My Project

Generated by Doxygen 1.12.0

1 readme 1

2 Hierarchical Index 7

2.1 Class Hierarchy . . . 7

3 Class Index 9

3.1 Class List . . . 9

4 File Index 11

4.1 File List . . . 11

5 Class Documentation 13

5.1 Stud Class Reference . . . 13

5.1.1 Detailed Description . . . 14

5.1.2 Constructor & Destructor Documentation . . . 14

5.1.2.1 Stud() [1/2] . . . 14

5.1.2.2 Stud() [2/2] . . . 14

5.1.2.3 ~Stud() . . . 15

5.1.3 Member Function Documentation . . . 15

5.1.3.1 addND() . . . 15

5.1.3.2 apskaiciuotiGalutinius() . . . 15

5.1.3.3 atvaizduoti() . . . 16

5.1.3.4 clearData() . . . 16

5.1.3.5 getEgz() . . . 16

5.1.3.6 getGalutinisMed() . . . 16

5.1.3.7 getGalutinisVid() . . . 17

5.1.3.8 getND() . . . 17

5.1.3.9 operator=() . . . 17

5.1.3.10 setEgz() . . . 17

5.1.3.11 setND() . . . 18

5.1.4 Friends And Related Symbol Documentation . . . 18

5.1.4.1 operator<< . . . 18

5.1.4.2 operator>> . . . 18

5.2 zmogus Class Reference . . . 20

5.2.1 Detailed Description . . . 21

5.2.2 Constructor & Destructor Documentation . . . 21

5.2.2.1 zmogus() . . . 21

5.2.2.2 ~zmogus() . . . 21

5.2.3 Member Function Documentation . . . 22

5.2.3.1 atvaizduoti() . . . 22

5.2.3.2 getPavarde() . . . 23

5.2.3.3 getVardas() . . . 23

5.2.3.4 setPavarde() . . . 23

5.2.3.5 setVardas() . . . 23

5.2.4 Friends And Related Symbol Documentation . . . 24
5.2.4.1 operator<< . . . 24

6 File Documentation **25**

6.1 FailoGeneravimas.cpp File Reference . . . 25
6.1.1 Detailed Description . . . 25
6.1.2 Function Documentation . . . 25
6.1.2.1 sugeneruotiStudentoFaila() . . . 25
6.2 FailoGeneravimas.cpp . . . 26
6.3 FailoNuskaitymas.cpp File Reference . . . 27
6.3.1 Detailed Description . . . 27
6.3.2 Function Documentation . . . 28
6.3.2.1 nuskaitytiIsFailo() . . . 28
6.3.2.2 nuskaitytiIsfailo() . . . 29
6.4 FailoNuskaitymas.cpp . . . 29
6.5 Mylib.h File Reference . . . 31
6.5.1 Detailed Description . . . 31
6.6 Mylib.h . . . 31
6.7 readme.md File Reference . . . 32
6.8 Stud.cpp File Reference . . . 32
6.8.1 Detailed Description . . . 32
6.8.2 Function Documentation . . . 32
6.8.2.1 operator<<() . . . 32
6.8.2.2 operator>>() . . . 33
6.8.2.3 val() . . . 34
6.9 Stud.cpp . . . 35
6.10 Stud.h File Reference . . . 36
6.10.1 Detailed Description . . . 37
6.10.2 Function Documentation . . . 37
6.10.2.1 irasytikietiakiaiList() . . . 37
6.10.2.2 irasytiKietiakiaiVector() . . . 38
6.10.2.3 irasytivargsiukusList() . . . 38
6.10.2.4 irasytiVargsiukusVector() . . . 38
6.10.2.5 nuskaitytiIsFailo() . . . 38
6.10.2.6 nuskaitytiIsfailo() . . . 39
6.10.2.7 sugeneruotiStudentoFaila() . . . 40
6.11 Stud.h . . . 40
6.12 Studentu_duomenys.cpp File Reference . . . 42
6.12.1 Detailed Description . . . 42
6.12.2 Function Documentation . . . 43
6.12.2.1 apskaiciuotiMediana() . . . 43
6.12.2.2 irasytikietiakiaiList() . . . 43

6.12.2.3 irasytiKietiakiaiVector() 44
6.12.2.4 irasytivargsiukusList() 44
6.12.2.5 irasytiVargsiukusVector() 45
6.12.2.6 main() 45
6.13 Studentu_duomenys.cpp 50

Index **57**

Chapter 1

readme

#Studentų galutinio balo apskaičiavimo programa. **(v2.0 versija)**

#Ši programa skirta apskaičiuoti galutiniams balams, įvedant arba nuskaitant iš failo studento vardą, pavardę, namų darbų rezultatus bei egzamino balą.

#Galutinis balas skaičiuojamas tokia formule: **Galutinis = 0.4** $*$ **vidurkis + 0.6** $*$ **egzaminas** (Kai reikia galutinio balo medianos pavidalu tai tiesiog vietoj vidurkio įstatoma mediana)

$**$#Norint naudotis programa, reikia atlikti šiuos veiksmus:$**$

- Pasirinkti, ar norite sugeneruoti failus(taip/ne).

Jei pasirinksite, kad norite sugeneruoti, tuomet failai bus sukurti ir išvedime bus rodomas failų kūrimo laikas.

- Atsakyti programai, ar norite įvesti studentų duomenis ar nuskaityti juos iš failo(ivesti/nuskaityti).
- Pasirinkti dalijimo į dvi kategorijas strategiją (1 - pirmoji, 2 - antroji, 3 - trečioji).
- Pasirinkti norimo naudoti konteinerio tipą (1 - vector, 2 - list).
- Pasirinkti rūšiavimo kritetijų (1 - pagal vardą, 2 - pagal pavardę, 3 - pagal galutinį balą).

Jei pasirenkate nuskaityti, tai programa tiesiogiai nuskaitys failą, naudodama pasirinktą konteinerio tipą, surušiuos studentus pagal galutinį balą(pagal vidurkį) į dvi grupes:Vargšiukai(galutinis balas $<$ 5) ir Kietiakiai(galutinis balas $>$= 5), surušiuos pagal pasirinktą kriterijų ir išves į du naujus failus.

Jei pasirenkate įvesti, tuomet toliau reikes atlikti šiuos veiksmus:

- Įvesti studentų skaičių.
- Pasirinkti norimą naudoti konteinerį.
- Įvesti studento vardą ir pavardę.
- Pasirinkti ar namų darbų ir egzamino rezultatus reikia generuoti atsitiktinai(taip/ne).
- Atsakyti programai, ar žinai koks yra namų darbų skaičius(taip/ne).
- Įvesti namų darbų skaičių.
- Įvesti namų darbų visus rezultatus(10-balėje sistemoje).

- Galiausiai įvesti egzamino balą. Išvedime prie studento duomenų matysite ir objekto saugojimo atmintyje adresą.

∗∗#Realisai:∗∗

∗∗#1 ir 2 releasai(v.pradinė ir v0.1)∗∗ - realizuoja programa pagal aprašytus užduoties reikalavimus nuskaito vartotojų įvedamus reikiamus duomenis ir pateikia studentu duomenis. Tyrimai ir rezultatai:

Laiko efektyvumas: programa greitai apdoroja nuskaitytus studentų duomenis, tačiau kai yra didesnis studentų skaičius, pastebimas ilgesnis laukimo laikas, kol programa pateikia rezultatus. Jei buvo pasirinkta įvesti duomenis, tuomet programoje įvedant mažą studentų skaičių(tarkim du), ji apdoroja įvestus studentų duomenis gana greitai. Tačiau kai yra didesnis studentų skaičius(tarkim dešimt), įvedimas užtrunka žymiai ilgiau.

Atminties efektyvumas: programoje naudojama struktūra(std::vector), kuri leidžia efektyviai saugoti ir tvarkyti studentų namų darbų rezultatus. Užtikrinama, kad programa galėtų veikti su dideliu studentų skaičiumi.

Vartotojo sąsajos paprastumas: programoje yra leidžiama lengvai įvesti duomenis ir gauti rezultatus. Aiškiai nurodyti visi privalomi įvedimai ir rezultatas gaunamas greitai.

#Rezultatas - Failai nuskaitomi teisingai ir programa išveda studentų vardus, pavardes ir galutinį balą(medianos ir vidurkio pavidalu).

∗∗#3 releasas(v0.2)∗∗ - Programa patobulinta, kad generuotu failus, surušiuotu nuskaitytus duomenis ir įrašytų į atskirus failus. Efektyvumo tyrimai ir rezultatai:

Laiko efektyvumas: Programa greitai apdoroja nuskaitytus studentų duomenis, tačiau kai yra didesnis studentų skaičius, pastebimas ilgesnis laukimo laikas, kol programa pateikia rezultatus. Galima pamatyti,kad didėjant failo dydžiui, apdorojimo laikas ilgėja, ypač nuskaitymo ir rūšiavimo etapuose. Rūšiavimo laikas augo dramatiškai nuo 0.↩
01251s(1000 įrašų) iki 544.98148s(10000000 įrašų), o dalijimo laikas taip pat didėjo, bet išlieka gerokai greitesnis už rūšiavimo laiką. Bendras testo laikas nuosekliai didėja, atspindėdamas procesų sudėtingumą. Kai buvo pasirinkta įvesti duomenis, tuomet programoje įvedant mažą studentų skaičių(tarkim du), ji apdoroja įvestus studentų duomenis gana greitai. Tačiau kai yra didesnis studentų skaičius(tarkim dešimt), įvedimas užtrunka žymiai ilgiau. Failų kūrimo efektyvumas mažėja didėjant duomenų kiekiui. Pastaba. Nors kiekvieno testavimo metu rezultatai gali nežymiai skirtis dėl atsitiktinių veiksnių, bendros laiko tendencijos išlieka tos pačios.

Atminties efektyvumas: programoje naudojama struktūra(std::vector), kuri leidžia efektyviai saugoti ir tvarkyti studentų namų darbų rezultatus. Užtikrinama, kad programa galėtų veikti su dideliu studentų skaičiumi.

Vartotojo sąsajos paprastumas: programoje yra leidžiama lengvai įvesti duomenis ir gauti rezultatus. Aiškiai nurodyti visi privalomi įvedimai ir rezultatas gaunamas greitai.

#Rezultatas - Iš įvesties studentų duomenys nuskaitomi teisingai ir programa išveda studentų vardus, pavardes ir galutinį balą(medianos ir vidurkio pavidalu). Taip pat kai nuskaitomas failas, studentai surušiuojami į dvi grupes ir išvedami į naujus failus. Išvedime rodoma programos veikimo greičio analizė.

∗∗#4 releasas(v0.3)∗∗ - Ismatuojama patobulintos v0.2 realizacijos veikimo spartą priklausomai nuo naudojamo vieno iš dvejų konteinerių(vector ir list).

Konteinerių testavimas. Buvo atlikta po 5 bandymus su kiekvieno dydžio failu. Matavimas sekundėmis.

Tyrimo rezultatai rodo, kad naudojant sąrašo tipo konteinerį, nuskaitymo, rūšiavimo bei dalijimo į dvi grupes laikai yra žymiai mažesni nei su vektoriaus konteineriu. Įrašymo laikai nesiskiria, o bendra testo trukmė yra gerokai mažesnė su sąrašo konteineriu, kas pabrėžia šio konteinerio efektyvumą. Taip pat pastebėta, kad didėjant duomenų kiekiui, skirtumas tarp laikų rezultatų dar labiau išryškėja.

∗∗#5 releasas(v1.0)∗∗ - Optimizuota studentų rūsiavimo (dalijimo) i dvi kategorijas ("vargšiukų" ir "kietiakų") realizacija (v0.3).

Buvo atlikti studentų dalijimo į dvi kategorijas strategijų bandymai

1 strategija: Bendro studentai konteinerio (vector ir list tipų) skaidymas (rūšiavimas) į du naujus to paties tipo konteinerius: "vargšiukų" ir "kietiakų". Tokiu būdu tas pats studentas yra dvejuose konteineriuose: bendrame studentai ir viename iš suskaidytų (vargšiukai arba kietiakai). 2 strategija: Bendro studentų konteinerio (vector ir list) skaidymas (rūšiavimas) panaudojant tik vieną naują konteinerį: "vargšiukai". 3 strategija: Bendro studentų konteinerio (vector ir list) skaidymas (rūšiavimas) panaudojant greičiau veikiančia 2 strategiją įtraukiant į ją "efektyvius" darbo su konteineriais metodus. Šiame tyrime pritaikyti tinkami algoritmai studentų dalijimo procedūrai paspartinti (optimizuoti) ant vieno fiksuoto konteinerio - vektoriaus.

Išvadele:Palyginus pagal 1 ir 2 strategias gautus vidutinius dalijimo laikus esant tam tikram duomenų kiekiui, pastebėta, kad dirbant su 2 strategiją, programos veikimo sparta atsižvelgiant į daljimo procesą yra greitesnė nei kai naudojama 1 strategija.

Rezultatas: Atlikus tyrimą naudojant 3 strategiją, pastebėta, kad po kiekvieno algortimo panaudojimo,programos veikimo sparta dalijimo atžvilgiu greičiausia buvo naudojant algortimą std::partition.

∗∗#6 realisas(v1.1)∗∗ - Perėjimas iš struktūros į klasę. #Palyginamos abiejų programų(versija v1.0 su struktūromis ir v1.1 su klasėmis) veikimo sparta.

#Naudojamas vienas fiksuotas konteineris - vektorius ir pati greičiausia dalijimo strategija - trečioji(su std::partition algoritmu) bei 100000 ir 1000000 dydžio failai. Rezultatas: Galima pastebėti, kad naudojant struct tipo duomenis, programos veikimo sparta yra žymiai greitesnė nei naudojant class tipo duomenis. Tai rodo, kad struct yra efektyvesnis tiek mažesniuose, tiek didesniuose duomenų kiekiuose.

#Toliau atlikita eksperimentinė analizė priklausomai nuo kompiliatoriaus optimizavimo lygio, nurodomo per flag'us: O1, O2, O3. #Paaiškinimai:

- **Optimizavimo lygiai:**:
  - O1 - Pagrindinė optimizacija, kurios tikslas pagerinti programos našumą, nepadidinus jos dydžio per daug.
  - O2 - Aukštesnis optimizavimo lygis, kuris bando pasiekti dar geresnį našumą.
  - O3 - Maksimalus optimizavimas, kuris žymiai pagerina našumą, bet taip pat gali padidinti '.exe' failo dydį.
- **Veikimo laiko matavimas:** Laiko matavimai buvo atlikti su 100000 ir 1000000 įrašų failais.
- **'.exe' failo dydis:∗∗Failo dydžiai priklauso nuo optimizavimo lygio.**

  #Rezultatas: Naudojant struct ir class tipus su skirtingais optimizavimo lygiais, matome, kad optimizavimo lygiai turi teigiamą poveikį veikimo laikui. Tačiau patys skirtumai tarp optimizavimo lygių(O1, O2, O3) yra maži ir perėjimas nuo vieno lygio į kitą neturi daug įtakos veikimo laikui. Optimizavimo lygiai turi įtakos ir .exe failo dydžiui. Su struct tipo duomenimis failo dydis pasikeičia nuo 69.5KB(O1) iki 81KB(O3), o su class - failo dydis didėja nuo 77.5KB(O1) iki 94.5KB(O3).

  #Išvada: Lyginant su rezultatais is aukščiau nurodytos lentelės, kur nebuvo tikrinami optimizavimo lygiai, matome, kad panaudoju optimizavimo lygius, veikimo laikas sumažėjo, programa pradėjo veikti greičiau.Išvada: Lyginant su rezultatais is aukščiau nurodytos lentelės, kur nebuvo tikrinami optimizavimo lygiai, matome, kad panaudoju optimizavimo lygius, veikimo laikas sumažėjo, programa pradėjo veikti greičiau.

  ∗∗#7 realisas(v1.2)∗∗ - Realizuoti visi reikiami "Rule of three" ir įvesties/išvesties operatoriai turimai Studentas klasei.

  #"Rule of three" operatorių realizavimas Studentas klasei.

  -Destruktorius yra skirtas atlaisvinti dinaminę atmintį, kuri buvo priskirta objektui per jo gyvavimo laikotarpį. Jis automatiškai kviečiamas, kai objektas išeina iš veikimo srities, užtikrindamas, kad nebūtų atminties nutekėjimų.

  -Kopijavimo konstruktorius yra naudojamas kuriant naują objektą, kuris tampa esamo objekto kopija. Tinkama kopija užtikrina, kad kiekvienas objektas turi savo atskirą duomenų kopiją.

-Kopijavimo priskirimo operatorius leidžia priskirti vieną objektą kitam. Svarbu įsitikinti, kad priskyrimas į save yra tinkamai valdomas ir kad seni objekto ištekliai yra tinkamai atlaisvinami prieš priskiriant naujas reikšmes.

#Perdengti įvesties ir išvesties metodai darbui su Studentų klasę.

-Operatoriai deklaruojami klasės viduje kaip draugiškos funkcijos, kad turėtų tiesioginę prieigą prie klasės privačių ir apsaugotų narių. Jie leidžia įvesti duomenis į Stud objekto kintamuosius naudojant std::istream, bei išvesti Stud objekto duomenis į std::ostream.

-Įvesties operatorius realizutoas stud.cpp faile.

-Išvesties operatorius realizutoas stud.cpp faile.

#Perdengtų metodų veikimas.

#Duomenų įvestis:

Rankiniu būdu - Vartotojas gali įvesti duomenis klaviatūra, kai programa naudoja std::cin srautą. Naudojant operatorių operator>>, programa leidžia įvesti vardą, pavardę, namų darbų rezultatus, egzamino balą.

Automatiniu - Duomenys gali būti įvedami automatiškai pagal tam tikrą procesą(įvesti iš anksto paruoštus duomenis)

Iš failo - Duomenys gali būti nuskaityti iš failo naudojant std::ifstream srautą. Perdengtas operator>> metodas leidžia nuskaityti duomenis iš failo, kurio turinys turi būti struktūrizuotas pagal tam tikrus reikalavimus ir užpildyti Stud objekto laukus.

#Duomenų išvestis

Į ekraną - Duomenys atspausdinami ekrane naudojant std::cout. Su perdengtu operatoriumi operator<< studento informacija bus gražiai suformuluota ir pateikta vartotojui tiesiogiai ekrane.

Į failą - Duomenys įrašomi į failą naudojant std::ofstream srautą. Operatorius operator<< užtikrina, kad studento duomenys būtų įrašyti į failą, kurį vėliau vartotojas gali peržiūrėti.

Išvadėlė: Perdengti metodai (operator>> ir operator<<) leidžia paprastai atlikti duomenų įvedimą ir išvedimą tiek interaktyviai su vartotoju, tiek automatiškai. Operator>> leidžia nuskaityti duomenis (rankiniu būdu, automatiniu būdu arba iš failo), o operator<< padeda išvesti duomenis ekrane arba į failą.

Išvadėlė: "Rule of three" taisyklė užtikrina, kad klasės objektai, kurie dirba su dinaminiais ištekliais, būtų teisingai kopijuojami, priskiriami ir sunaikinami, taip išvengiant atminties nutekėjimo ir kitų valdymo klaidų.

#Rezultatas - Iš įvesties studentų duomenys nuskaitomi teisingai ir programa išveda studentų vardus, pavardes ir galutinį balą(medianos ir vidurkio pavidalu). Taip pat kai nuskaitomas failas, studentai suruošiuojami į dvi grupes ir išvedami į naujus failus. Išvedime rodoma programos veikimo greičio analizė.

#8 realisas(v1.5) - Vietoje turimos Studentas klasės sukurtos dvi: bazinė (abstrakti) klasė, skirta bendrai aprašyti žmogų ir tuomet iš jos išvestinė (derived) klasė - Studentas.

Abstrakčios klasės Žmogus realizavimas. Rezultatas: Turi bendrą informaciją apie žmogų (vardas, pavardė). Ji negali būti naudojama tiesiogiai objektų kūrimui. Vietoj to, ji veikia kaip bazinė klasė, kuria remiasi kitos išvestinės klasės. Mūsų atvėju - Studentas. virtual ∼zmogus() - destruktorius yra virtualus, nes klasė paveldima. virtual void atvaizduoti() - tai grynai virtuali funkcija, todėl kiekviena išvestinė klasė privalo ją įgyvendinti. Tai leidžia naudoti polimorfizmą, nes funkcijos implementacija priklauso nuo objekto tipo.

Abstrakčios klasės Žmogus objektų kūrimas negalimas. Rezultatas: Metama klaida rodo, kad bandoma sukurti abstrakčios klasės zmogus objektą, o tai negalima, nes ji turi grynai virtualią funkciją.

Studento klasė išvestinė (derived) iš Žmogaus, kuri palaiko 1.2 versijoje realizuotą trejų metodų taisyklę. Rezultatas: Klasė Studentas paveldi bazinę klasę žmogus ir įgyvendina jos metodus. Ji prideda studento specifinius duomenis. Ši klasė privalo įgyvendinti visus grynai virtualius metodus iš bazinės klasės. Taip pat ji įgyvendina metodą atvaizduoti(), kad galėtų parodyti visą informaciją apie studentą.

Išvadėlė: Žmogus klasė suteikia bendrą šabloną, o Studentas prideda konkrečius duomenis ir funkcionalumą.

#Rezultatas - Iš įvesties studentų duomenys nuskaitomi teisingai ir programa išveda studentų vardus, pavardes ir galutinį balą(medianos ir vidurkio pavidalu). Taip pat kai nuskaitomas failas, studentai suruošiuojami į dvi grupes ir išvedami į naujus failus. Išvedime rodoma programos veikimo greičio analizė.

##Visų iki šios v2.0 versijos atliktų releasu apibendrinimas:

- 1 ir 2 releasai(v.pradinė ir v0.1) realizuoja programa pagal aprašytus užduoties reikalavimus nuskaito vartotojų įvedamus reikiamus duomenis ir pateikia studentu duomenis.

- 3 releasas(v0.2) - Programa patobulinta, kad generuotu failus, surusiuotu nuskaitytus duomenis ir įrašytų į atskirus failus.
- 4 releasas(v0.3) - Ismatuojama patobulintos v0.2 realizacijos veikimo spartą priklausomai nuo naudojamo vieno iš dvejų konteinerių(vector ir list)
- 5 releasas(v1.0) - Optimizuota studentų rūsiavimo (dalijimo) i dvi kategorijas ("vargšiukų" ir "kietiakų") realizacija (v0.3)
- 6 realisas(v1.1) - Perėjimas iš struktūros į klasę.
- 7 realisas(v1.2) - Realizuoti visi reikiami "Rule of three" ir įvesties/išvesties operatoriai turimai Studentas klasei.
- 8 realisas(v1.5) - Vietoje turimos Studentas klasės sukurtos dvi: bazinė (abstrakti) klasė, skirta bendrai aprašyti žmogų ir tuomet iš jos išvestinė (derived) klasė - Studentas.

##Diegimas

Šis projektas naudoja **CMake** kaip projektų valdymo įrankį ir gali būti sukurtas naudojant bet kurią operacinę sistemą, kuri palaiko CMake (Windows).

#CMakeLists.txt sukūrimas:

cmake_minimum_required(Version 3.10)

project(Studentu_duomenys)

set(CMAKE_CXX_STANDART 17) set(CMAKE_CXX_STANDART_REQUIRED ON)

set(SOURCES Studentu_duomenys.cpp Stud.cpp FailoGeneravimas.cpp FailoNuskaitymas.cpp Mylib.h Stud.h)

add_executable(studentai_program ${SOURCES})

##Sukūrimas naudojant cmake:

1. Atidarykite Developer Command Prompt for Visual Studio
2. Eikite į projekto katalogą: cd C:\path\to\Studentu_duomenys
3. Sukurkite build katalogą: mkdir build cd build
4. Paleiskite CMake, kad sugeneruotumėte Visual Studio projektą: cmake ..
5. Paleiskite sukūrimo procesą: cmake –build .
6. Po sėkmingo sukūrimo, galite rasti vykdomąjį failą studentai_program.exe Release kataloge

#Naudotos bibliotekos:

- <iostream>
- <iomanip>
- <string>
- <vector>
- <algorithm>
- <random>
- <fstream>
- <sstream>
- <chrono>
- <list>

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

zmogus 20
Stud 13

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Stud
Klasė, kuri aprašo studentą, paveldintį iš žmogaus 13
zmogus
Klasė, kuri aprašo žmogų su vardu ir pavarde 20

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

FailoGeneravimas.cpp
    Studentų duomenų generavimas į failą 25
FailoNuskaitymas.cpp
    Studentų duomenų nuskaitymas iš failo 27
Mylib.h
    Ši biblioteka apima pagrindines C++ standartines bibliotekas ir prideda aliasus 31
Stud.cpp
    Studentų objektų realizacija 32
Stud.h
    Studentų klasės antraštinis failas 36
Studentu_duomenys.cpp
    Pagrindinis programos vykdymo failas 42

Chapter 5

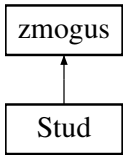
Class Documentation

5.1 Stud Class Reference

Klasė, kuri aprašo studentą, paveldintį iš žmogaus.

`#include <Stud.h>`

Inheritance diagram for Stud:



Public Member Functions

- Stud (std::string v="", std::string p="", std::vector< int > nd={}, double e=0.0)
  Konstruktorius, kuriame nustatomi studento vardas, pavardė, namų darbai ir egzaminas.
- Stud (const Stud &other) noexcept
  Kopijavimo konstruktorius.
- Stud & operator= (const Stud &other) noexcept
  Kopijavimo priskyrimo operatorius.
- void addND (int nd)
  Pridėti naują namų darbų rezultatą.
- ~Stud ()
  Destruktorius, kuris išvalo duomenis.
- void clearData ()
  Išvalo studento duomenis.
- std::vector< int > getND () const
  Grąžina namų darbų rezultatus.
- double getEgz () const
  Grąžina egzamino rezultatą.
- double getGalutinisVid () const
  Grąžina galutinį įvertinimą (vidurkį).
- double getGalutinisMed () const
  Grąžina galutinį įvertinimą (medianą).
- void setND (const std::vector< int > &nd)
  Nustato namų darbų rezultatus.
- void setEgz (double e)
  Nustato egzamino rezultatą.
- void apskaiciuotiGalutinius ()
  Apskaičiuoja galutinius įvertinimus (vidurkį ir medianą).
- void atvaizduoti (std::ostream &os) const
  Atvaizduoja studento duomenis.

Public Member Functions inherited from zmogus

- zmogus (std::string v="", std::string p="")
  Konstruktorius, kuriame nustatomi vardas ir pavardė.
- virtual ~zmogus ()=default
  Virtualus destruktorius (default).
- std::string getVardas () const
  Grąžina žmogaus vardą.
- std::string getPavarde () const
  Grąžina žmogaus pavardę.
- virtual void setVardas (const std::string &v)
  Nustato žmogaus vardą.
- virtual void setPavarde (const std::string &p)
  Nustato žmogaus pavardę.

Friends

- std::istream & operator>> (std::istream &is, Stud &stud)
  Draugiškas operatorius, kuris įveda studentą iš srauto.
- std::ostream & operator<< (std::ostream &os, const Stud &stud)
  Draugiškas operatorius, kuris išveda studentą į srautą.

5.1.1 Detailed Description

Klasė, kuri aprašo studentą, paveldintį iš žmogaus.

Ši klasė praplečia klasės `zmogus` funkcionalumą, kad apimtų studento pažymius, egzaminą ir galutinį įvertinimą.

Definition at line 108 of file Stud.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Stud() [1/2]

```
Stud::Stud (
            std::string v = "",
            std::string p = "",
            std::vector< int > nd = {},
            double e = 0.0)  [inline]
```

Konstruktorius, kuriame nustatomi studento vardas, pavardė, namų darbai ir egzaminas.

Parameters

v	Vardas.
p	Pavardė.
nd	Namų darbų rezultatai.
e	Egzamino rezultatas.

Definition at line 123 of file Stud.h.

```
00123                                                                {}, double e = 0.0)//
Konstruktorius su parametrais
00124         : zmogus(v, p), ND(nd), egz(e), GalutinisVid(0), GalutinisMed(0) {}
```

5.1.2.2 Stud() [2/2]

```
Stud::Stud (
            const Stud & other)  [inline], [noexcept]
```

Kopijavimo konstruktorius.

Parameters

other	Kitas studentas, kurio duomenys bus nukopijuoti.

Definition at line 130 of file Stud.h.

```
00131         : zmogus(other.getVardas(), other.getPavarde()),// Kopijavimo inicializacija iš zmogus
klasės
00132         ND(other.ND),
00133         egz(other.egz),
00134         GalutinisVid(other.GalutinisVid),
00135         GalutinisMed(other.GalutinisMed) {}
```

5.1.2.3 ~Stud()

```
Stud::~Stud ()  [inline]
```

Destruktorius, kuris išvalo duomenis.

Definition at line 163 of file Stud.h.

```
00163 { clearData(); }// Destruktorius, kuris išvalo studento duomenis
```

5.1.3 Member Function Documentation

5.1.3.1 addND()

```
void Stud::addND (
            int nd)  [inline]
```

Pridėti naują namų darbų rezultatą.

Parameters

nd	Namų darbų rezultatas.

Definition at line 156 of file Stud.h.

```
00156                         {// Pridedame naują namų darbų rezultatą į vektorių
00157         ND.push_back(nd);
00158     }
```

5.1.3.2 apskaiciuotiGalutinius()

```
void Stud::apskaiciuotiGalutinius ()
```

Apskaičiuoja galutinius įvertinimus (vidurkį ir medianą).

Apskaičiuoja galutinius balus pagal vidurkį ir medianą.

Ši funkcija apskaičiuoja studento galutinį balą naudojant namų darbų vidurkį ir mediana, pridedant egzaminų balą su nustatytais svoriais:

- 40% vidurkis arba mediana iš namų darbų
- 60% egzamino balas.

Definition at line 57 of file Studentu_duomenys.cpp.

```
00057                                   {
00058     if (ND.empty()) {
00059         cout « "Nd yra tuscias, negalima suskaiciuoti galutinio balo" « endl;
00060         GalutinisVid = 0.0;
00061         GalutinisMed = 0.0;
00062         return;
00063     }
00064     double vidutinis = 0.0;
00065     for (double nd : ND) {
00066         vidutinis += nd;// Skaičiuojame namų darbų vidurkį.
00067     }
00068     vidutinis /= ND.size();
00069     GalutinisVid = 0.4 * vidutinis + 0.6 * egz;
00070
00071     double mediana = apskaiciuotiMediana(ND);// Skaičiuojame namų darbų medianą.
00072     GalutinisMed = 0.4 * mediana + 0.6 * egz;
00073 }
```

5.1.3.3 atvaizduoti()

```
void Stud::atvaizduoti (
            std::ostream & os) const  [virtual]
```

Atvaizduoja studento duomenis.

Atvaizduoja studento informaciją į srautą.

Parameters

os	Išvesties srautas.

Ši funkcija išveda studento vardą, pavardę, galutinį vidurkį ir galutinį mediana į srautą.

Parameters

os	Išvesties srautas, į kurį bus rašoma informacija apie studentą.

Implements zmogus.

Definition at line 32 of file Stud.cpp.

```
00032                                           {// Išvedame studento vardą, pavardę, galutinį
vidurkį ir galutinę medianą į išvesties srautą
00033     os « "Vardas:" « getVardas() « ", Pavarde:" « getPavarde() « ", GalutinisVid:" «
GalutinisVid
00034         « ", GalutinisMed:" « GalutinisMed « endl;
00035 };
```

5.1.3.4 clearData()

```
void Stud::clearData ()  [inline]
```

Išvalo studento duomenis.

Definition at line 168 of file Stud.h.

```
00168                     {
00169         ND.clear();// Išvalome namų darbų rezultatus
00170         egz = 0.0; // Nustatome egzamino rezultatą į 0
00171         GalutinisVid = 0.0; // Nustatome galutinį vidurkį į 0
00172         GalutinisMed = 0.0; // Nustatome galutinę medianą į 0
00173     }
```

5.1.3.5 getEgz()

```
double Stud::getEgz () const  [inline]
```

Grąžina egzamino rezultatą.

Returns

Egzamino rezultatas.

Definition at line 185 of file Stud.h.

```
00185 { return egz; }
```

5.1.3.6 getGalutinisMed()

```
double Stud::getGalutinisMed () const  [inline]
```

Grąžina galutinį įvertinimą (medianą).

Returns

Galutinis įvertinimas (mediana).

Definition at line 197 of file Stud.h.

```
00197 { return GalutinisMed; }
```

5.1.3.7 getGalutinisVid()

```
double Stud::getGalutinisVid () const  [inline]
```

Grąžina galutinį įvertinimą (vidurkį).

Returns

Galutinis įvertinimas (vidurkis).

Definition at line 191 of file Stud.h.

```
00191 { return GalutinisVid; }
```

5.1.3.8 getND()

```
std::vector< int > Stud::getND () const  [inline]
```

Grąžina namų darbų rezultatus.

Returns

Namų darbų rezultatai.

Definition at line 179 of file Stud.h.

```
00179 { return ND; }
```

5.1.3.9 operator=()

```
Stud & Stud::operator= (
            const Stud & other)  [inline], [noexcept]
```

Kopijavimo priskyrimo operatorius.

Parameters

other	Kitas studentas, kurio duomenys bus priskiriami.

Returns

Nuoroda į šį objektą.

Definition at line 142 of file Stud.h.

```
00142                                                {
00143         if (this == &other) return *this;// Patikrinimas, kad neatsitiktų savitarpio priskyrimas
00144         zmogus::operator = (other);// Kviečiamas paveldėtas operatorius
00145         ND = other.ND;// Priskiriame namų darbų rezultatus
00146         egz = other.egz;// Priskiriame egzamino rezultatą
00147         GalutinisVid = other.GalutinisVid;// Priskiriame galutinį vidurkį
00148         GalutinisMed = other.GalutinisMed; // Priskiriame galutinę medianą
00149         return *this;// Grąžiname šį objektą
00150     }
```

5.1.3.10 setEgz()

```
void Stud::setEgz (
            double e)  [inline]
```

Nustato egzamino rezultatą.

Parameters

e	Egzamino rezultatas.

Definition at line 209 of file Stud.h.

```
00209 { egz = e; }
```

5.1.3.11 setND()

```
void Stud::setND (
            const std::vector< int > & nd)  [inline]
```

Nustato namų darbų rezultatus.

Parameters

nd	Namų darbų rezultatai.

Definition at line 203 of file Stud.h.

```
00203 { ND = nd; }
```

5.1.4 Friends And Related Symbol Documentation

5.1.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Stud & stud)  [friend]
```

Draugiškas operatorius, kuris išveda studentą į srautą.

Parameters

os	Išvesties srautas.
stud	Studentas.

Returns

Išvesties srautas su studento duomenimis.

Ši funkcija leidžia atvaizduoti visus studento duomenis naudojant << operatorių.

Parameters

os	Išvesties srautas, į kurį bus rašoma informacija apie studentą.
stud	Studentas, kurio informacija bus atvaizduota.

Returns

std::ostream& Nuoroda į išvesties srautą.

Definition at line 173 of file Stud.cpp.

```
00173                                                 {
00174     stud.atvaizduoti(os);// Kviečiama funkciją, kuri atvaizduoja studento duomenis
00175     return os;// Grąžinamas išvesties srautas
00176
00177 }
```

5.1.4.2 operator>>

```
std::istream & operator>> (
            std::istream & is,
            Stud & stud)  [friend]
```

Draugiškas operatorius, kuris įveda studentą iš srauto.

Parameters

is	Įvesties srautas.
stud	Studentas.

Returns

Įvesties srautas su studento duomenimis.

Šis operatorius leidžia įvesti studento duomenis, tokius kaip vardas, pavardė, namų darbų rezultatai ir egzamino rezultatas. Duomenys gali būti įvedami tiek atsitiktinai, tiek rankiniu būdu.

Parameters

is	Įvesties srautas, iš kurio bus nuskaityti duomenys.
stud	Objektas, į kurį bus saugomi nuskaityti duomenys.

Returns

std::istream& Nuoroda į įvesties srautą.

Paprašo vartotojo įvesti vardą ir pavardę. Šie duomenys priskiriami studento objektui.

Nustato įvestą vardą ir pavardę studento objektui.

Patikrina, ar rezultatai turėtų būti generuojami atsitiktinai.

Inicijuojamas atsitiktinių skaičių generatorius. Naudojamas sugeneruoti namų darbų ir egzamino balams.

Generuoja nurodyto dydžio atsitiktinius namų darbų balus. Balai priskiriami studento objektui.

Sugeneruojamas atsitiktinis egzamino balas ir priskiriamas studentui.

Vartotojas įveda konkrečių namų darbų skaičių ir rezultatus. Visi rezultatai pridedami prie studento objekto

Naudotojas įveda dinamiškai namų darbų rezultatus. Įvedimas baigiamas įvedus -1. Kiekvienas rezultatas pridedamas prie studento objekto

Definition at line 48 of file Stud.cpp.

```
00048                                                 {
00049
00053
00054     cout « "Ivesti varda ir pavarde: ";// Prašome vartotojo įvesti vardą ir pavardę
00055     string vardas, pavarde;
00056     is » vardas » pavarde;// Nuskaitomas vardas ir pavarde
00057
00060
00061      // Nustatomas įvestas vardas ir pavarde studento objektui
00062     stud.setVardas(vardas);
00063     stud.setPavarde(pavarde);
00064
00065     cout « "Konstruktoriu: Objekto " « vardas « " " « pavarde « " sukurimas" « endl;
00066
00067     string pasirinkimas;
00068
00071
00072     cout « "Ar reikia namu darbu ir egzamino rezultatus generuoti atsitiktinai?(taip/ne) ";//
Patikriname, ar rezultatai turi būti generuojami atsitiktinai
00073     is » pasirinkimas;
00074
00075     if (pasirinkimas == "taip") {
00076
00080
00081          // Inicijuojamas atsitiktinių skaičių generatorius
00082         random_device rd;
00083         mt19937 gen(rd());
00084         uniform_int_distribution<> dist(1, 10);
00085
00086         int ndCount;
00087         cout « "Ivesti namu darbu skaiciu: ";
00088         is » ndCount;// Nuskaityti namų darbų skaičių
00089
00093
00094          // Sugeneruojame nurodyto dydžio atsitiktinius namų darbų balus
00095         for (int i = 0; i < ndCount; ++i) {
00096             stud.addND(dist(gen));// Sugeneruojame ir pridedame atsitiktinį namų darbų balą
```

```
00097         }
00098
00101
00102         // Sugeneruojamas atsitiktinis egzamino balas ir priskiriamas studentui
00103         stud.setEgz(dist(gen));
00104         stud.apskaiciuotiGalutinius();// Apskaičiuojamas galutinis įvertinimas
00105
00106     }
00107     else {
00108         cout « "Ar zinai, koks yra namu darbu skaicius?(taip/ne): ";
00109         is » pasirinkimas;
00110
00111         if (pasirinkimas == "taip") {
00112
00116
00117              // Vartotojas įveda konkrečių namų darbų skaičių ir rezultatus
00118             int ndCount;
00119             cout « "Ivesti namu darbu skaiciu: ";
00120             is » ndCount;
00121
00122
00123
00124             cout « "Ivesti namu darbu rezultatus(10-baleje sistemoje): ";
00125             for (int i = 0; i < ndCount; ++i) {
00126                 int nd;
00127                 is » nd;// Vartotojas įveda kiekvieną namų darbų rezultatą
00128                 stud.addND(nd);// Pridedamas rezultatas prie studento objekto
00129
00130             }
00131         }
00132         else if (pasirinkimas == "ne") {
00133
00138
00139             // Naudotojas įveda dinamiškai namų darbų rezultatus
00140             double nd;
00141             cout « "Ivesti namu darbu rezultatus(ivesti '-1',kad baigti):" « endl;
00142             while (true) {
00143                 is » nd;// Vartotojas įveda kiekvieną namų darbų rezultatą
00144                 if (nd == -1) {// Baigiasi įvedimas, jei įvedama -1
00145                     break;
00146                 }
00147                 stud.addND(nd);// Pridedamas rezultatas prie studento objekto
00148             }
00149         }
00150
00151         // Naudotojas įveda egzamino rezultatą ir priskiria jį studento objektui.
00152         cout « "Ivesti egzamino rezultata: ";
00153         double egz;
00154         is » egz; // Vartotojas įveda egzamino rezultatą
00155         stud.setEgz(egz);// Priskiriamas egzamino rezultatas studentui
00156
00157
00158     }
00159
00160     return is;// Grąžinamas įvesties srautas
00161
00162 }
```

The documentation for this class was generated from the following files:

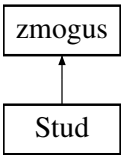
- Stud.h
- Stud.cpp
- Studentu_duomenys.cpp

5.2 zmogus Class Reference

Klasė, kuri aprašo žmogų su vardu ir pavarde.

`#include <Stud.h>`

Inheritance diagram for zmogus:



Public Member Functions

- zmogus (std::string v="", std::string p="")
  Konstruktorius, kuriame nustatomi vardas ir pavardė.
- virtual ~zmogus ()=default
  Virtualus destruktorius (default).
- std::string getVardas () const
  Grąžina žmogaus vardą.
- std::string getPavarde () const
  Grąžina žmogaus pavardę.
- virtual void setVardas (const std::string &v)
  Nustato žmogaus vardą.
- virtual void setPavarde (const std::string &p)
  Nustato žmogaus pavardę.
- virtual void atvaizduoti (std::ostream &os) const =0
  Grynoji virtuali funkcija, kuri turi būti implementuota paveldėtojų klasėse.

Friends

- std::ostream & operator<< (std::ostream &os, const zmogus &zmog)
  Draugiškas operatorius, kuris išveda žmogų į srautą.

5.2.1 Detailed Description

Klasė, kuri aprašo žmogų su vardu ir pavarde.

Ši klasė suteikia pagrindines funkcijas, leidžiančias nustatyti ir gauti žmogaus vardą ir pavardę.

Definition at line 41 of file Stud.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 zmogus()

```
zmogus::zmogus (
            std::string v = "",
            std::string p = "")  [inline]
```

Konstruktorius, kuriame nustatomi vardas ir pavardė.

Parameters

v	Žmogaus vardas (numatytasis tuščias).
p	Žmogaus pavardė (numatytasis tuščias).

Definition at line 51 of file Stud.h.

```
00052         : vardas(v), pavarde(p) {}
```

5.2.2.2 ~zmogus()

```
virtual zmogus::~zmogus ()  [virtual], [default]
```

Virtualus destruktorius (default).

5.2.3 Member Function Documentation

5.2.3.1 atvaizduoti()

```
virtual void zmogus::atvaizduoti (
            std::ostream & os) const  [pure virtual]
```

Grynoji virtuali funkcija, kuri turi būti implementuota paveldėtojų klasėse.

Parameters

os	Išvesties srautas.

Implemented in Stud.

5.2.3.2 getPavarde()

```
std::string zmogus::getPavarde () const  [inline]
```

Grąžina žmogaus pavardę.

Returns

Žmogaus pavardė.

Definition at line 69 of file Stud.h.

```
00069 { return pavarde; }// Pavardės grąžinimo funkcija
```

5.2.3.3 getVardas()

```
std::string zmogus::getVardas () const  [inline]
```

Grąžina žmogaus vardą.

Returns

Žmogaus vardas.

Definition at line 63 of file Stud.h.

```
00063 { return vardas; }// Vardo grąžinimo funkcija
```

5.2.3.4 setPavarde()

```
virtual void zmogus::setPavarde (
            const std::string & p)  [inline], [virtual]
```

Nustato žmogaus pavardę.

Parameters

p	Žmogaus pavardė.

Definition at line 82 of file Stud.h.

```
00082 { pavarde = p; } // Pavardės nustatymo funkcija
```

5.2.3.5 setVardas()

```
virtual void zmogus::setVardas (
            const std::string & v)  [inline], [virtual]
```

Nustato žmogaus vardą.

Parameters

v	Žmogaus vardas.

Definition at line 75 of file Stud.h.

```
00075 { vardas = v; }// Vardo nustatymo funkcija
```

5.2.4 Friends And Related Symbol Documentation

5.2.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const zmogus & zmog)  [friend]
```

Draugiškas operatorius, kuris išveda žmogų į srautą.

Parameters

os	Išvesties srautas.
zmog	Žmogaus objektas.

Returns

Išvesties srautas su žmogaus duomenimis.

Definition at line 96 of file Stud.h.

```
00096                                                                   {
00097         zmog.atvaizduoti(os);// Kvieciama atvaizduoti funkcija
00098         return os;// Grąžina išvesties srautą su žmogaus duomenimis
00099     }
```

The documentation for this class was generated from the following file:

- Stud.h

Chapter 6

File Documentation

6.1 FailoGeneravimas.cpp File Reference

Studentų duomenų generavimas į failą.

```
#include "Stud.h"
#include "Mylib.h"
```

Functions

- void sugeneruotiStudentoFaila (const string &fileName, int studentCount, int ndCount)
  Sugeneruoja studentų duomenų failą.

6.1.1 Detailed Description

Studentų duomenų generavimas į failą.

Šiame faile įgyvendinamas atsitiktinių studentų duomenų generavimas ir išsaugojimas į tekstinį failą. Kiekvienam studentui sugeneruojami atsitiktiniai vardai, pavardės, namų darbų rezultatai, egzamino rezultatas, kurie vėliau įrašomi į tekstinį failą nurodytu formatu.

Failas naudoja „mylib“ biblioteką ir susijusius metodus efektyviam duomenų rašymui.

Definition in file FailoGeneravimas.cpp.

6.1.2 Function Documentation

6.1.2.1 sugeneruotiStudentoFaila()

```
void sugeneruotiStudentoFaila (
            const string & fileName,
            int studentCount,
            int ndCount)
```

Sugeneruoja studentų duomenų failą.

Ši funkcija sukuria failą su studentų vardais, pavardėmis, jų namų darbų (ND) rezultatais ir egzamino rezultatais. Kiekvienam studentui sugeneruojami atsitiktiniai ND ir egzaminų balai, kurie įrašomi į nurodytą failą.

Parameters

fileName	Failo pavadinimas, kuriame bus išsaugoti studentų duomenys.
studentCount	Kiek studentų duomenų turi būti sugeneruota.
ndCount	Kiek namų darbų rezultatų turi būti sugeneruota kiekvienam studentui.

Definition at line 30 of file FailoGeneravimas.cpp.

```
00030                                                                                                                {
00031     auto pradzia = std::chrono::high_resolution_clock::now();  // Laiko pradžia
00032
00033
00034     ofstream outFile(fileName); // Sukuriame failą su nurodytu pavadinimu
00035     if (!outFile) {
00036         std::cerr « "Nepavyko sukurti failo: " « fileName « endl;
00037         return;
00038     }
00039
00040     // Pirmos eilutės su antraštėmis įrašymas į failą
00041     outFile « setw(25) « left « "Vardas"
00042         « setw(25) « left « "Pavarde";
00043
00044     // Generuojame ND antraštes
00045     for (int i = 1; i <= ndCount; i++) {
00046         outFile « setw(10) « right « "ND" + std::to_string(i);
00047     }
00048     outFile « setw(10) « right « "Egz." « endl;
00049
00050
00051
00052     // Atsitiktinių skaičių generatorius(1 - 10 balai).
00053     random_device rd;  // Atsitiktinių skaičių generatorius.
00054     mt19937 gen(rd());  // Inicializuojame generatorių.
00055     uniform_int_distribution<> dist(1, 10);  // Skaičių intervalas nuo 1 iki 10
00056
00057     // Generuojame duomenis kiekvienam studentui
00058     for (int i = 1; i <= studentCount; ++i) {
00059
00060         // Sukuriame studentą su vardu ir pavarde
00061         Stud studentas("Vardas" + std::to_string(i), "Pavarde" + std::to_string(i));
00062
00063         // Generuojame ND balus
00064         for (int j = 0; j < ndCount; ++j) {
00065             studentas.addND(dist(gen)); // Pridedame atsitiktinį ND balą
00066         }
00067
00068         // Generuojame egzamino balą
00069         studentas.setEgz(dist(gen));
00070
00071         // Įrašome studento duomenis į failą
00072         outFile « setw(25) « left « studentas.getVardas()
00073             « setw(25) « left « studentas.getPavarde();
00074
00075         // Įrašome ND rezultatus
00076         for (int nd : studentas.getND()) {
00077             outFile « setw(10) « right « nd;
00078         }
00079
00080         // Įrašome egzamino rezultatą
00081         outFile « setw(10) « right « studentas.getEgz() « endl;
00082     }
00083
00084     //Uždaromas failas
00085     outFile.close();
00086     cout « "Failas sukurtas: " « fileName « endl;
00087
00088     // Apskaičiuojame ir išvedame failo kūrimo trukmę
00089     auto pabaiga = std::chrono::high_resolution_clock::now();
00090     std::chrono::duration<double> elapsed = pabaiga - pradzia;
00091     cout « "Failo kurimo trukme: " « fixed « setprecision(5) « elapsed.count() « " s" « endl;
00092 }
```

6.2 FailoGeneravimas.cpp

Go to the documentation of this file.

```
00001
00015 #include "Stud.h"
00016 #include "Mylib.h"
00017
00018
00030 void sugeneruotiStudentoFaila(const string& fileName, int studentCount, int ndCount) {
00031     auto pradzia = std::chrono::high_resolution_clock::now();  // Laiko pradžia
00032
00033
00034     ofstream outFile(fileName); // Sukuriame failą su nurodytu pavadinimu
00035     if (!outFile) {
```

```
00036         std::cerr « "Nepavyko sukurti failo: " « fileName « endl;
00037         return;
00038     }
00039
00040     // Pirmos eilutės su antraštėmis įrašymas į failą
00041     outFile « setw(25) « left « "Vardas"
00042         « setw(25) « left « "Pavarde";
00043
00044     // Generuojame ND antraštes
00045     for (int i = 1; i <= ndCount; i++) {
00046         outFile « setw(10) « right « "ND" + std::to_string(i);
00047     }
00048     outFile « setw(10) « right « "Egz." « endl;
00049
00050
00051
00052     // Atsitiktinių skaičių generatorius(1 - 10 balai).
00053     random_device rd;  // Atsitiktinių skaičių generatorius.
00054     mt19937 gen(rd());  // Inicializuojame generatorių.
00055     uniform_int_distribution<> dist(1, 10);  // Skaičių intervalas nuo 1 iki 10
00056
00057     // Generuojame duomenis kiekvienam studentui
00058     for (int i = 1; i <= studentCount; ++i) {
00059
00060         // Sukuriame studentą su vardu ir pavarde
00061         Stud studentas("Vardas" + std::to_string(i), "Pavarde" + std::to_string(i));
00062
00063         // Generuojame ND balus
00064         for (int j = 0; j < ndCount; ++j) {
00065             studentas.addND(dist(gen)); // Pridedame atsitiktinį ND balą
00066         }
00067
00068         // Generuojame egzamino balą
00069         studentas.setEgz(dist(gen));
00070
00071         // Įrašome studento duomenis į failą
00072         outFile « setw(25) « left « studentas.getVardas()
00073             « setw(25) « left « studentas.getPavarde();
00074
00075         // Įrašome ND rezultatus
00076         for (int nd : studentas.getND()) {
00077             outFile « setw(10) « right « nd;
00078         }
00079
00080         // Įrašome egzamino rezultatą
00081         outFile « setw(10) « right « studentas.getEgz() « endl;
00082     }
00083
00084     //Uždaromas failas
00085     outFile.close();
00086     cout « "Failas sukurtas: " « fileName « endl;
00087
00088     // Apskaičiuojame ir išvedame failo kūrimo trukmę
00089     auto pabaiga = std::chrono::high_resolution_clock::now();
00090     std::chrono::duration<double> elapsed = pabaiga - pradzia;
00091     cout « "Failo kurimo trukme: " « fixed « setprecision(5) « elapsed.count() « " s" « endl;
00092 }
```

6.3 FailoNuskaitymas.cpp File Reference

Studentų duomenų nuskaitymas iš failo.

```
#include "Stud.h"
#include "Mylib.h"
```

Functions

- void nuskaitytiIsFailo (std::vector< Stud > &Vec1, const std::string &failoVardas)
  Nuskaityti studentų duomenis iš failo ir įkelti į vektorių.
- void nuskaitytiIsfailo (std::list< Stud > &list1, const std::string &failoVardas)
  Nuskaityti studentų duomenis iš failo ir įkelti į sąrašą.

6.3.1 Detailed Description

Studentų duomenų nuskaitymas iš failo.

Šiame faile apdorojamas tekstinio failo nuskaitymas, kuriame saugomi studentų duomenys. Nuskaityti duomenys apdorojami ir įrašomi į atitinkamus struktūrų konteinerius (pvz., vektorius arba sąrašus), kuriuose saugomi studentų vardai, pavardės, namų darbų ir egzamino rezultatai.

Failas naudoja „mylib“ biblioteką.

Definition in file FailoNuskaitymas.cpp.

6.3.2 Function Documentation

6.3.2.1 nuskaitytiIsFailo()

```
void nuskaitytiIsFailo (
            std::vector< Stud > & Vec1,
            const std::string & failoVardas)
```

Nuskaityti studentų duomenis iš failo ir įkelti į vektorių.

Nuskaityti studentus iš failo ir įrašyti juos į vektorių.

Ši funkcija atidaro nurodytą failą, nuskaito kiekvieną eilutę ir iš jos sukuria studento objektą, kurio vardas, pavardė, namų darbai ir egzamino rezultatai yra įrašomi ir perduodami į `std::vector`. Jeigu duomenų eilutėje yra klaida arba trūksta reikalingų duomenų, studentas ignoruojamas.

Parameters

Vec1	Nuoroda į `std::vector`, kuriame bus saugomi nuskaityti studentų duomenys.
failoVardas	Failo, iš kurio bus nuskaityti studentų duomenys, pavadinimas.

< Atidarome failą su nurodytu pavadinimu

Definition at line 28 of file FailoNuskaitymas.cpp.

```
00028                                                                                {
00029     try {
00030         std::ifstream inFile(failoVardas);
00031         if (!inFile) {
00032             throw runtime_error("Nepavyko atidaryti failo:" + failoVardas);
00033         }
00034         string line;
00035         while (getline(inFile, line)) {
00036 
00037             std::stringstream ss(line);
00038             Stud temp;
00039             temp.clearData(); // Išvalome ankstesnius duomenis
00040 
00041             std::string vardas, pavardė;
00042             if (!(ss » vardas » pavardė)) {
00043                 std::cerr « "Nepavyko nuskaityti studento vardo ir pavardės" « endl;
00044                 continue; // Jei vardo ir pavardės nuskaityti nepavyko, pereiname prie kitos
eilutės
00045             }
00046 
00047             temp.setVardas(vardas);
00048             temp.setPavarde(pavardė);
00049 
00050             int nd;
00051             while (ss » nd) {
00052                 temp.addND(nd); // Pridedame namų darbų balus
00053             }
00054             if (temp.getND().size() < 1) {
00055                 continue; // Jei ND trūksta, ignoruojame studentą
00056             }
00057             temp.setEgz(temp.getND().back()); // Paskutinį namų darbų balą priskiriame egzamino
rezultatui
00058             temp.getND().pop_back();  // Pašaliname paskutinį ND, kad liktų tik namų darbų
rezultatai
00059 
00060             Vec1.push_back(temp); // Pridedame studentą į vektorių
00061         }
00062         inFile.close(); // Uždaromas failas po nuskaitymo
00063 
00064     }
00065     catch (const std::exception& e) {
00066         std::cerr « "Klaida: " « e.what() « endl; // Jei įvyko klaida, išvedame klaidos
pranešimą
00067     }
00068 }
```

6.3.2.2 nuskaitytilsfailo()

```
void nuskaitytiIsfailo (
            std::list< Stud > & list1,
            const std::string & failoVardas)
```

Nuskaityti studentų duomenis iš failo ir įkelti į sąrašą.

Nuskaityti studentus iš failo ir įrašyti juos į sąrašą.

Ši funkcija atidaro nurodytą failą, nuskaito kiekvieną eilutę ir iš jos sukuria studento objektą, kurio vardas, pavardė, namų darbai ir egzamino rezultatai yra įrašomi ir perduodami `std::list`. Jeigu duomenų eilutėje yra klaida arba trūksta reikalingų duomenų, studentas ignoruojamas.

Parameters

list1	Nuoroda į `std::list`, kuriame bus saugomi nuskaityti studentų duomenys.
failoVardas	Failo, iš kurio bus nuskaityti studentų duomenys, pavadinimas.

Definition at line 81 of file FailoNuskaitymas.cpp.

```
00081                                                                                {
00082     try {
00083         std::ifstream inFile(failoVardas); // Atidarome failą su nurodytu pavadinimu
00084         if (!inFile) {
00085             throw runtime_error("Nepavyko atidaryti failo:" + failoVardas);
00086         }
00087         string line;
00088         while (getline(inFile, line)) {
00089
00090             std::stringstream ss(line);
00091             Stud temp;
00092             temp.clearData(); //Išvalome ankstesnius duomenis
00093
00094             std::string vardas, pavardė;
00095             if (!(ss » vardas » pavardė)) {
00096                 std::cerr « "Nepavyko nuskaityti studento vardo ir pavardės" « endl;
00097                 continue; // Jei vardo ir pavardės nuskaityti nepavyko, pereiname prie kitos
eilutės
00098             }
00099             temp.setVardas(vardas);
00100             temp.setPavarde(pavardė);
00101
00102             int nd;
00103             while (ss » nd) {
00104                 temp.addND(nd); // Pridedame namų darbų balus
00105             }
00106             if (temp.getND().size() < 1) {
00107                 continue; // Jei ND trūksta, ignoruojame studentą
00108             }
00109             temp.setEgz(temp.getND().back()); // Paskutinį namų darbų balą priskiriame egzamino
rezultatui
00110             temp.getND().pop_back();  // Pašaliname paskutinį ND, kad liktų tik namų darbų
rezultatai
00111
00112             list1.push_back(temp); // Pridedame studentą į sąrašą
00113         }
00114         inFile.close(); // Uždaromas failas po nuskaitymo
00115
00116     }
00117     catch (const std::exception& e) {
00118         std::cerr « "Klaida: " « e.what() « endl; // Jei įvyko klaida, išvedame klaidos
pranešimą
00119     }
00120 }
```

6.4 FailoNuskaitymas.cpp

Go to the documentation of this file.

```
00001
00014 #include "Stud.h"
00015 #include "Mylib.h"
00016
00028 void nuskaitytiIsFailo(std::vector<Stud>& Vec1, const std::string& failoVardas) {
00029     try {
00030         std::ifstream inFile(failoVardas);
00031         if (!inFile) {
```

```
00032             throw runtime_error("Nepavyko atidaryti failo:" + failoVardas);
00033         }
00034         string line;
00035         while (getline(inFile, line)) {
00036 
00037             std::stringstream ss(line);
00038             Stud temp;
00039             temp.clearData(); // Išvalome ankstesnius duomenis
00040 
00041             std::string vardas, pavardė;
00042             if (!(ss » vardas » pavardė)) {
00043                 std::cerr « "Nepavyko nuskaityti studento vardo ir pavardės" « endl;
00044                 continue; // Jei vardo ir pavardės nuskaityti nepavyko, pereiname prie kitos
eilutės
00045             }
00046 
00047             temp.setVardas(vardas);
00048             temp.setPavarde(pavardė);
00049 
00050             int nd;
00051             while (ss » nd) {
00052                 temp.addND(nd); // Pridedame namų darbų balus
00053             }
00054             if (temp.getND().size() < 1) {
00055                 continue; // Jei ND trūksta, ignoruojame studentą
00056             }
00057             temp.setEgz(temp.getND().back()); // Paskutinį namų darbų balą priskiriame egzamino
rezultatui
00058             temp.getND().pop_back();  // Pašaliname paskutinį ND, kad liktų tik namų darbų
rezultatai
00059 
00060             Vec1.push_back(temp); // Pridedame studentą į vektorių
00061         }
00062         inFile.close(); // Uždaromas failas po nuskaitymo
00063 
00064     }
00065     catch (const std::exception& e) {
00066         std::cerr « "Klaida: " « e.what() « endl; // Jei įvyko klaida, išvedame klaidos
pranešimą
00067     }
00068 }
00069 
00081 void nuskaitytiIsfailo(std::list<Stud>& list1, const std::string& failoVardas) {
00082     try {
00083         std::ifstream inFile(failoVardas); // Atidarome failą su nurodytu pavadinimu
00084         if (!inFile) {
00085             throw runtime_error("Nepavyko atidaryti failo:" + failoVardas);
00086         }
00087         string line;
00088         while (getline(inFile, line)) {
00089 
00090             std::stringstream ss(line);
00091             Stud temp;
00092             temp.clearData(); //Išvalome ankstesnius duomenis
00093 
00094             std::string vardas, pavardė;
00095             if (!(ss » vardas » pavardė)) {
00096                 std::cerr « "Nepavyko nuskaityti studento vardo ir pavardės" « endl;
00097                 continue; // Jei vardo ir pavardės nuskaityti nepavyko, pereiname prie kitos
eilutės
00098             }
00099             temp.setVardas(vardas);
00100             temp.setPavarde(pavardė);
00101 
00102             int nd;
00103             while (ss » nd) {
00104                 temp.addND(nd); // Pridedame namų darbų balus
00105             }
00106             if (temp.getND().size() < 1) {
00107                 continue; // Jei ND trūksta, ignoruojame studentą
00108             }
00109             temp.setEgz(temp.getND().back()); // Paskutinį namų darbų balą priskiriame egzamino
rezultatui
00110             temp.getND().pop_back();  // Pašaliname paskutinį ND, kad liktų tik namų darbų
rezultatai
00111 
00112             list1.push_back(temp); // Pridedame studentą į sąrašą
00113         }
00114         inFile.close(); // Uždaromas failas po nuskaitymo
00115 
00116     }
00117     catch (const std::exception& e) {
00118         std::cerr « "Klaida: " « e.what() « endl; // Jei įvyko klaida, išvedame klaidos
pranešimą
00119     }
00120 }
00121 
```

6.5 Mylib.h File Reference

Ši biblioteka apima pagrindines C++ standartines bibliotekas ir prideda aliasus.

```
#include <iostream>
#include <iomanip>
#include <string>
#include <vector>
#include <algorithm>
#include <random>
#include <fstream>
#include <sstream>
#include <chrono>
#include <list>
```

6.5.1 Detailed Description

Ši biblioteka apima pagrindines C++ standartines bibliotekas ir prideda aliasus.

Ši antraštė suteikia naudotojui prieigą prie populiarių C++ funkcionalumų, tokių kaip įvesties/išvesties srautai, konteineriai (pvz., std::vector), atsitiktinių skaičių generavimas ir failų įvedimas/istrynimas.

Definition in file Mylib.h.

6.6 Mylib.h

Go to the documentation of this file.

```
00001 #ifndef MYLIB_H_INCLUDED
00002 #define MYLIB_H_INCLUDED
00003
00010 #include <iostream>   // Apima įvesties/išvesties funkcionalumus.
00011 #include <iomanip>    // Apima funkcijas, susijusias su formatavimu (pvz., nustatyti skaitmenų
tikslumą).
00012 #include <string>     // Apima klasę std::string ir funkcijas dirbti su tekstu.
00013 #include <vector>     // Apima std::vector konteinerį (dinamiškai valdomus masyvus).
00014 #include <algorithm>  // Apima algoritmus, tokius kaip sort, find, ir kt.
00015 #include <random>     // Apima atsitiktinių skaičių generavimo funkcijas.
00016 #include <fstream>    // Apima failų įvesties/išvesties funkcijas.
00017 #include <sstream>    // Apima srautų klasę, skirtą dirbti su eilutėmis.
00018 #include <chrono>     // Apima laikrodžio funkcijas ir laiką.
00019 #include <list>       // Apima std::list konteinerį (dviašius sąrašus).
00020
00021  // Pagrindinių funkcijų naudojimo paprastinimas:
00022 using std::endl;              // Leidžia naudoti std::endl.
00023 using std::cout;              // Leidžia naudoti std::cout (standartinis išvesties srautas).
00024 using std::cin;               // Leidžia naudoti std::cin (standartinis įvesties srautas).
00025 using std::left;              // Leidžia naudoti std::left (kairiojo lygiavimo manipuliatorius).
00026 using std::right;             // Leidžia naudoti std::right (dešiniojo lygiavimo
manipuliatorius).
00027 using std::setw;              // Leidžia naudoti std::setw (išvesties pločio nustatymas).
00028 using std::setprecision;      // Leidžia naudoti std::setprecision (tikslumo nustatymas
skaičiams).
00029 using std::fixed;             // Leidžia naudoti std::fixed (formatavimas, kad skaičiai būtų
rodomi fiksuotu taškų tikslumu).
00030 using std::string;            // Leidžia naudoti std::string (standartinė eilutės klasė).
00031 using std::vector;            // Leidžia naudoti std::vector (dinaminis masyvas).
00032 using std::random_device;     // Leidžia naudoti std::random_device (atsitiktinių skaičių
generatorius).
00033 using std::mt19937;           // Leidžia naudoti std::mt19937 (atsitiktinių skaičių
generatorius).
00034 using std::uniform_int_distribution; // Leidžia naudoti std::uniform_int_distribution
(atsitiktinių sveikųjų skaičių paskirstymas).
00035 using std::fstream;           // Leidžia naudoti std::fstream (failų srautai).
00036 using std::sort;              // Leidžia naudoti std::sort (rūšiavimo funkcija).
00037 using std::runtime_error;     // Leidžia naudoti std::runtime_error (klaidos tipas).
00038 using std::ofstream;          // Leidžia naudoti std::ofstream (išvesties failo srautas).
00039 using std::list;              // Leidžia naudoti std::list (dviašiai sąrašai).
00040 using std::remove_if;         // Leidžia naudoti std::remove_if (elementų pašalinimas pagal
sąlygą).
00041 using std::partition;         // Leidžia naudoti std::partition (konteinerių dalijimas pagal
sąlygą).
00042 using std::stable_partition;  // Leidžia naudoti std::stable_partition (stabilus konteinerių
dalijimas).
```

```
00043
00044
00045 #endif // MYLIB_H_INCLUDED
```

6.7 readme.md File Reference

6.8 Stud.cpp File Reference

Studentų objektų realizacija.

```
#include "Stud.h"
#include "Mylib.h"
```

Functions

- std::istream & operator>> (std::istream &is, Stud &stud)
  Nuskaitymo operatorius studento duomenims įvesti iš srauto.
- std::ostream & operator<< (std::ostream &os, const Stud &stud)
  Išvesties operatorius studento duomenims atvaizduoti.
- void val (Stud &Lok)
  Nustato studento duomenis į pradinius (tuščius) reikšmes.

6.8.1 Detailed Description

Studentų objektų realizacija.

Šiame faile įgyvendinamos studento duomenų struktūros ir metodai, leidžiantys apdoroti studentų informaciją. Tai apima tokius veiksmus, kaip duomenų įvedimas, išvedimas, skaičiavimai, taip pat vidurkių ir medianų apskaičiavimas pagal studentų namų darbų ir egzamino rezultatus.

Failas įgyvendina funkcijas, kurios leidžia nustatyti studento duomenis, pavyzdžiui, vardą, pavardę, namų darbų rezultatus ir egzamino rezultatą, bei atlikti atitinkamus skaičiavimus, tokius kaip galutinio įvertinimo apskaičiavimas (vidurkis ir mediana). Taip pat realizuoti įvedimo ir išvedimo metodai, kurie leidžia vartotojui bendrauti su studentų objektais.

Failas naudoja „mylib“ biblioteką.

Definition in file Stud.cpp.

6.8.2 Function Documentation

6.8.2.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Stud & stud)
```

Išvesties operatorius studento duomenims atvaizduoti.

Draugiškas operatorius, kuris išveda studentą į srautą.

Ši funkcija leidžia atvaizduoti visus studento duomenis naudojant << operatorių.

Parameters

os	Išvesties srautas, į kurį bus rašoma informacija apie studentą.
stud	Studentas, kurio informacija bus atvaizduota.

Returns

std::ostream& Nuoroda į išvesties srautą.

Definition at line 173 of file Stud.cpp.

```
00173                                                   {
00174     stud.atvaizduoti(os);// Kviečiama funkcija, kuri atvaizduoja studento duomenis
00175     return os;// Grąžinamas išvesties srautas
00176
00177 }
```

6.8.2.2 operator>>()

```
std::istream & operator>> (
            std::istream & is,
            Stud & stud)
```

Nuskaitymo operatorius studento duomenims įvesti iš srauto.

Draugiškas operatorius, kuris įveda studentą iš srauto.

Šis operatorius leidžia įvesti studento duomenis, tokius kaip vardas, pavardė, namų darbų rezultatai ir egzamino rezultatas. Duomenys gali būti įvedami tiek atsitiktinai, tiek rankiniu būdu.

Parameters

is	Įvesties srautas, iš kurio bus nuskaityti duomenys.
stud	Objektas, į kurį bus saugomi nuskaityti duomenys.

Returns

std::istream& Nuoroda į įvesties srautą.

Paprašo vartotojo įvesti vardą ir pavardę. Šie duomenys priskiriami studento objektui.

Nustato įvestą vardą ir pavardę studento objektui.

Patikrina, ar rezultatai turėtų būti generuojami atsitiktinai.

Inicijuojamas atsitiktinių skaičių generatorius. Naudojamas sugeneruoti namų darbų ir egzamino balams.

Generuoja nurodyto dydžio atsitiktinius namų darbų balus. Balai priskiriami studento objektui.

Sugeneruojamas atsitiktinis egzamino balas ir priskiriamas studentui.

Vartotojas įveda konkrečių namų darbų skaičių ir rezultatus. Visi rezultatai pridedami prie studento objekto

Naudotojas įveda dinamiškai namų darbų rezultatus. Įvedimas baigiamas įvedus -1. Kiekvienas rezultatas pridedamas prie studento objekto

Definition at line 48 of file Stud.cpp.

```
00048                                                   {
00049
00053
00054     cout « "Ivesti varda ir pavarde: ";// Prašome vartotojo įvesti vardą ir pavardę
00055     string vardas, pavarde;
00056     is » vardas » pavarde;// Nuskaitomas vardas ir pavarde
00057
00060
00061      // Nustatomas įvestas vardas ir pavarde studento objektui
00062     stud.setVardas(vardas);
00063     stud.setPavarde(pavarde);
00064
00065     cout « "Konstruktoriu: Objekto " « vardas « " " « pavarde « " sukurimas" « endl;
00066
00067     string pasirinkimas;
00068
00071
00072     cout « "Ar reikia namu darbu ir egzamino rezultatus generuoti atsitiktinai?(taip/ne) ";//
Patikriname, ar rezultatai turi būti generuojami atsitiktinai
00073     is » pasirinkimas;
00074
00075     if (pasirinkimas == "taip") {
00076
00080
00081          // Inicijuojamas atsitiktinių skaičių generatorius
```

```
00082         random_device rd;
00083         mt19937 gen(rd());
00084         uniform_int_distribution<> dist(1, 10);
00085 
00086         int ndCount;
00087         cout « "Ivesti namu darbu skaiciu: ";
00088         is » ndCount;// Nuskaityti namų darbų skaičių
00089 
00093 
00094          // Sugeneruojame nurodyto dydžio atsitiktinius namų darbų balus
00095         for (int i = 0; i < ndCount; ++i) {
00096             stud.addND(dist(gen));// Sugeneruojame ir pridedame atsitiktinį namų darbų balą
00097         }
00098 
00101 
00102         // Sugeneruojamas atsitiktinis egzamino balas ir priskiriamas studentui
00103         stud.setEgz(dist(gen));
00104         stud.apskaiciuotiGalutinius();// Apskaičiuojamas galutinis įvertinimas
00105 
00106     }
00107     else {
00108         cout « "Ar zinai, koks yra namu darbu skaicius?(taip/ne): ";
00109         is » pasirinkimas;
00110 
00111         if (pasirinkimas == "taip") {
00112 
00116 
00117              // Vartotojas įveda konkrečių namų darbų skaičių ir rezultatus
00118             int ndCount;
00119             cout « "Ivesti namu darbu skaiciu: ";
00120             is » ndCount;
00121 
00122 
00123 
00124             cout « "Ivesti namu darbu rezultatus(10-baleje sistemoje): ";
00125             for (int i = 0; i < ndCount; ++i) {
00126                 int nd;
00127                 is » nd;// Vartotojas įveda kiekvieną namų darbų rezultatą
00128                 stud.addND(nd);// Pridedamas rezultatas prie studento objekto
00129 
00130             }
00131         }
00132         else if (pasirinkimas == "ne") {
00133 
00138 
00139             // Naudotojas įveda dinamiškai namų darbų rezultatus
00140             double nd;
00141             cout « "Ivesti namu darbu rezultatus(ivesti '-1',kad baigti):" « endl;
00142             while (true) {
00143                 is » nd;// Vartotojas įveda kiekvieną namų darbų rezultatą
00144                 if (nd == -1) {// Baigiasi įvedimas, jei įvedama -1
00145                     break;
00146                 }
00147                 stud.addND(nd);// Pridedamas rezultatas prie studento objekto
00148             }
00149         }
00150 
00151         // Naudotojas įveda egzamino rezultatą ir priskiria jį studento objektui.
00152         cout « "Ivesti egzamino rezultata: ";
00153         double egz;
00154         is » egz; // Vartotojas įveda egzamino rezultatą
00155         stud.setEgz(egz);// Priskiriamas egzamino rezultatas studentui
00156 
00157 
00158     }
00159 
00160     return is;// Grąžinamas įvesties srautas
00161 
00162 }
```

6.8.2.3 val()

```
void val (
             Stud & Lok)
```

Nustato studento duomenis į pradinius (tuščius) reikšmes.

Parameters

Lok	Studentas, kurio duomenys bus nustatyti į pradinius.

Definition at line 185 of file Stud.cpp.

```
00185                   {
00186     Lok.setVardas("");// Nustatome vardą į tuščią
00187     Lok.setPavarde("");// Nustatome pavardę į tuščią
00188     Lok.setND({});// Nustatome namų darbų rezultatus į tuščią vektorių
00189 }
```

6.9 Stud.cpp

Go to the documentation of this file.

```
00001
00021 #include "Stud.h"
00022 #include "Mylib.h"
00023
00024
00032 void Stud::atvaizduoti(std::ostream& os) const {// Išvedame studento vardą, pavardę, galutinį
vidurkį ir galutinę medianą į išvesties srautą
00033     os « "Vardas:" « getVardas() « ", Pavarde:" « getPavarde() « ", GalutinisVid:" «
GalutinisVid
00034         « ", GalutinisMed:" « GalutinisMed « endl;
00035 };
00036
00037
00048 std::istream& operator»(std::istream& is, Stud& stud) {
00049
00054     cout « "Ivesti varda ir pavarde: ";// Prašome vartotojo įvesti vardą ir pavardę
00055     string vardas, pavarde;
00056     is » vardas » pavarde;// Nuskaitomas vardas ir pavarde
00057
00061      // Nustatomas įvestas vardas ir pavarde studento objektui
00062     stud.setVardas(vardas);
00063     stud.setPavarde(pavarde);
00064
00065     cout « "Konstruktoriu: Objekto " « vardas « " " « pavarde « " sukurimas" « endl;
00066
00067     string pasirinkimas;
00068
00072     cout « "Ar reikia namu darbu ir egzamino rezultatus generuoti atsitiktinai?(taip/ne) ";//
Patikriname, ar rezultatai turi būti generuojami atsitiktinai
00073     is » pasirinkimas;
00074
00075     if (pasirinkimas == "taip") {
00076
00081          // Inicijuojamas atsitiktinių skaičių generatorius
00082         random_device rd;
00083         mt19937 gen(rd());
00084         uniform_int_distribution<> dist(1, 10);
00085
00086         int ndCount;
00087         cout « "Ivesti namu darbu skaiciu: ";
00088         is » ndCount;// Nuskaityti namų darbų skaičių
00089
00094          // Sugeneruojame nurodyto dydžio atsitiktinius namų darbų balus
00095         for (int i = 0; i < ndCount; ++i) {
00096             stud.addND(dist(gen));// Sugeneruojame ir pridedame atsitiktinį namų darbų balą
00097         }
00098
00102         // Sugeneruojamas atsitiktinis egzamino balas ir priskiriamas studentui
00103         stud.setEgz(dist(gen));
00104         stud.apskaiciuotiGalutinius();// Apskaičiuojamas galutinis įvertinimas
00105
00106     }
00107     else {
00108         cout « "Ar zinai, koks yra namu darbu skaicius?(taip/ne): ";
00109         is » pasirinkimas;
00110
00111         if (pasirinkimas == "taip") {
00112
00117              // Vartotojas įveda konkrečių namų darbų skaičių ir rezultatus
00118             int ndCount;
00119             cout « "Ivesti namu darbu skaiciu: ";
00120             is » ndCount;
00121
00122
00123
00124             cout « "Ivesti namu darbu rezultatus(10-baleje sistemoje): ";
00125             for (int i = 0; i < ndCount; ++i) {
00126                 int nd;
00127                 is » nd;// Vartotojas įveda kiekvieną namų darbų rezultatą
```

```
00128                 stud.addND(nd);// Pridedamas rezultatas prie studento objekto
00129
00130             }
00131         }
00132         else if (pasirinkimas == "ne") {
00133
00139             // Naudotojas įveda dinamiškai namų darbų rezultatus
00140             double nd;
00141             cout « "Ivesti namu darbu rezultatus(ivesti '-1',kad baigti):" « endl;
00142             while (true) {
00143                 is » nd;// Vartotojas įveda kiekvieną namų darbų rezultatą
00144                 if (nd == -1) {// Baigiasi įvedimas, jei įvedama -1
00145                     break;
00146                 }
00147                 stud.addND(nd);// Pridedamas rezultatas prie studento objekto
00148             }
00149         }
00150
00151         // Naudotojas įveda egzamino rezultatą ir priskiria jį studento objektui.
00152         cout « "Ivesti egzamino rezultata: ";
00153         double egz;
00154         is » egz; // Vartotojas įveda egzamino rezultatą
00155         stud.setEgz(egz);// Priskiriamas egzamino rezultatas studentui
00156
00157
00158     }
00159
00160     return is;// Grąžinamas įvesties srautas
00161
00162 }
00163
00173 std::ostream& operator«(std::ostream& os, const Stud& stud) {
00174     stud.atvaizduoti(os);// Kviečiama funkcija, kuri atvaizduoja studento duomenis
00175     return os;// Grąžinamas išvesties srautas
00176
00177 }
00178
00179
00185 void val(Stud& Lok) {
00186     Lok.setVardas("");// Nustatome vardą į tuščią
00187     Lok.setPavarde("");// Nustatome pavardę į tuščią
00188     Lok.setND({});// Nustatome namų darbų rezultatus į tuščią vektorių
00189 }
```

6.10 Stud.h File Reference

Studentų klasės antraštinis failas.

```
#include "Mylib.h"
```

Classes

- class zmogus
  Klasė, kuri aprašo žmogų su vardu ir pavarde.
- class Stud
  Klasė, kuri aprašo studentą, paveldintį iš žmogaus.

Functions

- void nuskaitytiIsFailo (std::vector< Stud > &Vec1, const std::string &failoVardas)
  Nuskaityti studentus iš failo ir įrašyti juos į vektorių.
- void nuskaitytiIsfailo (std::list< Stud > &list1, const std::string &failoVardas)
  Nuskaityti studentus iš failo ir įrašyti juos į sąrašą.
- void sugeneruotiStudentoFaila (const std::string &fileName, int studentCount, int ndCount)
  Sugeneruoti studentų failą su atsitiktiniais duomenimis.
- void irasytiVargsiukusVector (const std::vector< Stud > &vargsiukai, const std::string &failo↩ Pavadinimas)
  Įrašyti "vargsiukus" į failą iš vektoriaus.
- void irasytivargsiukusList (const std::list< Stud > &vargsiukai, const std::string &failoPavadinimas)
  Įrašyti "vargsiukus" į failą iš sąrašo.

- void irasytikietiakiaiList (const std::list< Stud > &kietiakiai, const std::string &failoPavadinimas)
  Įrašyti "kietiakiai" į failą iš sąrašo.
- void irasytiKietiakiaiVector (const std::vector< Stud > &kietiakiai, const std::string &failoPavadinimas)
  Įrašyti "kietiakiai" į failą iš vektoriaus.

6.10.1 Detailed Description

Studentų klasės antraštinis failas.

Šiame faile aprašoma studentų duomenų struktūra, jos kintamieji ir pagrindiniai metodai, kurie bus naudojami „stud.cpp“ faile.

Šiame faile aprašomos dvi pagrindinės klasės:

- **Abstrakti klasė `zmogus`**, kuri aprašo žmogaus duomenis, įskaitant vardą ir pavardę.
- **Išvestinė klasė `Stud`**, kuri paveldi klasę `zmogus` ir praplečia ją pridėdama studentų specifinius duomenis ir metodus, tokius kaip namų darbų rezultatai, egzaminas ir galutiniai įvertinimai.

`zmogus` klasė:

- Tai **abstrakti klasė**, nes ji turi **grynąją virtualią funkciją** `atvaizduoti()`, kuri nėra įgyvendinta šioje klasėje ir turi būti įgyvendinta išvestinėse klasėse.
- Ši klasė aprašo bendrą žmogaus informaciją, tokią kaip vardas ir pavardė, ir pateikia metodus jų nustatymui bei gavimui.
- Klasė `zmogus` negali būti tiesiogiai instancijuojama, tačiau ji suteikia pagrindą kitoms klasėms, pvz., `Stud`, paveldėti ir įgyvendinti savo specifinius metodus.
- Naudojami operatoriai, leidžiantys lengvai įvesti ir išvesti žmogaus duomenis į/iš srauto.

`Stud` klasė:

- Tai **išvestinė klasė**, paveldinti klasę `zmogus`. Ji prideda papildomus duomenis ir funkcijas, susijusias su studentų rezultatais.
- Klasė turi metodus, skirtus apdoroti studentų namų darbų rezultatus (`addND()`), apskaičiuoti galutinius įvertinimus (vidurkį ir medianą), taip pat atvaizduoti studento informaciją (`atvaizduoti()`).
- `Stud` klasė įgyvendina grynąją virtualią funkciją `atvaizduoti()` iš bazinės klasės `zmogus`, todėl ši funkcija tampa specifinė studentui.
- Pateikiami operatoriai, leidžiantys įvesti ir išvesti studentų duomenis į/iš srauto.
- Klasė turi ir papildomas funkcijas, kaip duomenų kopijavimas, destruktorius, skirtas duomenų valymui, bei papildomos funkcijos studentų duomenims apdoroti.

Failas naudoja „mylib“ biblioteką.

Definition in file Stud.h.

6.10.2 Function Documentation

6.10.2.1 irasytikietiakiaiList()

```
void irasytikietiakiaiList (
            const std::list< Stud > & kietiakiai,
            const std::string & failoPavadinimas)
```

Įrašyti "kietiakiai" į failą iš sąrašo.

Parameters

kietiakiai	Kietiakiai (studentai).
failoPavadinimas	Failo pavadinimas.

6.10.2.2 irasytiKietiakiaiVector()

```
void irasytiKietiakiaiVector (
            const std::vector< Stud > & kietiakiai,
            const std::string & failoPavadinimas)
```

Įrašyti "kietiakiai" į failą iš vektoriaus.

Parameters

kietiakiai	Kietiakiai (studentai).
failoPavadinimas	Failo pavadinimas.

6.10.2.3 irasytivargsiukusList()

```
void irasytivargsiukusList (
            const std::list< Stud > & vargsiukai,
            const std::string & failoPavadinimas)
```

Įrašyti "vargsiukus" į failą iš sąrašo.

Parameters

vargsiukai	Vargsiukai (studentai).
failoPavadinimas	Failo pavadinimas.

6.10.2.4 irasytiVargsiukusVector()

```
void irasytiVargsiukusVector (
            const std::vector< Stud > & vargsiukai,
            const std::string & failoPavadinimas)
```

Įrašyti "vargsiukus" į failą iš vektoriaus.

Parameters

vargsiukai	Vargsiukai (studentai).
failoPavadinimas	Failo pavadinimas.

6.10.2.5 nuskaitytiIsFailo()

```
void nuskaitytiIsFailo (
            std::vector< Stud > & Vec1,
            const std::string & failoVardas)
```

Nuskaityti studentus iš failo ir įrašyti juos į vektorių.

Parameters

Vec1	Vektorius, kuriame bus saugomi studentai.
failoVardas	Failo pavadinimas.

Nuskaityti studentus iš failo ir įrašyti juos į vektorių.

Ši funkcija atidaro nurodytą failą, nuskaito kiekvieną eilutę ir iš jos sukuria studento objektą, kurio vardas, pavardė, namų darbai ir egzamino rezultatai yra įrašomi ir perduodami į `std::vector`. Jeigu duomenų eilutėje yra klaida arba trūksta reikalingų duomenų, studentas ignoruojamas.

Parameters

Vec1	Nuoroda į `std::vector`, kuriame bus saugomi nuskaityti studentų duomenys.
failoVardas	Failo, iš kurio bus nuskaityti studentų duomenys, pavadinimas.

< Atidarome failą su nurodytu pavadinimu

Definition at line 28 of file FailoNuskaitymas.cpp.

```
00028                                                                                {
00029     try {
00030         std::ifstream inFile(failoVardas);
00031         if (!inFile) {
00032             throw runtime_error("Nepavyko atidaryti failo:" + failoVardas);
00033         }
00034         string line;
00035         while (getline(inFile, line)) {
00036 
00037             std::stringstream ss(line);
00038             Stud temp;
00039             temp.clearData(); // Išvalome ankstesnius duomenis
00040 
00041             std::string vardas, pavardė;
00042             if (!(ss » vardas » pavardė)) {
00043                 std::cerr « "Nepavyko nuskaityti studento vardo ir pavardės" « endl;
00044                 continue; // Jei vardo ir pavardės nuskaityti nepavyko, pereiname prie kitos
eilutės
00045             }
00046 
00047             temp.setVardas(vardas);
00048             temp.setPavarde(pavardė);
00049 
00050             int nd;
00051             while (ss » nd) {
00052                 temp.addND(nd); // Pridedame namų darbų balus
00053             }
00054             if (temp.getND().size() < 1) {
00055                 continue; // Jei ND trūksta, ignoruojame studentą
00056             }
00057             temp.setEgz(temp.getND().back()); // Paskutinį namų darbų balą priskiriame egzamino
rezultatui
00058             temp.getND().pop_back();  // Pašaliname paskutinį ND, kad liktų tik namų darbų
rezultatai
00059 
00060             Vec1.push_back(temp); // Pridedame studentą į vektorių
00061         }
00062         inFile.close(); // Uždaromas failas po nuskaitymo
00063 
00064     }
00065     catch (const std::exception& e) {
00066         std::cerr « "Klaida: " « e.what() « endl; // Jei įvyko klaida, išvedame klaidos
pranešimą
00067     }
00068 }
```

6.10.2.6 nuskaitytilsfailo()

```
void nuskaitytiIsfailo (
            std::list< Stud > & listl,
            const std::string & failoVardas)
```

Nuskaityti studentus iš failo ir įrašyti juos į sąrašą.

Parameters

Vec1	Sąrašas, kuriame bus saugomi studentai.
failoVardas	Failo pavadinimas.

Nuskaityti studentus iš failo ir įrašyti juos į sąrašą.

Ši funkcija atidaro nurodytą failą, nuskaito kiekvieną eilutę ir iš jos sukuria studento objektą, kurio vardas, pavardė, namų darbai ir egzamino rezultatai yra įrašomi ir perduodami `std::list`. Jeigu duomenų eilutėje yra klaida arba trūksta reikalingų duomenų, studentas ignoruojamas.

Parameters

list1	Nuoroda į `std::list`, kuriame bus saugomi nuskaityti studentų duomenys.
failoVardas	Failo, iš kurio bus nuskaityti studentų duomenys, pavadinimas.

Definition at line 81 of file FailoNuskaitymas.cpp.

```
00081                                                                                {
00082     try {
00083         std::ifstream inFile(failoVardas); // Atidarome failą su nurodytu pavadinimu
00084         if (!inFile) {
00085             throw runtime_error("Nepavyko atidaryti failo:" + failoVardas);
00086         }
00087         string line;
00088         while (getline(inFile, line)) {
00089 
00090             std::stringstream ss(line);
00091             Stud temp;
00092             temp.clearData(); //Išvalome ankstesnius duomenis
00093 
00094             std::string vardas, pavardė;
00095             if (!(ss » vardas » pavardė)) {
00096                 std::cerr « "Nepavyko nuskaityti studento vardo ir pavardės" « endl;
00097                 continue; // Jei vardo ir pavardės nuskaityti nepavyko, pereiname prie kitos
eilutės
00098             }
00099             temp.setVardas(vardas);
00100             temp.setPavarde(pavardė);
00101 
00102             int nd;
00103             while (ss » nd) {
00104                 temp.addND(nd); // Pridedame namų darbų balus
00105             }
00106             if (temp.getND().size() < 1) {
00107                 continue; // Jei ND trūksta, ignoruojame studentą
00108             }
00109             temp.setEgz(temp.getND().back()); // Paskutinį namų darbų balą priskiriame egzamino
rezultatui
00110             temp.getND().pop_back();  // Pašaliname paskutinį ND, kad liktų tik namų darbų
rezultatai
00111 
00112             list1.push_back(temp); // Pridedame studentą į sąrašą
00113         }
00114         inFile.close(); // Uždaromas failas po nuskaitymo
00115 
00116     }
00117     catch (const std::exception& e) {
00118         std::cerr « "Klaida: " « e.what() « endl; // Jei įvyko klaida, išvedame klaidos
pranešimą
00119     }
00120 }
```

6.10.2.7 sugeneruotiStudentoFaila()

```
void sugeneruotiStudentoFaila (
            const std::string & fileName,
            int studentCount,
            int ndCount)
```

Sugeneruoti studentų failą su atsitiktiniais duomenimis.

Parameters

fileName	Failo pavadinimas.
studentCount	Kiek studentų bus sugeneruota.
ndCount	Kiek namų darbų bus sugeneruota kiekvienam studentui.

6.11 Stud.h

Go to the documentation of this file.

```
00001
00031 #ifndef ZMOGUS_H_INCLUDED
00032 #define ZMOGUS_H_INCLUDED
00033 #include "Mylib.h"
00034
00041 class zmogus {
00042 private:
00043     std::string vardas, pavarde;// Privatus kintamasis vardui ir pavardei saugoti
00044
00045 public:
00051     zmogus(std::string v = "", std::string p = "")// Konstruktorius su numatytomis reikšmėmis
00052         : vardas(v), pavarde(p) {}
00053
00057     virtual ~zmogus() = default;// Numatytoji destrukcija
00058
00063     std::string getVardas() const { return vardas; }// Vardo grąžinimo funkcija
00064
00069     std::string getPavarde() const { return pavarde; }// Pavardės grąžinimo funkcija
00070
00075     virtual void setVardas(const std::string& v) { vardas = v; }// Vardo nustatymo funkcija
00076
00077
00082     virtual void setPavarde(const std::string& p) { pavarde = p; } // Pavardės nustatymo
funkcija
00083
00088     virtual void atvaizduoti(std::ostream& os) const = 0;// Grynoji virtuali funkcija, kurią
turi implementuoti paveldėtos klasės
00089
00096     friend std::ostream& operator «(std::ostream& os, const zmogus& zmog) {
00097         zmog.atvaizduoti(os);// Kvieciama atvaizduoti funkcija
00098         return os;// Grąžina išvesties srautą su žmogaus duomenimis
00099     }
00100 };
00101
00108 class Stud : public zmogus {
00109 private:
00110     std::vector<int> ND; // Namų darbų rezultatai.
00111     double egz; // Egzamino rezultatas.
00112     double GalutinisVid; // Galutinis įvertinimas (vidurkis).
00113     double GalutinisMed; // Galutinis įvertinimas (mediana).
00114
00115 public:
00123     Stud(std::string v = "", std::string p = "", std::vector<int> nd = {}, double e = 0.0)//
Konstruktorius su parametrais
00124         : zmogus(v, p), ND(nd), egz(e), GalutinisVid(0), GalutinisMed(0) {}
00125
00130     Stud(const Stud& other) noexcept
00131         : zmogus(other.getVardas(), other.getPavarde()),// Kopijavimo inicializacija iš zmogus
klasės
00132         ND(other.ND),
00133         egz(other.egz),
00134         GalutinisVid(other.GalutinisVid),
00135         GalutinisMed(other.GalutinisMed) {}
00136
00142     Stud& operator = (const Stud& other) noexcept {
00143         if (this == &other) return *this;// Patikrinimas, kad neatsitiktų savitarpio priskyrimas
00144         zmogus::operator = (other);// Kviečiamas paveldėtas operatorius
00145         ND = other.ND;// Priskiriame namų darbų rezultatus
00146         egz = other.egz;// Priskiriame egzamino rezultatą
00147         GalutinisVid = other.GalutinisVid;// Priskiriame galutinį vidurkį
00148         GalutinisMed = other.GalutinisMed; // Priskiriame galutinę medianą
00149         return *this;// Grąžiname šį objektą
00150     }
00151
00156     void addND(int nd) {// Pridedame naują namų darbų rezultatą į vektorių
00157         ND.push_back(nd);
00158     }
00159
00163     ~Stud() { clearData(); }// Destruktorius, kuris išvalo studento duomenis
00164
00168     void clearData() {
00169         ND.clear();// Išvalome namų darbų rezultatus
00170         egz = 0.0; // Nustatome egzamino rezultatą į 0
00171         GalutinisVid = 0.0; // Nustatome galutinį vidurkį į 0
00172         GalutinisMed = 0.0; // Nustatome galutinę medianą į 0
00173     }
00174
00179     std::vector<int> getND() const { return ND; }
00180
00185     double getEgz() const { return egz; }
00186
00191     double getGalutinisVid() const { return GalutinisVid; }
00192
00197     double getGalutinisMed() const { return GalutinisMed; }
00198
00203     void setND(const std::vector<int>& nd) { ND = nd; }
00204
```

```
00209     void setEgz(double e) { egz = e; }
00210
00214     void apskaiciuotiGalutinius();// Apskaičiuoja galutinį įvertinimą (vidurkį ir medianą)
00215
00220     void atvaizduoti(std::ostream& os) const;// Atvaizduoja studento duomenis
00221
00228     friend std::istream& operator»(std::istream& is, Stud& stud);// Draugiškas operatorius
įvedimui
00229
00236     friend std::ostream& operator«(std::ostream& os, const Stud& stud);// Draugiškas operatorius
išvedimui
00237 };
00238
00244 void nuskaitytiIsFailo(std::vector<Stud>& Vec1, const std::string& failoVardas);// Nuskaityti
studentus iš failo į vektorių
00245
00251 void nuskaitytiIsfailo(std::list<Stud>& list1, const std::string& failoVardas);// Nuskaityti
studentus iš failo į sąrašą
00252
00259 void sugeneruotiStudentoFaila(const std::string& fileName, int studentCount, int ndCount); //
Sugeneruoti studentų failą su atsitiktiniais duomenimis
00260
00266 void irasytiVargsiukusVector(const std::vector<Stud>& vargsiukai, const std::string&
failoPavadinimas); // Įrašyti "vargsiukus" į failą iš vektoriaus
00267
00273 void irasytivargsiukusList(const std::list<Stud>& vargsiukai, const std::string&
failoPavadinimas); // Įrašyti "vargsiukus" į failą iš sąrašo
00274
00280 void irasytikietiakiaiList(const std::list<Stud>& kietiakiai, const std::string&
failoPavadinimas); // Įrašyti "kietiakiai" į failą iš sąrašo
00281
00287 void irasytiKietiakiaiVector(const std::vector<Stud>& kietiakiai, const std::string&
failoPavadinimas);// Įrašyti "kietiakiai" į failą iš vektoriaus
00288
00289 #endif // ZMOGUS_H_INCLUDED;
```

6.12 Studentu_duomenys.cpp File Reference

Pagrindinis programos vykdymo failas.

```
#include "Mylib.h"
#include "Stud.h"
```

Functions

- double apskaiciuotiMediana (std::vector< int > &nd)
  Apskaičiuoja studentų namų darbų mediana.
- void irasytiVargsiukusVector (const vector< Stud > &vargsiukai, const string &failoPavadinimas)
  Įrašo vargsiukų studentų duomenis į failą.
- void irasytiKietiakiaiVector (const vector< Stud > &kietiakiai, const string &failoPavadinimas)
  Įrašo kietiakiai studentų duomenis į failą.
- void irasytivargsiukusList (const list< Stud > &vargsiukai, const string &failoPavadinimas)
  Įrašo vargsiukų studentų duomenis į failą.
- void irasytikietiakiaiList (const list< Stud > &kietiakiai, const string &failoPavadinimas)
  Įrašo kietiakiai studentų duomenis į failą.
- int main ()
  Pagrindinė programa, skirta studentų duomenų nuskaitymui, apdorojimui ir rūšiavimui.

6.12.1 Detailed Description

Pagrindinis programos vykdymo failas.

Šis failas sujungia visas reikalingas bibliotekas ir paleidžia programą, koordinuodamas studentų duomenų apdorojimą. Jame atliekami veiksmai, tokie kaip atsitiktinių studentų duomenų generavimas, studentų duomenų nuskaitymas iš failo, jų apdorojimas ir galutinių įvertinimų apskaičiavimas.

Failas vykdo programos logiką, įskaitant šiuos pagrindinius veiksmus:

– Atsitiktinių studentų duomenų generavimas ir įrašymas į failą.

- Studentų duomenų nuskaitymas iš failo į atitinkamas duomenų struktūras.
- Studentų rezultatų apdorojimas, įskaitant galutinių įvertinimų (vidurkio ir medianos) apskaičiavimą.
- Studentų rezultatų klasifikavimas ir įrašymas į atskirus failus pagal jų pasiekimus.

Šis failas taip pat valdys vartotojo sąsają, leidžiančią pasirinkti norimus veiksmus, pavyzdžiui, generuoti duomenis, nuskaityti juos iš failo arba išvesti studentų rezultatus į failą.

Failas naudoja „mylib“ biblioteką.

Definition in file Studentu_duomenys.cpp.

6.12.2 Function Documentation

6.12.2.1 apskaiciuotiMediana()

```
double apskaiciuotiMediana (
            std::vector< int > & nd)
```

Apskaičiuoja studentų namų darbų mediana.

Ši funkcija rūšiuoja namų darbų rezultatus ir grąžina medianą. Jei yra lyginis namų darbų skaičius, grąžinama vidutinė vidurinių elementų reikšmė. Jei nelyginis, grąžinamas vidurinysis elementas.

Parameters

nd	Vektorius, kuriame saugomi studento namų darbų balai.

Returns

Grąžina mediana kaip double reikšmę.

Definition at line 38 of file Studentu_duomenys.cpp.

```
00038                                           {
00039     std::sort(nd.begin(), nd.end());//Rūšiuojame namų darbų balus.
00040     int n = nd.size();
00041     if (n % 2 == 0) {
00042         return(static_cast<double>(nd[n / 2 - 1]) + static_cast<double>(nd[n / 2])) / 2.0;
00043     }
00044     else {
00045         return static_cast<double>(nd[n / 2]);
00046     }
00047 }
```

6.12.2.2 irasytikietiakiaiList()

```
void irasytikietiakiaiList (
            const list< Stud > & kietiakiai,
            const string & failoPavadinimas)
```

Įrašo kietiakiai studentų duomenis į failą.

Ši funkcija įrašo studentų, kurių galutiniai balai yra aukšti, duomenis į nurodytą failą.

Parameters

kietiakiai	Sąrašas studentų, kurių galutiniai balai yra aukšti.
failoPavadinimas	Failo pavadinimas, į kurį bus įrašyti studentų duomenys.

Definition at line 171 of file Studentu_duomenys.cpp.

```
00171                                                                                   {
00172     ofstream failas(failoPavadinimas);
00173     if (failas.is_open()) {
00174         failas « setw(15) « left « "Vardas"
00175             « setw(15) « left « "Pavarde"
00176             « setw(20) « left « "Galutinis(Vid.)" « endl;
00177
```

```
00178         for (const auto& studentas : kietiakiai) {
00179             failas « setw(15) « left « studentas.getVardas()
00180                 « setw(15) « left « studentas.getPavarde()
00181                 « setw(20) « left « fixed « setprecision(2) « studentas.getGalutinisVid() «
endl;
00182         }
00183         failas.close();
00184
00185     }
00186     else {
00187         cout « "Nepavyko atidaryti failo: " « failoPavadinimas « endl;
00188     }
00189 }
```

6.12.2.3 irasytiKietiakiaiVector()

```
void irasytiKietiakiaiVector (
             const vector< Stud > & kietiakiai,
             const string & failoPavadinimas)
```

Įrašo kietiakiai studentų duomenis į failą.

Ši funkcija įrašo studentų, kurių galutiniai balai yra aukšti, duomenis į nurodytą failą.

Parameters

kietiakiai	Vektorius studentų, kurių galutiniai balai yra aukšti.
failoPavadinimas	Failo pavadinimas, į kurį bus įrašyti studentų duomenys.

Definition at line 113 of file Studentu_duomenys.cpp.

```
00113                                                                                                {
00114     ofstream failas(failoPavadinimas);
00115     if (failas.is_open()) {
00116         failas « setw(15) « left « "Vardas"
00117             « setw(15) « left « "Pavarde"
00118             « setw(20) « left « "Galutinis(Vid.)" « endl;
00119
00120         for (const auto& studentas : kietiakiai) {
00121             failas « setw(15) « left « studentas.getVardas()
00122                 « setw(15) « left « studentas.getPavarde()
00123                 « setw(20) « left « fixed « setprecision(2) « studentas.getGalutinisVid() «
endl;
00124         }
00125         failas.close();
00126
00127     }
00128     else {
00129         cout « "Nepavyko atidaryti failo: " « failoPavadinimas « endl;
00130     }
00131 }
```

6.12.2.4 irasytivargsiukusList()

```
void irasytivargsiukusList (
             const list< Stud > & vargsiukai,
             const string & failoPavadinimas)
```

Įrašo vargsiukų studentų duomenis į failą.

Ši funkcija įrašo studentų, kurių galutiniai balai yra žemesni, duomenis į nurodytą failą.

Parameters

vargsiukai	Sąrašas studentų, kurių galutiniai balai yra žemesni.
failoPavadinimas	Failo pavadinimas, į kurį bus įrašyti studentų duomenys.

Definition at line 141 of file Studentu_duomenys.cpp.

```
00141                                                                                                {
00142     ofstream failas(failoPavadinimas);
00143     if (failas.is_open()) {
```

```
00144         failas « setw(15) « left « "Vardas"
00145             « setw(15) « left « "Pavarde"
00146             « setw(20) « left « "Galutinis(Vid.)" « endl;
00147
00148
00149         for (const auto& studentas : vargsiukai) {
00150             failas « setw(15) « left « studentas.getVardas()
00151                 « setw(15) « left « studentas.getPavarde()
00152                 « setw(20) « left « fixed « setprecision(2) « studentas.getGalutinisVid() «
endl;
00153
00154         }
00155         failas.close();
00156
00157     }
00158     else {
00159         cout « "Nepavyko atidaryti failo: " « failoPavadinimas « endl;
00160     }
00161 }
```

6.12.2.5 irasytiVargsiukusVector()

```
void irasytiVargsiukusVector (
            const vector< Stud > & vargsiukai,
            const string & failoPavadinimas)
```

Įrašo vargsiukų studentų duomenis į failą.

Ši funkcija įrašo studentų, kurių galutiniai balai yra žemesni, duomenis į nurodytą failą.

Parameters

vargsiukai	Vektorius studentų, kurių galutiniai balai yra žemesni.
failoPavadinimas	Failo pavadinimas, į kurį bus įrašyti studentų duomenys.

Definition at line 83 of file Studentu_duomenys.cpp.

```
00083                                                                                                {
00084     ofstream failas(failoPavadinimas);
00085     if (failas.is_open()) {
00086         failas « setw(15) « left « "Vardas"
00087             « setw(15) « left « "Pavarde"
00088             « setw(20) « left « "Galutinis(Vid.)" « endl;
00089
00090
00091         for (const auto& studentas : vargsiukai) {
00092             failas « setw(15) « left « studentas.getVardas()
00093                 « setw(15) « left « studentas.getPavarde()
00094                 « setw(20) « left « fixed « setprecision(2) « studentas.getGalutinisVid() «
endl;
00095
00096         }
00097         failas.close();
00098
00099     }
00100     else {
00101         cout « "Nepavyko atidaryti failo: " « failoPavadinimas « endl;
00102     }
00103 }
```

6.12.2.6 main()

```
int main ()
```

Pagrindinė programa, skirta studentų duomenų nuskaitymui, apdorojimui ir rūšiavimui.

Ši funkcija leidžia vartotojui nuskaityti studentų duomenis iš failo, suskirstyti juos į grupes ir rūšiuoti pagal pasirinkimus. Vartotojas gali pasirinkti konteinerio tipą (vector arba list), rusiavimo kriterijų (pagal vardą, pavardę arba galutinį balą) ir studentų dalijimo strategiją. Laikai matuojami kiekvienam etapui.

Returns

0 Jei programa baigiasi sėkmingai.

< Laiko pradžia rūšiavimui

Įveda studentų duomenis ir prideda juos į sąrašą

Definition at line 201 of file Studentu_duomenys.cpp.

```
00202 {
00203
00204
00205     cout « "Ar norite sugeneruoti studentu failus?(taip/ne): ";
00206     string generuotiFailoPasirinkima; // Kintamasis vartotojo atsakymui saugoti
00207     cin » generuotiFailoPasirinkima;// Vartotojo atsakymo įvedimas
00208
00209     if (generuotiFailoPasirinkima == "taip") {// Jei vartotojas pasirinko "taip", generuojami
failai
00210         sugeneruotiStudentoFaila("studentai1000.txt", 1000, 5);
00211         sugeneruotiStudentoFaila("studentai10000.txt", 10000, 7);
00212         sugeneruotiStudentoFaila("studentai100000.txt", 100000, 6);
00213         sugeneruotiStudentoFaila("studentai1000000.txt", 1000000, 8);
00214         sugeneruotiStudentoFaila("studentai10000000.txt", 10000000, 4);
00215         cout « "Failai sugeneruoti!" « endl;//Informacija apie failų sugeneravimą
00216     }
00217
00218     vector<Stud> Vec1;// Vektorius studentų saugojimui
00219     Stud Temp;// Laikinas studento objektas
00220
00221     cout « "Ar norite ivesti studentu duomenis ar nuskaityti is failo?(ivesti/nuskaityti): ";
00222     string pasirinkimas;// Kintamasis vartotojo pasirinkimui saugoti
00223     cin » pasirinkimas;// Vartotojo atsakymo įvedimas
00224
00225
00226
00227     if (pasirinkimas == "nuskaityti") {// Jei pasirinktas duomenų nuskaitymas iš failo
00228
00229         cout « "Pasirinkti strategija(1 - pirmoji, 2 - antroji, 3 - trecioji): "; // Klausimas
apie strategiją
00230         int strategija;// Kintamasis strategijai saugoti
00231         cin » strategija;//Vartotojo pasirinkimo įvedimas
00232
00233         cout « "Pasirinkite konteinerio tipa? (1 - vector, 2 - list): "; //Klausimas apie
konteinerio tipą
00234         int konteinerioTipas;//Kintamasis konteinerio tipui saugoti
00235         cin » konteinerioTipas;//Vartotojo pasirinkimo įvedimas
00236
00237
00238         if (konteinerioTipas == 1) {// Jei pasirenkamas vektorius
00239             vector<Stud> Vec1;// Studentų vektorius
00240             cout « "Naudojamas vector." « endl;
00241
00242             cout « "Pasirinkite rusiavimo kriteriju(1 - pagal varda, 2 - pagal pavarde, 3 -
pagal galutini bala: ";// Klausimas apie rūšiavimą
00243             int rusiavimoKriterijus;// Kintamasis rūšiavimo kriterijui saugoti
00244             cin » rusiavimoKriterijus;// Vartotojo pasirinkimo įvedimas
00245
00246             auto pradziaNuskaitymui = std::chrono::high_resolution_clock::now();// Laiko pradžia
nuskaitymui
00247             nuskaitytiIsFailo(Vec1, "studentai1000.txt");
00248             auto pabaigaNuskaitymui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
nuskaitymui
00249
00250             cout « "Failas uzdarytas" « endl;// Informacija, kad failas uždarytas
00251             cout « "Failo is " « Vec1.size() « " irasu nuskaitymo laikas: "
00252                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaNuskaitymui -
pradziaNuskaitymui).count() « " s" « endl;
00253
00254             for (auto& studentas : Vec1) {// Iteracija per studentus
00255                 studentas.apskaiciuotiGalutinius();// Galutinio balo skaičiavimas kiekvienam
studentui
00256             }
00257
00258             auto pradziaRusiavimui = std::chrono::high_resolution_clock::now();
00259
00260             if (rusiavimoKriterijus == 1) {// Rūšiavimas pagal vardą
00261                 sort(Vec1.begin(), Vec1.end(), [](const Stud& a, const Stud& b) {
00262                     return a.getVardas() < b.getVardas();
00263                     });
00264             }
00265             else if (rusiavimoKriterijus == 2) {// Rūšiavimas pagal pavardę
00266                 sort(Vec1.begin(), Vec1.end(), [](const Stud& a, const Stud& b) {
00267                     return a.getPavarde() < b.getPavarde();
00268                     });
00269             }
00270             else if (rusiavimoKriterijus == 3) {// Rūšiavimas pagal galutinį balą
00271                 sort(Vec1.begin(), Vec1.end(), [](const Stud& a, const Stud& b) {
```

```
00272                     return a.getGalutinisVid() < b.getGalutinisVid();
00273                     });
00274             }
00275             else {
00276                 cout « "Netinkamas pasirinkimas. Rusiavimas pagal varda" « endl;//
Klaida,netinkamas pasirinkimas
00277                 sort(Vec1.begin(), Vec1.end(), [](const Stud& a, const Stud& b) {
00278                     return a.getVardas() < b.getVardas();// Numatytoji rūšiavimo funkcija pagal
vardą
00279                     });
00280             }
00281
00282             auto pabaigaRusiavimui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
rūšiavimui
00283             cout « Vec1.size() « "irasu rusiavimas didejimo tvarka, su sort funkcija: "
00284                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaRusiavimui -
pradziaRusiavimui).count() « " s " « endl;
00285
00286             vector<Stud> vargsiukai; //Sukuriamas vektorius vargsiukams saugoti
00287             vector<Stud> kietiakiai; // Sukuriamas vektorius kietiakams saugoti
00288
00289             auto pradziaDalijimui = std::chrono::high_resolution_clock::now();// Laiko pradžia
dalijimui į grupes
00290
00291             if (strategija == 1) {// Patikrinama, ar pasirinkta pirmoji dalijimo strategija
00292
00293                 for (auto& studentas : Vec1) {// Pereinama per visus studentus
00294                     if (studentas.getGalutinisVid() < 5.0) {
00295                         vargsiukai.push_back(studentas);// Jei galutinis vidurkis mažesnis už 5,
pridedama į vargsiukus
00296                     }
00297                     else {
00298                         kietiakiai.push_back(studentas);//Jei galutinis vidurkis didesnis arba
lygus 5, pridedama į kietiakiai
00299                     }
00300                 }
00301
00302             }
00303             else if (strategija == 2) {// Patikrinama, ar pasirinkta antroji dalijimo strategija

00304
00305                 auto it = std::remove_if(Vec1.begin(), Vec1.end(), [](const Stud& studentas) {
00306                     return studentas.getGalutinisVid() >= 5.0;//Pašalinami studentai, kurių
galutinis vidurkis yra 5 ar didesnis
00307                     });
00308
00309                 vargsiukai.assign(it, Vec1.end()); // Vargsiukai priskiriami tiems, kurių
vidurkis mažesnis nei 5
00310                 Vec1.erase(it, Vec1.end()); // Pašalinami studentai, kurių vidurkis 5 ar
didesnis
00311
00312             }
00313             else if (strategija == 3) {// Patikrinama, ar pasirinkta trečioji dalijimo
strategija
00314
00315
00316                 auto it = std::partition(Vec1.begin(), Vec1.end(), [](const Stud& studentas) {
00317                     return studentas.getGalutinisVid() >= 5.0;// Padalinama pagal tai, ar
galutinis vidurkis yra didesnis arba lygus 5
00318                     });
00319                 vargsiukai = vector<Stud>(it, Vec1.end()); //Vargsiukai priskiriami tiems, kurių
vidurkis mažesnis nei 5
00320                 Vec1.erase(it, Vec1.end()); // Pašalinami studentai, kurių vidurkis 5 ar
didesnis
00321             }
00322
00323
00324             auto pabaigaDalijimui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
dalijimui į grupes
00325             cout « Vec1.size() « " irasu dalijimo i dvi grupes laikas, panaikinant pradini
Vektor: "
00326                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaDalijimui -
pradziaDalijimui).count() « " s" « endl;
00327
00328             auto pradziaVargsiukams = std::chrono::high_resolution_clock::now(); // Laiko
pradžia vargsiukų įrašymui į failą
00329             irasytiVargsiukusVector(vargsiukai, "vargsiukai.txt");
00330             auto pabaigaVargsiukams = std::chrono::high_resolution_clock::now(); // Laiko
pabaiga vargsiukų įrašymui į failą
00331             cout « vargsiukai.size() « " irasu  vargsiuku irasymo i faila laikas: "
00332                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaVargsiukams -
pradziaVargsiukams).count() « " s" « endl;
00333
00334             auto pabaigaTesto = std::chrono::high_resolution_clock::now();// Laiko pabaiga
testui
00335             cout « endl « Vec1.size() « " irasu testo laikas: "
00336                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaTesto -
```

```
pradziaNuskaitymui).count() « " s" « endl;
00337
00338
00339         }
00340         else if (konteinerioTipas == 2) {// Patikrinama, ar pasirinktas konteinerio tipas yra
'list'
00341             list<Stud> list1; //Sukuriama 'list' konteinerio struktūra
00342             cout « "Naudojamas list." « endl;
00343
00344             cout « "Pasirinkite rusiavimo kriteriju(1 - pagal varda, 2 - pagal pavarde, 3 -
pagal galutini bala: ";
00345             int rusiavimoKriterijus;
00346             cin » rusiavimoKriterijus; // Vartotojas pasirenka rusiavimo kriterijų
00347
00348             auto pradziaNuskaitymui = std::chrono::high_resolution_clock::now();// Laiko pradžia
failo nuskaitymui
00349             nuskaitytiIsfailo(list1, "studentai1000.txt");
00350             auto pabaigaNuskaitymui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
failo nuskaitymui
00351
00352             cout « "Failas uzdarytas" « endl;
00353             cout « "Failo is " « list1.size() « " irasu nuskaitymo laikas: "
00354                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaNuskaitymui -
pradziaNuskaitymui).count() « " s" « endl;
00355
00356             for (auto& studentas : list1) { //Apskaičiuojama kiekvieno studento galutinė
vidurkio reikšmė
00357                 studentas.apskaiciuotiGalutinius();
00358             }
00359
00360             auto pradziaRusiavimui = std::chrono::high_resolution_clock::now();// Laiko pradžia
rūšiavimui
00361
00362             if (rusiavimoKriterijus == 1) { // Patikrinama, pagal kokį kriterijų rūšiuoti
00363                 list1.sort([](const Stud& a, const Stud& b) {
00364                     return a.getVardas() < b.getVardas();// Rūšiuojama pagal vardą
00365                     });
00366             }
00367             else if (rusiavimoKriterijus == 2) {
00368                 list1.sort([](const Stud& a, const Stud& b) {
00369                     return a.getPavarde() < b.getPavarde();// Rūšiuojama pagal pavardę
00370                     });
00371             }
00372             else if (rusiavimoKriterijus == 3) {
00373                 list1.sort([](const Stud& a, const Stud& b) {
00374                     return a.getGalutinisVid() < b.getGalutinisVid();// Rūšiuojama pagal
galutinį vidurkį
00375                     });
00376             }
00377             else {
00378                 cout « "Netinkamas pasirinkimas. Rusiavimas pagal varda" « endl;
00379                 list1.sort([](const Stud& a, const Stud& b) {
00380                     return a.getVardas() < b.getVardas();// Jei pasirinktas netinkamas
kriterijus, rūšiuojama pagal vardą
00381                     });
00382             }
00383
00384             auto pabaigaRusiavimui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
rūšiavimui
00385             cout « list1.size() « "irasu rusiavimas didejimo tvarka, su sort funkcija: "
00386                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaRusiavimui -
pradziaRusiavimui).count() « " s " « endl;
00387
00388
00389
00390             auto pradziaDalijimui = std::chrono::high_resolution_clock::now();// Laiko pradžia
dalijimui į grupes
00391             list<Stud> vargsiukai, kietiakiai;// Sukuriami du sąrašai vargsiukams ir kietiakams
00392             if (strategija == 1) {
00393
00394                 for (auto& studentas : list1) {//Pereinama per visus studentus sąraše
00395                     if (studentas.getGalutinisVid() < 5.0) {
00396                         vargsiukai.push_back(studentas);// Jei galutinis vidurkis mažesnis už 5,
pridedama į vargsiukus
00397                     }
00398                     else {
00399                         kietiakiai.push_back(studentas); //Jei galutinis vidurkis didesnis arba
lygus 5, pridedama į kietiakiai
00400                     }
00401                 }
00402
00403             }
00404             else if (strategija == 2) { // Patikrinama, ar pasirinkta antroji dalijimo
strategija
00405
00406                 auto it = std::remove_if(list1.begin(), list1.end(), [](const Stud& studentas) {
00407                     return studentas.getGalutinisVid() >= 5.0; //Pašalinami studentai, kurių
```

```
galutinis vidurkis yra 5 ar didesnis
00408                     });
00409
00410                 vargsiukai.assign(it, list1.end()); // Vargsiukai priskiriami tiems, kurių
vidurkis mažesnis nei 5
00411                 list1.erase(it, list1.end()); //Pašalinami studentai, kurių vidurkis 5 ar
didesnis
00412             }
00413             else if (strategija == 3) {// Patikrinama, ar pasirinkta trečioji dalijimo
strategija
00414
00415                 auto it = std::partition(list1.begin(), list1.end(), [](const Stud& studentas) {
00416                     return studentas.getGalutinisVid() >= 5.0;//Padalinama pagal tai, ar
galutinis vidurkis yra didesnis arba lygus 5
00417                     });
00418                 vargsiukai = list<Stud>(it, list1.end()); // Vargsiukai priskiriami tiems, kurių
vidurkis mažesnis nei 5
00419                 list1.erase(it, list1.end()); // Pašalinami studentai, kurių vidurkis 5 ar
didesnis
00420             }
00421
00422             auto pabaigaDalijimui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
dalijimui į grupes
00423             cout « list1.size() « " irasu dalijimo i dvi grupes laikas: "
00424                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaDalijimui -
pradziaDalijimui).count() « " s" « endl;
00425
00426             auto pradziaVargsiukams = std::chrono::high_resolution_clock::now(); // Laiko
pradžia vargsiukų įrašymui į failą
00427             irasytivargsiukusList(vargsiukai, "vargsiukai.txt");
00428             auto pabaigaVargsiukams = std::chrono::high_resolution_clock::now(); // Laiko
pabaiga vargsiukų įrašymui į failą
00429             cout « vargsiukai.size() « " irasu  vargsiuku irasymo i faila laikas: "
00430                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaVargsiukams -
pradziaVargsiukams).count() « " s" « endl;
00431
00432
00433
00434             auto pabaigaTesto = std::chrono::high_resolution_clock::now();// Laiko pabaiga
testui
00435             cout « endl « list1.size() « " irasu testo laikas: "
00436                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaTesto -
pradziaNuskaitymui).count() « " s" « endl;
00437
00438         }
00439
00440
00441     }
00442     else {
00443
00444         // Klausiama vartotojo, kiek studentų norima įvesti
00445         cout « "Kiek yra studentu ?";
00446         int n;//Studentų skaičius
00447         cin » n;
00448
00449         // Prašoma vartotojo pasirinkti konteinerį (vector/list)
00450         cout « "Pasirinkite konteineri(vector/list): ";
00451         string konteineris;//Konteinerio pasirinkimas
00452         cin » konteineris;
00453
00454
00455         vector<Stud> Vec1; //Vektorius studentams, jei pasirenkamas "vector"
00456         list<Stud> list1; // Sąrašas studentams, jei pasirenkamas "list"
00457
00458         if (konteineris == "vector") {//Jei pasirinktas vektorius
00459
00460             for (int i = 0; i < n; i++) {
00461                 Stud Temp;// Laikinas studento objektas
00462                 cout « "Please input user data: " « endl;
00463                 cin » Temp;
00464                 Temp.apskaiciuotiGalutinius();// Apskaičiuojami galutiniai balai
00465
00466                 Vec1.push_back(Temp); // Studentas pridedamas į vektorių
00467
00468             }
00469
00470             sort(Vec1.begin(), Vec1.end(), [](const Stud& a, const Stud& b) {
00471                 return a.getVardas() < b.getVardas(); // Rūšiuoja studentus pagal vardą
00472                 });
00473
00474             // Spausdina stulpelius su studentų informacija
00475             cout « setw(15) « left « "Vardas" « setw(15) « left « "Pavarde"
00476                 « setw(5) « right « "Galutinis(Vid.)"
00477                 « setw(5) « right « " Galutinis(Med.)" « endl;
00478
00479             cout « string(60, '-') « endl;
00480
```

```
00481             for (const auto& studentas : Vec1) {
00482                 cout « studentas « endl;//Spausdina studentų informaciją
00483             }
00484 
00485         }
00486         else if (konteineris == "list") {// Jei pasirinktas sąrašas
00487 
00489             for (int i = 0; i < n; i++) {
00490                 Stud Temp; // Laikinas studento objektas
00491                 cout « "Please input user data: " « endl;
00492                 cin » Temp;
00493                 Temp.apskaiciuotiGalutinius(); // Apskaičiuojami galutiniai balai
00494                 list1.push_back(Temp);// Studentas pridedamas į sąrašą
00495 
00496             }
00497 
00498             list1.sort([](const Stud& a, const Stud& b) {
00499                 return a.getVardas() < b.getVardas();// Rūšiuoja studentus pagal vardą
00500                 });
00501 
00502             // Spausdina stulpelius su studentų informacija
00503             cout « setw(15) « left « "Vardas" « setw(15) « left « "Pavarde"
00504                 « setw(5) « right « "Galutinis(Vid.)"
00505                 « setw(5) « right « " Galutinis(Med.)" « endl;
00506 
00507             cout « string(60, '-') « endl;
00508 
00509 
00510             for (const auto& studentas : list1) {
00511                 cout « studentas « endl; // Spausdina studentų informaciją
00512             }
00513         }
00514         else {
00515             cout « "Neteisingas konteinerio pasirinkimas." « endl;
00516         }
00517 
00518 
00519         cout « "Programos pabaiga, sunaikinami visi objektai" « endl;
00520 
00521         return 0;// Programos pabaiga
00522 
00523     }
00524 }
```

6.13 Studentu_duomenys.cpp

Go to the documentation of this file.

```
00001 
00025 #include "Mylib.h"
00026 #include "Stud.h"
00027 
00038 double apskaiciuotiMediana(std::vector<int>& nd) {
00039     std::sort(nd.begin(), nd.end());//Rūšiuojame namų darbų balus.
00040     int n = nd.size();
00041     if (n % 2 == 0) {
00042         return(static_cast<double>(nd[n / 2 - 1]) + static_cast<double>(nd[n / 2])) / 2.0;
00043     }
00044     else {
00045         return static_cast<double>(nd[n / 2]);
00046     }
00047 }
00048 
00057 void Stud::apskaiciuotiGalutinius() {
00058     if (ND.empty()) {
00059         cout « "Nd yra tuscias, negalima suskaiciuoti galutinio balo" « endl;
00060         GalutinisVid = 0.0;
00061         GalutinisMed = 0.0;
00062         return;
00063     }
00064     double vidutinis = 0.0;
00065     for (double nd : ND) {
00066         vidutinis += nd;// Skaičiuojame namų darbų vidurkį.
00067     }
00068     vidutinis /= ND.size();
00069     GalutinisVid = 0.4 * vidutinis + 0.6 * egz;
00070 
00071     double mediana = apskaiciuotiMediana(ND);// Skaičiuojame namų darbų medianą.
00072     GalutinisMed = 0.4 * mediana + 0.6 * egz;
00073 }
00074 
00083 void irasytiVargsiukusVector(const vector<Stud>& vargsiukai, const string& failoPavadinimas) {
00084     ofstream failas(failoPavadinimas);
00085     if (failas.is_open()) {
```

```
00086         failas « setw(15) « left « "Vardas"
00087             « setw(15) « left « "Pavarde"
00088             « setw(20) « left « "Galutinis(Vid.)" « endl;
00089 
00090 
00091         for (const auto& studentas : vargsiukai) {
00092             failas « setw(15) « left « studentas.getVardas()
00093                 « setw(15) « left « studentas.getPavarde()
00094                 « setw(20) « left « fixed « setprecision(2) « studentas.getGalutinisVid() «
endl;
00095 
00096         }
00097         failas.close();
00098 
00099     }
00100     else {
00101         cout « "Nepavyko atidaryti failo: " « failoPavadinimas « endl;
00102     }
00103 }
00104 
00113 void irasytiKietiakiaiVector(const vector<Stud>& kietiakiai, const string& failoPavadinimas) {
00114     ofstream failas(failoPavadinimas);
00115     if (failas.is_open()) {
00116         failas « setw(15) « left « "Vardas"
00117             « setw(15) « left « "Pavarde"
00118             « setw(20) « left « "Galutinis(Vid.)" « endl;
00119 
00120         for (const auto& studentas : kietiakiai) {
00121             failas « setw(15) « left « studentas.getVardas()
00122                 « setw(15) « left « studentas.getPavarde()
00123                 « setw(20) « left « fixed « setprecision(2) « studentas.getGalutinisVid() «
endl;
00124         }
00125         failas.close();
00126 
00127     }
00128     else {
00129         cout « "Nepavyko atidaryti failo: " « failoPavadinimas « endl;
00130     }
00131 }
00132 
00141 void irasytivargsiukusList(const list<Stud>& vargsiukai, const string& failoPavadinimas) {
00142     ofstream failas(failoPavadinimas);
00143     if (failas.is_open()) {
00144         failas « setw(15) « left « "Vardas"
00145             « setw(15) « left « "Pavarde"
00146             « setw(20) « left « "Galutinis(Vid.)" « endl;
00147 
00148 
00149         for (const auto& studentas : vargsiukai) {
00150             failas « setw(15) « left « studentas.getVardas()
00151                 « setw(15) « left « studentas.getPavarde()
00152                 « setw(20) « left « fixed « setprecision(2) « studentas.getGalutinisVid() «
endl;
00153 
00154         }
00155         failas.close();
00156 
00157     }
00158     else {
00159         cout « "Nepavyko atidaryti failo: " « failoPavadinimas « endl;
00160     }
00161 }
00162 
00171 void irasytikietiakiaiList(const list<Stud>& kietiakiai, const string& failoPavadinimas) {
00172     ofstream failas(failoPavadinimas);
00173     if (failas.is_open()) {
00174         failas « setw(15) « left « "Vardas"
00175             « setw(15) « left « "Pavarde"
00176             « setw(20) « left « "Galutinis(Vid.)" « endl;
00177 
00178         for (const auto& studentas : kietiakiai) {
00179             failas « setw(15) « left « studentas.getVardas()
00180                 « setw(15) « left « studentas.getPavarde()
00181                 « setw(20) « left « fixed « setprecision(2) « studentas.getGalutinisVid() «
endl;
00182         }
00183         failas.close();
00184 
00185     }
00186     else {
00187         cout « "Nepavyko atidaryti failo: " « failoPavadinimas « endl;
00188     }
00189 }
00190 
00191 
00201 int main()
```

```
00202 {
00203
00204
00205     cout « "Ar norite sugeneruoti studentu failus?(taip/ne): ";
00206     string generuotiFailoPasirinkima; // Kintamasis vartotojo atsakymui saugoti
00207     cin » generuotiFailoPasirinkima;// Vartotojo atsakymo įvedimas
00208
00209     if (generuotiFailoPasirinkima == "taip") {// Jei vartotojas pasirinko "taip", generuojami
failai
00210         sugeneruotiStudentoFaila("studentai1000.txt", 1000, 5);
00211         sugeneruotiStudentoFaila("studentai10000.txt", 10000, 7);
00212         sugeneruotiStudentoFaila("studentai100000.txt", 100000, 6);
00213         sugeneruotiStudentoFaila("studentai1000000.txt", 1000000, 8);
00214         sugeneruotiStudentoFaila("studentai10000000.txt", 10000000, 4);
00215         cout « "Failai sugeneruoti!" « endl;//Informacija apie failų sugeneravimą
00216     }
00217
00218     vector<Stud> Vec1;// Vektorius studentų saugojimui
00219     Stud Temp;// Laikinas studento objektas
00220
00221     cout « "Ar norite ivesti studentu duomenis ar nuskaityti is failo?(ivesti/nuskaityti): ";
00222     string pasirinkimas;// Kintamasis vartotojo pasirinkimui saugoti
00223     cin » pasirinkimas;// Vartotojo atsakymo įvedimas
00224
00225
00226
00227     if (pasirinkimas == "nuskaityti") {// Jei pasirinktas duomenų nuskaitymas iš failo
00228
00229         cout « "Pasirinkti strategija(1 - pirmoji, 2 - antroji, 3 - trecioji): "; // Klausimas
apie strategiją
00230         int strategija;// Kintamasis strategijai saugoti
00231         cin » strategija;//Vartotojo pasirinkimo įvedimas
00232
00233         cout « "Pasirinkite konteinerio tipa? (1 - vector, 2 - list): "; //Klausimas apie
konteinerio tipą
00234         int konteinerioTipas;//Kintamasis konteinerio tipui saugoti
00235         cin » konteinerioTipas;//Vartotojo pasirinkimo įvedimas
00236
00237
00238         if (konteinerioTipas == 1) {// Jei pasirenkamas vektorius
00239             vector<Stud> Vec1;// Studentų vektorius
00240             cout « "Naudojamas vector." « endl;
00241
00242             cout « "Pasirinkite rusiavimo kriteriju(1 - pagal varda, 2 - pagal pavarde, 3 -
pagal galutini bala: ";// Klausimas apie rūšiavimą
00243             int rusiavimoKriterijus;// Kintamasis rūšiavimo kriterijui saugoti
00244             cin » rusiavimoKriterijus;// Vartotojo pasirinkimo įvedimas
00245
00246             auto pradziaNuskaitymui = std::chrono::high_resolution_clock::now();// Laiko pradžia
nuskaitymui
00247             nuskaitytiIsFailo(Vec1, "studentai1000.txt");
00248             auto pabaigaNuskaitymui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
nuskaitymui
00249
00250             cout « "Failas uzdarytas" « endl;// Informacija, kad failas uždarytas
00251             cout « "Failo is " « Vec1.size() « " irasu nuskaitymo laikas: "
00252                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaNuskaitymui -
pradziaNuskaitymui).count() « " s" « endl;
00253
00254             for (auto& studentas : Vec1) {// Iteracija per studentus
00255                 studentas.apskaiciuotiGalutinius();// Galutinio balo skaičiavimas kiekvienam
studentui
00256             }
00257
00258             auto pradziaRusiavimui = std::chrono::high_resolution_clock::now();
00260             if (rusiavimoKriterijus == 1) {// Rūšiavimas pagal vardą
00261                 sort(Vec1.begin(), Vec1.end(), [](const Stud& a, const Stud& b) {
00262                     return a.getVardas() < b.getVardas();
00263                     });
00264             }
00265             else if (rusiavimoKriterijus == 2) {// Rūšiavimas pagal pavardę
00266                 sort(Vec1.begin(), Vec1.end(), [](const Stud& a, const Stud& b) {
00267                     return a.getPavarde() < b.getPavarde();
00268                     });
00269             }
00270             else if (rusiavimoKriterijus == 3) {// Rūšiavimas pagal galutinį balą
00271                 sort(Vec1.begin(), Vec1.end(), [](const Stud& a, const Stud& b) {
00272                     return a.getGalutinisVid() < b.getGalutinisVid();
00273                     });
00274             }
00275             else {
00276                 cout « "Netinkamas pasirinkimas. Rusiavimas pagal varda" « endl;//
Klaida,netinkamas pasirinkimas
00277                 sort(Vec1.begin(), Vec1.end(), [](const Stud& a, const Stud& b) {
00278                     return a.getVardas() < b.getVardas();// Numatytoji rūšiavimo funkcija pagal
vardą
00279                     });
```

```
00280             }
00281
00282             auto pabaigaRusiavimui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
rūšiavimui
00283             cout « Vec1.size() « "irasu rusiavimas didejimo tvarka, su sort funkcija: "
00284                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaRusiavimui -
pradziaRusiavimui).count() « " s " « endl;
00285
00286             vector<Stud> vargsiukai; //Sukuriamas vektorius vargsiukams saugoti
00287             vector<Stud> kietiakiai; // Sukuriamas vektorius kietiakams saugoti
00288
00289             auto pradziaDalijimui = std::chrono::high_resolution_clock::now();// Laiko pradžia
dalijimui į grupes
00290
00291             if (strategija == 1) {// Patikrinama, ar pasirinkta pirmoji dalijimo strategija
00292
00293                 for (auto& studentas : Vec1) {// Pereinama per visus studentus
00294                     if (studentas.getGalutinisVid() < 5.0) {
00295                         vargsiukai.push_back(studentas);// Jei galutinis vidurkis mažesnis už 5,
pridedama į vargsiukus
00296                     }
00297                     else {
00298                         kietiakiai.push_back(studentas);//Jei galutinis vidurkis didesnis arba
lygus 5, pridedama į kietiakiai
00299                     }
00300                 }
00301
00302             }
00303             else if (strategija == 2) {// Patikrinama, ar pasirinkta antroji dalijimo strategija

00304
00305                 auto it = std::remove_if(Vec1.begin(), Vec1.end(), [](const Stud& studentas) {
00306                     return studentas.getGalutinisVid() >= 5.0;//Pašalinami studentai, kurių
galutinis vidurkis yra 5 ar didesnis
00307                     });
00308
00309                 vargsiukai.assign(it, Vec1.end()); // Vargsiukai priskiriami tiems, kurių
vidurkis mažesnis nei 5
00310                 Vec1.erase(it, Vec1.end()); // Pašalinami studentai, kurių vidurkis 5 ar
didesnis
00311
00312             }
00313             else if (strategija == 3) {// Patikrinama, ar pasirinkta trečioji dalijimo
strategija
00314
00315
00316                 auto it = std::partition(Vec1.begin(), Vec1.end(), [](const Stud& studentas) {
00317                     return studentas.getGalutinisVid() >= 5.0;// Padalinama pagal tai, ar
galutinis vidurkis yra didesnis arba lygus 5
00318                     });
00319                 vargsiukai = vector<Stud>(it, Vec1.end()); //Vargsiukai priskiriami tiems, kurių
vidurkis mažesnis nei 5
00320                 Vec1.erase(it, Vec1.end()); // Pašalinami studentai, kurių vidurkis 5 ar
didesnis
00321             }
00322
00323
00324             auto pabaigaDalijimui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
dalijimui į grupes
00325             cout « Vec1.size() « " irasu dalijimo i dvi grupes laikas, panaikinant pradini
Vektor: "
00326                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaDalijimui -
pradziaDalijimui).count() « " s" « endl;
00327
00328             auto pradziaVargsiukams = std::chrono::high_resolution_clock::now(); // Laiko
pradžia vargsiukų įrašymui į failą
00329             irasytiVargsiukusVector(vargsiukai, "vargsiukai.txt");
00330             auto pabaigaVargsiukams = std::chrono::high_resolution_clock::now(); // Laiko
pabaiga vargsiukų įrašymui į failą
00331             cout « vargsiukai.size() « " irasu  vargsiuku irasymo i faila laikas: "
00332                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaVargsiukams -
pradziaVargsiukams).count() « " s" « endl;
00333
00334             auto pabaigaTesto = std::chrono::high_resolution_clock::now();// Laiko pabaiga
testui
00335             cout « endl « Vec1.size() « " irasu testo laikas: "
00336                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaTesto -
pradziaNuskaitymui).count() « " s" « endl;
00337
00338
00339         }
00340         else if (konteinerioTipas == 2) {// Patikrinama, ar pasirinktas konteinerio tipas yra
'list'
00341             list<Stud> list1; //Sukuriama 'list' konteinerio struktūra
00342             cout « "Naudojamas list." « endl;
00343
00344             cout « "Pasirinkite rusiavimo kriteriju(1 - pagal varda, 2 - pagal pavarde, 3 -
```

```
pagal galutini bala: ";
00345             int rusiavimoKriterijus;
00346             cin » rusiavimoKriterijus; // Vartotojas pasirenka rusiavimo kriterijų
00347
00348             auto pradziaNuskaitymui = std::chrono::high_resolution_clock::now();// Laiko pradžia
failo nuskaitymui
00349             nuskaitytiIsfailo(list1, "studentai1000.txt");
00350             auto pabaigaNuskaitymui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
failo nuskaitymui
00351
00352             cout « "Failas uzdarytas" « endl;
00353             cout « "Failo is " « list1.size() « " irasu nuskaitymo laikas: "
00354                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaNuskaitymui -
pradziaNuskaitymui).count() « " s" « endl;
00355
00356             for (auto& studentas : list1) { //Apskaičiuojama kiekvieno studento galutinė
vidurkio reikšmė
00357                 studentas.apskaiciuotiGalutinius();
00358             }
00359
00360             auto pradziaRusiavimui = std::chrono::high_resolution_clock::now();// Laiko pradžia
rūšiavimui
00361
00362             if (rusiavimoKriterijus == 1) { // Patikrinama, pagal kokį kriterijų rūšiuoti
00363                 list1.sort([](const Stud& a, const Stud& b) {
00364                     return a.getVardas() < b.getVardas();// Rūšiuojama pagal vardą
00365                     });
00366             }
00367             else if (rusiavimoKriterijus == 2) {
00368                 list1.sort([](const Stud& a, const Stud& b) {
00369                     return a.getPavarde() < b.getPavarde();// Rūšiuojama pagal pavardę
00370                     });
00371             }
00372             else if (rusiavimoKriterijus == 3) {
00373                 list1.sort([](const Stud& a, const Stud& b) {
00374                     return a.getGalutinisVid() < b.getGalutinisVid();// Rūšiuojama pagal
galutinį vidurkį
00375                     });
00376             }
00377             else {
00378                 cout « "Netinkamas pasirinkimas. Rusiavimas pagal varda" « endl;
00379                 list1.sort([](const Stud& a, const Stud& b) {
00380                     return a.getVardas() < b.getVardas();// Jei pasirinktas netinkamas
kriterijus, rūšiuojama pagal vardą
00381                     });
00382             }
00383
00384             auto pabaigaRusiavimui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
rūšiavimui
00385             cout « list1.size() « "irasu rusiavimas didejimo tvarka, su sort funkcija: "
00386                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaRusiavimui -
pradziaRusiavimui).count() « " s " « endl;
00387
00388
00389
00390             auto pradziaDalijimui = std::chrono::high_resolution_clock::now();// Laiko pradžia
dalijimui į grupes
00391             list<Stud> vargsiukai, kietiakiai;// Sukuriami du sąrašai vargsiukams ir kietiakams
00392             if (strategija == 1) {
00393
00394                 for (auto& studentas : list1) {//Pereinama per visus studentus sąraše
00395                     if (studentas.getGalutinisVid() < 5.0) {
00396                         vargsiukai.push_back(studentas);// Jei galutinis vidurkis mažesnis už 5,
pridedama į vargsiukus
00397                     }
00398                     else {
00399                         kietiakiai.push_back(studentas); //Jei galutinis vidurkis didesnis arba
lygus 5, pridedama į kietiakiai
00400                     }
00401                 }
00402
00403             }
00404             else if (strategija == 2) { // Patikrinama, ar pasirinkta antroji dalijimo
strategija
00405
00406                 auto it = std::remove_if(list1.begin(), list1.end(), [](const Stud& studentas) {
00407                     return studentas.getGalutinisVid() >= 5.0; //Pašalinami studentai, kurių
galutinis vidurkis yra 5 ar didesnis
00408                     });
00409
00410                 vargsiukai.assign(it, list1.end()); // Vargsiukai priskiriami tiems, kurių
vidurkis mažesnis nei 5
00411                 list1.erase(it, list1.end()); //Pašalinami studentai, kurių vidurkis 5 ar
didesnis
00412             }
00413             else if (strategija == 3) {// Patikrinama, ar pasirinkta trečioji dalijimo
strategija
```

```
00414
00415                 auto it = std::partition(list1.begin(), list1.end(), [](const Stud& studentas) {
00416                     return studentas.getGalutinisVid() >= 5.0;//Padalinama pagal tai, ar
galutinis vidurkis yra didesnis arba lygus 5
00417                     });
00418                 vargsiukai = list<Stud>(it, list1.end()); // Vargsiukai priskiriami tiems, kurių
vidurkis mažesnis nei 5
00419                 list1.erase(it, list1.end()); // Pašalinami studentai, kurių vidurkis 5 ar
didesnis
00420             }
00421
00422             auto pabaigaDalijimui = std::chrono::high_resolution_clock::now();// Laiko pabaiga
dalijimui į grupes
00423             cout « list1.size() « " irasu dalijimo i dvi grupes laikas: "
00424                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaDalijimui -
pradziaDalijimui).count() « " s" « endl;
00425
00426             auto pradziaVargsiukams = std::chrono::high_resolution_clock::now(); // Laiko
pradžia vargsiukų įrašymui į failą
00427             irasytivargsiukusList(vargsiukai, "vargsiukai.txt");
00428             auto pabaigaVargsiukams = std::chrono::high_resolution_clock::now(); // Laiko
pabaiga vargsiukų įrašymui į failą
00429             cout « vargsiukai.size() « " irasu  vargsiuku irasymo i faila laikas: "
00430                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaVargsiukams -
pradziaVargsiukams).count() « " s" « endl;
00431
00432
00433
00434             auto pabaigaTesto = std::chrono::high_resolution_clock::now();// Laiko pabaiga
testui
00435             cout « endl « list1.size() « " irasu testo laikas: "
00436                 « fixed « setprecision(5) « std::chrono::duration<double>(pabaigaTesto -
pradziaNuskaitymui).count() « " s" « endl;
00437
00438         }
00439
00440
00441     }
00442     else {
00443
00444         // Klausiama vartotojo, kiek studentų norima įvesti
00445         cout « "Kiek yra studentu ?";
00446         int n;//Studentų skaičius
00447         cin » n;
00448
00449         // Prašoma vartotojo pasirinkti konteinerį (vector/list)
00450         cout « "Pasirinkite konteineri(vector/list): ";
00451         string konteineris;//Konteinerio pasirinkimas
00452         cin » konteineris;
00453
00454
00455         vector<Stud> Vec1; //Vektorius studentams, jei pasirenkamas "vector"
00456         list<Stud> list1; // Sąrašas studentams, jei pasirenkamas "list"
00457
00458         if (konteineris == "vector") {//Jei pasirinktas vektorius
00459
00460             for (int i = 0; i < n; i++) {
00461                 Stud Temp;// Laikinas studento objektas
00462                 cout « "Please input user data: " « endl;
00463                 cin » Temp;
00464                 Temp.apskaiciuotiGalutinius();// Apskaičiuojami galutiniai balai
00465
00466                 Vec1.push_back(Temp); // Studentas pridedamas į vektorių
00467
00468             }
00469
00470             sort(Vec1.begin(), Vec1.end(), [](const Stud& a, const Stud& b) {
00471                 return a.getVardas() < b.getVardas(); // Rūšiuoja studentus pagal vardą
00472                 });
00473
00474             // Spausdina stulpelius su studentų informacija
00475             cout « setw(15) « left « "Vardas" « setw(15) « left « "Pavarde"
00476                 « setw(5) « right « "Galutinis(Vid.)"
00477                 « setw(5) « right « " Galutinis(Med.)" « endl;
00478
00479             cout « string(60, '-') « endl;
00480
00481             for (const auto& studentas : Vec1) {
00482                 cout « studentas « endl;//Spausdina studentų informaciją
00483             }
00484
00485         }
00486         else if (konteineris == "list") {// Jei pasirinktas sąrašas
00487
00489             for (int i = 0; i < n; i++) {
00490                 Stud Temp; // Laikinas studento objektas
00491                 cout « "Please input user data: " « endl;
```

```
00492                 cin » Temp;
00493                 Temp.apskaiciuotiGalutinius(); // Apskaičiuojami galutiniai balai
00494                 list1.push_back(Temp);// Studentas pridedamas į sąrašą
00495 
00496             }
00497 
00498             list1.sort([](const Stud& a, const Stud& b) {
00499                 return a.getVardas() < b.getVardas();// Rūšiuoja studentus pagal vardą
00500                 });
00501 
00502             // Spausdina stulpelius su studentų informacija
00503             cout « setw(15) « left « "Vardas" « setw(15) « left « "Pavarde"
00504                 « setw(5) « right « "Galutinis(Vid.)"
00505                 « setw(5) « right « " Galutinis(Med.)" « endl;
00506 
00507             cout « string(60, '-') « endl;
00508 
00509 
00510             for (const auto& studentas : list1) {
00511                 cout « studentas « endl; // Spausdina studentų informaciją
00512             }
00513         }
00514         else {
00515             cout « "Neteisingas konteinerio pasirinkimas." « endl;
00516         }
00517 
00518 
00519         cout « "Programos pabaiga, sunaikinami visi objektai" « endl;
00520 
00521         return 0;// Programos pabaiga
00522 
00523     }
00524 }
00525 
```

Index

~Stud
    Stud, 15
~zmogus
    zmogus, 21

addND
    Stud, 15
apskaiciuotiGalutinius
    Stud, 15
apskaiciuotiMediana
    Studentu_duomenys.cpp, 43
atvaizduoti
    Stud, 15
    zmogus, 22

clearData
    Stud, 16

FailoGeneravimas.cpp, 25
    sugeneruotiStudentoFaila, 25
FailoNuskaitymas.cpp, 27
    nuskaitytiIsFailo, 28
    nuskaitytiIsfailo, 28

getEgz
    Stud, 16
getGalutinisMed
    Stud, 16
getGalutinisVid
    Stud, 16
getND
    Stud, 17
getPavarde
    zmogus, 23
getVardas
    zmogus, 23

irasytikietiakiaiList
    Stud.h, 37
    Studentu_duomenys.cpp, 43
irasytiKietiakiaiVector
    Stud.h, 37
    Studentu_duomenys.cpp, 44
irasytivargsiukusList
    Stud.h, 38
    Studentu_duomenys.cpp, 44
irasytiVargsiukusVector
    Stud.h, 38
    Studentu_duomenys.cpp, 45

main
    Studentu_duomenys.cpp, 45
Mylib.h, 31

nuskaitytiIsFailo
    FailoNuskaitymas.cpp, 28
    Stud.h, 38
nuskaitytiIsfailo
    FailoNuskaitymas.cpp, 28
    Stud.h, 39

operator<<
    Stud, 18
    Stud.cpp, 32
    zmogus, 24
operator>>
    Stud, 18
    Stud.cpp, 33
operator=
    Stud, 17

readme, 1
readme.md, 32

setEgz
    Stud, 17
setND
    Stud, 17
setPavarde
    zmogus, 23
setVardas
    zmogus, 23
Stud, 13
    ~Stud, 15
    addND, 15
    apskaiciuotiGalutinius, 15
    atvaizduoti, 15
    clearData, 16
    getEgz, 16
    getGalutinisMed, 16
    getGalutinisVid, 16
    getND, 17
    operator<<, 18
    operator>>, 18
    operator=, 17
    setEgz, 17
    setND, 17
    Stud, 14
Stud.cpp, 32
    operator<<, 32
    operator>>, 33

val, 34
Stud.h, 36
irasytikietiakiaiList, 37
irasytiKietiakiaiVector, 37
irasytivargsiukusList, 38
irasytiVargsiukusVector, 38
nuskaitytiIsFailo, 38
nuskaitytiIsfailo, 39
sugeneruotiStudentoFaila, 40
Studentu_duomenys.cpp, 42
apskaiciuotiMediana, 43
irasytikietiakiaiList, 43
irasytiKietiakiaiVector, 44
irasytivargsiukusList, 44
irasytiVargsiukusVector, 45
main, 45
sugeneruotiStudentoFaila
FailoGeneravimas.cpp, 25
Stud.h, 40

val
Stud.cpp, 34

zmogus, 20
∼zmogus, 21
atvaizduoti, 22
getPavarde, 23
getVardas, 23
operator<<, 24
setPavarde, 23
setVardas, 23
zmogus, 21