

OOP projektas

Sugeneruota Doxygen 1.13.2



<b>1 Hierarchijos Indeksas</b>	<b>1</b>
1.1 Klasų hierarchija	1
<b>2 Klasės Indeksas</b>	<b>3</b>
2.1 Klasės	3
<b>3 Failo Indeksas</b>	<b>5</b>
3.1 Failai	5
<b>4 Klasės Dokumentacija</b>	<b>7</b>
4.1 duom Klasė	7
4.2 human Klasė	7
4.2.1 Metodų Dokumentacija	7
4.2.1.1 abstractClassFunction()	7
4.2.2 Atributų Dokumentacija	8
4.2.2.1 pav	8
4.2.2.2 var	8
4.3 temp Struktūra	8
4.3.1 Atributų Dokumentacija	8
4.3.1.1 exam	8
4.3.1.2 mark	8
4.3.1.3 pav	8
4.3.1.4 pazymiai	8
4.3.1.5 var	9
4.3.1.6 vid_med	9
<b>5 Failo Dokumentacija</b>	<b>11</b>
5.1 build/CMakeFiles/4.0.0-rc4/CompilerIdC/CMakeCCompilerId.c Failo Nuoroda	11
5.1.1 Apibrėžimų Dokumentacija	12
5.1.1.1 __has_include	12
5.1.1.2 ARCHITECTURE_ID	12
5.1.1.3 C_STD_11	12
5.1.1.4 C_STD_17	12
5.1.1.5 C_STD_23	12
5.1.1.6 C_STD_99	12
5.1.1.7 C_VERSION	12
5.1.1.8 COMPILER_ID	12
5.1.1.9 DEC	13
5.1.1.10 HEX	13
5.1.1.11 PLATFORM_ID	13
5.1.1.12 STRINGIFY	13
5.1.1.13 STRINGIFY_HELPER	13
5.1.2 Funkcijos Dokumentacija	13
5.1.2.1 main()	13

5.1.3 Kintamojo Dokumentacija	14
5.1.3.1 info_arch	14
5.1.3.2 info_compiler	14
5.1.3.3 info_language_extensions_default	14
5.1.3.4 info_language_standard_default	14
5.1.3.5 info_platform	14
5.2 build/CMakeFiles/4.0.0-rc4/CompilerIdCXX/CMakeCXXCompilerId.cpp Failo Nuoroda	14
5.2.1 Apibrėžimų Dokumentacija	15
5.2.1.1 __has_include	15
5.2.1.2 ARCHITECTURE_ID	15
5.2.1.3 COMPILER_ID	15
5.2.1.4 CXX_STD	15
5.2.1.5 CXX_STD_11	15
5.2.1.6 CXX_STD_14	15
5.2.1.7 CXX_STD_17	15
5.2.1.8 CXX_STD_20	16
5.2.1.9 CXX_STD_23	16
5.2.1.10 CXX_STD_98	16
5.2.1.11 DEC	16
5.2.1.12 HEX	16
5.2.1.13 PLATFORM_ID	16
5.2.1.14 STRINGIFY	16
5.2.1.15 STRINGIFY_HELPER	17
5.2.2 Funkcijos Dokumentacija	17
5.2.2.1 main()	17
5.2.3 Kintamojo Dokumentacija	17
5.2.3.1 info_arch	17
5.2.3.2 info_compiler	17
5.2.3.3 info_language_extensions_default	17
5.2.3.4 info_language_standard_default	17
5.2.3.5 info_platform	18
5.3 main.cpp Failo Nuoroda	18
5.3.1 Funkcijos Dokumentacija	18
5.3.1.1 average()	18
5.3.1.2 check_menu()	18
5.3.1.3 compare()	19
5.3.1.4 main()	19
5.3.1.5 median()	19
5.3.1.6 menu()	19
5.3.1.7 method_test()	19
5.3.1.8 print_answers_console()	19
5.3.1.9 print_answers_to_file()	19

5.3.1.10 print_data_to_file()	20
5.3.1.11 random_grades()	20
5.3.1.12 random_names_grades()	20
5.3.1.13 read_file()	20
5.3.1.14 read_from_console()	20
5.3.1.15 read_names_from_console()	20
5.3.1.16 sorting()	21
5.3.1.17 vid_med_calc()	21
5.4 std.h Failo Nuoroda	21
5.5 std.h	21
5.6 template.h Failo Nuoroda	22
5.6.1 Funkcijos Dokumentacija	22
5.6.1.1 average()	22
5.6.1.2 check_menu()	23
5.6.1.3 compare()	23
5.6.1.4 median()	23
5.6.1.5 menu()	23
5.6.1.6 method_test()	23
5.6.1.7 print_answers_console()	23
5.6.1.8 print_answers_to_file()	23
5.6.1.9 print_data_to_file()	24
5.6.1.10 random_grades()	24
5.6.1.11 random_names_grades()	24
5.6.1.12 read_file()	24
5.6.1.13 read_from_console()	24
5.6.1.14 read_names_from_console()	24
5.6.1.15 sorting()	25
5.6.1.16 vid_med_calc()	25
5.6.2 Kintamojo Dokumentacija	25
5.6.2.1 test_file_location	25
5.6.2.2 vardai	25
5.7 template.h	25
5.8 tests.cpp Failo Nuoroda	29
5.8.1 Funkcijos Dokumentacija	29
5.8.1.1 TEST()	29
<b>Rodyklė</b>	<b>31</b>



## skyrius 1

# Hierarchijos Indeksas

### 1.1 Klasių hierarchija

Šis paveldėjimo sąrašas yra beveik surikiuotas abėcėlės tvarka:

human . . . . .	7
duom . . . . .	7
temp . . . . .	8





## skyrius 2

# Klasės Indeksas

### 2.1 Klasės

Klasės, struktūros, sąjungos ir sąsajos su trumpais aprašymais:

duom	7
human	7
temp	8



## skyrius 3

# Failo Indeksas

### 3.1 Failai

Visų failų sąrašas su trumpais aprašymais:

<a href="#">main.cpp</a>	18
<a href="#">std.h</a>	21
<a href="#">template.h</a>	22
<a href="#">tests.cpp</a>	29
build/CMakeFiles/4.0.0-rc4/CompilerIdC/ <a href="#">CMakeCCompilerId.c</a>	11
build/CMakeFiles/4.0.0-rc4/CompilerIdCXX/ <a href="#">CMakeCXXCompilerId.cpp</a>	14



## 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/CMakeCXXCompilerId.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.](#)

```
00001 #ifndef STD_H_INCLUDED
00002 #define STD_H_INCLUDED
00003
00004 using std::cout;
00005 using std::cin;
00006 using std::string;
00007 using std::endl;
00008 using std::vector;
00009 using std::ifstream;
00010 using std::setprecision;
00011 using std::fixed;
00012 using std::stringstream;
00013 using std::terminate;
00014 using std::sort;
00015 using std::setw;
00016 using std::left;
00017 using std::stringstream;
00018 using std::invalid_argument;
00019 using std::exception;
00020 using std::cerr;
00021 using std::stoi;
00022 using std::tolower;
00023 using std::numeric_limits;
00024 using std::streamsize;
00025 using std::random_device;
00026 using std::mt19937;
00027 using std::uniform_int_distribution;
00028 using std::ofstream;
00029 using std::to_string;
00030 using std::ios;
00031
00032 #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.](#)

```
00001 #ifndef TEMPLATE_H_INCLUDED
00002 #define TEMPLATE_H_INCLUDED
00003
00004 #include <iostream>
00005 #include <fstream>
00006 #include <iomanip>
00007 #include <string>
00008 #include <vector>
00009 #include <sstream>
00010 #include <cstdlib>
00011 #include <ctime>
00012 #include <algorithm>
00013 #include <random>
00014 #include <string>
00015 #include <chrono>
00016 #include <execution>
00017 #include "std.h"
00018
00019 struct temp
00020 {
00021     string var="test";
00022     string pav="test";
00023     vector<int> pazymiai;
00024     int exam=0;
00025     double vid_med=0;
00026     double mark;
00027 };
00028
00029 class human
00030 {
00031     public:
```

```

00032     virtual void abstractClassFunction() = 0;
00033     //
00034     string var = "";
00035     string pav = "";
00036 };
00037
00038
00039 class duom : private human
00040 {
00041     public:
00042         void abstractClassFunction() override{};
00043
00044     private:
00045         string var = "";
00046         string pav = "";
00047         vector<int> pazymiai;
00048         int exam = 0;
00049         double vid_med = 0.0;
00050         double mark = 0.0;
00051     public:
00052         duom(temp &a)
00053         {
00054             this->var=a.var;
00055             this->pav=a.pav;
00056             this->pazymiai=a.pazymiai;
00057             this->exam=a.exam;
00058             this->vid_med=a.vid_med;
00059             this->mark=a.mark;
00060         }
00061         duom(string var, string pav)
00062         {
00063             this->var=var;
00064             this->pav=pav;
00065         }
00066         duom(string var, string pav, vector<int> &pazymiai, int exam)
00067         {
00068             this->var=var;
00069             this->pav=pav;
00070             this->pazymiai=pazymiai;
00071             this->exam=exam;
00072         }
00073         duom()
00074         {}
00075         ~duom()
00076         {
00077             var.clear();
00078             pav.clear();
00079             pazymiai.clear();
00080             exam=0;
00081             vid_med=0;
00082             mark=0;
00083         }
00084     //
00085     duom(const duom &to_copy)
00086     {
00087         this->var=to_copy.var;
00088         this->pav=to_copy.pav;
00089         this->pazymiai=to_copy.pazymiai;
00090         this->exam=to_copy.exam;
00091         this->vid_med=to_copy.vid_med;
00092         this->mark=to_copy.mark;
00093     }
00094
00095     duom(duom &&to_move) noexcept
00096     {
00097         this->var=std::move(to_move.var);
00098         this->pav=std::move(to_move.pav);
00099         this->pazymiai=std::move(to_move.pazymiai);
00100         this->exam=to_move.exam;
00101         to_move.exam=0;
00102         this->vid_med=to_move.vid_med;
00103         to_move.vid_med=0;
00104         this->mark=to_move.mark;
00105         to_move.mark=0;
00106         //move naudojam tik su elementais saugomais heap'e
00107     }
00108     //
00109     duom& operator=(const duom &to_copy)
00110     {
00111         if(this == &to_copy) return *this;
00112         //
00113         this->var=to_copy.var;
00114         this->pav=to_copy.pav;
00115         this->pazymiai=to_copy.pazymiai;
00116         this->exam=to_copy.exam;
00117         this->vid_med=to_copy.vid_med;
00118         this->mark=to_copy.mark;

```



```

00119         return *this;
00120     }
00121     duom& operator=(duom &&to_move) noexcept
00122     {
00123         if(this == &to_move) return *this;
00124         //
00125         this->var=std::move(to_move.var);
00126         this->pav=std::move(to_move.pav);
00127         this->pazymiai=std::move(to_move.pazymiai);
00128         this->exam=to_move.exam;
00129         to_move.exam=0;
00130         this->vid_med=to_move.vid_med;
00131         to_move.vid_med=0;
00132         this->mark=to_move.mark;
00133         to_move.mark=0;
00134         //move naudojam tik su elementais saugomais heap'e
00135         return *this;
00136     }
00137     bool operator==(duom &&to_compare) noexcept
00138     {
00139         if(this->var==to_compare.var && this->pav==to_compare.pav &&
00140            this->pazymiai==to_compare.pazymiai && this->exam==to_compare.exam &&
00141            this->vid_med==to_compare.vid_med
00142            && this->mark==to_compare.mark)
00143             return true;
00144         else
00145             return false;
00146     }
00147     friend bool operator==(const duom &a, const duom &b) noexcept
00148     {
00149         if(a.var==b.var && a.pav==b.pav && a.pazymiai==b.pazymiai && a.exam==b.exam &&
00150            a.vid_med==b.vid_med
00151            && a.mark==b.mark)
00152             return true;
00153         else
00154             return false;
00155     }
00156     //
00157     friend std::istream& operator>(std::istream& in, duom& student)
00158     {
00159         string eil;
00160         try
00161         {
00162             std::getline(in, eil);
00163         }
00164         catch(const std::exception& e)
00165         {
00166             std::cerr << e.what() << '\n';
00167         }
00168         stringstream line(eil);
00169         line>>student.var>>student.pav;
00170
00171         double grade;
00172         student.pazymiai.clear();
00173         while(line>>grade)
00174             student.pazymiai.push_back(grade);
00175
00176         if(student.pazymiai.size()==0)
00177             throw invalid_argument("Truksta pazymiu");
00178
00179         student.exam=student.pazymiai.back();
00180         student.pazymiai.pop_back();
00181
00182         return in;
00183     }
00184     friend std::ostream& operator<(std::ostream& out, duom& student)
00185     {
00186         return out << left << fixed << setprecision(2) << setw(20) << student.var << " " << setw(20) <<
00187            student.pav << " " << setw(20) << student.mark << endl;
00188     }
00189     //
00190     string getVar() const
00191     {
00192         return var;
00193     }
00194     string getPav() const
00195     {
00196         return pav;
00197     }
00198     int getPazymiai_at(int i) const
00199     {
00200         return pazymiai[i];
00201     }
00202     vector<int> getPazymiai() const
00203     {
00204         return pazymiai;
00205     }

```

```

00202     int getExam() const
00203     {
00204         return exam;
00205     }
00206     double getVid_med() const
00207     {
00208         return vid_med;
00209     }
00210     double getMark() const
00211     {
00212         return mark;
00213     }
00214     //
00215     void setPazymiai(vector<int> &pazymiai)
00216     {
00217         this->pazymiai=pazymiai;
00218     }
00219     void addPazymiai(int grade)
00220     {
00221         if(grade>=1 && grade <=10)
00222             pazymiai.push_back(grade);
00223         else
00224             throw("Neteisingas pazymys");
00225     }
00226     void setExam(int exam)
00227     {
00228         if(exam>=1 && exam <=10)
00229             this->exam = exam;
00230         else
00231             throw("Neteisingas pazymys");
00232     }
00233     void setVid_med(double vid_med)
00234     {
00235         if(vid_med>=1 && vid_med <=10)
00236             this->vid_med=vid_med;
00237         else
00238             throw("Neteisingas pazymys");
00239     }
00240     void setMark(double mark)
00241     {
00242         if(mark>=1 && mark <=10)
00243             this->mark=mark;
00244         else
00245             throw("Neteisingas pazymys");
00246     }
00247     //
00248 };
00249
00250 vector<string> vardai={
00251     "Tomas",
00252     "Andrius",
00253     "Daumantas",
00254     "Jonas",
00255     "Petras",
00256     "Kestas",
00257     "Paulius",
00258     "Juozas",
00259     "Rokas",
00260     "Adomas",
00261     "Amelija",
00262     "Motiejus",
00263     "Jonas",
00264     "Olivija",
00265     "Lukas",
00266     "Emilija",
00267     "Jokubas",
00268     "Adele",
00269     "Benas",
00270     "Ema",
00271     "Dominykas",
00272     "Liepa",
00273     "Nojus",
00274     "Ugne",
00275     "Matas",
00276     "Lukne",
00277     "Markas",
00278     "Barbora",
00279     "Augustas"
00280 };
00281
00282 const string test_file_location = TEST_FILE_LOCATION; //CMake version
00283 //const string test_file_location = "../test_files/"; ///Manual compilation version (debug)
00284
00285 void menu(vector <duom> &grupe);
00286 void read_from_console(vector <duom> &grupe);
00287 void read_file(vector <duom> &grupe, string filename);
00288 void read_names_from_console(vector <duom> &grupe);

```

```
00289 void random_grades(vector <duom> &grupe, int m);
00290 void random_names_grades(vector <duom> &grupe, int record_amount, int mark_amount);
00291 void print_data_to_file(vector <duom> &grupe, int mark_amount, string filename);
00292 void print_answers_to_file(vector <duom> &grupe, string filename);
00293 void print_answers_console(vector <duom> &grupe);
00294 //
00295 char check_menu();
00296 void vid_med_calc(vector <duom> &grupe);
00297 bool compare(const string a, const string b, string rule);
00298 void sorting(vector <duom> &grupe, char rule);
00299 double average(duom given);
00300 double median(duom given);
00301 //
00302 void method_test(vector <duom> &grupe);
00303
00304 #endif
00305
00306
```

## 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:



# Rodyklè

- [\\_\\_has\\_include](#)
    - [CMakeCCompilerId.c, 12](#)
    - [CMakeCXXCompilerId.cpp, 15](#)
- [abstractClassFunction](#)
  - [human, 7](#)
- [ARCHITECTURE\\_ID](#)
  - [CMakeCCompilerId.c, 12](#)
  - [CMakeCXXCompilerId.cpp, 15](#)
- [average](#)
  - [main.cpp, 18](#)
  - [template.h, 22](#)
- [build/CMakeFiles/4.0.0-rc4/CompilerIdC/CMakeCCompilerId.c, 11](#)
- [build/CMakeFiles/4.0.0-rc4/CompilerIdCXX/CMakeCXXCompilerId.cpp, 14](#)
- [C\\_STD\\_11](#)
  - [CMakeCCompilerId.c, 12](#)
- [C\\_STD\\_17](#)
  - [CMakeCCompilerId.c, 12](#)
- [C\\_STD\\_23](#)
  - [CMakeCCompilerId.c, 12](#)
- [C\\_STD\\_99](#)
  - [CMakeCCompilerId.c, 12](#)
- [C\\_VERSION](#)
  - [CMakeCCompilerId.c, 12](#)
- [check\\_menu](#)
  - [main.cpp, 18](#)
  - [template.h, 22](#)
- [CMakeCCompilerId.c](#)
  - [\\_\\_has\\_include, 12](#)
  - [ARCHITECTURE\\_ID, 12](#)
  - [C\\_STD\\_11, 12](#)
  - [C\\_STD\\_17, 12](#)
  - [C\\_STD\\_23, 12](#)
  - [C\\_STD\\_99, 12](#)
  - [C\\_VERSION, 12](#)
  - [COMPILER\\_ID, 12](#)
  - [DEC, 12](#)
  - [HEX, 13](#)
  - [info\\_arch, 14](#)
  - [info\\_compiler, 14](#)
  - [info\\_language\\_extensions\\_default, 14](#)
  - [info\\_language\\_standard\\_default, 14](#)
  - [info\\_platform, 14](#)
  - [main, 13](#)
  - [PLATFORM\\_ID, 13](#)
  - [STRINGIFY, 13](#)
- [STRINGIFY\\_HELPER, 13](#)
- [CMakeCXXCompilerId.cpp](#)
  - [\\_\\_has\\_include, 15](#)
  - [ARCHITECTURE\\_ID, 15](#)
  - [COMPILER\\_ID, 15](#)
  - [CXX\\_STD, 15](#)
  - [CXX\\_STD\\_11, 15](#)
  - [CXX\\_STD\\_14, 15](#)
  - [CXX\\_STD\\_17, 15](#)
  - [CXX\\_STD\\_20, 15](#)
  - [CXX\\_STD\\_23, 16](#)
  - [CXX\\_STD\\_98, 16](#)
  - [DEC, 16](#)
  - [HEX, 16](#)
  - [info\\_arch, 17](#)
  - [info\\_compiler, 17](#)
  - [info\\_language\\_extensions\\_default, 17](#)
  - [info\\_language\\_standard\\_default, 17](#)
  - [info\\_platform, 17](#)
  - [main, 17](#)
  - [PLATFORM\\_ID, 16](#)
  - [STRINGIFY, 16](#)
  - [STRINGIFY\\_HELPER, 16](#)
- [compare](#)
  - [main.cpp, 18](#)
  - [template.h, 23](#)
- [COMPILER\\_ID](#)
  - [CMakeCCompilerId.c, 12](#)
  - [CMakeCXXCompilerId.cpp, 15](#)
- [CXX\\_STD](#)
  - [CMakeCXXCompilerId.cpp, 15](#)
- [CXX\\_STD\\_11](#)
  - [CMakeCXXCompilerId.cpp, 15](#)
- [CXX\\_STD\\_14](#)
  - [CMakeCXXCompilerId.cpp, 15](#)
- [CXX\\_STD\\_17](#)
  - [CMakeCXXCompilerId.cpp, 15](#)
- [CXX\\_STD\\_20](#)
  - [CMakeCXXCompilerId.cpp, 15](#)
- [CXX\\_STD\\_23](#)
  - [CMakeCXXCompilerId.cpp, 16](#)
- [CXX\\_STD\\_98](#)
  - [CMakeCXXCompilerId.cpp, 16](#)
- [DEC](#)
  - [CMakeCCompilerId.c, 12](#)
  - [CMakeCXXCompilerId.cpp, 16](#)
- [duom, 7](#)
- [exam](#)

- temp, 8
- HEX
  - CMakeCCompilerId.c, 13
  - CMakeCXXCompilerId.cpp, 16
- human, 7
  - abstractClassFunction, 7
  - pav, 8
  - var, 8
- info\_arch
  - CMakeCCompilerId.c, 14
  - CMakeCXXCompilerId.cpp, 17
- info\_compiler
  - CMakeCCompilerId.c, 14
  - CMakeCXXCompilerId.cpp, 17
- info\_language\_extensions\_default
  - CMakeCCompilerId.c, 14
  - CMakeCXXCompilerId.cpp, 17
- info\_language\_standard\_default
  - CMakeCCompilerId.c, 14
  - CMakeCXXCompilerId.cpp, 17
- info\_platform
  - CMakeCCompilerId.c, 14
  - CMakeCXXCompilerId.cpp, 17
- main
  - CMakeCCompilerId.c, 13
  - CMakeCXXCompilerId.cpp, 17
  - main.cpp, 19
- main.cpp, 18
  - average, 18
  - check\_menu, 18
  - compare, 18
  - main, 19
  - median, 19
  - menu, 19
  - method\_test, 19
  - print\_answers\_console, 19
  - print\_answers\_to\_file, 19
  - print\_data\_to\_file, 19
  - random\_grades, 20
  - random\_names\_grades, 20
  - read\_file, 20
  - read\_from\_console, 20
  - read\_names\_from\_console, 20
  - sorting, 20
  - vid\_med\_calc, 21
- mark
  - temp, 8
- median
  - main.cpp, 19
  - template.h, 23
- menu
  - main.cpp, 19
  - template.h, 23
- method\_test
  - main.cpp, 19
  - template.h, 23
- pav
  - human, 8
  - temp, 8
- pazymiai
  - temp, 8
- PLATFORM\_ID
  - CMakeCCompilerId.c, 13
  - CMakeCXXCompilerId.cpp, 16
- print\_answers\_console
  - main.cpp, 19
  - template.h, 23
- print\_answers\_to\_file
  - main.cpp, 19
  - template.h, 23
- print\_data\_to\_file
  - main.cpp, 19
  - template.h, 23
- random\_grades
  - main.cpp, 20
  - template.h, 24
- random\_names\_grades
  - main.cpp, 20
  - template.h, 24
- read\_file
  - main.cpp, 20
  - template.h, 24
- read\_from\_console
  - main.cpp, 20
  - template.h, 24
- read\_names\_from\_console
  - main.cpp, 20
  - template.h, 24
- sorting
  - main.cpp, 20
  - template.h, 24
- std.h, 21
- STRINGIFY
  - CMakeCCompilerId.c, 13
  - CMakeCXXCompilerId.cpp, 16
- STRINGIFY\_HELPER
  - CMakeCCompilerId.c, 13
  - CMakeCXXCompilerId.cpp, 16
- temp, 8
  - exam, 8
  - mark, 8
  - pav, 8
  - pazymiai, 8
  - var, 8
  - vid\_med, 9
- template.h, 22
  - average, 22
  - check\_menu, 22
  - compare, 23
  - median, 23
  - menu, 23
  - method\_test, 23

- print\_answers\_console, [23](#)
- print\_answers\_to\_file, [23](#)
- print\_data\_to\_file, [23](#)
- random\_grades, [24](#)
- random\_names\_grades, [24](#)
- read\_file, [24](#)
- read\_from\_console, [24](#)
- read\_names\_from\_console, [24](#)
- sorting, [24](#)
- test\_file\_location, [25](#)
- vardai, [25](#)
- vid\_med\_calc, [25](#)
- TEST
  - tests.cpp, [29](#)
- test\_file\_location
  - template.h, [25](#)
- tests.cpp, [29](#)
  - TEST, [29](#)
- var
  - human, [8](#)
  - temp, [8](#)
- vardai
  - template.h, [25](#)
- vid\_med
  - temp, [9](#)
- vid\_med\_calc
  - main.cpp, [21](#)
  - template.h, [25](#)