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

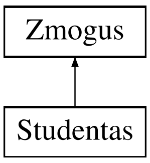
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Studentas Class Reference

Inheritance diagram for Studentas:



Public Member Functions

- **Studentas** (const string &vardas, const string &pavarde, int egzas=0)
- **Studentas** (std::istream &is)
- **Studentas** (int s)
- **Studentas** (const Studentas &s)
- **Studentas** (Studentas &&s) noexcept
- Studentas & **operator=** (const Studentas &s)
- Studentas & **operator=** (Studentas &&s) noexcept
- vector< int > **getNd** () const
- int **getEgzas** () const
- double **calculateFinalGrade** () const
- double **calculateMedian** () const
- istream & **readStudent** (std::istream &is)
- void setVardas (const string &v) override
- void setPavarde (const string &p) override
- void **setEgzas** (int e)
- void **addNd** (int n)

Public Member Functions inherited from Zmogus

- **Zmogus** (const string &vardas, const string &pavarde)
- string **getVardas** () const
- string **getPavarde** () const
- **Zmogus** (const Zmogus &o)
- **Zmogus** (Zmogus &&o) noexcept
- Zmogus & **operator=** (const Zmogus &o)
- Zmogus & **operator=** (Zmogus &&z) noexcept

Friends

- ostream & **operator**<< (ostream &out, const Studentas &a)
- istream & **operator**>> (istream &in, Studentas &a)

Additional Inherited Members

Protected Attributes inherited from Zmogus

- string **vardas**
- string **pavarde**
- int **amzius**

4.1.1 Member Function Documentation

4.1.1.1 setPavarde()

```
void Studentas::setPavarde (
             const string & p)  [inline], [override], [virtual]
```

Implements Zmogus.

4.1.1.2 setVardas()

```
void Studentas::setVardas (
             const string & v)  [inline], [override], [virtual]
```

Implements Zmogus.

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

- OneDrive/Desktop/klase/studentas.h
- OneDrive/Desktop/klase/studentas.cpp

4.2 studentas Struct Reference

Public Attributes

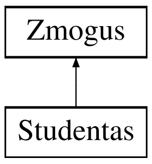
- std::string **vardas**
- std::string **pavarde**
- std::vector< int > **nd**
- int **egzas**
- string **vardas**
- string **pavarde**
- int * **nd**
- vector< int > **nd**

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

- OneDrive/Desktop/klase/funkcijosVector.h
- OneDrive/Desktop/klase/main.cpp
- OneDrive/Desktop/klase/test.cpp

4.3 Zmogus Class Reference

Inheritance diagram for Zmogus:



Public Member Functions

- **Zmogus** (const string &vardas, const string &pavarde)
- string **getVardas** () const
- string **getPavarde** () const
- virtual void **setVardas** (const string &v)=0
- virtual void **setPavarde** (const string &p)=0
- **Zmogus** (const Zmogus &o)
- **Zmogus** (Zmogus &&o) noexcept
- Zmogus & **operator=** (const Zmogus &o)
- Zmogus & **operator=** (Zmogus &&z) noexcept

Protected Attributes

- string **vardas**
- string **pavarde**
- int **amzius**

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

- OneDrive/Desktop/klase/studentas.h
- OneDrive/Desktop/klase/studentas.cpp

Chapter 5

File Documentation

5.1 funkcijosVector.h

```
#ifndef FUNKCIJOSVECTOR_H
#define FUNKCIJOSVECTOR_H

#include <string>
#include <vector>
#include <functional>
using namespace std;
struct studentas {
    std::string vardas;
    std::string pavarde;
    std::vector<int> nd;
    int egzas;
};
void studrus2(vector<studentas>& students, vector<studentas>& vargsai, vector<studentas>& galva, const
    string& filename, int dydis);
double calculateFinalGrade(const studentas& s);
double calculateMedian(const studentas& s);
void clearFiles();
int partition(vector<studentas>& students, vector<double>& galrez, vector<double>& median, int low,
    int high, int rusis);
void quickSort(vector<studentas>& students, vector<double>& galrez, vector<double>& median, int low,
    int high, int rusis);
void studrus(vector<studentas>& students, vector<studentas>& vargsai, vector<studentas>& galva, const
    string& filename, int dydis);
void studrus1(vector<studentas>& students, vector<studentas>& vargsai, const string& filename, int
    dydis);
void createfile(const std::string& filename, const int& kiekis);
void skaitymas(std::vector<studentas>& students, std::vector<double>& galrez, std::vector<double>&
    median);
void rasytiranka(std::vector<studentas>& students, std::vector<double>& galrez, std::vector<double>&
    median);
int readInt(const std::string& prompt);
bool isNumeric(const std::string& str);
bool compareNames(const std::string& a, const std::string& b);
void skaityti(std::vector<studentas>& students, std::vector<double>& galrez, std::vector<double>&
    median);
void spausdint(const std::vector<studentas>& students, const std::vector<double>& galrez, const
    std::vector<double>& median);
void spausdintfaila(const std::vector<studentas>& students, const std::vector<double>& galrez, const
    std::vector<double>& median);
std::string randname();

#endif
```

5.2 studentas.h

```
#ifndef STUDENTAS_H
#define STUDENTAS_H

#include <iostream>
#include <string>
#include <vector>
#include <numeric>
#include <algorithm>
```

```cpp
#include <iomanip>

using namespace std;

class Zmogus
{
protected:
    string vardas;
    string pavarde;
    int amzius;
public:
    Zmogus() = default;
    Zmogus(const string& vardas, const string& pavarde) : vardas(vardas), pavarde(pavarde) {}
    virtual ~Zmogus() = default;

    // Getteriai
    inline string getVardas() const { return vardas; }
    inline string getPavarde() const { return pavarde; }

    // Setteriai
    virtual void setVardas(const string& v) = 0;
    virtual void setPavarde(const string& p) = 0;

    // Copy constructor
    Zmogus(const Zmogus& o) : vardas(o.vardas), pavarde(o.pavarde) {}

    // Move constructor
    Zmogus(Zmogus&& o) noexcept : vardas(move(o.vardas)), pavarde(move(o.pavarde)) {}

    // Copy assignment operator
    Zmogus& operator=(const Zmogus& o);

    // Move assignment operator
    Zmogus& operator=(Zmogus&& z) noexcept;
};

class Studentas: public Zmogus {
private:
    vector<int> nd;
    int egzas;

public:
    // Constructors
    Studentas() : Zmogus(), egzas(0) {}
    Studentas(const string& vardas, const string& pavarde, int egzas = 0)
        : Zmogus(vardas, pavarde), egzas(egzas) {}
    Studentas(std::istream& is) { readStudent(is); }
    Studentas(int s) : Zmogus(), egzas(s) {}

    // Copy constructor
    Studentas(const Studentas& s) : Zmogus(s.vardas, s.pavarde), nd(s.nd), egzas(s.egzas) {}

    // Move constructor
    Studentas(Studentas&& s) noexcept : Zmogus(move(s)), nd(move(s.nd)), egzas(s.egzas) {
        s.egzas = 0;
    }

    //Naudojamas patikrinimui
    //~Studentas() { cout « "Sunaikintas " « vardas « " " « pavarde « " kurio egzamino rezultatas " « egzas « endl;}
    ~Studentas() {}

    // Copy assignment operator
    Studentas& operator=(const Studentas& s);
    // Move assignment operator
    Studentas& operator=(Studentas&& s) noexcept;

    // Getters
    inline vector<int> getNd() const { return nd; }
    inline int getEgzas() const { return egzas; }

    //Calculate funkcijos
    double calculateFinalGrade() const;
    double calculateMedian() const;

    istream& readStudent(std::istream& is);

    // Setters
    void setVardas(const string& v) override { vardas = v; }
    void setPavarde(const string& p) override { pavarde = p; }
    void setEgzas(int e) { egzas = e; }
    void addNd(int n) { nd.push_back(n); }

    //Out
    friend ostream& operator«(ostream& out, const Studentas &a) {
        const int ilgis = 20;
        out « setw(ilgis) « left « a.getPavarde() « " " « setw(ilgis) « left « a.getVardas() « "
";
        out « setw(ilgis) « left « a.calculateFinalGrade() « "      ";
        out « setw(ilgis) « left « a.calculateMedian() « endl;
        return out;
    }

    //In
    friend istream& operator»(istream& in, Studentas &a) {
        a.nd.clear();
        a.nd.reserve(15);
        in » a.vardas » a.pavarde;
        for (int k = 0; k < 15; k++) {
            int nd_val;
            in » nd_val;
            a.nd.push_back(nd_val);
        }
        in » a.egzas;
        return in;
    }
};

void printStudentState(const Studentas& student, const string& name);
void studrus2(std::vector<Studentas>& students, std::vector<Studentas>& vargsai,
std::vector<Studentas>& galva, const std::string& filename, int dydis);
void clearFiles();
int partition(std::vector<Studentas>& students, int low, int high, int rusis);
void quickSort(std::vector<Studentas>& students, int low, int high, int rusis);
void studrus(std::vector<Studentas>& students, std::vector<Studentas>& vargsai,
std::vector<Studentas>& galva, const std::string& filename, int dydis);
void studrus1(std::vector<Studentas>& students, std::vector<Studentas>& vargsai, const std::string&
filename, int dydis);
void createfile(const std::string& filename, const int& kiekis);
void skaitymas(std::vector<Studentas>& students);
void rasytiranka(std::vector<Studentas>& students);
void skaityti(std::vector<Studentas>& students);
void spausdint(const std::vector<Studentas>& students);
void spausdintfaila(const std::vector<Studentas>& students);
std::string randname();

#endif
```

Index