Pasvalio Petro Vileišio gimnazija

KRISTUPAS JAKUBONIS

**KTU programavimo pamokėlių užduočių sprendimas**

Projektinis darbas

Darbo konsultantė Dalia Monkevičienė, informacinių technologijų mokytoja metodininkė

**Pasvalys, 2020**

# Turinys

[Įvadas 4](#_Toc41577339)

[Dėstymas 5](#_Toc41577340)

[Užduotys pagal temas 6](#_Toc41577341)

[Funkcijos, grąžinančios apskaičiuotą reikšmę per vardą 6](#_Toc41577342)

[1. Kuriam amžiui priklauso metai? 6](#_Toc41577343)

[2. Laikas 7](#_Toc41577344)

[3. Knygų pasirinkimas 8](#_Toc41577345)

[4. Šešioliktainė sistema 10](#_Toc41577346)

[5. Datos dekodavimas 12](#_Toc41577347)

[6. Fibonačio sekos savybės 15](#_Toc41577348)

[7. Cikliškas skaičiaus skaitmenų perstatymas 18](#_Toc41577349)

[8. Balsu valdomo roboto kelias 20](#_Toc41577350)

[Funkcijos su parametrais-nuorodomis. 22](#_Toc41577351)

[1. Komandos 22](#_Toc41577352)

[2. Dovanų metas 24](#_Toc41577353)

[3. Santaupos 27](#_Toc41577354)

[4. Kambarių sienų dažymas 29](#_Toc41577355)

[5. Senovės Anglijos pinigai 32](#_Toc41577356)

[6. Civilinio karo simuliavimas 35](#_Toc41577357)

[7. Miltai 39](#_Toc41577358)

[8. Namų darbai 41](#_Toc41577359)

[9. Informacijos matavimo vienetai 43](#_Toc41577360)

[10. Geriausias krepšininkas 45](#_Toc41577361)

[11. Varikliukų pakavimas 48](#_Toc41577362)

[12. Kambarių sienų dažymas 50](#_Toc41577363)

[Masyvas. Masyvo elementų sumos, kiekio, vidurkio skaičiavimas. 53](#_Toc41577364)

[1. Pirkėjų srautai 53](#_Toc41577365)

[2. Krepšinio čempionatas 56](#_Toc41577366)

[3. Sodinukai 59](#_Toc41577367)

[4. Detalės 62](#_Toc41577368)

[5. Įmonės sandoriai 65](#_Toc41577369)

[6. Tritaškių konkursas 68](#_Toc41577370)

[7. Populiarumo konkursas 70](#_Toc41577371)

[8. Ateiviai žemėje: iššūkis 72](#_Toc41577372)

[9. Ateiviai žemėje: kelionė namo 75](#_Toc41577373)

[10. Finansų programėlė 77](#_Toc41577374)

[11. Pirmasis automobilis 79](#_Toc41577375)

[12. Studentų stipendijos 81](#_Toc41577376)

[13. Akcijų rinkos statistika 84](#_Toc41577377)

[14. Susitikimų kelionės 86](#_Toc41577378)

[15. Kelionė 89](#_Toc41577379)

[Naujo masyvo formavimas iš kelių masyvų. Didžiausios ir mažiausios masyvo elementų reikšmių vieta. 91](#_Toc41577380)

[1. Dalelė 91](#_Toc41577381)

[2. Spalvoti taškai 94](#_Toc41577382)

[3. Paprasti uždaviniai 98](#_Toc41577383)

[4. Protmūšis 101](#_Toc41577384)

[5. Sportininkai 104](#_Toc41577385)

[6. Pasirinkimas 108](#_Toc41577386)

[Išvados 110](#_Toc41577387)

[Literatūra 111](#_Toc41577388)

[Priedai 112](#_Toc41577389)

# Įvadas

Kauno technologijos universitetas nuo 2013 metų rengia nuotolinį ir nemokamą mokymų ciklą pavadinimu „Programavimo pamokėlės“. Šie mokymai jungia visus moksleivius, norinčius išmokti programuoti C++ programavimo kalba. Oraganozatoriai suteikia visą savarankiškam mokymuisi reikalingą medžiagą: vaizdo pamokas, teorinę medžiagą, praktines užduotis, konsultanto pagalbą. Šis mokymų ciklas vyksta lapkričio–birželio mėnesiais. Kiekvieną mėnesį įkeliama teorinė medžiaga ir ją atitinkančios praktinės užduotys.

Programavimu domiuosi jau ketverius metus. Viskas prasidėjo nuo programavimo pamokų su „Python“ programavimo kalba. Nors iš pradžių buvo sunkoka, tačiau šis užsiėmimas mane labai sudomino. Šį kūrybinį darbą pasirinkau, nes visi užsiėmimai, paskaitos, programavimo pamokos mane labai įtraukė, be galo domino. Manau, kad ateitį siesiu su šia kryptimi, todėl šis kūrybinis darbas man padės įvertinti savo turimas žinias ir suprasti IT srities projektinio darbo svarbą.

**Tema**. KTU programavimo pamokėlių sprendimas.

**Tikslas**.Naudojantis nuotolinio mokymosi svetainės suteikta medžiaga išspręsti programavimo pamokėlių užduotis.

**Uždaviniai**.

1. Išmokti nagrinėti ir įsiskaityti į programavimo užduotis.
2. Parašyti programas pateiktų užduočių sprendimui.
3. Išanalizuoti savo parašomą programinį kodą, stengtis jį padaryti kuo geresnį.
4. Išmokti planuoti savo laiką ir atitinkamai jį paskirstyti skiriamoms užduotims.
5. Aprašyti užduotis ir suformuluoti išvadą.

# Dėstymas

Pateiktos programavimo užduočių sąlygos, reikalavimai ir sprendimai. Visi sprendimai atitinka užduočių reikalavimus ir atspausdina teisingus atsakymus. Darbą apsunkino tai, kad neveikė uždavinių testavimo sistema, todėl teko pačiam galvoti papildomus testus uždaviniams patikrinti. Iš viso padariau 41 užduotį, kurios susideda iš 2019–2020 metų programavimo pamokėlių.

Užduotys yra suskirstytos į 4 temas:

1. Funkcijos, grąžinančios apskaičiuotą reikšmę per vardą.
2. Funkcijos su parametrais-nuorodomis.
3. Masyvas. Masyvo elementų sumos, kiekio, vidurkio skaičiavimas.
4. Naujo masyvo formavimas iš kelių masyvų. Didžiausios ir mažiausios masyvo elementų reikšmių vieta.

## Užduotys pagal temas

### Funkcijos, grąžinančios apskaičiuotą reikšmę per vardą

#### Kuriam amžiui priklauso metai?

1. **Užduotis**

Yra priimta: jei turime metus nuo 1900 iki 1999, tai jie yra 20 amžiaus metai, jei metai yra iš intervalo 1800-1899, tuomet turime 19 amžiaus metus ir pan. Parašykite programą, kuri iš pradinių duomenų failo **amzius\_data.txt** perskaitytų metus **m**, ir į rezultatų failą **amžius\_res.txt** įrašytų, kuriam amžiui **a** priklauso metai **m**.

|  |  |
| --- | --- |
| amzius\_data.txt | amzius\_rez.txt |
| 1968 | 20 |

1. **Reikalavimai**

* Parašykite funkciją **int Amzius (int m)**, grąžinančią nustatytą amžiaus reikšmę per funkcijos vardą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

int Amzius (int m);

int main(){

int m;

ifstream df ("amzius\_data.txt");

ofstream rf ("amzius\_res.txt");

df >> m;

rf << Amzius(m);

df.close();

rf.close();

return 0;

}

int Amzius (int m){

m = m / 1000 \* 10 + m / 100 % 10 + 1;

return m;

}

#### Laikas

1. **Užduotis**

Parašykite programą, skaičiuojančią, kelinta valandos sekundė **sd** yra dabar, jei nuo paros pradžios praėjo **s** sekundžių. Pradinis duomuo **s** perskaitomas iš pradinių duomenų failo **laikas\_data.txt**. Rezultatas įrašomas į failą **laikas\_res.txt**.

|  |  |
| --- | --- |
| laikas\_data.txt | laikas\_res.txt |
| 3615 | 15 |

1. **Reikalavimai**

* Parašykite funkciją **int Sekundes (int s)**, grąžinančią, kelinta valandos sekundė yra dabar, per funkcijos vardą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

int Sekundes (int s);

int main(){

int m;

ifstream df ("laikas\_data.txt");

ofstream rf ("laikas\_res.txt");

df >> m;

rf << Sekundes(m);

df.close();

rf.close();

return 0;

}

int Sekundes (int s){

s = s % 60;

return s;

}

#### Knygų pasirinkimas

1. **Užduotis**

Internetinė knygų parduotuvė paskelbė akciją. Eglė nusprendė knygoms paskirti **k** (realusis skaičius) eurų. Mergaitė renkasi norimas knygas iš sąrašo. Jei norimai knygai įsigyti užtenka pinigų, tuomet ji knygą deda į pirkinių krepšelį, jei ne – eina prie kitos knygos ir t.t. Pirmoje pradinių duomenų failo **knygos\_data.txt** eilutėje įrašyta pinigų suma **k** (realusis skaičius), bei knygų skaičius **n** (sveikasis skaičius). Tolesnėse **n** eilučių įrašyta po vieną sveikąjį skaičių **kd** – knygos kodas ir po vieną realųjį skaičių **kk** – knygos kaina.

Parašykite programą, kuri į rezultatų failą **knygos\_res.txt** išvestų **n** eilučių su informacija: knygos kodas, knygos kaina, kaip keičiasi (mažėja) Eglės turimų pinigų suma, ir ar Eglė gali įsigyti knygą (GALI arba NEGALI).

|  |  |
| --- | --- |
| knygos\_data.txt | knygos\_res.txt |
| 15.25 5  123 7.25  254 3.45  451 5.46  256 8.46  562 4.55 | 123 7.25 8.00 GALI  254 3.45 4.55 GALI  451 5.46 4.55 NEGALI  256 8.46 4.55 NEGALI  562 4.55 0.00 GALI |

1. **Reikalavimai**

* Parašykite funkciją **double Lieka (double suma, double knygos\_kaina)**, grąžinančią, kokia pinigų suma lieka Eglei įsigijus knygą iš sąrašo;
* Parašykite funkciją **string ArGali(double suma, double knygos\_kaina)**, grąžinanti žodį GALI, jei Eglė gali įsigyti knygą, arba NEGALI, jei knygai įsigyti trūksta pinigų.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

double Lieka (double suma, double knygos\_kaina);

string ArGali(double suma, double knygos\_kaina);

int main(){

int kiek, i, nr;

double pin, kain;

ifstream df ("knygos\_data.txt");

ofstream rf ("knygos\_res.txt");

df >> pin >> kiek;

for (i = 1; i <= kiek; i++){

df >> nr >> kain;

rf << nr << " " << kain << " " << fixed << setprecision(2) << Lieka(pin, kain) << " " << ArGali(pin, kain) << endl;

pin = Lieka(pin, kain);

}

rf.close();

df.close();

return 0;

}

double Lieka (double suma, double knygos\_kaina)

{

if (ArGali(suma, knygos\_kaina) == "GALI"){

return suma - knygos\_kaina;

}

else return suma;

}

string ArGali(double suma, double knygos\_kaina)

{

if (suma - knygos\_kaina >= 0) return "GALI";

else return "NEGALI";

}

#### Šešioliktainė sistema

1. **Užduotis**

Pradinių duomenų faile **data.txt** įrašytas dešimtainis natūralusis skaičius. Parašykite programą, kuri pervestų skaičių į šešioliktainę skaičiavimo sistemą ir įrašytų šešioliktainį skaičių į failą **res.txt**

|  |  |
| --- | --- |
| data.txt | res.txt |
| 165 | A5 |

1. **Reikalavimai**

* Parašykite funkciją **String Sesioliktainis(long desimtainis)**, kuri grąžina šešioliktainį skaičių.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

#include <string>

using namespace std;

string Sesioliktainis(long desimtainis);

int main(){

int sk;

ifstream df ("data.txt");

ofstream rf ("res.txt");

df >> sk;

rf << Sesioliktainis(sk);

df.close();

rf.close();

return 0;

}

string Sesioliktainis(long desimtainis)

{

int liek;

string ats;

while (desimtainis > 0){

liek = desimtainis % 16;

desimtainis = desimtainis / 16;

if (liek == 15) ats = "F" + ats;

else if (liek == 14)ats = "E" + ats;

else if (liek == 13)ats = "D" + ats;

else if (liek == 12)ats = "C" + ats;

else if (liek == 11)ats = "B" + ats;

else if (liek == 10)ats = "A" + ats;

else if (liek == 9)ats = "9" + ats;

else if (liek == 8)ats = "8" + ats;

else if (liek == 7)ats = "7" + ats;

else if (liek == 6)ats = "6" + ats;

else if (liek == 5)ats = "5" + ats;

else if (liek == 4)ats = "4" + ats;

else if (liek == 3)ats = "3" + ats;

else if (liek == 2)ats = "2" + ats;

else if (liek == 1)ats = "1" + ats;

else ats = "0" + ats;

}

return ats;

}

#### Datos dekodavimas

1. **Užduotis**

Datos pateikiamos tokiu formatu: pirmieji keturi skaitmenys - metai, sekantys du skaitmenys - mėnuo, paskutiniai du skaitmenys - diena. Parašykite programą, kuri paverstų kiekvieną aštuonių skaitmenų kratinį į žmogui lengvai skaitomą datą.

Pirmoje pradinių duomenų failo **datos\_data.txt** eilutėje įrašytas datų kiekis **n**. Sekančiose **n** eilučių įrašyta po vieną datą **data** aukščiau aprašytu formatu.

Rezultatų faile **datos\_res.txt** turi būti išspausdinta **n** eilučių - kiekvienoje eilutėje atitinkama dekoduota data (tokiu formatu, koks pateiktas pavyzdyje).

PASTABOS:

1. Pradinių duomenų faile pateiktos datos gali būti klaidingos, pavyzdžiui, 20010230 - vasaris trisdešimties dienų turėti negali (tai tik vienas galimas atvejis, apgalvokite ir daugiau galimų atvejų, kai per parametrus perduotos reikšmės reiškia neteisingą datą). Tokiu atveju rezultatų failo atitinkamoje eilutėje išspausdinkite žodį "KLAIDA" (didžiosiomis raidėmis, be taško pabaigoje).
2. Neignoruokite keliamųjų metų. Pavyzdžiui, nors 2019-aisiais vasaris turėjo tik 28 dienas, 2020-aisiais jis turės 29 dienas. Žinoma, programa turi veikti su bet kokiomis datomis, ne tik 2020 metais.
3. Nenaudokite lietuviškų raidžių.
4. Naudokite funkcijų prototipus (t.y. apsirašykite grąžinamą reikšmę, pavadinimą bei parametrus virš main(), tačiau funkcijos kodą rašykite po main()).

|  |  |
| --- | --- |
| datos\_data.txt | datos\_res.txt |
| 3  20010230  20191101  20150312 | KLAIDA  2019 m. lapkricio 1 d.  2015 m. kovo 12 d. |

1. **Reikalavimai**

* Parašykite funkciją **int Data(int data, int kas)**, grąžinančią iš aštuonių skaitmenų formato išsskirtus metus, mėnesį, arba dienas. **kas** - parametras, kuris nurodo, ką reikia išskirti. Jeigu jis lygus 1, reikia išskirti metus, jeigu jis lygus 2, reikia išskirti mėnesį, bet kokiu kitu atveju reikia išskirti dienas.
* Parašykite funkciją **bool ArTinkama(int metai, int menuo, int diena)**, grąžinančią reikšmę true, jeigu nurodyta data yra teisingo formato, arba false, jeigu nurodyta data yra neteisingo formato.
* Parašykite funkciją **string Menuo(int menuo)**, grąžinančią nurodyto mėnesio žodinį pavadinimą vienaskaitos kilmininko linksnyje.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

int Data(int data, int kas);

bool ArTinkama(int metai, int menuo, int diena);

string Menuo(int menuo);

int main(){

int n, i, met, men, dien, data;

ifstream df ("datos\_data.txt");

ofstream rf ("datos\_res.txt");

df >> n;

for (i = 1; i <= n; i++){

df >> data;

met = Data(data,1);

men = Data(data,2);

dien = Data(data,3);

if (ArTinkama(met, men, dien))rf << met << " m. " << Menuo(men) << " " << dien << " d." << endl;

else rf << "KLAIDA" << endl;

}

rf.close();

df.close();

return 0;

}

int Data(int data, int kas){

if (kas == 1){

data = data / 10000;

}

else if (kas == 2){

data = data / 100 % 100;

}

else data = data % 100;

return data;

}

bool ArTinkama(int metai, int menuo, int diena)

{

if (metai < 0 || menuo <= 0 || diena <= 0 || menuo > 12 || diena > 31) return false;

else if (menuo == 2 && metai % 4 != 0 && diena > 28) return false;

else if (metai % 4 == 0 && menuo == 2 && diena > 29) return false;

else if (menuo == 4 && diena > 30 || menuo == 6 && diena > 30 || menuo == 9 && diena > 30 || menuo == 11 && diena > 30)return false;

else return true;

}

string Menuo(int menuo)

{

if (menuo == 1)return "sausio";

else if (menuo == 2)return "vasario";

else if (menuo == 3)return "kovo";

else if (menuo == 4)return "balandzio";

else if (menuo == 5)return "geguzes";

else if (menuo == 6)return "birzelio";

else if (menuo == 7)return "liepos";

else if (menuo == 8)return "rugpjucio";

else if (menuo == 9)return "rugsejo";

else if (menuo == 10)return "spalio";

else if (menuo == 11)return "lapkricio";

else return "gruodzio";

}

#### Fibonačio sekos savybės

1. **Užduotis**

**Fibonačio skaičių seka** – sveikųjų skaičių seka, nusakoma taip: F1 = 1, F2 = 1, Fn = Fn-2 + Fn-1. Seka prasideda šiais skaičiais: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233. Kiekvienas šios sekos skaičius lygus dviejų prieš jį einančių skaičių sumai.

Knygoje „Matematikos pasaulyje“ pateikiamos tokios Fibonačio sekos savybės:

1) sekos kiekvienas trečias skaičius yra lyginis;

2) sekos kiekvienas ketvirtas skaičius dalijasi iš 3;

3) sekos kiekvienas penktas skaičius dalijasi iš 5;

4) sekos kiekvienas penkioliktas skaičius dalijasi iš 10.

Parašykite programą, kuri patikrintų n (3 ≤ n ≤ 93) narių Fibonačio seką ir išspausdintų tuos narius, kurie atitinka aukščiau nurodytas savybes.

Pradinių duomenų faile **fibonati\_data.txt** įrašyta **n** reikšmė. Rezultatų faile **fibonati\_res.txt** turi būti įrašyti Fibonačio sekos nariai, tenkinantys užduotyje nurodytas sekos savybes: nario reikšmė, kelintas tai yra sekos narys, kurią savybę tenkina.

|  |  |
| --- | --- |
| fibonati\_data.txt | fibonati\_res.txt |
| 13 | 2 3 1  3 4 2  5 5 3  8 6 1  21 8 2  34 9 1  55 10 3  144 12 1  144 12 2 |

1. **Reikalavimai**

* Parašykite funkciją **unsigned long long Narys(unsigned long long f1, unsigned long long f2)**, grąžinančią naują Fibonačio sekos narį.
* Parašykite funkciją **int Savybe(unsigned long long narys, int kelintas)**, grąžinančią, kurią savybę atitinka Fibonačio sekos narys. Kadangi vienas sekos narys gali atitikti kelias savybes, ši funkcija turi nebūtinai grąžinti tikslią savybę. Ji turėtų grąžinti tokį skaičių, iš kurio eitų sužinoti, kurias savybes atitinka narys.
* Programoje gali būti daugiau funkcijų, grąžinančių apskaičiuotą reikšmę per funkcijos vardą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

unsigned long long Narys(unsigned long long f1, unsigned long long f2);

int Savybe(unsigned long long narys, int kelintas);

int main(){

int n, n1, pir = 1, antr = 1, temp, real;

ifstream df ("fibonati\_data.txt");

ofstream rf ("fibonati\_res.txt");

df >> n;

n1 = n;

n = n - 2;

while (n > 0){

temp = Narys(pir, antr);

antr = pir;

pir = temp;

n = n - 1;

real = Savybe(temp, n1 - n);

while (real > 0){

if (Savybe(temp, n1 - n))rf << temp << " " << n1 - n << " " << real % 10 << endl;

real = real / 10;

}

}

rf.close();

df.close();

return 0;

}

unsigned long long Narys(unsigned long long f1, unsigned long long f2)

{

return f1 + f2;

}

int Savybe(unsigned long long narys, int kelintas)

{

int indeks = 0, sk = 0;

if (kelintas % 3 == 0){

sk = sk + 1 \* pow(10, indeks);

indeks++;

}

if (kelintas % 4 == 0){

sk = sk + 2 \* pow(10, indeks);

indeks++;

}

if (kelintas % 5 == 0){

sk = sk + 3 \* pow(10, indeks);

indeks++;

}

if (kelintas % 15 == 0){

sk = sk + 4 \* pow(10, indeks);

indeks++;

}

if (sk > 0)return sk;

else return false;

}

#### Cikliškas skaičiaus skaitmenų perstatymas

1. **Užduotis**

Parašykite programą, kuri **cikliškai** perstatytų iš failo **cikliskas\_data.txt** perskaityto skaičiaus **x** (**x** yra long tipo) skaitmenis ir naujai gautus skaičius surašytų faile **cikliskas\_res.txt** po vieną eilutėje.

|  |  |
| --- | --- |
| cikliskas\_data.txt | cikliskas\_res.txt |
| 578 | 578  857  785 |

1. **Reikalavimai**

* Parašykite funkciją **long Naujas (long x)**, grąžinančią naujai suformuotą skaičių atlikus ciklinį skaitmenų perstatymą.
* Programoje gali būti daugiau funkcijų, grąžinančių apskaičiuotą reikšmę per funkcijos vardą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

long Naujas (long x);

int main(){

long n, n1;

ifstream df ("cikliskas\_data.txt");

ofstream rf ("cikliskas\_res.txt");

df >> n;

rf << n << endl;

n1 = n;

n1 = n1 / 10;

while (n1 > 0){

n = Naujas(n);

rf << n << endl;

n1 = n1 / 10;

}

rf.close();

df.close();

return 0;

}

long Naujas (long x)

{

long x2;

int k = -1;

x2 = x;

while (x2 > 0){

x2 = x2 / 10;

k++;

}

return x % 10 \* pow(10, k) + x / 10;

}

#### Balsu valdomo roboto kelias

1. **Užduotis**

Parašykite programą, skaičiuojančią atstumą tarp taškų A ir B. Pradinių duomenų failo **robotas\_data.txt** pirmoje eilutėje įrašytas komandų skaičius **n**. Tolesnėse **n** eilučių įrašyta po komandą: K – kairėn, D – dešinėn, A – aukštyn, Z – žemyn. Rezultatų faile **robotas\_res.txt** turi būti įrašytas atstumas tarp taškų A ir B dviejų skaitmenų po kablelio tikslumu.

|  |  |
| --- | --- |
| robotas\_data.txt | robotas\_res.txt |
| 3  D  D  A | 2.24 |

1. **Reikalavimai**

* Parašykite funkciją **int Koordinate (char komanda)**, grąžinančią koordinatės, kuri pasikeitė įvykdžius komandą, reikšmę. Pvz., jei robotas vykdo komandą D – jo koordinatė x padidėja 1.
* Parašykite funkciją **double Atstumas (int x, int y)**, grąžinančią atstumą tarp taškų A ir B.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

int Koordinate (char komanda);

double Atstumas (int x, int y);

int main(){

int n, i, x = 0, y = 0;

char nurod;

ifstream df ("robotas\_data.txt");

ofstream rf ("robotas\_res.txt");

df >> n;

for (i = 1; i <= n; i++){

df >> nurod;

if (nurod == 'A' || nurod == 'D')x = x + Koordinate(nurod);

else y = y + Koordinate(nurod);

}

rf << fixed << setprecision(2) << Atstumas(x, y);

df.close();

rf.close();

return 0;

}

int Koordinate (char komanda)

{

if (komanda == 'D' || komanda == 'A') return 1;

else return -1;

}

double Atstumas (int x, int y)

{

return sqrt(pow(x, 2) + pow(y, 2));

}

### Funkcijos su parametrais-nuorodomis.

#### Komandos

1. **Užduotis**

Grupė draugų prieš pasiskirstydami į komandas sugalvojo tokį būdą:

* kiekvienas iš draugų užrašo ant lapelio po tris skaičius iš sveikųjų skaičių intervalo [1; 20] ir visus lapelius sudeda į krūvą.
* Kiekviename lapelyje paliekami didžiausias ir mažiausias užrašyti skaičiai.
* Jei didžiausi, arba mažiausi skaičiai yra keli, tuomet paliekamas tik vienas iš jų. Jei visi skaičiai vienodi, tai paliekami du iš jų.

Pavyzdžiui:

* jei lapelyje įrašyti skaičiai 1 5 1, tuomet turi likti 1 ir 5.
* jei įrašyti skaičiai 5 1 5, tuomet turi likti 1 ir 5.
* jei įrašyti skaičiai 5 5 5, tuomet turi likti 5 ir 5.
* Tie žaidėjai, kurių lapeliuose likusių skaičių suma yra lyginė, patenka į pirmą komandą, o tie, kurių lapeliuose likusių skaičių suma nelyginė – į antrą.

Parašykite programą, kuri į rezultatų failą **komandarez.txt** išvestų kiekvieno draugo lapelyje likusius neišbrauktus skaičius ir į kurią komandą draugas pateks.

Vienam draugui skiriama viena eilutė.

Pradinių duomenų failo **komanda.txt** pirmoje eilutėje įrašytas draugų skaičius **n**.

Tolesnėse **n** eilučių įrašyta po tris sveikuosius skaičius nuo 1 iki 20 – kiekvieno draugo sugalvoti skaičiai.

|  |  |
| --- | --- |
| komanda.txt | komandarez.txt |
| 6  3 2 3  14 5 9  6 9 17  9 9 2  8 7 8  4 4 4 | 2 3 antra  5 14 antra  6 17 antra  2 9 antra  7 8 antra  4 4 pirma |

1. **Reikalavimai**

* Įvesties failo vardas **komanda.txt**
* Rezultatų failo vardas **komandarez.txt**
* Funkcija **void()**, grąžinanti per parametrus kiekvieno draugo lapelyje likusius neišbrauktus skaičius ir į kurią komandą draugas pateks.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Komanda(int sk, int sk1, int sk2, int &maz, int &did, string &priklausymas);

int main(){

int n, maz, did, sk, sk1, sk2;

string priklausymas;

ifstream df ("komanda.txt");

ofstream rf ("komandarez.txt");

df >> n;

for (int i = 0; i < n; i++){

df >> sk >> sk1 >> sk2;

Komanda(sk, sk1, sk2, maz, did, priklausymas);

rf << maz << " " << did << " " << priklausymas << endl;

}

df.close();

rf.close();

return 0;

}

void Komanda(int sk, int sk1, int sk2, int &maz, int &did, string &priklausymas)

{

maz = sk;

did = sk;

if (maz > sk1) maz = sk1;

if (maz > sk2) maz = sk2;

if (did < sk1) did = sk1;

if (did < sk2) did = sk2;

if ((maz + did)%2 == 0)priklausymas = "pirma";

else priklausymas = "antra";

}

#### Dovanų metas

1. **Užduotis**

Artėjant kalėdoms, mokykloje buvo nuspręsta pradžiuginti pažangius moksleivius. Mokytojos pasiūlė padaryti staigmeną gerai besimokantiems mokiniams. Moksleiviai, kurių trimestro vidurkis yra įvertintas daugiau **nei 8 balais**, laikomi pažangiais mokiniais. Jiems nuspręsta prieš kalėdų atostogas įteikti kalėdinių saldainių dėžutę. Šią akciją nusprendė paremti netoliese esanti parduotuvė, kuri sutiko taikyti didesnes nuolaidas, priklausomai nuo nupirktų saldainių dėžučių kiekio, kiekvienai klasės auklėtojai. Jei perkama **4** ar daugiau dėžučių taikoma **25%** nuolaidą, jei **7** ar daugiau – **35%**, o jei daugiau nei **10** – net **40%** nuolaida.

Jūsų užduotis bus padėti mokytojai sužinoti kiek mokinių gaus kalėdinių dovanų, kokią nuolaidą gaus vienos klasės auklėtoja ir kiek iš viso pinigų už jas sumokės.

Rezultatai:

* Kiek išviso reikės pirkti saldainių dėžučių;
* Kokia vienos dėžutės kaina;
* Kiek iš viso sumokės pinigų.

|  |  |
| --- | --- |
| klase.txt | rezultatai.txt |
| IC  5  8.9  5 8 9 10 5 8  7 8 7 10 7 8 10 9  6 8 9 8 9 10 8  7 5 8 9 7 10 8 9  6 8 9 10 8 7 9 | IC  3  8.9  26.7 |

1. **Reikalavimai**

Reikalavimai, parašykite **void** tipo funkcijas:

* kuri skaitytų pirmą duomenų failą ir per funkcijos parametrus-nuorodas gražintų klasės pavadinimą, mokinių skaičių bei vienos dėžutės kainą;
* kuri apskaičiuotų kiek mokinių gaus dovanų;
* kuri grąžintų naują saldainių dėžutės kainą ir visą reikalingą pinigų sumą

Pradiniai duomenys saugomi dviejuose tekstiniuose failuose. Failas **klase.txt** – pirmoje eilutėje įrašytas klasės pavadinimas, antroje – mokiniu skaičius **n** (0<n<30), trečioje saldainių dėžutės kaina be nuolaidos – **salDezute**, toliau kiekvienoje eilutėje surašyti atskirų mokinių pažymiai, pirmas skaičius – kiek pažymių mokinys turi, toliau – visi jo pažymiai.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Info(string &klase, int &moksk, double &dezk);

void Kiekm(int n, int &mok);

void Kaina(int mok, double &dez, double &vis);

int main(){

string klase;

int n, mok = 0;

double dezk, vis = 0;

ofstream rf ("rezultatai.txt");

Info(klase, n, dezk);

Kiekm(n, mok);

Kaina(mok, dezk, vis);

rf << klase << endl << mok << endl << fixed << setprecision(1) << dezk << endl << vis << endl;

rf.close();

return 0;

}

void Info(string &klase, int &moksk, double &dezk)

{

ifstream df ("klase.txt");

df >> klase >> moksk >> dezk;

df.close();

}

void Kiekm(int n, int &mok)

{

int paz, n1;

double s = 0;

string laik;

ifstream df ("klase.txt");

df >> laik >> laik >> laik;

for (int i = 0; i < n; i++){

df >> n1;

for(int i1 = 0; i1 < n1; i1++){

df >> paz;

s = s + paz;

}

if(s / n1 > 8)mok++;

s = 0;

}

df.close();

}

#### Santaupos

1. **Užduotis**

**n** draugų taupė į taupykles mesdami 1, 2 ir 5 euro centų monetas. Prieš pavasario atostogas draugai nusprendė iškeisti turimus pinigus į didesnės vertės monetas: 2 ir 1 euro, bei 50, 20 ir 10 euro centų. Jei keičiant pinigus liktų neiškeistų smulkių centų, juos paaukos beglobiams gyvūnams.

Pirmoje pradinių duomenų failo **santaupos.txt** eilutėje įrašytas draugų skaičius **n**. Tolesnėse n eilučių įrašyta po 3 sveikuosius skaičius vieno žmogaus turimų 1, 2 ir 5 centų vertės monetų kiekiai.

Rezultatų failo **santauposrez.txt** failo yra **n** eilučių, kuriose turi būti išvardinta, kiek kokio nominalo monetų gaus kiekvienas draugas, pradedant nuo 2 eurų nominalo, po to 1 euro nominalo, po to 50, 20 ir 10 euro centų ir kokią pinigų sumą paaukos beglobiams gyvūnams.

|  |  |
| --- | --- |
| santaupos.txt | santauposrez.txt |
| 2  45 78 49  32 2 87 | 2 0 0 2 0 6  2 0 1 1 0 1 |

1. **Reikalavimai**

* Programoje turi būti **void funkcija Monetos()**, skaičiuojanti norimo nominalo monetų kiekį. Į funkciją reikia kreiptis 5 kartus: su 2 ir 1 euro, bei 50, 20 ir 10 euro centų monetomis.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Monetos(int vien, int du, int penk, int &dueur, int &eur, int &penkdc, int &dvidc, int &desc, int &auk);

int main(){

int vien, du, penk, dueur, eur, penkdc, dvidc, desc, auk, n;

ifstream df ("santaupos.txt");

ofstream rf ("santauposrez.txt");

df >> n;

for (int i = 0; i < n; i++){

df >> vien >> du >> penk;

Monetos(vien, du, penk, dueur, eur, penkdc, dvidc, desc, auk);

rf << dueur << " " << eur << " " << penkdc << " " << dvidc << " " << desc << " " << auk << endl;

}

df.close();

rf.close();

return 0;

}

void Monetos(int vien, int du, int penk, int &dueur, int &eur, int &penkdc, int &dvidc, int &desc, int &auk)

{

int vis = 0;

vis = vien + du \* 2 + penk \* 5;

dueur = vis / 200;

vis = vis % 200;

eur = vis / 100;

vis = vis % 100;

penkdc = vis / 50;

vis = vis % 50;

dvidc = vis / 20;

vis = vis % 20;

desc = vis / 10;

auk = vis % 10;

}

#### Kambarių sienų dažymas

1. **Užduotis**

Parašykite programą, kuri atliktų veiksmus su duotais 2 skaičiais: užrašytų šiuos skaičius iš kitos pusės, patikrintų ar naujieji skaičiai yra pirminiai. Jeigu gautieji skaičiai yra pirminiai, užrašytų didesniojo skaičiaus skaitmenų sumą, o jeigu ne – mažesniojo skaičiaus skaitmenų sumą.

**Pradiniai duomenys:** tekstiniame faile **„Skaitmenys.txt“** pateikti pradiniai duomenys. Pateikiami du mokytojo Galvočiaus duoti sveikieji skaičiai atskirti tarpo simboliu.

**Rezultatai:** tekstiniame faile **„Rezultatai.txt“** pateikiami rezultatai. Pirmoje eilutėje užrašyti 2 skaičiai iš kitos pusės (atskirti tarpu), kitoje eilutėje – skaitmenų suma pagal atitinkančią sąlygą (jeigu abu iš kitos pusės parašyti skaičiai yra pirminiai – išvedama didesniojo skaičiaus skaitmenų sumą, kitu atveju;– mažesniojo skaičiaus skaitmenų suma).

|  |  |
| --- | --- |
| Skaitmenys.txt | Rezultatai.txt |
| 799 938 | 997 839  25 |

1. **Reikalavimai**

* Skaičiai **int** tipo kintamieji;
* Sukurkite **void** tipo funkciją, kuri parašytų abu skaičius iš kitos pusės ir juos grąžintų per parametrus-nuorodas;
* Sukurkite **bool** tipo funkciją, kuri patikrintų ar skaičius pirminis;
* Sukurkite **int** tipo funkciją kuri apskaičiuotų skaičiaus skaitmenų sumą;
* **Nenaudokite** masyvų!

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Apsukimas (int &sk, int &sk1);

bool Pirminis (int sk);

int Suma(int sk);

int Kiekis(int sk);

int main(){

int sk, sk1, s;

ifstream df ("Skaitmenys.txt");

ofstream rf ("Rezultatai.txt");

df >> sk >> sk1;

Apsukimas(sk, sk1);

if (Pirminis(sk) && Pirminis(sk1))

{

if (sk > sk1)s = Suma(sk);

else s = Suma(sk1);

}

else{

if (sk < sk1)s = Suma(sk);

else s = Suma(sk1);

}

rf << sk << " " << sk1 << endl << s;

rf.close();

df.close();

return 0;

}

void Apsukimas (int &sk, int &sk1)

{

int k, k1, s = 0;

k = Kiekis(sk) - 1;

k1 = Kiekis(sk1) - 1;

for (k; k >= 0; k--)

{

s = s + (sk % 10) \* pow(10, k);

sk = sk / 10;

}

sk = s + 1;

s = 0;

for (k1; k1 >= 0; k1--)

{

s = s + (sk1 % 10) \* pow(10, k1);

sk1 = sk1 / 10;

}

sk1 = s + 1;

//Suluzta ir truksta +1, visur veikia skirtingai.

}

bool Pirminis (int sk)

{

int k = 2;

while (sk % k != 0)

{

if (k > sk / 2 + 1)return true;

k++;

}

return false;

}

int Suma(int sk)

{

int s = 0;

while (sk > 0){

s = s + sk % 10;

sk = sk / 10;

}

return s;

}

int Kiekis(int sk)

{

int k = 0;

while (sk > 0){

k++;

sk = sk / 10;

}

return k;

}

#### Senovės Anglijos pinigai

1. **Užduotis**

Senovės Anglijoje buvo trys piniginiai matavimo vienetai: svarai, šilingai ir pensai. Vienas svaras – 20 šilingų, šilingas – 12 pensų. Parašykite programą, skaičiuojančią dviejų skirtingų pinigų kiekių, išreikštų svarais, šilingais ir pensais, sumą ir skirtumą, išreikštą svarais, šilingais ir pensais.

Pradinių duomenų failo **pinigai.txt** pirmoje eilutėje įrašyta, kelių pinigų rinkinių sumą ir skirtumą reikia apskaičiuoti. Tolesnėse **n** eilučių įrašyta po 6 sveikuosius skaičius: pirmieji trys skaičiai reiškia pirmojo pinigų kiekio svarus **sv1**, šilingus **s1** ir pensus **p1**, paskutinieji trys skaičiai – antrojo pinigų kiekio svarus **sv2**, šilingus **s2** ir pensus **p2**.

Rezultatų faile pinigairez.txt turi būti n eilučių su 6 sveikaisiais skaičiais kiekvienoje eilutėje: pirmieji trys skaičiai reiškia sumos svarus **ssv**, sumos šilingus **ss**, sumos pensus **sp**; paskutinieji trys skaičiai – skirtumo svarus **sksv**, skirtumo šilingus **sks**, skirtumo pensus **skp**.

|  |  |
| --- | --- |
| pinigai.txt | pinigairez.txt |
| 2  2 3 2 1 4 5  3 5 4 2 6 8 | 3 7 7 0 18 9  5 12 0 0 18 8 |

1. **Reikalavimai**

* Turi būti funkcija **void Suma()**, skaičiuojanti dviejų pinigų kiekių sumą.
* Turi būti funkcija **void Skirtumas()**, skaičiuojanti dviejų pinigų kiekių skirtumą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Suma(int sv, int st, int p, int sv1, int st1, int p1, int &ssv, int &sst, int &sp);

void Skirtumas(int sv, int st, int p, int sv1, int st1, int p1, int &ssv, int &sst, int &sp);

int main(){

int sv, st, p, sv1, st1, p1, ssv, sst, sp, sksv, skst, skp, n;

ifstream df ("pinigai.txt");

ofstream rf ("pinigairez.txt");

df >> n;

for (int i = 0; i < n; i++){

df >> sv >> st >> p >> sv1 >> st1 >> p1;

Suma(sv, st, p, sv1, st1, p1, ssv, sst, sp);

Skirtumas(sv, st, p, sv1, st1, p1, sksv, skst, skp);

rf << ssv << " " << sst << " " << sp << " " << sksv << " " << skst << " " << skp << endl;

df.close();

rf.close();

}

return 0;

}

void Suma(int sv, int st, int p, int sv1, int st1, int p1, int &ssv, int &sst, int &sp)

{

sp = (p + p1) % 12;

sst = (st + st1 + (p + p1) / 12) % 20;

ssv = sv + sv1 + (st + st1 + (p + p1) / 12) / 20;

}

void Skirtumas(int sv, int st, int p, int sv1, int st1, int p1, int &ssv, int &sst, int &sp)

{

int s, s1, sk = 0;

s = sv \* 12 \* 20 + st \* 12 + p;

s1 = sv1 \* 20 \* 12 + st1 \* 12 + p1;

sk = s - s1;

sp = sk % 12;

ssv = sk / 12 / 20 ;

sst = sk / 12 % 20;

//Uzduotyje blogai nurodyti atsakymai

} if (ArTinkama(met, men, dien))rf << met << " m. " << Menuo(men) << " " << dien << " d." << endl;

else rf << "KLAIDA" << endl;

}

rf.close();

df.close();

return 0;

}

int Data(int data, int kas){

if (kas == 1){

data = data / 10000;

}

else if (kas == 2){

data = data / 100 % 100;

}

else data = data % 100;

return data;

}

bool ArTinkama(int metai, int menuo, int diena)

{

if (metai < 0 || menuo <= 0 || diena <= 0 || menuo > 12 || diena > 31) return false;

else if (menuo == 2 && metai % 4 != 0 && diena > 28) return false;

else if (metai % 4 == 0 && menuo == 2 && diena > 29) return false;

else if (menuo == 4 && diena > 30 || menuo == 6 && diena > 30 || menuo == 9 && diena > 30 || menuo == 11 && diena > 30)return false;

else return true;

}

string Menuo(int menuo)

{

if (menuo == 1)return "sausio";

else if (menuo == 2)return "vasario";

else if (menuo == 3)return "kovo";

else if (menuo == 4)return "balandzio";

else if (menuo == 5)return "geguzes";

else if (menuo == 6)return "birzelio";

else if (menuo == 7)return "liepos";

else if (menuo == 8)return "rugpjucio";

else if (menuo == 9)return "rugsejo";

else if (menuo == 10)return "spalio";

else if (menuo == 11)return "lapkricio";

else return "gruodzio";

}

#### Civilinio karo simuliavimas

1. **Užduotis**

Kovose gali dalyvauti 5 tipo kareiviai, kuris kiekvienas turi savo puolimo bei gynybos taškus.

Programos užduotis - pasakyti ar besiginanti pusė (gynėjas) laimėjo karą ir kiek gynėjas prarado kiekvieno tipo kareivių.

**Karių lentelė:**

|  |  |  |
| --- | --- | --- |
| **Kario pavadinimas** | **Puolimo taškai** | **Gynybos taškai** |
| Pretorionas | 20 | 64 |
| Legionierius | 54 | 54 |
| Lankininkas | 44 | 64 |
| Barbaras | 82 | 12 |
| Centurionas | 137 | 62 |

**Skaičiavimų paaiškinimas**:

Laimėtojas nustatomas lyginant puolimo stiprumą (PS) ir gynybos tvirtumą (GT).

Jeigu PS yra didesnis arba lygus GT, laikoma, kad puolėjas laimėjo kovą. Jeigu GT yra didesnis už PS – laimėjo gynėjas.

**PS yra visų kareivių puolimų taškų suma:**

|  |  |
| --- | --- |
| Duomenų failas | Skaičiavimas |
| 10 | 10 (kareivių skaičius iš duomenų failo) \* 20 ( kareivio puolimo taškai) = 200 |
| 0 | 0 \* 54 = 0 |
| 0 | 0 \* 44 = 0 |
| 500 | 500 \* 82 = 41000 |
| 100 | 100 \* 137 = 13700 |
| **BPP** | **54900** |

**GT yra visų kareivių gynimo taškų suma:**

|  |  |
| --- | --- |
| Duomenų failas | Skaičiavimas |
| 10 | 10 (kareivių skaičius iš duomenų failo) \* 64 ( kareivio gynimo taškai) = 640 |
| 5 | 5 \* 54 = 270 |
| 30 | 30 \* 64 = 1920 |
| 3000 | 3000 \* 12 = 36000 |
| 0 | 0 \* 62 = 0 |
| **BPP** | **38830** |

Siekiant nustatyti gynėjo prarastų karių skaičių, reikia apskaičiuoti santykį efektyvumą (SE).

**Santykinis efektyvumas skaičiuojamas taip:**

Jeigu PS yra lygus arba didesnis už GT:

SE = GT / PS.

Jeigu GT didesnis už PS:

SE = PS / GT.

SE yra suapvalinamas iki 4 skaičių po kablelio.

**Gynėjo aukų skaičiavimas:**

Jeigu gynėjas pralaimėjo, tai aukos (per kario tipą) skaičiuojamos taip:

KarioAukos = KariųSkaičius \* (1 - SE)

Jeigu gynėjas laimėjo, tai aukos (per kario tipą) skaičiuojamos taip:

KarioAukos = KariųSkaičius \* SE

**Įvesties paaiškinimas:**

Duomenų failas turi 10 eilučių, kuriose nuo 1 iki 5 eilutės yra parašyti puolėjo karių skaičius, o nuo 6 iki 10 gynėjo karių skaičius.

|  |  |
| --- | --- |
| Duomenų failas (data\_input.txt) | Paaiškinimas |
| 1 (pirma puolėjo eilutė) | Puolėjo pretorionų skaičius |
| 2 | Puolėjo legionierių skaičius |
| 3 | Puolėjo lankininkų skaičius |
| 4 | Puolėjo barbarų skaičius |
| 5 (paskutinė puolėjo eilutė) | Puolėjo centurionų skaičius |
| 6 (pirma gynėjo eilutė) | Gynėjo pretorionų skaičius |
| 7 | Gynėjo legionierių skaičius |
| 8 | Gynėjo lankininkų skaičius |
| 9 | Gynėjo barbarų skaičius |
| 10 (paskutinė gynėjo eilutė) | Gynėjo centurionų skaičius |

**Išvesties paaiškinimas:**

|  |  |
| --- | --- |
| Išvesties failas (data\_output.txt) | Paaiškinimas |
| true | Gynėjas laimėjo |
| 41 | Gynėjas prarado 41 pretorioną |
| 0 | Gynėjas neprarado legionierių |
| 10 | Gynėjas prarado 10 lankininkų |
| 0 | Gynėjas neprarado barbarų |
| 0 | Gynėjas neprarado centurionų |

|  |  |
| --- | --- |
| data\_input.txt | data\_output.txt |
| 0  0  0  0  5000  500  100  0  0  0 | false  437  874  0  0  0 |

1. **Reikalavimai**

* Nenaudoti masyvų.
* Apvalinti pajėgumų santykį iki 4 skaičiaus po kableliu.
* Skaičiuojant gynėjo aukų skaičių, reikšmę apvalinti naudojant **floor** funkciją.
* Naudokite įvesties failo vardą **data\_input.txt**, o išvesties **data\_output.txt.**
* Skaičių tipų konversijai naudoti **static\_cast <>()**.
* Naudoti **void** funkciją santykinio efektyvumui (**SE**) skaičiavimui, kuri rezultatą grąžintų per parametrus.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Santykis(double pt, double gt, double &SE);

int main(){

double P, La, Le, B, C, P1, La1, Le1, B1, C1, pt, gt;

double SE;

ifstream df ("data\_input.txt");

ofstream rf ("data\_output.txt");

df >> P >> Le >> La >> B >> C >> P1 >> Le1 >> La1 >> B1 >> C1;

pt = P \* 20 + Le \* 54 + La \* 44 + B \* 82 + C \* 137;

gt = P1 \* 64 + Le1 \* 54 + La1 \* 64 + B1 \* 12 + C1 \* 62;

Santykis(pt, gt, SE);

if (pt >= gt){

rf << "false" << endl;

SE = 1 - SE;

}

else{

rf << "true" << endl;

}

P1 = P1 \* SE;

Le1 = Le1 \* SE;

La1 = La1 \* SE;

B1 = B1 \* SE;

C1 = C1 \* SE;

rf << floor(P1) << endl << floor(Le1) << endl << floor(La1) << endl << floor(B1) << endl << floor(C1) << endl;

df.close();

rf.close();

return 0;

}

void Santykis(double pt, double gt, double &SE)

{

if (pt >= gt)SE = gt / pt;

else SE = pt / gt;

SE = round(SE \* 10000) / 10000;

}

#### Miltai

1. **Užduotis**

Žinoma, kad perkant daugiau prekių, jų vienetas kainuoja pigiau. Kepyklos užsakinėja miltus. Miltų pakuotėje yra **m** kilogramų miltų, stambesnėje pakuotėje – **s** pakuočių. Kilogramas miltų kainuoja **mm** eurų, pakuotė – **pp** eurų, stambesnė pakuotė – **ss** eurų. Parašykite programą, kuri apskaičiuotų, kiek kepyklai kainuos įsigyti **n** kilogramų miltų. Pirmiausia įsigijama kiek išeina stambesnių pakuočių, po to – pakuočių ir po to atskirai supakuotų miltų po kilogramą skaičiai.

Pradinių duomenų failo miltai.txt pirmoje eilutėje įrašyti 3 sveikieji skaičiai: kepyklų skaičius **k**, kilogramų kiekis pakuotėje **m**, pakuočių skaičius stambesnėje pakuotėje **s**, toliau įrašyti trys realieji skaičiai: miltų kilogramo kaina **mm**, pakuotės kaina **pp** ir stambesnės pakuotės kaina **ss**. Tolesnėse **k** eilučių įrašyta po vieną sveikąjį skaičių – kiekvienos kepyklos norimų įsigyti miltų kiekis kilogramais **n**.

Rezultatų faile miltairez.txt turi būti **k** eilučių su 3 sveikaisiais ir 3 realiaisiais skaičiais kiekvienoje eilutėje: pirmieji trys skaičiai reiškia, kiek stambesnių pakuočių **sn**, pakuočių **pn** ir atskirai po kilogramą supakuotų miltų **mn** įsigis kepykla; antrasis realiųjų skaičių rinkinys reiškia įsigytų stambesnių miltų pakuočių kainą **sk**, pakuočių kainą **pk** ir atskirai supakuotų po kilogramą miltų kainą **mk**.

|  |  |
| --- | --- |
| miltai.txt | miltairez.txt |
| 3 50 2 4.00 190.00 350.00  30  72  125 | 0 0 30 0.00 0.00 120.00  0 1 22 0.00 190.00 88.00  1 0 25 350.00 0.00 100.00 |

1. **Reikalavimai**

* Turi būti funkcija **void Kiekiai()**, skaičiuojanti, kelias stambesnes pakuotes, pakuotes ir po kilogramą supakuotų miltų kiekius įsigijo kepykla.
* Turi būti funkcija **void Kaina()**, skaičiuojanti įsigytų miltų stambesnių pakuočių, pakuočių ir miltų, supakuotų po kilogramą, kainas.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Kiekiai(double kiek, int kgp, int psp, int &pakkiek, int &stambkiek, double &atskr);

void Kaina(double mkgk, double pk, double spk, int pakkiek, int stambkiek, double atskr, double &stambkain, double &pakkain, double &kgkain);

int main(){

int n, kgp, psp, pakkiek, stambkiek;

double mkgk, pk, spk, kg, atskr, stambkain, pakkain, kgkain;

ifstream df ("miltai.txt");

ofstream rf ("miltairez.txt");

df >> n >> kgp >> psp >> mkgk >> pk >> spk;

for (int i = 0; i < n; i++){

df >> kg;

Kiekiai(kg, kgp, psp, pakkiek, stambkiek, atskr);

Kaina(mkgk, pk, spk, pakkiek, stambkiek, atskr, stambkain, pakkain, kgkain);

rf << stambkiek << " " << pakkiek << " " << atskr << " " << fixed << setprecision(2) << stambkain << " " << pakkain << " " << kgkain << endl;

}

df.close();

rf.close();

return 0;

}

void Kiekiai(double kiek, int kgp, int psp, int &pakkiek, int &stambkiek, double &atskr)

{

atskr = int(kiek) % kgp;

stambkiek = int(kiek - atskr) / kgp / psp;

pakkiek = int(kiek - atskr) / kgp % psp;

}

void Kaina(double mkgk, double pk, double spk, int pakkiek, int stambkiek, double atskr, double &stambkain, double &pakkain, double &kgkain)

{

stambkain = spk \* stambkiek;

pakkain = pk \* pakkiek;

kgkain = mkgk \* atskr;

}

#### Namų darbai

1. **Užduotis**

Visiem padariusiems namų darnus mokytoja prideda vieną balą prie kontrolinio pažymio, o nepadariusiems – vienu balu sumažina.

Reikia sukurti programą, kuri pakeistų mokinių pažymius, atsižvelgiant į tai, ar jie padarė namų darbus.

Failo **pazymiai.txt** pirmoje eilutėje pateiktas mokinių skaičius, kitose eilutėse pateikti mokinių pažymiai ir ar jie atliko namų darbus (žodis **atliko**, jie namų darbus padarė, žodis **neatliko** kitu atveju). Rezultatų faile **rez.txt** kiekvienoje eilutėje turi būti pateikti pakoreguoti pažymiai. Atkreipk dėmesį, kad **pažymys gali būti tik intervale [1:10]**, todėl jei mokinio pažymys lygus 1 ir jis nepadarė namų darbų, tai pažymys daugiau nemažinamas.

|  |  |
| --- | --- |
| pazymiai.txt | rez.txt |
| 3  10 atliko  3 neatliko  1 atliko | 10  2  2 |

1. **Reikalavimai**

* Funkcija **void**, kuri pakeičia mokinio pažymį, atsižvelgiant į jo statusą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Pazimys (int &paz, string nd);

int main(){

int n, paz;

string nd;

ifstream df ("pazymiai.txt");

ofstream rf ("rez.txt");

df >> n;

for (int i = 0; i < n; i++){

df >> paz >> nd;

Pazimys(paz, nd);

rf << paz << endl;

}

df.close();

rf.close();

return 0;

}

void Pazimys (int &paz, string nd)

{

if (paz > 1 && nd == "neatliko")paz--;

else if (paz < 9 && nd == "atliko")paz++;

}

#### Informacijos matavimo vienetai

1. **Užduotis**

Parašykite programą, kuri gamintojo nurodytą atmintuko talpą GB paverstų kompiuterio rodoma reikšme gibibaitais (rezultatas pateikiamas vieno skaitmens po kablelio tikslumu) ir tebibaitais (rezultatas pateikiamas dviejų skaitmenų po kablelio tikslumu).

Pradinių duomenų failo **informacija\_data.txt** pirmoje eilutėje įrašytas atmintukų skaičius **n**. Tolesnėse **n** eilučių įrašytos gamintojo nurodytos atmintukų talpos GB (sveikieji skaičiai).

Rezultatų faile **informacija\_res.txt** turi būti **n** eilučių, kurių kiekvienoje yra po 2 realiuosius skaičius, vienas nuo kito atskirtus tarpais: atmintuko talpa gibibaitais ir tebibaitais.

|  |  |
| --- | --- |
| informacija\_data.txt | informacija\_res.txt |
| 4  64  32  16  5 | 59.6 0.06  29.8 0.03  14.9 0.01  4.7 0 |

1. **Reikalavimai**

* Parašykite funkciją **void Atmintukai (int atmintukoTalpa, double & gibibaitai, double & tebibaitai)**, kuri per parametrus grąžina apskaičiuotą vieno atmintuko talpą gibibaitais ir tebibaitais.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Atmintukai(int atmintukoTalpa, double &gibibaitai, double &tebibaitai);

int main(){

int n, talp;

double gig, ter;

ifstream df ("informacija\_data.txt");

ofstream rf ("informacija\_res.txt");

df >> n;

for(int i = 0; i < n; i++){

df >> talp;

Atmintukai(talp, gig, ter);

rf << fixed << setprecision(1) << gig << " " << fixed << setprecision(2) << ter << endl;

}

df.close();

rf.close();

return 0;

}

void Atmintukai(int atmintukoTalpa, double &gibibaitai, double &tebibaitai)

{

gibibaitai = atmintukoTalpa \* (pow(10, 9) / pow(2, 30));

tebibaitai = atmintukoTalpa / (pow(2, 40) / pow(10, 9));

}

#### Geriausias krepšininkas

1. **Užduotis**

Du draugai, Šarūnas ir Dainius žaidžia skirtingose krepšinio mėgėjų komandose skirtinguose miestuose. Ir vieną dieną susiginčijo, kuris iš jų yra geresnis krepšinio žaidėjas. Todėl surinko geriausių savo varžybų rodiklius ir nusprendė juos sulyginti.

Tačiau jie nežinojo kaip apskaičiuoti svarbiausią rodiklį – žaidėjo efektyvumą. Padėkite draugams išsiaiškinti, kuris iš jų dviejų yra geresnis.

Duota abiejų krepšininkų duomenys: vardas ir pavardė **Krepsininkas** (simbolių eilutė iki 30 simbolių). Kitoje eilutėje duota krepšininko per vienerias rungtynes mestų dvitaškių metimų skaičių **PTS2**, įmestų tritaškių metimų skaičių **PTS3**, netikslių metimų skaičius **MISSED**, atkovotų kamuolių skaičius **REB**, perimtų kamuolių skaičius **STL**, rezultatyvių perdavimų skaičius, blokuotų metimų skaičius **BLK**, padarytų klaidų skaičius **T**, ir žaidėjo žaidimo laikas **TIME** (minimum 7 min).

Reikia rasti abiejų krepšininkų efektyvumą ir iš viso mestų metimų skaičių. Naudosime paprastesnę krepšininkų efektyvumo skaičiavimo formulę, kuri dažnai yra naudojama Europoje. Metimų skaičius apskaičiuojamas sudedant įmestų dvitaškių, tritaškių ir netaiklių metimų skaičių.

**INDEX**= (POINTS– MISSED SHOTS+ REBOUNDS+ ASSISTS+ STEALS+BLOCKS-T/OVER) /TIME PLAYED

*Kur: INDEX – efektyvumas; POINTS – surinkta taškų; MISSED SHOTS – netikslūs metimai; REBOUNDS –atkovota kamuolių; STEALS –perimta kamuolių;ASSISTST - rezultatyvūs perdavimai; BLOCKS – blokuota metimų T/OVER – klaidos (prarasta kamuolių), TIME PLAYED – žaidėjo žaidimo laikas (minimum 7 min)*

Rezultatų faile turite išvesti abiejų krepšininkų vardus ir pavarde bei apskaičiuotą kiekvieno krepšininko efektyvumą ir iš viso mestų metimų skaičių. Taip pat, trečioje rezultatų failo eilutėje turite išvesti eilutę, kurioje būtų nurodytas geresnis žaidėjas (žaidėjas, kuris yra efektyvesnis iš dviejų žaidėjų).

|  |  |
| --- | --- |
| Krepsininkai\_duom.txt | Krepsininkai\_rez.txt |
| Dainius Adomaitis  10 1 5 5 1 4 4 15 24  Šarūnas Jasikevičius  12 5 2 8 8 8 1 3 24 | Dainius Adomaitis 0,7 16  Šarūnas Jasikevičius 2,13 19  Geresnis yra Šarūnas Jasikevičius |

1. **Reikalavimai**

* Būtina funkcija void, grąžinanti per parametrus: vieno žaidėjo efektyvumą ir mestų metimų skaičių.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Krepsinis(int T[], double &indeks, int &met);

int main(){

string V[100];

int T[100], met = 0, sk;

char eil[30];

double indeks = 0, maxi = -1;

ifstream df ("Krepsininkai\_duom.txt");

ofstream rf ("Krepsininkai\_rez.txt");

for (int i = 0; i < 2; i++){

df.ignore();

df.get(eil, 30);

V[i] = eil;

for (int i1 = 0; i1 < 9; i1++){

df >> T[i1];

}

Krepsinis(T, indeks, met);

rf << V[i] << " " << fixed << setprecision(2) << indeks << " " << met << endl;

if (maxi < indeks){

maxi = indeks;

sk = i;

}

}

rf << "Geresnis yra " << V[sk];

rf.close();

df.close();

return 0;

}

void Krepsinis(int T[], double &indeks, int &met)

{

met = T[0] + T[1] + T[2];

indeks = double((T[0] \* 2 + T[1] \* 3 - T[2] + T[3] + T[4] + T[5] + T[6] - T[7])) / T[8];

}

#### Varikliukų pakavimas

1. **Užduotis**

Kai parduotuvė užsisako **k** robotų varikliukų, prieš išvežant juos reikia sudėlioti į dežes. Šios dėžės yra trijų tipų pagal dydį: pirmo tipo dėžėje telpa **k1** varikliukų, antro tipo dėžėje telpa **k2** varikliukų, trečio tipo dėžėje telpa **k3** varikliukų (k1 > k2 > k3).

Pirmiausia užpildoma kiek įmanoma pirmo tipo dėžių, paskui antro tipo dėžių ir tik tada pildomos trečio tipo dėžės. Nepilnai užpildytos gali būti tik trečio tipo dėžės, kuriose telpa **k3** varikliukų (t.y., jeigu k < k2, pildome tik trečio tipo dėžes, analogiškai elgiamės, jei telpa į antrą, bet ne į pirmą).

Duomenų failo pirmoje eilutėje parašyti keturi skaičiai: n - kiek parduotuvių užsisake varikliukų, **k1**, **k2**, **k3**. Sekančiose n eilučių parašyta po vieną skaičių - **k**, t.y. kiek atitinkama parduotuvė užsisakė varikliukų.

Rezultatų faile turi būti **n** eilučių, kiekvienoje jų parašyti trys skaičiai - kiek kiekvieno tipo dėžių buvo panaudota pakavimui.

|  |  |
| --- | --- |
| varikliukai\_data.txt | varikliukai\_res.txt |
| 3 20 10 5  64  32  16 | 3 0 1  1 1 1  0 1 2 |

1. **Reikalavimai**

* Aprašyti funkciją **void Kiekis(int &kiek, int talpa, int &deziuSkaicius)**, kuri per parametrus grąžina apskaičiuotą vieno tipo dėžių skaičių.
* Rašyti tvarkingą kodą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

void Kiekis(int &kiek, int talpa, int &deziuSkaicius);

using namespace std;

int main(){

int n, k, k1, k2, ds, var, rod = 0;

ifstream df ("varikliukai\_data.txt");

ofstream rf ("varikliukai\_res.txt");

df >> n >> k >> k1 >> k2;

for (int i = 0; i < n; i++){

df >> var;

Kiekis(var, k, ds);

rf << ds << " ";

Kiekis(var, k1, ds);

rf << ds << " ";

Kiekis(var, k2, ds);

if (var > 0)ds = ds + 1;

rf << ds << " " << endl;

}

df.close();

rf.close();

return 0;

}

void Kiekis(int &kiek, int talpa, int &deziuSkaicius)

{

deziuSkaicius = kiek / talpa;

kiek = kiek % talpa;

}

#### Kambarių sienų dažymas

1. **Užduotis**

Parašykite programą, kuri apskaičiuoja, kiek litrų dažų prireiks padengti kiekvieno kambario sienas 1 kartą ir aptinka didžiausią kiekį dažų, reikalingą kambario sienų padengimui tarp visų dažomų kambarių. (rezultatas pateikiamas dviejų skaitmenų po kablelio tikslumu)

Planuojama dažyti **n** kambarių sienas neskaitant lubų. Nuperkami dažai su **k** m²/l išeiga. Žinoma, kiek skirtingų sienų **s** yra kiekviename kambaryje, taip pat ir jų ilgiai bei aukščiai metrais, kambario durų skaičius **d** bei langų skaičius **r**. Durų plotas visuose kambariuose yra vienodas ir yra lygus 2 m², langų plotas taip pat ir yra lygus 1.5 m². Durys ir langai nedažomi.

Pirmoje pradinių duomenų failo dazai\_data.txt eilutėje įrašytas sveikasis skaičius kambarių **n** ir dažų išeiga **k**. Toliau **n** kartų surašyti kambarių parametrai: eilutėse yra skirtingų sienų skaičius **s**, kambario durų skaičius **d** bei langų skaičius **r**. Po kiekvienos kambario parametrų eilutės yra **s** kartų surašyti skirtingų sienų ilgiai ir aukščiai. Jei kambarys kvadrato formos, tada skirtingų sienų skaičius **s** = 1, jei stačiakampio, tai skirtingų sienų skaičius **s** = 2. Jei kambarys kito daugiakampio formos, tai skirtingų sienų skaičius **s** > 2.

Rezultatų faile dazai\_res.txt turi būti išvesti kiekvieno kambario sienoms padengti reikalingi dažų kiekiai litrais, paskutinėje eilutėje išvedamas didžiausias iš visų kambarių sienoms padengti reikalingas dažų kiekis.

|  |  |
| --- | --- |
| dazai\_data.txt | dazai\_res.txt |
| 3 6  2 1 1  4 3  5 3  5 2 3  2.5 3  1.5 3  2.7 3  4 3  3.5 3  1 1 2  4 3 | 7.42  8.42  6.52  Didžiausias kiekis: 8.42 |

1. **Reikalavimai**

* Parašykite funkciją **void naujasPlotas(double ilgis, double aukstis, int sienuSkaicius, double & plotas)**, kuri per parametrus grąžina apskaičiuotą naują plotą, pridedant einamos sienos apskačiuotą plotą, atsižvelgiant į sienų skaičių.
* Parašykite funkciją **void arDidziausias(double dazuKiekis, double & didziausiasKiekis)**, kuri per parametrus grąžina naują didžiausio reikiamo dažų kiekio reikšmę, jei paduodamas dažų kiekis didesnis už paduotą didžiausią dažų kiekį.
* Nenaudokite masyvų.
* Skaitymui galima naudoti ciklą cikle, n kartų einant per kiekvieną kambarį ir s kartų per kiekvieno kambario sienas.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void NaujasPlotas(double ilgis, double aukstis, int sienuSkaicius, double &plotas);

void ArDidziausias(double dazuKiekis, double &didziausiasKiekis);

int main(){

int n, s, lang, dur;

double dazkof, ilg, aukst, S = 0, kiek = 0, did = 0;

ifstream df ("dazai\_data.txt");

ofstream rf ("dazai\_res.txt");

df >> n >> dazkof;

for(int i = 0; i < n; i++){

df >> s >> dur >> lang;

for(int i1 = 0; i1 < s; i1++){

df >> ilg >> aukst;

NaujasPlotas(ilg, aukst, s, S);

}

S = S - dur \* 2 - lang \* 1.5;

kiek = S / dazkof;

rf << fixed << setprecision(2) << kiek << endl;

ArDidziausias(kiek, did);

S = 0;

}

rf << "Didziausias kiekis: " << did;

rf.close();

df.close();

return 0;

}

void NaujasPlotas(double ilgis, double aukstis, int sienuSkaicius, double &plotas)

{

if (sienuSkaicius == 1){

plotas = plotas + 4 \* ilgis \* aukstis;

}

else if (sienuSkaicius == 2){

plotas = plotas + 2 \* ilgis \* aukstis;

}

else{

plotas = plotas + ilgis \* aukstis;

}

}

void ArDidziausias(double dazuKiekis, double &didziausiasKiekis)

{

if (dazuKiekis > didziausiasKiekis)didziausiasKiekis = dazuKiekis;

}

### Masyvas. Masyvo elementų sumos, kiekio, vidurkio skaičiavimas.

#### Pirkėjų srautai

1. **Užduotis**

Darbo laikas skirstomas taip: 8.00-8.30 – 1 pusvalandis; 8.30-9.00 – 2 pusvalandis, ..., 19.30-20.00 – 24 pusvalandis.

Pradinių duomenų failo **pirkejai\_data.txt** pirmoje eilutėje įrašytas per dieną aptarnautų pirkėjų skaičius **n**. Tolesnėse **n** eilučių įrašyta po 2 sveikuosius skaičius – kasos čekyje nurodytas pirkėjo aptarnavimo laikas – valanda ir minutė.

Rezultatų faile **pirkejai\_res.txt** turi būti išvardinti visi 24 pusvalandžiai, nurodant pusvalandžio pradžios valandą ir minutę, pabaigos valandą ir minutę, bei tą pusvalandį aptarnautų pirkėjų skaičių. Vienam pusvalandžiui skiriama viena eilutė.

Jei pirkėjo aptarnavimo laikas sutampa su vieno pusvalandžio pabaiga, o kito pradžia, tokį pirkėją priskirkite ankstesniam pusvalandžiui (tam pusvalandžiui, su kurio pabaiga sutampa pirkėjo aptarnavimo laikas).

|  |  |
| --- | --- |
| pirkejai\_data.txt | pirkejai\_data.txt |
| 5  8 45  17 25  18 45  19 13  19 35 | 8 0 8 30 0  8 30 9 0 1  9 0 9 30 0  9 30 10 0 0  10 0 10 30 0  10 30 11 0 0  11 0 11 30 0  11 30 12 0 0  12 0 12 30 0  12 30 13 0 0  13 0 13 30 0  13 30 14 0 0  14 0 14 30 0  14 30 15 0 0  15 0 15 30 0  15 30 16 0 0  16 0 16 30 0  16 30 17 0 0  17 0 17 30 1  17 30 18 0 0  18 0 18 30 0  18 30 19 0 1  19 0 19 30 1  19 30 20 0 1 |

1. **Reikalavimai**

* Main funkcijoje privalo būti tik lokaliųjų kintamųjų apsirašymas bei skaitymo ir rašymo funkcijos.
* Sukurti funkciją **Skaitymas**.
* Sukurti funkciją **Rasymas**.
* Nenaudoti globalių kintamųjų.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito (int A[], int &n);

void Raso(int A[], int n);

void Laikas(int A[], int n, int B[]);

void Klientura(int A[], int n, int val, int minut, int minut1, int &klient);

int main(){

int A[1000], n, B[1000];

Skaito(A, n);

Laikas(A, n, B);

Raso(B, 120);

return 0;

}

void Skaito (int A[], int &n)

{

ifstream df ("pirkejai\_data.txt");

df >> n;

for (int i = 0; i < n \* 2; i++)df >> A[i];

df.close();

}

void Raso(int A[], int n)

{

ofstream rf ("pirkejai\_res.txt");

for (int i = 0; i < n; i = i + 5)rf << A[i] << " " << A[i + 1] << " " << A[i + 2] << " " << A[i + 3] << " " << A[i + 4] << endl;

rf.close();

}

void Laikas(int A[], int n, int B[])

{

int ind = 0, k = 0, i1 = 0;

for (int i = 8; i < 25; i = i + 1){

for(int i1 = 0; i1 <= 30; i1 = i1 + 30){

B[ind] = i;

ind++;

B[ind] = i1;

ind++;

}

Klientura(A, n, i, 0, 30, k);

B[ind] = k;

ind++;

for(int i1 = 30; i1 >= 0; i1 = i1 - 30){

B[ind] = i;

ind++;

B[ind] = i1;

ind++;

}

k = 0;

B[ind - 2] = B[ind - 2] + 1;

Klientura(A, n, i, 30, 60, k);

B[ind] = k;

k = 0;

ind++;

}

}

void Klientura(int A[], int n, int val, int minut, int minut1, int &klient)

{

for(int i = 0; i < n \* 2; i = i + 2){

if (A[i] == val && minut < A[i + 1] && A[i + 1] < minut1)klient++;

else if (minut1 == 60 && A[i] == val + 1 && minut1 % 60 == A[i + 1])klient++;

else if (minut1 == 30 && A[i] == val && minut1== A[i + 1])klient++;

}

}

#### Krepšinio čempionatas

1. **Užduotis**

Programuotojas Martynas labai domisi krepšiniu ir šiemet vykstantis pasaulio krepšinio čempionatas jį labai džiugina, tačiau Martynui nusibodo stebėti rungtynes televizoriaus ekrane, taigi jis nusprendė rungtynes stebėti gyvai.

Šiemet vyksiantis pasaulio krepšinio čempionatas organizuojamas Kinijoje, taigi Martynas nori pamatyti kiek įmanoma daugiau rungtynių gyvai, taigi turi suplanuoti savo išlaidas rungtynių, lėktuvų bilietams, bei apsistojimui ir negali viršyti čempionatui skirtų pinigų limito.

Parašykite programą, kuri suskaičiuotų, kuri nurodytų kurių rungtynių bilietus Martynas turėtų nusipirkti norėdamas aplankyti kuo daugiau rungtynių.

Pirmoje pradinių duomenų failo **Duomenys.txt** eilutėje pateikta Martyna turima suma skirta pasaulio krepšinio čempionatui. Antroje eilutėje nurodyta lėktuvo bilieto kaina **bilKaina** į vieną pusę. Bilietas atgal perkamas papildomai ir jo kaina yra tokia pati, kaip ir už bilietą pirmyn. Trečioje eilutėje nurodyta apsistojimo kaina **apsKaina**. Trečioje nurodytas preliminarus rungtynių skaičius **n**, kuriose dalyvaus Lietuvos rinktinė. Sekančiose **n** eilučių nurodytos šalys, prieš kurias žais Lietuva, bei rungtynių bilieto kaina. Rezultatų faile **Rezultatai.txt** reikia išvesti rungtynes, bei rungtynių kaina, kurias stebės Martynas.

**Pastaba**: Martynas privalo pirkti bilietus pirmyn ir atgal, bei apsistoti viešbutyje.

|  |  |
| --- | --- |
| Duomenys.txt | Rezultatai.txt |
| 1060  250.50  350.10  5  Kroatija 90  Rusija 45  Kinija 79  Ispanija 120  JAV 78 | Lietuva - Rusija  Lietuva - JAV  Lietuva - Kinija |

1. **Reikalavimai**

* Naudokite sveikųjų skaičių ir eilutės tipo (string) masyvus
* Parašykite pradinių duomenų skaitymo funkciją **void**.
* Parašykite funkciją **void** atrenkančią, kurias rungtynes stebės Martynas (sudaryti masyvą su string tipo reikšmėmis, kuriose nurodomos rungtynių dalyvės)
* Parašykite rezultatų įrašymo į failą funkciją **void**.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito (double A[], int &n, double &vis, string B[]);

void Raso(string A[], int n);

void Atrinkimas(double A[], int &n, string B[], double vis);

int main(){

double A[100], vis;

string B[100];

int n;

Skaito(A, n, vis, B);

Atrinkimas(A, n, B, vis);

Raso(B, n);

return 0;

}

void Skaito (double A[], int &n, double &vis, string B[])

{

ifstream df ("Duomenys.txt");

double lekt, aps;

df >> vis >> lekt >> aps >> n;

vis = vis - lekt \* 2 - aps;

for (int i = 0; i < n; i++)df >> B[i] >> A[i];

df.close();

}

void Raso(string A[], int n)

{

ofstream rf ("Rezultatai.txt");

for (int i = 0; i < n; i++)rf << "Lietuva - " << A[i] << endl;

rf.close();

}

void Atrinkimas(double A[], int &n, string B[], double vis)

{

int laik, k = 0;

string laik1;

for (int i = 0; i < n; i++){

for (int i1 = i; i1 < n; i1++){

if(A[i] > A[i1]){

laik = A[i];

A[i] = A[i1];

A[i1] = laik;

laik1 = B[i];

B[i] = B[i1];

B[i1] = laik1;

}

}

}

while(vis - A[k] >= 0){

vis = vis - A[k];

k++;

}

n = k;

}

#### Sodinukai

1. **Užduotis**

Pranas augina medelių sodinukus, kuriuos pardavinėja kitiems sodininkams. Pardavimui skirti sodinukai turi būti ne žemesni už **a1** cm ir ne aukštesni už **a2** cm. Prano medelyne auga **n** sodinukų. Rengdamasis rudeninei mugei, Pranas išmatavo kiekvieno sodinuko aukštį ir surašė duomenis faile **sodinukai\_data.txt**. Pirmoje eilutėje įrašytas sodinukų kiekis **n** (1 ≤ n ≤ 100). Sekančiose **n** eilučių įrašyti sodinukų aukščiai (sveikieji skaičiai). Paskutinėje eilutėje – du sveikieji skaičiai **a1** ir **a2**, nurodantys, kokiame aukščių intervale (imtinai) yra pardavimui tinkamų sodinukų aukščiai.

Parašykite programą, kuri rezultatų faile **sodinukai\_res.txt** išspausdintų:

* Vidutinį visų medelyne augančių sodinukų aukštį vieno skaitmens po kablelio tikslumu.
* Rudeninei mugei atrinktų sodinukų kiekį.
* Rudeninei mugei atrinktų sodinukų vidutinį aukštį vieno skaitmens po kablelio tikslumu.
* Rudeninei mugei atrinktų sodinukų aukščių sąrašą vienoje eilutėje (jeigu tokių sodinukų nėra, išspausdinti žodį 'NE' didžiosiomis raidėmis).

|  |  |
| --- | --- |
| sodinukai\_data.txt | sodinukai\_res.txt |
| 5  53  45  56  54  56  50 60 | 52.8  4  54.8  53 56 54 56 |

1. **Reikalavimai**

* Duomenų saugojimui naudoti statinius (!) masyvus.
* Aprašyti funkciją **void Skaityti()**, kuri nuskaito pradinius duomenis.
* Aprašyti funkciją **void Atrinkti()**, kuri sudeda rudeninei mugei atrinktus sodinukus į naują masyvą.
* Aprašyti funkciją **double Vidurkis()**, kuri grąžina vidutinį nurodytų sodinukų aukštį (į ją reikėtų kreiptis du kartus).
* Aprašyti funkciją **void Rasyti()**, kuri išspausdina rezultatus.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito (int A[], int &n, int &a, int &a1);

void Raso(int A[], int n, double vidvis, double vidpard, int k, int B[]);

void Atrinkti(int A[], int n, int a, int a1, int B[], int &k);

double Vidurkis(int A[], int n);

int main(){

int A[100], n, a, a1, B[100], k = 0;

Skaito(A, n, a, a1);

Atrinkti(A, n, a, a1, B, k);

Raso(A, n, Vidurkis(A, n), Vidurkis(B, k), k, B);

return 0;

}

void Skaito (int A[], int &n, int &a, int &a1)

{

ifstream df ("sodinukai\_data.txt");

df >> n;

for (int i = 0; i < n; i++)df >> A[i];

df >> a >> a1;

df.close();

}

void Raso(int A[], int n, double vidvis, double vidpard, int k, int B[])

{

ofstream rf ("sodinukai\_res.txt");

rf << fixed << setprecision(1) << vidvis << endl;

rf << k << endl;

rf << fixed << setprecision(1) << vidpard << endl;

if (B[0] == -1)rf << "NE" << endl;

else{

for(int i = 0; i < k; i++){

rf << B[i] << " ";

}

}

rf << endl;

rf.close();

}

void Atrinkti(int A[], int n, int a, int a1, int B[], int &k)

{

for (int i = 0; i < n; i++){

if(A[i] >= a && A[i] <= a1){

B[k] = A[i];

k++;

}

}

if (k == 0)B[0] = -1;

}

double Vidurkis(int A[], int n)

{

if (n != 0){

double s = 0;

for(int i = 0; i < n; i++)s = s + A[i];

return s / n;

}

else return 0;

}

#### Detalės

1. **Užduotis**

Pradinių duomenų failo **detales\_data.txt** pirmoje eilutėje įrašytas bendrovės sandėlyje esančių detalių rūšių kiekis **n** (n < 1000). Likusiose **n** eilučių įrašyti duomenys apie kiekvieną detalę: triženklis detalės kodas (sveikasis skaičius), tos rūšies detalių skaičius (sveikasis skaičius), detalės vieneto kaina (realusis skaičius) ir brokuotų detalių kiekis (sveikasis skaičius).

Parašykite programą, skaičiuojančią:

* kiek yra kiekvienos rūšies nebrokuotų detalių;
* kiek kainavo kiekvienos rūšies nebrokuotos detalės;
* kiek iš viso buvo brokuotų detalių;
* kiek kainavo visos detalės (įskaitant ir brokuotas);
* kiek procentų nuostolių patirs bendrovė dėl brokuotų detalių.

Skaičiavimų rezultatus išveskite į rezultatų failą **detales\_res.txt**. Rezultatų faile turi būti: pirmosiose **n** eilučių: pradiniai duomenys, nebrokuotų detalių skaičius, nebrokuotų detalių kaina, visų detalių kaina (įskaitant ir brokuotas), po to išvedamas bendras brokuotų detalių skaičius, pinigų suma, kurią kainavo visos detalės ir kiek procentų nuostolių patirs bendrovė dviejų ženklų po kablelio tikslumu.

|  |  |
| --- | --- |
| detales\_data.txt | detales\_res.txt |
| 3  123 12 1.20 5  213 15 2.30 9  415 14 3.21 2 | 123 12 1.20 5 7 8.40 14.40  213 15 2.30 9 6 13.80 34.50  415 14 3.21 2 12 38.52 44.94  16  93.84  35.29 |

1. **Reikalavimai**

* Naudokite sveikųjų ir realiųjų skaičių masyvus: sveikųjų – kiekiams ir kodams, realiųjų – detalių kainoms saugoti.
* Sukurkite pradinių duomenų skaitymo funkciją **void**. Joje į masyvus sudėkite detalių kodus, kiekius (visų ir brokuotų) ir detalės vieneto kainas.
* Sukurkite funkciją **void**, kuri suformuotų: nebrokuotų detalių masyvą (sveikųjų skaičių masyvas) bei nebrokuotų ir visų detalių kainų masyvus (realiųjų skaičių).
* Sukurkite detalių kainų sumos skaičiavimo funkciją, grąžinančią apskaičiuotą sumą per funkcijos vardą. Funkciją panaudokite atlikdami d ir e užduotis.
* Sukurkite brokuotų detalių kiekio skaičiavimo funkciją, grąžinančią apskaičiuotą kiekį per funkcijos vardą. Funkciją panaudokite atlikdami c užduotį.
* Rezultatų išvedimui į failą parašykite funkciją **void**.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito (int &n, int A[], int B[], double C[], int D[]);

void Formuoja(int n, int A[], int B[], int C[], double D[], double E[], double F[]);

void Raso(int n, int A[], int B[], double C[], int D[], int E[], double G[], double F[], int vis, double sum, double nuost);

int Detsum(int n, int A[]);

double Detkain(int n, double A[]);

int main(){

int A[100], B[100], D[100], E[100], n, N[100];

double C[100], F[100], G[100];

Skaito(n, A, B, C, D);

Formuoja(n, B, D, E, C, F, G);

Raso(n, A, B, C, D, E, G, F, Detsum(n, D), Detkain(n, F), (Detkain(n, F) - Detkain(n, G)) / Detkain(n, F) \* 100);

return 0;

}

void Skaito (int &n, int A[], int B[], double C[], int D[])

{

ifstream df ("detales\_data.txt");

df >> n;

for (int i = 0; i < n; i++)df >> A[i] >> B[i] >> C[i] >> D[i];

df.close();

}

void Formuoja(int n, int A[], int B[], int C[], double D[], double E[], double F[])

{

for (int i = 0; i < n; i++){

C[i] = A[i] - B[i];

E[i] = D[i] \* A[i];

F[i] = C[i] \* D[i];

}

}

int Detsum(int n, int A[])

{

int s = 0;

for (int i = 0; i < n; i++)s = s + A[i];

return s;

}

void Raso(int n, int A[], int B[], double C[], int D[], int E[], double G[], double F[], int vis, double sum, double nuost)

{

ofstream rf ("detales\_res.txt");

for (int i = 0; i < n; i++)rf << A[i] << " " << B[i] << " " << fixed << setprecision(2) << C[i] << " " << D[i] << " " << E[i] << " " << fixed << setprecision(2) << G[i] << " " << F[i] << endl;

rf << vis << endl << fixed << setprecision(2) << sum << endl << nuost;

rf.close();

}

double Detkain(int n, double A[])

{

double s = 0;

for (int i = 0; i < n; i++)s = s + A[i];

return s;

}

#### Įmonės sandoriai

1. **Užduotis**

Įmonė sudarinėja daug sandorių ir dėl to mėnesio gale nori pamatyti statistiką apie juos.

Pradinių duomenų failo **sandoriai.txt** pirmoje eilutėje įrašytas įmonės sandorių kiekis per mėnesį **n**. Likusiose **n** eilučių: sandorio suma (*gali būti ir teigiama ir neigiama*).

Parašykite programą, skaičiuojančią:

* kiek yra įeinančių (teigiamų) sandorių;
* įeinančių (teigiamų) sandorių vidurkį;
* kiek yra išeinančių (neigiamų) sandorių;
* išeinančių (neigiamų) sandorių vidurkį;
* kokį pelną (arba nuostolį) įmonė gavo.

PASTABA: pelnas (arba nuostolis) suskaičiuojamas pagal tokią formulę: **PELNAS = PAJAMOS - IŠLAIDOS**

Skaičiavimų rezultatus išveskite į rezultatų failą **statistika.txt**. Rezultatų faile turi būti:

* įeinančių sandorių kiekis
* įeinančių sandorių vidutinė suma
* išeinančių sandorių kiekis
* išeinančių sandorių vidutinė suma
* įmonės pelnas (arba nuostolis).

Skaičiai turi būti atskirti nauja eilute.

PASTABA: jei nėra įeinančių ar išeinančių sandorių, jų kiekis ir vidurkis yra 0.

|  |  |
| --- | --- |
| sandoriai.txt | statistika.txt |
| 5100.2  200.0  -300.5  -500.7  45.1 | 3  115.1  2  -400.6  -455.9 |

1. **Reikalavimai**

* Sukurkite pradinių duomenų skaitymo funkciją void. Joje į masyvą sudėkite sandorių sumas.
* Naudokite realiųjų skaičių masyvą sandorių sumoms saugoti.
* Suapvalinkite vidurkius ir pelną iki 2 skaičių po kablelio.
* Sandorių kiekiams naudokite sveikuosius skaičius.
* Rezultatų išvedimui į failą parašykite funkciją **void**.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito (double A[], int &n);

void Raso(double A[], int n);

void Statistika(double A[], int n, double B[]);

int main(){

double A[100], B[100];

int n;

Skaito(A, n);

Statistika(A, n, B);

Raso(B, n);

return 0;

}

void Skaito (double A[], int &n)

{

ifstream df ("sandoriai.txt");

df >> n;

for (int i = 0; i < n; i++)df >> A[i];

df.close();

}

void Raso(double A[], int n)

{

ofstream rf ("statistika.txt");

for (int i = 0; i < n; i++){

if (i == 4 || i == 1 || i == 3)rf << fixed << setprecision(2) << A[i] << endl;

else rf << fixed << setprecision(0) << A[i] << endl;

}

rf.close();

}

void Statistika(double A[], int n, double B[])

{

int kiein = 0, kisein = 0;

double ieinvid = 0, iseinvid = 0, peln;

for (int i = 0; i < n; i++){

if (A[i] > 0){

kiein++;

ieinvid = ieinvid + A[i];

}

else if (A[i] < 0){

kisein++;

iseinvid = iseinvid + A[i];

}

}

B[0] = kiein;

B[4] = ieinvid + iseinvid;

B[2] = kisein;

if (kiein == 0)B[1] = 0;

else B[1] = ieinvid / kiein;

if (kisein == 0)B[3] = 0;

else B[3] = iseinvid / kisein;

}

#### Tritaškių konkursas

1. **Užduotis**

Kiemo draugai krepšinio aikštelėje sugalvojo išbandyti jėgas „Tritaškių konkurse“ ir taip išsiaiškinti, kuris iš jų yra taikliausias nuo tritaškio zonos. Kiekvienas jaunuolis metė po 2 metimus iš 5 pozicijų, taigi iš viso 10 metimų kiekvienam draugui.

Parašykite programą, kuri apskaičiuoja, kiek iš viso kartu sudėjus kiemo draugai įmetė tritaškių, tų tritaškių bendra vidurkį .

**Pradiniai duomenys:**

* Duomenų faile pirmojoje eilutėje yra įrašytas skaičius **n**, nurodantis, kiek draugų kovojo dėl „ Tritaškių konkurso“ nugalėtojo vardo.
* Kitose eilutėse yra įrašyta po vieną skaičių, kurie parodo, kiek kiekvienas draugas įmetė tritaškių.

**Rezultatai:**

* Rezultatų faile turi būti du skaičiai, kurie būtų atskirti tarpo simboliu.
* Kiek iš viso taiklių tritaškių įmetė visi draugai.
* Koks yra pataikytų tritaškių bendras vidurkis.

|  |  |
| --- | --- |
| duom.txt | rez.txt |
| 8  3  7  5  2  8  4  6  1 | 36 4,5 |

1. **Reikalavimai**

* Funkciją **void Skaityti()**, nuskaityti duomenis iš failo.
* Funkciją **int Suma()**, suskaičiuoti visų draugų įmestų tritaškių sumą.
* Funkciją **double Vidurkis()**, suskaičiuoti visų įmestų tritaškių aritmetinį vidurkį.
* Funkciją **void Spausdinti()**, išspausdinti rezultatus į failą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito (int A[], int &n);

void Raso(int tri, double vid);

int Suma (int A[], int n);

double Vidurkis(int A[], int n);

int main(){

int A[100], n;

Skaito(A, n);

Raso(Suma(A, n), Vidurkis(A, n));

return 0;

}

void Skaito (int A[], int &n)

{

ifstream df ("duom.txt");

df >> n;

for (int i = 0; i < n; i++)df >> A[i];

df.close();

}

void Raso(int tri, double vid)

{

ofstream rf ("rez.txt");

rf << tri << " " << fixed << setprecision(1) << vid << endl;

rf.close();

}

int Suma(int A[], int n)

{

int s = 0;

for(int i = 0; i < n; i++)s = s + A[i];

return s;

}

double Vidurkis(int A[], int n)

{

double s = 0;

for (int i = 0; i < n; i++)s = s + A[i];

return s / n;

}

#### Populiarumo konkursas

1. **Užduotis**

Klasės draugai sumąstė, jog nori patikrinti kas daugiausiai surinko like`u facebook`e ant savo patalpintų nuotraukų

Kiekvienas draugas surašė visų savo nuotraukų like`ų skaičių.

Palyginkite kuris draugas yra surinkęs daugiausiai like`ų vidutiniškai.

Pradinių duomenų failo **nuotraukos\_data.txt** pirmoje eilutėje įrašytas draugų kiekis **n** ir nuotraukų per draugą kiekis **m**.

Suskaičiuokite kuris draugas vidutiniškai surinko daugiausiai like`ų,

Duomenų faile **nuotraukos\_res.txt** išveskite tik to draugo nuotraukų like`us ir kitoje eilutėje vidurkį.

Vidurkį apvalinti iki vieno skaičiaus po kablelio. Vidurkiai nebus vienodi.

|  |  |
| --- | --- |
| nuotraukos\_data.txt | nuotraukos\_res.txt |
| 4 2  13 14  2 5  6 8  14 67 | 14, 67  40.5 |

1. **Reikalavimai**

* Sukurkite funkcija double Vidurkis(int masyvas), kuri gražintų suskaičiuotą masyvo vidurkį per vardą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito (int A[], int &n, int &m);

void Raso(int A[], int m, double vid);

void Laikaidid(int n, int m, int A[], int B[]);

double Vidurkis(int A[], int n);

int main(){

int A[100], n, m, B[100];

Skaito(A, n, m);

Laikaidid(n, m, A, B);

Raso(B, m, Vidurkis(B, m));

return 0;

}

void Skaito (int A[], int &n, int &m)

{

ifstream df ("nuotraukos\_data.txt");

df >> n >> m;

for (int i = 0; i < n \* m; i++)df >> A[i];

df.close();

}

void Raso(int A[], int m, double vid)

{

ofstream rf ("nuotraukos\_res.txt");

for (int i = 0; i < m; i++){

rf << A[i];

if(i != m - 1)rf << ", ";

}

rf << endl << fixed << setprecision(1) << vid;

rf.close();

}

void Laikaidid(int n, int m, int A[], int B[])

{

int ind = 0, s = 0, s1 = 0;

for (int i = 0; i < n; i++){

for(int i1 = 0; i1 < m; i1++)s1 = s1 + A[i \* m + i1];

if (s1 > s){

s = s1;

ind = i;

}

}

for (int i = 0; i < m; i++)B[i] = A[ind \* m + i];

}

double Vidurkis(int A[], int n)

{

double s = 0;

for (int i = 0; i < n; i++)s = s + A[i];

return s / n;

}

#### Ateiviai žemėje: iššūkis

1. **Užduotis**

Lemtingą sausio sekmadienį žemę užpuolė ateiviai. Pamatę, kad žemiečiai nėra pasiruošę tarpgalaktiniams mūšiams, ateiviai suteikė šansą mesdami mums iššūkį ateiviškame žaidime.

Žaidimo taisyklės

Žaidimą sudaro du žaidėjai ir kortos. Ant kortos gali būti užrašytas skaičius nuo 1 iki 4. Žaidime yra keturi vienetai, keturi dvejetai, keturi trejetai ir keturi ketvertai – 16 kortų. Žaidimo tikslas - surinkti kortas, kurių suma kuo artimesnė septynetui, bet ne daugiau.

Kortų kaladė yra išmaišoma ir kiekvienam žaidėjui padedama po 5 užverstas kortas. Iš pradžių kiekvienam žaidėjui yra atverčiama po kortą. Atverstos kortos skaičius yra pridedamas prie bendros žaidėjo sumos pradedant 0. Žaidėjas gali rinktis, prašyti atversti kitą kortą, arba užbaigti žaidimą su turima suma. Žaidėjui nusprendus užbaigti žaidimą likusios žaidėjo kortos yra atverčiamos nepridedant jų prie bendros žaidėjo sumos. Žaidėjas pralaimi jei galutinė suma viršija 7 arba yra mažesnė už kito žaidėjo

Užduotis

Kaip ir buvo galima suprasti, žaidimą žais ateivis prieš žemietį. Tavo kaip programuotojo tikslas – apdoroti žaisto žaidimo duomenis ir pasakyti kas laimėjo ir, jei neviršijo septyneto, ar laiku žemietis užbaigė žaidimą, t.y. ar galėjo žemietis prašyti dar vienos kortos ir nepralaimėti.

Turėk omeny, kad žaidimą gali laimėti arba pralaimėti abu žaidėjai, pvz. abiems surinkus tą patį taškų skaičių, laimi abu. Abiems viršijus 7 - pralaimi abu.

Duomenų ir rezultatų paaiškinimai

Pradinių duomenų formatas:

Z(1) Z(2) Z(3) 0 Z(4) Z(5)

A(1) 0 A(2) A(3) A(4) A(5)

Z(n) - Žemiečio korta.

A(n) - Ateivio korta.

0 eilutėje reiškia žmogaus arba ateivio pasirinkimą nebeimti likusių kortų.

Rezultatų formatas:

W K

W: 0 - laimėjo žemietis, 1 - laimėjo ateivis, 2 - laimėjo abu, 3 - pralaimėjo abu

K: 0 - žemietietis dar galėjo nepralaimėdamas paprašyti kitos kortos, 1 - žemietis laiku užbaigė žaidimą. Nieko nereikia išvesti jei žemiečio suma >= 7.

|  |  |
| --- | --- |
| data.txt | results.txt |
| 4 3 0 1 2 2  4 1 2 0 4 4 | 2 |

1. **Reikalavimai**

* Funkcija duomenų **skaitymui** iš failo į masyvą.
* Funkcija randanti kas laimėjo žaidimą pagal paduotus **masyvus**.
* Funkcija **tikrinanti** ar žemietis galėjo traukti kitą kortą.
* Funkciją rezultatų **rašymui** į failą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito (int A[], int B[]);

void Raso(int l, int t);

int Laimejo(int A[], int B[]);

int Traukimas(int A[]);

int main(){

int A[100], B[100], l, t;

Skaito(A, B);

l = Laimejo(A, B);

t = Traukimas(A);

Raso(l, t);

return 0;

}

void Skaito (int A[], int B[])

{

ifstream df ("data.txt");

for (int i = 0; i < 6; i++)df >> A[i];

for (int i = 0; i < 6; i++)df >> B[i];

df.close();

}

void Raso(int l, int t)

{

ofstream rf ("results.txt");

rf << l << " ";

if (t != 2)rf << t;

rf.close();

}

int Laimejo(int A[], int B[])

{

int s = 0, s1 = 0, i = 0;

while (A[i]!=0 && i < 5){

s = s + A[i];

i++;

}

i = 0;

while (B[i]!= 0 && i < 6){

s1 = s1 + B[i];

i++;

}

if (s > s1 && s <= 7 || s1 > 7 && s <= 7)return 0;

else if (s1 > s && s1 <= 7 || s > 7 && s1 <= 7)return 1;

else if (s1 == s && s <= 7)return 2;

else return 3;

}

int Traukimas(int A[])

{

int s = 0, i = 0;

while (A[i]!=0 && i < 5){

s = s + A[i];

i++;

}

if(i != 5 && s < 7 && s + A[i + 1] <= 7)return 0;

else if(i != 5 && s < 7 && s + A[i + 1] > 7)return 1;

else return 2;

}

#### Ateiviai žemėje: kelionė namo

1. **Užduotis**

Po netikėto ateivių mesto iššūkio paaiškėjo, kad ateivių tikslas buvo ne užpulti žemę, o paprašyti pagalbos.

Ateivių kuras

Ateiviams pritrūko degalų grįžti namo! Ateivių kurą sudaro anglis, druska ir vanduo. Šviesmečiui nukeliauti reikia 10.3kg. anglies, 15.2kg. druskos ir 100l. vandens. Ateiviai davė sąrašą skaičių žyminčių kelionės etapus šviesmečiais.

Tavo užduotis - paskaičiuoti kiek anglies, druskos ir vandens reiks ateivių kelionei.

Duomenų ir rezultatų paaiškinimai

Pradinių duomenų formatas:

n

b(1) b(2) b(3) … b(n)

n - etapų skaičius.

b(n) - etapo atstumas šviesmečiais

Rezultatų formatas:

C N H

C - anglies kiekis (kg.), N - druskos kiekis (kg.), H - vandens kiekis (l.)

Visus kiekius pateikite suapvalinę iki artimiausio sveikojo skaičiaus (0.5 -> 1)

|  |  |
| --- | --- |
| data.txt | results.txt |
| 8  10 3 50 1 2 10 7 8 | 937 1383 9100 |

1. **Reikalavimai**

* Funkcija duomenų **skaitymui** iš failo į masyvą.
* Funkcija apskaičiuojanti ir grąžinanti **masyvą** su reikalingais anglies, druskos ir vandens kiekiais.
* Funkciją rezultatų **rašymui** į failą

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int A[], int &n);

void Kiekis(int A[], int n, double B[]);

void Raso(double A[], int n);

int main(){

int A[100], n;

double B[100];

Skaito(A, n);

Kiekis(A, n, B);

Raso(B, 3);

return 0;

}

void Skaito (int A[], int &n)

{

ifstream df ("data.txt");

df >> n;

for (int i = 0; i < n; i++)df >> A[i];

df.close();

}

void Kiekis(int A[], int n, double B[])

{

for(int i = 0; i < 3; i++)B[i] = 0;

for (int i = 0; i < n; i++){

B[0] = B[0] + A[i] \* 10.3;

B[1] = B[1] + A[i] \* 15.2;

B[2] = B[2] + A[i] \* 100;

}

}

void Raso(double A[], int n)

{

ofstream rf ("results.txt");

for (int i = 0; i < n; i++)rf << fixed << setprecision(0) << A[i] << " ";

rf.close();

}

#### Finansų programėlė

1. **Užduotis**

Virginija sumąstė paskaičiuoti savo finansus išleistus maisto produktams, ji nenorėjo siųstis papildomos programos, todėl pradėjo rašyti savo išlaidas tiesiai į mobiliojo telefono užrašinę. Tačiau ji ne tik nori sužinoti kiek išleidžia pinigų produktams, bet nori žinoti ir kokioms produktų kategorijoms kiek pinigų išleidžia.

Iš viso yra 5 kategorijos: M - mėsa, D - daržovės, G - gėrimai, S - saldumynai ir K - kita.

Pradinių duomenų failo **produktai\_data.txt** pirmoje eilutėje įrašytas produktų kiekis n.

Kitose eilutėse surašyta produktų informacija, pirmoje vietoje produkto kategorija, antroje kiekis ir trečioje kaina.

Atsakymų faile **produktai\_rez.txt** išveskite kiek pinigų yra išleista kiekvienai produktų kategorijai ir visų produktų pinigų sumą.

Jeigu nėra kurios nors kategorijos ją vis tiek išvesti su 0 kaip sumą.

Kategorijos visada išvedamos tokia seka kaip parašytą aukščiau.

|  |  |
| --- | --- |
| produktai\_data.txt | produktai\_rez.txt |
| 8  M 2 2.5  G 4 6  K 1 1.99  M 2 0.4  D 1 0.6  K 3 5.6  S 3 1.2  S 5 2.4 | Mesa: 5.8  Darzoves: 0.6  Gerimai: 24.0  Saldumynai: 15.6  Kita: 18.8  64.8 |

1. **Reikalavimai**

* Atsakymą apvalinti iki vieno skaičiaus po kablelio ir rašyti tik lotyniškomis raidėmis.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int &n, string P[], int K[], double KA[]);

void Kainos(int n, string P[], int K[], double KA[], double V[]);

void Raso(double V[]);

int main(){

int n, K[100];

double KA[100], V[100];

string P[100];

Skaito(n, P, K, KA);

Kainos(n, P, K, KA, V);

Raso(V);

return 0;

}

void Skaito(int &n, string P[100], int K[], double KA[])

{

ifstream df ("produktai\_data.txt");

df >> n;

for (int i = 0; i < n; i++)df >> P[i] >> K[i] >> KA[i];

df.close();

}

void Kainos(int n, string P[], int K[], double KA[], double V[])

{

int ind;

for(int i = 0; i < 6; i++)V[i] = 0;

for (int i = 0; i < n; i++){

if (P[i] == "M")ind = 0;

else if (P[i] == "D")ind = 1;

else if (P[i] == "G")ind = 2;

else if (P[i] == "S")ind = 3;

else ind = 4;

V[ind] = V[ind] + K[i] \* KA[i];

V[5] = V[5] + K[i] \* KA[i];

}

}

void Raso(double V[])

{

ofstream rf ("produktai\_rez.txt");

rf << fixed << setprecision(1) << "Mesa: " << V[0] << endl << "Darzoves: " << V[1] << endl << "Gerimai: " << V[2] << endl << "Saldumynai: " << V[3] << endl << "Kita: " << V[4] << endl << V[5];

rf.close();

}

#### Pirmasis automobilis

1. **Užduotis**

Išlaikiusi vairavimo egzaminą Margarita nusprendė nuo kovo 1 d. iki vasaros (birželio 1d.) taupyti savo pirmajam automobiliui. Margaritai tėvai nori padėti, todėl pažadėjo duoti lygiai tiek pat kiek turės ji. Margarita kasdien gauna tam tikrą sumą pinigų ir viską kas lieką deda į kiaulę taupyklę, kurioje jau dabar yra 500. Padėkime Margaritai nustatyti preliminarią pirmojo automobilio kainą, pagal sutaupomų kišenpinigių vidurkį, kad ji jau dabar galėtų pradėti rinktis automobilį.

Pirmoje failo eilutėje pateiktas prieš tai buvusių dienų skaičius **n** rodantis kiek dienų iki kovo 1d Margarita atsimena sutaupytų ir išleistų kišenpinigių kiekius. Kitose **n** eilučių pateikta buvusios dienos gautų ir išleistų kišenpinigių kiekis

PASTABA: laikykime kad iki vasaros (birželio 1d.) yra 92 dienos.

|  |  |
| --- | --- |
| duomenys.txt | rezultatai.txt |
| 5  10 3.5  9 4.2  11 7.2  8 3.1  4 2.6 | 1787.52 |

1. **Reikalavimai**

* Sukurkite pradinių duomenų skaitymo funkciją **void**. Joje į masyvą sudėkite likusius kišenpinigius.
* Naudokite realiųjų skaičių masyvą sandorių sumoms saugoti.
* Atsakymą pateikite suapvalinę iki dviejų skaičių po kablelio.
* Rezultatų išvedimui į failą parašykite funkciją **void**.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(double A[], double B[], int &n);

void Skirtumas(double A[], double B[], double C[], int n);

double Vidurkis(double A[], int n);

double Santaupos(double vid);

void Raso(double s);

int main(){

int n;

double Pr[100], Po[100], Sk[100];

Skaito(Pr, Po, n);

Skirtumas(Pr, Po, Sk, n);

Raso(Santaupos(Vidurkis(Sk, n)));

return 0;

}

void Skaito(double A[], double B[], int &n)

{

ifstream df ("duomenys.txt");

df >> n;

for (int i = 0; i < n; i++)df >> A[i] >> B[i];

df.close();

}

void Skirtumas(double A[], double B[], double C[], int n)

{

for (int i = 0; i < n; i++)C[i] = A[i] - B[i];

}

double Vidurkis(double A[], int n){

double s = 0;

for (int i = 0; i < n; i++)s = s + A[i];

return s / n;

}

double Santaupos(double vid)

{

return (vid \* 92 + 500) \* 2;

}

void Raso(double s)

{

ofstream rf ("rezultatai.txt");

rf << fixed << setprecision(2) << s;

rf.close();

}

#### Studentų stipendijos

1. **Užduotis**

Baigiantis kiekvienam semestrui 2 geriausiems studentam yra skiriamos stipendijos. Parenkite programą, kuri atrinktų, kurie studentai gaus stipendijas.

Geriausias studentas yra tas, kurio pažymių vidurkis yra aukščiausias.

**Pradiniai duomenys:**

Duomenų faile **pazymiai.txt** pirmoje eilutėje skaičius (**n**), nurodantis studentų kiekį ir skaičius (**m**), rodantis iš kelių modulių studentai gavo pažymius. Kitose **n** eilučių yra **m** pažymių. Visi skaičiai yra atskirti tarpo simboliu.

Apskaičiuokite kiekvieno studento pažymių **vidurkį**(2 skaitmenys po kablelio) ir atrinkite **2** geriausius studentus, kurie gaus stipendijas.

Į rezultatų failą (**rezultatai.txt**) išveskite **2** geriausių studentų pažymių vidurkius(2 skaitmenys po kablelio.)

|  |  |
| --- | --- |
| pazymiai.txt | rezultatai.txt |
| 20 5  10 8 9 9 7  6 6 7 10 10  10 5 8 6 10  7 9 9 6 5  8 7 10 5 9  9 10 5 9 6  7 7 5 9 7  6 5 10 10 5  8 6 8 8 9  5 6 9 10 10  9 8 7 9 5  5 5 7 8 6  5 10 9 10 9  10 6 9 8 9  7 8 7 6 6  5 8 10 8 7  5 5 9 8 8  8 6 7 7 10  8 10 5 6 7  5 8 6 6 5 | 8.60 8.60 |

1. **Reikalavimai**

* **int Suma()** funkcija, skaičiuojantis, kiekvieno studento bendrą pažymių sumą.
* **double Vidurkis()** funkcija, skaičiuojanti, kiekvieno studento pažymių vidurkį.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int A[], int &n, int &m);

double Suma(int A[], int n);

double Vidurkis(double s, int n);

void Vidurkiai(int A[], int n, int m, double B[]);

void Dudid(double A[], int n, double &pirm, double &antr);

void Raso(double vid, double vid1);

int main(){

int A[1000], n, m;

double B[100], pirm = 0, antr = 0;

Skaito(A, n, m);

Vidurkiai(A, n, m, B);

Dudid(B, n, pirm, antr);

Raso(pirm, antr);

return 0;

}

void Skaito(int A[], int &n, int &m)

{

ifstream df ("pazymiai.txt");

df >> n >> m;

for(int i = 0; i < n \* m; i++)df >> A[i];

df.close();

}

void Vidurkiai(int A[], int n, int m, double B[])

{

int C[100];

for(int i = 0; i < n; i++){

for (int i1 = 0; i1 < m; i1++)C[i1] = A[i \* m + i1];

B[i] = Vidurkis(Suma(C, m), m);

}

}

double Vidurkis(double s, int n)

{

return s / n;

}

void Dudid(double A[], int n, double &pirm, double &antr)

{

for(int i = 0; i < n; i++){

if (A[i] > pirm){

antr = pirm;

pirm = A[i];

}

else if (A[i] > antr)antr = A[i];

}

}

void Raso(double vid, double vid1)

{

ofstream rf ("rezultatai.txt");

rf << fixed << setprecision(2) << vid << " " << vid1;

rf.close();

}

double Suma(int A[], int n)

{

int s = 0;

for(int i = 0; i < n; i++)s = s + A[i];

return s;

}

#### Akcijų rinkos statistika

1. **Užduotis**

Akcijų rinkoje nuolatos parduodamos didžiųjų kompanijų (tokių kaip Google, Netflix, Microsoft, Silvercorp Metals, Taseko) akcijos. Kiekviena akcija turi numatytą tam tikrą akcijos kaina, kuri pastoviai svyruoja dėl ekonominių, politinių ar vidinių kompanijos veiksnių. Padėkime gauti konkrečios dienos parduotų akcijų statistiką:

* Kiek vidutiniškai akcijų pardavė kompanijos?
* Kokia vidutinė akcijos kaina kompanijai?
* Kiek kompanijos uždirbo iš akcijų vidutiniškai?

Pirmoje duomenų failo eilutėje pateiktas kompanijų kiekis **n**, kurių akcijomis buvo prekiauta tą dieną. Tolimesnėse **n** eilučių pateikta kompanijos parduotų akcijų skaičius, bei vienos akcijos kaina. Rezultatų failo pirmoje eilutėje pateikta kiek vidutiniškai akcijų pardavė kompanijos, antroje: kokia vidutinė akcijos kaina kompanijai ir trečioje: kiek kompanijos uždirbo iš akcijų vidutiniškai.

|  |  |
| --- | --- |
| duomenys.txt | rezultatai.txt |
| 7  467 600  125 550  366 420  566 125  488 233  566 358  895 400 | 496  359  178250 |

1. **Reikalavimai**

* Sukurkite pradinių duomenų skaitymo funkciją **void**. Joje į 2 atskirus masyvus sudėkite akcijų kiekius ir kainas.
* Naudokite sveikųjų skaičių masyvus.
* Atsakymus pateikite suapvalinę iki sveiko skaičiaus.
* Rezultatų išvedimui į failą parašykite funkciją **void**.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int &n, int A[], int B[]);

void Duom(int n, int A[], int B[], double &ak, double &aka, double &akav);

void Raso(double ak, double aka, double akav);

int main(){

int n, K[100], Ka[100];

double ak, aka, akav;

Skaito(n, K, Ka);

Duom(n, K, Ka, ak, aka, akav);

Raso(ak, aka, akav);

return 0;

}

void Skaito(int &n, int A[], int B[])

{

ifstream df ("duomenys.txt");

df >> n;

for (int i = 0; i < n; i++)df >> A[i] >> B[i];

df.close();

}

void Duom(int n, int A[], int B[], double &ak, double &aka, double &akav)

{

double skiek = 0, skain = 0;

for(int i = 0; i < n; i++){

skiek = skiek + A[i];

skain = skain + A[i] \* B[i];

}

ak = skiek / n;

aka = skain / skiek;

akav = skain / n;

}

void Raso(double ak, double aka, double akav)

{

ofstream rf ("rezultatai.txt");

rf << fixed << setprecision(0) << ak << endl << aka << endl << akav;

rf.close();

}

#### Susitikimų kelionės

1. **Užduotis**

Vaikinas išvyko studijuoti į Kauną, o mergina gimtajame mieste vis dar mokosi mokykloje. Norint jiems susitikti arba vaikinas, arba mergina turi važiuoti vienas pas kitą. Kadangi nei vienas, nei kitas dar neturi transporto priemonės, tad susitikimams reikia skirti nemažą dalį pinigų važiuoti traukiniu arba autobusu.

Pirmoje eilutėje yra du skaičiai: **n** – susitikimų skaičius ir **m** – visada bus 2 (pirmasis skaičius rodo nuvažiavimo sumą, antrasis skaičius grįžimo).

Apskaičiuokite, kiek vaikinas ir mergina kartu išleido pinigų vien kelionėms pirmyn/atgal, vidurkį bei kiek kiekvienas atskirai sumokėjo už keliones.

Jei sudėjus važiavimo pirmyn ir grįžimo sumokėta suma viršija 20.00 eurų, tuomet tai bus merginos išleisti pinigai, kadangi vaikinas yra studentas, jis turi 50 % nuolaidą. (Naudoti sąlyginį sakinį).

|  |  |
| --- | --- |
| duom.txt | rez.txt |
| 6 2  5.85 7.65  13.20 11.80  7.55 6.3  5.6 8.4  12.9 10.4  8.4 5.6 | 103.65  17.275  Vaikinas išleido – 55.35  Mergina išleido – 48.30 |

1. **Reikalavimai**

* Funkciją **void Skaityti()**, nuskaityti duomenis iš failo.
* Funkciją **double Suma()**, suskaičiuoti tiek vaikino, tiek merginos išleistą pinigų sumą.
* Funkciją **double Vidurkis()**, suskaičiuoti pirmyn/atgal vidurkį. (Skaičiuojama ne vienos kelionės vidurkis, o abiejų pirmyn/atgal).
* Funkciją **double Vaikinas()**, suskaičiuoti kiek pinigų išleido vaikinas.
* Funkciją **double Mergina()**, suskaičiuoti kiek pinigų išleido mergina.
* Funkciją **void Spausdinti()**, išspausdinti rezultatus į failą.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int &n, int &m, double A[]);

double Suma(int n, int m, double A[]);

double Vidurkis(double s, int n);

double Vaikinas(int n, int m, double A[]);

double Mergina(int n, int m, double A[]);

void Raso(double vis, double vid, double vaik, double merg);

int main(){

int n, m;

double A[100];

Skaito(n, m, A);

Raso(Suma(n, m, A), Vidurkis(Suma(n, m, A), n), Vaikinas(n, m, A), Mergina(n, m, A));

return 0;

}

void Skaito(int &n, int &m, double A[])

{

ifstream df ("duom.txt");

df >> n >> m;

for(int i = 0; i < n \* m; i++)df >> A[i];

df.close();

}

double Suma(int n, int m, double A[])

{

double s = 0;

for (int i = 0; i < n \* m; i++)s = s + A[i];

return s;

}

double Vidurkis(double s, int n)

{

return s / n;

}

double Vaikinas(int n, int m, double A[])

{

double s = 0, s1 = 0;

for (int i = 0; i < n; i++){

for(int i1 = 0; i1 < m; i1++)s1 = s1 + A[i \* m + i1];

if(s1 <= 20)s = s + s1;

s1 = 0;

}

return s;

}

double Mergina(int n, int m, double A[])

{

double s = 0, s1 = 0;

for (int i = 0; i < n; i++){

for(int i1 = 0; i1 < m; i1++)s1 = s1 + A[i \* m + i1];

if(s1 > 20)s = s + s1;

s1 = 0;

}

return s;

}

void Raso(double vis, double vid, double vaik, double merg)

{

ofstream rf ("rez.txt");

rf << fixed << setprecision(2) << vis << endl << fixed << setprecision(3) << vid << endl << "Vaikinas isleido - " << fixed << setprecision(2) << vaik << endl << "Mergina isleido - " << merg;

rf.close();

}

#### Kelionė

1. **Užduotis**

Tomas keliaudamas po Lietuvą, kelionės gale norėjo išsiaiškinti, koks visą kelionę buvo jo vidutinis automobilio greitis.

Parenkite programą, kuri padėtų Tomui išsiaiškinti, koks buvo jo vidutinis greitis.

**Pradiniai duomenys:**

Skaičius **n**, rodantis kiek skirtingų tarpų buvo, kai jo greitis kito. Kitose **n** eilučių yra **2** skaičiai - pirmas, rodantis atstumą (**metrais**), kurį nuvažiavo Tomas, o antras - laiko tarpas (**sekundėmis**) per kurį nuvažiavo tą atstumą. Skaičiai atskirti 1 tarpo simboliu.

Apskaičiuokite, koks buvo Tomas vidutinis greitis kelionės metu.

Rezultatų faile išspausdinkite vidutinį greitį m/s ir km/h atskirtus tarpo simboliu (**2 skaičių po kablelio tikslumu**).

|  |  |
| --- | --- |
| kelione.txt | rezultatai.txt |
| 5  3000 120  1500 70  2200 180  500 20  1300 45 | 19.54 70.34 |

1. **Reikalavimai**

* 2 masyvai atstumui laikyti bei laikui.
* **void Skaityti()** funkcija, duomenims skaityti.
* **int Suma()** funkcija, sumuojanti visą atstumą bei visą laiką.
* **double Vidurkis()** funkcija, skaičiuojanti vidutinį greitį.
* **void Spausdinti()** funkcija, rezultatams į failą spausdint.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int &n, int A[], int B[]);

int Suma(int n, int A[]);

double Vidurkis(int n, int A[], int B[]);

void Raso(double vidm, double vidkm);

int main(){

int n, K[100], L[100];

Skaito(n, K, L);

Raso(Vidurkis(n, K, L), Vidurkis(n, K, L) \* 3600 / 1000);

return 0;

}

void Skaito(int &n, int A[], int B[])

{

ifstream df ("kelione.txt");

df >> n;

for (int i = 0; i < n; i++)df >> A[i] >> B[i];

df.close();

}

int Suma(int n, int A[])

{

int s = 0;

for (int i = 0; i < n; i++)s = s + A[i];

return s;

}

double Vidurkis(int n, int A[], int B[])

{

double vid;

vid = Suma(n, A);

return vid / Suma(n, B);

}

void Raso(double vidm, double vidkm)

{

ofstream rf ("rezultatai.txt");

rf << fixed << setprecision(2) << vidm << " " << vidkm;

rf.close();

}

### Naujo masyvo formavimas iš kelių masyvų. Didžiausios ir mažiausios masyvo elementų reikšmių vieta.

#### Dalelė

1. **Užduotis**

Mokslininkas stebi vandens paviršiuje pakibusios kietosios dalelės judėjimą ir kas **N** sekundžių užrašo jos koordinates. Parašykite programą, kuri apskaičiuotų kietosios dalelės mažiausią, vidutinį, didžiausią greitį ir dalelės nukeliautą kelią.

Tekstiniame faile **dalele\_data.txt** pirmoje eilutėje užrašytas dalelės koordinačių skaičius **M** (**1 < M ≤ 100**) ir koordinačių fiksavimo intervalas **N** (**N > 0**) sekundėmis. Sekančiose failo eilutėse užrašytos dalelės koordinatės **Xi** ir **Yi** (**1 ≤ i ≤ M**) milimetrais (**Xi** ir **Yi** – realieji skaičiai).

Tekstiniame faile **dalele\_res.txt** spausdinkite kietosios dalelės mažiausią, vidutinį, didžiausią greitį ir dalelės nukeliautą kelią kaip nurodyta pavyzdyje.

|  |  |
| --- | --- |
| dalele\_data.txt | dalele\_res.txt |
| 5 2  1.8 2.7  -0.5 4.3  -1.4 3.2  0.1 1.4  0.7 -0.5 | Maziausias greitis: 0.71 mm/s  Vidutinis greitis: 1.07 mm/s  Didziausias greitis: 1.40 mm/s  Nukeliautas kelias: 8.56 mm |

1. **Reikalavimai**

* Naudokite realiųjų skaičių masyvus.
* Pradinių duomenų skaitymo funkcija **void**.
* Funkcija, surandanti mažiausią greitį.
* Funkcija, skaičiuojanti vidutinį greitį.
* Funkcija, surandanti didžiausią greitį.
* Funkcija, skaičiuojanti nukeliautą kelią.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

void Skaito (double A[], double B[], int &n, int &l);

void Greitis(double A[], double B[], int n, int l, double D[]);

double Maziausias(double A[], int n);

double Didziausias(double A[], int n);

double Kelias(double A[], int n, int l);

double Vidutinis(double A[], int n, int l);

using namespace std;

int main(){

double A[200], B[200], D[200];

int n, l;

ofstream rf ("dalele\_res.txt");

Skaito(A, B, n, l);

Greitis(A, B, n, l, D);

rf << "Maziausias greitis: " << fixed << setprecision(2) << Maziausias(D, n - 1) << " mm/s" << endl << "Vidutinis greitis: " << Vidutinis(D, n - 1, l) << " mm/s" << endl;

rf << "Didziausias greitis: " << fixed << setprecision(2) << Didziausias(D, n - 1) << " mm/s" << endl << "Nukeliautas kelias: " << Kelias(D, n - 1, l) << " mm" << endl;

rf.close();

return 0;

}

void Skaito (double A[], double B[], int &n, int &l)

{

ifstream df ("dalele\_data.txt");

df >> n >> l;

for (int i = 0; i < n; i++)df >> A[i] >> B[i];

df.close();

}

void Greitis(double A[], double B[], int n, int l, double D[])

{

for (int i = 0; i < n - 1; i++)D[i] = sqrt(pow(A[i] - A[i + 1], 2) + pow(B[i] - B[i + 1], 2)) / l;

}

double Maziausias(double A[], int n)

{

int ind = 0;

for (int i = 0; i < n; i++)if(A[ind] > A[i])ind = i;

return A[ind];

}

double Didziausias(double A[], int n)

{

int ind = 0;

for (int i = 0; i < n; i++)if(A[ind] < A[i])ind = i;

return A[ind];

}

double Kelias(double A[], int n, int l)

{

double s = 0;

for(int i = 0; i < n; i++)s = s + A[i] \* l;

return s;

}

double Vidutinis(double A[], int n, int l)

{

return Kelias(A, n, l)/ (l \* n);

}

#### Spalvoti taškai

1. **Užduotis**

Dvimatėje koordinačių plokštumoje yra **n** (1 ≤ n ≤ 12) trijų spalvų taškų. Jungiant vienodos spalvos taškus po tris, galima sudaryti daug skirtingų trikampių. Parašykite programą, kuri kiekvienos spalvos taškų aibėje surastų tuos tris taškus, kuriuos sujungus gautųsi lygiašonis trikampis su didžiausiu įmanomu perimetru.

Pradinių duomenų faile **taskai\_data.txt** pirmoje eilutėje užrašytas visų taškų kiekis **n**. Sekančiose **n** eilučių yra išvardinti visi taškai formatu 'SPALVA X\_KOORDINATE Y\_KOORDINATE'. Spalvų atitinkami kodai: r = raudona, m = mėlyna, z = žalia. Tiek X, tiek Y koordinatė yra sveikieji skaičiai intervale [-100;100].

Rezultatų faile **taskai\_res.txt** išspausdinkite tris eilutes: kiekvienoje eilutėje spalvą (pavyzdyje nurodyta tvarka) bei atitinkamos spalvos lygiašonio trikampio su didžiausiu perimetru perimetrą (vieno skaitmens po kablelio tikslumu). Jeigu tokio trikampio sudaryti negalima, išspausdinkite žodį 'nera' mažosiomis raidėmis. Nenaudokite lietuviškų raidžių.

|  |  |
| --- | --- |
| taskai\_data.txt | taskai\_res.txt |
| 12  r -3 -2  z -6 2  r -5 10  z 0 10  m 0 12  m 1 8  r 2 3  z 4 6  z 4 11  r 2 -3  r 6 0  z 6 2 | Raudona: 16.0  Melyna: nera  Zalia: 32.0 |

1. **Reikalavimai**

* Naudokite sveikųjų, realiųjų skaičių bei simbolių masyvus.
* Formuokite skirtingų spalvų trikampių perimetrų masyvus.
* Pradinių duomenų skaitymo funkcija **void**.
* Atstumo tarp dviejų taškų skaičiavimo funkcija, grąžinanti apskaičiuotą reikšmę per funkcijos vardą.
* Loginė funkcija, tikrinanti, ar iš 3 taškų galima sudaryti lygiašonį trikampį (nepamirškite, kad jungti galima tik vienodos spalvos taškus).
* Funkcija, skaičiuojanti trikampio perimetrą, grąžinanti apskaičiuotą reikšmę per funkcijos vardą.
* Funkcija, surandanti didžiausios reikšmės vietą realiųjų skaičių masyve (indeksą, ne pačią reikšmę).
* Rezultatų spausdinimo funkcija **void**.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int RX[], int RY[], int ZX[], int ZY[], int MX[], int MY[], int &r, int &z, int &m);

void MasP(int X[], int Y[], int &n, double Z[]);

double Atstumas(int x, int y, int x1, int y1);

bool Lygiasonis(int x, int y, int x1, int y1, int x2, int y2);

double Perimetras(int x, int y, int x1, int y1, int x2, int y2);

void Raso(double R, double Z, double M);

int Didziausias(double A[], int n);

int main(){

int RX[100], RY[100], ZX[100], ZY[100], MX[100], MY[100], r = 0, z = 0, m = 0;

double RP[100], ZP[100], MP[100];

Skaito(RX, RY, ZX, ZY, MX, MY, r, z, m);

MasP(RX, RY, r, RP);

MasP(ZX, ZY, z, ZP);

MasP(MX, MY, m, MP);

Raso(RP[Didziausias(RP, r)], ZP[Didziausias(ZP, z)], MP[Didziausias(MP, m)]);

return 0;

}

void Skaito(int RX[], int RY[], int ZX[], int ZY[], int MX[], int MY[], int &r, int &z, int &m)

{

int n, x, y;

string s;

ifstream df ("taskai\_data.txt");

df >> n;

for (int i = 0; i < n; i++){

df >> s >> x >> y;

if (s == "r"){

RX[r] = x;

RY[r] = y;

r++;

}

else if (s == "z"){

ZX[z] = x;

ZY[z] = y;

z++;

}

else{

MX[m] = x;

MY[m] = y;

m++;

}

}

df.close();

}

void MasP(int X[], int Y[], int &n, double Z[])

{

double pirm;

int ind = 0;

for (int i = 0; i < n - 2; i++){

for (int i1 = i + 1; i1 < n - 1; i1++){

for(int i2 = i1 + 1; i2 < n; i2++){

if(Lygiasonis(X[i], Y[i], X[i1], Y[i1], X[i2], Y[i2])){

Z[ind] = Perimetras(X[i], Y[i], X[i1], Y[i1], X[i2], Y[i2]);

ind++;

}

}

}

}

n = ind;

}

double Atstumas(int x, int y, int x1, int y1)

{

return sqrt(pow(x - x1, 2) + pow(y - y1, 2));

}

bool Lygiasonis(int x, int y, int x1, int y1, int x2, int y2){

if (Atstumas(x, y, x1, y1) == Atstumas(x1, y1, x2, y2) || Atstumas(x1, y1, x2, y2) == Atstumas(x2, y2, x, y) || Atstumas(x2, y2, x, y) == Atstumas(x, y, x1, y1))return true;

else return false;

}

double Perimetras(int x, int y, int x1, int y1, int x2, int y2)

{

return Atstumas(x, y, x1, y1) + Atstumas(x1, y1, x2, y2) + Atstumas(x2, y2, x, y);

}

void Raso(double R, double Z, double M)

{

ofstream rf ("taskai\_res.txt");

rf << "Raudona: ";

if (R > 0){

rf << fixed << setprecision(1) << R << endl;

}

else rf << "nera" << endl;

rf << "Melyna: ";

if (M > 0){

rf << M << endl;

}

else rf << "nera" << endl;

rf << "Zalia: ";

if (Z > 0){

rf << Z;

}

else rf << "nera" << endl;

rf.close();

}

int Didziausias(double A[], int n)

{

int ind = 0;

for (int i = 0; i < n; i++)if(A[i] > A[ind])ind = i;

return ind;

}

#### Paprasti uždaviniai

1. **Užduotis**

Trijų ekspertų komisija programavimo konkursui nori parinkti keletą paprastų uždavinių, tačiau jie nesutaria, kurie už pasiūlytų uždavinių yra paprasti. Kad pasiektų sutarimą, ekspertai nusprendžia naudoti tokią vertinimo sistemą: Kiekvienas ekspertas kiekvieną iš siūlomų uždavinių įvertina sudėtingumo balu 0 ≤ b ≤ 100 (keletas uždavinių gali būti vertinami vienodu balu). Galiausiai palyginami visų ekspertų vertinimai – paprastu laikomas toks uždavinys, kurį mažiausiu balu įvertino bent 2 ekspertai iš 3, ir nei vienas neįvertino didžiausiu balu. Pagal ekspertų vertinimus nustatykite, kurie uždaviniai yra paprasti.

Tekstinio failo **paprasti\_uzdaviniai\_data.txt** pirmoje eilutėje nurodytas siūlomų uždavinių skaičius **n**. Tolesnėse trijose eilutėse surašyti kiekvienam uždaviniui ekspertų skirti sudėtingumo balai – vienam ekspertui viena eilutė.

Rezultatus įrašykite į failą **paprasti\_uzdaviniai\_res.txt**, kuriame išvardinkite paprastų uždavinių numerius (uždaviniai numeruojami nuo 1).

Paaiškinimas: 1-ąjį uždavinį mažiausiais balais įvertino pirmasis (1) ir antrasis (5) ekspertai. 3-ąjį uždavinį mažiausiais balais įvertino pirmasis (1) ir trečiasis (8) ekspertai. 5-asis uždavinys nėra paprastas, nes jį didžiausiu balu (13) įvertino antrasis ekspertas.

|  |  |
| --- | --- |
| paprasti\_uzdaviniai\_data.txt | paprasti\_uzdaviniai\_res.txt |
| 5  1 2 1 7 1  5 9 6 8 13  9 10 8 15 8 | 1 3 |

1. **Reikalavimai**

* Naudokite vienmačius sveikųjų skaičių masyvus.
* Pradinių duomenų skaitymo funkcija **void**.
* Didžiausios reikšmės paieškos funkcija, grąžinanti reikšmę per funkcijos vardą.
* Mažiausios reikšmės paieškos funkcija, grąžinanti reikšmę per funkcijos vardą.
* Rezultatų rašymo į failą funkcija **void**.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

void Skaito(int P[], int A[], int T[], int &n);

int Maziausias(int A[], int n);

int Didziausias(int A[], int n);

void Paprasti(int P[], int A[], int T[], int n, int U[], int &ind);

void Raso(int A[], int n);

using namespace std;

int main(){

int P[100], A[100], T[100], n, U[100], ind = 0;

Skaito(P, A, T, n);

Paprasti(P, A, T, n, U, ind);

Raso(U, ind);

return 0;

}

void Skaito(int P[], int A[], int T[], int &n)

{

ifstream df ("paprasti\_uzdaviniai\_data.txt");

df >> n;

for (int i = 0; i < n; i++)df >> P[i];

for (int i = 0; i < n; i++)df >> A[i];

for (int i = 0; i < n; i++)df >> T[i];

df.close();

}

int Maziausias(int A[], int n)

{

int ind = 0;

for (int i = 0; i < n; i++)if(A[i] < A[ind])ind = i;

return A[ind];

}

int Didziausias(int A[], int n)

{

int ind = 0;

for (int i = 0; i < n; i++)if(A[i] > A[ind])ind = i;

return A[ind];

}

void Paprasti(int P[], int A[], int T[], int n, int U[], int &ind)

{

int pm = Maziausias(P, n), am = Maziausias(A, n), tm = Maziausias(T, n), pd = Didziausias(P, n), ad = Didziausias(A, n), td = Didziausias(T, n);

for(int i = 0; i < n; i++){

if(P[i] == pm && A[i] == am && T[i] != td || P[i] == pm && T[i] == tm && A[i] != ad || T[i] == tm && A[i] == am && P[i] != pd){

U[ind] = i + 1;

ind++;

}

}

}

void Raso(int A[], int n){

ofstream rf ("paprasti\_uzdaviniai\_res.txt");

for (int i = 0; i < n; i++){

rf << A[i] << " ";

}

rf.close();

}

#### Protmūšis

1. **Užduotis**

Protmūšyje dalyvauja **n** komandų. Kiekviena komanda turėjo atsakyti į **m** klausimų. Pradinių duomenų failo **protmusis\_data.txt** pirmoje failo eilutėje įrašytas komandų skaičius n ir klausimų skaičius m, likusiose n eilučių įrašytas komandos pavadinimas (jį sudaro vienas žodis) ir m sveikųjų skaičių: 0 – jei į klausimą komanda neatsakė, 1 – jei komanda į klausimą atsakė teisingai.

Parašykite programą, kuri į rezultatų failą **protmusis\_res.txt** išvestų rezultatus, kaip pateikta pavyzdyje: nurodomas komandos pavadinimas, bei į kelis klausimus komandos atsakė teisingai ir į kelis klaidingai. Toliau faile surašomas komandų, **atsakiusių į daugiau kaip pusę klausimų teisingai**, sąrašas.

Jei nėra tokių komandų, tuomet faile turi būti įrašytas žodis „nera“.

|  |  |
| --- | --- |
| protmusis\_data.txt | protmusis\_res.txt |
| 5 6  Zalieji 1 1 1 0 1 1  Melynieji 0 0 1 1 1 1  Raudonieji 1 1 1 1 0 1  Geltonieji 0 0 0 1 1 1  Baltieji 1 1 1 1 1 1 | VISOS KOMANDOS:  Zalieji 5 1  Melynieji 4 2  Raudonieji 5 1  Geltonieji 3 3  Baltieji 6 0  GERIAUSIOS KOMANDOS:  Zalieji 5 1  Melynieji 4 2  Raudonieji 5 1  Baltieji 6 0 |

1. **Reikalavimai**

* Naudojamas masyvo duomenų tipas.
* Pradinių duomenų skaitymo funkcija **void Skaitymas()**.
* Vienos komandos teisingai ir klaidingai atsakytų klausimų skaičiaus skaičiavimo funkcija **void Atsakymai()**, grąžinanti apskaičiuotas reikšmes per parametrus.
* Komandų, atsakiusių į daugiau kaip pusę klausimų teisingai, atrinkimo funkcija **void Geriausios()**. Funkcija turi suformuoti geriausių komandų masyvą.
* Rezultatų rašymo funkcija **void Rasymas()**.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(string A[], int B[], int &n, int &m);

void Atsakymai(int B[], int &m, int k, int &t, int &n);

void Geriausios(int T[], int NT[], int &n, int G[], int &ind);

void Raso(string A[], int T[], int NT[], int &n, int G[], int &ind);

int main(){

string A[100];

int B[100], n, m, t = 0, nt = 0, T[100], N[100], G[100], ind = 0;

Skaito(A, B, n, m);

for (int i = 0; i < n; i++){

Atsakymai(B, m, i, t, nt);

T[i] = t;

N[i] = nt;

t = 0;

nt = 0;

}

Geriausios(T, N, n, G, ind);

Raso(A, T, N, n, G, ind);

return 0;

}

void Skaito(string A[], int B[], int &n, int &m)

{

ifstream df ("protmusis\_data.txt");

df >> n >> m;

for(int i = 0; i < n; i++){

df >> A[i];

for(int i1 = 0; i1 < m; i1++)df >> B[i1 + i \* m];

}

df.close();

}

void Atsakymai(int B[], int &m, int k, int &t, int &n)

{

for(int i = 0; i < m; i++){

if(B[i + k \* m])t++;

else n++;

}

}

void Geriausios(int T[], int NT[], int &n, int G[], int &ind){

for (int i = 0; i < n; i++){

if(T[i] > NT[i]){

G[ind] = i;

ind++;

}

}

}

void Raso(string A[], int T[], int NT[], int &n, int G[], int &ind)

{

ofstream rf ("protmusis\_res.txt");

rf << "VISOS KOMANDOS:" << endl;

for (int i = 0; i < n; i++){

rf << A[i] << " " << T[i] << " " << NT[i] << endl;

}

rf << "GERIAUSIOS KOMANDOS:" << endl;

if (ind > 0){

for (int i = 0; i < ind; i++)rf << A[G[i]] << " " << T[G[i]] << " " << NT[G[i]] << endl;

}

else rf << "nera";

rf.close();

}

#### Sportininkai

1. **Užduotis**

Tarptautinis Olimpinis Komitetas (TOK) sportininkams rekomenduoja maisto sudėtinių dalių pasiskirstymą per parą: angliavandeniai turėtų sudaryti **a** procentų, riebalai – **r** procentų, baltymai – **b** procentų gaunamų kilokalorijų (kcal). Žinomi sportininkų duomenys: ūgis, masė ir suvartojamas energijos kiekis kcal per parą.

Parašykite programą, kuri rastų aukščiausio, žemiausio, lengviausio ir sunkiausio sportininko masę, ūgį, suvartojamos energijos kiekį kcal per parą ir kūno masės indeksą (KMI). Sportininko kūno masės indeksas (KMI), apskaičiuojamas pagal Adolphe Quetelet formulę: KMI = masė (kg) / ūgis2 (cm2).

Jeigu yra keli vienodai aukščiausi (žemiausi, sunkiausi, lengviausi) sportininkai, tai spausdinami visų duomenys.

Pradinių duomenų faile **sportininkai\_data.txt**:

* pirmoje eilutėje įrašyti trys sveikieji skaičiai **a**, **r**, **b** – rekomenduojamas angliavandenių, baltymų ir riebalų kiekis procentais;
* antroje failo eilutėje įrašytas krepšinio komandos sportininkų skaičius **n**;
* tolesnėse **n** eilučių įrašyta po du realiuosius skaičius: sportininko kūno masė (kg), ūgis (m) ir sveikasis skaičius – per parą suvartojamas energijos kiekis (kcal).

Rezultatų faile **sportininkai\_res.txt** turi būti įrašyta:

* pirmosiose n eilučių – kiekvieno sportininko eilės numeris ir kūno masės indeksas (KMI), apvalinamas iki sveikojo skaičiaus;
* tolesnėse eilutėse turi būti spausdinama:
  + aukščiausio sportininko duomenys: eilės numeris, masė, ūgis, suvartojamas energijos kiekis, KMI. Jei yra keli tokie sportininkai, tuomet turi būti spausdinami visų sportininkų duomenys;
  + žemiausio sportininko duomenys: eilės numeris, masė, ūgis, suvartojamas energijos kiekis, KMI. Jei yra keli tokie sportininkai, tuomet turi būti spausdinami visų sportininkų duomenys;
  + lengviausio sportininko duomenys: eilės numeris, masė, ūgis, suvartojamas energijos kiekis, KMI. Jei yra keli tokie sportininkai, tuomet turi būti spausdinami visų sportininkų duomenys;
  + sunkiausio sportininko duomenys: eilės numeris, masė, ūgis, suvartojamas energijos kiekis, KMI. Jei yra keli tokie sportininkai, tuomet turi būti spausdinami visų sportininkų duomenys.

|  |  |
| --- | --- |
| sportininkai\_data.txt | sportininkai\_res.txt |
| 60 25 15  12  82.00 1.85 3608  98.00 1.98 4073  92.00 1.95 3893  80.00 1.90 3541  125.00 2.08 4853  107.00 2.04 4358  107.00 2.05 4367  97.00 2.03 4094  117.00 2.11 4677  98.00 2.05 4138  93.00 1.98 3946  85.00 1.88 3650 | 1 24  2 25  3 24  4 22  5 29  6 26  7 25  8 24  9 26  10 23  11 24  12 24  9 117.00 2.11 4677 26  1 82.00 1.85 3608 24  4 80.00 1.90 3541 22  5 125.00 2.08 4853 29 |

1. **Reikalavimai**

* Naudokite sveikųjų ir realiųjų skaičių masyvus.
* Sukurkite pradinių duomenų skaitymo procedūrą (**void**).
* Sukurkite didžiausios reikšmės paieškos funkciją. Ją panaudokite ieškodami aukščiausio ir sunkiausio sportininko.
* Sukurkite mažiausios reikšmės paieškos funkciją. Ją panaudokite ieškodami žemiausio ir lengviausio sportininko.
* Sukurkite rezultatų rašymo į failą procedūrą (**void**).

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int &a, int &b, int &r, int &n, double M[], double U[], int E[]);

void Kmi(int n, double M[], double U[], double K[]);

void Maziausias(double A[], int n, int M[], int &ind);

void Didziausias(double A[], int n, int M[], int &ind);

void Raso(double K[], int &n, double M[], int MM[], int mm, int MD[], int md, int UM[], int um, int UD[], int ud, int E[], double U[]);

int main(){

int a, b, r, n, MM[100], MD[100], UM[100], UD[100], mm = 0, md = 0, ud = 0, um = 0, E[100];

double M[100], U[100], K[100];

Skaito(a, b, r, n, M, U, E);

Kmi(n, M, U, K);

Maziausias(M, n, MM, mm);

Didziausias(M, n, MD, md);

Maziausias(U, n, UM, um);

Didziausias(U, n, UD, ud);

Raso(K, n, M, MM, mm, MD, md, UM, um, UD, ud, E, U);

return 0;

}

void Skaito(int &a, int &b, int &r, int &n, double M[], double U[], int E[])

{

ifstream df ("sportininkai\_data.txt");

df >> a >> b >> r >> n;

for (int i = 0; i < n; i++)df >> M[i] >> U[i] >> E[i];

df.close();

}

void Kmi(int n, double M[], double U[], double K[])

{

for(int i = 0; i < n; i++)K[i] = M[i] / pow(U[i], 2);

}

void Maziausias(double A[], int n, int M[], int &ind)

{

int m = 0;

M[0] = 0;

ind++;

for (int i = 1; i < n; i++){

if (A[m] > A[i]){

ind = 0;

m = i;

M[ind] = i;

ind++;

}

else if(A[m] == A[i]){

M[ind] = i;

ind++;

}

}

}

void Didziausias(double A[], int n, int M[], int &ind)

{

int m = 0;

M[0] = 0;

ind++;

for (int i = 1; i < n; i++){

if (A[m] < A[i]){

ind = 0;

m = i;

M[ind] = i;

ind++;

}

else if(A[m] == A[i]){

M[ind] = i;

ind++;

}

}

}

void Raso(double K[], int &n, double M[], int MM[], int mm, int MD[], int md, int UM[], int um, int UD[], int ud, int E[], double U[])

{

ofstream rf ("sportininkai\_res.txt");

for (int i = 0; i < n; i++)rf << i + 1 << " " << fixed << setprecision(0) << K[i] << endl;

for (int i = 0; i < ud; i++)rf << fixed << setprecision(2) << UD[i] + 1 << " " << M[UD[i]] << " " << U[UD[i]] << " " << E[UD[i]] << " " << fixed << setprecision(0) << K[UD[i]] << endl;

for (int i = 0; i < um; i++)rf << fixed << setprecision(2) << UM[i] + 1 << " " << M[UM[i]] << " " << U[UM[i]] << " " << E[UM[i]] << " " << fixed << setprecision(0) << K[UM[i]] << endl;

for (int i = 0; i < mm; i++)rf << fixed << setprecision(2) << MM[i] + 1 << " " << M[MM[i]] << " " << U[MM[i]] << " " << E[MM[i]] << " " << fixed << setprecision(0) << K[MM[i]] << endl;

for (int i = 0; i < md; i++)rf << fixed << setprecision(2) << MD[i] + 1 << " " << M[MD[i]] << " " << U[MD[i]] << " " << E[MD[i]] << " " << fixed << setprecision(0) << K[MD[i]] << endl;

rf.close();

}

#### Pasirinkimas

1. **Užduotis**

Baigdami 10 klasę gimnazistai susidaro ugdymo planą. Ugdymo plane yra **n** dalykų. Yra žinoma, keli mokiniai renkasi dalyką, taip pat maksimalus mokinių skaičius to dalyko mobilioje grupėje. Pradinių duomenų failo **pasirinkimas\_data.txt** pirmoje eilutėje įrašytas dalykų skaičius **n** ir būsimųjų vienuoliktokų skaičius **m**. Tolesnėse **n** eilučių įrašyta informacija apie kiekvieną dalyką: dalyko pavadinimas (vienas žodis), mokinių, pasirinkusių dalyką, skaičius ir maksimalus mokinių skaičius to dalyko mobilioje grupėje.

Parašykite programą, kuri į rezultatų failą **pasirinkimas\_res.txt** išvestų rezultatus, kaip pateikta pavyzdyje: nurodomas dalyko pavadinimas, bei kiek mažiausiai mobilių to dalyko grupių bus sudaryta. Toliau faile surašomas dalykų, **kuriuos pasirinko mažiau negu pusė būsimųjų vienuoliktokų**, sąrašas su mobilių grupių skaičiumi.

Jei nėra tokių dalykų, tuomet faile turi būti įrašytas žodis „nera“.

|  |  |
| --- | --- |
| pasirinkimas\_data.txt | pasirinkimas\_res.txt |
| 5 60  Lietuviu 60 25  Matematika 60 27  Informatika 25 15  Anglu 45 12  Istorija 40 30 | VISI DALYKAI:  Lietuviu 3  Matematika 3  Informatika 2  Anglu 4  Istorija 2  ATRINKTI DALYKAI:  Informatika 2 |

1. **Reikalavimai**

* Pradinių duomenų skaitymo funkcija **void Skaitymas()**.
* **Vieno dalyko** mažiausio mobilių grupių kiekio skaičiavimo funkcija **int Kelios()**, grąžinanti apskaičiuotą reikšmę per funkcijos vardą.
* Dalykų, kuriuos pasirinko mažiau kaip pusė būsimųjų vienuoliktokų, atrinkimo funkcija **void Atrinkti()**. Funkcija turi suformuoti naują dalykų masyvą.
* Rezultatų rašymo funkcija **void Rasymas()**.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int &n, string D[], int K[], int P[], int &vis);

void Kiekis(int n, int K[], int P[], int G[]);

int Kelios(int kiek, int did);

void Atrinkti(int P[], int n, int I[], int &ind, int vis);

void Raso(string D[], int G[], int n, int ind, int I[]);

int main(){

int n, vis, K[100], P[100], ind = 0, I[100], G[100];

string D[100];

Skaito(n, D, K, P, vis);

for (int i = 0; i < n; i++)G[i] = Kelios(P[i], K[i]);

Atrinkti(P, n, I, ind, vis);

Raso(D, G, n, ind, I);

return 0;

}

void Skaito(int &n, string D[], int K[], int P[], int &vis)

{

ifstream df ("pasirinkimas\_data.txt");

df >> n >> vis;

for(int i = 0; i < n; i++)df >> D[i] >> P[i] >> K[i];

df.close();

}

int Kelios(int kiek, int did)

{

if (kiek % did > 0)return kiek / did + 1;

else return kiek / did;

}

void Atrinkti(int P[], int n, int I[], int &ind, int vis)

{

for (int i = 0; i < n; i++){

if(P[i] < vis / 2){

I[ind] = i;

ind++;

}

}

}

void Raso(string D[], int G[], int n, int ind, int I[])

{

ofstream rf ("pasirinkimas\_res.txt");

rf << "VISI DALYKAI:" << endl;

for(int i = 0; i < n; i++)rf << D[i] << " " << G[i] << endl;

rf << "ATRINKTI DALYKAI:" << endl;

for(int i = 0; i < ind; i++)rf << D[I[i]] << " " << G[I[i]] << endl;

if(ind == 0)rf << "nera";

rf.close();

}

# Išvados

Šis projektinis darbas sekėsi gerai. Pritaikius savo turimas programavimo žinias, mokyklinį kursą ir tinklalapio skiriamą medžiagą galėjau išspręsti dauguma užduočių. Šis darbas pagerino ne tik programavimo žinias, bet ir išmokė analizuoti, įsiskaityti į užduotis, planuoti laiką. Supratau, kaip pažvelgti į tam tikrą situaciją programuotojo akimis ir stengtis viską daryti, kuo efektyviau ir kompaktiškiau.

Susidurta ir su sunkumais. Svetainės vertinimo sistemos neveikimas bei mažas programavimo užduočių papildymas. Teko juos įveikti savaip: stengiausi pats surasti savo padarytas klaidas ir ištestuoti savo programas, papildomų užduočių ėmiau iš praeitų metų. Kiekviename darbe bei kiekvienoje užduotyje iškyla sunkumų, tačiau tas yra gerai – tas moko mus nepasiduoti, o mąstyti, kaip išspęsti iškilusį sunkumą.

Visus savo išsikeltus uždavinius įvykdžiau ir tikslą pasiekiau.

# Literatūra

<https://konkursai.if.ktu.lt/index.php/teorija-3/179-salyginis-sakinys-4p-cpp> [žiūrėta spalio mėn.]

<https://konkursai.if.ktu.lt/index.php/teorija-3/330-9-funkcija-su-parametrais-nuorodomis> [žiūrėta lapkričio mėn.]

<https://konkursai.if.ktu.lt/index.php/teorija-3/171-pazintis-su-masyvu-4p-cpp-2l-teorija> [žiūrėta sausio mėn.]

<https://konkursai.if.ktu.lt/index.php/teorija-3/183-paieska-masyve-4p-cpp-teor> [žiūrėta vasario mėn.]

<https://konkursai.if.ktu.lt/index.php/pamokeli-aprasymai> [naudotasi visus metus]

<https://konkursai.if.ktu.lt/index.php/pamokeliu-uzduotys-2> [naudotasi visus metus]

# Priedai

**RECENZIJA**

**2020-05-19**

**Pasvalys**

**Projekto pavadinimas:** KTU „Programavimo pamokėlių“ užduočių sprendimas

**Projekto autorius:** Kristupas Jakubonis, 3d klasės mokinys

**Recenzentas**: Dalia Monkevičienė, IT mokytoja metodininkė

**Projektinio darbo tema ir problemos aktualumas:**

Pasirinkta projektinio darbo tema labai aktuali, nes šiuo metu jaučiams IT specialistų poreikis. Skelbiama, kad ateityje prireiks keliolika tūkstančių IT specialistų, o norint juo tapti, reikia baigti atitinkamas studijas. Tam reikia įgyti programavimo pagrindus, išmokti juos taikyti uždavinių sprendimui.

**Projektinio darbo tikslai, problemų sprendimas:**

Pagrindinis šio darbo tikslas – išspręsti kuo daugiau pateiktų programavimo užduočių ir surinkti kuo daugiau taškų. Tuo pačiu įgyti daugiau žinių, bei įgūdžių pritaikant žinomus algoritmus uždavinių sprendimui, atrasti naujų sprendimo būdų.

**Darbo rezultatų ir išvadų apibūdinimas:**

Užduotys pateikiamos 3 sudėtingumo lygių. Pirmas lygis skirtas pradedantiesiems programuotojams, 2 lygis ­– susipažinusiems su programavimo pradmenimis, 3 lygis skirtas pažengusiems programuotojams. Mokinys sprendė 2 lygio užduotis ir sudėtingesnes 1 lygio užduotis. Metų eigoje buvo pateiktos 4 programavimo pamokėlių užduotys. Kiekviena pamokėlė buvo skirta tam tikrai temai išmokti. Dalį uždavinių mokinys ėmė iš ankstesnių metų pamokėlių. Mokinys išsprendė 41 užduotį. Tai tikrai didelis skaičius, ypač kai dalis temų dar nesimokyta mokykloje, reikėjo teoriją, reikalingą užduočių sprendimui išmokti savarankiškai ir vienai užduočiai išspręsti bei aprašyti kartais reikia ir kelių valandų. Be to darbą apsunkino tai, kad neveikė užduočių testavimo sistema ir mokiniui pačiam teko sudarinėti testus programoms tikrinti.

**Projektinio darbo privalumai ir trūkumai:**

Darbo privalumas tas, kad mokinys išsprendė tikrai daug uždavinių ir įgijo programavimo patirties.

Mokinio atliktą darbą vertinu puikiai.

Recenzentas Dalia Monkevičienė

(parašas)

**Projektinio darbo vertinimo lentelė**

Mokinio pavardė, vardas Jakubonis Kristupas

Klasė 3d

Projektinio darbo pavadinimas KTU „Programavimo pamokėlių“ užduočių sprendimas

Darbo konsultantas Dalia Monkevičienė

| **Vertintojas**  **ir skiriamų taškų maksimumas** | **Vertinimo kriterijai** | **Darbo  konsultanto įver­tinimas** | **Recen­zento įvertini­mas** | **Komisijos**  **įvertini­mas** |
| --- | --- | --- | --- | --- |
| **Darbo konsultantas**  E-dienyne įskaita | Kaupiamasis vertinimas **spalio–sausio mėn**.  Darbo idėja, idėjos įgyvendinimo nuoseklumas, kruopšti informacijos paieška, kūrybiniai bandymai, aptarimai, diskusijos, darbo formų šaltinių įvairovė, darbo tikslų ir uždavinių kėlimas, savarankiškumas ir kt. | Įsk. | X | X |
| **Darbo**  **konsultantas**  (40 taškų) | **Baigiamojo darbo aprašas ( kūrybinė–praktinė dalis)**  Projektinio darbo tema ir problemos aktualumas.  Projektinio darbo tikslai, problemų sprendimas.  Darbo rezultatų ir išvadų apibūdinimas.  Projektinio darbo privalumai ir trūkumai.  Aprašo struktūros atitikimas.  Akademinis sąžiningumas.  Įgyta(-os) arba labiausiai patobulinta(-os) 1–2 kompetencijos:  1. mokėjimo planuoti darbą kompetencija;  2. komunikavimo kompetencija;  3. pažinimo kompetencija;  4. socialinė kompetencija;  5. iniciatyvumo ir kūrybingumo kompetencija. | 40 | X | X |
| **Recenzentas, jeigu konsultantas nedirba gimnazijoje**  (40 taškų) | Projektinio darbo tema ir problemos aktualumas.  Projektinio darbo tikslai, problemų sprendimas.  Darbo rezultatų ir išvadų apibūdinimas.  Projektinio darbo privalumai ir trūkumai.  Aprašo struktūros atitikimas. | X |  | X |
| **Darbo vertinimo komisija**  (40 taškų) | **Darbo pristatymas:**  Praktinio darbo pritaikymo galimybės.  Pristatymo glaustumas (iki 5 min.).  Gebėjimas atsakyti į klausimus.  Vaizdinės informacijos kokybė ir informatyvumas.  Darbo struktūra.  Estetinis darbo aprašo apipavidalinimas. | X | X |  |
| **IT mokytojas**  (10 taškų) | Kompiuterinis raštingumas. Kriterijai:   * titulinis lapas, * turinys, * vieningas maketavimo stilius, * raštvedybos taisyklių laikymasis, * iliustracijų teisingas naudojimas bei numeravimas. | 10 | | |
| **Lietuvių kalbos mokytojas**  (10 taškų) | Lietuvių kalbos raštingumas  Kriterijai. Vertinamas 500 žodžių tekstas. Žymimos šios klai­dos: rašybos, skyrybos, gramatikos ir leksikos, stiliaus.  Vertinimo normos:  10 balų – 0–2 klaidos, 9 balai – 3-4 klaidos,  8 balai – 5-6 klaidos, 7 balai – 7-8 klaidos,  6 balai – 9-10 klaidų, 5 balai – 11-14 klaidų,  4 balai – 15-19 klaidų, 3 balai – 20-24 klaidos,  2 balai – 25-30 klaidų, 1 balas – 30-35 klaidos,  0 balų – daugiau nei 35 klaidos. |  | | |
| I pusmečio įvertinimas neįsk. (-10 taškų) | | Įskaityta | | |
| Darbo konsultanto įvertinimas | | 40 | | |
| Recenzento įvertinimas | |  | | |
| Kompiuterinio raštingumo įvertinimas | | 10 | | |
| Lietuvių kalbos raštingumo įvertinimas | |  | | |
| Komisijos įvertinimas | |  | | |
| Iš viso taškų ir gautas pažymys | |  | | |

Darbo konsultantas Dalia Monkevičienė

Recenzentas

Informacinių technologijų mokytojas Dalia Monkevičienė

Lietuvių kalbos mokytojas

Komisijos pirmininkas

**Taškų ir pažymių atitikimo lentelė**

|  |  |  |  |
| --- | --- | --- | --- |
| **Taškai** | **Pažymys** | **Taškai** | **Pažymys** |
| 100–95 | 10 | 60–51 | 6 |
| 94–85 | 9 | 50–41 | 5 |
| 85–71 | 8 | 40–31 | 4 |
| 70–61 | 7 | 30–0 | „neatsiskaitė“ |