oop2

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.

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:

5.2 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

5.2.1 Detailed Description

Definition at line 4 of file human.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Zmogus() [1/2]

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

Definition at line 9 of file human.h.

Here is the caller graph for this function:

5.2.2.2 Zmogus() [2/2]

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

Definition at line 10 of file human.h.

5.2.2.3 ∼Zmogus()

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

5.2.3 Member Function Documentation

5.2.3.1 getPavarde()

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

Definition at line 15 of file human.h.

Here is the caller graph for this function:

5.2.3.2 getVardas()

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

Definition at line 14 of file human.h.

Here is the caller graph for this function:

5.2.3.3 print()

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

Implemented in Stud.

5.2.3.4 setPavarde()

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

Definition at line 18 of file human.h.

Here is the caller graph for this function:

5.2.3.5 setVardas()

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

Definition at line 17 of file human.h.

Here is the caller graph for this function:

5.2.4 Member Data Documentation

5.2.4.1 Pavarde

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

Definition at line 6 of file human.h.

5.2.4.2 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"
```

Include dependency graph for functions.h: This graph shows which files directly or indirectly include this file:

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.

Here is the caller graph for this function:

6.1.1.2 GenerateEverything()

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

Definition at line 56 of file functions.h.

Here is the call graph for this function: Here is the caller graph for this function:

6.1.1.3 GenerateFile()

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

Definition at line 275 of file functions.h.

Here is the caller graph for this function:

6.1.1.4 GenerateScores()

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

Definition at line 100 of file functions.h.

Here is the call graph for this function: Here is the caller graph for this function:

6.1.1.5 ManualInput()

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

Definition at line 140 of file functions.h.

Here is the caller graph for this function:

6.1.1.6 OutputToFile()

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

Definition at line 259 of file functions.h.

Here is the caller graph for this function:

6.1.1.7 OutputToTerminal()

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

Definition at line 246 of file functions.h.

Here is the caller graph for this function:

6.1.1.8 ReadFile()

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

Definition at line 170 of file functions.h.

Here is the call graph for this function: Here is the caller graph for this function:

6.1.1.9 Sorting()

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

Definition at line 212 of file functions.h.

Here is the call graph for this function: Here is the caller graph for this function:

6.1.1.10 SpeedTesting()

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

Definition at line 317 of file functions.h.

Here is the call graph for this function: Here is the caller graph for this function:

6.1.1.11 SplitFile()

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

Definition at line 373 of file functions.h.

Here is the caller graph for this function:

6.1.1.12 TestStud()

```
void TestStud ()
```

Definition at line 8 of file functions.h.

Here is the caller graph for this function:

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     }
```

```

    cout « "How many homework scores do you want to generate? ";
    while (!(cin » x) || x < 1) {
        cout « "Invalid input. Please enter a positive number: ";
        cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    }

    for (int i = 0; i < n; i++) {
        Stud laik;
        int gender = rand() % 2;
        if (gender == 0) {
            laik.setVardas(FNames[rand() % 25]);
            laik.setPavarde(FSurnames[rand() % 25]);

        } else {
            laik.setVardas(MNames[rand() % 25]);
            laik.setPavarde(MSurnames[rand() % 25]);
        }

        for (int j = 0; j < x; j++) {
            laik.addPaz(rand() % 10);

        }
        laik.setEgz(rand() % 10);

        grupe.push_back(laik);
    }
    return grupe;
}

// Vardo ivedimas, pazymiu generavimas
template <typename Container>
Container GenerateScores() {
    cout « "Selected 2-Input names, generate scores" « endl;
    cout « endl;
    Container grupe;

    while (true) {
        Stud laik;
        string Vardas, Pavarde;
        cout « "Input name: ";
        cin » Vardas;
        laik.setVardas(Vardas);
        cout « "Input surname: ";
        cin » Pavarde;
        laik.setPavarde(Pavarde);

        cout « "How many homework scores do you want to generate? ";
        int n;
        cin » n;

        for (int i = 0; i < n; i++) {
            laik.addPaz((rand() % 10));
        }
        laik.setEgz((rand() % 10));

        grupe.push_back(laik);

        cout « "Enter more students? (y/n) ";
        char x;
        cin » x;
        while (x != 'y' && x != 'n') {
            cout « "Invalid input. Enter y or n" « endl;
            cin » x;
        }
        if (x == 'n') break;
    }
    return grupe;
}

// Visko ivedimas ranka
template <typename Container>
Container ManualInput() {
    Container grupe;
    std::cout « "Manual student input selected.\n" « std::endl;

    while (true) {
        Stud laik;

        // Naudojamas » klasės operatorius
        std::cin » laik;

        grupe.push_back(laik);

        char more;
        std::cout « "Add another student? (y/n): ";
        std::cin » more;
        while (more != 'y' && more != 'n') {
            std::cout « "Invalid input. Enter y or n: ";
            std::cin » more;
        }
        if (more == 'n') break;

        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Clear leftover input
        std::cout « std::endl;
    }

    return grupe;
}

//Skaitymas is failo
template <typename Container>
Container ReadFile(string filename) {
    Container grupe;

    ifstream fd(filename);
    while (!fd) {
        cerr « "File not found!" « endl;
        cout « "Enter existing file name: ";
        cin » filename;
        fd.open(filename);
    }

    string line;
    getline(fd, line); // Skip first line

    while (getline(fd, line)) {
        istringstream iss(line);
        Stud laik;
        string vardas, pavarde;
        iss » vardas » pavarde;
        laik.setVardas(vardas);
        laik.setPavarde(pavarde);
        int num;

        while (iss » num) {
            laik.addPaz(num);
        }

        vector<int> pazymiai = laik.getPaz();
        if (!pazymiai.empty()) {
            laik.setEgz(pazymiai.back());
            laik.removeLastPaz();
        }

        grupe.push_back(laik);
    }

    fd.close();
    return grupe;
}

//Rusiavimas
template <typename Container>
void Sorting(Container &grupe) {
    cout « "How do you want to sort the students?" « endl;
    cout « "1 - By name" « endl;
    cout « "2 - By surname" « endl;
    cout « "3 - By final score descending" « endl;
    cout « "4 - By final score ascending" « endl;

    char x;
    cin » x;
    while (x != '1' && x != '2' && x != '3' && x != '4') {
        cout « "Invalid input. Enter 1, 2, 3, or 4: ";
        cin » x;
    }
    auto chrono_start = std::chrono::high_resolution_clock::now();

    auto comparator = [&](const Stud &a, const Stud &b) {
        if (x == '1') return a.getVardas() < b.getVardas();
        if (x == '2') return a.getPavarde() < b.getPavarde();
        if (x == '3') return a.getGalutinis() < b.getGalutinis();
        else return a.getGalutinis() > b.getGalutinis();
    };
    // constexpr apskaiciuoja kompiliavimo metu, o ne runtime metu
    if constexpr (is_same_v<Container, list<Stud») {
        grupe.sort(comparator);  // listo sortas
            } else {
        sort(grupe.begin(), grupe.end(), comparator);  // std::sort vectoriui ir deque
    }
    auto chrono_end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> duration = chrono_end - chrono_start;
    cout « "SORTING TOOK: " « fixed « setprecision(5) « duration.count() « " s" « endl;
}

// Templated function to output results
template <typename Container>
void OutputToTerminal(Container &grupe) {
    cout « left « setw(15) « "Vardas" « setw(15) « "Pavarde"
         « setw(15) « "Galutinis (Vid.)"
         « " / "
         « "Galutinis (Med.)" « endl;
    cout « "-----------------------------------------------------------" « endl;
    for (const auto &n : grupe) {
        // Output naudojant « klasės operatoriu
        std::cout « n;
    }
}

template <typename Container>
void OutputToFile(Container& grupe)
{
    ofstream out("rezultatai.txt");
    out«std::left«setw(15)«"Vardas"«setw(15)«"Pavarde"
    «setw(15)«"Galutinis (Vid.)"«" / "«"Galutinis (Med.)"«endl;
    out«"-----------------------------------------------------------"«endl;
for(auto n :grupe)
  {
    out«std::left«setw(15)«n.getVardas()«setw(18)«n.getPavarde()«setw(7);
    if(n.getVm() == 'a') out«std::fixed«std::setprecision(2)«n.getGalutinis()«"            -"«endl;
    else out«" -                "«std::fixed«std::setprecision(2)«n.getGalutinis()«endl;
  }
out.close();

}

string GenerateFile(int StudentCount)
{
    string filename = "Studentai"+std::to_string(StudentCount)+".txt";
    ifstream fd(filename);
    if(fd.good())
    {
        cout«filename«" already exists"«endl;
        return filename;
    }
    fd.close();

    auto start = std::chrono::high_resolution_clock::now();
    ofstream fr(filename);
    if(!fr)
    {
        cout«"Error creating file" «filename«endl;
    }

    fr«std::left«setw(16)«"Vardas Pavarde "«std::left«setw(20)«"Pazymiai   "«"Egzaminas"«endl;
    for(int i=0; i<StudentCount; i++)
    {
        if(rand()%2==0)
        {
            fr«MNames[rand()%25]«" "«MSurnames[rand()%25]«" ";
        }
        else
        {
            fr«FNames[rand()%25]«" "«FSurnames[rand()%25]«" ";
        }
        for(int j=0; j<10; j++)
        {
            fr«rand()%10«" ";
        }
        fr«rand()%10«endl;
    }
    auto end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> duration = end - start;
    cout«filename «" sukurtas per "«fixed«setprecision(5)«duration.count() « " s" « endl;
    return filename;
}

template <typename Container>
Container SpeedTesting()
{
    Container grupe;
    string filename;

    cout « "Ar norite generuoti faila? (y/n): ";
    char choice;
    cin » choice;

    if (choice == 'y' || choice == 'Y')
    {
        int StudentCount;
        cout « "Enter the number of students: ";
        cin » StudentCount;

        filename = GenerateFile(StudentCount);
    }

    if (filename.empty()) // If filename is still empty, ask for input
    {
        cout « "Iveskite testo faila: ";
        cin » filename;
    }

    cout « "Chosen file: " « filename « endl;

    auto startRead = std::chrono::high_resolution_clock::now(); // Timer for file reading
    grupe = ReadFile<Container>(filename);
    auto endRead = std::chrono::high_resolution_clock::now();

    FinalScore(grupe);

    auto startSort = std::chrono::high_resolution_clock::now(); // Timer for sorting
    Sorting(grupe);
    auto endSort = std::chrono::high_resolution_clock::now();

    auto startSplit = std::chrono::high_resolution_clock::now();
     SplitFile(grupe);
    auto endSplit = std::chrono::high_resolution_clock::now();

    // Calculate and display durations
    std::chrono::duration<double> durationRead = endRead - startRead;
    std::chrono::duration<double> durationSort = endSort - startSort;
    std::chrono::duration<double> durationSplit = endSplit - startSplit;

    cout « filename « " failo nuskaitymo laikas: " « fixed « setprecision(5) « durationRead.count() « " s" « endl;
    cout « filename « " failo rusiavimas: " « fixed « setprecision(5) « durationSort.count() « " s" « endl;
    cout « filename « " failo paskirstymo ir irasymo laikas: " « fixed « setprecision(5) « durationSplit.count() « " s" « endl;
    cout « filename « " is viso uztruko: " « fixed « setprecision(5)
         « (durationRead.count() + durationSort.count() + durationSplit.count()) « " s" « endl;

    return grupe;
}

//Failo dalijimas i du (kietiakai, vargsiukai)
template <typename Container>
void SplitFile(Container& grupe) {
    auto start_split = std::chrono::high_resolution_clock::now();

    // padalina konteineri i 2
    auto it = std::partition(grupe.begin(), grupe.end(), [](const auto student) {
        return student.getGalutinis() < 5;
    });

    // sukuria konteineri vargsiukams is atskirtu elementu
    Container vargsai;
    vargsai.reserve(std::distance(grupe.begin(), it));
    std::move(grupe.begin(), it, std::back_inserter(vargsai));
    grupe.erase(grupe.begin(), it); // istrina atskirtus elem is pradinio konteinerio
    grupe.shrink_to_fit();

    auto end_split = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> split_duration = end_split - start_split;

    std::ofstream fr1("Vargsiukai.txt");
    std::ofstream fr2("Kietiakai.txt");

    if (!fr1 || !fr2) {
        std::cerr « "Error opening output files!" « std::endl;
        return;
    }

    auto startV = std::chrono::high_resolution_clock::now();
    fr1 « std::left « std::setw(15) « "Vardas" « std::setw(15) « "Pavarde"
        « std::setw(15) « "Galutinis (Vid.)" « " / " « "Galutinis (Med.)" « std::endl;
    fr1 « "-----------------------------------------------------------" « std::endl;

    for (const auto& n : vargsai) {
        fr1 « std::left « std::setw(15) « n.getVardas() « std::setw(18) « n.getPavarde() « std::setw(7);
        if (n.getVm() == 'a')
            fr1 « std::fixed « std::setprecision(2) « n.getGalutinis() « "            -" « std::endl;
        else
            fr1 « " -                " « std::fixed « std::setprecision(2) « n.getGalutinis() « std::endl;
    }
    auto endV = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> Vduration = endV - startV;

    auto startK = std::chrono::high_resolution_clock::now();
    fr2 « std::left « std::setw(15) « "Vardas" « std::setw(15) « "Pavarde"
        « std::setw(15) « "Galutinis (Vid.)" « " / " « "Galutinis (Med.)" « std::endl;
    fr2 « "----------------------------------------------------------" « std::endl;
    for (const auto& n : grupe) {
        fr2 « std::left « std::setw(15) « n.getVardas() « std::setw(18) « n.getPavarde() « std::setw(7);
        if (n.getVm() == 'a')
            fr2 « std::fixed « std::setprecision(2) « n.getGalutinis() « "             -" « std::endl;
        else
            fr2 « " -                " « std::fixed « std::setprecision(2) « n.getGalutinis() « std::endl;
    }
    auto endK = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> Kduration = endK - startK;

    fr1.close();
    fr2.close();

    std::cout « "Skirstymas ir irasymas: " « Kduration.count() + Vduration.count() + split_duration.count() « " s" « std::endl;
}


template <typename Container>
void FinalScore(Container& grupe)
{
    cout«"Calculate final scores using average or median? (a/m)"«endl;
    char am;
    cin»am;
    while(am!= 'a' && am!= 'm')
    {
        cout«"Invalid input. Enter a or m"«endl;
        cin»am;
    }

    for(auto &n :grupe)
    {
        vector<int> paz = n.getPaz();
        sort(paz.begin(), paz.end());
        n.setVm(am);
        int suma=0;
            for(auto n: paz)
            {
            suma=suma+n;}
            if(am=='a'){
                n.setGalutinis(0.4*(suma/paz.size())+0.6*n.getEgz());
            }
            else if (paz.size()%2==0){
                n.setGalutinis(0.4*(paz[paz.size()/2] + paz[paz.size()/2-1])/2 +0.6*n.getEgz());
            }
            else{
                n.setGalutinis(0.4*paz[paz.size()/2] +0.6*n.getEgz());
            }
        }
}

#endif
```

6.3 include/human.h File Reference

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

Include dependency graph for human.h: This graph shows which files directly or indirectly include this file:

Classes

- class Zmogus

6.4 human.h

Go to the documentation of this file.

```
#pragma once
#include "manolib.h"

class Zmogus {
protected:
    string Vardas, Pavarde;

public:
    Zmogus() : Vardas(""), Pavarde("") {}
    Zmogus(const string& v, const string& p) : Vardas(v), Pavarde(p) {}
    virtual ~Zmogus() = default;


    string getVardas() const { return Vardas; }
    string getPavarde() const { return Pavarde; }

    void setVardas(const string& v) { Vardas = v; }
    void setPavarde(const string& p) { Pavarde = p; }

    virtual void print() const = 0;
};
```

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>
```

Include dependency graph for manolib.h: This graph shows which files directly or indirectly include this file:

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.

```
#ifndef MANOLIB_H
#define MANOLIB_H

#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>


using std::cout;
using std::cin;
using std::endl;
using std::vector;
using std::string;
using std::setw;
using std::sort;
using std::left;
using std::fixed;
using std::setprecision;
using std::getline;
using std::ifstream;
using std::ofstream;
using std::istringstream;
using std::list;
using std::deque;
using std::cerr;
using std::vector;
using std::string;
using std::setw;
using std::is_same_v;


const string MNames[25] = {
    "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"
};
const string MSurnames[25] = {
    "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",
};
const string FNames[25] = {
    "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"

};
const string FSurnames[25] = {
"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"
};

#endif
```

6.7 include/student.h File Reference

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

Include dependency graph for student.h: This graph shows which files directly or indirectly include this file:

Classes

- class Stud

6.8 student.h

Go to the documentation of this file.

```
// Stud.h
#pragma once
#include "human.h"
#include "manolib.h"


class Stud : public Zmogus {
private:
    std::vector<int> paz;
    int egz;
    char vm;
    double galutinis;

public:
    // Constructors
    Stud() : Zmogus(), egz(0), vm(' '), galutinis(0.0) {}
    Stud(const std::string& v, const std::string& p, const std::vector<int>& pazymiai, int e, char
vmod, double gal)
        : Zmogus(v, p), paz(pazymiai), egz(e), vm(vmod), galutinis(gal) {}

    // Destructor
    ~Stud() { paz.clear(); }

    // Copy constructor
    Stud(const Stud& other)
        : Zmogus(other.Vardas, other.Pavarde), paz(other.paz), egz(other.egz), vm(other.vm),
galutinis(other.galutinis) {}

    // Copy assignment
    Stud& operator=(const Stud& other) {
        if (this == &other) return *this;
        Vardas = other.Vardas;
        Pavarde = other.Pavarde;
        paz = other.paz;
        egz = other.egz;
        vm = other.vm;
        galutinis = other.galutinis;
        return *this;
    }

    // Move constructor
    Stud(Stud&& other)
        : Zmogus(std::move(other.Vardas), std::move(other.Pavarde)), paz(std::move(other.paz)),
          egz(other.egz), vm(other.vm), galutinis(other.galutinis) {
        other.egz = 0;
        other.vm = ' ';
        other.galutinis = 0.0;
    }

    // Move assignment
    Stud& operator=(Stud&& other)  {
        if (this == &other) return *this;
        Vardas = std::move(other.Vardas);
        Pavarde = std::move(other.Pavarde);
        paz = std::move(other.paz);
        egz = other.egz;
        vm = other.vm;
        galutinis = other.galutinis;
        other.egz = 0;
        other.vm = ' ';
        other.galutinis = 0.0;
        return *this;
    }

    // Input operator
    friend std::istream& operator»(std::istream& in, Stud& s) {
        std::cout « "Iveskite varda: ";
        in » s.Vardas;
        std::cout « "Iveskite pavarde: ";
        in » s.Pavarde;

        std::cout « "Iveskite pazymiu kieki: ";
        int kiekis;
        in » kiekis;

        s.paz.clear();
        std::cout « "Iveskite pazymius: ";
        for (int i = 0; i < kiekis; ++i) {
            int pazymys;
            in » pazymys;
            s.paz.push_back(pazymys);
        }

        std::cout « "Iveskite egzamino rezultata: ";
        in » s.egz;

        std::cout « "Iveskite vertinimo metoda (a/m): ";
        in » s.vm;

        s.FinalScore();
        return in;
    }

    // Output operator
    friend std::ostream& operator«(std::ostream& out, const Stud& s) {
        out « std::left « std::setw(15) « s.Vardas
            « std::setw(18) « s.Pavarde;

        if (s.vm == 'a')
            out « std::fixed « std::setprecision(2) « std::setw(7) « s.galutinis « "            -" « std::endl;
        else
            out « " -               " « std::fixed « std::setprecision(2) « s.galutinis « std::endl;

        return out;
    }

    // getters & setters
    void setEgz(int e) { egz = e; }
    void setVm(char v) { vm = v; }
    void setGalutinis(double g) { galutinis = g; }
    void addPaz(int pazymys) { paz.push_back(pazymys); }

    int getEgz() const { return egz; }
    char getVm() const { return vm; }
    double getGalutinis() const { return galutinis; }
    std::vector<int> getPaz() const { return paz; }
    void removeLastPaz() { paz.pop_back(); }

    // Score calculation
    void FinalScore() {
        if (paz.empty()) {
            galutinis = 0.0;
            return;
        }
        if (vm == 'a') {
            double sum = 0.0;
            for (int pazymys : paz) sum += pazymys;
            galutinis = 0.4 * (sum / paz.size()) + 0.6 * egz;
        } else if (vm == 'm') {
            std::sort(paz.begin(), paz.end());
            int medianas = paz[paz.size() / 2];
            galutinis = 0.4 * medianas + 0.6 * egz;
        }
    }

    void print() const override {
        cout « *this;
    }
};
```

6.9 README.md File Reference

6.10 src/main.cpp File Reference

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

`#include "student.h"`
Include dependency graph for main.cpp:

Typedefs

- using Container = std::vector<Stud>

Functions

- int main ()

6.10.1 Typedef Documentation

6.10.1.1 Container

```
using Container = std::vector<Stud>
```

Definition at line 6 of file main.cpp.

6.10.2 Function Documentation

6.10.2.1 main()

```
int main ()
```

Definition at line 10 of file main.cpp.

Here is the call graph for this function:

6.11 main.cpp

Go to the documentation of this file.

```
00001 #include "manolib.h"
00002 #include "functions.h"
00003 #include "student.h"
00004
00005
00006 using Container = std::vector<Stud>;
00007 //using Container = std::deque<Stud>;
00008 //using Container = std::list<Stud>;
00009
00010 int main()
00011 {
00012     srand(static_cast<unsigned int>(time(0)));
00013
00014     try
00015     {
00016         Container grupe;
00017         cout « "Using container: " « typeid(Container).name() « endl;
00018         char a;
00019
00020         cout « "1 - Input everything manually" « endl;
00021         cout « "2 - Input names, generate scores" « endl;
00022         cout « "3 - Generate everything" « endl;
00023         cout « "4 - Read from file" « endl;
00024         cout « "5 - Performance test" « endl;
```

```
        cout « "6 - Class tests" « endl;

        cin » a;
        cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');

        while (a < '1' || a > '6')
        {
            cout « "Invalid input. Enter 1, 2, 3, 4, 5 or 6: ";
            cin » a;
            cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        }

        if (a == '1')
        {
            grupe = ManualInput<Container>();
        }
        else if (a == '2')
        {
            grupe = GenerateScores<Container>();
        }
        else if (a == '3')
        {
            grupe = GenerateEverything<Container>();
        }
        else if (a == '4')
        {
            string filename;
            cout « "Enter file name: ";
            cin » filename;
            cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');

            grupe = ReadFile<Container>(filename);

            if (grupe.empty())
            {
                throw std::runtime_error("Error: Could not read file or file is empty.");
            }
        }
        else if (a == '5')
        {
            grupe = SpeedTesting<Container>();
            return 0;
        }
        else if (a == '6')
        {
            TestStud();  // Run the test function
            return 0;
        }

        if (grupe.empty())
        {
            throw std::runtime_error("Error: No data to process.");
        }

        FinalScore(grupe); // Calculating final scores

        Sorting(grupe); // Sorting students

        cout « "Show results in file or terminal?" « endl;
        cout « "1 - File" « endl;
        cout « "2 - Terminal" « endl;

        int y;
        cin » y;

        while (cin.fail() || (y != 1 && y != 2))
        {
            cin.clear();
            cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Ignore invalid input
            cout « "Invalid input. Enter 1 or 2: ";
            cin » y;
        }
        if (y == 2)
            OutputToTerminal(grupe);
        else
            OutputToFile(grupe);
    }
    catch (const std::exception& e)
    {
        cerr « "An error occurred: " « e.what() « endl;
        return 1;
    }
    catch (...)
    {
        cerr « "An unknown error occurred." « endl;
        return 1;
    }
```

```
        return 0;
}
```

6.12 src/tests.cpp File Reference

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

Include dependency graph for tests.cpp:

Macros

- #define CATCH_CONFIG_MAIN

Functions

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

6.12.1 Macro Definition Documentation

6.12.1.1 CATCH_CONFIG_MAIN

```
#define CATCH_CONFIG_MAIN
```

Definition at line 1 of file tests.cpp.

6.12.2 Function Documentation

6.12.2.1 TEST_CASE() [1/2]

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

Definition at line 68 of file tests.cpp.

Here is the call graph for this function:

6.12.2.2 TEST_CASE() [2/2]

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

Definition at line 9 of file tests.cpp.

Here is the call graph for this function:

6.13 tests.cpp

Go to the documentation of this file.

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


TEST_CASE("Studentu klases penkiu pirstu taisykles testas")
{
    Stud student1("Jonas", "Jonaitis", {10, 10, 5, 6, 2, 8}, 7, 'a', 7.0);

    SECTION("Copy konstruktorius")
    {
        Stud student2(student1);
        REQUIRE(student1.getVardas() == student2.getVardas());
        REQUIRE(student1.getPavarde() == student2.getPavarde());
        REQUIRE(student1.getEgz() == student2.getEgz());
        REQUIRE(student1.getVm() == student2.getVm());
        REQUIRE(student1.getGalutinis() == student2.getGalutinis());
    }

    SECTION("Copy priskyrimo operatorius")
    {
        Stud student3 = student1;

        REQUIRE(student1.getVardas() == student3.getVardas());
        REQUIRE(student1.getPavarde() == student3.getPavarde());
        REQUIRE(student1.getEgz() == student3.getEgz());
        REQUIRE(student1.getVm() == student3.getVm());
        REQUIRE(student1.getGalutinis() == student3.getGalutinis());
    }

    SECTION("Move konstruktorius")
    {
        Stud student4(std::move(student1));
        REQUIRE(student4.getVardas() == "Jonas");
        REQUIRE(student4.getPavarde() == "Jonaitis");
        REQUIRE(student4.getEgz() == 7);
        REQUIRE(student4.getVm() == 'a');
        REQUIRE(student4.getGalutinis() == 7.0);
    }

    SECTION("Move priskyrimo operatorius")
    {
        Stud student5;
        student5 = std::move(student1);
        REQUIRE(student5.getVardas() == "Jonas");
        REQUIRE(student5.getPavarde() == "Jonaitis");
        REQUIRE(student5.getEgz() == 7);
        REQUIRE(student5.getVm() == 'a');
        REQUIRE(student5.getGalutinis() == 7.0);
    }
    SECTION("Input operatorius")
    {
        std::istringstream input("Petras Petraitis 10 3 8 7 6 5 10 3 5 1 7 8 a");
        Stud student6;
        input » student6;

        REQUIRE(student6.getVardas() == "Petras");
        REQUIRE(student6.getPavarde() == "Petraitis");
        REQUIRE(student6.getEgz() == 8);
        REQUIRE(student6.getVm() == 'a');
    }


}
TEST_CASE("Kitu programos funkciju testai")
{
    Stud student7("Petras", "Petraitis", {10, 9, 8}, 7, 'a', 0.0);
    Stud student8("Petras", "Petraitis", {7, 6, 5}, 7, 'm', 0.0);

    SECTION("FinalScore() testas")
    {
        student7.FinalScore();
        REQUIRE(student7.getGalutinis() == Approx(7.8));

        student8.FinalScore();
        REQUIRE(student8.getGalutinis() == Approx(6.6));
    }
}
```

Index