Objektinio-programavimo-uzd-2

Generated by Doxygen 1.13.2

Chapter 1

Objektinio-programavimo-uzd

Ši programa del skirtingu konteineriu laikų testavimų naudojant struktūras Kad paleist programa reikia tik paleisti run.bat faila

CPU: AMD Ryzen 7 8845, 8 Cores, 3,8 GHZ
RAM: 16 GB, 5600 MT/s
SSD NVMe

2.0

Paleisti programa naudokite run.bat file, o unit testus paleisti reikia minimum 3.14 cmake versijos ir paleisti test.bat faila

Peržiurėti dokumentacija per html, reikia eiti į html aplankala ir rastie index faila Taipat yra sukompiliuotas pdf failas latex aplankale

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

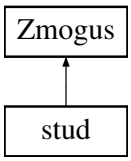
Chapter 5

Class Documentation

5.1 stud Class Reference

`#include <stud.h>`

Inheritance diagram for stud:



Public Member Functions

- stud ()
- stud (const string &vardas, const string &pavarde)
- stud (std::istringstream &line)
- stud (const stud &kitas)
- stud (stud &&kitas) noexcept
- ~stud ()
- stud & operator= (const stud &kitas)
- stud & operator= (stud &&kitas) noexcept
- void addTarpPazymys (int paz)
- void setEgzaminas (double egzaminas)
- void setVid (double vid)
- void setMed (double med)
- vector< int > & getTarp ()
- double getEgz () const
- double getGalutinisVid () const
- double getGalutinisMed () const
- void calculateGalutinis ()

Public Member Functions inherited from Zmogus

- Zmogus ()=default
- Zmogus (const string &vard, const string &pava)
- virtual ~Zmogus ()=default
- Zmogus (const Zmogus &other)
- Zmogus & operator= (const Zmogus &other)
- Zmogus (Zmogus &&other) noexcept
- Zmogus & operator= (Zmogus &&other) noexcept
- void setvard (const string &vardas)
- void setpava (const string &pavard)
- string getVardas () const
- string getPavarde () const

Friends

- std::ostream & operator<< (std::ostream &out, const stud &a)
- std::istream & operator>> (std::istream &in, stud &a)

Additional Inherited Members

Protected Attributes inherited from Zmogus

- string vard
- string pava

5.1.1 Detailed Description

Definition at line 8 of file stud.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 stud() [1/5]

```
stud::stud ()  [inline]
```

Definition at line 19 of file stud.h.

5.1.2.2 stud() [2/5]

```
stud::stud (
            const string & vardas,
            const string & pavarde)  [inline]
```

Definition at line 21 of file stud.h.

5.1.2.3 stud() [3/5]

```
stud::stud (
            std::istringstream & line)  [inline]
```

Definition at line 23 of file stud.h.

5.1.2.4 stud() [4/5]

```
stud::stud (
            const stud & kitas)  [inline]
```

Definition at line 36 of file stud.h.

5.1.2.5 stud() [5/5]

```
stud::stud (
            stud && kitas)  [inline], [noexcept]
```

Definition at line 44 of file stud.h.

5.1.2.6 ~stud()

```
stud::~stud ()  [inline]
```

Definition at line 56 of file stud.h.

5.1.3 Member Function Documentation

5.1.3.1 addTarpPazymys()

```
void stud::addTarpPazymys (
            int paz)  [inline], [virtual]
```

Implements Zmogus.

Definition at line 90 of file stud.h.

5.1.3.2 calculateGalutinis()

```
void stud::calculateGalutinis ()  [inline]
```

Definition at line 105 of file stud.h.

5.1.3.3 getEgz()

```
double stud::getEgz () const  [inline]
```

Definition at line 100 of file stud.h.

5.1.3.4 getGalutinisMed()

```
double stud::getGalutinisMed () const  [inline]
```

Definition at line 102 of file stud.h.

5.1.3.5 getGalutinisVid()

```
double stud::getGalutinisVid () const  [inline]
```

Definition at line 101 of file stud.h.

5.1.3.6 getTarp()

```
vector< int > & stud::getTarp ()  [inline]
```

Definition at line 99 of file stud.h.

5.1.3.7 operator=() [1/2]

```
stud & stud::operator= (
            const stud & kitas)  [inline]
```

Definition at line 62 of file stud.h.

5.1.3.8 operator=() [2/2]

```
stud & stud::operator= (
            stud && kitas)  [inline], [noexcept]
```

Definition at line 74 of file stud.h.

5.1.3.9 setEgzaminas()

```
void stud::setEgzaminas (
            double egzaminas)  [inline]
```

Definition at line 94 of file stud.h.

5.1.3.10 setMed()

```
void stud::setMed (
            double med)  [inline]
```

Definition at line 96 of file stud.h.

5.1.3.11 setVid()

```
void stud::setVid (
            double vid)  [inline]
```

Definition at line 95 of file stud.h.

5.1.4 Friends And Related Symbol Documentation

5.1.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & out,
            const stud & a)  [friend]
```

Definition at line 120 of file stud.h.

5.1.4.2 operator>>

```
std::istream & operator>> (
            std::istream & in,
            stud & a)  [friend]
```

Definition at line 126 of file stud.h.

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

- stud.h

5.2 Timer Class Reference

```
#include <Timer.h>
```

Public Member Functions

- Timer ()
- void reset ()
- double elapsed () const

5.2.1 Detailed Description

Definition at line 6 of file Timer.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Timer()

```
Timer::Timer ()  [inline]
```

Definition at line 13 of file Timer.h.

5.2.3 Member Function Documentation

5.2.3.1 elapsed()

```
double Timer::elapsed () const  [inline]
```

Definition at line 17 of file Timer.h.

5.2.3.2 reset()

```
void Timer::reset ()  [inline]
```

Definition at line 14 of file Timer.h.

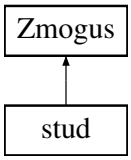
The documentation for this class was generated from the following file:

- Timer.h

5.3 Zmogus Class Reference

```
#include <Zmogus.h>
```

Inheritance diagram for Zmogus:



Public Member Functions

- Zmogus ()=default
- Zmogus (const string &vard, const string &pava)
- virtual ~Zmogus ()=default
- Zmogus (const Zmogus &other)
- Zmogus & operator= (const Zmogus &other)
- Zmogus (Zmogus &&other) noexcept
- Zmogus & operator= (Zmogus &&other) noexcept
- void setvard (const string &vardas)
- void setpava (const string &pavard)
- string getVardas () const
- string getPavarde () const
- virtual void addTarpPazymys (int paz)=0

Protected Attributes

- string vard
- string pava

5.3.1 Detailed Description

Definition at line 7 of file Zmogus.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Zmogus() [1/4]

```
Zmogus::Zmogus ()  [default]
```

5.3.2.2 Zmogus() [2/4]

```
Zmogus::Zmogus (
            const string & vard,
            const string & pava)  [inline]
```

Definition at line 12 of file Zmogus.h.

5.3.2.3 ~Zmogus()

```
virtual Zmogus::~Zmogus ()  [virtual], [default]
```

5.3.2.4 Zmogus() [3/4]

```
Zmogus::Zmogus (
            const Zmogus & other)  [inline]
```

Definition at line 18 of file Zmogus.h.

5.3.2.5 Zmogus() [4/4]

```
Zmogus::Zmogus (
            Zmogus && other)  [inline], [noexcept]
```

Definition at line 31 of file Zmogus.h.

5.3.3 Member Function Documentation

5.3.3.1 addTarpPazymys()

```
virtual void Zmogus::addTarpPazymys (
            int paz)  [pure virtual]
```

Implemented in stud.

5.3.3.2 getPavarde()

```
string Zmogus::getPavarde () const  [inline]
```

Definition at line 47 of file Zmogus.h.

5.3.3.3 getVardas()

```
string Zmogus::getVardas () const  [inline]
```

Definition at line 46 of file Zmogus.h.

5.3.3.4 operator=() [1/2]

```
Zmogus & Zmogus::operator= (
            const Zmogus & other)  [inline]
```

Definition at line 22 of file Zmogus.h.

5.3.3.5 operator=() [2/2]

```
Zmogus & Zmogus::operator= (
            Zmogus && other)  [inline], [noexcept]
```

Definition at line 35 of file Zmogus.h.

5.3.3.6 setpava()

```
void Zmogus::setpava (
            const string & pavard)  [inline]
```

Definition at line 44 of file Zmogus.h.

5.3.3.7 setvard()

```
void Zmogus::setvard (
            const string & vardas)  [inline]
```

Definition at line 43 of file Zmogus.h.

5.3.4 Member Data Documentation

5.3.4.1 pava

```
string Zmogus::pava  [protected]
```

Definition at line 54 of file Zmogus.h.

5.3.4.2 vard

```
string Zmogus::vard  [protected]
```

Definition at line 53 of file Zmogus.h.

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

- Zmogus.h

Chapter 6

File Documentation

6.1 automatiskas.cpp File Reference

```
#include "bibl.h"
```

Functions

- void automatiskas (vector< stud > &A)

Variables

- string Vard [5] {"Jonas", "Vytautas", "Antanas", "Tomas", "Juozas"}
- string Pava [5] {"Kazlauskas", "Stankevicius", "Petrauskas", "Janauskas", "Zukauskas"}

6.1.1 Function Documentation

6.1.1.1 automatiskas()

```
void automatiskas (
            vector< stud > & A)
```

Definition at line 6 of file automatiskas.cpp.

6.1.2 Variable Documentation

6.1.2.1 Pava

```
string Pava[5] {"Kazlauskas", "Stankevicius", "Petrauskas", "Janauskas", "Zukauskas"}
```

Definition at line 4 of file automatiskas.cpp.

6.1.2.2 Vard

```
string Vard[5] {"Jonas", "Vytautas", "Antanas", "Tomas", "Juozas"}
```

Definition at line 3 of file automatiskas.cpp.

6.2 automatiskas.cpp

Go to the documentation of this file.

```
00001 #include "bibl.h"
00002
00003 string Vard[5] {"Jonas", "Vytautas", "Antanas", "Tomas", "Juozas"};
00004 string Pava[5] {"Kazlauskas", "Stankevicius", "Petrauskas", "Janauskas", "Zukauskas"};
00005
00006 void automatiskas (vector <stud> &A){
00007     std::random_device rd;
00008     std::mt19937 mt(rd());
00009     std::uniform_int_distribution<int> vardui(0,4);
00010     std::uniform_int_distribution<int> pazymiui(0,10);
00011
00012     stud temp(Vard[vardui(mt)],Pava[vardui(mt)]);
00013
00014
00015     for (int i=0;i<pazymiui(mt);i++){
00016         int paz=pazymiui(mt);
00017         temp.addTarpPazymys(paz);
00018     }
00019     temp.setEgzaminas(pazymiui(mt));
00020
00021     temp.calculateGalutinis();
00022     A.push_back(temp);
00023 }
```

6.3 bibl.h File Reference

```
#include "stud.h"
#include "std.h"
#include "Timer.h"
```

Functions

- void spausdina (vector< stud >)
- void rankinis (vector< stud > &)
- void pusrankis (vector< stud > &)
- void automatiskas (vector< stud > &)
- void compare (vector< stud > &, double &)
- void compare (vector< stud > &)
- void failoNusk (vector< stud > &)
- void failoGen ()
- void rusiavimas ()
- void spausdinaFaila (vector< stud > &A, string)
- void test ()
- bool compVardas (stud &, stud &)
- bool compPavard (stud &, stud &)
- bool compVid (stud &, stud &)
- bool compMed (stud &, stud &)

Variables

- string Vard [5]
- string Pava [5]

6.3.1 Function Documentation

6.3.1.1 automatiskas()

```
void automatiskas (
            vector< stud > & A)
```

Definition at line 6 of file automatiskas.cpp.

6.3.1.2 compare() [1/2]

```
void compare (
            vector< stud > & A)
```

Definition at line 3 of file compare.cpp.

6.3.1.3 compare() [2/2]

```
void compare (
            vector< stud > & A,
            double & time)
```

Definition at line 43 of file compare.cpp.

6.3.1.4 compMed()

```
bool compMed (
            stud & a,
            stud & b)
```

Definition at line 98 of file compare.cpp.

6.3.1.5 compPavard()

```
bool compPavard (
            stud & a,
            stud & b)
```

Definition at line 90 of file compare.cpp.

6.3.1.6 compVardas()

```
bool compVardas (
            stud & a,
            stud & b)
```

Definition at line 86 of file compare.cpp.

6.3.1.7 compVid()

```
bool compVid (
            stud & a,
            stud & b)
```

Definition at line 94 of file compare.cpp.

6.3.1.8 failoGen()

```
void failoGen ()
```

Definition at line 3 of file failoGen.cpp.

6.3.1.9 failoNusk()

```
void failoNusk (
            vector< stud > & A)
```

Definition at line 3 of file failoNusk.cpp.

6.3.1.10 pusrankis()

```
void pusrankis (
            vector< stud > & A)
```

Definition at line 3 of file pusrankis.cpp.

6.3.1.11 rankinis()

```
void rankinis (
            vector< stud > & A)
```

Definition at line 3 of file rankinis.cpp.

6.3.1.12 rusiavimas()

```
void rusiavimas ()
```

plz fix kazkodel cia zymiai leciau veikia negu turetu(galimai destructor ir move reik?)

Definition at line 3 of file rusiavimas.cpp.

6.3.1.13 spausdina()

```
void spausdina (
            vector< stud > A)
```

Definition at line 3 of file spausdina.cpp.

6.3.1.14 spausdinaFaila()

```
void spausdinaFaila (
            vector< stud > & A,
            string failas)
```

Definition at line 12 of file spausdina.cpp.

6.3.1.15 test()

```
void test ()
```

Definition at line 93 of file test.cpp.

6.3.2 Variable Documentation

6.3.2.1 Pava

```
string Pava[5]  [extern]
```

Definition at line 4 of file automatiskas.cpp.

6.3.2.2 Vard

```
string Vard[5]  [extern]
```

Definition at line 3 of file automatiskas.cpp.

6.4 bibl.h

Go to the documentation of this file.

```
00001 # ifndef BIBL_H
00002 # define BIBL_H
00003
00004
00005 #include "stud.h"
00006 #include "std.h"
00007 #include "Timer.h"
00008
00009
00010 void spausdina(vector <stud> );
00011 void rankinis(vector <stud> &);
00012 void pusrankis(vector <stud> &);
00013 void automatiskas (vector <stud> &);
00014 void compare(vector <stud> &, double &);
00015 void compare(vector <stud> &);
00016 void failoNusk (vector <stud> &);
00017 void failoGen();
00018 void rusiavimas();
00019 void spausdinaFaila(vector <stud> &A, string);
00020 void test();
00021
00022 bool compVardas(stud &, stud &);
00023 bool compPavard(stud &, stud &);
00024 bool compVid(stud &, stud &);
00025 bool compMed(stud &, stud &);
00026
00027 extern string Vard[5];
00028 extern string Pava[5];
00029
00030
00031 # endif
```

6.5 compare.cpp File Reference

```
#include "bibl.h"
```

Functions

- void compare (vector< stud > &A)
- void compare (vector< stud > &A, double &time)
- bool compVardas (stud &a, stud &b)
- bool compPavard (stud &a, stud &b)
- bool compVid (stud &a, stud &b)
- bool compMed (stud &a, stud &b)

6.5.1 Function Documentation

6.5.1.1 compare() [1/2]

```
void compare (
            vector< stud > & A)
```

Definition at line 3 of file compare.cpp.

6.5.1.2 compare() [2/2]

```
void compare (
            vector< stud > & A,
            double & time)
```

Definition at line 43 of file compare.cpp.

6.5.1.3 compMed()

```
bool compMed (
            stud & a,
            stud & b)
```

Definition at line 98 of file compare.cpp.

6.5.1.4 compPavard()

```
bool compPavard (
            stud & a,
            stud & b)
```

Definition at line 90 of file compare.cpp.

6.5.1.5 compVardas()

```
bool compVardas (
            stud & a,
            stud & b)
```

Definition at line 86 of file compare.cpp.

6.5.1.6 compVid()

```
bool compVid (
            stud & a,
            stud & b)
```

Definition at line 94 of file compare.cpp.

6.6 compare.cpp

Go to the documentation of this file.

```
00001 #include "bibl.h"
00002
00003 void compare(vector <stud> &A){
00004     while(true){
00005         cout « "1 - Pagal Varda " « endl;
00006         cout « "2 - Pagal Pavarde " « endl;
00007         cout « "3 - Pagal pazymiu vidurki " « endl;
00008         cout « "4 - Pagal pazymiu mediana " « endl;
00009         int input;
00010         try {
00011             if (!(cin»input)||input<1 || input>4){
00012                 cin.clear();
00013                 cin.ignore();
00014                 throw "Ivestas neteisingas simbolis";
00015             }
00016             switch(input){
00017                 case 1:
00018                     std::sort(A.begin(),A.end(), compVardas);
00019                     break;
00020
00021                 case 2:
00022                     std::sort(A.begin(),A.end(), compPavard);
00023                     break;
00024
00025                 case 3:
00026                     std::sort(A.begin(),A.end(), compVid);
00027                     break;
00028
00029                 case 4:
00030                     std::sort(A.begin(),A.end(), compMed);
00031                     break;
00032             }
00033             break;
00034         }
00035         catch (char const *x){
00036             cout « x « endl;
00037             continue;
00038             cout « "Pagal ka isrusiuoti duomenis?" « endl;
00039         }
00040     }
00041 }
00042
00043 void compare(vector <stud> &A,double &time){
00044     Timer laik;
00045     while(true){
00046         cout « "1 - Pagal Varda " « endl;
00047         cout « "2 - Pagal Pavarde " « endl;
00048         cout « "3 - Pagal pazymiu vidurki " « endl;
00049         cout « "4 - Pagal pazymiu mediana " « endl;
00050         int input;
00051         try {
00052             if (!(cin»input)||input<1 || input>4){
00053                 cin.clear();
00054                 cin.ignore();
00055                 throw "Ivestas neteisingas simbolis";
00056             }
00057             laik.reset();
00058             switch(input){
00059                 case 1:
00060                     std::sort(A.begin(),A.end(), compVardas);
00061                     break;
00062
00063                 case 2:
00064                     std::sort(A.begin(),A.end(), compPavard);
00065                     break;
00066
00067                 case 3:
00068                     std::sort(A.begin(),A.end(), compVid);
00069                     break;
00070
00071                 case 4:
00072                     std::sort(A.begin(),A.end(), compMed);
00073                     break;
00074             }
00075             time+=laik.elapsed();
00076             break;
00077         }
00078         catch (char const *x){
00079             cout « x « endl;
00080             continue;
00081             cout « "Pagal ka isrusiuoti duomenis?" « endl;
00082         }
```

```
00083     }
00084 }
00085
00086 bool compVardas(stud &a, stud &b){
00087     return a.getVardas()<b.getVardas();
00088   }
00089
00090   bool compPavard(stud &a, stud &b){
00091     return a.getPavarde()<b.getPavarde();
00092   }
00093
00094   bool compVid(stud &a, stud &b){
00095     return a.getGalutinisVid()<b.getGalutinisVid();
00096   }
00097
00098   bool compMed(stud &a, stud &b){
00099     return a.getGalutinisMed()<b.getGalutinisMed();
00100   }
```

6.7 failoGen.cpp File Reference

```
#include "bibl.h"
```

Functions

- void failoGen ()

6.7.1 Function Documentation

6.7.1.1 failoGen()

```
void failoGen ()
```

Definition at line 3 of file failoGen.cpp.

6.8 failoGen.cpp

Go to the documentation of this file.

```
00001 #include "bibl.h"
00002
00003 void failoGen(){
00004     string failas;
00005     int kiek;
00006     int pazkiek;
00007     std::random_device rd;
00008     std::mt19937 mt(rd());
00009     std::uniform_int_distribution <int> pazymiui(0,10);
00010     Timer t;
00011
00012     cout « "Iveskite failo pavadinima (pvz. kursiokai)" « endl;
00013     cin » failas;
00014     cout « "Iveskite kiek sugeneruoti studentu" « endl;
00015     cin » kiek;
00016     cout « "Iveskite kiek pazymiu tures studentai (neskaiciuojant egzamino)" « endl;
00017     cin » pazkiek;
00018
00019     t.reset();
00020     std::stringstream eil;
00021
00022     std::ofstream rf(failas+".txt");
00023
00024     eil «std::left «setw(15)« "Vardas"« setw(15) « "Pavarde" ;
```

```
00025     for (int i=1;i<=pazkiek;i++){
00026         eil  « "ND"« setw(5) «std::to_string(i);
00027     }
00028     eil «  "Egz." « "\n";
00029
00030     rf « eil.str();
00031
00032     eil.str("");
00033
00034     for (int i=1;i<=kiek;i++){
00035         eil«setw(15) «"Vardas" + to_string(i) «setw(15)« "Pavarde" + to_string(i);
00036         for (int j=0;j<pazkiek;j++){
00037             eil « setw(7) « pazymiui(mt);
00038         }
00039         eil « setw(7) « pazymiui(mt) « "\n";
00040         rf « eil.str();
00041         eil.str("");
00042     }
00043     rf.close();
00044     cout « "failu kurimas ir jo uzdarymas uztruko " « t.elapsed() « endl;
00045 }
```

6.9 failoNusk.cpp File Reference

```
#include "bibl.h"
```

Functions

- void failoNusk (vector< stud > &A)

6.9.1 Function Documentation

6.9.1.1 failoNusk()

```
void failoNusk (
            vector< stud > & A)
```

Definition at line 3 of file failoNusk.cpp.

6.10 failoNusk.cpp

Go to the documentation of this file.

```
00001 #include "bibl.h"
00002
00003 void failoNusk (vector <stud> &A){
00004     string failas;
00005
00006     cout « "Iveskite failo pavadinima (pvz. kursiokai.txt)" « endl;
00007     while(true){
00008         cin » failas;
00009         if (!(std::filesystem::exists(failas))){
00010             cout « "Toks failas neegzistuoja, pabandykite vel" « endl;
00011             continue;
00012         }
00013         break;
00014     }
00015
00016     string eil;
00017     Timer t;
00018
00019     std::ifstream df(failas);
```

```
00020     getline(df,eil);
00021
00022     while(getline(df,eil)){
00023         std::istringstream line(eil);
00024         stud temp(line);
00025         temp.calculateGalutinis();
00026         A.push_back(temp);
00027     }
00028     cout « "Perskaityt ir suskaiciuot vidurkius uztruko " « t.elapsed() « endl;
00029     df.close();
00030 }
```

6.11 main.cpp File Reference

```
#include "bibl.h"
```

Functions

- int main ()

Variables

- vector< stud > A

6.11.1 Function Documentation

6.11.1.1 main()

```
int main ()
```

Definition at line 5 of file main.cpp.

6.11.2 Variable Documentation

6.11.2.1 A

```
vector<stud> A
```

Definition at line 3 of file main.cpp.

6.12 main.cpp

Go to the documentation of this file.

```
00001 #include "bibl.h"
00002
00003 vector <stud> A;
00004
00005 int main(){
00006     int input;
00007     string failas;
00008     while ((true)){
00009         cout « "Iveskite skaiciu kokiu budu norite ivesti duomenis " « endl;
00010         cout « "1 - Iveskite visus duomenis rankiniu budu " « endl;
00011         cout « "2 - Iveskite varda ir pavarde rankniu budu " « endl;
00012         cout « "3 - Sugeneruoti visus duomenis automatiskai " « endl;
00013         cout « "4 - Paiimti duomenis is failo " « endl;
00014         cout « "5 - Sugeneruoti nauja duomenu faila " « endl;
00015         cout « "6 - Surusiuoti faila i vargsiukus ir kietiakus " « endl;
00016         cout « "7 - Baigti darba ir spausdinti " « endl;
00017         cout « "8 - Testavimo atvejai " « endl;
00018         try {
00019             if (!(cin»input)||input<1 || input>8){
00020                 cin.clear();
00021                 cin.ignore();
00022                 throw "Ivestas neteisingas simbolis";
00023             }
00024
00025             switch(input){
00026                 case 1:
00027                     rankinis(A);
00028                     break;
00029
00030                 case 2:
00031                     pusrankis(A);
00032                     break;
00033
00034                 case 3:
00035                     int n;
00036                     cout « "Iveskite kiek mokiniu generuoti" « endl;
00037                     cin » n;
00038                     for (int i=0;i<n;i++){
00039                         automatiskas(A);
00040                     }
00041                     break;
00042
00043                 case 4:
00044                     failoNusk(A);
00045                     break;
00046
00047                 case 5:
00048                     failoGen();
00049                     break;
00050
00051                 case 6:
00052                     rusiavimas();
00053                     break;
00054
00055                 case 7:
00056                     cout « "Pagal ka isrusiuoti duomenis?" « endl;
00057                     compare(A);
00058                     spausdina(A);
00059                     return 0;
00060                 case 8:
00061                     test();
00062                     return 0;
00063                 default:
00064                     cout « "Ivedete neteisinga simobli, pabandykit vel! :)" « endl;
00065                     break;
00066         }
00067         }
00068         catch (char const *x){
00069             cout « x « endl;
00070             continue;
00071         }
00072
00073     }
00074  return 0;
00075 }
```

6.13 pusrankis.cpp File Reference

```
#include "bibl.h"
```

Functions

- void pusrankis (vector< stud > &A)

6.13.1 Function Documentation

6.13.1.1 pusrankis()

```
void pusrankis (
            vector< stud > & A)
```

Definition at line 3 of file pusrankis.cpp.

6.14 pusrankis.cpp

Go to the documentation of this file.

```
00001 #include "bibl.h"
00002
00003 void pusrankis(vector <stud> &A){
00004     std::random_device rd;
00005     std::mt19937 mt(rd());
00006     std::uniform_int_distribution <int> dist(0,10);
00007     string vardas;
00008     string pavard;
00009
00010     cout « "Iveskite studento Varda ir pavarde ";
00011     cin » vardas » pavard;
00012     stud temp(vardas,pavard);
00013     for (int i=0;i<dist(mt);i++){
00014         int paz=dist(mt);
00015         temp.addTarpPazymys(paz);
00016     }
00017     temp.setEgzaminas(dist(mt));
00018
00019     temp.calculateGalutinis();
00020     A.push_back(temp);
00021 }
```

6.15 rankinis.cpp File Reference

```
#include "bibl.h"
```

Functions

- void rankinis (vector< stud > &A)

6.15.1 Function Documentation

6.15.1.1 rankinis()

```
void rankinis (
            vector< stud > & A)
```

Definition at line 3 of file rankinis.cpp.

6.16 rankinis.cpp

Go to the documentation of this file.

```
00001 #include "bibl.h"
00002
00003 void rankinis(vector <stud> &A){
00004     string input;
00005     string vardas;
00006     string pavard;
00007     stud temp;
00008     cout « "Iveskite studento Varda ir pavarde ";
00009     cin » temp;
00010     temp.calculateGalutinis();
00011     A.push_back(temp);
00012 }
```

6.17 README.md File Reference

6.18 rusiavimas.cpp File Reference

```
#include "bibl.h"
```

Functions

- void rusiavimas ()

6.18.1 Function Documentation

6.18.1.1 rusiavimas()

```
void rusiavimas ()
```

plz fix kazkodel cia zymiai leciau veikia negu turetu(galimai destructor ir move reik?)

Definition at line 3 of file rusiavimas.cpp.

6.19 rusiavimas.cpp

Go to the documentation of this file.

```
00001 #include "bibl.h"
00002
00003 void rusiavimas(){
00004     string failas;
00005     vector <stud> visi;
00006     vector <stud> nuskriausti;
00007     bool vid;
00008     double visaTrukme=0;
00009     cout « "Iveskite failo pavadinima (pvz. kursiokai.txt)" « endl;
00010     while(true){
00011         cin » failas;
00012         if (!(std::filesystem::exists(failas))){
00013             cout « "Toks failas neegzistuoja, pabandykite vel" « endl;
00014             continue;
00015         }
00016         break;
00017     }
00018
00019     while(true){
00020         cout « "Pagal ka atrinkti studentus?" « endl;
00021         cout « "1 - Pagal pazymiu vidurki " « endl;
00022         cout « "2 - Pagal pazymiu mediana " « endl;
00023         int input;
00024         try {
00025             if (!(cin»input)||input<1 || input>2){
00026                 cin.clear();
00027                 cin.ignore();
00028                 throw "Ivestas neteisingas simbolis";
00029             }
00030             switch(input){
00031                 case 1:
00032                     vid=1;
00033                     break;
00034
00035                 case 2:
00036                     vid=0;
00037                     break;
00038             }
00039         break;
00040         }
00041         catch (char const *x){
00042             cout « x « endl;
00043             continue;
00044         }
00045     }
00046
00047     string eil;
00048     Timer t;
00049     std::ifstream df(failas);
00050     getline(df,eil);
00051
00052     while(getline(df,eil)){
00053         std::istringstream line(eil);
00054         stud temp(line);
00055         temp.calculateGalutinis();
00056         visi.push_back(std::move(temp));
00057
00058     }
00059     df.close();
00060     visaTrukme+=t.elapsed();
00061     cout « "Duomenis nuskaityti uztruko " « visaTrukme « endl;
00062     t.reset();
00064     if (vid==1){
00065         auto it = std::partition(visi.begin(), visi.end(), [](const stud &s)
00066         {
00067             return s.getGalutinisVid() >= 5.0; //
00068         });
00069         nuskriausti.insert(nuskriausti.end(), it, visi.end());
00070         visi.erase(it, visi.end());
00071         visi.shrink_to_fit();
00072     }
00073
00074     else if(vid==0){
00075         auto it = std::partition(visi.begin(), visi.end(), [](const stud &s)
00076         {
00077             return s.getGalutinisMed() >= 5.0; //
00078         });
00079         nuskriausti.insert(nuskriausti.end(), it, visi.end());
00080         visi.erase(it, visi.end());
00081         visi.shrink_to_fit();
00082     }
00083
```

```
00084     visaTrukme+=t.elapsed();
00085     cout « "Mokinius isrusiuoti i atskirus konteinerius uztruko " « t.elapsed() « endl;
00086     double trukme=0;
00087
00088
00089     cout « "Pagal ka isrikiuoti nuskriaustu duomenis?" « endl;
00090     compare(nuskriausti,trukme);
00091     cout « "Pagal ka isrikiuoti kietiaku duomenis?" « endl;
00092     compare(visi,trukme);
00093     visaTrukme+=trukme;
00094     cout « "Duomenis isrikiuoti uztruko " « trukme « endl;
00095
00096     t.reset();
00097     spausdinaFaila(nuskriausti,"nuskriausti "+failas);
00098     spausdinaFaila(visi,"kietiakai "+failas);
00099     visaTrukme+=t.elapsed();
00100     cout « "Duomenis atspausdinti uztruko "« t.elapsed()« endl;
00101     cout « "Isviso uztruko: "« visaTrukme« endl;
00102 }
00103
```

6.20 spausdina.cpp File Reference

```
#include "bibl.h"
```

Functions

- void spausdina (vector< stud > A)
- void spausdinaFaila (vector< stud > &A, string failas)

6.20.1 Function Documentation

6.20.1.1 spausdina()

```
void spausdina (
            vector< stud > A)
```

Definition at line 3 of file spausdina.cpp.

6.20.1.2 spausdinaFaila()

```
void spausdinaFaila (
            vector< stud > & A,
            string failas)
```

Definition at line 12 of file spausdina.cpp.

6.21 spausdina.cpp

Go to the documentation of this file.

```
00001 #include "bibl.h"
00002
00003 void spausdina(vector <stud> A){
00004     cout « "Vardas              Pavarde        Galutinis(vid.) / Galutinis(med.)" « endl;
00005     cout « "-------------------------------------------------------------------" « endl;
00006 for (int i=0;i<A.size();i++){
00007     cout « A[i];
00008 }
00009
00010 }
00011
00012 void spausdinaFaila(vector <stud> &A, string failas){
00013     std::ofstream rf (failas);
00014     rf « "Vardas              Pavarde        Galutinis(vid.) / Galutinis(med.)" « endl;
00015     rf « "-------------------------------------------------------------------" « endl;
00016 for (int i=0;i<A.size();i++){
00017     rf « A[i];
00018
00019 }
```

6.22 std.h File Reference

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <algorithm>
#include <ctime>
#include <random>
#include <stdlib.h>
#include <fstream>
#include <chrono>
#include <sstream>
#include <filesystem>
#include <string>
#include <list>
#include <deque>
```

6.23 std.h

Go to the documentation of this file.

```
00001
00002 # ifndef STD_H
00003 # define STD_H
00004
00005 #include <iostream>
00006 #include <iomanip>
00007 #include <vector>
00008 #include <algorithm>
00009 #include <ctime>
00010 #include <random>
00011 #include <stdlib.h>
00012 #include <fstream>
00013 #include <chrono>
00014 #include <sstream>
00015 #include <filesystem>
00016 #include <string>
00017 #include <list>
00018 #include <deque>
00019
00020
00021 using std::cin;
```

```
00022 using std::cout;
00023 using std::endl;
00024 using std::string;
00025 using std::vector;
00026 using std::setw;
00027 using std::to_string;
00028 using std::setw;
00029
00030 # endif
```

6.24 stud.h File Reference

```
#include "std.h"
#include <numeric>
#include "Zmogus.h"
```

Classes

- class stud

6.25 stud.h

Go to the documentation of this file.

```
00001 #ifndef STUD_H
00002 #define STUD_H
00003
00004 #include "std.h"
00005 #include <numeric>
00006 #include "Zmogus.h"
00007
00008 class stud : public Zmogus {
00009 private:
00010     vector<int> tarp;
00011     double tarpvid = 0;
00012     double tarpmed = 0;
00013     double egz = 0;
00014     double galutinisvid = 0;
00015     double galutinismed = 0;
00016
00017 public:
00018     // Constructors
00019     stud() : Zmogus() {}
00020
00021     stud(const string& vardas, const string& pavarde) : Zmogus(vardas, pavarde) {}
00022
00023     stud(std::istringstream& line) {
00024         int paz;
00025         line » vard » pava;
00026         while (line » paz) {
00027             addTarpPazymys(paz);
00028         }
00029         if (!tarp.empty()) {
00030             egz = tarp.back();
00031             tarp.pop_back();
00032         }
00033     }
00034
00035     // Copy constructor
00036     stud(const stud& kitas) : Zmogus(kitas) {
00037         tarp = kitas.tarp;
00038         egz = kitas.egz;
00039         galutinisvid = kitas.galutinisvid;
00040         galutinismed = kitas.galutinismed;
00041     }
00042
00043     // Move constructor
00044     stud(stud&& kitas) noexcept : Zmogus(std::move(kitas)) {
00045         tarp = std::move(kitas.tarp);
```

```
00046         egz = kitas.egz;
00047         galutinisvid = kitas.galutinisvid;
00048         galutinismed = kitas.galutinismed;
00049 
00050         kitas.egz = 0;
00051         kitas.galutinisvid = 0.0;
00052         kitas.galutinismed = 0.0;
00053     }
00054 
00055     // Destructor
00056     ~stud() {
00057         tarp.clear();
00058         tarp.shrink_to_fit();
00059     }
00060 
00061     // Copy assignment operator
00062     stud& operator=(const stud& kitas) {
00063         if (this != &kitas) {
00064             Zmogus::operator=(kitas); // Call base class assignment operator
00065             tarp = kitas.tarp;
00066             egz = kitas.egz;
00067             galutinisvid = kitas.galutinisvid;
00068             galutinismed = kitas.galutinismed;
00069         }
00070         return *this;
00071     }
00072 
00073     // Move assignment operator
00074     stud& operator=(stud&& kitas) noexcept {
00075         if (this != &kitas) {
00076             Zmogus::operator=(std::move(kitas)); // Call base class move assignment operator
00077             tarp = std::move(kitas.tarp);
00078             egz = kitas.egz;
00079             galutinisvid = kitas.galutinisvid;
00080             galutinismed = kitas.galutinismed;
00081 
00082             kitas.egz = 0;
00083             kitas.galutinisvid = 0.0;
00084             kitas.galutinismed = 0.0;
00085         }
00086         return *this;
00087     }
00088 
00089     // Setter methods
00090     void addTarpPazymys(int paz) {
00091         if (tarp.capacity() == 0) tarp.reserve(10);
00092         tarp.push_back(paz);
00093     }
00094     inline void setEgzaminas(double egzaminas) { egz = egzaminas; }
00095     inline void setVid(double vid) { galutinisvid = vid; }
00096     inline void setMed(double med) { galutinismed = med; }
00097 
00098     // Getter methods
00099     vector<int>& getTarp() { return tarp; }
00100     double getEgz() const { return egz; }
00101     double getGalutinisVid() const { return galutinisvid; }
00102     double getGalutinisMed() const { return galutinismed; }
00103 
00104     // Calculate final grades
00105     void calculateGalutinis() {
00106         if (!tarp.empty()) {
00107             tarpvid = accumulate(tarp.begin(), tarp.end(), 0) / double(tarp.size());
00108             std::sort(tarp.begin(), tarp.end());
00109             if (tarp.size() % 2 == 0) {
00110                 tarpmed = (tarp[tarp.size() / 2 - 1] + tarp[tarp.size() / 2]) / 2.0;
00111             } else {
00112                 tarpmed = tarp[tarp.size() / 2];
00113             }
00114         }
00115         galutinisvid = (tarpvid * 0.4) + (egz * 0.6);
00116         galutinismed = (tarpmed * 0.4) + (egz * 0.6);
00117     }
00118 
00119     // Overloaded operators
00120     friend std::ostream& operator<<(std::ostream& out, const stud& a) {
00121         out << std::left << setw(20) << a.getVardas() << setw(15) << a.getPavarde()
00122             << setw(18) << std::fixed << std::setprecision(2) << a.getGalutinisVid() << " " <<
    a.getGalutinisMed() << endl;
00123         return out;
00124     }
00125 
00126     friend std::istream& operator>>(std::istream& in, stud& a) {
00127         in >> a.vard >> a.pava;
00128         cout << "Veskite studento namu darbo pazymius arba N, kad sustoti ";
00129         string input;
00130         while (true) {
00131             try {
```

```
00132                 in » input;
00133                 if (input == "N" || input == "n") break;
00134                 int paz = std::stoi(input);
00135                 if (paz < 0 || paz > 10) {
00136                     throw "Ivestas neteisingas simbolis";
00137                 }
00138                 a.addTarpPazymys(paz);
00139             } catch (const std::invalid_argument&) {
00140                 cout « "Ivestas neteisingas simbolis" « endl;
00141                 continue;
00142             } catch (const char* x) {
00143                 cout « x « endl;
00144                 continue;
00145             }
00146         }
00147 
00148         cout « "Iveskite studento egzamino rezultata ";
00149         while (true) {
00150             try {
00151                 in » input;
00152                 int egz = std::stoi(input);
00153                 if (egz < 0 || egz > 10) {
00154                     throw "Ivestas neteisingas simbolis";
00155                 }
00156                 a.setEgzaminas(egz);
00157                 break;
00158             } catch (const std::invalid_argument&) {
00159                 cout « "Ivestas neteisingas simbolis" « endl;
00160                 continue;
00161             } catch (const char* x) {
00162                 cout « x « endl;
00163                 continue;
00164             }
00165         }
00166         return in;
00167     }
00168 };
00169 
00170 #endif
```

6.26 test.cpp File Reference

```
#include <cassert>
#include "bibl.h"
```

Functions

- void testDefaultConstructor ()
- void testSettersAndGetters ()
- void testCopyConstructor ()
- void testMoveConstructor ()
- void testCopyAssignment ()
- void testMoveAssignment ()
- void testOutput ()
- void testInput ()
- void test ()

6.26.1 Function Documentation

6.26.1.1 test()

```
void test ()
```

Definition at line 93 of file test.cpp.

6.26.1.2 testCopyAssignment()

```
void testCopyAssignment ()
```

Definition at line 53 of file test.cpp.

6.26.1.3 testCopyConstructor()

```
void testCopyConstructor ()
```

Definition at line 29 of file test.cpp.

6.26.1.4 testDefaultConstructor()

```
void testDefaultConstructor ()
```

Definition at line 4 of file test.cpp.

6.26.1.5 testInput()

```
void testInput ()
```

Definition at line 84 of file test.cpp.

6.26.1.6 testMoveAssignment()

```
void testMoveAssignment ()
```

Definition at line 63 of file test.cpp.

6.26.1.7 testMoveConstructor()

```
void testMoveConstructor ()
```

Definition at line 42 of file test.cpp.

6.26.1.8 testOutput()

```
void testOutput ()
```

Definition at line 73 of file test.cpp.

6.26.1.9 testSettersAndGetters()

```
void testSettersAndGetters ()
```

Definition at line 13 of file test.cpp.

6.27 test.cpp

Go to the documentation of this file.

```
00001 #include <cassert>
00002 #include "bibl.h"
00003
00004 void testDefaultConstructor() {
00005     stud s;
00006     assert(s.getVardas() == "");
00007     assert(s.getPavarde() == "");
00008     assert(s.getGalutinisVid() == 0);
00009     assert(s.getGalutinisMed() == 0);
00010     std::cout « "Default konstruktoriaus testas sekmingas!.\n";
00011 }
00012
00013 void testSettersAndGetters() {
00014     stud s;
00015     s.setvard("Jonas");
00016     s.setpava("Jonaitis");
00017     s.setEgzaminas(9.5);
00018     s.setVid(8.0);
00019     s.setMed(8.5);
00020
00021     assert(s.getVardas() == "Jonas");
00022     assert(s.getPavarde() == "Jonaitis");
00023     assert(s.getEgz() == 9.5);
00024     assert(s.getGalutinisVid() == 8);
00025     assert(s.getGalutinisMed() == 8.5);
00026     std::cout « "Setters ir getters testas sekmingas!\n";
00027 }
00028
00029 void testCopyConstructor() {
00030     stud original;
00031     original.setvard("Petras");
00032     original.setpava("Petraitis");
00033     original.getTarp().push_back(10);
00034     original.setEgzaminas(9);
00035
00036     stud copy(original);
00037     assert(copy.getVardas() == "Petras");
00038     assert(copy.getTarp()[0] == 10);
00039     std::cout « "Copy konstruktoriaus testas sekmingas!\n";
00040 }
00041
00042 void testMoveConstructor() {
00043     stud temp;
00044     temp.setvard("Move");
00045     temp.getTarp().push_back(7);
00046
00047     stud moved(std::move(temp));
00048     assert(moved.getVardas() == "Move");
00049     assert(moved.getTarp()[0] == 7);
00050     std::cout « "Move konstruktoriaus testas sekmingas!\n";
00051 }
00052
00053 void testCopyAssignment() {
00054     stud s1;
00055     s1.setpava("Kebabas");
00056
00057     stud s2;
00058     s2 = s1;
00059     assert(s2.getPavarde() == "Kebabas");
00060     std::cout « "Copy assignment testas sekmingas!\n";
00061 }
00062
00063 void testMoveAssignment() {
00064     stud s1;
00065     s1.setpava("Moved");
00066
00067     stud s2;
00068     s2 = std::move(s1);
00069     assert(s2.getPavarde() == "Moved");
00070     std::cout « "Move assignment testas sekmingas!\n";
00071 }
00072
00073 void testOutput() {
00074     stud original;
00075     original.setvard("Petras");
00076     original.setpava("Petraitis");
00077     original.getTarp().push_back(10);
00078     original.setEgzaminas(9);
00079     original.calculateGalutinis();
00080     cout « original;
00081     std::cout « "Output testas sekmingas!\n";
00082 }
```

```
00083
00084 void testInput() {
00085     stud original;
00086     cout « "iveskite varda ir pavarde\n";
00087     cin » original;
00088     original.calculateGalutinis();
00089     cout « original;
00090     std::cout « "Input testas sekmingas!\n";
00091 }
00092
00093 void test() {
00094     testDefaultConstructor();
00095     testSettersAndGetters();
00096     testCopyConstructor();
00097     testMoveConstructor();
00098     testCopyAssignment();
00099     testMoveAssignment();
00100     testOutput();
00101     //testInput();
00102
00103     std::cout « "\nAll tests passed!\n";
00104 }
```

6.28 Timer.h File Reference

```
#include "std.h"
```

Classes

- class Timer

6.29 Timer.h

Go to the documentation of this file.

```
00001 # ifndef TIMER_H
00002 # define TIMER_H
00003
00004 #include "std.h"
00005
00006 class Timer {
00007     private:
00008       // panaudojame using
00009       using hrClock = std::chrono::high_resolution_clock;
00010       using durationDouble = std::chrono::duration<double>;
00011       std::chrono::time_point<hrClock> start;
00012     public:
00013       Timer() : start{ hrClock::now() } {}
00014       void reset() {
00015         start = hrClock::now();
00016       }
00017       double elapsed() const {
00018         return durationDouble (hrClock::now() - start).count();
00019       }
00020   };
00021
00022   # endif
```

6.30 unit_test.cpp File Reference

```
#include <gtest/gtest.h>
#include "stud.h"
```

Functions

- TEST (StudTest, DefaultConstructor)
- TEST (StudTest, Setters)
- TEST (StudTest, RuleOfFive)
- int main (int argc, char **argv)

6.30.1 Function Documentation

6.30.1.1 main()

```
int main (
            int argc,
            char ** argv)
```

Definition at line 66 of file unit_test.cpp.

6.30.1.2 TEST() [1/3]

```
TEST (
            StudTest ,
            DefaultConstructor )
```

Definition at line 4 of file unit_test.cpp.

6.30.1.3 TEST() [2/3]

```
TEST (
            StudTest ,
            RuleOfFive )
```

Definition at line 26 of file unit_test.cpp.

6.30.1.4 TEST() [3/3]

```
TEST (
            StudTest ,
            Setters )
```

Definition at line 12 of file unit_test.cpp.

6.31 unit_test.cpp

Go to the documentation of this file.

```
00001 #include <gtest/gtest.h>
00002 #include "stud.h"
00003
00004 TEST(StudTest, DefaultConstructor) {
00005     stud s;
00006     EXPECT_EQ(s.getVardas(), "");
00007     EXPECT_EQ(s.getPavarde(), "");
00008     EXPECT_EQ(s.getGalutinisVid(), 0);
00009     EXPECT_EQ(s.getGalutinisMed(), 0);
00010 }
00011
00012 TEST(StudTest, Setters) {
00013     stud s;
00014     s.setvard("Jonas");
00015     s.setpava("Jonaitis");
00016     s.setEgzaminas(9.5);
00017     s.setVid(8.0);
00018     s.setMed(8.5);
00019     EXPECT_EQ(s.getVardas(), "Jonas");
00020     EXPECT_EQ(s.getPavarde(), "Jonaitis");
00021     EXPECT_EQ(s.getEgz(), 9.5);
00022     EXPECT_EQ(s.getGalutinisVid(), 8.0);
00023     EXPECT_EQ(s.getGalutinisMed(), 8.5);
00024 }
00025
00026 TEST(StudTest, RuleOfFive) {
00027     // Set up original object
00028     stud original;
00029     original.setvard("Jonas");
00030     original.setpava("Jonaitis");
00031     original.setEgzaminas(9.5);
00032     original.setVid(8.0);
00033     original.setMed(8.5);
00034
00035     // Copy constructor
00036     stud copyConstructed(original);
00037     EXPECT_EQ(copyConstructed.getVardas(), "Jonas");
00038     EXPECT_EQ(copyConstructed.getPavarde(), "Jonaitis");
00039     EXPECT_EQ(copyConstructed.getEgz(), 9.5);
00040     EXPECT_EQ(copyConstructed.getGalutinisVid(), 8.0);
00041     EXPECT_EQ(copyConstructed.getGalutinisMed(), 8.5);
00042
00043     // Copy assignment
00044     stud copyAssigned;
00045     copyAssigned = original;
00046     EXPECT_EQ(copyAssigned.getVardas(), "Jonas");
00047     EXPECT_EQ(copyAssigned.getPavarde(), "Jonaitis");
00048     EXPECT_EQ(copyAssigned.getEgz(), 9.5);
00049     EXPECT_EQ(copyAssigned.getGalutinisVid(), 8.0);
00050     EXPECT_EQ(copyAssigned.getGalutinisMed(), 8.5);
00051
00052     // Move constructor
00053     stud toMove;
00054     toMove.setvard("Moved");
00055     stud moveConstructed(std::move(toMove));
00056     EXPECT_EQ(moveConstructed.getVardas(), "Moved");
00057
00058     // Move assignment
00059     stud toMoveAssign;
00060     toMoveAssign.setvard("MoveAssign");
00061     stud moveAssigned;
00062     moveAssigned = std::move(toMoveAssign);
00063     EXPECT_EQ(moveAssigned.getVardas(), "MoveAssign");
00064 }
00065
00066 int main(int argc, char **argv) {
00067     ::testing::InitGoogleTest(&argc, argv);
00068     return RUN_ALL_TESTS();
00069 }
```

6.32 Zmogus.h File Reference

```
#include "std.h"
#include <numeric>
```

Classes

- class Zmogus

6.33 Zmogus.h

Go to the documentation of this file.

```
00001 #ifndef ZMOGUS_H
00002 #define ZMOGUS_H
00003
00004 #include "std.h"
00005 #include <numeric>
00006
00007 class Zmogus {
00008 public:
00009     // Default constructor
00010     Zmogus() = default;
00011
00012     Zmogus(const string& vard, const string& pava) : vard(vard), pava(pava) {}
00013
00014     // Virtual destructor
00015     virtual ~Zmogus() = default;
00016
00017     // Copy constructor
00018     Zmogus(const Zmogus& other)
00019         : vard(other.vard), pava(other.pava) {}
00020
00021     // Copy assignment operator
00022     Zmogus& operator=(const Zmogus& other) {
00023         if (this != &other) {
00024             vard = other.vard;
00025             pava = other.pava;
00026         }
00027         return *this;
00028     }
00029
00030     // Move constructor
00031     Zmogus(Zmogus&& other) noexcept
00032         : vard(std::move(other.vard)), pava(std::move(other.pava)) {}
00033
00034     // Move assignment operator
00035     Zmogus& operator=(Zmogus&& other) noexcept {
00036         if (this != &other) {
00037             vard = std::move(other.vard);
00038             pava = std::move(other.pava);
00039         }
00040         return *this;
00041     }
00042
00043     inline void setvard(const string& vardas) { vard = vardas; }
00044     inline void setpava(const string& pavard) { pava = pavard; }
00045
00046     inline string getVardas() const { return vard; }
00047     inline string getPavarde() const { return pava; }
00048
00049     // Kad clase butu abstrakti, reikia padaryti virtual function
00050     virtual void addTarpPazymys(int paz) = 0;
00051 protected:
00052
00053     string vard;
00054     string pava;
00055 };
00056
00057 #endif // ZMOGUS_H
```

Index