2.0

Generated by Doxygen 1.12.0

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

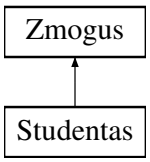
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Studentas Class Reference

Inheritance diagram for Studentas:



Public Member Functions

- **Studentas** (const string &vardas, const string &pavarde, const vector< int > &nd, int egz, const string &reikalavimas)
- const string & **getVardas** () const
- const string & **getPavarde** () const
- const vector< int > & **getNd** () const
- double **getGalutinis** () const
- int **getEgz** () const
- string **getReikalavimas** () const
- **Studentas** (const Studentas &other)
- Studentas & **operator=** (const Studentas &other)
- void **calculateGalutinis** ()
- void **setReikalavimas** (const string &reikalavimas_verte)
- void **setGalutinis** (double newGalutinis)
- void **setVardas** (const string &vardas)
- void **setPavarde** (const string &pavarde)
- void **setNd** (const vector< int > &n)
- void **setNd** (const list< int > &n)
- void **addNd** (int nd_verte)
- void **setEgz** (int egz_verte)
- void spausdintiInformacija () const override

Public Member Functions inherited from Zmogus

- **Zmogus** (const string &vardas, const string &pavarde)
- string **getVardas** () const
- string **getPavarde** () const
- void **setVardas** (const string &v)
- void **setPavarde** (const string &p)

Static Public Member Functions

- static string **generuoti_varda** (int indeksas)
- static string **generuoti_pavarde** (int indeksas)
- static Studentas **generuotiStudenta** ()
- static vector< Studentas > **nuskaitytiIsFailo** (const string &failoVardas)
- static void **rasytiIFaila** (const Studentas &studentas, const string &failoVardas)

Friends

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

Additional Inherited Members

Protected Attributes inherited from Zmogus

- string **vardas**
- string **pavarde**

4.1.1 Member Function Documentation

4.1.1.1 spausdintiInformacija()

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

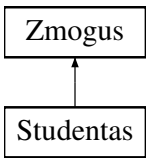
Implements Zmogus.

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

- Documents/2.0/Studentas.h

4.2 Zmogus Class Reference

Inheritance diagram for Zmogus:



Public Member Functions

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

Protected Attributes

- string **vardas**
- string **pavarde**

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

- Documents/2.0/Studentas.h

Chapter 5

File Documentation

5.1 Failai.h

```
00001 #ifndef FAILAI_H_INCLUDED
00002 #define FAILAI_H_INCLUDED
00003 #include "Mylib.h"
00004 #include "Studentas.h"
00005
00006 void kurti_faila(const std::vector<Studentas>& studentai, const std::string& failo_priedas);
00007 void generuoti_sarasus(int n, vector<Studentas>& studentai);
00008 void rusiavimas_2_grupes(const vector<Studentas>& studentai, vector<Studentas>& vargsiukai,
      vector<Studentas>& kietiakiai, int strategija);
00009 void darbas_su_failais(string failas, int duom_sk, string reikalavimas, string kriterijus, int
      strategija);
00010 void surusioti_failai(vector<Studentas>& studentai, string name);
00011 void sort_students_by_surname(vector<Studentas>& studentai);
00012 bool compare_by_surname(const Studentas& a, const Studentas& b);
00013 void sort_students_by_name(vector<Studentas>& studentai);
00014 bool compare_by_name(const Studentas& a, const Studentas& b);
00015 void sort_students_by_grade(vector<Studentas>& studentai);
00016 bool compare_by_grade(const Studentas& a, const Studentas& b);
00017 //void print_memory_usage(const vector<Studentas>& v, const string& name);
00018
00019 void rusiavimas_2_grupes_list(const list<Studentas>& studentai_list, list<Studentas>& vargsiukai_list,
      list<Studentas>& kietiakiai_list, int strategija);
00020 void darbas_su_failais_list(string failas, int duom_sk, string reikalavimas, string kriterijus, int
      strategija);
00021 void surusioti_failai_list(list<Studentas>& studentai_list, const string& name);
00022 void sort_students_by_name_list(list<Studentas>& studentai_list);
00023 void sort_students_by_surname_list(list<Studentas>& studentai_list);
00024 void sort_students_by_grade_list(list<Studentas>& studentai_list);
00025 //void print_memory_usage_list(const vector<Studentas>& v, const string& name);
00026
00027 #endif // FAILAI_H_INCLUDED
```

5.2 Mylib.h

```
00001 #ifndef MYLIB_H_INCLUDED
00002 #define MYLIB_H_INCLUDED
00003
00004 #include <cstdio>
00005 #include <memory>
00006 #include <stdexcept>
00007 #include <array>
00008 #include <windows.h>
00009
00010
00011
00012 #include <iostream>
00013 #include <fstream>
00014 #include <sstream>
00015 #include <vector>
00016 #include <list>
00017 #include <string>
00018 #include <algorithm>
00019 #include <numeric>
00020 #include <iomanip>
```

```
00021 #include <limits>
00022 #include <random>
00023 #include <chrono>
00024 using namespace std;
00025
00026
00027 #endif // MYLIB_H_INCLUDED
```

5.3 Studentas.h

```
00001 #ifndef STUDENTAS_H_INCLUDED
00002 #define STUDENTAS_H_INCLUDED
00003 #include "Mylib.h"
00004
00005 class Zmogus {
00006 protected:
00007     string vardas;
00008     string pavarde;
00009
00010 public:
00011     Zmogus() : vardas(" "), pavarde(" ") {}
00012
00013     Zmogus(const string& vardas, const string& pavarde)
00014         : vardas(vardas), pavarde(pavarde) {}
00015
00016     virtual ~Zmogus() {}
00017
00018     /*Zmogus& operator=(const Zmogus& other) {
00019         if (this != &other) {
00020             vardas = other.vardas;
00021             pavarde = other.pavarde;
00022         }
00023         return *this;
00024     }
00025
00026     Zmogus(const Zmogus& other)
00027         : vardas(other.vardas), pavarde(other.pavarde) {}
00028
00029 */
00030     virtual void spausdintiInformacija() const = 0;
00031    // virtual string gautiTipa() const = 0;
00032
00033     string getVardas() const { return vardas; }
00034     string getPavarde() const { return pavarde; }
00035
00036
00037     void setVardas(const string& v) { vardas = v; }
00038     void setPavarde(const string& p) { pavarde = p; }
00039 };
00040
00041
00042 class Studentas : public Zmogus {
00043 private:
00044    // string vardas;
00045     //string pavarde;
00046     vector<int> nd;
00047     int egz;
00048     double galutinis;
00049     string reikalavimas;
00050
00051 public:
00052      Studentas() : Zmogus(), egz(0), galutinis(0.0), reikalavimas(" ") {}
00053
00054
00055     Studentas(const string& vardas, const string& pavarde, const vector<int>& nd, int egz, const
      string& reikalavimas)
00056     : Zmogus(vardas, pavarde), nd(nd), egz(egz), reikalavimas(reikalavimas) {
00057     calculateGalutinis();
00058 }
00059
00060     const string& getVardas() const { return vardas; }
00061     const string& getPavarde() const { return pavarde; }
00062     const vector<int>& getNd() const { return nd; }
00063     double getGalutinis() const { return galutinis; }
00064     int getEgz() const { return egz; }
00065     string getReikalavimas() const { return reikalavimas; }
00066
00067     static string generuoti_varda(int indeksas) {
00068         return "Vardas" + to_string(indeksas);
00069     }
00070
00071
00072     static string generuoti_pavarde(int indeksas) {
```

```
00073         return "Pavarde" + to_string(indeksas);
00074     }
00075
00076
00077     Studentas(const Studentas& other)
00078         : Zmogus(other.getVardas(), other.getPavarde()), nd(other.nd), egz(other.egz),
      galutinis(other.galutinis), reikalavimas(other.reikalavimas) {
00079
00080     }
00081
00082
00083     ~Studentas() {
00084         nd.clear();
00085         //cout « "Objekto destruktorius kvieciamas " « vardas « " " « pavarde « " objektui." «
      std::endl;
00086     }
00087
00088
00089     Studentas& operator=(const Studentas& other) {
00090     if (this == &other) return *this;
00091     vardas = other.vardas;
00092     pavarde = other.pavarde;
00093     nd = other.nd;
00094     egz = other.egz;
00095     galutinis = other.galutinis;
00096     reikalavimas = other.reikalavimas;
00097     return *this;
00098 }
00099
00100     void calculateGalutinis()  {
00101         double vidurkis_nd = 0.0;
00102         if (!nd.empty()) {
00103             vidurkis_nd = accumulate(nd.begin(), nd.end(), 0.0) / nd.size();
00104         }
00105         galutinis = 0.4 * vidurkis_nd + 0.6 * egz;
00106     }
00107
00108    void setReikalavimas(const string& reikalavimas_verte) {
00109         reikalavimas = reikalavimas_verte;
00110     }
00111
00112
00113     void setGalutinis(double newGalutinis) {
00114         galutinis = newGalutinis;
00115     }
00116
00117
00118
00119
00120     void setVardas(const string& vardas) {
00121         this->vardas = vardas;
00122     }
00123
00124
00125     void setPavarde(const string& pavarde) {
00126         this->pavarde = pavarde;
00127     }
00128
00129
00130     void setNd(const vector<int>& n) { nd = n; }
00131
00132     void setNd(const list<int>& n) { nd.assign(n.begin(), n.end()); }
00133
00134
00135     void addNd(int nd_verte) {
00136         nd.push_back(nd_verte);
00137     }
00138
00139     void setEgz(int egz_verte) {
00140         if (egz_verte >= 1 && egz_verte <= 10) {
00141             egz = egz_verte;
00142         } else {
00143             cout « "Egzamino ivertinimas turi buti nuo 1 iki 10." « endl;
00144         }
00145     }
00146
00147
00148      friend istream& operator»(istream& in, Studentas& s) {
00149         cout « "Iveskite varda: ";
00150         in » s.vardas;
00151         cout « "Iveskite pavarde: ";
00152         in » s.pavarde;
00153         cout « "Iveskite egzamino pazimi: ";
00154         in » s.egz;
00155         s.nd.clear();
00156         cout « "Iveskite namu darbu pazymius (spauskite 0, kad baigti): ";
00157         int pazymys;
```

```
00158         while (in » pazymys && pazymys != 0) {
00159             s.nd.push_back(pazymys);
00160         }
00161         s.calculateGalutinis();
00162         return in;
00163     }
00164 
00165 
00166     friend ostream& operator«(ostream& out, const Studentas& s) {
00167         out « "Vardas: " « s.vardas « ", Pavarde: " « s.pavarde « "\n";
00168         out « "Egzaminas: " « s.egz « "\n";
00169         out « "Namu darbu pazymiai: ";
00170         for (int pazymys : s.nd) {
00171             out « pazymys « " ";
00172         }
00173         out « "\nGalutinis: " « fixed « setprecision(2) « s.galutinis « "\n";
00174         return out;
00175     }
00176 
00177     static Studentas generuotiStudenta() {
00178         Studentas s;
00179         s.vardas = "Vardas" + to_string(rand() % 100 + 1);
00180         s.pavarde = "Pavarde" + to_string(rand() % 100 + 1);
00181         s.egz = rand() % 10 + 1;
00182         int pazymiuKiekis = rand() % 5 + 1;
00183         for (int i = 0; i < pazymiuKiekis; ++i) {
00184             s.nd.push_back(rand() % 10 + 1);
00185         }
00186         s.calculateGalutinis();
00187         return s;
00188     }
00189 
00190     void spausdintiInformacija() const override {
00191         cout « "Vardas: " « getVardas() « ", Pavarde: " « getPavarde() « "\n";
00192         cout « "Egzaminas: " « egz « "\n";
00193         cout « "Namu darbu pazymiai: ";
00194         for (int pazymys : nd) {
00195             cout « pazymys « " ";
00196         }
00197         cout « "\nGalutinis: " « fixed « setprecision(2) « galutinis « "\n";
00198     }
00199 
00200 
00201     static vector<Studentas> nuskaitytiIsFailo(const string& failoVardas) {
00202         ifstream failas(failoVardas);
00203         vector<Studentas> studentai;
00204         if (!failas) {
00205             cerr « "Nepavyko atidaryti failo: " « failoVardas « endl;
00206             return studentai;
00207         }
00208         Studentas s;
00209         while (failas » s) {
00210             studentai.push_back(s);
00211         }
00212         failas.close();
00213         return studentai;
00214     }
00215      static void rasytiIFaila(const Studentas& studentas, const string& failoVardas) {
00216     ofstream failas(failoVardas);
00217     if (!failas) {
00218         cerr « "Nepavyko sukurti failo: " « failoVardas « endl;
00219         return;
00220     }
00221     failas « studentas;
00222     failas.close();
00223 }
00224 
00225 
00226 };
00227 
00228 //void galutinis_balas_vid(vector<Studentas>& studentai);
00229 //void galutinis_balas_med(vector<Studentas>& studentai);
00230 void print_results(const vector<Studentas>& studentai);
00231 void patikrinimas(vector<Studentas>& studentai);
00232 void skaityti(vector<Studentas>& studentai, string name, string kriterijus);
00233 void duomenys(vector<Studentas>& studentai);
00234 void sort_students(vector<Studentas>& studentai);
00235 bool compare_students(const Studentas& a, const Studentas& b);
00236 string capitalize(string var);
00237 string tolowers(string var);
00238 
00239 void print_results_list(const list<Studentas>& studentai_list);
00240 void duomenys_list(list<Studentas>& studentai_list);
00241 //void galutinis_balas_med_list(list<Studentas>& studentai_list);
00242 //void galutinis_balas_vid_list(list<Studentas>& studentai_list);
00243 void skaityti_list(list<Studentas>& studentai_list, const string& name, const string& kriterijus);
00244 //string exec(const char* cmd);
```

```
00245
00246
00247
00248 #endif // STUDENTAS_H_INCLUDED
```

Index