Objektinis-programavimas

Generated by Doxygen 1.12.0

Chapter 1

Objektinis-programavimas

1.1 Programos aprašymas

Ši programa leidžia vartotojams įvesti arba importuoti studentų duomenis, įskaitant vardą, pavardę, namų darbų rezultatus ir egzamino įvertinimus. Galutiniai rezultatai apskaičiuojami pagal formulę:

Galutinis balas = 0,4 × namų darbų vidurkis + 0,6 × egzamino įvertinimas

Vartotojai gali pasirinkti, ar namų darbų rezultatus skaičiuoti naudojant medianą, ar vidurkį. Rezultatai su studento informacija ir galutiniu balu gali būti išvedami į naują failą arba rodomi terminale.

Išsimčių valdymas:

- Klausimuose, kuriuose reikalaujama vartotojo pasirinkti vieną iš dviejų variantų yra naudojama try and catch norint apsaugoti nuo neteisingų įvedimų ir taip pat sistemos lūžimo, jeigu būtų įvesta raidė
- Norint įvesti namų darbų ar egzamino pažymį vartotojas yra apribojamas sveikaisiais skaičiais nuo 1 iki 10
- Nuskaitant failą yra tikrinama ar failas atsidaro, ar nuskaitytoje eilutėje yra namų darbų įrašai ir ar įrašyti namų darbų duomenys atitinka nustatytus standartus(sveikieji skaičiai nuo 1 iki 10 ir ar nėra įrašyta raidė). Pasitaikius išimčiai išmetama atitinkama klaida.

1.2 Failų aprašymas

- main.cpp – pagrindinis programos failas, kuris inicijuoja ir vykdo pagrindinę programos logiką.
- Stud.cpp – faile įgyvendinti visi metodai, naudojami programoje, susiję su studentų duomenų apdorojimu.
- Files.cpp – faile aprašyti metodai, skirti failų generavimui ir nuskaitymui.
- Grupavimas.cpp – faile pateiktos trys strategijos, skirtos studentų grupavimui.
- MyLib.h – faile deklaruotos visos programoje naudojamos bibliotekos.
- Stud.h – faile aprašyta Stud struktūra, apibrėžianti studentą, ir deklaruotos su ja susijusios funkcijos.
- Timer.h – faile aprašyta klasė, skirta laiko matavimui programos vykdymo metu.
- Test.pp - faile aprašyti konstruktoriaus, getteriu ir seteriu, kopijavimo konstruktoriaus testai.

Metodai:

- Vartotojas gali pasirinkti, ar studentų informaciją nuskaityti iš tekstinio failo, ar įvesti rankiniu būdu.
- Jei vartotojas pasirenka nuskaityti informaciją iš tekstinio failo, suteikiama galimybė sugeneruoti naujus tekstinius failus.
- Įvedant informaciją rankiniu būdu, vartotojas gali sugeneruoti atsitiktinius namų darbų ir egzamino rezultatus.
- Vartotojas gali pasirinkti, kaip skaičiuoti galutinį rezultatą – naudojant namų darbų medianą arba vidurkį.

Metodai su failų generavimu:

- Sugeneruojami 5 nauji tekstiniai failai.
- Nauji failai nuskaitomi ir studentai grupuojami pagal galutinį pažymį: jei galutinis pažymys yra nemažesnis už 5, studentas priskiriamas „smart“ grupei, kitaip – „dumb“ grupei.
- Naudotojui leidžiama pasirinkti, pagal kokius kriterijus išrūšiuoti studentus, įrašant juos į rezultatų failus, taip pat pasirinkti, pagal kurią strategiją juos grupuoti.

1.3 Programos įdiegimo instrukcija

Nukopijuokite projekto direktoriją, sukuriame buid direktoriją ir į ją persikeliame:
```
cd projekto_direktorija
mkdir build
cd build
```

Paleidžiame CMake, kad sugeneruotų reikalingus failus:
```
cmake ..
```

Sukompiliuojame kodą, sukuriame .exe failą:
```
cmake --build . --config Release
```

Lengvesniam paleidimui unzipinus projektą paleiskite **run.bat** failą.

1.4 Testavimas

V0.2, v0.3 ir v1.0 dalyse buvo atliktas testavimas, naudojant sugeneruotus 5 failus. Testavimas buvo atliktas **6** kartus, o gauti programos vykdymo laikai buvo panaudoti vidurkiui apskaičiuoti. Buvo naudojami dviejų tipų konteineriai: `std::vector` ir `std::list`. Žemiau pateikiamos v0.2, v0.3 ir v1.0 dalių testavimo rezultatų suvestinės.
Kompiuterio su kuriuo buvo atliekami testai specifikacijos:

- CPU: Intel(R) Core(TM) Ultra 7 155H **3.80 GHz**
- RAM: 16 GB
- SDD: 1 TB

1.5 v2.0

v2.0 atnaujinimai:

- Sukurta programos dokumentacija naudojant Doxygen
- Realizuoti trys unit testai su klase: konstruktoriui, geteriam ir seteriam, kopijavimo konstruktoriui. Naudojant google test framework.

1.5.0.1 Unit tests

```
//Test of constructor using google tests
TEST(Student, Constructor) {
   <...>
}
//Test of getters and setters for name and last name, using google tests
TEST(Student, GettersAndSetters) {

    <...>
}

//Test of the copy constructor, using google tests
TEST(Student, CopyConstructor) {
   <...>
}
```

1.5.0.2 Programos paleidimas

Paleidus run.bat filą paleidžiama List.exe failas(pagrindinis programos failas), ProjectTest.exe(Unit testų paleidimas).

1.6 v1.5

v1.5 atnaujinimai:

- Sukurta bazinė klasė Human
- Padaryta, kad Žmogui skirta bazinė klasė Human būtų abstrakčioji
- Stud klasė nuo šiol yra Human išvestinė klasė

1.6.1 ### Klasių diagramos

1.7 v1.2

v1.2 atnaujinimai:

- Realizuota rule of three (destructor, copy constructor and copy assigment) turimai Stud klasei.
- Sukurti įvesties ir išvestie operatoriai
- Perdengti įvesties ir išvesties metodai darbui su Studentų klasę.

1.7.1 Plačiau apie Įvesties ir išvesties metodų perdengimą

Sukurti keli metodus su tuo pačiu pavadinimu. Kitaip tariant duomenų iš failo nuskaitymo, failu generavimo ir duomenų nuskaitymo rankiniu būdu metodai dabar turi vienodus pavadinimus, bet atlieka skirtingas funkcijas.
Įvesties metodų perdengimas:

- `void input(Stud& Lok);` - metodas naudojamas studentų nuskaitymui, kai informacija įvedama rankiniu būdu
- `void input(const string& fileName, const int& number);` - metodas naudojamas studentų informacijos generavimui naujuose failuose
- `void input(const string& fileName, list <Stud>& stud);` - metodas naudojamas studentų failų nuskaitymui

Išvesties metodų perdengimas:

- `void output(list<Stud>& stud, int choice);` - metodas naudojamas studentų duomenų išvedimui į terminalą
- `void output(const string& fileName, list<Stud>& stud);` - metodas naudojamas studentų duomenų išvedimui į failus

1.8 V1.1

v1.1 atnaujinimai: Programam nuo šiol naudoja class tipo realizacija, vietoje struct.
Testavimui naudojamos didesnės apimties failai, norint išsiaiškinti kaip pokytis iš struct į class tipą paveikė didesnės apimties failų veikimo spartą. Naudojama 3 grupavimo strategija ir lists.

1.8.0.1 Failas: Studentai 1000000

VEIKSMAS	VYKDYMO VID.(struct - lst)	VYKDYMO VID.(class - lst)
Failo nuskaitymas	6.109984 s	7.88116 s
Failo rušiavimas	0.2250646 s	0.32491 s
Failo grupavimas	0.08983824 s	0.10684 s
Studentų išvedimas į smart failą	0.6137734 s	0.76192 s
Studentų išvedimas į dumb failą	0.422014 s	0.55333 s

1.8.0.2 Failas: Studentai 10000000

VEIKSMAS	VYKDYMO VID.(struct - lst)	VYKDYMO VID.(class - lst)
Failo nuskaitymas	57.26326 s	77.97883 s
Failo rušiavimas	2.928294 s	4.32859 s
Failo grupavimas	0.8334702 s	1.20533 s
Studentų išvedimas į smart failą	5.74896 s	7.71917 s
Studentų išvedimas į dumb failą	3.777118 s	5.48454 s

∗∗Testavimo išvados: ∗∗

- Naudojant class vietoj struct programos veikimo efektyvumas sumažėja, tačiau tai gali būti dėl to, nes class paprastai turi daugiau funkcionalumo, pavyzdžiui, įtrauktus metodus ir papildomas saugumo savybes, kurios gali padidinti atminties ir procesoriaus naudojimą

1.8.1 Struct ir Klasės versija ištirta su optimizavimo "flagais" (yra lentelė: greitis, exe failo dydis)

1.8.1.1 Class - list

FLAG	LAIKAS	.exe FAILO DYDIS
-O1	181.3833 s	163.8496 KB
-O2	168.82811 s	163.7412 KB
-O3	170.6521 s	168.7304 KB

1.8.1.2 Class - vector

FLAG	LAIKAS	.exe FAILO DYDIS
-O1	140.3943 s	185.3867 KB
-O2	139.7370 s	180.3212 KB
-O3	138.5772 s	218.7763 KB

1.8.1.3 Struct - list

FLAG	LAIKAS	.exe FAILO DYDIS
-O1	169.1964 s	162.8779 KB
-O2	162.7343 s	161.9873 KB
-O3	163.4852 s	165.8007 KB

1.8.1.4 Struct - vector

FLAG	LAIKAS	.exe FAILO DYDIS
-O1	137.7089 s	191.4091 KB
-O2	125.9297 s	185.9580 KB
-O3	125.8531 s	203.1005 KB

Eksperimento tyrimo išvados:

- Naudojant vektorių, vykdomojo failo dydis buvo žymiai didesnis (iki 203 KB), o mažiausias dydis užfiksuotas struktūros su list atveju ($\sim$161 KB).
- -O2 optimizacija pasiūlė geriausią balansą tarp vykdymo laiko ir failo dydžio; -O3 dažnai pagerindavo tik laiką, bet padidindavo failo dydį.
- Struktūra buvo efektyvesnė pagal laiką visais atvejais, tačiau klasės su vektoriumi našumas buvo panašus į struktūrą su vektoriumi.

Rekomendacijos: Struktūros su vektoriumi kombinacija ir -O2 optimizacija yra tinkamiausia, jei svarbus greitis, o struktūra su sąrašu – jei prioritetas mažas failo dydis.

1.9 V1.0

v1.0 atnaujinimas: Vartotojui suteikta galimybė pasirinkti, kokią strategiją jis/ji nori naudoti atliekant studentų grupavimą.

- 1 strategija - skaidymas į du naujus to paties tipo konteinerius: "dumb" ir "smart".
- 2 strategija - skaidymas (rūšiavimas) panaudojant tik vieną naują konteinerį: "smart".
- 3 startefija - 2 strategijos patobulinimas naudojant funkcija partition.

1.9.0.1 Failas: Studentai 1000

VEIKSMAS	VYKDYMO VID.(vector-2str)	VYKDYMO VID.(list-2str)	VYKDYMO VID.(vector-3str)	VYKDYMO VID.(list-3str)
Failo nuskaitymas	0.01044758 s	0.019020 s	0.0102307 s	0.020861 s
Failo rušiavimas	0.0003314 s	0.000228 s	0.0002818 s	0.0000924 s
Failo grupavimas	0.00020226 s	0.000085 s	0.0000639 s	0.000096 s
Studentų išvedimas į smart failą	0.00250062 s	0.003435 s	0.0025763 s	0.002818 s
Studentų išvedimas į dumb failą	0.00140596 s	0.016829 s	0.0014608 s	0.0013931 s

1.9.0.2 Failas: Studentai 10000

VEIKSMAS	VYKDYMO VID.(vector-2str)	VYKDYMO VID.(list-2str)	VYKDYMO VID.(vector-3str)	VYKDYMO VID.(list-3str)
Failo nuskaitymas	0.0816417 s	0.085794 s	0.0905467 s	0.092395 s
Failo rušiavimas	0.00451634 s	0.002108 s	0.0042513 s	0.0012073 s
Failo grupavimas	0.00206306 s	0.001052 s	0.0006844 s	0.0007259 s
Studentų išvedimas į smart failą	0.0117262 s	0.011443 s	0.0109514 s	0.0107712 s
Studentų išvedimas į dumb failą	0.00751514 s	0.009215 s	0.0070881 s	0.0078009 s

1.9.0.3 Failas: Studentai 100000

VEIKSMAS	VYKDYMO VID.(vector-2str)	VYKDYMO VID.(list-2str)	VYKDYMO VID.(vector-3str)	VYKDYMO VID.(list-3str)
Failo nuskaitymas	0.6415938 s	0.750852 s	0.7453039 s	0.6484366 s
Failo rušiavimas	0.04712624 s	0.015056 s	0.0495826 s	0.0182077 s
Failo grupavimas	0.01539674 s	0.016468 s	0.0072107 s	0.00993212 s
Studentų išvedimas į smart failą	0.07577556 s	0.083223 s	0.0806573 s	0.08293438 s
Studentų išvedimas į dumb failą	0.05242182 s	0.058222 s	0.0531508 s	0.05512132 s

1.9.0.4 Failas: Studentai 1000000

VEIKSMAS	VYKDYMO VID.(vector-2str)	VYKDYMO VID.(list-2str)	VYKDYMO VID.(vector-3str)	VYKDYMO VID.(list-3str)
Failo nuskaitymas	6.098156 s	6.669537 s	7.2731120 s	6.109984 s
Failo rušiavimas	0.4605068 s	0.121612 s	0.5361923 s	0.2250646 s
Failo grupavimas	0.09134838 s	0.229607 s	0.0622364 s	0.08983824 s
Studentų išvedimas į smart failą	0.5152744 s	0.616712 s	0.6443484 s	0.6137734 s
Studentų išvedimas į dumb failą	0.3749948 s	0.458501 s	0.4263662 s	0.422014 s

1.9.0.5 Failas: Studentai 10000000

VEIKSMAS	VYKDYMO VID.(vector-2str)	VYKDYMO VID.(list-2str)	VYKDYMO VID.(vector-3str)	VYKDYMO VID.(list-3str)
Failo nuskaitymas	62.77496 s	64.72142 s	72.2938054 s	57.26326 s
Failo rušiavimas	5.333358 s	3.13935 s	6.2909252 s	2.928294 s
Failo grupavimas	0.9854816 s	1.25798 s	0.5787044 s	0.8334702 s
Studentų išvedimas į smart failą	5.417268 s	6.67649 s	6.5107646 s	5.74896 s
Studentų išvedimas į dumb failą	3.800548 s	4.57997 s	4.6918214 s	3.777118 s

1.9.1 Testavimo išvados:

Buvo testuojama programa naudojant dvi skirtingas struktūras: `std::vector` ir `std::list`. Programos veikimo laikai buvo lyginami. **Galima padaryti išvadas:**

- 3 srategija(tiek vector, tiek list) dažniausiai yra spartesnė nei 2 strategija, ypač rūšiavimo ir grupavimo operacijose. Tai rodo, kad 3 strategija efektyviau tvarko duomenis, greičiau atlieka šias operacijas.
- Didėjant duomenų kiekiui, tiek vector, tiek list struktūrose išlieka panaši vykdymo laiko tendencija – 3 strategija yra greitesnė nei 2 strategija. Tai rodo, kad pasirinkus tinkamą formatą, galima sumažinti vykdymo laiką nepriklausomai nuo struktūros tipo.
- List struktūros šiek tiek praranda našumą esant labai dideliems duomenų rinkiniams. Tai leidžia manyti, kad vector struktūros yra labiau skalabilios su didėjančiu duomenų kiekiu.

1.10 v0.3

V0.3 atnaujinimas:

- Norint atlikti testavimus ir palyginti dviejų skirtingų rūšių konteinerius, programa buvo pritaikyta ne tik vector tipo konteineriams, bet ir list tipo konteineriams.
- Surenkant studentu duomenis ranka, programa apskaičiuoja ne tik galutinį rezultatą, tačiau ir struktūros adresą kompiuterio atmintyje.

1.10.0.1 Failas: Studentai 1000

Pirma strategija

VEIKSMAS	VYKDYMO VIDURKIS(vector)	VYKDYMO VIDURKIS(list)
Failo nuskaitymas	0.0159676167 s	0.0123800667 s
Failo grupavimas	0.0019471667 s	0.0013408 s
Failo rušiavimas	0.0002987667 s	0.0004841333 s
Studentų išvedimas į smart failą	0.0030953833 s	0.0031049833 s
Studentų išvedimas į dumb failą	0.00191965 s	0.0015635833 s

1.10.0.2 Failas: Studentai 10000

Pirma strategija

VEIKSMAS	VYKDYMO VIDURKIS(vector)	VYKDYMO VIDURKIS(list)
Failo nuskaitymas	0.0985081333 s	0.0950360667 s
Failo grupavimas	0.0076673167 s	0.0069883833 s
Failo rušiavimas	0.0052359 s	0.0018065333 s
Studentų išvedimas į smart failą	0.0130139 s	0.0126294 s
Studentų išvedimas į dumb failą	0.0099617833 s	0.0095311167 s

1.10.0.3 Failas: Studentai 100000

Pirma strategija

VEIKSMAS	VYKDYMO VIDURKIS(vector)	VYKDYMO VIDURKIS(list)
Failo nuskaitymas	0.751953875 s	0.7402695 s
Failo grupavimas	0.0603202 s	0.0593329667 s
Failo rušiavimas	0.0599166875 s	0.0301281833 s
Studentų išvedimas į smart failą	0.1135195 s	0.0991398333 s
Studentų išvedimas į dumb failą	0.0726314667 s	0.0720664 s

1.10.0.4 Failas: Studentai 1000000

Pirma strategija

VEIKSMAS	VYKDYMO VIDURKIS(vector)	VYKDYMO VIDURKIS(list)
Failo nuskaitymas	8.2884075 s	8.067803333 s
Failo grupavimas	0.851942625 s	0.6628698333 s
Failo rušiavimas	0.71661653 s	0.3381395 s
Studentų išvedimas į smart failą	1.0000588167 s	0.822413 s
Studentų išvedimas į dumb failą	0.6644465 s	0.5990218333 s

1.10.0.5 Failas: Studentai 10000000

Pirma strategija

VEIKSMAS	VYKDYMO VIDURKIS(vector)	VYKDYMO VIDURKIS(list)
Failo nuskaitymas	85.1526125 s	82.4037 s
Failo grupavimas	5.68810375 s	5.05564 s
Failo rušiavimas	6.61167625 s	6.0088216667 s
Studentų išvedimas į smart failą	10.1445716667 s	9.6545516667 s
Studentų išvedimas į dumb failą	7.6079633333 s	7.146921 s

1.10.1

1.10.2 Testavimo išvados

Buvo testuojama programa naudojant dvi skirtingas struktūras: `std::vector` ir `std::list`. Programos veikimo laikai buvo lyginami. **Galima padaryti išvadas:**

- Lyginant vector ir list, programa sparčiau veikia, tačiau pokyčiai yra neženklūs. Atsižvelgiant į skaičius, didesnius pokyčius galime pastebėti tik didesniuose duomenų failuose.
- List yra labiau tinkamas naudoti su dideliais duomenų kiekiais, priešingai negu vector, kuris geriau veikia mažesnio duomenų kiekio failuose.

1.11 v0.2

V0.2 atnaujinimas:

- Vartotojas nuo šiol gali pasirinkti failus ne tik nuskaityti, bet ir sugeneruoti. Naudotojui pasirinkus generavimo metodą, bus sugeneruojami 5 skirtingi failai: studentai 1000, studentai 10000, studentai 100000, studentai 1000000 studentai 10000000
- Studentai pagal galutinį vidurkį buvo sugrupuoti į protingus (galutinis įvertinimas $>$ 5) ir ne tiek protingus studentus (galutinis įvertinimas $<$ 5). Dvi skirtingos grupės yra išvedamos į skirtinguus failus
- Vartotojui suteikiama galimybė pasirinkti kaip rušiuoti studentų failus
- Buvo atliekami testavimai su vektoriu sturktūromis.

1.11.0.1 Failas: Studentai 1000

VEIKSMAS	VYKDYMO VIDURKIS(vector)
Failo nuskaitymas	0.0159676167 s
Failo grupavimas	0.0019471667 s
Failo rušiavimas	0.0002987667 s
Studentų išvedimas į smart failą	0.0030953833 s
Studentų išvedimas į dumb failą	0.00191965 s

1.11.0.2 Failas: Studentai 10000

VEIKSMAS	VYKDYMO VIDURKIS(vector)
Failo nuskaitymas	0.0985081333 s
Failo grupavimas	0.0076673167 s
Failo rušiavimas	0.0052359 s
Studentų išvedimas į smart failą	0.0130139 s
Studentų išvedimas į dumb failą	0.0099617833 s

1.11.0.3 Failas: Studentai 100000

VEIKSMAS	VYKDYMO VIDURKIS(vector)
Failo nuskaitymas	0.751953875 s
Failo grupavimas	0.0603202 s
Failo rušiavimas	0.0599166875 s
Studentų išvedimas į smart failą	0.1135195 s
Studentų išvedimas į dumb failą	0.0726314667 s

1.11.0.4 Failas: Studentai 1000000

VEIKSMAS	VYKDYMO VIDURKIS(vector)
Failo nuskaitymas	8.2884075 s
Failo grupavimas	0.851942625 s
Failo rušiavimas	0.71661653 s
Studentų išvedimas į smart failą	1.0000588167 s
Studentų išvedimas į dumb failą	0.6644465 s

1.11.0.5 Failas: Studentai 10000000

VEIKSMAS	VYKDYMO VIDURKIS(vector)
Failo nuskaitymas	85.1526125 s
Failo grupavimas	5.68810375 s
Failo rušiavimas	6.61167625 s
Studentų išvedimas į smart failą	10.1445716667 s
Studentų išvedimas į dumb failą	7.6079633333 s

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

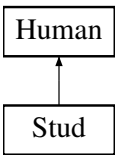
Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 Human Class Reference

`#include <Human.h>`
Inheritance diagram for Human:



Public Member Functions

- Human ()
- Human (string vardas, string pavarde)
- void setName (string tempVardas)
- string getName () const
- void setLastName (string tempPavarde)
- string getLastName () const
- Human (const Human &copy)
- Human & operator= (const Human &other)
- ~Human ()
- virtual void whoAmI () const =0

Protected Attributes

- string vardas
- string pavarde

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Human() [1/3]

```
Human::Human ()  [inline]
```

5.1.1.2 Human() [2/3]

```
Human::Human (
            string vardas,
            string pavarde)  [inline]
```

5.1.1.3 Human() [3/3]

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

5.1.1.4 ~Human()

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

5.1.2 Member Function Documentation

5.1.2.1 getLastName()

```
string Human::getLastName () const  [inline]
```

5.1.2.2 getName()

```
string Human::getName () const  [inline]
```

5.1.2.3 operator=()

```
Human & Human::operator= (
            const Human & other)  [inline]
```

5.1.2.4 setLastName()

```
void Human::setLastName (
            string tempPavarde)  [inline]
```

5.1.2.5 setName()

```
void Human::setName (
            string tempVardas)  [inline]
```

5.1.2.6 whoAmI()

```
virtual void Human::whoAmI () const  [pure virtual]
```

Implemented in Stud.

5.1.3 Member Data Documentation

5.1.3.1 pavarde

```
string Human::pavarde  [protected]
```

5.1.3.2 vardas

```
string Human::vardas  [protected]
```

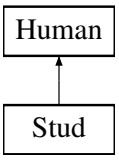
The documentation for this class was generated from the following file:

- List/include/Human.h

5.2 Stud Class Reference

```
#include <Stud.h>
```

Inheritance diagram for Stud:



Public Member Functions

- Stud ()
- Stud (string vardas, string pavarde, vector< int > nd, int egz)
- Stud (std::istream &is)
- void input (Stud &Lok)
- void input (const string &fileName, const int &number)
- void input (const string &fileName, list< Stud > &stud)
- void output (list< Stud > &stud, int choice)
- void output (const string &fileName, list< Stud > &stud)
- ~Stud ()
- Stud (const Stud &copy)
- Stud & operator= (const Stud &other)

  III. copy assigment.
- void whoAmI () const

Public Member Functions inherited from Human

- Human ()
- Human (string vardas, string pavarde)
- void setName (string tempVardas)
- string getName () const
- void setLastName (string tempPavarde)
- string getLastName () const
- Human (const Human &copy)
- Human & operator= (const Human &other)
- ~Human ()

Public Attributes

- vector< int > ND
- double vid
- double med
- double egz

Friends

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

Additional Inherited Members

Protected Attributes inherited from Human

- string vardas
- string pavarde

5.2.1 Constructor & Destructor Documentation

5.2.1.1 Stud() [1/4]

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

5.2.1.2 Stud() [2/4]

```
Stud::Stud (
            string vardas,
            string pavarde,
            vector< int > nd,
            int egz)  [inline]
```

5.2.1.3 Stud() [3/4]

```
Stud::Stud (
            std::istream & is)  [inline]
```

5.2.1.4 ~Stud()

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

5.2.1.5 Stud() [4/4]

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

5.2.2 Member Function Documentation

5.2.2.1 input() [1/3]

```
void Stud::input (
            const string & fileName,
            const int & number)
```

5.2.2.2 input() [2/3]

```
void Stud::input (
            const string & fileName,
            list< Stud > & stud)
```

5.2.2.3 input() [3/3]

```
void Stud::input (
            Stud & Lok)
```

5.2.2.4 operator=()

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

III. copy assigment.

5.2.2.5 output() [1/2]

```
void Stud::output (
            const string & fileName,
            list< Stud > & stud)
```

5.2.2.6 output() [2/2]

```
void Stud::output (
            list< Stud > & stud,
            int choice)
```

5.2.2.7 whoAmI()

```
void Stud::whoAmI () const  [inline], [virtual]
```
Implements Human.

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 & in,
            Stud & student)  [friend]
```

5.2.4 Member Data Documentation

5.2.4.1 egz

```
double Stud::egz
```

5.2.4.2 med

```
double Stud::med
```

5.2.4.3 ND

```
vector<int> Stud::ND
```

5.2.4.4 vid

```
double Stud::vid
```
The documentation for this class was generated from the following files:

- List/include/Stud.h
- List/src/Files.cpp
- List/src/Stud.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:

- List/include/Timer.h

Chapter 6

File Documentation

6.1 List/include/Human.h File Reference

```
#include "Mylib.h"
```

Classes

- class Human

6.2 Human.h

Go to the documentation of this file.

```
00001 #include "Mylib.h"
00002
00003 class Human {
00004 protected:
00005     string vardas, pavarde;
00006 public:
00007     Human() : vardas(""), pavarde("") {};
00008     Human(string vardas, string pavarde) :
00009         vardas(vardas), pavarde(pavarde) {}
00010
00011     //setters and getters
00012     void setName(string tempVardas) {
00013         vardas = tempVardas;
00014     }
00015     inline string getName() const { return vardas; }
00016
00017     void setLastName(string tempPavarde) {
00018         pavarde = tempPavarde;
00019     }
00020     inline string getLastName() const { return pavarde; }
00021
00022     //II. copy constructor
00023     Human(const Human& copy) {
00024         vardas = copy.vardas;
00025         pavarde = copy.pavarde;
00026     }
00027     //III.copy assigment
00028     Human& operator=(const Human& other) {
00029         if (this != &other) {
00030             vardas = other.vardas;
00031             pavarde = other.pavarde;
00032         }
00033         return *this;
00034     }
00035
00036     ~Human() { vardas.clear(); pavarde.clear(); }
00037
00038     virtual void whoAmI() const = 0;
00039 };
00040
```

6.3 List/include/Mylib.h File Reference

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

6.4 Mylib.h

Go to the documentation of this file.

```
00001 #include<iostream>
00002 #include<fstream>
00003 #include<sstream>
00004 #include<iomanip>
00005 #include<string>
00006 #include<vector>
00007 #include<list>
00008 #include<algorithm>
00009 #include<limits>
00010 #include<ios>
00011 #include<cstdlib>
00012 #include<stdexcept>
00013 #include<exception>
00014
00015 using std::list;
00016 using std::endl;
00017 using std::cout;
00018 using std::cin;
00019 using std::left;
00020 using std::right;
00021 using std::setw;
00022 using std::setprecision;
00023 using std::fixed;
00024 using std::sort;
00025 using std::getline;
00026 using std::streamsize;
00027 using std::ifstream;
00028 using std::ofstream;
00029 using std::ostream;
00030 using std::istream;
00031 using std::numeric_limits;
00032 using std::string;
00033 using std::vector;
00034 using std::runtime_error;
00035 using std::istringstream;
00036 using std::cerr;
00037 using std::exception;
00038 using std::invalid_argument;
00039 using std::out_of_range;
00040 using std::noskipws;
00041 using std::to_string;
00042 using std::flush;
00043 using std::stringstream;
00044 using std::sort;
00045 using std::partition;
```

6.5 List/include/Stud.h File Reference

```
#include "Mylib.h"
#include "Human.h"
```

Classes

- class Stud

Functions

- void val (Stud &Lok)
- void finalgrade (Stud &Lok)
- double median (const vector< int > &ND)
- double mean (const vector< int > &ND)
- void grouping1 (list< Stud > &smart, list< Stud > &dumb, list< Stud > &list)
- void grouping2 (list< Stud > &smart, list< Stud > &list)
- void grouping3 (list< Stud > &smart, list< Stud > &list)
- bool sortByName (Stud &a, Stud &b)
- bool sortBySurname (Stud &a, Stud &b)
- bool sortDecending (Stud &a, Stud &b)
- bool sortAscending (Stud &a, Stud &b)
- void sortByChoice (list< Stud > &stud, const int &choice)
- void demonstration ()

6.5.1 Function Documentation

6.5.1.1 demonstration()

```
void demonstration ()
```

6.5.1.2 finalgrade()

```
void finalgrade (
            Stud & Lok)
```

6.5.1.3 grouping1()

```
void grouping1 (
            list< Stud > & smart,
            list< Stud > & dumb,
            list< Stud > & list)
```

6.5.1.4 grouping2()

```
void grouping2 (
            list< Stud > & smart,
            list< Stud > & list)
```

6.5.1.5 grouping3()

```
void grouping3 (
            list< Stud > & smart,
            list< Stud > & list)
```

6.5.1.6 mean()

```
double mean (
            const vector< int > & ND)
```

6.5.1.7 median()

```
double median (
            const vector< int > & ND)
```

6.5.1.8 sortAscending()

```
bool sortAscending (
            Stud & a,
            Stud & b)
```

6.5.1.9 sortByChoice()

```
void sortByChoice (
            list< Stud > & stud,
            const int & choice)
```

6.5.1.10 sortByName()

```
bool sortByName (
            Stud & a,
            Stud & b)
```

6.5.1.11 sortBySurname()

```
bool sortBySurname (
            Stud & a,
            Stud & b)
```

6.5.1.12 sortDecending()

```
bool sortDecending (
            Stud & a,
            Stud & b)
```

6.5.1.13 val()

```
void val (
            Stud & Lok)
```

6.6 Stud.h

Go to the documentation of this file.

```
00001 #include "Mylib.h"
00002 #include "Human.h"
00003
00004 class Stud : public Human {
00005 public:
00006     vector<int>ND;
00007     double vid, med, egz;
00008
00009     Stud() : vid(0), med(0), egz(0) {}
00010     Stud(string vardas, string pavarde, vector<int> nd, int egz) :
00011          Human(vardas,pavarde), ND(nd), egz(egz), vid(0), med(0) {}
00012
00013     Stud(std::istream& is) : Stud() {
00014         is » *this;
00015     }
00016
00017     //operatoriai
00018     friend istream& operator»(istream& in, Stud& student);
00019     friend ostream& operator«(ostream& out, const Stud& student);
00020
00021     //metodu perdengimas
00022     void input(Stud& Lok);
00023     void input(const string& fileName, const int& number);
00024     void input(const string& fileName, list <Stud>& stud);
00025     void output(list<Stud>& stud, int choice);
00026     void output(const string& fileName, list<Stud>& stud);
00027
00028     ~Stud() { ND.clear(); }//I. destruktorius
00029
00030     //II. copy constructor
00031     Stud(const Stud& copy)
```

```
00032         : Human(copy),
00033             ND(copy.ND),
00034             vid(copy.vid),
00035             med(copy.med),
00036             egz(copy.egz){}
00037
00039     Stud& operator=(const Stud& other) {
00040         if (this != &other) {
00041             Human::operator=(other);
00042             ND = other.ND;
00043             vid = other.vid;
00044             med = other.med;
00045             egz = other.egz;
00046         }
00047         return *this;
00048     }
00049
00050     void whoAmI() const { std::cout « "As esu is Stud klasės\n"; }
00051 };
00052
00053 void val(Stud& Lok);
00054 void finalgrade(Stud& Lok);
00055 double median(const vector <int>& ND);
00056 double mean(const vector <int>& ND);
00057 void grouping1(list<Stud>& smart, list<Stud>& dumb, list<Stud>& list);
00058 void grouping2(list<Stud>& smart, list<Stud>& list);
00059 void grouping3(list<Stud>& smart, list<Stud>& list);
00060 bool sortByName(Stud& a, Stud& b);
00061 bool sortBySurname(Stud& a, Stud& b);
00062 bool sortDecending(Stud& a, Stud& b);
00063 bool sortAscending(Stud& a, Stud& b);
00064 void sortByChoice(list<Stud>& stud, const int& choice);
00065 void demonstration();
```

6.7 List/include/Timer.h File Reference

```
#include "Mylib.h"
#include <chrono>
```

Classes

- class Timer

6.8 Timer.h

Go to the documentation of this file.

```
00001 #include "Mylib.h"
00002 #include <chrono>
00003
00004 class Timer {
00005 private:
00006     std::chrono::time_point<std::chrono::high_resolution_clock> start;
00007 public:
00008     Timer() : start{ std::chrono::high_resolution_clock::now() } {}
00009     void reset() {
00010         start = std::chrono::high_resolution_clock::now();
00011     }
00012     double elapsed() const {
00013         return std::chrono::duration<double>(std::chrono::high_resolution_clock::now() -
      start).count();
00014     }
00015 };
```

6.9 List/src/Files.cpp File Reference

```
#include "Stud.h"
#include "Timer.h"
```

6.10 List/src/Grouping.cpp File Reference

```
#include "Stud.h"
#include "Timer.h"
```

Functions

- void grouping1 (list< Stud > &smart, list< Stud > &dumb, list< Stud > &list)
- void grouping2 (list< Stud > &smart, list< Stud > &list)
- void grouping3 (list< Stud > &smart, list< Stud > &list)

6.10.1 Function Documentation

6.10.1.1 grouping1()

```
void grouping1 (
            list< Stud > & smart,
            list< Stud > & dumb,
            list< Stud > & list)
```

6.10.1.2 grouping2()

```
void grouping2 (
            list< Stud > & smart,
            list< Stud > & list)
```

6.10.1.3 grouping3()

```
void grouping3 (
            list< Stud > & smart,
            list< Stud > & list)
```

6.11 List/src/List.cpp File Reference

```
#include "Mylib.h"
#include "Stud.h"
#include "Timer.h"
```

Functions

- int main ()

6.11.1 Function Documentation

6.11.1.1 main()

```
int main ()
```

6.12 List/src/Stud.cpp File Reference

```
#include "Stud.h"
#include "Timer.h"
```

Functions

- istream & operator>> (istream &is, Stud &student)
- ostream & operator<< (ostream &out, const Stud &student)
- void val (Stud &Lok)
- void finalgrade (Stud &Lok)
- double median (const vector< int > &ND)
- double mean (const vector< int > &ND)
- bool sortByName (Stud &a, Stud &b)
- bool sortBySurname (Stud &a, Stud &b)
- bool sortDecending (Stud &a, Stud &b)
- bool sortAscending (Stud &a, Stud &b)
- void sortByChoice (list< Stud > &stud, const int &choice)
- void demonstration ()

6.12.1 Function Documentation

6.12.1.1 demonstration()

```
void demonstration ()
```

6.12.1.2 finalgrade()

```
void finalgrade (
            Stud & Lok)
```

6.12.1.3 mean()

```
double mean (
            const vector< int > & ND)
```

6.12.1.4 median()

```
double median (
            const vector< int > & ND)
```

6.12.1.5 operator<<()

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

6.12.1.6 operator>>()

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

6.12.1.7 sortAscending()

```
bool sortAscending (
            Stud & a,
            Stud & b)
```

6.12.1.8 sortByChoice()

```
void sortByChoice (
            list< Stud > & stud,
            const int & choice)
```

6.12.1.9 sortByName()

```
bool sortByName (
            Stud & a,
            Stud & b)
```

6.12.1.10 sortBySurname()

```
bool sortBySurname (
            Stud & a,
            Stud & b)
```

6.12.1.11 sortDecending()

```
bool sortDecending (
            Stud & a,
            Stud & b)
```

6.12.1.12 val()

```
void val (
            Stud & Lok)
```

6.13 List/tests/test.cpp File Reference

```
#include "gtest/gtest.h"
#include "Stud.h"
```

Functions

- TEST (Student, Constructor)
- TEST (Student, GettersAndSetters)
- TEST (Student, CopyConstructor)

6.13.1 Function Documentation

6.13.1.1 TEST() [1/3]

```
TEST (
            Student ,
            Constructor )
```

6.13.1.2 TEST() [2/3]

```
TEST (
            Student ,
            CopyConstructor )
```

6.13.1.3 TEST() [3/3]

```
TEST (
            Student ,
            GettersAndSetters )
```

6.14 README.md File Reference

Index