

Курсовая работа

Создано системой Doxygen 1.9.4



---

1 Иерархический список классов	1
1.1 Иерархия классов	1
2 Алфавитный указатель классов	3
2.1 Классы	3
3 Список файлов	5
3.1 Файлы	5
4 Классы	7
4.1 Класс Base	7
4.1.1 Подробное описание	7
4.1.2 Методы	7
4.1.2.1 connect()	7
4.1.2.2 getData()	8
4.2 Класс Calc	8
4.2.1 Подробное описание	8
4.2.2 Конструктор(ы)	8
4.2.2.1 Calc()	8
4.2.3 Методы	9
4.2.3.1 getResult()	9
4.3 Класс Communicator	9
4.3.1 Подробное описание	9
4.3.2 Методы	10
4.3.2.1 connection()	10
4.3.2.2 generateSalt()	11
4.3.2.3 MD5()	11
4.4 Класс CritError	12
4.4.1 Подробное описание	12
4.4.2 Конструктор(ы)	12
4.4.2.1 CritError()	13
4.5 Класс Interface	13
4.5.1 Подробное описание	13
4.5.2 Конструктор(ы)	13
4.5.2.1 Interface()	14
4.5.3 Методы	14
4.5.3.1 displayHelp()	14
4.5.3.2 getDatabaseFile()	14
4.5.3.3 getLogFile()	14
4.5.3.4 getServerPort()	15
4.5.3.5 parseArguments()	15
4.5.3.6 setupConnection()	15
4.6 Класс Logger	16
4.6.1 Подробное описание	16

4.6.2 Конструктор(ы)	16
4.6.2.1 Logger() [1/2]	16
4.6.2.2 Logger() [2/2]	16
4.6.3 Методы	17
4.6.3.1 getPath()	17
4.6.3.2 setPath()	17
4.6.3.3 writeLog()	17
4.7 Класс NoCritError	19
4.7.1 Подробное описание	20
4.7.2 Конструктор(ы)	20
4.7.2.1 NoCritError()	20
5 Файлы	21
5.1 Файл base.h	21
5.1.1 Подробное описание	22
5.2 base.h	22
5.3 Файл calc.h	23
5.3.1 Подробное описание	23
5.4 calc.h	24
5.5 Файл communicator.h	24
5.5.1 Подробное описание	25
5.6 communicator.h	25
5.7 Файл error.h	26
5.7.1 Подробное описание	27
5.8 error.h	27
5.9 Файл interface.h	27
5.9.1 Подробное описание	28
5.10 interface.h	28
5.11 Файл logger.h	29
5.11.1 Подробное описание	29
5.12 logger.h	30
Предметный указатель	31

# Глава 1

## Иерархический список классов

### 1.1 Иерархия классов

Иерархия классов.

Base . . . . .	7
Calc . . . . .	8
Communicator . . . . .	9
Interface . . . . .	13
Logger . . . . .	16
std::runtime_error	
CritError . . . . .	12
NoCritError . . . . .	19



## Глава 2

# Алфавитный указатель классов

### 2.1 Классы

Классы с их кратким описанием.

<a href="#">Base</a>	Класс для работы с базой данных . . . . .	7
<a href="#">Calc</a>	Класс для выполнения вычислений . . . . .	8
<a href="#">Communicator</a>	Класс коммуникатора . . . . .	9
<a href="#">CritError</a>	Класс для возбуждения критических ошибок Возбуждает критические ошибки .	12
<a href="#">Interface</a>	Класс интерфейса . . . . .	13
<a href="#">Logger</a>	Класс для работы с журналом логов . . . . .	16
<a href="#">NoCritError</a>	Класс для возбуждения некритических ошибок Возбуждает некритические ошибки . . . . .	19





## Глава 3

# Список файлов

### 3.1 Файлы

Полный список документированных файлов.

<a href="#">base.h</a>	Заголовочный файл для модуля базы данных . . . . .	<a href="#">21</a>
<a href="#">calc.h</a>	Заголовочный файл для модуля вычислений . . . . .	<a href="#">23</a>
<a href="#">communicator.h</a>	Заголовочный файл для коммуникатора сервера . . . . .	<a href="#">24</a>
<a href="#">error.h</a>	Заголовочный файл для обработки ошибок . . . . .	<a href="#">26</a>
<a href="#">interface.h</a>	Заголовочный файл для интерфейса . . . . .	<a href="#">27</a>
<a href="#">logger.h</a>	Заголовочный файл для модуля логирования . . . . .	<a href="#">29</a>



## Глава 4

# Классы

### 4.1 Класс Base

Класс для работы с базой данных

```
#include <base.h>
```

Открытые члены

- void [connect](#) (std::string filePath)  
Установка соединения с базой данных
- std::map< std::string, std::string > [getData](#) ()  
Получить базу данных

#### 4.1.1 Подробное описание

Класс для работы с базой данных

Контейнер `dataBase` хранит в себе логины и пароли пользователей. Для получения данных используется метод [getData\(\)](#).

#### 4.1.2 Методы

##### 4.1.2.1 connect()

```
void Base::connect (  
    std::string filePath )
```

Установка соединения с базой данных

Читает данные из указанного файла и заполняет контейнер `dataBase`.

## Аргументы

in	filePath	Путь к файлу базы данных.
----	----------	---------------------------

## Исключения

<a href="#">CritError</a>	Если файл не открывается или имеет неверный формат.
---------------------------	---

## 4.1.2.2 getData()

```
std::map< std::string, std::string > Base::getData ( ) [inline]
```

Получить базу данных

Возвращает

`std::map<std::string, std::string>` Контейнер с логинами и паролями пользователей.

Объявления и описания членов классов находятся в файлах:

- [base.h](#)
- [base.cpp](#)

## 4.2 Класс Calc

Класс для выполнения вычислений

```
#include <calc.h>
```

## Открытые члены

- [Calc](#) (`std::vector< double > numbers`)  
Конструктор класса [Calc](#).
- `double getResult ( )`  
Получить результат вычислений

## 4.2.1 Подробное описание

Класс для выполнения вычислений

Класс принимает вектор чисел и производит вычисления.

## 4.2.2 Конструктор(ы)

## 4.2.2.1 Calc()

```
Calc::Calc (
    std::vector< double > numbers )
```

Конструктор класса [Calc](#).

Инициализирует объект и выполняет вычисления на основе переданного вектора.

Аргументы

in	numbers	Вектор чисел для вычислений.
----	---------	------------------------------

### 4.2.3 Методы

#### 4.2.3.1 getResult()

```
double Calc::getResult ( ) [inline]
```

Получить результат вычислений

Возвращает

double Результат вычислений.

Объявления и описания членов классов находятся в файлах:

- [calc.h](#)
- [calc.cpp](#)

## 4.3 Класс Communicator

Класс коммуникатора

```
#include <communicator.h>
```

Открытые члены

- int [connection](#) (int port, const std::map< std::string, std::string > &database, [Logger](#) \*logger)  
Установка соединения с клиентами
- std::string [MD5](#) (const std::string &inputString)  
Вычисление MD5 хеша
- std::string [generateSalt](#) ()  
Генерация соли

#### 4.3.1 Подробное описание

Класс коммуникатора

Устанавливает соединение с сервером, производит авторизацию клиента.

### 4.3.2 Методы

#### 4.3.2.1 connection()

```
int Communicator::connection (
    int port,
    const std::map< std::string, std::string > & database,
    Logger * logger )
```

Установка соединения с клиентами

Производит соединение с сервером и авторизует пользователя.

## Аргументы

in	port	Номер порта для соединения.
in	database	Контейнер с базой данных пользователей.
in	logger	Указатель на объект <a href="#">Logger</a> для записи событий.

## Возвращает

int Код результата операции.

## 4.3.2.2 generateSalt()

```
std::string Communicator::generateSalt ( )
```

## Генерация соли

Генерирует соль для безопасного хранения паролей.

## Возвращает

std::string Сгенерированная соль.

## 4.3.2.3 MD5()

```
std::string Communicator::MD5 (
    const std::string & inputString )
```

## Вычисление MD5 хеша

Производит вычисление MD5 хеша для заданной строки.

## Аргументы

in	inputString	Входная строка для хэширования.
----	-------------	---------------------------------

## Возвращает

std::string Результат хэширования.

Объявления и описания членов классов находятся в файлах:

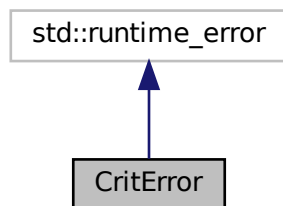
- [communicator.h](#)
- [communicator.cpp](#)

## 4.4 Класс CritError

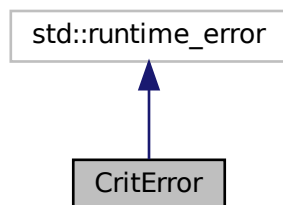
Класс для возбуждения критических ошибок Возбуждает критические ошибки.

```
#include <error.h>
```

Граф наследования: CritError:



Граф связей класса CritError:



Открытые члены

- [CritError](#) (const std::string &message)  
Конструктор класса [CritError](#).

### 4.4.1 Подробное описание

Класс для возбуждения критических ошибок Возбуждает критические ошибки.

### 4.4.2 Конструктор(ы)



## 4.4.2.1 CritError()

```
CritError::CritError (
    const std::string & message ) [inline]
```

Конструктор класса [CritError](#).

Аргументы

in	message	Сообщение об ошибке.
----	---------	----------------------

Объявления и описания членов класса находятся в файле:

- [error.h](#)

## 4.5 Класс Interface

Класс интерфейса

```
#include <interface.h>
```

Открытые члены

- [Interface](#) ()  
Конструктор по умолчанию
- bool [parseArguments](#) (int argc, const char \*\*argv)  
Парсер аргументов командной строки
- void [setupConnection](#) (const std::string &databaseFile, const std::string &logFile)  
Настройка соединения
- void [displayHelp](#) (const boost::program\_options::options\_description &options)  
Отображение справки
- int [getServerPort](#) () const  
Получение порта сервера
- std::string [getDatabaseFile](#) ()  
Получение пути к файлу базы данных
- std::string [getLogFile](#) ()  
Получение пути к файлу журнала

## 4.5.1 Подробное описание

Класс интерфейса

Управляет настройками сервера и взаимодействием с пользователем.

## 4.5.2 Конструктор(ы)

#### 4.5.2.1 Interface()

```
Interface::Interface ( ) [inline]
```

Конструктор по умолчанию

Инициализирует параметры по умолчанию.

#### 4.5.3 Методы

##### 4.5.3.1 displayHelp()

```
void Interface::displayHelp (
    const boost::program_options::options_description & options )
```

Отображение справки

Аргументы

in	options	Описание доступных опций.
----	---------	---------------------------

##### 4.5.3.2 getDatabaseFile()

```
std::string Interface::getDatabaseFile ( ) [inline]
```

Получение пути к файлу базы данных

Возвращает

std::string Путь к файлу базы данных.

##### 4.5.3.3 getLogFile()

```
std::string Interface::getLogFile ( ) [inline]
```

Получение пути к файлу журнала

Возвращает

std::string Путь к файлу журнала.

## 4.5.3.4 getServerPort()

```
int Interface::getServerPort ( ) const [inline]
```

Получение порта сервера

Возвращает

int Номер порта сервера.

## 4.5.3.5 parseArguments()

```
bool Interface::parseArguments (
    int argc,
    const char ** argv )
```

Парсер аргументов командной строки

Читает операнды командной строки. В случае передачи операнда -h производится вызов справки.

Аргументы

in	argc	Количество аргументов.
in	argv	Массив аргументов.

Возвращает

bool true, если аргументы обработаны корректно; false в противном случае.

Исключения

<b>CritError</b>	В случае некорректного значения порта.
------------------	--

## 4.5.3.6 setupConnection()

```
void Interface::setupConnection (
    const std::string & databaseFile,
    const std::string & logFile )
```

Настройка соединения

Устанавливает соединение с базой данных и журналом лога.

## Аргументы

in	databaseFile	Путь к файлу базы данных.
in	logFile	Путь к файлу журнала.

Объявления и описания членов классов находятся в файлах:

- [interface.h](#)
- [interface.cpp](#)

## 4.6 Класс Logger

Класс для работы с журналом логов

```
#include <logger.h>
```

### Открытые члены

- [Logger](#) ()  
Конструктор по умолчанию
- [Logger](#) (const std::string &path)  
Конструктор с указанием пути
- int [setPath](#) (const std::string &pathFile)  
Установка пути к файлу лога
- int [writeLog](#) (const std::string &message)  
Запись события в журнал
- std::string [getPath](#) () const  
Получение пути к файлу лога

### 4.6.1 Подробное описание

Класс для работы с журналом логов

Позволяет записывать события в лог-файл.

### 4.6.2 Конструктор(ы)

#### 4.6.2.1 [Logger](#)() [1/2]

```
Logger::Logger ( )
```

Конструктор по умолчанию

Инициализирует объект [Logger](#) с умолчательным путем.

#### 4.6.2.2 [Logger](#)() [2/2]

```
Logger::Logger (
    const std::string & path )
```

Конструктор с указанием пути

## Аргументы

in	path	Путь к файлу лога.
----	------	--------------------

## 4.6.3 Методы

## 4.6.3.1 getPath()

```
std::string Logger::getPath ( ) const [inline]
```

Получение пути к файлу лога

Возвращает

std::string Путь к файлу лога.

## 4.6.3.2 setPath()

```
int Logger::setPath (
    const std::string & pathFile )
```

Установка пути к файлу лога

## Аргументы

in	pathFile	Путь к файлу лога.
----	----------	--------------------

Возвращает

int Код результата операции.

## Исключения

<a href="#">CriticalSection</a>	Если файл не открывается.
---------------------------------	---------------------------

## 4.6.3.3 writeLog()

```
int Logger::writeLog (
    const std::string & message )
```

Запись события в журнал

## Аргументы

in	message	Сообщение для записи в лог.
----	---------	-----------------------------

## Возвращает

int Код результата операции.

## Исключения

<a href="#">CritError</a>	Если файл не открывается на запись.
---------------------------	-------------------------------------

Объявления и описания членов классов находятся в файлах:

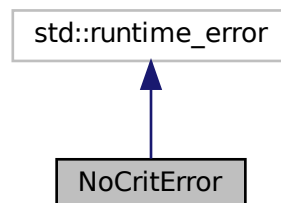
- [logger.h](#)
- [logger.cpp](#)

## 4.7 Класс NoCritError

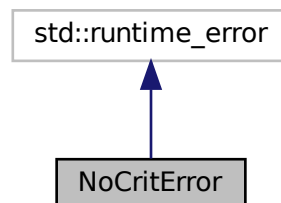
Класс для возбуждения не критических ошибок Возбуждает не критические ошибки.

```
#include <error.h>
```

Граф наследования:NoCritError:



Граф связей класса NoCritError:



## Открытые члены

- [NoCritError](#) (const std::string message)  
Конструктор класса [NoCritError](#).

### 4.7.1 Подробное описание

Класс для возбуждения некритических ошибок Возбуждает некритические ошибки.

### 4.7.2 Конструктор(ы)

#### 4.7.2.1 NoCritError()

```
NoCritError::NoCritError (  
    const std::string message )    [inline]
```

Конструктор класса [NoCritError](#).

Аргументы

in	message	Сообщение об ошибке.
----	---------	----------------------

Объявления и описания членов класса находятся в файле:

- [error.h](#)



## Глава 5

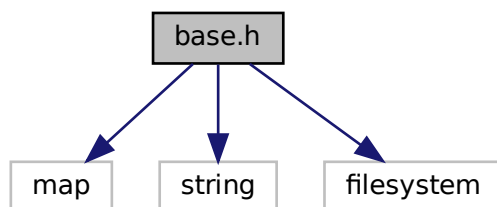
# Файлы

### 5.1 Файл base.h

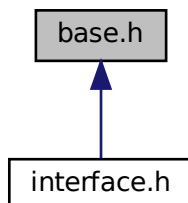
Заголовочный файл для модуля базы данных

```
#include <map>
#include <string>
#include <filesystem>
```

Граф включаемых заголовочных файлов для base.h:



Граф файлов, в которые включается этот файл:



## Классы

- class [Base](#)

Класс для работы с базой данных

### 5.1.1 Подробное описание

Заголовочный файл для модуля базы данных

Автор

Чувашов М.С.

Версия

1.0

Дата

19.12.2024

Авторство

ИБСТ ПГУ

## 5.2 base.h

[См. документацию.](#)

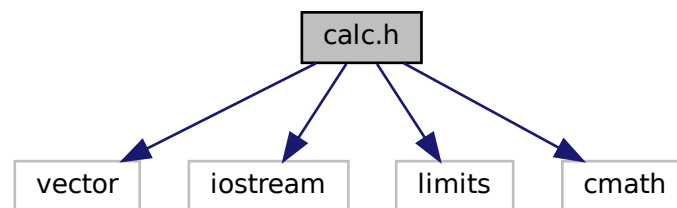
```
1 #pragma once
2 #include <map>
3 #include <string>
4 #include <filesystem>
5
18 class Base
19 {
20 private:
21     std::map<std::string, std::string> dataBase;
22 public:
23     void connect(std::string filePath);
24
27     std::map<std::string, std::string> getData() { return dataBase; }
28 };
```

## 5.3 Файл calc.h

Заголовочный файл для модуля вычислений

```
#include <vector>
#include <iostream>
#include <limits>
#include <cmath>
```

Граф включаемых заголовочных файлов для calc.h:



### Классы

- class [Calc](#)

Класс для выполнения вычислений

### 5.3.1 Подробное описание

Заголовочный файл для модуля вычислений

Автор

Чувашов М.С.

Версия

1.0

Дата

19.12.2024

Авторство

ИБСТ ПГУ

## 5.4 calc.h

См. документацию.

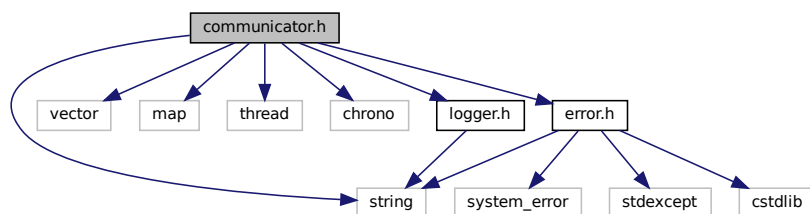
```
1 #pragma once
2 #include <vector>
3 #include <iostream>
4 #include <limits>
5 #include <cmath>
6
18 class Calc
19 {
20 private:
21     double result;
22 public:
29     Calc(std::vector<double> numbers);
30
36     double getResult() { return result; }
37 };
```

## 5.5 Файл communicator.h

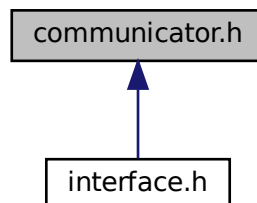
Заголовочный файл для коммуникатора сервера

```
#include <string>
#include <vector>
#include <map>
#include <thread>
#include <chrono>
#include "logger.h"
#include "error.h"
```

Граф включаемых заголовочных файлов для communicator.h:



Граф файлов, в которые включается этот файл:



## Классы

- class [Communicator](#)  
Класс коммуникатора

## Макросы

- `#define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1`

### 5.5.1 Подробное описание

Заголовочный файл для коммуникатора сервера

Автор

Чувашов М.С.

Версия

1.0

Дата

19.12.2024

Авторство

ИБСТ ПГУ

## 5.6 communicator.h

[См. документацию.](#)

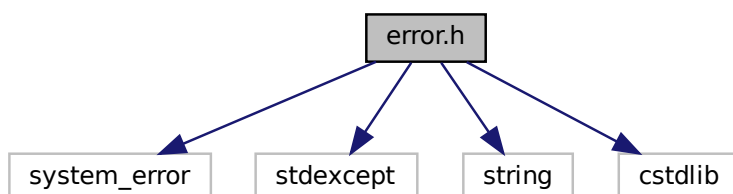
```
1 #pragma once
2 #include <string>
3 #include <vector>
4 #include <map>
5 #include <thread>
6 #include <chrono>
7 #include "logger.h"
8 #include "error.h"
9 #define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1
10
22 class Communicator
23 {
24 public:
34     int connection(int port, const std::map<std::string, std::string>& database, Logger* logger);
35
43     std::string MD5(const std::string& inputString);
44
51     std::string generateSalt();
52 };
```

## 5.7 Файл error.h

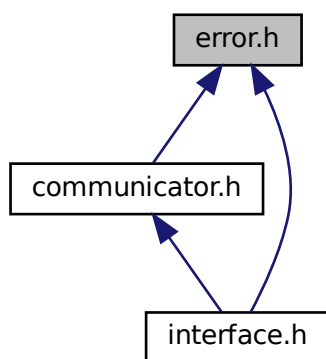
Заголовочный файл для обработки ошибок

```
#include <system_error>
#include <stdexcept>
#include <string>
#include <cstdlib>
```

Граф включаемых заголовочных файлов для error.h:



Граф файлов, в которые включается этот файл:



## Классы

- class [CritError](#)  
Класс для возбуждения критических ошибок Возбуждает критические ошибки.
- class [NoCritError](#)  
Класс для возбуждения некритических ошибок Возбуждает некритические ошибки.

### 5.7.1 Подробное описание

Заголовочный файл для обработки ошибок

Автор

Чувашов М.С.

Версия

1.0

Дата

19.12.2024

Авторство

ИБСТ ПГУ

## 5.8 error.h

[См. документацию.](#)

```

1 #pragma once
2 #include <system_error>
3 #include <stdexcept>
4 #include <string>
5 #include <cstdlib>
6
18 class CritError : public std::runtime_error
19 {
20 public:
26     CritError(const std::string& message) : std::runtime_error(message) {}
27 };
28
32 class NoCritError : public std::runtime_error
33 {
34 public:
40     NoCritError(const std::string message) : std::runtime_error(message) {}
41 };

```

## 5.9 Файл interface.h

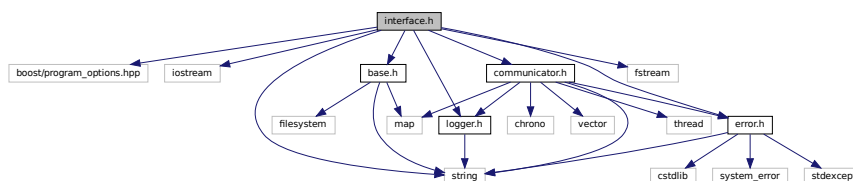
Заголовочный файл для интерфейса

```

#include <boost/program_options.hpp>
#include <iostream>
#include <string>
#include <fstream>
#include "logger.h"
#include "base.h"
#include "communicator.h"
#include "error.h"

```

Граф включаемых заголовочных файлов для interface.h:



## Классы

- class [Interface](#)  
Класс интерфейса

### 5.9.1 Подробное описание

Заголовочный файл для интерфейса

Автор

Чувашов М.С.

Версия

1.0

Дата

19.12.2024

Авторство

ИБСТ ПГУ

### 5.10 interface.h

[См. документацию.](#)

```
1 #pragma once
2 #include <boost/program_options.hpp>
3 #include <iostream>
4 #include <string>
5 #include <fstream>
6 #include "logger.h"
7 #include "base.h"
8 #include "communicator.h"
9 #include "error.h"
10 namespace po = boost::program_options;
11
12
13 class Interface {
14 private:
15     int serverPort;
16     std::string databaseFile;
17     std::string logFile;
18
19 public:
20     Interface() : serverPort(33333), databaseFile("base.txt"), logFile("log.txt") {}
21
22     bool parseArguments(int argc, const char** argv);
23
24     void setupConnection(const std::string& databaseFile, const std::string& logFile);
25
26     void displayHelp(const boost::program_options::options_description& options);
27
28     int getServerPort()const { return serverPort; }
29
30     std::string getDatabaseFile() { return databaseFile; }
31
32     std::string getLogFile() { return logFile; }
33 };
```

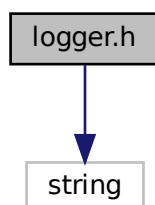


## 5.11 Файл logger.h

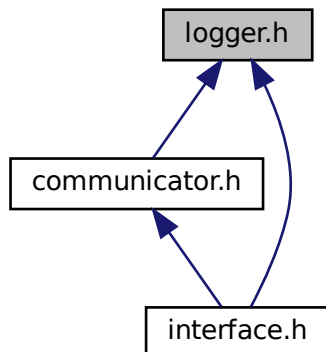
Заголовочный файл для модуля логирования

```
#include <string>
```

Граф включаемых заголовочных файлов для logger.h:



Граф файлов, в которые включается этот файл:



Классы

- class [Logger](#)

Класс для работы с журналом логов

### 5.11.1 Подробное описание

Заголовочный файл для модуля логирования

Автор

Чувашов М.С.

Версия

1.0

Дата

19.12.2024

Авторство

ИБСТ ПГУ

## 5.12 logger.h

[См. документацию.](#)

```
1 #pragma once
2 #include <string>
3
15 class Logger
16 {
17 private:
18     std::string pathToLogFile;
19
25     std::string getTime();
26
27 public:
33     Logger();
34
40     Logger(const std::string& path);
41
49     int setPath(const std::string& pathFile);
50
58     int writeLog(const std::string& message);
59
65     std::string getPath()const { return pathToLogFile; }
66 };
```

# Предметный указатель

- Base, [7](#)
  - connect, [7](#)
  - getData, [8](#)
- base.h, [21](#)
- Calc, [8](#)
  - Calc, [8](#)
  - getResult, [9](#)
- calc.h, [23](#)
- Communicator, [9](#)
  - connection, [10](#)
  - generateSalt, [11](#)
  - MD5, [11](#)
- communicator.h, [24](#)
- connect
  - Base, [7](#)
- connection
  - Communicator, [10](#)
- CritError, [12](#)
  - CritError, [12](#)
- displayHelp
  - Interface, [14](#)
- error.h, [26](#)
- generateSalt
  - Communicator, [11](#)
- getData
  - Base, [8](#)
- getDatabaseFile
  - Interface, [14](#)
- getLogFile
  - Interface, [14](#)
- getPath
  - Logger, [17](#)
- getResult
  - Calc, [9](#)
- getServerPort
  - Interface, [14](#)
- Interface, [13](#)
  - displayHelp, [14](#)
  - getDatabaseFile, [14](#)
  - getLogFile, [14](#)
  - getServerPort, [14](#)
  - Interface, [13](#)
  - parseArguments, [15](#)
  - setupConnection, [15](#)
- interface.h, [27](#)
- Logger, [16](#)
  - getPath, [17](#)
  - Logger, [16](#)
  - setPath, [17](#)
  - writeLog, [17](#)
- logger.h, [29](#)
- MD5
  - Communicator, [11](#)
- NoCritError, [19](#)
  - NoCritError, [20](#)
- parseArguments
  - Interface, [15](#)
- setPath
  - Logger, [17](#)
- setupConnection
  - Interface, [15](#)
- writeLog
  - Logger, [17](#)