

v2.0

Generated by Doxygen 1.10.0

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Student Class Reference	7
4.1.1 Member Function Documentation	8
4.1.1.1 ppp()	8
4.2 Zmogus Class Reference	8
5 File Documentation	11
5.1 failu-generavimas.h	11
5.2 student.h	11
5.3 vektoriai.h	13
Index	15

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Zmogus	8
Student	7

Chapter 2

Class Index

2.1 Class List

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

Student	7
Zmogus	8

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

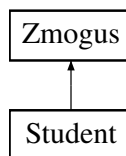
sources/ failu-generavimas.h	11
sources/ student.h	11
sources/ vektoriai.h	13

Chapter 4

Class Documentation

4.1 Student Class Reference

Inheritance diagram for Student:



Public Member Functions

- void `ppp` () const override
- **Student** (const string &fName, const string &lName, const vector< int > &grades, int finalExamGrade, double median, double average)
- **Student** (const `Student` &other)
- **Student** (`Student` &&other) noexcept
- `Student` & **operator=** (const `Student` &other)
- `Student` & **operator=** (`Student` &&other) noexcept
- const vector< int > & **getGrades** () const
- int **getFinalExamGrade** () const
- double **getMedian** () const
- double **getAverage** () const
- double **getFinalMedian** () const
- double **getFinalAverage** () const
- double **getFinalGrade** () const
- void **setGrades** (const vector< int > &newGrades)
- void **setFinalExamGrade** (int examGrade)
- void **setMedian** (double medianValue)
- void **setAverage** (double averageValue)
- void **setFinalMedian** (double finalMedian)
- void **setFinalAverage** (double finalAverage)
- void **setFinalGrade** (double finalGradeValue)

Public Member Functions inherited from [Zmogus](#)

- string **getFirstName** () const
- string **getLastName** () const
- void **setFirstName** (const string &fName)
- void **setLastName** (const string &lName)

Friends

- std::istream & **operator>>** (istream &i, [Student](#) &student)
- std::ostream & **operator<<** (std::ostream &os, const [Student](#) &student)

Additional Inherited Members

Protected Member Functions inherited from [Zmogus](#)

- **Zmogus** (const string &fName, const string &lName)

Protected Attributes inherited from [Zmogus](#)

- string **firstName**
- string **lastName**

4.1.1 Member Function Documentation

4.1.1.1 ppp()

```
void Student::ppp ( ) const [inline], [override], [virtual]
```

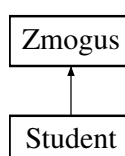
Implements [Zmogus](#).

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

- sources/student.h

4.2 Zmogus Class Reference

Inheritance diagram for Zmogus:



Public Member Functions

- string **getFirstName** () const
- string **getLastName** () const
- void **setFirstName** (const string &fName)
- void **setLastName** (const string &lName)

Protected Member Functions

- **Zmogus** (const string &fName, const string &lName)
- virtual void **ppp** () const =0

Protected Attributes

- string **firstName**
- string **lastName**

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

- sources/student.h

Chapter 5

File Documentation

5.1 failu-generavimas.h

```
00001 #ifndef FAILU_GENERAVIMAS_H
00002 #define FAILU_GENERAVIMAS_H
00003
00004 #include <iostream>
00005 #include <fstream>
00006 #include <iomanip>
00007 #include <vector>
00008 #include <chrono>
00009 #include <thread>
00010 #include <string>
00011 #include <ctime>
00012 #include <cstdlib>
00013 #include "vektoriai.h"
00014 #include "student.h"
00015
00016 using namespace std;
00017
00018 void writeCategorizedStudents(const vector<Student>& students, const string& filename);
00019 void generateFiles();
00020 void sortAndWriteToFile(const string& inputFilename);
00021 void generatingFinal();
00022
00023 #endif
```

5.2 student.h

```
00001 #ifndef STUDENT_H
00002 #define STUDENT_H
00003
00004 #include "vektoriai.h"
00005
00006 #include <vector>
00007 #include <string>
00008 #include <iomanip>
00009 #include <algorithm>
00010
00011 using namespace std;
00012
00013 class Zmogus {
00014 protected:
00015     string firstName;
00016     string lastName;
00017
00018     Zmogus() = default;
00019     Zmogus(const string& fName, const string& lName) : firstName(fName), lastName(lName) {}
00020     virtual void ppp() const = 0;
00021
00022 public:
00023     virtual ~Zmogus() {}
00024     string getFirstName() const { return firstName; }
00025     string getLastName() const { return lastName; }
00026
00027     void setFirstName(const string& fName) { firstName = fName; }
00028     void setLastName(const string& lName) { lastName = lName; }
```

```

00029 };
00030
00031 class Student : public Zmogus {
00032 private:
00033     vector<int> grades;
00034     int finalExamGrade;
00035     double median, average;
00036     double fin_median, fin_average, finalGrade;
00037
00038 public:
00039     void ppp() const override {}
00040     Student() : finalExamGrade(0), median(0.0), average(0.0), fin_median(0.0), fin_average(0.0),
00041     finalGrade(0.0) {}
00042     Student(const string& fName, const string& lName, const vector<int>& grades, int finalExamGrade,
00043     double median, double average)
00044     : Zmogus(fName, lName), grades(grades), finalExamGrade(finalExamGrade), median(median),
00045     average(average), fin_median(0.0), fin_average(0.0), finalGrade(0.0) {}
00046
00047 // Destructor
00048 ~Student() {
00049     grades.clear();
00050     firstName.clear();
00051     lastName.clear();
00052 }
00053
00054 // Copy Constructor
00055 Student(const Student& other)
00056 : Zmogus(other.firstName, other.lastName), grades(other.grades),
00057     finalExamGrade(other.finalExamGrade), median(other.median), average(other.average),
00058     fin_median(other.fin_median), fin_average(other.fin_average), finalGrade(other.finalGrade) {}
00059
00060 // Move Constructor
00061 Student(Student&& other) noexcept
00062 : Zmogus(move(other.firstName), move(other.lastName)),
00063     grades(move(other.grades)),
00064     finalExamGrade(move(other.finalExamGrade)),
00065     median(move(other.median)),
00066     average(move(other.average)),
00067     fin_median(move(other.fin_median)),
00068     fin_average(move(other.fin_average)),
00069     finalGrade(move(other.finalGrade)) {}
00070
00071 other.firstName.clear();
00072 other.lastName.clear();
00073 other.grades.clear();
00074 }
00075
00076 // Copy Assignment Operator
00077 Student& operator=(const Student& other) {
00078     if (this != &other) { // self-assignment check
00079         Zmogus::setFirstName(other.getFirstName());
00080         Zmogus::setLastName(other.getLastName());
00081         grades = other.grades;
00082         finalExamGrade = other.finalExamGrade;
00083         median = other.median;
00084         average = other.average;
00085         fin_median = other.fin_median;
00086         fin_average = other.fin_average;
00087         finalGrade = other.finalGrade;
00088     }
00089     return *this;
00090 }
00091
00092 // Move Assignment Operator
00093 Student& operator=(Student&& other) noexcept {
00094     if (this != &other) {
00095         Zmogus::setFirstName(move(other.getFirstName()));
00096         Zmogus::setLastName(move(other.getLastName()));
00097         grades = move(other.grades);
00098         finalExamGrade = move(other.finalExamGrade);
00099         median = move(other.median);
00100         average = move(other.average);
00101         fin_median = move(other.fin_median);
00102         fin_average = move(other.fin_average);
00103         finalGrade = move(other.finalGrade);
00104         other.firstName.clear();
00105         other.lastName.clear();
00106         other.grades.clear();
00107     }
00108     return *this;
00109 }
00110
00111 // Input Operator
00112 friend std::istream& operator>>(istream& i, Student& student) {
00113     string firstName, lastName;
00114     i >> firstName >> lastName;
00115     student.setFirstName(firstName);

```



```

00111     student.setLastName(lastName);
00112
00113     vector<int> grades;
00114     for (int j = 0; j < 15; ++j) {
00115         int grade;
00116         i » grade;
00117         grades.push_back(grade);
00118     }
00119     student.setGrades(grades);
00120
00121     i » student.finalExamGrade;
00122
00123     // final average
00124     double sum = 0;
00125     for (int grade : grades) {
00126         sum += grade;
00127     }
00128     double average = sum / grades.size();
00129     double finalAverage = average * 0.4 + student.finalExamGrade * 0.6;
00130     student.setFinalAverage(finalAverage);
00131
00132     // final median
00133     sort(grades.begin(), grades.end());
00134     double finalMedian;
00135     if (grades.size() % 2 == 0) {
00136         finalMedian = (grades[grades.size() / 2 - 1] + grades[grades.size() / 2]) / 2.0;
00137     } else {
00138         finalMedian = grades[grades.size() / 2];
00139     }
00140     finalMedian = finalMedian * 0.4 + student.finalExamGrade * 0.6;
00141     student.setFinalMedian(finalMedian);
00142
00143     return i;
00144 }
00145
00146 // Output Operator
00147 friend std::ostream& operator<<(std::ostream& os, const Student& student) {
00148     os << setw(10) << student.getFirstName() << setw(20) << student.getLastName();
00149     double average = student.getAverage() * 0.4 + student.getFinalExamGrade() * 0.6;
00150     double median = student.getMedian() * 0.4 + student.getFinalExamGrade() * 0.6;
00151     os << fixed << setw(25) << setprecision(2) << average;
00152     os << fixed << setw(25) << setprecision(2) << median << '\n';
00153     return os;
00154 }
00155
00156 const vector<int>& getGrades() const { return grades; }
00157 int getFinalExamGrade() const { return finalExamGrade; }
00158 double getMedian() const { return median; }
00159 double getAverage() const { return average; }
00160 double getFinalMedian() const { return fin_median; }
00161 double getFinalAverage() const { return fin_average; }
00162 double getFinalGrade() const { return finalGrade; }
00163
00164
00165 void setGrades(const vector<int>& newGrades) { grades = newGrades; }
00166 void setFinalExamGrade(int examGrade) { finalExamGrade = examGrade; }
00167 void setMedian(double medianValue) { median = medianValue; }
00168 void setAverage(double averageValue) { average = averageValue; }
00169 void setFinalMedian(double finalMedian) { fin_median = finalMedian; }
00170 void setFinalAverage(double finalAverage) { fin_average = finalAverage; }
00171 void setFinalGrade(double finalGradeValue) { finalGrade = finalGradeValue; }
00172 };
00173
00174 #endif

```

5.3 vektoriai.h

```

00001 #ifndef VEKTORIAI_H
00002 #define VEKTORIAI_H
00003
00004 // #include "student.h"
00005
00006 #include <iostream>
00007 #include <fstream>
00008 #include <vector>
00009 #include <string>
00010 #include <algorithm>
00011 #include <iomanip>
00012 #include <ctime>
00013 #include <cstdlib>
00014 #include <sstream>
00015 #include <limits>
00016 #include <numeric>

```

```
00017 #include <chrono>
00018 #include <cassert>
00019
00020
00021 using namespace std;
00022
00023 class Student;
00024 bool isValidName(const string& name);
00025 bool isValidGrade(const string& grade);
00026 double calculateAverage(const Student& student);
00027 double calculateMedian(const Student& student);
00028 void randomGradeGenerator(int number, Student& student);
00029 void generateNames(Student& student);
00030 void readFromFile(const string& filename, vector<Student>& students);
00031 void tests();
00032
00033 #endif
```

Index

ppp

Student, [8](#)

sources/failu-generavimas.h, [11](#)

sources/student.h, [11](#)

sources/vektoriai.h, [13](#)

Student, [7](#)

ppp, [8](#)

Zmogus, [8](#)