v2.0

Generated by Doxygen 1.11.0

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

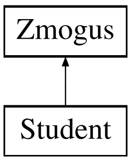
Class Documentation

4.1 Student Class Reference

Class representing a student, derived from Zmogus.

```
#include <student.h>
```

Inheritance diagram for Student:



Public Member Functions

- Student ()

    Default constructor.
- ∼Student ()

    Destructor.
- void setName (const string &vardas) override

    Set the name of the person.
- string getName () const override

    Get the name of the person.
- void setSurname (const string &pavarde) override

    Set the surname of the person.
- string getSurname () const override

    Get the surname of the person.
- void setExamResult (int egzaminas)

    Set the exam result.
- int getExamResult () const

    Get the exam result.
- void setFinalAvg (double Gal_vid)

    Set the final average.
- double getFinalAvg () const

Get the final average.
- void setFinalMedian (double Gal_med)

  Set the final median.
- double getFinalMedian () const

  Get the final median.
- void setSingleGrade (int naujasnd)

  Add a single grade.
- int getSingleGrade (int i) const

  Get a single grade by index.
- void setGrades (const vector< int > &ND)

  Set all grades.
- vector< int > getGrades () const

  Get all grades.
- void clearData ()

  Clear all student data.

Public Member Functions inherited from Zmogus

- virtual ~Zmogus ()

  Virtual destructor.

Friends

- std::istream & operator>> (istream &in, Student &student)

  Input operator for Student.
- ostream & operator<< (ostream &out, const Student &student)

  Output operator for Student.

Additional Inherited Members

Protected Attributes inherited from Zmogus

- string name

  Name of the person.
- string surname

  Surname of the person.

4.1.1 Detailed Description

Class representing a student, derived from Zmogus.

The Student class contains additional attributes and methods specific to a student, including grades, exam results, and methods to calculate averages and medians.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Student()

```
Student::Student ()  [inline]
```

Default constructor.

4.1.2.2 ∼Student()

```
Student::∼Student ()  [inline]
```

Destructor.

4.1.3 Member Function Documentation

4.1.3.1 clearData()

```
void Student::clearData ()  [inline]
```

Clear all student data.

4.1.3.2 getExamResult()

```
int Student::getExamResult () const  [inline]
```

Get the exam result.

Returns

The exam result.

4.1.3.3 getFinalAvg()

```
double Student::getFinalAvg () const  [inline]
```

Get the final average.

Returns

The final average.

4.1.3.4 getFinalMedian()

```
double Student::getFinalMedian () const  [inline]
```

Get the final median.

Returns

The final median.

4.1.3.5 getGrades()

```
vector< int > Student::getGrades () const  [inline]
```

Get all grades.

Returns

A vector of all grades.

4.1.3.6 getName()

```
string Student::getName () const  [inline], [override], [virtual]
```

Get the name of the person.

Returns

The name of the person.

Implements Zmogus.

4.1.3.7 getSingleGrade()

```
int Student::getSingleGrade (
            int i) const  [inline]
```

Get a single grade by index.

Parameters

i	The index of the grade.

Returns

The grade at the specified index.

4.1.3.8 getSurname()

```
string Student::getSurname () const  [inline], [override], [virtual]
```

Get the surname of the person.

Returns

The surname of the person.

Implements Zmogus.

4.1.3.9 setExamResult()

```
void Student::setExamResult (
             int egzaminas)  [inline]
```

Set the exam result.

Parameters

egzaminas	The exam result to set.

4.1.3.10 setFinalAvg()

```
void Student::setFinalAvg (
             double Gal_vid)  [inline]
```

Set the final average.

Parameters

Gal_vid	The final average to set.

4.1.3.11 setFinalMedian()

```
void Student::setFinalMedian (
             double Gal_med)  [inline]
```

Set the final median.

Parameters

Gal_med	The final median to set.

4.1.3.12 setGrades()

```
void Student::setGrades (
             const vector< int > & ND)  [inline]
```

Set all grades.

Parameters

ND	Vector of grades to set.

4.1.3.13 setName()

```
void Student::setName (
             const string & vardas)  [inline], [override], [virtual]
```

Set the name of the person.

Parameters

vardas	The name to set.

Implements Zmogus.

4.1.3.14 setSingleGrade()

```
void Student::setSingleGrade (
             int naujasnd)  [inline]
```

Add a single grade.

Parameters

naujasnd	The grade to add.

4.1.3.15 setSurname()

```
void Student::setSurname (
             const string & pavarde)  [inline], [override], [virtual]
```

Set the surname of the person.

Parameters

pavarde	The surname to set.

Implements Zmogus.

4.1.4 Friends And Related Symbol Documentation

4.1.4.1 operator<<

```
ostream & operator<< (
             ostream & out,
             const Student & student)  [friend]
```

Output operator for Student.

Parameters

out	Output stream.
student	Student object to output data from.

Returns

Reference to the output stream.

4.1.4.2 operator>>

```
std::istream & operator>> (
             istream & in,
             Student & student)  [friend]
```

Input operator for Student.

Parameters

in	Input stream.
student	Student object to input data into.

Returns

Reference to the input stream.

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

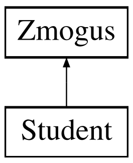
- Desktop/v2.0/student.h

4.2 Zmogus Class Reference

Abstract base class for a person.

```
#include <student.h>
```

Inheritance diagram for Zmogus:



Public Member Functions

- virtual void setName (const string &vardas)=0

  Set the name of the person.
- virtual string getName () const =0

  Get the name of the person.
- virtual void setSurname (const string &pavarde)=0

  Set the surname of the person.
- virtual string getSurname () const =0

  Get the surname of the person.
- virtual ∼Zmogus ()

  Virtual destructor.

Protected Attributes

- string name

  Name of the person.
- string surname

  Surname of the person.

4.2.1 Detailed Description

Abstract base class for a person.

The Zmogus class provides a base for derived classes representing people, with pure virtual methods for setting and getting names and surnames.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 ∼Zmogus()

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

Virtual destructor.

4.2.3 Member Function Documentation

4.2.3.1 getName()

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

Get the name of the person.

Returns

The name of the person.

Implemented in Student.

4.2.3.2 getSurname()

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

Get the surname of the person.

Returns

The surname of the person.

Implemented in Student.

4.2.3.3 setName()

```
virtual void Zmogus::setName (
             const string & vardas)  [pure virtual]
```

Set the name of the person.

Parameters

vardas	The name to set.

Implemented in Student.

4.2.3.4 setSurname()

```
virtual void Zmogus::setSurname (
             const string & pavarde)  [pure virtual]
```

Set the surname of the person.

Parameters

pavarde	The surname to set.

Implemented in Student.

4.2.4 Member Data Documentation

4.2.4.1 name

```
string Zmogus::name  [protected]
```

Name of the person.

4.2.4.2 surname

```
string Zmogus::surname  [protected]
```

Surname of the person.

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

- Desktop/v2.0/student.h

Chapter 5

File Documentation

5.1 Desktop/v2.0/main.cpp File Reference

```
#include "student.h"
```

Functions

- int main ()
  The main function of the program.

5.1.1 Function Documentation

5.1.1.1 main()

```
int main ()
```

The main function of the program.

Returns

int Returns 0 upon successful execution.

< Number of students

< Number of homework assignments

< Initial time point for generating

< User's menu choice

< Vector to store student objects

Handles case '3' for displaying randomly generated student data.

Parameters

none	

Returns

void

Handles case '4' for reading student data from a file and displaying it.

Parameters

none	

Returns

void

Case '5' of the main menu: Perform sorting and display data in a specified format.

This case reads data from a file, sorts it based on user preference, and outputs the sorted data to a file. The user can choose to display either the average or median of student grades.

Parameters

students	Vector containing student objects.
hw	Vector containing homework grades.

Executes the selected functionality based on user input until the user chooses to exit.

Returns

0 upon successful execution.

5.2 Desktop/v2.0/student.cpp File Reference

```
#include "student.h"
```

Functions

- void generateRandomGrades (vector< Student > &students, double hw)

  Generates random grades for students.
- void generateRandomData (vector< Student > &students, double hw)

  Generates random data for students.
- double calculateMedian (vector< int > &arr)

  Calculates the median of a vector of integers.
- void readDataFromFile (vector< Student > &students, double &hw, int N)

  Reads data from a file and populates a vector of Student objects.
- void enterDataManually (vector< Student > &students, double hw)

  Manually enters data for students.
- bool compareByName (const Student &a, const Student &b)

  Comparator function to compare students by name.
- bool compareBySurname (const Student &a, const Student &b)

  Comparator function to compare students by surname.
- bool compareByMedian (const Student &a, const Student &b)

  Comparator function to compare students by final median grade.
- bool compareByAvg (const Student &a, const Student &b)

  Comparator function to compare students by final average grade.
- int generateRandomNumber (int min, int max)

  Function to generate a random number within a specified range.
- double calculateAverage (const vector< int > &pazymiai)

  Function to calculate the average of a vector of integers.
- void failuGeneravimas (int studentu_kiekis, const std::string &failo_pavadinimas)

  Function to generate a file with student data.
- bool Nuskaitymas (const std::string &failo_pavadinimas, std::vector< Student > &students, int studentukiekis)

  Reads student data from a file and populates a vector of Student objects.
- std::chrono::steady_clock::time_point DabartinisLaikas ()

  Returns the current system time.
- double LaikoSkirtumas (const std::chrono::steady_clock::time_point &pradzia, const std::chrono::steady_↩
clock::time_point &pabaiga)

  Calculates the time difference between two time points.
- void calculateResults (std::vector< Student > &stud)

  Calculates final results for each student.
- double skaiciuotiVidurki (const std::vector< int > &pazymiai)

  Function to calculate the average based on student grades.
- bool arGerasStudentas (const Student &student)

  Function to check if a student is good based on the average grade.
- bool lygintiPagalVidurki (const Student &a, const Student &b)

  Function to compare two students based on their average grades.
- void rūšiuoja_ir_rašo_failus (std::vector< Student > &students)

  Function to sort students, separate them into good and bad, and write to files.
- void testavimoRezultatai (bool success, const std::string &testName)

  Function to display test results.
- void testas ()

  A function to test various operations of the Student class.

5.2.1 Function Documentation

5.2.1.1 arGerasStudentas()

```
bool arGerasStudentas (
            const Student & student)
```

Function to check if a student is good based on the average grade.

Check if a student is a good student.

Parameters

student	The Student object to evaluate.

Returns

bool True if the student's average grade is greater than or equal to 5.0, false otherwise.

5.2.1.2 calculateAverage()

```
double calculateAverage (
            const vector< int > & pazymiai)
```

Function to calculate the average of a vector of integers.

Calculate the average of a vector of integers.

Parameters

pazymiai	A vector of integers representing grades.

Returns

double The average grade.

5.2.1.3 calculateMedian()

```
double calculateMedian (
            vector< int > & arr)
```

Calculates the median of a vector of integers.

Calculate the median of a vector of integers.

Parameters

arr	Vector of integers.

Returns

Median value.

5.2.1.4 calculateResults()

```
void calculateResults (
             std::vector< Student > & stud)
```

Calculates final results for each student.

Calculate results for a list of students.

This function calculates final average and median grades for each student based on their individual grades and exam results. It updates the Student objects accordingly.

Parameters

stud	Reference to a vector of Student objects.

5.2.1.5 compareByAvg()

```
bool compareByAvg (
             const Student & a,
             const Student & b)
```

Comparator function to compare students by final average grade.

Compare two students by average grade.

5.2.1.6 compareByMedian()

```
bool compareByMedian (
             const Student & a,
             const Student & b)
```

Comparator function to compare students by final median grade.

Compare two students by median grade.

5.2.1.7 compareByName()

```
bool compareByName (
             const Student & a,
             const Student & b)
```

Comparator function to compare students by name.

Compare two students by name.

5.2.1.8 compareBySurname()

```
bool compareBySurname (
            const Student & a,
            const Student & b)
```

Comparator function to compare students by surname.

Compare two students by surname.

5.2.1.9 DabartinisLaikas()

```
std::chrono::steady_clock::time_point DabartinisLaikas ()
```

Returns the current system time.

Get the current time point.

Returns

The current system time as a steady_clock::time_point object.

5.2.1.10 enterDataManually()

```
void enterDataManually (
            vector< Student > & students,
            double hw)
```

Manually enters data for students.

Enter student data manually.

Parameters

students	Vector of Student objects.
hw	Number of homework assignments.

Function to manually enter student data.

This function allows manual entry of student data, including names, homework grades, and exam results. It calculates the final average grade for each student based on the provided weights.

Parameters

students	A vector of Student objects to store student data.
hw	The number of homework assignments.

5.2.1.11 failuGeneravimas()

```
void failuGeneravimas (
            int studentu_kiekis,
            const std::string & failo_pavadinimas)
```

Function to generate a file with student data.

Generate student data files.

This function generates a file containing student data with randomly generated grades for each homework assignment and exam.

Parameters

studentu_kiekis	The number of students.
failo_pavadinimas	The filename for the generated file.

5.2.1.12 generateRandomData()

```
void generateRandomData (
             vector< Student > & students,
             double hw)
```

Generates random data for students.

Generate random data for a list of students.

Parameters

students	Vector of Student objects.
hw	Number of homework assignments.

5.2.1.13 generateRandomGrades()

```
void generateRandomGrades (
             vector< Student > & students,
             double hw)
```

Generates random grades for students.

Generate random grades for a list of students.

Parameters

students	Vector of Student objects.
hw	Number of homework assignments.

5.2.1.14 generateRandomNumber()

```
int generateRandomNumber (
             int min,
             int max)
```

Function to generate a random number within a specified range.

Generate a random number within a range.

Parameters

min	The minimum value of the range.
max	The maximum value of the range.

Returns

int A random integer within the specified range.

5.2.1.15 LaikoSkirtumas()

```
double LaikoSkirtumas (
            const std::chrono::steady_clock::time_point & pradzia,
            const std::chrono::steady_clock::time_point & pabaiga)
```

Calculates the time difference between two time points.

Calculate the time difference between two time points.

Parameters

pradzia	The starting time point.
pabaiga	The ending time point.

Returns

The time difference between the two time points in seconds.

5.2.1.16 lygintiPagalVidurki()

```
bool lygintiPagalVidurki (
            const Student & a,
            const Student & b)
```

Function to compare two students based on their average grades.

Compare two students by average grade.

Parameters

a	The first Student object.
b	The second Student object.

Returns

bool True if the average grade of student 'a' is greater than that of student 'b', false otherwise.

5.2.1.17 Nuskaitymas()

```
bool Nuskaitymas (
            const std::string & failo_pavadinimas,
            std::vector< Student > & students,
            int studentukiekis)
```

Reads student data from a file and populates a vector of Student objects.

Read student data from a file.

This function reads student data from a specified file, extracts student names and grades, calculates exam results, and populates a vector of Student objects.

Parameters

failo_pavadinimas	The name of the file to read student data from.
students	Reference to a vector of Student objects to populate.
studentukiekis	The number of students expected in the file.

Returns

True if the file is successfully opened and data is read, false otherwise.

5.2.1.18 rūšiuoja_ir_rašo_failus()

```
void rūšiuoja_ir_rašo_failus (
            std::vector< Student > & students)
```

Function to sort students, separate them into good and bad, and write to files.

Sort students and write to files.

Parameters

students	Vector of Student objects.

5.2.1.19 readDataFromFile()

```
void readDataFromFile (
            vector< Student > & students,
            double & hw,
            int N)
```

Reads data from a file and populates a vector of Student objects.

Read student data from a file.

Parameters

students	Vector of Student objects.
hw	Number of homework assignments.
N	Maximum number of students to read.

5.2.1.20 skaiciuotiVidurki()

```
double skaiciuotiVidurki (
            const std::vector< int > & pazymiai)
```

Function to calculate the average based on student grades.

Calculate the average of grades.

Parameters

pazymiai	Vector of integers representing student grades.

Returns

double The calculated average of the grades.

5.2.1.21 testas()

`void testas ()`

A function to test various operations of the Student class.

Run test functions.

This function tests the default constructor, setters, getters, copy constructor, move constructor, copy assignment, and move assignment of the Student class. < Assert that name is empty

< Assert that surname is empty

< Assert that exam result is 0

< Assert that final average is 0.0

< Assert that final median is 0.0

< Set name

< Assert that name is set correctly

< Set surname

< Assert that surname is set correctly

< Set exam result

< Assert that exam result is set correctly

< Set final average

< Assert that final average is set correctly

< Set final median

< Assert that final median is set correctly

< Assert that copied name is correct

< Assert that copied surname is correct

< Assert that copied exam result is correct

< Test copy constructor

< Assert that moved name is correct

< Assert that moved surname is correct

< Assert that moved exam result is correct

< Test move constructor

< Assert that copied name is correct

< Assert that copied surname is correct

< Assert that copied exam result is correct

< Test copy assignment

< Assert that moved name is correct

< Assert that moved surname is correct

< Assert that moved exam result is correct

< Test move assignment

5.2.1.22 testavimoRezultatai()

```
void testavimoRezultatai (
             bool success,
             const std::string & testName)
```

Function to display test results.

Parameters

success	Indicates whether the test passed (true) or failed (false).
testName	The name of the test.

5.3 Desktop/v2.0/student.h File Reference

This file contains the declaration of the Zmogus and Student classes and related functions.

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <limits>
#include <string>
#include <vector>
#include <sstream>
#include <algorithm>
#include <numeric>
#include <random>
#include <ctime>
#include <cstring>
#include <cassert>
#include <utility>
#include <chrono>
```

Classes

- class Zmogus

  Abstract base class for a person.
- class Student

  Class representing a student, derived from Zmogus.

Functions

- void generateRandomGrades (vector< Student > &students, double hw)

  Generate random grades for a list of students.
- void generateRandomData (vector< Student > &students, double hw)

  Generate random data for a list of students.
- double calculateMedian (vector< int > &arr)

  Calculate the median of a vector of integers.
- void readDataFromFile (vector< Student > &students, double &hw, int N)

Read student data from a file.
- void enterDataManually (vector< Student > &students, double hw)

  Enter student data manually.
- bool compareByName (const Student &a, const Student &b)

  Compare two students by name.
- bool compareBySurname (const Student &a, const Student &b)

  Compare two students by surname.
- bool compareByMedian (const Student &a, const Student &b)

  Compare two students by median grade.
- bool compareByAvg (const Student &a, const Student &b)

  Compare two students by average grade.
- int generateRandomNumber (int min, int max)

  Generate a random number within a range.
- double calculateAverage (const vector< int > &pazymiai)

  Calculate the average of a vector of integers.
- void failuGeneravimas (int studentu_kiekis, const std::string &failo_pavadinimas)

  Generate student data files.
- bool Nuskaitymas (const std::string &failo_pavadinimas, std::vector< Student > &students, int studen-tukiekis)

  Read student data from a file.
- std::chrono::steady_clock::time_point DabartinisLaikas ()

  Get the current time point.
- double LaikoSkirtumas (const std::chrono::steady_clock::time_point &pradzia, const std::chrono::steady_↩clock::time_point &pabaiga)

  Calculate the time difference between two time points.
- void calculateResults (std::vector< Student > &stud)

  Calculate results for a list of students.
- double skaiciuotiVidurki (const std::vector< int > &pazymiai)

  Calculate the average of grades.
- bool arGerasStudentas (const Student &student)

  Check if a student is a good student.
- bool lygintiPagalVidurki (const Student &a, const Student &b)

  Compare two students by average grade.
- void rūšiuoja_ir_rašo_failus (std::vector< Student > &students)

  Sort students and write to files.
- void testas ()

  Run test functions.

5.3.1 Detailed Description

This file contains the declaration of the Zmogus and Student classes and related functions.

The file provides a detailed description of the Zmogus base class and the Student derived class, along with several functions for handling student data.

5.3.2 Function Documentation

5.3.2.1 arGerasStudentas()

```
bool arGerasStudentas (
            const Student & student)
```

Check if a student is a good student.

Parameters

student	Student to check.

Returns

True if the student is good, false otherwise.

Check if a student is a good student.

Parameters

student	The Student object to evaluate.

Returns

bool True if the student's average grade is greater than or equal to 5.0, false otherwise.

5.3.2.2 calculateAverage()

```
double calculateAverage (
            const vector< int > & pazymiai)
```

Calculate the average of a vector of integers.

Parameters

pazymiai	Vector of integers.

Returns

The average value.

Calculate the average of a vector of integers.

Parameters

pazymiai	A vector of integers representing grades.

Returns

double The average grade.

5.3.2.3 calculateMedian()

```
double calculateMedian (
            vector< int > & arr)
```

Calculate the median of a vector of integers.

Parameters

arr	Vector of integers.

Returns

The median value.

Calculate the median of a vector of integers.

Parameters

arr	Vector of integers.

Returns

Median value.

5.3.2.4 calculateResults()

```
void calculateResults (
             std::vector< Student > & stud)
```

Calculate results for a list of students.

Parameters

stud	Vector of students.

Calculate results for a list of students.

This function calculates final average and median grades for each student based on their individual grades and exam results. It updates the Student objects accordingly.

Parameters

stud	Reference to a vector of Student objects.

5.3.2.5 compareByAvg()

```
bool compareByAvg (
             const Student & a,
             const Student & b)
```

Compare two students by average grade.

Parameters

a	First student.
b	Second student.

Returns

True if a's average is less than b's average.

Compare two students by average grade.

5.3.2.6 compareByMedian()

```
bool compareByMedian (
            const Student & a,
            const Student & b)
```

Compare two students by median grade.

Parameters

a	First student.
b	Second student.

Returns

True if a's median is less than b's median.

Compare two students by median grade.

5.3.2.7 compareByName()

```
bool compareByName (
            const Student & a,
            const Student & b)
```

Compare two students by name.

Parameters

a	First student.
b	Second student.

Returns

True if a's name is less than b's name.

Compare two students by name.

5.3.2.8 compareBySurname()

```
bool compareBySurname (
            const Student & a,
            const Student & b)
```

Compare two students by surname.

Parameters

a	First student.
b	Second student.

Returns

True if a's surname is less than b's surname.

Compare two students by surname.

5.3.2.9 DabartinisLaikas()

```
std::chrono::steady_clock::time_point DabartinisLaikas ()
```

Get the current time point.

Returns

Current time point.

Get the current time point.

Returns

The current system time as a steady_clock::time_point object.

5.3.2.10 enterDataManually()

```
void enterDataManually (
            vector< Student > & students,
            double hw)
```

Enter student data manually.

Parameters

students	Vector to store student data.
hw	Homework weight.

Enter student data manually.

Parameters

students	Vector of Student objects.
hw	Number of homework assignments.

Function to manually enter student data.

This function allows manual entry of student data, including names, homework grades, and exam results. It calculates the final average grade for each student based on the provided weights.

Parameters

students	A vector of Student objects to store student data.
hw	The number of homework assignments.

5.3.2.11 failuGeneravimas()

```
void failuGeneravimas (
             int studentu_kiekis,
             const std::string & failo_pavadinimas)
```

Generate student data files.

Parameters

studentu_kiekis	Number of students.
failo_pavadinimas	File name.

Generate student data files.

This function generates a file containing student data with randomly generated grades for each homework assignment and exam.

Parameters

studentu_kiekis	The number of students.
failo_pavadinimas	The filename for the generated file.

5.3.2.12 generateRandomData()

```
void generateRandomData (
             vector< Student > & students,
             double hw)
```

Generate random data for a list of students.

Parameters

students	Vector of students.
hw	Homework weight.

Generate random data for a list of students.

Parameters

students	Vector of Student objects.
hw	Number of homework assignments.

5.3.2.13 generateRandomGrades()

```
void generateRandomGrades (
             vector< Student > & students,
             double hw)
```

Generate random grades for a list of students.

Parameters

students	Vector of students.
hw	Homework weight.

Generate random grades for a list of students.

Parameters

students	Vector of Student objects.
hw	Number of homework assignments.

5.3.2.14 generateRandomNumber()

```
int generateRandomNumber (
             int min,
             int max)
```

Generate a random number within a range.

Parameters

min	Minimum value.
max	Maximum value.

Returns

A random number between min and max.

Generate a random number within a range.

Parameters

min	The minimum value of the range.
max	The maximum value of the range.

Returns

int A random integer within the specified range.

5.3.2.15 LaikoSkirtumas()

```
double LaikoSkirtumas (
             const std::chrono::steady_clock::time_point & pradzia,
             const std::chrono::steady_clock::time_point & pabaiga)
```

Calculate the time difference between two time points.

Parameters

pradzia	Start time point.
pabaiga	End time point.

Returns

Time difference in seconds.

Calculate the time difference between two time points.

Parameters

pradzia	The starting time point.
pabaiga	The ending time point.

Returns

The time difference between the two time points in seconds.

5.3.2.16 lygintiPagalVidurki()

```
bool lygintiPagalVidurki (
             const Student & a,
             const Student & b)
```

Compare two students by average grade.

Parameters

a	First student.
b	Second student.

Returns

True if a's average is greater than b's average.

Compare two students by average grade.

Parameters

a	The first Student object.
b	The second Student object.

Returns

bool True if the average grade of student 'a' is greater than that of student 'b', false otherwise.

5.3.2.17 Nuskaitymas()

```
bool Nuskaitymas (
             const std::string & failo_pavadinimas,
             std::vector< Student > & students,
             int studentukiekis)
```

Read student data from a file.

Parameters

failo_pavadinimas	File name.
students	Vector to store student data.
studentukiekis	Number of students.

Returns

True if successful, false otherwise.

Read student data from a file.

This function reads student data from a specified file, extracts student names and grades, calculates exam results, and populates a vector of Student objects.

Parameters

failo_pavadinimas	The name of the file to read student data from.
students	Reference to a vector of Student objects to populate.
studentukiekis	The number of students expected in the file.

Returns

True if the file is successfully opened and data is read, false otherwise.

5.3.2.18 rūšiuoja_ir_rašo_failus()

```
void rūšiuoja_ir_rašo_failus (
            std::vector< Student > & students)
```

Sort students and write to files.

Parameters

students	Vector of students.

Sort students and write to files.

Parameters

students	Vector of Student objects.

5.3.2.19 readDataFromFile()

```
void readDataFromFile (
            vector< Student > & students,
            double & hw,
            int N)
```

Read student data from a file.

Parameters

students	Vector to store student data.
hw	Homework weight.
N	Number of students.

Read student data from a file.

Parameters

students	Vector of Student objects.
hw	Number of homework assignments.
N	Maximum number of students to read.

5.3.2.20 skaiciuotiVidurki()

```
double skaiciuotiVidurki (
            const std::vector< int > & pazymiai)
```

Calculate the average of grades.

Parameters

pazymiai	Vector of grades.

Returns

The average grade.

Calculate the average of grades.

Parameters

pazymiai	Vector of integers representing student grades.

Returns

double The calculated average of the grades.

5.3.2.21 testas()

```
void testas ()
```

Run test functions.

Run test functions.

This function tests the default constructor, setters, getters, copy constructor, move constructor, copy assignment, and move assignment of the Student class. < Assert that name is empty

< Assert that surname is empty

< Assert that exam result is 0

< Assert that final average is 0.0

< Assert that final median is 0.0

< Set name

< Assert that name is set correctly

< Set surname

< Assert that surname is set correctly

< Set exam result

< Assert that exam result is set correctly

< Set final average

< Assert that final average is set correctly

< Set final median

< Assert that final median is set correctly

< Assert that copied name is correct

< Assert that copied surname is correct

< Assert that copied exam result is correct

< Test copy constructor

< Assert that moved name is correct

< Assert that moved surname is correct

< Assert that moved exam result is correct

< Test move constructor

< Assert that copied name is correct

< Assert that copied surname is correct

< Assert that copied exam result is correct

< Test copy assignment

< Assert that moved name is correct

< Assert that moved surname is correct

< Assert that moved exam result is correct

< Test move assignment

5.4 student.h

Go to the documentation of this file.

```cpp

#ifndef STUDENT_H
#define STUDENT_H


#include <iostream>
#include <fstream>
#include <iomanip>
#include <limits>
#include <string>
#include <vector>
#include <sstream>
#include <algorithm>
#include <numeric>
#include <random>
#include <ctime>
#include <cstring>
#include <cassert>
#include <utility>
#include <chrono>


using namespace std;
using namespace std::chrono;

class Zmogus {
protected:
    string name;
    string surname;

public:
    virtual void setName(const string& vardas) = 0;

    virtual string getName() const = 0;

    virtual void setSurname(const string& pavarde) = 0;

    virtual string getSurname() const = 0;

    virtual ~Zmogus() {}
};

class Student : public Zmogus {
private:
    vector<int> grades;
    int exam_result;
    double final_avg;
    double final_median;

public:
    Student() : grades(), exam_result(0), final_avg(0.0), final_median(0.0) {}

    ~Student() {}

    // Implementing abstract methods from Zmogus
    void setName(const string& vardas) override {
        name = vardas;
    }

    string getName() const override {
        return name;
    }

    void setSurname(const string& pavarde) override {
        surname = pavarde;
    }

    string getSurname() const override {
        return surname;
    }

    // Additional methods for Student
    void setExamResult(int egzaminas) {
        exam_result = egzaminas;
    }

    int getExamResult() const {
        return exam_result;
    }

    void setFinalAvg(double Gal_vid) {
        final_avg = Gal_vid;
    }

    double getFinalAvg() const {
        return final_avg;
    }

    void setFinalMedian(double Gal_med) {
        final_median = Gal_med;
    }

    double getFinalMedian() const {
        return final_median;
    }

    void setSingleGrade(int naujasnd) {
        grades.push_back(naujasnd);
    }

    int getSingleGrade(int i) const {
        return grades[i];
    }

    void setGrades(const vector<int>& ND) {
        grades = ND;
    }

    vector<int> getGrades() const {
        return grades;
    }

    void clearData() {
        grades.clear();
        exam_result = 0;
        final_avg = 0.0;
        final_median = 0.0;
    }

    friend std::istream& operator»(istream& in, Student& student) {
        string vardas, pavarde;
        int egzaminas;

        if (!(in » vardas » pavarde » egzaminas)){
            in.clear();
            in.ignore(numeric_limits<streamsize>::max(), '\n');
            return in;
        }
        student.setName(vardas);
        student.setSurname(pavarde);
        student.setExamResult(egzaminas);

        return in;
    }

    friend ostream& operator«(ostream& out, const Student& student) {
        out « "Vardas: " « student.getName() « endl;
        out « "Pavarde: " « student.getSurname() « endl;
        out « "Egzamino rezultatas: " « student.getExamResult() « endl;
        out « "Pazymiai: ";
        vector<int> grades = student.getGrades();
        for (int i = 0; i < grades.size(); i++){
            out « grades[i] « " ";
        }
        out « endl;
        return out;
    }
};

// Function declarations with brief descriptions

void generateRandomGrades(vector<Student>& students, double hw);

void generateRandomData(vector<Student>& students, double hw);

double calculateMedian(vector<int>& arr);

void readDataFromFile(vector<Student>& students, double& hw, int N);

void enterDataManually(vector<Student>& students, double hw);

bool compareByName(const Student& a, const Student& b);

bool compareBySurname(const Student& a, const Student& b);

bool compareByMedian(const Student& a, const Student& b);

bool compareByAvg(const Student& a, const Student& b);

```

```
int generateRandomNumber(int min, int max);

double calculateAverage(const vector<int>& pazymiai);

void failuGeneravimas(int studentu_kiekis, const std::string& failo_pavadinimas);

bool Nuskaitymas(const std::string& failo_pavadinimas, std::vector<Student>& students, int
studentukiekis);

std::chrono::steady_clock::time_point DabartinisLaikas();

double LaikoSkirtumas(const std::chrono::steady_clock::time_point& pradzia, const
std::chrono::steady_clock::time_point& pabaiga);

void calculateResults(std::vector<Student>& stud);

double skaiciuotiVidurki(const std::vector<int>& pazymiai);

bool arGerasStudentas(const Student& student);

bool lygintiPagalVidurki(const Student& a, const Student& b);

void rūšiuoja_ir_rašo_failus(std::vector<Student>& students);

void testas();

#endif // STUDENT_H
```