UniversityCourseV2

v2.0

Generated by Doxygen 1.10.0

Chapter 1

UniversityCourseV2

Objektinio programavimo antroji užduotis

1.1 Spartos analize

1.1.1 Be oprimizatoriaus

100.000 studentų
Struct. Student sorting to two groups took 0.0304037 seconds.
Class. Student sorting to two groups took 0.0443925 seconds.
1.000.000 studentų
Struct. Student sorting to two groups took 0.319136 seconds.
Class. Student sorting to two groups took 0.600509 seconds.

1.1.2 Kompiliatoriaus optimizavimo lygiai

1.1.2.1 100.000 studentų

Optimization	Struct	Class	File size Struct	File size Class
O1	0.0117796 seconds.	0.0195176 seconds.	262 KB	254 KB
O2	0.0114957 seconds.	0.0205766 seconds.	266 KB	256 KB
O3	0.0108180 seconds.	0.0202935 seconds.	264 KB	259 KB

1.1.2.2 1.000.000 studentų

Optimization	Struct	Class	File size Struct	File size Class
O1	0.127661 seconds.	0.345793 seconds.	262 KB	254 KB
O2	0.115778 seconds.	0.339243 seconds.	266 KB	256 KB
O3	0.115472 seconds.	0.333918 seconds.	264 KB	259 KB

1.2 Studentas Class

1.2.1 Class Structure

`Zmogus` abstrakčioje klasėje yra šie nariai:

- `Vardas_`: String tipo studento vardas.
- `Pavarde_`: String tipo studento pavarde.

`Studentas` klasėje yra šie nariai:

- `egzaminas_`: Double tipo egzamino pažymys.
- `namudarbas_`: Double vectoriaus tipo namų darbų pažymiai.

1.2.2 Data input

Studento duomenų (Vardas, Pavarde, Namu darbas, Egzaminas) rankinis įvedimas:

```
.setVardas();
.setPavarde();
.setNamudarbas();
.setEgzaminas();
```

```
Studentas data;
data.setVardas("John");
data.setPavarde("Doe");
data.setNamudarbas({8, 6, 9, 5});
data.setEgzaminas(7);
```

Studento duomenų (Vardas, Pavarde, Namu darbas, Egzaminas) programoje įvedimo pvz:

```
Studentas data;
cout << "Iveskite studento varda: ";
cin >> temp;
data.setVardas(temp);

cout << "Iveskite studento pavarde: ";
cin >> temp;
data.setPavarde(temp);

vector<double> namudarbas;
int NumberOfInputs = 5;
for (int i = 0; i < NumberOfInputs; i++) {
    cout << "Iveskite studento namu darbo rezultata: ";
    cin >> temp;
    namudarbas.push_back(temp);
}
duom.setNamudarbas(namudarbas);

cout << "Iveskite studento egzamino rezultata: ";
cin >> temp;
data.setEgzaminas(temp);
```

Studento duomenų (Vardas, Pavarde, Namu darbas, Egzaminas) is failo įvedimas pagal formata kuris pateiktas failuose pavyzdys. Pvz. 1000_GeneratedStudents.txt:

```
vector<Studentas> student;
ifstream inputFile("1000_GeneratedStudents.txt");

string line;
getline(inputFile, line); // Naikinama pirma header eilute

// Skaitoma ir apdorojama visa eilute
while (getline(inputFile, line)) {
    Studentas duom;
    istringstream iss(line);
    duom.readStudent(iss);
    student.push_back(duom);
}
```

1.2.3 Data output

Studento duomenų (Vardas, Pavarde, Namu darbas, Egzaminas, GalutinisMed, GalutinisVid) išvedimas:
`.Vardas();`
`.Pavarde();`
`.Namudarbas();`
`.Egzaminas();`
`GalutinisMed();`
`GalutinisVid();`

```
Studentas data;
cout << "Studento vardas " << data.Vardas() << endl;
cout << "Studento pavarde " << data.Pavarde() << endl;
cout << "Studento pazymiai ";
for (double grade : data.Namudarbas()) {
    std::cout << grade << " ";
}
cout << "Studento egzamino rezultatas " << data.Egzaminas() << endl;
cout << "Galutinis vidurkis naudojant mediana " << GalutinisMed(data);
cout << "Galutinis vidurkis paprastu budu " << GalutinisVid(data);
```

1.2.4 Usage

Norint naudotis `Studentas` klase, turite atsisiųsti failus `Studentas.h` `Studentas.cpp` ir `Zmogus.h`. Savo kode turite inlcudinti `Studentas.h`.

```
#include "Studentas.h"
```

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 RandInt Class Reference

Public Member Functions

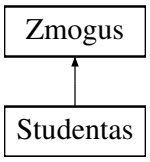
- **RandInt** (int low, int high)
- int **operator()** ()

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

- RandInt.hpp

5.2 Studentas Class Reference

Inheritance diagram for Studentas:



Public Member Functions

- **Studentas** (const std::string &vardas, const std::string &pavarde, double egzaminas, const std::vector< double > &namudarbas)
- **Studentas** (const Studentas &other)
- **Studentas** (Studentas &&other) noexcept
- Studentas & **operator=** (const Studentas &other)
- Studentas & **operator=** (Studentas &&other) noexcept
- std::string Vardas () const
- std::string Pavarde () const
- double **Egzaminas** () const
- std::vector< double > **Namudarbas** () const
- void setVardas (const std::string &vardas)
- void setPavarde (const std::string &pavarde)
- void **setEgzaminas** (double egzaminas)
- void **setNamudarbas** (const std::vector< double > &namudarbas)
- std::istream & **readStudent** (std::istream &)
- virtual void doSomething ()

Public Member Functions inherited from Zmogus

- **Zmogus** (const std::string &vardas, const std::string &pavarde)

Static Public Member Functions

- static double **Vidurkis** (const std::vector< double > &namudarbas)
- static double **Mediana** (const std::vector< double > &namudarbas)

Friends

- double **GalutinisVid** (const Studentas &duom)
- double **GalutinisMed** (const Studentas &duom)

Additional Inherited Members

Protected Attributes inherited from Zmogus

- std::string **Vardas_**
- std::string **Pavarde_**

5.2.1 Member Function Documentation

5.2.1.1 doSomething()

```
void Studentas::doSomething ( )  [virtual]
```

Implements Zmogus.

5.2.1.2 Pavarde()

```
std::string Studentas::Pavarde ( ) const  [inline], [virtual]
```

Reimplemented from Zmogus.

5.2.1.3 setPavarde()

```
void Studentas::setPavarde (
            const std::string & pavarde )  [virtual]
```

Reimplemented from Zmogus.

5.2.1.4 setVardas()

```
void Studentas::setVardas (
            const std::string & vardas )  [virtual]
```

Reimplemented from Zmogus.

5.2.1.5 Vardas()

```
std::string Studentas::Vardas ( ) const  [inline], [virtual]
```

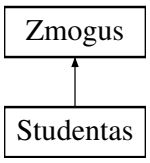
Reimplemented from Zmogus.

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

- Studentas.h
- Studentas.cpp

5.3 Zmogus Class Reference

Inheritance diagram for Zmogus:



Public Member Functions

- **Zmogus** (const std::string &vardas, const std::string &pavarde)
- virtual std::string **Vardas** () const
- virtual std::string **Pavarde** () const
- virtual void **setVardas** (const std::string &vardas)
- virtual void **setPavarde** (const std::string &pavarde)
- virtual void **doSomething** ()=0

Protected Attributes

- std::string **Vardas_**
- std::string **Pavarde_**

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

- Zmogus.h

Chapter 6

File Documentation

6.1 RandInt.hpp

```
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 int'ą
00007   private:
00008     std::random_device rd;
00009     std::mt19937 mt;
00010     std::uniform_int_distribution<int> dist;
00011 };
```

6.2 Studentas.h

```
00001 #ifndef STUDENTAS_H
00002 #define STUDENTAS_H
00003
00004 #include <algorithm>
00005 #include <string>
00006 #include <vector>
00007 #include <iostream>
00008 #include "Zmogus.h"
00009
00010 class Studentas : public Zmogus{
00011 private:
00012     double egzaminas_;
00013     std::vector<double> namudarbas_;
00014
00015 public:
00016     // Default konstruktoriai
00017     Studentas() : Zmogus(), egzaminas_(0), namudarbas_({}) {}
00018     Studentas(const std::string& vardas, const std::string& pavarde, double egzaminas, const
      std::vector<double>& namudarbas) : Zmogus(vardas, pavarde), egzaminas_(egzaminas),
      namudarbas_(namudarbas) {}
00019
00020     // Rule of Five
00021     ~Studentas();
00022     Studentas(const Studentas& other); // Copy constructor
00023     Studentas(Studentas&& other) noexcept; // Move constructor
00024     Studentas& operator=(const Studentas& other); // Copy assignment
00025     Studentas& operator=(Studentas&& other) noexcept; // Move assignment
00026
00027     // Outputeriai
00028     std::string Vardas() const { return Vardas_; }
00029     std::string Pavarde() const { return Pavarde_; }
00030     double Egzaminas() const { return egzaminas_; }
00031     std::vector<double> Namudarbas() const { return namudarbas_; }
00032
00033     // Set'eriai
00034     void setVardas(const std::string& vardas);
00035     void setPavarde(const std::string& pavarde);
00036     void setEgzaminas(double egzaminas);
00037     void setNamudarbas(const std::vector<double>& namudarbas);
00038     std::istream& readStudent(std::istream&);
```

```
00039
00040     // Calculations
00041     friend double GalutinisVid(const Studentas& duom);
00042     friend double GalutinisMed(const Studentas& duom);
00043
00044     // Static functions for calculations
00045     static double Vidurkis(const std::vector<double>& namudarbas);
00046     static double Mediana(const std::vector<double>& namudarbas);
00047
00048     // Virtualios funkcijos deklaracija, kad klase Zmogus butu abstrakti
00049     virtual void doSomething();
00050 };
00051 #endif
```

6.3 VectorHeader.h

```
00001 #ifndef HEADER_H // redefinition apsauga
00002 #define HEADER_H
00003
00004 #include "VectorIncludes.h"
00005 #include "studentas.h"
00006
00007 void input1(Studentas& duom);
00008 void input2(Studentas& duom);
00009 void input3(Studentas& duom, int n);
00010 void OutputBy(const vector<Studentas>& student);
00011 void manualmode();
00012 void readingmode(const string& fileName);
00013 void filegeneration();
00014 void SplitVector(const vector<Studentas>& student);
00015 void SplitVector2(vector<Studentas>& student);
00016 void SplitVector3(vector<Studentas>& student);
00017 int NumberVerification(const string& prompt, int minValue, int maxValue);
00018 int NumberVerification(const string& prompt, int minValue);
00019 int YesNoVerification(const string& prompt);
00020
00021 #endif // HEADER_H
```

6.4 VectorIncludes.h

```
00001 #ifndef INCLUDES_H // redefinition apsauga
00002 #define INCLUDES_H
00003
00004 // Standard Libraries
00005 #include <fstream>
00006 #include <iostream>
00007 #include <sstream>
00008 #include <vector>
00009 #include <iomanip>
00010 #include <string>
00011 #include <algorithm>
00012 #include <chrono>
00013 #include <cstdlib>
00014 #include <ctime>
00015 #include <random>
00016
00017 using namespace std;
00018 using namespace std::chrono;
00019
00020 // Custom Libraries
00021 #include "RandInt.hpp"
00022
00023 #endif // INCLUDES_H
```

6.5 Zmogus.h

```
00001 #ifndef ZMOGUS_H
00002 #define ZMOGUS_H
00003
00004 class Zmogus {
00005 protected:
00006     std::string Vardas_;
00007     std::string Pavarde_;
00008
00009 public:
```

```
00010     // Default konstruktorius
00011     Zmogus() : Vardas_(""), Pavarde_("") {}
00012     Zmogus(const std::string& vardas, const std::string& pavarde) : Vardas_(vardas), Pavarde_(pavarde)
      {}
00013
00014     // Virtualus destruktorius
00015     virtual ~Zmogus() {}
00016
00017     // Virtualus outputeriai
00018     virtual std::string Vardas() const { return Vardas_; }
00019     virtual std::string Pavarde() const { return Pavarde_; }
00020
00021     // Virtualus set'eriai
00022     virtual void setVardas(const std::string& vardas) { Vardas_ = vardas; }
00023     virtual void setPavarde(const std::string& pavarde) { Pavarde_ = pavarde; }
00024
00025     // Virtuali funkcija be realizacijos tam, kad klase Zmogus butu abstrakti
00026     virtual void doSomething() = 0;
00027 };
00028
00029 #endif
```

Index