V2.0

Generated by Doxygen 1.12.0

Chapter 1

Kompiuterio specifikacijos:

- **CPU: 12th Gen Intel(R) Core(TM) i7-1260P 2.10 GHz**
- **RAM: 16 GB**
- **SSD (1TB)**

1.1 Įdiegimo instrukcija

Naudokite terminalą ir CMake.

1. Parsisiųskite projektą kaip ZIP failą ir išskleiskite (extract).
2. Nukopijuokite projekto direktoriją, sukuriame build direktoriją ir į ją persikeliame:
   ```
   cd "[direktorija į projektą]"
   mkdir build
   cd build
   ```
3. Paleidžiame CMake, kad sugeneruotų reikalingus failus:
   ```
   cmake ..
   ```
4. Sukompiliuojame kodą, sukuriame .exe failą:
   ```
   cmake --build . --config Release
   ```

1.2 V2.0

1. main.cpp yra main failas.
2. MyLib3.h faile aprašytos bibliotekos.
3. Stud3.cpp faile surašytos visos funkcijos, realizuoti įvedimo/išvedimo operatoriai
4. Zmogus.h faile aprašyta bazinė klasė
5. Stud3.h faile aprašyta išvestinė (derived) klasė, jos konstruktoriai, funkcijų deklaracijos ir pnš.
6. Failu_kurimas.cpp faile yra failų generavimo ir duomenų nuskaitymo, išvedimo į failus funkcijos.
7. RandInt.h aprašoma atsitiktinių skaičių generavimo klasė.
8. Timer.h laiko skaičiavimo klasė.
9. CmakeLists.txt
10. run.bat paleidimo failas.
11. test.cpp faile surašyti testai.

1.2.1 Kas naujo?

- Sukurta projekto `Doxygen` dokumentacija.
- Sukurti `Unit Test`'ai, kurie testuoja Stud klasės konstruktorių, kopijavimo konstruktorių, *setter*'ius ir *output* operatorių. Naudojamas `GoogleTest` *framework*'as.
- *run.bat* failas sukompiliuoja kodą, sukuriamas ir *Vector.exe*, ir *ProjektasTest.exe*. Automatiškai paleidžiamas testų *exe*.

1.3 V1.5

1. main.cpp yra main failas.
2. MyLib3.h faile aprašytos bibliotekos.
3. Stud3.cpp faile surašytos visos funkcijos, realizuoti įvedimo/išvedimo operatoriai
4. Zmogus.h faile aprašyta bazinė klasė
5. Stud3.h faile aprašyta išvestinė (derived) klasė, jos konstruktoriai, funkcijų deklaracijos ir pnš.
6. Failu_kurimas.cpp faile yra failų generavimo ir duomenų nuskaitymo, išvedimo į failus funkcijos.
7. RandInt.h aprašoma atsitiktinių skaičių generavimo klasė.
8. Timer.h laiko skaičiavimo klasė.
9. CmakeLists.txt
10. run.bat paleidimo failas.

1.3.1 Kas naujo?

- Sukurta abstrakti bazinė klasė `Zmogus` su *protected* kintamaisiais: vardas, pavardė.
- `Stud` klasė tapo išvestine `Zmogus` klase.

1.3.2 # V1.2

1. main.cpp yra main failas.
2. MyLib3.h faile aprašytos bibliotekos.
3. Stud3.cpp faile surašytos visos funkcijos, realizuoti įvedimo/išvedimo operatoriai
4. Stud3.h faile aprašyta klasė, konstruktoriai, funkcijų deklaracijos ir pnš.
5. Failu_kurimas.cpp faile yra failų generavimo ir duomenų nuskaitymo, išvedimo į failus funkcijos.
6. RandInt.h aprašoma atsitiktinių skaičių generavimo klasė.
7. Timer.h laiko skaičiavimo klasė.
8. CmakeLists.txt
9. run.bat paleidimo failas.

1.3.3 Kas naujo?

- Realizuoti ir demonstruojami visi *Rule of three* metodai (*desrtuctor*, *copy constructor*, *copy assignment operator*).
- Perdengti įvesties ir išvesties metodai.
- Sukurti įvesties ir išvesties operatoriai.

Perdengti `output` ir `input` metodai, jie skiriasi savo parametrais:

- void output(const vector<Stud>& vec, const string& failoPav); - išveda duomenis į failą.
- void output(const vector<Stud>& vector1); - išveda duomenis į ekraną.
- void input(const string& failoVardas, vector<Stud>& studentai); - nuskaito duomenis iš failo.
- void input(const string& failas, int eil); - automatiškai sugeneruoja duomenis į failą.
- void input(); - duomenų įvedimas ranka.

Stud3.h realizuota nauja funkcija `demo`, kuri demonstruoja, kad veikia kopijavimo konstruktorius ir kopijavimo priskyrimo operatorius.

1.3.4 # V1.1

1. main.cpp yra main failas,
2. MyLib3.h faile aprašytos bibliotekos,
3. Stud3.cpp faile surašytos visos funkcijos,
4. Stud3.h faile aprašyta klasė ir funkcijų deklaracijos.
5. Failu_kurimas.cpp faile yra failų generavimo ir duomenų išvedimo į failus funkcijos.
6. RandInt.h aprašoma atsitiktinių skaičių generavimo klasė.
7. Timer.h laiko skaičiavimo klasė,
8. CmakeLists.txt
9. run.bat paleidimo failas.

1.3.5 Kas naujo?

- Programa realizuota naudojant `class`, vietoje `struct`
- Programa parašyta ir naudojimui su `vector`'iumi, ir su `list`'u
- Ir `struct`, ir `class` programos versijos ištirtos su optimizavimo flag'ais `-O1`, `-O2`, `-O3`

1.3.6 Testų vidurkiai:

Buvo atlikti 5 testai.
Lentelėse pateiktas vidutinis testų laikas sekundėmis naudojant `vector` ir 3 dalinimo į 2 grupes strategiją.

- 100 000 įrašų:

Funkcija	Struct_vector	Class_vector
Failo nuskaitymas	0.551681	0.596265
Rūšiavimas	0.042794	0.126591
Dalinimas į 2 grupes	0.003696	0.019929
"Sigma" išvedimas	0.087069	0.084964
"Beta" išvedimas	0.062781	0.056621
Viso testo trukmė	0.750380	0.884373

- 1 000 000 įrašų:

Funkcija	Struct_vector	Class_vector
Failo nuskaitymas	5.270314	5.751194
Rūšiavimas	0.518232	1.485370
Dalinimas į 2 grupes	0.046979	0.139557
"Sigma" išvedimas	0.733518	0.766893
"Beta" išvedimas	0.507979	0.530507
Viso testo trukmė	7.077026	8.673524

1.3.7 Analizė su −O1, −O2, −O3 flag'ais:

- Struct vector:

Flag	Greitis (s)	exe dydis (KB)
-O1	112.41	179.87
-O2	108.14	173.93
-O3	109.83	187.65

- Class vector:

Flag	Greitis (s)	exe dydis (KB)
-O1	147.91	174.36
-O2	136.50	168.03
-O3	136.35	181.86

- Struct list:

Flag	Greitis (s)	exe dydis (KB)
-O1	132.97	150.25
-O2	139.63	149.39
-O3	143.10	150.16

- Class list:

Flag	Greitis (s)	exe dydis (KB)
-O1	148.68	151.14
-O2	146.61	149.78
-O3	147.77	153.16

1.3.8 Tyrimo išvados:

1 000 000 ir 10 000 000 įrašų atveju, programa realizuota su `struct` veikia šiek tiek greičiau nei su `class`.
Programa realizuota su `vector` konteineriu su visais flag'ais veikia greičiau nei `list` programa, tačiau jos exe failai yra didesni.
Ir su `vector`'iumi, ir su `list`'u programos veikia greičiausiai su `-O2` flag'u. Išskyrus `class vector` programos versiją, jos `-O3` flag'as veikia nežymiai greičiau nei `-O2` flag'as.
Didžiausi exe failai yra `-O3` flag'o, mažiausi `-O2`.

- Taigi, atsižvelgiant į programos veikimo greitį ir exe failų dydį, optimaliausia būtų naudoti `vector struct` arba `list struct` programos versijas, naudojant `-O2` flag'ą.

1.3.9 # V1.0

1. main.cpp yra main failas,
2. MyLib3.h faile aprašytos bibliotekos,
3. Stud3.cpp faile surašytos visos funkcijos,
4. Stud3.h faile aprašyta struktūra ir funkcijų deklaracijos.
5. Failu_kurimas.cpp faile yra failų generavimo ir duomenų išvedimo į failus funkcijos.
6. RandInt.h aprašoma atsitiktinių skaičių generavimo klasė.
7. Timer.h laiko skaičiavimo klasė,
8. CmakeLists.txt
9. run.bat paleidimo failas.

1.3.10 Kas naujo?

- Programoje realizuotos 3 strategijos (naudotojas gali pasirinkti) studentų pagal galutinį balą skirstymui.
- Programa parašyta ir naudojimui su vector'iumi, ir su list'u.

1.3.11 Testų vidurkiai:

Buvo atlikti 8 testai.
Lentelėse pateiktas vidutinis testų laikas sekundėmis naudojant vector ir list.

- 1 000 įrašų:

Funkcija	2_strategija su vector	2_strategija su list	3_strategija su vector	3_strategija su list
Failo nuskaitymas	0.005733	0.005491	0.006346	0.006410
Rūšiavimas	0.000283	0.000089	0.000263	0.000096
Dalinimas į 2 grupes	***0.000210***	***0.000087***	***0.000069***	***0.000033***
"Sigma" išvedimas	0.001556	0.001523	0.002205	0.001856
"Beta" išvedimas	0.001008	0.000916	0.001464	0.001227
Viso testo trukmė	0.008792	0.008108	0.010349	0.009624

- 10 000 įrašų:

Funkcija	2_strategija su vector	2_strategija su list	3_strategija su vector	3_strategija su list
Failo nuskaitymas	0.046741	0.042193	0.054437	0.053289
Rūšiavimas	0.003621	0.001331	0.004881	0.002083
Dalinimas į 2 grupes	***0.002224***	***0.000858***	***0.000845***	***0.000472***
"Sigma" išvedimas	0.009532	0.007251	0.010891	0.012339
"Beta" išvedimas	0.007029	0.005353	0.008013	0.00839
Viso testo trukmė	0.069149	0.056989	0.079068	0.07657

- 100 000 įrašų:

Funkcija	2_strategija su vector	2_strategija su list	3_strategija su vector	3_strategija su list
Failo nuskaitymas	0.383984	0.332345	0.522110	0.512723
Rūšiavimas	0.041763	0.019096	0.06225	0.038709
Dalinimas į 2 grupes	***0.0242208***	***0.011417***	***0.007132***	***0.014610***
"Sigma" išvedimas	0.071508	0.063848	0.096222	0.103482
"Beta" išvedimas	0.046491	0.043425	0.072234	0.073484
Viso testo trukmė	0.567968	0.470132	0.759952	0.743010

- 1 000 000 įrašų:

Funkcija	2_strategija su vector	2_strategija su list	3_strategija su vector	3_strategija su list
Failo nuskaitymas	4.0932	3.350006	5.702961	5.391963
Rūšiavimas	0.526682	0.401233	0.724726	0.565133
Dalinimas į 2 grupes	***0.164642***	***0.141924***	***0.057141***	***0.114295***
"Sigma" išvedimas	0.723316	0.583383	0.938431	0.944535
"Beta" išvedimas	0.557543	0.398344	0.642816	0.654641
Viso testo trukmė	6.065383	4.874893	8.066082	7.670571

- 10 000 000 įrašų:

Funkcija	2_strategija su vector	2_strategija su list	3_strategija su vector	3_strategija su list
Failo nuskaitymas	44.880362	34.16475	55.102162	50.802437
Rūšiavimas	6.517658	6.104373	8.130297	8.220571
Dalinimas į 2 grupes	***1.941232***	***1.392186***	***0.576508***	***1.102128***
"Sigma" išvedimas	8.124385	6.21747	9.368383	9.161067
"Beta" išvedimas	5.698845	4.178442	6.60259	6.657571
Viso testo trukmė	67.3361	52.057225	79.779962	75.943775

1.3.12 Tyrimo išvados:

- Remiantis v0.3 versijos testų rezultatais antra studentu skirstymo pagal galutinį balą strategija veikia greičiau nei pirmoji.
- Visų dydžių failams trečia strategija (naudojamas partition algoritmas) veikia greičiausiai ir su vector'iais, ir su list'ais. Išimtis: 100 000 įrašų failas. Jam greičiausiai su list'u veikė antra strategija (laiko skirtumas labai nedidelis, apie 0.003s), su vector'iumi trečia.
- Naudojant vector'ius daug didesnis greičio skirtumas tarp antros ir trečios strategijos, nei naudojant list'us. Trečia strategija su vektoriais veikia maždaug 3 kartus greičiau.

1.3.13 # V0.3

1. Project_v0.3.cpp yra main failas,
2. MyLib3.h faile aprašytos bibliotekos,
3. Stud3.cpp faile surašytos visos funkcijos,
4. Stud3.h faile aprašyta struktūra ir funkcijų deklaracijos.

5. Failu_kurimas.cpp faile yra failų generavimo ir duomenų išvedimo į failus funkcijos.
6. RandInt.h aprašoma atsitiktinių skaičių generavimo klasė.
7. Timer.h laiko skaičiavimo klasė.

1.3.14 Kas naujo?

- Programa parašyta taip, kad vietoje vector naudotų list konteinerį informacijai apie studentus saugoti.
- Duomenis įvedus ranka, šalia studentų duomenų yra išvedamas to studento saugojimo atmintyje adresas.
- Atliekami programos greičio testai, jų vidurkiai lyginami su v0.2 versijos rezultatais (t.y. programa naudoja vector konteinerį).

1.3.15 Testų vidurkiai:

Buvo atlikti 8 testai.
Lentelėse pateiktas vidutinis testų laikas sekundėmis naudojant vector ir list.

- 1 000 įrašų:

Funkcija	Vector	List
Failo nuskaitymas	0.007049	0.006176
Rūšiavimas	0.000362	0.000100
Dalinimas į 2 grupes	0.000218	0.000480
"Sigma" išvedimas	0.004299	0.002947
"Beta" išvedimas	0.001367	0.001585
Viso testo trukmė	0.013487	0.011256

- 10 000 įrašų:

Funkcija	Vector	List
Failo nuskaitymas	0.065592	0.061209
Rūšiavimas	0.004222	0.001590
Dalinimas į 2 grupes	0.002387	0.004416
"Sigma" išvedimas	0.012350	0.013929
"Beta" išvedimas	0.008186	0.008271
Viso testo trukmė	0.092739	0.089417

- 100 000 įrašų:

Funkcija	Vector	List
Failo nuskaitymas	0.588068	0.527883
Rūšiavimas	0.045850	0.035308
Dalinimas į 2 grupes	0.033737	0.058177
"Sigma" išvedimas	0.101996	0.095644
"Beta" išvedimas	0.070402	0.064387
Viso testo trukmė	0.840056	0.781400

- 1 000 000 įrašų:

Funkcija	Vector	List
Failo nuskaitymas	5.833982	5.02694
Rūšiavimas	0.528940	0.512247
Dalinimas į 2 grupes	0.290187	0.497361
"Sigma" išvedimas	0.861029	0.839495
"Beta" išvedimas	0.626815	0.592574
Viso testo trukmė	8.140956	7.46862

- 10 000 000 įrašų:

Funkcija	Vector	List
Failo nuskaitymas	57.355937	52.606862
Rūšiavimas	6.242971	7.908942
Dalinimas į 2 grupes	3.301593	5.202556
"Sigma" išvedimas	8.64999	9.017396
"Beta" išvedimas	5.985857	5.814621
Viso testo trukmė	81.523975	80.550325

1.3.16 Tyrimo išvados:

Labiausiai skiriasi failo nuskaitymo ir dalinimo į dvi grupes testo vidurkiai su visų dydžių failais.

- Su list'ais nuskaitymas veikia greičiau nei su vektoriais.
- Tačiau duomenų skirstymas į dvi grupes list'uose vyksta beveik dvigubai lėčiau nei vector'iuose su visų dydžių failais.

1.3.17 # V0.2

1. Project_v0.2.cpp yra main failas,
2. MyLib.h faile aprašytos bibliotekos,
3. Stud.cpp faile surašytos visos funkcijos,
4. Stud.h faile aprašyta struktūra ir funkcijų deklaracijos.
5. Failu_kurimas.cpp faile yra failų generavimo ir duomenų išvedimo į failus funkcijos.
6. RandInt.h aprašoma atsitiktinių skaičių generavimo klasė.
7. Timer.h laiko skaičiavimo klasė.

1.3.18 Kas naujo?

Programa leidžia:

- pasirinkti ar vartotjas nori generuoti naujus 5 failus (1000, 10000, 1000000, 10000000 įrašų)
- pasirinkti pagal ką būtų rūšiuojami išvedimo duomenys (vardą, pavardę ar galutinį balą)
- apskaičiuoti failų generavimo laiką
- apskaičiuoti failų rūšiavimo, nuskaitymo ir išvedimo laiką
- padalinti studentus į dvi grupes pagal galutinį balą ("sigma" - galutinis balas $>=$ 5, "beta" - galutinis balas $<$ 5).

1.3.19 Testų vidurkiai:

Buvo atlikti 8 testai.

- 1 000 įrašų:

Funkcija	Vidutinis laikas (s)
Failo nuskaitymas	0.007049
Rūšiavimas	0.000362
Dalinimas į 2 grupes	0.000218
"Sigma" išvedimas	0.004299
"Beta" išvedimas	0.001367
Viso testo trukmė	0.013487

- 10 000 įrašų:

Funkcija	Vidutinis laikas (s)
Failo nuskaitymas	0.065592
Rūšiavimas	0.004222
Dalinimas į 2 grupes	0.002387
"Sigma" išvedimas	0.012350
"Beta" išvedimas	0.008186
Viso testo trukmė	0.092739

- 100 000 įrašų:

Funkcija	Vidutinis laikas (s)
Failo nuskaitymas	0.588068
Rūšiavimas	0.045850
Dalinimas į 2 grupes	0.033737
"Sigma" išvedimas	0.101996
"Beta" išvedimas	0.070402
Viso testo trukmė	0.840056

- 1 000 000 įrašų:

Funkcija	Vidutinis laikas (s)
Failo nuskaitymas	5.833982
Rūšiavimas	0.528940
Dalinimas į 2 grupes	0.290187
"Sigma" išvedimas	0.861029
"Beta" išvedimas	0.626815
Viso testo trukmė	8.140956

- 10 000 000 įrašų:

Funkcija	Vidutinis laikas (s)
Failo nuskaitymas	57.355937
Rūšiavimas	6.242971
Dalinimas į 2 grupes	3.301593
"Sigma" išvedimas	8.64999
"Beta" išvedimas	5.985857
Viso testo trukmė	81.523975

1.3.20 # V.1

1. Project_v0.1.cpp yra main failas,
2. MyLib.h faile aprašytos bibliotekos,
3. Stud.cpp faile surašytos visos funkcijos,
4. Stud.h faile aprašyta struktūra ir funkcijų deklaracijos.

1.3.21 Kas naujo?

Programa leidžia vartotojui pasirinkti atsitiktinai generuojamų namų darbų kiekį. Duomenis galima nuskaityti iš failo arba įvesti pačiam. Išimčių valdymas:

- visi "0", "1" atsakymai į užklausas nebūtų raidė ar kitas skaičius;
- įmanoma atidaryti tekstinį failą;
- tekstinio failo eilutėje yra įvestų namų darbų ar egzamino rezultatų;
- tekstiniame faile įvesti ND ir egzamino rezultatai yra tarp 1 ir 10;
- įvestas studentų ar atsitiktinai generuojamų namų darbų skaičius nėra raidė ar kitas simbolis;
- įvesti namų darbų ir egzamino rezultatas yra tarp 1 ir 10. Išvedimo duomenys surūšiuoti pagal pavardę. Papildomai užkomentuota yra funkcija, kuri išvestų iš tekstinio failo nuskaitytus duomenis patikrai. Taip pat užkomentuotos eilutės, kurios išvestų atsitiktinai sugeneruotus namų darbų ir egzamino rezultatus.

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 RandInt Class Reference

`#include <RandInt.h>`

Public Member Functions

- RandInt (int low, int high)
- int operator() ()

5.1.1 Constructor & Destructor Documentation

5.1.1.1 RandInt()

```
RandInt::RandInt (
            int low,
            int high)  [inline]
```

5.1.2 Member Function Documentation

5.1.2.1 operator()()

```
int RandInt::operator() ()  [inline]
```

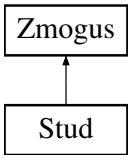
The documentation for this class was generated from the following file:

- Vector/include/RandInt.h

5.2 Stud Class Reference

`#include <Stud3.h>`

Inheritance diagram for Stud:



Public Member Functions

- Stud (string v="", string p="", vector< int > nd={}, int egz=0)
- Stud (const Stud &c)
- Stud & operator= (const Stud &op)
- ~Stud ()

- double getGalutinisVid () const
- double getGalutinisMed () const
- vector< int > getNd () const
- int getEgz () const
- void setNd (const vector< int > &nd)
- void setEgz (int egz)
- void output (const vector< Stud > &vec, const string &failoPav)
- void output (const vector< Stud > &vector1)
- void input (const string &failoVardas, vector< Stud > &studentai)
- void input (const string &failas, int eil)
- void input ()
- void skaiciuotiGalutiniBala ()
- void clean ()
- void demo (int demo)
- void kasAs () const

Public Member Functions inherited from Zmogus

- Zmogus (string v="", string p="")
- Zmogus (const Zmogus &copy)
- Zmogus & operator= (const Zmogus &op)
- string getVardas () const
- string getPavarde () const
- void setVardas (const string &v)
- void setPavarde (const string &p)
- ~Zmogus ()

Friends

- istream & operator>> (istream &is, Stud &student)
- ostream & operator<< (ostream &out, const Stud &student)

Additional Inherited Members

Protected Attributes inherited from Zmogus

- string vardas
- string pavarde

5.2.1 Constructor & Destructor Documentation

5.2.1.1 Stud() [1/2]

```
Stud::Stud (
            string v = "",
            string p = "",
            vector< int > nd = {},
            int egz = 0)  [inline]
```

5.2.1.2 Stud() [2/2]

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

5.2.1.3 ~Stud()

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

5.2.2 Member Function Documentation

5.2.2.1 clean()

```
void Stud::clean ()
```

5.2.2.2 demo()

```
void Stud::demo (
            int demo)
```

5.2.2.3 getEgz()

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

5.2.2.4 getGalutinisMed()

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

5.2.2.5 getGalutinisVid()

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

5.2.2.6 getNd()

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

5.2.2.7 input() [1/3]

```
void Stud::input ()
```

5.2.2.8 input() [2/3]

```
void Stud::input (
            const string & failas,
            int eil)
```

5.2.2.9 input() [3/3]

```
void Stud::input (
            const string & failoVardas,
            vector< Stud > & studentai)
```

5.2.2.10 kasAs()

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

Implements Zmogus.

5.2.2.11 operator=()

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

5.2.2.12 output() [1/2]

```
void Stud::output (
            const vector< Stud > & vec,
            const string & failoPav)
```

5.2.2.13 output() [2/2]

```
void Stud::output (
            const vector< Stud > & vector1)
```

5.2.2.14 setEgz()

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

5.2.2.15 setNd()

```
void Stud::setNd (
            const vector< int > & nd)  [inline]
```

5.2.2.16 skaiciuotiGalutiniBala()

```
void Stud::skaiciuotiGalutiniBala ()
```

5.2.3 Friends And Related Symbol Documentation

5.2.3.1 operator<<

```
ostream & operator<< (
            ostream & out,
            const Stud & student)  [friend]
```

5.2.3.2 operator>>

```
istream & operator>> (
            istream & is,
            Stud & student)  [friend]
```

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

- Vector/include/Stud3.h
- Vector/src/Failu_kurimas.cpp
- Vector/src/Stud3.cpp

5.3 Timer Class Reference

`#include <Timer.h>`

Public Member Functions

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

5.3.1 Constructor & Destructor Documentation

5.3.1.1 Timer()

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

5.3.2 Member Function Documentation

5.3.2.1 elapsed()

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

5.3.2.2 reset()

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

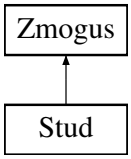
The documentation for this class was generated from the following file:

- Vector/include/Timer.h

5.4 Zmogus Class Reference

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

Inheritance diagram for Zmogus:



Public Member Functions

- Zmogus (string v="", string p="")
- Zmogus (const Zmogus &copy)
- Zmogus & operator= (const Zmogus &op)
- string getVardas () const
- string getPavarde () const
- void setVardas (const string &v)
- void setPavarde (const string &p)
- virtual void kasAs () const =0
- ~Zmogus ()

Protected Attributes

- string vardas
- string pavarde

5.4.1 Constructor & Destructor Documentation

5.4.1.1 Zmogus() [1/2]

```
Zmogus::Zmogus (
            string v = "",
            string p = "")  [inline]
```

5.4.1.2 Zmogus() [2/2]

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

5.4.1.3 ~Zmogus()

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

5.4.2 Member Function Documentation

5.4.2.1 getPavarde()

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

5.4.2.2 getVardas()

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

5.4.2.3 kasAs()

```
virtual void Zmogus::kasAs () const  [pure virtual]
```
Implemented in Stud.

5.4.2.4 operator=()

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

5.4.2.5 setPavarde()

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

5.4.2.6 setVardas()

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

5.4.3 Member Data Documentation

5.4.3.1 pavarde

```
string Zmogus::pavarde  [protected]
```

5.4.3.2 vardas

```
string Zmogus::vardas  [protected]
```
The documentation for this class was generated from the following file:

- Vector/include/Zmogus.h

Chapter 6

File Documentation

6.1 readme.md File Reference

6.2 Vector/include/MyLib3.h File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <iomanip>
#include <algorithm>
#include <fstream>
#include <sstream>
#include <ios>
#include <limits>
#include <cstdlib>
#include <stdexcept>
#include <exception>
#include <random>
#include <ctime>
#include <chrono>
```

6.3 MyLib3.h

Go to the documentation of this file.

```
00001 #ifndef MYLIB_H_INCLUDED
00002 #define MYLIB_H_INCLUDED
00003
00004 #include <iostream>
00005 #include <string>
00006 #include <vector>
00007 #include <iomanip>
00008 #include <algorithm>
00009 #include <fstream>
00010 #include <sstream>
00011 #include <ios>
00012 #include <limits>
00013 #include <cstdlib>
00014 #include <stdexcept>
00015 #include <exception>
00016 #include <random>
00017 #include <ctime>
00018 #include <chrono>
00019
00020
00021 using std::endl;
00022 using std::cout;
00023 using std::cin;
00024 using std::left;
00025 using std::right;
00026 using std::setw;
00027 using std::setprecision;
```

```
00028 using std::fixed;
00029 using std::sort;
00030 using std::max;
00031 using std::ifstream;
00032 using std::streamsize;
00033 using std::numeric_limits;
00034 using std::string;
00035 using std::vector;
00036 using std::runtime_error;
00037 using std::istringstream;
00038 using std::cerr;
00039 using std::exception;
00040 using std::ofstream;
00041 using std::to_string;
00042 using std::partition;
00043 using std::make_move_iterator;
00044 using std::istream;
00045 using std::ostream;
00046
00047
00048 #endif // MYLIB_H_INCLUDED
```

6.4 Vector/include/RandInt.h File Reference

Classes

- class RandInt

6.5 RandInt.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 class RandInt {
00004 public:
00005     RandInt(int low, int high) : mt{ rd() }, dist{ low, high } { }
00006     int operator()() { return dist(mt); } // generuok inta
00007 private:
00008     std::random_device rd;
00009     std::mt19937 mt;
00010     std::uniform_int_distribution<int> dist;
00011 };
```

6.6 Vector/include/Stud3.h File Reference

```
#include "MyLib3.h"
#include "Zmogus.h"
```

Classes

- class Stud

Functions

- void kategorijos3 (vector< Stud > &vector1, vector< Stud > &beta)
- bool lygintiVardas (const Stud &a, const Stud &b)
- bool lygintiPavarde (const Stud &a, const Stud &b)
- bool lygintiGalutinis (const Stud &a, const Stud &b)
- void sortByChoice (vector< Stud > &vec, int b)

6.6.1 Function Documentation

6.6.1.1 kategorijos3()

```
void kategorijos3 (
            vector< Stud > & vector1,
            vector< Stud > & beta)
```

6.6.1.2 lygintiGalutinis()

```
bool lygintiGalutinis (
            const Stud & a,
            const Stud & b)
```

6.6.1.3 lygintiPavarde()

```
bool lygintiPavarde (
            const Stud & a,
            const Stud & b)
```

6.6.1.4 lygintiVardas()

```
bool lygintiVardas (
            const Stud & a,
            const Stud & b)
```

6.6.1.5 sortByChoice()

```
void sortByChoice (
            vector< Stud > & vec,
            int b)
```

6.7 Stud3.h

Go to the documentation of this file.

```
00001 #ifndef STUD_H_INCLUDED
00002 #define STUD_H_INCLUDED
00003 #include "MyLib3.h"
00004 #include "Zmogus.h"
00005
00006 class Stud : public Zmogus {
00007 private:
00008     vector <int> nd_;
00009     int egz_;
00010     double galutinisVid_, galutinisMed_;
00011
00012 public:
00013
00014     Stud(string v = "", string p = "", vector<int> nd = {}, int egz = 0) : Zmogus(v, p), nd_(nd),
      egz_(egz), galutinisVid_(0), galutinisMed_(0) {}
00015
00016
00017     //1.copy constructor
00018     Stud(const Stud& c) : Zmogus(c), nd_(c.nd_), egz_(c.egz_), galutinisVid_(c.galutinisVid_),
      galutinisMed_(c.galutinisMed_) {};
00019
00020     //2.copy assignment operator
00021     Stud& operator=(const Stud& op) {
00022         if (this != &op) {
00023             Zmogus::operator=(op);
00024             nd_ = op.nd_;
00025             egz_ = op.egz_;
00026             galutinisVid_ = op.galutinisVid_;
00027             galutinisMed_ = op.galutinisMed_;
00028         }
00029         return *this;
00030     }
00031     //3.destructor
00032     ~Stud() { nd_.clear(); }
00033
00034     //getters:
00035     inline double getGalutinisVid() const { return galutinisVid_; }
00036     inline double getGalutinisMed() const { return galutinisMed_; }
00037     vector<int> getNd() const { return nd_; }
00038     int getEgz() const { return egz_; }
00039
00040     //setters:
00041
00042     void setNd(const vector<int>& nd) { nd_ = nd; }
00043     void setEgz(int egz) { egz_ = egz; }
00044
00045     //input/output operators:
```

```
00046     friend istream& operator»(istream& is, Stud& student);
00047     friend ostream& operator«(ostream& out, const Stud& student);
00048
00049     //perdengti metodai:
00050     void output(const vector<Stud>& vec, const string& failoPav);
00051     void output(const vector<Stud>& vector1);
00052
00053     void input(const string& failoVardas, vector<Stud>& studentai);
00054     void input(const string& failas, int eil);
00055     void input();
00056
00057     void skaiciuotiGalutiniBala();
00058     void clean();
00059
00060     void demo(int demo);
00061     void kasAs() const { cout « "As esu studentas" « endl; }
00062 };
00063
00064 static double skaiciuotiNdVid(const vector <int>& nd);
00065 static double skaiciuotiNdMed(vector <int>& nd);
00066
00067 void kategorijos3(vector<Stud>& vector1, vector<Stud>& beta);
00068
00069 bool lygintiVardas(const Stud& a, const Stud& b);
00070 bool lygintiPavarde(const Stud& a, const Stud& b);
00071 bool lygintiGalutinis(const Stud& a, const Stud& b);
00072 void sortByChoice(vector<Stud>& vec, int b);
00073
00074 #endif //STUD_H_INCLUDED
```

6.8 Vector/include/Timer.h File Reference

Classes

- class Timer

6.9 Timer.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 class Timer {
00004 private:
00005     std::chrono::time_point<std::chrono::high_resolution_clock> start;
00006 public:
00007     Timer() : start{ std::chrono::high_resolution_clock::now() } {}
00008     void reset() {
00009         start = std::chrono::high_resolution_clock::now();
00010     }
00011     double elapsed() const {
00012         return std::chrono::duration<double>(std::chrono::high_resolution_clock::now() -
      start).count();
00013     }
00014 };
```

6.10 Vector/include/Zmogus.h File Reference

```
#include "MyLib3.h"
```

Classes

- class Zmogus

6.11 Zmogus.h

Go to the documentation of this file.

```
00001 #ifndef ZMOGUS_H_INCLUDED
00002 #define ZMOGUS_H_INCLUDED
00003 #include "MyLib3.h"
00004
00005 class Zmogus {
```

```
00006 protected:
00007     string vardas, pavarde;
00008 public:
00009     Zmogus(string v = "", string p = "") : vardas(v), pavarde(p) {}
00010
00011     Zmogus(const Zmogus& copy) : vardas(copy.vardas), pavarde(copy.pavarde) {}
00012
00013     Zmogus& operator=(const Zmogus& op) {
00014         if (this != &op) {  // Apsauga nuo saves priskyrimo
00015             vardas = op.vardas;
00016             pavarde = op.pavarde;
00017         }
00018         return *this;
00019     }
00020
00021     string getVardas() const { return vardas; }
00022     string getPavarde() const { return pavarde; }
00023
00024
00025     void setVardas(const string& v) { vardas = v; }
00026     void setPavarde(const string& p) { pavarde = p; }
00027
00028     virtual void kasAs() const = 0;
00029     ~Zmogus() { vardas.clear(); }
00030
00031 };
00032
00033 #endif //ZMOGUS_H_INCLUDED
```

6.12 Vector/src/Failu_kurimas.cpp File Reference

```
#include <iostream>
#include "MyLib3.h"
#include "Stud3.h"
#include "RandInt.h"
#include "Timer.h"
```

6.13 Vector/src/main.cpp File Reference

```
#include <iostream>
#include "MyLib3.h"
#include "Stud3.h"
#include "Timer.h"
```

Functions

- int main ()

6.13.1 Function Documentation

6.13.1.1 main()

```
int main ()
```

6.14 Vector/src/Stud3.cpp File Reference

```
#include "Stud3.h"
#include "RandInt.h"
```

Functions

- double skaiciuotiNdVid (const vector< int > &nd)
- double skaiciuotiNdMed (vector< int > &nd)

- void kategorijos3 (vector< Stud > &vector1, vector< Stud > &beta)
- bool lygintiVardas (const Stud &a, const Stud &b)
- bool lygintiPavarde (const Stud &a, const Stud &b)
- bool lygintiGalutinis (const Stud &a, const Stud &b)
- void sortByChoice (vector< Stud > &vec, int b)
- istream & operator>> (istream &is, Stud &student)
- ostream & operator<< (ostream &out, const Stud &student)

6.14.1 Function Documentation

6.14.1.1 kategorijos3()

```
void kategorijos3 (
            vector< Stud > & vector1,
            vector< Stud > & beta)
```

6.14.1.2 lygintiGalutinis()

```
bool lygintiGalutinis (
            const Stud & a,
            const Stud & b)
```

6.14.1.3 lygintiPavarde()

```
bool lygintiPavarde (
            const Stud & a,
            const Stud & b)
```

6.14.1.4 lygintiVardas()

```
bool lygintiVardas (
            const Stud & a,
            const Stud & b)
```

6.14.1.5 operator<<()

```
ostream & operator<< (
            ostream & out,
            const Stud & student)
```

6.14.1.6 operator>>()

```
istream & operator>> (
            istream & is,
            Stud & student)
```

6.14.1.7 skaiciuotiNdMed()

```
double skaiciuotiNdMed (
            vector< int > & nd)
```

6.14.1.8 skaiciuotiNdVid()

```
double skaiciuotiNdVid (
            const vector< int > & nd)
```

6.14.1.9 sortByChoice()

```
void sortByChoice (
            vector< Stud > & vec,
            int b)
```

6.15 Stud3.cpp

Go to the documentation of this file.

```
00001 #include "Stud3.h"
00002 #include "RandInt.h"
00003
00004
00005 double skaiciuotiNdVid(const vector <int>& nd) {
00006     double sum = 0;
00007     for (int i = 0; i < nd.size(); i++) {
00008         sum += nd.at(i);
00009     }
00010     return sum / nd.size();
00011 }
00012 double skaiciuotiNdMed(vector <int>& nd) {
00013     sort(nd.begin(), nd.end());
00014     size_t size = nd.size();
00015     if (size % 2 == 0) {
00016         return (nd.at(size / 2 - 1) + nd.at(size / 2)) / 2.0;
00017     }
00018     else {
00019         return nd.at(size / 2);
00020     }
00021 }
00022
00023 void Stud::skaiciuotiGalutiniBala() {
00024
00025     if (nd_.empty()) {
00026         cerr « "Klaida: ND balai yra tusciame vektoriuje!" « endl;
00027         throw runtime_error("ND balai yra tusciame vektoriuje");
00028     }
00029     galutinisVid_ = 0.4 * skaiciuotiNdVid(nd_) + 0.6 * egz_;
00030     galutinisMed_ = 0.4 * skaiciuotiNdMed(nd_) + 0.6 * egz_;
00031
00032     galutinisVid_ = round(galutinisVid_ * 100.0) / 100.0;
00033     galutinisMed_ = round(galutinisMed_ * 100.0) / 100.0;
00034 }
00035
00036 void kategorijos3(vector<Stud>& vector1, vector<Stud>& beta) {
00037     auto partitionPoint = partition(vector1.begin(), vector1.end(), [](const Stud& s) {
00038         return s.getGalutinisVid() >= 5;
00039         });
00040
00041     beta.insert(beta.end(), make_move_iterator(partitionPoint), make_move_iterator(vector1.end()));
00042
00043     vector1.erase(partitionPoint, vector1.end());
00044 }
00045
00046 bool lygintiVardas(const Stud& a, const Stud& b) {
00047     return a.getVardas() < b.getVardas();
00048 }
00049 bool lygintiPavarde(const Stud& a, const Stud& b) {
00050     return a.getPavarde() < b.getPavarde();
00051 }
00052 bool lygintiGalutinis(const Stud& a, const Stud& b) {
00053     return a.getGalutinisVid() < b.getGalutinisVid();
00054 }
00055 void sortByChoice(vector<Stud>& vec, int b) {
00056     if (b == 0) {
00057         sort(vec.begin(), vec.end(), lygintiVardas);
00058     }
00059     else if (b == 1) {
00060         sort(vec.begin(), vec.end(), lygintiPavarde);
00061     }
00062     else if (b == 2) {
00063         sort(vec.begin(), vec.end(), lygintiGalutinis);
00064     }
00065 }
00066
00067 //is terminalo
00068 void Stud::input() {
00069     constexpr int max = 10;
00070     RandInt rnd{ 1, max };
00071     cout « "Input Name, Surname:" « endl;
00072     cin » vardas » pavarde;
00073
00074     cout « "Do you want randomized ND and Exam scores (0 - no, 1 - yes)?" « endl;
```

```
00075     int ats;
00076     try {
00077         cin » ats;
00078         if (cin.fail() || (ats != 0 && ats != 1)) {
00079             throw runtime_error("Error: wrong input");
00080         }
00081         if (ats == 0) {
00082             cout « "Input ND scores (press non numeric symbol and ENTER to finish):" « endl;
00083             int paz;
00084 
00085             while (cin » paz) {
00086                 if (cin.fail() || (paz < 1 || paz > 10)) {
00087                     throw runtime_error("Error: invalid ND input");
00088                 }
00089                 else {
00090                     nd_.push_back(paz);
00091                 }
00092             }
00093 
00094             cin.clear();
00095             cin.ignore(numeric_limits<streamsize>::max(), '\n');
00096 
00097             cout « "Input Exam score: ";
00098             cin » egz_;
00099             if (egz_ < 1 || egz_ > 10) {
00100                 throw runtime_error("Error: invalid Exam input");
00101             }
00102         }
00103         else if (ats == 1) {
00104             cout « "How many ND scores should program randomize?" « endl;
00105             int ndSk;
00106             cin » ndSk;
00107             if (cin.fail()) {
00108                 throw runtime_error("Error: invalid input");
00109             }
00110 
00111             for (int i = 0; i < ndSk; i++) {
00112                 nd_.push_back(rnd());
00113             }
00114             egz_ = rnd();
00115         }
00116     }
00117     catch (exception& e) {
00118         cerr « e.what() « endl;
00119         exit(EXIT_FAILURE);
00120     }
00121 }
00122 
00123 //i terminala
00124 void Stud::output(const vector <Stud>& vector1) {
00125     cout « setw(15) « left « "Name" « setw(15) « left « "Surname" « setw(30) « left « "Final average
      score (vid.)" « setw(10) « right « "Adress" « endl;
00126     for (const Stud& student : vector1) {
00127         cout « student « " "« setw(40) « right « &student « endl;
00128     }
00129 }
00130 
00131 istream& operator»(istream& is, Stud& student) {
00132     string vardas, pavarde, x;
00133     vector<int> nd;
00134     int score, egz;
00135     is » vardas » pavarde;
00136     student.setVardas(vardas);
00137     student.setPavarde(pavarde);
00138 
00139     nd.clear();
00140 
00141     while (is » x) {
00142         score = std::stoi(x);
00143         nd.push_back(score);
00144     }
00145 
00146     for (int i = 0; i < nd.size(); i++) {
00147         if ((nd.at(i) < 1 || nd.at(i) > 10)) {
00148             throw runtime_error("Error: ND score must be between 1 and 10");
00149         }
00150     }
00151     if (nd.empty()) {
00152         throw runtime_error("Error: no ND scores found");
00153     }
00154 
00155     egz = nd.back();
00156 
00157     nd.pop_back();
00158     student.setEgz(egz);
00159     student.setNd(nd);
00160 
```

```
00161     return is;
00162 }
00163
00164 ostream& operator«(ostream& out, const Stud& student) {
00165
00166     out « setw(15) « left « student.vardas « setw(15) « left « student.pavarde « setw(5) « right «
      student.galutinisVid_;
00167
00168     return out;
00169 }
00170
00171
00172 void Stud::clean() {
00173     vardas.clear();
00174     pavarde.clear();
00175     nd_.clear();
00176 }
00177
00178 void Stud::demo(int demo) {
00179     if (demo == 1) {
00180         Stud obj1("Zigmantas", "Alonis", { 7, 5, 9 }, 8);
00181         obj1.skaiciuotiGalutiniBala();
00182
00183         //1.copy constructor
00184         //Stud obj2 = obj1;
00185         Stud obj2(obj1);
00186
00187         //2.copy assignment operator
00188         Stud obj3;
00189         obj3 = obj1;
00190
00191         cout « "Obj1:" « endl;
00192         cout « obj1 « endl;
00193         cout « "Obj2:" « endl;
00194         cout « obj2 « endl;
00195         cout « "Obj3:" « endl;
00196         cout « obj3 « endl;
00197     }
00198 }
```

6.16 Vector/Tests/test.cpp File Reference

```
#include "gtest/gtest.h"
#include "../include/Stud3.h"
#include "../src/Stud3.cpp"
```

Functions

- TEST (StudTest, Constructor)
- TEST (StudTest, OutputOp)
- TEST (StudTest, Setters)
- TEST (StudTest, CopyConstructor)
- int main (int argc, char ∗∗argv)

6.16.1 Function Documentation

6.16.1.1 main()

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

6.16.1.2 TEST() [1/4]

```
TEST (
            StudTest ,
            Constructor )
```

EXPECT_EQ(student.getNd(), vector<int>{});

6.16.1.3 TEST() [2/4]

```
TEST (
            StudTest ,
            CopyConstructor )
```

6.16.1.4 TEST() [3/4]

```
TEST (
            StudTest ,
            OutputOp )
```

6.16.1.5 TEST() [4/4]

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

Index