

# Student Management System

Generated by Doxygen 1.13.2



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 Laikas Class Reference	7
4.1.1 Detailed Description	7
4.1.2 Constructor & Destructor Documentation	7
4.1.2.1 Laikas()	7
4.1.3 Member Function Documentation	8
4.1.3.1 baigti()	8
4.1.3.2 gautiLaikoSkirtuma()	8
4.1.3.3 pradeti()	8
4.1.4 Member Data Documentation	8
4.1.4.1 end	8
4.1.4.2 start	8
4.1.4.3 veiksmoPavadinimas	9
4.2 Studentas Class Reference	9
4.2.1 Detailed Description	11
4.2.2 Constructor & Destructor Documentation	11
4.2.2.1 Studentas() [1/5]	11
4.2.2.2 Studentas() [2/5]	11
4.2.2.3 Studentas() [3/5]	12
4.2.2.4 Studentas() [4/5]	12
4.2.2.5 Studentas() [5/5]	12
4.2.2.6 ~Studentas()	12
4.2.3 Member Function Documentation	12
4.2.3.1 addND()	12
4.2.3.2 clearND()	13
4.2.3.3 egzaminas()	13
4.2.3.4 galBalas()	13
4.2.3.5 galutinis()	13
4.2.3.6 nd()	14
4.2.3.7 nuskaitymasFile()	14
4.2.3.8 operator=() [1/2]	14
4.2.3.9 operator=() [2/2]	14
4.2.3.10 print()	15
4.2.3.11 read()	16

4.2.3.12 readStudent()	16
4.2.3.13 setEgzaminas()	16
4.2.3.14 setGalutinis()	16
4.2.3.15 skaiciuotiMed()	17
4.2.3.16 skaiciuotiVid()	17
4.2.4 Member Data Documentation	17
4.2.4.1 destruktoriuSk	17
4.2.4.2 egzaminas_	17
4.2.4.3 galutinis_	18
4.2.4.4 nd_	18
4.3 Zmogus Class Reference	18
4.3.1 Detailed Description	19
4.3.2 Constructor & Destructor Documentation	19
4.3.2.1 Zmogus() [1/4]	19
4.3.2.2 Zmogus() [2/4]	19
4.3.2.3 Zmogus() [3/4]	19
4.3.2.4 Zmogus() [4/4]	20
4.3.2.5 ~Zmogus()	20
4.3.3 Member Function Documentation	20
4.3.3.1 operator=() [1/2]	20
4.3.3.2 operator=() [2/2]	20
4.3.3.3 pavarde()	21
4.3.3.4 print()	21
4.3.3.5 read()	21
4.3.3.6 setPavarde()	21
4.3.3.7 setVardas()	22
4.3.3.8 vardas()	22
4.3.4 Member Data Documentation	22
4.3.4.1 pavarde_	22
4.3.4.2 vardas_	22
<b>5 File Documentation</b>	<b>23</b>
5.1 funkcijos.cpp File Reference	23
5.1.1 Function Documentation	24
5.1.1.1 compareByGalutinis()	24
5.1.1.2 compareByPavarde()	25
5.1.1.3 compareByVardas()	25
5.1.1.4 isvestiStudentusIfaila()	25
5.1.1.5 skaitytiIsFailo()	26
5.1.1.6 skirstytiStudentus()	26
5.1.1.7 sortStudentai()	26
5.1.1.8 testuotiDuomenuApdorojima()	26

5.1.1.9 testuotiStudentoMetodus()	27
5.1.1.10 testuotiZmogausKlase()	27
5.2 funkcijos.h File Reference	27
5.2.1 Function Documentation	28
5.2.1.1 compareByGalutinis()	28
5.2.1.2 compareByPavarde()	28
5.2.1.3 compareByVardas()	29
5.2.1.4 isvestiStudentusIFaila()	29
5.2.1.5 skaitytisFailo()	29
5.2.1.6 skirstytiStudentus()	30
5.2.1.7 sortStudentai()	30
5.2.1.8 testuotiDuomenuApdorojima()	30
5.2.1.9 testuotiStudentoMetodus()	31
5.2.1.10 testuotiZmogausKlase()	31
5.3 funkcijos.h	31
5.4 laikas.cpp File Reference	32
5.5 laikas.h File Reference	32
5.6 laikas.h	33
5.7 main.cpp File Reference	33
5.7.1 Function Documentation	33
5.7.1.1 main()	33
5.8 zmogus.cpp File Reference	33
5.8.1 Function Documentation	34
5.8.1.1 operator<<()	34
5.8.1.2 operator>>()	34
5.9 zmogus.h File Reference	34
5.9.1 Function Documentation	35
5.9.1.1 operator<<()	35
5.9.1.2 operator>>()	35
5.10 zmogus.h	36
<b>Index</b>	<b>37</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Laikas . . . . .	7
Zmogus . . . . .	18
Studentas . . . . .	9





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">Laikas</a>	Class for measuring execution time . . . . .	7
<a href="#">Studentas</a>	Student class that inherits from <a href="#">Zmogus</a> (Person) abstract class . . . . .	9
<a href="#">Zmogus</a>	Abstract base class representing a person . . . . .	18



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">funkcijos.cpp</a>	23
<a href="#">funkcijos.h</a>	27
<a href="#">laikas.cpp</a>	32
<a href="#">laikas.h</a>	32
<a href="#">main.cpp</a>	33
<a href="#">zmogus.cpp</a>	33
<a href="#">zmogus.h</a>	34



# Chapter 4

## Class Documentation

### 4.1 Laikas Class Reference

Class for measuring execution time.

```
#include <laikas.h>
```

#### Public Member Functions

- [Laikas](#) (const std::string &pavadinimas)  
*Constructor.*
- void [pradeti](#) ()  
*Start the timer Records the starting time point.*
- void [baigti](#) ()  
*Stop the timer and print the elapsed time Records the ending time point and outputs the time difference.*
- double [gautiLaikoSkirtuma](#) ()  
*Get the elapsed time in seconds.*

#### Private Attributes

- std::chrono::high\_resolution\_clock::time\_point [start](#)  
*Start time point.*
- std::chrono::high\_resolution\_clock::time\_point [end](#)  
*End time point.*
- std::string [veiksmaPavadinimas](#)  
*Name of the operation being timed.*

#### 4.1.1 Detailed Description

Class for measuring execution time.

This class provides functionality to measure and report the execution time of code segments using high-resolution clock.

#### 4.1.2 Constructor & Destructor Documentation

##### 4.1.2.1 Laikas()

```
Laikas::Laikas (  
    const std::string & pavadinimas)
```

Constructor.

**Parameters**

<i>pavadinimas</i>	Name of the operation to time
--------------------	-------------------------------

### 4.1.3 Member Function Documentation

#### 4.1.3.1 baigti()

```
void Laikas::baigti ()
```

Stop the timer and print the elapsed time Records the ending time point and outputs the time difference.

#### 4.1.3.2 gautiLaikoSkirtuma()

```
double Laikas::gautiLaikoSkirtuma ()
```

Get the elapsed time in seconds.

**Returns**

Time difference between start and end in seconds

#### 4.1.3.3 pradeti()

```
void Laikas::pradeti ()
```

Start the timer Records the starting time point.

### 4.1.4 Member Data Documentation

#### 4.1.4.1 end

```
std::chrono::high_resolution_clock::time_point Laikas::end [private]
```

End time point.

#### 4.1.4.2 start

```
std::chrono::high_resolution_clock::time_point Laikas::start [private]
```

Start time point.

#### 4.1.4.3 veiksmoPavadinimas

```
std::string Laikas::veiksmoPavadinimas [private]
```

Name of the operation being timed.

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

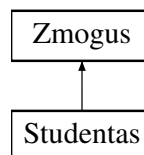
- [laikas.h](#)
- [laikas.cpp](#)

## 4.2 Studentas Class Reference

Student class that inherits from [Zmogus](#) (Person) abstract class.

```
#include <funkcijos.h>
```

Inheritance diagram for Studentas:



### Public Member Functions

- [Studentas](#) ()  
*Default constructor Initializes a student with default values.*
- [Studentas](#) (std::istream &is)  
*Constructor that initializes a student from input stream.*
- [Studentas](#) (const std::string &vardas, const std::string &pavarde)  
*Constructor with name and surname.*
- [Studentas](#) (const [Studentas](#) &other)  
*Copy constructor.*
- [Studentas](#) & operator= (const [Studentas](#) &other)  
*Copy assignment operator.*
- [Studentas](#) ([Studentas](#) &&other) noexcept  
*Move constructor.*
- [Studentas](#) & operator= ([Studentas](#) &&other) noexcept  
*Move assignment operator.*
- [~Studentas](#) () override  
*Virtual destructor Overrides the pure virtual destructor from [Zmogus](#).*
- void [print](#) (std::ostream &os) const override  
*Prints student information to output stream.*
- void [read](#) (std::istream &is) override  
*Reads student information from input stream.*
- std::vector< int > [nd](#) () const  
*Get homework grades.*

- int `egzaminas` () const  
*Get exam grade.*
- double `galutinis` () const  
*Get final grade.*
- void `setEgzaminas` (int `egzaminas`)  
*Set exam grade.*
- void `setGalutinis` (double `galutinis`)  
*Set final grade.*
- std::istream & `readStudent` (std::istream &is)  
*Read student data from input stream.*
- void `addND` (int pazymys)  
*Add a homework grade.*
- void `clearND` ()  
*Clear all homework grades.*
- double `skaiciuotiVid` () const  
*Calculate average of homework grades.*
- double `skaiciuotiMed` () const  
*Calculate median of homework grades.*
- double `galBalas` (bool naudotiVidurki=true) const  
*Calculate final grade.*

## Public Member Functions inherited from `Zmogus`

- `Zmogus` ()=default  
*Default constructor.*
- `Zmogus` (const std::string &`vardas`, const std::string &`pavarde`)  
*Constructor with name and surname.*
- `Zmogus` (const `Zmogus` &other)  
*Copy constructor.*
- `Zmogus` & `operator=` (const `Zmogus` &other)  
*Copy assignment operator.*
- `Zmogus` (`Zmogus` &&other) noexcept  
*Move constructor.*
- `Zmogus` & `operator=` (`Zmogus` &&other) noexcept  
*Move assignment operator.*
- virtual `~Zmogus` ()=0  
*Virtual destructor Pure virtual destructor makes this class abstract.*
- std::string `vardas` () const  
*Get first name.*
- std::string `pavarde` () const  
*Get last name.*
- void `setVardas` (const std::string &`vardas`)  
*Set first name.*
- void `setPavarde` (const std::string &`pavarde`)  
*Set last name.*

## Static Public Member Functions

- static void `nuskaitymasFile` (std::vector< `Studentas` > &grupe, const std::string &failoPavadinimas)  
*Static method for reading students from file.*



### Static Public Attributes

- static int `destruktoriuSk` = 0  
*Static counter for tracking destructor calls (for testing)*

### Private Attributes

- `std::vector< int >` `nd_`  
*Vector of homework (namų darbai) scores.*
- int `egzaminas_`  
*Exam score.*
- double `galutinis_`  
*Final grade.*

### Additional Inherited Members

### Protected Attributes inherited from `Zmogus`

- `std::string` `vardas_`  
*First name of the person.*
- `std::string` `pavarde_`  
*Last name of the person.*

## 4.2.1 Detailed Description

Student class that inherits from `Zmogus` (Person) abstract class.

This class represents a student with homework grades, exam score and final grade. It inherits from the abstract `Zmogus` class and implements its pure virtual methods.

## 4.2.2 Constructor & Destructor Documentation

### 4.2.2.1 `Studentas()` [1/5]

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

Default constructor Initializes a student with default values.

### 4.2.2.2 `Studentas()` [2/5]

```
Studentas::Studentas (  
    std::istream & is)
```

Constructor that initializes a student from input stream.

## Parameters

<i>is</i>	Input stream to read from
-----------	---------------------------

**4.2.2.3 Studentas()** [3/5]

```
Studentas::Studentas (  
    const std::string & vardas,  
    const std::string & pavarde) [inline]
```

Constructor with name and surname.

## Parameters

<i>vardas</i>	Student's first name
<i>pavarde</i>	Student's last name

**4.2.2.4 Studentas()** [4/5]

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

Copy constructor.

## Parameters

<i>other</i>	Another student to copy from
--------------	------------------------------

**4.2.2.5 Studentas()** [5/5]

```
Studentas::Studentas (  
    Studentas && other) [noexcept]
```

Move constructor.

## Parameters

<i>other</i>	Another student to move from
--------------	------------------------------

**4.2.2.6 ~Studentas()**

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

Virtual destructor Overrides the pure virtual destructor from [Zmogus](#).

**4.2.3 Member Function Documentation****4.2.3.1 addND()**

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

Add a homework grade.

## Parameters

<i>pazymys</i>	Grade to add
----------------	--------------

## Exceptions

<i>std::invalid_argument</i>	If grade is not between 1 and 10
------------------------------	----------------------------------

**4.2.3.2 clearND()**

```
void Studentas::clearND ()
```

Clear all homework grades.

**4.2.3.3 egzaminas()**

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

Get exam grade.

## Returns

Exam grade

**4.2.3.4 galBalas()**

```
double Studentas::galBalas (  
    bool naudotiVidurki = true) const
```

Calculate final grade.

## Parameters

<i>naudotiVidurki</i>	Whether to use average (true) or median (false)
-----------------------	---

## Returns

Final grade calculated as  $0.4 \cdot \text{homework} + 0.6 \cdot \text{exam}$

**4.2.3.5 galutinis()**

```
double Studentas::galutinis () const [inline]
```

Get final grade.

## Returns

Final grade

#### 4.2.3.6 nd()

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

Get homework grades.

##### Returns

Vector of homework grades

#### 4.2.3.7 nuskaitymasFile()

```
void Studentas::nuskaitymasFile (
    std::vector< Studentas > & grupe,
    const std::string & failoPavadinimas) [static]
```

Static method for reading students from file.

##### Parameters

<i>grupe</i>	Vector to store students
<i>failoPavadinimas</i>	Filename to read from

#### 4.2.3.8 operator=() [1/2]

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

Copy assignment operator.

##### Parameters

<i>other</i>	Another student to copy from
--------------	------------------------------

##### Returns

Reference to this student after assignment

#### 4.2.3.9 operator=() [2/2]

```
Studentas & Studentas::operator= (
    Studentas && other) [noexcept]
```

Move assignment operator.

##### Parameters

<i>other</i>	Another student to move from
--------------	------------------------------

##### Returns

Reference to this student after assignment

#### 4.2.3.10 print()

```
void Studentas::print (  
    std::ostream & os) const [override], [virtual]
```

Prints student information to output stream.

**Parameters**

<i>os</i>	Output stream to print to
-----------	---------------------------

Implements [Zmogus](#).

**4.2.3.11 read()**

```
void Studentas::read (  
    std::istream & is) [override], [virtual]
```

Reads student information from input stream.

**Parameters**

<i>is</i>	Input stream to read from
-----------	---------------------------

Implements [Zmogus](#).

**4.2.3.12 readStudent()**

```
std::istream & Studentas::readStudent (  
    std::istream & is)
```

Read student data from input stream.

**Parameters**

<i>is</i>	Input stream to read from
-----------	---------------------------

**Returns**

Reference to the input stream

**4.2.3.13 setEgzaminas()**

```
void Studentas::setEgzaminas (  
    int egzaminas) [inline]
```

Set exam grade.

**Parameters**

<i>egzaminas</i>	Exam grade to set
------------------	-------------------

**4.2.3.14 setGalutinis()**

```
void Studentas::setGalutinis (  
    double galutinis) [inline]
```

Set final grade.

## Parameters

<i>galutinis</i>	Final grade to set
------------------	--------------------

**4.2.3.15 skaiciuotiMed()**

```
double Studentas::skaiciuotiMed () const
```

Calculate median of homework grades.

## Returns

Median of homework grades

## Exceptions

<i>std::runtime_error</i>	If there are no homework grades
---------------------------	---------------------------------

**4.2.3.16 skaiciuotiVid()**

```
double Studentas::skaiciuotiVid () const
```

Calculate average of homework grades.

## Returns

Average of homework grades

## Exceptions

<i>std::runtime_error</i>	If there are no homework grades
---------------------------	---------------------------------

**4.2.4 Member Data Documentation****4.2.4.1 destruktoriuSk**

```
int Studentas::destruktoriuSk = 0 [static]
```

Static counter for tracking destructor calls (for testing)

**4.2.4.2 egzaminas\_**

```
int Studentas::egzaminas_ [private]
```

Exam score.

#### 4.2.4.3 galutinis\_

```
double Studentas::galutinis_ [private]
```

Final grade.

#### 4.2.4.4 nd\_

```
std::vector<int> Studentas::nd_ [private]
```

Vector of homework (namų darbų) scores.

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

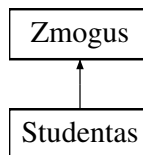
- [funkcijos.h](#)
- [funkcijos.cpp](#)

## 4.3 Zmogus Class Reference

Abstract base class representing a person.

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

Inheritance diagram for Zmogus:



### Public Member Functions

- [Zmogus](#) ()=default  
*Default constructor.*
- [Zmogus](#) (const std::string &[vardas](#), const std::string &[pavarde](#))  
*Constructor with name and surname.*
- [Zmogus](#) (const [Zmogus](#) &other)  
*Copy constructor.*
- [Zmogus](#) & [operator=](#) (const [Zmogus](#) &other)  
*Copy assignment operator.*
- [Zmogus](#) ([Zmogus](#) &&other) noexcept  
*Move constructor.*
- [Zmogus](#) & [operator=](#) ([Zmogus](#) &&other) noexcept  
*Move assignment operator.*
- virtual [~Zmogus](#) ()=0  
*Virtual destructor Pure virtual destructor makes this class abstract.*
- virtual void [print](#) (std::ostream &os) const =0  
*Print person information to output stream.*



- virtual void `read` (std::istream &is)=0  
*Read person information from input stream.*
- std::string `vardas` () const  
*Get first name.*
- std::string `pavarde` () const  
*Get last name.*
- void `setVardas` (const std::string &vardas)  
*Set first name.*
- void `setPavarde` (const std::string &pavarde)  
*Set last name.*

### Protected Attributes

- std::string `vardas_`  
*First name of the person.*
- std::string `pavarde_`  
*Last name of the person.*

## 4.3.1 Detailed Description

Abstract base class representing a person.

This class serves as an abstract base class for all person types. It implements common attributes (name, surname) and declares pure virtual methods.

## 4.3.2 Constructor & Destructor Documentation

### 4.3.2.1 Zmogus() [1/4]

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

Default constructor.

### 4.3.2.2 Zmogus() [2/4]

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

Constructor with name and surname.

#### Parameters

<code>vardas</code>	First name
<code>pavarde</code>	Last name

### 4.3.2.3 Zmogus() [3/4]

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

Copy constructor.

**Parameters**

<i>other</i>	Another person to copy from
--------------	-----------------------------

**4.3.2.4 Zmogus() [4/4]**

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

Move constructor.

**Parameters**

<i>other</i>	Another person to move from
--------------	-----------------------------

**4.3.2.5 ~Zmogus()**

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

Virtual destructor Pure virtual destructor makes this class abstract.

**4.3.3 Member Function Documentation****4.3.3.1 operator=() [1/2]**

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

Copy assignment operator.

**Parameters**

<i>other</i>	Another person to copy from
--------------	-----------------------------

**Returns**

Reference to this person after assignment

**4.3.3.2 operator=() [2/2]**

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

Move assignment operator.

## Parameters

<i>other</i>	Another person to move from
--------------	-----------------------------

## Returns

Reference to this person after assignment

**4.3.3.3 pavarde()**

```
std::string Zmogus::pavarde () const [inline]
```

Get last name.

## Returns

Last name of the person

**4.3.3.4 print()**

```
virtual void Zmogus::print (  
    std::ostream & os) const [pure virtual]
```

Print person information to output stream.

## Parameters

<i>os</i>	Output stream to print to
-----------	---------------------------

Pure virtual method that derived classes must implement

Implemented in [Studentas](#).

**4.3.3.5 read()**

```
virtual void Zmogus::read (  
    std::istream & is) [pure virtual]
```

Read person information from input stream.

## Parameters

<i>is</i>	Input stream to read from
-----------	---------------------------

Pure virtual method that derived classes must implement

Implemented in [Studentas](#).

**4.3.3.6 setPavarde()**

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

Set last name.

**Parameters**

<i>pavarde</i>	Last name to set
----------------	------------------

**4.3.3.7 setVardas()**

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

Set first name.

**Parameters**

<i>vardas</i>	First name to set
---------------	-------------------

**4.3.3.8 vardas()**

```
std::string Zmogus::vardas () const [inline]
```

Get first name.

**Returns**

First name of the person

**4.3.4 Member Data Documentation****4.3.4.1 pavarde\_**

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

Last name of the person.

**4.3.4.2 vardas\_**

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

First name of the person.

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

- [zmogus.h](#)
- [zmogus.cpp](#)

# Chapter 5

## File Documentation

### 5.1 funkcijos.cpp File Reference

```
#include "funkcijos.h"  
#include "laikas.h"
```

#### Functions

- bool [compareByVardas](#) (const [Studentas](#) &a, const [Studentas](#) &b)  
*Compare students by first name.*
- bool [compareByPavarde](#) (const [Studentas](#) &a, const [Studentas](#) &b)  
*Compare students by last name.*
- bool [compareByGalutinis](#) (const [Studentas](#) &a, const [Studentas](#) &b)  
*Compare students by final grade (descending)*
- void [skaitytisFailo](#) (std::vector< [Studentas](#) > &grupe, const std::string &failoPavadinimas)  
*Read students from file.*
- void [investiStudentusIFaila](#) (const std::vector< [Studentas](#) > &studentai, const std::string &failoPavadinimas, char ats)  
*Write students to file.*
- void [sortStudentai](#) (std::vector< [Studentas](#) > &grupe, char sortingOption)  
*Sort students by specified criterion.*
- void [skirstytiStudentus](#) (std::vector< [Studentas](#) > &grupe, std::vector< [Studentas](#) > &kietiakiai, std::vector< [Studentas](#) > &vargsai)  
*Split students into two groups based on final grade.*
- void [testuotiDuomenuApdorojima](#) (const std::string &aplankas, int skaicius)  
*Test data processing performance.*
- void [testuotiStudentoMetodus](#) ()  
*Test [Studentas](#) class methods.*
- void [testuotiZmogausKlase](#) ()  
*Test [Zmogus](#) class.*

## 5.1.1 Function Documentation

### 5.1.1.1 `compareByGalutinis()`

```
bool compareByGalutinis (  
    const Studentas & a,  
    const Studentas & b)
```

Compare students by final grade (descending)

**Parameters**

<i>a</i>	First student
<i>b</i>	Second student

**Returns**

True if a's final grade is higher than b's

**5.1.1.2 compareByPavarde()**

```
bool compareByPavarde (  
    const Studentas & a,  
    const Studentas & b)
```

Compare students by last name.

**Parameters**

<i>a</i>	First student
<i>b</i>	Second student

**Returns**

True if a's surname comes before b's

**5.1.1.3 compareByVardas()**

```
bool compareByVardas (  
    const Studentas & a,  
    const Studentas & b)
```

Compare students by first name.

**Parameters**

<i>a</i>	First student
<i>b</i>	Second student

**Returns**

True if a's name comes before b's

**5.1.1.4 isvestiStudentusIFaila()**

```
void isvestiStudentusIFaila (  
    const std::vector< Studentas > & studentai,  
    const std::string & failoPavadinimas,  
    char ats)
```

Write students to file.

## Parameters

<i>studentai</i>	Students to write
<i>failoPavadinimas</i>	Filename to write to
<i>ats</i>	Whether to use average ('v') or median ('m') for final grade

**5.1.1.5 skaitytiIsFailo()**

```
void skaitytiIsFailo (
    std::vector< Studentas > & grupe,
    const std::string & failoPavadinimas)
```

Read students from file.

## Parameters

<i>grupe</i>	Vector to store students
<i>failoPavadinimas</i>	Filename to read from

**5.1.1.6 skirstytiStudentus()**

```
void skirstytiStudentus (
    std::vector< Studentas > & grupe,
    std::vector< Studentas > & kietiakai,
    std::vector< Studentas > & vargsai)
```

Split students into two groups based on final grade.

## Parameters

<i>grupe</i>	Input vector of students
<i>kietiakai</i>	Output vector for students with final grade $\geq 5.0$
<i>vargsai</i>	Output vector for students with final grade $< 5.0$

**5.1.1.7 sortStudentai()**

```
void sortStudentai (
    std::vector< Studentas > & grupe,
    char sortingOption)
```

Sort students by specified criterion.

## Parameters

<i>grupe</i>	Students to sort
<i>sortingOption</i>	Sorting criterion: 'v' (name), 'p' (surname), 'g' (final grade)

**5.1.1.8 testuotiDuomenuApdorojima()**

```
void testuotiDuomenuApdorojima (
    const std::string & aplankas,
    int skaicius)
```

Test data processing performance.



## Parameters

<i>aplankas</i>	Directory for files
<i>skaicius</i>	Number of students

**5.1.1.9 testuotiStudentoMetodus()**

```
void testuotiStudentoMetodus ()
```

Test [Studentas](#) class methods.

**5.1.1.10 testuotiZmogausKlase()**

```
void testuotiZmogausKlase ()
```

Test [Zmogus](#) class.

**5.2 funkcijos.h File Reference**

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <stdexcept>
#include <execution>
#include <numeric>
#include <cassert>
#include "zmogus.h"
```

**Classes**

- class [Studentas](#)

*Student class that inherits from [Zmogus](#) (Person) abstract class.*

## Functions

- bool `compareByVardas` (const `Studentas` &a, const `Studentas` &b)  
*Compare students by first name.*
- bool `compareByPavarde` (const `Studentas` &a, const `Studentas` &b)  
*Compare students by last name.*
- bool `compareByGalutinis` (const `Studentas` &a, const `Studentas` &b)  
*Compare students by final grade (descending)*
- void `skaitytisFailo` (std::vector< `Studentas` > &grupe, const std::string &failoPavadinimas)  
*Read students from file.*
- void `isvestiStudentusIFaila` (const std::vector< `Studentas` > &studentai, const std::string &failoPavadinimas, char ats)  
*Write students to file.*
- void `sortStudentai` (std::vector< `Studentas` > &grupe, char sortingOption)  
*Sort students by specified criterion.*
- void `skirstytiStudentus` (std::vector< `Studentas` > &grupe, std::vector< `Studentas` > &kietiakiai, std::vector< `Studentas` > &vargsai)  
*Split students into two groups based on final grade.*
- void `testuotiDuomenuApdorojima` (const std::string &aplankas, int skaicius)  
*Test data processing performance.*
- void `testuotiStudentoMetodus` ()  
*Test `Studentas` class methods.*
- void `testuotiZmogausKlase` ()  
*Test `Zmogus` class.*

## 5.2.1 Function Documentation

### 5.2.1.1 `compareByGalutinis()`

```
bool compareByGalutinis (
    const Studentas & a,
    const Studentas & b)
```

Compare students by final grade (descending)

#### Parameters

<i>a</i>	First student
<i>b</i>	Second student

#### Returns

True if a's final grade is higher than b's

### 5.2.1.2 `compareByPavarde()`

```
bool compareByPavarde (
    const Studentas & a,
    const Studentas & b)
```

Compare students by last name.

## Parameters

<i>a</i>	First student
<i>b</i>	Second student

## Returns

True if a's surname comes before b's

**5.2.1.3 compareByVardas()**

```
bool compareByVardas (  
    const Studentas & a,  
    const Studentas & b)
```

Compare students by first name.

## Parameters

<i>a</i>	First student
<i>b</i>	Second student

## Returns

True if a's name comes before b's

**5.2.1.4 isvestiStudentusIFaila()**

```
void isvestiStudentusIFaila (  
    const std::vector< Studentas > & studentai,  
    const std::string & failoPavadinimas,  
    char ats)
```

Write students to file.

## Parameters

<i>studentai</i>	Students to write
<i>failoPavadinimas</i>	Filename to write to
<i>ats</i>	Whether to use average ('v') or median ('m') for final grade

**5.2.1.5 skaitytiIsFailo()**

```
void skaitytiIsFailo (  
    std::vector< Studentas > & grupe,  
    const std::string & failoPavadinimas)
```

Read students from file.

## Parameters

<i>grupe</i>	Vector to store students
<i>failoPavadinimas</i>	Filename to read from

**5.2.1.6 skirstytiStudentus()**

```
void skirstytiStudentus (
    std::vector< Studentas > & grupe,
    std::vector< Studentas > & kietiakiiai,
    std::vector< Studentas > & vargsai)
```

Split students into two groups based on final grade.

## Parameters

<i>grupe</i>	Input vector of students
<i>kietiakiiai</i>	Output vector for students with final grade $\geq 5.0$
<i>vargsai</i>	Output vector for students with final grade $< 5.0$

**5.2.1.7 sortStudentai()**

```
void sortStudentai (
    std::vector< Studentas > & grupe,
    char sortingOption)
```

Sort students by specified criterion.

## Parameters

<i>grupe</i>	Students to sort
<i>sortingOption</i>	Sorting criterion: 'v' (name), 'p' (surname), 'g' (final grade)

**5.2.1.8 testuotiDuomenuApdorojima()**

```
void testuotiDuomenuApdorojima (
    const std::string & aplankas,
    int skaicius)
```

Test data processing performance.

## Parameters

<i>aplankas</i>	Directory for files
<i>skaicius</i>	Number of students

### 5.2.1.9 testuotiStudentoMetodus()

```
void testuotiStudentoMetodus ()
```

Test [Studentas](#) class methods.

### 5.2.1.10 testuotiZmogausKlase()

```
void testuotiZmogausKlase ()
```

Test [Zmogus](#) class.

## 5.3 funkcijos.h

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

```
00001 #ifndef FUNKCIJOS_H
00002 #define FUNKCIJOS_H
00003
00004 #include <iostream>
00005 #include <string>
00006 #include <vector>
00007 #include <algorithm>
00008 #include <iomanip>
00009 #include <fstream>
00010 #include <sstream>
00011 #include <stdexcept>
00012 #include <execution>
00013 #include <numeric>
00014 #include <cassert>
00015 #include "zmogus.h"
00016
00024 class Studentas : public Zmogus
00025 {
00026 // realizacija
00027 private:
00028     std::vector<int> nd_;
00029     int egzaminas_;
00030     double galutinis_;
00031
00032 // interfeisas
00033 public:
00034     static int destruktoriusK;
00035
00041     Studentas() : Zmogus(), egzaminas_(0), galutinis_(0) { }
00042
00047     Studentas(std::istream& is);
00048
00054     Studentas(const std::string& vardas, const std::string& pavarde)
00055         : Zmogus(vardas, pavarde), egzaminas_(0), galutinis_(0) { }
00056
00057 // Rule of Five
00062     Studentas(const Studentas& other);
00063
00069     Studentas& operator=(const Studentas& other);
00070
00075     Studentas(Studentas&& other) noexcept;
00076
00082     Studentas& operator=(Studentas&& other) noexcept;
00083
00088     ~Studentas() override;
00089
00090 // Implementation of pure virtual methods
00095     void print(std::ostream& os) const override;
00096
00101     void read(std::istream& is) override;
00102
00103 // Getteriai
00108     inline std::vector<int> nd() const { return nd_; }
00109
00114     inline int egzaminas() const { return egzaminas_; }
00115
00120     inline double galutinis() const { return galutinis_; }
```

```

00121
00122 // Setteriai
00127 inline void setEgzaminas(int egzaminas) { egzaminas_ = egzaminas; }
00128
00133 inline void setGalutinis(double galutinis) { galutinis_ = galutinis; }
00134
00135 // Metodai
00141 std::istream& readStudent(std::istream& is);
00142
00148 void addND(int pazymys);
00149
00153 void clearND();
00154
00160 double skaiciuotiVid() const;
00161
00167 double skaiciuotiMed() const;
00168
00174 double galBalas(bool naudotiVidurki = true) const;
00175
00181 static void nuskaitymasFile(std::vector<Studentas>& grupe, const std::string& failoPavadinimas);
00182 };
00183
00190 bool compareByVardas(const Studentas& a, const Studentas& b);
00191
00198 bool compareByPavarde(const Studentas& a, const Studentas& b);
00199
00206 bool compareByGalutinis(const Studentas& a, const Studentas& b);
00207
00213 void skaitytiIsFailo(std::vector<Studentas>& grupe, const std::string& failoPavadinimas);
00214
00221 void isvestiStudentusIFaila(const std::vector<Studentas>& studentai, const std::string&
    failoPavadinimas, char ats);
00222
00228 void sortStudentai(std::vector<Studentas>& grupe, char sortingOption);
00229
00236 void skirstytiStudentus(std::vector<Studentas>& grupe, std::vector<Studentas>& kietiakai,
    std::vector<Studentas>& vargsai);
00237
00243 void testuotiDuomenuApdorojima(const std::string& aplankas, int skaicius);
00244
00248 void testuotiStudentoMetodus();
00249
00253 void testuotiZmogausKlase();
00254
00255 #endif // FUNKCIJOS_H

```

## 5.4 laikas.cpp File Reference

```

#include "Laikas.h"
#include "funkcijos.h"

```

## 5.5 laikas.h File Reference

```

#include <chrono>
#include <iostream>

```

### Classes

- class [Laikas](#)

*Class for measuring execution time.*

## 5.6 laikas.h

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

```
00001 #ifndef LAIKAS_H
00002 #define LAIKAS_H
00003
00004 #include <chrono>
00005 #include <iostream>
00006
00014 class Laikas
00015 {
00016 private:
00017     std::chrono::high_resolution_clock::time_point start;
00018     std::chrono::high_resolution_clock::time_point end;
00019     std::string veiksmoPavadinimas;
00020
00021 public:
00026     Laikas(const std::string& pavadinimas);
00027
00032     void pradeti();
00033
00038     void baigti();
00039
00044     double gautiLaikoSkirtuma();
00045 };
00046
00047 #endif
```

## 5.7 main.cpp File Reference

```
#include "funkcijos.h"
#include "laikas.h"
#include "zmogus.h"
```

### Functions

- int [main](#) ()

### 5.7.1 Function Documentation

#### 5.7.1.1 main()

```
int main ()
```

## 5.8 zmogus.cpp File Reference

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

### Functions

- std::ostream & [operator<<](#) (std::ostream &os, const [Zmogus](#) &zmogus)  
*Output operator for [Zmogus](#) class.*
- std::istream & [operator>>](#) (std::istream &is, [Zmogus](#) &zmogus)  
*Input operator for [Zmogus](#) class.*

## 5.8.1 Function Documentation

### 5.8.1.1 operator<<()

```
std::ostream & operator<< (  
    std::ostream & os,  
    const Zmogus & zmogus)
```

Output operator for Zmogus class.

#### Parameters

<i>os</i>	Output stream
<i>zmogus</i>	Person to output

#### Returns

Reference to output stream

### 5.8.1.2 operator>>()

```
std::istream & operator>> (  
    std::istream & is,  
    Zmogus & zmogus)
```

Input operator for Zmogus class.

#### Parameters

<i>is</i>	Input stream
<i>zmogus</i>	Person to input to

#### Returns

Reference to input stream

## 5.9 zmogus.h File Reference

```
#include <string>  
#include <iostream>
```

#### Classes

- class Zmogus

*Abstract base class representing a person.*



## Functions

- `std::ostream & operator<< (std::ostream &os, const Zmogus &zmogus)`  
*Output operator for Zmogus class.*
- `std::istream & operator>> (std::istream &is, Zmogus &zmogus)`  
*Input operator for Zmogus class.*

## 5.9.1 Function Documentation

### 5.9.1.1 operator<<()

```
std::ostream & operator<< (  
    std::ostream & os,  
    const Zmogus & zmogus)
```

Output operator for Zmogus class.

#### Parameters

<i>os</i>	Output stream
<i>zmogus</i>	Person to output

#### Returns

Reference to output stream

### 5.9.1.2 operator>>()

```
std::istream & operator>> (  
    std::istream & is,  
    Zmogus & zmogus)
```

Input operator for Zmogus class.

#### Parameters

<i>is</i>	Input stream
<i>zmogus</i>	Person to input to

#### Returns

Reference to input stream

## 5.10 zmogus.h

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

```

00001 #ifndef ZMOGUS_H
00002 #define ZMOGUS_H
00003
00004 #include <string>
00005 #include <iostream>
00006
00014 class Zmogus
00015 {
00016 protected:
00017     std::string vardas_;
00018     std::string pavarde_;
00019
00020 public:
00024     Zmogus() = default;
00025
00031     Zmogus(const std::string& vardas, const std::string& pavarde)
00032         : vardas_(vardas), pavarde_(pavarde) {}
00033
00034     // Rule of Five
00039     Zmogus(const Zmogus& other) : vardas_(other.vardas_), pavarde_(other.pavarde_) {}
00040
00046     Zmogus& operator=(const Zmogus& other);
00047
00052     Zmogus(Zmogus&& other) noexcept;
00053
00059     Zmogus& operator=(Zmogus&& other) noexcept;
00060
00065     virtual ~Zmogus() = 0;
00066
00073     virtual void print(std::ostream& os) const = 0;
00074
00081     virtual void read(std::istream& is) = 0;
00082
00083     // Getters and setters
00088     inline std::string vardas() const { return vardas_; }
00089
00094     inline std::string pavarde() const { return pavarde_; }
00095
00100     inline void setVardas(const std::string& vardas) { vardas_ = vardas; }
00101
00106     inline void setPavarde(const std::string& pavarde) { pavarde_ = pavarde; }
00107 };
00108
00115 std::ostream& operator<<(std::ostream& os, const Zmogus& zmogus);
00116
00123 std::istream& operator>>(std::istream& is, Zmogus& zmogus);
00124
00125 #endif // ZMOGUS_H

```

# Index

- ~Studentas
  - Studentas, [12](#)
- ~Zmogus
  - Zmogus, [20](#)
- addND
  - Studentas, [12](#)
- baigti
  - Laikas, [8](#)
- clearND
  - Studentas, [13](#)
- compareByGalutinis
  - funkcijos.cpp, [24](#)
  - funkcijos.h, [28](#)
- compareByPavarde
  - funkcijos.cpp, [25](#)
  - funkcijos.h, [28](#)
- compareByVardas
  - funkcijos.cpp, [25](#)
  - funkcijos.h, [29](#)
- destruktoriuSk
  - Studentas, [17](#)
- egzaminas
  - Studentas, [13](#)
- egzaminas\_
  - Studentas, [17](#)
- end
  - Laikas, [8](#)
- funkcijos.cpp, [23](#)
  - compareByGalutinis, [24](#)
  - compareByPavarde, [25](#)
  - compareByVardas, [25](#)
  - investiStudentusIFaila, [25](#)
  - skaitytilsFailo, [26](#)
  - skirstytiStudentus, [26](#)
  - sortStudentai, [26](#)
  - testuotiDuomenuApdorojima, [26](#)
  - testuotiStudentoMetodus, [27](#)
  - testuotiZmogausKlase, [27](#)
- funkcijos.h, [27](#)
  - compareByGalutinis, [28](#)
  - compareByPavarde, [28](#)
  - compareByVardas, [29](#)
  - investiStudentusIFaila, [29](#)
  - skaitytilsFailo, [29](#)
  - skirstytiStudentus, [30](#)
- sortStudentai, [30](#)
- testuotiDuomenuApdorojima, [30](#)
- testuotiStudentoMetodus, [30](#)
- testuotiZmogausKlase, [31](#)
- galBalas
  - Studentas, [13](#)
- galutinis
  - Studentas, [13](#)
- galutinis\_
  - Studentas, [17](#)
- gautiLaikoSkirtuma
  - Laikas, [8](#)
- investiStudentusIFaila
  - funkcijos.cpp, [25](#)
  - funkcijos.h, [29](#)
- Laikas, [7](#)
  - baigti, [8](#)
  - end, [8](#)
  - gautiLaikoSkirtuma, [8](#)
  - Laikas, [7](#)
  - pradeti, [8](#)
  - start, [8](#)
  - veiksmoPavadinimas, [8](#)
- laikas.cpp, [32](#)
- laikas.h, [32](#)
- main
  - main.cpp, [33](#)
- main.cpp, [33](#)
- main, [33](#)
- nd
  - Studentas, [13](#)
- nd\_
  - Studentas, [18](#)
- nuskaitymasFile
  - Studentas, [14](#)
- operator<<
  - zmogus.cpp, [34](#)
  - zmogus.h, [35](#)
- operator>>
  - zmogus.cpp, [34](#)
  - zmogus.h, [35](#)
- operator=
  - Studentas, [14](#)
  - Zmogus, [20](#)

pavarde  
     Zmogus, 21  
 pavarde\_  
     Zmogus, 22  
 pradeti  
     Laikas, 8  
 print  
     Studentas, 14  
     Zmogus, 21  
  
 read  
     Studentas, 16  
     Zmogus, 21  
 readStudent  
     Studentas, 16  
  
 setEgzaminas  
     Studentas, 16  
 setGalutinis  
     Studentas, 16  
 setPavarde  
     Zmogus, 21  
 setVardas  
     Zmogus, 22  
 skaiciuotiMed  
     Studentas, 17  
 skaiciuotiVid  
     Studentas, 17  
 skaitytisFailo  
     funkcijos.cpp, 26  
     funkcijos.h, 29  
 skirstytiStudentus  
     funkcijos.cpp, 26  
     funkcijos.h, 30  
 sortStudentai  
     funkcijos.cpp, 26  
     funkcijos.h, 30  
 start  
     Laikas, 8  
 Studentas, 9  
     ~Studentas, 12  
     addND, 12  
     clearND, 13  
     destruktoriuSk, 17  
     egzaminas, 13  
     egzaminas\_, 17  
     galBalas, 13  
     galutinis, 13  
     galutinis\_, 17  
     nd, 13  
     nd\_, 18  
     nuskaitymasFile, 14  
     operator=, 14  
     print, 14  
     read, 16  
     readStudent, 16  
     setEgzaminas, 16  
     setGalutinis, 16  
     skaiciuotiMed, 17  
     skaiciuotiVid, 17  
     Studentas, 11, 12  
  
 testuotiDuomenuApdorojima  
     funkcijos.cpp, 26  
     funkcijos.h, 30  
 testuotiStudentoMetodus  
     funkcijos.cpp, 27  
     funkcijos.h, 30  
 testuotiZmogausKlase  
     funkcijos.cpp, 27  
     funkcijos.h, 31  
  
 vardas  
     Zmogus, 22  
 vardas\_  
     Zmogus, 22  
 veiksmoPavadinimas  
     Laikas, 8  
  
 Zmogus, 18  
     ~Zmogus, 20  
     operator=, 20  
     pavarde, 21  
     pavarde\_, 22  
     print, 21  
     read, 21  
     setPavarde, 21  
     setVardas, 22  
     vardas, 22  
     vardas\_, 22  
     Zmogus, 19, 20  
 zmogus.cpp, 33  
     operator<<, 34  
     operator>>, 34  
 zmogus.h, 34  
     operator<<, 35  
     operator>>, 35