Studentų analizė

v2.0

Generated by Doxygen 1.13.2

1 README 1

2 Hierarchical Index 3

2.1 Class Hierarchy . . . 3

3 Class Index 5

3.1 Class List . . . 5

4 File Index 7

4.1 File List . . . 7

5 Class Documentation 9

5.1 Studentas Class Reference . . . 9

5.1.1 Member Function Documentation . . . 10

5.1.1.1 pavarde() . . . 10

5.1.1.2 setPavarde() . . . 10

5.1.1.3 setVardas() . . . 10

5.1.1.4 vardas() . . . 10

5.2 Zmogus Class Reference . . . 10

6 File Documentation 13

6.1 student.h . . . 13

Index 17

Chapter 1

README

v1.2

Pridėti ir ištestuoti visi reikiami konstruktoriai ir operatoriai, kad atitiktų "Rule of five"..

Sukurtas meniu su galimais pasirinkimais:

1 - Konstruktorius

2 - Destruktorius

3 - Copy constructor

4 - Copy assignment operator

5 - Move constructor

6 - Move operator

7 - Input operator

8 - Output operator

9 - studentų grupavimas ir išvedimas į failus

```
9.1 - įvesti duomenis ranka

9.2 - generuoti duomenis

9.3 - nuskaityti duomenis iš failo
```

---

v1.1
Kopiuterio parametrai: CPU - Apple M3 RAM - 16 GB SSD - 494,38 GB
2 tyrimas: su skirtingais optimizavimo flag'ais (O1, O2, O3):

Struktūra pakeista į klasę ir visas kodas atitinkamai pritaikytas. Matuojamas nuskaitymo iš failo laikas, rikiavimas ir grupavimas. Galimas pasirinkimas išvedimo į ekraną arba du failus.
1 tyrimas: atliktas su dviem failais. Išrinktas geričiausiai veikiantis konteineris ir strategija, palyginti rezultatai senos kodo versijos su struktūra, bei naujos su klase.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Zmogus 10
Studentas 9

Chapter 3

Class Index

3.1 Class List

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

Studentas 9
Zmogus 10

Chapter 4

File Index

4.1 File List

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

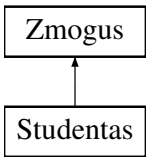
student.h 13

Chapter 5

Class Documentation

5.1 Studentas Class Reference

Inheritance diagram for Studentas:



Public Member Functions

- **Studentas** (const Studentas &other)
- Studentas & **operator=** (const Studentas &other)
- **Studentas** (Studentas &&other) noexcept
- Studentas & **operator=** (Studentas &&other) noexcept
- string vardas () const override
- string pavarde () const override
- double **galutinisVid** () const
- double **galutinisMed** () const
- vector< int > **getNamuDarbai** () const
- int **getEgzaminas** () const
- size_t **getNamuDarbaiSize** () const
- void setVardas (const string &vardas) override
- void setPavarde (const string &pavarde) override
- void **setEgzaminas** (int egzas)
- void **setNamuDarbai** (const vector< int > &nd)
- void **setGalutinisVid** (double vid)
- void **setGalutinisMed** (double med)
- istream & **readStudent** (istream &is)
- void **skaiciuotiGalutinius** ()

Public Member Functions inherited from Zmogus

- **Zmogus** (const string &vardas, const string &pavarde)

Static Public Member Functions

- static double **skaiciuotiVidurki** (const vector< int > &pazymiai)
- static double **skaiciuotiMediana** (vector< int > pazymiai)

Friends

- ostream & **operator**<< (ostream &os, const Studentas &s)
- istream & **operator**>> (istream &is, Studentas &s)

Additional Inherited Members

Protected Attributes inherited from Zmogus

- string **vardas_**
- string **pavarde_**

5.1.1 Member Function Documentation

5.1.1.1 pavarde()

```
string Studentas::pavarde () const  [inline], [override], [virtual]
```

Implements Zmogus.

5.1.1.2 setPavarde()

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

Implements Zmogus.

5.1.1.3 setVardas()

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

Implements Zmogus.

5.1.1.4 vardas()

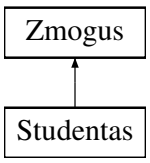
```
string Studentas::vardas () const  [inline], [override], [virtual]
```

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

- student.h
- student.cpp

5.2 Zmogus Class Reference

Inheritance diagram for Zmogus:



Public Member Functions

- **Zmogus** (const string &vardas, const string &pavarde)
- virtual string **vardas** () const =0
- virtual string **pavarde** () const =0
- virtual void **setVardas** (const string &vardas)=0
- virtual void **setPavarde** (const string &pavarde)=0

Protected Attributes

- string **vardas_**
- string **pavarde_**

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

- student.h

Chapter 6

File Documentation

6.1 student.h

```
00001 #ifndef STUDENT_H
00002 #define STUDENT_H
00003
00004 #include <iostream>
00005 #include <vector>
00006 #include <string>
00007 #include <iomanip>
00008 #include <algorithm>
00009 #include <numeric>
00010 #include <stdexcept>
00011 #include <fstream>
00012 #include <sstream>
00013
00014 using namespace std;
00015 using namespace std::chrono;
00016
00017 class Zmogus {
00018     protected:
00019         string vardas_;
00020         string pavarde_;
00021     public:
00022         Zmogus() = default;
00023         Zmogus(const string& vardas, const string& pavarde) : vardas_(vardas), pavarde_(pavarde) {}
00024         virtual ~Zmogus() = default;
00025
00026         virtual string vardas() const = 0;
00027         virtual string pavarde() const = 0;
00028
00029         virtual void setVardas(const string& vardas) = 0;
00030         virtual void setPavarde(const string& pavarde) = 0;
00031 };
00032
00033 class Studentas : public Zmogus {
00034     private:
00035         vector<int> namuDarbai_;
00036         int egzaminas_;
00037         double galutinisVid_;
00038         double galutinisMed_;
00039
00040     public:
00041         // Konstruktoriai ir destruktoriai
00042         Studentas();
00043         ~Studentas();
00044
00045         Studentas(const Studentas& other);
00046         Studentas& operator=(const Studentas& other);
00047         Studentas(Studentas&& other) noexcept;
00048         Studentas& operator=(Studentas&& other) noexcept;
00049
00050         // Getteriai
00051         string vardas() const override { return vardas_; }
00052         string pavarde() const override { return pavarde_; }
00053         double galutinisVid() const { return galutinisVid_; }
00054         double galutinisMed() const { return galutinisMed_; }
00055         vector<int> getNamuDarbai() const { return namuDarbai_; }
00056         int getEgzaminas() const { return egzaminas_; }
00057         size_t getNamuDarbaiSize() const { return namuDarbai_.size(); }
00058
00059         // Setteriai
00060         void setVardas(const string& vardas) override { vardas_ = vardas; }
00061         void setPavarde(const string& pavarde) override { pavarde_ = pavarde; }
```

```
00062         void setEgzaminas(int egzas) { egzaminas_ = egzas; }
00063         void setNamuDarbai(const vector<int>& nd) { namuDarbai_ = nd; }
00064         void setGalutinisVid(double vid) { galutinisVid_ = vid; }
00065         void setGalutinisMed(double med) { galutinisMed_ = med; }
00066 
00067         // Funkcijos
00068         istream& readStudent(istream& is);
00069 
00070         void skaiciuotiGalutinius();
00071         static double skaiciuotiVidurki(const vector<int>& pazymiai);
00072         static double skaiciuotiMediana(vector<int> pazymiai);
00073 
00074         // Operatoriai
00075         friend ostream& operator<<(ostream& os, const Studentas& s);
00076         friend istream& operator>>(istream& is, Studentas& s);
00077     };
00078 
00079 template <typename konteineris>
00080 void nuskaitytiIsFailo(konteineris &studentai) {
00081     string failoPavadinimas;
00082     ifstream failas;
00083 
00084     while (true) {
00085         try {
00086             cout << "\nĮveskite failo pavadinimą: ";
00087             cin >> failoPavadinimas;
00088 
00089             failas.open(failoPavadinimas);
00090             if (!failas) throw runtime_error("Nepavyko atidaryti failo!");
00091 
00092             break;
00093         }
00094         catch (const exception& e) {
00095             cout << e.what() << " Bandykite dar kartą.\n";
00096             cin.clear();
00097             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00098         }
00099     }
00100 
00101     auto start = steady_clock::now();
00102 
00103     string eilute;
00104     getline(failas, eilute);
00105 
00106     while (getline(failas, eilute)) {
00107         istringstream line(eilute);
00108         Studentas stud;
00109 
00110 
00111         stud.readStudent(line);
00112 
00113         studentai.push_back(stud);
00114     }
00115 
00116     failas.close();
00117 
00118     auto end = steady_clock::now();
00119     auto trukme = duration_cast<duration<double>>(end - start);
00120     cout << fixed << setprecision(3);
00121     cout << "Duomenų nuskaitymas užtruko: " << trukme.count() << " s\n" << endl;
00122 }
00123 
00124 template <typename konteineris>
00125 void rusiuotiStudentus(konteineris &studentai){
00126 
00127     int rusiavimoPasirinkimas;
00128 
00129     while (true) {
00130         try {
00131             cout << "Pasirinkite rikiavimo būdą:\n";
00132             cout << "1 - Pagal vardą (A-Z)\n";
00133             cout << "2 - Pagal pavardę (A-Z)\n";
00134             cout << "3 - Pagal galutinį vidurkį\n";
00135             cout << "4 - Pagal galutinę medianą\n";
00136             cout << "Jūsų pasirinkimas: ";
00137             cin >> rusiavimoPasirinkimas;
00138 
00139             if (cin.fail()) {
00140                 throw invalid_argument("Neteisinga įvestis! Įveskite tik skaičių.");
00141             }
00142             if (rusiavimoPasirinkimas < 1 || rusiavimoPasirinkimas > 4) {
00143                 throw out_of_range("Pasirinkimas turi būti nuo 1 iki 4.");
00144             }
00145             break;
00146         }
00147         catch (const exception &e) {
00148             cout << e.what() << " Bandykite dar kartą.\n";
```

```
00149             cin.clear();
00150             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00151         }
00152     }
00153 
00154     auto start = steady_clock::now();
00155 
00156     switch (rusiavimoPasirinkimas) {
00157         case 1:
00158             std::sort(studentai.begin(), studentai.end(), [](const Studentas& a, const Studentas& b) {
00159                 return a.vardas() < b.vardas();
00160             });
00161             break;
00162         case 2:
00163             std::sort(studentai.begin(), studentai.end(), [](const Studentas& a, const Studentas& b) {
00164                 return a.pavarde() < b.pavarde();
00165             });
00166             break;
00167         case 3:
00168             std::sort(studentai.begin(), studentai.end(), [](const Studentas& a, const Studentas& b) {
00169                 return a.galutinisVid() < b.galutinisVid();
00170             });
00171             break;
00172         case 4:
00173             std::sort(studentai.begin(), studentai.end(), [](const Studentas& a, const Studentas& b) {
00174                 return a.galutinisMed() < b.galutinisMed();
00175             });
00176             break;
00177     }
00178 
00179     auto end = steady_clock::now();
00180     auto trukme = duration_cast<duration<double»(end - start);
00181     cout « fixed « setprecision(3);
00182     cout « "Rikiavimas užtruko: " « trukme.count() « " s\n" « endl;
00183 }
00184 
00185 template <typename konteineris>
00186 void strategija_3(konteineris& studentai, konteineris& nuskriaustukai) {
00187 
00188     char grupavimoPasirinkimas;
00189     Studentas stud;
00190 
00191     while (true) {
00192         try {
00193             cout « "Pasirinkite pagal ką bus sugrupuoti studentai (V - pagal vidurkį, M - pagal
      medianą): ";
00194             cin » grupavimoPasirinkimas;
00195 
00196             if (grupavimoPasirinkimas != 'V' && grupavimoPasirinkimas != 'v' && grupavimoPasirinkimas
      != 'M' && grupavimoPasirinkimas != 'm') {
00197                 throw invalid_argument("Neteisinga įvestis! Pasirinkite V arba M.");
00198             }
00199 
00200             break;
00201         }
00202         catch (const invalid_argument& e) {
00203             cout « e.what() « " Bandykite dar kartą.\n";
00204             cin.clear();
00205             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00206             }
00207         }
00208 
00209     auto start = steady_clock::now();
00210 
00211     auto it = stable_partition(studentai.begin(), studentai.end(), [grupavimoPasirinkimas](const
      Studentas& stud) {
00212         if (grupavimoPasirinkimas == 'V' || grupavimoPasirinkimas == 'v') {
00213             return stud.galutinisVid() >= 5;
00214         } else {
00215             return stud.galutinisMed() >= 5;
00216         }
00217     });
00218 
00219     nuskriaustukai.assign(it, studentai.end());
00220     studentai.erase(it, studentai.end());
00221 
00222     auto end = steady_clock::now();
00223 
00224     if constexpr (is_same_v<konteineris, vector<Studentas» || is_same_v<konteineris, deque<Studentas»)
      {
00225         studentai.shrink_to_fit();
00226     }
00227 
00228     auto trukme = duration_cast<duration<double»(end - start);
00229     cout « fixed « setprecision(3);
00230     cout « "3 strategija užtruko: " « trukme.count() « " s\n" « endl;
00231 }
```

```
00232
00233 template <typename konteineris>
00234 void isvestiIDuFailus(konteineris& nuskriaustukai, konteineris& studentai){
00235
00236     ofstream outNuskriaustukai("nuskriaustukai.txt"), outKietekai("kietekai.txt");
00237     if (!outNuskriaustukai || !outKietekai) {
00238         cerr « "Nepavyko sukurti rezultatų failų!" « endl;
00239         return;
00240     }
00241
00242     outNuskriaustukai « left « setw(20) « "Pavarde" « setw(20) « "Vardas" « setw(20) « "Galutinis
      (Vid.)" « setw(20) « "Galutinis (Med.)" « endl;
00243     outNuskriaustukai « string(70, '-') « endl;
00244     outKietekai « left « setw(20) « "Pavarde" « setw(20) « "Vardas" « setw(20) « "Galutinis (Vid.)" «
     setw(20) « "Galutinis (Med.)" « endl;
00245     outKietekai « string(70, '-') « endl;
00246
00247     for (const auto& stud : nuskriaustukai) {
00248         outNuskriaustukai « left « setw(20) « stud.pavarde() « setw(20) « stud.vardas() « fixed «
     setprecision(2) « setw(20) « stud.galutinisVid() « setw(20) « stud.galutinisMed() « endl;
00249     }
00250     for (const auto& stud : studentai) {
00251         outKietekai « left « setw(20) « stud.pavarde() « setw(20) « stud.vardas() « fixed «
     setprecision(2) « setw(20) « stud.galutinisVid() « setw(20) « stud.galutinisMed() « endl;
00252     }
00253
00254     cout « "Failai \"nuskriaustukai.txt\" ir \"kietekai.txt\" sukurti!" « endl;
00255 }
00256
00257 void ivestiStudenta(vector<Studentas>& studentai);
00258
00259 void generuotiStudentus(vector<Studentas>& studentai);
00260
00261 void testDestructor();
00262
00263 void testConstructor();
00264
00265 void testCopyConstructor();
00266
00267 void testCopyAssignment();
00268
00269 void testMoveConstructor();
00270
00271 void testMoveAssignment();
00272
00273 void testInputOperator();
00274
00275 void testOutputOperator();
00276
00277 #endif
```

Index

pavarde
    Studentas, 10

README, 1

setPavarde
    Studentas, 10
setVardas
    Studentas, 10
Studentas, 9
    pavarde, 10
    setPavarde, 10
    setVardas, 10
    vardas, 10

vardas
    Studentas, 10

Zmogus, 10