Galutinis Balas

3.0

Generated by Doxygen 1.13.2

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 stuktūros 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/Objektinis.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/įvedimas

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 Vector klasės funkcijos

Šioje versijoje buvo realizuota nuosava vektoriaus klasė, čia pateiksiu kelių funkcijų implementacijos pavyzdžius:

1. Push_back(elementas)

   Jei vektoriaus viduje esancio masyvo dydis yra mažesnis už talpą, prie masyvo galo pridedamas naujas elementas, o kintamasis, kuris nurodo masyvo dydi, padidinamas vienetu.

   Jei vektorias viduje esančio masyvo dydis yra lygus talpai, tai pirma visi elementai perkopijuojami į nauja masyvą, kurio talpa būtų dvigubai didesnė už praeito, senasis masyvas ištrinamas, o prie naujojo galo pridedamas naujas elementas.

   Pvz:

   ```
   Vector<int> v;
   v.Push_back(10);
   v.Push_back(20);
   cout << v[0] << " " << v[1] << endl;
   ```

   Terminale gausite

   ```
   10 20
   ```

1. Shrink_to_fit()

   Naudojamas norint sumažinti vektoriaus viduje esančio masyvo talpą, kad ji būtų lygi masyvo dydžiui.

   Jei dydis jau yra lygus talpai, grįžtama iš funkcijos ir niekas neįvykta.

   Kitu atveju sukuriamas naujas masyvas, kurio talpa būtu lygi senojo masyvo dydžiui, į jį nukopijuojami visi elementai, esantys senajame masyve, senasis ištrinamas, o naujasis masyvas tampa vektoriaus masyvu.

   Pvz:

```
Vector<int> v;
v.Push_back(10);
v.Push_back(20);
v.Push_back(30);
cout << v.Capacity() << endl;
v.Shrink_to_fit();
cout << v.Capacity() << endl;
```

Terminale gausite

```
4
3
```

1. == operatorius

   Lygina, ar kiekvienas pirmojo vektoriaus elementas yra lygus antrojo vektoriaus elementui tame pačiame indekse.

   Jei dydžiai skiriasi, grąžinama false.

   Jei bent vienas elementas iš primojo vektoriaus neatitinka tame pačiame indekse esančio elemento iš antrojo vektoriaus, grąžinama false.

   Kitu atveju grąžinama true.

   Pvz:

```
Vector<int> a{1, 2, 3}, b{1, 2, 3}, c{1, 2, 3, 4};
cout << (a == b) << " " << (a == c) << endl;
```

   Terminale gausite

```
1 0
```

1. Resize(naujas_dydis(, reikšmė))

   Keičia vektoriaus viduje esančio masyvo dydį.

   Jei naujasis dydis lygus esamam dydžiui, grįžtama.

   Jei naujas dydis yra mažesnis už esamą, masyvo dydį rodančiam kintamajui priskiriama naujojo dydžio reikšmė.

   Jei naujas dydis didesnis už esamą, kuriamas naujas masyvas, kuriam rezervuojama didesnė reikšmė tarp naujojo dydžio ir talpos∗2. Į jį nukopijuojami visi senojo masyvo elementai ir nuo pabaigos iki norimo dydžio indeksų pridedama tuščio konstruktoriaus sukurtas elementas (jei nėra reikšmės) arba norima reikmė (jei ji yra).

   Pvz:

```
Vector<int> d = {1, 2, 3};
for (int i = 0; i < d.Size(); ++i)
{
    cout << d[i] << " ";
}
cout << endl;
d.Resize(5);
for (int i = 0; i < d.Size(); ++i)
{
    cout << d[i] << " ";
}
cout << endl;
d.Resize(2);
for (int i = 0; i < d.Size(); ++i)
{
    cout << d[i] << " ";
}
cout << endl;
d.Resize(6, 4);
for (int i = 0; i < d.Size(); ++i)
{
    cout << d[i] << " ";
}
cout << endl;
```

Terminale gausime

```
1 2 3
1 2 3 0 0
1 2
1 2 4 4 4 4
```

1. = operatorius

   Nukopijuoja visus antrojo vektoriaus elementus ir kintamuosius į pirmąjį.

   Pvz:

```
Vector<int> a{1, 2, 3};
Vector<int> b;
b = a;
for (int i=0; i < b.Size(); ++i)
    cout << b[i] << " ";
cout << endl;
```

   Terminale gausite

```
1 2 3
```

1.6 spartumo testavimai

Buvo atlikti testai, kurie lygino mano sukurtos Vector klasės ir įprasto vector konteinerio užtruktą laiką push_back() funkcija pildant juos tam tikru kiekiu elementų (skaičių). Atlikto testo rezultatai:

Tipas	10,000	100,000	1,000,000	10,000,000	100,000,000
Vector klasė	0.0000834	0.000503	0.00229	0.0231	0.201
vector konteineris	0.0000573	0.000345	0.00282	0.0258	0.227

Kaip galima pastebėti iš rezultatų, mano klasės vektorius yra nežymiai greitesnis, kai į vektorių dedama iki 100,000 skaičių, po to standartinis vektorius tampa greitesnis.

Abiejų vektorių atminties perskirstymu įvyksta tiek pat: 27.

1.7 Dokumentacija

Dokumentacijos failai yra docs/ aplanke. Ten rasite html/ ir latex/ aplankus taip pat ir jau pilnai sugeneruotą .pdf pavidalo dokumentaciją. Norint dokumentaciją matyti naršyklėje, html/ aplanke susiraskite failą "index.html" ir atidarykite jį naudodamiesi "Live Server" ar kitu .html failų paleidimo būdu.

1.8 Testavimo sistemos parametrai

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

1.9 Spartos Testavimo Rezultatai

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

1.9.1 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

1.9.2 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

1.9.3 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.

1.10 Vektorių spartos palyginimas

Žemiau pateiktose lentelėse pateikti vektorių laikai:

1.10.1 įprastas vector

Failo dydis	Duomenų skaitymas	Duomenų rūšiavimas	Studentų rūšiavimas	Išvedimas į failus	Bendras laikas
100,000 įrašų	0.451	0.0130	0.024	0.138	0.707
1,000,000 įrašų	4.934	0.101	0.234	1.103	6.478
10,000,000 įrašų	41.759	0.718	2.920	11.211	56.613

1.10.2 mano vector klasė

Failo dydis	Duomenų skaitymas	Duomenų rūšiavimas	Studentų rūšiavimas	Išvedimas į failus	Bendras laikas
100,000 įrašų	0.489	0.0121	0.025	0.177	1.585
1,000,000 įrašų	5.443	0.196	0.325	1.683	7.653
10,000,000 įrašų	37.09	1.188	2.603	9.946	50.837

Skirtumai nežymūs, mano vektorius yra greitesnis naudojant 10,000,000 studentų, tačiau lėtesnis kitais atvejais.

1.11 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

1.11.1 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

1.11.2 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

1.11.3 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 startegija yra greičiausia su visais konteineriais išskyrus list'ą. Norint naudoti list'ą būtų geriausia taikyti 2 strategiją. 2 strategijos neefktyvumą, 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.

1.12 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:

1.12.1 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

1.12.2 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.

1.13 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:

1.13.1 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

1.13.2 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.

1.14 Nuotraukos

Programos veikimo pavyzdys:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

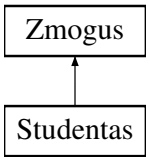
Chapter 5

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 perkelimo konstruktorius.
- Studentas & operator= (Studentas &&st) noexcept
  Studento perkelimo operacija.
- bool operator== (const Studentas &rhs)
- 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 ()

  Skaičiuojamas studento galutinis balas pagal vidurkį
- float SkaiciuotiM ()

  Skaičiuojamas 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

v	Vardas
p	Pavardė
n	Namų darbų pažymiai (vektorius)
e	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

st	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 perkelimo konstruktorius.

Parameters

st	Studentas, kurio duomenis norima perkelti

Definition at line 113 of file studentas.h.

5.1.2.5 ~Studentas()

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

Definition at line 279 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 268 of file studentas.h.

5.1.3.2 getEgz()

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

Definition at line 222 of file studentas.h.

5.1.3.3 getGalM()

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

Definition at line 225 of file studentas.h.

5.1.3.4 getGalV()

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

Definition at line 224 of file studentas.h.

5.1.3.5 getNd()

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

Definition at line 223 of file studentas.h.

5.1.3.6 getPavarde()

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

Implements Zmogus.

Definition at line 221 of file studentas.h.

5.1.3.7 getVardas()

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

Implements Zmogus.

Definition at line 220 of file studentas.h.

5.1.3.8 operator=() [1/2]

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

Kopijavimo operacija.

Parameters

st	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 perkelimo operacija.

Parameters

st	Studentas, kurio duomenis norima perkelti

Returns

Studentas, į kurį buvo perkelti duomenys

Definition at line 131 of file studentas.h.

5.1.3.10 operator==()

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

Definition at line 148 of file studentas.h.

5.1.3.11 setPaz()

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

Studento namų darbų ir egzamino pažymių set'eris.

Parameters

n	namų darbų pažymių vektorius
e	egzamino pažymys

Definition at line 213 of file studentas.h.

5.1.3.12 setVarPav()

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

Studento vardo ir pavardės set'eris.

Parameters

v	Vardas
p	Pavardė

Definition at line 202 of file studentas.h.

5.1.3.13 SkaiciuotiM()

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

Skaičiuojamas studento galutinis balas pagal medianą

Returns

Grąžinama suskaičiuota galutinio balo reikšmė, kuri išsaugoma "galutinisM"

Definition at line 246 of file studentas.h.

5.1.3.14 SkaiciuotiV()

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

Skaičiuojamas studento galutinis balas pagal vidurkį

Returns

Grąžinama suskaičiuota galutinio balo reikšmė, kuri išsaugoma "galutinisV"

Definition at line 231 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

os	Būdas, kuriuo išvedama
st	Išvedamas studentas

Returns

ostream&

Definition at line 159 of file studentas.h.

5.1.4.2 operator>>

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

Studento įvedimo operacija.

Parameters

is	Būdas, kuriuo įvedama
st	Įvedamas studentas

Returns

istream&

Definition at line 171 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 Vector< T > Class Template Reference

```
#include <vector.h>
```

Public Types

- using value_type = T
- using reference = T &
- using const_reference = const T &
- using pointer = T *
- using const_pointer = const T *
- using iterator = T *
- using const_iterator = const T *
- using reverse_iterator = T *
- using const_reverse_iterator = const T *
- using allocator_type = std::allocator<T>

Public Member Functions

- Vector ()

  Naujo vektoriaus konstruktorius.
- Vector (const Vector< T > &rhs)

  Kopijavimo konstruktorius.
- Vector (Vector< T > &&rhs) noexcept

  Vektoriaus perkėlimo konstruktorius.
- Vector (int elements, const T &value=T())

  Vektoriaus konstruktorius, kuris užpildo vektorių tam tikru skaičiumi ir tuo pačiu objektu.
- Vector (const std::initializer_list< T > &list)

  Vektoriaus konstruktorius iš pateikto sąrašo.
- ~Vector ()

  Vektoriaus destruktorius.
- void push_back (const T &value)

  prideda duotą objektą į vektoriaus galą
- void pop_back ()

  pašalina paskutinį vektoriaus elementą
- bool empty () const

  tikrina, ar vektorius tuščias
- int size () const

  grąžina vektoriaus dydį
- int capacity () const

  grąžina vektoriaus talpą
- void reserve (int new_cap)

  Pakeičia vektoriaus talpą į duotąją, jei ji nėra mažesnė už dabartinę
- void shrink_to_fit ()

  sumažina vektoriaus talpą iki jo dydžio
- bool operator== (const Vector< T > &rhs) const

  lygybės operatorius
- bool operator!= (const Vector< T > &rhs) const

  nelygybės operatorius
- bool operator> (const Vector< T > &rhs) const

  daugiau operatorius, pirmi nesutampantys elementai tame pačiame indekse nustato grąžinamą reikšmę, jei visi elementai lygūs, didesnis yra tas, kuris turi daugiau elementų
- bool operator>= (const Vector< T > &rhs) const

  daugiau operatorius, pirmi nesutampantys elementai tame pačiame indekse nustato grąžinamą reikšmę, jei visi elementai lygūs, didesnis yra tas, kuris turi daugiau elementų
- bool operator< (const Vector< T > &rhs) const

  daugiau operatorius, pirmi nesutampantys elementai tame pačiame indekse nustato grąžinamą reikšmę, jei visi elementai lygūs, didesnis yra tas, kuris turi daugiau elementų
- bool operator<= (const Vector< T > &rhs) const

  daugiau operatorius, pirmi nesutampantys elementai tame pačiame indekse nustato grąžinamą reikšmę, jei visi elementai lygūs, didesnis yra tas, kuris turi daugiau elementų
- Vector< T > & operator= (const Vector< T > &rhs)

  Kopijavimo operacija.
- Vector< T > & operator= (Vector< T > &&rhs) noexcept

  Perkėlimo operacija.
- allocator_type get_allocator () const
- value_type & operator[ ] (int index)

  indekso operatorius
- value_type & at (int index)

indekso funkcija

- value_type & front ()

  Pirmasis vektoriaus elementas.
- value_type & back ()

  Paskutinis vektoriaus elementas.
- pointer data ()
- const_pointer data () const
- void insert (int index, const T &value)

  įterpimas
- void erase (int index)

  elemento pašalinimas
- iterator erase (const_iterator pos)
- iterator erase (const_iterator first, const_iterator last)
- void clear ()

  viso vektoriaus išvalymas
- template<typename... Args>
  void emplace_back (Args &&...args)

  Elemento įterpimas gale.
- template<typename... Args>
  T * emplace (int index, Args &&...args)

  Elemento įterpimas indekse.
- void assign (int count, const T &value)

  Priskyrimas, visi prieš tai buvę elementai pašalinami, o naujasis įterpiamas tiek kartų, koks yra 'count'.
- void assign (std::initializer_list< T > ilist)

  Priskyrimas, visi prieš tai buvę elementai pašalinami, o vietoj tų elementų įterpiami sąrašo elementai.
- iterator begin ()

  Pradžios iteratorius.
- iterator end ()

  Pabaigos iteratorius, nurodo sekančią vietą po paskutinio vektoriaus elemento.
- const_iterator begin () const
- const_iterator end () const
- const_iterator cbegin () const
- const_iterator cend () const
- iterator rbegin ()

  Atvirkštinis pradžios iteratorius nurodo sekančią vietą prieš pirmąjį vektoriaus elementą
- iterator rend ()

  Atvirkštinis pabaigos iteratorius.
- const_iterator rbegin () const
- const_iterator rend () const
- void resize (int count)

  Vektoriaus dydžio pakeitimas, tuščios vektoriaus vietos užpildomos numatytu objektu.
- void resize (int count, const T &value)

  Vektoriaus dydžio pakeitimas, tuščios vektoriaus vietos užpildomos duotuoju objektu.

5.3.1 Detailed Description

template<typename T>
class Vector< T >

Definition at line 18 of file vector.h.

5.3.2 Member Typedef Documentation

5.3.2.1 allocator_type

```
template<typename T>
using Vector< T >::allocator_type = std::allocator<T>
```

Definition at line 30 of file vector.h.

5.3.2.2 const_iterator

```
template<typename T>
using Vector< T >::const_iterator = const T *
```

Definition at line 27 of file vector.h.

5.3.2.3 const_pointer

```
template<typename T>
using Vector< T >::const_pointer = const T *
```

Definition at line 25 of file vector.h.

5.3.2.4 const_reference

```
template<typename T>
using Vector< T >::const_reference = const T &
```

Definition at line 23 of file vector.h.

5.3.2.5 const_reverse_iterator

```
template<typename T>
using Vector< T >::const_reverse_iterator = const T *
```

Definition at line 29 of file vector.h.

5.3.2.6 iterator

```
template<typename T>
using Vector< T >::iterator = T *
```

Definition at line 26 of file vector.h.

5.3.2.7 pointer

```
template<typename T>
using Vector< T >::pointer = T *
```

Definition at line 24 of file vector.h.

5.3.2.8 reference

```
template<typename T>
using Vector< T >::reference = T &
```

Definition at line 22 of file vector.h.

5.3.2.9 reverse_iterator

```
template<typename T>
using Vector< T >::reverse_iterator = T *
```

Definition at line 28 of file vector.h.

5.3.2.10 value_type

```
template<typename T>
using Vector< T >::value_type = T
```

Definition at line 21 of file vector.h.

5.3.3 Constructor & Destructor Documentation

5.3.3.1 Vector() [1/5]

```
template<typename T>
Vector< T >::Vector ()
```

Naujo vektoriaus konstruktorius.

5.3.3.2 Vector() [2/5]

```
template<typename T>
Vector< T >::Vector (
            const Vector< T > & rhs)
```

Kopijavimo konstruktorius.

Parameters

rhs	Vektorius, iš kurio kopijuojama

5.3.3.3 Vector() [3/5]

```
template<typename T>
Vector< T >::Vector (
            Vector< T > && rhs)  [noexcept]
```

Vektoriaus perkėlimo konstruktorius.

Parameters

rhs	Vektorius, kurio duomenys perkeliami

5.3.3.4 Vector() [4/5]

```
template<typename T>
Vector< T >::Vector (
            int elements,
            const T & value = T())
```

Vektoriaus konstruktorius, kuris užpildo vektorių tam tikru skaičiumi ir tuo pačiu objektu.

Parameters

elements	Nusako kokio dydžio vektoriaus norime
value	Objekto reikšmė, kuria bus užpildomas vektorius

5.3.3.5 Vector() [5/5]

```
template<typename T>
Vector< T >::Vector (
            const std::initializer_list< T > & list)
```

Vektoriaus konstruktorius iš pateikto sąrašo.

Parameters

list	sąrašas, kurį norima įdėti į vektorių

5.3.3.6 ~Vector()

```
template<typename T>
Vector< T >::~Vector ()
```

Vektoriaus destruktorius.

5.3.4 Member Function Documentation

5.3.4.1 assign() [1/2]

```
template<typename T>
void Vector< T >::assign (
            int count,
            const T & value)
```

Priskyrimas, visi prieš tai buvę elementai pašalinami, o naujasis įterpiamas tiek kartų, koks yra 'count'.

Parameters

count	Norimas elementų skaičius
value	priskiriama reikšmė

5.3.4.2 assign() [2/2]

```
template<typename T>
void Vector< T >::assign (
            std::initializer_list< T > ilist)
```

Priskyrimas, visi prieš tai buvę elementai pašalinami, o vietoj tų elementų įterpiami sąrašo elementai.

Parameters

ilist	Sąrašas

5.3.4.3 at()

```
template<typename T>
value_type & Vector< T >::at (
            int index)
```

indekso funkcija

Parameters

index	Norimas indeksas

Returns

value_type& Elementas tame indekse

5.3.4.4 back()

```
template<typename T>
value_type & Vector< T >::back ()
```

Paskutinis vektoriaus elementas.

Returns

value_type& elemento adresas

5.3.4.5 begin() [1/2]

```
template<typename T>
iterator Vector< T >::begin ()  [inline]
```

Pradžios iteratorius.

Returns

iterator Iteratorius, kuris nurodo vektoriaus pradžią

Definition at line 276 of file vector.h.

5.3.4.6 begin() [2/2]

```
template<typename T>
const_iterator Vector< T >::begin () const  [inline]
```

Definition at line 283 of file vector.h.

5.3.4.7 capacity()

```
template<typename T>
int Vector< T >::capacity () const
```

grąžina vektoriaus talpą

Returns

int Talpa

5.3.4.8 cbegin()

```
template<typename T>
const_iterator Vector< T >::cbegin () const  [inline]
```

Definition at line 285 of file vector.h.

5.3.4.9 cend()

```
template<typename T>
const_iterator Vector< T >::cend () const  [inline]
```

Definition at line 286 of file vector.h.

5.3.4.10 clear()

```
template<typename T>
void Vector< T >::clear ()
```

viso vektoriaus išvalymas

5.3.4.11 data() [1/2]

```
template<typename T>
pointer Vector< T >::data ()
```

5.3.4.12 data() [2/2]

```
template<typename T>
const_pointer Vector< T >::data () const
```

5.3.4.13 emplace()

```
template<typename T>
template<typename...  Args>
T * Vector< T >::emplace (
            int index,
            Args &&...  args)
```

Elemento įterpimas indekse.

Template Parameters

Args	Objekto tipas

Parameters

index	Indeksas, kuriame norima įterpti
args	Objektas, kurį norima įterpti

Returns

T*

5.3.4.14 emplace_back()

```
template<typename T>
template<typename...  Args>
void Vector< T >::emplace_back (
            Args &&...  args)
```

Elemento įterpimas gale.

Template Parameters

Args	Objekto tipas

Parameters

args	Objektas, kurį norima įterpti

5.3.4.15 empty()

```
template<typename T>
bool Vector< T >::empty () const
```

tikrina, ar vektorius tuščias

Returns

true Vektorius tuščias

false Vektorius ne tuščias

5.3.4.16 end() [1/2]

```
template<typename T>
iterator Vector< T >::end ()  [inline]
```

Pabaigos iteratorius, nurodo sekančią vietą po paskutinio vektoriaus elemento.

Returns

iterator Iteratorius, kuris nurodo vektoriaus pabaigą

Definition at line 282 of file vector.h.

5.3.4.17 end() [2/2]

```
template<typename T>
const_iterator Vector< T >::end () const  [inline]
```

Definition at line 284 of file vector.h.

5.3.4.18 erase() [1/3]

```
template<typename T>
iterator Vector< T >::erase (
            const_iterator first,
            const_iterator last)
```

5.3.4.19 erase() [2/3]

```
template<typename T>
iterator Vector< T >::erase (
            const_iterator pos)
```

5.3.4.20 erase() [3/3]

```
template<typename T>
void Vector< T >::erase (
            int index)
```

elemento pašalinimas

Parameters

index	indeksas, kurio vietoje esantį objektą norima pašalinti

5.3.4.21 front()

```
template<typename T>
value_type & Vector< T >::front ()
```

Pirmasis vektoriaus elementas.

Returns

value_type& elemento adresas

5.3.4.22 get_allocator()

```
template<typename T>
allocator_type Vector< T >::get_allocator () const
```

5.3.4.23 insert()

```
template<typename T>
void Vector< T >::insert (
            int index,
            const T & value)
```

įterpimas

Parameters

index	indeksas, po kurio norimma įterpti
value	objektas, kurį norima įterpti

5.3.4.24 operator"!=()

```
template<typename T>
bool Vector< T >::operator!= (
            const Vector< T > & rhs) const
```

nelygybės operatorius

Parameters

rhs	Vektorius, su kuriuo lyginama

Returns

true Vektoriai nelygūs

false Vektoriai lygūs

5.3.4.25 operator<()

```
template<typename T>
bool Vector< T >::operator< (
            const Vector< T > & rhs) const
```

daugiau operatorius, pirmi nesutampantys elementai tame pačiame indekse nustato grąžinamą reikšmę, jei visi elementai lygūs, didesnis yra tas, kuris turi daugiau elementų

Parameters

rhs	Vektorius, su kuriuo lyginama

Returns

true Vektorius kairėje mažesnis

false Vektorius kairėje nemažesnis

5.3.4.26 operator<=()

```
template<typename T>
bool Vector< T >::operator<= (
            const Vector< T > & rhs) const
```

daugiau operatorius, pirmi nesutampantys elementai tame pačiame indekse nustato grąžinamą reikšmę, jei visi elementai lygūs, didesnis yra tas, kuris turi daugiau elementų

Parameters

rhs	Vektorius, su kuriuo lyginama

Returns

true Vektorius kairėje mažesnis arba lygus

false Vektorius kairėje didesnis

5.3.4.27 operator=() [1/2]

```
template<typename T>
Vector< T > & Vector< T >::operator= (
            const Vector< T > & rhs)
```

Kopijavimo operacija.

Parameters

rhs	Vektorius, iš kurio kopijuojama

Returns

Vector<T>&

5.3.4.28 operator=() [2/2]

```
template<typename T>
Vector< T > & Vector< T >::operator= (
            Vector< T > && rhs)  [noexcept]
```

Perkėlimo operacija.

Parameters

rhs	Vektorius, iš kurio perkeliama

Returns

Vector<T>&

5.3.4.29 operator==()

```
template<typename T>
bool Vector< T >::operator== (
            const Vector< T > & rhs) const
```

lygybės operatorius

Parameters

rhs	Vektorius, su kuriuo lyginama

Returns

true Vektoriai lygūs

false Vektoriai nelygūs

5.3.4.30 operator>()

```
template<typename T>
bool Vector< T >::operator> (
            const Vector< T > & rhs) const
```

daugiau operatorius, pirmi nesutampantys elementai tame pačiame indekse nustato grąžinamą reikšmę, jei visi elementai lygūs, didesnis yra tas, kuris turi daugiau elementų

Parameters

rhs	Vektorius, su kuriuo lyginama

Returns

true Vektorius kairėje didesnis

false Vektorius kairėje nedidesnis

5.3.4.31 operator>=()

```
template<typename T>
bool Vector< T >::operator>= (
            const Vector< T > & rhs) const
```

daugiau operatorius, pirmi nesutampantys elementai tame pačiame indekse nustato grąžinamą reikšmę, jei visi elementai lygūs, didesnis yra tas, kuris turi daugiau elementų

Parameters

rhs	Vektorius, su kuriuo lyginama

Returns

true Vektorius kairėje didesnis arba lygus

false Vektorius kairėje mažesnis

5.3.4.32 operator[]()

```
template<typename T>
value_type & Vector< T >::operator[] (
            int index)
```

indekso operatorius

Parameters

index	Norimas indeksas

Returns

value_type& Elementas tame indekse

5.3.4.33 pop_back()

```
template<typename T>
void Vector< T >::pop_back ()
```

pašalina paskutinį vektoriaus elementą

5.3.4.34 push_back()

```
template<typename T>
void Vector< T >::push_back (
            const T & value)
```

prideda duotą objektą į vektoriaus galą

Parameters

value	Objektas, kurį norima pridėti

5.3.4.35 rbegin() [1/2]

```
template<typename T>
iterator Vector< T >::rbegin ()  [inline]
```

Atvirkštinis pradžios iteratorius nurodo sekančią vietą prieš pirmąjį vektoriaus elementą

Returns

iterator Iteratorius, kuris nurodo vektoriaus atvirkštinę pradžią

Definition at line 293 of file vector.h.

5.3.4.36 rbegin() [2/2]

```
template<typename T>
const_iterator Vector< T >::rbegin () const  [inline]
```

Definition at line 300 of file vector.h.

5.3.4.37 rend() [1/2]

```
template<typename T>
iterator Vector< T >::rend ()  [inline]
```

Atvirkštinis pabaigos iteratorius.

Returns

iterator Iteratorius, kuris nurodo vektoriaus atvirkštinę pabaigą

Definition at line 299 of file vector.h.

5.3.4.38 rend() [2/2]

```
template<typename T>
const_iterator Vector< T >::rend () const  [inline]
```

Definition at line 301 of file vector.h.

5.3.4.39 reserve()

```
template<typename T>
void Vector< T >::reserve (
            int new_cap)
```

Pakeičia vektoriaus talpą į duotąją, jei ji nėra mažesnė už dabartinę

Parameters

new_cap	naujoji norima talpa

5.3.4.40 resize() [1/2]

```
template<typename T>
void Vector< T >::resize (
            int count)
```

Vektoriaus dydžio pakeitimas, tuščios vektoriaus vietos užpildomos numatytu objektu.

Parameters

count	Norimas naujasis dydis

5.3.4.41 resize() [2/2]

```
template<typename T>
void Vector< T >::resize (
            int count,
            const T & value)
```

Vektoriaus dydžio pakeitimas, tuščios vektoriaus vietos užpildomos duotuoju objektu.

Parameters

count	Norimas naujasis dydis
value	Objektas, kuriuo užpildomos tuščios vektoriaus vietos

5.3.4.42 shrink_to_fit()

```
template<typename T>
void Vector< T >::shrink_to_fit ()
```

sumažina vektoriaus talpą iki jo dydžio

5.3.4.43 size()

```
template<typename T>
int Vector< T >::size () const
```

grąžina vektoriaus dydį

Returns

int Dydis

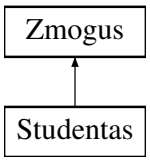
The documentation for this class was generated from the following file:

- vector.h

5.4 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

5.4.1 Detailed Description

Definition at line 8 of file zmogus.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Zmogus() [1/2]

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

Definition at line 15 of file zmogus.h.

5.4.2.2 Zmogus() [2/2]

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

Definition at line 16 of file zmogus.h.

5.4.2.3 ~Zmogus()

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

Definition at line 28 of file zmogus.h.

5.4.3 Member Function Documentation

5.4.3.1 getPavarde()

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

Implemented in Studentas.

5.4.3.2 getVardas()

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

Implemented in Studentas.

5.4.3.3 setVarPav()

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

Definition at line 21 of file zmogus.h.

5.4.4 Member Data Documentation

5.4.4.1 pavarde

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

Definition at line 12 of file zmogus.h.

5.4.4.2 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 rasytiIFaila (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();)
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 duomenu 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 duomenu ivedimo pasirinkimai:\n";
00117         cout « "1 - ranka, 2 - generuoti pazymius, 3 - generuoti ir pazymius ir studentų 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 irasymo 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 <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 rasytiIFaila (string pav, T &v, string pas)
- void rasytiIFaila (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

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 64 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 183 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 140 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 230 of file header.h.

6.5.2.14 skaitytiFaila()

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

Definition at line 89 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 <algorithm>
00018 #include <cstdlib>
00019 #include <ctime>
00020 #include <fstream>
00021 #include <sstream>
00022 #include <limits>
00023 #include <numeric>
00024 #include <chrono>
```

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

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

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

6.7 README.md File Reference

6.8 studentas.h File Reference

Studento klasė

```
#include "zmogus.h"
#include "vector.h"
#include <string>
#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

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 "vector.h"
00016 #include <string>
00017 #include <algorithm>
00018 #include <iostream>
00019 #include <sstream>
00020
00021 using std::copy;
00022 using std::cout;
00023 using std::istream;
00024 using std::istringstream;
00025 using std::move;
00026 using std::ostream;
00027 using std::sort;
00028 using std::string;
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 perkelimo 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 perkelimo 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     }
00148     bool operator==(const Studentas &rhs)
00149     {
00150         return vardas == rhs.getVardas() && pavarde == rhs.getPavarde() && nd == rhs.getNd() && egz ==
      rhs.getEgz();
00151     }
00159     friend ostream &operator«(ostream &os, const Studentas &st)
00160     {
00161         os « st.vardas « " " « st.pavarde « " " « st.galutinisV « " " « st.galutinisM « "\n";
00162         return os;
00163     }
00171     friend istream &operator»(istream &is, Studentas &st)
00172     {
00173         is » st.vardas » st.pavarde;
00174         st.nd.clear();
00175 
00176         string eilute;
00177         getline(is, eilute);
00178         istringstream iss(eilute);
00179 
00180         Vector<int> visiPazymiai;
00181         int paz;
00182 
00183         while (iss » paz)
00184         {
00185             visiPazymiai.push_back(paz);
00186         }
00187 
00188         st.egz = visiPazymiai.back();
00189         visiPazymiai.pop_back();
00190         st.nd = visiPazymiai;
00191         st.galutinisV = st.SkaiciuotiV();
00192         st.galutinisM = st.SkaiciuotiM();
00193 
00194         return is;
00195     }
00202     void setVarPav(string v, string p)
00203     {
00204         vardas = v;
00205         pavarde = p;
00206     }
00213     void setPaz(Vector<int> n, int e)
00214     {
00215         nd = n;
00216         egz = e;
00217         galutinisV = SkaiciuotiV();
00218         galutinisM = SkaiciuotiM();
00219     }
00220     inline string getVardas() const override { return vardas; }
```

```
00221     inline string getPavarde() const override { return pavarde; }
00222     inline int getEgz() const { return egz; }
00223     inline Vector<int> getNd() const { return nd; }
00224     inline int getGalV() const { return galutinisV; }
00225     inline int getGalM() const { return galutinisM; }
00231     float SkaiciuotiV()
00232     {
00233         int s = 0;
00234         for (int i = 0; i < nd.size(); i++)
00235         {
00236             s += nd[i];
00237         }
00238         float galutinis = 0.4 * (1.0 * s / nd.size()) + 0.6 * egz;
00239         return galutinis;
00240     }
00246     float SkaiciuotiM()
00247     {
00248
00249         float paz;
00250         sort(nd.begin(), nd.end());
00251
00252         if (nd.size() % 2 == 0)
00253         {
00254             paz = 1.0 * (nd[nd.size() / 2 - 1] + nd[nd.size() / 2]) / 2;
00255         }
00256         else
00257         {
00258             paz = nd[nd.size() / 2];
00259         }
00260
00261         float galutinis = 0.4 * paz + 0.6 * egz;
00262         return galutinis;
00263     }
00268     void clear()
00269     {
00270         if (TEST_MODE)
00271             cout « "Studento destruktorius\n";
00272         vardas.clear();
00273         pavarde.clear();
00274         nd.clear();
00275         egz = 0;
00276         galutinisV = 0;
00277         galutinisM = 0;
00278     }
00279     ~Studentas() { clear(); }
00280 };
00281
00282 #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

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 vector.h File Reference

Mano sukurto vektoriaus klasė, kuri turi beveik visą įprasto vektoriaus funkcionalumą

```
#include <iostream>
#include <memory>
#include "vector.tpp"
```

Classes

- class Vector< T >

6.12.1 Detailed Description

Mano sukurto vektoriaus klasė, kuri turi beveik visą įprasto vektoriaus funkcionalumą

Author

Eglė

Version

0.1

Date

2025-05-22

Copyright

Definition in file vector.h.

6.13 vector.h

Go to the documentation of this file.

```
00001
00011
00012 #pragma once
00013
00014 #include <iostream>
00015 #include <memory>
00016
00017 template <typename T>
00018 class Vector
00019 {
00020 public:
00021     using value_type = T;
00022     using reference = T &;
00023     using const_reference = const T &;
00024     using pointer = T *;
00025     using const_pointer = const T *;
00026     using iterator = T *;
00027     using const_iterator = const T *;
00028     using reverse_iterator = T *;
00029     using const_reverse_iterator = const T *;
00030     using allocator_type = std::allocator<T>;
00031
00032 private:
00033     int Size;
00034     int Capacity;
00035     T *array;
00036     allocator_type alloc;
00037
00038 public:
00043     Vector();
00049     Vector(const Vector<T> &rhs);
00055     Vector(Vector<T> &&rhs) noexcept;
00062     Vector(int elements, const T &value = T());
00068     Vector(const std::initializer_list<T> &list);
00073     ~Vector();
00074
00080     void push_back(const T &value);
00085     void pop_back();
00086
00093     bool empty() const;
```

```
00099     int size() const;
00105     int capacity() const;
00111     void reserve(int new_cap);
00116     void shrink_to_fit();
00117
00125     bool operator==(const Vector<T> &rhs) const;
00133     bool operator!=(const Vector<T> &rhs) const;
00142     bool operator>(const Vector<T> &rhs) const;
00151     bool operator>=(const Vector<T> &rhs) const;
00160     bool operator<(const Vector<T> &rhs) const;
00169     bool operator<=(const Vector<T> &rhs) const;
00170
00177     Vector<T> &operator=(const Vector<T> &rhs);
00184     Vector<T> &operator=(Vector<T> &&rhs) noexcept;
00185     allocator_type get_allocator() const;
00186
00193     value_type &operator[](int index);
00200     value_type &at(int index);
00206     value_type &front();
00212     value_type &back();
00213
00214     pointer data();
00215     const_pointer data() const;
00216
00223     void insert(int index, const T &value);
00229     void erase(int index);
00230     iterator erase(const_iterator pos);
00231     iterator erase(const_iterator first, const_iterator last);
00236     void clear();
00237
00244     template <typename... Args>
00245     void emplace_back(Args &&...args);
00254     template <typename... Args>
00255     T *emplace(int index, Args &&...args);
00256
00263     void assign(int count, const T &value);
00269     void assign(std::initializer_list<T> ilist);
00270
00276     iterator begin() { return array; }
00282     iterator end() { return array + Size; }
00283     const_iterator begin() const { return array; }
00284     const_iterator end() const { return array + Size; }
00285     const_iterator cbegin() const { return array; }
00286     const_iterator cend() const { return array + Size; }
00287
00293     iterator rbegin() { return array + Size - 1; }
00299     iterator rend() { return array - 1; }
00300     const_iterator rbegin() const { return array + Size - 1; }
00301     const_iterator rend() const { return array - 1; }
00302
00308     void resize(int count);
00315     void resize(int count, const T &value);
00316 };
00317
00318 #include "vector.tpp"
```

6.14 zmogus.h File Reference

```
#include <string>
```

Classes

- class Zmogus

6.15 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