$  ( $1 \leq n \leq 30$ ). Toliau kiekvienoje eilutėje yra įrašytas mokinio vardas (pirmos 15 pozicijų) ir jo surinktų taškų skaičius.

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Rezultatai
5 Petras 20 Jurgis 13 Algis 18 Robertas 9 Kasparas 26	Mokinių, surinkusių ne mažiau kaip 15 taškų, sąrašas ----- Vardas Taškai ----- Petras 20 Algis 18 Kasparas 26 ----- Klasės mokinys vidutiniškai surinko taškų: 17 Atrinktieji mokiniai vidutiniškai surinko taškų: 21

### Algoritmas

Užduotis gali būti sprendžiama taip:

1. Skaitomi pradiniai duomenys.
2. Sudaromas atrinktų mokinių sąrašas.
3. Apskaičiuojama, kiek mokiniai iš viso surinko taškų.
4. Apskaičiuojamas mokinio surinktų taškų vidurkis.
5. Apskaičiuojama, kiek iš viso taškų surinko atrinktieji mokiniai, ir rezultatas įrašomas į failą.
6. Apskaičiuojamas ir įrašomas į failą vieno mokinio surinktų taškų vidurkis.

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Skaityti()	Duomenų skaitymas iš failo
Spausdinti()	Rezultatų rašymas į failą
Atrinkti()	Mokinių atrinkimas
Suma()	Mokinių surinktų taškų sumos skaičiavimas



### Struktūros duomenų tipo kūrimas

- Sukurkite duomenų tipą vieno mokinio duomenims aprašyti ir šio tipo kintamąjį:

```
struct Mokinys {
    string pav; // mokinio vardas
    int kiek; // taškų skaičius
};
//-----
int main()
{
    Mokinys A; // vieno mokinio duomenys
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekране turėtumėte matyti tik informacinį pranešimą.



### Pradinių duomenų failo paruošimas, vieno mokinio duomenų skaitymas iš failo

- Tame pačiame kataloge, kur yra programos failas, sukurkite tekstinį pradinių duomenų failą *Duomenys14.txt*. Į jį įrašykite vieno mokinio duomenis:

```
Petras 20
```

Vardą pradėkite rašyti nuo eilutės pirmos pozicijos. Po to palikite bent vieną tarpą ir įrašykite mokinio surinktų taškų skaičių.

- Papildykite programą konstanta pradinių duomenų failo vardui atmintyje laikyti:

```
const char CDfv[] = "Duomenys14.txt";
```

- Parašykite funkcijos `Skaityti()`, skirtos vieno mokinio duomenims iš failo skaityti, prototipą:

```
void Skaityti(Mokinys A);
```

- Parašykite funkcijos `Skaityti()` tekstą:

```
// Iš failo skaitomi mokinio A duomenys
void Skaityti(Mokinys A)
{
    ifstream fd(CDfv);
    fd >> A.pav >> A.kiek;
    fd.close();
}
```

Operatoriumi `>>` skaitoma simbolių eilutė iki tarpo simbolio arba iki failo eilutės pabaigos, jeigu neapninkamas tarpo simbolis. Visi tarpai iki eilutės pradžios praleidžiami.

Operatorius `>>` praleidžia visus tarpus iki skaičiaus ir perskaito skaičių (skaičiaus pabaiga yra tarpo simbolis arba failo eilutės pabaigos simbolis, jeigu nebuvo tarpo simbolio).



- Papildykite pagrindinę funkciją `main()` kreipiniu į funkciją `Skaityti()`. Norėdami įsitikinti, kad duomenys perskaityti teisingai, pagrindinėje funkcijoje parašykite kintamojo reikšmių rodymo ekrane sakinį:

```
//-----
void Skaityti(Mokinys A);
//-----
int main()
{
    Mokinys A;           // vieno mokinio duomenys
    Skaityti(A);
    cout << A.pav << " " << A.kiek << endl; // perskaitytus duomenis parodo ekrane
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti, pavyzdžiui, tokį vaizdą:

```
Petras 20
```



### Visų mokinių, dalyvavusių varžybose, duomenų skaitymas iš failo

- Pradinių duomenų failą *Duomenys14.txt* papildykite kitų mokinių, dalyvavusių varžybose, duomenimis. Pirmoje failo eilutėje nurodykite, kiek mokinių dalyvavo varžybose, pavyzdžiui,

```
5
Petras      20
Jurgis      13
Algis       18
Robertas    9
Kasparas    26
```

Rekomenduojame duomenis pateikti stulpeliais, kad būtų galima patogiau juos apdoroti (taisyti, papildyti) ir išvengti klaidų. Reikia nutarti, koks gali būti ilgiausias vardas. Pavyzdžiui, pasirinkus 15 simbolių, taškų skaičių reikėtų rašyti nuo 17-os pozicijos.

- Struktūros tipo kintamąjį `Mokinys A` pakeiskite tokio pat tipo masyvu `Mokinys A[CMax]`. Aprašykite konstantą masyvo dydžiui atmintyje laikyti (žinoma, kad klasėje yra ne daugiau kaip 30 mokinių).

```
const int CMax = 30;           // didžiausias galimas mokinių skaičius klasėje
Mokinys A[CMax]; int n;       // klasės mokinių duomenys
```

- Papildykite funkciją `Skaityti()` sakiniais visiems duomenims skaityti:

```
// Pradinių duomenų skaitymas iš failo: A – duomenų masyvas, n – duomenų skaičius
void Skaityti(Mokinys A[], int & n)
{
    ifstream fd (CDFv);
    fd >> n;                               // perskaitomas klasės mokinių skaičius
    for (int i = 0; i < n; i++)
        fd >> A[i].pav >> A[i].kiek;
    fd.close();
}
```

- Pakoreguokite funkcijos `Skaityti()` prototipą:

```
void Skaityti(Mokinys A[], int & n);
```

- Papildykite funkciją `main()` kreipiniu į funkciją `Skaityti()` ir šiuo sakiniu:

```
cout << n; // mokinių skaičius klasėje
```

```
const char CRfv[] = "Duomenys14.txt";
const int CMax = 30; // didžiausias galimas mokinių skaičius klasėje
//-----
struct Mokinyas {
    string pav; // mokinio vardas
    int kiek; // taškų skaičius
};
//-----
void Skaityti(Mokinyas A[], int & n);
//-----
int main()
{
    Mokinyas A[CMax]; int n; // klasės mokinių duomenys
    Skaityti(A, n); // skaitomi duomenys
    cout << n; // klasės mokinių skaičius
    return 0;
}
```

- Iš programos pašalinkite eilutę, kad kintamojo `n` reikšmė nebūtų rodoma ekrane.
- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti, pavyzdžiui, tokį vaizdą:

```
5
```



#### Duomenų rašymas į failą

- Papildykite programą funkcija `Spausdinti()`, skirta pradinių duomenų masyvo reikšmėms rašyti į rezultatų failą. `A(n)` – masyvas, kurio duomenys rašomi į failą lentelė. Lentelės pavadinimas yra nuskaitomas simbolių eilutė `eil`.

```
// Mokinių sąrašas rašomas į failą, nurodytą konstanta CRfv
// A – duomenų masyvas, n – duomenų skaičius
void Spausdinti(Mokinyas A[], int n, string eil)
{
    ofstream fr(CRfv);
    fr << eil << endl;
    fr << "-----" << endl;
    fr << "    Vardas          Taškai    " << endl;
    fr << "-----" << endl;
    for (int i = 0; i < n; i++)
        fr << " " << setw(15) << left << A[i].pav << " " <<
            << setw(10) << A[i].kiek << endl;
    fr << "-----" << endl;
    fr.close();
}
```

- Papildykite programą – aprašykite konstantą rezultatų failo vardui atmintyje laikyti:

```
const char CRfv[] = "Rezultatai14.txt";
```

- Parašykite funkcijos `Spausdinti()` prototipą:

```
void Spausdinti(Mokinyas A[], int n, string eil);
```

- Papildykite pagrindinę funkciją `main()` tokiais sakiniais:

```
string eil = "Klasės mokinių sąrašas"; // lentelės pavadinimas
Spausdinti(A, n, eil); // kreipinys į spausdinimo funkciją
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą `Rezultatai14.txt`. Jame turėtumėte matyti:

```
Klasės mokinių sąrašas
-----
      Vardas      Taškai
-----
      Petras      20
      Jurgis      13
      Algis       18
      Robertas    9
      Kasperas    26
-----
```

5

### Mokinių, surinkusių ne mažiau kaip 15 taškų, sąrašo sudarymas

- Parašykite funkcijos `Atrinkti()`, kuri atrinktų mokinius, varžybose surinkusius ne mažiau kaip 15 taškų, prototipą:

```
void Atrinkti(Mokinyas A[], int n, Mokinyas B[], int & m);
```

- Parašykite funkcijos `Atrinkti()` tekstą:

```
// Iš masyvo A(n) atrenkami į masyvą B(m) mokiniai, varžybose surinkę ne mažiau kaip 15 taškų
void Atrinkti(Mokinyas A[], int n, Mokinyas B[], int & m)
{
    m = 0;
    for (int i = 0; i < n; i++)
        if (A[i].kiek >= 15) {
            B[m] = A[i]; // kopijuojami i-ojo mokinio duomenys iš masyvo A į masyvo B pabaigą
            m++; // masyvo B įrašų skaičius padidėjo vienetu
        }
}
```

- Papildykite pagrindinę funkciją naujo masyvo, skirto atrenkamų mokinių sąrašui atmintyje laikyti, aprašymu:

```
Mokinyas B[CMax]; int m;
```

- Pagrindinėje funkcijoje parašykite kreipinį į funkciją `Atrinkti()`:

```
Atrinkti(A, n, B, m);
```

- Įrašykite ir įvykdykite programą.
- Pagrindinėje funkcijoje pradinių duomenų skaitymo sakinius pakeiskite rezultatų rašymo sakiniais:

```
string eil = "Mokinių, surinkusių ne mažiau kaip 15 taškų, sąrašas";
Spausdinti(B, m, eil);
```

- Įrašykite ir įvykdykite programą.

- Atverkite rezultatų failą *Rezultatai14.txt*. Jame turėtumėte matyti atrinktų mokinių sąrašą:

```
Mokinių, surinkusių ne mažiau kaip 15 taškų, sąrašas
-----
Vardas      Taškai
-----
Petras      20
Algis       18
Kasparas    26
-----
```



### 6 Surinktų taškų sumos skaičiavimas

- Parašykite funkcijos `Suma()`, skirtos mokinių surinktų taškų sumai skaičiuoti, prototipą:

```
int Suma(Mokinys A[], int n);
```

- Papildykite programą funkcijos tekstu:

```
// Apskaičiuoja ir grąžina masyve A(n) esančių mokinių surinktų taškų sumą
int Suma(Mokinys A[], int n)
{
    int s = 0;
    for (int i = 0; i < n; i++)
        s += A[i].kiek;
    return s;
}
```

- Papildykite pagrindinę funkciją kreipiniu į funkciją `Suma()` ir sakiniiais, skirtais skaičiavimų rezultatams rašyti į failą:

```
ofstream fr(CRfv, ios::app); // srautas atveriamas papildyti
fr << "Klasės mokinys vidutiniškai surinko taškų: " << Suma(A, n) / n << endl;
fr << "Atrinkti mokiniai vidutiniškai surinko taškų: " << Suma(B, m) / m << endl;
fr.close();
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą *Rezultatai14.txt*. Jame turėtumėte matyti tokį vaizdą:

```
Mokinių, surinkusių ne mažiau kaip 15 taškų, sąrašas
-----
Vardas      Taškai
-----
Petras      20
Algis       18
Kasparas    26
-----
Klasės mokinys vidutiniškai surinko taškų: 17
Atrinkti mokiniai vidutiniškai surinko taškų: 21
```

### Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDFv[] = "Duomenys14.txt";
const char CRfv[] = "Rezultatai14.txt";
const int CMax = 30;
//-----
```

```

struct Mokinys {
    string pav; // mokinio vardas
    int kiek; // taškų skaičius
};
//-----
void Skaityti(Mokinys A[], int & n);
void Spausdinti(Mokinys A[], int n, string eil);
void Atrinkti(Mokinys A[], int n, Mokinys B[], int & m);
int Suma(Mokinys A[], int n);
//-----
int main()
{
    Mokinys A[CMax]; int n; // klasės mokinių duomenys
    Mokinys B[CMax]; int m; // atrinktų mokinių duomenys
    Skaityti(A, n);
    Atrinkti(A, n, B, m);
    string eil = "Mokinių, surinkusių ne mažiau kaip 15 taškų, sąrašas";
    Spausdinti(B, m, eil);
    ofstream fr(CRfv, ios::app); // srautas atveriamas papildyti
    fr << "Klasės mokinyms vidutiniškai surinko taškų: " << Suma(A, n) / n << endl;
    fr << "Atrinkti mokiniai vidutiniškai surinko taškų: " << Suma(B, m) / m << endl;
    fr.close();
    return 0;
}

```



### Programos patikrinimas

- Patikrinkite, kaip dirba programa esant skirtingiems pradinių duomenų rinkiniams, pavyzdžiui, kai į krepšį metė:
  - ✔ tik vienas mokinys ir surinko 15 taškų;
  - ✔ tik vienas mokinys ir surinko 18 taškų;
  - ✔ tik vienas mokinys ir surinko 10 taškų;
  - ✔ keli mokiniai ir visi surinko daugiau kaip po 15 taškų;
  - ✔ keli mokiniai ir nė vienas jų nesurinko 15 taškų (papildykite programą, kad ji įrašytų į failą pranešimą, jog neatrinktas nė vienas mokinys).

*Pastaba.* Pradinių duomenų rinkinius galima surašyti į skirtingus failus. Tada, prieš vykdant programą, kaskart pradinių duomenų failo vardą programoje reikia pakeisti nauju. Jeigu norima, kad ir rezultatai būtų rašomi į skirtingus failus, tai prieš vykdant programą reikia pakeisti rezultatų failo vardo konstantos reikšmę.

- Pamąstykite, kokiais atvejais programa dar nepatikrinta, ir paruoškite atitinkamus duomenų rinkinius.



### Programos papildymas

- Papildykite programą funkcija, kuri rastų taikliausią krepšinininką. Patikrinkite, kaip veikia programa su visais turimais duomenimis.
- Papildykite programą, kad tuo atveju, kai taikliausias krepšinininkas nesurinko 15 taškų, į rezultatų failą būtų įrašomas pranešimas.
- Programa turi pateikti rezultatus ir tada, kai pradinių duomenų failas tuščias, t. y. pirmoje eilutėje nurodytas skaičius 0 (nulis). Ką šiuo atveju matysite rezultatų faile? Papildykite programą reikalingais pranešimais, kurie būtų rašomi į rezultatų failą.



## Užduotys

### 1. Krepšinininkų komanda

Įvyko Lukių miesto mokinių krepšinio pirmenybės, kuriose dalyvavo miesto mokyklų rinktinės. Reikia sudaryti kandidatų į miesto rinktinę sąrašą. Kandidatu gali būti tas mokinys, kurio ūgis yra ne mažesnis kaip  $p$  % už aukščiausią turnyro dalyvį arba kuris per turnyrą įmetė ne mažiau kaip  $k$  % taškų už daugiausia įmetusį turnyro dalyvį.

Pirmoje pradinių duomenų failo eilutėje yra nurodytas dalyvių skaičius. Kitose eilutėse pateikti mokinių duomenys: vardas (skiriama 15 pozicijų), ūgis ir įmestų taškų skaičius. Paskutinėje failo eilutėje yra du skaičiai:  $p$  ir  $k$ . Parenkite programą, kuri sudarytų kandidatų į miesto rinktinę sąrašą.

Pradiniai duomenys			Rezultatai		
5			-----		
Rimas	195.5	45	Vardas	Ūgis	Taškai
Robertas	165	13	-----		
Jurgis	205	36	Rimas	195.5	45
Matas	158	50	Jurgis	205	36
Antanas	145	5	Matas	158	50
10 20			-----		

### 2. Valiuta

Turistų grupės gidas surinko lietuvių keliautojų turimus pinigus (litus ir centus) ir iškeitė juos į lankomos šalies valiutą (bitus ir centus). Parenkite programą, kuri apskaičiuotų:

- ✓ kiek gidas surinko lietuviškų pinigų;
- ✓ kiek bitų ir centų gavo gidas, iškeitęs surinktus pinigus lankomos šalies banke;
- ✓ kiek kuris turistas gavo pinigų nauja valiuta (bitais ir centais, 1 bitą sudaro 100 centų).

Keitimo metu gautos centų dalys atmetamos.

Pirmoje pradinių duomenų failo eilutėje nurodytas turistų skaičius. Toliau atskirose eilutėse pateikti turistų duomenys: vardas (skiriama 15 pozicijų), litai, centai. Paskutinėje eilutėje yra lito kursas nauja valiuta (vieno lito vertė bitais ir centais).

Pradiniai duomenys			Rezultatai		
5			-----		
Rimas	252	45	Gidas surinko:	848	49
Robertas	187	13	Gidas gavo:	8654	59
Jurgis	205	36	-----		
Matas	58	50	Vardas	Turi	Gavo
Antanas	145	5	-----		
10 20			Rimas	252 45	2574 99
			Robertas	187 13	1908 72
			Jurgis	205 36	2094 67
			Matas	58 50	596 70
			Antanas	145 5	1479 51
			-----		

### 3. Mokesčiai

Butų savininkai kas mėnesį gauna sąskaitas už šildymą, telefoną ir sunaudotą vandenį. Vieni žmonės moka tiksliai nurodytas sąskaitose pinigų sumas, kiti sumoka gerokai daugiau, kad kitą mėnesį reikėtų mokėti mažiau arba iš viso nereikėtų mokėti. Sąskaitoje pinigų suma nurodoma teigiamu skaičiumi, jeigu reikia mokėti, ir neigiamu skaičiumi, jeigu dar liko pinigų iš sumokėtų avansų.

Parenkite programą, kuri apskaičiuotų:

- ✓ kiek kuris žmogus turi sumokėti iš viso pinigų už paslaugas;
- ✓ kiek iš viso pinigų gaus kiekviena paslaugas teikianti įmonė.

Pirmoje pradinių duomenų failo eilutėje yra gavusių sąskaitas žmonių skaičius. Toliau atskirose eilutėse pateikti duomenys apie kiekvieną žmogų: vardas (skiriama 15 pozicijų), mokeskis už šilumą, telefoną ir vandenį. Mokesčių dydis nurodytas litais ir centais (realūs skaičiai).

Pradiniai duomenys	Rezultatai
4	Už šilumą turi būti sumokėta: 79.65
Rimas            2.5    13.0    -5.0	Už telefoną turi būti sumokėta: 63.16
Robertas        18.7   -13.95 -25.0	Už vandenį turi būti sumokėta: 14.45
Jurgis           -205.0 -36.0   -0.5	-----
Matas            58.45  50.16  14.45	Vardas                      Turi mokėti
	-----
	Rimas                            15.50
	Robertas                        18.70
	Matas                             123.06
	-----

#### 4. Lėktuvo bagažas

Registruodamiesi skrydžiui keleiviai pildo bagažo saugojimo formas, kuriose nurodo savo pavardę, bagažo vienetų skaičių ir kiekvieno vieneto masę.

Reikia parengti programą, kuri:

- ✓ apskaičiuotų, kiek lėktuve skrenda keleivių, kurie turi ne daugiau kaip  $m$  bagažo vienetų;
- ✓ apskaičiuotų, kelių keleivių turima bagažo masė viršija  $x$  kilogramų 0,1 kilogramo tikslumu;
- ✓ apskaičiuotų, kelių keleivių bagažo masė viršija vidutinę visų skrendančiųjų bagažo masę 1 kilogramo tikslumu;
- ✓ surašytų keleivius, kurių bagažo masė skiriasi nuo didžiausios vieno keleivio bagažo masės  $\pm y$  kilogramų.

Pirmoje pradinių duomenų failo eilutėje nurodytas keleivių skaičius  $n$  ( $n \leq 500$ ). Toliau yra  $n$  eilučių, kurių kiekvienoje įrašyta informacija apie vieną keleivį: pavardė (skiriama 20 pozicijų), bagažo vienetų skaičius  $k$  (natūralusis skaičius) ir  $k$  realiųjų skaičių – keleivio vežamo bagažo kiekvieno vieneto masė kilogramais. Po to nurodyti skaičiai  $m$ ,  $x$  ir  $y$ , atskirti tarpais.

Rezultatų faile turi būti:

- ✓ pirmoje eilutėje nurodytas skaičius keleivių, turinčių ne daugiau kaip  $m$  bagažo vienetų;
- ✓ antroje eilutėje įrašytas skaičius keleivių, kurių kiekvieno bagažo masė viršija  $x$  kilogramų 0,1 kilogramo tikslumu;
- ✓ trečioje eilutėje įrašyta, kelių keleivių bagažo masė viršija vidutinę visų skrendančiųjų bagažo masę 1 kilogramo tikslumu;
- ✓ tolesnėse eilutėse išvardytos pavardės keleivių, kurių bagažo masė nuo didžiausios vieno keleivio bagažo masės skiriasi  $\pm y$  kilogramų.

Pradiniai duomenys	Rezultatai
5	3
Jonaitis            3 2 3 2	2
Petraitis           2 3 2	2
Antanaitis        2 4 3	Petraitis
Kizlaitis           3 2 1 2	Kizlaitis
Pranaitis           2 2 2	
2 6 2	



## 1.15. paieška nesutvarkytame sąrašė

Atlikdami šį darbą:

- ✓ sukursite struktūros duomenų tipą ir masyvą, kurio elementai yra šio tipo;
- ✓ išmoksėte skaityti simbolių eilutes, kai jas sudaro keli žodžiai ir kai jos užima kelias failo eilutes;
- ✓ išmoksėte formuoti naują duomenų sąrašą;
- ✓ susipažinsite su paieškos nesutvarkytame sąrašė algoritmu.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.2. Operatoriai	3.9. Paieškos nerikiuotame masyve algoritmas
2.4. Duomenų skaitymas iš failo	
2.5. Rezultatų (duomenų) rašymas į failą	
2.6. Funkcijos	
2.7. Masyvas	
2.9. Simbolių eilutė string	
2.10. Struktūra	
2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	

### Užduotis

**Miestai.** Sudarytas pasaulio miestų sąrašas, kuriame nurodytas miesto pavadinimas ir valstybė, kurioje miestas yra. Reikia rasti, kiek kurios valstybės miestų yra sąrašė.

Pirmoje pradinių duomenų failo eilutėje yra miestų skaičius  $n$  ( $1 \leq n \leq 300$ ). Toliau kiekvienoje eilutėje pateiktas miesto pavadinimas (jis užima 20 pirmųjų eilutės pozicijų) ir valstybės, kurioje tas miestas yra, pavadinimas.

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Rezultatai
10	Valstybių sąrašas:
Trakai                    Lietuva	Lietuva            2
Madridas                Ispanija	Ispanija           3
Barselona               Ispanija	Italija             3
Roma                     Italija	Prancūzija        1
Paryžius                 Prancūzija	Estija              1
Saragosa                Ispanija	
Genuja                  Italija	
Tartu                     Estija	
Kudirkos Naumiestis   Lietuva	
Venecija                Italija	

### Algoritmas

Užduotis gali būti sprendžiama taip:

1. Skaitymi pradiniai duomenys į masyvą.
2. Formuojamas valstybių sąrašas.
3. Rezultatai rašomi į failą.

### Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Skaityti ()	Pradinių duomenų skaitymas iš failo
Spausdinti ()	Rezultatų rašymas į failą
Atrinkti ()	Sąrašo formavimas
Yra ()	Valstybės paieška





## Pradinių duomenų failo kūrimas, duomenų skaitymas iš failo

- Tame pačiame kataloge, kur yra programos failas, sukurkite tekstinį failą *Duomenys15.txt* ir į jį įrašykite pateiktus pavyzdyje pradinis duomenis.
- Papildykite programą konstanta pradinių duomenų failo vardui atmintyje laikyti:

```
const char CDFv[] = "Duomenys15.txt";
```

- Papildykite programą dviem konstantomis, skirtomis maksimaliam masyvo dydžiui ir maksimaliam miestų pavadinimų simbolių skaičiui (ilgiui) atmintyje laikyti:

```
const int CMax = 300;  
const int CPav = 20;
```

- Aprašykite struktūrą ir kintamuosius:

```
struct Miestas {  
    string pav,           // miesto pavadinimas  
        valst;          // valstybės pavadinimas  
    int kiek;           // miestų skaičius valstybėje  
};
```

Pradiniams duomenims atmintyje laikyti skirti pirmieji du struktūros kintamieji, o rezultatams (valstybės pavadinimui ir miestų skaičiui toje valstybėje) – antrasis ir trečiasis. Taip daryti yra patogiau, nei kurti dvi atskiras struktūras pradiniams duomenims ir rezultatams.

- Parašykite funkcijos *Skaityti()*, skirtos pradiniams duomenims skaityti iš failo į masyvą *A(n)*, prototipą:

```
void Skaityti(Miestas A[], int &n);
```

- Parašykite funkcijos *Skaityti()* tekstą:

```
// Skaitomi pradiniai duomenys iš failo, nurodyto konstanta CDFv, į masyvą A(n)  
void Skaityti(Miestas A[], int &n)  
{  
    char eil[CPav + 1];           // papildoma vieta eilutės pabaigos simboliui '\0'  
    ifstream fd(CDFv);  
    fd >> n;                       // miestų skaičius  
    fd.ignore(80, '\n');           // visi simboliai failo eilutėje iki jos pabaigos praleidžiami  
    for (int i = 0; i < n; i++) {  
        fd.get(eil, CPav);         // skaitomas miesto pavadinimas  
        A[i].pav = eil;            // miesto pavadinimas konvertuojamas į string tipo eilutę  
        getline(fd, A[i].valst);  // skaitomas miesto pavadinimas iki failo eilutės pabaigos  
        A[i].kiek = 1;            // apskaičiuojamas valstybės miestų skaičius  
    }  
    fd.close();  
}
```

*Pastaba.* Visus veiksmus sėkmingai buvo galima atlikti ir naudojantis *char[]* tipo eilutėmis. Tam yra daug standartinių funkcijų. Tačiau *string* tipo eilutėms lengviau taikyti priskyrimo ir palyginimo operandus. Iš failo *string* tipo eilutes galima skaityti dviem būdais: iki failo eilutės pabaigos arba iki nurodyto skirtuko, žymincio eilutės pabaigą. Antrasis būdas nėra patogus, nes nepakanka sugalvoti, kokį simbolį pasirinkti skirtuku, bet jį reikia įrašyti. Įvedant *char[]* tipo eilutes, galima nurodyti, kiek simbolių iš eilės reikia skaityti.

- Papildykite pagrindinę funkciją main() kreipiniu į funkciją Skaityti(). Norėdami įsitikinti, kad pradiniai duomenys perskaityti teisingai, pagrindinėje funkcijoje parašykite perskaityto miestų skaičiaus rodymo ekrane sakinį:

```
cout << "n = " << n << endl;
```

```
struct Miestas {
    string pav // miesto pavadinimas
    valst; // valstybės pavadinimas
    int kiek; // miestų skaičius valstybėje
};
//-----
const char CDfv[] = "Duomenys15.txt";
const int CMax = 300;
const int CPav = 20;
//-----
void Skaityti(Miestas A[], int & n);
//-----
int main()
{
    Miestas A[CMax]; int n; // pradinių duomenų masyvas
    Skaityti(A, n);
    cout << "n = " << n << endl; // rodomas ekrane perskaitytas duomenų skaičius
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti, pavyzdžiui, tokią n reikšmę:

```
n = 10
```

- Iš programos pašalinkite eilutę, kad kintamojo n reikšmė nebūtų rodoma ekrane.

2

## Rezultatų rašymas į failą

- Papildykite programą konstanta rezultatų failo vardui atmintyje laikyti:

```
const char CRfv[] = "Rezultatai15.txt";
```

- Parašykite funkcijos Spausdinti(), skirtos masyvo A(n) reikšmėms rašyti į rezultatų failą, prototipą:

```
void Spausdinti(Miestas A[], int n, string pav);
```

- Papildykite programą funkcijos Spausdinti() tekstu:

```
// Rašomi valstybių pavadinimai ir nurodoma, kiek toje valstybėje yra miestų
// pav – rezultatų sąrašo pavadinimas
void Spausdinti(Miestas A[], int n, string pav)
{
    ofstream fr(CRfv);
    fr << pav << endl;
    for (int i = 0; i < n; i++)
        fr << setw(15) << A[i].pav
            << setw(15) << A[i].valst
            << setw(6) << A[i].kiek << endl;
    fr.close();
}
```

- Papildykite pagrindinę funkciją `main()` kreipiniu į spausdinimo funkciją:

```
Spausdinti(A, n, "Valstybių sąrašas:");
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą *Rezultatai15.txt*. Jame turėtumėte matyti:

```
Valstybių sąrašas:
    Trakai          Lietuva      1
    Madridas        Ispanija    1
    Barcelona       Ispanija    1
    Roma            Italija     1
    Paryžius        Prancūzija 1
    Saragosa        Ispanija    1
    Genuja          Italija     1
    Tartu           Estija      1
    Kudirkos Naumiestis Lietuva     1
    Venecija        Italija     1
```

3

### Valstybių sąrašo sudarymas

- Papildykite programą funkcijų `Atrinkti()` ir `Yra()` prototipais:

```
void Atrinkti(Miestas A[], int n, Miestas B[], int & m);
int Yra(Miestas A[], int n, string pav);
```

Funkcija `Atrinkti()` skirta valstybių sąrašui sudaryti. Nuosekliai peržiūrimas duomenų masyvas `A(n)`. Kiekvienos masyvo reikšmės, laikomos atmintyje kintamajame `valst` (valstybės pavadinimas), ieškoma masyve `B(m)`. Jeigu tokios valstybės pavadinimo dar nėra, tuomet masyvo `A` elemento reikšmė kopijuojama į masyvo `B` pabaigą. Jeigu tokios valstybės pavadinimas yra, tuomet surastose masyve `B` vietoje kintamojo kiek reikšmė didinama vienetu. Paieškai skirta funkcija `Yra()`.

- Parašykite funkcijų `Atrinkti()` ir `Yra()` tekstą:

```
// Formuojamas sąrašas, kiek kurioje valstybėje yra miestų
// Masyvas A(n) – pradiniai duomenys, B(m) – rezultatai
void Atrinkti(Miestas A[], int n, Miestas B[], int & m)
{
    m = 0; // rezultatų masyvas tuščias
    for (int i = 0; i < n; i++){
        int k = Yra(B, m, A[i].valst); // masyve B(m) ieškoma valstybės pavadinimo
        if (k >= 0)
            B[k].kiek++; // valstybės pavadinimas surastas
        else {
            B[m] = A[i]; // valstybės pavadinimo nerasta; masyvas B papildomas nauja reikšme
            m++;
        }
    }
}

// -----
// Masyve A(n) ieškoma, ar pavadinimas pav yra tarp valst
int Yra(Miestas A[], int n, string pav)
{
    for (int i = 0; i < n; i++)
        if (A[i].valst == pav) return i; // paieška sėkminga; grąžinamas indeksas,
                                           // kur buvo rastas pav
    return -1; // paieška nesėkminga; grąžinama neigiama reikšmė
}
}
```

- Papildykite pagrindinę funkciją rezultatų masyvo B (m) aprašymu ir kreipiniu į funkciją Atrinkti (). Kreipinyje į spausdinimo funkciją masyvą A (n) pakeiskite masyvu B (m):

```
int main()
{
    Miestas A[CMax]; int n; // pradinių duomenų masyvas
    Miestas B[CMax]; int m; // rezultatų masyvas
    Skaityti(A, n);
    Atrinkti(A, n, B, m); // daromas valstybių sąrašas
    Spausdinti(B, m, "Valstybių sąrašas:");
    return 0;
}
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą *Rezultatai15.txt*. Jame turėtumėte matyti:

Valstybių sąrašas:		
Trakai	Lietuva	2
Madridas	Ispanija	3
Roma	Italija	3
Paryžius	Prancūzija	1
Tartu	Estija	1

- Pirmame stulpelyje rašomas vieno valstybės miesto pavadinimas. Jis rezultatų faile nereikalingas. Spausdinimo funkcijoje void Spausdinti () pašalinkite miesto pavadinimo rašymo į rezultatų failą sakinį.

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
struct Miestas {
    string pav // miesto pavadinimas
    valst; // valstybės pavadinimas
    int kiek; // miestų skaičius valstybėje
};
//-----
const char Cdfv[] = "Duomenys15.txt";
const char CRfv[] = "Rezultatai15.txt";
const int CMax = 300;
const int CPav = 20;
//-----
void Skaityti(Miestas A[], int & n);
void Spausdinti(Miestas A[], int n, string pav);
void Atrinkti(Miestas A[], int n, Miestas B[], int & m);
int Yra (Miestas A[], int n, string pav);
//-----
int main()
{
    Miestas A[CMax]; int n; // pradinių duomenų masyvas
    Miestas B[CMax]; int m; // rezultatų masyvas
    Skaityti(A, n);
    Atrinkti(A, n, B, m);
    Spausdinti(B, m, "Valstybių sąrašas:");
    return 0;
}
```



## Programos patikrinimas

- Patikrinkite, kaip dirba programa esant skirtingiems pradinių duomenų rinkiniams, pavyzdžiui, kai faile yra:
  - ✓ tik vieno miesto duomenys;
  - ✓ 300 miestų duomenys (ribinis duomenų skaičius);
  - ✓ nedaug miestų, bet visi priklauso vienai valstybei;
  - ✓ valstybių sostinės.

*Pastaba.* Pradinių duomenų rinkinius galima surašyti į skirtingus failus. Tada, prieš vykdant programą, kaskart pradinių duomenų failo vardą programoje reikia pakeisti nauju. Jeigu norima, kad ir rezultatai būtų įrašomi į skirtingus failus, tai prieš vykdant programą reikia pakeisti rezultatų failo vardo konstantos reikšmę.

- Pagalvokite, kokių dar galėtų būti nepatiktinų situacijų, ir paruoškite atitinkamus duomenų rinkinius.



## Programos papildymas

- Papildykite programą funkcija, kuri surikiuotų valstybių sąrašą pagal miestų skaičių mažėjančiai.
- Programa turi pateikti rezultatus ir tada, kai pradinių duomenų failas tuščias, t. y. pirmoje eilutėje nurodytas skaičius 0 (nulis). Ką šiuo atveju matysite rezultatų faile? Papildykite programą reikalingais pranešimais, kurie būtų rašomi į rezultatų failą.



## Užduotys

### 1. Vardai

Pateiktas mokyklos mokinių sąrašas. Parenkite programą, kuri surikiuotų mokinių sąrašą pagal vardų pasikartojimo dažnį ir pateiktų jį abėcėliškai.

Pirmoje pradinių duomenų failo eilutėje nurodytas mokinių skaičius. Toliau kiekvienoje eilutėje yra mokinio pavardė ir vardas. Pavardei skirta 20 pirmųjų eilutės pozicijų, vardui – dar 20 pozicijų.

2	Pradiniai duomenys	Rezultatai
7		Algis 1
	Margis Petras	Rima 1
	Batuotas Algis	Rimas 1
	Barsė Rita	Rita 2
	Barsis Rimas	Petras 2
	Liepa Petras	
	Liepa Rima	
	Liepa Rita	

### 2. Ledai

Parduotuvė IKS, atlikdama valgomųjų ledų populiarumo tyrimą, apklausė Smaližių mokyklos mokinius. Kiekvienas mokinys į failą surašė, kiek kokių ledų per savaitę suvalgė. Parenkite programą, kuri sudarytų ledų sąrašą. Turi būti nurodytas ledų pavadinimas ir pinigų suma, kurią sumokėjo mokiniai už tos rūšies ledus. Sąrašas turi būti surikiuotas pagal sumokėtą pinigų sumą mažėjančiai.

Pirmoje pradinių duomenų failo eilutėje nurodytas apklaustų mokinių skaičius. Kitose eilutėse įrašyti kiekvienos rūšies ledų duomenys: pavadinimas (20 pirmųjų pozicijų), kiek porcijų mokinys suvalgė, vienos porcijos kaina.

Pradiniai duomenys			Rezultatai	
6			Riešutiniai	76.56
Baltieji vaisiniai	5	3.5	Mégėjų	31.3
Mégėjų	3	6.3	Žalieji agurkiniai	28.75
Žalieji agurkiniai	25	1.15	Baltieji vaisiniai	17.5
Aromatiniai	15	0.57	Aromatiniai	8.55
Riešutiniai	22	3.48		
Mégėjų	2	6.2		

### 3. Gyvūnai

Vakario miesto gyventojai laiko namuose daug įvairių gyvūnų. Vienas savininkas gali turėti kelių rūšių gyvūnų. Parenkite programą, kuri sudarytų mieste esančių gyvūnų sąrašą ir surikiuotą pagal jų skaičių mažėjančiai.

Pirmoje pradinių duomenų failo eilutėje nurodytas gyvūnų savininkų skaičius. Toliau kiekvienoje eilutėje yra gyvūno pavadinimas (15 pirmųjų pozicijų), gyvūnų skaičius ir savininko vardas bei pavardė (20 pozicijų skaičiuojant nuo 21 eilutės pozicijos).

Pradiniai duomenys			Rezultatai	
5			Pelė baltoji	14
Katinas	3	Petras Rudasis	Žiurkėnas	5
Pelė baltoji	14	Jurgis Rudasis	Katinas	4
Katinas	1	Rita Rudoji	Vilkas pilkas	2
Vilkas pilkas	2	Petras Rudasis		
Žiurkėnas	5	Jurgis Rudasis		

### 4. Rinkimai

Į mokyklos mokinių tarybą renkama  $n$  mokinių. Kandidatų sąrašė yra  $k$  mokinių ( $n \leq k$ ). Visi mokiniai, gavę biuletenius su kandidatų sąrašais, paliko juose  $n$  neišbrauktų kandidatų. Biuletenių duomenys buvo perkelti į failą: kiekvienoje eilutėje nurodytas kandidato vardas ir pavardė. Parenkite programą, kuri pateiktų failė balsavimo rezultatus, surikiuotus mažėjančiai pagal surinktų balsų skaičių.

Pirmoje pradinių duomenų failo eilutėje nurodytas duomenų skaičius. Toliau kiekvienoje eilutėje yra kandidato vardas ir pavardė (vardui ir pavardei skiriama po 15 pozicijų eilutėje).

Pradiniai duomenys			Rezultatai		
6			Algis	Liepa	2
Algis	Liepa		Petras	Kaladė	2
Petras	Kaladė		Rita	Kaladė	1
Petras	Kaladė		Algis	Kaladė	1
Rita	Kaladė				
Algis	Liepa				
Algis	Kaladė				



## 1.16. Duomenų atranka

Atlikdami šį darbą:

- ✓ sukursite struktūros duomenų tipą ir masyvą, kurio elementai yra šio tipo;
- ✓ išmoksite rašyti funkcijas, kurias galima panaudoti daug kartų su skirtingais duomenimis;
- ✓ pakartosite struktūros tipo masyvo elementų reikšmių rikiavimo algoritmą;
- ✓ išmoksite formuoti naują duomenų sąrašą.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.1. Operatoriai	3.11. Naujo masyvo formavimo algoritmas
2.4. Duomenų skaitymas iš failo	3.12. Kiti rikiavimo algoritmai
2.5. Rezultatų (duomenų) rašymas į failą	
2.6. Funkcijos	
2.7. Masyvas	
2.9. Simbolių eilutė <code>string</code>	
2.10. Struktūra	
2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	

### Užduotis

**Bėgikai.** Jovaro gimnazijoje mokslo metų pabaigoje vyksta sporto šventė. Stadione yra aštuoni bėgimo takeliai. Norinčių bėgti 800 metrų užsiregistravo  $n$  ( $8 < n < 16$ ) bėgikų. Jie buvo paskirstyti į du pogrupius taip, kad viename bėgime varžytųsi ne mažiau kaip du bėgikai. Reikia parašyti programą, kuri atrinktų finaliniam bėgimui pusę geriausiųjų pirmojo ir pusę geriausiųjų antrojo bėgimo dalyvių ir paskirtų jiems bėgimo takelius. Jeigu bėgikų grupėje buvo nelyginis skaičius, prieš atranką reikia atmesti bėgiką, kurio rezultatas buvo blogiausias. Takeliai skiriami tokia tvarka: blogiausiam bėgikui – pirmasis takelis, iš likusiųjų blogiausiam – antrasis takelis ir t. t. Geriausio rezultato savininkui skiriamas paskutinis iš eilės dar laisvas takelis, pavyzdžiui, jeigu finale bus 6 mokiniai, tai geriausias bėgs šeštuoju takeliu. Į rezultatų failą reikia lentelė išvesti finalinio bėgimo dalyvius, surikiuotus didėjančiai pagal atrankos bėgimuose parodytą laiką, ir greta nurodyti finalinio bėgimo takelio numerį.

Pirmojo atrankos bėgimo rezultatai įrašyti į vieną pradinių duomenų failą, antrojo – į kitą. Pirmoje failo eilutėje yra bėgikų skaičius. Toliau kiekvienoje eilutėje yra bėgiko duomenys: vardas, pavardė ir rezultatas (laikas minutėmis ir sekundėmis).

*Pradinių duomenų ir rezultatų failų pavyzdys*

Pradiniai duomenys	Rezultatai
<b>Pirmas failas</b>	
5	
Petras Pirmas           2 25	
Jurgis Antras           3 1	
Algis Trečias           0 59	
Robertas Ketvirtas    3 13	
Kasparas Penktas     2 45	
<b>Antras failas</b>	
6	
Petras Didysis        1 25	
Jurgis Greitasis     3 11	
Algis Skrajūnas      0 56	
Robertas Pirmūnas   3 13	
Kasparas Eiklusis    2 45	
Rimas Batuotas      4 15	
	Bėgikų finalinio bėgimo sąrašas
	-----
	Vardas ir                   Parodytas    Bėgimo
	pavardė                    laikas        takelis
	-----
	Algis Skrajūnas           0 : 56        5
	Algis Trečias             0 : 59        4
	Petras Didysis           1 : 25        3
	Petras Pirmas            2 : 25        2
	Kasparas Eiklusis       2 : 45        1
	-----

## Algoritmas

Užduotis gali būti sprendžiama taip:

1. Skaitomi pirmojo failo duomenys į masyvą.
2. Rikiuojami masyvo duomenys.
3. Pusės geriausių bėgikų duomenys kopijuojami į finalinio bėgimo dalyvių sąrašą (masyvą).
4. Skaitomi antrojo failo duomenys į masyvą.
5. Rikiuojami masyvo duomenys.
6. Pusės geriausių duomenys bėgikų kopijuojami į finalinio bėgimo dalyvių sąrašą (masyvą).
7. Rikiuojamas rezultatų masyvas.
8. Rezultatai rašomi į tekstinį failą (nurodomas kiekvieno bėgiko bėgimo takelio numeris).

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
Skaityti ()	Vieno failo duomenų skaitymas iš failo
Spausdinti ()	Rezultatų rašymas į failą
Rikiuoti ()	Sąrašo rikiavimas
Atrinkti ()	Sportininkų atranka



### Pradinių duomenų failo kūrimas, duomenų skaitymas iš failo

- Tame pačiame kataloge, kur ir programos failas, sukurkite pirmąjį tekstinį failą *Duomenys16a.txt* ir į jį įrašykite pateiktus pavyzdyje pradinius duomenis.
- Papildykite programą konstanta pirmojo pradinių duomenų failo vardui atmintyje laikyti:

```
const char CDfva[] = "Duomenys16a.txt";
```

- Papildykite programą dviem konstantomis. Viena jų skirta maksimaliam masyvo dydžiui, kita – maksimaliam mokinio vardo ir pavardės simbolių skaičiui (ilgiui) atmintyje laikyti:

```
const int CMax = 30;  
const int CPav = 20;
```

- Aprašykite duomenų struktūrą:

```
struct Sportininkas {  
    string pav;           // vardas ir pavardė  
    int laikas;          // laikas sekundėmis  
};
```

- Papildykite pagrindinę funkciją `main()` kintamaisiais pradiniais duomenimis atmintyje laikyti:

```
Sportininkas A[CMax]; int n; // masyvas pradiniais duomenimis laikyti
```

- Parašykite funkcijos `Skaityti()`, skirtos pradiniais duomenimis iš failo skaityti į masyvą `A(n)`, prototipą:

```
void Skaityti(const char fv[], Sportininkas A[], int & n);
```



- Parašykite funkcijos `Skaityti()` tekstą:

```
// Skaitomi pradiniai duomenys iš failo į masyvą A(n)
void Skaityti(const char fv[], Sportininkas A[], int & n)
{
    ifstream fd (fv);
    int min, sek;
    char eil[CPav+1];
    fd >> n; // bėgikų skaičius
    fd.ignore(80, '\n'); // pereinama į kitą eilutę
    for (int i = 0; i < n; i++){
        fd.get(eil, CPav); // bėgiko vardas ir pavardė
        A[i].pav = eil;
        fd >> min >> sek; // bėgiko laikas minutėmis ir sekundėmis
        fd.ignore(80, '\n'); // pereinama į kitą eilutę
        A[i].laikas = min * 60 + sek; // apskaičiuojamas laikas sekundėmis
    }
    fd.close();
}
```

- Papildykite pagrindinę funkciją `kreipiniu` į funkciją `Skaityti()`. Norėdami įsitikinti, kad pradiniai duomenys perskaityti teisingai, pagrindinėje funkcijoje parašykite kontrolinio perskaityto bėgikų skaičiaus išvedimo į ekraną sakinį:

```
const char Cdfva[] = "Duomenys16a.txt";
const int CMax = 30;
const int CPav = 20;
//-----
struct Sportininkas {
    string pav; // vardas ir pavardė
    int laikas; // laikas sekundėmis
};
//-----
void Skaityti(char fv[], Sportininkas A[], int & n);
//-----
int main()
{
    Sportininkas A[CMax]; int n; // masyvas pradiniam duomenims laikyti
    Skaityti(Cdfva, A, n);
    cout << "n = " << n << endl;
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekране turėtumėte matyti, pavyzdžiui, tokią `n` reikšmę:

```
n = 5
```

- Pašalinkite iš pagrindinės funkcijos sakinį, kad kintamojo `n` reikšmė nebūtų rodoma ekrane.



## 2 Duomenų rašymas į failą

- Papildykite programą – aprašykite konstantą rezultatų failo vardui atmintyje laikyti:

```
const char CRfv[] = "Rezultatai16.txt";
```

- Parašykite funkcijos `Spausdinti()`, skirtos masyvo `A(n)` elementų reikšmėms rašyti į rezultatų failą, prototipą:

```
void Spausdinti(Sportininkas A[], int n);
```

➤ Parašykite funkcijos tekstą:

```
// Masyvo A(n) duomenys rašomi į failą, nurodytą konstanta CRFv
void Spausdinti(Sportininkas A[], int n)
{
    ofstream fr(CRFv);
    fr << "Bėgikų sąrašas" << endl;
    fr << "-----" << endl;
    fr << "Vardas ir          Parodytas " << endl;
    fr << "pavardė          laikas " << endl;
    fr << "-----" << endl;
    for (int i = 0; i < n; i++)
        fr << setw(20) << left << A[i].pav << " "
            << A[i].laikas / 60 << " : "
            << A[i].laikas % 60 << endl;
    fr << "-----" << endl;
    fr.close();
}
```

➤ Papildykite pagrindinę funkciją main() kreipiniu į spausdinimo funkciją:

```
Spausdinti(A, n);
```

➤ Įrašykite ir įvykdykite programą.

➤ Atverkite rezultatų failą *Rezultatai16.txt*. Jame turėtumėte matyti pirmojo pradinių duomenų failo duomenis:

```
Bėgikų sąrašas
-----
Vardas ir          Parodytas
pavardė          laikas
-----
Petras Pirmas      2 : 25
Jurgis Antras     3 : 1
Algis Trečias     0 : 59
Robertas Ketvirtas 3 : 13
Kasparas Penktas  2 : 45
-----
```

### 3 Duomenų sąrašo rikiavimas

➤ Papildykite programą duomenų rikiavimo funkcija. Parašykite funkcijos Rikiuoti() prototipą:

```
void Rikiuoti(Sportininkas A[], int n);
```

➤ Parašykite funkcijos Rikiuoti() tekstą:

```
// Rikiuojami masyvo A(n) duomenys pagal laiką didėjančiai
void Rikiuoti(Sportininkas A[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = i + 1; j < n; j++)
            if (A[j].laikas < A[i].laikas){
                Sportininkas sp = A[i];
                A[i] = A[j];
                A[j] = sp;
            }
}
```

- Papildykite pagrindinę funkciją kreipiniu į funkciją Rikiuoti():

```
int main()
{
    Sportininkas A[CMax]; int n; // masyvas pradiniais duomenims laikyti
    Skaityti(Cdfva,A, n);
    Rikiuoti(A, n);
    Spausdinti(A, n);
    return 0;
}
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą *Rezultatai16.txt*. Jame turėtumėte matyti surikiuotus pagal laiką pirmojo pradinųjų duomenų failo duomenis:

Bėgikų sąrašas	
Vardas ir pavardė	Parodytas laikas
Algis Trečias	0 : 59
Petras Pirmas	2 : 25
Kasparas Penktas	2 : 45
Jurgis Antras	3 : 1
Robertas Ketvirtas	3 : 13

4

#### Pusės geriausių bėgikų atrinkimas finaliniam bėgimui

- Papildykite pagrindinę funkciją – aprašykite naują masyvą, kuriame bus laikomas finalinio bėgimo dalyvių sąrašas:

```
Sportininkas B[CMax]; int m = 0; // sąrašas iš pradžių turi būti tuščias (m = 0)
```

- Parašykite funkcijos Atrinkti(), skirtos pusei dalyvių duomenų perrašyti iš surikiuoto masyvo į finalinio bėgimo sąrašo masyvą, prototipą:

```
void Atrinkti(Sportininkas A[], int n, Sportininkas B[], int & m);
```

- Parašykite funkcijos Atrinkti() tekstą:

```
// Iš masyvo A(n) pusė duomenų kopijuojama į masyvą B(m)
void Atrinkti(Sportininkas A[], int n, Sportininkas B[], int & m)
{
    for (int i = 0; i < n / 2; i++) {
        B[m] = A[i];
        m++;
    }
}
```

- Papildykite programą kreipiniu į funkciją Atrinkti() ir kreipinyje į spausdinimo funkciją įrašykite masyvo B(m) vardą B ir duomenų skaičių m:

```
int main()
{
    Sportininkas A[CMax]; int n; // masyvas pradiniais duomenims laikyti
    Sportininkas B[CMax]; int m = 0; // sąrašas iš pradžių turi būti tuščias (m = 0)
```

```

Skaityti(CDfva,A, n);
Rikiuoti(A, n);
Atrinkti(A, n, B, m);
Spausdinti(B, m);
return 0;
}

```

- Įrašykite ir įvykdysite programą.
- Atverkite rezultatų failą *Rezultatai16.txt*. Jame turėtumėte matyti atrinktus pagal laiką pirmojo pradinių duomenų failo duomenis:

Bėgikų sąrašas	
Vardas ir pavardė	Parodytas laikas
Algis Trečias	0 : 59
Petras Pirmas	2 : 25

5

### Programos papildymas veiksmais antrojo pradinių duomenų failo duomenims apdoroti

- Tame pačiame kataloge, kur ir programos failas, sukurkite antrąjį tekstinį failą *Duomenys16b.txt*. Į jį įrašykite pateiktus pavyzdyje pradinius duomenis.
- Papildykite programą konstanta antrojo pradinių duomenų failo vardui atmintyje laikyti:

```
const char CDfvb[] = "Duomenys16b.txt";
```

- Papildykite pagrindinę funkciją veiksmis duomenims iš antrojo pradinių duomenų failo skaityti, rikiuoti ir atrinkti:

```

int main()
{
    Sportininkas A[CMax]; int n; // masyvas pradiniais duomenimis laikyti
    Sportininkas B[CMax]; int m = 0; // sąrašas iš pradžių turi būti tuščias (m = 0)
    Skaityti(CDfva,A, n);
    Rikiuoti(A, n);
    Atrinkti(A, n, B, m);
    Skaityti(CDfvb,A, n);
    Rikiuoti(A, n);
    Atrinkti(A, n, B, m);
    Spausdinti(B, m);
    return 0;
}

```

- Įrašykite ir įvykdysite programą.
- Atverkite rezultatų failą *Rezultatai16.txt*. Jame turėtumėte matyti atrinktus pagal laiką abiejų pradinių duomenų failų duomenis:

Bėgikų sąrašas	
Vardas ir pavardė	Parodytas laikas
Algis Trečias	0 : 59
Petras Pirmas	2 : 25
Algis Skrajūnas	0 : 56
Petras Didysis	1 : 25
Kasparas Eiklusis	2 : 45

- Papildykite pagrindinę funkciją kreipiniu į rikiavimo funkciją:

```
Rikiuoti(B, m);
```

- Įrašykite ir įvykdykite programą. Rezultatų faile turėtumėte matyti tuos pačius duomenis, kaip ir po ankstesnio žingsnio, tik surikiuotus pagal laiką didėjančiai.



6

### Finalinio bėgimo dalyvių sąrašo papildymas bėgimo takelio numeriu

- Su bėgimo takelių numeriais nereikia atlikti kokių nors papildomų veiksmų, todėl jų nereikia įsiminti. Vadinasi, bėgimo takelių numeriams įrašyti į failą nebūtina kurti atskiros funkcijos ir nereikia pertvarkyti struktūros duomenų tipo. Takelių numerius galima nurodyti duomenis rašant į failą. Tam reikia šiek tiek pakoreguoti funkciją Spausdinti():

```
// Masyvo A(n) duomenys rašomi į failą, nurodytą konstanta CRfv
void Spausdinti(Sportininkas A[], int n)
{
    ofstream fr(CRfv);
    fr << "Bėgikų finalinio bėgimo sąrašas" << endl;
    fr << "-----" << endl;
    fr << "Vardas ir          Parodytas    Bėgimo " << endl;
    fr << "pavardė            laikas      takelis" << endl;
    fr << "-----" << endl;
    for (int i = 0; i < n; i++)
        fr << setw(20) << left << A[i].pav << "    "
            << A[i].laikas / 60 << " : "
            << A[i].laikas % 60 << "    "
            << "    " << n - i << endl; // bėgimo takelio nr.
    fr << "-----" << endl;
    fr.close();
}
```

- Įrašykite ir įvykdykite programą. Rezultatų faile turėtumėte matyti pateiktus pavyzdyje duomenis.

### Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char Cdfva[] = "Duomenys16a.txt";
const char Cdfvb[] = "Duomenys16b.txt";
const char CRfv[] = "Rezultatai16.txt";
const int CMax = 30;
const int CPav = 20;
//-----
struct Sportininkas {
    string pav; // vardas ir pavardė
    int laikas; // laikas sekundėmis
};
//-----
void Skaityti(const char fv[], Sportininkas A[], int & n);
void Spausdinti(Sportininkas A[], int n);
void Rikiuoti(Sportininkas A[], int n);
void Atrinkti(Sportininkas A[], int n, Sportininkas B[], int & m);
//-----
```

```

int main()
{
    Sportininkas A[CMax]; int n; // masyvas pradiniais duomenimis laikyti
    Sportininkas B[CMax]; int m = 0; // sąrašas iš pradžių turi būti tuščias (m = 0)
    Skaityti(CDfva, A, n);
    Rikiuoti(A, n);
    Atrinkti(A, n, B, m);
    Skaityti(CDfva, A, n);
    Rikiuoti(A, n);
    Atrinkti(A, n, B, m);
    Rikiuoti(B, m);
    Spausdinti(B, m);
    return 0;
}

```



## Programos patikrinimas

- Patikrinkite, kaip dirba programa esant skirtingiems pradinių duomenų rinkiniams, pavyzdžiui, kai pradinių duomenų:
  - ✓ pirmame faile yra tik du, o antrajame – aštuoni bėgikai;
  - ✓ pirmame faile yra tik du, o antrajame – septyni bėgikai;
  - ✓ abiejuose failuose yra po aštuonis bėgikus;
  - ✓ pirmame faile visų bėgikų rezultatas vienodas.

*Pastaba.* Pradinių duomenų rinkinius galima surašyti į skirtingus failus. Tada, prieš vykdant programą, kaskart pradinių duomenų failo vardą programoje reikia pakeisti nauju. Jeigu norima, kad ir rezultatai būtų rašomi į skirtingus failus, tai prieš vykdant programą reikia pakeisti rezultatų failo vardo konstantos reikšmę.

- Pagalvokite, kokių dar galėtų pasitaikyti situacijų, ir paruoškite atitinkamus duomenų rinkinius.



## Programos papildymas

- Patikrinkite, kaip dirba programa, kai pirmame ir antrame atrankos bėgimuose buvo tik po du bėgikus.
- Papildykite programą veiksmis, kad visi keturi bėgikai būtų perkelti į finalinį bėgikų sąrašą.
- Patikrinkite, kaip dirba programa, kai abiejuose bėgimuose buvo tik po vieną bėgiką. Papildykite programą veiksmis, kuriais abu bėgikai būtų atrenkami finaliniam bėgimui.
- Patikrinkite, kaip dirba programa, kai antrame bėgime nstartavo nė vienas bėgikas.
- Patikrinkite, kaip dirba programa, kai viename iš atrankos bėgimų didesnės dalies bėgikų rezultatas yra geriausias ir vienodas. Ką šiuo atveju patartumėte daryti programuotojui?



## Užduotys

### 1. Kolekcija

Martynas kolekcionuoja lietuvių atlikėjų muzikos albumus. Albumų kolekcija nuolat papildoma, todėl Martynas nusprendė sukurti elektroninį muzikos albumų katalogą. Informaciją apie katalogus jis surašė į tekstinį failą *DuomAlbumai.txt*. Pirmoje jo eilutėje nurodė turimų albumų skaičių  $n$  ( $3 \leq n \leq 500$ ). Tolesnėse  $n$  eilučių nurodė kiekvieno albumo pavadinimą (jam skyrė 20 pozicijų), albumo išleidimo metus, albume esančių įrašų trukmę valandomis ir minutėmis bei kiek kartų buvo perklaustas albumas.

Parente programą, kuri į tekstinį failą *RezAlbumai.txt* įrašytų:

- ✓ pirmoje eilutėje – visų katalogo dainų trukmę (valandomis ir minutėmis);
- ✓ antroje eilutėje – kiek vidutiniškai kartų buvo perklausomas vienas albumas;
- ✓ tolesnėse  $n$  eilučių albumų pavadinimus, surikiuotus pagal populiarumą didėjančiai (kuo daugiau kartų perklaustas albumas, tuo populiarsnis). Jei albumai vienodai populiari, tai jų rikiuoti pagal kitą požymį nereikia.

<i>DuomAlbumai.txt</i>	<i>RezAlbumai.txt</i>
3	6 0
Window of My Soul 2008 2 5 5	3.67
Paskutinis šokis 2007 2 3 3	Išklausyk 2007
Išklausyk 2007 1 52 3	Paskutinis šokis 2007
	Window of My Soul 2008

## 2. Kontrolinis darbas

Informatikos mokytoja parengė devintokams kontrolinį darbą, kurį sudaro  $k$  užduočių. Mokinys gali gauti po 10 taškų už kiekvieną teisingai išspręstą užduotį. Gautas taškų skaičius dauginamas iš užduoties svorio koeficiento. Kontrolinio darbo pažymys atitinka visų užduočių įvertinimų sumą.

Pirmoje pradinių duomenų failo *Kontras.txt* eilutėje nurodyti du tarpu atskirti sveikieji skaičiai – mokinių skaičius  $n$  ( $5 \leq n \leq 30$ ) ir užduočių skaičius  $k$  ( $3 \leq k \leq 10$ ). Antroje failo eilutėje įrašyta  $k$  realiųjų skaičių – užduočių svoriai, kurie vienas nuo kito atskirti tarpais. Tolesnėse  $n$  eilučių nurodytas kiekvieno mokinio vardas (jam skiriama 15 pozicijų) ir  $k$  sveikųjų skaičių – kiekvienos užduoties įvertinimas taškais nuo 0 iki 10.

Parente programą, kuri apskaičiuotų ir tekstiniame faile *RezKontras.txt* pateiktų:

- ✓ pirmose  $n$  eilučių – mokinių kontrolinio darbo rezultatus (reikia nurodyti mokinio vardą ir už darbą gautą taškų skaičių, suapvalintą iki sveikiejo skaičiaus);
- ✓ kontrolinio darbo įvertinimų vidurkį, suapvalintą iki sveikiejo skaičiaus;
- ✓ geriausią įvertinimą gavusio mokinio vardą ir įvertinimą (jei yra keli tokie mokiniai, reikia nurodyti juos visus);
- ✓ blogiausią įvertinimą gavusio mokinio vardą ir įvertinimą (jei yra keli tokie mokiniai, reikia nurodyti juos visus);
- ✓ kiek mokinių gavo teigiamą (ne mažesnę kaip 4) įvertinimą.

<i>Kontras.txt</i>	<i>RezKontras.txt</i>
6 4	Tomas 8
0.3 0.2 0.2 0.3	Domas 9
Tomas 6 9 10 7	Rimas 8
Domas 8 10 10 9	Simas 9
Rimas 7 9 9 8	Romas 9
Simas 9 9 9 9	Rokas 8
Romas 10 10 10 8	9
Rokas 8 9 9 7	Domas 9
	Simas 9
	Romas 9
	Tomas 8
	Rimas 8
	Rokas 8
	6

## 3. Praktiški skaitytojai

„Šviesuolių“ miestelį aptarnaujantis laiškininkas išnešioja gyventojų gautus laiškus ir prenumeruojamą spaudą. Metų pradžioje laiškininkas gavo leidimą organizuoti akciją „Prenumeruok didesnę kiekį – mokėsi tris kartus pigiau“. „Šviesuolių“ miestelio gyventojai labai praktiški žmonės. Jie sugalvojo, kad vienas žmogus gali užsiprenumeruoti kelis to paties pavadinimo leidinius, po to vieną pasilikti sau, o kitus išdalyti kaimynams. Kitas žmogus gali užsiprenumeruoti kito pavadinimo kelis leidinius, po to vieną pasilikti sau, o kitus išdalyti kaimynams ir t. t. Kaip tarė, taip padarė.

Pirmoje tekstinio failo *Skaitytojai.txt* eilutėje nurodytas prenumeratorių skaičius  $n$  ( $3 \leq n \leq 20$ ) ir prenumeruojamų leidinių skaičius  $k$  ( $3 \leq k \leq 10$ ), atskirti tarpais. Antroje failo eilutėje surašytos leidinių kainos (realieji skaičiai), viena nuo kitos atskirtos tarpais. Toliau faile yra  $n$  eilučių, kuriose išvardyti miestelio gyventojų vardai (jiems skiriama 15 pozicijų), jų prenumeruojamų leidinių kiekiai nuo pirmojo iki  $k$ -ojo. Jei kurio nors leidinio žmogus neprenumeruoja, toje vietoje nurodytas nulis.

Parente programą, kuri tekstiname faile *RezSkaitytojai.txt* pateiktų:

- ✓ pirmosiose  $n$  eilučių – už kokią pinigų sumą kiekvienas skaitytojas užsiprenumeravo leidinių;
- ✓ už kokią pinigų sumą leidinių užsiprenumeravo visi „Šviesuolių“ miestelio gyventojai;
- ✓ kuris „Šviesuolių“ miestelio gyventojas užsiprenumeravo daugiausia leidinių;
- ✓ kuris „Šviesuolių“ miestelio gyventojas užsiprenumeravo leidinių už mažiausią pinigų sumą.

<i>Skaitytojai.txt</i>	<i>RezSkaitytojai.txt</i>
5 4	Tomas 8.00
2.00 3.00 2.00 2.00	Romas 5.00
Tomas 2 0 2 0	Rimas 18.00
Romas 0 1 1 0	Simas 7.00
Rimas 2 2 1 3	Tadas 7.00
Simas 0 1 0 2	45.00
Tadas 1 1 1 0	Rimas 8
	Romas 5.00

#### 4. Katinų globėjai

Viena pagyvenusi moteriškė yra didelė katinų mylėtoja. Ji kiekvieną dieną daugiabučių namų katinų bendriją aprūpina šviežiu pienu. Taip gerai prižiūrimos katinų bendrijos narių skaičius sparčiai didėja. Moteriškė suprato, kad šitaip lepindama savo numylėtinius ji greitai pritrūks pensijos pienui pirkti, todėl nusprendė į akciją įtraukti kiemo vaikus. Vaikams ji siūlo už sutaupytus perkant ledus pinigus nupirkti katėms pieno. Vaikai sutiko pagelbėti katinų globėjai.

Pirmoje pradinių duomenų failo *Katinai.txt* eilutėje įrašytas sutikusių dalyvauti akcijoje vaikų skaičius  $n$  ( $2 \leq n \leq 10$ ). Tolesnėse  $n$  eilučių nurodytas kiekvieno vaiko vardas (jam skiriama 15 pozicijų), vaiko turimi pinigai (litais ir centais), kelias dienas  $d$  vaikas valgė ledus, kiek porcijų ledų  $k$  suvalgydavo kiekvieną dieną ir vienos porcijos kaina  $kp$ .

Parente programą, kuri apskaičiuotų ir į rezultatų failą *RezKatinai.txt* įrašytų:

- ✓ pirmose  $n$  failo eilučių – kiekvieno vaiko sutaupytus perkant ledus pinigus, t. y. turi būti du sveikieji skaičiai (litai ir centai);
- ✓ tolesnėje failo eilutėje – kiek pinigų liko pienui, t. y. turi būti du sveikieji skaičiai (litai ir centai);
- ✓ po jų – vaiko, kuris skyrė didžiausią pinigų sumą katinams parenti, vardą ir skirtą pinigų sumą (litais ir centais).

<i>Katinai.txt</i>	<i>RezKatinai.txt</i>
3	Tomas 2 95
Tomas 5 15 2 1 1.25 1 0.95	Lukas 1 19
Lukas 3 12 2 1 0.95 1 0.98	Saulius 3 20
Saulius 7 45 2 1 1.25 2 1.50	7 34
	Saulius 3 20



## 1.17. Duomenų šalinimas ir papildymas

Atlikdami šį darbą:

- ✓ sukursite struktūros duomenų tipą ir masyvą, kurio elementai yra šio tipo;
- ✓ išmoksite rašyti funkcijas, kurias galima panaudoti daug kartų su skirtingais duomenimis;
- ✓ prisiminsite struktūros tipo masyvo elementų reikšmių rikiavimo algoritmą;
- ✓ išmoksite iš masyvo pašalinti reikšmes;
- ✓ sužinosite, kaip galima masyvą papildyti naujomis reikšmėmis.



Nuorodos į C++ kalbos ir duomenų struktūrų žinyną	Nuorodos į algoritmų žinyną
2.2. Operatoriai	3.6. Reikšmės šalinimo algoritmas
2.4. Duomenų skaitymas iš failo	3.7. Reikšmės įterpimo algoritmas
2.5. Rezultatų (duomenų) rašymas į failą	3.12. Kiti rikiavimo algoritmai
2.6. Funkcijos	
2.7. Masyvas	
2.9. Simbolių eilutė <code>string</code>	
2.10. Struktūra	
2.11. Knygoje naudojamų įterpiamųjų failų sąrašas	

### Užduotis

Klasės sąrašas. Eglyno gimnazijos keletas mokinių iš „Erelių“ klasės perėjo į kitas klases, o keletas mokinių iš kitų klasių atėjo į „Erelių“ klasę. Reikia parašyti programą, kuri sutvarkytų „Erelių“ klasės mokinių sąrašą: pašalintų išvykusiuosius, įrašytų naujokus, po to sudarytų ir abėcėliškai išrikiuotą sąrašą mokinių, kurie pasirinko etikos dalyką.

Visi trys sąrašai yra viename pradinį duomenų faile.

Failo pradžioje pateiktas išvykusių mokinių sąrašas:

- ✓ pirmoje sąrašo eilutėje yra išvykusiųjų skaičius;
- ✓ kitose eilutėse – jų pavardės ir vardai.

Toliau pateiktas naujokų sąrašas:

- ✓ pirmoje sąrašo eilutėje yra naujokų skaičius;
- ✓ kitose eilutėse – jų pavardės, vardai ir kokį dorinio ugdymo dalyką jie pasirinko (etiką ar tikybą).

Pabaigoje yra klasės mokinių sąrašas, kurį reikia koreguoti:

- ✓ pirmoje sąrašo eilutėje nurodytas mokinių skaičius;
- ✓ kitose eilutėse – mokinių pavardės, vardai ir kokį dorinio ugdymo dalyką jie pasirinko (etiką ar tikybą).

Pradiniai duomenys	Rezultatai
3	Klasės sąrašas
Pirmasis Petras *	-----
Antrutė Jurgita *	Mokinys
Trečiasis Algis *	Dalykas
2	-----
Taiklioji Akis Tikyba	Baravykaitė Marijona Tikyba
Smalsuolė Marcelliuė Etika	Baravykas Algis Etika
10	Eglaitė Ramunė Etika
Baravykas Algis Etika	Gaurius Svajūnas Etika
Baravykaitė Marijona Tikyba	Lepšius Martynas Tikyba
Lepšius Martynas Tikyba	Lepšytė Barbora Etika
Pirmasis Petras Etika	Liepaitė Rita Tikyba
Lepšytė Barbora Etika	Smalsuolė Marcelliuė Etika
Eglaitė Ramunė Etika	Taiklioji Akis Tikyba
Antrutė Jurgita Tikyba	-----
Liepaitė Rita Tikyba	Etiką pasirinkusieji mokiniai
Trečiasis Algis Tikyba	-----
Gaurius Svajūnas Etika	Mokinys
	Dalykas
	-----
	Baravykas Algis Etika
	Eglaitė Ramunė Etika
	Gaurius Svajūnas Etika
	Lepšytė Barbora Etika
	Smalsuolė Marcelliuė Etika
	-----

*Pastaba.* Atliekant veiksmus su simbolių eilutėmis, svarbu, kad jos būtų vienodo ilgio, t. y. simbolių skaičius jose būtų toks pat. Pavyzdžiui, jei programoje eilučių, skirtų mokinio vardui ir pavardei laikyti, didžiausias ilgis nurodytas 25, tuomet, pradinių duomenų faile užrašius mokinio pavardę ir vardą, būtina dar įvesti (klaviatūra arba programiškai) tiek tarpo simbolių, kiek trūksta iki 25. Kad įvestų simbolių kaskart nereikėtų skaičiuoti, patogiu 26 pozicijoje parašyti kokį nors simbolį (pvz., \*), kuris žymėtų eilutės pabaigą. Jeigu tas simbolis eilutėje yra paskutinis, tuomet skaitant duomenis reikia pereiti į kitą eilutę. Jeigu už jo yra kiti duomenys, tuomet skaitant, jį reikia praleisti (pvz., perskaityti jį ir neįrašyti į atmintinę).

## Algoritmas

Užduotis gali būti sprendžiama taip:

1. Pradinių duomenų failo duomenys skaitomi į tris atskirus masyvus: išvykusiųjų, naujokų ir visų klasės mokinių.
2. Iš klasės sąrašo pašalinami išvykusieji.
3. Klasės sąrašas papildomas naujokais.
4. Klasės sąrašas surikiuojamas abėcėliškai.
5. Klasės mokiniai surašomi į rezultatų failą lentele.
6. Atrinkami etikos dalyką pasirinkę mokiniai.
7. Atrinkti mokiniai surašomi į rezultatų failą lentele.

## Programos struktūra

Funkcijos pavadinimas	Funkcijos paskirtis
SkaitytiPav()	Išvykusių mokinių sąrašo formavimas
SkaitytiKlase()	Klasės mokinių sąrašo formavimas
Trinti()	Išvykusiųjų šalinimas
Papildyti()	Papildymas naujokais
Atrinkti()	Mokinių atrinkimas
Rikiuoti()	Sąrašo rikiavimas
Spausdinti()	Rezultatų rašymas į failą



## Pradinių duomenų failo kūrimas, konstantų ir kintamųjų aprašymas

- Tame pačiame kataloge, kur ir programos failas, sukurkite tekstinį failą *Duomenys17.txt*. Į jį įrašykite pateiktus pavyzdyje pradinius duomenis.
- Papildykite programą konstantomis, skirtomis pradinių duomenų ir rezultatų failų vardams, maksimaliam masyvo dydžiui ir maksimaliam pavardės, vardo simbolių skaičiui (ilgiui) atmintyje laikyti:

```
const char CDfv[] = "Duomenys17.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai17.txt"; // rezultatų failo vardas
const int Cmax = 30; // maksimalus masyvų dydis
const int CPav = 25; // maksimalus vardo ir pavardės simbolių skaičius
```

- Aprašykite struktūrą ir duomenų elementus, skirtus duomenims atmintyje laikyti:

```
struct Mokinys {
    string pav; // pavardė ir vardas
    string dalykas; // pasirinkto dalyko pavadinimas
};
```

- Papildykite pagrindinę funkciją `main()` masyvais:

```
Mokinys Klase[CMax]; int n; // klasės mokinių sąrašas
Mokinys Ate[CMax]; int m; // išvykusiųjų sąrašas
Mokinys Nauji[CMax]; int k; // naujokų sąrašas
```

- Išvalykite rezultatų failą:

```
ofstream fr(CRfv);
fr.close();
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tik informacinį pranešimą. Rezultatų failas turi būti tuščias. Jeigu darbiname kataloge tokio failo nebuvo, tuomet jis bus sukurtas.



## Išvykusių mokinių sąrašo skaitymas

- Parašykite funkciją `SkaitytiPav()`, skirtą pirmajam esančiam faile sąrašui skaityti. Šiame sąraše yra tik mokinių pavardės ir vardai, todėl jis skaitomas kitaip nei kiti du sąrašai. Visi sąrašai yra viename faile, todėl jį atverti reikia pagrindinėje funkcijoje, o skaitymo funkcijai perduoti srauto adresą (srauto kintamąjį).

```
// Iš nurodyto duomenų srauto fd skaitoma n reikšmių į masyvą A (išvykusių mokinių pavardės)
void SkaitytiPav(ifstream & fd, Mokinys A[], int & n)
{
    char eil[CPav + 1];
    fd >> n; // skaičius mokinių, kurių pavardės yra surašytos
    fd.ignore(80, '\n'); // pereinama į kitą eilutę
    for (int i = 0; i < n; i++) {
        fd.get(eil, CPav); // perskaitoma vieno mokinio pavardė ir vardas
        A[i].pav = eil;
        A[i].dalykas = ""; // dalykas nenurodytas, todėl į masyvą įrašoma tuščia eilutė
        fd.ignore(80, '\n'); // pereinama į kitą eilutę
    }
}
```

- Papildykite programą funkcijos prototipu:

```
void SkaitytiPav(ifstream & fd, Mokinys A[], int & n);
```

- Papildykite pagrindinę funkciją main() sakiniiais, kuriais pradinį duomenų failas paruošiamas skaityti, kreipiamasi į parašytą skaitymo funkciją ir duomenų failas užveriamas. Norėdami įsitikinti, kad pradiniai duomenys perskaityti teisingai, pagrindinėje funkcijoje parašykite perskaitytų pavardžių skaičiaus rodymo ekrane sakinį.

```
int main()
{
    Mokinys Klase[CMax]; int n; // klasės mokinių sąrašas
    Mokinys Ate[CMax]; int m; // išvykusių mokinių sąrašas
    Mokinys Nauji[CMax]; int k; // naujokų sąrašas
    ofstream fr(CRfv);
    fr.close();
    ifstream fd(CDfv);
    SkaitytiPav(fd, Ate, m);
    cout << "m = " << m << endl;
    fd.close();
    return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti, pavyzdžiui, tokią m reikšmę:

```
m = 3
```

3

### Klasės mokinių sąrašo skaitymas

- Parašykite funkciją SkaitytiKlase() antrajam ir trečiajam sąrašams, esantiems pradinį duomenų faile, skaityti. Sąrašai pateikti ta pačia tvarka, todėl funkcija tiks jiems abiem skaityti. Kadangi duomenų failas jau atvertas ir iš jo reikia skaityti toliau, tai funkcijai reikia perduoti srauto adresą (srauto kintamąjį).

```
// Skaitomi klasės mokinių duomenys
// fd – duomenų srautas, A(n) – masyvas, į kurį skaitomi duomenys
void SkaitytiKlase(ifstream & fd, Mokinys A[], int & n)
{
    char eil[CPav + 1];
    fd >> n; // skaičius mokinių, kurių pavardės surašytos
    fd.ignore(80, '\n'); // pereinama į kitą eilutę
    for (int i = 0; i < n; i++){
        fd.get(eil, CPav); // perskaitoma vieno mokinio pavardė ir vardas
        A[i].pav = eil;
        fd >> ws; // praleidžiami tarpo simboliai iki kitos simbolių eilutės
        getline(fd, A[i].dalykas); // perskaitomas dalyko pavadinimas (iki eilutės pabaigos)
    }
}
```

- Parašykite funkcijos SkaitytiKlase() prototipą:

```
void SkaitytiKlase(ifstream & fd, Mokinys A[], int & n);
```

- Papildykite pagrindinę funkciją kreipiniais į funkciją antrajam ir trečiajam sąrašams skaityti:

```
SkaitytiKlase(fd, Nauji, k);
SkaitytiKlase(fd, Klase, n);
```

- Norėdami įsitikinti, kad duomenys teisingai perskaityti, pagrindinėje funkcijoje parašykite perskaitytų kiekvieno sąrašo simbolių eilučių skaičiaus rodymo ekrane sakinius:

```
cout << "k = " << k << endl;
cout << "n = " << n << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti:

```
m = 3
k = 2
n = 10
```

- Iš pagrindinės funkcijos pašalinkite sakinius, kad kintamųjų  $m$ ,  $k$  ir  $n$  reikšmės nebūtų rodoma ekrane.



#### Klasės mokinių sąrašo formavimas

- Papildykite programą funkcija Spausdinti():

```
// Rašomi masyvo A(n) duomenys lentelė, kurios pavadinimas perduodamas eilute eil
void Spausdinti(Mokinys A[], int n, string eil)
{
    ofstream fr(CRfv, ios::app); // failas paruošiamas papildyti
    fr << eil << endl;
    fr << "-----" << endl;
    fr << "    Mokinys           Dalykas " << endl;
    fr << "-----" << endl;
    for (int i = 0; i < n; i++)
        fr << setw(20) << left << A[i].pav << "    "
            << A[i].dalykas << endl;
    fr << "-----" << endl;
    fr.close();
}
```

- Papildykite pagrindinę funkciją kreipiniais į funkciją Spausdinti(). Kadangi naujokų ir klasės sąrašai yra pateikti vienodai, todėl parašykite du kreipinius:

```
Spausdinti(Klase, n, "Klasės sąrašas");
Spausdinti(Nauji, k, "Naujokų sąrašas");
```

- Įrašykite ir įvykdykite programą.
- Papildykite programą parašytos funkcijos prototipu:

```
void Spausdinti(Mokinys A[], int n, string eil);
```

- Atverkite rezultatų failą *Rezultatai17.txt*. Jame turėtumėte matyti dviejų sąrašų duomenis:

Klasės sąrašas	
Mokinys	Dalykas
Baravykas Algis	Etika
Baravykaitė Marijona	Tikyba
Lepšius Martynas	Tikyba
Pirmasis Petras	Etika
Lepšytė Barbora	Etika
Eglaitė Ramunė	Etika
Antrutė Jurgita	Tikyba
Liepaitė Rita	Tikyba
Trečiasis Algis	Tikyba
Gaurius Svajūnas	Etika

Naujokų sąrašas	
Mokinys	Dalykas
Taiklioji Akis	Tikyba
Smalsuolė Marceliukė	Etika

5

### Įšvykusių mokinių šalinimas iš klasės sąrašo

- Parašykite funkciją `Trinti()` išvykusiems mokiniams pašalinti iš klasės sąrašo:

```
// Iš masyvo A(n) pašalinami masyvo B(m) duomenys
void Trinti(Mokinys B[], int m, Mokinys A[], int &n)
{
    for (int i = 0; i < m; i++) // peržiūrimas šalinamųjų sąrašas
        for (int j = 0; j < n; j++) // ieškoma klasės sąrašo
            if (B[i].pav == A[j].pav) { // lyginamos pavardžių ir vardų eilutės
                A[j] = A[n-1]; // šalinamos reikšmės vietoje įrašoma paskutinė masyvo reikšmė
                n--; // sumažinamas klasės sąrašo reikšmių skaičius
                j = n; // vidinio ciklo darbas nutraukiamas
            }
}
```

- Papildykite programą parašytos funkcijos prototipu:

```
void Trinti(Mokinys B[], int m, Mokinys A[], int &n);
```

- Papildykite programą kreipiniu į parašytą funkciją:

```
Trinti(Ate, m, Klase, n);
```

- Iš pagrindinės funkcijos pašalinkite kreipinį, kuriuo buvo rašomas į failą naujokų sąrašas.
- Įrašykite ir įvykdykite programą.

- Atverkite rezultatų failą *Rezultatai17.txt*. Jame turėtumėte matyti:

Klasės sąrašas	
Mokinys	Dalykas
Baravykas Algis	Etika
Baravykaitė Marijona	Tikyba
Lepšius Martynas	Tikyba
Gaurius Svajūnas	Etika
Lepšytė Barbora	Etika
Eglaitė Ramunė	Etika
Liepaitė Rita	Tikyba



### 6 Klasės sąrašo papildymas naujokais

- Parašykite funkciją `Papildyti()`, skirtą klasės sąrašui papildyti naujokais. Klasės sąrašas nesutvarkytas, todėl paprasčiausiai bus naujokus surašyti turimo klasės sąrašo pabaigoje.

```
// Masyvo C(k) reikšmes kopijuoja į masyvo A(n) pabaigą
void Papildyti(Mokinys C[], int k, Mokinys A[], int & n)
{
    for (int i = 0; i < k; i++) {
        A[n] = C[i];
        n++;
    }
}
```

- Papildykite programą parašytos funkcijos prototipu:

```
void Papildyti(Mokinys C[], int k, Mokinys A[], int & n);
```

- Papildykite programą kreipiniu į parašytą funkciją:

```
Papildyti(Nauji, k, Klase, n);
```

- Įrašykite ir įvykdykite programą.
- Atverkite rezultatų failą *Rezultatai17.txt*. Jame turėtumėte matyti:

Klasės sąrašas	
Mokinys	Dalykas
Baravykas Algis	Etika
Baravykaitė Marijona	Tikyba
Lepšius Martynas	Tikyba
Gaurius Svajūnas	Etika
Lepšytė Barbora	Etika
Eglaitė Ramunė	Etika
Liepaitė Rita	Tikyba
Taiklioji Akis	Tikyba
Smalsuolė Marcellukė	Etika



## Klasės mokinių sąrašo rikiavimas abėcėliškai

- Parašykite funkciją Rikiuoti():

```
// Masyvas A(n) rikiuojamas pagal kintamojo pav reikšmę abėcėliškai
void Rikiuoti(Mokinys A[], int n)
{
    for (int i = 0; i < n; i++)
        for (int j = i + 1; j < n; j++)
            if (A[j].pav < A[i].pav){
                Mokinys sp = A[i];
                A[i] = A[j];
                A[j] = sp;
            }
}
```

- Papildykite programą parašytos funkcijos prototipu:

```
void Rikiuoti(Mokinys A[], int n);
```

- Papildykite programą kreipiniu į parašytą funkciją:

```
Rikiuoti(Klase, n);
```

- Įrašykite ir įvykdykite programą.

- Atverkite rezultatų failą *Rezultatai17.txt*. Jame turėtumėte matyti:

Klasės sąrašas

Mokinys	Dalykas
Baravykaitė Marijona	Tikyba
Baravykas Algis	Etika
Eglaitė Ramunė	Etika
Gaurius Svajūnas	Etika
Lepšius Martynas	Tikyba
Lepšytė Barbora	Etika
Liepaitė Rita	Tikyba
Smalsuolė Marceliukė	Etika
Taiklioji Akis	Tikyba



## Etikos dalyką pasirinkusių sąrašo sudarymas

- Parašykite funkciją Atrinkti(), skirtą etikos dalyką pasirinkusių mokinių duomenims perrašyti į kitą masyvą:

```
// Iš masyvo A(n) perrašomi į masyvą B(m) duomenys,
// kurių kintamojo dalykas reikšmė sutampa su funkcijos parametro dalykas reikšme
void Atrinkti(Mokinys A[], int n, string dalykas, Mokinys B[], int & m)
{
    m = 0;
    for (int i = 0; i < n; i++)
        if (A[i].dalykas == dalykas) {
            B[m] = A[i];
            m++;
        }
}
```



- Papildykite programą parašytos funkcijos prototipu:

```
void Atrinkti(Mokinys A[], int n, string dalykas, Mokinys B[], int & m);
```

- Papildykite programą kreipiniu į parašytą funkciją. Atrenkamų mokinių sąrašui atmintyje laikyti galima sukurti naują masyvą arba panaudoti jau nereikalingą, pavyzdžiui, masyvą Ate:

```
Atrinkti(Klase, n, "Etika", Ate, m);
```

- Papildykite programą kreipiniu į funkciją Spausdinti(), kad atrinktus mokinius būtų galima surašyti į rezultatų failą. Kadangi klasės sąrašas buvo sutvarkytas, tai atrinktų mokinių sąrašo rikiuoti nereikia:

```
Spausdinti(Ate, m, "Etiką pasirinkusieji mokiniai");
```

- Įrašykite ir įvykdykite programą. Rezultatų faile turėtumėte matyti pateiktus pavyzdyje rezultatus.

## Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDfv[] = "Duomenys17.txt"; // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai17.txt"; // rezultatų failo vardas
const int CMax = 30; // maksimalus masyvų dydis
const int CPav = 25; // maksimalus vardo ir pavardės simbolių skaičius
//-----
struct Mokinys {
    string pav; // pavardė ir vardas
    string dalykas; // pasirinkto dalyko pavadinimas
};
//-----
void SkaitytiPav(ifstream & fd, Mokinys A[], int & n);
void SkaitytiKlase(ifstream & fd, Mokinys A[], int & n);
void Trinti(Mokinys B[], int m, Mokinys A[], int & n);
void Papildyti(Mokinys C[], int k, Mokinys A[], int & n);
void Atrinkti(Mokinys A[], int n, string dalykas, Mokinys B[], int & m);
void Rikiuoti(Mokinys A[], int n);
void Spausdinti(Mokinys A[], int n, string eil);
//-----
int main()
{
    Mokinys Klase[CMax]; int n; // klasės mokinių sąrašas
    Mokinys Ate[CMax]; int m; // išvykusių mokinių sąrašas
    Mokinys Nauji[CMax]; int k; // naujų sąrašas
    ofstream fr(CRfv); fr.close();
    ifstream fd(CDfv);
    SkaitytiPav(fd, Ate, m);
    SkaitytiKlase(fd, Nauji, k);
    SkaitytiKlase(fd, Klase, n);
    fd.close();
    Trinti(Ate, m, Klase, n);
    Papildyti(Nauji, k, Klase, n);
    Rikiuoti(Klase, n);
    Spausdinti(Klase, n, "Klasės sąrašas");
    Atrinkti(Klase, n, "Etika", Ate, m);
    Spausdinti(Ate, m, "Etiką pasirinkusieji mokiniai");
    return 0;
}
```



## Programos patikrinimas

- Patikrinkite, kaip dirba programa esant skirtingiems pradinių duomenų rinkiniams, pavyzdžiui, kai:
  - ✓ pirmame sąrašė yra tik vienas mokinys;
  - ✓ antrame sąrašė yra tik vienas mokinys;
  - ✓ pirmas sąrašas tuščias (mokinių skaičius 0);
  - ✓ antras sąrašas tuščias (mokinių skaičius 0);
  - ✓ pirmas ir antras sąrašai tušti (mokinių skaičiai 0);
  - ✓ pirmas ir trečias sąrašai tušti, o antrasis netuščias (yra tik naujokai).

*Pastaba.* Pradinių duomenų rinkinius galima surašyti į skirtingus failus. Tada, prieš vykdant programą, kaskart pradinių duomenų failo vardą programoje reikia pakeisti nauju. Jeigu norima, kad ir rezultatai būtų įrašomi į skirtingus failus, tai prieš vykdant programą reikia pakeisti rezultatų failo vardo konstantos reikšmę.

- Pagalvokite, kokių dar galėtų pasitaikyti situacijų, ir paruoškite atitinkamus duomenų rinkinius.



## Programos papildymas

- Patikrinkite, kaip dirba programa, visiems klasės mokiniams pasirinkus tikybos dalyką. Ką matysite rezultatų faile? Jei reikia, patikslinkite programą.
- Papildykite spausdinimo funkciją taip, kad spausdinamo sąrašo eilutės būtų numeruojamos pradedant vienetu.
- Pakeiskite programą taip, kad visi trys sąrašai būtų skaitomi iš atskirų duomenų failų.



## Užduotys

### 1. Mokesčiai

Pradinių duomenų faile yra informacija apie gyventojus ir jų mokami mokesčiai už komunalines paslaugas: nurodyta pavardė ir vardas (skiriama 20 pozicijų), buto numeris, mokestis už šilumą, elektrą, vandenį (teigiamieji skaičiai – permoka, neigiamieji skaičiai – skola).

Kitame pradinių duomenų faile yra tokio pat tipo informacija apie kai kurių asmenų padarytas įmokas.

Parenkite programą, kuri pirmame rezultatų faile pateiktų namo gyventojų sąrašą, sutvarkytą pagal įmokas. Į antrą rezultatų failą reikia surašyti lentelę gyventojų, kurie liko skolingi bent už vieną patarnavimą, duomenis.

Pradiniai duomenys				
5				
Pirmasis Petras	15	5.5	0.0	-54.0
Anrutė Jurgita	13	-25.05	2.5	-13
Katytė Pilkutė	1	0.0	0.0	0.0
Katinas Batuotas	16	-5.5	-18.2	-45.0
Basas Algis	14	2.2	2.2	2.2
3				
Pirmasis Petras	15	0.0	0.0	14.0
Katytė Pilkutė	1	3.0	0.0	3.0
Basas Algis	14	15.0	0.2	12.2

Rezultatai				
Pavardė, vardas	Buto Nr.	Šildymas	Elektra	Vanduo
Pirmasis Petras	15	5.50	0.00	-30.00
Anrutė Jurgita	13	-25.05	2.50	-13.00
Katytė Pilkutė	1	3.00	0.00	3.00
Katinas Batuotas	16	-5.50	-18.20	-45.00
Basas Algis	14	17.20	2.40	14.40

Pavardė, vardas	Buto Nr.	Šildymas	Elektra	Vanduo
Pirmasis Petras	15	5.50	0.00	-30.0
Anrutė Jurgita	13	-25.05	2.50	-13.00
Katinas Batuotas	16	-5.50	-18.20	-45.00

## 2. Prekės

Pradinių duomenų faile pateikti duomenys apie parduotuvėje esančias prekes: prekės pavadinimas (skiriama 20 pozicijų), pagaminimo data, vartojimo laikas dienomis. Parenkite programą, kuri į rezultatų failą surašytų lentelę duomenis apie prekes, pagamintas per nurodytą laikotarpį (laikotarpio duomenys įvedami klaviatūra). Sąrašas turi būti surikiuotas pagal prekės pavadinimą ir vartojimo laiką. Lentelės pabaigoje reikia nurodyti sąrašė esančių prekių vartojimo laiko aritmetinį vidurkį.

Pradiniai duomenys	
5	
Virtos bulvės	2011 01 15 2
Mėgėjų salotos	2011 01 10 1
Sūris su aguonomis	2011 01 22 15
Duona su aguonomis	2010 12 20 20
Rauginti agurkai	2010 09 15 100

Rezultatai		
Nurodytas laikotarpis: 2010 12 15 - 2011 01 15		
Prekės pavadinimas	Pagaminimo Data	Vartojimo laikas
Duona su aguonomis	2010 12 20	20
Mėgėjų salotos	2011 01 10	1
Virtos bulvės	2011 01 15	2
Vartojimo trukmės vidurkis: 7.67		

## 3. Viešbučio gyventojai

Pirmoje pradinių duomenų failo eilutėje yra viešbučio gyventojų skaičius. Kitose eilutėse pateiktas viešbučio gyventojų sąrašas: nurodyta pavardė ir vardas (skiriama 20 pozicijų), atvykimo data, kiek parų gyveno, paros kaina.

Parenkite programą, kuri paruoštų atskaitą: nurodytą pavardę, vardą, atvykimo datą, išvykimo datą, mokestį. Sąrašą, surikiuotą mažėjančiai pagal mokestį, reikia įrašyti į failą lentelę. Lentelės pabaigoje turi būti nurodyta pinigų suma, kurią sumokėjo visi gyventojai.

Pradiniai duomenys	
3	
Pirmasis Petras	2010 12 25 15 25.5
Anrutė Jurgita	2011 01 15 65 24.36
Pilkoji Pelytė	2010 11 01 155 0.1

Rezultatai			
Pavardė, vardas	Atvyko	Išvyko	Sumokėjo
Anrutė Jurgita	2011 01 15	2011 03 20	1583.40
Pirmasis Petras	2010 12 25	2011 01 8	382.50
Pilkoji Pelytė	2010 11 01	2011 04 05	15.50
-----			
Viešbučio pajamos: 1981.40			

#### 4. Abonentai

Telefono ryšių stotis mėnesio pabaigoje formuoja dokumentą, kuriame yra duomenys apie abonentus: pavardė ir vardas (skiriama 20 pozicijų), telefono numeris (su kuo kalbėta), miestas, prakalbėtas laikas (minutėmis). Pradiniai duomenys surašyti faile: pirmoje eilutėje nurodytas duomenų skaičius, toliau pateikti dokumento duomenys. Atskirame pradinių duomenų faile yra kainoraštis: miestas, minutės kaina. Pirmoje failo eilutėje nurodytas kainoraščio įrašų skaičius.

Parente programą, kuri suformuotų ir surašytų abėcėliškai mokėjimo ataskaitą: vardas, pavardė, pinigų suma, kurią reikia mokėti. Ataskaitos pabaigoje nurodykite visų abonentų pinigų sumą už prakalbėtą laiką.

Pradiniai duomenys			
5			
Pirmasis Petras	868999999	Klaipėda	12
Anrutė Jurgita	868999999	Klaipėda	24
Jurgutis Jurgis	888824242	Vilnius	12
Anrutė Jurgita	758455547	Vilnius	15
Rimutis Rimas	131231133	Raseiniai	12
7			
Ariogala	12.5		
Baisogala	6.25		
Klaipėda	10.89		
Kaunas	5.5		
Raseiniai	6.5		
Vilnius	5.68		
Voskoniai	7.8		
Rezultatai			
Pavardė, vardas	Mokėti		
-----			
Anrutė Jurgita	346.56		
Jurgutis Jurgis	68.16		
Pirmasis Petras	130.68		
Rimutis Rimas	78.00		
-----			
Pajamos:		407.40	



## 2.1. Konstantos

Programa atlieka veiksmus su duomenimis (pvz.: sveikaisiais ir realiaisiais skaičiais; simboliais; simbolių eilutėmis). Duomenys gali būti pastovūs ir kintami. Duomuo, kuris atliekant programą nekinta, vadinamas *konstanta*. Jos aprašymas pradodamas žodžiu `const`. Po to nurodomas konstantos reikšmės tipas, vardas ir reikšmė:

```
const Tipas Vardas = reikšmė;
```

Konstantos vardą rekomenduojame pradėti raide `C`, kad programos tekste būtų galima lengviau atpažinti, kurie vardai yra kintamųjų, o kurie – konstantų. Kai kurie kompiliatoriai (pvz., *Microsoft Visual C++ 2005/2008*) programoje leidžia naudoti konstantų, kintamųjų, funkcijų vardus su diakritiniais ženklais.

Konstantų aprašymo pavyzdžiai:

```
const int Cn = 100;
const double Cd = 100.15;
const bool Cb = false;
const char Cs = 'A';
const string Ce = "C++ programavimo kalba";
const char Ce[] = "C++ programavimo kalba";
const int CA[] = {25, 13, 4, 9};
```

*Microsoft Visual Studio 2005* (32 bitų procesoriaus) baziniai duomenų tipai ir jų ribinės reikšmės

Tipo pavadinimas	Galimos reikšmės arba jų intervalas	Atminties dydis
<code>bool</code>	true (1), false (0)	1 baitas
<code>char</code>	-128 .. 127	1 baitas
<code>unsigned char</code>	0 .. 255	1 baitas
<code>int</code>	-2147483648 .. 2147483647	4 baitai
<code>short</code>	-32768 .. 32767	2 baitai
<code>long</code>	-2147483647 .. 2147483647	4 baitai
<code>unsigned long</code>	0 .. 4294967295	4 baitai
<code>unsigned short</code>	0 .. 65535	2 baitai
<code>unsigned int</code>	0 .. 4294967295	4 baitai
<code>float</code>	$-1.18 \times 10^{-38}$ .. $3.4 \times 10^{38}$	4 baitai
<code>double</code>	$-2.23 \times 10^{-308}$ .. $1.79 \times 10^{308}$	8 baitai

Jei programoje naudojama konstanta, kurios duomenų tipą sukūrė programuotojas, tai pirma aprašomas duomenų tipas, po to – konstanta. Pavyzdžiui:

```
struct MM {
    int kk;
    double rr;
};
const MM Cu = {15, 25.5};
const MM CAu[] = {15, 13.13, 14, 456.2, 9, 9.9};
```

Jeigu masyvo tipo konstantos apraše už vardo tarp laužtinių skliaustų masyvo dydis nenurodytas, tuomet jis būna toks, kiek reikšmių pateikta tarp riestinių skliaustų (reikšmės viena nuo kitos skiriamos kableliais). Jei masyvo dydis nurodytas, tai reikia pateikti ir atitinkamą skaičių reikšmių. Reikšmės numeruojamos iš eilės, pradedant nuo 0. Veiksmuose naudojamos masyvo tipo konstantų reikšmės indeksuojamos taip pat, kaip ir masyvo tipo kintamųjų. Pavyzdžiui:

```
const int CA[] = {25, 13, 4, 9};
const int CB[5] = {25, 13, 4, 9, 0};
```

Kai konstanta yra struktūros tipo, jos reikšmės pateikiamos tokia eilės tvarka, kokia struktūroje aprašyti kintamieji. Pavyzdžiui, konstantų  $C_u$  ir  $C_{Au}$  apraše poromis pateikti sveikieji ir realieji skaičiai, atitinkantys struktūros  $MM$  kintamuosius  $kk$  ir  $rr$ .

Sukurta konstanta (kintamasis, duomenų tipas ir kt.) galioja nuo jos aprašymo vietos.

Programoje konstantų reikšmių keisti negalima.

## 2.2. Operatoriai

Su duomenimis atliekami veiksmai vadinami *operacijomis*. Operacijos būna: aritmetinės, loginės, lyginimo ir kt. Jos žymimos sutartiniais simboliais, vadinamaisiais *operatoriumis* (pvz.: +, %, ++, != ir kt.). Operatorių argumentai vadinami *operandais*. Jais gali būti konstantos, kintamieji, kreipiniai į funkcijas, reiškiniai.

Operatoriai gali būti *vienanariai* arba *dvinariai*. Vienanariai operatoriai turi vieną operandą, kuris gali būti operatoriaus kairėje arba dešinėje pusėje (pvz.: ++i, i++). Dvinariai operatoriai turi du operandus: vienas jų yra kairiojo, kitas – dešiniojo operando pusėje (pvz.: a + b, x += y).

Aptarsime kai kuriuos dažnai naudojamus operatorius ir jų paskirtį.

C++ kalboje sveikojo tipo kintamųjų reikšmės galima didinti arba mažinti vienetu naudojantis atitinkamai operatoriais ++ ir --.

Operatorius	Paskirtis
++operandas	Operando reikšmė didinama vienetu, po to naudojama
--operandas	Operando reikšmė mažinama vienetu, po to naudojama
operandas++	Operando reikšmė naudojama, po to didinama vienetu
operandas--	Operando reikšmė naudojama, po to mažinama vienetu

### Pavyzdžiai

Sakinys	Ekvivalentus sakiny	Pavyzdys
x++;	x = x + 1;	b = ++a; // a reikšmė didinama vienetu, o po to priskiriama b
x--;	x = x - 1;	b = a--; // a reikšmė priskiriama b, o po to mažinama vienetu

Reiškinyje gali būti daug skirtingų operatorių, todėl svarbu žinoti, kokia eilės tvarka reiškinyje atliekami veiksmai. Ją nurodo operatorių *prioritetas* (operacijų atlikimo pirmenybė) ir skliaustai.

### Operatorių prioritetai

Prioritetas	Operatoriaus simbolis	Operatoriaus pavadinimas	Asociatyvumas
1	::	Priklausomybės	→
2	. -> ( ) [] ++ --	Struktūrinio objekto komponento tiesioginis elemento išrinkimas Struktūrinio objekto komponento netiesioginis elemento išrinkimas Funkcija Masyvas Priešdėlinis reikšmių didinimas vienetu Priešdėlinis reikšmių mažinimas vienetu	→
3	! ++ -- + -	Loginis neigimas Priesaginis reikšmių didinimas vienetu Priesaginis reikšmių mažinimas vienetu Vienanaris pliusas Vienanaris minusas	←

	& * new delete tipo vardas	Adresas Rodyklė Atminties išskyrimas Atminties išlaisvinimas Tipo keitimas	
4	* / %	Daugyba Dalyba Sveikųjų skaičių dalybos liekana	→
5	+ -	Sudėtis Atimtis	→
6	<< >>	Postūmis į kairę Postūmis į dešinę	→
7	< <= >= >	Mažiau Mažiau arba lygu Daugiau arba lygu Daugiau	→
8	== !=	Lygu Nelygu	→
9	&&	Loginis IR	→
10		Loginis ARBA	→
11	?:	Sąlygos operacija	←
12	= * = / = % = + = - = & = << = >> = ^ =   =	Paprastas ir sudėtinis priskyrimai	←

*Pastaba.* Ženklas → parodo vieno tipo operacijų atlikimo kryptį iš kairės į dešinę, ženklas ← – iš dešinės į kairę.

Visi operatorių simboliai, išskyrus [ ] ( ) ? : , atsižvelgiant į kontekstą, gali reikšti skirtingus operatorius. Pavyzdžiui, dvinaris operatorius & atitinka bitų laukų loginę daugybą, o vienas operatorius & – adreso operatorių.

Pirmasis prioritetas – aukščiausias. Vienodo prioriteto operatorių atlikimo tvarka ta pati, jie vykdomi pagal asociatyvumo taisykles.

Jei tas pats operatorius programos tekste atitinka vienanarį ir dvinarį operatorių, tai pirmiausia vykdomas vienanaris operatorius, o po to – dvinaris.

Kompilatorius pagal operatoriaus simbolį ir operandų tipus nustato, ar galima atlikti operaciją. Jei taip, tai operacija įvykdoma ir gaunamas apibrėžto tipo rezultatas.

Operatoriai klasifikuojami pagal įvairius požymius: operandų kiekį, paskirtį, rezultatų tipą ir kt. Pagal paskirtį operatoriai yra: aritmetiniai (+, -, \*, /, %), lyginimo (<, >, <=, >=, ==, !=), priskyrimo (=, \*=, +=, -=, /=, %=), loginiai (&&, ||, !), bitų laukų (&, |, ^, ~), postūmio (<<, >>) ir kt.

*Aritmetiniai operatoriai* (+, -, \*, /) naudojami su sveikosiomis ir realiosiomis reikšmėmis. Dalybos liekanos operatorius (%) naudojamas tik su sveikosiomis reikšmėmis. Aritmetinio operatoriaus rezultatas yra realusis, kai nors vienas operandas realusis. Vienodo prioriteto aritmetiniai operatoriai atliekami iš eilės. Pavyzdžiui, įvykdžius programos fragmentą

```
double y;
int x = 2;
y = 2.0 / 3 + x * x - (2 * x - 5);
```

kintamasis y įgis reikšmę, lygią 5.66667.

*Lyginimo operatorių* == (ar lygu) ir != (ar nelygu) operandai turi būti diskretieji dydžiai, nes realieji skaičiai yra lyginami tam tikru tikslumu.

Operatoriai == ir != turi žemesnį prioritetą negu kiti lyginimo operatoriai (<, <=, >, >=). Pavyzdžiui, įvykdžius programos fragmentą

```
bool y;  
int x = 2;  
y = x > 2 && x != 0;
```

kintamasis y įgis reikšmę, lygią 0 (false kompiuteryje visuomet yra 0, true - 1).

Priskyrimo operatoriai yra dveji: paprastojo priskyrimo (=) ir sudėtinio (\*=, /=, %=, +=, -=).

Sudėtinio priskyrimo operatoriais nurodoma, kad rezultato kintamasis turi būti ir dešiniojoje priskyrimo operatoriaus pusėje. Sudėtinio priskyrimo sakinio sintaksė yra tokia:

```
kintamasis operatorius= reiškinys;  
ekvivalentu  
kintamasis = kintamasis operatorius reiškinys;
```

Sudėtinio priskyrimo operatoriaus naudojimo pavyzdžiai:

```
i *= (i1 + i2); // ekvivalentu i = i * (i1 + i2);  
x += y; // ekvivalentu x = x + y;
```

Struktūrinio objekto komponentas parenkamas operatoriumi . (taškas).

Pavyzdžiui, įvykdžius programos fragmentą

```
struct Studentas {  
    string vardas;  
    int amzius;  
    double vidurkis;  
};  
Studentas stud;  
stud.amzius = 20;  
cout << stud.amzius << endl;
```

bus išspausdinta reikšmė 20.

Kartais vieną duomenų tipą reikia pakeisti kitu (pvz., realųjį skaičių reikia paversti sveikuju arba, atvirkščiai, sveikąjį skaičių reikia pakeisti realiuoju). Tam naudojami *duomenų tipo keitimo operatoriai*. Jų sintaksė tokia:

```
(tipas) operandas  
tipas (operandas)
```

Operandu gali būti kintamasis, konstanta arba reiškinys.

Pavyzdžiui, įvykdę programos fragmentą

```
int a = 5, b = 7;  
double c = a / b; // dalybos rezultatas yra sveikasis skaičius  
double d = (double) a / b; // dalybos rezultatas yra realusis skaičius  
cout << c << " " << d << endl;
```

ekrane matysime

```
0 0.714286
```

Duomenų tipą reikia keisti atsargiai, nes gali pasikeisti skaitinė operando reikšmė.

## 2.3. Kintamasis, rodyklė, adresas

Kiekvienas programos kintamasis kompiuterio atmintyje užima tam tikrą skaičių baitų. Tai priklauso nuo kintamojo tipo (pvz., kiekvienam kintamajam tipo char kompiuterio atmintyje laikyti skiriamas vienas baitas). Kompiuteryje visos atminties ląstelės (baitai) turi adresus, koduojamus šešioliktainiais skaičiais. Kintamojo reikšmės vieta atmintyje nusakoma jos vietos pirmojo baito adresu. Programuotojas programoje rašo kintamųjų vardus. Kompiliatorius visuomet sukuria kintamųjų ir jų adresų lentelę, pagal kurią paruoštoje vykdyti programoje kintamųjų vardai keičiami jų reikšmių adresais kompiuterio atmintyje.



Programuotojas gali susikurti kintamąjį (-uosius), kuris atmintyje laiko ne reikšmės, o kitų kintamųjų adresus. Sakoma, kad toks kintamasis parodo, kur yra tam tikra reikšmė, todėl jis vadinamas *rodyklės tipo kintamuoju*, arba tiesiog *rodyklė*.

Rodyklės tipo kintamasis aprašomas kartu su ženklu `*`.

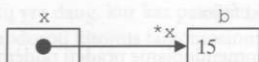
*Pavyzdys*

```
int *x; // aprašomas kintamasis reikšmės kompiuterio atmintyje adresu laikyti (int tipo reikšmė)
int b = 15; // sveikojo tipo kintamajam suteikiama reikšmė, lygi 15
x = &b; // rodyklės tipo kintamajam x priskiriamas kintamojo b reikšmės kompiuterio atmintyje adresas
cout << x // ekrane parodoma kintamojo x reikšmė (kintamojo b reikšmės kompiuterio atmintyje adresas)
    << endl
    << *x // ekrane parodoma reikšmė, kurią nurodo į kintamąjį x įrašytas adresas
    << endl
    << b // ekrane parodoma kintamojo b reikšmė
    << endl
    << &b // ekrane parodoma kintamojo b reikšmės kompiuterio atmintyje adresas
    << endl;
```

Ekrane matysite, pavyzdžiui, tokį vaizdą:

```
0012FE94
15
15
0012FE94
```

Visa tai grafiškai galima pavaizduoti taip:



Čia rodykle vaizduojamas adresas, kuris parodo, kur kompiuterio atmintyje yra rodyklės (kintamojo) rodoma reikšmė.

Rodyklės tipo kintamiesiems rekomenduojama suteikti pradinę nulinę reikšmę (konstantą `NULL`). Tai specialiai parinkta skaitinė reikšmė, nesutampanti su jokia atminties adresu.

*Pavyzdžiui:*

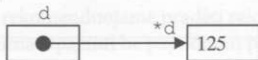
```
int *x;
x = NULL;
```

Rodyklės tipo kintamieji, kurių reikšmių vietai kompiuterio atmintyje išskirti naudojamas operatorius `new`, vadinami *dinaminiais kintamaisiais*.

*Pavyzdys*

```
int *d; // aprašomas rodyklės tipo kintamasis adresu atmintyje laikyti (int tipo reikšmė)
// Kompiuterio atmintyje išskiriama vieta sveikojo tipo dinaminiam kintamajam (*d)
// ir šio kintamojo adresas priskiriamas rodyklės tipo kintamajam d
d = new int;
*d = 125; // rodyklės tipo kintamojo laikomu adresu įrašoma reikšmė, lygi 125
cout << *d << endl // ekrane parodomas dinaminio kintamojo d reikšmė
    << d << endl; // ekrane parodomas rodyklės d laikomas adresas
```

Šio pavyzdžio situaciją grafiškai galima pavaizduoti taip:



## 2.4. Duomenų skaitymas iš failo

Kai pradinių duomenų daug, juos įvedinėti klaviatūra nėra patogiu. 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 skaityti iš failo, reikia:

- ▼ Aprašyti įvesties srauto `ifstream` kintamąjį, pavyzdžiui,

```
ifstream fd;
```

- ▼ Programoje jį susieti su tekstiniu failu, pavyzdžiui,

```
fd.open("Duomenys.txt");
```

Galima įvesties srauto `ifstream` kintamąjį aprašyti ir susieti su tekstiniu failu vienu sakiniu, pavyzdžiui,

```
ifstream fd("Duomenys.txt");
```

Programoje `ifstream` tipo kintamasis, susietas su konkrečiu duomenų srautu (failu), vadinamas *įvesties srautu*.

- ▼ Baigus darbą, srautą būtina užverti, pavyzdžiui,

```
fd.close();
```

Duomenų failas turi būti darbiname kataloge (tame pačiame, kaip ir programos teksto failas). Jei failas yra darbinio katalogo pakatalogyje, failo vardą reikia nurodyti su pakatalogio vardu, pavyzdžiui,

```
fd.open("Duom/Duomenys.txt");
```

Kitais atvejais reikia nurodyti visą kelią iki failo.

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

Failo vardas programoje gali būti nurodomas simbolių eilutės tipo konstanta arba kintamuoju.

*Pavyzdys*

Failo vardas – konstanta	Failo vardas – kintamojo reikšmė
<pre>const char CDfv[] = "Duomenys.txt"; ifstream fd(CDfv);</pre>	<pre>char fVardas[] = "Duomenys.txt"; ifstream fd(fVardas);</pre>

Čia `fd` yra duomenų srauto kintamasis. Jis susiejamas su failu, kurio vardą laiko eilutės tipo konstanta `CDfv` arba kintamasis `fVardas`.

Kaip ir įvedant klaviatūra, taip ir skaitant duomenis iš failo naudojamas tas pats operatorius `>>`. Tik įvesties srauto vardas `cin` yra keičiamas įvesties srauto, susijusio su duomenų failu, kintamojo vardu, pavyzdžiui,

```
fd >> x;
```

Įvesties ir išvesties srautų, susijusių su duomenų ir rezultatų failais, priemonės aprašytos antraštiniame faile `fstream`. Jis įkeliamas į programą tokiu sakiniu:

```
#include <fstream>
```

Pateikiame pavyzdį, kai iš tekstinio failo *Duomenys.txt* perskaitomi du sveikieji skaičiai ir ekrane parodoma perskaitytų skaičių suma.

*Pavyzdys*

*Duomenys.txt*

```
15 134
```

```
int main()
{
    int a, b, s;
    ifstream fd("Duomenys.txt");
    fd >> a >> b;
    fd.close();
    s = a + b;
    cout << s << endl;
    return 0;
}
```

Kai kurios darbo su įvesties srautais funkcijos

Funkcijos prototipas	Paaiškinimai
void <b>ignore</b> (max, skyriklis);	Iš srauto paima ir pašalina iki max simbolių arba iki nurodyto simbolio skyriklis imtinai (iš anksto numatytas '\n')
int <b>peek</b> ();	Iš srauto nukopijuoja, grąžina vieną int tipo simbolį ir palieka jį sraute. Grąžina true, jeigu aptikta srauto (failo) pabaiga
bool <b>eof</b> ();	Grąžina true, jeigu nustatyta srauto pabaiga, kitaip grąžina false
bool <b>good</b> ();	Grąžina true, jeigu nebuvo klaidos įvedimo metu, kitaip grąžina false

## 2.5. Rezultatų (duomenų) rašymas į failą

Akivaizdu, kad skaičiavimų rezultatus ekrane galima parodyti, jei jų yra nedaug. Tuomet nesunku juos peržiūrėti ir / ar nusrašyti. Tačiau jei rezultatų yra daug, kur kas patogiau juos įrašyti į tekstinį failą. Tada duomenis paprasčiau panaudoti dokumentuose, apdoroti kitomis taikomosiomis programomis (pvz., skaičiuokle).

Norint duomenis įrašyti į failą, reikia:

- ✓ Aprašyti išvesties srauto `ofstream` kintamąjį, pavyzdžiui,

```
ofstream fr;
```

- ✓ Programoje išvesties srauto kintamąjį susieti su tekstiniu failu, pavyzdžiui,

```
fr.open("Rezultatas.txt");
```

Galima išvesties srauto `ofstream` kintamąjį aprašyti ir susieti su tekstiniu failu vienu sakiniu, pavyzdžiui,

```
ofstream fr("Rezultatas.txt");
```

Programoje `ofstream` tipo kintamasis, susietas su konkrečiu duomenų srautu (failu), vadinamas **išvesties srautu**.

Jei vykdančią programą darbiname kataloge nurodytu vardu tekstinis failas jau yra, tai jame esantys duomenys pašalinami. Priešingu atveju sukuriamas naujas failas.

Jei failą norima sukurti ne darbiname kataloge, būtina nurodyti kelią iki failo, pavyzdžiui,

```
fr.open("C:/Rez/Rezultatai.txt");
```

Norint jau esantį rezultatų failą papildyti, reikia jį atverti, pavyzdžiui, taip:

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

Tuomet esami duomenys išlieka, o failas paruošiamas naujiems duomenims įrašyti į failo pabaigą.

- ✓ Baigus darbą, srautą būtina užverti, pavyzdžiui,

```
fr.close();
```

Visų išvesties srauto kintamųjų vardus rekomenduojame pradėti raidėmis `fr` (raidė `f` reiškia failą, `r` – rezultatus). Tuomet programos tekste juos galima atpažinti be papildomų paaiškinimų.

Failo vardas programoje gali būti nurodomas simbolių eilutės tipo konstanta arba kintamuoju.

Failo vardas – konstanta	Failo vardas – kintamojo reikšmė
const char CRfv[] = "Rezultatas.txt"; ofstream fr(CRfv);	char fVardas[] = "Rezultatas.txt"; ofstream fr(fVardas);

Čia `fr` yra duomenų srauto kintamasis. Jis susiejamas su failu, kurio vardą laiko eilutės tipo konstanta `CRfv` arba kintamasis `fVardas`.

Kaip ir rodant ekrane, taip ir rašant duomenis į failą naudojamas tas pats operatorius `<<`. Tik srauto vardas `cout` yra keičiamas išvesties srauto, susijusio su duomenų failu, kintamojo vardu, pavyzdžiui,

```
fr << x;
```

Įvesties ir išvesties srautų, susijusių su duomenų ir rezultatų failais, priemonės aprašytos antraštiniame faile `fstream`. Jis įkeliamas į programą tokiu sakiniu:

```
#include <fstream>
```

Pateikiame pavyzdį, kai į tekstinį failą *Rezultatas.txt* įrašoma dviejų sveikųjų skaičių, įvestų klaviatūra, suma.

#### Pavyzdys

```
#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;
}
```

Įvykdę programą ir klaviatūra įvedę skaičius 49 ir 100, rezultatų faile *Rezultatas.txt* matysite:

```
149
```

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

```
#include <iomanip>
```

#### Dažniausiai naudojami manipuliatoriai

Manipuliatorius	Paaiškinimai
<b>left</b>	Išvedamus duomenis lygiuoja pagal kairįjį kraštą
<b>right</b>	Išvedamus duomenis lygiuoja pagal dešinįjį kraštą
<b>endl</b>	Į srautą nusiunčia eilučių skyriklio simbolį <code>'\n'</code> ir išvalo buferį
<b>setw</b> (int n)	Nustato išvedamų duomenų lauko plotį n
<b>setprecision</b> (int n)	Nustato realiojo skaičiaus išvedamų skaitmenų skaičių n Jeigu skaičiaus sveikąjoje dalyje yra daugiau skaitmenų, tuomet jis vaizduojamas standartinė išraiška, kurioje yra daugiklis $10^n$ . Laipsnio pagrindas žymimas simboliu <code>e</code> ir po jo rašomas laipsnio rodiklis. Pavyzdžiui, užrašas $3.1416e+006$ atitinka skaičių $3.14 \cdot 10^6$ Jeigu prieš manipuliatorių <code>setprecision</code> rašomas manipuliatorius <code>fixed</code> , tuomet n nustato, kiek realiojo skaičiaus trupmeninės dalies skaitmenų reikia rašyti, įskaitant ir nulius skaičiaus gale, pavyzdžiui, <code>3141592653.400</code>

Išvesties sraute galima nurodyti, kiek pozicijų skirti atskiriems duomenims. Nurodymai rašomi prieš išvedamus duomenis.

Pavyzdys

```
int a = 45;
double b = 123.258;
fr << a;
fr << setw(4) << a << " "
  << fixed << setprecision(2) << b;
```

Faile matysite:

```
 4 5 1 2 3 2 6
```

Realiojo skaičiaus trupmeninei daliai buvo nurodyta per mažai pozicijų, todėl pateikti tik du skaitmenys po kablelio, paskutinis – suapvalintas.

## 2.6. Funkcijos

*Funkcija* yra vadinama programos konstrukcija, kuri atlieka savarankiškus veiksmus. Vykdamt programą, į funkcijas galima kreiptis daug kartų. Funkcijos padeda struktūrizuoti programą. Tokią programą lengviau skaityti ir analizuoti.

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

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

```
rezultatoTipas funkcijosVardas(formaliejiParametrai);
```

Pavyzdžiui, `int Suma(int a, int b);`.

Funkcijos realizavimo aprašas (funkcijos antraštė ir veiksmai, kuriuos ji turi atlikti) pateikiamas už funkcijos `main()`. Jei funkcijos realizavimo aprašas yra prieš `main()`, tuomet nereikia rašyti funkcijos prototipo.

Funkcijos aprašas atrodo taip:

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

Funkcijos antraštėje pirmiausia nurodomas grąžinamos reikšmės tipas. Jei funkcija per savo vardą 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 parametru, tuomet rašomi tušti skliaustai.

Funkcijos duomenis su programos duomenimis susieja `formaliejiParametrai`. Parametrai apibrėžiami pagal tas pačias taisykles, kaip ir kintamieji: nurodant jų reikšmės tipą ir vardą. Vienas nuo kito parametrai atskiriami kableliais.

Veiksmai, kuriuos turi atlikti funkcija, nurodomi dalyje `funkcijosKamienas`. Jei funkcija skirta kokiai nors reikšmei grąžinti, tai tarp funkcijos veiksmų turi būti bent vienas sakiny, kuriuo apskaičiuota reikšmė būtų grąžinama į programą. Funkcijos kamienas gali būti aprašomi kintamieji, reikalingi funkcijos veiksmams atlikti. Jie galioja tik funkcijoje.

Reikšmei grąžinti per funkcijos vardą naudojamas sakiny `return`. Sakinio `return` sintaksė yra tokia:

```
return Reiškiny;
```

Reiškinys – tai bet koks reiškinys, kurio reikšmė grąžinama atlikus veiksmą. Grąžinamos reikšmės tipas ir funkcijos antraštėje nurodytas rezultato tipas turi būti tarpusavyje suderinti. Sakiniu `return` ne tik grąžinama nurodyta reikšmė, bet ir nutraukiamas funkcijos darbas. Sakiniu `return;` tiesiog nutraukiamas funkcijos darbas.

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

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

Jei funkcija grąžina apskaičiuotas reikšmes per parametrus (`void`), į ją kreipiamasi taip:

```
funkcijosVardas(faktiniaiParametrai);
```

Duomenis, su kuriais bus atliekami funkcijos veiksmai, nurodo `faktiniaiParametrai`. Tai gali būti reikšmės, kintamieji, reiškiniai.

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 rašomi 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;                 // per funkcijos vardą grąžinama apskaičiuota stačiakampio ploto reikšmė
}
```

Įvykdę programą, ekrane matysite:

```
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 tų pačių tipų kintamuosius, kaip ir perduodami parametrai, ir jiems priskiria parametrų reikšmes. Vadinasi, funkcija dirba su parametrų reikšmių kopijomis, bet ne su pačiais parametrais. Tai patogu, 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 adresus). Vadinasi, funkcija tiesiogiai naudoja perduodamus kintamuosius.

Pateikiame pavyzdį, kuriame funkcija `Sukeisti()`, naudodama parametrų vardus `pirmas` ir `antras`, faktiškai naudojami 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;
}
```

Ivykdę programą, ekrane matysite:

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

## 2.7. Masyvas

### Masyvo duomenų struktūra

*Masyvas* – viena pagrindinių duomenų struktūrų, naudojamų programavimo kalbose. Ši struktūra leidžia laikyti atmintyje (vienoje vietoje) vienu vardu daug to paties tipo reikšmių. Pavyzdžiui, žinomos vieno mėnesio pokalbių telefonu trukmės minutėmis. Reikia apskaičiuoti, kiek iš viso minučių prakalbėta, ilgiausio pokalbio trukmę, ilgiausių pokalbių skaičių ir kt. Rašant programą bet kurio mėnesio pokalbių telefonu trukmėms apdoroti, negalima žinoti, kiek pokalbių bus tą mėnesį. Todėl negalima iš anksto numatyti, kiek kintamųjų reikia pokalbių trukmėms laikyti. Be to, atskirų pokalbių trukmėms naudojant skirtingus kintamuosius, programa bus gremėzdiška ir neuniversali (kito mėnesio duomenims apdoroti programą reikės taisyti). Masyve visas reikšmės galima laikyti vienu vardu. Reikšmių skaičius masyve gali būti skirtingas.

Masyvo aprašyme nurodomas elementų reikšmių tipas, vardas ir dydis:

```
int A[10];
```

Masyvo dydis  
Masyvo vardas  
Masyvo elementų reikšmių tipas

Naudojantis masyvu, reikia prisiminti, kad:

- ✓ Visos masyvo elementų reikšmės yra to paties tipo.
- ✓ Norint reikšmę įrašyti į masyvą arba paimti iš jo, reikia nurodyti reikšmės masyve eilės numerį. Tas numeris vadinamas *indeksu*. Jis rašomas už masyvo vardo laužtiniuose skliaustuose.
- ✓ Masyvų elementai vadinami *indeksuotaisiais kintamaisiais*. Jie reiškiniuose ir kitose duomenų struktūrose naudojami taip pat, kaip ir paprasti kintamieji.

Paprasciausiai atveju masyvą galima įšivaizduoti kaip vienodo tipo reikšmių, sunumeruotų iš eilės ir turinčių tą patį vardą, seką. C++ kalboje masyvų elementai pradedami numeruoti nuo nulio.

### Pavyzdys

Masyvo A elementų reikšmės atmintyje	15	-3	18	45	25	-4524	-75
Elementų indeksai	0	1	2	3	4	5	6
Kreipinys į masyvo A elementus programavimo kalboje C++	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]

Elemento indeksas gali būti sveikasis skaičius, sveikąjo tipo kintamasis arba reiškinys, kurio rezultatas yra sveikasis skaičius. Pavyzdžiui:

```
A[12]; A[i]; A[i + 2]; B[k + 6 * i];
```

Masyvo elementų reikšmių tipas gali būti bazinis (int, double, char ir kt.) arba programuotojo sukurtas.

Masyve galima laikyti ne daugiau reikšmių, negu nurodyta jo apraše. Pavyzdžiui, aprašas `int A[10];` parodo, kad masyve A galima laikyti 10 skaičių: pirmąjį – nuliniame elemente (A[0]), antrąjį – pirmame elemente (A[1]) ir t. t. Paskutinio elemento indeksas – vienetu mažesnis už masyvo dydį (9). Todėl rašant programas labai svarbu sekti, kad indekso reikšmė visuomet būtų intervale [0, masyvoDydis - 1].

Programoje galima aprašyti ir naudoti keletą skirtingo tipo ir skirtingo dydžio masyvų. Jų vardai programos bloke turi būti skirtingi.

Masyvo dydis apraše nurodo kompiliatoriui, kiek vietos atmintyje reikia skirti masyvo elementų reikšmėms.

### Pavyzdys

```
int A[50], // masyvas, kuriame bus galima laikyti iki 50 sveikųjų skaičių
C[30]; // masyvas, kuriame bus galima laikyti iki 30 sveikųjų skaičių
double B[60]; // masyvas, kuriame bus galima laikyti iki 60 realiųjų skaičių
```

Aiškinamajame tekste (jei jis rašomas) masyvo dydis paprastai nurodomas už masyvo vardo paprastuosiuose skliaustuose. Pavyzdžiui:

```
A(50), C(30), P(n), D(n * m + 5)
```

Masyvo elementų reikšmės patogų apdoroti, kai kreipinyje į masyvo elementus indekso vietoje rašomas kintamasis. Pavyzdžiui:

```
int A[50];
for (int i = 0; i < 50; i++) A[i] = i * i;
```

Čia masyve bus surašyti (didėjančiai) skaičių nuo 0 iki 49 kvadratai. Skaičius surašyti mažėjančiai galima pakeltus ciklo antraštę:

```
for (int i = 49; i >= 0; i--) A[i] = i * i;
```

Aprašant masyvą, galima nurodyti jo elementų pradines reikšmes. Pavyzdžiui:

```
int A[5] = {50, 18, -3, 9, 10};
double B[4] = {1.3, 9.0, 18.34, 9.5};
```

Toks aprašas gali būti funkcijos viduje (lokalieji masyvai) arba už funkcijos `main()` ribų (globalieji masyvai).

Jei programoje masyvo elementams reikšmės nesuteikiamos, tai dažniausiai masyvui skirtoje atminties vietoje įrašomos atsitiktinės reikšmės, vadinamosios šiukšlės. Pavyzdžiui, masyvo `int A[5] = {8, 4, 15};` paskutinės dvi reikšmės A[3] ir A[4] yra šiukšlės.



Jei masyvo dydis nenurodomas, masyvui atmintyje skiriama tiek vietos, kiek masyvo reikšmių aprašyta:

```
int A[ ] = {6, 89, 21}; // masyvo A dydis bus lygus 3
```

Aprašyti masyvą nenurodant jo dydžio galima tik tada, kai apibrėžiamos masyvo elementų pradinės reikšmės. Taigi aprašas `int A[ ]`; negalimas.

Masyvo dydis paprastai nurodomas konstanta, kad prirėikus jos reikšmę būtų patogu keisti. Masyvo dydis programoje pasirenkamas toks, kad masyve tilptų didžiausias galimas skaičiavimuose naudojamų reikšmių skaičius. Žinoma, masyvas negali būti begalinis. Jo dydį riboja turimos atmintyje laisvos vietos dydis. Kai reikšmių labai daug ir jų visų vienu metu negalima laikyti masyve, duomenys apdorojami kitais būdais.

## Reikšmių priskyrimas masyvo elementams

Masyvo elementams pradinės reikšmės galima suteikti aprašant masyvą arba priskyrimo sakinius. Dažniausiai pradiniai duomenys laikomi failuose. Prieš atliekant skaičiavimus su duomenimis, reikia juos iš failo įkelti į masyvą.

Prieš skaitant duomenis iš failo, būtina išsiaiškinti, kaip jie surašyti į failą. Paprastai pirmasis į failą įrašytas skaičius nurodo reikšmių skaičių masyve. Tokiu atveju reikia perskaityti pirmąjį į failą įrašytą skaičių ir patikrinti, ar tiek reikšmių tilps programoje aprašytame masyve. Jei taip, tuomet skaitomos masyvo reikšmės iš failo. Priešingu atveju reikia numatyti galimus tolesnius veiksmus (pvz.: nutraukti programos darbą, perskaityti didžiausią galimą reikšmių skaičių, pasiūlyti naudotojui pakeisti masyvo reikšmių skaičių ir pan.), kurie priklauso nuo papildomų uždavinio sprendimo sąlygų.

### Pavyzdys

```
const int Cn = 100;           // masyvo dydis (elementų skaičius)
int A[Cn], n;                // masyvas ir jo reikšmių skaičius
//-----
// Masyvo D(k) reikšmės skaitomos iš failo, kurio vardas fv
bool Skaityti(char fv[], int D[], int &k)
{
    ifstream fd(fv);
    fd >> k;                  // perskaitomas pirmasis (masyvo reikšmių) skaičius
    if (k > Cn) return false; // masyve trūksta vietos
    for (int i = 0; i < k; i++)
        fd >> D[i];          // skaitomos masyvo reikšmės
    fd.close();
    return true;             // duomenys sėkmingai perkelti į masyvą
}
```

### Duomenų failo pavyzdys

```
12
15 -45 8 4 3 5 8 10
-1 -89 125 14 -85
```

Funkcijos parametrų sąrašė prieš masyvo tipo parametą nereikia rašyti jokių specialių simbolių, nes kreipiantis į funkciją tokiam parametrui visuomet perduodamas argumentas, t. y. nulinio masyvo elemento, adresas. Kreipinys į pavyzdyje pateiktą funkciją gali būti toks:

```
if (Skaityti("Duomenys.txt", A, n)) ...
```

## Masyvo reikšmių rašymas

Paprastiausiai masyvo reikšmės rašyti į failą arba rodyti ekrane.

### Pavyzdys

```
// Nuoseklus masyvo reikšmių rašymas
// Masyvo D(k) reikšmės rašomos į failą, kurio vardas fv
void Spausdinti(char fv[], int D[], int k)
{
    ofstream fr(fv);
    for (int i = 0; i < k; i++)
        fr << D[i] << " ";
    fr << endl;
    fr.close();
}
```

Kreipinys į pavyzdyje pateiktą funkciją gali būti toks:

```
Spausdinti("Rezultatai.txt", A, n);
```

Pavyzdyje nurodytos duomenų failo reikšmės į failą bus rašomos nuosekliai viena po kitos:

```
15 -45 8 4 3 5 8 10 -1 -89 125 14 -85
```

Jeigu išvesties srautas yra atvertas, tai jį galima perduoti funkcijai. Tokiu atveju funkcijos antraštėje vietoj failo nurodomas srauto vardas. Parametro tipas turi būti keičiamas į srauto tipą. Srauto kopijos perduoti funkcijai negalima, todėl perduodamas srauto adresas.

### Pavyzdys

```
// Nuoseklus masyvo reikšmių rašymas
// Masyvo D(k) reikšmės rašomos į srautą, kurio vardas srautas
void Spausdinti(ofstream & srautas, int D[], int k)
{
    for (int i = 0; i < k; i++)
        srautas << D[i] << " ";
    srautas << endl;
}
```

Čia srautas tik naudojamas. Užverti jį reikia tame programos bloke, kur jis buvo atvertas.

Kreipinys į pavyzdyje pateiktą funkciją galėtų būti toks:

```
Spausdinti(fr, A, n); // fr – išvesties srautas, susietas su failu
Spausdinti(cout, A, n); // cout – išvesties srautas, susietas su ekranu (tekstinis failas)
```

## Masyvo perdavimas funkcijai

Masyvo vardas laiko masyvo nulinio elemento adresą, kurio programiškai negalima keisti. Adresas rodo, kur kompiuterio atmintyje prasideda masyvo reikšmių sąrašas. Todėl sakoma, kad masyvo vardas yra rodyklė-konstanta į nulinį masyvo elementą.

Masyvai gali būti funkcijos parametrai. Kai funkcijos parametras yra rodyklė (masyvas), tai visi pakeitimai, atlikti su masyvo elementų reikšmėmis funkcijos kamiene, matomi ir funkcijos išorėje. Norint uždrausti keitimus masyvo viduje, parametras reikia aprašyti su baziniu žodžiu `const`.

Kai funkcijos parametras yra masyvas, tai funkcija iš tikrųjų turi du parametrus: patį masyvą ir masyvo reikšmių kiekį. Funkcijai neužtenka perduoti vieną parametras (patį masyvą). Reikia nurodyti ir masyvo reikšmių kiekį. Jo galima nepateikti tik tam tikrais atvejais, pavyzdžiui, kai masyvo reikšmių kiekis iš anksto žinomas ir niekada nesikeičia, kai jis įrašytas į nulinį masyvo elementą arba visada yra lygus masyvo dydžiui (elementų skaičiui).

Panagrinėkime dvi funkcijas: vieną masyvo reikšmėms įvesti klaviatūra, kitą – masyvo reikšmėms rodyti ekrane. Funkcijos, skirtos masyvo elementų reikšmėms įvesti, prototipas, pavyzdžiui, yra toks:

```
void Ivesti(int duom[], int & kiek);
```

Čia pirmasis parametras yra masyvas (jo požymis – laužtiniai skliaustai), antrasis – reikšmių kiekis masyve. Antrasis funkcijos parametras kartu yra ir nuoroda, nes tik funkcijos viduje galima sužinoti būsimą masyvo reikšmių kiekį, kurį būtinais reikės perduoti į funkcijos išorę.

Masyvo reikšmių išvedimo funkcijos prototipas gali būti toks:

```
void Isvesti(int duom[], int kiek);
```

Norint kreiptis į funkciją, kurios parametras yra masyvas, kreipinyje tereikia nurodyti masyvo vardą be jokių papildomų ženklų, pavyzdžiui:

```
Ivesti(Am, kiek);  
Isvesti(Am, kiek);
```

*Pavyzdys.* Pagrindinė programa ir jos funkcijos

```
// Masyvo reikšmių įvedimas klaviatūra  
// Masyvas – funkcijos parametras  
#include <iostream>  
using namespace std;  
//-----  
bool Ivesti(int Duom[], int & kiek);  
void Isvesti(int Duom[], int kiek);  
//-----  
const int Cilgis = 10;  
int main()  
{  
    int Am[Cilgis], Akiek;  
    bool sekme = Ivesti(Am, Akiek);  
    if (sekme)  
        Isvesti(Am, Akiek);  
    else {  
        cout << "Per daug elementų paprašėte įvesti" << Akiek << endl;  
        cout << "Galima tik " << Cilgis;  
    }  
    return 0;  
}  
//-----  
// Įvedamos klaviatūra masyvo Duom(kiek) reikšmės  
bool Ivesti(int Duom[], int & kiek)  
{  
    cout << "Įveskite elementų skaičių ";  
    cin >> kiek;  
    if (kiek > Cilgis)  
        return false;  
    cout << "Įveskite " << kiek << " masyvo elementus (-u) "  
        << endl;  
    for (int i = 0; i < kiek; i++)  
        cin >> Duom[i];  
    return true;  
}  
//-----  
// Rodomos ekrane masyvo Duom(kiek) reikšmės  
void Isvesti(int Duom[], int kiek)  
{  
    cout << "Masyvo elementai " << endl;  
    for (int i = 0; i < kiek; i++)  
        cout << Duom[i] << " ";  
    cout << endl;  
}
```

```
Įveskite elementų skaičių 5
Įveskite 5 masyvo elementus (-ų)
10 11 12 54 8
```

Realizuodami funkciją `Ivesti()`, šiek tiek pakeitėme jos antraštę. Dabar funkcija grąžina pranešimą `true`, jei duomenys įvesti į masyvą, arba `false`, jei neįvesti. Pagrindinėje funkcijoje tikrinama, ar pavyko įvesti. Tai leidžia lengviau aptikti programos klaidas. Programoje aprašytas masyvas yra fiksuoto dydžio. Programos darbo metu klaviatūra nurodoma, kiek reikšmių reikia įvesti į masyvą. Jeigu prašoma įvesti daugiau, nei gali tilpti masyve, tai funkcija `Ivesti()` grąžina reikšmę `false`. Tai rodo, kad į masyvą reikšmės nėra įvestos. Jeigu įvesties sraute yra daugiau reikšmių, nei gali tilpti masyve, tuomet veiksmams su visomis reikšmėmis atlikti būtina taikyti kitus duomenų apdorojimo algoritmus.

## 2.8. Simbolių eilutė `char []`

C++ programavimo kalbos abėcėlę sudaro šie simboliai (ženklai):

- 1) 26 lotynų abėcėlės didžiosios ir mažosios raidės;
- 2) 10 arabiškų skaitmenų (0, 1, 2, 3, 4, 5, 6, 7, 8, 9);
- 3) specialieji simboliai (" , { } | [ ] ( ) + - \* / \ ; : ' ? < > \_ ! & # ~ = % ^ \$ @ . )

Simbolių sekas (simbolių eilutes) patogu laikyti `char` tipo masyvuose. Jie aprašomi panašiai, kaip ir kitų tipų masyvai, pavyzdžiui:

```
char s1[10];
```

Skirtumas tik tas, kad už paskutinės reikšmės masyve įrašomas eilutės pabaigos simbolis – nulis ('`\0`'). Simbolių eilutės `char` tipo dažnai vadinamos *C eilutėmis*.

Aprašant eilutę galima priskirti reikšmę, pavyzdžiui:

```
char s2[] = "C++ kalba";
```

arba

```
char s2[12] = "C++ kalba";
```

Pirmuoju atveju sukuriamas tokio ilgio masyvas `s2`, kuriame galėtų tilpti kabutėse užrašyta simbolių seka ir eilutės pabaigos simbolis, t. y. 10 elementų masyvas. Antruoju atveju kompiuterio atmintyje sukuriamas nurodyto dydžio masyvas. Jo pradžioje įrašoma kabutėse nurodyta simbolių seka:

Eilutės <code>s2[]</code> simboliai	C	+	+		k	a	l	b	a	\0		
Eilutės <code>s2[12]</code> simboliai	C	+	+		k	a	l	b	a	\0		
Indeksai	0	1	2	3	4	5	6	7	8	9	10	11

Simbolių eilutėms apdoroti yra daug standartinių funkcijų. Jas sąlygiškai galima suskirstyti į tris grupes:

- 1) *Analizės funkcijos* – nekeisdamos eilutės randa kokių nors eilutės parametrų reikšmes.
- 2) *Konvertavimo ir kopijavimo funkcijos* – konvertuoja eilutę į skaičių arba kopijuoja ją į kitą eilutę.
- 3) *Modifikavimo funkcijos* – koku nors būdu keičia eilutės reikšmę.

Funkcijos, kurios suformuoja naują eilutę, išskyrus `strncpy()`, automatiškai įrašo jos pabaigos simbolį. Jei eilutė formuojama simbolis po simbolio, programuotojas pats turi įrašyti eilutės pabaigos simbolį. Naudojantis modifikavimo funkcijomis, būtina sekti, kad pakeistos eilutės ilgis nebūtų didesnis už nurodytą apraše.

Visos operacijos su `char[]` tipo eilutėmis (simbolių masyvais) atliekamos naudojant tam skirtas funkcijas. Lentelėse pateiktos funkcijos, kurios naudojamos vadovėlyje arba gali būti reikalingos pateiktoms užduotims spręsti.

Funkcijos prototipas	Paiškinimai
<pre>int strcmp(const char *s1,            const char *s2);</pre>	<p>Palygina eilutes s1 ir s2. Eilučių simbolių (ženklų) kodai iš eilės lyginami tol, kol randami skirtingi simboliai arba kurios nors eilutės pabaiga (didžiosios ir mažosios raidės skiriamos). Funkcija grąžina:</p> <ul style="list-style-type: none"> <li>• 0, jei eilutės sutampa (s1 == s2);</li> <li>• teigiamąjį skaičių, jei eilutės s1 kodas didesnis už eilutės s2 kodą (s1 &gt; s2);</li> <li>• neigiamąjį skaičių, jei eilutės s1 kodas mažesnis už eilutės s2 kodą (s1 &lt; s2).</li> </ul> <p><i>Pavyzdys</i></p> <pre>char e[] = "medis"; char s[] = "medelis"; if (strcmp(e, s) == 0)     cout &lt;&lt; "e = s"; else if (strcmp(e, s) &lt; 0)     cout &lt;&lt; "e &lt; s"; else cout &lt;&lt; "e &gt; s";</pre> <p>Rezultatas – pranešimas ekrane: e &gt; s.</p>
<pre>int stricmp(const char *s1,             const char *s2);</pre>	<p>Palygina eilutes s1 ir s2. Eilučių simbolių (ženklų) kodai iš eilės lyginami tol, kol randami skirtingi simboliai arba kurios nors eilutės pabaiga (didžiosios ir mažosios raidės neskiriamos). Funkcija grąžina:</p> <ul style="list-style-type: none"> <li>• 0, jei eilutės sutampa (s1 == s2);</li> <li>• teigiamąjį skaičių, jei eilutės s1 kodas didesnis už eilutės s2 kodą (s1 &gt; s2);</li> <li>• neigiamąjį skaičių, jei eilutės s1 kodas mažesnis už eilutės s2 kodą (s1 &lt; s2).</li> </ul> <p><i>Pavyzdys</i></p> <pre>char e[] = "medis"; char s[] = "Medis"; if (stricmp(e, s) == 0)     cout &lt;&lt; "e = s"; else if (stricmp(e, s) &lt; 0)     cout &lt;&lt; "e &lt; s"; else cout &lt;&lt; "e &gt; s";</pre> <p>Rezultatas – pranešimas ekrane: e = s.</p>
<pre>int strncmp(const char *s1,             const char *s2,             int m);</pre>	<p>Palygina eilučių s1 ir s2 pirmuosius m simbolių (didžiosios ir mažosios raidės skiriamos). Funkcija grąžina:</p> <ul style="list-style-type: none"> <li>• 0, jei eilučių dalys (m pirmųjų simbolių) sutampa;</li> <li>• teigiamąjį skaičių, jei s1 dalis didesnė už s2 dalį;</li> <li>• neigiamąjį skaičių, jei s1 dalis mažesnė už s2 dalį.</li> </ul> <p><i>Pavyzdys</i></p> <pre>char e[] = "medis"; char s[] = "medelis"; if (strncmp(e, s, 3) == 0)     cout &lt;&lt; "e = s"; else if (strncmp(e, s, 3) &lt; 0)     cout &lt;&lt; "e &lt; s"; else cout &lt;&lt; "e &gt; s";</pre> <p>Rezultatas – pranešimas ekrane: e = s.</p>
<pre>int strnicmp(const char *s1,              const char *s2,              int m);</pre>	<p>Palygina eilučių s1 ir s2 pirmuosius m simbolių (didžiosios ir mažosios raidės neskiriamos). Funkcija grąžina:</p> <ul style="list-style-type: none"> <li>• 0, jei eilučių dalys (m pirmųjų simbolių) sutampa;</li> <li>• teigiamąjį skaičių, jei s1 dalis didesnė už s2 dalį;</li> <li>• neigiamąjį skaičių, jei s1 dalis mažesnė už s2 dalį.</li> </ul>

	<p><i>Pavyzdys</i></p> <pre>char e[] = "medis"; char s[] = "Medelis"; if (strnicmp(e, s, 3) == 0)     cout &lt;&lt; "e = s"; else if (strnicmp(e, s, 3) &lt; 0)     cout &lt;&lt; "e &lt; s"; else cout &lt;&lt; "e &gt; s";</pre> <p>Rezultatas – pranešimas ekrane: e = s.</p>
unsigned <b>strlen</b> (char *s);	<p>Apskaičiuoja ir grąžina eilutės s (be nulinio simbolio) ilgį.</p> <p><i>Pavyzdys</i></p> <pre>char e[12] = "C++ kalba"; int n = strlen(e);</pre> <p>Rezultatas – kintamojo n reikšmė (9).</p>

### Eilučių konvertavimo ir kopijavimo funkcijos

Funkcijos prototipas	Paaiškinimai
double <b>atof</b> (const char *s);	<p>Konvertuoja eilutės s pirmuosius simbolius į realųjį skaičių, pateikiamą standartine išraiška, ir jį grąžina.</p> <p><i>Pavyzdys</i></p> <pre>char *s = "45.23e3abc"; double d = atof(s);</pre> <p>Rezultatas – kintamojo d reikšmė 45.230.</p>
int <b>atoi</b> (const char *s);	<p>Konvertuoja eilutės s pirmuosius simbolius į sveikąjį skaičių ir jį grąžina.</p> <p><i>Pavyzdys</i></p> <pre>char *s = "45.23e3abc"; int n = atoi(s);</pre> <p>Rezultatas – kintamojo n reikšmė 45.</p>
long <b>atol</b> (const char *s);	<p>Konvertuoja eilutės s pirmuosius simbolius į ilgąjį sveikąjį skaičių ir jį grąžina.</p> <p><i>Pavyzdys</i></p> <pre>char *s = "45.23e3abc"; long n = atol(s);</pre> <p>Rezultatas – kintamojo n reikšmė 45.</p>
char * <b>strcpy</b> (char *d, const char *s);	<p>Kopijuoja eilutę s į eilutę d ir grąžina rodyklę į eilutę d.</p> <p><i>Pavyzdys</i></p> <pre>char e[12]; strcpy(e, "C++ kalba");</pre> <p>Rezultatas – eilutės e reikšmė: "C++ kalba".</p>
char * <b>strncpy</b> (char *d, char *s, unsigned m);	<p>Kopijuoja m simbolių iš eilutės s į eilutę d ir grąžina rodyklę į eilutę d (eilutės pabaigos simbolio ('\0') neįrašo).</p> <p><i>Pavyzdys</i></p> <pre>char e[12]; char s[] = "C++ kalba"; strncpy(e, s, 3); e[3]='\0';</pre> <p>Rezultatas – eilutės e reikšmė: "C++".</p>
char * <b>strlwr</b> (const char *s);	<p>Perrašo visą eilutę s mažosiomis raidėmis ir grąžina rodyklę į pakeistą eilutę s (mažųjų raidžių ir kitų simbolių nekeičia).</p> <p><i>Pavyzdys</i></p> <pre>char e[] = "C++ Builder"; strlwr(e);</pre> <p>Rezultatas – eilutės e nauja reikšmė: "c++ builder".</p>

<pre>char *strupr(const char *s);</pre>	<p>Perrašo visą eilutę s didžiosiomis raidėmis ir grąžina rodyklę į pakeistą eilutę s (didžiųjų raidžių ir kitų simbolių nekeičia).</p> <p><i>Pavyzdys</i></p> <pre>char e[] = "C++ Builder"; strupr(e);</pre> <p>Rezultatas – eilutės e nauja reikšmė: "C++ BUILDER".</p>
---	--

### Eilučių modifikavimo funkcijos

Funkcijos prototipas	Paaiškinimai
<pre>char *strcat(char *d,               const char *s);</pre>	<p>Prie eilutės d prijungia eilutę s ir grąžina rodyklę į sujungtą eilutę d.</p> <p><i>Pavyzdys</i></p> <pre>char e[12] = "C++ "; char s[] = "kalba"; strcat(e, s);</pre> <p>Rezultatas – eilutės e nauja reikšmė: "C++ kalba".</p>
<pre>char *strncat(char *d,               const char *s,               unsigned m);</pre>	<p>Prie eilutės d prijungia m simbolių iš eilutės s ir grąžina rodyklę į sujungtą eilutę d.</p> <p><i>Pavyzdys</i></p> <pre>char e[15] = "C++ "; char s[] = "programavimas"; strncat(e, s, 8);</pre> <p>Rezultatas – eilutės e nauja reikšmė: "C++ programa".</p>
<pre>char *strnset(char *s,               char c,               unsigned m);</pre>	<p>Užpildo s eilutės m elementų simboliu c ir grąžina rodyklę į eilutę s.</p> <p><i>Pavyzdys</i></p> <pre>char e[] = "C++ Builder"; strnset(e, '*', 4);</pre> <p>Rezultatas – eilutės e nauja reikšmė: "****Builder".</p>
<pre>char *strset(char *s,               char c);</pre>	<p>Užpildo eilutę s simboliu c ir grąžina rodyklę į eilutę s.</p> <p><i>Pavyzdys</i></p> <pre>char e[] = "C++ Builder"; strset(e, '*');</pre> <p>Rezultatas – eilutės e nauja reikšmė: "*****".</p>

### Eilučių skaitymas ir rašymas

Simbolių eilutėms įvesti naudojamas operatorius >>. Jis priskiria įvesties sraute nurodytam simbolių eilutės kintamajam visus simbolius iki pirmo tarpo simbolio arba iki eilutės pabaigos simbolio '\n' (visi tarpo simboliai iki eilutės pradžios praleidžiami). Eilutės pabaigoje įterpia eilutės pabaigos simbolį '\0'. Jeigu aprašant eilutę pabaigos simbolio vieta nebuvo numatyta arba eilutė įvesties sraute yra ilgesnė, nei numatyta eilutės apraše, tuomet eilutės pabaigos simbolis į ją neįrašomas ir veiksmai su tokia eilute tampa klaidingi arba neįmanomi.

Pavyzdžiui, jei įvesties sraute fd yra simbolių eilutė

```
Man patinka C++ programavimo kalba
```

tai, atlikę veiksmus

```
char eil[100];
fd >> eil; // fd-įvesties srautas
cout << eil;
```

ekrane matysime simbolių eilutę:

```
Man
```

Norint įvesti simbolių eilutę ne iki tarpo, reikia naudoti globaliąją funkciją get():

```
istream & get(char eil[], int n);
```

Funkcija `get()` iš įvesties srauto įveda  $n-1$  simbolių ir palieka vietą eilutės pabaigos simboliui `'\0'`. Pavyzdžiui, jeigu įvesties sraute `fd` yra simbolių eilutė

```
Man patinka C++ programavimo kalba
```

tai, atlikę veiksmus

```
char eil[100];
fd.get(eil, 10);
cout << eil;
```

ekrane matysime tokią simbolių eilutę:

```
Man patin
```

Įvesties sraute iki eilutės pradžios esantys tarpo simboliai nepraleidžiami. Jeigu įvesties sraute yra mažiau simbolių, nei numatyta eilutės apraše, tuomet įvestis baigiama, aptikus eilutės pabaigos simbolių. Jeigu reikia įvesti tiek simbolių, kiek nurodyta eilutės apraše, tuomet patogiu naudoti operatorių `sizeof`, kuris apskaičiuoja kintamojo ilgį baitais:

```
fd.get(eil, sizeof eil);
```

*Pavyzdys.* Funkcija iš srauto `fd` nukopijuoja  $n$  eilučių į srautą `fr`

```
void KopijuotiSrautoEilutes(istream & fd, ostream & fr, int n)
{
    char E[100];
    for (int i = 0; (i < n) && (!fd.eof()); i++) {
        fd.get(E, sizeof E);
        fr << E << endl;
    }
}
```

Simbolių eilutėms išvesti naudojamas operatorius `<<`. Jis išveda visus eilutės simbolius iki jos pabaigos simbolio `'\0'`, pavyzdžiui:

```
fr << eil;
```

## Simbolių masyvas

Veiksmams su simbolių eilutėmis atlikti yra daug funkcijų. Tačiau simbolių eilutės – tokie pat masyvai, kaip sveikųjų ar realiųjų skaičių. Tik simbolių masyvo elementų reikšmės yra simboliai. Todėl darbui su simbolių masyvo elementais galima taikyti tuos pačius algoritmus, kaip ir su sveikųjų ar realiųjų skaičių masyvų elementais.

*Pavyzdys.* Simboliai skaitomi iš failo

```
void Skaityti(char E[], int & n)
{
    ifstream fd("Sim.txt");
    for (n = 0; !fd.eof() && n < Cn; n++) // Cn – masyvo dydis
        fd >> E[n];
    fd.close();
}
```

Operatorius `>>` praleidžia visus tarpo, tabulavimo žymės ir eilutės pabaigos simbolius. Norint masyve laikyti visus tekste esančius simbolius, įvedimo takinį reikia pakeisti tokiu:

```
fd.get(E[n]);
```



## Pavyzdys. Rašomos simbolių masyvo reikšmės

```
void Spausdinti(char E[], int n)
{
    ofstream fr("SimRez.txt");
    for (int i = 0; i < n; i++)
        fr << E[i];
    fr.close();
}
```

## Pavyzdys. Skaičiuojama, kiek kartų masyve pasitaiko nurodytas simbolis

```
int Kiek(char E[], int n, char sim)
{
    int k = 0;
    for (int i = 0; i < n; ++i)
        if (E[i] == sim) k++;
    return k;
}
```

Norint išsiaiškinti, kiek a raidžių yra masyve, reikia parašyti kreipinį į funkciją Kiek():

```
cout << Kiek(E, n, 'a');
```

Pavyzdžiui, norint rasti, kuri mažoji raidė simbolių masyve pasikartoja dažniausiai, taip pat galima pasinaudoti jau turima Kiek() funkcija:

```
char Kuri(char E[], int n)
{
    int sk, k = 0;
    char sim = ' ';
    for (char i = 'a'; i <= 'z'; i++) {
        sk = Kiek(E, n, i);
        if (sk > k) {
            k = sk;
            sim = i;
        }
    }
    return sim;
}
```

Jeigu simbolių masyvo gale nėra eilutės pabaigos simbolio '\0', tuomet negalima taikyti darbo su simbolių eilutėmis funkcijų ir kitų veiksmų visam masyvui, pavyzdžiui, cout << E;

## Simbolių eilučių masyvas

Viena simbolių eilutė yra simbolių masyvas. Jeigu eilučių yra daug, patogų jas susirašyti į vieną masyvą, pavyzdžiui:

```
char Mas[100][15];
```

Šiame masyve gali būti laikoma 100 eilučių, kurių kiekviena gali turėti po 15 simbolių. Pavyzdžiui,

```
cout << Mas[5] << endl; // parodo ekrane penktoje vietoje esančią eilutę
cout << Mas[2][12] << endl; // parodo ekrane antroje vietoje esančios eilutės 12-ą simbolį
```

```
void Skaityti(char Mas[][15], int &n)
{
    ifstream fd("Simb.txt");
    for (n = 0; !fd.eof() && n < CMax; n++) {
        fd.get(Mas[n], sizeof Mas[n]);
        fd.ignore(80, '\n'); // praleidžiami visi likę simboliai iki eilutės pabaigos
        // paprastai tekstiniuose failuose nebūna ilgesnių kaip 80 simbolių eilučių
    }
    fd.close();
}
```

Masyvo eilutes išvesties sraute `fr` galima spausdinti taip:

```
for (int i = 0; i < r; i++)
    fr << A[i] << endl;
```

Užrašius

```
fr << A[3][5] << endl;
```

bus išspausdintas masyvo trečioje vietoje esančios eilutės penktas simbolis.

## 2.9. Simbolių eilutė `string`

Darbu su simbolių eilutėmis yra sukurta klasė (tipas) `string`. Failas, kuriame yra aprašyta klasė `string`, į programą įterpiamas sakiniu

```
#include <string>
```

Simbolių eilučių tipo kintamieji (objektai) aprašomi taip pat, kaip ir kitų tipų kintamieji, pavyzdžiui,

```
string eil, e1;
```

Sukurtos eilutės yra tuščios. Jas aprašant galima priskirti reikšmes, pavyzdžiui,

```
string eil = "Jau moku programuoti C++ kalba";
```

Klasės `string` objektai literatūroje dažnai vadinami C++ *eilutėmis*. Eilučių simboliai indeksuojami taip pat, kaip ir masyve, t. y. pradedant 0. Darbu su eilutėmis yra naudojami tokie operatoriai:

### 1) Priskyrimo (=).

Klasės `string` objektui galima priskirti kitos eilutės ar net simbolio reikšmę. Pavyzdžiui:

```
string s1, s2;
string s = "12";
char c = 'z';
s1 = s; s2 = c;
```

Rezultatas – `s1` reikšmė ("12") ir `s2` reikšmė ("z").

### 2) Sudėties (+).

Sudėdant eilutes `s1` ir `s2`, prie pirmos eilutės pabaigos pridedama antros eilutės kopija. Galima sudėti dvi C++ eilutes arba C++ eilutę ir simbolį. Pavyzdžiui:

```
string s1 = "a", s2 = "b";
string s = "12"; char c = 'z';
s1 = s1 + s; s2 = s2 + c;
```

Rezultatas – `s1` reikšmė ("a12") ir `s2` reikšmė ("bz").

### 3) Lyginimo (==, !=, < ir kt.).

Lyginami eilutes sudarančių simbolių kodai. Dvi eilutės lygios, jei jos yra vienodo ilgio ir sutampa visų jų simbolių kodai poromis. C++ eilutę galima lyginti tik su kita C++ eilute.

Klasės `string` objektams (kintamiesiems) galima taikyti įvesties srauto operatorių `>>` ir išvesties srauto operatorių `<<`.

Operatorius >> įveda į string tipo eilutę simbolių seką iki tarpo arba iki eilutės pabaigos simbolio '\n'.

Pavyzdžiui, jei įvesties sraute fd yra simbolių eilutė

```
Man patinka C++ programavimo kalba
```

tai, atlikę veiksmus

```
string eil
fd >> eil; // fd-įvesties srautas
cout << eil;
```

ekrane matysime eilutę:

```
Man
```

Norint įvesti dalį simbolių eilutės (pvz., ne iki tarpo), reikia naudotis globaliaja funkcija getline(). Galimi du šios funkcijos formatai:

```
istream & getline(istream & fd, string & str); // įveda iki eilutės pabaigos simbolio '\n'
istream & getline(istream & fd, string & eil, char skyr); // įveda iki skyriklio simbolio skyr
```

Tarkime, faile fd yra eilutė

```
Dangaus laiptelis * Andersen
```

Pavyzdys. Veiksmai ir jų rezultatai, rodomi ekrane

Veiksmai	Rezultatai ekrane
<pre>string eil fd &gt;&gt; eil; // fd-įvesties srautas cout &lt;&lt; eil;</pre>	Dangaus
<pre>string eil getline(fd, eil); // fd-įvesties srautas cout &lt;&lt; eil;</pre>	Dangaus laiptelis * Andersen
<pre>string eil getline(fd, eil, '*'); // fd-įvesties srautas cout &lt;&lt; eil;</pre>	Dangaus laiptelis

Išvesties srauto operatorius << išveda string tipo eilutę eil:

```
fr << eil;
```

Skaidydami tekstą, prarandame eilutės pabaigos simbolį '\n', todėl reikia rašyti:

```
fr << eil << endl;
```

Pavyzdžiui, iš srauto fd nukopijuoti n eilučių į srautą fr galima naudojantis šia funkcija:

```
void KopijuotiSrautoEilutes(istream & fd, ostream & fr, int n)
{
    string E;
    for (int i = 0; (i < n) && (!fd.eof()); i++) {
        getline(fd, E);
        fr << E << endl;
    }
}
```

Jei norima iš C++ eilutės paimti atskirus simbolius, tuomet laužtiniuose skliaustuose reikia nurodyti jų indeksus.

Pavyzdžiui, turime eilutę:

```
string eil = "Skanus agurkas";
```

Tuomet sakiniu

```
cout << eil[5];
```

ekrane parodysiame eilutės simbolį s, kuris masyve yra 5-oje vietoje (masyvo elementai indeksuojami pradedant nuo 0).

Funkcijos prototipas	Paiškinimai
size_type <b>max_size</b> () const;	Grąžina elementų, kurie gali būti eilutėje, skaičių. <i>Pavyzdys</i> string s("C++ kalba"); int n = s.max_size(); Rezultatas – n reikšmė: 4294967281.
size_type <b>length</b> () const;	Grąžina eilutės ilgį (simbolių skaičių eilutėje). <i>Pavyzdys</i> string s("C++ kalba"); int n = s.length(); Rezultatas – n reikšmė: 9.
const char * <b>c_str</b> ();	Grąžina rodyklę-konstantą į C eilutę, kurios reikšmė sutampa su eilute, kuriai taikoma ši funkcija. Naudojama failų atidarymo sakiniuose. <i>Pavyzdys</i> string s = "Duomenys.txt"; ifstream fd(s.c_str()); Rezultatas – eilutė s, konvertuota į C eilutę.
string <b>substr</b> (size_type k, size_type n = npos);	Iš eilutės, kuriai taikoma ši funkcija, kopijuojama n simbolių, pradedant k-uoju. <i>Pavyzdys</i> string s = "Programavimo kalba C++"; string s1 = "Matematika"; s1 = s.substr(13, 5); Rezultatas – nauja eilutės s1 reikšmė: "kalba".

## 2.10. Struktūra

### Struktūros aprašymas, kintamieji

*Struktūros* duomenų tipu (angl. *struct*) vadinamas įvairių duomenų elementų rinkinys. Analogiška duomenų struktūra *Pascal* programavimo kalbų šeimoje vadinama *įrašu* (angl. *record*). Struktūros tipo kintamasis laiko atmintyje tarpusavyje logiškai susietus duomenis.

Struktūros tipo aprašo sintaksė yra tokia:

```
struct vardas { kintamųjųSąrašas };
```

Struktūros aprašo bazinis žodis yra *struct*. Po jo rašomas struktūros vardas. Kintamųjų rinkinys pateikiamas riestiniuose skliaustuose. Struktūros tipo kintamieji aprašomi pagal bendras taisykles: tipo vardas, jo kintamųjų vardai, atskirti kableliais. Sąrašo pabaigoje (už riestinio skliausto) rašomas kabliataškis.

Pavyzdžiui, prekės pavadinimui, vieneto kainai, kiekiui ir pinigų sumai laikyti galima sudaryti tokią struktūrą:

```
struct Pirkiny {
    string pav;           // pavadinimas
    double kaina;        // kaina
    int kiekis;          // kiekis
    double suma;         // pinigų suma
};
```

Tipo pavadinimas
Kintamųjų sąrašas

Struktūros tipo bazinis žodis

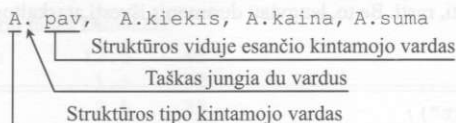
Struktūros tipo kintamieji aprašomi naudojant sukurto tipo vardą, pavyzdžiui, Pirkinyns A, B;

Grafiškai struktūros tipo kintamuosius A ir B galima pavaizduoti taip:

A				B			
pav	kaina	kiekis	suma	pav	kaina	kiekis	suma

Struktūros tipo kintamojo kopiją galima gauti priskyrimo veiksmu:  $B = A$ ; Tai vienintelis galimas veiksmas, atliekamas su struktūros tipo kintamuoju. Kiti veiksmai galimi tik su konkrečiais duomenimis, kuriuos laiko vidiniai struktūros kintamieji.

Vidinio struktūros kintamojo vardas yra sudėtinis. Jis sudarytas iš struktūros tipo kintamojo vardo ir vidinio kintamojo vardo, tarp kurių rašomas taškas, pavyzdžiui:



Kiekvienas struktūros tipo kintamasis turi visus to tipo vidinių kintamųjų vardus. Pavyzdžiui, pirkinio A duomenis galima priskirti taip:

```
A.pav = "Agurkas";  
A.kaina = 2.56;  
A.kiekis = 15;  
A.suma = A.kaina * A.kiekis;
```

Jeigu kintamojo A atskiras reikšmės laiko kiti kintamieji, tai juos galima panaudoti priskyrimo veiksmuose, pavyzdžiui:

```
A.pav = vardas; // kopijuojama kintamojo vardas reikšmė (simbolių eilutė)  
A.kaina = kainaNauja;  
A.kiekis = kiekisNaujas;
```

## Struktūros kintamojo reikšmių skaitymas / rašymas

Struktūros tipo kintamojo reikšmės rodomos ekrane arba rašomos į failą taip pat, kaip ir kitų nagrinėtų tipų kintamųjų reikšmės. Reikia tik nepamiršti visur rašyti sudėtinį kintamojo vardą:

```
cout << A.pav << " " << A.kaina << " "  
<< A.kiekis << " " << A.suma;
```

Kai faile duomenys surašyti atskiromis eilutėmis arba visos reikšmės yra skaitinės, struktūros tipo duomenims skaityti naudojamos tos pačios priemonės, kaip ir kitų tipų kintamųjų. Kai struktūroje yra eilutės tipo kintamieji ir jų reikšmės faile surašytos ne atskiromis eilutėmis, o kartu su kitomis reikšmėmis, tuomet skaitymo metu reikia atpažinti, kur baigiasi simbolių eilutė. Pavyzdžiui,

```
Agurkas 2.56 5
```

Jeigu pavadinimai yra iš kelių žodžių, tuomet duomenų skaitymo programa gali tapti labai sudėtinga. Pavyzdžiui:

```
Pekino salotos 2.56 6  
Raudonieji gūžiniai kopūstai 2.15 4
```

Skaitant skaičius, visi tarpo simboliai iki skaičiaus praleidžiami. Skaičiaus užrašas baigiamas tarpo arba eilutės pabaigos simboliu. Tačiau jeigu po skaičiaus yra rašoma simbolių eilutė, tarpo simbolių skaičius turi didelę reikšmę, nes reikia žinoti, keli iš jų priklauso pavadinimui, o keli – ne. Todėl reikia parašyti programos fragmentą, skirtą tiems simboliams skaitymo metu praleisti. Žinoma, galima po skaičiaus iš karto rašyti simbolių eilutę (nepalikant tarpo), tačiau toks duomenų pateikimo būdas yra labai nevaizdus ir gali būti klaidų šaltinis.

Duomenims faile pateikti reikia parinkti tokią formą, kuri būtų nesudėtinga, vaizdi ir leistų neklysti juos įvedant. Tai *pozicinis* duomenų surašymas. Kiekviena duomenų reikšmė yra pateikiama pradedant sutarta pozicija

(simbolio eilės numeriu nuo eilutės pradžios). Numatoma, kiek daugiausia pozicijų bus skiriama kiekvienai duomenų reikšmei. Duomenų pateikimo forma primena stulpelius:

Nuo 1 pozicijos	Nuo 32 pozicijos	Nuo 45 pozicijos
Pekino salotos	2.56	5
Raudonieji gūžiniai kopūstai	12.15	415

Esant tokiai duomenų formai, yra žinomas maksimalus kiekvienos simbolių sekos ilgis. Skaitant eilutę, perskaitomi visi jos pabaigoje esantys tarpo simboliai. Atmintyje laikomos simbolių eilutės yra vienodo ilgio. Taip paprasčiau jas rikiuoti, atpažinti, rasti. Be to, lengviau duomenis išvesti atskaitos forma.

#### Pavyzdys

```
ifstream fd ("Duom.txt");
char simb[30];
fd.get(simb, sizeof simb);
A.pav = simb; // C eilutė konvertuojama į C++ eilutę
fd >> A.kiekis >> A.kaina;
fd.ignore();
fd.get(simb, sizeof simb);
B.pav = simb; // C eilutė konvertuojama į C++ eilutę
fd >> B.kiekis >> B.kaina;
fd.ignore();
fd.close();
```

Įvedant struktūros tipo duomenis, vienoje duomenų eilutėje paprastai būna kelios simbolių eilutės ir skaičiai. Žinome, kad skaitant visi tarpo simboliai iki skaičiaus yra praleidžiami. Tačiau skaitant simbolių eilutes taip nėra. Skaitymo procesui valdyti taikomas įvesties srauto metodas:

**ignore**(max, skyriklis)

Iš įvesties srauto pašalina ne daugiau kaip max simbolių arba visus simbolius iki eilutės pabaigos, jeigu jų yra mažiau nei max. Šalinimas sraute nutraukiamas, jeigu aptinkamas simbolis skyriklis.

Pavyzdžiai:

```
fd.ignore(); // iš srauto šalinamas vienas simbolis; jeigu tai buvo '\n', pereinama į kitą failo eilutę
fd.ignore(20, 'k'); // iš srauto šalinama ne daugiau kaip 20 simbolių, kol aptinkamas nurodytas simbolis
Dažnai vartojama forma, kai reikia pašalinti iš įvesties srauto visus simbolius iki eilutės pabaigos ir pereiti į naują eilutę.
Paprastai tekstiniuose failuose nebūna ilgesnių kaip 80 simbolių eilučių
fd.ignore(80, '\n');
```

Paprastai įvesties sraute simbolių eilutės viena nuo kitos skiriamos tarpais. Norint visus tarpo simbolius iki eilutės pradžios praleisti, patogų pasinaudoti manipuliatoriumi ws, pavyzdžiui,

```
fd >> ws; // sraute praleidžiami visi tarpai iki pirmojo netarpo
```

## Struktūrų masyvas

Realiuose uždaviniuose naudojami struktūrų masyvai. Jie aprašomi taip pat, kaip ir skaičių masyvai, pavyzdžiui,

```
Pirkinyas A[50];
```

Taip pat rašomi ir kreipiniai į struktūrų masyvo elementus:

```
A[5], A[i], A[k + 2]
```

Kreipiniai į elementų laukus rašomi, pavyzdžiui, taip:

```
A[5].pav, A[i].kiekis, A[k+r].pav
```

Struktūrų masyvo duomenys dažniausiai pateikiami tekstiniam faile. Pavadinimus sudarantys simboliai paprastai rašomi toje pačioje eilutėje, kaip ir struktūrai priklausantys skaičiai, todėl reikia ypač atidžiai rašyti duomenų skaitymo sakinius. Duomenų skaitymo iš failo priemonių yra daug, todėl pateikiant faile duomenis reikia įvertinti, kaip jie bus skaitomi.

Dirbant su masyvu, jo dydį nusakančią reikšmę pravartu apibrėžti kaip konstantą. Tuomet skaitant bus galima lengviau sekti įvedamų duomenų skaičių (į masyvą galima įrašyti ne daugiau duomenų, nei nurodyta apraše).

Panagrinėkime pavyzdį, kai tekstiniam duomenų faile surašytos pirkėjo pirktos prekės: nurodytas prekės pavadinimas, kaina ir kiekis. Reikia apskaičiuoti, kiek pirkejas sumokėjo už kiekvieną prekę atskirai ir už visas prekes.

### Pavyzdys

Tarkime, kad yra toks duomenų failas *Duom.txt*:

Olandiškas sūris	12.5	45
Vilniaus duona	1.5	12
Pienas	2.5	25

Duomenų struktūra tokia pat, kaip jau aprašyta anksčiau. Duomenų yra daug. Todėl sukuriamas struktūrų masyvas, į kurį duomenys perkeliama iš failo.

Skaitymo metu apskaičiuojama, kiek reikia sumokėti už kiekvieną prekę atskirai. Po to apskaičiuojama visų perkamų prekių piniginė vertė ir gauti rezultatai įrašomi į failą.

### Programos pavyzdys

```
// Veiksmai su struktūrų masyvo duomenimis
const char CDuomenys[] = "Duom.txt";
const char CRezultatai[] = "Rez.txt";
const Cn = 100; // masyvo dydis
const Cpav = 20; // prekės pavadinimo ilgis
//-----
struct Pirkinys {
    string pav; // pavadinimas
    double kaina; // kaina
    int kiekis; // kiekis
    double suma; // pinigų suma
};
//-----
Pirkinys A[Cn]; int n; // duomenų masyvas A(n)
void Skaityti(const char failoVardas[]);
double Pinigai();
void Spausdinti(const char failoVardas[], double suma);
//-----
int main()
{
    Skaityti(CDuomenys);
    double suma = Pinigai();
    Spausdinti(CRezultatai, suma);
    return 0;
}
//-----
// Duomenys įvedami iš failo failoVardas į masyvą A(n) ir
// skaičiuojama struktūros vidinio kintamojo suma reikšmė
void Skaityti(const char failoVardas[])
{
    ifstream fd(FailoVardas);
    n = 0;
    char simb[Cpav + 1]; // papildoma vieta reikalinga nuliniam simboliui
```

```

// Skaitomi duomenys iki failo pabaigos arba tiek, kiek telpa masyve, jeigu duomenų daug
while (!fd.eof() && (n < Cn)) {
    fd.get(simb, Cpav);
    A[n].pav = simb;
    fd >> A[n].kaina >> A[n].kiekis;
    fd.ignore(80, '\n'); // praleidžiami visi likę simboliai iki eilutės pabaigos
    A[n].suma = A[n].kaina * A[n].kiekis;
    n++;
}
fd.close();
}
//-----
// Masyvo A(n) ir kintamojo suma reikšmės rašomos į failą failoVardas
void Spausdinti(const char failoVardas[], double suma)
{
    ofstream fr(failoVardas);
    fr << "M O K Ė J I M A I " << endl;
    fr << "- - - - -" << endl;
    for (int i = 0; i < n; i++)
        fr << A[i].pav << " " << A[i].suma << " Lt" << endl;
    fr << "- - - - -" << endl;
    fr << "Mokinys sumokėjo: " << suma << " Lt" << endl;
    fr.close();
}
//-----
// Skaičiuojama ir gražinama apskaičiuota prekių pinigine vertė
double Pinigai()
{
    double s = 0;
    for (int i = 0; i < n; i++)
        s += A[i].suma;
    return s;
}
//-----

```

### Rezultatų failas

```

M O K Ė J I M A I
- - - - -
Olandiškas sūris      562.5 Lt
Vilniaus duona       18 Lt
Pienas                62.5 Lt
- - - - -
Mokinys sumokėjo:    643 Lt

```

### Sudėtinės struktūros

Struktūroje gali būti ne tik paprastų duomenų tipų kintamieji. Čia gali būti masyvai, kitos struktūros, rodyklės ir kt. Atliekant veiksmus su sudėtinės struktūros kintamojo reikšmėmis, būtina žinoti, kad nuosekliai rašomi visi kintamųjų vardai, pradedant struktūros kintamuoju ir baigiant konkrečią reikšmę saugančio kintamojo vardu. Sudėtinio vardo dalys skiriamos taškais. Pavyzdžiui:

```

struct PvzA {
    int A[Cn];
    int na;
    int suma;
};
PvzA P1, P2;

```

Jau žinome, kad galimas priskyrimo veiksmas  $P1 = P2$ ;



Masyvo vardas užrašomas P1.A, o masyvo elementas nurodomas indeksu, pavyzdžiui, taip:

```
P1.A[k]; // k - sveikąjo tipo kintamasis
```

Jeigu masyvo pabaigoje norima įrašyti papildomą reikšmę, skaitomą iš srauto, tuomet galima nurodyti taip:

```
cin >> P1.A[P1.na];  
P1.na++;
```

Galima sukurti struktūrą, kurios viduje kintamieji būtų jau sukurtų struktūrų tipų, pavyzdžiui:

```
struct PvzB {  
    PvzA Mas[Cn]; // struktūrų masyvas  
    int n;  
};  
PvzB P2;
```

Užrašas P2.Mas nurodo struktūros masyvą, o užrašas P2.Mas[5] – penktąjį masyvo elementą. Masyvo elementas yra struktūros tipo, todėl jo konkrečios reikšmės vardus nusako tokie užrašai:

```
P2.Mas[5].A[15]  
P2.Mas[5].na
```

## 2.11. Knygoje naudojamų įterpiamųjų failų sąrašas

Programos pradžioje *parengiamajai doroklei* (angl. *preprocessor*) rašomos instrukcijos, kuriose nurodoma, ko-kių failų tekstai turi būti įterpiami į programą pirminio apdorojimo metu. Įterpimo instrukcijos pradžioje rašoma #include, toliau tarp simbolių < > nurodomi įterpiamųjų failų vardai. Pavyzdžiui, antraštinio failo iostream priemonės perkeliama į programą sakiniu

```
#include <iostream>
```

Praktikos darbuose naudojami antraštiniai failai

Įterpiamasis failas	Paaishkinimas
iostream	Duomenų įvedimo klaviatūra ir rodymo ekrane priemonės
fstream	Duomenų skaitymo iš failo ir rašymo į failą priemonės
iomanip	Duomenų išvedimo į failų srautus (ekraną, failą) priemonės
cmath	Matematinų funkcijų rinkinys
string	Darbo su string tipo eilutėmis priemonės

# 3

## ALGORITMŲ ŽINYNAS

Visi jau žinomi ir nauji algoritmai pristatomi taikant juos darbui su sveikųjų skaičių masyvų reikšmėmis. Pakeitus masyvo reikšmių tipą, algoritmai nesikeičia. Jeigu prireikia papildomų veiksmų, tai algoritmus tenka modifikuoti. Pavyzdžiui, kai reikia rasti masyve laikomų teigiamųjų reikšmių sumą, tuomet, prieš atliekant sumos veiksmą, reikia patikrinti, ar masyvo elemento reikšmė teigiama. Apskritai sprendžiant konkrečius uždavinius nagrinėjamus algoritmus visuomet reikia modifikuoti. Kai kurie algoritmai pritaikyti masyvams su struktūros tipo reikšmėmis. Tuomet struktūros tipo kintamųjų vardai rašomi nurodant kelis vardus, atskirtus tašku.

*Pavyzdys*

```
struct Namas {
    string pav; // pavadinimas
    int kiek; // kambarių skaičius
    double plotas // visas naudingas namo plotas
};
Namas muziejus = {"Velnių", 3, 125.5};
```

Kintamojo muziejus reikšmę sudaro trijų vidinių kintamųjų pav, kiek ir plotas reikšmės. Veiksmus galima atlikti tik su konkrečiomis reikšmėmis, kurias nusako sudėtinis vardas:

muziejus.pav, muziejus.kiek, muziejus.plotas.

Su kintamojo muziejus reikšme galima atlikti tik priskyrimo veiksmą, pavyzdžiui,

Namas kitasMuziejus = muziejus;

Galimi veiksmai su keliomis reikšmėmis vienu metu. Pavyzdžiui, skaičiuokle rikiavimo veiksmą galima atlikti pagal kelis raktus. Tą patį galima padaryti ir naudojantis rikiavimo algoritmu. Tam tereikia parašyti sudėtingesnę lyginimo reiškinį, kuris paprastai rikiavimo ar paieškos algoritmuose vadinamas *rikiavimo / paieškės raktū*. Tokiu atveju raktui kuriama atskira funkcija, kurios rezultatas gali būti tiek loginė, tiek skaitinė reikšmė.

Žinyne algoritmai pateikiami naudojantis *struktūrogrāmomis*.

*Pagrindiniai struktūrogramų simboliai*

Grafinis simbolis	Pavyzdys	Paskirtis
	$y = \sin(15 * a) + \text{Plotas}(s, t)$	Šiuo simboliu vaizduojami tokie algoritmų veiksmai: pradžia, pabaiga, skaičiavimai, kreipiniai į funkcijas ir kt. Veiksmų dedamųjų gali būti kiek norima daug ir įvairių. Viename blokelyje galima rašyti kelis veiksmus, jei jie yra to paties tipo (pvz., priskyrimo).
	<p>a) </p> <p>b) </p>	<p>Šis grafinis simbolis skirtas ciklams vaizduoti. Viršutinėje dalyje rašomos ciklo vykdymo sąlygos, o apatinėje – veiksmai. Kairiojoje pusėje paliekama vertikali „kišenė“ rodo ciklo galiojimo sritį.</p> <p>Žinomo kartojimų skaičiaus ciklui rašoma ciklo kintamojo pradinė reikšmė, galutinė reikšmė ir ciklo kintamojo keitimo reikšmė (jei ši reikšmė lygi 1, tuomet ji nerašoma). Reikšmės skiriamos kableliais.</p>

<p>a)</p>		<p>Šis grafinis simbolis skirtas sąlyginiam sakiniui vaizduoti. Viršuje per vidurį rašoma sąlyga. Vienoje trikampėje dalyje paprastai rašoma raidė T (Taip, sąlyga tenkinama), kitoje – N (Ne, sąlyga netenkinama). Apačioje kairiojoje pusėje rašomi veiksmai, kuriuos reikia atlikti, kai sąlyga tenkinama, dešiniojoje – kai netenkinama. Šios dalys gali būti sudėtinės, t.y. sudarytos iš kitų struktūrinių simbolių. Antrasis grafinio simbolio variantas (b) vartojamas tada, kai sąlygai užrašyti reikia daug vietos. Vertikaliąjį veiksmų skiriamąjį liniją gali būti brėžiama nebūtinai per vidurį.</p>
<p>b)</p>		

### 3.1. Sumos skaičiavimo algoritmas

Pradžia Suma
suma = 0
i = 0, n-1
suma = suma + A[i]
Grąžinti suma
Pabaiga

```
// Apskaičiuoja ir grąžina masyvo A(n) reikšmių sumą
int Suma(int A[], int n)
{
    int suma = 0;
    for (int i = 0; i < n; i++)
        suma = suma + A[i];
    return suma;
}
```

### 3.2. Sandaugos skaičiavimo algoritmas

Pradžia Sandauga
sand = 1
i = 0, n-1
sand = sand * A[i]
Grąžinti sand
Pabaiga

```
// Apskaičiuoja ir grąžina masyvo A(n) reikšmių sandaugą
int Sandauga(int A[], int n)
{
    int sand = 1;
    for (int i = 0; i < n; i++)
        sand = sand * A[i];
    return sand;
}
```

### 3.3. Kiekio skaičiavimo algoritmas

Pradžia TeigKiekis
kiekt = 0
i = 0, n-1
A[i] > 0
kiekt = kiekt + 1
Grąžinti kiekt
Pabaiga

```
// Apskaičiuoja ir grąžina masyvo A(n) teigiamųjų reikšmių kiekį
int TeigKiekis(int A[], int n)
{
    int kiekt = 0;
    for (int i = 0; i < n; i++)
        if (A[i] > 0)
            kiekt = kiekt + 1;
    return kiekt;
}
```

### 3.4. Aritmetinio vidurkio skaičiavimo algoritmas

Pradžia AritmVidurkis
suma = 0
i = 0, n-1
suma = suma + A[i]
Grąžinti suma / n
Pabaiga

```
// Apskaičiuoja ir grąžina masyvo A(n) reikšmių aritmetinį vidurkį
double AritmVidurkis(int A[], int n)
{
    int suma = 0;
    for (int i = 0; i < n; i++)
        suma = suma + A[i];
    double y = (double) suma / n;
    return y;
}
```

### 3.5. Didžiausios reikšmės paieškos algoritmas

Pradžia Didžiausias
vieta = 0;
i = 1, n-1
T A[i] > A[vieta] N
vieta = i
Grąžinti vieta
Pabaiga

```
// Randa ir grąžina didžiausios masyvo A(n) reikšmės vietą masyve
int Didžiausias(int A[], int n)
{
    int vieta = 0;
    for (int i = 1; i < n; i++)
        if (A[i] > A[vieta])
            vieta = i;
    return vieta;
}
```

### 3.6. Reikšmės šalinimo algoritmas

Pradžia Salinti
i = k, n-2
A[i] = A[i+1]
n = n - 1
Pabaiga

```
// Pašalina iš masyvo A(n) k-ąją reikšmę
void Salinti(int A[], int &n, int k)
{
    for (int i = k; i < n - 1; i++)
        A[i] = A[i+1];
    n = n - 1;
}
```

### 3.7. Reikšmės įterpimo algoritmas

Pradžia Iterpti
i = n, k+1, -1
A[i] = A[i-1]
n = n + 1
A[k] = b
Pabaiga

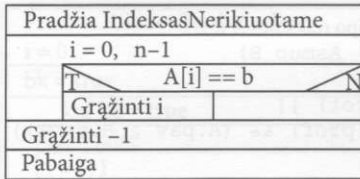
```
// Įterpia į masyvą A(n) k-oje vietoje reikšmę b
void Iterpti(int A[], int &n, int k, int b)
{
    for (int i = n; i > k; i--)
        A[i] = A[i-1];
    n = n + 1;
    A[k] = b;
}
```

### 3.8. Rikiavimo išrinkimo būdu algoritmas

Pradžia Rikiuoti
i = 0, n-2
mini = i
j = i+1, n-1
T A[j] > A[mini] N
mini = j
b = A[i]
A[i] = A[mini]
A[mini] = b
Pabaiga

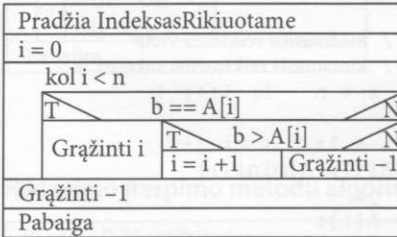
```
// Rikiuoja masyvo A(n) reikšmes didėjančiai
void Rikiuoti(int A[], int n)
{
    int mini; // mažiausios reikšmės vieta
    int b; // kintamasis reikšmėms sukeisti
    for (int i = 0; i < n - 1; i++) {
        mini = i;
        for (int j = i + 1; j < n; j++)
            if (A[j] < A[mini])
                mini = j;
        b = A[i];
        A[i] = A[mini];
        A[mini] = b;
    }
}
```

### 3.9. Paieškos nerikiuotame masyve algoritmas



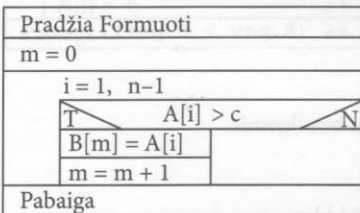
```
// Gražina surastos masyve A(n) reikšmės b indeksą arba
// -1, jei tokios reikšmės nebuvo
int IndeksasNerikiuotame(int A[], int n, int b)
{
    for (int i = 0; i < n; i++)
        if (A[i] == b)
            return i; // reikšmė surasta
    return -1; // reikšmės nerasta
}
```

### 3.10. Paieškos surikiuotame masyve algoritmas



```
// Gražina surastos masyve A(n) reikšmės b indeksą arba
// -1, jei tokios reikšmės masyve nebuvo
// Pastaba: masyvo A(n) reikšmės surikiuotos didėjančiai
int IndeksasRikiuotame(int A[], int n, int b)
{
    int i = 0;
    while (i < n)
        if (b == A[i])
            return i; // reikšmė surasta
        else if (b > A[i])
            i = i + 1;
        else
            return -1; // reikšmės nerasta
    return -1; // reikšmės nerasta
}
```

### 3.11. Naujo masyvo formavimo algoritmas



```
// Masyvo A(n) reikšmės, didesnes už c, surašo į masyvą B(m)
void Formuoti(int A[], int n, int c,
              int B[], int & m)
{
    m = 0;
    for (int i = 0; i < n; i++)
        if (A[i] > c) {
            B[m] = A[i];
            m = m + 1;
        }
}
```

### 3.12. Kiti rikiavimo algoritmai

Tarkime, turime tokį struktūros tipą:

```
struct Asmuo {
    string pav; // pavardė ir vardas
    string prof; // profesija
    int metai; // darbo stažas metais
};
```

Duomenys, kuriuos reikia surikiuoti pagal profesiją ir asmenį (pavardė, vardas), yra masyvas A:

```
Asmuo A[1000]; int n;
```

## Rikiavimas išrinkimo būdu

Pradžia Raktas	
(A.prof > B.prof) arba ((A.prof == B.prof) ir (A.pav > B.pav))	
T	N
Gražinti true	Gražinti false
Pabaiga	

```
// Palygina dvi struktūros tipo reikšmes A ir B
bool Raktas(Asmuo A, Asmuo B)
{
    if ((A.prof > B.prof) ||
        ((A.prof == B.prof) && (A.pav > B.pav)))
        return true;
    else return false;
}
```

Pradžia Rikiuoti	
i = 0, n-2	
mini = i	
j = i+1, n-1	
T	N
Raktas(A[j], A[mini])	
mini = j	
b = A[i]	
A[i] = A[mini]	
A[mini] = b	
Pabaiga	

```
// Rikiuoja struktūrų masyvo A(n) reikšmes pagal raktą
void Rikiuoti(Asmuo A[], int n)
{
    int mini; // mažiausios reikšmės vieta
    Asmuo b; // kintamasis reikšmėms sukeisti
    for (int i = 0; i < n - 1; i++) {
        mini = i;
        for (int j = i + 1; j < n; j++)
            if (Raktas(A[j], A[mini]))
                mini = j;
        b = A[i];
        A[i] = A[mini];
        A[mini] = b;
    }
}
```

Jeigu struktūrų masyvą reikėtų surikiuoti pagal visus tris požymius (iš eilės): profesiją, darbo stažą ir asmens pavardę bei vardą, tai raktas būtų toks:

```
if ((A.prof > B.prof) ||
    ((A.prof == B.prof) && (A.metai > B.metai)) ||
    ((A.prof == B.prof) && (A.metai == B.metai) && (A.pav > B.pav)))
    return true;
else return false;
```

Žinoma, kad tokį loginį reiškinį galima užrašyti paprasčiau, tačiau užrašas bus ilgesnis:

```
if ((A.prof > B.prof)
    return true;
else if ((A.prof == B.prof) && (A.metai > B.metai))
    return true;
else if ((A.prof == B.prof) && (A.metai == B.metai) && (A.pav > B.pav))
    return true;
else return false;
```

## Rikiavimo burbuliuko (porinių sukeitimų) metodu algoritmas (variantas A)

Pradžia RikiuotiBurbulasA	
i = 0, n-2	
j = n-1, i+1, -1	
T	N
A[j] < A[j-1]	
c = A[j]	
A[j] = A[j-1]	
A[j-1] = c	
Pabaiga	

```
// Rikiuoja masyvo A(n) reikšmes didėjančiai
void RikiuotiBurbulasA(int A[], int n)
{
    int c; // pagalbiniis kintamasis
    for (int i = 0; i < n - 1; i++)
        for (int j = n - 1; j > i; j--)
            if (A[j] < A[j-1]) {
                c = A[j];
                A[j] = A[j-1];
                A[j-1] = c;
            }
}
```

## Rikiavimo burbuliuko (porinių sukeitimų) metodu algoritmas (variantas B)

Pradžia RikiuotiBurbulasB
i = 0
bk = true
kol bk = true
bk = false
i = i + 1
j = n - 1, i, -1
T A[j] < A[j-1] N
bk = true
c = A[j]
A[j] = A[j-1]
A[j-1] = c
Pabaiga

```
// Rikiuoja masyvo A(n) reikšmes didėjančiai
void RikiuotiBurbulasB(int A[], int n)
{
    int i = 0, c; // pagalbinis kintamasis
    bool bk = true; // buvo keitimai
    while (bk) { // kol yra sukeičiamų vietomis reikšmių
        bk = false;
        i = i + 1;
        for (int j = n - 1; j > i - 1; j--)
            if (A[j] < A[j-1]) {
                bk = true;
                c = A[j];
                A[j] = A[j-1];
                A[j-1] = c;
            }
    }
}
```

## Rikiavimo įterpimo metodu algoritmas

Pradžia RikiuotiIterpimu
i = 1, n-1
b = A[i]
j = 0
kol b < A[j]
j = j + 1
k = i-1, j, -1
A[k+1] = A[k]
A[j] = b
Pabaiga

```
// Rikiuoja masyvo A(n) reikšmes didėjančiai
void RikiuotiIterpimu(int A[], int n)
{
    int i, j, k, b;
    for (int i = 1; i < n; i++) {
        b = A[i]; // nesurikiuotos reikšmės paėmimas
                // Įterpimo pozicijos paieška
        j = 0;
        while (b < A[j]) j = j + 1;
                // Reikšmių postūmis įterpimo pozicijai atlaisvinti
        for (int k = i - 1; k > j - 1; k--)
            A[k+1] = A[k];
        A[j] = b; // paimtos reikšmės įterpimas
    }
}
```