Objektinis_2

Generated by Doxygen 1.10.0

Chapter 1

Objektinio užduotis 2

Programa skaičiuojanti galutinį studento rezultatą pagal pateiktus namų darbų ir egzamino rezultatus.

1.1 Funkcionalumas

- Meniu kuriame galima pasirinkti, ką atsitiktinai generuoti.
- Duomenų skaitymas iš išankstinio failo.
- Duomenų įvedimas.
- Duomenų failo kūrimas.
- Galutinio balo skaičiavimas pagal vidurkį ir medianą.
- Galima skaityti duomenis iš tam tikru formatu pateiktų teksto failų.
- "Pažengusių" ir "Žlugusių" mokinių išvedimas atskiruose failuose.

1.2 Perdengti operatoriai

Jei naudojama "cin $>>$ klasė" arba "cout $<<$ klasė$>$" atspausdinama arba įrašoma atitinkama klasės informacija visur pasitaikanti šioje užduotyje. Jei rašoma kažkas kitas, o ne klasė, jie veikia kaip įprastai. Pakeisti kur spausdinama galima arba keičiant freopen parametrus arba kode pritaikant ifstream ir ofstream, kas leistų naudoti skirtingus raktažodžius spausdinimui į skirtingas vietas.

1.3 Naudojimosi instrukcijos

- Įjungti programą.
- Sekti terminale matomus žingsnius.
- Jei prašome vesti failo pavadinimą, vesti be ".txt" pabaigoje.
- Gauti rezultatus.

1.4 Sistemos specifikacijos

- **CPU**: AMD Ryzen 5 5600H 3.30 GHz
- **RAM**: DDR4 16GB
- **HDD**: SSD 512GB

1.5 Greičio testai (5 testų vidurkis) ms

1.5.0.1 Vector

Failas	Skaitymo trukmė	Rūšiavimo trukmė	Skirstymo trukmė
stud1000	11	0	1
stud10000	87	16	1
stud100000	907	147	25
stud1000000	8296	1911	292
stud10000000	90697	25683	2911

1.5.0.2 List

Failas	Skaitymo trukmė	Rūšiavimo trukmė	Skirstymo trukmė
stud1000	14	0	1
stud10000	160	6	17
stud100000	1160	78	144
stud1000000	11753	1202	1570
stud10000000	120384	18116	40532

1.5.0.3 Deque

Failas	Skaitymo trukmė	Rūšiavimo trukmė	Skirstymo trukmė
stud1000	12	2	1
stud10000	97	29	6
stud100000	918	394	71
stud1000000	9375	5177	937
stud10000000	96751	69256	61005

1.6 Skirstymas pagal skirtingas strategijas (3 testų vidurkis) ms

1.6.0.1 Vector

Failas	1 strategija	2 strategija (originali)	3 strategija
stud1000	0	1	1
stud10000	3	1	6
stud100000	28	25	31
stud1000000	372	262	335
stud10000000	5580	2911	3398

1.6.0.2 List

Failas	1 strategija	2 strategija (originali)	3 strategija
stud1000	1	1	1
stud10000	20	17	10
stud100000	242	144	149
stud1000000	2628	1570	1806
stud10000000	60549	40532	25499

1.6.0.3 Deque

Failas	1 strategija	2 strategija (originali)	3 strategija
stud1000	1	1	0
stud10000	12	6	6
stud100000	159	71	70
stud1000000	1698	937	884
stud10000000	102817	61005	52611

1.7 Klasių ir struktūrų spartos palyginimas (naudojant vektorių)

1.7.0.1 Struktūra

Failas	Skaitymo trukmė	Rušiavimo trukmė	Skirtsymo trukmė
stud1000000	8996	1911	292
stud10000000	90697	25683	2911

1.7.0.2 Klasė

Failas	Skaitymo trukmė	Rušiavimo trukmė	Skirtsymo trukmė
stud1000000	8152	2805	125
stud10000000	86862	37644	1424

1.8 Optimizavimo "flag'ų" palyginimas (stud1000000)

	Skaitymo, rūšiavimo ir skirtsymo trukmė (ms)	.exe dysis
Struct -O1	10118	466 KB
Struct -O2	10166	466 KB
Struct -O3	10388	466 KB
Class -O1	12071	451 KB
Class -O2	11781	451 KB
Class -O3	11082	451 KB

1.9 Release'ų istorija

- V.pradinė: pirma prelimenati programa, kuri skaičiuoja ranka įvestus mokinio duomeis ir išveda galutinius rezultatus.
- v0.1: nereikia iš anksto nustatyti duomenų kiekio, padarytas atsitiktinės generacijos funkcionalumas. Programa padaryta naudojant vektorius ir, atskirai, naudojant masyvus.
- v0.2: programa gali duomenis priimti iš failo.
- v0.3: programa paskirstyta per kelis failus, pridėtas išimčių valdymas.
- v0.4: programoje galima generuoti naujus failus, duomenys atspausdinami į 2 atskirus failus, atliekama laiko analizė.
- v1.0: atliktas programos testavimas su skirtingais konteineriais, naudotos skirtingos mokinių skirstymo strategijos, padarytos jų efektyvumo strategijos.
- v1.1: programa perdaryta naudojant custom klases o ne struktūras.
- v1.2: pritaikyta rule of five, sukurti move ir copy operatoriai
- v1.5: klasė padalinta į dvi dalis, viena iš kurių abstrakti bazinė.

1.10 Tyrimas

Buvo atliktas mokinių dalijimo į 2 konteinerius skirtingų strategijų testavimas, kuris padėjo paoptimizuoti greitį naudojant list ir deque, bet sparta naudojant vector sumažėjo. Visais atvejais 1 stratejija buvo pati lėčiausia ir taip pat ji prasčiausia programos naudojamos vietos atžvilgiu.

1.11 Kompiuterio paruošimas programai

Čia gidas Windows sistemoms.

- Atsisiųskite c++ kompiliatorių. Gidas čia: https://www.geeksforgeeks.org/installing-mingw-tools-fo
- Atsisiųskite make. Gidas čia: https://linuxhint.com/install-use-make-windows/.

1.12 Programos diegimas ir paleidimas

1. Atsisiųskite programos kodą iš repozitorijos.
2. Terminale pasiekite atsisiuntimo aplanką.
3. Terminale parašykite "make" (pirmą kartą, kai leidžiate programą).
4. Jei norite patikrinti testus, naudokite "make test".
5. Paleiskite programą terminale įvesdami .\prog.exe (Windows) arba .\prog (Linux)

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

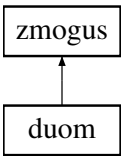
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 duom Class Reference

Inheritance diagram for duom:



Public Member Functions

- **duom** (istream &cin)
- **duom** (const duom &temp)
- **duom** (duom &&temp) noexcept
- duom & **operator=** (const duom &temp)
- duom & **operator=** (duom &&temp) noexcept
- string vard () const override
- string pav () const override
- double **galvid** () const
- double **galmed** () const
- void vardas (const string &va) override
- void pavarde (const string &pa) override
- void **nd** (int nd)
- void **egz** (int egz)
- void **calc** ()
- void **vpskait** ()
- void **skaitduom** ()
- void **spausdinti** ()
- void **vardoGen** ()
- void **ndGen** ()
- void **egzGen** ()

Public Member Functions inherited from zmogus

- **zmogus** (const zmogus &temp)
- **zmogus** (zmogus &&temp) noexcept
- zmogus & **operator=** (const zmogus &temp)
- zmogus & **operator=** (zmogus &&temp) noexcept

Friends

- istream & **operator>>** (istream &cin, duom &s)
- ostream & **operator<<** (ostream &cout, const duom &s)

Additional Inherited Members

Protected Attributes inherited from zmogus

- string **vard_**
- string **pav_**

5.1.1 Member Function Documentation

5.1.1.1 pav()

```
string duom::pav ( ) const  [inline], [override], [virtual]
```

Implements zmogus.

5.1.1.2 pavarde()

```
void duom::pavarde (
            const string & pa )  [inline], [override], [virtual]
```

Implements zmogus.

5.1.1.3 vard()

```
string duom::vard ( ) const  [inline], [override], [virtual]
```

Implements zmogus.

5.1.1.4 vardas()

```
void duom::vardas (
            const string & va )  [inline], [override], [virtual]
```

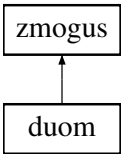
Implements zmogus.

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

- virtual void **vardas** (const string &va)=0
- virtual void **pavarde** (const string &pa)=0
- virtual string **vard** () const =0
- virtual string **pav** () const =0
- **zmogus** (const zmogus &temp)
- **zmogus** (zmogus &&temp) noexcept
- zmogus & **operator=** (const zmogus &temp)
- zmogus & **operator=** (zmogus &&temp) noexcept

Protected Attributes

- string **vard_**
- string **pav_**

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

- funkcijos.h

Chapter 6

File Documentation

6.1 funkcijos.h

```
00001 #ifndef FUNKCIJOS_H
00002 #define FUNKCIJOS_H
00003
00004 #include <bits/stdc++.h>
00005
00006 using namespace std;
00007 using namespace std::chrono;
00008 using std::setw;
00009 using std::left;
00010
00011 class zmogus{
00012     protected:
00013         string vard_;
00014         string pav_;
00015     public:
00016         virtual void vardas(const string &va) = 0;
00017         virtual void pavarde(const string &pa) = 0;
00018
00019         virtual string vard() const = 0;
00020         virtual string pav() const = 0;
00021
00022         zmogus() {}
00023         ~zmogus() {}
00024
00025         // copy c
00026         zmogus(const zmogus &temp)
00027             : vard_(temp.vard_), pav_(temp.pav_) {}
00028
00029         // move c
00030         zmogus(zmogus &&temp) noexcept
00031             : vard_(move(temp.vard_)), pav_(move(temp.pav_)) {}
00032
00033         // copy a
00034         zmogus& operator=(const zmogus &temp) {
00035             if(this!=&temp){
00036                 vard_=temp.vard_;
00037                 pav_=temp.pav_;
00038             }
00039             return *this;
00040         }
00041
00042         // move a
00043         zmogus& operator=(zmogus &&temp) noexcept {
00044             if(this!=&temp){
00045                 vard_=move(temp.vard_);
00046                 pav_=move(temp.pav_);
00047             }
00048             return *this;
00049         }
00050 };
00051
00052 class duom : public zmogus{
00053     private:
00054         vector<int> ndrez_;
00055         int egzrez_;
00056         double galvid_, galmed_;
00057     public:
00058         duom() : egzrez_(0), galvid_(0), galmed_(0) {}
```

```
00059         ~duom() {}
00060         duom(istream &cin);
00061
00062         // copy c
00063         duom(const duom &temp)
00064             : zmogus(temp), ndrez_(temp.ndrez_), egzrez_(temp.egzrez_), galvid_(temp.galvid_),
      galmed_(temp.galmed_) {}
00065
00066         // move c
00067         duom(duom &&temp) noexcept
00068             : zmogus(move(temp)), ndrez_(move(temp.ndrez_)), egzrez_(temp.egzrez_),
      galvid_(temp.galvid_), galmed_(temp.galmed_) {
00069                 temp.egzrez_={};
00070                 temp.galvid_={};
00071                 temp.galmed_={};
00072                 temp.ndrez_.clear();
00073             }
00074
00075         // copy a
00076         duom& operator=(const duom &temp) {
00077             if(this!=&temp){
00078                 zmogus::operator=(temp);
00079                 ndrez_=temp.ndrez_;
00080                 egzrez_=temp.egzrez_;
00081                 galvid_=temp.galvid_;
00082                 galmed_=temp.galmed_;
00083             }
00084             return *this;
00085         }
00086
00087         // move a
00088         duom& operator=(duom &&temp) noexcept {
00089             if(this!=&temp){
00090                 zmogus::operator=(move(temp));
00091                 ndrez_=move(temp.ndrez_);
00092                 egzrez_=move(temp.egzrez_);
00093                 temp.egzrez_={};
00094                 galvid_=move(temp.galvid_);
00095                 temp.galvid_={};
00096                 galmed_=move(temp.galmed_);
00097                 temp.galmed_={};
00098                 temp.ndrez_.clear();
00099             }
00100             return *this;
00101         }
00102
00103         inline string vard() const override { return this->vard_; }
00104         inline string pav() const override { return this->pav_; }
00105         inline double galvid() const { return galvid_; }
00106         inline double galmed() const { return galmed_; }
00107
00108         void vardas(const string &va) override { this->vard_=va; }
00109         void pavarde(const string &pa) override { this->pav_=pa; }
00110         void nd(int nd) { ndrez_.push_back(nd); }
00111         void egz(int egz) { egzrez_=egz; }
00112         void calc();
00113
00114         void vpskait();
00115         void skaitduom();
00116         void spausdinti();
00117         void vardoGen();
00118         void ndGen();
00119         void egzGen();
00120
00121         friend istream& operator>>(istream &cin, duom &s);
00122         friend ostream& operator<<(ostream &cout, const duom &s);
00123
00124 };
00125
00126 bool sort1(const duom &, const duom &);
00127 bool sort2(const duom &, const duom &);
00128 bool sort3(const duom &, const duom &);
00129 bool sort4(const duom &, const duom &);
00130 bool sort1u(const duom &, const duom &);
00131 bool sort2u(const duom &, const duom &);
00132 bool sort3u(const duom &, const duom &);
00133 bool sort4u(const duom &, const duom &);
00134 bool pagalVid(const duom &x, const double d);
00135 bool pagalMed(const duom &x, const double d);
00136
00137 template <typename sk=int, typename talpa>
00138 void rusiuoti(sk &, sk &, talpa &);
00139
00140 template <typename talpa, typename sk>
00141 void strategija3(talpa &, talpa &, sk);
00142
00143 template <typename talpa, typename sk>
```

```
00144 void strategija2(talpa &, talpa &, sk);
00145
00146 template <typename talpa, typename sk>
00147 void strategija1(talpa &, talpa &, sk, talpa &);
00148
00149 template <typename sk, typename talpa>
00150 void skaitymas(sk &, talpa &);
00151
00152 template <typename talpa, typename sk=int>
00153 void isfailo(talpa &, sk &);
00154
00155 void kurtifaila();
00156
00157 template <typename sk, typename talpa>
00158 double rankinis(sk &, talpa &, sk &);
00159
00160 void input();
00161
00162 void testas();
00163
00164 #endif
```

Index