# OOP projektas

Sugeneruota Doxygen 1.13.2

# skyrius 1

# Hierarchijos Indeksas

## 1.1 Klasių hierarchija

Šis paveldėjimo sąrašas yra beveik surikiuotas abėcėlės tvarka:

# skyrius 2

# Klasės Indeksas

## 2.1 Klasės

Klasės, struktūros, sąjungos ir sąsajos su trumpais aprašymais:

# skyrius 3

# Failo Indeksas

## 3.1 Failai

Visų failų sąrašas su trumpais aprašymais:

# skyrius 4

# Klasės Dokumentacija

## 4.1 duom Klasė

```
#include <template.h>
```

Paveldimumo diagrama duom:

## 4.2 human Klasė

```
#include <template.h>
```

Paveldimumo diagrama human:

#### Vieši Metodai

- virtual void abstractClassFunction ()=0

#### Vieši Atributai

- string var = ""
- string pav = ""

### 4.2.1 Metodų Dokumentacija

#### 4.2.1.1 abstractClassFunction()

```
virtual void human::abstractClassFunction () [pure virtual]
```

Realizuota duom.

### 4.2.2 Atributų Dokumentacija

#### 4.2.2.1 pav

```
string human::pav = ""
```

#### 4.2.2.2 var

```
string human::var = ""
```

Dokumentacija šiai klasei sugeneruota iš šio failo:

- template.h

## 4.3 temp Struktūra

```
#include <template.h>
```

**Vieši Atributai**

- string var ="test"
- string pav ="test"
- vector< int > pazymiai
- int exam =0
- double vid_med =0
- double mark

### 4.3.1 Atributų Dokumentacija

#### 4.3.1.1 exam

```
int temp::exam =0
```

#### 4.3.1.2 mark

```
double temp::mark
```

#### 4.3.1.3 pav

```
string temp::pav ="test"
```

#### 4.3.1.4 pazymiai

```
vector<int> temp::pazymiai
```

#### 4.3.1.5 var

```
string temp::var ="test"
```

#### 4.3.1.6 vid_med

```
double temp::vid_med =0
```

Dokumentacija šiai struktūrai sugeneruota iš šio failo:

- template.h

skyrius 5

# Failo Dokumentacija

## 5.1 build/CMakeFiles/4.0.0-rc4/CompilerIdC/CMakeCCompilerId.c Failo Nuoroda

### Apibrėžimai

- #define __has_include(x)
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X)
- #define STRINGIFY(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_STD_99 199901L
- #define C_STD_11 201112L
- #define C_STD_17 201710L
- #define C_STD_23 202311L
- #define C_VERSION

### Funkcijos

- int main (int argc, char ∗argv[ ])

### Kintamieji

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

### 5.1.1 Apibrėžimų Dokumentacija

#### 5.1.1.1 __has_include

```
#define __has_include(
             x)
```

**Reikšmė:**

```
0
```

#### 5.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

#### 5.1.1.3 C_STD_11

```
#define C_STD_11 201112L
```

#### 5.1.1.4 C_STD_17

```
#define C_STD_17 201710L
```

#### 5.1.1.5 C_STD_23

```
#define C_STD_23 202311L
```

#### 5.1.1.6 C_STD_99

```
#define C_STD_99 199901L
```

#### 5.1.1.7 C_VERSION

```
#define C_VERSION
```

#### 5.1.1.8 COMPILER_ID

```
#define COMPILER_ID ""
```

#### 5.1.1.9 DEC

```
#define DEC(
               n)
```

**Reikšmė:**

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

#### 5.1.1.10 HEX

```
#define HEX(
               n)
```

**Reikšmė:**

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

#### 5.1.1.11 PLATFORM_ID

```
#define PLATFORM_ID
```

#### 5.1.1.12 STRINGIFY

```
#define STRINGIFY(
               X)
```

**Reikšmė:**

```
STRINGIFY_HELPER(X)
```

#### 5.1.1.13 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
               X)
```

**Reikšmė:**

```
#X
```

### 5.1.2 Funkcijos Dokumentacija

#### 5.1.2.1 main()

```
int main (
              int argc,
              char * argv[])
```

### 5.1.3 Kintamojo Dokumentacija

#### 5.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

#### 5.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

#### 5.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

**Pradinė reikšmė:**

```
= "INFO" ":" "extensions_default["




  "OFF"
"]"
```

#### 5.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

**Pradinė reikšmė:**

```
=
  "INFO" ":" "standard_default[" C_VERSION "]"
```

#### 5.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

## 5.2 build/CMakeFiles/4.0.0-rc4/CompilerIdCXX/CMakeCXXCompiler↩Id.cpp Failo Nuoroda

**Apibrėžimai**

- #define __has_include(x)
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X)
- #define STRINGIFY(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define CXX_STD_98 199711L
- #define CXX_STD_11 201103L
- #define CXX_STD_14 201402L
- #define CXX_STD_17 201703L
- #define CXX_STD_20 202002L
- #define CXX_STD_23 202302L
- #define CXX_STD __cplusplus

**Funkcijos**

- int main (int argc, char ∗argv[ ])

**Kintamieji**

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

## 5.2.1 Apibrėžimų Dokumentacija

### 5.2.1.1 __has_include

```
#define __has_include(
            x)
```

**Reikšmė:**

```
0
```

### 5.2.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

### 5.2.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

### 5.2.1.4 CXX_STD

```
#define CXX_STD __cplusplus
```

### 5.2.1.5 CXX_STD_11

```
#define CXX_STD_11 201103L
```

### 5.2.1.6 CXX_STD_14

```
#define CXX_STD_14 201402L
```

### 5.2.1.7 CXX_STD_17

```
#define CXX_STD_17 201703L
```

#### 5.2.1.8 CXX_STD_20

```
#define CXX_STD_20 202002L
```

#### 5.2.1.9 CXX_STD_23

```
#define CXX_STD_23 202302L
```

#### 5.2.1.10 CXX_STD_98

```
#define CXX_STD_98 199711L
```

#### 5.2.1.11 DEC

```
#define DEC(
              n)
```

**Reikšmė:**

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

#### 5.2.1.12 HEX

```
#define HEX(
              n)
```

**Reikšmė:**

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

#### 5.2.1.13 PLATFORM_ID

```
#define PLATFORM_ID
```

#### 5.2.1.14 STRINGIFY

```
#define STRINGIFY(
              X)
```

**Reikšmė:**

```
STRINGIFY_HELPER(X)
```

#### 5.2.1.15 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X)
```

**Reikšmė:**

```
#X
```

### 5.2.2 Funkcijos Dokumentacija

#### 5.2.2.1 main()

```
int main (
             int argc,
             char * argv[])
```

### 5.2.3 Kintamojo Dokumentacija

#### 5.2.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

#### 5.2.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

#### 5.2.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

**Pradinė reikšmė:**

```
= "INFO" ":" "extensions_default["




  "OFF"

"]"
```

#### 5.2.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

**Pradinė reikšmė:**

```
= "INFO" ":" "standard_default["










  "98"

"]"
```

#### 5.2.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

## 5.3 main.cpp Failo Nuoroda

```
#include "template.h"
```

Įtraukimo priklausomybių diagrama main.cpp:

**Funkcijos**

- int main ()
- void menu (vector< duom > &grupe)
- void read_from_console (vector< duom > &grupe)
- void read_file (vector< duom > &grupe, string filename)
- void read_names_from_console (vector< duom > &grupe)
- void random_grades (vector< duom > &grupe, int m)
- void random_names_grades (vector< duom > &grupe, int record_amount, int mark_amount)
- void print_data_to_file (vector< duom > &grupe, int mark_amount, string filename)
- void print_answers_to_file (vector< duom > &grupe, string filename)
- void print_answers_console (vector< duom > &grupe)
- char check_menu ()
- void vid_med_calc (vector< duom > &grupe)
- bool compare (const string a, const string b, string rule)
- void sorting (vector< duom > &grupe, char rule)
- double average (duom given)
- double median (duom given)
- void method_test (vector< duom > &grupe)

### 5.3.1 Funkcijos Dokumentacija

#### 5.3.1.1 average()

```
double average (
            duom given)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

#### 5.3.1.2 check_menu()

```
char check_menu ()
```

Here is the caller graph for this function:

#### 5.3.1.3 compare()

```
bool compare (
             const string a,
             const string b,
             string rule)
```

Here is the caller graph for this function:

#### 5.3.1.4 main()

```
int main ()
```

Funkcijos kvietimo grafas:

#### 5.3.1.5 median()

```
double median (
             duom given)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

#### 5.3.1.6 menu()

```
void menu (
             vector< duom > & grupe)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

#### 5.3.1.7 method_test()

```
void method_test (
             vector< duom > & grupe)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

#### 5.3.1.8 print_answers_console()

```
void print_answers_console (
             vector< duom > & grupe)
```

Here is the caller graph for this function:

#### 5.3.1.9 print_answers_to_file()

```
void print_answers_to_file (
             vector< duom > & grupe,
             string filename)
```

Here is the caller graph for this function:

#### 5.3.1.10 print_data_to_file()

```
void print_data_to_file (
            vector< duom > & grupe,
            int mark_amount,
            string filename)
```

Here is the caller graph for this function:

#### 5.3.1.11 random_grades()

```
void random_grades (
            vector< duom > & grupe,
            int m)
```

Here is the caller graph for this function:

#### 5.3.1.12 random_names_grades()

```
void random_names_grades (
            vector< duom > & grupe,
            int record_amount,
            int mark_amount)
```

Here is the caller graph for this function:

#### 5.3.1.13 read_file()

```
void read_file (
            vector< duom > & grupe,
            string filename)
```

Here is the caller graph for this function:

#### 5.3.1.14 read_from_console()

```
void read_from_console (
            vector< duom > & grupe)
```

Here is the caller graph for this function:

#### 5.3.1.15 read_names_from_console()

```
void read_names_from_console (
            vector< duom > & grupe)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

#### 5.3.1.16 sorting()

```
void sorting (
             vector< duom > & grupe,
             char rule)
```

Here is the caller graph for this function:

#### 5.3.1.17 vid_med_calc()

```
void vid_med_calc (
             vector< duom > & grupe)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

## 5.4 std.h Failo Nuoroda

Šis grafas rodo, kuris failas tiesiogiai ar netiesiogiai įtraukia šį failą:

## 5.5 std.h

Eiti į šio failo dokumentaciją.

```
#ifndef STD_H_INCLUDED
#define STD_H_INCLUDED

using std::cout;
using std::cin;
using std::string;
using std::endl;
using std::vector;
using std::ifstream;
using std::setprecision;
using std::fixed;
using std::istringstream;
using std::terminate;
using std::sort;
using std::setw;
using std::left;
using std::stringstream;
using std::invalid_argument;
using std::exception;
using std::cerr;
using std::stoi;
using std::tolower;
using std::numeric_limits;
using std::streamsize;
using std::random_device;
using std::mt19937;
using std::uniform_int_distribution;
using std::ofstream;
using std::to_string;
using std::ios;

#endif
```

## 5.6 template.h Failo Nuoroda

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <vector>
#include <sstream>
#include <cstdlib>
#include <ctime>
#include <algorithm>
#include <random>
#include <chrono>
#include <execution>
#include "std.h"
```

Įtraukimo priklausomybių diagrama template.h: Šis grafas rodo, kuris failas tiesiogiai ar netiesiogiai įtraukia šį failą:

**Klasės**

- struct temp
- class human
- class duom

**Funkcijos**

- void menu (vector< duom > &grupe)
- void read_from_console (vector< duom > &grupe)
- void read_file (vector< duom > &grupe, string filename)
- void read_names_from_console (vector< duom > &grupe)
- void random_grades (vector< duom > &grupe, int m)
- void random_names_grades (vector< duom > &grupe, int record_amount, int mark_amount)
- void print_data_to_file (vector< duom > &grupe, int mark_amount, string filename)
- void print_answers_to_file (vector< duom > &grupe, string filename)
- void print_answers_console (vector< duom > &grupe)
- char check_menu ()
- void vid_med_calc (vector< duom > &grupe)
- bool compare (const string a, const string b, string rule)
- void sorting (vector< duom > &grupe, char rule)
- double average (duom given)
- double median (duom given)
- void method_test (vector< duom > &grupe)

**Kintamieji**

- vector< string > vardai
- const string test_file_location = TEST_FILE_LOCATION

### 5.6.1 Funkcijos Dokumentacija

#### 5.6.1.1 average()

```
double average (
            duom given)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

#### 5.6.1.2 check_menu()

```
char check_menu ()
```

Here is the caller graph for this function:

#### 5.6.1.3 compare()

```
bool compare (
            const string a,
            const string b,
            string rule)
```

Here is the caller graph for this function:

#### 5.6.1.4 median()

```
double median (
            duom given)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

#### 5.6.1.5 menu()

```
void menu (
            vector< duom > & grupe)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

#### 5.6.1.6 method_test()

```
void method_test (
            vector< duom > & grupe)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

#### 5.6.1.7 print_answers_console()

```
void print_answers_console (
            vector< duom > & grupe)
```

Here is the caller graph for this function:

#### 5.6.1.8 print_answers_to_file()

```
void print_answers_to_file (
            vector< duom > & grupe,
            string filename)
```

Here is the caller graph for this function:

#### 5.6.1.9 print_data_to_file()

```
void print_data_to_file (
            vector< duom > & grupe,
            int mark_amount,
            string filename)
```

Here is the caller graph for this function:

#### 5.6.1.10 random_grades()

```
void random_grades (
            vector< duom > & grupe,
            int m)
```

Here is the caller graph for this function:

#### 5.6.1.11 random_names_grades()

```
void random_names_grades (
            vector< duom > & grupe,
            int record_amount,
            int mark_amount)
```

Here is the caller graph for this function:

#### 5.6.1.12 read_file()

```
void read_file (
            vector< duom > & grupe,
            string filename)
```

Here is the caller graph for this function:

#### 5.6.1.13 read_from_console()

```
void read_from_console (
            vector< duom > & grupe)
```

Here is the caller graph for this function:

#### 5.6.1.14 read_names_from_console()

```
void read_names_from_console (
            vector< duom > & grupe)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

#### 5.6.1.15 sorting()

```
void sorting (
            vector< duom > & grupe,
            char rule)
```

Here is the caller graph for this function:

#### 5.6.1.16 vid_med_calc()

```
void vid_med_calc (
            vector< duom > & grupe)
```

Funkcijos kvietimo grafas: Here is the caller graph for this function:

### 5.6.2 Kintamojo Dokumentacija

#### 5.6.2.1 test_file_location

```
const string test_file_location = TEST_FILE_LOCATION
```

#### 5.6.2.2 vardai

```
vector<string> vardai
```

## 5.7 template.h

Eiti į šio failo dokumentaciją.

```
#ifndef TEMPLATE_H_INCLUDED
#define TEMPLATE_H_INCLUDED

#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <vector>
#include <sstream>
#include <cstdlib>
#include <ctime>
#include <algorithm>
#include <random>
#include <string>
#include <chrono>
#include <execution>
#include "std.h"

struct temp
{
    string var="test";
    string pav="test";
    vector<int> pazymiai;
    int exam=0;
    double vid_med=0;
    double mark;
};

class human
{
    public:
        virtual void abstractClassFunction() = 0;
        //
        string var = "";
        string pav = "";
};


class duom : private human
{
    public:
        void abstractClassFunction() override{};

    private:
        string var = "";
        string pav = "";
        vector<int> pazymiai;
        int exam = 0;
        double vid_med = 0.0;
        double mark = 0.0;
    public:
        duom(temp &a)
        {
            this->var=a.var;
            this->pav=a.pav;
            this->pazymiai=a.pazymiai;
            this->exam=a.exam;
            this->vid_med=a.vid_med;
            this->mark=a.mark;
        }
        duom(string var, string pav)
        {
            this->var=var;
            this->pav=pav;
        }
        duom(string var, string pav, vector<int> &pazymiai, int exam)
        {
            this->var=var;
            this->pav=pav;
            this->pazymiai=pazymiai;
            this->exam=exam;
        }
        duom()
        {}
        ~duom()
        {
            var.clear();
            pav.clear();
            pazymiai.clear();
            exam=0;
            vid_med=0;
            mark=0;
        }
    //
        duom(const duom &to_copy)
        {
            this->var=to_copy.var;
            this->pav=to_copy.pav;
            this->pazymiai=to_copy.pazymiai;
            this->exam=to_copy.exam;
            this->vid_med=to_copy.vid_med;
            this->mark=to_copy.mark;

        }
        duom(duom &&to_move) noexcept
        {
            this->var=std::move(to_move.var);
            this->pav=std::move(to_move.pav);
            this->pazymiai=std::move(to_move.pazymiai);
            this->exam=to_move.exam;
            to_move.exam=0;
            this->vid_med=to_move.vid_med;
            to_move.vid_med=0;
            this->mark=to_move.mark;
            to_move.mark=0;
            //move naudojam tik su elementais saugomais heap'e
        }
        //
        duom& operator=(const duom &to_copy)
        {
            if(this == &to_copy) return *this;
            //
            this->var=to_copy.var;
            this->pav=to_copy.pav;
            this->pazymiai=to_copy.pazymiai;
            this->exam=to_copy.exam;
            this->vid_med=to_copy.vid_med;
            this->mark=to_copy.mark;
            return *this;
        }
        duom& operator=(duom &&to_move) noexcept
        {
            if(this == &to_move) return *this;
            //
            this->var=std::move(to_move.var);
            this->pav=std::move(to_move.pav);
            this->pazymiai=std::move(to_move.pazymiai);
            this->exam=to_move.exam;
            to_move.exam=0;
            this->vid_med=to_move.vid_med;
            to_move.vid_med=0;
            this->mark=to_move.mark;
            to_move.mark=0;
            //move naudojam tik su elementais saugomais heap'e
            return *this;
        }
        bool operator==(duom &&to_compare) noexcept
        {
            if(this->var==to_compare.var && this->pav==to_compare.pav &&
this->pazymiai==to_compare.pazymiai && this->exam==to_compare.exam &&
this->vid_med==to_compare.vid_med
            && this->mark==to_compare.mark)
                return true;
            else
             return false;
        }
        friend bool operator==(const duom &a, const duom &b) noexcept
        {
            if(a.var==b.var && a.pav==b.pav && a.pazymiai==b.pazymiai && a.exam==b.exam &&
a.vid_med==b.vid_med
            && a.mark==b.mark)
                return true;
            return false;
        }
        //
        friend std::ifstream& operator»(std::ifstream& in, duom& student)
        {
            string eil;
            try
            {
                std::getline(in, eil);
            }
            catch(const std::exception& e)
            {
                std::cerr « e.what() « '\n';
            }

            stringstream line(eil);
            line»student.var»student.pav;

            double grade;
            student.pazymiai.clear();
            while(line»grade)
                student.pazymiai.push_back(grade);

            if(student.pazymiai.size()==0)
                throw invalid_argument("Truksta pazymiu");

            student.exam=student.pazymiai.back();
            student.pazymiai.pop_back();

            return in;
        }
        friend std::ostream& operator«(std::ostream& out, duom& student)
        {
            return out « left « fixed « setprecision(2) « setw(20) « student.var « " " « setw(20) «
student.pav « " " « setw(20) « student.mark « endl;
        }
    //
        string getVar() const
        {
            return var;
        }
        string getPav() const
        {
            return pav;
        }
        int getPazymiai_at(int i) const
        {
            return pazymiai[i];
        }
        vector<int> getPazymiai() const
        {
            return pazymiai;
        }
        int getExam() const
        {
            return exam;
        }
        double getVid_med() const
        {
            return vid_med;
        }
        double getMark() const
        {
            return mark;
        }
    //
        void setPazymiai(vector<int> &pazymiai)
        {
            this->pazymiai=pazymiai;
        }
        void addPazymiai(int grade)
        {
            if(grade>=1 && grade <=10)
                pazymiai.push_back(grade);
            else
                throw("Neteisingas pazymys");
        }
        void setExam(int exam)
        {
            if(exam>=1 && exam <=10)
                this->exam = exam;
            else
                throw("Neteisingas pazymys");
        }
        void setVid_med(double vid_med)
        {
            if(vid_med>=1 && vid_med <=10)
                this->vid_med=vid_med;
            else
                throw("Neteisingas pazymys");
        }
        void setMark(double mark)
        {
            if(mark>=1 && mark <=10)
                this->mark=mark;
            else
                throw("Neteisingas pazymys");
        }
    //
};

vector<string> vardai={
"Tomas",
"Andrius",
"Daumantas",
"Jonas",
"Petras",
"Kestas",
"Paulius",
"Juozas",
"Rokas",
"Adomas",
"Amelija",
"Motiejus",
"Jonas",
"Olivija",
"Lukas",
"Emilija",
"Jokubas",
"Adele",
"Benas",
"Ema",
"Dominykas",
"Liepa",
"Nojus",
"Ugne",
"Matas",
"Lukne",
"Markas",
"Barbora",
"Augustas"
};

const string test_file_location = TEST_FILE_LOCATION; //CMake version
//const string test_file_location = "../../test_files/"; ///Manual complilation version (debug)

void menu(vector <duom> &grupe);
void read_from_console(vector <duom> &grupe);
void read_file(vector <duom> &grupe, string filename);
void read_names_from_console(vector <duom> &grupe);
void random_grades(vector <duom> &grupe, int m);
void random_names_grades(vector <duom> &grupe, int record_amount, int mark_amount);
void print_data_to_file(vector <duom> &grupe, int mark_amount, string filename);
void print_answers_to_file(vector <duom> &grupe, string filename);
void print_answers_console(vector <duom> &grupe);
//
char check_menu();
void vid_med_calc(vector <duom> &grupe);
bool compare(const string a, const string b, string rule);
void sorting(vector <duom> &grupe, char rule);
double average(duom given);
double median(duom given);
//
void method_test(vector <duom> &grupe);

#endif


```

## 5.8 tests.cpp Failo Nuoroda

```
#include <gtest/gtest.h>
#include "template.cpp"
```

Įtraukimo priklausomybių diagrama tests.cpp:

### Funkcijos

- TEST (Compare, CompareFunctionTest)

### 5.8.1 Funkcijos Dokumentacija

#### 5.8.1.1 TEST()

```
TEST (
            Compare ,
            CompareFunctionTest )
```

Funkcijos kvietimo grafas:

# Rodyklė