

vektoriai

Generated by Doxygen 1.11.0

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 Studentas Class Reference	7
4.1.1 Member Function Documentation	8
4.1.1.1 print()	8
4.2 Zmogus Class Reference	8
5 File Documentation	11
5.1 Struktura.h	11
Index	15

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Zmogus	8
Studentas	7

Chapter 2

Class Index

2.1 Class List

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

Studentas	7
Zmogus	8

Chapter 3

File Index

3.1 File List

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

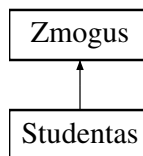
Desktop/2024_antras_pusmetis/Objektinis_programavimas/1uzduotis/vektoriai/[Struktura.h](#) 11

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]
```

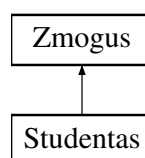
Implements [Zmogus](#).

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

```
00001 #ifndef STRUKTURA_H_INCLUDED
00002 #define STRUKTURA_H_INCLUDED
00003
00004 #include <iostream>
00005 #include <string>
00006 #include <vector>
00007 #include <fstream>
00008 #include <iomanip>
00009 #include <algorithm>
00010 #include <chrono>
00011 #include <random>
00012 #include <sstream>
00013
00014 // Abstract base class for a person
00015 class Zmogus {
00016 protected:
00017     std::string vardas;
00018     std::string pavarde;
00019
00020 public:
00021     Zmogus() = default;
00022     Zmogus(std::string vardas1, std::string pavarde1) : vardas(vardas1), pavarde(pavarde1) {}
00023
00024     virtual ~Zmogus() = default;
00025
00026     void setVardas(std::string vardas1) {
00027         vardas = vardas1;
00028     }
00029     std::string getVardas() const {
00030         return vardas;
00031     }
00032
00033     void setPavarde(std::string pavarde1) {
00034         pavarde = pavarde1;
00035     }
00036     std::string getPavarde() const {
00037         return pavarde;
00038     }
00039
00040     // Pure virtual functions making this class abstract
00041     virtual void print() const = 0;
00042 };
00043
00044 // Derived class for a student
00045 class Studentas : public Zmogus {
00046 private:
00047     std::vector<double> nd;
00048     double eg;
00049     double ndvid;
00050     double galutinis;
00051     double mediana;
00052
00053 public:
00054     // Constructors
00055     Studentas() = default;
00056     Studentas(std::string vardas1, std::string pavarde1, std::vector<double> nd1, double eg1, double
        galutinis1, double median1)
```

```

00057         : Zmogus(vardas1, pavardel1, nd(nd1), eg(eg1), ndvid(0), galutinis(galutinis1),
medianana(median1) {}
00058
00059     // Destructor
00060     ~Studentas() = default;
00061
00062     // Copy Constructor
00063     Studentas(const Studentas& other)
00064         : Zmogus(other.vardas, other.pavarde, nd(other.nd), eg(other.eg), ndvid(other.ndvid),
galutinis(other.galutinis), medianana(other.medianana) {}
00065
00066     // Copy Assignment Operator
00067     Studentas& operator=(const Studentas& other) {
00068         if (this != &other) {
00069             Zmogus::operator=(other);
00070             nd = other.nd;
00071             eg = other.eg;
00072             ndvid = other.ndvid;
00073             galutinis = other.galutinis;
00074             medianana = other.medianana;
00075         }
00076         return *this;
00077     }
00078
00079     // Move Constructor
00080     Studentas(Studentas&& other) noexcept
00081         : Zmogus(std::move(other)), nd(std::move(other.nd)), eg(other.eg), ndvid(other.ndvid),
galutinis(other.galutinis), medianana(other.medianana) {}
00082
00083     // Move Assignment Operator
00084     Studentas& operator=(Studentas&& other) noexcept {
00085         if (this != &other) {
00086             Zmogus::operator=(std::move(other));
00087             nd = std::move(other.nd);
00088             eg = other.eg;
00089             ndvid = other.ndvid;
00090             galutinis = other.galutinis;
00091             medianana = other.medianana;
00092         }
00093         return *this;
00094     }
00095
00096     void setNd(std::vector<double> nd1) {
00097         nd = nd1;
00098     }
00099     std::vector<double> getNd() const {
00100         return nd;
00101     }
00102
00103     void addNd(double nd1) {
00104         nd.push_back(nd1);
00105     }
00106
00107     void setEg(double eg1) {
00108         eg = eg1;
00109     }
00110     double getEg() const {
00111         return eg;
00112     }
00113
00114     void setNdvid(double ndvid1) {
00115         ndvid = ndvid1;
00116     }
00117     double getNdvid() const {
00118         return ndvid;
00119     }
00120
00121     void setGalutinis(double galutinis1) {
00122         galutinis = galutinis1;
00123     }
00124     double getGalutinis() const {
00125         return galutinis;
00126     }
00127
00128     void setMedianana(double medianal1) {
00129         medianana = medianal1;
00130     }
00131     double getMedianana() const {
00132         return medianana;
00133     }
00134
00135     void setEgzFromNd() {
00136         if (!nd.empty()) {
00137             eg = nd.back();
00138             nd.pop_back();
00139         }
00140     }

```



```

00141
00142     void calculateNdVid() {
00143         ndvid = 0;
00144         for (size_t j = 0; j < nd.size(); j++) {
00145             ndvid += nd[j];
00146         }
00147         ndvid /= nd.size();
00148     }
00149
00150     // Input and Output Operators
00151     friend std::ostream& operator<<(std::ostream& os, const Studentas& studentas) {
00152         os << studentas.vardas << ' ' << studentas.pavarde << ' ';
00153         for (auto& grade : studentas.nd) {
00154             os << grade << ' ';
00155         }
00156         os << studentas.eg;
00157         return os;
00158     }
00159
00160     friend std::istream& operator>>(std::istream& is, Studentas& studentas) {
00161         is >> studentas.vardas >> studentas.pavarde;
00162         double grade;
00163         while (is >> grade) {
00164             studentas.nd.push_back(grade);
00165         }
00166         return is;
00167     }
00168
00169     void print() const override {
00170         std::cout << vardas << " " << pavarde << std::endl;
00171     }
00172 };
00173
00174 // Function declarations
00175 void testInputOutput();
00176 void testConstructor();
00177 void testCopyConstructor();
00178 void testMoveConstructor();
00179 void testCopyAssignment();
00180 void testMoveAssignment();
00181 double median(std::vector<double>& arr);
00182 void generuotiBalus(Studentas& studentas);
00183 void skaitytiIsFailo(std::vector<Studentas>& A, std::string& failoPavadinimas);
00184 void rikiuotiPagalVarda(std::vector<Studentas>& A);
00185 void rikiuotiPagalPavarde(std::vector<Studentas>& A);
00186 void rikiuotiPagalGalutiniVidurki(std::vector<Studentas>& A);
00187 void rikiuotiPagalMediana(std::vector<Studentas>& A);
00188 void atspausdintiDuomenis(std::vector<Studentas>& A, bool iFaila = false);
00189 void generuotiFailaSuStudentais(int irasuSkaicius);
00190 void rikiuotiStudentus(std::vector<Studentas>& geri_studentai, std::vector<Studentas>&
    blogi_studentai);
00191
00192 #endif // STRUKTURA_H_INCLUDED

```


Index

Desktop/2024_antras_pusmetis/Objektinis_programavimas/1_uzduotis/vektoriai/Struktura.h,
[11](#)

print
Studentas, [8](#)

Studentas, [7](#)
print, [8](#)

Zmogus, [8](#)