Dokumentacija

Generated by Doxygen 1.10.0

Chapter 1

README

v2.0

Makefile set-up: **Makefile idiegimas naudojant Chocolatey (Windows)**

Isitikinkite, kad turite Chocolatey idiegima: patikrinkite, ar jusu kompiuteryje yra idiegta Chocolatey. Jei ne, idiekite pagal instrukcijas https://chocolatey.org/install.

Idiekite Makefile: atidarykite PowerShell kaip administratorius ir ivykdykite sia komanda:

choco install make

Patikrinkite idiegima: patikrinkite, ar Makefile sekmingai idiegtas, vykdydami komanda:

make –version

Jei viskas sekminga, turetumete pamatyti Make versijos informacija

Makefile idiegimas naudojant kitus metodus **MacOS:** Makefile iprastai yra idiegtas standartinėje MacOS distribucijoje, todel papildomu veiksmu paprastai nereikia

Linux: Daugumoje Linux distribuciju Makefile taip pat yra idiegtas is anksto. Jei reikia, naudokite savo paketu tvarkykle, pvz., apt-get, yum, dnf, arba kita pagal distribucija

Windows (be Chocolatey): Noredami idiegti Makefile Windows sistemoje be Chocolatey, galite naudoti rankinius idiegimo failus, kuriuos galite rasti internete. Paprastai tie failai turi .exe pletini ir gali buti lengvai idiegti, sekdamie pridedamas instrukcijas

Norint pradeti, i terminala reikia ivesti "make", kai viskas bus sukompiliuota, galima testi su programa, jei norima, galima rasyti "make clean" norint istrinti .o ir .exe failus

Programos naudojimas veikimo metu:

Vartotojas pasirenka, su kokiu konteineriu norima vygdyti programa ir pasirinktinai i terminala parasoma: ./vektoriai , ./list arba ./deque

Vos paleidus programa atsiras pasirinkimas ar pratestuoti musu turimus klases metodus , jei paspaudziame 't', tada pasirenkame numeri nuo 1-5 ir gauname testo rezultata

Toliau musu bus klausiama, ar norime ivesti duomenis ar skaityti is failo

1. Jei bus pasirenkamas duomenu ivedimas, bus reikalaujama pasirinkti ar norima ivesti/generuoti duomenis

1.1 Ar vienu, ar kitu budu reikes ivesti studentu vardus ir pavardes, toliau reikes ivesti studentu namu darbu ir egzamino pazymius

1.2 Jei bus pasirinktas duomenu generavimas, po vardu ir pavardziu irasymo nieko daryti nebereikes

1.3 Galiausiai reikes pasirinkti kur norime matyti duomenis ekrane ar faile

1. Jei pacioje pradzioje bus pasirinktas skaitymas, jusu bus klausiama ar norite generuoti naujus failus, jei ivesite 't'(taip), bus generuojami nauji failai, jei ivesite bet koki kita simboli, programa veiks toliau

2.1 Toliau, jusu bus klausiama ar norite skaityti naujai sukurtus failus, ar jau turimus

2.2 Bus prasoma ivesti, pasirinktinai, turimu/nauju failu kieki, jie bus nuskaitomi, isvedami apytiksliai testavimu laikai ekrane bei sukuriami nauji failai, kuriuose yra surusiuoti studentai pagal vidurki (nuskriaustieji/mokslinciai)

2.3 Galiausiai, kaip ir anksciau, bus isvedami apytiksliai testavimu laikai ekrane bei sukuriami nauji failai, kuriuose yra surusiuoti studentai pagal vidurki (nuskriaustieji/mokslinciai)

RELEASAI

0.1

Sukurta nauja repozitorija, realizuotos elementarios funkcijos, kaip vidurkio ir medianos skaiciavimas. Rezultate gavome, kad vektorius naudoti yra zymiai efektyviau atminties atzvilgiu.

0.2

Programa padaryta prieinamesne vartotojui, galima ne tik irasyti, bet ir skaityti is failo. Testuojama su 10000, 100000 ir 1000000 dydzio failais.

0.3

Prideti header failai, try/catch blokai. Rezultate programa tapo labiau strukturizuota bei klaidu gaudymas uzdraus-davo programos luzima.

0.4

Programa pagal vartotojo pasirinkima sukuria naujus failus, isskirsto mokinius i vargsiukus ir mokslincius, isveda i failus. Padaryti tikslus laiko matavimai.

1.0

Programa padaryta veikti su atskiro tipo konteineriais: deque, list ir vector. Kiekvienas pagal tris strategijas. Pagal matavimo rezultatus greiciausiai buvo vykdoma vector programa naudojant 3 strategija.

1.1

Atliktas repozitorijos kopijavimas. Programoje is strukturu pereinama i klases. Rezultate, programa veikia nasiau naudojant klases.

1.2

Igyvendinti visi "Rule of Five" ir isviesties bei ivesties operatoriai savai klasei.

1.5

Sukurta dar viena **bazine**, abstrakcioji klase, kuriai priklauso klase derived. Prideti konstruktoriaus, copy konstruk-toriaus... testavimai.

2.0

Per Doxygen HTML formatu sukurta dokumentacija bei padaryti Unit testai naudojant patogu C++ framework'a supratimui.

Kaip atrodo Doxygen dokumentacija:

Unit testai:

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

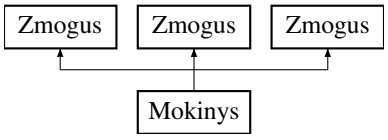
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Mokinys Class Reference

Inheritance diagram for Mokinys:



Public Member Functions

- **Mokinys** (string vard="", string pav="", vector< int > nd={}, int e=0, double vid=0.0, double med=0.0)
- void patikrinimas () const override
- **Mokinys** (const Mokinys &other)
- **Mokinys** (Mokinys &&other) noexcept
- Mokinys & **operator=** (const Mokinys &other)
- Mokinys & **operator=** (Mokinys &&other) noexcept
- string **getVardas** () const
- string **getPavarde** () const
- vector< int > **getND** () const
- int **getEgzaminas** () const
- double **getVID** () const
- double **getMED** () const
- void **setVardas** (const string &name)
- void **setPavarde** (const string &surname)
- void **addND** (int nd)
- void **clearND** ()
- void **setEgzaminas** (int exam)
- void **setVID** (double vid)
- void **setMED** (double med)
- void **Vidurkis** (vector< Mokinys > &A)
- void **Isvedimas** (const vector< Mokinys > &A, int MOK_kiekis, string isvedimas)
- void **Isvedimas2** (const vector< Mokinys > &A, int MOK_kiekis, string isvedimas)
- void **Skaitymas** (vector< Mokinys > &Nuskriaustieji, vector< Mokinys > &Mokslinciai, vector< int > &IrasuSk, string failas, vector< Mokinys > &A, int &temp, char strategija)

- void **StudentuRusiavimas** (vector< Mokinys > &Nuskriaustieji, vector< Mokinys > &Mokslinciai, vector< Mokinys > &A, vector< int > &IrasuSk, string failas, int &temp)
- void **StudentuRusiavimas2** (vector< Mokinys > &Nuskriaustieji, vector< Mokinys > &Mokslinciai, vector< Mokinys > &A, vector< int > &IrasuSk, string failas, int &temp)
- void **StudentuRusiavimas3** (vector< Mokinys > &Nuskriaustieji, vector< Mokinys > &Mokslinciai, vector< Mokinys > &A, vector< int > &IrasuSk, string failas, int &temp)
- void **Rikiavimas** (vector< Mokinys > &Mokslinciai, vector< Mokinys > &Nuskriaustieji, vector< int > &IrasuSk)
- **Mokinys** (string vard="", string pav="", list< int > nd={}, int e=0, double vid=0.0, double med=0.0)
- void patikrinimas () const override
- **Mokinys** (const Mokinys &other)
- **Mokinys** (Mokinys &&other) noexcept
- Mokinys & **operator=** (const Mokinys &other)
- Mokinys & **operator=** (Mokinys &&other) noexcept
- string **getVardas** () const
- string **getPavarde** () const
- list< int > **getND** () const
- int **getEgzaminas** () const
- double **getVID** () const
- double **getMED** () const
- void **setVardas** (const string &name)
- void **setPavarde** (const string &surname)
- void **addND** (int nd)
- void **clearND** ()
- void **setEgzaminas** (int exam)
- void **setVID** (double vid)
- void **setMED** (double med)
- void **Vidurkis** (list< Mokinys > &A)
- void **Isvedimas** (const list< Mokinys > &A, int MOK_kiekis, string isvedimas)
- void **Isvedimas2** (const list< Mokinys > &A, int MOK_kiekis, string isvedimas)
- void **Skaitymas** (list< Mokinys > &Nuskriaustieji, list< Mokinys > &Mokslinciai, list< int > &IrasuSk, string failas, list< Mokinys > &A, int &temp, char strategija)
- void **StudentuRusiavimas** (list< Mokinys > &Nuskriaustieji, list< Mokinys > &Mokslinciai, list< Mokinys > &A, list< int > &IrasuSk, string failas, int &temp)
- void **StudentuRusiavimas2** (list< Mokinys > &Nuskriaustieji, list< Mokinys > &Mokslinciai, list< Mokinys > &A, list< int > &IrasuSk, string failas, int &temp)
- void **StudentuRusiavimas3** (list< Mokinys > &Nuskriaustieji, list< Mokinys > &Mokslinciai, list< Mokinys > &A, list< int > &IrasuSk, string failas, int &temp)
- void **Rikiavimas** (list< Mokinys > &Mokslinciai, list< Mokinys > &Nuskriaustieji, list< int > &IrasuSk)
- **Mokinys** (string vard="", string pav="", deque< int > nd={}, int e=0, double vid=0.0, double med=0.0)
- void patikrinimas () const override
- **Mokinys** (const Mokinys &other)
- **Mokinys** (Mokinys &&other) noexcept
- Mokinys & **operator=** (const Mokinys &other)
- Mokinys & **operator=** (Mokinys &&other) noexcept
- string **getVardas** () const
- string **getPavarde** () const
- deque< int > **getND** () const
- int **getEgzaminas** () const
- double **getVID** () const
- double **getMED** () const
- void **setVardas** (const string &name)
- void **setPavarde** (const string &surname)
- void **addND** (int nd)
- void **clearND** ()

- void **setEgzaminas** (int exam)
- void **setVID** (double vid)
- void **setMED** (double med)
- void **Vidurkis** (deque< Mokinys > &A)
- void **Isvedimas** (const deque< Mokinys > &A, int MOK_kiekis, string isvedimas)
- void **Isvedimas2** (const deque< Mokinys > &A, int MOK_kiekis, string isvedimas)
- void **Skaitymas** (deque< Mokinys > &Nuskriaustieji, deque< Mokinys > &Mokslinciai, deque< int > &IrasuSk, string failas, deque< Mokinys > &A, int &temp, char strategija)
- void **StudentuRusiavimas** (deque< Mokinys > &Nuskriaustieji, deque< Mokinys > &Mokslinciai, deque< Mokinys > &A, deque< int > &IrasuSk, string failas, int &temp)
- void **StudentuRusiavimas2** (deque< Mokinys > &Nuskriaustieji, deque< Mokinys > &Mokslinciai, deque< Mokinys > &A, deque< int > &IrasuSk, string failas, int &temp)
- void **StudentuRusiavimas3** (deque< Mokinys > &Nuskriaustieji, deque< Mokinys > &Mokslinciai, deque< Mokinys > &A, deque< int > &IrasuSk, string failas, int &temp)
- void **Rikiavimas** (deque< Mokinys > &Mokslinciai, deque< Mokinys > &Nuskriaustieji, deque< int > &IrasuSk)

Public Member Functions inherited from Zmogus

- **Zmogus** (string vard="", string pav="")
- **Zmogus** (string vard="", string pav="")
- **Zmogus** (string vard="", string pav="")

Static Public Member Functions

- static bool **PagalVidurki** (const Mokinys &a, const Mokinys &b)
- static bool **PagalMediana** (const Mokinys &a, const Mokinys &b)
- static bool **PagalVarda** (const Mokinys &a, const Mokinys &b)
- static bool **PagalPavarde** (const Mokinys &a, const Mokinys &b)
- static bool **PagalVidurki** (const Mokinys &a, const Mokinys &b)
- static bool **PagalMediana** (const Mokinys &a, const Mokinys &b)
- static bool **PagalVarda** (const Mokinys &a, const Mokinys &b)
- static bool **PagalPavarde** (const Mokinys &a, const Mokinys &b)
- static bool **PagalVidurki** (const Mokinys &a, const Mokinys &b)
- static bool **PagalMediana** (const Mokinys &a, const Mokinys &b)
- static bool **PagalVarda** (const Mokinys &a, const Mokinys &b)
- static bool **PagalPavarde** (const Mokinys &a, const Mokinys &b)

Friends

- std::ostream & **operator**<< (std::ostream &fr, const Mokinys &temp1)
- istream & **operator**>> (istream &fd, Mokinys &temp1)
- std::ostream & **operator**<< (std::ostream &fr, const Mokinys &temp1)
- istream & **operator**>> (istream &fd, Mokinys &temp1)
- std::ostream & **operator**<< (std::ostream &fr, const Mokinys &temp1)
- istream & **operator**>> (istream &fd, Mokinys &temp1)

Additional Inherited Members

Protected Attributes inherited from Zmogus

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

5.1.1 Member Function Documentation

5.1.1.1 patikrinimas() [1/3]

```
void Mokinys::patikrinimas ( ) const  [inline], [override], [virtual]
```

Implements Zmogus.

5.1.1.2 patikrinimas() [2/3]

```
void Mokinys::patikrinimas ( ) const  [inline], [override], [virtual]
```

Implements Zmogus.

5.1.1.3 patikrinimas() [3/3]

```
void Mokinys::patikrinimas ( ) const  [inline], [override], [virtual]
```

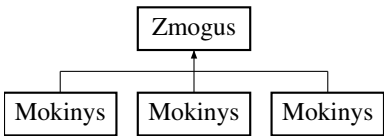
Implements Zmogus.

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

- mokinys.h
- mokinys1.h
- mokinys2.h
- mokinys.cpp
- mokinys1.cpp
- mokinys2.cpp

5.2 Zmogus Class Reference

Inheritance diagram for Zmogus:



Public Member Functions

- **Zmogus** (string vard="", string pav="")
- virtual void **patikrinimas** () const =0
- **Zmogus** (string vard="", string pav="")
- virtual void **patikrinimas** () const =0
- **Zmogus** (string vard="", string pav="")
- virtual void **patikrinimas** () const =0

Protected Attributes

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

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

- mokinys.h
- mokinys1.h
- mokinys2.h

Chapter 6

File Documentation

6.1 funkcijos.h

```
00001 #ifndef FUNKCIJOS_H
00002 #define FUNKCIJOS_H
00003 #include "mokinys.h"
00004
00005 bool Patikrinimas(string kint);
00006 void GeneruotiFailus(vector<Mokinys>& Nuskriaustieji, vector<Mokinys>& Mokslinciai, vector<int>&
      IrasuSk, vector<Mokinys>& A);
00007 void testConstructor();
00008 void testCopyConstructor();
00009 void testMoveConstructor();
00010 void testCopyAssignment();
00011 void testMoveAssignment();
00012
00013 #endif
```

6.2 funkcijos1.h

```
00001 #ifndef FUNKCIJOS_H
00002 #define FUNKCIJOS_H
00003
00004 #include "mokinys1.h"
00005
00006 bool Patikrinimas (string kint);
00007 void GeneruotiFailus (list<Mokinys>& Nuskriaustieji, list<Mokinys>& Mokslinciai, list<int>& IrasuSk,
      list<Mokinys>& A);
00008 void testConstructor();
00009 void testCopyConstructor();
00010 void testMoveConstructor();
00011 void testCopyAssignment();
00012 void testMoveAssignment();
00013
00014 #endif
```

6.3 funkcijos2.h

```
00001 #ifndef FUNKCIJOS_H
00002 #define FUNKCIJOS_H
00003
00004 #include "mokinys2.h"
00005
00006 bool Patikrinimas(string kint);
00007 void GeneruotiFailus(deque<Mokinys>& Nuskriaustieji, deque<Mokinys>& Mokslinciai, deque<int>& IrasuSk,
      deque<Mokinys>& A);
00008 void testConstructor();
00009 void testCopyConstructor();
00010 void testMoveConstructor();
00011 void testCopyAssignment();
00012 void testMoveAssignment();
00013
00014 #endif
```

6.4 mokinys.h

```
00001 #ifndef MOKINYS_H
00002 #define MOKINYS_H
00003
00004 #include <iostream>
00005 #include <fstream>
00006 #include <iomanip>
00007 #include <string>
00008 #include <vector>
00009 #include <sstream>
00010 #include <algorithm>
00011 #include <chrono>
00012 #include <cstring>
00013 #include <stdexcept>
00014 #include <list>
00015 #include <deque>
00016 #include <cassert>
00017 #include <utility>
00018
00019
00020 using namespace std;
00021
00022 const char CRfv[] = "rezultatai.txt";
00023 const char CRfv2[] = "naujas_failas.txt";
00024 const char CRfv3[] = "mokslinciai.txt";
00025 const char CRfv4[] = "nuskriaustieji.txt";
00026
00027 // const char CDfv[] = "kursiokai.txt";
00028 const char CDfv0[] = "studentai10000.txt";
00029 const char CDfv1[] = "studentai100000.txt";
00030 const char CDfv2[] = "studentai1000000.txt";
00031
00032 class Zmogus
00033 {
00034 protected:
00035     string vardas;
00036     string pavarde;
00037
00038 public:
00039     Zmogus(string vard = "", string pav = "") : vardas(move(vard)), pavarde(move(pav)) {}
00040     virtual ~Zmogus() = default;
00041     virtual void patikrinimas() const = 0;
00042 };
00043
00044 class Mokinys : public Zmogus
00045 {
00046 private:
00047     /* string vardas;
00048     string pavarde; */
00049     vector<int> ND;
00050     int egzaminas;
00051     double VID;
00052     double MED;
00053
00054 public:
00055     // Constructor
00056     Mokinys(string vard = "", string pav = "", vector<int> nd = {}, int e = 0, double vid = 0.0,
      double med = 0.0)
00057         : Zmogus(move(vard), move(pav)), ND(nd), egzaminas(e), VID(vid), MED(med) {}
00058
00059     // Destructor
00060     ~Mokinys() = default;
00061
00062     void patikrinimas() const override{};
00063
00064     // Copy constructor
00065     Mokinys(const Mokinys &other) : Zmogus(other), ND(other.ND), egzaminas(other.egzaminas),
      VID(other.VID), MED(other.MED) {}
00066
00067     // Move contructor
00068     Mokinys(Mokinys &&other) noexcept
00069         : Zmogus((move(other.vardas)), (move(other.pavarde))),
00070           ND(move(other.ND)), egzaminas(exchange(other.egzaminas, 0)),
00071           VID(exchange(other.VID, 0)), MED(exchange(other.MED, 0)) {}
00072
00073     // Copy Assignment Operator
00074     Mokinys &operator=(const Mokinys &other)
00075     {
00076         Zmogus::operator=(other);
00077         ND = other.ND;
00078         egzaminas = other.egzaminas;
00079         VID = other.VID;
00080         MED = other.MED;
00081         return *this;
00082     }
00083
```

```
00084     // Move Assignment Operator
00085     Mokinys &operator=(Mokinys &&other) noexcept
00086     {
00087         vardas = move(other.vardas);
00088         pavarde = move(other.pavarde);
00089         ND = move(other.ND);
00090         egzaminas = exchange(other.egzaminas, 0);
00091         VID = exchange(other.VID, 0);
00092         MED = exchange(other.MED, 0);
00093         return *this;
00094     }
00095
00096     friend std::ostream &operator«(std::ostream &fr, const Mokinys &temp1)
00097     {
00098         fr « "Vardas: " « temp1.vardas « endl;
00099         fr « "Pavarde: " « temp1.pavarde « endl;
00100         fr « "Namu darbai: ";
00101         for (int pazymys : temp1.ND)
00102         {
00103             fr « pazymys « " ";
00104         }
00105         cout « endl;
00106         fr « "Egzamino pazymys: " « temp1.egzaminas « endl;
00107         fr « "Mediana: " « temp1.MED « endl;
00108         fr « "Vidurkis: " « temp1.VID « endl;
00109         return fr;
00110     }
00111
00112     friend istream &operator»(istream &fd, Mokinys &temp1)
00113     {
00114         cout « "Iveskite varda: ";
00115         fd » temp1.vardas;
00116         cout « "Iveskite pavarde: ";
00117         fd » temp1.pavarde;
00118         cout « "Iveskite namu darbus: ";
00119         int pazymys;
00120         temp1.ND.clear();
00121         while (fd » pazymys && pazymys != 0)
00122         {
00123             temp1.ND.push_back(pazymys);
00124         }
00125         cout « "Iveskite egzamino pazymi: ";
00126         fd » temp1.egzaminas;
00127         cout « "Iveskite mediana: ";
00128         fd » temp1.MED;
00129         cout « "Iveskite vidurki: ";
00130         fd » temp1.VID;
00131         return fd;
00132     }
00133
00134     // Getter functions
00135     string getVardas() const { return vardas; }
00136     string getPavarde() const { return pavarde; }
00137     vector<int> getND() const { return ND; }
00138     int getEgzaminas() const { return egzaminas; }
00139     double getVID() const { return VID; }
00140     double getMED() const { return MED; }
00141
00142     // Setter functions
00143     void setVardas(const string &name) { vardas = name; }
00144     void setPavarde(const string &surname) { pavarde = surname; }
00145     void addND(int nd) { ND.push_back(nd); }
00146     void clearND() { ND.clear(); }
00147     void setEgzaminas(int exam) { egzaminas = exam; }
00148     void setVID(double vid) { VID = vid; }
00149     void setMED(double med) { MED = med; }
00150
00151     // Utility functions
00152     void Vidurkis(vector<Mokinys> &A);
00153     void Isvedimas(const vector<Mokinys> &A, int MOK_kiekis, string isvedimas);
00154     void Isvedimas2(const vector<Mokinys> &A, int MOK_kiekis, string isvedimas);
00155     static bool PagalVidurki(const Mokinys &a, const Mokinys &b);
00156     static bool PagalMediana(const Mokinys &a, const Mokinys &b);
00157     static bool PagalVarda(const Mokinys &a, const Mokinys &b);
00158     static bool PagalPavarde(const Mokinys &a, const Mokinys &b);
00159     void Skaitymas(vector<Mokinys> &Nuskriaustieji, vector<Mokinys> &Mokslinciai, vector<int>
      &IrasuSk, string failas, vector<Mokinys> &A, int &temp, char strategija);
00160     void StudentuRusiavimas(vector<Mokinys> &Nuskriaustieji, vector<Mokinys> &Mokslinciai,
      vector<Mokinys> &A, vector<int> &IrasuSk, string failas, int &temp);
00161     void StudentuRusiavimas2(vector<Mokinys> &Nuskriaustieji, vector<Mokinys> &Mokslinciai,
      vector<Mokinys> &A, vector<int> &IrasuSk, string failas, int &temp);
00162     void StudentuRusiavimas3(vector<Mokinys> &Nuskriaustieji, vector<Mokinys> &Mokslinciai,
      vector<Mokinys> &A, vector<int> &IrasuSk, string failas, int &temp);
00163     void Rikiavimas(vector<Mokinys> &Mokslinciai, vector<Mokinys> &Nuskriaustieji, vector<int>
      &IrasuSk);
00164 };
00165
```

```
00166 #endif
```

6.5 mokinys1.h

```
00001 #ifndef MOKINYS_H
00002 #define MOKINYS_H
00003
00004 #include <iostream>
00005 #include <fstream>
00006 #include <iomanip>
00007 #include <string>
00008 #include <list>
00009 #include <sstream>
00010 #include <algorithm>
00011 #include <chrono>
00012 #include <cstring>
00013 #include <stdexcept>
00014 #include <list>
00015 #include <deque>
00016 #include <cassert>
00017 #include <utility>
00018
00019 using namespace std;
00020
00021 const char CRfv[] = "rezultatai.txt";
00022 const char CRfv2[] = "naujas_failas.txt";
00023 const char CRfv3[] = "mokslinciai.txt";
00024 const char CRfv4[] = "nuskriaustieji.txt";
00025
00026 // const char CDfv[] = "kursiokai.txt";
00027 const char CDfv0[] = "studentai10000.txt";
00028 const char CDfv1[] = "studentai100000.txt";
00029 const char CDfv2[] = "studentai1000000.txt";
00030
00031 class Zmogus
00032 {
00033 protected:
00034     string vardas;
00035     string pavarde;
00036
00037 public:
00038     Zmogus(string vard = "", string pav = "") : vardas(move(vard)), pavarde(move(pav)) {}
00039     virtual ~Zmogus() = default;
00040     virtual void patikrinimas() const = 0;
00041 };
00042
00043 class Mokinys : public Zmogus
00044 {
00045 private:
00046     /* string vardas;
00047     string pavarde; */
00048     list<int> ND;
00049     int egzaminas;
00050     double VID;
00051     double MED;
00052
00053 public:
00054     // Constructor
00055     Mokinys(string vard = "", string pav = "", list<int> nd = {}, int e = 0, double vid = 0.0, double
      med = 0.0)
00056         : Zmogus(move(vard), move(pav)), ND(nd), egzaminas(e), VID(vid), MED(med) {}
00057
00058     // Destructor
00059     ~Mokinys() = default;
00060
00061     void patikrinimas() const override{};
00062
00063     // Copy constructor
00064     Mokinys(const Mokinys &other) : Zmogus(other), ND(other.ND), egzaminas(other.egzaminas),
      VID(other.VID), MED(other.MED) {}
00065
00066     // Move contructor
00067     Mokinys(Mokinys &&other) noexcept
00068         : Zmogus((move(other.vardas)), (move(other.pavarde))),
00069           ND(move(other.ND)), egzaminas(exchange(other.egzaminas, 0)),
00070           VID(exchange(other.VID, 0)), MED(exchange(other.MED, 0)) {}
00071
00072     // Copy Assignment Operator
00073     Mokinys &operator=(const Mokinys &other)
00074     {
00075         Zmogus::operator=(other);
00076         ND = other.ND;
00077         egzaminas = other.egzaminas;
```

```
00078         VID = other.VID;
00079         MED = other.MED;
00080         return *this;
00081     }
00082 
00083     // Move Assignment Operator
00084     Mokinys &operator=(Mokinys &&other) noexcept
00085     {
00086         vardas = move(other.vardas);
00087         pavarde = move(other.pavarde);
00088         ND = move(other.ND);
00089         egzaminas = exchange(other.egzaminas, 0);
00090         VID = exchange(other.VID, 0);
00091         MED = exchange(other.MED, 0);
00092         return *this;
00093     }
00094 
00095      friend std::ostream &operator«(std::ostream &fr, const Mokinys &temp1)
00096     {
00097         fr « "Vardas: " « temp1.vardas « endl;
00098         fr « "Pavarde: " « temp1.pavarde « endl;
00099         fr « "Namu darbai: ";
00100         for (int pazymys : temp1.ND)
00101         {
00102             fr « pazymys « " ";
00103         }
00104         cout « endl;
00105         fr « "Egzamino pazymys: " « temp1.egzaminas « endl;
00106         fr « "Mediana: " « temp1.MED « endl;
00107         fr « "Vidurkis: " « temp1.VID « endl;
00108         return fr;
00109     }
00110 
00111     friend istream &operator»(istream &fd, Mokinys &temp1)
00112     {
00113         cout « "Iveskite varda: ";
00114         fd » temp1.vardas;
00115         cout « "Iveskite pavarde: ";
00116         fd » temp1.pavarde;
00117         cout « "Iveskite namu darbus: ";
00118         int pazymys;
00119         temp1.ND.clear();
00120         while (fd » pazymys && pazymys != 0)
00121         {
00122             temp1.ND.push_back(pazymys);
00123         }
00124         cout « "Iveskite egzamino pazymi: ";
00125         fd » temp1.egzaminas;
00126         cout « "Iveskite mediana: ";
00127         fd » temp1.MED;
00128         cout « "Iveskite vidurki: ";
00129         fd » temp1.VID;
00130         return fd;
00131     }
00132 
00133     // Getter functions
00134     string getVardas() const { return vardas; }
00135     string getPavarde() const { return pavarde; }
00136     list<int> getND() const { return ND; }
00137     int getEgzaminas() const { return egzaminas; }
00138     double getVID() const { return VID; }
00139     double getMED() const { return MED; }
00140 
00141     // Setter functions
00142     void setVardas(const string &name) { vardas = name; }
00143     void setPavarde(const string &surname) { pavarde = surname; }
00144     void addND(int nd) { ND.push_back(nd); }
00145     void clearND() { ND.clear(); }
00146     void setEgzaminas(int exam) { egzaminas = exam; }
00147     void setVID(double vid) { VID = vid; }
00148     void setMED(double med) { MED = med; }
00149 
00150     // Utility functions
00151     void Vidurkis(list<Mokinys> &A);
00152     void Isvedimas(const list<Mokinys> &A, int MOK_kiekis, string isvedimas);
00153     void Isvedimas2(const list<Mokinys> &A, int MOK_kiekis, string isvedimas);
00154     static bool PagalVidurki(const Mokinys &a, const Mokinys &b);
00155     static bool PagalMediana(const Mokinys &a, const Mokinys &b);
00156     static bool PagalVarda(const Mokinys &a, const Mokinys &b);
00157     static bool PagalPavarde(const Mokinys &a, const Mokinys &b);
00158     void Skaitymas(list<Mokinys> &Nuskriaustieji, list<Mokinys> &Mokslinciai, list<int> &IrasuSk,
      string failas, list<Mokinys> &A, int &temp, char strategija);
00159     void StudentuRusiavimas(list<Mokinys> &Nuskriaustieji, list<Mokinys> &Mokslinciai, list<Mokinys>
      &A, list<int> &IrasuSk, string failas, int &temp);
00160     void StudentuRusiavimas2(list<Mokinys> &Nuskriaustieji, list<Mokinys> &Mokslinciai, list<Mokinys>
      &A, list<int> &IrasuSk, string failas, int &temp);
00161     void StudentuRusiavimas3(list<Mokinys> &Nuskriaustieji, list<Mokinys> &Mokslinciai, list<Mokinys>
```

```
      &A, list<int> &IrasuSk, string failas, int &temp);
00162     void Rikiavimas(list<Mokinys> &Mokslinciai, list<Mokinys> &Nuskriaustieji, list<int> &IrasuSk);
00163 };
00164
00165 #endif
```

6.6 mokinys2.h

```
00001 #ifndef MOKINYS_H
00002 #define MOKINYS_H
00003
00004 #include <iostream>
00005 #include <fstream>
00006 #include <iomanip>
00007 #include <string>
00008 #include <deque>
00009 #include <sstream>
00010 #include <algorithm>
00011 #include <chrono>
00012 #include <cstring>
00013 #include <stdexcept>
00014 #include <list>
00015 #include <deque>
00016 #include <cassert>
00017 #include <utility>
00018
00019 using namespace std;
00020
00021 const char CRfv[] = "rezultatai.txt";
00022 const char CRfv2[] = "naujas_failas.txt";
00023 const char CRfv3[] = "mokslinciai.txt";
00024 const char CRfv4[] = "nuskriaustieji.txt";
00025
00026 // const char CDfv[] = "kursiokai.txt";
00027 const char CDfv0[] = "studentai10000.txt";
00028 const char CDfv1[] = "studentai100000.txt";
00029 const char CDfv2[] = "studentai1000000.txt";
00030
00031 class Zmogus
00032 {
00033 protected:
00034     string vardas;
00035     string pavarde;
00036
00037 public:
00038     Zmogus(string vard = "", string pav = "") : vardas(move(vard)), pavarde(move(pav)) {}
00039     virtual ~Zmogus() = default;
00040     virtual void patikrinimas() const = 0;
00041 };
00042
00043 class Mokinys : public Zmogus
00044 {
00045 private:
00046     /* string vardas;
00047     string pavarde; */
00048     deque<int> ND;
00049     int egzaminas;
00050     double VID;
00051     double MED;
00052
00053 public:
00054     // Constructor
00055     Mokinys(string vard = "", string pav = "", deque<int> nd = {}, int e = 0, double vid = 0.0, double
      med = 0.0)
00056         : Zmogus(move(vard), move(pav)), ND(nd), egzaminas(e), VID(vid), MED(med) {}
00057
00058     // Destructor
00059     ~Mokinys() = default;
00060
00061     void patikrinimas() const override{};
00062
00063     // Copy constructor
00064     Mokinys(const Mokinys &other) : Zmogus(other), ND(other.ND), egzaminas(other.egzaminas),
      VID(other.VID), MED(other.MED) {}
00065
00066     // Move contructor
00067     Mokinys(Mokinys &&other) noexcept
00068         : Zmogus((move(other.vardas)), (move(other.pavarde))),
00069           ND(move(other.ND)), egzaminas(exchange(other.egzaminas, 0)),
00070           VID(exchange(other.VID, 0)), MED(exchange(other.MED, 0)) {}
00071
00072     // Copy Assignment Operator
00073     Mokinys &operator=(const Mokinys &other)
```

```
00074     {
00075         Zmogus::operator=(other);
00076         ND = other.ND;
00077         egzaminas = other.egzaminas;
00078         VID = other.VID;
00079         MED = other.MED;
00080         return *this;
00081     }
00082 
00083     // Move Assignment Operator
00084     Mokinys &operator=(Mokinys &&other) noexcept
00085     {
00086         vardas = move(other.vardas);
00087         pavarde = move(other.pavarde);
00088         ND = move(other.ND);
00089         egzaminas = exchange(other.egzaminas, 0);
00090         VID = exchange(other.VID, 0);
00091         MED = exchange(other.MED, 0);
00092         return *this;
00093     }
00094 
00095      friend std::ostream &operator«(std::ostream &fr, const Mokinys &temp1)
00096     {
00097         fr « "Vardas: " « temp1.vardas « endl;
00098         fr « "Pavarde: " « temp1.pavarde « endl;
00099         fr « "Namu darbai: ";
00100         for (int pazymys : temp1.ND)
00101         {
00102             fr « pazymys « " ";
00103         }
00104         cout « endl;
00105         fr « "Egzamino pazymys: " « temp1.egzaminas « endl;
00106         fr « "Mediana: " « temp1.MED « endl;
00107         fr « "Vidurkis: " « temp1.VID « endl;
00108         return fr;
00109     }
00110 
00111     friend istream &operator»(istream &fd, Mokinys &temp1)
00112     {
00113         cout « "Iveskite varda: ";
00114         fd » temp1.vardas;
00115         cout « "Iveskite pavarde: ";
00116         fd » temp1.pavarde;
00117         cout « "Iveskite namu darbus: ";
00118         int pazymys;
00119         temp1.ND.clear();
00120         while (fd » pazymys && pazymys != 0)
00121         {
00122             temp1.ND.push_back(pazymys);
00123         }
00124         cout « "Iveskite egzamino pazymi: ";
00125         fd » temp1.egzaminas;
00126         cout « "Iveskite mediana: ";
00127         fd » temp1.MED;
00128         cout « "Iveskite vidurki: ";
00129         fd » temp1.VID;
00130         return fd;
00131     }
00132 
00133     // Getter functions
00134     string getVardas() const { return vardas; }
00135     string getPavarde() const { return pavarde; }
00136     deque<int> getND() const { return ND; }
00137     int getEgzaminas() const { return egzaminas; }
00138     double getVID() const { return VID; }
00139     double getMED() const { return MED; }
00140 
00141     // Setter functions
00142     void setVardas(const string &name) { vardas = name; }
00143     void setPavarde(const string &surname) { pavarde = surname; }
00144     void addND(int nd) { ND.push_back(nd); }
00145     void clearND() { ND.clear(); }
00146     void setEgzaminas(int exam) { egzaminas = exam; }
00147     void setVID(double vid) { VID = vid; }
00148     void setMED(double med) { MED = med; }
00149 
00150     // Utility functions
00151     void Vidurkis(deque<Mokinys> &A);
00152     void Isvedimas(const deque<Mokinys> &A, int MOK_kiekis, string isvedimas);
00153     void Isvedimas2(const deque<Mokinys> &A, int MOK_kiekis, string isvedimas);
00154     static bool PagalVidurki(const Mokinys &a, const Mokinys &b);
00155     static bool PagalMediana(const Mokinys &a, const Mokinys &b);
00156     static bool PagalVarda(const Mokinys &a, const Mokinys &b);
00157     static bool PagalPavarde(const Mokinys &a, const Mokinys &b);
00158     void Skaitymas(deque<Mokinys> &Nuskriaustieji, deque<Mokinys> &Mokslinciai, deque<int> &IrasuSk,
      string failas, deque<Mokinys> &A, int &temp, char strategija);
00159     void StudentuRusiavimas(deque<Mokinys> &Nuskriaustieji, deque<Mokinys> &Mokslinciai,
```

```
      deque<Mokinys> &A, deque<int> &IrasuSk, string failas, int &temp);
00160     void StudentuRusiavimas2(deque<Mokinys> &Nuskriaustieji, deque<Mokinys> &Mokslinciai,
      deque<Mokinys> &A, deque<int> &IrasuSk, string failas, int &temp);
00161     void StudentuRusiavimas3(deque<Mokinys> &Nuskriaustieji, deque<Mokinys> &Mokslinciai,
      deque<Mokinys> &A, deque<int> &IrasuSk, string failas, int &temp);
00162     void Rikiavimas(deque<Mokinys> &Mokslinciai, deque<Mokinys> &Nuskriaustieji, deque<int> &IrasuSk);
00163 };
00164
00165 #endif
```

Index