

Galutinis balas

Generated by Doxygen 1.13.2

1 2.0 versija	1
1.1 Praeitų versijų aprašymai	1
1.2 Platesnis programos aprašymas	1
1.2.1 Failų generavimas:	1
1.2.2 Studentų rūšiavimas:	2
1.2.3 Matuojamas laikas:	2
1.3 Kaip Paleisti Programą	2
1.4 Naudojimosi instrukcija	2
1.4.1 Duomenų išvedimas/jvedimas	3
1.5 Testavimo sistemos parametrai	3
1.6 Spartos Testavimo Rezultatai	3
1.6.1 vector	3
1.6.2 deque	3
1.6.3 list	4
1.7 Strategijos testavimo rezultatai	4
1.7.1 vector	4
1.7.2 deque	4
1.7.3 list	4
1.8 Struktūros ir klasės testavimo rezultatai	5
1.8.1 klasė	5
1.8.2 struktūra	5
1.9 Kompiliatoriaus flag'ų testavimo rezultatai	5
1.9.1 Klasė	5
1.9.2 Struktūra	6
1.10 Nuotraukos	6
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 Studentas Class Reference	13
5.1.1 Detailed Description	14
5.1.2 Constructor & Destructor Documentation	14
5.1.2.1 Studentas() [1/4]	14
5.1.2.2 Studentas() [2/4]	14
5.1.2.3 Studentas() [3/4]	15
5.1.2.4 Studentas() [4/4]	15
5.1.2.5 ~Studentas()	15

5.1.3 Member Function Documentation	15
5.1.3.1 clear()	15
5.1.3.2 getEgz()	16
5.1.3.3 getGalM()	16
5.1.3.4 getGalV()	16
5.1.3.5 getNd()	16
5.1.3.6 getPavarde()	16
5.1.3.7 getVardas()	16
5.1.3.8 operator=() [1/2]	16
5.1.3.9 operator=() [2/2]	17
5.1.3.10 setPaz()	17
5.1.3.11 setVarPav()	17
5.1.3.12 SkaiciuotiM()	18
5.1.3.13 SkaiciuotiV()	18
5.1.4 Friends And Related Symbol Documentation	18
5.1.4.1 operator<<	18
5.1.4.2 operator>>	18
5.2 Timer Class Reference	19
5.2.1 Detailed Description	19
5.2.2 Constructor & Destructor Documentation	19
5.2.2.1 Timer()	19
5.2.3 Member Function Documentation	20
5.2.3.1 elapsed()	20
5.2.3.2 reset()	20
5.3 Zmogus Class Reference	20
5.3.1 Detailed Description	21
5.3.2 Constructor & Destructor Documentation	21
5.3.2.1 Zmogus() [1/2]	21
5.3.2.2 Zmogus() [2/2]	21
5.3.2.3 ~Zmogus()	21
5.3.3 Member Function Documentation	21
5.3.3.1 getPavarde()	21
5.3.3.2 getVardas()	21
5.3.3.3 setVarPav()	22
5.3.4 Member Data Documentation	22
5.3.4.1 pavarde	22
5.3.4.2 vardas	22
6 File Documentation	23
6.1 funkcijos.cpp File Reference	23
6.1.1 Function Documentation	23
6.1.1.1 cinEx()	23

6.1.1.2 generuotiFaila()	23
6.1.1.3 generuotiPazymius()	24
6.1.1.4 generuotiVardus()	24
6.1.1.5 irasytiPazymius()	24
6.1.1.6 irasytiVarda()	24
6.1.1.7 rasytiFaila()	24
6.1.1.8 rusiuotiStudentus()	24
6.1.1.9 rusiuotiStudentus3()	25
6.1.1.10 testuotiKurima()	25
6.2 funkcijos.cpp	25
6.3 galutinis_balas.cpp File Reference	29
6.3.1 Function Documentation	29
6.3.1.1 main()	29
6.4 galutinis_balas.cpp	29
6.5 header.h File Reference	34
6.5.1 Detailed Description	35
6.5.2 Function Documentation	36
6.5.2.1 cinEx()	36
6.5.2.2 generuotiFaila()	36
6.5.2.3 generuotiPazymius()	36
6.5.2.4 generuotiVardus()	36
6.5.2.5 irasytiPazymius()	36
6.5.2.6 irasytiVarda()	36
6.5.2.7 rasytiFaila() [1/2]	37
6.5.2.8 rasytiFaila() [2/2]	37
6.5.2.9 rusiuotiStudentus() [1/3]	37
6.5.2.10 rusiuotiStudentus() [2/3]	37
6.5.2.11 rusiuotiStudentus() [3/3]	37
6.5.2.12 rusiuotiStudentus3() [1/2]	38
6.5.2.13 rusiuotiStudentus3() [2/2]	38
6.5.2.14 skaitytiFaila()	38
6.5.2.15 testuotiKurima()	38
6.6 header.h	38
6.7 README.md File Reference	42
6.8 studentas.h File Reference	42
6.8.1 Detailed Description	42
6.8.2 Variable Documentation	43
6.8.2.1 TEST_MODE	43
6.9 studentas.h	43
6.10 timeris.h File Reference	45
6.10.1 Detailed Description	45
6.11 timeris.h	46

6.12 zmogus.h File Reference	46
6.13 zmogus.h	47
Index	49

Chapter 1

2.0 versija

Projekto Apžvalga

Ši programa sugeneruoja studentų duomenų failus, suskaičiuoja studentų galutinį balą pagal pasirinktą kriterijų suskirsto studentus į dvi kategorijas pagal jų galutinį balą ir matuoja užtruktą laiką kiekviename etape.

Šioje versijoje taip pat pridėta galimybė pasirinkti, kokio tipo konteinerį (vector, deque ar list) ir kokią strategiją norite naudoti.

1.1 Praeitų versijų aprašymai

- v0.1: programa leidžia įrašyti arba sugeneruoti savus studentus, o rezultatus rodo ekrane,
- v0.2: pridėtas skaitymas iš failo/ surašymas į failą,
- v0.3: funkcijos ir struktūros perkeltos į atskirus failus, pridėtas išimčių valdymas, leidžiantis sumažinti programos užstigimų šansą,
- v0.4: programa pati generuoja reikiamus failus, kuriuos naudoja tolimesniems skaičiavimams, pridėti laiko matavimai,
- v1.0: programa papildyta dar dviem konteineriais ir 3-jomis strategijomis,
- v1.1: "Studentas" pakeistas iš struktūros į klasę,
- v1.2: Sukurti Studento klasės kopijavimo ir perkėlimo konstruktoriai ir metodai ir perdengtos įvesties ir išvesties operacijos.
- v1.5: Sukurta bazinė (abstrakti) klasė "Žmogus", o "Studentas" padaryta "Žmogaus" išvestine klase

1.2 Platesnis programos aprašymas

1.2.1 Failų generavimas:

- Sugeneruojami penki skirtingo dydžio studentų sąrašų failai (1,000, 10,000, 100,000, 1,000,000 ir 10,000,000 įrašų).

1.2.2 Studentų rūšiavimas:

- Studentai skirstomi į dvi grupes:
 - „Vargšiukai“ – Galutinis balas < 5.0
 - „Galvočiai“ – Galutinis balas ≥ 5.0

1.2.3 Matuojamas laikas:

- Failo sukūrimui
- Duomenų nuskaitymui
- Duomenų failo rūšiavimui
- Studentų rūšiavimui
- Surūšiuotų duomenų išvedimui į naujus failus
- Bendram programos veikimo laikui

1.3 Kaip Paleisti Programą

Klonuokite repozitoriją:

```
git clone https://github.com/Eagle780/Objektninis.git
```

Pereikite į v1.0 šaką:

```
git checkout v1.0
```

Aplanke atsidarykite terminalą ir įrašykite:

```
mkdir build
cd build
cmake ..
cmake --build . --config Release
```

Paleidžiamasis failas (GalutinisBalas.exe) atsiras build/Release aplanke. Pirmą kartą paleidus programą bus sugeneruojami visi 5 testavimo failai. Visi sekantys paleidimai naudos tuos pačius 5 failus. Naujas failas bus sukurtas tik tada, jei senasis bus ištrintas.

1.4 Naudojimosi instrukcija

Pradžioje Jūsų bus klausiama, ar norite atlikti metodų testą, po to bus prašoma pasirinkti norimą konteinerį, po to pateiktas meniu leis išsirinkti naujų studentų įrašymų būdą. Baigus įrašymą (arba norint iškart skaityti studentus iš failo) pasirenkamas 4-tas meniu variantas. Tada galėsite pasirinkti, pagal ką norite rūšiuoti pagrindinį konteinerį. Po rūšiavimo liks pasirinkti būdą, kuriuo norite suskirstyti "vargšiukus" ir "galvočius" į skirtingus konteinerius. Baigus darbą spaudžiama "Enter".

1.4.1 Duomenų išvedimas/išvedimas

Naudojant ne vien .exe programą, o ir patį kodą, "studentas.h" galima rasti perdengtus >> (įvedimo) ir << (išvedimo) metodus. Jie leidžia patogiau ir greičiau naudotis klase. Jei norite pamatyti paprastą pavyzdį, kaip jie veikia, paleidus .exe failą paleiskite testą (įrašykite t raidę).

Iškart po raidės t įvedimo jums reikės suvesti studento duomenis, pvz:

```
vardas pavarde 10 9 10 10
```

Paskutinis pažimys bus nuskaitytas kaip egzamino rezultatas.

Įvedimo operatorių galima naudoti ne vien rankiniam įvedimui, bet ir nukaitymui iš failo, kadangi perdengtas operatorius kaip kintamąjį taip pat turi ir įvedimo būdą.

Studentą taip pat galima ir išvesti, panaudojus praeito pavyzdžio duomenis, bus išvedama

```
vardas pavarde 9.86667 10
```

Išvestis taip pat, kaip ir įvestis, gali būti naudojama ir darbui su failais, kadangi operacija kaip kintamąjį paima išvesties metodą.

1.5 Testavimo sistemos parametrai

- CPU - AMD Ryzen 5 5600H, 3.30GHz
- RAM - 2x8GB DDR4 3200MHz
- SSD - PCIe gen 3 NVMe M.2 512GB

1.6 Spartos Testavimo Rezultatai

Programa buvo testuojama Release režimu, siekiant gauti tikslius laiko matavimus. Žemiau pateikiamos rezultatų lentelės:

1.6.1 vector

Failo dydis	Duomenų skaitymas	Duomenų rūšiavimas	Studentų rūšiavimas
1,000 įrašų	0.00169	0.00203	0.00154
10,000 įrašų	0.01694	0.0303	0.0268
100,000 įrašų	0.354	0.376	0.354
1,000,000 įrašų	1.937	4.932	4.205
10,000,000 įrašų	18.483	76.858	4.717

1.6.2 deque

Failo dydis	Duomenų skaitymas	Duomenų rūšiavimas	Studentų rūšiavimas
1,000 įrašų	0.00178	0.00215	0.00164
10,000 įrašų	0.0148	0.0252	0.0276
100,000 įrašų	0.179	0.387	0.374
1,000,000 įrašų	1.952	5.101	4.349
10,000,000 įrašų	18.483	77.137	4.821

1.6.3 list

Failo dydis	Duomenų skaitymas	Duomenų rūšiavimas	Studentų rūšiavimas
1,000 įrašų	0.00196	0.00169	0.00111
10,000 įrašų	0.0178	0.0382	0.0231
100,000 įrašų	0.185	0.342	0.275
1,000,000 įrašų	2.093	5.427	3.029
10,000,000 įrašų	18.732	77.154	6.214

Kaip matome, laikai, naudojant vien tik skirtingus konteinerius, skiriasi, nors ir nežymiai. Vektorius vidutiniškai veikia greičiausiai, o list'as - lėčiausiai.

1.7 Strategijos testavimo rezultatai

Testuojama buvo ne vien lyginant skirtingų konteinerių naudojimą, bet ir pritaikant skirtingas studentų rūšiavimo strategijas

1 strategija: Studentai rūšiuojami į du skirtingus to pačio tipo konteinerius, 2 strategija: Naudojamas tik vienas papildomas konteineris, o perkelti įrašai ištrinami iš pradinio konteinerio 3 strategija: Greitesnis iš 2-ų buvusių taip pat taikant ir bibliotekas

1.7.1 vector

Failo dydis	1 strategija	2 strategija	3 strategija
1,000 įrašų	0.00030	0.0055	0.00024
10,000 įrašų	0.0027	0.511	0.0018
100,000 įrašų	0.030	40.92	0.013
1,000,000 įrašų	0.457	>1000	0.235
10,000,000 įrašų	4.868	-	5.268

1.7.2 deque

Failo dydis	1 strategija	2 strategija	3 strategija
1,000 įrašų	0.00039	0.00109	0.00023
10,000 įrašų	0.0027	0.1232	0.0019
100,000 įrašų	0.031	12.62	0.016
1,000,000 įrašų	0.427	>1000	0.253
10,000,000 įrašų	4.866	-	5.411

1.7.3 list

Failo dydis	1 strategija	2 strategija	3 strategija
1,000 įrašų	0.00052	0.00023	0.00032
10,000 įrašų	0.0041	0.0020	0.0024
100,000 įrašų	0.043	0.025	0.029
1,000,000 įrašų	0.430	0.302	0.392
10,000,000 įrašų	6.158	4.731	6.329

Išbandžius visas strategijas su visais konteineriais, galime pastebėti įdomų dalyką, kad 3 strategija yra greičiausia su visais konteineriais išskyrus list'ą. Norint naudoti list'ą būtų geriausia taikyti 2 strategiją. 2 strategijos neefektivumą, atsižvelgiant į vektorių ir deque laikus, galima paaiškinti tuo, kad atliekant kiekvieną perkėlimo operaciją, visi elementai turi būti perkopijuoti į naują vietą, neskaitant išmesto elemento. List'as nepatiria tokių minusų, nes jam reikia tik pakeisti rodykles į kitus narius. Tai taip pat iš dalies paaiškina, kodėl 2 strategija yra greitesnė nei 3, kai kalba eina apie list'ą.

1.8 Struktūros ir klasės testavimo rezultatai

Šioje versijoje taip pat reikėjo palyginti realizacijos spartą naudojant struktūrą (v1.0) ir klasę (dabartinė, v1.1). Abi programos naudojo vektorių ir 3 strategiją:

1.8.1 klasė

Failo dydis	Duomenų skaitymas	Rūšiavimas	Studentų rūšiavimas	Išvedimas į failus	Bendras laikas
100,000 įrašų	0.203	0.103	0.020	0.266	0.689
1,000,000 įrašų	2.142	1.196	0.251	2.165	5.901

1.8.2 struktūra

Failo dydis	Duomenų skaitymas	Rūšiavimas	Studentų rūšiavimas	Išvedimas į failus	Bendras laikas
100,000 įrašų	0.260	0.128	0.021	0.257	0.733
1,000,000 įrašų	2.626	1.609	0.277	3.286	7.835

Iš rezultatų matome, kad, nors ir nežymiai, klasė veikia greičiau nei struktūra.

1.9 Kompiliatoriaus flag'ų testavimo rezultatai

Programos greitis gali priklausyti ne vien nuo pačio kodo pateikimų, bet ir nuo paprasto flag'o (kompiliatoriaus optimizavimo lygio) pakeitimo kompiliuojant kodą. Testavimui naudotas 1,000,000 studentų failas bei vektorius ir 3 strategija:

1.9.1 Klasė

Flag	Duomenų skaitymas	Rūšiavimas	Studentų rūšiavimas	Išvedimas į failus	Bendras laikas	dydis (KB)
Be flag	1.953	15.422	0.606	15.954	33.939	722
O1	1.124	2.935	0.237	4.270	8.570	379
O2	1.086	2.796	0.229	4.079	8.195	349
O3	1.137	3.108	0.239	4.194	8.652	358

1.9.2 Struktūra

Flag	Duomenų skaitymas	Rūšiavimas	Studentų rūšiavimas	Išvedimas į failus	Bendras laikas	dydis (KB)
Be flag	1.869	16.665	0.623	16.668	34.533	721
O1	1.093	3.549	0.269	5.143	10.349	383
O2	1.146	3.458	0.275	4.878	9.891	351
O3	1.112	3.475	0.276	4.946	10.043	354

Kaip galima pastebėti iš rezultatų, greičiausias (ir mažiausiai vietos užimantis) flag'as yra ne O3, bet O2. Taip gali atsitikti todėl, nes O3 taiko sudėtingesnius kompiliavimo metodus, kurie parodo savo naudą naudojant dar daugiau duomenų, nei buvo naudojama dabar.

1.10 Nuotraukos

Programos veikimo pavyzdys:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Timer	19
Zmogus	20
Studentas	13

Chapter 3

Class Index

3.1 Class List

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

Studentas	13
Timer	19
Zmogus	20

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

funkcijos.cpp	23
galutinis_balas.cpp	29
header.h	
Header failas, kuriame aprašytos visos naudojamos funkcijos	34
studentas.h	
Studento klasė	42
timeris.h	
Klasė naudojama atliekamų funkcijų matavimui	45
zmogus.h	46

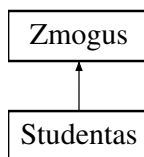
Chapter 5

Class Documentation

5.1 Studentas Class Reference

```
#include <studentas.h>
```

Inheritance diagram for Studentas:



Public Member Functions

- `Studentas ()`
Tuščias studento konstruktorius.
- `Studentas (string v, string p, vector< int > n, int e)`
Pilnas studento konstruktorius.
- `Studentas (const Studentas &st)`
Kopijavimo konstruktorius.
- `Studentas & operator= (const Studentas &st)`
Kopijavimo operacija.
- `Studentas (Studentas &&st) noexcept`
Studento perkavimo konstruktorius.
- `Studentas & operator= (Studentas &&st) noexcept`
Studento perkavimo operacija.
- `void setVarPav (string v, string p)`
Studento vardo ir pavardės set'eris.
- `void setPaz (vector< int > n, int e)`
Studento namų darbų ir egzamino pažymių set'eris.
- `string getVardas () const` override
- `string getPavarde () const` override
- `int getEgz () const`
- `vector< int > getNd () const`
- `int getGalV () const`

- int `getGalM ()` const
- float `SkaiciuotiV ()`
Skačiuojamas studento galutinis balas pagal vidurkj
- float `SkaiciuotiM ()`
Skačiuojamas studento galutinis balas pagal medianą
- void `clear ()`
funkcija, kuri iškviečiama iškvietus destruktorių
- `~Studentas ()`

Public Member Functions inherited from `Zmogus`

- `Zmogus ()`
- `Zmogus (string v, string p)`
- void `setVarPav (string v, string p)`
- virtual `~Zmogus ()`

Friends

- ostream & `operator<<` (ostream &os, const `Studentas` &st)
Studento išvedimo operacija.
- istream & `operator>>` (istream &is, `Studentas` &st)
Studento įvedimo operacija.

Additional Inherited Members

Protected Attributes inherited from `Zmogus`

- string `vardas`
- string `pavarde`

5.1.1 Detailed Description

Definition at line 32 of file `studentas.h`.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `Studentas()` [1/4]

```
Studentas::Studentas () [inline]
```

Tuščias studento konstruktorius.

Definition at line 47 of file `studentas.h`.

5.1.2.2 `Studentas()` [2/4]

```
Studentas::Studentas (
    string v,
    string p,
    vector< int > n,
    int e) [inline]
```

Pilnas studento konstruktorius.

Parameters

<i>v</i>	Vardas
<i>p</i>	Pavardė
<i>n</i>	Namų darbų pažymiai (vektorius)
<i>e</i>	Egzamino pažymys

Definition at line 56 of file [studentas.h](#).

5.1.2.3 Studentas() [3/4]

```
Studentas::Studentas (  
    const Studentas & st) [inline]
```

Kopijavimo konstruktorius.

Parameters

<i>st</i>	Studentas , kurio duomenis norima kopijuoti
-----------	---

Definition at line 70 of file [studentas.h](#).

5.1.2.4 Studentas() [4/4]

```
Studentas::Studentas (  
    Studentas && st) [inline], [noexcept]
```

Studento perkeltimo konstruktorius.

Parameters

<i>st</i>	Studentas , kurio duomenis norima perkelti
-----------	--

Definition at line 113 of file [studentas.h](#).

5.1.2.5 ~Studentas()

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

Definition at line 275 of file [studentas.h](#).

5.1.3 Member Function Documentation

5.1.3.1 clear()

```
void Studentas::clear () [inline]
```

funkcija, kuri iškviečiama iškvietus destruktorių

Definition at line 264 of file [studentas.h](#).

5.1.3.2 getEgz()

```
int Studentas::getEgz () const [inline]
```

Definition at line 218 of file [studentas.h](#).

5.1.3.3 getGalM()

```
int Studentas::getGalM () const [inline]
```

Definition at line 221 of file [studentas.h](#).

5.1.3.4 getGalV()

```
int Studentas::getGalV () const [inline]
```

Definition at line 220 of file [studentas.h](#).

5.1.3.5 getNd()

```
vector< int > Studentas::getNd () const [inline]
```

Definition at line 219 of file [studentas.h](#).

5.1.3.6 getPavarde()

```
string Studentas::getPavarde () const [inline], [override], [virtual]
```

Implements [Zmogus](#).

Definition at line 217 of file [studentas.h](#).

5.1.3.7 getVardas()

```
string Studentas::getVardas () const [inline], [override], [virtual]
```

Implements [Zmogus](#).

Definition at line 216 of file [studentas.h](#).

5.1.3.8 operator=() [1/2]

```
Studentas & Studentas::operator= (  
    const Studentas & st) [inline]
```

Kopijavimo operacija.

Parameters

<i>st</i>	Studentas , kurio duomenis norima kopijuoti
-----------	---

Returns

[Studentas](#), į kurį buvo nukopijuoti duomenys

Definition at line 88 of file [studentas.h](#).

5.1.3.9 operator=() [2/2]

```
Studentas & Studentas::operator= (  
    Studentas && st) [inline], [noexcept]
```

Studento perkeltimo operacija.

Parameters

<i>st</i>	Studentas , kurio duomenis norima perkelti
-----------	--

Returns

[Studentas](#), į kurį buvo perkelti duomenys

Definition at line 131 of file [studentas.h](#).

5.1.3.10 setPaz()

```
void Studentas::setPaz (  
    vector< int > n,  
    int e) [inline]
```

Studento namų darbų ir egzamino pažymių set'eris.

Parameters

<i>n</i>	namų darbų pažymių vektorius
<i>e</i>	egzamino pažymys

Definition at line 209 of file [studentas.h](#).

5.1.3.11 setVarPav()

```
void Studentas::setVarPav (  
    string v,  
    string p) [inline]
```

Studento vardo ir pavardės set'eris.

Parameters

<i>v</i>	Vardas
<i>p</i>	Pavardė

Definition at line 198 of file [studentas.h](#).

5.1.3.12 SkaiciuotiM()

```
float Studentas::SkaiciuotiM () [inline]
```

Skaičiuojamas studento galutinis balas pagal medianą

Returns

Grąžinama suskaiciuota galutinio balo reikšmė, kuri išsaugoma "galutinisM"

Definition at line 242 of file [studentas.h](#).

5.1.3.13 SkaiciuotiV()

```
float Studentas::SkaiciuotiV () [inline]
```

Skaičiuojamas studento galutinis balas pagal vidurkį

Returns

Grąžinama suskaiciuota galutinio balo reikšmė, kuri išsaugoma "galutinisV"

Definition at line 227 of file [studentas.h](#).

5.1.4 Friends And Related Symbol Documentation**5.1.4.1 operator<<**

```
ostream & operator<< (
    ostream & os,
    const Studentas & st) [friend]
```

Studento išvedimo operacija.

Parameters

<i>os</i>	Būdas, kuriuo išvedama
<i>st</i>	Išvedamas studentas

Returns

ostream&

Definition at line 155 of file [studentas.h](#).

5.1.4.2 operator>>

```
istream & operator>> (
    istream & is,
    Studentas & st) [friend]
```

Studento įvedimo operacija.

Parameters

<i>is</i>	Būdas, kuriuo įvedama
<i>st</i>	Įvedamas studentas

Returns

istream&

Definition at line 167 of file [studentas.h](#).

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

- [studentas.h](#)

5.2 Timer Class Reference

```
#include <timeris.h>
```

Public Member Functions

- [Timer](#) ()
Konstruktorius, kuris išsaugo laiką, kada buvo iškviestas.
- void [reset](#) ()
restartuoja laiko kintamąjį
- double [elapsed](#) () const
iš dabartinio laiko atima išsaugotą

5.2.1 Detailed Description

Definition at line 16 of file [timeris.h](#).

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Timer()

```
Timer::Timer () [inline]
```

Konstruktorius, kuris išsaugo laiką, kada buvo iškviestas.

Definition at line 28 of file [timeris.h](#).

5.2.3 Member Function Documentation

5.2.3.1 elapsed()

```
double Timer::elapsed () const [inline]
```

iš dabartinio laiko atima išsaugotą

Returns

Laikas nuo konstruktoriaus iškvietimo iki dabar

Definition at line [42](#) of file [timeris.h](#).

5.2.3.2 reset()

```
void Timer::reset () [inline]
```

restartuoja laiko kintamąjį

Definition at line [33](#) of file [timeris.h](#).

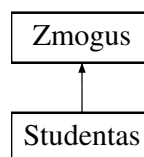
The documentation for this class was generated from the following file:

- [timeris.h](#)

5.3 Zmogus Class Reference

```
#include <zmogus.h>
```

Inheritance diagram for Zmogus:



Public Member Functions

- [Zmogus](#) ()
- [Zmogus](#) (string v, string p)
- void [setVarPav](#) (string v, string p)
- virtual string [getVardas](#) () const =0
- virtual string [getPavarde](#) () const =0
- virtual [~Zmogus](#) ()

Protected Attributes

- string [vardas](#)
- string [pavarde](#)

5.3.1 Detailed Description

Definition at line 8 of file [zmogus.h](#).

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Zmogus() [1/2]

```
Zmogus::Zmogus () [inline]
```

Definition at line 15 of file [zmogus.h](#).

5.3.2.2 Zmogus() [2/2]

```
Zmogus::Zmogus (  
    string v,  
    string p) [inline]
```

Definition at line 16 of file [zmogus.h](#).

5.3.2.3 ~Zmogus()

```
virtual Zmogus::~Zmogus () [inline], [virtual]
```

Definition at line 28 of file [zmogus.h](#).

5.3.3 Member Function Documentation

5.3.3.1 getPavarde()

```
virtual string Zmogus::getPavarde () const [pure virtual]
```

Implemented in [Studentas](#).

5.3.3.2 getVardas()

```
virtual string Zmogus::getVardas () const [pure virtual]
```

Implemented in [Studentas](#).

5.3.3.3 setVarPav()

```
void Zmogus::setVarPav (
    string v,
    string p) [inline]
```

Definition at line 21 of file [zmogus.h](#).

5.3.4 Member Data Documentation

5.3.4.1 pavarde

```
string Zmogus::pavarde [protected]
```

Definition at line 12 of file [zmogus.h](#).

5.3.4.2 vardas

```
string Zmogus::vardas [protected]
```

Definition at line 11 of file [zmogus.h](#).

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

- [zmogus.h](#)

Chapter 6

File Documentation

6.1 funkcijos.cpp File Reference

```
#include "header.h"
#include "timeris.h"
```

Functions

- void [cinEx](#) ()
- [Studentas generuotiPazymius](#) ([Studentas](#) temp)
- [Studentas generuotiVardus](#) ([Studentas](#) temp)
- [Studentas irasytiPazymius](#) ([Studentas](#) temp)
- [Studentas irasytiVarda](#) ([Studentas](#) temp)
- void [rasytiFaila](#) (string pav, list< [Studentas](#) > &v, string pas)
- bool [generuotiFaila](#) (string &failas, int ndDydis, int &dydis)
- string [rusiuotiStudentus](#) (list< [Studentas](#) > &A, list< [Studentas](#) > &v, int var)
- string [rusiuotiStudentus3](#) (list< [Studentas](#) > &A, list< [Studentas](#) > &v, int var)
- void [testuotiKurima](#) (string &failas, int ndDydis, int &dydis)

6.1.1 Function Documentation

6.1.1.1 cinEx()

```
void cinEx ()
```

Definition at line [4](#) of file [funkcijos.cpp](#).

6.1.1.2 generuotiFaila()

```
bool generuotiFaila (
    string & failas,
    int ndDydis,
    int & dydis)
```

Definition at line [169](#) of file [funkcijos.cpp](#).

6.1.1.3 generuotiPazymius()

```
Studentas generuotiPazymius (  
    Studentas temp)
```

Definition at line 9 of file [funkcijos.cpp](#).

6.1.1.4 generuotiVardus()

```
Studentas generuotiVardus (  
    Studentas temp)
```

Definition at line 51 of file [funkcijos.cpp](#).

6.1.1.5 irasytiPazymius()

```
Studentas irasytiPazymius (  
    Studentas temp)
```

Definition at line 65 of file [funkcijos.cpp](#).

6.1.1.6 irasytiVarda()

```
Studentas irasytiVarda (  
    Studentas temp)
```

Definition at line 130 of file [funkcijos.cpp](#).

6.1.1.7 rasytiIFaila()

```
void rasytiIFaila (  
    string pav,  
    list< Studentas > & v,  
    string pas)
```

Definition at line 142 of file [funkcijos.cpp](#).

6.1.1.8 rusiuotiStudentus()

```
string rusiuotiStudentus (  
    list< Studentas > & A,  
    list< Studentas > & v,  
    int var)
```

Definition at line 198 of file [funkcijos.cpp](#).

6.1.1.9 rusiuotiStudentus3()

```
string rusiuotiStudentus3 (
    list< Studentas > & A,
    list< Studentas > & v,
    int var)
```

Definition at line 250 of file funkcijos.cpp.

6.1.1.10 testuotiKurima()

```
void testuotiKurima (
    string & failas,
    int ndDydis,
    int & dydis)
```

Definition at line 292 of file funkcijos.cpp.

6.2 funkcijos.cpp

[Go to the documentation of this file.](#)

```
00001 #include "header.h"
00002 #include "timeris.h"
00003
00004 void cinEx()
00005 {
00006     cin.exceptions(ios::failbit | ios::badbit);
00007 }
00008
00009 Studentas generuotiPazymius(Studentas temp)
00010 {
00011     vector<int> nd;
00012     int n = 0;
00013     int a = 0;
00014     while (true)
00015     {
00016         cout << "Iveskite pazymiu skaiciu: ";
00017
00018         try
00019         {
00020             cin >> n;
00021         }
00022         catch (ios_base::failure &e)
00023         {
00024             cout << "Neteisinga ivestis\n";
00025             cin.clear();
00026             cin.ignore(1000, '\n');
00027             continue;
00028         }
00029         if (n == 0)
00030         {
00031             cout << "Netinkamas skaicius\n";
00032             continue;
00033         }
00034         break;
00035     }
00036
00037     srand(time(0));
00038
00039     for (int i = 0; i < n; i++)
00040     {
00041         a = 1 + rand() % 10;
00042         nd.push_back(a);
00043     }
00044
00045     int egz = 1 + rand() % 10;
00046     temp.setPaz(nd, egz);
00047
00048     return temp;
```

```

00049 }
00050
00051 Studentas generuotiVardus(Studentas temp)
00052 {
00053     vector<string> Vardai = {"Jonas", "Antanas", "Petras", "Dovydas", "Tomas"};
00054     vector<string> Pavardes = {"Jonaitis", "Petrauskas", "Kazlauskas", "Antanaitis", "Ivanauskas"};
00055
00056     srand(time(0));
00057
00058     string vardas = Vardai[rand() % Vardai.size()];
00059     string pavarde = Pavardes[rand() % Pavardes.size()];
00060     temp.setVarPav(vardas, pavarde);
00061
00062     return temp;
00063 }
00064
00065 Studentas irasytiPazymius(Studentas temp)
00066 {
00067     vector<int> nd;
00068     int a;
00069     cout << "Iveskite pazymius (norint baigti pazymiu rasyma, irasykite 0):\n";
00070     while (true)
00071     {
00072         try
00073         {
00074             cin >> a;
00075         }
00076         catch (ios_base::failure &e)
00077         {
00078             cout << "Neteisinga ivestis\n";
00079             cin.clear();
00080             cin.ignore(1000, '\n');
00081             continue;
00082         }
00083
00084         if (a > 0 && a <= 10)
00085         {
00086             nd.push_back(a);
00087         }
00088         else if (a == 0)
00089         {
00090             if (nd.size() == 0)
00091                 cout << "Iveskite bent viena pazymi\n";
00092             else
00093                 break;
00094         }
00095         else
00096             cout << "Neteisingas pazymys\n";
00097     }
00098
00099     a = 0;
00100     while (true)
00101     {
00102         cout << "Iveskite egzamino pazymi: ";
00103         try
00104         {
00105             cin >> a;
00106         }
00107         catch (ios_base::failure &e)
00108         {
00109             cout << "Neteisinga ivestis\n";
00110             cin.clear();
00111             cin.ignore(1000, '\n');
00112             continue;
00113         }
00114
00115         if (a <= 10 && a > 0)
00116         {
00117             break;
00118         }
00119         else
00120         {
00121             cout << "Netinkamas skaicius\n";
00122         }
00123     }
00124
00125     temp.setPaz(nd, a);
00126
00127     return temp;
00128 }
00129
00130 Studentas irasytiVarda(Studentas temp)
00131 {
00132     string vardas, pavarde;
00133     cout << "Iveskite varda: ";
00134     cin >> vardas;
00135     cout << "Iveskite pavarde: ";

```



```

00136     cin » pavarde;
00137     temp.setVarPav(vardas, pavarde);
00138
00139     return temp;
00140 }
00141
00142 void rasytiIFaila(string pav, list<Studentas> &v, string pas)
00143 {
00144     if (pas == "v")
00145     {
00146         v.sort([](const Studentas &a, const Studentas &b)
00147             { return a.getGalV() > b.getGalV(); });
00148     }
00149     else if (pas == "m")
00150     {
00151         v.sort([](const Studentas &a, const Studentas &b)
00152             { return a.getGalM() > b.getGalM(); });
00153     }
00154     ofstream fr(pav);
00155     if (!fr.is_open())
00156     {
00157         std::cerr << "Nepavyko atidaryti failo: " << pav << "\n";
00158         return;
00159     }
00160     fr << "Vardas      Pavardė      Galutinis (Vid.)  Galutinis (Med.)\n";
00161     fr << "-----\n";
00162     for (const Studentas &i : v)
00163     {
00164         fr << left << setw(12) << i.getVardas() << setw(16) << i.getPavarde();
00165         fr << fixed << setw(17) << setprecision(2) << i.getGalV() << i.getGalM() << "\n";
00166     }
00167 }
00168
00169 bool generuotiFaila(string &failas, int ndDydis, int &dydis)
00170 {
00171     if (std::filesystem::exists(failas))
00172     {
00173         return false;
00174     }
00175
00176     ofstream fr(failas);
00177     fr << left << setw(17) << "Vardas" << setw(18) << "Pavarde";
00178     for (int i = 0; i < ndDydis; i++)
00179     {
00180         fr << "ND" << setw(3) << i + 1;
00181     }
00182     fr << setw(5) << "Egz.\n";
00183
00184     srand(time(0));
00185     for (int i = 0; i < dydis; i++)
00186     {
00187         fr << "VardasNR" << setw(9) << i + 1 << "PavardeNR" << setw(10) << i + 1;
00188         for (int j = 0; j < ndDydis + 1; j++)
00189         {
00190             fr << setw(5) << 1 + rand() % 10;
00191         }
00192         fr << "\n";
00193     }
00194     fr.close();
00195     return true;
00196 }
00197
00198 string rusiuotiStudentus(list<Studentas> &A, list<Studentas> &v, int var)
00199 {
00200     string pas = "";
00201
00202     if (var == 3)
00203     {
00204         pas = "v";
00205     }
00206     else if (var == 4)
00207     {
00208         pas = "m";
00209     }
00210
00211     while (pas != "v" && pas != "m")
00212     {
00213         cout << "Studentus rusiuoti pagal vidurki ar mediana? (v/m)\n";
00214         cin » pas;
00215     }
00216
00217     if (pas == "v")
00218     {
00219         for (auto t = A.end(); t != A.begin(); t--)
00220         {
00221             if (t->getGalV() < 5.0)
00222             {

```

```

00223         v.push_back(*t);
00224         t = A.erase(t);
00225     }
00226     else
00227     {
00228         --t;
00229     }
00230 }
00231 }
00232 else
00233 {
00234     for (auto t = A.end(); t != A.begin(); )
00235     {
00236         if (t->getGalM() < 5.0)
00237         {
00238             v.push_back(*t);
00239             t = A.erase(t);
00240         }
00241         else
00242         {
00243             --t;
00244         }
00245     }
00246 }
00247 return pas;
00248 }
00249
00250 string rusiuotiStudentus3(list<Studentas> &A, list<Studentas> &v, int var)
00251 {
00252     string pas = "";
00253
00254     if (var == 3)
00255     {
00256         pas = "v";
00257     }
00258     else if (var == 4)
00259     {
00260         pas = "m";
00261     }
00262
00263     while (pas != "v" && pas != "m")
00264     {
00265         cout << "Studentus rusiuoti pagal vidurki ar mediana? (v/m)\n";
00266         cin >> pas;
00267     }
00268
00269     if (pas == "v")
00270     {
00271         A.remove_if([&](const Studentas &st)
00272         {
00273             if (st.getGalV() < 5.0) {
00274                 v.push_back(st);
00275                 return true;
00276             }
00277             return false; });
00278     }
00279     else
00280     {
00281         A.remove_if([&](const Studentas &st)
00282         {
00283             if (st.getGalM() < 5.0) {
00284                 v.push_back(st);
00285                 return true;
00286             }
00287             return false; });
00288     }
00289     return pas;
00290 }
00291
00292 void testuotiKurima(string &failas, int ndDydis, int &dydis)
00293 {
00294     Timer t;
00295     bool arEgzistuoja = generuotiFaila(failas, ndDydis, dydis);
00296     double laikas = t.elapsed();
00297     if (!arEgzistuoja)
00298     {
00299         return;
00300     }
00301     cout << dydis << " studentu failo generavimo laikas: " << laikas << "\n";
00302 }

```

6.3 galutinis_balas.cpp File Reference

```
#include "header.h"
#include "timeris.h"
```

Functions

- int [main](#) ()

6.3.1 Function Documentation

6.3.1.1 main()

```
int main ()
```

Definition at line 4 of file [galutinis_balas.cpp](#).

6.4 galutinis_balas.cpp

[Go to the documentation of this file.](#)

```
00001 #include "header.h"
00002 #include "timeris.h"
00003
00004 int main()
00005 {
00006     int dydis = 100;
00007     string failas = "studentai" + to_string(dydis) + ".txt";
00008     int ndDydis = 5;
00009     double laikas = 0;
00010     int k = 0;
00011     string pas;
00012
00013     cinEx();
00014
00015     vector<Studentas> A;
00016     deque<Studentas> B;
00017     list<Studentas> C;
00018
00019     vector<Studentas> vargsiukai, galvociai;
00020     deque<Studentas> vargsiukaiB, galvociaiB;
00021     list<Studentas> vargsiukaiC, galvociaiC;
00022
00023     while (dydis < 10000000)
00024     {
00025         dydis *= 10;
00026         failas = "studentai" + to_string(dydis) + ".txt";
00027         testuotiKurima(failas, ndDydis, dydis);
00028     }
00029
00030     char test;
00031     while (true)
00032     {
00033         cout << "ar atlikti klases testa? (t/n)\n";
00034         try
00035         {
00036             cin >> test;
00037         }
00038         catch (ios_base::failure &e)
00039         {
00040             cout << "Neteisinga ivestis\n";
00041             cin.clear();
00042             cin.ignore(1000, '\n');
00043             continue;
00044         }
00045         if (test != 't' && test != 'n')
```

```

00046     {
00047         cout << "Neteisinga ivestis\n";
00048         continue;
00049     }
00050     break;
00051 }
00052
00053 if (test == 't')
00054 {
00055     TEST_MODE = true;
00056     Zmogus *temp = new Studentas();
00057     cout << temp << "\n";
00058     delete temp;
00059
00060     Studentas temp1;
00061     cin >> temp1;
00062     cout << temp1;
00063     cout << "---\n";
00064
00065     Studentas temp2 = temp1;
00066     cout << "pradinis temp1: " << temp1;
00067     cout << "nukopijuotas temp2: " << temp2;
00068     cout << "---\n";
00069
00070     Studentas temp3 = move(temp2);
00071     cout << "perkeltas temp3: " << temp3;
00072     cout << "perkeltas temp2: " << temp2;
00073     cout << "---\n";
00074
00075     Studentas temp4("vardas", "pavarde", {10, 10, 10}, 8);
00076     cout << "sukurtas temp4: " << temp4;
00077     cout << "---\n";
00078
00079     temp2 = temp4;
00080     temp3 = move(temp4);
00081     cout << "nukopijuotas temp2: " << temp2;
00082     cout << "perkeltas temp3: " << temp3;
00083     cout << "perkeltas temp4: " << temp4;
00084 }
00085 TEST_MODE = false;
00086 cout << "\n";
00087
00088 while (true)
00089 {
00090     cout << "Koki konteineri naudoti duomeniu saugojimui?\n";
00091     cout << "1 - vector, 2 - deque, 3 - list\n";
00092     try
00093     {
00094         cin >> k;
00095     }
00096     catch (ios_base::failure &e)
00097     {
00098         cout << "Neteisinga ivestis\n";
00099         cin.clear();
00100         cin.ignore(1000, '\n');
00101         continue;
00102     }
00103     if (k <= 0 || k >= 4)
00104     {
00105         cout << "Neteisingas skaicius\n";
00106         continue;
00107     }
00108     break;
00109 }
00110
00111 while (true)
00112 {
00113     Studentas temp;
00114     int a = 0;
00115
00116     cout << "Studento duomeniu ivedimo pasirinkimai:\n";
00117     cout << "1 - ranka, 2 - generuoti pazymius, 3 - generuoti ir pazymius ir studentu vardus,
pavardes, 4 - baigti ivedima/skaityti duomenis is failo\n";
00118
00119     try
00120     {
00121         cin >> a;
00122     }
00123     catch (ios_base::failure &e)
00124     {
00125         cout << "Neteisinga ivestis\n";
00126         cin.clear();
00127         cin.ignore(1000, '\n');
00128         continue;
00129     }
00130
00131     if (a <= 0 || a >= 5)

```

```

00132     {
00133         cout << "Neteisingas skaicius\n";
00134         continue;
00135     }
00136
00137     if (a == 1)
00138     {
00139         temp = irasytiVarda(temp);
00140         temp = irasytiPazymius(temp);
00141
00142         A.push_back(temp);
00143     }
00144
00145     else if (a == 2)
00146     {
00147         temp = irasytiVarda(temp);
00148         temp = generuotiPazymius(temp);
00149
00150         A.push_back(temp);
00151     }
00152
00153     else if (a == 3)
00154     {
00155         temp = generuotiVardus(temp);
00156         temp = generuotiPazymius(temp);
00157
00158         A.push_back(temp);
00159     }
00160
00161     else
00162     {
00163         break;
00164     }
00165 }
00166
00167 while (true)
00168 {
00169     cout << "Kokio dydzio faila naudoti? ";
00170     try
00171     {
00172         cin >> dydis;
00173     }
00174
00175     catch (ios_base::failure &e)
00176     {
00177         cout << "Neteisinga ivestis\n";
00178         cin.clear();
00179         cin.ignore(1000, '\n');
00180         continue;
00181     }
00182
00183     if (!std::filesystem::exists("studentai" + to_string(dydis) + ".txt"))
00184     {
00185         cout << "Tokio dydzio failo nera\n";
00186         continue;
00187     }
00188     failas = "studentai" + to_string(dydis) + ".txt";
00189     break;
00190 }
00191
00192 Timer tmain;
00193 Timer t;
00194 if (k == 1)
00195 {
00196     skaitytiFaila(failas, A);
00197 }
00198 else if (k == 2)
00199 {
00200     skaitytiFaila(failas, B);
00201 }
00202 else if (k == 3)
00203 {
00204     skaitytiFaila(failas, C);
00205 }
00206 laikas = t.elapsed();
00207 cout << dydis << " studentu failo nuskaitymo laikas: " << laikas << "\n";
00208
00209 int variantas = 0;
00210
00211 while (true)
00212 {
00213     cout << "Duomenis rusiuoti pagal:\n";
00214     cout << "1 - varda, 2 - pavarde, 3 - galutini (vid.), 4 - galutini (med.)\n";
00215
00216     try
00217     {
00218         cin >> variantas;

```

```

00219     }
00220     catch (ios_base::failure &e)
00221     {
00222         cout << "Neteisinga ivestis\n";
00223         cin.clear();
00224         cin.ignore(1000, '\n');
00225         continue;
00226     }
00227
00228     if (variantas < 1 || variantas > 4)
00229     {
00230         cout << "Neteisingas skaicius\n";
00231         continue;
00232     }
00233     break;
00234 }
00235
00236 Timer t3;
00237 if (k == 1)
00238 {
00239     if (variantas == 1)
00240     {
00241         sort(A.begin(), A.end(), [](const Studentas &a, const Studentas &b)
00242             { return a.getVardas() < b.getVardas(); });
00243     }
00244
00245     else if (variantas == 2)
00246     {
00247         sort(A.begin(), A.end(), [](const Studentas &a, const Studentas &b)
00248             { return a.getPavarde() < b.getPavarde(); });
00249     }
00250
00251     else if (variantas == 3)
00252     {
00253         sort(A.begin(), A.end(), [](const Studentas &a, const Studentas &b)
00254             { return a.getGalV() > b.getGalV(); });
00255     }
00256
00257     else if (variantas == 4)
00258     {
00259         sort(A.begin(), A.end(), [](const Studentas &a, const Studentas &b)
00260             { return a.getGalM() > b.getGalM(); });
00261     }
00262 }
00263 else if (k == 2)
00264 {
00265     if (variantas == 1)
00266     {
00267         sort(B.begin(), B.end(), [](const Studentas &a, const Studentas &b)
00268             { return a.getVardas() < b.getVardas(); });
00269     }
00270
00271     else if (variantas == 2)
00272     {
00273         sort(B.begin(), B.end(), [](const Studentas &a, const Studentas &b)
00274             { return a.getPavarde() < b.getPavarde(); });
00275     }
00276
00277     else if (variantas == 3)
00278     {
00279         sort(B.begin(), B.end(), [](const Studentas &a, const Studentas &b)
00280             { return a.getGalV() > b.getGalV(); });
00281     }
00282
00283     else if (variantas == 4)
00284     {
00285         sort(B.begin(), B.end(), [](const Studentas &a, const Studentas &b)
00286             { return a.getGalM() > b.getGalM(); });
00287     }
00288 }
00289 else if (k == 3)
00290 {
00291     if (variantas == 1)
00292     {
00293         C.sort([](const Studentas &a, const Studentas &b)
00294             { return a.getVardas() < b.getVardas(); });
00295     }
00296
00297     else if (variantas == 2)
00298     {
00299         C.sort([](const Studentas &a, const Studentas &b)
00300             { return a.getPavarde() < b.getPavarde(); });
00301     }
00302
00303     else if (variantas == 3)
00304     {
00305         C.sort([](const Studentas &a, const Studentas &b)

```

```

00306         { return a.getGalV() > b.getGalV(); });
00307     }
00308
00309     else if (variantas == 4)
00310     {
00311         C.sort([](const Studentas &a, const Studentas &b)
00312             { return a.getGalM() > b.getGalM(); });
00313     }
00314 }
00315 laikas = t3.elapsed();
00316 cout << dydis << " studentu konteinerio rusiavimo laikas: " << laikas << "\n";
00317
00318 int var2 = 0;
00319
00320 while (true)
00321 {
00322     cout << "Koki duomenu rusiavimo buda naudoti?\n";
00323     cout << "1 - vargsiuku ir galvociu, 2 - tik vargsiuku, 3 - greitesni su algoritmais\n";
00324
00325     try
00326     {
00327         cin >> var2;
00328     }
00329     catch (ios_base::failure &e)
00330     {
00331         cout << "Neteisinga ivestis\n";
00332         cin.clear();
00333         cin.ignore(1000, '\n');
00334         continue;
00335     }
00336
00337     if (var2 < 1 || var2 > 4)
00338     {
00339         cout << "Neteisingas skaicius\n";
00340         continue;
00341     }
00342     break;
00343 }
00344
00345 if (var2 == 1)
00346 {
00347     if (k == 1)
00348     {
00349         Timer t4;
00350         pas = rusiuotiStudentus(A, vargsiukai, galvociai, variantas);
00351         laikas = t4.elapsed();
00352     }
00353     else if (k == 2)
00354     {
00355         Timer t4;
00356         pas = rusiuotiStudentus(B, vargsiukaiB, galvociaiB, variantas);
00357         laikas = t4.elapsed();
00358     }
00359     else if (k == 3)
00360     {
00361         Timer t4;
00362         pas = rusiuotiStudentus(C, vargsiukaiC, galvociaiC, variantas);
00363         laikas = t4.elapsed();
00364     }
00365     cout << dydis << " studentu surusiavimo i 2 konteinerius laikas: " << laikas << "\n";
00366 }
00367 else if (var2 == 2)
00368 {
00369     if (k == 1)
00370     {
00371         Timer t4;
00372         pas = rusiuotiStudentus(A, vargsiukai, variantas);
00373         laikas = t4.elapsed();
00374     }
00375     else if (k == 2)
00376     {
00377         Timer t4;
00378         pas = rusiuotiStudentus(B, vargsiukaiB, variantas);
00379         laikas = t4.elapsed();
00380     }
00381     else if (k == 3)
00382     {
00383         Timer t4;
00384         pas = rusiuotiStudentus(C, vargsiukaiC, variantas);
00385         laikas = t4.elapsed();
00386     }
00387     cout << dydis << " studentu surusiavimo i 1 konteineri laikas: " << laikas << "\n";
00388 }
00389 else if (var2 == 3)
00390 {
00391     if (k == 1)
00392     {

```

```

00393         Timer t4;
00394         pas = rusiuotiStudentus3(A, vargsiukai, variantas);
00395         laikas = t4.elapsed();
00396     }
00397     else if (k == 2)
00398     {
00399         Timer t4;
00400         pas = rusiuotiStudentus3(B, vargsiukaiB, variantas);
00401         laikas = t4.elapsed();
00402     }
00403     else if (k == 3)
00404     {
00405         Timer t4;
00406         pas = rusiuotiStudentus3(C, vargsiukaiC, variantas);
00407         laikas = t4.elapsed();
00408     }
00409     cout << dydis << " studentu surusiavimo greitesniu budu su algoritmais laikas: " << laikas <<
"\n";
00410 }
00411
00412 Timer t2;
00413 if (var2 == 1)
00414 {
00415     if (k == 1)
00416     {
00417         rasytiIFaila("vargsiukai.txt", vargsiukai, pas);
00418         rasytiIFaila("galvociai.txt", galvociai, pas);
00419     }
00420     else if (k == 2)
00421     {
00422         rasytiIFaila("vargsiukai.txt", vargsiukaiB, pas);
00423         rasytiIFaila("galvociai.txt", galvociaiB, pas);
00424     }
00425     else if (k == 3)
00426     {
00427         rasytiIFaila("vargsiukai.txt", vargsiukaiC, pas);
00428         rasytiIFaila("galvociai.txt", galvociaiC, pas);
00429     }
00430 }
00431 else if (var2 == 2 || var2 == 3)
00432 {
00433     if (k == 1)
00434     {
00435         rasytiIFaila("vargsiukai.txt", vargsiukai, pas);
00436         rasytiIFaila("galvociai.txt", A, pas);
00437     }
00438     else if (k == 2)
00439     {
00440         rasytiIFaila("vargsiukai.txt", vargsiukaiB, pas);
00441         rasytiIFaila("galvociai.txt", B, pas);
00442     }
00443     else if (k == 3)
00444     {
00445         rasytiIFaila("vargsiukai.txt", vargsiukaiC, pas);
00446         rasytiIFaila("galvociai.txt", C, pas);
00447     }
00448 }
00449 laikas = t2.elapsed();
00450 cout << dydis << " surusiuotu studentu irasyimo laikas: " << laikas << "\n";
00451
00452 double visasLaikas = tmain.elapsed();
00453 cout << dydis << " studentu programos veikimo (nuo failo pasirinkimo) laikas: " << visasLaikas <<
"\n";
00454
00455 cout << "Paspauskite \"Enter\", kad iseitumete...";
00456 cin.get();
00457 cin.get();
00458
00459 return 0;
00460 }

```

6.5 header.h File Reference

Header failas, kuriame aprašytos visos naudojamos funkcijos.

```

#include "Studentas.h"
#include <iostream>
#include <iomanip>
#include <vector>

```



```
#include <algorithm>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <sstream>
#include <limits>
#include <numeric>
#include <chrono>
#include <filesystem>
#include <list>
#include <deque>
```

Functions

- void [cinEx](#) ()
- [Studentas generuotiPazymius](#) ([Studentas](#) temp)
- [Studentas generuotiVardus](#) ([Studentas](#) temp)
- [Studentas irasytiPazymius](#) ([Studentas](#) temp)
- [Studentas irasytiVarda](#) ([Studentas](#) temp)
- template<typename T>
void [rasytiFaila](#) (string pav, T &v, string pas)
- void [rasytiFaila](#) (string pav, list< [Studentas](#) > &v, string pas)
- template<typename T>
void [skaitytiFaila](#) (string failas, T &A)
- bool [generuotiFaila](#) (string &failas, int ndDydis, int &dydis)
- template<typename T>
string [rusiuotiStudentus](#) (T &A, T &v, T &g, int var)
- template<typename T>
string [rusiuotiStudentus](#) (T &A, T &v, int var)
- string [rusiuotiStudentus](#) (list< [Studentas](#) > &A, list< [Studentas](#) > &v, int var)
- template<typename T>
string [rusiuotiStudentus3](#) (T &A, T &v, int var)
- string [rusiuotiStudentus3](#) (list< [Studentas](#) > &A, list< [Studentas](#) > &v, int var)
- void [testuotiKurima](#) (string &failas, int ndDydis, int &dydis)

6.5.1 Detailed Description

Header failas, kuriame aprašytos visos naudojamos funkcijos.

Author

Eglė

Version

0.1

Date

2025-05-09

Copyright

Copyright (c) 2025

Definition in file [header.h](#).

6.5.2 Function Documentation

6.5.2.1 cinEx()

```
void cinEx ()
```

Definition at line 4 of file [funkcijos.cpp](#).

6.5.2.2 generuotiFaila()

```
bool generuotiFaila (  
    string & failas,  
    int ndDydis,  
    int & dydis)
```

Definition at line 169 of file [funkcijos.cpp](#).

6.5.2.3 generuotiPazymius()

```
Studentas generuotiPazymius (  
    Studentas temp)
```

Definition at line 9 of file [funkcijos.cpp](#).

6.5.2.4 generuotiVardus()

```
Studentas generuotiVardus (  
    Studentas temp)
```

Definition at line 51 of file [funkcijos.cpp](#).

6.5.2.5 irasytiPazymius()

```
Studentas irasytiPazymius (  
    Studentas temp)
```

Definition at line 65 of file [funkcijos.cpp](#).

6.5.2.6 irasytiVarda()

```
Studentas irasytiVarda (  
    Studentas temp)
```

Definition at line 130 of file [funkcijos.cpp](#).

6.5.2.7 rasytiIFaila() [1/2]

```
void rasytiIFaila (  
    string pav,  
    list< Studentas > & v,  
    string pas)
```

Definition at line 142 of file [funkcijos.cpp](#).

6.5.2.8 rasytiIFaila() [2/2]

```
template<typename T>  
void rasytiIFaila (  
    string pav,  
    T & v,  
    string pas)
```

Definition at line 65 of file [header.h](#).

6.5.2.9 rusiuotiStudentus() [1/3]

```
string rusiuotiStudentus (  
    list< Studentas > & A,  
    list< Studentas > & v,  
    int var)
```

Definition at line 198 of file [funkcijos.cpp](#).

6.5.2.10 rusiuotiStudentus() [2/3]

```
template<typename T>  
string rusiuotiStudentus (  
    T & A,  
    T & v,  
    int var)
```

Definition at line 184 of file [header.h](#).

6.5.2.11 rusiuotiStudentus() [3/3]

```
template<typename T>  
string rusiuotiStudentus (  
    T & A,  
    T & v,  
    T & g,  
    int var)
```

Definition at line 141 of file [header.h](#).

6.5.2.12 rusiuotiStudentus3() [1/2]

```
string rusiuotiStudentus3 (  
    list< Studentas > & A,  
    list< Studentas > & v,  
    int var)
```

Definition at line 250 of file [funkcijos.cpp](#).

6.5.2.13 rusiuotiStudentus3() [2/2]

```
template<typename T>  
string rusiuotiStudentus3 (  
    T & A,  
    T & v,  
    int var)
```

Definition at line 231 of file [header.h](#).

6.5.2.14 skaitytiFaila()

```
template<typename T>  
void skaitytiFaila (  
    string failas,  
    T & A)
```

Definition at line 90 of file [header.h](#).

6.5.2.15 testuotiKurima()

```
void testuotiKurima (  
    string & failas,  
    int ndDydis,  
    int & dydis)
```

Definition at line 292 of file [funkcijos.cpp](#).

6.6 header.h

[Go to the documentation of this file.](#)

```
00001  
00011 #ifndef HEADER_H  
00012 #define HEADER_H  
00013  
00014 #include "Studentas.h"  
00015 #include <iostream>  
00016 #include <iomanip>  
00017 #include <vector>  
00018 #include <algorithm>  
00019 #include <cstdlib>  
00020 #include <ctime>  
00021 #include <fstream>  
00022 #include <sstream>  
00023 #include <limits>  
00024 #include <numeric>
```

```

00025 #include <chrono>
00026 #include <filesystem>
00027 #include <list>
00028 #include <deque>
00029
00030 using std::accumulate;
00031 using std::cin;
00032 using std::copy;
00033 using std::cout;
00034 using std::deque;
00035 using std::exception;
00036 using std::fixed;
00037 using std::getline;
00038 using std::ifstream;
00039 using std::ios;
00040 using std::ios_base;
00041 using std::istreamstream;
00042 using std::left;
00043 using std::list;
00044 using std::numeric_limits;
00045 using std::ofstream;
00046 using std::partition;
00047 using std::rand;
00048 using std::setprecision;
00049 using std::setw;
00050 using std::sort;
00051 using std::srand;
00052 using std::streamsize;
00053 using std::string;
00054 using std::time;
00055 using std::to_string;
00056 using std::vector;
00057
00058 void cinEx();
00059 Studentas generuotiPazymius(Studentas temp);
00060 Studentas generuotiVardus(Studentas temp);
00061 Studentas irasytiPazymius(Studentas temp);
00062 Studentas irasytiVarda(Studentas temp);
00063
00064 template <typename T>
00065 void rasytiIFaila(string pav, T &v, string pas)
00066 {
00067     if (pas == "v")
00068     {
00069         sort(v.begin(), v.end(), [](const Studentas &a, const Studentas &b)
00070             { return a.getGalV() > b.getGalV(); });
00071     }
00072     else if (pas == "m")
00073     {
00074         sort(v.begin(), v.end(), [](const Studentas &a, const Studentas &b)
00075             { return a.getGalM() > b.getGalM(); });
00076     }
00077     ofstream fr(pav);
00078     fr << "Vardas      Pavardė          Galutinis (Vid.)  Galutinis (Med.)\n";
00079     fr << "-----\n";
00080     for (const Studentas &i : v)
00081     {
00082         fr << left << setw(12) << i.getVardas() << setw(16) << i.getPavarde();
00083         fr << fixed << setw(17) << setprecision(2) << i.getGalV() << i.getGalM() << "\n";
00084     }
00085 }
00086
00087 void rasytiIFaila(string pav, list<Studentas> &v, string pas);
00088
00089 template <typename T>
00090 void skaitytiFaila(string failas, T &A)
00091 {
00092
00093     vector<int> nd;
00094     try
00095     {
00096         ifstream fd(failas);
00097
00098         if (!fd)
00099         {
00100             throw std::ios_base::failure("Nepavyko atidaryti failo");
00101         }
00102
00103         fd.ignore(numeric_limits<streamsize>::max(), '\n');
00104
00105         string vardas, pavarde;
00106         while (fd >> vardas >> pavarde)
00107         {
00108             int egz = 0;
00109             nd.clear();
00110
00111             string eilute;

```

```

00112         getline(fd, eilute);
00113         istream iss(eilute);
00114
00115         vector<int> visiPazymiai;
00116         int pazimys;
00117         while (iss » pazimys)
00118         {
00119             visiPazymiai.push_back(pazimys);
00120         }
00121
00122         egz = visiPazymiai.back();
00123         visiPazymiai.pop_back();
00124         nd = visiPazymiai;
00125
00126         Studentas temp(vardas, pavarde, nd, egz);
00127         A.push_back(temp);
00128     }
00129     fd.close();
00130 }
00131 catch (ios_base::failure &e)
00132 {
00133     cout << "Nepavyko atidaryti failo\n";
00134     return;
00135 }
00136 }
00137
00138 bool generuotiFaila(string &failas, int ndDydis, int &dydis);
00139
00140 template <typename T>
00141 string rusiuotiStudentus(T &A, T &v, T &g, int var)
00142 {
00143     string pas = "";
00144
00145     if (var == 3)
00146     {
00147         pas = "v";
00148     }
00149     else if (var == 4)
00150     {
00151         pas = "m";
00152     }
00153
00154     while (pas != "v" && pas != "m")
00155     {
00156         cout << "Studentus rusiuoti pagal vidurki ar mediana? (v/m)\n";
00157         cin » pas;
00158     }
00159
00160     if (pas == "v")
00161     {
00162         for (Studentas st : A)
00163         {
00164             if (st.getGalV() < 5.0)
00165                 v.push_back(st);
00166             else
00167                 g.push_back(st);
00168         }
00169     }
00170     else
00171     {
00172         for (Studentas st : A)
00173         {
00174             if (st.getGalM() < 5.0)
00175                 v.push_back(st);
00176             else
00177                 g.push_back(st);
00178         }
00179     }
00180     return pas;
00181 }
00182
00183 template <typename T>
00184 string rusiuotiStudentus(T &A, T &v, int var)
00185 {
00186     string pas = "";
00187
00188     if (var == 3)
00189     {
00190         pas = "v";
00191     }
00192     else if (var == 4)
00193     {
00194         pas = "m";
00195     }
00196
00197     while (pas != "v" && pas != "m")
00198     {

```

```

00199         cout << "Studentus rusiuoti pagal vidurki ar mediana? (v/m)\n";
00200         cin >> pas;
00201     }
00202
00203     if (pas == "v")
00204     {
00205         for (int i = A.size() - 1; i >= 0; --i)
00206         {
00207             if (A[i].getGalV() < 5.0)
00208             {
00209                 v.push_back(A[i]);
00210                 A.pop_back();
00211             }
00212         }
00213     }
00214     else
00215     {
00216         for (int i = A.size() - 1; i >= 0; --i)
00217         {
00218             if (A[i].getGalM() < 5.0)
00219             {
00220                 v.push_back(A[i]);
00221                 A.pop_back();
00222             }
00223         }
00224     }
00225     return pas;
00226 }
00227
00228 string rusiuotiStudentus(list<Studentas> &A, list<Studentas> &v, int var);
00229
00230 template <typename T>
00231 string rusiuotiStudentus3(T &A, T &v, int var)
00232 {
00233     string pas = "";
00234
00235     if (var == 3)
00236     {
00237         pas = "v";
00238     }
00239     else if (var == 4)
00240     {
00241         pas = "m";
00242     }
00243
00244     while (pas != "v" && pas != "m")
00245     {
00246         cout << "Studentus rusiuoti pagal vidurki ar mediana? (v/m)\n";
00247         cin >> pas;
00248     }
00249
00250     if (pas == "v")
00251     {
00252         A.erase(remove_if(A.begin(), A.end(), [&](const Studentas &st)
00253         {
00254             if (st.getGalV() < 5.0)
00255             {
00256                 v.push_back(st);
00257                 return true;
00258             }
00259             return false; })),
00260             A.end());
00261     }
00262     else
00263     {
00264         A.erase(remove_if(A.begin(), A.end(), [&](const Studentas &st)
00265         {
00266             if (st.getGalM() < 5.0)
00267             {
00268                 v.push_back(st);
00269                 return true;
00270             }
00271             return false; })),
00272             A.end());
00273     }
00274     return pas;
00275 }
00276
00277 string rusiuotiStudentus3(list<Studentas> &A, list<Studentas> &v, int var);
00278
00279 void testuotiKurima(string &failas, int ndDydis, int &dydis);
00280
00281 #endif

```

6.7 README.md File Reference

6.8 studentas.h File Reference

Studento klasė

```
#include "zmogus.h"
#include <string>
#include <vector>
#include <algorithm>
#include <iostream>
#include <sstream>
```

Classes

- class [Studentas](#)

Variables

- bool [TEST_MODE](#) = false

6.8.1 Detailed Description

Studento klasė

Author

Eglė

Version

0.1

Date

2025-05-07

Copyright

Copyright (c) 2025

Definition in file [studentas.h](#).

6.8.2 Variable Documentation

6.8.2.1 TEST_MODE

bool TEST_MODE = false [inline]

Definition at line 30 of file [studentas.h](#).

6.9 studentas.h

[Go to the documentation of this file.](#)

```

00001
00011 #ifndef STUDENT_H
00012 #define STUDENT_H
00013
00014 #include "zmogus.h"
00015 #include <string>
00016 #include <vector>
00017 #include <algorithm>
00018 #include <iostream>
00019 #include <sstream>
00020
00021 using std::cout;
00022 using std::istream;
00023 using std::istringstream;
00024 using std::move;
00025 using std::ostream;
00026 using std::sort;
00027 using std::string;
00028 using std::vector;
00029
00030 inline bool TEST_MODE = false;
00031
00032 class Studentas : public Zmogus
00033 {
00034 private:
00035     string vardas;
00036     string pavarde;
00037     vector<int> nd;
00038     int egz;
00039     float galutinisV;
00040     float galutinisM;
00041
00042 public:
00047     Studentas() : egz(0) {}
00056     Studentas(string v, string p, vector<int> n, int e)
00057     {
00058         vardas = v;
00059         pavarde = p;
00060         nd = n;
00061         egz = e;
00062         galutinisV = SkaiciuotiV();
00063         galutinisM = SkaiciuotiM();
00064     };
00070     Studentas(const Studentas &st)
00071     {
00072         if (TEST_MODE)
00073             cout << "Studento kopijavimo konstruktorius\n";
00074         vardas = st.vardas;
00075         pavarde = st.pavarde;
00076         nd.resize(st.nd.size());
00077         copy(st.nd.begin(), st.nd.end(), nd.begin());
00078         egz = st.egz;
00079         galutinisV = st.galutinisV;
00080         galutinisM = st.galutinisM;
00081     }
00088     Studentas &operator=(const Studentas &st)
00089     {
00090         if (TEST_MODE)
00091             cout << "Studento kopijavimo operacija\n";
00092         if (this == &st)
00093             return *this;
00094
00095         if (!nd.empty())
00096             nd.clear();
00097

```

```

00098         vardas = st.vardas;
00099         pavarde = st.pavarde;
00100         nd.resize(st.nd.size());
00101         copy(st.nd.begin(), st.nd.end(), nd.begin());
00102         egz = st.egz;
00103         galutinisV = st.galutinisV;
00104         galutinisM = st.galutinisM;
00105
00106         return *this;
00107     }
00113     Studentas(Studentas &&st) noexcept
00114     {
00115         if (TEST_MODE)
00116             cout << "Studento perkeliimo konstruktorius\n";
00117         vardas = move(st.vardas);
00118         pavarde = move(st.pavarde);
00119         nd = move(st.nd);
00120         egz = st.egz;
00121         galutinisV = st.galutinisV;
00122         galutinisM = st.galutinisM;
00123         st.clear();
00124     }
00131     Studentas &operator=(Studentas &&st) noexcept
00132     {
00133         if (TEST_MODE)
00134             cout << "Studento perkeliimo operacija\n";
00135         if (this == &st)
00136             return *this;
00137
00138         vardas = move(st.vardas);
00139         pavarde = move(st.pavarde);
00140         nd = move(st.nd);
00141         egz = st.egz;
00142         galutinisV = st.galutinisV;
00143         galutinisM = st.galutinisM;
00144         st.clear();
00145
00146         return *this;
00147     }
00155     friend ostream &operator<<(ostream &os, const Studentas &st)
00156     {
00157         os << st.vardas << " " << st.pavarde << " " << st.galutinisV << " " << st.galutinisM << "\n";
00158         return os;
00159     }
00167     friend istream &operator>>(istream &is, Studentas &st)
00168     {
00169         is >> st.vardas >> st.pavarde;
00170         st.nd.clear();
00171
00172         string eilute;
00173         getline(is, eilute);
00174         istringstream iss(eilute);
00175
00176         vector<int> visiPazymiai;
00177         int paz;
00178
00179         while (iss >> paz)
00180         {
00181             visiPazymiai.push_back(paz);
00182         }
00183
00184         st.egz = visiPazymiai.back();
00185         visiPazymiai.pop_back();
00186         st.nd = visiPazymiai;
00187         st.galutinisV = st.SkaiciuotiV();
00188         st.galutinisM = st.SkaiciuotiM();
00189
00190         return is;
00191     }
00198     void setVarPav(string v, string p)
00199     {
00200         vardas = v;
00201         pavarde = p;
00202     }
00209     void setPaz(vector<int> n, int e)
00210     {
00211         nd = n;
00212         egz = e;
00213         galutinisV = SkaiciuotiV();
00214         galutinisM = SkaiciuotiM();
00215     }
00216     inline string getVardas() const override { return vardas; }
00217     inline string getPavarde() const override { return pavarde; }
00218     inline int getEgz() const { return egz; }
00219     inline vector<int> getNd() const { return nd; }
00220     inline int getGalV() const { return galutinisV; }
00221     inline int getGalM() const { return galutinisM; }

```

```

00227     float SkaiciuotiV()
00228     {
00229         int s = 0;
00230         for (int i = 0; i < nd.size(); i++)
00231         {
00232             s += nd[i];
00233         }
00234         float galutinis = 0.4 * (1.0 * s / nd.size()) + 0.6 * egz;
00235         return galutinis;
00236     }
00242     float SkaiciuotiM()
00243     {
00244
00245         float paz;
00246         sort(nd.begin(), nd.end());
00247
00248         if (nd.size() % 2 == 0)
00249         {
00250             paz = 1.0 * (nd[nd.size() / 2 - 1] + nd[nd.size() / 2]) / 2;
00251         }
00252         else
00253         {
00254             paz = nd[nd.size() / 2];
00255         }
00256
00257         float galutinis = 0.4 * paz + 0.6 * egz;
00258         return galutinis;
00259     }
00264     void clear()
00265     {
00266         if (TEST_MODE)
00267             cout << "Studento destruktoriaus\n";
00268         vardas.clear();
00269         pavarde.clear();
00270         nd.clear();
00271         egz = 0;
00272         galutinisV = 0;
00273         galutinisM = 0;
00274     }
00275     ~Studentas() { clear(); }
00276 };
00277
00278 #endif

```

6.10 timeris.h File Reference

Klasė naudojama atliekamų funkcijų matavimui.

```
#include "header.h"
```

Classes

- class [Timer](#)

6.10.1 Detailed Description

Klasė naudojama atliekamų funkcijų matavimui.

Author

Eglė

Version

0.1

Date

2025-05-09

Copyright

Copyright (c) 2025

Definition in file [timeris.h](#).

6.11 timeris.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef TIMER_H
00012 #define TIMER_H
00013
00014 #include "header.h"
00015
00016 class Timer
00017 {
00018 private:
00019     using hrClock = std::chrono::high_resolution_clock;
00020     using durationDouble = std::chrono::duration<double>;
00021     std::chrono::time_point<hrClock> start;
00022
00023 public:
00028     Timer() : start{hrClock::now()} {}
00033     void reset()
00034     {
00035         start = hrClock::now();
00036     }
00042     double elapsed() const
00043     {
00044         return durationDouble(hrClock::now() - start).count();
00045     }
00046 };
00047
00048 #endif
```

6.12 zmogus.h File Reference

```
#include <string>
```

Classes

- class [Zmogus](#)

6.13 zmogus.h

[Go to the documentation of this file.](#)

```
00001 #ifndef ZMOGUS_H
00002 #define ZMOGUS_H
00003
00004 #include <string>
00005
00006 using std::string;
00007
00008 class Zmogus
00009 {
00010 protected:
00011     string vardas;
00012     string pavarde;
00013
00014 public:
00015     Zmogus() {}
00016     Zmogus(string v, string p)
00017     {
00018         vardas = v;
00019         pavarde = p;
00020     };
00021     void setVarPav(string v, string p)
00022     {
00023         vardas = v;
00024         pavarde = p;
00025     }
00026     virtual string getVardas() const = 0;
00027     virtual string getPavarde() const = 0;
00028     virtual ~Zmogus() {}
00029 };
00030
00031 #endif
```


Index

- ~Studentas
 - Studentas, [15](#)
- ~Zmogus
 - Zmogus, [21](#)
- 2.0 versija, [1](#)
- cinEx
 - funkcijos.cpp, [23](#)
 - header.h, [36](#)
- clear
 - Studentas, [15](#)
- elapsed
 - Timer, [20](#)
- funkcijos.cpp, [23](#)
 - cinEx, [23](#)
 - generuotiFaila, [23](#)
 - generuotiPazymius, [23](#)
 - generuotiVardus, [24](#)
 - irasytiPazymius, [24](#)
 - irasytiVarda, [24](#)
 - rasytilFaila, [24](#)
 - rusiuotiStudentus, [24](#)
 - rusiuotiStudentus3, [24](#)
 - testuotiKurima, [25](#)
- galutinis_balas.cpp, [29](#)
 - main, [29](#)
- generuotiFaila
 - funkcijos.cpp, [23](#)
 - header.h, [36](#)
- generuotiPazymius
 - funkcijos.cpp, [23](#)
 - header.h, [36](#)
- generuotiVardus
 - funkcijos.cpp, [24](#)
 - header.h, [36](#)
- getEgz
 - Studentas, [15](#)
- getGalM
 - Studentas, [16](#)
- getGalV
 - Studentas, [16](#)
- getNd
 - Studentas, [16](#)
- getPavarde
 - Studentas, [16](#)
 - Zmogus, [21](#)
- getVardas
 - Studentas, [16](#)
 - Zmogus, [21](#)
- header.h, [34](#)
 - cinEx, [36](#)
 - generuotiFaila, [36](#)
 - generuotiPazymius, [36](#)
 - generuotiVardus, [36](#)
 - irasytiPazymius, [36](#)
 - irasytiVarda, [36](#)
 - rasytilFaila, [36, 37](#)
 - rusiuotiStudentus, [37](#)
 - rusiuotiStudentus3, [37, 38](#)
 - skaitytiFaila, [38](#)
 - testuotiKurima, [38](#)
- irasytiPazymius
 - funkcijos.cpp, [24](#)
 - header.h, [36](#)
- irasytiVarda
 - funkcijos.cpp, [24](#)
 - header.h, [36](#)
- main
 - galutinis_balas.cpp, [29](#)
- operator<<
 - Studentas, [18](#)
- operator>>
 - Studentas, [18](#)
- operator=
 - Studentas, [16, 17](#)
- pavarde
 - Zmogus, [22](#)
- rasytilFaila
 - funkcijos.cpp, [24](#)
 - header.h, [36, 37](#)
- README.md, [42](#)
- reset
 - Timer, [20](#)
- rusiuotiStudentus
 - funkcijos.cpp, [24](#)
 - header.h, [37](#)
- rusiuotiStudentus3
 - funkcijos.cpp, [24](#)
 - header.h, [37, 38](#)
- setPaz
 - Studentas, [17](#)

- setVarPav
 - Studentas, [17](#)
 - Zmogus, [21](#)
- SkaiciuotiM
 - Studentas, [18](#)
- SkaiciuotiV
 - Studentas, [18](#)
- skaitytiFaila
 - header.h, [38](#)
- Studentas, [13](#)
 - ~Studentas, [15](#)
 - clear, [15](#)
 - getEgz, [15](#)
 - getGalM, [16](#)
 - getGalV, [16](#)
 - getNd, [16](#)
 - getPavarde, [16](#)
 - getVardas, [16](#)
 - operator<<, [18](#)
 - operator>>, [18](#)
 - operator=, [16](#), [17](#)
 - setPaz, [17](#)
 - setVarPav, [17](#)
 - SkaiciuotiM, [18](#)
 - SkaiciuotiV, [18](#)
 - Studentas, [14](#), [15](#)
- studentas.h, [42](#)
 - TEST_MODE, [43](#)
- TEST_MODE
 - studentas.h, [43](#)
- testuotiKurima
 - funkcijos.cpp, [25](#)
 - header.h, [38](#)
- Timer, [19](#)
 - elapsed, [20](#)
 - reset, [20](#)
 - Timer, [19](#)
- timeris.h, [45](#)
- vardas
 - Zmogus, [22](#)
- Zmogus, [20](#)
 - ~Zmogus, [21](#)
 - getPavarde, [21](#)
 - getVardas, [21](#)
 - pavarde, [22](#)
 - setVarPav, [21](#)
 - vardas, [22](#)
 - Zmogus, [21](#)
- zmogus.h, [46](#)