vektoriai

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

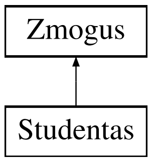
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Studentas Class Reference

Inheritance diagram for Studentas:



Public Member Functions

- **Studentas** (std::string vardas1, std::string pavarde1, std::vector< double > nd1, double eg1, double galutinis1, double median1)
- **Studentas** (const Studentas &other)
- Studentas & **operator=** (const Studentas &other)
- **Studentas** (Studentas &&other) noexcept
- Studentas & **operator=** (Studentas &&other) noexcept
- void **setNd** (std::vector< double > nd1)
- std::vector< double > **getNd** () const
- void **addNd** (double nd1)
- void **setEg** (double eg1)
- double **getEg** () const
- void **setNdvid** (double ndvid1)
- double **getNdvid** () const
- void **setGalutinis** (double galutinis1)
- double **getGalutinis** () const
- void **setMediana** (double mediana1)
- double **getMediana** () const
- void **setEgzFromNd** ()
- void **calculateNdVid** ()
- void print () const override

Public Member Functions inherited from Zmogus

- **Zmogus** (std::string vardas1, std::string pavarde1)
- void **setVardas** (std::string vardas1)
- std::string **getVardas** () const
- void **setPavarde** (std::string pavarde1)
- std::string **getPavarde** () const

Friends

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

Additional Inherited Members

Protected Attributes inherited from Zmogus

- std::string **vardas**
- std::string **pavarde**

4.1.1 Member Function Documentation

4.1.1.1 print()

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

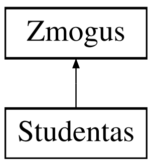
Implements Zmogus.

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

- Desktop/2024_antras_pusmetis/Objektinis_programavimas/1uzduotis/vektoriai/Struktura.h

4.2 Zmogus Class Reference

Inheritance diagram for Zmogus:



Public Member Functions

- **Zmogus** (std::string vardas1, std::string pavarde1)
- void **setVardas** (std::string vardas1)
- std::string **getVardas** () const
- void **setPavarde** (std::string pavarde1)
- std::string **getPavarde** () const
- virtual void **print** () const =0

Protected Attributes

- std::string **vardas**
- std::string **pavarde**

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

- Desktop/2024_antras_pusmetis/Objektinis_programavimas/1uzduotis/vektoriai/Struktura.h

Chapter 5

File Documentation

5.1 Struktura.h

```
#ifndef STRUKTURA_H_INCLUDED
#define STRUKTURA_H_INCLUDED

#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <iomanip>
#include <algorithm>
#include <chrono>
#include <random>
#include <sstream>

// Abstract base class for a person
class Zmogus {
protected:
    std::string vardas;
    std::string pavarde;

public:
    Zmogus() = default;
    Zmogus(std::string vardas1, std::string pavarde1) : vardas(vardas1), pavarde(pavarde1) {}

    virtual ~Zmogus() = default;

    void setVardas(std::string vardas1) {
        vardas = vardas1;
    }
    std::string getVardas() const {
        return vardas;
    }

    void setPavarde(std::string pavarde1) {
        pavarde = pavarde1;
    }
    std::string getPavarde() const {
        return pavarde;
    }

    // Pure virtual functions making this class abstract
    virtual void print() const = 0;
};

// Derived class for a student
class Studentas : public Zmogus {
private:
    std::vector<double> nd;
    double eg;
    double ndvid;
    double galutinis;
    double mediana;

public:
    // Constructors
    Studentas() = default;
    Studentas(std::string vardas1, std::string pavarde1, std::vector<double> nd1, double eg1, double
galutinis1, double median1)
```

```cpp
        : Zmogus(vardas1, pavarde1), nd(nd1), eg(eg1), ndvid(0), galutinis(galutinis1), mediana(median1) {}

    // Destructor
    ~Studentas() = default;

    // Copy Constructor
    Studentas(const Studentas& other)
        : Zmogus(other.vardas, other.pavarde), nd(other.nd), eg(other.eg), ndvid(other.ndvid), galutinis(other.galutinis), mediana(other.mediana) {}

    // Copy Assignment Operator
    Studentas& operator=(const Studentas& other) {
        if (this != &other) {
            Zmogus::operator=(other);
            nd = other.nd;
            eg = other.eg;
            ndvid = other.ndvid;
            galutinis = other.galutinis;
            mediana = other.mediana;
        }
        return *this;
    }

    // Move Constructor
    Studentas(Studentas&& other) noexcept
        : Zmogus(std::move(other)), nd(std::move(other.nd)), eg(other.eg), ndvid(other.ndvid), galutinis(other.galutinis), mediana(other.mediana) {}

    // Move Assignment Operator
    Studentas& operator=(Studentas&& other) noexcept {
        if (this != &other) {
            Zmogus::operator=(std::move(other));
            nd = std::move(other.nd);
            eg = other.eg;
            ndvid = other.ndvid;
            galutinis = other.galutinis;
            mediana = other.mediana;
        }
        return *this;
    }

    void setNd(std::vector<double> nd1) {
        nd = nd1;
    }
    std::vector<double> getNd() const {
        return nd;
    }

    void addNd(double nd1) {
        nd.push_back(nd1);
    }

    void setEg(double eg1) {
        eg = eg1;
    }
    double getEg() const {
        return eg;
    }

    void setNdvid(double ndvid1) {
        ndvid = ndvid1;
    }
    double getNdvid() const {
        return ndvid;
    }

    void setGalutinis(double galutinis1) {
        galutinis = galutinis1;
    }
    double getGalutinis() const {
        return galutinis;
    }

    void setMediana(double mediana1) {
        mediana = mediana1;
    }
    double getMediana() const {
        return mediana;
    }

    void setEgzFromNd() {
        if (!nd.empty()) {
            eg = nd.back();
            nd.pop_back();
        }
    }

    void calculateNdVid() {
        ndvid = 0;
        for (size_t j = 0; j < nd.size(); j++) {
            ndvid += nd[j];
        }
        ndvid /= nd.size();
    }

    // Input and Output Operators
    friend std::ostream& operator«(std::ostream& os, const Studentas& studentas) {
        os « studentas.vardas « ' ' « studentas.pavarde « ' ';
        for (auto& grade : studentas.nd) {
            os « grade « ' ';
        }
        os « studentas.eg;
        return os;
    }

    friend std::istream& operator»(std::istream& is, Studentas& studentas) {
        is » studentas.vardas » studentas.pavarde;
        double grade;
        while (is » grade) {
            studentas.nd.push_back(grade);
        }
        return is;
    }

    void print() const override {
        std::cout « vardas « " " « pavarde « std::endl;
    }
};

// Function declarations
void testInputOutput();
void testConstructor();
void testCopyConstructor();
void testMoveConstructor();
void testCopyAssignment();
void testMoveAssignment();
double median(std::vector<double>& arr);
void generuotiBalus(Studentas& studentas);
void skaitytiIsFailo(std::vector<Studentas>& A, std::string& failoPavadinimas);
void rikiuotiPagalVarda(std::vector<Studentas>& A);
void rikiuotiPagalPavarde(std::vector<Studentas>& A);
void rikiuotiPagalGalutiniVidurki(std::vector<Studentas>& A);
void rikiuotiPagalMediana(std::vector<Studentas>& A);
void atspausdintiDuomenis(std::vector<Studentas>& A, bool iFaila = false);
void generuotiFailaSuStudentais(int irasuSkaicius);
void rikiuotiStudentus(std::vector<Studentas>& geri_studentai, std::vector<Studentas>&
blogi_studentai);

#endif // STRUKTURA_H_INCLUDED
```

Index