Chapter 1

v2.0

1.1 Diegimo instrukcija

- Atsisiųskite projekto kodą iš GitHub naudodami `git clone` komandą su projekto URL:
  `git clone https://github.com/nojusta/LabDarbas_nr2`

- Pereikite į projekto katalogą naudodami cd komandą. Pavyzdžiui:

```
cd *projekto vieta kompiuteryje*
```

- Sukurkite Makefile su reikiamomis taisyklėmis. Jūsų Makefile turėtų atrodyti maždaug taip (Unix OS atveju):

```
# Kompiliatorius
CXX = g++

# Kompiliatoriaus parametrai
CXXFLAGS = -std=c++14 -O3

# Vykdymo failo pavadinimas
TARGET = v2

# Šaltinio failai
SRCS = main.cpp functionality.cpp input.cpp calculations.cpp student.cpp

# Objektų failai
OBJS = $(SRCS:.cpp=.o)

# Taisyklė programa susieti
$(TARGET): $(OBJS)
    $(CXX) $(CXXFLAGS) -o $(TARGET) $(OBJS)

# Taisyklė kompiliuoti šaltinio failus
.cpp.o:
    $(CXX) $(CXXFLAGS) -c $<  -o $@

# Taisyklė išvalyti tarpinius failus
clean:
    $(RM) $(OBJS)

# Taisyklė išvalyti viską
distclean: clean
    $(RM) $(TARGET)
```

- Sukompiliuokite programą naudodami make komandą:
  ```
  make
  ```
  Tada gausite tokį rezultatą:

- Paleiskite programą naudodami šią komandą:
  ```
  ./v2
  ```

1.1.1 Valymo instrukcija

- Jei norite išvalyti sukompiliuotus failus, galite naudoti šias make komandas:
  ```
  make clean
  make distclean
  ```

1.2 Naudojimosi instrukcija

- Paleiskite programą naudodami šią komandą:
  ```
  ./v2
  ```

- Programa pateiks meniu su įvairiomis funkcijomis. Pasirinkite funkciją įvedę atitinkamą numerį ir spauskite `Enter`.

1.2.1 Programos funkcijos:

- Nuskaito duomenis iš naudotojo arba failo ir patikrina ar jie yra teisingi (naudojant išimčių valdymą).
- Duoda naudotojui galimybė pasirinkti du galutinio balo skaičiavimo būdus - skaičiuojant vidurkį ar medianą.
- Leidžia pasirinkti 5 skirtingus būdus įvesti, nuskaityti ar sugeneruoti duomenis.
- Dinamiškai paskiria atmintį pagal įvesta / nuskaitytą duomenų kiekį.
- Atidaro testavimo failus ir apskaičiuoja laiką, kurį praleidžia apdorojant duomenis iš failų.
- Visi pranešimai išvedami lietuvių kalbą.
- Projektas išskaidytas į kelis failus (.h ir .cpp).
- Generuoja penkis atsitiktinius studentų sąrašų failus, sudarytus isˇ: 1 000, 10 000, 100 000, 1 000 000, 10 000 000 įrasˇų
- Atlieka tyrimus / testavimus su sugeneruotais failais.
- Surūšiuoja studentus ir išveda į du naujus failus.

- Yra 3 skirtingi konteinerio tipo pasirinkimai testavimui - vector, deque, list.
- Yra 3 skirtingos strategijos duomenų skirstymui.
- Naudojama klasė, saugojant studentų duomenis.
- Galima testuoti visus "Rule of five" konstruktorius ir I/O operatorius.
- Yra abstrakti klasė Person. Student klasė yra Person klasės išvestinė klasė.
- DoxyGen sugeneruota HTML/TEX formato dokumentacija.

Norėdami baigti darbą su programa, pasirinkite atitinkamą skaičių.

1.3 Klasės naudojami "Rule of five" ir I/O operatoriai.

"Rule of five" yra C++ programavimo kalbos konceptas, kuris apima penkis pagrindinius komponentus, reikalingus objektų valdymui: destruktorius, kopijavimo konstruktorius, kopijavimo priskyrimo operatorius, perkeliamasis konstruktorius ir perkeliamasis priskyrimo operatorius. Šiame projekte "Rule of five" yra taikomas `Student` klasei.

1. **Destruktorius**: Šis komponentas naudojamas išvalyti `Student` objektą, kai jis nebereikalingas.
2. **Kopijavimo konstruktorius**: Šis komponentas leidžia sukurti naują `Student` objektą, kuris yra identiškas esamam `Student` objektui.
3. **Kopijavimo priskyrimo operatorius**: Šis operatorius leidžia priskirti vieno `Student` objekto vertę kitam `Student` objektui.
4. **Perkeliamasis konstruktorius**: Šis komponentas leidžia "perkelti" `Student` objektą, o ne kopijuoti jį. Tai yra efektyvesnis būdas sukurti naują `Student` objektą, kai turime laikiną `Student` objektą, kurio mums nebereikia.
5. **Perkeliamasis priskyrimo operatorius**: Šis operatorius leidžia "perkelti" vieno `Student` objekto vertę į kitą `Student` objektą, o ne kopijuoti ją.

Be to, šiame projekte yra naudojami įvesties (>>) ir išvesties (<<) operatoriai.

Įvesties operatorius (>>): **Šis operatorius naudojamas nuskaitant duomenis iš įvesties srauto (pvz., `std::cin` ar `std::istringstream`) į `Student` objektą.**

Išvesties operatorius (<<): Šis operatorius naudojamas rašant `Student` objektą į išvesties srautą (pvz., `std::cout` ar `std::ostringstream`).

Šie operatoriai leidžia lengvai ir efektyviai manipuliuoti `Student` objektais, nuskaitant duomenis iš įvesties srautų ir rašant juos į išvesties srautus.

Visiems konstruktoriams / operatoriams yra atlikti testai, siekiantys patikrinti ar visi jie veikia. Testus galima atlikti pasirinkus tai per meniu.

1.4 Programos sparta naudojant skirtingus kompiliatoriaus optimizavimo lygius

1.4.1 Testavimo sistemos parametrai:

- Saugykla: 256 GB, Integruota NVMe SSD
- Atmintis: 8 GB RAM
- Procesorius: Apple M1

1.4.2 Be optimizavimo testavimas

	Greitis(1mln.)	**Greitis(10mln.)**	**Failo dydis**
Struct	12.111s	130.058s	411kb
Klasė	11.358s	118.430s	404kb

Peržiūrėti
Struktūros testavimo rezultatai
Struktūros failo dydis
Klasės testavimo rezultatai
Klasės failo dydis

1.4.3 O1 optimizavimo lygio testavimas

	Greitis(1mln.)	**Greitis(10mln.)**	**Failo dydis**
Struct	2.790s	30.391s	162kb
Klasė	11.489s	117.949s	147kb

- Su struktūra greitesni testavimo rezultatai, naudojant šį optimizavimo raktą.
- Struktūros ir klasių failų dydžiai mažesni, naudojant šį optimizavimo raktą.

Peržiūrėti
Struktūros testavimo rezultatai
Struktūros failo dydis
Klasės testavimo rezultatai
Klasės failo dydis

1.4.4 O2 optimizavimo lygio testavimas

	Greitis(1mln.)	Greitis(10mln.)	Failo dydis
Struct	2.662s	30.459s	162kb
Klasė	2.767s	29.540s	147kb

- Su klase greitesni testavimo rezultatai, naudojant šį optimizavimo raktą.
- Struktūros ir klasių failų dydžiai išliko tokie patys, kaip su praeitu optimizavimo raktu.

Peržiūrėti
Struktūros testavimo rezultatai
Struktūros failo dydis
Klasės testavimo rezultatai
Klasės failo dydis

1.4.5 O3 optimizavimo lygio testavimas

	Greitis(1mln.)	Greitis(10mln.)	Failo dydis
Struct	2.734s	28.984s	161kb
Klasė	2.738s	29.735s	162kb

- Testavimo greičio rezultatai panašūs su praeitu optimizavimo raktu.
- Failų dydžiai minimaliai pasikeitė.

Peržiūrėti
Struktūros testavimo rezultatai
Struktūros failo dydis
Klasės testavimo rezultatai
Klasės failo dydis

1.5 Konteinerių testavimas

Peržiūrėti

1.5.1 Testavimo sistemos parametrai:

- Saugykla: 256 GB, Integruota NVMe SSD
- Atmintis: 8 GB RAM
- Procesorius: Apple M1

1.6 1 strategija

1.6.1 Naudojant Vector tipo konteinerius:

Failo dydis	Skaitymo laikas	Rūšiavimo laikas	Skirstymo laikas	Veikimo laikas
1 000	0.023s	0.004s	0.005s	0.033s
10 000	0.113s	0.018s	0.023s	0.155s
100 000	0.763s	0.188s	0.237s	1.189s
1 000 000	7.448s	2.105s	2.673s	12.227s
10 000 000	74.810s	24.911s	31.783s	131.505s

peržiūrėti

1.6.2 Naudojant Deque tipo konteinerius:

Failo dydis	Skaitymo laikas	Rūšiavimo laikas	Skirstymo laikas	Veikimo laikas
1 000	0.023s	0.004s	0.005s	0.034s
10 000	0.093s	0.019s	0.023s	0.136s
100 000	0.766s	0.193s	0.239s	1.198s
1 000 000	7.312s	2.160s	2.638s	12.111s
10 000 000	73.857s	25.580s	30.620s	130.058s

peržiūrėti

1.6.3 Naudojant List tipo konteinerius:

Failo dydis	Skaitymo laikas	Rūšiavimo laikas	Skirstymo laikas	Veikimo laikas
1 000	0.023s	0.003s	0.004s	0.031s
10 000	0.112s	0.018s	0.023s	0.154s
100 000	0.761s	0.247s	0.296s	1.305s
1 000 000	7.352s	3.322s	3.818s	14.493s
10 000 000	73.862s	42.406s	47.849s	164.118s

peržiūrėti

1.7 2 strategija

1.7.1 Naudojant Vector tipo konteinerius:

Failo dydis	Skaitymo laikas	Rūšiavimo laikas	Skirstymo laikas	Veikimo laikas
1 000	0.024s	0.004s	0.035s	0.064s
10 000	0.112s	0.018s	1.879s	2.010s
100 000	0.762s	0.184s	184.637s	184.637s

peržiūrėti

- Rezultatų su 1 000 000 ir 10 000 000 nėra, nes per ilgai trunka skaičiavimai (>10min).

1.7.2 Naudojant Deque tipo konteinerius:

Failo dydis	Skaitymo laikas	Rūšiavimo laikas	Skirstymo laikas	Veikimo laikas
1 000	0.021s	0.004s	0.006s	0.032s
10 000	0.114s	0.019s	0.025s	0.158s
100 000	0.765s	0.192s	0.252s	1.210s
1 000 000	7.337s	2.128s	2.772s	12.238s

peržiūrėti

- Rezultatų su 10 000 000 nėra, nes per ilgai trunka skaičiavimai (>10min).

1.7.3 Naudojant List tipo konteinerius:

Failo dydis	Skaitymo laikas	Rūšiavimo laikas	Skirstymo laikas	Veikimo laikas
1 000	0.023s	0.003s	0.001s	0.028s
10 000	0.113s	0.018s	0.005s	0.137s
100 000	0.761s	0.239s	0.061s	1.062s
1 000 000	7.433s	3.324s	0.668s	11.344s
10 000 000	72.891s	42.406s	8.586s	123.646s

peržiūrėti

1.8 3 strategija

1.8.1 Naudojant Vector tipo konteinerius:

Failo dydis	Skaitymo laikas	Rūšiavimo laikas	Skirstymo laikas	Veikimo laikas
1 000	0.023s	0.004s	0.001s	0.028s
10 000	0.112s	0.019s	0.005s	0.137s
100 000	0.758s	0.191s	0.051s	1.001s
1 000 000	7.303s	2.263s	0.622s	10.189s
10 000 000	73.373s	24.204s	8.597s	106.176s

peržiūrėti

1.9 Išvados

Remiantis atliktų testų rezultatais, galime padaryti keletą išvadų:

1. **Vector tipo konteineriai**: Vector tipo konteineriai parodė geriausius rezultatus su mažesniais failais. Tačiau, kai studentų kiekis padidėjo iki 1 000 000 ir 10 000 000, Vector tipo konteinerių veikimo laikas žymiai padidėjo, naudojant antrąją strategiją, kuri yra neefektyvi su šiuo konteineriu tipu. Pirmoji ir trečioji strategija parodė greičiausius rezultatus.
2. **Deque tipo konteineriai**: Deque tipo konteineriai parodė panašius rezultatus kaip ir Vector tipo konteineriai. Tačiau, jie buvo šiek tiek greitesni su didesniais failų dydžiais. Kaip ir su vektoriais, antroji strategija buvo neefektyvi.
3. **List tipo konteineriai**: List tipo konteineriai parodė geriausius rezultatus su didesniais failų dydžiais. Jie buvo ypač efektyvūs naudojant antrąją strategiją.

Bendra išvada yra, kad konteinerio tipo ir strategijos pasirinkimas gali turėti didelę įtaką programos veikimo laikui, ypač dirbant su didelėmis duomenų apimtimis.

1.10 Senesnių versijų aprašymai

- v.pradinė: Pradinė versija. Nuskaito vartotojo įvestį, patikrina ją, leidžia vartotojui pasirinkti tarp dviejų galutinio balo skaičiavimo būdų (vidurkis ar mediana), ir išveda duomenis ekrane.

- v0.1: Prideda galimybę pasirinkti iš 4 skirtingų būdų įvesti arba generuoti duomenis / baigti programą. Dinamiškai paskiria atmintį pagal įvestų duomenų kiekį. Išveda duomenis ekrane.
- v0.2: Prideda galimybę skaityti duomenis iš failo. Leidžia vartotojui pasirinkti iš 5 skirtingų būdų įvesti, skaityti arba generuoti duomenis. Dinamiškai skiria atmintį pagal įvestą / nuskaitytą duomenų kiekį. Atidaro testavimo failus ir apskaičiuoja laiką, kurį praleidžia apdorojant duomenis iš failų. Išveda duomenis pasirinktinai ekrane arba faile.
- v0.3: Prideda išimčių valdymą duomenų nuskaitymui. Visi pranešimai išvedami lietuvių kalba. Projektas išskaidytas į kelis failus (.h ir .cpp).
- v0.4: Prideda galimybę generuoti penkis atsitiktinius studentų sąrašų failus, sudarytus iš: 1 000, 10 000, 100 000, 1 000 000, 10 000 000 įrašų. Atlieka tyrimus / testavimus su sugeneruotais failais. Surūšiuoja studentus ir išveda į du naujus failus.
- v1.0_pradinė: Prideda 3 skirtingus konteinerio tipo pasirinkimus testavimui - vector, deque, list.
- v1.0: Yra 3 skirtingos konteinerių testavimo strategijos ir galimybė sukompiliuoti programą, naudojant Makefile.
- v1.1: Naudojamos klasės su destruktoriais ir konstruktoriais, vietoj struktūrų.
- v1.2: Prideda galimybė testuoti visus "Rule of five" konstruktorius ir I/O operatorius.
- v1.5: Abstrakti klasė Person. Student klasė yra Person klasės išvestinė klasė.

Chapter 2

Hierarchijos Indeksas

2.1 Klasių hierarchija

Šis paveldėjimo sąrašas yra beveik surikiuotas abėcėlės tvarka:

Chapter 3

Klasės Indeksas

3.1 Klasės

Klasės, struktūros, sąjungos ir sąsajos su trumpais aprašymais:

Chapter 4

Failo Indeksas

4.1 Failai

Visų dokumentuotų failų sąrašas su trumpais aprašymais:

Chapter 5

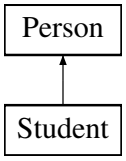
Klasės Dokumentacija

5.1 Person Klasė

Klasė, aprašanti žmogų.

`#include <person.h>`

Paveldimumo diagrama Person:



Vieši Metodai

- virtual ∼**Person** ()=default

  Virtualus destruktorius.
- virtual std::string getFirstName () const =0

  Virtualus metodas, grąžinantis asmens vardą.
- virtual std::string getLastName () const =0

  Virtualus metodas, grąžinantis asmens pavardę.
- virtual std::string getName () const =0

  Virtualus metodas, grąžinantis pilną asmens vardą ir pavardę.

5.1.1 Smulkus aprašymas

Klasė, aprašanti žmogų.

Ši klasė yra abstrakti bazinė klasė, nuo kurios paveldės Student klasė.

5.1.2 Metodų Dokumentacija

getFirstName()

`virtual std::string Person::getFirstName ( ) const  [pure virtual]`

Virtualus metodas, grąžinantis asmens vardą.

Gražina

Asmens vardas.

Realizuota Student.

getLastName()

`virtual std::string Person::getLastName ( ) const  [pure virtual]`

Virtualus metodas, grąžinantis asmens pavardę.

Gražina

Asmens pavardė.

Realizuota Student.

getName()

`virtual std::string Person::getName ( ) const  [pure virtual]`

Virtualus metodas, grąžinantis pilną asmens vardą ir pavardę.

Gražina

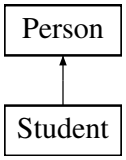
Pilnas asmens vardas ir pavardė.

Realizuota Student.
Dokumentacija šiai klasei sugeneruota iš šio failo:

- person.h

5.2 Student Klasė

Klasė, aprašanti studentą.
`#include <student.h>`
Paveldimumo diagrama Student:



Vieši Metodai

- **Student** ()

  Konstruktorius be parametrų.
- Student (const std::string &firstName, const std::string &lastName, int examResults, const std::vector< int > &homeworkResults)

  Konstruktorius su parametrais.
- Student (const Student &other)

  Kopijavimo konstruktorius.
- Student (Student &&other) noexcept

  Perkėlimo konstruktorius.
- Student & operator= (const Student &other)

  Kopijavimo priskyrimo operatorius.
- Student & operator= (Student &&other) noexcept

  Perkėlimo priskyrimo operatorius.
- ~**Student** ()

  Destruktorius.
- std::string getFirstName () const

  Grąžina studento vardą.
- std::string getLastName () const

  Grąžina studento pavardę.
- std::string getName () const

  Grąžina studento vardą ir pavardę.
- const std::vector< int > & getHomeworkResults () const

  Grąžina studento namų darbų rezultatus.
- int getExamResults () const

  Grąžina studento egzamino rezultatą.
- int getExamGrade () const

  Grąžina studento egzamino pažymį.
- void **removeLastHomeworkGrade** ()

  Pašalina paskutinį namų darbo pažymį.
- void setFirstName (std::string firstName)

  Nustato studento vardą.
- void setLastName (std::string lastName)

  Nustato studento pavardę.
- void addHomeworkResult (int result)

  Prideda namų darbo rezultatą.
- void **clearHomeworkResults** ()

  Išvalo namų darbų rezultatus.
- void setExamResults (int examResults)

Nustato egzamino rezultatą.

- void setHomeworkResults (std::vector< int > results)

  Nustato namų darbų rezultatus.
- double calculateMedian () const

  Skaičiuoja medianą iš namų darbų rezultatų.
- double calculateAverage () const

  Skaičiuoja vidurkį iš namų darbų rezultatų.
- double calculateFinalGrade (bool median) const

  Skaičiuoja galutinį pažymį.

Vieši Metodai inherited from Person

- virtual ~**Person** ()=default

  Virtualus destruktorius.

Draugai

- std::istream & operator>> (std::istream &is, Student &s)

  Įvesties operatorius.
- std::ostream & operator<< (std::ostream &os, const Student &s)

  Išvesties operatorius.

5.2.1 Smulkus aprašymas

Klasė, aprašanti studentą.

Ši klasė paveldi Person klasę ir prideda papildomus studento atributus.

5.2.2 Konstruktoriaus ir Destruktoriaus Dokumentacija

Student() [1/3]

```
Student::Student (
              const std::string & firstName,
              const std::string & lastName,
              int examResults,
              const std::vector< int > & homeworkResults )
```

Konstruktorius su parametrais.

Parametrai

firstName	Studento vardas.
lastName	Studento pavardė.
examResults	Studento gzamino rezultatas.
homeworkResults	Studento namų darbų rezultatai.

Student() [2/3]

```
Student::Student (
            const Student & other )
```

Kopijavimo konstruktorius.

Parametrai

other	Kita Student klasės objekto instancija.

Student() [3/3]

```
Student::Student (
            Student && other )  [noexcept]
```

Perkėlimo konstruktorius.

Parametrai

other	Kita Student klasės objekto instancija.

5.2.3 Metodų Dokumentacija

addHomeworkResult()

```
void Student::addHomeworkResult (
            int result )  [inline]
```

Prideda namų darbo rezultatą.

Parametrai

result	Namų darbo rezultatas.

calculateAverage()

```
double Student::calculateAverage ( ) const
```

Skaičiuoja vidurkį iš namų darbų rezultatų.

Gražina

Vidurkis.

calculateFinalGrade()

```
double Student::calculateFinalGrade (
            bool median ) const
```

Skaičiuoja galutinį pažymį.

Parametrai

median	Jei true, naudoja medianą, jei false, naudoja vidurkį.

Gražina

Galutinis pažymys.

calculateMedian()

```
double Student::calculateMedian ( ) const
```

Skaičiuoja medianą iš namų darbų rezultatų.

Gražina

Mediana.

getExamGrade()

```
int Student::getExamGrade ( ) const  [inline]
```

Grąžina studento egzamino pažymį.

Gražina

Egzamino pažymys.

getExamResults()

```
int Student::getExamResults ( ) const  [inline]
```

Grąžina studento egzamino rezultatą.

Gražina

Egzamino rezultatas.

getFirstName()

```
std::string Student::getFirstName ( ) const  [inline], [virtual]
```

Grąžina studento vardą.

Gražina

Studento vardas.

Realizuoja Person.

getHomeworkResults()

```
const std::vector< int > & Student::getHomeworkResults ( ) const  [inline]
```

Grąžina studento namų darbų rezultatus.

Gražina

Namų darbų rezultatai.

getLastName()

```
std::string Student::getLastName ( ) const  [inline], [virtual]
```

Grąžina studento pavardę.

Gražina

Studento pavardė.

Realizuoja Person.

getName()

```
std::string Student::getName ( ) const  [inline], [virtual]
```

Grąžina studento vardą ir pavardę.

Gražina

Studento vardas ir pavardė.

Realizuoja Person.

operator=() [1/2]

```
Student & Student::operator= (
            const Student & other )
```

Kopijavimo priskyrimo operatorius.

Parametrai

other	Kita Student klasės objekto instancija.

Gražina

Grąžina ∗this.

operator=() [2/2]

```
Student & Student::operator= (
            Student && other )  [noexcept]
```

Perkėlimo priskyrimo operatorius.

Parametrai

other	Kita Student klasės objekto instancija.

Gražina

Grąžina ∗this.

setExamResults()

```
void Student::setExamResults (
              int examResults )  [inline]
```

Nustato egzamino rezultatą.

Parametrai

examResults	Egzamino rezultatas.

setFirstName()

```
void Student::setFirstName (
              std::string firstName )  [inline]
```

Nustato studento vardą.

Parametrai

firstName	Studento vardas.

setHomeworkResults()

```
void Student::setHomeworkResults (
              std::vector< int > results )  [inline]
```

Nustato namų darbų rezultatus.

Parametrai

results	Namų darbų rezultatai.

setLastName()

```
void Student::setLastName (
              std::string lastName )  [inline]
```

Nustato studento pavardę.

Parametrai

lastName	Studento pavardė.

5.2.4 Draugiškų Ir Susijusių Funkcijų Dokumentacija

operator<<

```
std::ostream & operator<< (
              std::ostream & os,
              const Student & s )  [friend]
```

Išvesties operatorius.

Parametrai

os	Išvesties srautas.
s	Student klasės objekto instancija.

Gražina

Grąžina išvesties srautą.

operator>>

```
std::istream & operator>> (
              std::istream & is,
              Student & s )  [friend]
```

Įvesties operatorius.

Parametrai

is	Įvesties srautas.
s	Student klasės objekto instancija.

Gražina

Grąžina įvesties srautą.

Dokumentacija šiai klasei sugeneruota iš šių failų:

- student.h
- student.cpp

Chapter 6

Failo Dokumentacija

6.1 calculations.h

```
00001 #ifndef CALCULATIONS_H
00002 #define CALCULATIONS_H
00003
00004 #include <vector>
00005 #include "student.h"
00006 #include "input.h"
00007
00008 bool compareByFirstName(const Student& a, const Student& b);
00009
00010 bool compareByLastName(const Student& a, const Student& b);
00011
00012 bool compareByGrade(const Student& a, const Student& b);
00013
00014 double calculateAverage(const std::vector<int>& homeworkResults);
00015
00016 double calculateFinalGrade(const Student& data, bool Median);
00017
00018 double calculateMedian(std::vector<int> homeworkResults);
00019
00020 template <typename Container>
00021 void sortStudents(Container& students, int criteria);
00022
00023 template <>
00024 void sortStudents<std::list<Student>>(std::list<Student>& students, int criteria);
00025
00026 #endif // CALCULATIONS_H
```

6.2 functionality.h

```
00001 #ifndef FUNCTIONALITY_H
00002 #define FUNCTIONALITY_H
00003
00004 #include <string>
00005 #include <cstdlib>
00006 #include <vector>
00007 #include "student.h"
00008 #include "calculations.h"
00009
00010 int getContainerTypeFromUser();
00011 int generateGrade();
00012 std::string generateName();
```

```
00013 std::string generateLastName();
00014 std::string isString(const std::string& prompt);
00015 int isGrade(const std::string& prompt);
00016 void generateFile(int n);
00017 void outputToTerminal(const std::vector<Student>& studentsLow, const
      std::vector<Student>& studentsHigh, bool Median);
00018 template <typename Container>
00019 void outputToFile(const Container& students, size_t m, bool Median, const std::string&
      filename);
00020
00021 #endif // FUNCTIONALITY_H
```

6.3 input.h

```
00001 #ifndef INPUT_H
00002 #define INPUT_H
00003
00004 #include <string>
00005 #include <vector>
00006 #include <fstream>
00007 #include "student.h"
00008 #include "calculations.h"
00009 #include "functionality.h"
00010
00011 bool getMedianPreference();
00012
00013 template <typename Container>
00014 void processStudents(Container& students, bool Median,
      std::chrono::high_resolution_clock::time_point startTotal);
00015
00016 int Menu();
00017
00018 std::string getFilenameFromUser();
00019
00020 template <typename Container>
00021 void readData(std::ifstream& fin, Container& students);
00022
00023 template <typename Container>
00024 void openFiles(const std::vector<std::string>& filenames, Container& students, bool
      Median, int strategy);
00025
00026 void input(Student& data, bool& Median);
00027
00028 std::string studentData(const Student& s);
00029
00030 #endif // INPUT_H
```

6.4 person.h

```
00001 #ifndef PERSON_H
00002 #define PERSON_H
00003
00004 #include <string>
00005 #include <vector>
00006 #include <iostream>
00007 #include <algorithm>
00008 #include <numeric>
00009 #include <deque>
00010 #include <list>
00011 #include <istream>
00012 #include <ostream>
00013 #include <iomanip>
00014 #include <sstream>
```

```
00015
00022 class Person {
00023 public:
00027     virtual ~Person() = default;
00028
00033     virtual std::string getFirstName() const = 0;
00034
00039     virtual std::string getLastName() const = 0;
00040
00045     virtual std::string getName() const = 0;
00046 };
00047
00048 #endif // PERSON_H
```

6.5 student.h

```
00001 #ifndef STUDENT_H
00002 #define STUDENT_H
00003
00004 #include "person.h"
00005
00012 class Student : public Person {
00013 private:
00014     std::string firstName, lastName;
00015     std::vector<int> homeworkResults;
00016     int examResults;
00017
00018 public:
00022     Student();
00023
00031     Student(const std::string &firstName, const std::string &lastName, int
      examResults, const std::vector<int> &homeworkResults);
00032
00037     Student(const Student &other);
00038
00043     Student(Student &&other) noexcept;
00044
00050     Student &operator=(const Student &other);
00051
00057     Student &operator=(Student &&other) noexcept;
00058
00065     friend std::istream &operator»(std::istream &is, Student &s);
00066
00073     friend std::ostream &operator«(std::ostream &os, const Student &s);
00074
00078     ~Student() {
00079         homeworkResults.clear();
00080     }
00081
00082     // Get'eriai
00087     inline std::string getFirstName() const { return firstName; }
00088
00093     inline std::string getLastName() const { return lastName; }
00094
00099     std::string getName() const { return getFirstName() + " " + getLastName(); }
00100
00105     const std::vector<int> &getHomeworkResults() const { return homeworkResults; }
00106
00111     int getExamResults() const { return examResults; }
00112
00117     int getExamGrade() const { return homeworkResults.back(); }
00118
00122     void removeLastHomeworkGrade() { if (!homeworkResults.empty()) {
      homeworkResults.pop_back(); } }
00123
00124     // Set'eriai
```

```
00129     void setFirstName(std::string firstName) { this->firstName = std::move(firstName);
      }
00130
00135     void setLastName(std::string lastName) { this->lastName = std::move(lastName); }
00136
00141     void addHomeworkResult(int result) { homeworkResults.push_back(result); }
00142
00146     void clearHomeworkResults() { homeworkResults.clear(); }
00147
00152     void setExamResults(int examResults) { this->examResults = examResults; }
00153
00158     void setHomeworkResults(std::vector<int> results) { homeworkResults =
      std::move(results); }
00159
00160     // Funkcijos
00165     double calculateMedian() const;
00166
00171     double calculateAverage() const;
00172
00178     double calculateFinalGrade(bool median) const;
00179 };
00180
00181 #endif // STUDENT_H
```