

v2.0

Generated by Doxygen 1.11.0



---

<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy . . . . .	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List . . . . .	3
<b>3 File Index</b>	<b>5</b>
3.1 File List . . . . .	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 Studentas Class Reference . . . . .	7
4.1.1 Member Function Documentation . . . . .	8
4.1.1.1 setPavarde() . . . . .	8
4.1.1.2 setVardas() . . . . .	8
4.2 studentas Struct Reference . . . . .	8
4.3 Zmogus Class Reference . . . . .	9
<b>5 File Documentation</b>	<b>11</b>
5.1 funkcijosVector.h . . . . .	11
5.2 studentas.h . . . . .	11
<b>Index</b>	<b>15</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

studentas . . . . .	8
Zmogus . . . . .	9
Studentas . . . . .	7



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">Studentas</a>	.....	<a href="#">7</a>
<a href="#">studentas</a>	.....	<a href="#">8</a>
<a href="#">Zmogus</a>	.....	<a href="#">9</a>





## Chapter 3

# File Index

### 3.1 File List

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

OneDrive/Desktop/klase/ <a href="#">funkcijosVector.h</a> . . . . .	11
OneDrive/Desktop/klase/ <a href="#">studentas.h</a> . . . . .	11

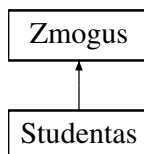


## Chapter 4

# Class Documentation

### 4.1 Studentas Class Reference

Inheritance diagram for Studentas:



#### Public Member Functions

- **Studentas** (const string &vardas, const string &pavarde, int egzاس=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

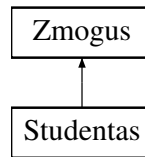
- std::string **vardas**
- std::string **pavarde**
- std::vector< int > **nd**
- int **egzas**
- string **vardas**
- string **pavarde**
- int \* **nd**
- vector< int > **nd**

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

```
00001 #ifndef FUNKCIJOSVECTOR_H
00002 #define FUNKCIJOSVECTOR_H
00003
00004 #include <string>
00005 #include <vector>
00006 #include <functional>
00007 using namespace std;
00008 struct studentas {
00009     std::string vardas;
00010     std::string pavarde;
00011     std::vector<int> nd;
00012     int egzas;
00013 };
00014 void studrus2(vector<studentas>& students, vector<studentas>& vargsai, vector<studentas>& galva, const
    string& filename, int dydis);
00015 double calculateFinalGrade(const studentas& s);
00016 double calculateMedian(const studentas& s);
00017 void clearFiles();
00018 int partition(vector<studentas>& students, vector<double>& galrez, vector<double>& median, int low,
    int high, int rusis);
00019 void quickSort(vector<studentas>& students, vector<double>& galrez, vector<double>& median, int low,
    int high, int rusis);
00020 void studrus(vector<studentas>& students, vector<studentas>& vargsai, vector<studentas>& galva, const
    string& filename, int dydis);
00021 void studrus1(vector<studentas>& students, vector<studentas>& vargsai, const string& filename, int
    dydis);
00022 void createfile(const std::string& filename, const int& kiekis);
00023 void skaitymas(std::vector<studentas>& students, std::vector<double>& galrez, std::vector<double>&
    median);
00024 void rasytiranka(std::vector<studentas>& students, std::vector<double>& galrez, std::vector<double>&
    median);
00025 int readInt(const std::string& prompt);
00026 bool isNumeric(const std::string& str);
00027 bool compareNames(const std::string& a, const std::string& b);
00028 void skaityti(std::vector<studentas>& students, std::vector<double>& galrez, std::vector<double>&
    median);
00029 void spausdint(const std::vector<studentas>& students, const std::vector<double>& galrez, const
    std::vector<double>& median);
00030 void spausdintfaila(const std::vector<studentas>& students, const std::vector<double>& galrez, const
    std::vector<double>& median);
00031 std::string randname();
00032
00033 #endif
```

### 5.2 studentas.h

```
00001 #ifndef STUDENTAS_H
00002 #define STUDENTAS_H
00003
00004 #include <iostream>
00005 #include <string>
00006 #include <vector>
00007 #include <numeric>
00008 #include <algorithm>
```

```

00009 #include <iomanip>
00010
00011 using namespace std;
00012
00013 class Zmogus
00014 {
00015 protected:
00016     string vardas;
00017     string pavarde;
00018     int amzius;
00019 public:
00020     Zmogus() = default;
00021     Zmogus(const string& vardas, const string& pavarde) : vardas(vardas), pavarde(pavarde) {}
00022     virtual ~Zmogus() = default;
00023
00024     // Getteriai
00025     inline string getVardas() const { return vardas; }
00026     inline string getPavarde() const { return pavarde; }
00027
00028     // Setteriai
00029     virtual void setVardas(const string& v) = 0;
00030     virtual void setPavarde(const string& p) = 0;
00031
00032     // Copy constructor
00033     Zmogus(const Zmogus& o) : vardas(o.vardas), pavarde(o.pavarde) {}
00034
00035     // Move constructor
00036     Zmogus(Zmogus&& o) noexcept : vardas(move(o.vardas)), pavarde(move(o.pavarde)) {}
00037
00038     // Copy assignment operator
00039     Zmogus& operator=(const Zmogus& o);
00040
00041     // Move assignment operator
00042     Zmogus& operator=(Zmogus&& z) noexcept;
00043 };
00044
00045 class Studentas: public Zmogus {
00046 private:
00047     vector<int> nd;
00048     int egzas;
00049 public:
00050     // Constructors
00051     Studentas() : Zmogus(), egzas(0) {}
00052     Studentas(const string& vardas, const string& pavarde, int egzas = 0)
00053         : Zmogus(vardas, pavarde), egzas(egzas) {}
00054     Studentas(std::istream& is) { readStudent(is); }
00055     Studentas(int s) : Zmogus(), egzas(s) {}
00056
00057     // Copy constructor
00058     Studentas(const Studentas& s) : Zmogus(s.vardas, s.pavarde), nd(s.nd), egzas(s.egzas) {}
00059
00060     // Move constructor
00061     Studentas(Studentas&& s) noexcept : Zmogus(move(s)), nd(move(s.nd)), egzas(s.egzas) {
00062         s.egzas = 0;
00063     }
00064
00065     //Naudojamas patikrinimui
00066     //~Studentas() { cout << "Sunaikintas " << vardas << " " << pavarde << " kurio egzamino rezultatas " <<
00067     egzas << endl;}
00068     ~Studentas() {}
00069
00070     // Copy assignment operator
00071     Studentas& operator=(const Studentas& s);
00072     // Move assignment operator
00073     Studentas& operator=(Studentas&& s) noexcept;
00074
00075     // Getters
00076     inline vector<int> getNd() const { return nd; }
00077     inline int getEgzas() const { return egzas; }
00078
00079     //Calculate funkcijos
00080     double calculateFinalGrade() const;
00081     double calculateMedian() const;
00082
00083     istream& readStudent(std::istream& is);
00084
00085     // Setters
00086     void setVardas(const string& v) override { vardas = v; }
00087     void setPavarde(const string& p) override { pavarde = p; }
00088     void setEgzas(int e) { egzas = e; }
00089     void addNd(int n) { nd.push_back(n); }
00090
00091     //Out
00092     friend ostream& operator<<(ostream& out, const Studentas &a) {
00093         const int ilgis = 20;
00094         out << setw(ilgis) << left << a.getPavarde() << " " << setw(ilgis) << left << a.getVardas() << "

```



```

";
00095         out << setw(ilgis) << left << a.calculateFinalGrade() << "          ";
00096         out << setw(ilgis) << left << a.calculateMedian() << endl;
00097         return out;
00098     }
00099
00100     //In
00101     friend ostream& operator<<(ostream& in, Studentas &a) {
00102         a.nd.clear();
00103         a.nd.reserve(15);
00104         in >> a.vardas >> a.pavarde;
00105         for (int k = 0; k < 15; k++) {
00106             int nd_val;
00107             in >> nd_val;
00108             a.nd.push_back(nd_val);
00109         }
00110         in >> a.egzas;
00111         return in;
00112     }
00113 };
00114
00115 void printStudentState(const Studentas& student, const string& name);
00116 void studrus2(std::vector<Studentas>& students, std::vector<Studentas>& vargsai,
00117               std::vector<Studentas>& galva, const std::string& filename, int dydis);
00118 void clearFiles();
00119 int partition(std::vector<Studentas>& students, int low, int high, int rusis);
00120 void quickSort(std::vector<Studentas>& students, int low, int high, int rusis);
00121 void studrus(std::vector<Studentas>& students, std::vector<Studentas>& vargsai,
00122               std::vector<Studentas>& galva, const std::string& filename, int dydis);
00123 void studrus1(std::vector<Studentas>& students, std::vector<Studentas>& vargsai, const std::string&
00124               filename, int dydis);
00125 void createfile(const std::string& filename, const int& kiekis);
00126 void skaitymas(std::vector<Studentas>& students);
00127 void rasytiranka(std::vector<Studentas>& students);
00128 void skaityti(std::vector<Studentas>& students);
00129 void spausdint(const std::vector<Studentas>& students);
00130 void spausdintfaila(const std::vector<Studentas>& students);
00131 std::string randname();
00132 #endif

```



# Index

OneDrive/Desktop/klase/funkcijosVector.h, [11](#)

OneDrive/Desktop/klase/studentas.h, [11](#)

setPavarde

Studentas, [8](#)

setVardas

Studentas, [8](#)

Studentas, [7](#)

setPavarde, [8](#)

setVardas, [8](#)

studentas, [8](#)

Zmogus, [9](#)