ObjektinisProgramavimas

v2.0

Generated by Doxygen 1.10.0

Chapter 1

ObjektinisProgramavimas

1.1 Release'ai

1. V.pradinė: Sukurtas programos karkasas. Naudotojas gali įvesti studentų kiekį, jų duomenis (vardą, pavardę, pažymius) ir ekrane matyti atspausdintus studento duomenis su apskaičiuotu galutiniu balu.
2. V0.1: Programa papildyta taip, kad studentų skaičius ir namų darbų skaičius nėra žinomi iš anksto. Pridėtas dar vienas programos failas (viename faile studentams saugoti naudojame C masyvus, kitame - std::vector konteinerius).
3. V0.2: Programoje atsirado galimybė nuskaityti duomenis iš failo, bei juos išrykiuoti.
4. V0.3: Funkcijos ir jų antraštės perkeltos į atskirus .cpp ir .h failus. Pridėtas išimčių valdymas.
5. V0.4: Pridėta failų generavimo funkcija. Pridėtas studentų rūšiavimas į atskirus konteinerius ir failus, atsižvelgiant į jų galutinius balus. Atlikti du programos spartos tyrimai.
6. V1.0: Atliktas programos testavimas su skirtingais konteineriais (Vector, List ir Deque). Taip pat naudojant skirtingus algoritmus, atliktas studentų skirstymas į dvi grupes testavimas.

1.2 Naudojimosi instrukcija

Norint naudoti 5 arba 6 parinktį (darbą su failas), pirmiausia turite susigeneruoti failus naudodami funkciją generuotiFaila().

1. Paleisti programą
2. Sekti programoje nurodomus žingsnius priklausomai nuo to, kaip jūs norite vykdyti programą.
3. Gauti studentų rezultatus ekrane arba faile (priklausomai nuo to, kokį išvedimo būdą jūs pasirinkote).

1.3 Programos diegimo ir paleidimo instrukcija

1. Privaloma turėti įsidiegus "MinGW" kompiliatorių ir "Make" - automatizavimo įrankį, kuris kuria vykdomąsias programas (Šis įrankis dažniausiai būna automatiškai instaliuotas Linux ir MacOS sistemose). Atsisiųsti MinGW galite čia: `MinGW` Pamoka, kaip atsisiųsti "Make" Windows naudotojams: `Make`
2. Atsisiųskite programos šaltinio kodą iš mūsų repozitorijos.
3. Atsidarę terminalą, naviguokite į atsisiųstos programos aplanką.
4. Įvykdykite komandą: make "konteineris" (vietoj "konteineris" įrašykite, su kokio tipo konteineriu norite testuoti programą: Vector, List ar Deque).
5. Tuomet terminale įrašykite ./mainVector, ./mainList arba ./mainDeque, kad paleistumėte norimą programą Linux sistemoje arba mainVector.exe, mainList.exe ar mainDeque.exe Windows sistemoje.

1.4 Testavimo parametrai

CPU: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz RAM: 16GB SSD: Micron NVMe 512GB

1.5 Darbo su vektoriais rezultatai, naudojant 1 rūšiavimo strategiją

Studentu skaicius	Failo generavimo trukme	Duomenu nuskaitymo trukme	Duomenu rikiavimo trukme	Duomenu skirstymo trukme	Duomenu isvedimo i failus trukme	Viso testo trukme
1000	0.015606	0.003097	0.000917	0.000369	0.006659	0.011042
10000	0.066734	0.021904	0.007164	0.003322	0.047934	0.080324
100000	0.579735	0.194111	0.096998	0.020499	0.384902	0.69651
1000000	5.70716	1.79173	1.33658	0.207118	3.42451	6.75994
10000000	56.1039	17.711	16.2768	1.79619	34.9862	70.7702

1.6 Darbo su list'ais rezultatai, naudojant 1 rūšiavimo strategiją

Studentu skaicius	Duomenu nuskaitymo trukme	Duomenu rikiavimo trukme	Duomenu skirstymo trukme	Viso testo trukme
1000	0.012964	0.000793	0.003106	0.031365
10000	0.079438	0.003847	0.011004	0.145519
100000	0.284524	0.056278	0.10717	0.837474
1000000	3.03823	0.999937	1.05213	9.01883
10000000	29.4128	15.8629	14.6962	101.946

1.7 Darbo su deque'ais rezultatai, naudojant 1 rūšiavimo strategiją

Studentu skaicius	Duomenu nuskaitymo trukme	Duomenu rikiavimo trukme	Duomenu skirstymo trukme	Viso testo trukme
1000	0.004163	0.002462	0.000821	0.017254
10000	0.024412	0.021921	0.006072	0.090725
100000	0.173054	0.348351	0.067596	1.04723
1000000	1.91702	4.24411	0.651649	11.0898
10000000	16.7831	54.2263	15.4799	124.884

1.8 Vektoriuje esanciu studentu rikiavimo rezultatai naudojant 2 strategija

Studentu skaicius	Duomenu skirstymo trukme
1000	0.009835
10000	0.876002

Studentu skaicius	Duomenu skirstymo trukme
100000	91.3894
1000000	1000+
10000000	10000+

Testuojant faila su 1 000 000 studentu skirstymo laikas toks ilgas, jog tiesiog neverta laukti pabaigos

1.9 Vektoriuje esanciu studentu rikiavimo rezultatai naudojant 3 strategija

Studentu skaicius	Duomenu skirstymo trukme
1000	0.000356
10000	0.003718
100000	0.02288
1000000	0.236657
10000000	2.38154

1.10 List'e esanciu studentu rikiavimo rezultatai naudojant 2 strategija

Studentu skaicius	Duomenu skirstymo trukme
1000	0.000423
10000	0.007395
100000	0.046986
1000000	0.483659
10000000	4.88548

Neapsakomai greičiau, nei naudojant 2 strategiją su vektoriais

1.11 List'e esanciu studentu rikiavimo rezultatai naudojant 3 strategija

Studentu skaicius	Duomenu skirstymo trukme
1000	0.00587
10000	0.008309
100000	0.079064
1000000	0.826309
10000000	8.46289

Programa vykdoma lėčiau, nei naudojant 2 strategiją

1.12 Deque esanciu studentu rikiavimo rezultatai naudojant 2 strategija

Studentu skaicius	Duomenu skirstymo trukme
1000	0.006382
10000	0.562682
100000	57.5754
1000000	1000+
10000000	10000+

Testuojant faila su 1 000 000 studentu skirstymo laikas toks ilgas, jog tiesiog neverta laukti pabaigos

1.13 Deque esanciu studentu rikiavimo rezultatai naudojant 3 strategija

Studentu skaicius	Duomenu skirstymo trukme
1000	0.001138
10000	0.010341
100000	0.099007
1000000	1.18494
10000000	30.9919

Vykdymo laikas ženkliai sutrumpėja, lyginant su 2 strategija

1.14 Programos veikimo laikų palyginimas naudojant Class ir Struct

1.14.1 Struct

Studentu skaicius	Duomenu nuskaitymo trukme	Duomenu rikiavimo trukme	Duomenu skirstymo trukme	Viso testo trukme
100000	0.184327	0.067112	0.023777	0.624784
1000000	1.74853	0.918207	0.24031	5.9056

1.14.2 Class

Studentu skaicius	Duomenu nuskaitymo trukme	Duomenu rikiavimo trukme	Duomenu skirstymo trukme	Viso testo trukme
100000	0.201868	0.119731	0.036706	0.601262
1000000	1.84009	1.63147	0.372546	6.44111

1.15 Programos veikimo laikų palyginimas naudojant optimizavimo flag'us

1.15.1 Studentų kiekis: 100000

	Duomenu nuskaitymo trukme	Duomenu rikiavimo trukme	Duomenu skirstymo trukme	Viso testo trukme	.exe failo dydis
Struct -O1	0.140682	0.013065	0.012014	0.393933	3233 kB
Struct -O2	0.138285	0.013137	0.011664	0.403994	3216 kB
Struct -O3	0.137578	0.011896	0.011614	0.400341	3206 kB
Class -O1	0.146403	0.061518	0.024968	0.492828	3225 kB
Class -O2	0.15007	0.050086	0.021801	0.461427	3208 kB
Class -O3	0.152853	0.048282	0.022229	0.462433	3205 kB

1.15.2 Studentų kiekis: 1000000

	Duomenu nuskaitymo trukme	Duomenu rikiavimo trukme	Duomenu skirstymo trukme	Viso testo trukme
Struct -O1	1.28276	0.186754	0.165579	4.07632
Struct -O2	1.22172	0.184022	0.126522	4.02027
Struct -O3	1.26782	0.169268	0.12368	4.10841
Class -O1	1.34906	0.709025	0.235499	4.71324
Class -O2	1.31023	0.698598	0.224778	4.59237
Class -O3	1.35358	0.711891	0.241283	4.60437

1.16 Rule Of Five Pritaikymas

1.16.1 Testavimo funkcijos rezultatai

1.16.2 Pridėtų dalykų aprašas

1. Copy konstruktorius - naujo "studentas" objekto kūrimo metu mes nukopijuojame visus duomenis į naują objektą iš kažkurio seno objekto.
2. Copy Assignment operatorius - naudodami lygybės ženklą mes galime nukopijuoti visus vieno objekto duomenis kitam objektui.
3. Move konstruktorius - naujo "studentas" objekto kūrimo metu mes perkeliame visus duomenis iš senesnio objekto į naujai kuriamą (senasis objektas lieka galioti, bet jo būsena nėra tiksliai žinoma).
4. Move Assignment operatorius - naudodami lygybės ženklą ir "move" raktažodį, mes galime jau sukurtam objektui perkelti visus duomenis iš seno objekto (senasis objektas lieka galioti, bet jo būsena nėra tiksliai žinoma).

1.16.3 Perdengtų metodų aprašas

1.16.3.1 Įvestis

1. Rankinis būdas: programoje parašius, tarkim, cin >> studentas, vartotojas turės galimybę ranka įvesti visus objekto duomenis, jei parinktis (gauta programos pradžioje, bus lygi 1).
2. Automatinis būdas: jei parinktis bus lygi 2 arba 3, tuomet vartotojas galės įvesti tik vardą ir pavardę arba apskritai visi duomenys bus generuojami. (Programoje šios įvesties užrašymas taip pat atrodo cin >> studentas).
3. Nuskaitymas iš failo: liko nepakitęs.

1.16.3.2 Išvestis

1. Į ekraną: panaudojant operatorių $<<$, tarkim cout $<<$ studentas, visi "studentas" klasės duomenys bus išvesti į ekraną.
2. Į failą: panaudojant operatorių $<<$, tarkim vargsiukai $<<$ studentas, visi "studentas" klasės duomenys bus išvesti į failą "vargsiukai".

1.17 Paveldėjimo pritaikymas

1.17.1 Pridėtų dalykų aprašas

1. Nauja klasė "zmogus", iš kurios išvedame mūsų senąją klasę "studentas".
2. Į naująją klasę "zmogus" iš klasės "studentas" mes perkėlėme kintamuosius "vardas" ir "pavarde", taip pat naujoji klasė turi konstruktorių, destruktorių, get'erius bei keturias virtualias funkcijas.

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

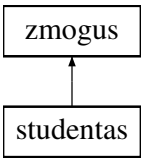
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 studentas Class Reference

Inheritance diagram for studentas:



Public Member Functions

- **studentas** (istream &is)
- **studentas** (const studentas &other)
- **studentas** (studentas &&other) noexcept
- studentas & **operator=** (const studentas &other)
- studentas & **operator=** (studentas &&other) noexcept
- double **galutinis** () const
- int **getEgz** () const
- const vector< int > & **getNd** () const
- void **clearNd** ()
- int **gautiPaskutiniPazymi** ()
- void **generuotiEgzPazymi** ()
- void **generuotiNdPazymi** ()
- void **baloSkaiciavimas** (string)
- void didziosiosVardas ()
- void **didziosiosPavarde** ()
- void generuotiVarda (int i) override
- void generuotiPavarde (int i) override

Public Member Functions inherited from zmogus

- string **vardas** () const
- string **pavarde** () const

Friends

- bool **palygintiMazejant** (const studentas &, const studentas &)
- bool **palygintiDidejant** (const studentas &, const studentas &)
- ostream & **operator**<< (ostream &, const studentas &)
- istream & **operator**>> (istream &, studentas &)

Additional Inherited Members

Protected Member Functions inherited from zmogus

- **zmogus** (string vardas="", string pavarde="")

Protected Attributes inherited from zmogus

- string **vardas_**
- string **pavarde_**

5.1.1 Member Function Documentation

5.1.1.1 didziosiosVardas()

```
void studentas::didziosiosVardas ( )  [inline], [virtual]
```

Implements zmogus.

5.1.1.2 generuotiPavarde()

```
void studentas::generuotiPavarde (
            int i )  [inline], [override], [virtual]
```

Reimplemented from zmogus.

5.1.1.3 generuotiVarda()

```
void studentas::generuotiVarda (
            int i )  [inline], [override], [virtual]
```

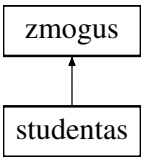
Reimplemented from zmogus.

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

- funkcijos.h
- funkcijos.cpp

5.2 zmogus Class Reference

Inheritance diagram for zmogus:



Public Member Functions

- string **vardas** () const
- string **pavarde** () const
- virtual void **didziosiosVardas** ()=0
- virtual void **generuotiVarda** (int i)
- virtual void **generuotiPavarde** (int i)

Protected Member Functions

- **zmogus** (string vardas="", string pavarde="")

Protected Attributes

- string **vardas_**
- string **pavarde_**

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

- funkcijos.h

Chapter 6

File Documentation

6.1 funkcijos.h

```
00001 #include <iostream>
00002 #include <numeric>
00003 #include <functional>
00004 #include <iomanip>
00005 #include <string>
00006 #include <limits>
00007 #include <vector>
00008 #include <cmath>
00009 #include <random>
00010 #include <ctime>
00011 #include <fstream>
00012 #include <sstream>
00013 #include <algorithm>
00014 #include <chrono>
00015 #include <list>
00016 #include <deque>
00017 #include <utility>
00018
00019 using namespace std;
00020 using namespace chrono;
00021
00022 extern string skaiciavimoBudas;
00023 extern int pazymiuKiekis, parinktis, papildymas, k, i, randomPazymiuKiekis;
00024
00025 // Zmogaus duomenis sauganti klase
00026 class zmogus
00027 {
00028     protected:
00029         string vardas_, pavarde_; // Klases kintamieji: zmogaus vardas ir pavarde
00030     protected:
00031         zmogus(string vardas = "", string pavarde = "") : vardas_(vardas), pavarde_(pavarde) {} //
      Default konstruktorius
00032         ~zmogus() { vardas_.clear(), pavarde_.clear(); } // Destruktorius
00033     public:
00034         inline string vardas() const { return vardas_; }    // get'eriai, inline
00035         inline string pavarde() const { return pavarde_; }  // get'eriai, inline
00036         virtual void didziosiosVardas() = 0; // Visiskai virtuali funkcija
00037         virtual void generuotiVarda(int i) // Funkcija, skirta generuoti zmogaus vardui
00038         {
00039             string vardas;
00040             vardas = "Vardas" + to_string(i + 1);
00041             vardas_ = vardas;
00042         }
00043         virtual void generuotiPavarde(int i) // Funkcija, skirta generuoti zmogaus pavardei
00044         {
00045             string pavarde;
00046             pavarde = "Pavarde" + to_string(i + 1);
00047             pavarde_ = pavarde;
00048         }
00049 };
00050
00051 // Studento duomenis sauganti klase
00052 class studentas : public zmogus
00053 {
00054     private:
00055         vector<int> nd_; // Studento namu darbu pazymiu vektorius
00056         int egz_; // Studento egzamino pazymys
00057         double vidurkis_, mediana_, galutinis_; // Studento pazymiu vidurkis, mediana ir galutinis
      balas
00058     public:
```

```
00059         studentas() : zmogus(), egz_(0) {}  // default konstruktorius
00060         ~studentas() { clearNd(); } // destruktorius
00061         studentas(istream& is); // Konstruktorius su nuoroda i istream objekta, kaip parametru
00062         studentas(const studentas& other) : // Copy konstruktorius
00063             zmogus(other.vardas_, other.pavarde_),
00064             nd_(other.nd_),
00065             egz_(other.egz_),
00066             vidurkis_(other.vidurkis_),
00067             mediana_(other.mediana_),
00068             galutinis_(other.galutinis_) {}
00069         studentas(studentas&& other) noexcept : // Move konstruktorius
00070             zmogus(move(other.vardas_), move(other.pavarde_)),
00071             nd_(move(other.nd_)),
00072             egz_(move(other.egz_)),
00073             vidurkis_(move(other.vidurkis_)),
00074             mediana_(move(other.mediana_)),
00075             galutinis_(move(other.galutinis_)) {}
00076         studentas& operator=(const studentas& other) { return *this = studentas(other); } // Copy
      assignment operatorius
00077         studentas& operator=(studentas&& other) noexcept // Move assignment operatorius
00078         {
00079             swap(vardas_, other.vardas_);
00080             swap(pavarde_, other.pavarde_);
00081             swap(nd_, other.nd_);
00082             swap(egz_, other.egz_);
00083             swap(vidurkis_, other.vidurkis_);
00084             swap(mediana_, other.mediana_);
00085             swap(galutinis_, other.galutinis_);
00086             return *this;
00087         }
00088         double galutinis() const { return galutinis_; } // Galutinio balo get'eris
00089         int getEgz() const { return egz_; } // Egzamino pazymio get'eris
00090         const vector<int>& getNd() const { return nd_; } // Namu darbu pazymiu vektoriaus get'eris
00091         void clearNd() { nd_.clear(); } // Funkcija, isvalanti namu darbu pazymiu vektoriu
00092         int gautiPaskutiniPazymi();
00093         void generuotiEgzPazymi();
00094         void generuotiNdPazymi();
00095         void baloSkaiciavimas(string);
00096         void didziosiosVardas() { for(char &c : vardas_) c = toupper(c); } // Funkcija, skirta visas
      vardo raides paversti i didziasias
00097         void didziosiosPavarde() { for(char &c : pavarde_) c = toupper(c); } // Funkcija, skirta visas
      pavardes raides paversti i didziasias
00098         void generuotiVarda(int i) override { zmogus::generuotiVarda(i); } // Funkcija, skirta
      generuoti varda
00099         void generuotiPavarde(int i) override { zmogus::generuotiPavarde(i); } // Funkcija, skirta
      generuoti pavarde
00100         friend bool palygintiMazejant(const studentas&, const studentas&);
00101         friend bool palygintiDidejant(const studentas&, const studentas&);
00102         friend ostream& operator<<(ostream&, const studentas&);
00103         friend istream& operator>>(istream&, studentas&);
00104 };
00105
00106 int generuotiPazymi();
00107 string didziosios(string&);
00108 bool tikRaides(string);
00109 int tarpuSkaicius(string);
00110 void printHeader(ostream&);
00111 void testas(studentas&);
00112 void generuotiFaila(int, int, string);
00113 template <typename Cont>
00114 void failoSkaitymas(ifstream&, Cont&);
00115 template <typename Cont>
00116 void strategija3(Cont&, Cont&);
00117 template <typename Cont>
00118 void rikiuotiDidejant(Cont&);
00119 template <typename Cont>
00120 void rikiuotiMazejant(Cont&);
```

Index