Studentų galutinio balo skaičiuoklė

Generated by Doxygen 1.13.2

Chapter 1

Studentų galutinio balo skaičiavimo programa

Šis projektas yra C++ programa, kuri apskaičiuoja galutinį studento balą pagal jų namų darbų, bei egzamino įvertinimus.

1.1 Projekto paleidimas naudojant CMake

1.1.0.1 1. Reikalingi įrankiai

Prieš paleidžiant projektą, įsitikinkite, kad turite šiuos įrankius:

- **CMake**: Atsisiųsti CMake (minimum v3.10)
- **C++ kompiliatorius** (GCC, CLANG, MSVC)

1.1.0.2 2. Parsisiųskite projektą, jei jo dar neturite

1.1.0.2.1 Projekto klonavimas iš git:

```
git clone https://github.com/nupustas/oop.vp
```

Paklonave projektą, atidarykite jo aplanką.

1.1.0.2.2 Projekto kompiliavimas:

```
mkdir build
cd build
cmake ..
cmake --build . --config Release
```

1.1.0.2.3 Projekto paleidimas:

```
cd release
OOP.exe
```

1.2 Projekto struktūra:

- **`include/`**: Aplankalas, kuriame laikomi projekto header failai.
- **`src/`**: Pagrindinis programos kodas.
- **`CMakeLists.txt`**: CMake instrukcijos kompiliavimui.
- **`ReadME.md`**: Programos instrukcija.

1.3 V3.0

1.3.1 Vector klasės funkcijų pavyzdžiai

1.3.1.1 1. `void erase(size_t index)`

Ši funkcija pašalina vieną konteinerio elementą, kurio index yra paduotas.

1.3.1.1.0.1 Veikimas:

Iš pradžių patikrinama ar paduotas **index** nėra didesnis nei esamo konteinerio dydis, jei ne, visi elementai esantys dešinėje index'o pastumiami į kairę ir pakeičiamas konteinerio dydis.

1.3.1.2 2. `V* erase(V* first, V* last)`

Ši funkcija, pašalina konteinerio elementus nuo į ja paduotų index: nuo ** V* first ** iki ** V* last **

1.3.1.2.0.1 Veikimas:

Patikrinama ar first nėra mažesnis už masyvo pradžią, ar last nėra didesnis už masyvo pabaigą, ir ar first $<$ last. Tada pointeriai first ir last paverčiami į index'us, ir elementai kurie yra už 'last' yra pastumiami į kairę. Pakeičiamas konteinerio 'dydis' ir grąžinamas pirmo elemento po ištrintų iteratorius.

1.3.1.3 3. `V& operator[](size_t index)`

Operatorius [], leidžia prieti prie norimo elemento konteineryje. Norint pasiekti bet kokį elementą konteineryje, naudojamas šis operatorius per norimo elemento index'ą. `Pvz.: Vector[i]`

1.3.1.3.0.1 Veikimas:

Patikrinama ar index nėra didesnis už konteinerio dydį, jei ne grąžinamas duomenų masyvo elementas. `return duom[index];`

1.3.1.4 4. bool operator==(const Vektor<V>& other)

Operatorius ==, šis leidžia patikrinti ar du konteineriai yra lygūs.

1.3.1.4.0.1 Veikimas:

Visų pirmą patikrinama ar konteinerių dydis lygus, jei ne iškart grąžinama **false** reikšmė. Jei konteinerių dydis toks pat, tuomet tikrinama ar kiekvinas konteinerių elementas lygus. Jei visi elementai lygūs, grąžinama **true** reikšmė.

1.3.1.5 5. void pop_back()

Ši funkcija leidžia pašalinti konteinerio paskutinį elementą.

1.3.1.5.0.1 Veikimas:

Patikrinama ar konteineris nėra tusčias, jei ne, tuomet jo dydis yra pamažinamas vienu.
`--dydis;`

1.4 Testavimas

1.4.0.1 Šiose lentelėse pateikiami skirtingų C++ konteinerių (vector, list, deque) testavimo rezultatai.

1.4.1 Testavimo sistemos parametrai:

- **Procesorius: Intel Core i5-10300H**
- **Operatyvioji atmintis: 2×4GB DDR4 3200MHz**

1.5 -Diskas: 512GB NVMe SSD

1.5.0.1 Originalus Vector vs Vektor klasė

- **Užpildymas naudojant push_back() funkciją**

Dydis	Vector	Klasė
10000	0.00010 s	0.00006 s
100000	0.00101 s	0.00051 s
1000000	0.00414 s	0.00477 s
10000000	0.04605 s	0.04761 s
100000000	0.50926 s	0.39100 s
1000000000	9.06876 s	6.91821 s

- **Programos spartos analizė**

Failas	Vector	Klasė
Studentai10000	0.17188 s	0.13205 s
Studentai100000	1.03871 s	1.09587 s
Studentai1000000	9.88531 s	10.2013 s
Studentai10000000	100.456 s	110.858 s

1.5.0.2 Testų analizė

- Užpildant konteinerį naudojant push_back() funkciją, vektoriaus klasė yra spartesnė
- Naudojant programoje, klasė yra šiek tiek pranašesnė apdorojant mažus studentų kiekius, tačiau kai studentų daug originalus vektorius vistiek veikia sparčiau.

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:

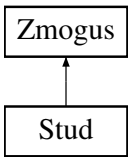
Chapter 5

Class Documentation

5.1 Stud Class Reference

`#include <student.h>`

Inheritance diagram for Stud:



Public Member Functions

- Stud ()
- Stud (const std::string &v, const std::string &p, const std::vector< int > &pazymiai, int e, char vmod, double gal)
- ~Stud ()
- Stud (const Stud &other)
- Stud & operator= (const Stud &other)
- Stud (Stud &&other)
- Stud & operator= (Stud &&other)
- void setEgz (int e)
- void setVm (char v)
- void setGalutinis (double g)
- void addPaz (int pazymys)
- int getEgz () const
- char getVm () const
- double getGalutinis () const
- std::vector< int > getPaz () const
- void removeLastPaz ()
- void FinalScore ()
- void print () const override

Public Member Functions inherited from Zmogus

- Zmogus ()
- Zmogus (const string &v, const string &p)
- virtual ~Zmogus ()=default
- string getVardas () const
- string getPavarde () const
- void setVardas (const string &v)
- void setPavarde (const string &p)

Friends

- std::istream & operator>> (std::istream &in, Stud &s)
- std::ostream & operator<< (std::ostream &out, const Stud &s)

Additional Inherited Members

Protected Attributes inherited from Zmogus

- string Vardas
- string Pavarde

5.1.1 Detailed Description

Definition at line 7 of file student.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Stud() [1/4]

```
Stud::Stud ()  [inline]
```

Definition at line 16 of file student.h.

5.1.2.2 Stud() [2/4]

```
Stud::Stud (
            const std::string & v,
            const std::string & p,
            const std::vector< int > & pazymiai,
            int e,
            char vmod,
            double gal)  [inline]
```

Definition at line 17 of file student.h.

5.1.2.3 ~Stud()

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

Definition at line 21 of file student.h.

5.1.2.4 Stud() [3/4]

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

Definition at line 24 of file student.h.

5.1.2.5 Stud() [4/4]

```
Stud::Stud (
            Stud && other)  [inline]
```

Definition at line 40 of file student.h.

5.1.3 Member Function Documentation

5.1.3.1 addPaz()

```
void Stud::addPaz (
            int pazymys)  [inline]
```

Definition at line 109 of file student.h.

5.1.3.2 FinalScore()

```
void Stud::FinalScore ()  [inline]
```

Definition at line 118 of file student.h.

5.1.3.3 getEgz()

```
int Stud::getEgz () const  [inline]
```

Definition at line 111 of file student.h.

5.1.3.4 getGalutinis()

```
double Stud::getGalutinis () const  [inline]
```

Definition at line 113 of file student.h.

5.1.3.5 getPaz()

```
std::vector< int > Stud::getPaz () const  [inline]
```

Definition at line 114 of file student.h.

5.1.3.6 getVm()

```
char Stud::getVm () const  [inline]
```

Definition at line 112 of file student.h.

5.1.3.7 operator=() [1/2]

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

Definition at line 28 of file student.h.

5.1.3.8 operator=() [2/2]

```
Stud & Stud::operator= (
            Stud && other)  [inline]
```

Definition at line 49 of file student.h.

5.1.3.9 print()

```
void Stud::print () const  [inline], [override], [virtual]
```

Implements Zmogus.

Definition at line 134 of file student.h.

5.1.3.10 removeLastPaz()

```
void Stud::removeLastPaz ()  [inline]
```

Definition at line 115 of file student.h.

5.1.3.11 setEgz()

```
void Stud::setEgz (
            int e)  [inline]
```

Definition at line 106 of file student.h.

5.1.3.12 setGalutinis()

```
void Stud::setGalutinis (
            double g)  [inline]
```

Definition at line 108 of file student.h.

5.1.3.13 setVm()

```
void Stud::setVm (
            char v)  [inline]
```

Definition at line 107 of file student.h.

5.1.4 Friends And Related Symbol Documentation

5.1.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & out,
            const Stud & s)  [friend]
```

Definition at line 93 of file student.h.

5.1.4.2 operator>>

```
std::istream & operator>> (
            std::istream & in,
            Stud & s)  [friend]
```

Definition at line 64 of file student.h.

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

- include/student.h

5.2 Vektor< V > Class Template Reference

```
#include <vector.h>
```

Public Types

- using value_type = V
- using reference = V&
- using const_reference = const V&
- using iterator = V*
- using const_iterator = const V*
- using size_type = size_t

Public Member Functions

- Vektor ()
- Vektor (size_t d)
- Vektor (size_t d, const V &value)
- ~Vektor ()
- size_t size () const
- size_t max_size () const
- size_t capacity () const
- bool empty () const
- void reserve (size_t n)
- void erase (size_t index)
- V * erase (V *first, V *last)
- void swap (Vektor< V > &other)
- void shrink_to_fit ()
- void push_back (const V &value)
- void pop_back ()
- V * begin ()
- V * end ()
- V & front ()
- V & back ()
- V * clear ()
- V & operator[ ] (size_t index)
- Vektor (const Vektor< V > &other)
- Vektor< V > & operator= (const Vektor< V > &other)
- Vektor (Vektor< V > &&other) noexcept
- Vektor< V > & operator= (Vektor< V > &&other) noexcept
- bool operator== (const Vektor< V > &other) const

5.2.1 Detailed Description

template<typename V>
class Vektor< V >

Definition at line 5 of file vector.h.

5.2.2 Member Typedef Documentation

5.2.2.1 const_iterator

```
template<typename V>
using Vektor< V >::const_iterator = const V*
```

Definition at line 28 of file vector.h.

5.2.2.2 const_reference

```
template<typename V>
using Vektor< V >::const_reference = const V&
```

Definition at line 26 of file vector.h.

5.2.2.3 iterator

```
template<typename V>
using Vektor< V >::iterator = V*
```

Definition at line 27 of file vector.h.

5.2.2.4 reference

```
template<typename V>
using Vektor< V >::reference = V&
```

Definition at line 25 of file vector.h.

5.2.2.5 size_type

```
template<typename V>
using Vektor< V >::size_type = size_t
```

Definition at line 29 of file vector.h.

5.2.2.6 value_type

```
template<typename V>
using Vektor< V >::value_type = V
```

Definition at line 24 of file vector.h.

5.2.3 Constructor & Destructor Documentation

5.2.3.1 Vektor() [1/5]

```
template<typename V>
Vektor< V >::Vektor ()  [inline]
```

Definition at line 34 of file vector.h.

5.2.3.2 Vektor() [2/5]

```
template<typename V>
Vektor< V >::Vektor (
            size_t d)  [inline]
```

Definition at line 36 of file vector.h.

5.2.3.3 Vektor() [3/5]

```
template<typename V>
Vektor< V >::Vektor (
            size_t d,
            const V & value)  [inline]
```

Definition at line 39 of file vector.h.

5.2.3.4 ~Vektor()

```
template<typename V>
Vektor< V >::~Vektor ()  [inline]
```

Definition at line 45 of file vector.h.

5.2.3.5 Vektor() [4/5]

```
template<typename V>
Vektor< V >::Vektor (
            const Vektor< V > & other)  [inline]
```

Definition at line 134 of file vector.h.

5.2.3.6 Vektor() [5/5]

```
template<typename V>
Vektor< V >::Vektor (
            Vektor< V > && other)  [inline], [noexcept]
```

Definition at line 154 of file vector.h.

5.2.4 Member Function Documentation

5.2.4.1 back()

```
template<typename V>
V & Vektor< V >::back ()  [inline]
```

Definition at line 117 of file vector.h.

5.2.4.2 begin()

```
template<typename V>
V * Vektor< V >::begin ()  [inline]
```

Definition at line 114 of file vector.h.

5.2.4.3 capacity()

```
template<typename V>
size_t Vektor< V >::capacity () const  [inline]
```

Definition at line 51 of file vector.h.

5.2.4.4 clear()

```
template<typename V>
V * Vektor< V >::clear ()  [inline]
```

Definition at line 118 of file vector.h.

5.2.4.5 empty()

```
template<typename V>
bool Vektor< V >::empty () const  [inline]
```

Definition at line 52 of file vector.h.

5.2.4.6 end()

```
template<typename V>
V * Vektor< V >::end ()  [inline]
```

Definition at line 115 of file vector.h.

5.2.4.7 erase() [1/2]

```
template<typename V>
void Vektor< V >::erase (
            size_t index)  [inline]
```

Definition at line 58 of file vector.h.

5.2.4.8 erase() [2/2]

```
template<typename V>
V * Vektor< V >::erase (
            V * first,
            V * last)  [inline]
```

Definition at line 65 of file vector.h.

5.2.4.9 front()

```
template<typename V>
V & Vektor< V >::front ()  [inline]
```

Definition at line 116 of file vector.h.

5.2.4.10 max_size()

```
template<typename V>
size_t Vektor< V >::max_size () const  [inline]
```

Definition at line 50 of file vector.h.

5.2.4.11 operator=() [1/2]

```
template<typename V>
Vektor< V > & Vektor< V >::operator= (
            const Vektor< V > & other)  [inline]
```

Definition at line 141 of file vector.h.

5.2.4.12 operator=() [2/2]

```
template<typename V>
Vektor< V > & Vektor< V >::operator= (
            Vektor< V > && other)  [inline], [noexcept]
```

Definition at line 162 of file vector.h.

5.2.4.13 operator==()

```
template<typename V>
bool Vektor< V >::operator== (
            const Vektor< V > & other) const  [inline]
```

Definition at line 175 of file vector.h.

5.2.4.14 operator[]()

```
template<typename V>
V & Vektor< V >::operator[] (
            size_t index)  [inline]
```

Definition at line 129 of file vector.h.

5.2.4.15 pop_back()

```
template<typename V>
void Vektor< V >::pop_back ()  [inline]
```

Definition at line 108 of file vector.h.

5.2.4.16 push_back()

```
template<typename V>
void Vektor< V >::push_back (
            const V & value)  [inline]
```

Definition at line 96 of file vector.h.

5.2.4.17 reserve()

```
template<typename V>
void Vektor< V >::reserve (
            size_t n)  [inline]
```

Definition at line 53 of file vector.h.

5.2.4.18 shrink_to_fit()

```
template<typename V>
void Vektor< V >::shrink_to_fit ()  [inline]
```

Definition at line 85 of file vector.h.

5.2.4.19 size()

```
template<typename V>
size_t Vektor< V >::size () const  [inline]
```

Definition at line 49 of file vector.h.

5.2.4.20 swap()

```
template<typename V>
void Vektor< V >::swap (
            Vektor< V > & other)  [inline]
```

Definition at line 80 of file vector.h.

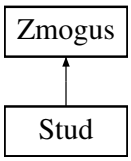
The documentation for this class was generated from the following file:

- include/vector.h

5.3 Zmogus Class Reference

`#include <human.h>`

Inheritance diagram for Zmogus:



Public Member Functions

- Zmogus ()
- Zmogus (const string &v, const string &p)
- virtual ~Zmogus ()=default
- string getVardas () const
- string getPavarde () const
- void setVardas (const string &v)
- void setPavarde (const string &p)
- virtual void print () const =0

Protected Attributes

- string Vardas
- string Pavarde

5.3.1 Detailed Description

Definition at line 4 of file human.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Zmogus() [1/2]

```
Zmogus::Zmogus ()  [inline]
```

Definition at line 9 of file human.h.

5.3.2.2 Zmogus() [2/2]

```
Zmogus::Zmogus (
            const string & v,
            const string & p)  [inline]
```

Definition at line 10 of file human.h.

5.3.2.3 ~Zmogus()

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

5.3.3 Member Function Documentation

5.3.3.1 getPavarde()

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

Definition at line 15 of file human.h.

5.3.3.2 getVardas()

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

Definition at line 14 of file human.h.

5.3.3.3 print()

```
virtual void Zmogus::print () const  [pure virtual]
```

Implemented in Stud.

5.3.3.4 setPavarde()

```
void Zmogus::setPavarde (
            const string & p)  [inline]
```

Definition at line 18 of file human.h.

5.3.3.5 setVardas()

```
void Zmogus::setVardas (
            const string & v)  [inline]
```

Definition at line 17 of file human.h.

5.3.4 Member Data Documentation

5.3.4.1 Pavarde

```
string Zmogus::Pavarde  [protected]
```

Definition at line 6 of file human.h.

5.3.4.2 Vardas

```
string Zmogus::Vardas  [protected]
```

Definition at line 6 of file human.h.

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

- include/human.h

Chapter 6

File Documentation

6.1 include/functions.h File Reference

```
#include "manolib.h"
#include "student.h"
```

Functions

- void TestStud ()
- template<typename Container>
  Container GenerateEverything ()
- template<typename Container>
  Container GenerateScores ()
- template<typename Container>
  Container ManualInput ()
- template<typename Container>
  Container ReadFile (string filename)
- template<typename Container>
  void Sorting (Container &grupe)
- template<typename Container>
  void OutputToTerminal (Container &grupe)
- template<typename Container>
  void OutputToFile (Container &grupe)
- string GenerateFile (int StudentCount)
- template<typename Container>
  Container SpeedTesting ()
- template<typename Container>
  void SplitFile (Container &grupe)
- template<typename Container>
  void FinalScore (Container &grupe)

6.1.1 Function Documentation

6.1.1.1 FinalScore()

```
template<typename Container>
void FinalScore (
            Container & grupe)
```

Definition at line 436 of file functions.h.

6.1.1.2 GenerateEverything()

```
template<typename Container>
Container GenerateEverything ()
```

Definition at line 56 of file functions.h.

6.1.1.3 GenerateFile()

```
string GenerateFile (
            int StudentCount)
```

Definition at line 275 of file functions.h.

6.1.1.4 GenerateScores()

```
template<typename Container>
Container GenerateScores ()
```

Definition at line 100 of file functions.h.

6.1.1.5 ManualInput()

```
template<typename Container>
Container ManualInput ()
```

Definition at line 140 of file functions.h.

6.1.1.6 OutputToFile()

```
template<typename Container>
void OutputToFile (
            Container & grupe)
```

Definition at line 259 of file functions.h.

6.1.1.7 OutputToTerminal()

```
template<typename Container>
void OutputToTerminal (
            Container & grupe)
```

Definition at line 246 of file functions.h.

6.1.1.8 ReadFile()

```
template<typename Container>
Container ReadFile (
            string filename)
```

Definition at line 170 of file functions.h.

6.1.1.9 Sorting()

```
template<typename Container>
void Sorting (
            Container & grupe)
```

Definition at line 212 of file functions.h.

6.1.1.10 SpeedTesting()

```
template<typename Container>
Container SpeedTesting ()
```

Definition at line 317 of file functions.h.

6.1.1.11 SplitFile()

```
template<typename Container>
void SplitFile (
            Container & grupe)
```

Definition at line 373 of file functions.h.

6.1.1.12 TestStud()

```
void TestStud ()
```

Definition at line 8 of file functions.h.

6.2 functions.h

Go to the documentation of this file.

```
00001 #ifndef FUNCTIONS_H
00002 #define FUNCTIONS_H
00003
00004 #include "manolib.h"
00005 #include "student.h"
00006
00007 // Klasės testavimas
00008 void TestStud() {
00009     cout « "Student klases testavimas:" «endl;
00010     // TEST COPY CONSTRUCTOR
00011     cout « "Sukuriamas student1" «endl;
00012     Stud student1("Jonas", "Jonaitis", {10, 9, 8,8,10,9}, 8, 'a', 9.0);
00013
00014     cout « "\n TEST COPY CONSTRUCTOR" « endl;
00015     Stud student2(student1);
00016
00017     cout « "Original student: \n " « student1 ;
00018     cout « "Copied student: \n " « student2 « endl;
00019
00020     //COPY ASSIGNMENT OPERATOR
00021     cout « "\n TEST COPY ASSIGNMENT OPERATOR" « endl;
00022     Stud student3;
00023     student3 = student1;
00024
00025     cout « "Assigned student:\n  " « student3 « endl;
00026
00027     //MOVE CONSTRUCTOR
00028     cout « "TEST MOVE CONSTRUCTOR" « endl;
00029     Stud student4(std::move(student1));
00030
00031     cout « "Moved student: \n " « student4;
00032     cout « "Original student: \n " « student1 « endl;
00033
00034     //MOVE ASSIGNMENT OPERATOR
00035     cout « "\n TEST MOVE ASSIGNMENT OPERATOR" « endl;
00036     Stud student5;
00037     student5 = std::move(student2);
00038
00039     cout « "Moved-assigned student: \n " « student5 « endl;
00040     cout « "Original student: \n " « student2 « endl;
00041
00042     //INPUT OPERATOR
00043     cout « "\n TEST INPUT OPERATOR" « endl;
00044     Stud student6;
00045     cin » student6;
00046
00047     cout « "Entered student: " « student6 « endl;
00048
00049     //OUTPUT OPERATOR
00050     cout « "\n TEST OUTPUT OPERATOR" « endl;
00051     cout « "Final output of student:\n  " « student6 « endl;
00052 }
00053
00054 // Visko generavimas
00055 template <typename Container>
00056 Container GenerateEverything() {
00057     Container grupe;
00058     cout « "Selected '3-Generate everything' " « endl;
00059     cout « endl;
00060     int n, x;
00061     cout « "How many students do you want to generate? ";
00062     while (!(cin » n) || n < 1) {
00063         cout « "Invalid input. Please enter a positive number: ";
00064         cin.clear();
00065         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00066     }
00067
00068     cout « "How many homework scores do you want to generate? ";
00069     while (!(cin » x) || x < 1) {
00070         cout « "Invalid input. Please enter a positive number: ";
00071         cin.clear();
00072         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00073     }
00074
00075     for (int i = 0; i < n; i++) {
00076         Stud laik;
00077         int gender = rand() % 2;
00078         if (gender == 0) {
00079             laik.setVardas(FNames[rand() % 25]);
00080             laik.setPavarde(FSurnames[rand() % 25]);
00081
00082         } else {
```

```
00083             laik.setVardas(MNames[rand() % 25]);
00084             laik.setPavarde(MSurnames[rand() % 25]);
00085         }
00086
00087         for (int j = 0; j < x; j++) {
00088             laik.addPaz(rand() % 10);
00089
00090         }
00091         laik.setEgz(rand() % 10);
00092
00093         grupe.push_back(laik);
00094     }
00095     return grupe;
00096 }
00097
00098 // Vardo ivedimas, pazymiu generavimas
00099 template <typename Container>
00100 Container GenerateScores() {
00101     cout « "Selected 2-Input names, generate scores" « endl;
00102     cout « endl;
00103     Container grupe;
00104
00105     while (true) {
00106         Stud laik;
00107         string Vardas, Pavarde;
00108         cout « "Input name: ";
00109         cin » Vardas;
00110         laik.setVardas(Vardas);
00111         cout « "Input surname: ";
00112         cin » Pavarde;
00113         laik.setPavarde(Pavarde);
00114
00115         cout « "How many homework scores do you want to generate? ";
00116         int n;
00117         cin » n;
00118
00119         for (int i = 0; i < n; i++) {
00120             laik.addPaz((rand() % 10));
00121         }
00122         laik.setEgz((rand() % 10));
00123
00124         grupe.push_back(laik);
00125
00126         cout « "Enter more students? (y/n) ";
00127         char x;
00128         cin » x;
00129         while (x != 'y' && x != 'n') {
00130             cout « "Invalid input. Enter y or n" « endl;
00131             cin » x;
00132         }
00133         if (x == 'n') break;
00134     }
00135     return grupe;
00136 }
00137
00138 // Visko ivedimas ranka
00139 template <typename Container>
00140 Container ManualInput() {
00141     Container grupe;
00142     std::cout « "Manual student input selected.\n" « std::endl;
00143
00144     while (true) {
00145         Stud laik;
00146
00147         // Naudojamas » klasės operatorius
00148         std::cin » laik;
00149
00150         grupe.push_back(laik);
00151
00152         char more;
00153         std::cout « "Add another student? (y/n): ";
00154         std::cin » more;
00155         while (more != 'y' && more != 'n') {
00156             std::cout « "Invalid input. Enter y or n: ";
00157             std::cin » more;
00158         }
00159         if (more == 'n') break;
00160
00161         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Clear leftover input
00162         std::cout « std::endl;
00163     }
00164
00165     return grupe;
00166 }
00167
00168 //Skaitymas is failo
00169 template <typename Container>
```

```
00170 Container ReadFile(string filename) {
00171     Container grupe;
00172 
00173     ifstream fd(filename);
00174     while (!fd) {
00175         cerr « "File not found!" « endl;
00176         cout « "Enter existing file name: ";
00177         cin » filename;
00178         fd.open(filename);
00179     }
00180 
00181     string line;
00182     getline(fd, line); // Skip first line
00183 
00184     while (getline(fd, line)) {
00185         istringstream iss(line);
00186         Stud laik;
00187         string vardas, pavarde;
00188         iss » vardas » pavarde;
00189         laik.setVardas(vardas);
00190         laik.setPavarde(pavarde);
00191         int num;
00192 
00193         while (iss » num) {
00194             laik.addPaz(num);
00195         }
00196 
00197         vector<int> pazymiai = laik.getPaz();
00198         if (!pazymiai.empty()) {
00199             laik.setEgz(pazymiai.back());
00200             laik.removeLastPaz();
00201         }
00202 
00203         grupe.push_back(laik);
00204     }
00205 
00206     fd.close();
00207     return grupe;
00208 }
00209 
00210 //Rusiavimas
00211 template <typename Container>
00212 void Sorting(Container &grupe) {
00213     cout « "How do you want to sort the students?" « endl;
00214     cout « "1 - By name" « endl;
00215     cout « "2 - By surname" « endl;
00216     cout « "3 - By final score descending" « endl;
00217     cout « "4 - By final score ascending" « endl;
00218 
00219     char x= '3';
00220     //cin » x;
00221     while (x != '1' && x != '2' && x != '3' && x != '4') {
00222         cout « "Invalid input. Enter 1, 2, 3, or 4: ";
00223         cin » x;
00224     }
00225     auto chrono_start = std::chrono::high_resolution_clock::now();
00226 
00227     auto comparator = [&](const Stud &a, const Stud &b) {
00228         if (x == '1') return a.getVardas() < b.getVardas();
00229         if (x == '2') return a.getPavarde() < b.getPavarde();
00230         if (x == '3') return a.getGalutinis() < b.getGalutinis();
00231         else return a.getGalutinis() > b.getGalutinis();
00232     };
00233     // constexpr apskaiciuoja kompiliavimo metu, o ne runtime metu
00234     if constexpr (is_same_v<Container, list<Stud»>) {
00235         grupe.sort(comparator);  // listo sortas
00236             } else {
00237         sort(grupe.begin(), grupe.end(), comparator);  // std::sort vectoriui ir deque
00238     }
00239     auto chrono_end = std::chrono::high_resolution_clock::now();
00240     std::chrono::duration<double> duration = chrono_end - chrono_start;
00241     cout « "SORTING TOOK: " « fixed « setprecision(5) « duration.count() « " s" « endl;
00242 }
00243 
00244 // Templated function to output results
00245 template <typename Container>
00246 void OutputToTerminal(Container &grupe) {
00247     cout « left « setw(15) « "Vardas" « setw(15) « "Pavarde"
00248          « setw(15) « "Galutinis (Vid.)"
00249          « " / "
00250          « "Galutinis (Med.)" « endl;
00251     cout « "-------------------------------------------------------------" « endl;
00252     for (const auto &n : grupe) {
00253         // Output naudojant « klasės operatoriu
00254         std::cout « n;
00255     }
00256 }
```

```
00257
00258 template <typename Container>
00259 void OutputToFile(Container& grupe)
00260 {
00261     ofstream out("rezultatai.txt");
00262     out«std::left«setw(15)«"Vardas"«setw(15)«"Pavarde"
00263     «setw(15)«"Galutinis (Vid.)"«" / "«"Galutinis (Med.)"«endl;
00264     out«"-------------------------------------------------------------"«endl;
00265 for(auto n :grupe)
00266   {
00267     out«std::left«setw(15)«n.getVardas()«setw(18)«n.getPavarde()«setw(7);
00268     if(n.getVm() == 'a') out«std::fixed«std::setprecision(2)«n.getGalutinis()«"            -"«endl;
00269     else out«" -                "«std::fixed«std::setprecision(2)«n.getGalutinis()«endl;
00270   }
00271 out.close();
00272
00273 }
00274
00275 string GenerateFile(int StudentCount)
00276 {
00277     string filename = "Studentai"+std::to_string(StudentCount)+".txt";
00278     ifstream fd(filename);
00279     if(fd.good())
00280     {
00281         cout«filename«" already exists"«endl;
00282         return filename;
00283     }
00284     fd.close();
00285
00286     auto start = std::chrono::high_resolution_clock::now();
00287     ofstream fr(filename);
00288     if(!fr)
00289     {
00290         cout«"Error creating file" «filename«endl;
00291     }
00292
00293     fr«std::left«setw(16)«"Vardas Pavarde "«std::left«setw(20)«"Pazymiai   "«"Egzaminas"«endl;
00294     for(int i=0; i<StudentCount; i++)
00295     {
00296         if(rand()%2==0)
00297         {
00298             fr«MNames[rand()%25]«" "«MSurnames[rand()%25]«" ";
00299         }
00300         else
00301         {
00302             fr«FNames[rand()%25]«" "«FSurnames[rand()%25]«" ";
00303         }
00304         for(int j=0; j<10; j++)
00305         {
00306             fr«rand()%10«" ";
00307         }
00308         fr«rand()%10«endl;
00309     }
00310     auto end = std::chrono::high_resolution_clock::now();
00311     std::chrono::duration<double> duration = end - start;
00312     cout«filename «" sukurtas per "«fixed«setprecision(5)«duration.count() « " s" « endl;
00313     return filename;
00314 }
00315
00316 template <typename Container>
00317 Container SpeedTesting()
00318 {
00319     Container grupe;
00320     string filename;
00321
00322     cout « "Ar norite generuoti faila? (y/n): ";
00323     char choice;
00324     cin » choice;
00325
00326     if (choice == 'y' || choice == 'Y')
00327     {
00328         int StudentCount;
00329         cout « "Enter the number of students: ";
00330         cin » StudentCount;
00331
00332         filename = GenerateFile(StudentCount);
00333     }
00334
00335     if (filename.empty()) // If filename is still empty, ask for input
00336     {
00337         cout « "Iveskite testo faila: ";
00338         cin » filename;
00339     }
00340
00341     cout « "Chosen file: " « filename « endl;
00342
00343     auto startRead = std::chrono::high_resolution_clock::now(); // Timer for file reading
```

```
00344     grupe = ReadFile<Container>(filename);
00345     auto endRead = std::chrono::high_resolution_clock::now();
00346 
00347     FinalScore(grupe);
00348 
00349     auto startSort = std::chrono::high_resolution_clock::now(); // Timer for sorting
00350     Sorting(grupe);
00351     auto endSort = std::chrono::high_resolution_clock::now();
00352 
00353     auto startSplit = std::chrono::high_resolution_clock::now();
00354      SplitFile(grupe);
00355     auto endSplit = std::chrono::high_resolution_clock::now();
00356 
00357     // Calculate and display durations
00358     std::chrono::duration<double> durationRead = endRead - startRead;
00359     std::chrono::duration<double> durationSort = endSort - startSort;
00360     std::chrono::duration<double> durationSplit = endSplit - startSplit;
00361 
00362     cout « filename « " failo nuskaitymo laikas: " « fixed « setprecision(5) « durationRead.count() « " s" « endl;
00363     cout « filename « " failo rusiavimas: " « fixed « setprecision(5) « durationSort.count() « " s" « endl;
00364     cout « filename « " failo paskirstymo ir irasymo laikas: " « fixed « setprecision(5) « durationSplit.count() « " s" « endl;
00365     cout « filename « " is viso uztruko: " « fixed « setprecision(5)
00366          « (durationRead.count() + durationSort.count() + durationSplit.count()) « " s" « endl;
00367 
00368     return grupe;
00369 }
00370 
00371 //Failo dalijimas i du (kietiakai, vargsiukai)
00372 template <typename Container>
00373 void SplitFile(Container& grupe) {
00374     auto start_split = std::chrono::high_resolution_clock::now();
00375 
00376     // padalina konteineri i 2
00377     auto it = std::partition(grupe.begin(), grupe.end(), [](const auto student) {
00378         return student.getGalutinis() < 5;
00379     });
00380 
00381     // sukuria konteineri vargsiukams is atskirtu elementu
00382     Container vargsai;
00383     vargsai.reserve(std::distance(grupe.begin(), it));
00384     std::move(grupe.begin(), it, std::back_inserter(vargsai));
00385     grupe.erase(grupe.begin(), it); // istrina atskirtus elem is pradinio konteinerio
00386     grupe.shrink_to_fit();
00387 
00388     auto end_split = std::chrono::high_resolution_clock::now();
00389     std::chrono::duration<double> split_duration = end_split - start_split;
00390 
00391     std::ofstream fr1("Vargsiukai.txt");
00392     std::ofstream fr2("Kietiakai.txt");
00393 
00394     if (!fr1 || !fr2) {
00395         std::cerr « "Error opening output files!" « std::endl;
00396         return;
00397     }
00398 
00399     auto startV = std::chrono::high_resolution_clock::now();
00400     fr1 « std::left « std::setw(15) « "Vardas" « std::setw(15) « "Pavarde"
00401         « std::setw(15) « "Galutinis (Vid.)" « " / " « "Galutinis (Med.)" « std::endl;
00402     fr1 « "-------------------------------------------------------------" « std::endl;
00403 
00404     for (const auto& n : vargsai) {
00405         fr1 « std::left « std::setw(15) « n.getVardas() « std::setw(18) « n.getPavarde() « std::setw(7);
00406         if (n.getVm() == 'a')
00407             fr1 « std::fixed « std::setprecision(2) « n.getGalutinis() « "            -" « std::endl;
00408         else
00409             fr1 « " -               " « std::fixed « std::setprecision(2) « n.getGalutinis() « std::endl;
00410     }
00411     auto endV = std::chrono::high_resolution_clock::now();
00412     std::chrono::duration<double> Vduration = endV - startV;
00413 
00414     auto startK = std::chrono::high_resolution_clock::now();
00415     fr2 « std::left « std::setw(15) « "Vardas" « std::setw(15) « "Pavarde"
00416         « std::setw(15) « "Galutinis (Vid.)" « " / " « "Galutinis (Med.)" « std::endl;
00417     fr2 « "-------------------------------------------------------------" « std::endl;
00418     for (const auto& n : grupe) {
00419         fr2 « std::left « std::setw(15) « n.getVardas() « std::setw(18) « n.getPavarde() « std::setw(7);
00420         if (n.getVm() == 'a')
00421             fr2 « std::fixed « std::setprecision(2) « n.getGalutinis() « "            -" « std::endl;
00422         else
00423             fr2 « " -               " « std::fixed « std::setprecision(2) « n.getGalutinis() « std::endl;
```

```
00424     }
00425     auto endK = std::chrono::high_resolution_clock::now();
00426     std::chrono::duration<double> Kduration = endK - startK;
00427 
00428     fr1.close();
00429     fr2.close();
00430 
00431     std::cout « "Skirstymas ir irasymas: " « Kduration.count() + Vduration.count() +
      split_duration.count() « " s" « std::endl;
00432 }
00433 
00434 
00435 template <typename Container>
00436 void FinalScore(Container& grupe)
00437 {
00438     cout«"Calculate final scores using average or median? (a/m)"«endl;
00439     char am = 'a';
00440     //cin»am;
00441     while(am!= 'a' && am!= 'm')
00442     {
00443         cout«"Invalid input. Enter a or m"«endl;
00444         cin»am;
00445     }
00446 
00447     for(auto &n :grupe)
00448     {
00449         vector<int> paz = n.getPaz();
00450         sort(paz.begin(), paz.end());
00451         n.setVm(am);
00452         int suma=0;
00453             for(auto n: paz)
00454             {
00455             suma=suma+n;}
00456             if(am=='a'){
00457                 n.setGalutinis(0.4*(suma/paz.size())+0.6*n.getEgz());
00458             }
00459             else if (paz.size()%2==0){
00460                 n.setGalutinis(0.4*(paz[paz.size()/2] + paz[paz.size()/2-1])/2 +0.6*n.getEgz());
00461             }
00462             else{
00463                 n.setGalutinis(0.4*paz[paz.size()/2] +0.6*n.getEgz());
00464             }
00465         }
00466 }
00467 
00468 #endif
```

6.3 include/human.h File Reference

```
#include "manolib.h"
```

Classes

- class Zmogus

6.4 human.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "manolib.h"
00003 
00004 class Zmogus {
00005 protected:
00006     string Vardas, Pavarde;
00007 
00008 public:
00009     Zmogus() : Vardas(""), Pavarde("") {}
00010     Zmogus(const string& v, const string& p) : Vardas(v), Pavarde(p) {}
00011     virtual ~Zmogus() = default;
```

```
00012
00013
00014     string getVardas() const { return Vardas; }
00015     string getPavarde() const { return Pavarde; }
00016
00017     void setVardas(const string& v) { Vardas = v; }
00018     void setPavarde(const string& p) { Pavarde = p; }
00019
00020     virtual void print() const = 0;
00021 };
```

6.5 include/manolib.h File Reference

```
#include <vector>
#include <list>
#include <deque>
#include <iomanip>
#include <iostream>
#include <ctime>
#include <algorithm>
#include <fstream>
#include <sstream>
#include <chrono>
#include <limits>
#include <ios>
#include <string>
#include <type_traits>
#include <exception>
```

Variables

- const string MNames [25]
- const string MSurnames [25]
- const string FNames [25]
- const string FSurnames [25]

6.5.1 Variable Documentation

6.5.1.1 FNames

```
const string FNames[25]
```

Initial value:

```
= {
    "Egle", "Indre", "Lina","Neringa","Sigute","Ugne","Laura","Viktorija",
    "Rasa","Gintare","Agne","Ieva", "Milda","Margarita","Aiste", "Vilma",
    "Ruta","Aiste","Gabija","Jurate","Jurgita", "Vaiva", "Ula", "Greta",
    "Kotryna"

}
```

Definition at line 57 of file manolib.h.

6.5.1.2 FSurnames

```
const string FSurnames[25]
```

Initial value:

```
= {
"Norkute","Petronyte","Seskinyte","Pakalnaite","Daugelaite","Simonaityte",
"Giedre","Zukaite", "Norkute", "Kaminskaite", "Dapsyte", "Kucinskaite",
"Vaitkeviciute", "Vasiliauskaite", "Navickaite", "Urbonaite", "Grigoniene",
 "Rutkauskaite", "Vaitkute", "Pakalnyte", "Norkute", "Skripkaite", "Butkeviciute",
"Mickeviciute", "Brazaite"
}
```

Definition at line 64 of file manolib.h.

6.5.1.3 MNames

```
const string MNames[25]
```

Initial value:

```
= {
    "Andrius", "Dainius", "Jonas" , "Marius", "Orestas", "Povilas",
    "Aidas",  "Tomas",  "Vejas", "Zygimantas",  "Vaidotas",
    "Linas", "Kestutis", "Vaidotas", "Martynas",   "Gintaras",
     "Tomas", "Antanas", "Paulius",   "Jonas",  "Mantas",
     "Mindaugas", "Rokas", "Lukas", "Kazimieras"
}
```

Definition at line 44 of file manolib.h.

6.5.1.4 MSurnames

```
const string MSurnames[25]
```

Initial value:

```
= {
    "Petrauskas", "Jankauskas", "Kazlauskas", "Zukauskas", "Kavaliauskas", "Stankevicius", "Bieliauskas",
    "Budvytis", "Giedraitis",  "Rimkus",  "Valiukas", "Juknevicius", "Vaitkevicius",
     "Vasiliauskas", "Navickas",  "Urbonas", "Grigonis", "Rutkauskas",
    "Vaitkus", "Pakalnis", "Norkus", "Skripka", "Butkevicius", "Nedzinskas", "Mickevicius",
}
```

Definition at line 51 of file manolib.h.

6.6 manolib.h

Go to the documentation of this file.

```
00001 #ifndef MANOLIB_H
00002 #define MANOLIB_H
00003
00004 #include<vector>
00005 #include<list>
00006 #include<deque>
00007 #include<iomanip>
00008 #include<iostream>
00009 #include<ctime>
00010 #include<algorithm>
00011 #include<fstream>
00012 #include<sstream>
00013 #include<chrono>
00014 #include<limits>
00015 #include<ios>
```

```
00016 #include<string>
00017 #include<type_traits>
00018 #include<exception>
00019
00020
00021 using std::cout;
00022 using std::cin;
00023 using std::endl;
00024 using std::vector;
00025 using std::string;
00026 using std::setw;
00027 using std::sort;
00028 using std::left;
00029 using std::fixed;
00030 using std::setprecision;
00031 using std::getline;
00032 using std::ifstream;
00033 using std::ofstream;
00034 using std::istringstream;
00035 using std::list;
00036 using std::deque;
00037 using std::cerr;
00038 using std::vector;
00039 using std::string;
00040 using std::setw;
00041 using std::is_same_v;
00042
00043
00044 const string MNames[25] = {
00045     "Andrius", "Dainius", "Jonas" , "Marius", "Orestas", "Povilas",
00046     "Aidas",  "Tomas",  "Vejas", "Zygimantas",  "Vaidotas",
00047     "Linas", "Kestutis", "Vaidotas", "Martynas",   "Gintaras",
00048      "Tomas", "Antanas", "Paulius",   "Jonas",  "Mantas",
00049      "Mindaugas", "Rokas", "Lukas", "Kazimieras"
00050 };
00051 const string MSurnames[25] = {
00052     "Petrauskas", "Jankauskas", "Kazlauskas", "Zukauskas", "Kavaliauskas", "Stankevicius",
      "Bieliauskas",
00053     "Budvytis", "Giedraitis",  "Rimkus",  "Valiukas", "Juknevicius", "Vaitkevicius",
00054      "Vasiliauskas", "Navickas",  "Urbonas", "Grigonis", "Rutkauskas",
00055     "Vaitkus", "Pakalnis", "Norkus", "Skripka", "Butkevicius", "Nedzinskas", "Mickevicius",
00056 };
00057 const string FNames[25] = {
00058     "Egle", "Indre", "Lina","Neringa","Sigute","Ugne","Laura","Viktorija",
00059     "Rasa","Gintare","Agne","Ieva", "Milda","Margarita","Aiste", "Vilma",
00060     "Ruta","Aiste","Gabija","Jurate","Jurgita", "Vaiva", "Ula", "Greta",
00061     "Kotryna"
00062
00063 };
00064 const string FSurnames[25] = {
00065 "Norkute","Petronyte","Seskinyte","Pakalnaite","Daugelaite","Simonaityte",
00066 "Giedre","Zukaite", "Norkute", "Kaminskaite", "Dapsyte", "Kucinskaite",
00067 "Vaitkeviciute", "Vasiliauskaite", "Navickaite", "Urbonaite", "Grigoniene",
00068  "Rutkauskaite", "Vaitkute", "Pakalnyte", "Norkute", "Skripkaite", "Butkeviciute",
00069 "Mickeviciute", "Brazaite"
00070 };
00071
00072 #endif
```

6.7 include/student.h File Reference

```
#include "human.h"
#include "manolib.h"
```

Classes

- class Stud

6.8 student.h

Go to the documentation of this file.

```
00001 // Stud.h
00002 #pragma once
00003 #include "human.h"
00004 #include "manolib.h"
00005
00006
00007 class Stud : public Zmogus {
00008 private:
00009     std::vector<int> paz;
00010     int egz;
00011     char vm;
00012     double galutinis;
00013
00014 public:
00015     // Constructors
00016     Stud() : Zmogus(), egz(0), vm(' '), galutinis(0.0) {}
00017     Stud(const std::string& v, const std::string& p, const std::vector<int>& pazymiai, int e, char
      vmod, double gal)
00018         : Zmogus(v, p), paz(pazymiai), egz(e), vm(vmod), galutinis(gal) {}
00019
00020     // Destructor
00021     ~Stud() { paz.clear(); }
00022
00023     // Copy constructor
00024     Stud(const Stud& other)
00025         : Zmogus(other.Vardas, other.Pavarde), paz(other.paz), egz(other.egz), vm(other.vm),
      galutinis(other.galutinis) {}
00026
00027     // Copy assignment
00028     Stud& operator=(const Stud& other) {
00029         if (this == &other) return *this;
00030         Vardas = other.Vardas;
00031         Pavarde = other.Pavarde;
00032         paz = other.paz;
00033         egz = other.egz;
00034         vm = other.vm;
00035         galutinis = other.galutinis;
00036         return *this;
00037     }
00038
00039     // Move constructor
00040     Stud(Stud&& other)
00041         : Zmogus(std::move(other.Vardas), std::move(other.Pavarde)), paz(std::move(other.paz)),
00042           egz(other.egz), vm(other.vm), galutinis(other.galutinis) {
00043         other.egz = 0;
00044         other.vm = ' ';
00045         other.galutinis = 0.0;
00046     }
00047
00048     // Move assignment
00049     Stud& operator=(Stud&& other)  {
00050         if (this == &other) return *this;
00051         Vardas = std::move(other.Vardas);
00052         Pavarde = std::move(other.Pavarde);
00053         paz = std::move(other.paz);
00054         egz = other.egz;
00055         vm = other.vm;
00056         galutinis = other.galutinis;
00057         other.egz = 0;
00058         other.vm = ' ';
00059         other.galutinis = 0.0;
00060         return *this;
00061     }
00062
00063     // Input operator
00064     friend std::istream& operator»(std::istream& in, Stud& s) {
00065         std::cout « "Iveskite varda: ";
00066         in » s.Vardas;
00067         std::cout « "Iveskite pavarde: ";
00068         in » s.Pavarde;
00069
00070         std::cout « "Iveskite pazymiu kieki: ";
00071         int kiekis;
00072         in » kiekis;
00073
00074         s.paz.clear();
00075         std::cout « "Iveskite pazymius: ";
00076         for (int i = 0; i < kiekis; ++i) {
00077             int pazymys;
00078             in » pazymys;
00079             s.paz.push_back(pazymys);
00080         }
```

```
00081
00082         std::cout « "Iveskite egzamino rezultata: ";
00083         in » s.egz;
00084
00085         std::cout « "Iveskite vertinimo metoda (a/m): ";
00086         in » s.vm;
00087
00088         s.FinalScore();
00089         return in;
00090     }
00091
00092     // Output operator
00093     friend std::ostream& operator«(std::ostream& out, const Stud& s) {
00094         out « std::left « std::setw(15) « s.Vardas
00095             « std::setw(18) « s.Pavarde;
00096
00097         if (s.vm == 'a')
00098             out « std::fixed « std::setprecision(2) « std::setw(7) « s.galutinis « "            -" «
      std::endl;
00099         else
00100             out « " -               " « std::fixed « std::setprecision(2) « s.galutinis « std::endl;
00101
00102         return out;
00103     }
00104
00105     // getters & setters
00106     void setEgz(int e) { egz = e; }
00107     void setVm(char v) { vm = v; }
00108     void setGalutinis(double g) { galutinis = g; }
00109     void addPaz(int pazymys) { paz.push_back(pazymys); }
00110
00111     int getEgz() const { return egz; }
00112     char getVm() const { return vm; }
00113     double getGalutinis() const { return galutinis; }
00114     std::vector<int> getPaz() const { return paz; }
00115     void removeLastPaz() { paz.pop_back(); }
00116
00117     // Score calculation
00118     void FinalScore() {
00119         if (paz.empty()) {
00120             galutinis = 0.0;
00121             return;
00122         }
00123         if (vm == 'a') {
00124             double sum = 0.0;
00125             for (int pazymys : paz) sum += pazymys;
00126             galutinis = 0.4 * (sum / paz.size()) + 0.6 * egz;
00127         } else if (vm == 'm') {
00128             std::sort(paz.begin(), paz.end());
00129             int medianas = paz[paz.size() / 2];
00130             galutinis = 0.4 * medianas + 0.6 * egz;
00131         }
00132     }
00133
00134     void print() const override {
00135         cout « *this;
00136     }
00137 };
```

6.9 include/vector.h File Reference

```
#include "manolib.h"
```

Classes

- class Vektor< V >

6.10 vector.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "manolib.h"
00003
00004 template<typename V>
00005 class Vektor{
00006 private:
00007     V* duom;
00008     size_t dydis;
00009     size_t talpa;
00010
00011 void resize(size_t n)
00012 {
00013     V* temp = new V[n];
00014     for(size_t i=0; i<dydis; i++)
00015     {
00016         temp[i] = duom[i];
00017     }
00018     delete[] duom;
00019     duom = temp;
00020     talpa = n;
00021 }
00022 public:
00023 // Standard typedefs required by STL compatibility
00024 using value_type = V;
00025 using reference = V&;
00026 using const_reference = const V&;
00027 using iterator = V*;
00028 using const_iterator = const V*;
00029 using size_type = size_t;
00030
00031
00032
00033 //Konstruktoriai
00034 Vektor(): duom(nullptr), dydis(0), talpa(0) {}
00035 //inicilizavimas su talpaa
00036 Vektor(size_t d): dydis(0), talpa(d) {
00037     duom = new V[d];
00038 }
00039 Vektor(size_t d, const V& value): dydis(d), talpa(d) {
00040     duom = new V[d];
00041     for (size_t i = 0; i < d; ++i) {
00042         duom[i] = value;
00043     }
00044 }
00045 ~Vektor() {
00046     delete[] duom;
00047 }
00048
00049 size_t size() const {return dydis;}
00050 size_t max_size() const {return std::numeric_limits<size_t>::max();}
00051 size_t capacity() const {return talpa;}
00052 bool empty() const {return dydis ==0;}
00053 void reserve(size_t n) {
00054     if (n > talpa) {
00055         resize(n);
00056     }
00057 }
00058 void erase(size_t index) {
00059     if (index >= dydis) throw std::out_of_range("Index out of range");
00060     for (size_t i = index; i < dydis - 1; ++i) {
00061         duom[i] = duom[i + 1];
00062     }
00063     --dydis;
00064 }
00065 V* erase(V* first, V* last) {
00066     if (first < duom || last > duom + dydis || first > last)
00067         throw std::out_of_range("Invalid iterator range");
00068
00069     size_t start = first - duom;
00070     size_t end = last - duom;
00071     size_t range = end - start;
00072
00073     for (size_t i = end; i < dydis; ++i) {
00074         duom[i - range] = duom[i];
00075     }
00076
00077     dydis -= range;
00078     return duom + start;
00079 }
00080 void swap(Vektor<V>& other){
00081     std::swap(duom, other.duom);
00082     std::swap(dydis, other.dydis);
00083     std::swap(talpa, other.talpa);
00084 }
00085 void shrink_to_fit() {
00086     if (talpa > dydis) {
00087         V* temp = new V[dydis];
```

```
00088         for (size_t i = 0; i < dydis; ++i) {
00089             temp[i] = duom[i];
00090         }
00091         delete[] duom;
00092         duom = temp;
00093         talpa = dydis;}}
00094
00095 //Vektoriaus funkciju realizacijos
00096 void push_back(const V& value) {
00097     if (dydis == talpa) {
00098         size_t new_talpa;
00099         if (talpa == 0) {
00100             new_talpa = 1;
00101         } else {
00102             new_talpa = talpa * 2;
00103         }
00104         resize(new_talpa);
00105     }
00106     duom[dydis++] = value;
00107 }
00108 void pop_back() {
00109     if (dydis == 0) {
00110         throw std::out_of_range("Cannot pop_back from empty vector");
00111     }
00112     --dydis;}
00113
00114 V* begin() {return duom;}
00115 V* end() {return duom+dydis;}
00116 V& front() { return duom[0]; }
00117 V& back() { return duom[dydis - 1]; }
00118 V* clear()
00119 {
00120     delete[] duom;
00121     duom = nullptr;
00122     dydis = 0;
00123     talpa = 0;
00124     return duom;
00125 }
00126
00127
00128 //
00129 V& operator[](size_t index) {
00130     if (index >= dydis) throw std::out_of_range("Index out of range");
00131     return duom[index];
00132 }
00133 // Copy constructor
00134 Vektor(const Vektor<V>& other) : dydis(other.dydis), talpa(other.talpa) {
00135     duom = new V[talpa];
00136     for (size_t i = 0; i < dydis; ++i) {
00137         duom[i] = other.duom[i];
00138     }
00139 }
00140 // Copy assignment operator
00141 Vektor<V>& operator=(const Vektor<V>& other) {
00142     if (this != &other) {
00143         delete[] duom;
00144         dydis = other.dydis;
00145         talpa = other.talpa;
00146         duom = new V[dydis];
00147         for (size_t i = 0; i < dydis; ++i) {
00148             duom[i] = other.duom[i];
00149         }
00150     }
00151     return *this;
00152 }
00153 // Move constructor
00154 Vektor(Vektor<V>&& other) noexcept
00155     : duom(other.duom), dydis(other.dydis), talpa(other.talpa) {
00156     other.duom = nullptr;
00157     other.dydis = 0;
00158     other.talpa = 0;
00159 }
00160
00161 // Move assignment operator
00162 Vektor<V>& operator=(Vektor<V>&& other) noexcept {
00163     if (this != &other) {
00164         delete[] duom;
00165         duom = other.duom;
00166         dydis = other.dydis;
00167         talpa = other.talpa;
00168
00169         other.duom = nullptr;
00170         other.dydis = 0;
00171         other.talpa = 0;
00172     }
00173     return *this;
00174 }
```

```
00175 bool operator==(const Vektor<V>& other) const {
00176     if (dydis != other.dydis) return false;
00177     for (size_t i = 0; i < dydis; ++i) {
00178         if (!(duom[i] == other.duom[i])) return false;
00179     }
00180     return true;
00181 }
00182 //setteriai
00183
00184
00185 //getteriai
00186
00187
00188 };
```

6.11 README.md File Reference

6.12 src/main.cpp File Reference

```
#include "manolib.h"
#include "functions.h"
#include "student.h"
#include "vector.h"
```

Typedefs

- using Container = Vektor<Stud>

Functions

- int main ()

6.12.1 Typedef Documentation

6.12.1.1 Container

```
using Container = Vektor<Stud>
```

Definition at line 7 of file main.cpp.

6.12.2 Function Documentation

6.12.2.1 main()

```
int main ()
```

Definition at line 11 of file main.cpp.

6.13 main.cpp

Go to the documentation of this file.

```
00001 #include "manolib.h"
00002 #include "functions.h"
00003 #include "student.h"
00004 #include "vector.h"
00005
00006
00007 using Container = Vektor<Stud>;
00008 //using Container = std::vector<Stud>;
00009
00010
00011 int main()
00012 {
00013     srand(static_cast<unsigned int>(time(0)));
00014
00015     try
00016     {
00017         Container grupe;
00018         //cout « "Using container: " « typeid(Container).name() « endl;
00019         char a;
00020
00021         cout « "1 - Input everything manually" « endl;
00022         cout « "2 - Input names, generate scores" « endl;
00023         cout « "3 - Generate everything" « endl;
00024         cout « "4 - Read from file" « endl;
00025         cout « "5 - Performance test" « endl;
00026         cout « "6 - Class tests" « endl;
00027         cout « "7 - Vektor class tests" « endl;
00028
00029         cin » a;
00030         cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00031
00032         while (a < '1' || a > '7')
00033         {
00034             cout « "Invalid input. Enter 1, 2, 3, 4, 5 or 6: ";
00035             cin » a;
00036             cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00037         }
00038
00039         if (a == '1')
00040         {
00041             grupe = ManualInput<Container>();
00042         }
00043         else if (a == '2')
00044         {
00045             grupe = GenerateScores<Container>();
00046         }
00047         else if (a == '3')
00048         {
00049             grupe = GenerateEverything<Container>();
00050         }
00051         else if (a == '4')
00052         {
00053             string filename;
00054             cout « "Enter file name: ";
00055             cin » filename;
00056             cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00057
00058             grupe = ReadFile<Container>(filename);
00059
00060             if (grupe.empty())
00061             {
00062                 throw std::runtime_error("Error: Could not read file or file is empty.");
00063             }
00064         }
00065         else if (a == '5')
00066         {
00067             grupe = SpeedTesting<Container>();
00068             return 0;
00069         }
00070         else if (a == '6')
00071         {
00072             TestStud();  // Run the test function
00073             return 0;
00074         }
00075         else if (a == '7')
00076         {
00077             std::vector<int> og;
00078             Vektor <int> klase;
00079             size_t sz = 100000; // 100000, 1000000, 10000000, 100000000
00080             cin»sz;
00081             cout« "Size: " « sz « endl;
00082             auto start_split = std::chrono::high_resolution_clock::now();
```

```
00083             for (int i = 1; i <= sz; ++i) og.push_back(i);
00084             auto end_split = std::chrono::high_resolution_clock::now();
00085             std::chrono::duration<double> split_duration = end_split - start_split;
00086             cout« "OG vector: " « fixed « setprecision(5) « split_duration.count() « " s" « endl;
00087 
00088 
00089             auto start = std::chrono::high_resolution_clock::now();
00090             for (int i = 1; i <= sz; ++i) klase.push_back(i);
00091             auto end = std::chrono::high_resolution_clock::now();
00092             std::chrono::duration<double> duration = end - start;
00093             cout« "Vektor class: " « fixed « setprecision(5) « duration.count() « " s" « endl;
00094             return 0;
00095         }
00096         if (grupe.empty())
00097         {
00098             throw std::runtime_error("Error: No data to process.");
00099         }
00100 
00101         FinalScore(grupe); // Calculating final scores
00102 
00103         Sorting(grupe); // Sorting students
00104 
00105         cout « "Show results in file or terminal?" « endl;
00106         cout « "1 - File" « endl;
00107         cout « "2 - Terminal" « endl;
00108 
00109         int y;
00110         cin » y;
00111 
00112         while (cin.fail() || (y != 1 && y != 2))
00113         {
00114             cin.clear();
00115             cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Ignore invalid input
00116             cout « "Invalid input. Enter 1 or 2: ";
00117             cin » y;
00118         }
00119         if (y == 2)
00120             OutputToTerminal(grupe);
00121         else
00122             OutputToFile(grupe);
00123     }
00124     catch (const std::exception& e)
00125     {
00126         cerr « "An error occurred: " « e.what() « endl;
00127         return 1;
00128     }
00129     catch (...)
00130     {
00131         cerr « "An unknown error occurred." « endl;
00132         return 1;
00133     }
00134 
00135     return 0;
00136 }
```

6.14 src/tests.cpp File Reference

```
#include "catch.hpp"
#include "student.h"
#include "human.h"
#include "manolib.h"
#include "functions.h"
#include "vector.h"
```

Macros

- #define CATCH_CONFIG_MAIN

Functions

- TEST_CASE ("Studentu klases penkiu pirstu taisykles testas")
- TEST_CASE ("Kitu programos funkciju testai")
- TEST_CASE ("Vektoriaus klasės testai")

6.14.1 Macro Definition Documentation

6.14.1.1 CATCH_CONFIG_MAIN

```
#define CATCH_CONFIG_MAIN
```

Definition at line 1 of file tests.cpp.

6.14.2 Function Documentation

6.14.2.1 TEST_CASE() [1/3]

```
TEST_CASE (
             "Kitu programos funkciju testai" )
```

Definition at line 69 of file tests.cpp.

6.14.2.2 TEST_CASE() [2/3]

```
TEST_CASE (
             "Studentu klases penkiu pirstu taisykles testas" )
```

Definition at line 10 of file tests.cpp.

6.14.2.3 TEST_CASE() [3/3]

```
TEST_CASE (
             "Vektoriaus klasės testai" )
```

Definition at line 83 of file tests.cpp.

6.15 tests.cpp

Go to the documentation of this file.

```
00001 #define CATCH_CONFIG_MAIN
00002 #include "catch.hpp"
00003 #include "student.h"
00004 #include "human.h"
00005 #include "manolib.h"
00006 #include "functions.h"
00007 #include "vector.h"
00008
00009
00010 TEST_CASE("Studentu klases penkiu pirstu taisykles testas")
00011 {
00012     Stud student1("Jonas", "Jonaitis", {10, 10, 5, 6, 2, 8}, 7, 'a', 7.0);
00013
00014     SECTION("Copy konstruktorius")
00015     {
00016         Stud student2(student1);
00017         REQUIRE(student1.getVardas() == student2.getVardas());
00018         REQUIRE(student1.getPavarde() == student2.getPavarde());
00019         REQUIRE(student1.getEgz() == student2.getEgz());
00020         REQUIRE(student1.getVm() == student2.getVm());
00021         REQUIRE(student1.getGalutinis() == student2.getGalutinis());
00022     }
```

```
00023
00024     SECTION("Copy priskyrimo operatorius")
00025     {
00026         Stud student3 = student1;
00027
00028         REQUIRE(student1.getVardas() == student3.getVardas());
00029         REQUIRE(student1.getPavarde() == student3.getPavarde());
00030         REQUIRE(student1.getEgz() == student3.getEgz());
00031         REQUIRE(student1.getVm() == student3.getVm());
00032         REQUIRE(student1.getGalutinis() == student3.getGalutinis());
00033     }
00034
00035     SECTION("Move konstruktorius")
00036     {
00037         Stud student4(std::move(student1));
00038         REQUIRE(student4.getVardas() == "Jonas");
00039         REQUIRE(student4.getPavarde() == "Jonaitis");
00040         REQUIRE(student4.getEgz() == 7);
00041         REQUIRE(student4.getVm() == 'a');
00042         REQUIRE(student4.getGalutinis() == 7.0);
00043     }
00044
00045     SECTION("Move priskyrimo operatorius")
00046     {
00047         Stud student5;
00048         student5 = std::move(student1);
00049         REQUIRE(student5.getVardas() == "Jonas");
00050         REQUIRE(student5.getPavarde() == "Jonaitis");
00051         REQUIRE(student5.getEgz() == 7);
00052         REQUIRE(student5.getVm() == 'a');
00053         REQUIRE(student5.getGalutinis() == 7.0);
00054     }
00055     SECTION("Input operatorius")
00056     {
00057         std::istringstream input("Petras Petraitis 10 3 8 7 6 5 10 3 5 1 7 8 a");
00058         Stud student6;
00059         input » student6;
00060
00061         REQUIRE(student6.getVardas() == "Petras");
00062         REQUIRE(student6.getPavarde() == "Petraitis");
00063         REQUIRE(student6.getEgz() == 8);
00064         REQUIRE(student6.getVm() == 'a');
00065     }
00066
00067
00068 }
00069 TEST_CASE("Kitu programos funkciju testai")
00070 {
00071     Stud student7("Petras", "Petraitis", {10, 9, 8}, 7, 'a', 0.0);
00072     Stud student8("Petras", "Petraitis", {7, 6, 5}, 7, 'm', 0.0);
00073
00074     SECTION("FinalScore() testas")
00075     {
00076         student7.FinalScore();
00077         REQUIRE(student7.getGalutinis() == Approx(7.8));
00078
00079         student8.FinalScore();
00080         REQUIRE(student8.getGalutinis() == Approx(6.6));
00081     }
00082 }
00083 TEST_CASE("Vektoriaus klasės testai")
00084 {
00085     Vektor<Stud> vektorius;
00086     vektorius.push_back(Stud("Jonas", "Jonaitis", {10, 9, 8}, 7, 'a', 0.0));
00087     vektorius.push_back(Stud("Petras", "Petraitis", {7, 6, 5}, 7, 'm', 0.0));
00088
00089     SECTION("Vektoriaus dydis")
00090     {
00091         REQUIRE(vektorius.size() == 2);
00092     }
00093
00094     SECTION("Vektoriaus indeksavimas []")
00095     {
00096         REQUIRE(vektorius[0].getVardas() == "Jonas");
00097         REQUIRE(vektorius[1].getPavarde() == "Petraitis");
00098     }
00099     Vektor<int>test;
00100     test.push_back(1);
00101     test.push_back(2);
00102
00103     SECTION("pop_back() testas")
00104     {
00105         test.pop_back();
00106         REQUIRE(test.size() == 1);
00107         REQUIRE(test[0] == 1);
00108     }
00109     SECTION("reserve() testas")
```

```
00110     {
00111         test.reserve(25);
00112         REQUIRE(test.capacity() == 25);
00113     }
00114     SECTION("shrink_to_fit() testas")
00115     {
00116         test.reserve(25);
00117         test.shrink_to_fit();
00118         REQUIRE(test.capacity() == 2);
00119     }
00120     SECTION("swap() testas")
00121     {
00122         Vektor<int>test2;
00123         test2.push_back(3);
00124         test2.push_back(4);
00125         test.swap(test2);
00126         REQUIRE(test.size() == 2);
00127         REQUIRE(test[0] == 3);
00128         REQUIRE(test2.size() == 2);
00129         REQUIRE(test2[0] == 1);
00130     }
00131 }
```

Index