VU OP Darbas

v2.0

Generated by Doxygen 1.12.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

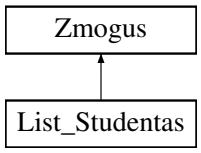
Class Documentation

4.1 List_Studentas Class Reference

Klasė, paveldėta iš Zmogus, skirta studentų duomenims valdyti naudojant `std::list`.

`#include <List_funkcijos.h>`

Inheritance diagram for List_Studentas:



Public Member Functions

- List_Studentas ()

  Numatytasis konstruktorius.
- List_Studentas (const std::string &vardas, const std::string &pavarde, const std::list< int > &pazymiai, int egzaminoPazymys)

  Konstruktorius su parametrais.
- ~List_Studentas ()

  Destruktorius.
- List_Studentas (const List_Studentas &other)

  Kopijavimo konstruktorius.
- List_Studentas & operator= (const List_Studentas &other)

  Kopijavimo priskyrimo operatorius.
- std::list< int > getPazymiai () const

  Grąžina pažymių sąrašą.
- int getEgzaminoPazymys () const

  Grąžina egzamino pažymį.
- float getVidurkis () const

  Grąžina pažymių vidurkį.
- float getMediana () const

  Grąžina pažymių medianą.

- float getGalutinisVidurkis () const

  Grąžina galutinį rezultatą pagal vidurkį.
- float getGalutineMediana () const

  Grąžina galutinį rezultatą pagal medianą.
- void List_setPazymiai (const std::list< int > &pazymiai)

  Nustato pažymių sąrašą.
- void setEgzaminoPazymys (int egzaminoPazymys)

  Nustato egzamino pažymį.
- void setVidurkis (float vidurkis)

  Nustato pažymių vidurkį.
- void setMediana (float mediana)

  Nustato pažymių medianą.
- void setGalutinisVidurkis (float galutinisVidurkis)

  Nustato galutinį rezultatą pagal vidurkį.
- void setGalutineMediana (float galutineMediana)

  Nustato galutinį rezultatą pagal medianą.
- void printInfo () const override

  Spausdina informaciją apie studentą.
- void pridetiPazymi (int pazymys)

  Prideda pažymį į sąrašą.
- void List_skaiciuotiRezultatus ()

  Apskaičiuoja rezultatus, tokius kaip vidurkis ir mediana.

Public Member Functions inherited from Zmogus

- Zmogus ()

  Numatytasis konstruktorius.
- Zmogus (const std::string &vardas, const std::string &pavarde)

  Konstruktorius su parametrais.
- virtual ~Zmogus ()

  Virtualus destruktorius.
- Zmogus (const Zmogus &other)

  Kopijavimo konstruktorius.
- Zmogus & operator= (const Zmogus &other)

  Kopijavimo priskyrimo operatorius.
- std::string getVardas () const

  Grąžina asmens vardą.
- std::string getPavarde () const

  Grąžina asmens pavardę.
- void setVardas (const std::string &vardas)

  Nustato asmens vardą.
- void setPavarde (const std::string &pavarde)

  Nustato asmens pavardę.

Friends

- std::ostream & operator<< (std::ostream &os, const List_Studentas &studentas)

  Perkrautas operatorius << išvedimui.
- std::istream & operator>> (std::istream &is, List_Studentas &studentas)

  Perkrautas operatorius >> įvedimui.

Additional Inherited Members

Protected Attributes inherited from Zmogus

- std::string vardas
  Asmens vardas.
- std::string pavarde
  Asmens pavardė.

4.1.1 Detailed Description

Klasė, paveldėta iš Zmogus, skirta studentų duomenims valdyti naudojant `std::list`.

Definition at line 16 of file List_funkcijos.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 List_Studentas() [1/3]

```
List_Studentas::List_Studentas ()
```

Numatytasis konstruktorius.

Definition at line 5 of file List_studentas.cpp.

4.1.2.2 List_Studentas() [2/3]

```
List_Studentas::List_Studentas (
            const std::string & vardas,
            const std::string & pavarde,
            const std::list< int > & pazymiai,
            int egzaminoPazymys)
```

Konstruktorius su parametrais.

Parameters

vardas	Studentų vardas.
pavarde	Studentų pavardė.
pazymiai	Pažymių sąrašas.
egzaminoPazymys	Egzamino pažymys.

Definition at line 10 of file List_studentas.cpp.

4.1.2.3 ~List_Studentas()

```
List_Studentas::~List_Studentas ()
```

Destruktorius.

Definition at line 19 of file List_studentas.cpp.

4.1.2.4 List_Studentas() [3/3]

```
List_Studentas::List_Studentas (
            const List_Studentas & other)
```

Kopijavimo konstruktorius.

Parameters

other	Kitas List_Studentas objektas, iš kurio kopijuojama.

Definition at line 25 of file List_studentas.cpp.

4.1.3 Member Function Documentation

4.1.3.1 getEgzaminoPazymys()

```
int List_Studentas::getEgzaminoPazymys () const
```

Grąžina egzamino pažymį.

Returns

Egzamino pažymys.

Definition at line 59 of file List_studentas.cpp.

4.1.3.2 getGalutineMediana()

```
float List_Studentas::getGalutineMediana () const
```

Grąžina galutinį rezultatą pagal medianą.

Returns

Galutinis rezultatas pagal medianą.

Definition at line 63 of file List_studentas.cpp.

4.1.3.3 getGalutinisVidurkis()

```
float List_Studentas::getGalutinisVidurkis () const
```

Grąžina galutinį rezultatą pagal vidurkį.

Returns

Galutinis rezultatas pagal vidurkį.

Definition at line 62 of file List_studentas.cpp.

4.1.3.4 getMediana()

```
float List_Studentas::getMediana () const
```

Grąžina pažymių medianą.

Returns

Mediana.

Definition at line 61 of file List_studentas.cpp.

4.1.3.5 getPazymiai()

```
std::list< int > List_Studentas::getPazymiai () const
```

Grąžina pažymių sąrašą.

Returns

Pažymių sąrašas.

Definition at line 58 of file List_studentas.cpp.

4.1.3.6 getVidurkis()

```
float List_Studentas::getVidurkis () const
```

Grąžina pažymių vidurkį.

Returns

Vidurkis.

Definition at line 60 of file List_studentas.cpp.

4.1.3.7 List_setPazymiai()

```
void List_Studentas::List_setPazymiai (
            const std::list< int > & pazymiai)
```

Nustato pažymių sąrašą.

Parameters

pazymiai	Sąrašas, kuris bus nustatytas.

Definition at line 66 of file List_studentas.cpp.

4.1.3.8 List_skaiciuotiRezultatus()

```
void List_Studentas::List_skaiciuotiRezultatus ()
```

Apskaičiuoja rezultatus, tokius kaip vidurkis ir mediana.

Definition at line 80 of file List_studentas.cpp.

4.1.3.9 operator=()

```
List_Studentas & List_Studentas::operator= (
            const List_Studentas & other)
```

Kopijavimo priskyrimo operatorius.

Parameters

other	Kitas List_Studentas objektas, kuris priskiriamas.

Returns

Nuoroda į esamą objektą.

Definition at line 34 of file List_studentas.cpp.

4.1.3.10 pridetiPazymi()

```
void List_Studentas::pridetiPazymi (
            int pazymys)
```

Prideda pažymį į sąrašą.

Parameters

pazymys	Pridedamas pažymys.

Definition at line 74 of file List_studentas.cpp.

4.1.3.11 printInfo()

```
void List_Studentas::printInfo () const  [override], [virtual]
```

Spausdina informaciją apie studentą.

Implements Zmogus.

Definition at line 99 of file List_studentas.cpp.

4.1.3.12 setEgzaminoPazymys()

```
void List_Studentas::setEgzaminoPazymys (
            int egzaminoPazymys)
```

Nustato egzamino pažymį.

Parameters

egzaminoPazymys	Pažymys, kuris bus nustatytas.

Definition at line 69 of file List_studentas.cpp.

4.1.3.13 setGalutineMediana()

```
void List_Studentas::setGalutineMediana (
            float galutineMediana)
```

Nustato galutinį rezultatą pagal medianą.

Parameters

galutineMediana	Galutinis rezultatas, kuris bus nustatytas.

Definition at line 71 of file List_studentas.cpp.

4.1.3.14 setGalutinisVidurkis()

```
void List_Studentas::setGalutinisVidurkis (
            float galutinisVidurkis)
```

Nustato galutinį rezultatą pagal vidurkį.

Parameters

galutinisVidurkis	Galutinis rezultatas, kuris bus nustatytas.

Definition at line 70 of file List_studentas.cpp.

4.1.3.15 setMediana()

```
void List_Studentas::setMediana (
            float mediana)
```

Nustato pažymių medianą.

Parameters

mediana	Mediana, kuri bus nustatyta.

Definition at line 68 of file List_studentas.cpp.

4.1.3.16 setVidurkis()

```
void List_Studentas::setVidurkis (
            float vidurkis)
```

Nustato pažymių vidurkį.

Parameters

vidurkis	Vidurkis, kuris bus nustatytas.

Definition at line 67 of file List_studentas.cpp.

4.1.4 Friends And Related Symbol Documentation

4.1.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const List_Studentas & studentas)  [friend]
```

Perkrautas operatorius << išvedimui.

Parameters

os	Išvesties srautas.
studentas	Studentas, kurio duomenys spausdinami.

Returns

Išvesties srautas.

4.1.4.2 operator>>

```
std::istream & operator>> (
            std::istream & is,
            List_Studentas & studentas)  [friend]
```

Perkrautas operatorius >> įvedimui.

Parameters

is	Įvesties srautas.
studentas	Studentas, kurio duomenys skaitomi.

Returns

Įvesties srautas.

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

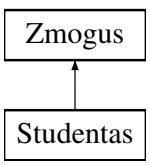
- include/List_funkcijos.h
- src/List_studentas.cpp

4.2 Studentas Class Reference

Išvestinė klasė, reprezentuojanti studentą su pažymiais ir rezultatais.

```
#include <Vec_funkcijos.h>
```

Inheritance diagram for Studentas:



Public Member Functions

- Studentas ()

  Numatytasis konstruktorius.
- Studentas (const std::string &vardas, const std::string &pavarde, const std::vector< int > &pazymiai, int egzaminoPazymys)

  Konstruktorius su parametrais.
- ~Studentas ()

  Destruktorius.
- Studentas (const Studentas &other)

  Kopijavimo konstruktorius.
- Studentas & operator= (const Studentas &other)

  Kopijavimo priskyrimo operatorius.
- std::vector< int > getPazymiai () const
- int getEgzaminoPazymys () const
- float getVidurkis () const
- float getMediana () const
- float getGalutinisVidurkis () const
- float getGalutineMediana () const
- void setPazymiai (const std::vector< int > &pazymiai)
- void setEgzaminoPazymys (int egzaminoPazymys)
- void setVidurkis (float vidurkis)
- void setMediana (float mediana)
- void setGalutineMediana (float galutineMediana)
- void setGalutinisVidurkis (float galutinisVidurkis)
- void skaiciuotiRezultatus ()

  Apskaičiuoja tokius rezultatus kaip vidurkis ir mediana.
- void pridetiPazymi (int pazymys)

  Prideda pažymį į sąrašą.
- void printInfo () const override

  Spausdina informaciją apie studentą.

Public Member Functions inherited from Zmogus

- Zmogus ()

  Numatytasis konstruktorius.
- Zmogus (const std::string &vardas, const std::string &pavarde)

  Konstruktorius su parametrais.
- virtual ~Zmogus ()

  Virtualus destruktorius.
- Zmogus (const Zmogus &other)

  Kopijavimo konstruktorius.
- Zmogus & operator= (const Zmogus &other)

  Kopijavimo priskyrimo operatorius.
- std::string getVardas () const

  Grąžina asmens vardą.
- std::string getPavarde () const

  Grąžina asmens pavardę.
- void setVardas (const std::string &vardas)

  Nustato asmens vardą.
- void setPavarde (const std::string &pavarde)

  Nustato asmens pavardę.

Friends

- std::ostream & operator<< (std::ostream &os, const Studentas &studentas)
- std::istream & operator>> (std::istream &is, Studentas &studentas)

Additional Inherited Members

Protected Attributes inherited from Zmogus

- std::string vardas

  Asmens vardas.
- std::string pavarde

  Asmens pavardė.

4.2.1 Detailed Description

Išvestinė klasė, reprezentuojanti studentą su pažymiais ir rezultatais.

Definition at line 87 of file Vec_funkcijos.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Studentas() [1/3]

```
Studentas::Studentas ()
```

Numatytasis konstruktorius.

Definition at line 5 of file studentas.cpp.

4.2.2.2 Studentas() [2/3]

```
Studentas::Studentas (
            const std::string & vardas,
            const std::string & pavarde,
            const std::vector< int > & pazymiai,
            int egzaminoPazymys)
```

Konstruktorius su parametrais.

Parameters

vardas	Studentų vardas.
pavarde	Studentų pavardė.
pazymiai	Pažymių sąrašas.
egzaminoPazymys	Egzamino pažymys.

Definition at line 8 of file studentas.cpp.

4.2.2.3 ~Studentas()

```
Studentas::~Studentas ()
```

Destruktorius.

Definition at line 15 of file studentas.cpp.

4.2.2.4 Studentas() [3/3]

```
Studentas::Studentas (
            const Studentas & other)
```

Kopijavimo konstruktorius.

Parameters

other	Kitas Studentas objektas, iš kurio kopijuojama.

Definition at line 33 of file studentas.cpp.

4.2.3 Member Function Documentation

4.2.3.1 getEgzaminoPazymys()

```
int Studentas::getEgzaminoPazymys () const
```

Definition at line 59 of file studentas.cpp.

4.2.3.2 getGalutineMediana()

```
float Studentas::getGalutineMediana () const
```

Definition at line 63 of file studentas.cpp.

4.2.3.3 getGalutinisVidurkis()

```
float Studentas::getGalutinisVidurkis () const
```

Definition at line 62 of file studentas.cpp.

4.2.3.4 getMediana()

```
float Studentas::getMediana () const
```

Definition at line 61 of file studentas.cpp.

4.2.3.5 getPazymiai()

```
std::vector< int > Studentas::getPazymiai () const
```

Definition at line 58 of file studentas.cpp.

4.2.3.6 getVidurkis()

```
float Studentas::getVidurkis () const
```

Definition at line 60 of file studentas.cpp.

4.2.3.7 operator=()

```
Studentas & Studentas::operator= (
            const Studentas & other)
```

Kopijavimo priskyrimo operatorius.

Parameters

other	Kitas Studentas objektas, kuris priskiriamas.

Returns

Nuoroda į dabartinį objektą.

Definition at line 43 of file studentas.cpp.

4.2.3.8 pridetiPazymi()

```
void Studentas::pridetiPazymi (
            int pazymys)
```

Prideda pažymį į sąrašą.

Parameters

pazymys	Pridedamas pažymys.

Definition at line 87 of file studentas.cpp.

4.2.3.9 printInfo()

```
void Studentas::printInfo () const  [override], [virtual]
```

Spausdina informaciją apie studentą.

Implements Zmogus.

Definition at line 375 of file studentas.cpp.

4.2.3.10 setEgzaminoPazymys()

```
void Studentas::setEgzaminoPazymys (
            int egzaminoPazymys)
```

Definition at line 69 of file studentas.cpp.

4.2.3.11 setGalutineMediana()

```
void Studentas::setGalutineMediana (
            float galutineMediana)
```

Definition at line 71 of file studentas.cpp.

4.2.3.12 setGalutinisVidurkis()

```
void Studentas::setGalutinisVidurkis (
            float galutinisVidurkis)
```

Definition at line 70 of file studentas.cpp.

4.2.3.13 setMediana()

```
void Studentas::setMediana (
            float mediana)
```

Definition at line 68 of file studentas.cpp.

4.2.3.14 setPazymiai()

```
void Studentas::setPazymiai (
            const std::vector< int > & pazymiai)
```

Definition at line 66 of file studentas.cpp.

4.2.3.15 setVidurkis()

```
void Studentas::setVidurkis (
            float vidurkis)
```

Definition at line 67 of file studentas.cpp.

4.2.3.16 skaiciuotiRezultatus()

```
void Studentas::skaiciuotiRezultatus ()
```

Apskaičiuoja tokius rezultatus kaip vidurkis ir mediana.

Definition at line 73 of file studentas.cpp.

4.2.4 Friends And Related Symbol Documentation

4.2.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Studentas & studentas)  [friend]
```

Definition at line 92 of file studentas.cpp.

4.2.4.2 operator>>

```
std::istream & operator>> (
            std::istream & is,
            Studentas & studentas)  [friend]
```

Definition at line 133 of file studentas.cpp.

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

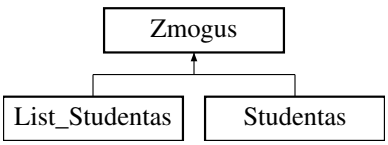
- include/Vec_funkcijos.h
- src/studentas.cpp

4.3 Zmogus Class Reference

Abstrakti bazinė klasė, reprezentuojanti asmenį su vardu ir pavarde.

```
#include <Vec_funkcijos.h>
```

Inheritance diagram for Zmogus:



Public Member Functions

- Zmogus ()
  Numatytasis konstruktorius.
- Zmogus (const std::string &vardas, const std::string &pavarde)
  Konstruktorius su parametrais.
- virtual ~Zmogus ()
  Virtualus destruktorius.
- Zmogus (const Zmogus &other)
  Kopijavimo konstruktorius.
- Zmogus & operator= (const Zmogus &other)
  Kopijavimo priskyrimo operatorius.

- virtual void printInfo () const =0

  Grynas virtualus metodas informacijai apie asmenį atspausdinti.
- std::string getVardas () const

  Grąžina asmens vardą.
- std::string getPavarde () const

  Grąžina asmens pavardę.
- void setVardas (const std::string &vardas)

  Nustato asmens vardą.
- void setPavarde (const std::string &pavarde)

  Nustato asmens pavardę.

Protected Attributes

- std::string vardas

  Asmens vardas.
- std::string pavarde

  Asmens pavardė.

4.3.1 Detailed Description

Abstrakti bazinė klasė, reprezentuojanti asmenį su vardu ir pavarde.

Ši klasė naudojama kaip bazinė klasė išvestinėms klasėms, tokioms kaip Studentas.

Definition at line 17 of file Vec_funkcijos.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Zmogus() [1/3]

```
Zmogus::Zmogus ()
```

Numatytasis konstruktorius.

Definition at line 5 of file human.cpp.

4.3.2.2 Zmogus() [2/3]

```
Zmogus::Zmogus (
            const std::string & vardas,
            const std::string & pavarde)
```

Konstruktorius su parametrais.

Parameters

vardas	Asmens vardas.
pavarde	Asmens pavardė.

Definition at line 8 of file human.cpp.

4.3.2.3 ~Zmogus()

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

Virtualus destruktorius.

Definition at line 12 of file human.cpp.

4.3.2.4 Zmogus() [3/3]

```
Zmogus::Zmogus (
            const Zmogus & other)
```

Kopijavimo konstruktorius.

Parameters

other	Kitas Zmogus objektas, iš kurio kopijuojama.

Definition at line 15 of file human.cpp.

4.3.3 Member Function Documentation

4.3.3.1 getPavarde()

```
std::string Zmogus::getPavarde () const
```

Grąžina asmens pavardę.

Returns

Asmens pavardė.

Definition at line 28 of file human.cpp.

4.3.3.2 getVardas()

```
std::string Zmogus::getVardas () const
```

Grąžina asmens vardą.

Returns

Asmens vardas.

Definition at line 27 of file human.cpp.

4.3.3.3 operator=()

```
Zmogus & Zmogus::operator= (
            const Zmogus & other)
```

Kopijavimo priskyrimo operatorius.

Parameters

other	Kitas Zmogus objektas, kuris priskiriamas.

Returns

Nuoroda į dabartinį objektą.

Definition at line 18 of file human.cpp.

4.3.3.4 printInfo()

```
virtual void Zmogus::printInfo () const  [pure virtual]
```

Grynas virtualus metodas informacijai apie asmenį atspausdinti.

Implemented in List_Studentas, and Studentas.

4.3.3.5 setPavarde()

```
void Zmogus::setPavarde (
            const std::string & pavarde)
```

Nustato asmens pavardę.

Parameters

pavarde	Naujasis pavardė.

Definition at line 30 of file human.cpp.

4.3.3.6 setVardas()

```
void Zmogus::setVardas (
            const std::string & vardas)
```

Nustato asmens vardą.

Parameters

vardas	Naujasis vardas.

Definition at line 29 of file human.cpp.

4.3.4 Member Data Documentation

4.3.4.1 pavarde

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

Asmens pavardė.

Definition at line 20 of file Vec_funkcijos.h.

4.3.4.2 vardas

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

Asmens vardas.

Definition at line 19 of file Vec_funkcijos.h.

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

- include/Vec_funkcijos.h
- include/human.cpp
- src/human.cpp

Chapter 5

File Documentation

5.1 include/human.cpp File Reference

```
#include "Vec_funkcijos.h"
#include <iostream>
```

5.2 human.cpp

Go to the documentation of this file.

```
00001 #include "Vec_funkcijos.h"
00002 #include <iostream>
00003
00004 // Default Constructor
00005 Zmogus::Zmogus() : vardas(""), pavarde("") {}
00006
00007 // Parameterized Constructor
00008 Zmogus::Zmogus(const std::string &vardas, const std::string &pavarde)
00009     : vardas(vardas), pavarde(pavarde) {}
00010
00011 // Destructor
00012 Zmogus::~Zmogus() {}
00013
00014 // Copy Constructor
00015 Zmogus::Zmogus(const Zmogus &other) : vardas(other.vardas), pavarde(other.pavarde) {}
00016
00017 // Copy Assignment Operator
00018 Zmogus &Zmogus::operator=(const Zmogus &other) {
00019     if (this != &other) {
00020         vardas = other.vardas;
00021         pavarde = other.pavarde;
00022     }
00023     return *this;
00024 }
00025
00026 // Getters and Setters
00027 std::string Zmogus::getVardas() const { return vardas; }
00028 std::string Zmogus::getPavarde() const { return pavarde; }
00029 void Zmogus::setVardas(const std::string &vardas) { this->vardas = vardas; }
00030 void Zmogus::setPavarde(const std::string &pavarde) { this->pavarde = pavarde; }
```

5.3 src/human.cpp File Reference

```
#include "Vec_funkcijos.h"
#include <iostream>
```

5.4 human.cpp

Go to the documentation of this file.

```
00001 #include "Vec_funkcijos.h"
00002 #include <iostream>
00003
00004 // Default Constructor
00005 Zmogus::Zmogus() : vardas(""), pavarde("") {}
00006
00007 // Parameterized Constructor
00008 Zmogus::Zmogus(const std::string &vardas, const std::string &pavarde)
00009     : vardas(vardas), pavarde(pavarde) {}
00010
00011 // Destructor
00012 Zmogus::~Zmogus() {}
00013
00014 // Copy Constructor
00015 Zmogus::Zmogus(const Zmogus &other) : vardas(other.vardas), pavarde(other.pavarde) {}
00016
00017 // Copy Assignment Operator
00018 Zmogus &Zmogus::operator=(const Zmogus &other) {
00019     if (this != &other) {
00020         vardas = other.vardas;
00021         pavarde = other.pavarde;
00022     }
00023     return *this;
00024 }
00025
00026 // Getters and Setters
00027 std::string Zmogus::getVardas() const { return vardas; }
00028 std::string Zmogus::getPavarde() const { return pavarde; }
00029 void Zmogus::setVardas(const std::string &vardas) { this->vardas = vardas; }
00030 void Zmogus::setPavarde(const std::string &pavarde) { this->pavarde = pavarde; }
```

5.5 include/List_Biblioteka.h File Reference

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <string>
#include <fstream>
#include <sstream>
#include <stdexcept>
#include <limits>
#include <algorithm>
#include <chrono>
#include <numeric>
#include <random>
#include <cstring>
#include <cctype>
#include <locale>
#include <list>
```

5.6 List_Biblioteka.h

Go to the documentation of this file.

```
00001 #ifndef LIST_BIBLIOTEKA_H_INCLUDED
00002 #define LIST_BIBLIOTEKA_H_INCLUDED
00003
00004 #include <iostream>
00005 #include <iomanip>
```

```
00006 #include <vector>
00007 #include <cstdlib>
00008 #include <ctime>
00009 #include <string>
00010 #include <fstream>
00011 #include <sstream>
00012 #include <stdexcept>
00013 #include <limits>
00014 #include <algorithm>
00015 #include <chrono>
00016 #include <numeric>
00017 #include <random>
00018 #include <cstring>
00019 #include <cctype>
00020 #include <locale>
00021 #include <list>
00022
00023 using std::cin;
00024 using std::cout;
00025 using std::endl;
00026 using std::exception;
00027 using std::fixed;
00028 using std::getline;
00029 using std::ifstream;
00030 using std::invalid_argument;
00031 using std::ios;
00032 using std::isdigit;
00033 using std::isspace;
00034 using std::istringstream;
00035 using std::left;
00036 using std::list;
00037 using std::max;
00038 using std::move;
00039 using std::numeric_limits;
00040 using std::ofstream;
00041 using std::ostringstream;
00042 using std::out_of_range;
00043 using std::right;
00044 using std::runtime_error;
00045 using std::setprecision;
00046 using std::setw;
00047 using std::sort;
00048 using std::stof;
00049 using std::streamsize;
00050 using std::string;
00051 using std::to_string;
00052 using std::vector;
00053
00054 #endif
```

5.7 include/List_failo_apdorojimas.h File Reference

```
#include "List_Biblioteka.h"
#include "List_funkcijos.h"
```

Functions

- void List_skaitytiDuomenisIsFailo (const string &failoPavadinimas, list< List_Studentas > &studentai, long long &trukmeSkaitymo, long long &trukmeVidurkio)
- void List_skaiciuotiIsFailo (List_Studentas &studentas, bool tinkamiPazymiai, list< List_Studentas > &studentai)
- void skaitytiIrIsvestiDuomenis (const string &ivestiesFailoPavadinimas, const string &outputFileName, long long &trukmeSkaitymo, long long &trukmeVidurkio, long long &trukmeIrasymo)
- void List_padalintiRezultatuFaila (const string &ivestiesFailoPavadinimas, const string &islaikiusiu↩FailoPavadinimas, const string &neislaikiusiuFailoPavadinimas, long long &laikasSkaitymo, long long &rusiavimoLaikas, long long &laikasRasymo)
- void List_skaitytiIrIsvestiDuomenis (const string &ivestiesFailoPavadinimas, const string &irasymoFailo↩Pavadinimas, long long &trukmeSkaitymo, long long &trukmeVidurkio, long long &trukmeIrasymo)

5.7.1 Function Documentation

5.7.1.1 List_padalintiRezultatuFaila()

```
void List_padalintiRezultatuFaila (
            const string & ivestiesFailoPavadinimas,
            const string & islaikiusiuFailoPavadinimas,
            const string & neislaikiusiuFailoPavadinimas,
            long long & laikasSkaitymo,
            long long & rusiavimoLaikas,
            long long & laikasRasymo)
```

5.7.1.2 List_skaiciuotiIsFailo()

```
void List_skaiciuotiIsFailo (
            List_Studentas & studentas,
            bool tinkamiPazymiai,
            list< List_Studentas > & studentai)
```

5.7.1.3 List_skaitytiDuomenisIsFailo()

```
void List_skaitytiDuomenisIsFailo (
            const string & failoPavadinimas,
            list< List_Studentas > & studentai,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio)
```

5.7.1.4 List_skaitytiIrIsvestiDuomenis()

```
void List_skaitytiIrIsvestiDuomenis (
            const string & ivestiesFailoPavadinimas,
            const string & irasymoFailoPavadinimas,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio,
            long long & trukmeIrasymo)
```

5.7.1.5 skaitytiIrIsvestiDuomenis()

```
void skaitytiIrIsvestiDuomenis (
            const string & ivestiesFailoPavadinimas,
            const string & outputFileName,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio,
            long long & trukmeIrasymo)
```

5.8 List_failo_apdorojimas.h

Go to the documentation of this file.

```
00001 #ifndef LIST_FAILO_APDOROJIMAS_H
00002 #define LIST_FAILO_APDOROJIMAS_H
00003
00004 #include "List_Biblioteka.h"
00005 #include "List_funkcijos.h"
00006
00007 void List_skaitytiDuomenisIsFailo(const string &failoPavadinimas, list<List_Studentas> &studentai,
      long long &trukmeSkaitymo, long long &trukmeVidurkio);
00008 void List_skaiciuotiIsFailo(List_Studentas &studentas, bool tinkamiPazymiai, list<List_Studentas>
      &studentai);
00009 void skaitytiIrIsvestiDuomenis(const string &ivestiesFailoPavadinimas, const string &outputFileName,
      long long &trukmeSkaitymo, long long &trukmeVidurkio, long long &trukmeIrasymo);
00010 void List_padalintiRezultatuFaila(const string &ivestiesFailoPavadinimas, const string
      &islaikiusiuFailoPavadinimas, const string &neislaikiusiuFailoPavadinimas, long long &laikasSkaitymo,
      long long &rusiavimoLaikas, long long &laikasRasymo);
00011 void List_skaitytiIrIsvestiDuomenis(const string &ivestiesFailoPavadinimas, const string
      &irasymoFailoPavadinimas, long long &trukmeSkaitymo, long long &trukmeVidurkio, long long
      &trukmeIrasymo);
00012
00013 #endif
```

5.9 include/List_funkcijos.h File Reference

Antraštinis failas, kuriame apibrėžiama klasė List_Studentas ir su sąrašais susijusios funkcijos.

```
#include "List_Biblioteka.h"
#include "Vec_funkcijos.h"
```

Classes

- class List_Studentas

  Klasė, paveldėta iš Zmogus, skirta studentų duomenims valdyti naudojant `std::list`.

Functions

- void List_programa ()

  Vykdo programos funkcionalumą su sąrašais.
- float List_skaiciuotiVidurki (std::list< int > &pazymiai)

  Apskaičiuoja pažymių vidurkį.
- float List_skaiciuotiMediana (std::list< int > &pazymiai)

  Apskaičiuoja pažymių medianą.
- void List_ivestiStudentoDuomenis (List_Studentas &studentas)

  Įveda studento duomenis.
- List_Studentas List_generuotiAtsitiktiniStudenta ()

  Generuoja atsitiktinį studentą.
- void List_rusiuotiStudentus (std::list< List_Studentas > &studentai)

  Rūšiuoja studentų sąrašą.
- void List_skaitytiDuomenisIsFailo (const std::string &failoPavadinimas, std::list< List_Studentas > &studentai, long long &trukmeSkaitymo, long long &trukmeVidurkio)

  Skaito studentų duomenis iš failo.

5.9.1 Detailed Description

Antraštinis failas, kuriame apibrėžiama klasė List_Studentas ir su sąrašais susijusios funkcijos.

Definition in file List_funkcijos.h.

5.9.2 Function Documentation

5.9.2.1 List_generuotiAtsitiktiniStudenta()

```
List_Studentas List_generuotiAtsitiktiniStudenta ()
```

Generuoja atsitiktinį studentą.

Returns

Atsitiktinis studentas su sugeneruotais duomenimis.

Definition at line 108 of file List_funkcijos.cpp.

5.9.2.2 List_ivestiStudentoDuomenis()

```
void List_ivestiStudentoDuomenis (
            List_Studentas & studentas)
```

Įveda studento duomenis.

Parameters

studentas	Studentas, kurio duomenys įvedami.

Definition at line 5 of file List_funkcijos.cpp.

5.9.2.3 List_programa()

```
void List_programa ()
```

Vykdo programos funkcionalumą su sąrašais.

Definition at line 217 of file List_funkcijos.cpp.

5.9.2.4 List_rusiuotiStudentus()

```
void List_rusiuotiStudentus (
            std::list< List_Studentas > & studentai)
```

Rūšiuoja studentų sąrašą.

Parameters

studentai	Sąrašas, kuris bus rūšiuojamas.

Definition at line 76 of file List_funkcijos.cpp.

5.9.2.5 List_skaiciuotiMediana()

```
float List_skaiciuotiMediana (
            std::list< int > & pazymiai)
```

Apskaičiuoja pažymių medianą.

Parameters

pazymiai	Pažymių sąrašas.

Returns

Apskaičiuota mediana.

Definition at line 47 of file List_funkcijos.cpp.

5.9.2.6 List_skaiciuotiVidurki()

```
float List_skaiciuotiVidurki (
            std::list< int > & pazymiai)
```

Apskaičiuoja pažymių vidurkį.

Parameters

pazymiai	Pažymių sąrašas.

Returns

Apskaičiuotas vidurkis.

Definition at line 38 of file List_funkcijos.cpp.

5.9.2.7 List_skaitytiDuomenisIsFailo()

```
void List_skaitytiDuomenisIsFailo (
            const std::string & failoPavadinimas,
            std::list< List_Studentas > & studentai,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio)
```

Skaito studentų duomenis iš failo.

Parameters

failoPavadinimas	Failo pavadinimas.
studentai	Sąrašas, kuriame bus saugomi studentų duomenys.
trukmeSkaitymo	Skaitymo trukmė milisekundėmis.
trukmeVidurkio	Vidurkių skaičiavimo trukmė milisekundėmis.

Definition at line 38 of file List_failo_apdorojimas.cpp.

5.10 List_funkcijos.h

Go to the documentation of this file.

```
00001 #ifndef LIST_FUNKCIJOS_H
00002 #define LIST_FUNKCIJOS_H
00003
00004 #include "List_Biblioteka.h"
00005 #include "Vec_funkcijos.h"
00006
00016 class List_Studentas : public Zmogus {
00017 private:
00018     std::list<int> pazymiai;
00019     int egzaminoPazymys = 0;
00020     float vidurkis = 0;
00021     float mediana = 0;
00022     float galutinisVidurkis = 0;
00023     float galutineMediana = 0;
00024
00025 public:
00029     List_Studentas();
00030
00038     List_Studentas(const std::string &vardas, const std::string &pavarde,
00039                    const std::list<int> &pazymiai, int egzaminoPazymys);
00040
00044     ~List_Studentas();
00045
00050     List_Studentas(const List_Studentas &other);
00051
00057     List_Studentas &operator=(const List_Studentas &other);
00058
00059     // Getteriai
00064     std::list<int> getPazymiai() const;
00065
00070     int getEgzaminoPazymys() const;
00071
00076     float getVidurkis() const;
00077
00082     float getMediana() const;
00083
00088     float getGalutinisVidurkis() const;
00089
00094     float getGalutineMediana() const;
00095
00096     // Setteriai
00101     void List_setPazymiai(const std::list<int> &pazymiai);
00102
00107     void setEgzaminoPazymys(int egzaminoPazymys);
00108
00113     void setVidurkis(float vidurkis);
00114
00119     void setMediana(float mediana);
00120
00125     void setGalutinisVidurkis(float galutinisVidurkis);
00126
00131     void setGalutineMediana(float galutineMediana);
00132
00136     void printInfo() const override;
00137
00138     // Papildomi metodai
00143     void pridetiPazymi(int pazymys);
00144
00148     void List_skaiciuotiRezultatus();
00149
00156     friend std::ostream &operator<<(std::ostream &os, const List_Studentas &studentas);
00157
00164     friend std::istream &operator>>(std::istream &is, List_Studentas &studentas);
```

```
00165 };
00166
00167 // Funkcijų deklaracijos
00171 void List_programa();
00172
00178 float List_skaiciuotiVidurki(std::list<int> &pazymiai);
00179
00185 float List_skaiciuotiMediana(std::list<int> &pazymiai);
00186
00191 void List_ivestiStudentoDuomenis(List_Studentas &studentas);
00192
00197 List_Studentas List_generuotiAtsitiktiniStudenta();
00198
00203 void List_rusiuotiStudentus(std::list<List_Studentas> &studentai);
00204
00212 void List_skaitytiDuomenisIsFailo(
00213     const std::string &failoPavadinimas,
00214     std::list<List_Studentas> &studentai,
00215     long long &trukmeSkaitymo,
00216     long long &trukmeVidurkio);
00217
00218 #endif // LIST_FUNKCIJOS_H
```

5.11 include/Vec_Biblioteka.h File Reference

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <string>
#include <fstream>
#include <sstream>
#include <stdexcept>
#include <limits>
#include <algorithm>
#include <chrono>
#include <numeric>
#include <random>
#include <cstring>
#include <cctype>
#include <locale>
```

5.12 Vec_Biblioteka.h

Go to the documentation of this file.

```
00001 #ifndef BIBLIOTEKA_H_INCLUDED
00002 #define BIBLIOTEKA_H_INCLUDED
00003
00004 #include <iostream>
00005 #include <iomanip>
00006 #include <vector>
00007 #include <cstdlib>
00008 #include <ctime>
00009 #include <string>
00010 #include <fstream>
00011 #include <sstream>
00012 #include <stdexcept>
00013 #include <limits>
00014 #include <algorithm>
00015 #include <chrono>
00016 #include <numeric>
00017 #include <random>
00018 #include <cstring>
00019 #include <cctype>
00020 #include <locale>
```

```
00021
00022 using std::cin;
00023 using std::cout;
00024 using std::endl;
00025 using std::exception;
00026 using std::fixed;
00027 using std::getline;
00028 using std::ifstream;
00029 using std::invalid_argument;
00030 using std::ios;
00031 using std::isdigit;
00032 using std::isspace;
00033 using std::istringstream;
00034 using std::left;
00035 using std::max;
00036 using std::move;
00037 using std::numeric_limits;
00038 using std::ofstream;
00039 using std::ostringstream;
00040 using std::out_of_range;
00041 using std::right;
00042 using std::runtime_error;
00043 using std::setprecision;
00044 using std::setw;
00045 using std::sort;
00046 using std::stof;
00047 using std::streamsize;
00048 using std::string;
00049 using std::string_view;
00050 using std::to_string;
00051 using std::vector;
00052
00053 #endif
```

5.13 include/Vec_failo_apdorojimas.h File Reference

```
#include "Vec_Biblioteka.h"
#include "Vec_funkcijos.h"
```

Functions

- void skaitytiDuomenisIsFailo (const string &failoPavadinimas, vector< Studentas > &studentai, long long &trukmeSkaitymo, long long &trukmeVidurkio)
- void skaiciuotiIsFailo (Studentas &studentas, bool tinkamiPazymiai, vector< Studentas > &studentai)
- void skaitytiIrIsvestiDuomenis (const string &ivestiesFailoPavadinimas, const string &outputFileName, long long &trukmeSkaitymo, long long &trukmeVidurkio, long long &trukmeIrasymo)
- void padalintiRezultatuFaila (const string &ivestiesFailoPavadinimas, const string &islaikiusiuFailo↩ Pavadinimas, const string &neislaikiusiuFailoPavadinimas, long long &laikasSkaitymo, long long &rusiavimoLaikas, long long &laikasRasymo)

5.13.1 Function Documentation

5.13.1.1 padalintiRezultatuFaila()

```
void padalintiRezultatuFaila (
            const string & ivestiesFailoPavadinimas,
            const string & islaikiusiuFailoPavadinimas,
            const string & neislaikiusiuFailoPavadinimas,
            long long & laikasSkaitymo,
            long long & rusiavimoLaikas,
            long long & laikasRasymo)
```

5.13.1.2 skaiciuotiIsFailo()

```
void skaiciuotiIsFailo (
            Studentas & studentas,
            bool tinkamiPazymiai,
            vector< Studentas > & studentai)
```

5.13.1.3 skaitytiDuomenisIsFailo()

```
void skaitytiDuomenisIsFailo (
            const string & failoPavadinimas,
            vector< Studentas > & studentai,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio)
```

5.13.1.4 skaitytiIrIsvestiDuomenis()

```
void skaitytiIrIsvestiDuomenis (
            const string & ivestiesFailoPavadinimas,
            const string & outputFileName,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio,
            long long & trukmeIrasymo)
```

5.14 Vec_failo_apdorojimas.h

Go to the documentation of this file.

```
00001 #ifndef FAILO_APDOROJIMAS_H
00002 #define FAILO_APDOROJIMAS_H
00003
00004 #include "Vec_Biblioteka.h"
00005 #include "Vec_funkcijos.h"
00006
00007 void skaitytiDuomenisIsFailo(const string &failoPavadinimas, vector<Studentas> &studentai, long long
      &trukmeSkaitymo, long long &trukmeVidurkio);
00008 void skaiciuotiIsFailo(Studentas &studentas, bool tinkamiPazymiai, vector<Studentas> &studentai);
00009 void skaitytiIrIsvestiDuomenis(const string &ivestiesFailoPavadinimas, const string &outputFileName,
      long long &trukmeSkaitymo, long long &trukmeVidurkio, long long &trukmeIrasymo);
00010 void padalintiRezultatuFaila(const string &ivestiesFailoPavadinimas, const string
      &islaikiusiuFailoPavadinimas, const string &neislaikiusiuFailoPavadinimas, long long &laikasSkaitymo,
      long long &rusiavimoLaikas, long long &laikasRasymo);
00011
00012 #endif
```

5.15 include/Vec_funkcijos.h File Reference

```
#include "Vec_Biblioteka.h"
```

Classes

- class Zmogus

  Abstrakti bazinė klasė, reprezentuojanti asmenį su vardu ir pavarde.
- class Studentas

  Išvestinė klasė, reprezentuojanti studentą su pažymiais ir rezultatais.

Functions

- void Vec_programa ()

  Vykdo Vec programos funkcionalumą.
- Studentas generuotiAtsitiktiniStudenta ()

  Generuoja atsitiktinį studentą.
- int gautiPazymi (const std::string &klausimas)

  Grąžina pažymį, pagrįstą vartotojo įvestimi.
- float skaiciuotiVidurki (const std::vector< int > &vidurkis)

  Apskaičiuoja pažymių vidurkį.
- float skaiciuotiMediana (const std::vector< int > &mediana)

  Apskaičiuoja pažymių medianą.
- int generuotiSkaiciu (int min, int max)

  Generuoja atsitiktinį skaičių intervalo ribose.
- std::string generuotiVardaPavarde ()

  Generuoja atsitiktinį vardą ir pavardę.
- void generuotiFaila ()

  Generuoja studentų failą su atsitiktiniais duomenimis.
- void generuotiAtsitiktiniusFailus ()

  Generuoja kelis atsitiktinius studentų failus.
- void generuotiStudentuFaila (int studentuKiekis, const std::string &failoPavadinimas)

  Generuoja failą su tam tikru studentų skaičiumi.
- void vykdytiVisusZingsnius ()

  Vykdo visus programos žingsnius.
- void skaitytiDuomenisIsFailo (const std::string &failoPavadinimas, std::vector< Studentas > &studentai, long long &trukmeSkaitymo, long long &trukmeVidurkio)

  Nuskaito studentų duomenis iš failo.

5.15.1 Function Documentation

5.15.1.1 gautiPazymi()

```
int gautiPazymi (
            const std::string & klausimas)
```

Grąžina pažymį, pagrįstą vartotojo įvestimi.

Parameters

klausimas	Klausimas, užduodamas vartotojui.

Returns

Įvestas pažymys.

5.15.1.2 generuotiAtsitiktiniStudenta()

```
Studentas generuotiAtsitiktiniStudenta ()
```

Generuoja atsitiktinį studentą.

Returns

Studentas su atsitiktiniais duomenimis.

Definition at line 132 of file Vec_funkcijos_papildomos.cpp.

5.15.1.3 generuotiAtsitiktiniusFailus()

```
void generuotiAtsitiktiniusFailus ()
```

Generuoja kelis atsitiktinius studentų failus.

Definition at line 173 of file Vec_funkcijos_papildomos.cpp.

5.15.1.4 generuotiFaila()

```
void generuotiFaila ()
```

Generuoja studentų failą su atsitiktiniais duomenimis.

Definition at line 54 of file Vec_generuoti_failus.cpp.

5.15.1.5 generuotiSkaiciu()

```
int generuotiSkaiciu (
            int min,
            int max)
```

Generuoja atsitiktinį skaičių intervalo ribose.

Parameters

min	Mažiausia reikšmė.
max	Didžiausia reikšmė.

Returns

Generuotas atsitiktinis skaičius.

Definition at line 109 of file Vec_funkcijos_papildomos.cpp.

5.15.1.6 generuotiStudentuFaila()

```
void generuotiStudentuFaila (
            int studentuKiekis,
            const std::string & failoPavadinimas)
```

Generuoja failą su tam tikru studentų skaičiumi.

Parameters

studentuKiekis	Kiekis studentų, kurie bus sugeneruoti.
failoPavadinimas	Išvesties failo pavadinimas.

5.15.1.7 generuotiVardaPavarde()

```
std::string generuotiVardaPavarde ()
```

Generuoja atsitiktinį vardą ir pavardę.

Returns

Sugeneruotas vardas ir pavardė.

Definition at line 118 of file Vec_funkcijos_papildomos.cpp.

5.15.1.8 skaiciuotiMediana()

```
float skaiciuotiMediana (
            const std::vector< int > & mediana)
```

Apskaičiuoja pažymių medianą.

Parameters

mediana	Pažymių sąrašas.

Returns

Apskaičiuota mediana.

Definition at line 77 of file Vec_funkcijos_papildomos.cpp.

5.15.1.9 skaiciuotiVidurki()

```
float skaiciuotiVidurki (
            const std::vector< int > & vidurkis)
```

Apskaičiuoja pažymių vidurkį.

Parameters

vidurkis	Pažymių sąrašas.

Returns

Apskaičiuotas vidurkis.

Definition at line 100 of file Vec_funkcijos_papildomos.cpp.

5.15.1.10 skaitytiDuomenisIsFailo()

```
void skaitytiDuomenisIsFailo (
            const std::string & failoPavadinimas,
            std::vector< Studentas > & studentai,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio)
```

Nuskaito studentų duomenis iš failo.

Parameters

failoPavadinimas	Įvesties failo pavadinimas.
studentai	Vektorius, kuriame saugomi studentų duomenys.
trukmeSkaitymo	Skaitymo trukmė milisekundėmis.
trukmeVidurkio	Vidurkių skaičiavimo trukmė milisekundėmis.

Definition at line 30 of file Vec_failo_apdorojimas.cpp.

5.15.1.11 Vec_programa()

```
void Vec_programa ()
```

Vykdo Vec programos funkcionalumą.

Definition at line 7 of file Vec_funkcijos.cpp.

5.15.1.12 vykdytiVisusZingsnius()

```
void vykdytiVisusZingsnius ()
```

Vykdo visus programos žingsnius.

Definition at line 184 of file Vec_funkcijos_papildomos.cpp.

5.16 Vec_funkcijos.h

Go to the documentation of this file.

```
00001 #ifndef FUNKCIJOS_H
00002 #define FUNKCIJOS_H
00003
00004 #include "Vec_Biblioteka.h"
00005
00017 class Zmogus {
00018 protected:
00019     std::string vardas;
00020     std::string pavarde;
00021
00022 public:
00026     Zmogus();
00027
00033     Zmogus(const std::string &vardas, const std::string &pavarde);
00034
00038     virtual ~Zmogus();
00039
00044     Zmogus(const Zmogus &other);
00045
00051     Zmogus &operator=(const Zmogus &other);
00052
00056     virtual void printInfo() const = 0;
00057
00062     std::string getVardas() const;
00063
00068     std::string getPavarde() const;
00069
00074     void setVardas(const std::string &vardas);
00075
00080     void setPavarde(const std::string &pavarde);
00081 };
00082
00087 class Studentas : public Zmogus {
00088 private:
```

```
00089     std::vector<int> pazymiai;
00090     int egzaminoPazymys = 0;
00091     float vidurkis = 0;
00092     float mediana = 0;
00093     float galutinisVidurkis = 0;
00094     float galutineMediana = 0;
00095
00096 public:
00100     Studentas();
00101
00109     Studentas(const std::string &vardas, const std::string &pavarde, const std::vector<int> &pazymiai,
      int egzaminoPazymys);
00110
00114     ~Studentas();
00115
00120     Studentas(const Studentas &other);
00121
00127     Studentas &operator=(const Studentas &other);
00128
00129     // Getteriai ir setteriai (modeliuojama pagal visų funkcijų veikimo principą)
00130     std::vector<int> getPazymiai() const;
00131     int getEgzaminoPazymys() const;
00132     float getVidurkis() const;
00133     float getMediana() const;
00134     float getGalutinisVidurkis() const;
00135     float getGalutineMediana() const;
00136
00137     void setPazymiai(const std::vector<int> &pazymiai);
00138     void setEgzaminoPazymys(int egzaminoPazymys);
00139     void setVidurkis(float vidurkis);
00140     void setMediana(float mediana);
00141     void setGalutineMediana(float galutineMediana);
00142     void setGalutinisVidurkis(float galutinisVidurkis);
00143
00147     void skaiciuotiRezultatus();
00148
00153     void pridetiPazymi(int pazymys);
00154
00158     void printInfo() const override;
00159
00160     // Draugiškos funkcijos
00161     friend std::ostream &operator<<(std::ostream &os, const Studentas &studentas);
00162     friend std::istream &operator>>(std::istream &is, Studentas &studentas);
00163 };
00164
00165 // Kitos funkcijos
00169 void Vec_programa();
00170
00175 Studentas generuotiAtsitiktiniStudenta();
00176
00182 int gautiPazymi(const std::string &klausimas);
00183
00189 float skaiciuotiVidurki(const std::vector<int> &vidurkis);
00190
00196 float skaiciuotiMediana(const std::vector<int> &mediana);
00197
00204 int generuotiSkaiciu(int min, int max);
00205
00210 std::string generuotiVardaPavarde();
00211
00215 void generuotiFaila();
00216
00220 void generuotiAtsitiktiniusFailus();
00221
00227 void generuotiStudentuFaila(int studentuKiekis, const std::string &failoPavadinimas);
00228
00232 void vykdytiVisusZingsnius();
00233
00241 void skaitytiDuomenisIsFailo(
00242     const std::string &failoPavadinimas,
00243     std::vector<Studentas> &studentai,
00244     long long &trukmeSkaitymo,
00245     long long &trukmeVidurkio);
00246
00247 #endif
```

5.17 include/Vec_funkcijos_papildomos.h File Reference

```
#include "Vec_Biblioteka.h"
#include "Vec_funkcijos.h"
```

Functions

- void rusiuotiStudentusPagalPavarde (std::vector< Studentas > &studentai)
- void rusiuotiStudentusPagalVarda (std::vector< Studentas > &studentai)
- void rusiuotiPagalVidurkiDidejanciai (std::vector< Studentas > &studentai)
- void rusiuotiPagalVidurkiMazejanciai (std::vector< Studentas > &studentai)
- float skaiciuotiMediana (vector< int > &pazymiai)
- float skaiciuotiVidurki (vector< int > &pazymiai)
- int generuotiSkaiciu (int min, int max)

  Generuoja atsitiktinį skaičių intervalo ribose.
- string generuotiVardaPavarde ()

  Generuoja atsitiktinį vardą ir pavardę.
- Studentas generuotiAtsitiktiniStudenta ()

  Generuoja atsitiktinį studentą.
- void generuotiAtsitiktiniusFailus ()

  Generuoja kelis atsitiktinius studentų failus.
- void vykdytiVisusZingsnius ()

  Vykdo visus programos žingsnius.
- void vykdytiKeliskart (int &kartai)
- void rusiuotiRezultatus (long long &trukmeRezultatuSkaitymo, long long &trukmeRezultatuSkaidymas, long long &trukmeSkaidymoIrasymas)
- void ivestiDuomenisRanka (vector< Studentas > &studentai)
- void automatiskaiGeneruotiDuomenis (vector< Studentas > &studentai)
- void nuskaitytiDuomenisIsFailo (vector< Studentas > &studentai, long long &trukmeSkaitymo, long long &trukmeVidurkio)
- void skaiciuotiRezultatus (long long &trukmeSkaitymo, long long &trukmeVidurkio, long long &trukmeIrasymo)

5.17.1 Function Documentation

5.17.1.1 automatiskaiGeneruotiDuomenis()

```
void automatiskaiGeneruotiDuomenis (
            vector< Studentas > & studentai)
```

Definition at line 298 of file Vec_funkcijos_papildomos.cpp.

5.17.1.2 generuotiAtsitiktiniStudenta()

```
Studentas generuotiAtsitiktiniStudenta ()
```

Generuoja atsitiktinį studentą.

Returns

Studentas su atsitiktiniais duomenimis.

Definition at line 132 of file Vec_funkcijos_papildomos.cpp.

5.17.1.3 generuotiAtsitiktiniusFailus()

```
void generuotiAtsitiktiniusFailus ()
```

Generuoja kelis atsitiktinius studentų failus.

Definition at line 173 of file Vec_funkcijos_papildomos.cpp.

5.17.1.4 generuotiSkaiciu()

```
int generuotiSkaiciu (
            int min,
            int max)
```

Generuoja atsitiktinį skaičių intervalo ribose.

Parameters

min	Mažiausia reikšmė.
max	Didžiausia reikšmė.

Returns

Generuotas atsitiktinis skaičius.

Definition at line 109 of file Vec_funkcijos_papildomos.cpp.

5.17.1.5 generuotiVardaPavarde()

```
string generuotiVardaPavarde ()
```

Generuoja atsitiktinį vardą ir pavardę.

Returns

Sugeneruotas vardas ir pavardė.

Definition at line 118 of file Vec_funkcijos_papildomos.cpp.

5.17.1.6 ivestiDuomenisRanka()

```
void ivestiDuomenisRanka (
            vector< Studentas > & studentai)
```

Definition at line 262 of file Vec_funkcijos_papildomos.cpp.

5.17.1.7 nuskaitytiDuomenisIsFailo()

```
void nuskaitytiDuomenisIsFailo (
            vector< Studentas > & studentai,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio)
```

Definition at line 328 of file Vec_funkcijos_papildomos.cpp.

5.17.1.8 rusiuotiPagalVidurkiDidejanciai()

```
void rusiuotiPagalVidurkiDidejanciai (
            std::vector< Studentas > & studentai)
```

Definition at line 63 of file Vec_funkcijos_papildomos.cpp.

5.17.1.9 rusiuotiPagalVidurkiMazejanciai()

```
void rusiuotiPagalVidurkiMazejanciai (
            std::vector< Studentas > & studentai)
```

Definition at line 70 of file Vec_funkcijos_papildomos.cpp.

5.17.1.10 rusiuotiRezultatus()

```
void rusiuotiRezultatus (
            long long & trukmeRezultatuSkaitymo,
            long long & trukmeRezultatuSkaidymas,
            long long & trukmeSkaidymoIrasymas)
```

Definition at line 412 of file Vec_funkcijos_papildomos.cpp.

5.17.1.11 rusiuotiStudentusPagalPavarde()

```
void rusiuotiStudentusPagalPavarde (
            std::vector< Studentas > & studentai)
```

Definition at line 37 of file Vec_funkcijos_papildomos.cpp.

5.17.1.12 rusiuotiStudentusPagalVarda()

```
void rusiuotiStudentusPagalVarda (
            std::vector< Studentas > & studentai)
```

Definition at line 50 of file Vec_funkcijos_papildomos.cpp.

5.17.1.13 skaiciuotiMediana()

```
float skaiciuotiMediana (
            vector< int > & pazymiai)
```

5.17.1.14 skaiciuotiRezultatus()

```
void skaiciuotiRezultatus (
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio,
            long long & trukmeIrasymo)
```

Definition at line 383 of file Vec_funkcijos_papildomos.cpp.

5.17.1.15 skaiciuotiVidurki()

```
float skaiciuotiVidurki (
            vector< int > & pazymiai)
```

5.17.1.16 vykdytiKeliskart()

```
void vykdytiKeliskart (
            int & kartai)
```

Definition at line 438 of file Vec_funkcijos_papildomos.cpp.

5.17.1.17 vykdytiVisusZingsnius()

```
void vykdytiVisusZingsnius ()
```

Vykdo visus programos žingsnius.

Definition at line 184 of file Vec_funkcijos_papildomos.cpp.

5.18 Vec_funkcijos_papildomos.h

Go to the documentation of this file.

```
00001 #ifndef VEC_FUNKCIJOS_PAPILDOMOS_H
00002 #define VEC_FUNKCIJOS_PAPILDOMOS_H
00003
00004 #include "Vec_Biblioteka.h"
00005 #include "Vec_funkcijos.h"
00006
00007 void rusiuotiStudentusPagalPavarde(std::vector<Studentas> &studentai);
00008 void rusiuotiStudentusPagalVarda(std::vector<Studentas> &studentai);
00009 void rusiuotiPagalVidurkiDidejanciai(std::vector<Studentas> &studentai);
00010 void rusiuotiPagalVidurkiMazejanciai(std::vector<Studentas> &studentai);
00011 float skaiciuotiMediana(vector<int> &pazymiai);
00012 float skaiciuotiVidurki(vector<int> &pazymiai);
00013 int generuotiSkaiciu(int min, int max);
00014 string generuotiVardaPavarde();
00015 Studentas generuotiAtsitiktiniStudenta();
00016 void generuotiAtsitiktiniusFailus();
00017 void vykdytiVisusZingsnius();
00018 void vykdytiKeliskart(int &kartai);
00019 void rusiuotiRezultatus(long long &trukmeRezultatuSkaitymo, long long &trukmeRezultatuSkaidymas, long
      long &trukmeSkaidymoIrasymas);
00020 void ivestiDuomenisRanka(vector<Studentas> &studentai);
00021 void automatiskaiGeneruotiDuomenis(vector<Studentas> &studentai);
00022 void nuskaitytiDuomenisIsFailo(vector<Studentas> &studentai, long long &trukmeSkaitymo, long long
      &trukmeVidurkio);
00023 void skaiciuotiRezultatus(long long &trukmeSkaitymo, long long &trukmeVidurkio, long long
      &trukmeIrasymo);
00024
00025 #endif
```

5.19 src/List_failo_apdorojimas.cpp File Reference

```
#include "List_failo_apdorojimas.h"
#include "List_funkcijos.h"
#include "Vec_funkcijos_papildomos.h"
```

Functions

- void List_skaiciuotiIsFailo (List_Studentas &studentas, bool tinkamiPazymiai, std::list< List_Studentas > &studentai)
- void List_skaitytiDuomenisIsFailo (const std::string &failoPavadinimas, std::list< List_Studentas > &studentai, long long &trukmeSkaitymo, long long &trukmeVidurkio)

  Skaito studentų duomenis iš failo.
- void List_skaitytiIrIsvestiDuomenis (const std::string &ivestiesFailoPavadinimas, const std::string &irasymoFailoPavadinimas, long long &trukmeSkaitymo, long long &trukmeVidurkio, long long &trukmeIrasymo)
- void List_padalintiRezultatuFaila (const std::string &ivestiesFailoPavadinimas, const std::string &islaikiusiuFailoPavadinimas, const std::string &neislaikiusiuFailoPavadinimas, long long &laikasSkaitymo, long long &rusiavimoLaikas, long long &laikasRasymo)

5.19.1 Function Documentation

5.19.1.1 List_padalintiRezultatuFaila()

```
void List_padalintiRezultatuFaila (
            const std::string & ivestiesFailoPavadinimas,
            const std::string & islaikiusiuFailoPavadinimas,
            const std::string & neislaikiusiuFailoPavadinimas,
            long long & laikasSkaitymo,
            long long & rusiavimoLaikas,
            long long & laikasRasymo)
```

Definition at line 202 of file List_failo_apdorojimas.cpp.

5.19.1.2 List_skaiciuotiIsFailo()

```
void List_skaiciuotiIsFailo (
            List_Studentas & studentas,
            bool tinkamiPazymiai,
            std::list< List_Studentas > & studentai)
```

Definition at line 5 of file List_failo_apdorojimas.cpp.

5.19.1.3 List_skaitytiDuomenisIsFailo()

```
void List_skaitytiDuomenisIsFailo (
            const std::string & failoPavadinimas,
            std::list< List_Studentas > & studentai,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio)
```

Skaito studentų duomenis iš failo.

Parameters

failoPavadinimas	Failo pavadinimas.
studentai	Sąrašas, kuriame bus saugomi studentų duomenys.
trukmeSkaitymo	Skaitymo trukmė milisekundėmis.
trukmeVidurkio	Vidurkių skaičiavimo trukmė milisekundėmis.

Definition at line 38 of file List_failo_apdorojimas.cpp.

5.19.1.4 List_skaitytiIrIsvestiDuomenis()

```
void List_skaitytiIrIsvestiDuomenis (
            const std::string & ivestiesFailoPavadinimas,
            const std::string & irasymoFailoPavadinimas,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio,
            long long & trukmeIrasymo)
```

Definition at line 154 of file List_failo_apdorojimas.cpp.

5.20 List_failo_apdorojimas.cpp

Go to the documentation of this file.

```
00001 #include "List_failo_apdorojimas.h"
00002 #include "List_funkcijos.h"
00003 #include "Vec_funkcijos_papildomos.h"
00004
00005 void List_skaiciuotiIsFailo(List_Studentas &studentas, bool tinkamiPazymiai, std::list<List_Studentas>
      &studentai)
00006 {
00007     if (tinkamiPazymiai && !studentas.getPazymiai().empty())
00008     {
00009         std::list<int> pazymiai = studentas.getPazymiai();
00010
00011         // Paskutinis pažymys - egzaminas
00012         studentas.setEgzaminoPazymys(pazymiai.back());
00013         pazymiai.pop_back();
00014
00015         float vidurkis = List_skaiciuotiVidurki(pazymiai);
00016         float mediana = List_skaiciuotiMediana(pazymiai);
00017
00018         studentas.setVidurkis(vidurkis);
00019         studentas.setMediana(mediana);
00020
00021         const float egzaminoBalas = 0.6 * studentas.getEgzaminoPazymys();
00022         const float vidurkioBalas = 0.4 * vidurkis;
00023         const float medianosBalas = 0.4 * mediana;
00024
00025         studentas.setGalutinisVidurkis(vidurkioBalas + egzaminoBalas);
00026         studentas.setGalutineMediana(medianosBalas + egzaminoBalas);
00027
00028         studentai.push_back(studentas);
00029     }
00030     else
00031     {
00032         std::cout « "Klaida: truksta pazymiu studentui "
00033                  « studentas.getVardas() « " "
00034                  « studentas.getPavarde() « "\n";
00035     }
00036 }
00037
00038 void List_skaitytiDuomenisIsFailo(const std::string &failoPavadinimas,
00039                                   std::list<List_Studentas> &studentai,
00040                                   long long &trukmeSkaitymo,
00041                                   long long &trukmeVidurkio)
00042 {
00043     auto pradziaSkaitymo = std::chrono::high_resolution_clock::now();
```

```
00044
00045     std::ifstream failas(failoPavadinimas, std::ios::in | std::ios::binary);
00046     if (!failas)
00047     {
00048         throw std::runtime_error("Failo " + failoPavadinimas + " nera.");
00049     }
00050
00051     std::string buffer;
00052     buffer.reserve(1048576); // Buferio dydis baitais
00053
00054     // Praleidžia antraštę
00055     getline(failas, buffer);
00056
00057     while (getline(failas, buffer))
00058     {
00059         if (buffer.length() < 52)
00060         { // Minimalaus ilgio patikrinimas
00061             throw std::runtime_error("Netinkamas eilutes ilgis");
00062         }
00063
00064         List_Studentas studentas;
00065
00066         // Skaito vardą ir pavardę
00067         studentas.setVardas(buffer.substr(0, 16));
00068         studentas.setPavarde(buffer.substr(16, 32));
00069
00070         // Funkcija ištrinti whitespace iš string
00071         auto trim = [](std::string &str)
00072         {
00073             // Ištrinti pradžioję esantį whitespace
00074             str.erase(str.begin(), std::find_if(str.begin(), str.end(), [](unsigned char ch)
00075                                                 { return !std::isspace(ch); }));
00076             // Ištrinti pabaigoje esantį whitespace
00077             str.erase(std::find_if(str.rbegin(), str.rend(), [](unsigned char ch)
00078                                    { return !std::isspace(ch); })
00079                           .base(),
00080                       str.end());
00081         };
00082
00083         std::string vardas = studentas.getVardas();
00084         trim(vardas);
00085         std::string pavarde = studentas.getPavarde();
00086         trim(pavarde);
00087         studentas.setVardas(vardas);
00088         studentas.setPavarde(pavarde);
00089
00090         // Pažymiai prasideda nuo 52 simbolio
00091         size_t pozicija = 52;
00092         bool tinkamiPazymiai = true;
00093         std::list<int> pazymiai;
00094
00095         while (pozicija < buffer.length())
00096         {
00097             // Praleidžia whitespace
00098             while (pozicija < buffer.length() && isspace(buffer[pozicija]))
00099                 pozicija++;
00100             if (pozicija >= buffer.length())
00101                 break;
00102
00103             int grade = 0;
00104             bool tinkamas = true;
00105
00106             // Patikrina ar skaičius
00107             if (isdigit(buffer[pozicija]))
00108             {
00109                 while (pozicija < buffer.length() && isdigit(buffer[pozicija]))
00110                 {
00111                     grade = grade * 10 + (buffer[pozicija] - '0');
00112                     pozicija++;
00113                 }
00114
00115                 // Patikrina ar tarp 0 ir 10
00116                 if (grade < 0 || grade > 10)
00117                 {
00118                     tinkamas = false;
00119                 }
00120             }
00121             else
00122             {
00123                 // Jeigu ne skaičius:
00124                 tinkamas = false; // Netinkamas
00125                 pozicija++;       // Eina į kitą poziciją
00126             }
00127
00128             if (tinkamas)
00129             {
00130                 pazymiai.push_back(grade);
```

```
00131             }
00132             else
00133             {
00134                 tinkamiPazymiai = false;
00135                 break;
00136             }
00137
00138             // Praleidžia whitespace
00139             while (pozicija < buffer.length() && isspace(buffer[pozicija]))
00140                 pozicija++;
00141         }
00142
00143         // Nustatome pažymius
00144         studentas.List_setPazymiai(pazymiai);
00145
00146         List_skaiciuotiIsFailo(studentas, tinkamiPazymiai, studentai);
00147     }
00148
00149     auto pabaigaSkaitymo = std::chrono::high_resolution_clock::now();
00150     trukmeSkaitymo = std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaSkaitymo -
      pradziaSkaitymo).count();
00151     trukmeVidurkio = 0;
00152 }
00153
00154 void List_skaitytiIrIsvestiDuomenis(const std::string &ivestiesFailoPavadinimas,
00155                                     const std::string &irasymoFailoPavadinimas,
00156                                     long long &trukmeSkaitymo,
00157                                     long long &trukmeVidurkio,
00158                                     long long &trukmeIrasymo)
00159 {
00160     std::list<List_Studentas> studentai;
00161     List_skaitytiDuomenisIsFailo(ivestiesFailoPavadinimas, studentai, trukmeSkaitymo, trukmeVidurkio);
00162
00163     auto pradziaIrasimo = std::chrono::high_resolution_clock::now();
00164
00165     // Naudoja stringstream buferiui
00166     std::ostringstream buffer;
00167
00168     // Įrašo antraštę į buferį
00169     buffer « std::left « std::setw(16) « "Pavarde"
00170            « std::setw(16) « "Vardas"
00171            « std::setw(25) « "Galutinis Vidurkis"
00172            « "Galutine Mediana\n";
00173     buffer « std::string(70, '-') « "\n";
00174
00175     // Sort students by surname
00176     studentai.sort([](const List_Studentas &a, const List_Studentas &b)
00177                    { return a.getPavarde() < b.getPavarde(); });
00178
00179     for (const auto &studentas : studentai)
00180     {
00181         buffer « std::left « std::setw(16) « studentas.getPavarde()
00182                « std::setw(16) « studentas.getVardas()
00183                « std::setw(25) « std::fixed « std::setprecision(2) « studentas.getGalutinisVidurkis()
00184                « std::fixed « std::setprecision(2) « studentas.getGalutineMediana()
00185                « "\n";
00186     }
00187
00188     // Atidaro failą įrašymui
00189     std::ofstream irasymoFailas(irasymoFailoPavadinimas, std::ios::out | std::ios::binary);
00190     if (!irasymoFailas)
00191     {
00192         throw std::runtime_error("Nepavyko atidaryti isvesties failo " + irasymoFailoPavadinimas);
00193     }
00194
00195     // Įrašo visą buferį vienu metu
00196     irasymoFailas « buffer.str();
00197     irasymoFailas.close();
00198
00199     auto pabaigaIrasimo = std::chrono::high_resolution_clock::now();
00200     trukmeIrasymo = std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaIrasimo -
      pradziaIrasimo).count();
00201 }
00202 void List_padalintiRezultatuFaila(const std::string &ivestiesFailoPavadinimas,
00203                                   const std::string &islaikiusiuFailoPavadinimas,
00204                                   const std::string &neislaikiusiuFailoPavadinimas,
00205                                   long long &laikasSkaitymo,
00206                                   long long &rusiavimoLaikas,
00207                                   long long &laikasRasymo)
00208 {
00209     auto pradziaSkaitymo = std::chrono::high_resolution_clock::now();
00210
00211     // Atidaro duomenis binariniu režimu
00212     std::ifstream ivestiesFailas(ivestiesFailoPavadinimas, std::ios::in | std::ios::binary);
00213     if (!ivestiesFailas)
00214     {
00215         throw std::runtime_error("Nepavyko atidaryti ivesties failo " + ivestiesFailoPavadinimas);
```

```
00216     }
00217
00218     // Perskaito duomenis į buferį
00219     ivestiesFailas.seekg(0, std::ios::end);
00220     size_t failoDydis = ivestiesFailas.tellg();
00221     ivestiesFailas.seekg(0, std::ios::beg);
00222     std::string failoTurinys(failoDydis, '\0');
00223     ivestiesFailas.read(&failoTurinys[0], failoDydis);
00224     ivestiesFailas.close();
00225
00226     std::ofstream islaikiusiuFailas(islaikiusiuFailoPavadinimas, std::ios::out | std::ios::binary);
00227     std::ofstream neislaikiusiuFailas(neislaikiusiuFailoPavadinimas, std::ios::out |
      std::ios::binary);
00228     if (!islaikiusiuFailas || !neislaikiusiuFailas)
00229     {
00230         throw std::runtime_error("Nepavyko atidaryti išvesties failų");
00231     }
00232
00233     size_t pos = 0;
00234     size_t newline_pos = failoTurinys.find('\n');
00235     islaikiusiuFailas « failoTurinys.substr(0, newline_pos) « '\n';
00236     neislaikiusiuFailas « failoTurinys.substr(0, newline_pos) « '\n';
00237     pos = newline_pos + 1;
00238
00239     newline_pos = failoTurinys.find('\n', pos);
00240     islaikiusiuFailas « failoTurinys.substr(pos, newline_pos - pos) « '\n';
00241     neislaikiusiuFailas « failoTurinys.substr(pos, newline_pos - pos) « '\n';
00242     pos = newline_pos + 1;
00243
00244     // Naudojame list su List_Studentas struktūra
00245     std::list<List_Studentas> studentai;
00246     std::list<List_Studentas> vargsiukai;
00247
00248     while (pos < failoTurinys.size())
00249     {
00250         newline_pos = failoTurinys.find('\n', pos);
00251         std::string_view line(failoTurinys.data() + pos, newline_pos - pos);
00252         pos = newline_pos + 1;
00253
00254         List_Studentas student;
00255         size_t word_start = line.find_first_not_of(" \t");
00256         size_t word_end = line.find(' ', word_start);
00257
00258         if (word_end != std::string_view::npos)
00259         {
00260             student.setPavarde(std::string(line.substr(word_start, word_end - word_start)));
00261             word_start = line.find_first_not_of(" \t", word_end);
00262             word_end = line.find(' ', word_start);
00263
00264             if (word_end != std::string_view::npos)
00265             {
00266                 student.setVardas(std::string(line.substr(word_start, word_end - word_start)));
00267                 word_start = line.find_first_not_of(" \t", word_end);
00268                 word_end = line.find(' ', word_start);
00269
00270                 if (word_end != std::string_view::npos)
00271                 {
00272                     float galutinisVidurkis = std::stof(std::string(line.substr(word_start,
00273                                                                                 word_end -
      word_start)));
00274                     word_start = line.find_first_not_of(" \t", word_end);
00275                     float galutineMediana = std::stof(std::string(line.substr(word_start)));
00276
00277                     student.setGalutinisVidurkis(galutinisVidurkis);
00278                     student.setGalutineMediana(galutineMediana);
00279
00280                     if (galutinisVidurkis >= 5.0f)
00281                     {
00282                         studentai.push_back(student);
00283                     }
00284                     else
00285                     {
00286                         vargsiukai.push_back(student);
00287                     }
00288                 }
00289             }
00290         }
00291     }
00292
00293     auto pabaigaSkaitymo = std::chrono::high_resolution_clock::now();
00294     laikasSkaitymo = std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaSkaitymo -
      pradziaSkaitymo).count();
00295     std::cout « "Failo skaitymas uztruko " « laikasSkaitymo « " ms." « std::endl;
00296
00297     auto pradetiRusiavima = std::chrono::high_resolution_clock::now();
00298
00299     // Rūšiuojame abu sąrašus
```

```
00300     studentai.sort([](const List_Studentas &a, const List_Studentas &b)
00301                    { return a.getGalutinisVidurkis() > b.getGalutinisVidurkis(); });
00302
00303     vargsiukai.sort([](const List_Studentas &a, const List_Studentas &b)
00304                     { return a.getGalutinisVidurkis() > b.getGalutinisVidurkis(); });
00305
00306     auto pabaigaRusiavimo = std::chrono::high_resolution_clock::now();
00307     rusiavimoLaikas = std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaRusiavimo -
    pradetiRusiavima).count();
00308     std::cout << "Rusiavimas uztruko " << rusiavimoLaikas << " ms." << std::endl;
00309
00310     auto pradetiRasyma = std::chrono::high_resolution_clock::now();
00311
00312     // Rašome kietiakus (likusius studentai konteineryje)
00313     for (const auto &studentas : studentai)
00314     {
00315         islaikiusiuFailas << std::left << std::setw(15) << studentas.getPavarde() << " "
00316                           << std::setw(15) << studentas.getVardas() << " "
00317                           << std::setw(24) << studentas.getGalutinisVidurkis() << " "
00318                           << studentas.getGalutineMediana() << "\n";
00319     }
00320
00321     // Rašome vargšiukus
00322     for (const auto &studentas : vargsiukai)
00323     {
00324         neislaikiusiuFailas << std::left << std::setw(15) << studentas.getPavarde() << " "
00325                             << std::setw(15) << studentas.getVardas() << " "
00326                             << std::setw(24) << studentas.getGalutinisVidurkis() << " "
00327                             << studentas.getGalutineMediana() << "\n";
00328     }
00329
00330     auto pabaigaRasymo = std::chrono::high_resolution_clock::now();
00331     laikasRasymo = std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaRasymo -
    pradetiRasyma).count();
00332
00333     // Uždaryti failus
00334     islaikiusiuFailas.close();
00335     neislaikiusiuFailas.close();
00336 }
```

5.21 src/List_funkcijos.cpp File Reference

```
#include "List_funkcijos.h"
#include "List_failo_apdorojimas.h"
#include "Vec_funkcijos_papildomos.h"
```

Functions

- void List_ivestiStudentoDuomenis (List_Studentas &studentas)

  Įveda studento duomenis.
- float List_skaiciuotiVidurki (std::list< int > &pazymiai)

  Apskaičiuoja pažymių vidurkį.
- float List_skaiciuotiMediana (std::list< int > &pazymiai)

  Apskaičiuoja pažymių medianą.
- void List_rusiuotiStudentus (std::list< List_Studentas > &studentai)

  Rūšiuoja studentų sąrašą.
- void List_rusiuotiStudentusPagalVarda (std::list< List_Studentas > &studentai)
- void List_rusiuotiStudentusPagalPavarde (std::list< List_Studentas > &studentai)
- List_Studentas List_generuotiAtsitiktiniStudenta ()

  Generuoja atsitiktinį studentą.
- void List_vykdytiVisusZingsnius ()
- void List_programa ()

  Vykdo programos funkcionalumą su sąrašais.

5.21.1 Function Documentation

5.21.1.1 List_generuotiAtsitiktiniStudenta()

```
List_Studentas List_generuotiAtsitiktiniStudenta ()
```

Generuoja atsitiktinį studentą.

Returns

Atsitiktinis studentas su sugeneruotais duomenimis.

Definition at line 108 of file List_funkcijos.cpp.

5.21.1.2 List_ivestiStudentoDuomenis()

```
void List_ivestiStudentoDuomenis (
            List_Studentas & studentas)
```

Įveda studento duomenis.

Parameters

studentas	Studentas, kurio duomenys įvedami.

Definition at line 5 of file List_funkcijos.cpp.

5.21.1.3 List_programa()

```
void List_programa ()
```

Vykdo programos funkcionalumą su sąrašais.

Definition at line 217 of file List_funkcijos.cpp.

5.21.1.4 List_rusiuotiStudentus()

```
void List_rusiuotiStudentus (
            std::list< List_Studentas > & studentai)
```

Rūšiuoja studentų sąrašą.

Parameters

studentai	Sąrašas, kuris bus rūšiuojamas.

Definition at line 76 of file List_funkcijos.cpp.

5.21.1.5 List_rusiuotiStudentusPagalPavarde()

```
void List_rusiuotiStudentusPagalPavarde (
            std::list< List_Studentas > & studentai)
```

Definition at line 98 of file List_funkcijos.cpp.

5.21.1.6 List_rusiuotiStudentusPagalVarda()

```
void List_rusiuotiStudentusPagalVarda (
            std::list< List_Studentas > & studentai)
```

Definition at line 87 of file List_funkcijos.cpp.

5.21.1.7 List_skaiciuotiMediana()

```
float List_skaiciuotiMediana (
            std::list< int > & pazymiai)
```

Apskaičiuoja pažymių medianą.

Parameters

pazymiai	Pažymių sąrašas.

Returns

Apskaičiuota mediana.

Definition at line 47 of file List_funkcijos.cpp.

5.21.1.8 List_skaiciuotiVidurki()

```
float List_skaiciuotiVidurki (
            std::list< int > & pazymiai)
```

Apskaičiuoja pažymių vidurkį.

Parameters

pazymiai	Pažymių sąrašas.

Returns

Apskaičiuotas vidurkis.

Definition at line 38 of file List_funkcijos.cpp.

5.21.1.9 List_vykdytiVisusZingsnius()

```
void List_vykdytiVisusZingsnius ()
```

Definition at line 138 of file List_funkcijos.cpp.

5.22 List_funkcijos.cpp

Go to the documentation of this file.

```
00001 #include "List_funkcijos.h"
00002 #include "List_failo_apdorojimas.h"
00003 #include "Vec_funkcijos_papildomos.h"
00004
00005 void List_ivestiStudentoDuomenis(List_Studentas &studentas)
00006 {
00007     std::string vardas, pavarde;
00008     std::list<int> pazymiai;
00009     int egzaminoPazymys;
00010
00011     std::cout « "Vardas: ";
00012     std::cin » vardas;
00013
00014     std::cout « "Pavarde: ";
00015     std::cin » pavarde;
00016
00017     std::cout « "Iveskite pazymius (iveskite -1, kad baigtumete):\n";
00018     while (true)
00019     {
00020         int pazymys = gautiPazymi("Pazymys (arba -1, kad baigtumete): ");
00021         if (pazymys == -1)
00022             break;
00023         pazymiai.push_back(pazymys);
00024     }
00025
00026     egzaminoPazymys = gautiPazymi("Egzamino pazymys: ");
00027     if (egzaminoPazymys == -1)
00028         egzaminoPazymys = 0;
00029
00030     studentas.setVardas(vardas);
00031     studentas.setPavarde(pavarde);
00032     studentas.List_setPazymiai(pazymiai);
00033     studentas.setEgzaminoPazymys(egzaminoPazymys);
00034
00035     studentas.List_skaiciuotiRezultatus();
00036 }
00037
00038 float List_skaiciuotiVidurki(std::list<int> &pazymiai)
00039 {
00040     if (pazymiai.empty())
00041     {
00042         return 0.0f;
00043     }
00044     return std::accumulate(pazymiai.begin(), pazymiai.end(), 0.0f) / pazymiai.size();
00045 }
00046
00047 float List_skaiciuotiMediana(std::list<int> &pazymiai)
00048 {
00049     if (pazymiai.empty())
00050         return 0.0f;
00051
00052     // Create a copy to sort
00053     std::list<int> sortedPazymiai = pazymiai;
00054     sortedPazymiai.sort();
00055
00056     size_t n = sortedPazymiai.size();
00057     auto it = sortedPazymiai.begin();
00058
00059     // Go to middle
00060     std::advance(it, n / 2);
00061
00062     // If even size, take average of two middle elements
00063     if (n % 2 == 0)
00064     {
00065         auto it_prev = it;
00066         --it_prev;
00067         return (*it + *it_prev) / 2.0f;
00068     }
00069     else
```

```
00070     {
00071         return *it;
00072     }
00073 }
00074
00075 // Rūšiuoja studentus pagal pavardę, o jei pavardės vienodos - pagal vardą
00076 void List_rusiuotiStudentus(std::list<List_Studentas> &studentai)
00077 {
00078     studentai.sort([](const List_Studentas &a, const List_Studentas &b)
00079                    {
00080         if (a.getPavarde() == b.getPavarde()) {
00081             return a.getVardas() < b.getVardas();
00082         }
00083         return a.getPavarde() < b.getPavarde(); });
00084 }
00085
00086 // Rūšiuoja studentus pagal vardą, o jei vardai vienodi - pagal pavardę
00087 void List_rusiuotiStudentusPagalVarda(std::list<List_Studentas> &studentai)
00088 {
00089     studentai.sort([](const List_Studentas &a, const List_Studentas &b)
00090                    {
00091         if (a.getVardas() == b.getVardas()) {
00092             return a.getPavarde() < b.getPavarde();
00093         }
00094         return a.getVardas() < b.getVardas(); });
00095 }
00096
00097 // Rūšiuoja studentus pagal pavardę, o jei pavardės vienodos - pagal vardą
00098 void List_rusiuotiStudentusPagalPavarde(std::list<List_Studentas> &studentai)
00099 {
00100     studentai.sort([](const List_Studentas &a, const List_Studentas &b)
00101                    {
00102         if (a.getPavarde() == b.getPavarde()) {
00103             return a.getVardas() < b.getVardas();
00104         }
00105         return a.getPavarde() < b.getPavarde(); });
00106 }
00107
00108 List_Studentas List_generuotiAtsitiktiniStudenta()
00109 {
00110     // Create an empty student object using default constructor
00111     List_Studentas studentas;
00112
00113     // Generuojami atsitiktiniai vardas ir pavarde
00114     studentas.setVardas(generuotiVardaPavarde());
00115     studentas.setPavarde(generuotiVardaPavarde());
00116
00117     // Pre-allocate space for pazymiai to avoid reallocations
00118     int pazymiuKiekis = generuotiSkaiciu(1, 20);
00119
00120     // Generuojami atsitiktiniai pažymiai
00121     std::list<int> pazymiai;
00122     for (int i = 0; i < pazymiuKiekis; i++)
00123     {
00124         pazymiai.push_back(generuotiSkaiciu(0, 10));
00125     }
00126     studentas.List_setPazymiai(pazymiai);
00127
00128     // Generuojamas egzamino pažymys
00129     int egzaminoPazymys = generuotiSkaiciu(0, 10);
00130     studentas.setEgzaminoPazymys(egzaminoPazymys);
00131
00132     // Use the class method to calculate results
00133     studentas.List_skaiciuotiRezultatus();
00134
00135     return studentas;
00136 }
00137
00138 void List_vykdytiVisusZingsnius()
00139 {
00140     vector<int> studentuKiekiai = {1000000, 10000000};
00141
00142     // Atidaryti CSV failą rašymui
00143     ofstream csvFile("performance_data.csv", std::ios::app);
00144     if (!csvFile.is_open())
00145     {
00146         throw runtime_error("Nepavyko atidaryti CSV failo");
00147     }
00148
00149     // Įrašo CSV antraštę, jei failas tuščias
00150     csvFile.seekp(0, std::ios::end);
00151     if (csvFile.tellp() == 0)
00152     {
00153         csvFile « "Testavimo Laikas;Studentu Kiekis;Studentu generavimo laikas;Sugeneruotu duomenu
      skaitymo laikas;Rezultatu irasymo laikas;Rezultatu skaitymo laikas;Rezultatu skaidymo laikas;Skaidymo
      irasymas;Bendras Laikas\n";
00154     }
```

```
00155
00156     for (int kiekis : studentuKiekiai)
00157     {
00158         cout « "Vykdomi zingsniai su " « kiekis « " studentu:\n";
00159
00160         // Gauti dabartinį laiką
00161         auto now = std::chrono::system_clock::now();
00162         auto in_time_t = std::chrono::system_clock::to_time_t(now);
00163         std::stringstream ss;
00164         ss « std::put_time(std::localtime(&in_time_t), "%Y-%m-%d %X");
00165         string timestamp = ss.str();
00166
00167         // Generuoti studentų failą
00168         string studentuFailas = "txt_failai/studentai_" + to_string(kiekis) + ".txt";
00169         auto pradziaGeneravimo = std::chrono::high_resolution_clock::now();
00170         generuotiStudentuFaila(kiekis, studentuFailas);
00171         auto pabaigaGeneravimo = std::chrono::high_resolution_clock::now();
00172         auto trukmeGeneravimo =
      std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaGeneravimo - pradziaGeneravimo);
00173         cout « "Failo su " « kiekis « "studentais generavimas uztruko " « trukmeGeneravimo.count() « "
      ms.\n";
00174
00175         // Skaitomas sugeneruotas failas, apskaičiuoja galutinius rezultatus ir išvedamas į rezultatų
      failą
00176         string rezultatuFailas = "txt_failai/rezultatai_" + to_string(kiekis) + ".txt";
00177         cout « "Skaitomi duomenys ir isvedami i " « rezultatuFailas « "...\n";
00178         long long trukmeSkaitymo, trukmeVidurkio, trukmeIrasymo;
00179         auto pradziaSkaitymo = std::chrono::high_resolution_clock::now();
00180         List_skaitytiIrIsvestiDuomenis(studentuFailas, rezultatuFailas, trukmeSkaitymo,
      trukmeVidurkio, trukmeIrasymo);
00181         auto pabaigaSkaitymo = std::chrono::high_resolution_clock::now();
00182         auto trukmeSkaitymoLaikas =
      std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaSkaitymo - pradziaSkaitymo);
00183
00184         cout « "Skaitymo laikas: " « trukmeSkaitymo « " ms.\n";
00185         cout « "Duomenu isvedimas i " « rezultatuFailas « " uztruko " « trukmeIrasymo « " ms.\n";
00186
00187         // Rezultatų failo padalijimas į išlaikiusius ir neišlaikiusius
00188         string islaikeFailas = "txt_failai/rezultatai_" + to_string(kiekis) + "_islaike.txt";
00189         string neislaikeFailas = "txt_failai/rezultatai_" + to_string(kiekis) + "_neislaike.txt";
00190         cout « "Dalinamas rezultatu failas i islaikiusius ir neislaikiusius...\n";
00191         long long trukmeRezultatuSkaitymo, trukmeRezultatuSkaidymas, trukmeSkaidymoIrasymas;
00192         List_padalintiRezultatuFaila(rezultatuFailas, islaikeFailas, neislaikeFailas,
      trukmeRezultatuSkaitymo, trukmeRezultatuSkaidymas, trukmeSkaidymoIrasymas);
00193         cout « "Rezultatu failo dalinimas uztruko " « trukmeRezultatuSkaitymo +
      trukmeRezultatuSkaidymas + trukmeSkaidymoIrasymas « " ms.\n";
00194
00195         // Skaičuoti bendrą laiką
00196         long long bendrasLaikas = trukmeGeneravimo.count() + trukmeSkaitymo + trukmeIrasymo +
      trukmeRezultatuSkaitymo + trukmeRezultatuSkaidymas + trukmeSkaidymoIrasymas;
00197         cout « "Visi zingsniai su " « kiekis « " studentu baigti. Trukme: " « bendrasLaikas « "
      ms.\n\n";
00198
00199         // Surašyti laikus į CSV failą
00200         csvFile « timestamp « ";"
00201                 « kiekis « ";"
00202                 « trukmeGeneravimo.count() « ";"
00203                 « trukmeSkaitymo « ";"
00204                 « trukmeIrasymo « ";"
00205                 « trukmeRezultatuSkaitymo « ";"
00206                 « trukmeRezultatuSkaidymas « ";"
00207                 « trukmeSkaidymoIrasymas « ";"
00208                 « bendrasLaikas « "\n";
00209     }
00210
00211     csvFile.close();
00212     cout « "Visi zingsniai visiems studentu kiekiams baigti.\n";
00213     cout « "Duomenys issaugoti faile 'performance_data.csv'\n";
00214 }
00215
00216 // Pagrindinė sąrašo (list) programos funkcija
00217 void List_programa()
00218 {
00219
00220     int pasirinkimas;
00221     int failoPasirinkimas;
00222     int studentuKiekis;
00223     bool gerasPasirinkimas = false;
00224
00225     // Meniu
00226     cout « "1. Ivesti duomenis ranka\n";
00227     cout « "2. Automatiskai generuoti duomenis\n";
00228     cout « "3. Nuskaityti duomenis is failo\n";
00229     cout « "4. Sukurti atsitiktiniu studentu failus\n";
00230     cout « "5. Suskaiciuoti rezultatus\n";
00231     cout « "6. Padalinti rezultatu faila i islaikius ir neislaikius\n";
00232     cout « "7. Sugeneruoti 5 atsitiktinius failus\n";
```

```
00233     cout « "8. Vykdyti visus zingsnius visiems studentu kiekiams\n";
00234     cout « "Jusu pasirinkimas: ";
00235 
00236     // Vartotojo pasirinkimo tikrinimas
00237     while (!gerasPasirinkimas)
00238     {
00239         cout « "Iveskite pasirinkima (1-8): ";
00240         cin » pasirinkimas;
00241 
00242         if (cin.fail() || pasirinkimas < 1 || pasirinkimas > 8)
00243         {
00244             cin.clear();
00245             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00246             cout « "Neteisingas pasirinkimas. Prasome ivesti skaiciu nuo 1 iki 8.\n";
00247         }
00248         else
00249         {
00250             gerasPasirinkimas = true;
00251         }
00252     }
00253 
00254     list<List_Studentas> studentai;
00255     try
00256     {
00257         long long trukmeSkaitymo, trukmeVidurkio, trukmeIrasymo;
00258         switch (pasirinkimas)
00259         {
00260         case 1:
00261         {
00262             // Duomenų įvedimas ranka
00263             cout « "Kiek studentu norite irasyti? ";
00264             bool gerasPasirinkimas = false;
00265 
00266             while (!gerasPasirinkimas)
00267             {
00268                 cin » studentuKiekis;
00269                 if (cin.fail() || studentuKiekis < 1)
00270                 {
00271                     cin.clear();
00272                     cin.ignore(numeric_limits<streamsize>::max(), '\n');
00273                     cout « "Klaida: Ivestas neteisingas studentu skaicius. Prasome ivesti skaiciu
      didesni uz 0: ";
00274                 }
00275                 else
00276                 {
00277                     gerasPasirinkimas = true;
00278                 }
00279             }
00280 
00281             for (int i = 0; i < studentuKiekis; i++)
00282             {
00283                 List_Studentas studentas;
00284 
00285                 // Įvedami 1 studento duomenys
00286                 List_ivestiStudentoDuomenis(studentas);
00287 
00288                 // Pridedama studento informacija į list
00289                 studentai.push_back(studentas);
00290 
00291                 // Išvadamas sąrašo adresas atmintyje
00292                 cout « "Studento saraso objektas atmintyje saugomas adresu: " « &studentas « endl;
00293             }
00294 
00295             break;
00296         }
00297 
00298         case 2:
00299         {
00300             // Automatinis duomenų generavimas
00301             cout « "Kiek studentu norite sugeneruoti? ";
00302             bool gerasPasirinkimas = false;
00303 
00304             while (!gerasPasirinkimas)
00305             {
00306                 cin » studentuKiekis;
00307                 if (cin.fail() || studentuKiekis < 1)
00308                 {
00309                     cin.clear();
00310                     cin.ignore(numeric_limits<streamsize>::max(), '\n');
00311                     cout « "Klaida: Ivestas neteisingas studentu skaicius. Prasome ivesti skaiciu
      didesni uz 0: ";
00312                 }
00313                 else
00314                 {
00315                     gerasPasirinkimas = true;
00316                 }
00317             }
```

```
00318
00319             for (int i = 0; i < studentuKiekis; i++)
00320             {
00321                 studentai.push_back(List_generuotiAtsitiktiniStudenta());
00322             }
00323             break;
00324         }
00325
00326         case 3:
00327         {
00328             // Duomenų nuskaitymas iš failo
00329             cout « "Pasirinkite faila (1. studentai10.txt, 2. studentai100.txt, 3. studentai10000.txt,
      4. studentai100000.txt, 5. studentai1000000.txt, 6. studentai10_blog.txt, 7. tuscias.txt): ";
00330             bool gerasPasirinkimas = false;
00331
00332             while (!gerasPasirinkimas)
00333             {
00334                 cin » failoPasirinkimas;
00335                 if (cin.fail() || failoPasirinkimas < 1 || failoPasirinkimas > 7)
00336                 {
00337                     cin.clear();
00338                     cin.ignore(numeric_limits<streamsize>::max(), '\n');
00339                     cout « "Neteisingas pasirinkimas. Prasome ivesti skaiciu nuo 1 iki 7: ";
00340                 }
00341                 else
00342                 {
00343                     gerasPasirinkimas = true;
00344                 }
00345             }
00346
00347             string failoPavadinimas;
00348             switch (failoPasirinkimas)
00349             {
00350             case 1:
00351                 failoPavadinimas = "txt_failai/studentai10.txt";
00352                 break;
00353             case 2:
00354                 failoPavadinimas = "txt_failai/studentai100.txt";
00355                 break;
00356             case 3:
00357                 failoPavadinimas = "txt_failai/studentai10000.txt";
00358                 break;
00359             case 4:
00360                 failoPavadinimas = "txt_failai/studentai100000.txt";
00361                 break;
00362             case 5:
00363                 failoPavadinimas = "txt_failai/studentai1000000.txt";
00364                 break;
00365             case 6:
00366                 failoPavadinimas = "txt_failai/studentai10_blog.txt";
00367                 break;
00368             case 7:
00369                 failoPavadinimas = "txt_failai/tuscias.txt";
00370                 break;
00371             }
00372
00373             List_skaitytiDuomenisIsFailo(failoPavadinimas, studentai, trukmeSkaitymo, trukmeVidurkio);
00374             cout « "Duomenys nuskaityti is " « failoPavadinimas « " per " « trukmeSkaitymo « " ms\n";
00375             break;
00376         }
00377
00378         case 4:
00379         {
00380             // Atsitiktinių studentų failų kūrimas
00381             generuotiFaila();
00382             return;
00383         }
00384         case 5:
00385         {
00386             // Skaičiuoti rezultatus
00387             vector<string> studentuSkaicius = {"_1000", "_10000", "_100000", "_1000000", "_10000000",
      "1000", "10000", "100000", "1000000", "10000000"};
00388             cout « "Pasirinkite rezultatu faila:\n";
00389             cout « "Kodo generuoti duomenys\n1. studentai_1000.txt\n2. studentai_10000.txt\n3.
      studentai_100000.txt\n4. studentai_1000000.txt\n5. studentai_10000000.txt\n";
00390             cout « "Pavyzdiniai duomenys\n6. studentai1000.txt\n7. studentai10000.txt\n8.
      studentai100000.txt\n9. studentai1000000.txt\n10. studentai10000000.txt\n";
00391
00392             int failoPasirinkimas;
00393             cout « "Jusu pasirinkimas: ";
00394             cin » failoPasirinkimas;
00395             if (failoPasirinkimas < 1 || failoPasirinkimas > 10)
00396             {
00397                 throw runtime_error("Neteisingas failo pasirinkimas.");
00398             }
00399
00400             string duomenuFailas = "txt_failai/studentai" + (studentuSkaicius[failoPasirinkimas - 1])
```

```
    + ".txt";
00401             string isvestiesFailoPavadinimas = "txt_failai/rezultatai" +
    (studentuSkaicius[failoPasirinkimas - 1]) + ".txt";
00402
00403             List_skaitytiIrIsvestiDuomenis(duomenuFailas, isvestiesFailoPavadinimas, trukmeSkaitymo,
    trukmeVidurkio, trukmeIrasymo);
00404             cout « "Duomenys nuskaityti is " « duomenuFailas « " per " « trukmeSkaitymo « "ms ir
    isvesti i " « isvestiesFailoPavadinimas « " per " « trukmeIrasymo « " ms.\n";
00405             return;
00406         }
00407
00408         case 6:
00409         {
00410             // Rūšiuoti į išlaikiusius ir neišlaikiusius
00411             vector<string> studentuSkaicius = {"_1000", "_10000", "_100000", "_1000000", "_10000000",
    "1000", "10000", "100000", "1000000", "10000000"};
00412             cout « "Pasirinkite rezultatu faila:\n";
00413             cout « "Kodo generuoti duomenys\n1. rezultatai_1000.txt\n2. rezultatai_10000.txt\n3.
    rezultatai_100000.txt\n4. rezultatai_1000000.txt\n5. rezultatai_10000000.txt\n";
00414             cout « "Pavyzdiniai duomenys\n6. rezultatai1000.txt\n7. rezultatai10000.txt\n8.
    rezultatai100000.txt\n9. rezultatai1000000.txt\n10. rezultatai10000000.txt\n";
00415
00416             int failoPasirinkimas;
00417             cout « "Jusu pasirinkimas: ";
00418             cin » failoPasirinkimas;
00419             if (failoPasirinkimas < 1 || failoPasirinkimas > 10)
00420             {
00421                 throw runtime_error("Neteisingas failo pasirinkimas.");
00422             }
00423
00424             string duomenuFailas = "txt_failai/rezultatai" + (studentuSkaicius[failoPasirinkimas - 1])
    + ".txt";
00425             string islaikiusiuFailoPavadinimas = "txt_failai/rezultatai" +
    (studentuSkaicius[failoPasirinkimas - 1]) + "_islaike.txt";
00426             string neislaikiusiuFailoPavadinimas = "txt_failai/rezultatai" +
    (studentuSkaicius[failoPasirinkimas - 1]) + "_neislaike.txt";
00427
00428             long long trukmeRezultatuSkaitymo, trukmeRezultatuSkaidymas, trukmeSkaidymoIrasymas;
00429             List_padalintiRezultatuFaila(duomenuFailas, islaikiusiuFailoPavadinimas,
    neislaikiusiuFailoPavadinimas, trukmeRezultatuSkaitymo, trukmeRezultatuSkaidymas,
    trukmeSkaidymoIrasymas);
00430             cout « "Rezultatu failas padalintas i " « islaikiusiuFailoPavadinimas « " ir " «
    neislaikiusiuFailoPavadinimas « '\n';
00431             return;
00432         }
00433
00434         case 7:
00435         {
00436             // Generuoti 5 atsitiktinio dydžio studentų failus
00437             generuotiAtsitiktiniusFailus();
00438             return;
00439         }
00440
00441         case 8:
00442         {
00443             // Kelius kartus sukamas kodas, kad sužinoti kiek laiko užtrunka kodas
00444             int kartai;
00445             bool validInput = false;
00446
00447             while (!validInput)
00448             {
00449                 cout « "Kiek kartu norite paleisti funkcija 'vykdytiVisusZingsnius'? ";
00450                 cin » kartai;
00451
00452                 if (cin.fail() || kartai <= 0)
00453                 {
00454                     cin.clear();
00455                     cin.ignore(numeric_limits<streamsize>::max(), '\n');
00456                     cout « "Neteisingas skaicius. Prasome ivesti teigiama skaiciu.\n";
00457                 }
00458                 else
00459                 {
00460                     validInput = true;
00461                 }
00462             }
00463
00464             for (int i = 0; i < kartai; i++)
00465             {
00466                 cout « "Vykdoma " « i + 1 « " karta:\n";
00467                 List_vykdytiVisusZingsnius();
00468             }
00469
00470             return;
00471         }
00472         }
00473
00474         if (studentai.empty())
```

```
00475         {
00476             throw runtime_error("Nera studentu duomenu.");
00477         }
00478 
00479         int pasirinkimas;
00480         cout « "Jeigu norite rusiuoti pagal varda - rasykite 1, jeigu pagal pavarde - 2: ";
00481         cin » pasirinkimas;
00482 
00483         // Rūšiuoja studentus
00484         if (pasirinkimas == 1)
00485         {
00486             List_rusiuotiStudentusPagalVarda(studentai);
00487         }
00488         else if (pasirinkimas == 2)
00489         {
00490             List_rusiuotiStudentusPagalPavarde(studentai);
00491         }
00492         else
00493         {
00494             cout « "Blogas pasirinkimas, rusiuojama pagal varda.\n";
00495             List_rusiuotiStudentusPagalVarda(studentai);
00496         }
00497 
00498         // Spausdina rezultatus
00499         std::cout « std::left « std::setw(16) « "Pavarde"
00500                   « std::setw(16) « "Vardas"
00501                   « std::setw(25) « "Galutinis Vidurkis"
00502                   « " / " « "Galutine Mediana" « '\n';
00503         std::cout « "-------------------------------------------------------------------------------\n";
00504         std::cout « std::fixed « std::setprecision(2);
00505 
00506         // Iterate through the list of students and print their details
00507         for (const List_Studentas &studentas : studentai)
00508         {
00509             std::cout « std::left « std::setw(16) « studentas.getPavarde()
00510                       « std::setw(16) « studentas.getVardas()
00511                       « std::setw(25) « studentas.getGalutinisVidurkis()
00512                       « "   " « studentas.getGalutineMediana() « '\n';
00513         }
00514     }
00515     catch (const exception &e)
00516     {
00517         cout « "Ivyko klaida: " « e.what() « '\n';
00518     }
00519 }
```

5.23 src/List_studentas.cpp File Reference

```
#include "List_funkcijos.h"
#include "Vec_funkcijos.h"
```

5.24 List_studentas.cpp

Go to the documentation of this file.

```
00001 #include "List_funkcijos.h"
00002 #include "Vec_funkcijos.h"
00003 
00004 // Constructors and Destructor
00005 List_Studentas::List_Studentas() : Zmogus()
00006 {
00007     // std::cout « "Default constructor called\n";
00008 }
00009 
00010 List_Studentas::List_Studentas(const std::string &vardas, const std::string &pavarde,
00011                                const std::list<int> &pazymiai, int egzaminoPazymys)
00012     : Zmogus(vardas, pavarde), pazymiai(pazymiai), egzaminoPazymys(egzaminoPazymys)
00013 {
00014     // std::cout « "Parameterized constructor called\n";
00015     List_skaiciuotiRezultatus();
00016 }
00017 
00018 // Destructor
00019 List_Studentas::~List_Studentas()
```

```
00020 {
00021     // std::cout « "Destructor called\n";
00022 }
00023
00024 // Copy Constructor
00025 List_Studentas::List_Studentas(const List_Studentas &other)
00026     : Zmogus(other), pazymiai(other.pazymiai), egzaminoPazymys(other.egzaminoPazymys),
00027       vidurkis(other.vidurkis), mediana(other.mediana),
00028       galutinisVidurkis(other.galutinisVidurkis), galutineMediana(other.galutineMediana)
00029 {
00030     // std::cout « "Copy constructor called\n";
00031 }
00032
00033 // Copy Assignment Operator
00034 List_Studentas &List_Studentas::operator=(const List_Studentas &other)
00035 {
00036     if (this == &other)
00037     {
00038         return *this; // Handle self-assignment
00039     }
00040
00041     // std::cout « "Copy assignment operator called\n";
00042
00043     // Call base class assignment operator
00044     Zmogus::operator=(other);
00045
00046     // Copy fields specific to List_Studentas
00047     pazymiai = other.pazymiai;
00048     egzaminoPazymys = other.egzaminoPazymys;
00049     vidurkis = other.vidurkis;
00050     mediana = other.mediana;
00051     galutinisVidurkis = other.galutinisVidurkis;
00052     galutineMediana = other.galutineMediana;
00053
00054     return *this;
00055 }
00056
00057 // Getters
00058 std::list<int> List_Studentas::getPazymiai() const { return pazymiai; }
00059 int List_Studentas::getEgzaminoPazymys() const { return egzaminoPazymys; }
00060 float List_Studentas::getVidurkis() const { return vidurkis; }
00061 float List_Studentas::getMediana() const { return mediana; }
00062 float List_Studentas::getGalutinisVidurkis() const { return galutinisVidurkis; }
00063 float List_Studentas::getGalutineMediana() const { return galutineMediana; }
00064
00065 // Setters
00066 void List_Studentas::List_setPazymiai(const std::list<int> &pazymiai) { this->pazymiai = pazymiai; }
00067 void List_Studentas::setVidurkis(float vidurkis) { this->vidurkis = vidurkis; }
00068 void List_Studentas::setMediana(float mediana) { this->mediana = mediana; }
00069 void List_Studentas::setEgzaminoPazymys(int egzaminoPazymys) { this->egzaminoPazymys =
      egzaminoPazymys; }
00070 void List_Studentas::setGalutinisVidurkis(float galutinisVidurkis) { this->galutinisVidurkis =
      galutinisVidurkis; }
00071 void List_Studentas::setGalutineMediana(float galutineMediana) { this->galutineMediana =
      galutineMediana; }
00072
00073 // Method to add a grade
00074 void List_Studentas::pridetiPazymi(int pazymys)
00075 {
00076     pazymiai.push_back(pazymys);
00077 }
00078
00079 // Method to calculate results
00080 void List_Studentas::List_skaiciuotiRezultatus()
00081 {
00082     // Calculate average and median
00083     float vidurkis = List_skaiciuotiVidurki(pazymiai);
00084     float mediana = List_skaiciuotiMediana(pazymiai);
00085
00086     setVidurkis(vidurkis);
00087     setMediana(mediana);
00088
00089     // Calculate final grades
00090     const float egzaminoBalas = 0.6f * egzaminoPazymys;
00091     const float vidurkioBalas = 0.4f * vidurkis;
00092     const float medianosBalas = 0.4f * mediana;
00093
00094     setGalutinisVidurkis(vidurkioBalas + egzaminoBalas);
00095     setGalutineMediana(medianosBalas + egzaminoBalas);
00096 }
00097
00098 // Override Zmogus's pure virtual function
00099 void List_Studentas::printInfo() const
00100 {
00101     std::cout « "Studentas: " « getVardas() « " " « getPavarde() « "\n"
00102               « "Egzamino pazymys: " « egzaminoPazymys « "\n"
00103               « "Vidurkis: " « vidurkis « "\n"
```

```
00104               « "Mediana: " « mediana « "\n"
00105               « "Galutinis Vidurkis: " « galutinisVidurkis « "\n"
00106               « "Galutine Mediana: " « galutineMediana « "\n";
00107 }
```

5.25 src/main.cpp File Reference

```
#include "Vec_funkcijos.h"
#include "List_funkcijos.h"
```

Functions

- int main ()

5.25.1 Function Documentation

5.25.1.1 main()

```
int main ()
```

Definition at line 4 of file main.cpp.

5.26 main.cpp

Go to the documentation of this file.

```
00001 #include "Vec_funkcijos.h"
00002 #include "List_funkcijos.h"
00003
00004 int main()
00005 {
00006     char pabaiga = 'N';
00007     char vec_li;
00008
00009     do
00010     {
00011         // Vartotojas įveda konteinerio tipą (vektorius arba sąrašas)
00012         do
00013         {
00014             cout « "Ar norite naudoti Vector ar List? Jeigu Vector - rasykite V, jeigu List - rasykite
      L: ";
00015             cin » vec_li;
00016             vec_li = toupper(vec_li);
00017
00018             if (vec_li != 'V' && vec_li != 'L')
00019             {
00020                 cout « "Neteisingas pasirinkimas. Prasome ivesti V arba L: ";
00021                 cin.clear();
00022                 cin.ignore(numeric_limits<streamsize>::max(), '\n');
00023             }
00024         } while (vec_li != 'V' && vec_li != 'L');
00025
00026         // Vykdyti atitinkamą funkciją pagal naudotojo įvestį
00027         if (vec_li == 'V')
00028         {
00029             Vec_programa();
00030         }
00031         else
00032         {
00033             List_programa();
00034         }
00035
00036         // Vartotojo įvestis programos užbaigimui
```

```
00037         do
00038         {
00039             cout « "Ar norite uzdaryti programa? Jeigu taip - rasykite T, jeigu ne - rasykite N: ";
00040             cin » pabaiga;
00041             pabaiga = toupper(pabaiga);
00042 
00043             if (pabaiga != 'T' && pabaiga != 'N')
00044             {
00045                 cout « "Neteisingas pasirinkimas. Prasome ivesti T arba N: ";
00046                 cin.clear();
00047                 cin.ignore(numeric_limits<streamsize>::max(), '\n');
00048             }
00049         } while (pabaiga != 'T' && pabaiga != 'N');
00050 
00051         cout « "\n";
00052 
00053     } while (pabaiga != 'T');
00054 
00055     return 0;
00056 }
```

5.27 src/studentas.cpp File Reference

```
#include "Vec_funkcijos.h"
#include "Vec_failo_apdorojimas.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const Studentas &studentas)
- int gautiPazymi (std::istream &is, const std::string &klausimas)
- std::istream & operator>> (std::istream &is, Studentas &studentas)
- std::ifstream & operator>> (std::ifstream &failas, std::vector< Studentas > &studentai)

5.27.1 Function Documentation

5.27.1.1 gautiPazymi()

```
int gautiPazymi (
            std::istream & is,
            const std::string & klausimas)
```

Definition at line 103 of file studentas.cpp.

5.27.1.2 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Studentas & studentas)
```

Definition at line 92 of file studentas.cpp.

5.27.1.3 operator>>() [1/2]

```
std::ifstream & operator>> (
            std::ifstream & failas,
            std::vector< Studentas > & studentai)
```

Definition at line 274 of file studentas.cpp.

5.27.1.4 operator>>() [2/2]

```
std::istream & operator>> (
            std::istream & is,
            Studentas & studentas)
```

Definition at line 133 of file studentas.cpp.

5.28 studentas.cpp

Go to the documentation of this file.

```
00001 #include "Vec_funkcijos.h"
00002 #include "Vec_failo_apdorojimas.h"
00003
00004 // Default Constructor
00005 Studentas::Studentas() : Zmogus() {}
00006
00007 // Parameterized Constructor
00008 Studentas::Studentas(const std::string &vardas, const std::string &pavarde, const std::vector<int>
      &pazymiai, int egzaminoPazymys)
00009     : Zmogus(vardas, pavarde), pazymiai(pazymiai), egzaminoPazymys(egzaminoPazymys) {
00010     // std::cout « "Parameterized constructor called\n";
00011     skaiciuotiRezultatus();
00012 }
00013
00014 // Destructor
00015 Studentas::~Studentas()
00016 {
00017     // Clear the vector to release its memory
00018     pazymiai.clear();
00019     // Reset other members to default values
00020     vardas = "";
00021     pavarde = "";
00022     egzaminoPazymys = 0;
00023     vidurkis = 0.0f;
00024     mediana = 0.0f;
00025     galutinisVidurkis = 0.0f;
00026     galutineMediana = 0.0f;
00027
00028     // Debug message to indicate destruction
00029     // std::cout « "Destructor called: Studentas object cleaned up.\n";
00030 }
00031
00032 // Copy Constructor
00033 Studentas::Studentas(const Studentas &other)
00034     : Zmogus(other), pazymiai(other.pazymiai), egzaminoPazymys(other.egzaminoPazymys),
00035       vidurkis(other.vidurkis), mediana(other.mediana),
00036       galutinisVidurkis(other.galutinisVidurkis), galutineMediana(other.galutineMediana)
00037 {
00038     // Explicit copy of all members
00039     // std::cout « "Copy constructor called.\n";
00040 }
00041
00042 // Copy Assignment Operator
00043 Studentas &Studentas::operator=(const Studentas &other) {
00044     if (this != &other) {
00045         Zmogus::operator=(other);
00046         pazymiai = other.pazymiai;
00047         egzaminoPazymys = other.egzaminoPazymys;
00048         vidurkis = other.vidurkis;
00049         mediana = other.mediana;
00050         galutinisVidurkis = other.galutinisVidurkis;
00051         galutineMediana = other.galutineMediana;
```

```
00052     }
00053     // std::cout « "Copy assignment operator called.\n";
00054     return *this;
00055 }
00056
00057 // Getters (unchanged)
00058 std::vector<int> Studentas::getPazymiai() const { return pazymiai; }
00059 int Studentas::getEgzaminoPazymys() const { return egzaminoPazymys; }
00060 float Studentas::getVidurkis() const { return vidurkis; }
00061 float Studentas::getMediana() const { return mediana; }
00062 float Studentas::getGalutinisVidurkis() const { return galutinisVidurkis; }
00063 float Studentas::getGalutineMediana() const { return galutineMediana; }
00064
00065 // Corrected Setters
00066 void Studentas::setPazymiai(const std::vector<int> &pazymiai) { this->pazymiai = pazymiai; }
00067 void Studentas::setVidurkis(float vidurkis) { this->vidurkis = vidurkis; }
00068 void Studentas::setMediana(float mediana) { this->mediana = mediana; }
00069 void Studentas::setEgzaminoPazymys(int egzaminoPazymys) { this->egzaminoPazymys = egzaminoPazymys; }
00070 void Studentas::setGalutinisVidurkis(float galutinisVidurkis) { this->galutinisVidurkis =
      galutinisVidurkis; }
00071 void Studentas::setGalutineMediana(float galutineMediana) { this->galutineMediana = galutineMediana; }
00072
00073 void Studentas::skaiciuotiRezultatus()
00074 {
00075     // Implement calculation of vidurkis, mediana, galutinisVidurkis, galutineMediana
00076     if (!pazymiai.empty())
00077     {
00078         vidurkis = skaiciuotiVidurki(pazymiai);
00079         mediana = skaiciuotiMediana(pazymiai);
00080
00081         // Example calculation of final grades (adjust as needed)
00082         galutinisVidurkis = 0.4 * vidurkis + 0.6 * egzaminoPazymys;
00083         galutineMediana = 0.4 * mediana + 0.6 * egzaminoPazymys;
00084     }
00085 }
00086
00087 void Studentas::pridetiPazymi(int pazymys)
00088 {
00089     pazymiai.push_back(pazymys);
00090 }
00091
00092 std::ostream &operator«(std::ostream &os, const Studentas &studentas)
00093 {
00094
00095     os « std::left « std::setw(16) « studentas.getPavarde()
00096        « std::setw(16) « studentas.getVardas()
00097        « std::setw(24) « std::fixed « std::setprecision(2)
00098        « studentas.getGalutinisVidurkis()
00099        « studentas.getGalutineMediana();
00100     return os;
00101 }
00102
00103 int gautiPazymi(std::istream &is, const std::string &klausimas)
00104 {
00105     while (true)
00106     {
00107         std::string skaicius;
00108         std::cout « klausimas;
00109         is » skaicius;
00110
00111         if (skaicius == "-1")
00112             return -1; // End input
00113
00114         try
00115         {
00116             int pazymys = std::stoi(skaicius);
00117             if (pazymys >= 0 && pazymys <= 10)
00118             {
00119                 return pazymys;
00120             }
00121             else
00122             {
00123                 std::cout « "Klaida: pazymys turi buti tarp 0 ir 10.\n";
00124             }
00125         }
00126         catch (const std::invalid_argument &)
00127         {
00128             std::cout « "Klaida: iveskite teisinga skaiciu.\n";
00129         }
00130     }
00131 }
00132
00133 std::istream &operator»(std::istream &is, Studentas &studentas)
00134 {
00135     if (is.rdbuf() == std::cin.rdbuf())
00136     {
00137         // Handle manual input
```

```
00138         std::string vardas, pavarde;
00139         std::vector<int> pazymiai;
00140         int egzaminoPazymys;
00141 
00142         std::cout « "Vardas: ";
00143         is » vardas;
00144         std::cout « "Pavarde: ";
00145         is » pavarde;
00146 
00147         std::cout « "Iveskite pazymius (iveskite -1, kad baigtumete):\n";
00148         while (true)
00149         {
00150             int pazymys;
00151             is » pazymys;
00152 
00153             // Check if the input was successful
00154             if (is.fail())
00155             {
00156                 // Clear the error state
00157                 is.clear();
00158                 // Ignore the invalid input
00159                 is.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00160                 std::cout « "Klaida: prasome ivesti skaiciu.\n";
00161                 continue; // Retry the input
00162             }
00163 
00164             if (pazymys == -1)
00165                 break;
00166             if (pazymys < 0 || pazymys > 10)
00167             {
00168                 std::cout « "Klaida: pazymys turi buti tarp 0 ir 10.\n";
00169             }
00170             else
00171             {
00172                 pazymiai.push_back(pazymys);
00173             }
00174         }
00175 
00176         std::cout « "Egzamino pazymys: ";
00177         while (true)
00178         {
00179             is » egzaminoPazymys;
00180 
00181             // Check if the input was successful
00182             if (is.fail())
00183             {
00184                 // Clear the error state
00185                 is.clear();
00186                 // Ignore the invalid input
00187                 is.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00188                 std::cout « "Klaida: prasome ivesti teisinga skaiciu egzamino pazymys.\n";
00189             }
00190             else
00191             {
00192                 // Break the loop if input is valid
00193                 break;
00194             }
00195         }
00196 
00197         // After getting a valid egzaminoPazymys, set values in the Studentas object
00198         studentas.setVardas(vardas);
00199         studentas.setPavarde(pavarde);
00200         studentas.setPazymiai(pazymiai);
00201         studentas.setEgzaminoPazymys(egzaminoPazymys);
00202         studentas.skaiciuotiRezultatus();
00203     }
00204     else
00205     {
00206         // Handle fixed-width file parsing (same as file input logic)
00207         std::string line;
00208         if (!std::getline(is, line))
00209         {
00210             return is; // Return early if no more lines
00211         }
00212 
00213         if (line.length() < 52)
00214         {
00215             throw std::runtime_error("Netinkamas eilutes ilgis");
00216         }
00217 
00218         // Extract fixed-width fields
00219         std::string vardas = line.substr(0, 16);
00220         std::string pavarde = line.substr(16, 32);
00221         auto trim = [](std::string &str)
00222         {
00223             str.erase(str.begin(), std::find_if(str.begin(), str.end(), [](unsigned char ch)
00224                                                 { return !std::isspace(ch); }));
```

```
00225             str.erase(std::find_if(str.rbegin(), str.rend(), [](unsigned char ch)
00226                                       { return !std::isspace(ch); })
00227                           .base(),
00228                       str.end());
00229         };
00230         trim(vardas);
00231         trim(pavarde);
00232 
00233         // Parse grades
00234         std::vector<int> pazymiai;
00235         size_t pozicija = 52;
00236         while (pozicija < line.length())
00237         {
00238             // Skip whitespace
00239             while (pozicija < line.length() && std::isspace(line[pozicija]))
00240                 pozicija++;
00241             if (pozicija >= line.length())
00242                 break;
00243 
00244             // Parse grade
00245             int grade = 0;
00246             bool tinkamas = true;
00247             while (pozicija < line.length() && std::isdigit(line[pozicija]))
00248             {
00249                 grade = grade * 10 + (line[pozicija] - '0');
00250                 pozicija++;
00251             }
00252 
00253             if (grade < 0 || grade > 10)
00254             {
00255                 tinkamas = false;
00256                 throw std::runtime_error("Netinkamas pazymys");
00257             }
00258             pazymiai.push_back(grade);
00259         }
00260 
00261         // Set values in the Studentas object
00262         studentas.setVardas(vardas);
00263         studentas.setPavarde(pavarde);
00264         studentas.setPazymiai(pazymiai);
00265         studentas.skaiciuotiRezultatus();
00266 
00267         return is;
00268     }
00269 
00270     return is;
00271 }
00272 
00273 // Global function to overload stream input operator for file reading
00274 std::ifstream &operator»(std::ifstream &failas, std::vector<Studentas> &studentai)
00275 {
00276     if (!failas)
00277     {
00278         throw std::runtime_error("Failas negali būti atidarytas.");
00279     }
00280 
00281     std::string buffer;
00282     buffer.reserve(1048576); // Buferio dydis baitais
00283 
00284     // Praleidžia antraštę
00285     std::getline(failas, buffer);
00286 
00287     while (std::getline(failas, buffer))
00288     {
00289         if (buffer.length() < 52)
00290         {
00291             throw std::runtime_error("Netinkamas eilutes ilgis");
00292         }
00293 
00294         Studentas studentas;
00295 
00296         // Trim funkcija
00297         auto trim = [](std::string &str)
00298         {
00299             str.erase(str.begin(), std::find_if(str.begin(), str.end(), [](unsigned char ch)
00300                                                 { return !std::isspace(ch); }));
00301             str.erase(std::find_if(str.rbegin(), str.rend(), [](unsigned char ch)
00302                                       { return !std::isspace(ch); })
00303                           .base(),
00304                       str.end());
00305         };
00306 
00307         // Vardas ir pavardė
00308         std::string vardas = buffer.substr(0, 16);
00309         std::string pavarde = buffer.substr(16, 32);
00310         trim(vardas);
00311         trim(pavarde);
```

```
00312
00313         studentas.setVardas(vardas);
00314         studentas.setPavarde(pavarde);
00315
00316         // Pažymių skaitymas
00317         size_t pozicija = 52;
00318         bool tinkamiPazymiai = true;
00319         std::vector<int> pazymiai;
00320
00321         while (pozicija < buffer.length())
00322         {
00323             // Praleidžia whitespace
00324             while (pozicija < buffer.length() && std::isspace(buffer[pozicija]))
00325                 pozicija++;
00326             if (pozicija >= buffer.length())
00327                 break;
00328
00329             int grade = 0;
00330             bool tinkamas = true;
00331
00332             if (std::isdigit(buffer[pozicija]))
00333             {
00334                 while (pozicija < buffer.length() && std::isdigit(buffer[pozicija]))
00335                 {
00336                     grade = grade * 10 + (buffer[pozicija] - '0');
00337                     pozicija++;
00338                 }
00339
00340                 if (grade < 0 || grade > 10)
00341                 {
00342                     tinkamas = false;
00343                 }
00344             }
00345             else
00346             {
00347                 tinkamas = false;
00348                 pozicija++;
00349             }
00350
00351             if (tinkamas)
00352             {
00353                 pazymiai.push_back(grade);
00354             }
00355             else
00356             {
00357                 tinkamiPazymiai = false;
00358                 break;
00359             }
00360
00361             // Praleidžia whitespace
00362             while (pozicija < buffer.length() && std::isspace(buffer[pozicija]))
00363                 pozicija++;
00364         }
00365
00366         studentas.setPazymiai(pazymiai);
00367
00368         // Paskaičiuoja rezultatus
00369         skaiciuotiIsFailo(studentas, tinkamiPazymiai, studentai);
00370     }
00371
00372     return failas;
00373 }
00374
00375 void Studentas::printInfo() const {
00376     std::cout « "Vardas: " « vardas « ", Pavarde: " « pavarde « std::endl;
00377     std::cout « "Egzamino Pazymys: " « egzaminoPazymys « std::endl;
00378     std::cout « "Vidurkis: " « vidurkis « ", Mediana: " « mediana « std::endl;
00379     std::cout « "Galutinis Vidurkis: " « galutinisVidurkis « ", Galutine Mediana: " « galutineMediana
      « std::endl;
00380     std::cout « "Pazymiai: ";
00381     for (const auto &pazymys : pazymiai) {
00382         std::cout « pazymys « " ";
00383     }
00384     std::cout « std::endl;
00385 }
```

5.29 src/test_Studentas.cpp File Reference

```
#include <gtest/gtest.h>
#include "Vec_funkcijos.h"
```

Functions

- TEST (StudentasTest, DefaultConstructor)
- TEST (StudentasTest, ParameterizedConstructor)

5.29.1 Function Documentation

5.29.1.1 TEST() [1/2]

```
TEST (
            StudentasTest ,
            DefaultConstructor )
```

Definition at line 5 of file test_Studentas.cpp.

5.29.1.2 TEST() [2/2]

```
TEST (
            StudentasTest ,
            ParameterizedConstructor )
```

Definition at line 13 of file test_Studentas.cpp.

5.30 test_Studentas.cpp

Go to the documentation of this file.

```
00001 #include <gtest/gtest.h>
00002 #include "Vec_funkcijos.h"
00003
00004 // Test default constructor
00005 TEST(StudentasTest, DefaultConstructor) {
00006     Studentas student;
00007     EXPECT_EQ(student.getVardas(), "");   // Default name
00008     EXPECT_EQ(student.getPavarde(), ""); // Default surname
00009     EXPECT_EQ(student.getEgzaminoPazymys(), 0); // Default grade
00010 }
00011
00012 // Test parameterized constructor
00013 TEST(StudentasTest, ParameterizedConstructor) {
00014     std::vector<int> grades = {10, 9, 8};
00015     Studentas student("Jonas", "Jonaitis", grades, 10);
00016     EXPECT_EQ(student.getVardas(), "Jonas");
00017     EXPECT_EQ(student.getPavarde(), "Jonaitis");
00018     EXPECT_EQ(student.getEgzaminoPazymys(), 10);
00019 }
```

5.31 src/Vec_failo_apdorojimas.cpp File Reference

```
#include "Vec_failo_apdorojimas.h"
```

Functions

- void skaiciuotiIsFailo (Studentas &studentas, bool tinkamiPazymiai, std::vector< Studentas > &studentai)
- void skaitytiDuomenisIsFailo (const std::string &failoPavadinimas, std::vector< Studentas > &studentai, long long &trukmeSkaitymo, long long &trukmeVidurkio)

  Nuskaito studentų duomenis iš failo.
- void skaitytiIrIsvestiDuomenis (const std::string &ivestiesFailoPavadinimas, const std::string &irasymoFailoPavadinimas, long long &trukmeSkaitymo, long long &trukmeVidurkio, long long &trukmeIrasymo)
- void padalintiRezultatuFaila (const std::string &ivestiesFailoPavadinimas, const std::string &islaikiusiuFailoPavadinimas, const std::string &neislaikiusiuFailoPavadinimas, long long &laikasSkaitymo, long long &rusiavimoLaikas, long long &laikasRasymo)

5.31.1 Function Documentation

5.31.1.1 padalintiRezultatuFaila()

```
void padalintiRezultatuFaila (
            const std::string & ivestiesFailoPavadinimas,
            const std::string & islaikiusiuFailoPavadinimas,
            const std::string & neislaikiusiuFailoPavadinimas,
            long long & laikasSkaitymo,
            long long & rusiavimoLaikas,
            long long & laikasRasymo)
```

Definition at line 99 of file Vec_failo_apdorojimas.cpp.

5.31.1.2 skaiciuotiIsFailo()

```
void skaiciuotiIsFailo (
            Studentas & studentas,
            bool tinkamiPazymiai,
            std::vector< Studentas > & studentai)
```

Definition at line 3 of file Vec_failo_apdorojimas.cpp.

5.31.1.3 skaitytiDuomenisIsFailo()

```
void skaitytiDuomenisIsFailo (
            const std::string & failoPavadinimas,
            std::vector< Studentas > & studentai,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio)
```

Nuskaito studentų duomenis iš failo.

Parameters

failoPavadinimas	Įvesties failo pavadinimas.
studentai	Vektorius, kuriame saugomi studentų duomenys.
trukmeSkaitymo	Skaitymo trukmė milisekundėmis.
trukmeVidurkio	Vidurkių skaičiavimo trukmė milisekundėmis.

Definition at line 30 of file Vec_failo_apdorojimas.cpp.

5.31.1.4 skaitytiIrIsvestiDuomenis()

```
void skaitytiIrIsvestiDuomenis (
             const std::string & ivestiesFailoPavadinimas,
             const std::string & irasymoFailoPavadinimas,
             long long & trukmeSkaitymo,
             long long & trukmeVidurkio,
             long long & trukmeIrasymo)
```

Definition at line 51 of file Vec_failo_apdorojimas.cpp.

5.32 Vec_failo_apdorojimas.cpp

Go to the documentation of this file.

```
00001 #include "Vec_failo_apdorojimas.h"
00002
00003 void skaiciuotiIsFailo(Studentas &studentas, bool tinkamiPazymiai, std::vector<Studentas> &studentai)
00004 {
00005     if (tinkamiPazymiai && !studentas.getPazymiai().empty())
00006     {
00007         // Extract and set the exam score
00008         std::vector<int> pazymiai = studentas.getPazymiai();
00009         int egzaminoPazymys = pazymiai.back();
00010         pazymiai.pop_back();
00011
00012         studentas.setEgzaminoPazymys(egzaminoPazymys);
00013         studentas.setPazymiai(pazymiai);
00014
00015         // Calculate results
00016         studentas.skaiciuotiRezultatus();
00017
00018         // Add the student to the vector
00019         studentai.push_back(std::move(studentas));
00020     }
00021     else
00022     {
00023         std::cout « "Klaida: truksta pazymiu studentui "
00024                   « studentas.getVardas() « " "
00025                   « studentas.getPavarde() « "\n";
00026     }
00027 }
00028
00029 // Modified skaitytiDuomenisIsFailo function to use the new operator
00030 void skaitytiDuomenisIsFailo(
00031     const std::string &failoPavadinimas,
00032     std::vector<Studentas> &studentai,
00033     long long &trukmeSkaitymo,
00034     long long &trukmeVidurkio)
00035 {
00036     auto pradziaSkaitymo = std::chrono::high_resolution_clock::now();
00037
00038     std::ifstream failas(failoPavadinimas, std::ios::in | std::ios::binary);
00039     if (!failas)
00040     {
00041         throw std::runtime_error("Failo " + failoPavadinimas + " nera.");
00042     }
00043
00044     failas » studentai;
00045
00046     auto pabaigaSkaitymo = std::chrono::high_resolution_clock::now();
00047     trukmeSkaitymo = std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaSkaitymo -
      pradziaSkaitymo).count();
00048     trukmeVidurkio = 0;
00049 }
00050
00051 void skaitytiIrIsvestiDuomenis(
00052     const std::string &ivestiesFailoPavadinimas,
00053     const std::string &irasymoFailoPavadinimas,
00054     long long &trukmeSkaitymo,
00055     long long &trukmeVidurkio,
00056     long long &trukmeIrasymo)
00057 {
00058     std::vector<Studentas> studentai;
00059
00060     // Skaitymas iš failo
00061     skaitytiDuomenisIsFailo(ivestiesFailoPavadinimas, studentai, trukmeSkaitymo, trukmeVidurkio);
```

```
00062
00063     auto pradziaIrasimo = std::chrono::high_resolution_clock::now();
00064
00065     // Naudoja stringstream buferiui
00066     std::ostringstream buffer;
00067
00068     // Įrašo antraštę į buferį
00069     buffer « std::left « std::setw(16) « "Pavarde"
00070            « std::setw(16) « "Vardas"
00071            « std::setw(25) « "Galutinis Vidurkis"
00072            « "Galutine Mediana\n";
00073     buffer « std::string(70, '-') « "\n";
00074
00075     for (const auto &studentas : studentai)
00076     {
00077         buffer « std::left « std::setw(16) « studentas.getPavarde()
00078                « std::setw(16) « studentas.getVardas()
00079                « std::setw(25) « std::fixed « std::setprecision(2) « studentas.getGalutinisVidurkis()
00080                « std::fixed « std::setprecision(2) « studentas.getGalutineMediana()
00081                « "\n";
00082     }
00083
00084     // Atidaro failą įrašymui
00085     std::ofstream irasymoFailas(irasymoFailoPavadinimas, std::ios::out | std::ios::binary);
00086     if (!irasymoFailas)
00087     {
00088         throw std::runtime_error("Nepavyko atidaryti isvesties failo " + irasymoFailoPavadinimas);
00089     }
00090
00091     // Įrašo visą buferį vienu metu
00092     irasymoFailas « buffer.str();
00093     irasymoFailas.close();
00094
00095     auto pabaigaIrasimo = std::chrono::high_resolution_clock::now();
00096     trukmeIrasymo = std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaIrasimo -
      pradziaIrasimo).count();
00097 }
00098
00099 void padalintiRezultatuFaila(const std::string &ivestiesFailoPavadinimas,
00100                              const std::string &islaikiusiuFailoPavadinimas,
00101                              const std::string &neislaikiusiuFailoPavadinimas,
00102                              long long &laikasSkaitymo,
00103                              long long &rusiavimoLaikas,
00104                              long long &laikasRasymo)
00105 {
00106     auto pradziaSkaitymo = std::chrono::high_resolution_clock::now();
00107
00108     // Open input file with optimized flags
00109     std::ifstream ivestiesFailas(ivestiesFailoPavadinimas, std::ios::in | std::ios::binary |
      std::ios::ate);
00110     if (!ivestiesFailas)
00111     {
00112         throw std::runtime_error("Nepavyko atidaryti ivesties failo: " + ivestiesFailoPavadinimas);
00113     }
00114
00115     // Read file content into string
00116     const auto failoDydis = ivestiesFailas.tellg();
00117     std::string failoTurinys(failoDydis, '\0');
00118     ivestiesFailas.seekg(0);
00119     ivestiesFailas.read(&failoTurinys[0], failoDydis);
00120     ivestiesFailas.close();
00121
00122     // Prepare output files
00123     std::ofstream islaikiusiuFailas(islaikiusiuFailoPavadinimas, std::ios::out | std::ios::binary);
00124     std::ofstream neislaikiusiuFailas(neislaikiusiuFailoPavadinimas, std::ios::out |
      std::ios::binary);
00125     if (!islaikiusiuFailas || !neislaikiusiuFailas)
00126     {
00127         throw std::runtime_error("Nepavyko atidaryti išvesties failų");
00128     }
00129
00130     // Read header
00131     std::istringstream inputStream(failoTurinys);
00132     std::string headerLine;
00133     std::getline(inputStream, headerLine);
00134
00135     islaikiusiuFailas « headerLine « "\n";
00136     neislaikiusiuFailas « headerLine « "\n";
00137
00138     std::vector<Studentas> studentai, vargsiukai;
00139     std::string line;
00140
00141     // Parse student data
00142     while (std::getline(inputStream, line))
00143     {
00144         if (line.empty())
00145             continue;
```

```
00146
00147         std::istringstream lineStream(line);
00148         Studentas studentas;
00149
00150         std::string vardas, pavarde;
00151         float galutinisVidurkis, galutineMediana;
00152
00153         lineStream » pavarde » vardas » galutinisVidurkis » galutineMediana;
00154
00155         studentas.setPavarde(pavarde);
00156         studentas.setVardas(vardas);
00157         studentas.setGalutinisVidurkis(galutinisVidurkis);
00158         studentas.setGalutineMediana(galutineMediana);
00159
00160         studentai.push_back(studentas);
00161     }
00162
00163     auto pabaigaSkaitymo = std::chrono::high_resolution_clock::now();
00164     laikasSkaitymo = std::chrono::duration_cast<std::chrono::milliseconds>(
00165                          pabaigaSkaitymo - pradziaSkaitymo)
00166                          .count();
00167
00168     // Sort and partition students
00169     auto pradetiRusiavima = std::chrono::high_resolution_clock::now();
00170
00171     auto partitionPoint = std::partition(studentai.begin(), studentai.end(),
00172                                          [](const Studentas &studentas)
00173                                          {
00174                                              return studentas.getGalutinisVidurkis() >= 5.0f;
00175                                          });
00176
00177     vargsiukai.insert(vargsiukai.end(), partitionPoint, studentai.end());
00178     studentai.erase(partitionPoint, studentai.end());
00179
00180     // Sort both groups
00181     std::sort(studentai.begin(), studentai.end(), [](const Studentas &a, const Studentas &b)
00182               { return a.getGalutinisVidurkis() > b.getGalutinisVidurkis(); });
00183
00184     std::sort(vargsiukai.begin(), vargsiukai.end(), [](const Studentas &a, const Studentas &b)
00185               { return a.getGalutinisVidurkis() > b.getGalutinisVidurkis(); });
00186
00187     auto pabaigaRusiavimo = std::chrono::high_resolution_clock::now();
00188     rusiavimoLaikas = std::chrono::duration_cast<std::chrono::milliseconds>(
00189                           pabaigaRusiavimo - pradetiRusiavima)
00190                           .count();
00191
00192     // Write results to files
00193     auto pradetiRasyma = std::chrono::high_resolution_clock::now();
00194
00195     for (const auto &studentas : studentai)
00196     {
00197         islaikiusiuFailas « studentas « std::endl;
00198     }
00199
00200     for (const auto &studentas : vargsiukai)
00201     {
00202         neislaikiusiuFailas « studentas « std::endl;
00203     }
00204
00205     auto pabaigaRasymo = std::chrono::high_resolution_clock::now();
00206     laikasRasymo = std::chrono::duration_cast<std::chrono::milliseconds>(
00207                        pabaigaRasymo - pradetiRasyma)
00208                        .count();
00209
00210     islaikiusiuFailas.close();
00211     neislaikiusiuFailas.close();
00212 }
```

5.33 src/Vec_funkcijos.cpp File Reference

```
#include "Vec_funkcijos.h"
#include "Vec_funkcijos_papildomos.h"
#include "Vec_failo_apdorojimas.h"
```

Functions

- void Vec_programa ()

  Vykdo Vec programos funkcionalumą.

5.33.1 Function Documentation

5.33.1.1 Vec_programa()

```
void Vec_programa ()
```

Vykdo Vec programos funkcionalumą.

Definition at line 7 of file Vec_funkcijos.cpp.

5.34 Vec_funkcijos.cpp

Go to the documentation of this file.

```
00001 // Pagalbinės funkcijos
00002 #include "Vec_funkcijos.h"
00003 #include "Vec_funkcijos_papildomos.h"
00004 #include "Vec_failo_apdorojimas.h"
00005
00006 // Pagrindinė vektoriaus programos funkcija
00007 void Vec_programa()
00008 {
00009
00010     // Vėliau naudojami kintamieji
00011     int pasirinkimas;
00012     int failoPasirinkimas;
00013     int studentuKiekis;
00014     bool gerasPasirinkimas = false;
00015
00016     // Meniu
00017     cout « "1. Ivesti duomenis ranka\n"
00018          « "2. Automatiskai generuoti duomenis\n"
00019          « "3. Nuskaityti duomenis is failo\n"
00020          « "4. Sukurti atsitiktiniu studentu failus\n"
00021          « "5. Suskaiciuoti rezultatus\n"
00022          « "6. Padalinti rezultatu faila i islaikius ir neislaikius\n"
00023          « "7. Sugeneruoti 5 atsitiktinius failus\n"
00024          « "8. Vykdyti visus zingsnius visiems studentu kiekiams\n"
00025          « "9. Abstrakti klases Zmogus, jos objektu kurimo neleidimo parodymas\n"
00026          « "Jusu pasirinkimas: ";
00027
00028     // Vartotojo pasirinkimo tikrinimas
00029     while (!gerasPasirinkimas)
00030     {
00031         cin.clear();
00032         cin.ignore(numeric_limits<streamsize>::max(), '\n');
00033
00034         cout « "Iveskite pasirinkima (1-9): ";
00035         cin » pasirinkimas;
00036
00037         if (cin.fail() || pasirinkimas < 1 || pasirinkimas > 9)
00038         {
00039             cin.clear();
00040             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00041             cout « "Neteisingas pasirinkimas. Prasome ivesti skaiciu nuo 1 iki 8.\n";
00042         }
00043         else
00044         {
00045             gerasPasirinkimas = true;
00046         }
00047     }
00048
00049     vector<Studentas> studentai;
00050     try
00051     {
00052         long long trukmeSkaitymo = 0, trukmeVidurkio = 0, trukmeIrasymo = 0;
00053         switch (pasirinkimas)
00054         {
00055         case 1:
00056         {
00057             // Duomenų įvedimas ranka
00058             ivestiDuomenisRanka(studentai);
00059             break;
00060         }
00061
00062         case 2:
00063         {
```

```
00064             // Automatinis duomenų generavimas
00065             automatiskaiGeneruotiDuomenis(studentai);
00066             break;
00067         }
00068
00069         case 3:
00070         {
00071             // Duomenų nuskaitymas iš failo
00072             nuskaitytiDuomenisIsFailo(studentai, trukmeSkaitymo, trukmeVidurkio);
00073             break;
00074         }
00075
00076         case 4:
00077         {
00078             // Atsitiktinių studentų failų kūrimas
00079             generuotiFaila();
00080             return;
00081         }
00082         case 5:
00083         {
00084             // Skaičiuoti rezultatus
00085             skaiciuotiRezultatus(trukmeSkaitymo, trukmeVidurkio, trukmeIrasymo);
00086             break;
00087         }
00088         case 6:
00089         {
00090             // Rūšiuoti į išlaikiusius ir neišlaikiusius
00091             long long trukmeRezultatuSkaitymo, trukmeRezultatuSkaidymas, trukmeSkaidymoIrasymas;
00092             rusiuotiRezultatus(trukmeRezultatuSkaitymo, trukmeRezultatuSkaidymas,
    trukmeSkaidymoIrasymas);
00093             break;
00094         }
00095         case 7:
00096         {
00097             // Generuoti 5 atsitiktinio dydžio studentų failus
00098             generuotiAtsitiktiniusFailus();
00099             return;
00100         }
00101
00102         case 8:
00103         {
00104             // Kelius kartus sukamas kodas, kad sužinoti kiek laiko užtrunka kodas
00105             int kartai;
00106             vykdytiKeliskart(kartai);
00107
00108             return;
00109         }
00110         case 9:
00111         {
00112             // Abstralčioji klasė Zmogus
00113             std::cout « "Parodysime, kad bandant sukurti abstrakcios klases Zmogus objekta ivyksta
    klaida:\n";
00114
00115             try
00116             {
00117                 // Zmogus *h = new Zmogus("John", "Doe");
00118             }
00119             catch (const std::runtime_error &e)
00120             {
00121                 std::cout « "Caught an error: " « e.what() « "\n";
00122             }
00123
00124             // Parodysime, kad 5manoma sukurti išvestinės klasės objektus
00125             Studentas studentas("Jonas", "Jonaitis", {10, 9, 8}, 9);
00126             std::cout « "Sekmingai sukurtas isvestines klases 'Studentas' objektas:\n";
00127             std::cout « "Vardas: " « studentas.getVardas() « ", Pavarde: " « studentas.getPavarde() «
    "\n";
00128
00129             return;
00130         }
00131         }
00132         if (pasirinkimas != 5 && pasirinkimas != 6)
00133         {
00134             if (studentai.empty())
00135             {
00136                 throw runtime_error("Nera studentu duomenu.");
00137             }
00138             else
00139             {
00140                 cout « "Studentai: " « studentai.size() « endl;
00141             }
00142
00143             int pasirinkimasRik;
00144             bool validInput = false;
00145
00146             while (!validInput)
00147             {
```

```
00148                 cout « "Jeigu norite rusiuoti pagal varda - 1, jeigu pagal pavarde - 2, pagal vidurki
      didejanciai - 3, pagal vidurki mazejanciai - 4: ";
00149                 cin » pasirinkimasRik;
00150
00151                 if (cin.fail() || (pasirinkimasRik < 1 || pasirinkimasRik > 4))
00152                 {
00153                     cin.clear();
00154                     cin.ignore(numeric_limits<streamsize>::max(), '\n');
00155                     cout « "Blogas pasirinkimas. Prasome ivesti 1, 2, 3 arba 4.\n";
00156                 }
00157                 else
00158                 {
00159                     validInput = true;
00160                 }
00161             }
00162
00163             // Rūšiuoja studentus
00164             if (pasirinkimasRik == 1)
00165             {
00166                 rusiuotiStudentusPagalVarda(studentai);
00167             }
00168             else if (pasirinkimasRik == 2)
00169             {
00170                 rusiuotiStudentusPagalPavarde(studentai);
00171             }
00172             else if (pasirinkimasRik == 3)
00173             {
00174                 rusiuotiPagalVidurkiDidejanciai(studentai);
00175             }
00176             else if (pasirinkimasRik == 4)
00177             {
00178                 rusiuotiPagalVidurkiMazejanciai(studentai);
00179             }
00180             else
00181             {
00182                 cout « "Blogas pasirinkimas. Prasome ivesti 1, 2, 3 arba 4.\n";
00183             }
00184
00185             // Spausdina rezultatus
00186             std::cout « std::left « std::setw(16) « "Pavarde"
00187                       « std::setw(16) « "Vardas"
00188                       « std::setw(25) « "Galutinis Vidurkis"
00189                       « " / " « "Galutine Mediana\n";
00190             std::cout « "-------------------------------------------------------------------------\n";
00191             std::cout « std::fixed « std::setprecision(2);
00192
00193             for (const Studentas &studentas : studentai)
00194             {
00195                 std::cout « studentas « '\n';
00196             }
00197         }
00198     }
00199     catch (const exception &e)
00200     {
00201         cout « "Vec_funkcijos.cpp faile ivyko klaida: " « e.what() « '\n';
00202     }
00203 }
```

5.35 src/Vec_funkcijos_papildomos.cpp File Reference

```
#include "Vec_funkcijos.h"
#include "Vec_funkcijos_papildomos.h"
#include "Vec_failo_apdorojimas.h"
```

Functions

- int gautiPazymi (const string &klausimas)
- void rusiuotiStudentusPagalPavarde (std::vector< Studentas > &studentai)
- void rusiuotiStudentusPagalVarda (std::vector< Studentas > &studentai)
- void rusiuotiPagalVidurkiDidejanciai (std::vector< Studentas > &studentai)
- void rusiuotiPagalVidurkiMazejanciai (std::vector< Studentas > &studentai)
- float skaiciuotiMediana (const std::vector< int > &pazymiai)

Apskaičiuoja pažymių medianą.

- float skaiciuotiVidurki (const std::vector< int > &pazymiai)

  Apskaičiuoja pažymių vidurkį.
- int generuotiSkaiciu (int min, int max)

  Generuoja atsitiktinį skaičių intervalo ribose.
- string generuotiVardaPavarde ()

  Generuoja atsitiktinį vardą ir pavardę.
- Studentas generuotiAtsitiktiniStudenta ()

  Generuoja atsitiktinį studentą.
- void generuotiAtsitiktiniusFailus ()

  Generuoja kelis atsitiktinius studentų failus.
- void vykdytiVisusZingsnius ()

  Vykdo visus programos žingsnius.
- void ivestiDuomenisRanka (vector< Studentas > &studentai)
- void automatiskaiGeneruotiDuomenis (vector< Studentas > &studentai)
- void nuskaitytiDuomenisIsFailo (vector< Studentas > &studentai, long long &trukmeSkaitymo, long long &trukmeVidurkio)
- void skaiciuotiRezultatus (long long &trukmeSkaitymo, long long &trukmeVidurkio, long long &trukmeIrasymo)
- void rusiuotiRezultatus (long long &trukmeRezultatuSkaitymo, long long &trukmeRezultatuSkaidymas, long long &trukmeSkaidymoIrasymas)
- void vykdytiKeliskart (int &kartai)

5.35.1 Function Documentation

5.35.1.1 automatiskaiGeneruotiDuomenis()

```
void automatiskaiGeneruotiDuomenis (
            vector< Studentas > & studentai)
```

Definition at line 298 of file Vec_funkcijos_papildomos.cpp.

5.35.1.2 gautiPazymi()

```
int gautiPazymi (
            const string & klausimas)
```

Definition at line 6 of file Vec_funkcijos_papildomos.cpp.

5.35.1.3 generuotiAtsitiktiniStudenta()

```
Studentas generuotiAtsitiktiniStudenta ()
```

Generuoja atsitiktinį studentą.

Returns

Studentas su atsitiktiniais duomenimis.

Definition at line 132 of file Vec_funkcijos_papildomos.cpp.

5.35.1.4 generuotiAtsitiktiniusFailus()

```
void generuotiAtsitiktiniusFailus ()
```

Generuoja kelis atsitiktinius studentų failus.

Definition at line 173 of file Vec_funkcijos_papildomos.cpp.

5.35.1.5 generuotiSkaiciu()

```
int generuotiSkaiciu (
            int min,
            int max)
```

Generuoja atsitiktinį skaičių intervalo ribose.

Parameters

min	Mažiausia reikšmė.
max	Didžiausia reikšmė.

Returns

Generuotas atsitiktinis skaičius.

Definition at line 109 of file Vec_funkcijos_papildomos.cpp.

5.35.1.6 generuotiVardaPavarde()

```
string generuotiVardaPavarde ()
```

Generuoja atsitiktinį vardą ir pavardę.

Returns

Sugeneruotas vardas ir pavardė.

Definition at line 118 of file Vec_funkcijos_papildomos.cpp.

5.35.1.7 ivestiDuomenisRanka()

```
void ivestiDuomenisRanka (
            vector< Studentas > & studentai)
```

Definition at line 262 of file Vec_funkcijos_papildomos.cpp.

5.35.1.8 nuskaitytiDuomenisIsFailo()

```
void nuskaitytiDuomenisIsFailo (
            vector< Studentas > & studentai,
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio)
```

Definition at line 328 of file Vec_funkcijos_papildomos.cpp.

5.35.1.9 rusiuotiPagalVidurkiDidejanciai()

```
void rusiuotiPagalVidurkiDidejanciai (
            std::vector< Studentas > & studentai)
```

Definition at line 63 of file Vec_funkcijos_papildomos.cpp.

5.35.1.10 rusiuotiPagalVidurkiMazejanciai()

```
void rusiuotiPagalVidurkiMazejanciai (
            std::vector< Studentas > & studentai)
```

Definition at line 70 of file Vec_funkcijos_papildomos.cpp.

5.35.1.11 rusiuotiRezultatus()

```
void rusiuotiRezultatus (
            long long & trukmeRezultatuSkaitymo,
            long long & trukmeRezultatuSkaidymas,
            long long & trukmeSkaidymoIrasymas)
```

Definition at line 412 of file Vec_funkcijos_papildomos.cpp.

5.35.1.12 rusiuotiStudentusPagalPavarde()

```
void rusiuotiStudentusPagalPavarde (
            std::vector< Studentas > & studentai)
```

Definition at line 37 of file Vec_funkcijos_papildomos.cpp.

5.35.1.13 rusiuotiStudentusPagalVarda()

```
void rusiuotiStudentusPagalVarda (
            std::vector< Studentas > & studentai)
```

Definition at line 50 of file Vec_funkcijos_papildomos.cpp.

5.35.1.14 skaiciuotiMediana()

```
float skaiciuotiMediana (
            const std::vector< int > & mediana)
```

Apskaičiuoja pažymių medianą.

Parameters

mediana	Pažymių sąrašas.

Returns

Apskaičiuota mediana.

Definition at line 77 of file Vec_funkcijos_papildomos.cpp.

5.35.1.15 skaiciuotiRezultatus()

```
void skaiciuotiRezultatus (
            long long & trukmeSkaitymo,
            long long & trukmeVidurkio,
            long long & trukmeIrasymo)
```

Definition at line 383 of file Vec_funkcijos_papildomos.cpp.

5.35.1.16 skaiciuotiVidurki()

```
float skaiciuotiVidurki (
            const std::vector< int > & vidurkis)
```

Apskaičiuoja pažymių vidurkį.

Parameters

vidurkis	Pažymių sąrašas.

Returns

Apskaičiuotas vidurkis.

Definition at line 100 of file Vec_funkcijos_papildomos.cpp.

5.35.1.17 vykdytiKeliskart()

```
void vykdytiKeliskart (
            int & kartai)
```

Definition at line 438 of file Vec_funkcijos_papildomos.cpp.

5.35.1.18 vykdytiVisusZingsnius()

```
void vykdytiVisusZingsnius ()
```

Vykdo visus programos žingsnius.

Definition at line 184 of file Vec_funkcijos_papildomos.cpp.

5.36 Vec_funkcijos_papildomos.cpp

Go to the documentation of this file.

```
00001 #include "Vec_funkcijos.h"
00002 #include "Vec_funkcijos_papildomos.h"
00003 #include "Vec_failo_apdorojimas.h"
00004
00005 // Gauna pažymį iš vartotojo įvesties
00006 int gautiPazymi(const string &klausimas)
00007 {
00008     while (true)
00009     {
00010         string skaicius;
00011         cout « klausimas;
00012         cin » skaicius;
00013
00014         if (skaicius == "-1")
00015             return -1; // Grąžina -1, jei vartotojas nori baigti įvestį
00016
00017         try
00018         {
00019             int pazymys = stoi(skaicius);
00020             if (pazymys >= 0 && pazymys <= 10)
00021             {
00022                 return pazymys;
00023             }
00024             else
00025             {
00026                 cout « "Klaida: pazymys turi buti tarp 0 ir 10.\n";
00027             }
00028         }
00029         catch (const invalid_argument &)
00030         {
00031             cout « "Klaida: iveskite teisinga skaiciu.\n";
00032         }
00033     }
00034 }
00035
00036 // Rūšiuoja studentus pagal pavardę, o jei pavardės vienodos - pagal vardą
00037 void rusiuotiStudentusPagalPavarde(std::vector<Studentas> &studentai)
00038 {
00039     std::sort(studentai.begin(), studentai.end(), [](const Studentas &a, const Studentas &b)
00040               {
00041                   if (a.getPavarde() == b.getPavarde())
00042                   {
00043                       return a.getVardas() < b.getVardas(); // Sort by `vardas` if `pavarde` is the
      same
00044                   }
00045                   return a.getPavarde() < b.getPavarde(); // Sort by `pavarde`
00046               });
00047 }
00048
00049 // Rūšiuoja studentus pagal vardą, o jei vardai vienodos - pagal pavardę
00050 void rusiuotiStudentusPagalVarda(std::vector<Studentas> &studentai)
00051 {
00052     std::sort(studentai.begin(), studentai.end(), [](const Studentas &a, const Studentas &b)
00053               {
00054                   if (a.getVardas() == b.getVardas())
00055                   {
00056                       return a.getPavarde() < b.getPavarde(); // Sort by `pavarde` if `vardas` is the
      same
00057                   }
00058                   return a.getVardas() < b.getVardas(); // Sort by `vardas`
00059               });
00060 }
00061
00062 // Rūšiuoja studentus pagal galutinį vidurkį nuo mažiausio iki didžiausio
00063 void rusiuotiPagalVidurkiDidejanciai(std::vector<Studentas> &studentai)
00064 {
00065     std::sort(studentai.begin(), studentai.end(), [](const Studentas &a, const Studentas &b)
00066               { return a.getGalutinisVidurkis() < b.getGalutinisVidurkis(); });
00067 }
00068
00069 // Rūšiuoja studentus pagal galutinį vidurkį nuo didžiausio iki mažiausio
00070 void rusiuotiPagalVidurkiMazejanciai(std::vector<Studentas> &studentai)
00071 {
00072     std::sort(studentai.begin(), studentai.end(), [](const Studentas &a, const Studentas &b)
00073               { return a.getGalutinisVidurkis() > b.getGalutinisVidurkis(); });
00074 }
00075
00076 // Skaičiuoja pažymių medianą
00077 float skaiciuotiMediana(const std::vector<int> &pazymiai)
00078 {
00079     if (pazymiai.empty())
00080         return 0;
```

```
00081
00082     std::vector<int> tempPazymiai = pazymiai; // Create a copy to avoid modifying original
00083     size_t n = tempPazymiai.size();
00084     size_t middle = n / 2;
00085
00086     if (n % 2 == 0)
00087     {
00088         std::nth_element(tempPazymiai.begin(), tempPazymiai.begin() + middle - 1, tempPazymiai.end());
00089         std::nth_element(tempPazymiai.begin() + middle - 1, tempPazymiai.begin() + middle,
      tempPazymiai.end());
00090         return (tempPazymiai[middle - 1] + tempPazymiai[middle]) / 2.0f;
00091     }
00092     else
00093     {
00094         std::nth_element(tempPazymiai.begin(), tempPazymiai.begin() + middle, tempPazymiai.end());
00095         return tempPazymiai[middle];
00096     }
00097 }
00098
00099 // Skaičiuoja pažymių vidurkį
00100 float skaiciuotiVidurki(const std::vector<int> &pazymiai)
00101 {
00102     if (pazymiai.empty())
00103     {
00104         return 0.0f;
00105     }
00106     return std::accumulate(pazymiai.begin(), pazymiai.end(), 0.0f) / pazymiai.size();
00107 }
00108
00109 int generuotiSkaiciu(int min, int max)
00110 {
00111     static std::random_device rd;
00112     static std::mt19937 gen(rd());
00113     std::uniform_int_distribution<int> distrib(min, max);
00114     return distrib(gen);
00115 }
00116
00117 // Generuoja atsitiktinį vardą arba pavardę
00118 string generuotiVardaPavarde()
00119 {
00120     static std::random_device rd;
00121     static std::mt19937 gen(rd());
00122     static const char raides[] = "abcdefghijklmnopqrstuvwxyz";
00123     std::uniform_int_distribution<int> raideDistrib(0, 25);
00124     string vardasPavarde(4, ' ');
00125     for (int i = 0; i < 4; ++i)
00126     {
00127         vardasPavarde[i] = raides[raideDistrib(gen)];
00128     }
00129     return vardasPavarde;
00130 }
00131
00132 Studentas generuotiAtsitiktiniStudenta()
00133 {
00134     Studentas studentas;
00135
00136     // Set atsitiktinis vardas ir pavarde
00137     studentas.setVardas(generuotiVardaPavarde());
00138     studentas.setPavarde(generuotiVardaPavarde());
00139
00140     // Pre-allocate space for pazymiai
00141     int pazymiuKiekis = generuotiSkaiciu(1, 20);
00142     std::vector<int> pazymiai;
00143     pazymiai.reserve(pazymiuKiekis);
00144
00145     // Generuojami atsitiktiniai pažymiai
00146     for (int i = 0; i < pazymiuKiekis; i++)
00147     {
00148         pazymiai.push_back(generuotiSkaiciu(0, 10));
00149     }
00150     studentas.setPazymiai(pazymiai); // Set pazymiai
00151
00152     // Generuojamas egzamino pažymys
00153     int egzaminoPazymys = generuotiSkaiciu(0, 10);
00154     studentas.setEgzaminoPazymys(egzaminoPazymys);
00155
00156     // Apskaičiuojami vidurkis ir mediana
00157     float vidurkis = skaiciuotiVidurki(studentas.getPazymiai());
00158     float mediana = skaiciuotiMediana(studentas.getPazymiai());
00159     studentas.setVidurkis(vidurkis); // Set vidurkis
00160     studentas.setMediana(mediana);   // Set mediana
00161
00162     // Apskaičiuojami galutiniai įvertinimai
00163     const double egzaminoBalas = 0.6 * egzaminoPazymys;
00164     const double vidurkioBalas = 0.4 * vidurkis;
00165     const double medianosBalas = 0.4 * mediana;
00166
```

```
00167     studentas.setGalutinisVidurkis(vidurkioBalas + egzaminoBalas); // Set galutinis vidurkis
00168     studentas.setGalutineMediana(medianosBalas + egzaminoBalas);   // Set galutine mediana
00169
00170     return studentas;
00171 }
00172
00173 void generuotiAtsitiktiniusFailus()
00174 {
00175     for (int i = 1; i <= 5; ++i)
00176     {
00177         int studentuKiekis = generuotiSkaiciu(1, 1000000);
00178         string failoPavadinimas = "txt_failai/studentai_random_" + to_string(i) + ".txt";
00179         generuotiStudentuFaila(studentuKiekis, failoPavadinimas);
00180         cout « "Sugeneruotas failas " « failoPavadinimas « " su " « studentuKiekis « " studentu.\n";
00181     }
00182 }
00183
00184 void vykdytiVisusZingsnius()
00185 {
00186     vector<int> studentuKiekiai = {1000000, 10000000};
00187
00188     // Atidaryti CSV failą rašymui
00189     ofstream csvFile("performance_data.csv", std::ios::app);
00190     if (!csvFile.is_open())
00191     {
00192         throw runtime_error("Nepavyko atidaryti CSV failo");
00193     }
00194
00195     // Įrašo CSV antraštę, jei failas tuščias
00196     csvFile.seekp(0, std::ios::end);
00197     if (csvFile.tellp() == 0)
00198     {
00199         csvFile « "Testavimo Laikas;Studentu Kiekis;Studentu generavimo laikas;Sugeneruotu duomenu
      skaitymo laikas;Rezultatu irasymo laikas;Rezultatu skaitymo laikas;Rezultatu skaidymo laikas;Skaidymo
      irasymas;Bendras Laikas\n";
00200     }
00201
00202     for (int kiekis : studentuKiekiai)
00203     {
00204         cout « "Vykdomi zingsniai su " « kiekis « " studentu:\n";
00205
00206         // Gauti dabartinį laiką
00207         auto now = std::chrono::system_clock::now();
00208         auto in_time_t = std::chrono::system_clock::to_time_t(now);
00209         std::stringstream ss;
00210         ss « std::put_time(std::localtime(&in_time_t), "%Y-%m-%d %X");
00211         string timestamp = ss.str();
00212
00213         // Generuoti studentų failą
00214         string studentuFailas = "txt_failai/studentai_" + to_string(kiekis) + ".txt";
00215         auto pradziaGeneravimo = std::chrono::high_resolution_clock::now();
00216         generuotiStudentuFaila(kiekis, studentuFailas);
00217         auto pabaigaGeneravimo = std::chrono::high_resolution_clock::now();
00218         auto trukmeGeneravimo =
      std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaGeneravimo - pradziaGeneravimo);
00219         cout « "Failo su " « kiekis « "studentais generavimas uztruko " « trukmeGeneravimo.count() « "
      ms.\n";
00220
00221         // Skaitomas sugeneruotas failas, apskaičiuoja galutinius rezultatus ir išvedamas į rezultatų
      failą
00222         string rezultatuFailas = "txt_failai/rezultatai_" + to_string(kiekis) + ".txt";
00223         cout « "Skaitomi duomenys ir isvedami i " « rezultatuFailas « "...\n";
00224         long long trukmeSkaitymo, trukmeVidurkio, trukmeIrasymo;
00225         auto pradziaSkaitymo = std::chrono::high_resolution_clock::now();
00226         skaitytiIrIsvestiDuomenis(studentuFailas, rezultatuFailas, trukmeSkaitymo, trukmeVidurkio,
      trukmeIrasymo);
00227         auto pabaigaSkaitymo = std::chrono::high_resolution_clock::now();
00228         auto trukmeSkaitymoLaikas =
      std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaSkaitymo - pradziaSkaitymo);
00229
00230         cout « "Skaitymo laikas: " « trukmeSkaitymo « " ms.\n";
00231         cout « "Duomenu isvedimas i " « rezultatuFailas « " uztruko " « trukmeIrasymo « " ms.\n";
00232
00233         // Rezultatų failo padalijimas į išlaikiusius ir neišlaikiusius
00234         string islaikeFailas = "txt_failai/rezultatai_" + to_string(kiekis) + "_islaike.txt";
00235         string neislaikeFailas = "txt_failai/rezultatai_" + to_string(kiekis) + "_neislaike.txt";
00236         cout « "Dalinamas rezultatu failas i islaikiusius ir neislaikiusius...\n";
00237         long long trukmeRezultatuSkaitymo, trukmeRezultatuSkaidymas, trukmeSkaidymoIrasymas;
00238         padalintiRezultatuFaila(rezultatuFailas, islaikeFailas, neislaikeFailas,
      trukmeRezultatuSkaitymo, trukmeRezultatuSkaidymas, trukmeSkaidymoIrasymas);
00239         cout « "Rezultatu failo dalinimas uztruko " « trukmeRezultatuSkaitymo +
      trukmeRezultatuSkaidymas + trukmeSkaidymoIrasymas « " ms.\n";
00240
00241         // Skaičuoti bendrą laiką
00242         long long bendrasLaikas = trukmeGeneravimo.count() + trukmeSkaitymo + trukmeIrasymo +
      trukmeRezultatuSkaitymo + trukmeRezultatuSkaidymas + trukmeSkaidymoIrasymas;
00243         cout « "Visi zingsniai su " « kiekis « " studentu baigti. Trukme: " « bendrasLaikas « "
```

```
ms.\n\n";
00244
00245         // Surašyti laikus į CSV failą
00246         csvFile « timestamp « ";"
00247                 « kiekis « ";"
00248                 « trukmeGeneravimo.count() « ";"
00249                 « trukmeSkaitymo « ";"
00250                 « trukmeIrasymo « ";"
00251                 « trukmeRezultatuSkaitymo « ";"
00252                 « trukmeRezultatuSkaidymas « ";"
00253                 « trukmeSkaidymoIrasymas « ";"
00254                 « bendrasLaikas « "\n";
00255     }
00256
00257     csvFile.close();
00258     cout « "Visi zingsniai visiems studentu kiekiams baigti.\n";
00259     cout « "Duomenys issaugoti faile 'performance_data.csv'\n";
00260 }
00261
00262 void ivestiDuomenisRanka(vector<Studentas> &studentai)
00263 {
00264     int studentuKiekis;
00265     bool gerasPasirinkimas = false;
00266
00267     cout « "Kiek studentu norite irasyti? ";
00268
00269     // Studentų skaičiaus įvesties patvirtinimas
00270     while (!gerasPasirinkimas)
00271     {
00272         cin » studentuKiekis;
00273         if (cin.fail() || studentuKiekis < 1)
00274         {
00275             cin.clear();
00276             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00277             cout « "Klaida: Ivestas neteisingas studentu skaicius. Prasome ivesti skaiciu didesni uz
    0: ";
00278         }
00279         else
00280         {
00281             gerasPasirinkimas = true;
00282         }
00283     }
00284     studentai.resize(studentuKiekis);
00285
00286     // Studentų duomenų įvedimas ir atminties adreso spausdinimas
00287     for (Studentas &studentas : studentai)
00288     {
00289         // Input student data
00290         std::cin » studentas;
00291
00292         // Output the entered data
00293         // std::cout « "Ivestas studentas:\n" « studentas « '\n';
00294         cout « "Studento vektoriaus objektas atmintyje saugomas adresu: " « &studentas « endl;
00295     }
00296 }
00297
00298 void automatiskaiGeneruotiDuomenis(vector<Studentas> &studentai)
00299 {
00300     int studentuKiekis;
00301     bool gerasPasirinkimas = false;
00302
00303     cout « "Kiek studentu norite sugeneruoti? ";
00304
00305     // Įvesties patvirtinimas
00306     while (!gerasPasirinkimas)
00307     {
00308         cin » studentuKiekis;
00309         if (cin.fail() || studentuKiekis < 1)
00310         {
00311             cin.clear();
00312             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00313             cout « "Klaida: Ivestas neteisingas studentu skaicius. Prasome ivesti skaiciu didesni uz
    0: ";
00314         }
00315         else
00316         {
00317             gerasPasirinkimas = true;
00318         }
00319     }
00320
00321     // Sugeneruoti atsitiktinius studentus ir pridėti juos prie vektoriaus
00322     for (int i = 0; i < studentuKiekis; i++)
00323     {
00324         studentai.push_back(generuotiAtsitiktiniStudenta());
00325     }
00326 }
00327
```

```
00328 void nuskaitytiDuomenisIsFailo(vector<Studentas> &studentai, long long &trukmeSkaitymo, long long
      &trukmeVidurkio)
00329 {
00330     int failoPasirinkimas;
00331     bool gerasPasirinkimas = false;
00332 
00333     // Leisti vartotoją pasirinkti failą
00334     cout « "Pasirinkite faila (1. studentai10.txt, 2. studentai100.txt, 3. studentai10000.txt, 4.
      studentai100000.txt, 5. studentai1000000.txt, 6. studentai10_blog.txt, 7. tuscias.txt): ";
00335 
00336     // Įvesties patvirtinimas
00337     while (!gerasPasirinkimas)
00338     {
00339         cin » failoPasirinkimas;
00340         if (cin.fail() || failoPasirinkimas < 1 || failoPasirinkimas > 7)
00341         {
00342             cin.clear();
00343             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00344             cout « "Neteisingas pasirinkimas. Prasome ivesti skaiciu nuo 1 iki 7: ";
00345         }
00346         else
00347         {
00348             gerasPasirinkimas = true;
00349         }
00350     }
00351 
00352     string failoPavadinimas;
00353     switch (failoPasirinkimas)
00354     {
00355     case 1:
00356         failoPavadinimas = "txt_failai/studentai10.txt";
00357         break;
00358     case 2:
00359         failoPavadinimas = "txt_failai/studentai100.txt";
00360         break;
00361     case 3:
00362         failoPavadinimas = "txt_failai/studentai10000.txt";
00363         break;
00364     case 4:
00365         failoPavadinimas = "txt_failai/studentai100000.txt";
00366         break;
00367     case 5:
00368         failoPavadinimas = "txt_failai/studentai1000000.txt";
00369         break;
00370     case 6:
00371         failoPavadinimas = "txt_failai/studentai10_blog.txt";
00372         break;
00373     case 7:
00374         failoPavadinimas = "txt_failai/tuscias.txt";
00375         break;
00376     }
00377 
00378     skaitytiDuomenisIsFailo(failoPavadinimas, studentai, trukmeSkaitymo, trukmeVidurkio);
00379 
00380     cout « "Duomenys nuskaityti is " « failoPavadinimas « " per " « trukmeSkaitymo « " ms\n";
00381 }
00382 
00383 void skaiciuotiRezultatus(long long &trukmeSkaitymo, long long &trukmeVidurkio, long long
      &trukmeIrasymo)
00384 {
00385     std::vector<std::string> studentuSkaicius = {
00386         "_1000", "_10000", "_100000", "_1000000", "_10000000",
00387         "1000", "10000", "100000", "1000000", "10000000"};
00388 
00389     std::cout « "Pasirinkite rezultatu faila:\n";
00390     std::cout « "Kodo generuoti duomenys\n1. studentai_1000.txt\n2. studentai_10000.txt\n3.
      studentai_100000.txt\n4. studentai_1000000.txt\n5. studentai_10000000.txt\n";
00391     std::cout « "Pavyzdiniai duomenys\n6. studentai1000.txt\n7. studentai10000.txt\n8.
      studentai100000.txt\n9. studentai1000000.txt\n10. studentai10000000.txt\n";
00392     std::cout « "Jusu pasirinkimas: ";
00393 
00394     // Įvesties patikrinimas
00395     int failoPasirinkimas;
00396     std::cin » failoPasirinkimas;
00397     if (failoPasirinkimas < 1 || failoPasirinkimas > 10)
00398     {
00399         throw std::runtime_error("Neteisingas failo pasirinkimas.");
00400     }
00401 
00402     // Generuoti failų pavadinimus pagal pasirinkimą
00403     std::string duomenuFailas = "txt_failai/studentai" + studentuSkaicius[failoPasirinkimas - 1] +
      ".txt";
00404     std::string isvestiesFailoPavadinimas = "txt_failai/rezultatai" +
      studentuSkaicius[failoPasirinkimas - 1] + ".txt";
00405 
00406     skaitytiIrIsvestiDuomenis(duomenuFailas, isvestiesFailoPavadinimas, trukmeSkaitymo,
      trukmeVidurkio, trukmeIrasymo);
```

```
00407     std::cout « "Duomenys nuskaityti is " « duomenuFailas
00408               « " per " « trukmeSkaitymo « "ms ir isvesti i "
00409               « isvestiesFailoPavadinimas « " per " « trukmeIrasymo « " ms.\n";
00410 }
00411
00412 void rusiuotiRezultatus(long long &trukmeRezultatuSkaitymo, long long &trukmeRezultatuSkaidymas, long
      long &trukmeSkaidymoIrasymas)
00413 {
00414     vector<string> studentuSkaicius = {"_1000", "_10000", "_100000", "_1000000", "_10000000", "1000",
      "10000", "100000", "1000000", "10000000"};
00415
00416     cout « "Pasirinkite rezultatu faila:\n";
00417     cout « "Kodo generuoti duomenys\n1. rezultatai_1000.txt\n2. rezultatai_10000.txt\n3.
      rezultatai_100000.txt\n4. rezultatai_1000000.txt\n5. rezultatai_10000000.txt\n";
00418     cout « "Pavyzdiniai duomenys\n6. rezultatai1000.txt\n7. rezultatai10000.txt\n8.
      rezultatai100000.txt\n9. rezultatai1000000.txt\n10. rezultatai10000000.txt\n";
00419     cout « "Jusu pasirinkimas: ";
00420
00421     // Įvesties patikrinimas
00422     int failoPasirinkimas;
00423     cin » failoPasirinkimas;
00424     if (failoPasirinkimas < 1 || failoPasirinkimas > 10)
00425     {
00426         throw runtime_error("Neteisingas failo pasirinkimas.");
00427     }
00428
00429     // Generuoti failų pavadinimus pagal pasirinkimą
00430     string duomenuFailas = "txt_failai/rezultatai" + studentuSkaicius[failoPasirinkimas - 1] + ".txt";
00431     string islaikiusiuFailoPavadinimas = "txt_failai/rezultatai" + studentuSkaicius[failoPasirinkimas
      - 1] + "_islaike.txt";
00432     string neislaikiusiuFailoPavadinimas = "txt_failai/rezultatai" +
      studentuSkaicius[failoPasirinkimas - 1] + "_neislaike.txt";
00433
00434     padalintiRezultatuFaila(duomenuFailas, islaikiusiuFailoPavadinimas, neislaikiusiuFailoPavadinimas,
      trukmeRezultatuSkaitymo, trukmeRezultatuSkaidymas, trukmeSkaidymoIrasymas);
00435     cout « "Rezultatu failas padalintas i " « islaikiusiuFailoPavadinimas « " ir " «
      neislaikiusiuFailoPavadinimas « '\n';
00436 }
00437
00438 void vykdytiKeliskart(int &kartai)
00439 {
00440     bool validInput = false;
00441
00442     // Įvesties patikrinimas
00443     while (!validInput)
00444     {
00445         cout « "Kiek kartu norite paleisti funkcija 'vykdytiVisusZingsnius'? ";
00446         cin » kartai;
00447
00448         if (cin.fail() || kartai <= 0)
00449         {
00450             cin.clear();
00451             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00452             cout « "Neteisingas skaicius. Prasome ivesti teigiama skaiciu.\n";
00453         }
00454         else
00455         {
00456             validInput = true;
00457         }
00458     }
00459
00460     // Vykdo visus žingsnius i kartų
00461     for (int i = 0; i < kartai; i++)
00462     {
00463         cout « "Vykdoma " « i + 1 « " karta:\n";
00464         vykdytiVisusZingsnius();
00465     }
00466 }
```

5.37 src/Vec_generuoti_failus.cpp File Reference

```
#include "Vec_funkcijos.h"
```

Functions

- void generuotiStudentuFaila (int studentuKiekis, const string &failoPavadinimas)
- void generuotiFaila ()

  Generuoja studentų failą su atsitiktiniais duomenimis.

5.37.1 Function Documentation

5.37.1.1 generuotiFaila()

```
void generuotiFaila ()
```

Generuoja studentų failą su atsitiktiniais duomenimis.

Definition at line 54 of file Vec_generuoti_failus.cpp.

5.37.1.2 generuotiStudentuFaila()

```
void generuotiStudentuFaila (
            int studentuKiekis,
            const string & failoPavadinimas)
```

Definition at line 3 of file Vec_generuoti_failus.cpp.

5.38 Vec_generuoti_failus.cpp

Go to the documentation of this file.

```
00001 #include "Vec_funkcijos.h"
00002
00003 void generuotiStudentuFaila(int studentuKiekis, const string &failoPavadinimas)
00004 {
00005     ofstream isvedimoFailas(failoPavadinimas);
00006     if (!isvedimoFailas.is_open())
00007     {
00008         throw runtime_error("Nepavyko atidaryti failo " + failoPavadinimas);
00009     }
00010
00011     // Generuojama antraštės eilutė
00012     ostringstream headerStream;
00013     headerStream « left « setw(16) « "Vardas" « setw(16) « "Pavarde" « setw(20) « " ";
00014     for (int i = 1; i <= 8; ++i)
00015     {
00016         headerStream « setw(10) « ("ND" + to_string(i));
00017     }
00018
00019     headerStream « setw(9) « ("ND9");
00020
00021     for (int i = 10; i <= 15; ++i)
00022     {
00023         headerStream « setw(10) « ("ND" + to_string(i));
00024     }
00025     headerStream « setw(9) « "Egz." « "\n";
00026     isvedimoFailas « headerStream.str();
00027
00028     // Naudojamas stringstream, kad kauptų studentų duomenis prieš įrašymą
00029     ostringstream outputStream;
00030     for (int i = 0; i < studentuKiekis; ++i)
00031     {
00032         string vardas = generuotiVardaPavarde(); // Generuojami vardai ir pavardės
00033         string pavarde = generuotiVardaPavarde();
00034
00035         outputStream « left « setw(16) « vardas « setw(16) « pavarde « setw(13) « " ";
00036
00037         // Generuojami 15 pažymių
00038         for (int j = 0; j < 15; ++j)
00039         {
00040             outputStream « right « setw(10) « generuotiSkaiciu(1, 10);
00041         }
00042
00043         // Generuojamas egzamino pažymys
00044         outputStream « right « setw(10) « generuotiSkaiciu(1, 10) « "\n";
00045     }
00046
00047     // Galutinis rašymas iš stream į failą
00048     isvedimoFailas « outputStream.str();
```

```
00049
00050     isvedimoFailas.close();
00051     cout « "Failas '" « failoPavadinimas « "' su " « studentuKiekis « " studentais buvo
      sugeneruotas.\n";
00052 }
00053
00054 void generuotiFaila()
00055 {
00056     vector<int> studentuSkaicius = {1000, 10000, 100000, 1000000, 10000000};
00057
00058     cout « "Kiek studentu norite sugeneruoti:\n";
00059     for (int i = 0; i < studentuSkaicius.size(); ++i)
00060     {
00061         cout « i + 1 « ". " « studentuSkaicius[i] « " studentu.\n";
00062     }
00063
00064     int pasirinkimas;
00065     while (true)
00066     {
00067         cout « "Prasome ivesti skaiciu (1-5): ";
00068         cin » pasirinkimas;
00069         if (cin.fail() || pasirinkimas < 1 || pasirinkimas > 5)
00070         {
00071             cin.clear();
00072             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00073             cout « "Neteisingas pasirinkimas. Prasome ivesti 1, 2, 3, 4 arba 5.\n";
00074         }
00075         else
00076         {
00077             break;
00078         }
00079     }
00080
00081     int studentuKiekis = studentuSkaicius[pasirinkimas - 1];
00082     string failoPavadinimas = "txt_failai/studentai_" + to_string(studentuKiekis) + ".txt";
00083
00084     try
00085     {
00086         auto pradziaSkaitymo = std::chrono::high_resolution_clock::now();
00087         generuotiStudentuFaila(studentuKiekis, failoPavadinimas);
00088         auto pabaigaSkaitymo = std::chrono::high_resolution_clock::now();
00089         long long trukmeSkaitymo =
      std::chrono::duration_cast<std::chrono::milliseconds>(pabaigaSkaitymo - pradziaSkaitymo).count();
00090         cout « "Duomenu generavimas ir failo irasymas truko " « trukmeSkaitymo « "ms.\n";
00091     }
00092     catch (const exception &e)
00093     {
00094         cout « "Klaida: " « e.what() « "\n";
00095     }
00096 }
```

Index