OOP3V3.0

V3.0

Generated by Doxygen 1.14.0

Chapter 1

Releases

Version	Functions	Comments
v.pradinė	Duomenų įvedimas. Studentų duomenų struktūra. Skaičiavimo funkcijos.	Viskas įvyko sklandžiai. Komentarų nėra
v0.1	Galimybė įvesti bet kokį kiekį studentų, nes naudojami STL konteineriai. Versija su dinaminiu masyvu. Atsitiktinis pažymių generavimas. Meniu.	Pirma kartą naudojau STL konteinerius, jie smarkiai palengvina darbą.
v0.2	Sąveika su failais. Rušiavimas pagal pasirinkimą.	Nieko specialaus, bet reikėjo kodo architektūros perdarymo įskaitomumui
v0.3	Naudojama išimčių valdymas.	Iš pradžių nenorėjau naudoti exep.handling, bet pabandžius buvo paprasta ir pagražino kodo išvazdą.
v0.4	Studentų generavimas. Failų generavimas. Studentų rušiavimo funkcija. Testavimo funkcija. Tyrimo README.md aprašas.	Užtruko labai daug laiko, nes buvo nemažai klaidų, teko daug testuoti. Galū gale viskas išėjo sklandžiai.
v1.0	Naudojama CMake. Trys skirtingos versijos su skirtingais STL konteineriais: List, Deque ir Vector. Spartumo tyrimai. Naudojami specialūs algoritmai Vector versijoje. Sutvarkyta repozitorija. Viso projekto aprašas. Naudojimosi ir diegimo instrukcijos.	Užtruko dar daugiau laiko dėl testavimo ir šios versijos monotoniško darbo. Darbas pavyko.
v1.1	Pakeista kodo struktura, implementuota klasė vietoj prieštai naudoto struct	Pradėti su klasėm nesinorėjo, bet vertėjo kodo aiškumui ir spartesniam kurimui
v1.2	Implementuota Rules of Five, perkrautas Ivesties operatorius ir perkrautas Išvesties operatorius. Testai turi menu, padaryti testai programos testavimo palengvinimui.	Versija padaryta greitai, bet naudingai
v1.5	Sukurtos abstrakti bazinė Zmogus ir išvestinė Stud klasės.	Greičiausia padaryta versija, supratimas apie bazines ir išvestines klases tikrai padidėjo. Testų keisti nereikėjo :)
v2.0	Sukurta HTML, LaTeX PDF dokumentacija naudojant Doxygen programėlę. Implementuoti Stud klasės Unit Testai naudojant "googletest" projektą.	Dėl korumpuotų failų ir nepastabumo šita versija buvo blogiausia patirtis. Bet galų gale viskas pavyko sekmingai.
v3.0	Sukuriau savo custom Vector klasę, atkartota didelė dalis STL vector funkcijų. Unit testai nuosavai vector funkcijai. Testavimas. Installeris.	Daug laiko užtruko, sudėtinga pradėti, bet atradus internetinius šaltinius pasidare lengviau.

1.1 Starting manual

Atsisiūsti v3.0 full release.

Naudojant WinRAR arba 7-Zip, atskleisti (extract) failą, bus sukurtas aplankas su programos failais.

"dependancies" aplanke įsidiegti "cmake-4.0.0-rc4-windows-x86_64.msi".

Atidarai aplanką VECTOR, paleidi run.bat scriptą.

Atsidarys "cmd" langas kuriame bus veikianti programa, jeigu programą uždarėte ir norite vėl ją atidaryti, tai nuo tos vietos kur yra run.bat paspauskite "build" aplanką, tada "Release" aplanką, kuriame rasite "OOP3V30.exe" paleidžiamąjį failą.

1.2 User manual

1 - Ranka įvedamas studentas ir jo pažymiai.

2 - Ranka įvedamas studento vardas/pavardė, pažymys generuojamas.

3 - Studentas(-ai) sukuriami automatiškai, tik reikia įvesti studentų kiekį.

4 - Failas, esantis aplanke kartu su "OOP3V30.exe" (...\VECTOR\build\Release) yra nuskaitomas į tam tikrą konteinerį.

5 - Sukuriamas failas pavadinimu "studList{įvestasStudentuKiekis}.txt"

6 - Atidaromas testinis menu

7 - Tęsiama toliau, grįžti negalima.

7.1-7.3 - Pasirinkimai ar isvesti i terminala, i faila arba i faila suskirstytus

7.x.1-7.x.3 - Pasirinkimai apskaičiuoti galutinius pažymius vidurkiu arba mediana arba abu

7.x.x.1-7.x.x.4 - Pasirinkimai pagal ką norite rušiuoti studentus.

Rezultatų failą rasite (...\VECTOR\build\Release).

1.3 5 Funkcijų aprašymai

Reserve	void Reserve(size_t newCapacity); Ši funkcija naudojama užrezervuoti pateiktą "vietų" (newCapacity) atmintyje. Funkcija gauna reikšmę newCapacity, kurią lyginu su esančia capacity reikšme, jeigu newCapacity mažesnis, tai funkcijas grįžta. Kitaip ji sukuria new newArray, kurią užpildo array, array išsitrina ir atgauna savo reikšmes.
PushBack	void PushBack(const T& value); Ši funkcija priima value, kuri įstato į Vectoriaus galą. Pirma - patikrina ar size >= capacity, jei taip, tai padidina array ir toliau eina, kur array paskutine vietele idedama value.
operator==	bool operator==(const Vector& rhs) const; Ši funkcija priima rhs reikšmę, kurią lygina su pradines reikšmės size, tada array elementais ir jeigu abu neismete false, duoda true.
Vector	Vector(const std::initializer_list<T>& list); Ši funkcija leidžia inicijuoti reikšmes pvz.: Vector<int> SK {1,2,3,4,5}. Pirma paima size(0), tada capacity yra pagal paduota list ilgi + 5, sukuriama array dinaminej atminty ir PushBack funkcija prideda visas reikšmes prie array.
At	T& At(size_t index); Priima indeksa, pagal kuri gražina elementa toj vietoj, jeigu indeksas neišeina iš Vectoriaus ribų.

1.4 Vector compare test

Fill size	My vector	STL vector
10000	0.00004790 sec	0.00005380 sec
100000	0.00037480 sec	0.00041790 sec
1000000	0.00198310 sec	0.00290890 sec
10000000	0.01952810 sec	0.03320920 sec
100000000	0.30923830 sec	0.35358800 sec
AVG	0.06051138 sec	0.07803556 sec
TOTAL	0.30636120 sec	0.39470780 sec
Perskirstymai	25 kartai	46 kartai

Kiekvienas failas nuskaitytas po 3 kartus.

File size	My vector (AVG)	My vector (TOT)	STL vector (FUN)	STL vector (AVG)	STL vector (TOT)	STL vector (FUN)
100000	0.42521847 sec	1.27565540 sec	0.37621842 sec	0.41326590 sec	1.23979770 sec	0.35512627 sec
1000000	4.52487077 sec	13.57461230 sec	4.02434072 sec	4.48646507 sec	13.45939520 sec	3.62357066 sec
10000000	49.94290833 sec	149.↩ 82872500 sec	37.95690113 sec	41.98002827 sec	125.↩ 94008480 sec	36.44295553 sec

Nuskaityme vienintele Vector funkcija naudojama yra push_back, arba PushBack, tikslaus laiko negausiu, bet operacija kuri suskirsto duomenis ivyksta yra pateikta. Po šios analizės paaiškėjo, kad aš nesu geresnis Vector klasės kurėjas negu žmonės susiėmę STL.

1.5 Unit testing

Unit testai vykdomi pirmą kartą paleidus run.bat failą ir priešais pagrindinę programą.

1.6 Klasės naudojimas

1.6.1 Išvesties operatorius

Išvesties operatorius perdengtas, naudojimas labai paprastas:

Stud studentas(Jonas, Jonaitis, 8, 5, 6, 7);

std::cout << studentas;

Tai padarant išves duotus duomenis apie studentą, jo vidurkį ir medianą.

1.6.2 Ivesties operatorius

Irgi labai paprastas naudojimas, rankiniu būdu parašant "std::cin >> studentas;" terminale prašys vartotojo įvesti duomenis.

Stud studentas;

std::cin >> studentas;

1.6.2.1 Kiti būdai įvesties

Iš failo įvestis daroma naudojant fileRead() funkciją. Norint nuskaityti savus failus, reikia pateikti juos (...\build\↩ Release\test.txt) formatu (studentu kiekis nesvarbu) ir idėti į (...\VECTOR\build\Release).

Automatinė daroma naudojant makeStud() funkciją.

1.7 V1.1 testas

Part	Specifications
CPU↩ :	Intel i5-10600K
RAM↩ :	2x16GB 2666MHz DDR4
STR↩ :	Kingston NV2 1 TB M.2 2280 NVMe

Flagai	100000 (Struct)	1000000 (Struct)	EXE size (Struct)	100000 (Class)	1000000 (Class)	EXE size (Class)
Be flagu	0.44610952 sec	4.20656290 sec	127 KB	0.44792966 sec	4.39760216 sec	141 KB
O1	0.44192624 sec	4.05293074 sec	119 KB	0.43307130 sec	4.24400640 sec	129 KB
O2	0.40628492 sec	4.08120978 sec	119 KB	0.43038714 sec	4.25291338 sec	129 KB
O3	0.43692144 sec	4.35214120 sec	119 KB	0.44872598 sec	4.40468464 sec	129 KB

1.8 V1.0 testas

1.8.1 Pradžia

Part	Specifications
CPU↩ :	Intel i5-10600K
RAM↩ :	2x16GB 2666MHz DDR4
STR↩ :	Kingston NV2 1 TB M.2 2280 NVMe

Įvestis:

1 Strategija:

6 studList1000.txt 1 1 3

6 studList10000.txt 1 1 3

6 studList100000.txt 1 1 3

6 studList1000000.txt 1 1 3

6 studList10000000.txt 1 1 3

2 Strategija:

6 studList1000.txt 1 2 3

6 studList10000.txt 1 2 3

6 studList100000.txt 1 2 3

6 studList1000000.txt 1 2 3

6 studList10000000.txt 1 2 3

1.8.2 Vector

Failo dydis	Pirma strategija	Antra strategija
1000	0.04915840 sec	0.08148740 sec
10000	0.43130660 sec	0.45130710 sec
100000	4.46657050 sec	4.48731360 sec
1000000	41.31402830 sec	42.56847090 sec
10000000	424.58098150 sec	431.82273370 sec

10 milijonų su pirma strategija užemė 7.7GB RAM, o antra 5.9GB RAM.

1.8.3 List

Failo dydis	Pirma strategija	Antra strategija
1000	0.05118660 sec	0.08393100 sec
10000	0.37497380 sec	0.37388180 sec
100000	3.70634520 sec	3.53147080 sec
1000000	35.40114870 sec	35.71196400 sec
10000000	350.78200190 sec	361.86110030 sec

10 milijonų su pirma strategija užemė 6.1GB RAM, o antra 5.3GB RAM.

1.8.4 Deque

Failo dydis	Pirma strategija	Antra strategija
1000	0.05772800 sec	0.05651100 sec
10000	0.39833120 sec	0.41024440 sec
100000	3.76838590 sec	4.06764040 sec
1000000	37.04322620 sec	39.36357640 sec
10000000	373.31425860 sec	394.22153330 sec

10 milijonų su pirma strategija užemė 10.1GB RAM, o antra 6.8GB RAM.

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:

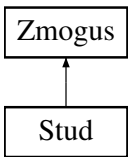
Chapter 5

Class Documentation

5.1 Stud Class Reference

`#include <studentas.h>`

Inheritance diagram for Stud:



Public Member Functions

- Stud (std::string var="", std::string pav="", Vector< int > pazymys={}, int egz={})

  Konstruktorius ir desktrutorius.
- ~Stud ()=default
- Stud (const Stud &other)

  Copy constructor.
- Stud & operator= (const Stud &other)

  Copy assignment operator.
- Stud (Stud &&other) noexcept

  Move constructor.
- Stud & operator= (Stud &&other) noexcept

  Move assignment operator.
- void setVar (const std::string &var) override

  Setteriai, kurie nustato studento varda, pavarde, uzduotis ir egzamino pazymi.
- void setPav (const std::string &pav) override
- void setEgz (const int egz)
- void addPazymys (const int pazymys)

  Papildomos funkcijos, kurios prideda ir pasalina uzduociu pazymius.
- void removePazymys ()
- void calculateGalVidurkis ()
- void calculateGalMediana ()
- std::string getVar () const override

Getteriai, kurie grazina studento varda, pavarde, uzduotis, egzamino pazymi ir galutini pazymi.

- std::string getPav () const override
- Vector< int > getPazymys () const
- int getEgz () const
- float getVidurkis () const
- float getMediana () const

Public Member Functions inherited from Zmogus

- Zmogus (std::string var="", std::string pav="")

  Konstruktorius ir desktrutorius.
- ~Zmogus ()=default
- Zmogus (const Zmogus &other)

  Copy constructor.
- Zmogus & operator= (const Zmogus &other)

  Copy assignment operator.
- Zmogus (Zmogus &&other) noexcept

  Move constructor.
- Zmogus & operator= (Zmogus &&other) noexcept

  Move assignment operator.

Static Public Member Functions

- static int ivestiesPatikrinimas (const int nuo, const int iki)

  Ivesties patikrinimas.
- static int ivestiesPatikrinimas (const int nuo, const int iki, const int sustabdymoSalyga)

Friends

- std::istream & operator>> (std::istream &is, Stud &s)

  Ivesties operatorius.
- std::ostream & operator<< (std::ostream &os, const Stud &s)

  Isvesties operatorius.

Additional Inherited Members

Protected Attributes inherited from Zmogus

- std::string var_ {}
- std::string pav_ {}

5.1.1 Detailed Description

Definition at line 8 of file studentas.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Stud() [1/3]

```
Stud::Stud (
            std::string var = "",
            std::string pav = "",
            Vector< int > pazymys = {},
            int egz = {})  [inline], [explicit]
```

Konstruktorius ir desktrutorius.

Definition at line 17 of file studentas.h.

5.1.2.2 ~Stud()

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

5.1.2.3 Stud() [2/3]

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

Copy constructor.

Definition at line 23 of file studentas.h.

5.1.2.4 Stud() [3/3]

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

Move constructor.

Definition at line 43 of file studentas.h.

5.1.3 Member Function Documentation

5.1.3.1 addPazymys()

```
void Stud::addPazymys (
            const int pazymys)  [inline]
```

Papildomos funkcijos, kurios prideda ir pasalina uzduociu pazymius.

Definition at line 115 of file studentas.h.

5.1.3.2 calculateGalMediana()

```
void Stud::calculateGalMediana ()
```

Definition at line 17 of file Stud.cpp.

5.1.3.3 calculateGalVidurkis()

```
void Stud::calculateGalVidurkis ()
```

Definition at line 5 of file Stud.cpp.

5.1.3.4 getEgz()

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

Definition at line 125 of file studentas.h.

5.1.3.5 getMediana()

```
float Stud::getMediana () const  [inline]
```

Definition at line 127 of file studentas.h.

5.1.3.6 getPav()

```
std::string Stud::getPav () const  [inline], [override], [virtual]
```

Reimplemented from Zmogus.

Definition at line 123 of file studentas.h.

5.1.3.7 getPazymys()

```
Vector< int > Stud::getPazymys () const  [inline]
```

Definition at line 124 of file studentas.h.

5.1.3.8 getVar()

```
std::string Stud::getVar () const  [inline], [override], [virtual]
```

Getteriai, kurie grazina studento varda, pavarde, uzduotis, egzamino pazymi ir galutini pazymi.

Reimplemented from Zmogus.

Definition at line 122 of file studentas.h.

5.1.3.9 getVidurkis()

```
float Stud::getVidurkis () const  [inline]
```

Definition at line 126 of file studentas.h.

5.1.3.10 ivestiesPatikrinimas() [1/2]

```
int Stud::ivestiesPatikrinimas (
            const int nuo,
            const int iki)  [inline], [static]
```

Ivesties patikrinimas.

Definition at line 130 of file studentas.h.

5.1.3.11 ivestiesPatikrinimas() [2/2]

```
int Stud::ivestiesPatikrinimas (
            const int nuo,
            const int iki,
            const int sustabdymoSalyga)  [inline], [static]
```

Definition at line 151 of file studentas.h.

5.1.3.12 operator=() [1/2]

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

Copy assignment operator.

Definition at line 31 of file studentas.h.

5.1.3.13 operator=() [2/2]

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

Move assignment operator.

Definition at line 56 of file studentas.h.

5.1.3.14 removePazymys()

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

Definition at line 116 of file studentas.h.

5.1.3.15 setEgz()

```
void Stud::setEgz (
            const int egz)  [inline]
```

Definition at line 112 of file studentas.h.

5.1.3.16 setPav()

```
void Stud::setPav (
            const std::string & pav)  [inline], [override], [virtual]
```

Implements Zmogus.

Definition at line 111 of file studentas.h.

5.1.3.17 setVar()

```
void Stud::setVar (
            const std::string & var)  [inline], [override], [virtual]
```

Setteriai, kurie nustato studento varda, pavarde, uzduotis ir egzamino pazymi.

Implements Zmogus.

Definition at line 110 of file studentas.h.

5.1.4 Friends And Related Symbol Documentation

5.1.4.1 operator<<

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

Isvesties operatorius.

Definition at line 99 of file studentas.h.

5.1.4.2 operator>>

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

Ivesties operatorius.

Definition at line 73 of file studentas.h.

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

- VECTOR/Include/studentas.h
- VECTOR/src/Stud.cpp

5.2 Vector< T > Class Template Reference

`#include <vector.h>`

Public Member Functions

- Vector ()
- Vector (const Vector &rhs)
- Vector (int elements, const T &value=T())
- Vector (const std::initializer_list< T > &list)
- ~Vector ()
- Vector & operator= (const Vector &rhs)
- void Assign (size_t count, const T &value)
- template<typename InputIt, typename = typename std::enable_if<!std::is_integral<InputIt>::value>::type>
  void Assign (InputIt first, InputIt last)
- T & At (size_t index)
- const T & At (size_t index) const
- T & operator[ ] (size_t index)
- T & Front ()
- const T & Front () const
- T & Back ()
- const T & Back () const
- T * begin ()
- T * end ()
- T * rbegin ()
- T * rend ()
- bool Empty () const
- size_t Size () const
- size_t MaxSize () const
- void Reserve (size_t newCapacity)
- size_t Capacity () const
- void ShrinkToFit ()
- void Clear ()
- void Insert (size_t index, const T &value)
- template<typename InputIt>
  void InsertRange (size_t index, InputIt first, InputIt last)
- template<typename... Args>
  void Emplace (size_t index, Args &&... args)
- void Erase (size_t index)
- void PushBack (const T &value)
- template<typename... Args>
  void EmplaceBack (Args &&... args)
- template<typename InputIt>
  void AppendRange (InputIt first, InputIt last)
- void PopBack ()
- void Resize (size_t newSize, const T &value=T())
- void Swap (Vector &other)
- bool operator== (const Vector &rhs) const
- bool operator!= (const Vector &rhs) const

Friends

- std::ostream & operator<< (std::ostream &ostr, const Vector &rhs)

5.2.1 Detailed Description

template<typename T>
class Vector< T >

Definition at line 8 of file vector.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Vector() [1/4]

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

Definition at line 15 of file vector.h.

5.2.2.2 Vector() [2/4]

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

Definition at line 20 of file vector.h.

5.2.2.3 Vector() [3/4]

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

Definition at line 30 of file vector.h.

5.2.2.4 Vector() [4/4]

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

Definition at line 40 of file vector.h.

5.2.2.5 ~Vector()

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

Definition at line 52 of file vector.h.

5.2.3 Member Function Documentation

5.2.3.1 AppendRange()

```
template<typename T>
template<typename InputIt>
void Vector< T >::AppendRange (
            InputIt first,
            InputIt last)  [inline]
```

Definition at line 310 of file vector.h.

5.2.3.2 Assign() [1/2]

```
template<typename T>
template<typename InputIt, typename = typename std::enable_if<!std::is_integral<InputIt>↩
::value>::type>
void Vector< T >::Assign (
            InputIt first,
            InputIt last)  [inline]
```

Definition at line 91 of file vector.h.

5.2.3.3 Assign() [2/2]

```
template<typename T>
void Vector< T >::Assign (
            size_t count,
            const T & value)  [inline]
```

Definition at line 78 of file vector.h.

5.2.3.4 At() [1/2]

```
template<typename T>
T & Vector< T >::At (
            size_t index)  [inline]
```

Definition at line 109 of file vector.h.

5.2.3.5 At() [2/2]

```
template<typename T>
const T & Vector< T >::At (
            size_t index) const  [inline]
```

Definition at line 117 of file vector.h.

5.2.3.6 Back() [1/2]

```
template<typename T>
T & Vector< T >::Back ()  [inline]
```

Definition at line 136 of file vector.h.

5.2.3.7 Back() [2/2]

```
template<typename T>
const T & Vector< T >::Back () const  [inline]
```

Definition at line 140 of file vector.h.

5.2.3.8 begin()

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

Definition at line 146 of file vector.h.

5.2.3.9 Capacity()

```
template<typename T>
size_t Vector< T >::Capacity () const  [inline]
```

Definition at line 189 of file vector.h.

5.2.3.10 Clear()

```
template<typename T>
void Vector< T >::Clear ()  [inline]
```

Definition at line 209 of file vector.h.

5.2.3.11 Emplace()

```
template<typename T>
template<typename...  Args>
void Vector< T >::Emplace (
            size_t index,
            Args &&...  args)  [inline]
```

Definition at line 260 of file vector.h.

5.2.3.12 EmplaceBack()

```
template<typename T>
template<typename...  Args>
void Vector< T >::EmplaceBack (
            Args &&...  args)  [inline]
```

Definition at line 301 of file vector.h.

5.2.3.13 Empty()

```
template<typename T>
bool Vector< T >::Empty () const  [inline]
```

Definition at line 163 of file vector.h.

5.2.3.14 end()

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

Definition at line 150 of file vector.h.

5.2.3.15 Erase()

```
template<typename T>
void Vector< T >::Erase (
            size_t index)  [inline]
```

Definition at line 276 of file vector.h.

5.2.3.16 Front() [1/2]

```
template<typename T>
T & Vector< T >::Front ()  [inline]
```

Definition at line 128 of file vector.h.

5.2.3.17 Front() [2/2]

```
template<typename T>
const T & Vector< T >::Front () const  [inline]
```

Definition at line 132 of file vector.h.

5.2.3.18 Insert()

```
template<typename T>
void Vector< T >::Insert (
            size_t index,
            const T & value)  [inline]
```

Definition at line 213 of file vector.h.

5.2.3.19 InsertRange()

```
template<typename T>
template<typename InputIt>
void Vector< T >::InsertRange (
            size_t index,
            InputIt first,
            InputIt last)  [inline]
```

Definition at line 241 of file vector.h.

5.2.3.20 MaxSize()

```
template<typename T>
size_t Vector< T >::MaxSize () const  [inline]
```

Definition at line 171 of file vector.h.

5.2.3.21 operator"!=()

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

Definition at line 363 of file vector.h.

5.2.3.22 operator=()

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

Definition at line 61 of file vector.h.

5.2.3.23 operator==()

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

Definition at line 354 of file vector.h.

5.2.3.24 operator[]()

```
template<typename T>
T & Vector< T >::operator[] (
            size_t index)  [inline]
```

Definition at line 124 of file vector.h.

5.2.3.25 PopBack()

```
template<typename T>
void Vector< T >::PopBack ()  [inline]
```

Definition at line 316 of file vector.h.

5.2.3.26 PushBack()

```
template<typename T>
void Vector< T >::PushBack (
            const T & value)  [inline]
```

Definition at line 287 of file vector.h.

5.2.3.27 rbegin()

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

Definition at line 154 of file vector.h.

5.2.3.28 rend()

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

Definition at line 158 of file vector.h.

5.2.3.29 Reserve()

```
template<typename T>
void Vector< T >::Reserve (
            size_t newCapacity)  [inline]
```

Definition at line 175 of file vector.h.

5.2.3.30 Resize()

```
template<typename T>
void Vector< T >::Resize (
            size_t newSize,
            const T & value = T())  [inline]
```

Definition at line 323 of file vector.h.

5.2.3.31 ShrinkToFit()

```
template<typename T>
void Vector< T >::ShrinkToFit ()  [inline]
```

Definition at line 193 of file vector.h.

5.2.3.32 Size()

```
template<typename T>
size_t Vector< T >::Size () const  [inline]
```

Definition at line 167 of file vector.h.

5.2.3.33 Swap()

```
template<typename T>
void Vector< T >::Swap (
            Vector< T > & other)  [inline]
```

Definition at line 338 of file vector.h.

5.2.4 Friends And Related Symbol Documentation

5.2.4.1 operator<<

```
template<typename T>
std::ostream & operator<< (
            std::ostream & ostr,
            const Vector< T > & rhs)  [friend]
```

Definition at line 370 of file vector.h.

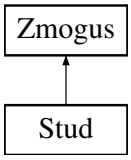
The documentation for this class was generated from the following file:

- VECTOR/Include/vector.h

5.3 Zmogus Class Reference

`#include <zmogus.h>`

Inheritance diagram for Zmogus:



Public Member Functions

- Zmogus (std::string var="", std::string pav="")
  Konstruktorius ir desktrutorius.
- ~Zmogus ()=default
- Zmogus (const Zmogus &other)
  Copy constructor.
- Zmogus & operator= (const Zmogus &other)
  Copy assignment operator.
- Zmogus (Zmogus &&other) noexcept
  Move constructor.
- Zmogus & operator= (Zmogus &&other) noexcept
  Move assignment operator.
- virtual void setVar (const std::string &var)=0
  Setteriai, kurie nustato zmogaus varda ir pavarde.
- virtual void setPav (const std::string &pav)=0
- virtual std::string getVar () const
  Getteriai, kurie grazina zmogaus varda ir pavarde.
- virtual std::string getPav () const

Protected Attributes

- std::string var_ {}
- std::string pav_ {}

Friends

- std::istream & operator>> (std::istream &is, Zmogus &s)
  Ivesties operatorius.
- std::ostream & operator<< (std::ostream &os, const Zmogus &s)
  Isvesties operatorius.

5.3.1 Detailed Description

Definition at line 4 of file zmogus.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Zmogus() [1/3]

```
Zmogus::Zmogus (
            std::string var = "",
            std::string pav = "")  [inline], [explicit]
```

Konstruktorius ir desktrutorius.

Definition at line 10 of file zmogus.h.

5.3.2.2 ~Zmogus()

```
Zmogus::~Zmogus ()  [default]
```

5.3.2.3 Zmogus() [2/3]

```
Zmogus::Zmogus (
            const Zmogus & other)  [inline]
```

Copy constructor.

Definition at line 16 of file zmogus.h.

5.3.2.4 Zmogus() [3/3]

```
Zmogus::Zmogus (
            Zmogus && other)  [inline], [noexcept]
```

Move constructor.

Definition at line 30 of file zmogus.h.

5.3.3 Member Function Documentation

5.3.3.1 getPav()

```
virtual std::string Zmogus::getPav () const  [inline], [virtual]
```

Reimplemented in Stud.

Definition at line 74 of file zmogus.h.

5.3.3.2 getVar()

```
virtual std::string Zmogus::getVar () const  [inline], [virtual]
```

Getteriai, kurie grazina zmogaus varda ir pavarde.

Reimplemented in Stud.

Definition at line 73 of file zmogus.h.

5.3.3.3 operator=() [1/2]

```
Zmogus & Zmogus::operator= (
            const Zmogus & other)  [inline]
```

Copy assignment operator.

Definition at line 21 of file zmogus.h.

5.3.3.4 operator=() [2/2]

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

Move assignment operator.

Definition at line 38 of file zmogus.h.

5.3.3.5 setPav()

```
virtual void Zmogus::setPav (
            const std::string & pav)  [pure virtual]
```

Implemented in Stud.

5.3.3.6 setVar()

```
virtual void Zmogus::setVar (
            const std::string & var)  [pure virtual]
```

Setteriai, kurie nustato zmogaus varda ir pavarde.

Implemented in Stud.

5.3.4 Friends And Related Symbol Documentation

5.3.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Zmogus & s)  [friend]
```

Isvesties operatorius.

Definition at line 63 of file zmogus.h.

5.3.4.2 operator>>

```
std::istream & operator>> (
            std::istream & is,
            Zmogus & s)  [friend]
```

Ivesties operatorius.

Definition at line 49 of file zmogus.h.

5.3.5 Member Data Documentation

5.3.5.1 pav_

```
std::string Zmogus::pav_ {}  [protected]
```

Definition at line 6 of file zmogus.h.

5.3.5.2 var_

```
std::string Zmogus::var_ {}  [protected]
```

Definition at line 6 of file zmogus.h.

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

- VECTOR/Include/zmogus.h

Chapter 6

File Documentation

6.1 README.md File Reference

6.2 VECTOR/Include/functionsCallsVector.h File Reference

```
#include "studentas.h"
#include "zmogus.h"
#include "meinelib.h"
```

Functions

- void readRanka (Stud &stu)
- void readName_makeGrade (Stud &stu)
- void makeStud (Stud &stu)
- void fileRead (Vector< Stud > &studentai, std::string vardas)
- void isvestiesMenu (Vector< Stud > &studentai)
- void isvestis (Vector< Stud > &studentai, std::ostream &isvestiesMetodas, const int galutinioPasirinkimas)
- void rusiavimas (Vector< Stud > &studentai, int rusiavimoPasirinkimas)
- void randomStudentas (Stud &studentas, bool vyras)
- void randomAtsitiktinisPazymys (Stud &stu)
- void failoKurimas (int studentuSk)
- void fileFilter (Vector< Stud > &studentai, const int galutinioPasirinkimas, const int rusiavimoPasirinkimas)
- void testMenu ()
- void nuskaitymoTestas ()
- void studentuTest ()
- void programTest ()
- void vectorCompare ()
- void atmintiesPerskirstymas ()
- int ivestiesPatikrinimas (const int nuo, const int iki)
- int ivestiesPatikrinimas (const int nuo, const int iki, const int sustabdymoSalyga)
- void studentuGalutiniuSkaiciavimas (Vector< Stud > &studentai)

6.2.1 Function Documentation

6.2.1.1 atmintiesPerskirstymas()

```
void atmintiesPerskirstymas ()
```

Definition at line 283 of file test.cpp.

6.2.1.2 failoKurimas()

```
void failoKurimas (
            int studentuSk)
```

Definition at line 5 of file fileGenerator.cpp.

6.2.1.3 fileFilter()

```
void fileFilter (
            Vector< Stud > & studentai,
            const int galutinioPasirinkimas,
            const int rusiavimoPasirinkimas)
```

Definition at line 5 of file studentuSkirstymas.cpp.

6.2.1.4 fileRead()

```
void fileRead (
            Vector< Stud > & studentai,
            std::string vardas)
```

Definition at line 48 of file ivestis.cpp.

6.2.1.5 isvestiesMenu()

```
void isvestiesMenu (
            Vector< Stud > & studentai)
```

Definition at line 5 of file isvestis.cpp.

6.2.1.6 isvestis()

```
void isvestis (
            Vector< Stud > & studentai,
            std::ostream & isvestiesMetodas,
            const int galutinioPasirinkimas)
```

Definition at line 43 of file isvestis.cpp.

6.2.1.7 ivestiesPatikrinimas() [1/2]

```
int ivestiesPatikrinimas (
            const int nuo,
            const int iki)
```

Definition at line 5 of file ivestisPatikrinimas.cpp.

6.2.1.8 ivestiesPatikrinimas() [2/2]

```
int ivestiesPatikrinimas (
            const int nuo,
            const int iki,
            const int sustabdymoSalyga)
```

Definition at line 26 of file ivestisPatikrinimas.cpp.

6.2.1.9 makeStud()

```
void makeStud (
            Stud & stu)
```

Definition at line 43 of file ivestis.cpp.

6.2.1.10 nuskaitymoTestas()

```
void nuskaitymoTestas ()
```

Definition at line 39 of file test.cpp.

6.2.1.11 programTest()

```
void programTest ()
```

Definition at line 101 of file test.cpp.

6.2.1.12 randomAtsitiktinisPazymys()

```
void randomAtsitiktinisPazymys (
            Stud & stu)
```

Definition at line 40 of file generators.cpp.

6.2.1.13 randomStudentas()

```
void randomStudentas (
            Stud & studentas,
            bool vyras)
```

Definition at line 29 of file generators.cpp.

6.2.1.14 readName_makeGrade()

```
void readName_makeGrade (
            Stud & stu)
```

Definition at line 32 of file ivestis.cpp.

6.2.1.15 readRanka()

```
void readRanka (
            Stud & stu)
```

Definition at line 5 of file ivestis.cpp.

6.2.1.16 rusiavimas()

```
void rusiavimas (
            Vector< Stud > & studentai,
            int rusiavimoPasirinkimas)
```

Definition at line 5 of file studentuRusiavimas.cpp.

6.2.1.17 studentuGalutiniuSkaiciavimas()

```
void studentuGalutiniuSkaiciavimas (
            Vector< Stud > & studentai)
```

Definition at line 35 of file Stud.cpp.

6.2.1.18 studentuTest()

```
void studentuTest ()
```

Konstruktoriaus testas

Tuscio konstruktoriaus testas

Definition at line 165 of file test.cpp.

6.2.1.19 testMenu()

```
void testMenu ()
```

Definition at line 5 of file test.cpp.

6.2.1.20 vectorCompare()

```
void vectorCompare ()
```

Definition at line 259 of file test.cpp.

6.3 functionsCallsVector.h

Go to the documentation of this file.

```
00001 #ifndef FUNCTIONSCALLSV_H
00002 #define FUNCTIONSCALLSV_H
00003
00004 #include "studentas.h"
00005 #include "zmogus.h"
00006 #include "meinelib.h"
00007
00008 // Laikomi funkciju prototipai.
00009
00010 // Prototipai
00011
00012 // ivestis.cpp
00013 void readRanka(Stud& stu);
00014 void readName_makeGrade(Stud& stu);
00015 void makeStud(Stud& stu);
00016 void fileRead(Vector<Stud>& studentai, std::string vardas);
00017
00018 // isvestis.cpp
00019 void isvestiesMenu(Vector<Stud>& studentai);
00020 void isvestis(Vector<Stud>& studentai, std::ostream& isvestiesMetodas, const int
      galutinioPasirinkimas);
00021
00022 // studentuRusiavimas.cpp
00023 void rusiavimas(Vector<Stud>& studentai, int rusiavimoPasirinkimas);
00024
00025 // generators.cpp
00026 void randomStudentas(Stud& studentas, bool vyras);
00027 void randomAtsitiktinisPazymys(Stud& stu);
00028
00029 // fileGenerator.cpp
00030 void failoKurimas(int studentuSk);
00031
00032 // studentuSkirstymas.cpp
00033 void fileFilter(Vector<Stud>& studentai, const int galutinioPasirinkimas, const int
      rusiavimoPasirinkimas);
00034
00035 // test.cpp
00036 void testMenu();
00037 void nuskaitymoTestas();
00038 void studentuTest();
00039 void programTest();
00040 void vectorCompare();
00041 void atmintiesPerskirstymas();
00042
00043 // ivestisPatikrinimas.cpp
00044 int ivestiesPatikrinimas(const int nuo, const int iki);
00045 int ivestiesPatikrinimas(const int nuo, const int iki, const int sustabdymoSalyga);
00046
00047 // Stud.cpp
00048 // Stud klases Medianos ir Vidurkio funkciju deklaracijos pacioje klaseje
00049 void studentuGalutiniuSkaiciavimas(Vector<Stud>& studentai);
00050
00051
00052 #endif
```

6.4 VECTOR/Include/meinelib.h File Reference

```
#include <iostream>
#include <iomanip>
#include <string>
#include "vector.h"
```

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

Typedefs

- using hrClock = std::chrono::high_resolution_clock
- using ms = std::chrono::milliseconds
- using sec = std::chrono::duration<double>

6.4.1 Typedef Documentation

6.4.1.1 hrClock

```
using hrClock = std::chrono::high_resolution_clock
```

Definition at line 32 of file meinelib.h.

6.4.1.2 ms

```
using ms = std::chrono::milliseconds
```

Definition at line 33 of file meinelib.h.

6.4.1.3 sec

```
using sec = std::chrono::duration<double>
```

Definition at line 34 of file meinelib.h.

6.5 meinelib.h

Go to the documentation of this file.

```
00001 #ifndef MEINELIB_H
00002 #define MEINELIB_H
00003
00004 // Biblioteku ir pavadinimu header failas.
00005
00006 #include <iostream>
00007 #include <iomanip>
00008 #include <string>
00009 #include "vector.h"
00010 #include <deque>
00011 #include <list>
00012 #include <fstream>
00013 #include <algorithm>
00014 #include <numeric>
00015 #include <sstream>
00016 #include <cstdlib>
00017 #include <cmath>
00018 #include <chrono>
00019 #include <filesystem>
00020
00021 namespace fs = std::filesystem;
00022
00023 using std::cout;
00024 using std::cout;
00025 using std::endl;
00026 using std::string;
00027 using std::deque;
00028 using std::list;
00029 using std::fixed;
00030 using std::setprecision;
00031 using std::sort;
00032 using hrClock = std::chrono::high_resolution_clock;
00033 using ms = std::chrono::milliseconds;
00034 using sec = std::chrono::duration<double>;
00035
00036
00037 #endif
```

6.6 VECTOR/Include/studentas.h File Reference

```
#include "zmogus.h"
#include "Vector.h"
```

Classes

- class Stud

6.7 studentas.h

Go to the documentation of this file.

```
00001 #ifndef STUDENTAS_H
00002 #define STUDENTAS_H
00003
00004 #include "zmogus.h"
00005 #include "Vector.h"
00006 // Struktura, kurioje laikomi studento duomenys.
00007
00008 class Stud : public Zmogus {
00009 private:
00010     Vector<int>pazymys_{};
00011     int egz_{};
00012     float galVidurkis_{};
00013     float galMediana_{};
```

```
00014
00015 public:
00017     explicit Stud(std::string var = "", std::string pav = "", Vector<int> pazymys = {}, int egz = {})
      :
00018         Zmogus(std::move(var), std::move(pav)), pazymys_(std::move(pazymys)), egz_(egz) {}
00019
00020     ~Stud() = default;
00021
00023     Stud(const Stud& other) :
00024         Zmogus(other.var_, other.pav_),
00025         pazymys_(other.pazymys_),
00026         egz_(other.egz_),
00027         galVidurkis_(other.galVidurkis_),
00028         galMediana_(other.galMediana_) {}
00029
00031     Stud& operator=(const Stud& other) {
00032         if (this != &other) {
00033             Zmogus::operator=(other);
00034             pazymys_ = other.pazymys_;
00035             egz_ = other.egz_;
00036             galVidurkis_ = other.galVidurkis_;
00037             galMediana_ = other.galMediana_;
00038         }
00039         return *this;
00040     }
00041
00043     Stud(Stud&& other) noexcept :
00044         Zmogus(std::move(other)),
00045         pazymys_(std::move(other.pazymys_)),
00046         egz_(other.egz_),
00047         galVidurkis_(other.galVidurkis_),
00048         galMediana_(other.galMediana_) {
00049         other.pazymys_.Clear();
00050         other.egz_ = 0;
00051         other.galVidurkis_ = 0.0f;
00052         other.galMediana_ = 0.0f;
00053     }
00054
00056     Stud& operator=(Stud&& other) noexcept {
00057         if (this != &other) {
00058             Zmogus::operator=(std::move(other));
00059             pazymys_ = std::move(other.pazymys_);
00060             egz_ = other.egz_;
00061             galVidurkis_ = other.galVidurkis_;
00062             galMediana_ = other.galMediana_;
00063
00064             other.pazymys_.Clear();
00065             other.egz_ = 0;
00066             other.galVidurkis_ = 0.0f;
00067             other.galMediana_ = 0.0f;
00068         }
00069         return *this;
00070     }
00071
00073     friend std::istream& operator»(std::istream& is, Stud& s) {
00074         is » static_cast<Zmogus&>(s);
00075
00076         int egz;
00077         Vector<int> pazymiai;
00078         int paz;
00079
00080         std::cout « "Iveskite egzamino pazymi: ";
00081         egz = ivestiesPatikrinimas(0, 10);
00082
00083         std::cout « "Iveskite pazymius 0 iki 10, norint baigti iveskite -1:\n";
00084         while(true) {
00085             paz = ivestiesPatikrinimas(0, 10, -1);
00086             if (paz == -1) break;
00087             pazymiai.PushBack(paz);
00088         }
00089
00090         s.setEgz(egz);
00091         for (int p : pazymiai) s.addPazymys(p);
00092
00093         s.calculateGalVidurkis();
00094         s.calculateGalMediana();
00095         return is;
00096     }
00097
00099     friend std::ostream& operator«(std::ostream& os, const Stud& s) {
00100         os « static_cast<const Zmogus&>(s);
00101         os « " Egzaminas: " « s.getEgz() « " Pazymiai: ";
00102         for (int p : s.getPazymys()) {
00103             os « p « " ";
00104         }
00105         os « "Vidurkis: " « s.getVidurkis() « " Mediana: " « s.getMediana();
00106         return os;
```

```
00107     }
00108
00110     void setVar(const std::string& var) override { var_ = var; }
00111     void setPav(const std::string& pav) override { pav_ = pav; }
00112     void setEgz(const int egz)                   { egz_ = egz; }
00113
00115     void addPazymys(const int pazymys)        {pazymys_.PushBack(pazymys);}
00116     void removePazymys()                    {pazymys_.PopBack();}
00117
00118     void calculateGalVidurkis();
00119     void calculateGalMediana();
00120
00122     std::string getVar() const override { return var_; }
00123     std::string getPav() const override { return pav_; }
00124     Vector<int> getPazymys() const { return pazymys_; }
00125     int getEgz() const                    { return egz_; }
00126     float getVidurkis() const           { return galVidurkis_; }
00127     float getMediana() const          { return galMediana_; }
00128
00130     static int ivestiesPatikrinimas(const int nuo, const int iki) {
00131         int input{};
00132         while (true) {
00133             try {
00134                 std::cin » input;
00135                 if (input < nuo || input > iki) {
00136                     std::cout « "\n\n!!!!Iveskite skaiciu nuo " « nuo « " iki " « iki « ".!!!!\n\n\n";
00137                     continue;
00138                 }
00139             }
00140             catch (...) {
00141                 std::cin.clear();
00142                 std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00143                 std::cout « "\n\n!!!!Ivestis neteisinga. Bandykite isnaujo.!!!!\n\n\n";
00144                 continue;
00145             }
00146             break;
00147         }
00148         return input;
00149     }
00150
00151     static int ivestiesPatikrinimas(const int nuo, const int iki, const int sustabdymoSalyga) {
00152         int input{};
00153         while (true) {
00154             try {
00155                 std::cin » input;
00156                 if (input == sustabdymoSalyga) {
00157                     return sustabdymoSalyga;
00158                 }
00159
00160                 if (input < nuo || input > iki) {
00161                     std::cout « "\n\n!!!!Iveskite skaiciu nuo " « nuo « " iki " « iki « ".!!!!\n\n\n";
00162                     continue;
00163                 }
00164             }
00165             catch (...) {
00166                 std::cin.clear();
00167                 std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00168                 std::cout « "\n\n!!!!Ivestis neteisinga. Bandykite isnaujo.!!!!\n\n\n";
00169                 continue;
00170             }
00171             break;
00172         }
00173         return input;
00174     }
00175 };
00176
00177 #endif
```

6.8 VECTOR/Include/vector.h File Reference

```
#include <iostream>
#include <iterator>
```

Classes

- class Vector< T >

6.9 vector.h

Go to the documentation of this file.

```
00001 #ifndef VECTOR_H
00002 #define VECTOR_H
00003
00004 #include <iostream>
00005 #include <iterator>
00006
00007 template<typename T>
00008 class Vector {
00009 private:
00010     size_t size;
00011     size_t capacity;
00012     T* array = nullptr;
00013 public:
00014     // Constructors
00015     Vector() :
00016         size(0),
00017         capacity(5),
00018         array(new T[capacity]) {}
00019
00020     Vector(const Vector& rhs) :
00021         size(rhs.size),
00022         capacity(rhs.capacity),
00023         array(new T[capacity])
00024     {
00025         for (size_t i = 0; i < size; ++i) {
00026             array[i] = rhs.array[i];
00027         }
00028     }
00029
00030     Vector(int elements, const T& value = T()) :
00031         size(elements),
00032         capacity(elements + 5),
00033         array(new T[capacity])
00034     {
00035         for (size_t i = 0; i < size; ++i) {
00036             array[i] = value;
00037         }
00038     }
00039
00040     Vector(const std::initializer_list<T>& list) :
00041         size(0),
00042         capacity(list.size() + 5),
00043         array(new T[capacity])
00044     {
00045         for (const T& value : list) {
00046             PushBack(value);
00047         }
00048     }
00049
00050
00051     //Destructor
00052     ~Vector() {
00053         delete[] array;
00054         array = nullptr;
00055         size = 0;
00056         capacity = 0;
00057     }
00058
00059
00060     // Operator=
00061     Vector& operator=(const Vector& rhs) {
00062         if (this != &rhs) {
00063             if (rhs.size > capacity) {
00064                 delete[] array;
00065                 capacity = rhs.size + 5;
00066                 array = new T[capacity];
00067             }
00068             for (size_t i = 0; i < rhs.size; ++i) {
00069                 array[i] = rhs.array[i];
00070             }
00071             size = rhs.size;
00072         }
00073         return *this;
00074     }
00075
00076
00077     // Assign
00078     void Assign(size_t count, const T& value) {
00079         if (count > capacity) {
00080             delete[] array;
00081             capacity = count + 5;
00082             array = new T[capacity];
```

```
00083         }
00084         for (size_t i = 0; i < count; ++i) {
00085             array[i] = value;
00086         }
00087         size = count;
00088     }
00089
00090     template <typename InputIt, typename = typename
      std::enable_if<!std::is_integral<InputIt>::value>::type>
00091     void Assign(InputIt first, InputIt last) {
00092         size_t count = std::distance(first, last);
00093         if (count > capacity) {
00094             delete[] array;
00095             capacity = count + 5;
00096             array = new T[capacity];
00097         }
00098
00099         size_t i = 0;
00100         for (InputIt it = first; it != last; ++it, ++i) {
00101             array[i] = *it;
00102         }
00103
00104         size = count;
00105     }
00106
00107
00108     // Element access
00109     T& At(size_t index) {
00110         if ((index < 0) || (index >= size))
00111         {
00112             throw std::exception("At - Index out of range");
00113         }
00114         return array[index];
00115     }
00116
00117     const T& At(size_t index) const {
00118         if ((index < 0) || (index >= size)) {
00119             throw std::exception("At - Index out of range");
00120         }
00121         return array[index];
00122     }
00123
00124     T& operator[](size_t index) {
00125         return array[index];
00126     }
00127
00128     T& Front() {
00129         return At(0);
00130     }
00131
00132     const T& Front() const {
00133         return At(0);
00134     }
00135
00136     T& Back() {
00137         return At(size - 1);
00138     }
00139
00140     const T& Back() const {
00141         return At(size - 1);
00142     }
00143
00144
00145     // Iterators+
00146     T* begin() {
00147         return array;
00148     }
00149
00150     T* end() {
00151         return array + size;
00152     }
00153
00154     T* rbegin() {
00155         return (size == 0) ? nullptr : array + size - 1;
00156     }
00157
00158     T* rend() {
00159         return (size == 0) ? nullptr : array - 1;
00160     }
00161
00162     // Capacity+
00163     bool Empty() const {
00164         return size == 0;;
00165     }
00166
00167     size_t Size() const {
00168         return size;
```

```
00169     }
00170
00171     size_t MaxSize() const {
00172         return static_cast<size_t>(-1) / sizeof(T);
00173     }
00174
00175     void Reserve(size_t newCapacity) {
00176         if (newCapacity <= capacity) return;
00177
00178         T* newArray = new T[newCapacity];
00179
00180         for (size_t i = 0; i < size; ++i) {
00181             newArray[i] = array[i];
00182         }
00183
00184         delete[] array;
00185         array = newArray;
00186         capacity = newCapacity;
00187     }
00188
00189     size_t Capacity() const {
00190         return capacity;
00191     }
00192
00193     void ShrinkToFit() {
00194         if (capacity == size) return;
00195
00196         T* newArray = new T[size];
00197
00198         for (size_t i = 0; i < size; ++i) {
00199             newArray[i] = array[i];
00200         }
00201
00202         delete[] array;
00203         array = newArray;
00204         capacity = size;
00205     }
00206
00207
00208     // Modifiers
00209     void Clear() {
00210         size = 0;
00211     }
00212
00213     void Insert(size_t index, const T& value) {
00214         if ((index < 0) || (index > size)) {
00215             throw std::exception("Insert - Index out of range");
00216         }
00217
00218         if (size >= capacity) {
00219             capacity *= 2;
00220             T* newarray = new T[capacity];
00221             for (size_t i = 0; i < index; ++i) {
00222                 newarray[i] = array[i];
00223             }
00224             newarray[index] = value;
00225             for (size_t i = index; i < size; ++i) {
00226                 newarray[i + 1] = array[i];
00227             }
00228             delete[] array;
00229             array = newarray;
00230         }
00231         else {
00232             for (size_t i = size; i > index; --i) {
00233                 array[i] = array[i - 1];
00234             }
00235             array[index] = value;
00236         }
00237         ++size;
00238     }
00239
00240     template<typename InputIt>
00241     void InsertRange(size_t index, InputIt first, InputIt last) {
00242         if (index < 0 || index > size) {
00243             throw std::out_of_range("InsertRange - Index out of range");
00244         }
00245         size_t count = std::distance(first, last);
00246         if (size + count > capacity) {
00247             Reserve(size + count + 5);
00248         }
00249         for (size_t i = size; i > index; --i) {
00250             array[i + count - 1] = array[i - 1];
00251         }
00252         for (size_t i = 0; i < count; ++i, ++first) {
00253             array[index + i] = *first;
00254         }
00255
```

```
00256         size += count;
00257     }
00258
00259     template<typename... Args>
00260     void Emplace(size_t index, Args&&... args) {
00261         if (index < 0 || index > size) {
00262             throw std::out_of_range("Emplace - Index out of range");
00263         }
00264
00265         if (size >= capacity) {
00266             Reserve(capacity * 2);
00267         }
00268         for (size_t i = size; i > index; --i) {
00269             array[i] = array[i - 1];
00270         }
00271
00272         array[index] = T(std::forward<Args>(args)...);
00273         ++size;
00274     }
00275
00276     void Erase(size_t index) {
00277         if ((index < 0) || (index >= size)) {
00278             throw std::exception("Erase - Index out of range");
00279         }
00280
00281         for (size_t i = index; i < size - 1; ++i) {
00282             array[i] = array[i + 1];
00283         }
00284         --size;
00285     }
00286
00287     void PushBack(const T& value) {
00288         if (size >= capacity) {
00289             capacity *= 2;
00290             T* newarray = new T[capacity];
00291             for (size_t i = 0; i < size; ++i) {
00292                 newarray[i] = array[i];
00293             }
00294             delete[] array;
00295             array = newarray;
00296         }
00297         array[size++] = value;
00298     }
00299
00300     template<typename... Args>
00301     void EmplaceBack(Args&&... args) {
00302         if (size >= capacity) {
00303             Reserve(capacity * 2);
00304         }
00305
00306         array[size++] = T(std::forward<Args>(args)...);
00307     }
00308
00309     template<typename InputIt>
00310     void AppendRange(InputIt first, InputIt last) {
00311         for (auto it = first; it != last; ++it) {
00312             PushBack(*it);
00313         }
00314     }
00315
00316     void PopBack() {
00317         if (size == 0) {
00318             throw std::exception("Pop back on empty vector!");
00319         }
00320         --size;
00321     }
00322
00323     void Resize(size_t newSize, const T& value = T()) {
00324         if (newSize < size) {
00325             size = newSize;
00326         }
00327         else {
00328             if (newSize > capacity) {
00329                 Reserve(newSize + 5);
00330             }
00331             for (size_t i = size; i < newSize; ++i) {
00332                 array[i] = value;
00333             }
00334             size = newSize;
00335         }
00336     }
00337
00338     void Swap(Vector& other) {
00339         size_t tempSize = size;
00340         size = other.size;
00341         other.size = tempSize;
00342
```

```
00343         size_t tempCapacity = capacity;
00344         capacity = other.capacity;
00345         other.capacity = tempCapacity;
00346 
00347         T* tempArray = array;
00348         array = other.array;
00349         other.array = tempArray;
00350     }
00351 
00352 
00353     // Non-member functions
00354     bool operator==(const Vector& rhs) const {
00355         if (Size() != rhs.Size()) return false;
00356 
00357         for (size_t i = 0; i < Size(); ++i) {
00358             if (array[i] != rhs.array[i]) return false;
00359         }
00360         return true;
00361     }
00362 
00363     bool operator!=(const Vector& rhs) const
00364     {
00365         return !(*this == rhs);
00366     }
00367 
00368 
00369     // Operator«
00370     friend std::ostream& operator«(std::ostream& ostr, const Vector& rhs)
00371     {
00372         for (size_t i = 0; i < rhs.size; ++i) {
00373             ostr « rhs.array[i] « " ";
00374         }
00375         /*ostr « " || ";
00376 
00377         for (int i = rhs.size; i < rhs.capacity; ++i) {
00378             ostr « rhs.array[i] « " ";
00379         }
00380 
00381         ostr « std::endl;*/
00382 
00383         return ostr;
00384     }
00385 
00386 
00387 };
00388 
00389 #endif
```

6.10 VECTOR/Include/zmogus.h File Reference

Classes

- class Zmogus

6.11 zmogus.h

Go to the documentation of this file.

```
00001 #ifndef ZMOGUS_H
00002 #define ZMOGUS_H
00003 
00004 class Zmogus {
00005 protected:
00006     std::string var_{}, pav_{};
00007 
00008 public:
00010     explicit Zmogus(std::string var = "", std::string pav = "") :
00011         var_(std::move(var)), pav_(std::move(pav)) {}
00012 
00013     ~Zmogus() = default;
00014 
00016     Zmogus(const Zmogus& other) :
00017         var_(other.var_),
00018         pav_(other.pav_){}
00019 
```

```
00021     Zmogus& operator=(const Zmogus& other) {
00022         if (this != &other) {
00023             var_ = other.var_;
00024             pav_ = other.pav_;
00025         }
00026         return *this;
00027     }
00028
00030     Zmogus(Zmogus&& other) noexcept :
00031         var_(std::move(other.var_)),
00032         pav_(std::move(other.pav_)){
00033         other.var_.clear();
00034         other.pav_.clear();
00035     }
00036
00038     Zmogus& operator=(Zmogus&& other) noexcept {
00039         if (this != &other) {
00040             var_ = std::move(other.var_);
00041             pav_ = std::move(other.pav_);
00042             other.var_.clear();
00043             other.pav_.clear();
00044         }
00045         return *this;
00046     }
00047
00049     friend std::istream& operator»(std::istream& is, Zmogus& s) {
00050         std::string var, pav;
00051
00052         std::cout « "Iveskite varda: ";
00053         is » var;
00054         std::cout « "Iveskite pavarde: ";
00055         is » pav;
00056
00057         s.setVar(var);
00058         s.setPav(pav);
00059         return is;
00060     }
00061
00063     friend std::ostream& operator«(std::ostream& os, const Zmogus& s) {
00064         os « "Vardas: " « s.getVar() « " Pavarde: " « s.getPav();
00065         return os;
00066     }
00067
00069     virtual void setVar(const std::string& var) = 0;
00070     virtual void setPav(const std::string& pav) = 0;
00071
00073     virtual std::string getVar() const { return var_; }
00074     virtual std::string getPav() const { return pav_; }
00075 };
00076
00077 #endif
```

6.12 VECTOR/src/fileGenerator.cpp File Reference

```
#include "meinelib.h"
#include "studentas.h"
#include "functionsCallsVector.h"
```

Functions

- void failoKurimas (int studentuSk)

6.12.1 Function Documentation

6.12.1.1 failoKurimas()

```
void failoKurimas (
            int studentuSk)
```

Definition at line 5 of file fileGenerator.cpp.

6.13 fileGenerator.cpp

Go to the documentation of this file.

```
00001 #include "meinelib.h"
00002 #include "studentas.h"
00003 #include "functionsCallsVector.h"
00004
00005 void failoKurimas(int studentuSk) {
00006     std::string failoVardas = "studList";
00007
00008     if (studentuSk == 0) {
00009     std::cout « "Kiek studentu norite sukurti? (1 - 10 000 000): \n";
00010     studentuSk = ivestiesPatikrinimas(1, 10000000);
00011     }
00012
00013     auto pradzia = hrClock::now();
00014     std::cout « "\nPalaukite, kuriamas failas...\n\n";
00015     failoVardas += std::to_string(studentuSk) + ".txt";
00016
00017     std::stringstream outputas;
00018     int pazymiuSk = 7;
00019
00020     outputas « std::left « std::setw(20) « "Vardas" « std::setw(20) « "Pavarde" « std::setw(20);
00021
00022     for (int i = 0; i < pazymiuSk; i++) {
00023         outputas « std::left « std::setw(6) « "ND" + std::to_string(i + 1);
00024
00025     }
00026     outputas « std::left « std::setw(6) « "Egz." « endl;
00027
00028     for (int i = 0; i < studentuSk; i++) {
00029         outputas « std::left « std::setw(20) « "Vardas" + std::to_string(i + 1) « std::setw(20) «
      "Pavarde" + std::to_string(i + 1);
00030
00031         for (int j = 0; j < pazymiuSk; j++) {
00032             outputas « std::setw(6) « rand() % 10 + 1;
00033
00034         }
00035         outputas « std::setw(6) « rand() % 10 + 1 « endl;
00036
00037     }
00038     std::ofstream rez(failoVardas);
00039     rez « outputas.str();
00040     rez.close();
00041     auto pabaiga = hrClock::now();
00042     auto trukme = std::chrono::duration_cast<sec>(pabaiga - pradzia);
00043     std::cout « "Failas sukurtas per " « fixed « setprecision(8) « trukme.count() « " sec.\n\n";
00044 }
```

6.14 VECTOR/src/generators.cpp File Reference

```
#include "meinelib.h"
#include "studentas.h"
#include "functionsCallsVector.h"
```

Functions

- void randomStudentas (Stud &studentas, bool vyras)
- void randomAtsitiktinisPazymys (Stud &stu)

Variables

- Vector< std::string > vyruVardai
- Vector< std::string > vyruPavardes
- Vector< std::string > moteruVardai
- Vector< std::string > moteruPavardes

6.14.1 Function Documentation

6.14.1.1 randomAtsitiktinisPazymys()

```
void randomAtsitiktinisPazymys (
            Stud & stu)
```

Definition at line 40 of file generators.cpp.

6.14.1.2 randomStudentas()

```
void randomStudentas (
            Stud & studentas,
            bool vyras)
```

Definition at line 29 of file generators.cpp.

6.14.2 Variable Documentation

6.14.2.1 moteruPavardes

```
Vector<std::string> moteruPavardes
```

Initial value:

```
= { "Jonate", "Petraite", "Kazlauskaite", "Baltrunaite", "Simkute", "Kairyte", "Marcinkeviciute",
      "Zabielskaite", "Bagdonaite", "Urbonaite",
                                  "Kavaliauskaite", "Griniute", "Bielskiute", "Matuleviciute", "Sulskite",
      "Sakalauskaite", "Butkute", "Karpaviciute", "Zilinskaite", "Stankeviciute",
                                  "Vasiliauskaite", "Simkeviciute", "Vainyte", "Paskeviciute",
      "Bagdonaviciute", "Aleknaviciute", "Kavoliute", "Miezutaviciute", "Giedraite", "Pavardenyte",
                                  "Sviderskyte", "Malinauskaite", "Gintalaite", "Budreckaite",
      "Tamasauskaite", "Zimnickaite", "Tamuleviciute", "Skorupskaite", "Gaigalaite", "Sadauskaite",
                                  "Janusonyte", "Leskeviciute", "Mikulenaite", "Kairaite",
      "Jarmalaviciute", "Milkeviciute", "Dumciute", "Tamulynaite", "Poskaite", "Savickaite" }
```

Definition at line 23 of file generators.cpp.

6.14.2.2 moteruVardai

```
Vector<std::string> moteruVardai
```

Initial value:

```
= { "Austeja", "Gabija", "Egle", "Ieva", "Lina", "Ruta", "Agne", "Laura", "Monika", "Jurgita",
                                "Kamile", "Indre", "Viktorija", "Justina", "Karolina", "Evelina", "Ugne",
      "Neringa", "Dovile", "Raminta",
                                "Erika", "Gintare", "Alina", "Simona", "Vaida", "Edita", "Julija", "Renata",
      "Sandra", "Svetlana",
                                "Laima", "Zita", "Gitana", "Greta", "Sigita", "Brigita", "Aleksandra",
      "Emilija", "Asta", "Ingrida",
                                "Joana", "Patricija", "Skaiste", "Vitalija", "Giedre", "Rasa", "Lilija",
      "Ona", "Aurelija", "Silvija" }
```

Definition at line 17 of file generators.cpp.

6.14.2.3 vyruPavardes

```
Vector<std::string> vyruPavardes
```

Initial value:

```
= { "Jonaitis", "Petraitis", "Kazlauskas", "Baltrunas", "Simkus", "Kairys", "Marcinkevicius", "Zabielskas",
      "Bagdonas", "Urbonas",
                                "Kavaliauskas", "Puidokas", "Bielskis", "Matulevicius", "Sulskis",
      "Sakalauskas", "Butkus", "Karpavicius", "Zilinskas", "Stankevicius",
                                "Vasiliauskas", "Simkevicius", "Daksevic", "Paskevicius", "Bagdonavicius",
      "Aleknavicius", "Kavolis", "Miezutavicius", "Giedraitis", "Pavardenis",
                                "Sviderskis", "Malinauskas", "Gintalas", "Budreckas", "Tamasauskas",
      "Zimnickas", "Tamulevicius", "Skorupskas", "Gaigalas", "Sadauskas",
                                "Janusonis", "Leskevicius", "Mikulenas", "Kairaitis", "Jarmalavicius",
      "Milkevicus", "Dumcius", "Tamulynas", "Poska", "Savickas" }
```

Definition at line 11 of file generators.cpp.

6.14.2.4 vyruVardai

```
Vector<std::string> vyruVardai
```

Initial value:

```
= { "Jonas", "Mantas", "Tomas", "Lukas", "Dovydas", "Andrius", "Gabrielius", "Erikas", "Vilius", "Domantas",
                              "Augustas", "Mindaugas", "Rokas", "Paulius", "Simas", "Arnas", "Edvinas",
      "Justas", "Kipras", "Martynas",
                              "Pijus", "Vytis", "Zygimantas", "Tautvydas", "Evaldas", "Eimantas",
      "Deividas", "Laurynas", "Karolis", "Gytis",
                              "Benas", "Titas", "Ignas", "Nojus", "Vytautas", "Aivaras", "Saulius",
      "Kristupas", "Orestas", "Armandas",
                              "Jokubas", "Dainius", "Sigitas", "Almantas", "Haroldas", "Julius", "Dziugas",
      "Gediminas", "Antanas", "Vytenis" }
```

Definition at line 5 of file generators.cpp.

6.15 generators.cpp

Go to the documentation of this file.

```
00001 #include "meinelib.h"
00002 #include "studentas.h"
00003 #include "functionsCallsVector.h"
00004
00005 Vector<std::string> vyruVardai = { "Jonas", "Mantas", "Tomas", "Lukas", "Dovydas", "Andrius",
      "Gabrielius", "Erikas", "Vilius", "Domantas",
00006                                 "Augustas", "Mindaugas", "Rokas", "Paulius", "Simas", "Arnas",
      "Edvinas", "Justas", "Kipras", "Martynas",
00007                                 "Pijus", "Vytis", "Zygimantas", "Tautvydas", "Evaldas", "Eimantas",
      "Deividas", "Laurynas", "Karolis", "Gytis",
00008                                 "Benas", "Titas", "Ignas", "Nojus", "Vytautas", "Aivaras", "Saulius",
      "Kristupas", "Orestas", "Armandas",
00009                                 "Jokubas", "Dainius", "Sigitas", "Almantas", "Haroldas", "Julius",
      "Dziugas", "Gediminas", "Antanas", "Vytenis" };
00010
00011 Vector<std::string> vyruPavardes = { "Jonaitis", "Petraitis", "Kazlauskas", "Baltrunas", "Simkus",
      "Kairys", "Marcinkevicius", "Zabielskas", "Bagdonas", "Urbonas",
00012                                  "Kavaliauskas", "Puidokas", "Bielskis", "Matulevicius", "Sulskis",
      "Sakalauskas", "Butkus", "Karpavicius", "Zilinskas", "Stankevicius",
00013                                 "Vasiliauskas", "Simkevicius", "Daksevic", "Paskevicius",
      "Bagdonavicius", "Aleknavicius", "Kavolis", "Miezutavicius", "Giedraitis", "Pavardenis",
00014                                 "Sviderskis", "Malinauskas", "Gintalas", "Budreckas", "Tamasauskas",
      "Zimnickas", "Tamulevicius", "Skorupskas", "Gaigalas", "Sadauskas",
00015                                 "Janusonis", "Leskevicius", "Mikulenas", "Kairaitis", "Jarmalavicius",
      "Milkevicus", "Dumcius", "Tamulynas", "Poska", "Savickas" };
00016
00017 Vector<std::string> moteruVardai = { "Austeja", "Gabija", "Egle", "Ieva", "Lina", "Ruta", "Agne",
      "Laura", "Monika", "Jurgita",
00018                                 "Kamile", "Indre", "Viktorija", "Justina", "Karolina", "Evelina",
      "Ugne", "Neringa", "Dovile", "Raminta",
```

```
00019                                "Erika", "Gintare", "Alina", "Simona", "Vaida", "Edita", "Julija",
      "Renata", "Sandra", "Svetlana",
00020                                "Laima", "Zita", "Gitana", "Greta", "Sigita", "Brigita", "Aleksandra",
      "Emilija", "Asta", "Ingrida",
00021                                "Joana", "Patricija", "Skaiste", "Vitalija", "Giedre", "Rasa",
      "Lilija", "Ona", "Aurelija", "Silvija" };
00022
00023 Vector<std::string> moteruPavardes = { "Jonate", "Petraite", "Kazlauskaite", "Baltrunaite", "Simkute",
      "Kairyte", "Marcinkeviciute", "Zabielskaite", "Bagdonaite", "Urbonaite",
00024                                "Kavaliauskaite", "Griniute", "Bielskiute", "Matuleviciute",
      "Sulskite", "Sakalauskaite", "Butkute", "Karpaviciute", "Zilinskaite", "Stankeviciute",
00025                                "Vasiliauskaite", "Simkeviciute", "Vainyte", "Paskeviciute",
      "Bagdonaviciute", "Aleknaviciute", "Kavoliute", "Miezutaviciute", "Giedraite", "Pavardenyte",
00026                                "Sviderskyte", "Malinauskaite", "Gintalaite", "Budreckaite",
      "Tamasauskaite", "Zimnickaite", "Tamuleviciute", "Skorupskaite", "Gaigalaite", "Sadauskaite",
00027                                "Janusonyte", "Leskeviciute", "Mikulenaite", "Kairaite",
      "Jarmalaviciute", "Milkeviciute", "Dumciute", "Tamulynaite", "Poskaite", "Savickaite" };
00028
00029 void randomStudentas(Stud& studentas, bool vyras) {
      // Sukuria atsitiktini studenta.
00030     if (vyras) {
00031         studentas.setVar (vyruVardai.At(rand() % vyruVardai.Size()));
00032         studentas.setPav (vyruPavardes.At(rand() % vyruPavardes.Size()));
00033     }
00034     else {
00035         studentas.setVar (moteruVardai.At(rand() % moteruVardai.Size()));
00036         studentas.setPav (moteruPavardes.At(rand() % moteruPavardes.Size()));
00037     }
00038 }
00039
00040 void randomAtsitiktinisPazymys(Stud& stu) {
      // Sugeneruoja atsitiktini pazymi.
00041     int pazymiai = rand() % 15 + 1;
00042     stu.setEgz(rand() % 10 + 1);
00043
00044     for (int i = 0; i < pazymiai; i++) {
00045         stu.addPazymys(rand() % 10 + 1);
00046     }
00047 }
```

6.16 VECTOR/src/isvestis.cpp File Reference

```
#include "meinelib.h"
#include "studentas.h"
#include "functionsCallsVector.h"
```

Functions

- void isvestiesMenu (Vector< Stud > &studentai)
- void isvestis (Vector< Stud > &studentai, std::ostream &isvestiesMetodas, const int galutinioPasirinkimas)

6.16.1 Function Documentation

6.16.1.1 isvestiesMenu()

```
void isvestiesMenu (
            Vector< Stud > & studentai)
```

Definition at line 5 of file isvestis.cpp.

6.16.1.2 isvestis()

```
void isvestis (
            Vector< Stud > & studentai,
            std::ostream & isvestiesMetodas,
            const int galutinioPasirinkimas)
```

Definition at line 43 of file isvestis.cpp.

6.17 isvestis.cpp

Go to the documentation of this file.

```
00001 #include "meinelib.h"
00002 #include "studentas.h"
00003 #include "functionsCallsVector.h"
00004
00005 void isvestiesMenu(Vector<Stud>& studentai) {
00006     std::cout « "Pasirinkite norima studentu isvedimo buda: \n";
00007     std::cout « "1 - Isvesti i terminala\n";
00008     std::cout « "2 - Isvesti i faila\n";
00009     std::cout « "3 - Isvesti i du failus, suskirsto\n";
00010     int isvestiesPasirinkimas = ivestiesPatikrinimas(1, 3);
00011
00012     std::cout « "Pasirinkite koki galutini norite matyti: \n";
00013     std::cout « "1 - Vidurki \n";
00014     std::cout « "2 - Mediana \n";
00015     std::cout « "3 - Abu \n";
00016     int galutinioPasirinkimas = ivestiesPatikrinimas(1, 3);
00017
00018     std::cout « "Pasirinkite studentu rusiavima:\n";
00019     std::cout « "1 - Pagal varda\n";
00020     std::cout « "2 - Pagal pavarde\n";
00021     std::cout « "3 - Pagal galutini pazymi pagal vidurki \n";
00022     std::cout « "4 - Pagal galutini pazymi pagal mediana \n";
00023     int rusiavimoPasirinkimas = ivestiesPatikrinimas(1, 4);
00024
00025
00026     if (isvestiesPasirinkimas == 1) {
00027         studentuGalutiniuSkaiciavimas(studentai);
00028         rusiavimas(studentai, rusiavimoPasirinkimas);
00029         isvestis(studentai, std::cout, galutinioPasirinkimas);
00030     }
00031
00032     else if (isvestiesPasirinkimas == 2){
00033         std::ofstream output("rezultatai.txt");
00034         studentuGalutiniuSkaiciavimas(studentai);
00035         rusiavimas(studentai, rusiavimoPasirinkimas);
00036         isvestis(studentai, output, galutinioPasirinkimas);
00037     }
00038     else if (isvestiesPasirinkimas == 3) {
00039         fileFilter(studentai, galutinioPasirinkimas, rusiavimoPasirinkimas);
00040     }
00041 }
00042
00043 void isvestis(Vector<Stud>& studentai, std::ostream& isvestiesMetodas, const int
      galutinioPasirinkimas) {
00044     int longest_name{};
00045     int longest_surname{};
00046     for (auto& studentas : studentai) {
00047         if (studentas.getVar().length() > longest_name) {
00048             longest_name = static_cast<int>(studentas.getVar().length());
00049         }
00050         if (studentas.getPav().length() > longest_surname) {
00051             longest_surname = static_cast<int>(studentas.getPav().length());
00052         }
00053     }
00054     int isvesties_pavardes_ilgis{ ((longest_surname > 7 ? longest_surname + 2 : 8)) };
00055     int isvesties_vardo_ilgis{ ((longest_name > 6 ? longest_name + 2 : 8)) };
00056
00057     std::stringstream isvestis{};
00058
00059     isvestis « std::left « std::setw(isvesties_vardo_ilgis) « "Vardas" «
      std::setw(isvesties_pavardes_ilgis) « "Pavarde";
00060
00061     if (galutinioPasirinkimas == 1 || galutinioPasirinkimas == 3) {
00062         isvestis « std::setw(17) « "Galutinis (Vid.)";
```

```
00063     }
00064     if (galutinioPasirinkimas == 2 || galutinioPasirinkimas == 3) {
00065         isvestis « std::setw(17) « "Galutinis (Med.)";
00066     }
00067
00068     isvestis « "\n";
00069
00070     if (galutinioPasirinkimas == 1) {
00071         for (auto& i : studentai) {
00072             isvestis « std::setw(isvesties_pavardes_ilgis) « i.getVar()
00073                 « std::setw(isvesties_vardo_ilgis) « i.getPav()
00074                 « std::setw(20) « std::left « i.getVidurkis() « "\n";
00075         }
00076     }
00077     else if (galutinioPasirinkimas == 2) {
00078         for (auto& i : studentai) {
00079             isvestis « std::setw(isvesties_pavardes_ilgis) « i.getVar()
00080                 « std::setw(isvesties_vardo_ilgis) « i.getPav()
00081                 « std::setw(20) « std::left « i.getMediana() « "\n";
00082         }
00083     }
00084     else if (galutinioPasirinkimas == 3) {
00085         for (auto& i : studentai) {
00086             isvestis « std::setw(isvesties_pavardes_ilgis) « i.getVar()
00087                 « std::setw(isvesties_vardo_ilgis) « i.getPav()
00088                 « std::setw(20) « std::left « i.getVidurkis() « "   "
00089                 « std::setw(20) « std::left « i.getMediana() « "\n";
00090         }
00091     }
00092
00093     isvestiesMetodas « isvestis.str();
00094 }
```

6.18 VECTOR/src/ivestis.cpp File Reference

```
#include "meinelib.h"
#include "studentas.h"
#include "functionsCallsVector.h"
```

Functions

- void readRanka (Stud &stu)
- void readName_makeGrade (Stud &stu)
- void makeStud (Stud &stu)
- void fileRead (Vector< Stud > &studentai, std::string ivestas_vardas)

6.18.1 Function Documentation

6.18.1.1 fileRead()

```
void fileRead (
            Vector< Stud > & studentai,
            std::string ivestas_vardas)
```

Definition at line 48 of file ivestis.cpp.

6.18.1.2 makeStud()

```
void makeStud (
            Stud & stu)
```

Definition at line 43 of file ivestis.cpp.

6.18.1.3 readName_makeGrade()

```
void readName_makeGrade (
            Stud & stu)
```

Definition at line 32 of file ivestis.cpp.

6.18.1.4 readRanka()

```
void readRanka (
            Stud & stu)
```

Definition at line 5 of file ivestis.cpp.

6.19 ivestis.cpp

Go to the documentation of this file.

```
00001 #include "meinelib.h"
00002 #include "studentas.h"
00003 #include "functionsCallsVector.h"
00004
00005 void readRanka(Stud& stu) {
      // Vartotojo praso ivesti studento varda, pavarde, pazymius.
00006     int input, i = 1;
00007     std::string a;
00008     std::cout « "Iveskite studento varda: ";
00009     std::cin » a;
00010     stu.setVar(a);
00011     std::cout « "Iveskite studento pavarde: ";
00012     std::cin » a;
00013     stu.setPav(a);
00014
00015     std::cout « "Iveskite studento egzamino pazymi: ";
00016     input = ivestiesPatikrinimas(0, 10);
00017     stu.setEgz(input);
00018     std::cout « "\n\n" « "Iveskite studento namu darbu pazymius. Ivedus visus pazymius, iveskite
      -1.\n";
00019     std::cout « "Jeigu darbas neatliktas iveskite 0.\n\n";
00020
00021     while (true) {
00022         std::cout « i « "-asis pazymys: ";
00023         input = ivestiesPatikrinimas(0, 10, -1);
00024         if (input == -1) {
00025             break;
00026         }
00027         stu.addPazymys(input);
00028         i++;
00029     }
00030 }
00031
00032 void readName_makeGrade(Stud& stu) {
00033     std::string a;
00034     std::cout « "Iveskite studento varda: ";
00035     std::cin » a;
00036     stu.setVar(a);
00037     std::cout « "Iveskite studento pavarde: ";
00038     std::cin » a;
00039     stu.setPav(a);
00040     randomAtsitiktinisPazymys(stu);
00041 }
00042
00043 void makeStud(Stud& stu) {
      // Sukuria atsitiktini studenta.
00044     randomStudentas(stu, rand() % 2);
00045     randomAtsitiktinisPazymys(stu);
00046 }
00047
00048 void fileRead(Vector<Stud>& studentai, std::string ivestas_vardas) {
00049     std::stringstream buffer;
00050     std::ifstream duom(ivestas_vardas);
00051     if (!duom) {
```

```
00052         throw std::runtime_error("\nFailas nerastas.\n\n");
00053         return;
00054     }
00055     buffer « duom.rdbuf();
00056     duom.close();
00057
00058     std::string line;
00059     getline(buffer, line);
00060     while (getline(buffer, line)) {
00061         Stud tempStu;
00062         std::istringstream iss(line);
00063         std::string vardas{}, pavarde{};
00064         iss » vardas » pavarde;
00065         tempStu.setVar(vardas);
00066         tempStu.setPav(pavarde);
00067         int pazymys{};
00068
00069         while (iss » pazymys) {
00070             tempStu.addPazymys(pazymys);
00071         }
00072         tempStu.setEgz(tempStu.getPazymys().Back());
00073         tempStu.removePazymys();
00074         studentai.PushBack(tempStu);
00075     }
00076 }
```

6.20 VECTOR/src/ivestisPatikrinimas.cpp File Reference

```
#include "meinelib.h"
#include "studentas.h"
#include "functionsCallsVector.h"
```

Functions

- int ivestiesPatikrinimas (const int nuo, const int iki)
- int ivestiesPatikrinimas (const int nuo, const int iki, const int sustabdymoSalyga)

6.20.1 Function Documentation

6.20.1.1 ivestiesPatikrinimas() [1/2]

```
int ivestiesPatikrinimas (
            const int nuo,
            const int iki)
```

Definition at line 5 of file ivestisPatikrinimas.cpp.

6.20.1.2 ivestiesPatikrinimas() [2/2]

```
int ivestiesPatikrinimas (
            const int nuo,
            const int iki,
            const int sustabdymoSalyga)
```

Definition at line 26 of file ivestisPatikrinimas.cpp.

6.21 ivestisPatikrinimas.cpp

Go to the documentation of this file.

```
00001 #include "meinelib.h"
00002 #include "studentas.h"
00003 #include "functionsCallsVector.h"
00004
00005 int ivestiesPatikrinimas(const int nuo, const int iki) {
00006     int input{};
00007     while (true) {
00008         try {
00009             std::cin » input;
00010             if (input < nuo || input > iki) {
00011                 std::cout « "\n\n!!!!Iveskite skaiciu nuo " « nuo « " iki " « iki « ".!!!!\n\n\n";
00012                 continue;
00013             }
00014         }
00015         catch (...) {
00016             std::cin.clear();
00017             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00018             std::cout « "\n\n!!!!Ivestis neteisinga. Bandykite isnaujo.!!!!\n\n\n";
00019             continue;
00020         }
00021         break;
00022     }
00023     return input;
00024 }
00025
00026 int ivestiesPatikrinimas(const int nuo, const int iki, const int sustabdymoSalyga) {
00027     int input{};
00028     while (true) {
00029         try {
00030             std::cin » input;
00031             if (input == sustabdymoSalyga) {
00032                 return sustabdymoSalyga;
00033             }
00034
00035             if (input < nuo || input > iki) {
00036                 std::cout « "\n\n!!!!Iveskite skaiciu nuo " « nuo « " iki " « iki « ".!!!!\n\n\n";
00037                 continue;
00038             }
00039         }
00040         catch (...) {
00041             std::cin.clear();
00042             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00043             std::cout « "\n\n!!!!Ivestis neteisinga. Bandykite isnaujo.!!!!\n\n\n";
00044             continue;
00045         }
00046         break;
00047     }
00048     return input;
00049 }
```

6.22 VECTOR/src/main.cpp File Reference

```
#include "meinelib.h"
#include "studentas.h"
#include "functionsCallsVector.h"
```

Functions

- int main ()

6.22.1 Function Documentation

6.22.1.1 main()

```
int main ()
```

Definition at line 5 of file main.cpp.

6.23 main.cpp

Go to the documentation of this file.

```
00001 #include "meinelib.h"
00002 #include "studentas.h"
00003 #include "functionsCallsVector.h"
00004
00005 int main() {
00006     std::cin.exceptions(std::ios::failbit);
00007     srand(time(NULL));
00008     Vector<Stud> studentai;
00009     int n = 0,baigimas = 7;
00010     while (true) {
00011         Stud studentas;
00012         std::cout « "Pasirinkite norima studento duomenu surasyma irasant skaiciu nuo 1 iki "«
      baigimas «".\n";
00013         std::cout « "--------------------------------------------------------------------------------\n";
00014         std::cout « "1 - Ivestis ranka\n";
00015         std::cout « "2 - Ivestis ranka (Generuojami tik pazymiai)\n";
00016         std::cout « "3 - Generuojamas studentas ir pazymiai\n";
00017         std::cout « "4 - Failo nuskaitymas\n";
00018         std::cout « "5 - Failo kurimas\n";
00019         std::cout « "6 - Testine aplinka\n";
00020         std::cout « baigimas « " - Baigti darba\n";
00021         std::cout « "--------------------------------------------------------------------------------\n";
00022         int menuPasirinkimas = ivestiesPatikrinimas(1, baigimas, baigimas);
00023
00024         if (menuPasirinkimas == baigimas) { break;}
00025
00026         switch (menuPasirinkimas) {
00027         case 1:
00028             readRanka(studentas);
00029             studentai.PushBack(studentas);
00030             break;
00031         case 2:
00032             readName_makeGrade(studentas);
00033             studentai.PushBack(studentas);
00034             break;
00035         case 3:
00036             std::cout « "Kiek studentu sugeneruoti?\n";
00037             std::cin » n;
00038             for (int i = 0; i < n; i++) {
00039                 makeStud(studentas);
00040                 studentai.PushBack(studentas);
00041             }
00042             break;
00043         case 4:
00044             try {
00045                 std::cout « "Iveskite norimo failo pavadinima be kabuciu:\n";
00046                 for (const auto& entry : fs::directory_iterator(".")) {
00047                     if (entry.path().extension() == ".txt") {
00048                         std::cout « entry.path().filename() « endl;
00049                     }
00050                 }
00051                 std::string ivestas_vardas;
00052                 std::cin » ivestas_vardas;
00053                 fileRead(studentai, ivestas_vardas);
00054             }
00055             catch (const std::exception& e) {
00056                 std::cerr « e.what() « std::endl;
00057                 continue;
00058             }
00059             break;
00060         case 5:
00061             failoKurimas(0);
00062             break;
00063         case 6:
00064             try {
00065                 testMenu();
00066             }
00067             catch (const std::exception& e) {
00068                 std::cerr « e.what() « std::endl;
00069                 break;
00070             }
00071             break;
00072         }
00073     }
00074     isvestiesMenu(studentai);
00075 }
```

6.24 VECTOR/src/Stud.cpp File Reference

```
#include "meinelib.h"
#include "studentas.h"
#include "functionsCallsVector.h"
```

Functions

- void studentuGalutiniuSkaiciavimas (Vector< Stud > &studentai)

6.24.1 Function Documentation

6.24.1.1 studentuGalutiniuSkaiciavimas()

```
void studentuGalutiniuSkaiciavimas (
            Vector< Stud > & studentai)
```

Definition at line 35 of file Stud.cpp.

6.25 Stud.cpp

Go to the documentation of this file.

```
00001 #include "meinelib.h"
00002 #include "studentas.h"
00003 #include "functionsCallsVector.h"
00004
00005 void Stud::calculateGalVidurkis() {
00006     int pazymiuSuma{};
00007     if (this->getPazymys().Empty()) {
00008         this->galVidurkis_ = 0.0f;
00009         return;
00010     }
00011     for (const auto& pazymys : this->getPazymys()) {
00012         pazymiuSuma += pazymys;
00013     }
00014     this->galVidurkis_ = pazymiuSuma / this->getPazymys().Size() * 0.4f + this->egz_ * 0.6f;
00015 }
00016
00017 void Stud::calculateGalMediana() {
00018     float galMediana{};
00019     if (this->getPazymys().Empty()) {
00020         this->galMediana_ = 0.0f;
00021         return;
00022     }
00023     Vector<int> sortedPazymiai = this->getPazymys();
00024     std::sort(sortedPazymiai.begin(), sortedPazymiai.end());
00025
00026     if (sortedPazymiai.Size() % 2 == 0) {
00027         galMediana = (sortedPazymiai[sortedPazymiai.Size() / 2 - 1] +
      sortedPazymiai[sortedPazymiai.Size() / 2]) / 2.0f;
00028     }
00029     else {
00030         galMediana = sortedPazymiai[sortedPazymiai.Size() / 2];
00031     }
00032     this->galMediana_ = galMediana * 0.4f + this->egz_ * 0.6f;
00033 }
00034
00035 void studentuGalutiniuSkaiciavimas(Vector<Stud>& studentai) {
00036     if (studentai.Back().getVidurkis() == 0 || studentai.Back().getMediana() == 0) {
00037         for (auto& studentas : studentai) {
00038             studentas.calculateGalVidurkis();
00039             studentas.calculateGalMediana();
00040         }
00041     }
00042 }
```

6.26 VECTOR/src/studentuRusiavimas.cpp File Reference

```
#include "meinelib.h"
#include "studentas.h"
#include "functionsCallsVector.h"
```

Functions

- void rusiavimas (Vector< Stud > &studentai, int rusiavimoPasirinkimas)

6.26.1 Function Documentation

6.26.1.1 rusiavimas()

```
void rusiavimas (
            Vector< Stud > & studentai,
            int rusiavimoPasirinkimas)
```

Definition at line 5 of file studentuRusiavimas.cpp.

6.27 studentuRusiavimas.cpp

Go to the documentation of this file.

```
00001 #include "meinelib.h"
00002 #include "studentas.h"
00003 #include "functionsCallsVector.h"
00004
00005 void rusiavimas(Vector<Stud>& studentai, int rusiavimoPasirinkimas) {
00006     auto compareByName = [](const Stud& a, const Stud& b) {
00007         return a.getVar() < b.getVar();
00008         };
00009     auto compareBySurname = [](const Stud& a, const Stud& b) {
00010         return a.getPav() < b.getPav();
00011         };
00012     auto compareByFinalGradeVid = [](const Stud& a, const Stud& b) {
00013         return a.getVidurkis() < b.getVidurkis();
00014         };
00015     auto compareByFinalGradeMed = [](const Stud& a, const Stud& b) {
00016         return a.getMediana() < b.getMediana();
00017         };
00018
00019     switch (rusiavimoPasirinkimas) {
00020     case 1:
00021         sort(studentai.begin(), studentai.end(), compareByName);
00022         break;
00023     case 2:
00024         sort(studentai.begin(), studentai.end(), compareBySurname);
00025         break;
00026     case 3:
00027         sort(studentai.begin(), studentai.end(), compareByFinalGradeVid);
00028         break;
00029     case 4:
00030         sort(studentai.begin(), studentai.end(), compareByFinalGradeMed);
00031         break;
00032     default:
00033         std::cout « "Neteisingas pasirinkimas. Nerusiavome.\n";
00034         break;
00035     }
00036 }
```

6.28 VECTOR/src/studentuSkirstymas.cpp File Reference

```
#include "meinelib.h"
#include "studentas.h"
#include "functionsCallsVector.h"
```

Functions

- void fileFilter (Vector< Stud > &studentai, const int galutinioPasirinkimas, const int rusiavimoPasirinkimas)

6.28.1 Function Documentation

6.28.1.1 fileFilter()

```
void fileFilter (
            Vector< Stud > & studentai,
            const int galutinioPasirinkimas,
            const int rusiavimoPasirinkimas)
```

Definition at line 5 of file studentuSkirstymas.cpp.

6.29 studentuSkirstymas.cpp

Go to the documentation of this file.

```
00001 #include "meinelib.h"
00002 #include "studentas.h"
00003 #include "functionsCallsVector.h"
00004
00005 void fileFilter(Vector<Stud>& studentai, const int galutinioPasirinkimas, const int
      rusiavimoPasirinkimas) {
00006     Vector<Stud> mokslinciai{};
00007     Vector<Stud> vargsai{};
00008     studentuGalutiniuSkaiciavimas(studentai);
00009
00010     if (rusiavimoPasirinkimas == 1 || rusiavimoPasirinkimas == 2) {
00011         rusiavimas(studentai, 3);
00012     }
00013     else {
00014         rusiavimas(studentai, rusiavimoPasirinkimas);
00015     }
00016
00017     if (rusiavimoPasirinkimas == 3) {
00018         auto partition_iteratorius = std::partition_point(studentai.begin(), studentai.end(),
00019             [](const Stud& studentas) {return studentas.getVidurkis() < 5.0f;}
00020         );
00021         vargsai.Assign(studentai.begin(), partition_iteratorius);
00022         mokslinciai.Assign(partition_iteratorius, studentai.end());
00023     }
00024     else if (rusiavimoPasirinkimas == 4) {
00025         auto partition_iteratorius = std::partition_point(studentai.begin(), studentai.end(),
00026             [](const Stud& studentas) {return studentas.getMediana() < 5.0f;}
00027         );
00028         vargsai.Assign(studentai.begin(), partition_iteratorius);
00029         mokslinciai.Assign(partition_iteratorius, studentai.end());
00030     }
00031
00032     try {
00033         std::ofstream mokslinciai_output("mokslinciai.txt");
00034         isvestis(mokslinciai, mokslinciai_output, galutinioPasirinkimas);
00035         mokslinciai_output.close();
00036
00037         std::ofstream vargsai_output("vargsai.txt");
00038         isvestis(vargsai, vargsai_output, galutinioPasirinkimas);
00039         vargsai_output.close();
00040     }
00041     catch (std::exception& e) {
00042         std::cerr « "Ivyko klaida isvedant i failus failus: " « e.what() « "\n";
00043     }
00044 }
```

6.30 VECTOR/src/test.cpp File Reference

```
#include "meinelib.h"
#include "studentas.h"
#include "functionsCallsVector.h"
```

Functions

- void testMenu ()
- void nuskaitymoTestas ()
- void programTest ()
- void studentuTest ()
- void vectorCompare ()
- void atmintiesPerskirstymas ()

6.30.1 Function Documentation

6.30.1.1 atmintiesPerskirstymas()

```
void atmintiesPerskirstymas ()
```

Definition at line 283 of file test.cpp.

6.30.1.2 nuskaitymoTestas()

```
void nuskaitymoTestas ()
```

Definition at line 39 of file test.cpp.

6.30.1.3 programTest()

```
void programTest ()
```

Definition at line 101 of file test.cpp.

6.30.1.4 studentuTest()

```
void studentuTest ()
```

Konstruktoriaus testas

Tuscio konstruktoriaus testas

Definition at line 165 of file test.cpp.

6.30.1.5 testMenu()

```
void testMenu ()
```

Definition at line 5 of file test.cpp.

6.30.1.6 vectorCompare()

```
void vectorCompare ()
```

Definition at line 259 of file test.cpp.

6.31 test.cpp

Go to the documentation of this file.

```
00001 #include "meinelib.h"
00002 #include "studentas.h"
00003 #include "functionsCallsVector.h"
00004
00005 void testMenu() {
00006     int baigimas = 6;
00007     while (true) {
00008         std::cout « "Pasirinkite norima testavimo buda: \n";
00009         std::cout « "1 - Nuskaitymo testas\n";
00010         std::cout « "2 - Programos unit testas\n";
00011         std::cout « "3 - Studentu unit testas\n";
00012         std::cout « "4 - Vector compare\n";
00013         std::cout « "5 - Vector atminties perskirstymas\n";
00014         std::cout « baigimas « " - Baigti testavima\n";
00015         int testPasirinkimas = ivestiesPatikrinimas(1, baigimas, baigimas);
00016
00017         if (testPasirinkimas == baigimas) return;
00018
00019         switch (testPasirinkimas) {
00020         case 1:
00021             nuskaitymoTestas();
00022             break;
00023         case 2:
00024             programTest();
00025             break;
00026         case 3:
00027             studentuTest();
00028             break;
00029         case 4:
00030             vectorCompare();
00031             break;
00032         case 5:
00033             atmintiesPerskirstymas();
00034             break;
00035         }
00036     }
00037 }
00038
00039 void nuskaitymoTestas() {
00040     int n = 0;
00041     sec dursum(0.0);
00042     std::string ivestas_vardas;
00043     Vector<Stud> studentai;
00044
00045     std::cout « "Kiek kartu norite nuskaitineti faila?\n";
00046     std::cin » n;
00047     std::cout « "Koki faila norite nuskaityti? (Iveskite be kabuciu)\n";
00048
00049     for (const auto& entry : fs::directory_iterator(".")) {
00050         if (entry.path().extension() == ".txt") {
00051             std::cout « entry.path().filename() « endl;
00052         }
00053     }
00054     std::cin » ivestas_vardas;
00055
00056     for (int i = 0; i < n; i++) {
00057         auto start = hrClock::now();
```

```
00058
00059         std::stringstream buffer;
00060         std::ifstream duom(ivestas_vardas);
00061         if (!duom) {
00062             throw std::runtime_error("\nFailas nerastas.\n\n");
00063             return;
00064         }
00065         buffer « duom.rdbuf();
00066         duom.close();
00067
00068         std::string line;
00069         getline(buffer, line);
00070             auto startVector = hrClock::now();
00071         while (getline(buffer, line)) {
00072             Stud tempStu;
00073             std::istringstream iss(line);
00074             std::string vardas{}, pavarde{};
00075             iss » vardas » pavarde;
00076             tempStu.setVar(vardas);
00077             tempStu.setPav(pavarde);
00078             int pazymys{};
00079
00080             while (iss » pazymys) {
00081                 tempStu.addPazymys(pazymys);
00082             }
00083             tempStu.setEgz(tempStu.getPazymys().Back());
00084             tempStu.removePazymys();
00085             studentai.PushBack(tempStu);
00086         }
00087             auto endVector = hrClock::now();
00088             sec durationVector = endVector - startVector;
00089             std::cout « "Studentas nuskaitytas per " « fixed « setprecision(8) «
      durationVector.count() « " sec\n";
00090
00091         sec duration = hrClock::now() - start;
00092         std::cout « "Failas nuskaitytas per " « fixed « setprecision(8) « duration.count() « " sec\n";
00093         dursum += duration;
00094     }
00095     studentai.Clear();
00096     studentai.ShrinkToFit();
00097     std::cout « "Viso laiko: " « fixed « setprecision(8) « dursum.count() « " sec\n";
00098     std::cout « "Avg: " « fixed « setprecision(8) « dursum.count() / n « " sec\n\n";
00099 }
00100
00101 void programTest() {
00102     Vector<Stud> studentai;
00103     Stud studentas;
00104
00105     // Test 1: Manual input
00106     readRanka(studentas);
00107     studentai.PushBack(studentas);
00108
00109     if (studentai.Size() == 1)
00110         std::cout « "[PASS] Added one student\n\n";
00111     else
00112         std::cout « "[FAIL] studentai size: " « studentai.Size() « ", expected 1\n\n";
00113
00114     if (studentai[0].getVar() == studentas.getVar())
00115         std::cout « "[PASS] Name matches\n\n";
00116     else
00117         std::cout « "[FAIL] Name mismatch\n\n";
00118
00119     if (studentai[0].getPav() == studentas.getPav())
00120         std::cout « "[PASS] Surname matches\n\n";
00121     else
00122         std::cout « "[FAIL] Surname mismatch\n\n";
00123
00124     // Test 2: Generate random grades
00125     readName_makeGrade(studentas);
00126     if (!studentas.getPazymys().Empty())
00127         std::cout « "[PASS] Random grades generated\n\n";
00128     else
00129         std::cout « "[FAIL] Random grades missing\n\n";
00130
00131     // Test 3: Create a random student
00132     makeStud(studentas);
00133     if (!studentas.getPazymys().Empty())
00134         std::cout « "[PASS] Random student created\n\n";
00135     else
00136         std::cout « "[FAIL] Random student creation failed\n\n";
00137
00138     // Test 4: File reading
00139     try {
00140         fileRead(studentai, "test.txt");
00141         if (!studentai.Empty())
00142             std::cout « "[PASS] File reading successful\n\n";
00143         else
```

```
00144             std::cout « "[FAIL] File read but no students loaded\n\n";
00145     }
00146     catch (const std::exception& e) {
00147         std::cout « "[FAIL] File reading exception: " « e.what() « '\n\n';
00148     }
00149
00150     // Test 5: File creation
00151     failoKurimas(100);
00152     if (fs::exists("studList100.txt"))
00153         std::cout « "[PASS] File creation successful\n\n";
00154     else
00155         std::cout « "[FAIL] File not created\n\n";
00156
00157     // Test 6: Output filtering
00158     isvestiesMenu(studentai);
00159
00160     // Test 7: Sorting
00161     rusiavimas(studentai, 1);
00162     std::cout « "[INFO] Sorting completed\n\n";
00163 }
00164
00165 void studentuTest() {
00166     Vector<Stud> studentai;
00167     Vector<int> pazymiai = { 5, 6, 7 };
00168
00169
00171     Stud studentas1("Jonas", "Jonaitis", pazymiai, 8);
00172     studentas1.calculateGalVidurkis();
00173     studentas1.calculateGalMediana();
00174     if (studentas1.getVar() == "Jonas" &&
00175         studentas1.getPav() == "Jonaitis" &&
00176         studentas1.getPazymys() == pazymiai) {
00177         std::cout « "[PASS] Constructor test passed\n\n";
00178     }
00179     else {
00180         std::cout « "[FAIL] Constructor test failed\n\n";
00181     }
00182
00184     Stud studentas7;
00185     if (studentas7.getVar().empty() &&
00186         studentas7.getPav().empty() &&
00187         studentas7.getPazymys().Empty()) {
00188         std::cout « "[PASS] Empty Constructor test passed\n\n";
00189     }
00190     else {
00191         std::cout « "[FAIL] Empty Constructor test failed\n\n";
00192     }
00193
00194     // Copy constructor testas
00195     Stud studentas2(studentas1);
00196     if (studentas1.getVar() == studentas2.getVar() &&
00197         studentas1.getPav() == studentas2.getPav() &&
00198         studentas1.getPazymys() == studentas2.getPazymys() &&
00199         studentas1.getVidurkis() == studentas2.getVidurkis() &&
00200         studentas1.getMediana() == studentas2.getMediana()) {
00201         std::cout « "[PASS] Copy constructor test passed\n\n";
00202     }
00203     else {
00204         std::cout « "[FAIL] Copy constructor test failed\n\n";
00205     }
00206
00207     // Copy assignment operator testas
00208     Stud studentas3;
00209     studentas3 = studentas1;
00210     if (studentas3.getVar() == studentas1.getVar() &&
00211         studentas3.getPav() == studentas1.getPav() &&
00212         studentas3.getPazymys() == studentas1.getPazymys() &&
00213         studentas3.getVidurkis() == studentas1.getVidurkis() &&
00214         studentas3.getMediana() == studentas1.getMediana()) {
00215         std::cout « "[PASS] Copy assignment operator test passed\n\n";
00216     }
00217     else {
00218         std::cout « "[FAIL] Copy assignment operator test failed\n\n";
00219     }
00220
00221     // Move constructor testas
00222     Stud studentas4(std::move(studentas1));
00223     if (studentas4.getVar() != studentas1.getVar() &&
00224         studentas4.getPav() != studentas1.getPav() &&
00225         studentas4.getPazymys() != studentas1.getPazymys()) {
00226         std::cout « "[PASS] Move constructor test passed\n\n";
00227     }
00228     else {
00229         std::cout « "[FAIL] Move constructor test failed\n\n";
00230     }
00231
00232     // Move assignment operator testas
```

```
00233     Stud studentas5;
00234     studentas5 = std::move(studentas4);
00235     if (studentas5.getVar() != studentas4.getVar() &&
00236         studentas5.getPav() != studentas4.getPav() &&
00237         studentas5.getPazymys() != studentas4.getPazymys()) {
00238         std::cout « "[PASS] Move assignment operator test passed\n\n";
00239     }
00240     else {
00241         std::cout « "[FAIL] Move assignment operator test failed\n\n";
00242     }
00243
00244     std::cout « "\n";
00245
00246     // Ivesties operatorius testas
00247     std::cout « "Input operator testas:\n";
00248     Stud studentas6;
00249     std::cin » studentas6;
00250
00251     std::cout « "\n";
00252
00253     // Isveties operatorius testas
00254     std::cout « "Ivestas studentas: \n" « studentas6;
00255
00256     std::cout « "\n\n";
00257 }
00258
00259 void vectorCompare() {
00260     auto start = hrClock::now();
00261     auto dursum = sec(0.0);
00262     Vector<int> vec1;
00263     int n = 10000;
00264     for (int i = 0; i < 5; i++) {
00265         auto start1 = hrClock::now();
00266         for (int j = 0; j < n; j++) {
00267             vec1.PushBack(j);
00268         }
00269         vec1.Clear();
00270         n *= 10;
00271         auto end1 = hrClock::now();
00272         sec duration1 = end1 - start1;
00273         dursum += duration1;
00274         cout «"Size " « n « " Vector PushBack duration: " « fixed « setprecision(8) «
      duration1.count() « " sec\n";
00275     }
00276
00277     auto end = hrClock::now();
00278     sec duration = end - start;
00279     std::cout « "Vector compare duration: " « fixed « setprecision(8) « duration.count() « " sec\n";
00280     std::cout « "Vector compare avg: " « fixed « setprecision(8) « dursum.count()/5 « " sec\n";
00281 }
00282
00283 void atmintiesPerskirstymas() {
00284     Vector<int> sk;
00285     int n = 100000000, j = 0;
00286     for (int i = 0; i < n; i++) {
00287         sk.PushBack(i);
00288         if (sk.Capacity() == sk.Size()) { j++; }
00289     }
00290     cout « "Atminties perskirstyta " « j « " kartu.\n";
00291     cout « "Atmintis perskirstyta ir studentai prideti.\n";
00292 }
```

6.32 VECTOR/test/studentas_tests.cpp File Reference

```
#include "../googletest-main/googletest/include/gtest/gtest.h"
#include "../Include/studentas.h"
#include "../Include/zmogus.h"
#include "../Include/functionsCallsVector.h"
#include "../Include/meinelib.h"
```

Functions

- TEST (StudentTest, ConstructorTest)

- TEST (StudentTest, EmptyConstructorTest)
- TEST (StudentTest, CopyConstructorTest)
- TEST (StudentTest, CopyAssignmentOperatorTest)
- TEST (StudentTest, MoveConstructorTest)
- TEST (StudentTest, MoveAssignmentOperatorTest)

6.32.1 Function Documentation

6.32.1.1 TEST() [1/6]

```
TEST (
            StudentTest ,
            ConstructorTest )
```

Definition at line 7 of file studentas_tests.cpp.

6.32.1.2 TEST() [2/6]

```
TEST (
            StudentTest ,
            CopyAssignmentOperatorTest )
```

Definition at line 35 of file studentas_tests.cpp.

6.32.1.3 TEST() [3/6]

```
TEST (
            StudentTest ,
            CopyConstructorTest )
```

Definition at line 22 of file studentas_tests.cpp.

6.32.1.4 TEST() [4/6]

```
TEST (
            StudentTest ,
            EmptyConstructorTest )
```

Definition at line 14 of file studentas_tests.cpp.

6.32.1.5 TEST() [5/6]

```
TEST (
            StudentTest ,
            MoveAssignmentOperatorTest )
```

Definition at line 58 of file studentas_tests.cpp.

6.32.1.6 TEST() [6/6]

```
TEST (
            StudentTest ,
            MoveConstructorTest )
```

Definition at line 49 of file studentas_tests.cpp.

6.33 studentas_tests.cpp

Go to the documentation of this file.

```
00001 #include "../googletest-main/googletest/include/gtest/gtest.h"
00002 #include "../Include/studentas.h"
00003 #include "../Include/zmogus.h"
00004 #include "../Include/functionsCallsVector.h"
00005 #include "../Include/meinelib.h"
00006
00007 TEST(StudentTest, ConstructorTest) {
00008     Stud student("TestVardas", "TestPavarde", { 7, 8, 9 }, 10);
00009     EXPECT_EQ(student.getVar(), "TestVardas");
00010     EXPECT_EQ(student.getPav(), "TestPavarde");
00011     EXPECT_EQ(student.getEgz(), 10);
00012 }
00013
00014 TEST(StudentTest, EmptyConstructorTest) {
00015     Stud student;
00016     EXPECT_EQ(student.getVar(), "");
00017     EXPECT_EQ(student.getPav(), "");
00018     EXPECT_TRUE(student.getPazymys().Empty());
00019     EXPECT_EQ(student.getEgz(), 0);
00020 }
00021
00022 TEST(StudentTest, CopyConstructorTest) {
00023     Stud original("OriginalVardas", "OriginalPavarde", { 7, 8, 9 }, 10);
00024     original.calculateGalMediana();
00025     original.calculateGalVidurkis();
00026     Stud copy(original);
00027     EXPECT_EQ(copy.getVar(), original.getVar());
00028     EXPECT_EQ(copy.getPav(), original.getPav());
00029     EXPECT_EQ(copy.getPazymys(), original.getPazymys());
00030     EXPECT_EQ(copy.getEgz(), original.getEgz());
00031     EXPECT_FLOAT_EQ(copy.getVidurkis(), original.getVidurkis());
00032     EXPECT_FLOAT_EQ(copy.getMediana(), original.getMediana());
00033 }
00034
00035 TEST(StudentTest, CopyAssignmentOperatorTest) {
00036     Stud original("OriginalVardas", "OriginalPavarde", { 7, 8, 9 }, 10);
00037     original.calculateGalMediana();
00038     original.calculateGalVidurkis();
00039     Stud copy;
00040     copy = original;
00041     EXPECT_EQ(copy.getVar(), original.getVar());
00042     EXPECT_EQ(copy.getPav(), original.getPav());
00043     EXPECT_EQ(copy.getPazymys(), original.getPazymys());
00044     EXPECT_EQ(copy.getEgz(), original.getEgz());
00045     EXPECT_FLOAT_EQ(copy.getVidurkis(), original.getVidurkis());
00046     EXPECT_FLOAT_EQ(copy.getMediana(), original.getMediana());
00047 }
00048
00049 TEST(StudentTest, MoveConstructorTest) {
00050     Stud original("OriginalVardas", "OriginalPavarde", { 7, 8, 9 }, 10);
00051     Stud moved(std::move(original));
00052     EXPECT_EQ(moved.getVar(), "OriginalVardas");
00053     EXPECT_EQ(moved.getPav(), "OriginalPavarde");
00054     EXPECT_EQ(moved.getPazymys(), Vector<int>({ 7, 8, 9 }));
00055     EXPECT_EQ(moved.getEgz(), 10);
00056 }
00057
00058 TEST(StudentTest, MoveAssignmentOperatorTest) {
00059     Stud original("OriginalVardas", "OriginalPavarde", { 7, 8, 9 }, 10);
00060     Stud moved;
00061     moved = std::move(original);
00062     EXPECT_EQ(moved.getVar(), "OriginalVardas");
00063     EXPECT_EQ(moved.getPav(), "OriginalPavarde");
00064     EXPECT_EQ(moved.getPazymys(), Vector<int>({ 7, 8, 9 }));
00065     EXPECT_EQ(moved.getEgz(), 10);
00066 }
```

6.34 VECTOR/test/vector_tests.cpp File Reference

```
#include "../googletest-main/googletest/include/gtest/gtest.h"
#include "../Include/functionsCallsVector.h"
#include "../Include/meinelib.h"
```

Functions

- TEST (VectorTest, DefaultConstructor)
- TEST (VectorTest, PushBackAndAt)
- TEST (VectorTest, AtOutOfRangeThrows)
- TEST (VectorTest, BracketOperator)
- TEST (VectorTest, FrontBack)
- TEST (VectorTest, ReserveIncreasesCapacity)
- TEST (VectorTest, ShrinkToFitReducesCapacity)
- TEST (VectorTest, ClearEmptiesVector)
- TEST (VectorTest, InsertAtPosition)
- TEST (VectorTest, InsertRangeWorks)
- TEST (VectorTest, EraseElement)
- TEST (VectorTest, PopBack)
- TEST (VectorTest, ResizeUpAndDown)
- TEST (VectorTest, SwapWorks)
- TEST (VectorTest, AssignWithCountValue)
- TEST (VectorTest, AssignWithIterators)
- TEST (VectorTest, Emplace)
- TEST (VectorTest, EmplaceBack)
- TEST (VectorTest, AppendRange)
- TEST (VectorTest, EqualityOperators)
- TEST (VectorTest, CopyConstructor)
- TEST (VectorTest, AssignmentOperator)
- TEST (VectorTest, BeginEnd)
- TEST (VectorTest, RBeginREnd)

6.34.1 Function Documentation

6.34.1.1 TEST() [1/24]

```
TEST (
            VectorTest ,
            AppendRange )
```

Definition at line 142 of file vector_tests.cpp.

6.34.1.2 TEST() [2/24]

```
TEST (
            VectorTest ,
            AssignmentOperator )
```

Definition at line 167 of file vector_tests.cpp.

6.34.1.3 TEST() [3/24]

```
TEST (
            VectorTest ,
            AssignWithCountValue )
```

Definition at line 114 of file vector_tests.cpp.

6.34.1.4 TEST() [4/24]

```
TEST (
            VectorTest ,
            AssignWithIterators )
```

Definition at line 122 of file vector_tests.cpp.

6.34.1.5 TEST() [5/24]

```
TEST (
            VectorTest ,
            AtOutOfRangeThrows )
```

Definition at line 21 of file vector_tests.cpp.

6.34.1.6 TEST() [6/24]

```
TEST (
            VectorTest ,
            BeginEnd )
```

Definition at line 175 of file vector_tests.cpp.

6.34.1.7 TEST() [7/24]

```
TEST (
            VectorTest ,
            BracketOperator )
```

Definition at line 26 of file vector_tests.cpp.

6.34.1.8 TEST() [8/24]

```
TEST (
            VectorTest ,
            ClearEmptiesVector )
```

Definition at line 56 of file vector_tests.cpp.

6.34.1.9 TEST() [9/24]

```
TEST (
            VectorTest ,
            CopyConstructor )
```

Definition at line 160 of file vector_tests.cpp.

6.34.1.10 TEST() [10/24]

```
TEST (
            VectorTest ,
            DefaultConstructor )
```

Definition at line 7 of file vector_tests.cpp.

6.34.1.11 TEST() [11/24]

```
TEST (
            VectorTest ,
            Emplace )
```

Definition at line 130 of file vector_tests.cpp.

6.34.1.12 TEST() [12/24]

```
TEST (
            VectorTest ,
            EmplaceBack )
```

Definition at line 136 of file vector_tests.cpp.

6.34.1.13 TEST() [13/24]

```
TEST (
            VectorTest ,
            EqualityOperators )
```

Definition at line 150 of file vector_tests.cpp.

6.34.1.14 TEST() [14/24]

```
TEST (
            VectorTest ,
            EraseElement )
```

Definition at line 80 of file vector_tests.cpp.

6.34.1.15 TEST() [15/24]

```
TEST (
            VectorTest ,
            FrontBack )
```

Definition at line 32 of file vector_tests.cpp.

6.34.1.16 TEST() [16/24]

```
TEST (
            VectorTest ,
            InsertAtPosition )
```

Definition at line 63 of file vector_tests.cpp.

6.34.1.17 TEST() [17/24]

```
TEST (
            VectorTest ,
            InsertRangeWorks )
```

Definition at line 70 of file vector_tests.cpp.

6.34.1.18 TEST() [18/24]

```
TEST (
            VectorTest ,
            PopBack )
```

Definition at line 87 of file vector_tests.cpp.

6.34.1.19 TEST() [19/24]

```
TEST (
            VectorTest ,
            PushBackAndAt )
```

Definition at line 14 of file vector_tests.cpp.

6.34.1.20 TEST() [20/24]

```
TEST (
            VectorTest ,
            RBeginREnd )
```

Definition at line 185 of file vector_tests.cpp.

6.34.1.21 TEST() [21/24]

```
TEST (
            VectorTest ,
            ReserveIncreasesCapacity )
```

Definition at line 40 of file vector_tests.cpp.

6.34.1.22 TEST() [22/24]

```
TEST (
            VectorTest ,
            ResizeUpAndDown )
```

Definition at line 94 of file vector_tests.cpp.

6.34.1.23 TEST() [23/24]

```
TEST (
            VectorTest ,
            ShrinkToFitReducesCapacity )
```

Definition at line 47 of file vector_tests.cpp.

6.34.1.24 TEST() [24/24]

```
TEST (
            VectorTest ,
            SwapWorks )
```

Definition at line 104 of file vector_tests.cpp.

6.35 vector_tests.cpp

Go to the documentation of this file.

```
00001 #include "../googletest-main/googletest/include/gtest/gtest.h"
00002 #include "../Include/functionsCallsVector.h"
00003 #include "../Include/meinelib.h"
00004
00005
00006
00007 TEST(VectorTest, DefaultConstructor) {
00008     Vector<int> vec;
00009     EXPECT_TRUE(vec.Empty());
00010     EXPECT_EQ(vec.Size(), 0);
00011     EXPECT_GE(vec.Capacity(), 5);
00012 }
00013
00014 TEST(VectorTest, PushBackAndAt) {
00015     Vector<int> vec;
00016     vec.PushBack(10);
00017     EXPECT_EQ(vec.Size(), 1);
00018     EXPECT_EQ(vec.At(0), 10);
00019 }
00020
00021 TEST(VectorTest, AtOutOfRangeThrows) {
00022     Vector<int> vec;
```

```
00023     EXPECT_THROW(vec.At(1), std::exception);
00024 }
00025
00026 TEST(VectorTest, BracketOperator) {
00027     Vector<int> vec;
00028     vec.PushBack(5);
00029     EXPECT_EQ(vec[0], 5);
00030 }
00031
00032 TEST(VectorTest, FrontBack) {
00033     Vector<int> vec;
00034     vec.PushBack(1);
00035     vec.PushBack(2);
00036     EXPECT_EQ(vec.Front(), 1);
00037     EXPECT_EQ(vec.Back(), 2);
00038 }
00039
00040 TEST(VectorTest, ReserveIncreasesCapacity) {
00041     Vector<int> vec;
00042     size_t oldCap = vec.Capacity();
00043     vec.Reserve(oldCap + 10);
00044     EXPECT_GT(vec.Capacity(), oldCap);
00045 }
00046
00047 TEST(VectorTest, ShrinkToFitReducesCapacity) {
00048     Vector<int> vec;
00049     vec.PushBack(1);
00050     vec.PushBack(2);
00051     vec.Reserve(100);
00052     vec.ShrinkToFit();
00053     EXPECT_EQ(vec.Capacity(), vec.Size());
00054 }
00055
00056 TEST(VectorTest, ClearEmptiesVector) {
00057     Vector<int> vec;
00058     vec.PushBack(1);
00059     vec.Clear();
00060     EXPECT_EQ(vec.Size(), 0);
00061 }
00062
00063 TEST(VectorTest, InsertAtPosition) {
00064     Vector<int> vec;
00065     vec.PushBack(1);
00066     vec.Insert(1, 2);
00067     EXPECT_EQ(vec[1], 2);
00068 }
00069
00070 TEST(VectorTest, InsertRangeWorks) {
00071     Vector<int> vec;
00072     std::vector<int> data = {3, 4, 5};
00073     vec.InsertRange(0, data.begin(), data.end());
00074     EXPECT_EQ(vec.Size(), 3);
00075     EXPECT_EQ(vec[0], 3);
00076     EXPECT_EQ(vec[1], 4);
00077     EXPECT_EQ(vec[2], 5);
00078 }
00079
00080 TEST(VectorTest, EraseElement) {
00081     Vector<int> vec;
00082     vec.PushBack(10);
00083     vec.Erase(0);
00084     EXPECT_EQ(vec.Size(), 0);
00085 }
00086
00087 TEST(VectorTest, PopBack) {
00088     Vector<int> vec;
00089     vec.PushBack(1);
00090     vec.PopBack();
00091     EXPECT_TRUE(vec.Empty());
00092 }
00093
00094 TEST(VectorTest, ResizeUpAndDown) {
00095     Vector<int> vec;
00096     vec.Resize(5, 7);
00097     EXPECT_EQ(vec.Size(), 5);
00098     for (int i = 0; i < 5; ++i)
00099         EXPECT_EQ(vec[i], 7);
00100     vec.Resize(2);
00101     EXPECT_EQ(vec.Size(), 2);
00102 }
00103
00104 TEST(VectorTest, SwapWorks) {
00105     Vector<int> vec;
00106     Vector<int> other;
00107     other.PushBack(100);
00108     vec.PushBack(5);
00109     vec.Swap(other);
```

```
00110     EXPECT_EQ(vec[0], 100);
00111     EXPECT_EQ(other[0], 5);
00112 }
00113
00114 TEST(VectorTest, AssignWithCountValue) {
00115     Vector<int> vec;
00116     vec.Assign(3, 9);
00117     EXPECT_EQ(vec.Size(), 3);
00118     for (int i = 0; i < 3; ++i)
00119         EXPECT_EQ(vec[i], 9);
00120 }
00121
00122 TEST(VectorTest, AssignWithIterators) {
00123     Vector<int> vec;
00124     int data[] = { 1, 2, 3 };
00125     vec.Assign(data, data + 3);
00126     EXPECT_EQ(vec.Size(), 3);
00127     EXPECT_EQ(vec[1], 2);
00128 }
00129
00130 TEST(VectorTest, Emplace) {
00131     Vector<int> vec;
00132     vec.Emplace(0, 42);
00133     EXPECT_EQ(vec[0], 42);
00134 }
00135
00136 TEST(VectorTest, EmplaceBack) {
00137     Vector<int> vec;
00138     vec.EmplaceBack(55);
00139     EXPECT_EQ(vec.Back(), 55);
00140 }
00141
00142 TEST(VectorTest, AppendRange) {
00143     Vector<int> vec;
00144     std::vector<int> vals = {5, 6};
00145     vec.AppendRange(vals.begin(), vals.end());
00146     EXPECT_EQ(vec[0], 5);
00147     EXPECT_EQ(vec[1], 6);
00148 }
00149
00150 TEST(VectorTest, EqualityOperators) {
00151     Vector<int> vec;
00152     Vector<int> other;
00153     vec.PushBack(1);
00154     other.PushBack(1);
00155     EXPECT_TRUE(vec == other);
00156     other.PushBack(2);
00157     EXPECT_TRUE(vec != other);
00158 }
00159
00160 TEST(VectorTest, CopyConstructor) {
00161     Vector<int> vec;
00162     vec.PushBack(7);
00163     Vector<int> copy(vec);
00164     EXPECT_EQ(copy[0], 7);
00165 }
00166
00167 TEST(VectorTest, AssignmentOperator) {
00168     Vector<int> vec;
00169     Vector<int> other;
00170     other.PushBack(3);
00171     vec = other;
00172     EXPECT_EQ(vec[0], 3);
00173 }
00174
00175 TEST(VectorTest, BeginEnd) {
00176     Vector<int> vec;
00177     vec.PushBack(1);
00178     vec.PushBack(2);
00179     auto it = vec.begin();
00180     EXPECT_EQ(*it, 1);
00181     ++it;
00182     EXPECT_EQ(*it, 2);
00183 }
00184
00185 TEST(VectorTest, RBeginREnd) {
00186     Vector<int> vec;
00187     vec.PushBack(1);
00188     vec.PushBack(2);
00189     auto rit = vec.rbegin();
00190     EXPECT_EQ(*rit, 2);
00191     --rit;
00192     EXPECT_EQ(*rit, 1);
00193 }
```

Index