

Server-MD5-int32\_t-saltOnClient

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 CLI	7
4.1.1 Подробное описание	7
4.1.2 Конструктор(ы)	7
4.1.2.1 CLI()	8
4.1.3 Методы	8
4.1.3.1 getAddr()	8
4.1.3.2 getDataFile()	8
4.1.3.3 getLogFile()	8
4.1.3.4 getPort()	9
4.1.3.5 parseArgs()	9
4.2 Класс DataWorker	9
4.2.1 Подробное описание	10
4.2.2 Конструктор(ы)	10
4.2.2.1 DataWorker()	10
4.2.3 Методы	10
4.2.3.1 getFilePath()	10
4.2.3.2 readCredentials()	10
4.3 Класс Error	11
4.4 Класс Logger	11
4.4.1 Подробное описание	11
4.4.2 Конструктор(ы)	11
4.4.2.1 Logger()	11
4.4.3 Методы	12
4.4.3.1 getLogFilePath()	12
4.4.3.2 write()	12
4.5 Класс Server	13
4.5.1 Подробное описание	13
4.5.2 Конструктор(ы)	13
4.5.2.1 Server()	13
4.5.3 Методы	14
4.5.3.1 authenticateClient()	14
4.5.3.2 calculateAndSendProduct()	14
4.5.3.3 getAddr()	14
4.5.3.4 getData()	15

4.5.3.5	getPort()	15
4.5.3.6	initializeServer()	15
4.5.3.7	terminateServer()	15
4.5.3.8	waitForClientConnection()	16
5	Файлы	17
5.1	cli.h	17
5.2	dataworker.h	17
5.3	error.h	18
5.4	logger.h	18
5.5	md5.h	18
5.6	server.h	18
	Предметный указатель	21

# Глава 1

## Иерархический список классов

### 1.1 Иерархия классов

Иерархия классов.

CLI . . . . .	7
DataWorker . . . . .	9
Logger . . . . .	11
runtime_error	
Error . . . . .	11
Server . . . . .	13



## Глава 2

# Алфавитный указатель классов

### 2.1 Классы

Классы с их кратким описанием.

<a href="#">CLI</a>	Класс для обработки аргументов командной строки . . . . .	<a href="#">7</a>
<a href="#">DataWorker</a>	Класс для работы с данными из файла . . . . .	<a href="#">9</a>
<a href="#">Error</a>	Класс для представления ошибок в приложении . . . . .	<a href="#">11</a>
<a href="#">Logger</a>	Класс для записи сообщений об ошибках в лог-файл . . . . .	<a href="#">11</a>
<a href="#">Server</a>	Класс для реализации сервера . . . . .	<a href="#">13</a>





## Глава 3

# Список файлов

### 3.1 Файлы

Полный список документированных файлов.

code/ <a href="#">cli.h</a> . . . . .	17
code/ <a href="#">dataworker.h</a> . . . . .	17
code/ <a href="#">error.h</a> . . . . .	18
code/ <a href="#">logger.h</a> . . . . .	18
code/ <a href="#">md5.h</a> . . . . .	18
code/ <a href="#">server.h</a> . . . . .	18



## Глава 4

# Классы

### 4.1 Класс CLI

Класс для обработки аргументов командной строки.

```
#include <cli.h>
```

Открытые члены

- [CLI](#) ()  
Конструктор по умолчанию.
- void [parseArgs](#) (int argc, char \*argv[])  
Парсит аргументы командной строки.
- void [showHelp](#) () const  
Печатает справку по использованию программы.
- string [getAddr](#) () const  
Возвращает адрес сервера.
- int [getPort](#) () const  
Возвращает порт сервера.
- string [getLogFile](#) () const  
Возвращает путь к лог-файлу.
- string [getDataFile](#) () const  
Возвращает путь к файлу данных.

#### 4.1.1 Подробное описание

Класс для обработки аргументов командной строки.

Класс предоставляет методы для парсинга аргументов командной строки и получения значений аргументов.

#### 4.1.2 Конструктор(ы)

#### 4.1.2.1 CLI()

`CLI::CLI ( )`

Конструктор по умолчанию.

Устанавливает значения аргументов командной строки по умолчанию.

#### 4.1.3 Методы

##### 4.1.3.1 getAddr()

`string CLI::getAddr ( ) const`

Возвращает адрес сервера.

Возвращает

`string` Адрес сервера.

##### 4.1.3.2 getDataFile()

`string CLI::getDataFile ( ) const`

Возвращает путь к файлу данных.

Возвращает

`string` Путь к файлу данных.

##### 4.1.3.3 getLogFile()

`string CLI::getLogFile ( ) const`

Возвращает путь к лог-файлу.

Возвращает

`string` Путь к лог-файлу.

## 4.1.3.4 getPort()

```
int CLI::getPort ( ) const
```

Возвращает порт сервера.

Возвращает

int Порт сервера.

## 4.1.3.5 parseArgs()

```
void CLI::parseArgs (
    int argc,
    char * argv[] )
```

Парсит аргументы командной строки.

Аргументы

argc	Количество аргументов.
argv	Массив аргументов.

Исключения

Error	Если возникает ошибка парсинга.
-------	---------------------------------

Объявления и описания членов классов находятся в файлах:

- code/cli.h
- code/cli.cpp

## 4.2 Класс DataWorker

Класс для работы с данными из файла.

```
#include <dataworker.h>
```

Открытые члены

- [DataWorker](#) (const string &filePath)  
Конструктор класса [DataWorker](#).
- vector< pair< string, string > > [readCredentials](#) () const  
Читает учетные данные из файла.
- string [getFilePath](#) () const  
Возвращает путь к файлу данных.

### 4.2.1 Подробное описание

Класс для работы с данными из файла.

Класс предоставляет методы для чтения учетных данных из файла.

### 4.2.2 Конструктор(ы)

#### 4.2.2.1 DataWorker()

```
DataWorker::DataWorker (
    const string & filePath ) [explicit]
```

Конструктор класса [DataWorker](#).

Аргументы

filePath	Путь к файлу данных.
----------	----------------------

### 4.2.3 Методы

#### 4.2.3.1 getFilePath()

```
string DataWorker::getFilePath ( ) const
```

Возвращает путь к файлу данных.

Возвращает

```
string Путь к файлу данных.
```

#### 4.2.3.2 readCredentials()

```
vector< pair< string, string > > DataWorker::readCredentials ( ) const
```

Читает учетные данные из файла.

Возвращает

```
vector<pair<string, string>> Вектор пар логин/пароль.
```

Исключения

Error	Если не удастся открыть файл или данные имеют неправильный формат.
-------	--

Объявления и описания членов классов находятся в файлах:

- code/dataworker.h
- code/dataworker.cpp

## 4.3 Класс Error

Класс для представления ошибок в приложении.

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

Граф наследования:Error:

## 4.4 Класс Logger

Класс для записи сообщений об ошибках в лог-файл.

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

Открытые члены

- [Logger](#) (const string &logFilePath)  
Конструктор класса [Logger](#).
- void [write](#) (const [Error](#) &exception)  
Записывает сообщение об ошибке в лог-файл.
- string [getLogFilePath](#) () const  
Возвращает путь к лог-файлу.

### 4.4.1 Подробное описание

Класс для записи сообщений об ошибках в лог-файл.

Класс предоставляет методы для записи сообщений об ошибках в лог-файл.

### 4.4.2 Конструктор(ы)

#### 4.4.2.1 Logger()

```
Logger::Logger (  
    const string & logFilePath ) [explicit]
```

Конструктор класса [Logger](#).

## Аргументы

logFilePath	Путь к лог-файлу.
-------------	-------------------

## Исключения

runtime_error	Если не удастся открыть лог-файл.
---------------	-----------------------------------

## 4.4.3 Методы

## 4.4.3.1 getLogFilePath()

```
string Logger::getLogFilePath ( ) const
```

Возвращает путь к лог-файлу.

Возвращает

string Путь к лог-файлу.

## 4.4.3.2 write()

```
void Logger::write (
    const Error & exception )
```

Записывает сообщение об ошибке в лог-файл.

## Аргументы

exception	Объект ошибки для записи.
-----------	---------------------------

## Исключения

runtime_error	Если лог-файл не открыт.
---------------	--------------------------

Объявления и описания членов классов находятся в файлах:

- code/logger.h
- code/logger.cpp



## 4.5 Класс Server

Класс для реализации сервера.

```
#include <server.h>
```

### Открытые члены

- [Server](#) (const std::string &addr, int port, const std::vector< std::pair< std::string, std::string > > &data)  
Конструктор класса [Server](#).
- void [initializeServer](#) ()  
Инициализирует сервер.
- void [terminateServer](#) ()  
Завершает работу сервера.
- std::string & [getAddr](#) ()  
Возвращает адрес сервера.
- int & [getPort](#) ()  
Возвращает порт сервера.
- std::vector< std::pair< std::string, std::string > > & [getData](#) ()  
Возвращает учетные данные для аутентификации.
- void [waitForClientConnection](#) ()  
Ожидает подключения клиента.
- void [authenticateClient](#) ()  
Аутентифицирует клиента.
- void [calculateAndSendProduct](#) ()  
Вычисляет произведения значений в векторе и отправляет их клиенту.

### 4.5.1 Подробное описание

Класс для реализации сервера.

Класс предоставляет методы для инициализации сервера, ожидания подключения клиентов, аутентификации клиентов и вычисления произведений значений.

### 4.5.2 Конструктор(ы)

#### 4.5.2.1 Server()

```
Server::Server (  
    const std::string & addr,  
    int port,  
    const std::vector< std::pair< std::string, std::string > > & data )
```

Конструктор класса [Server](#).

## Аргументы

addr	Адрес сервера.
port	Порт сервера.
data	Вектор пар логин/пароль для аутентификации.

## 4.5.3 Методы

## 4.5.3.1 authenticateClient()

```
void Server::authenticateClient ( )
```

Аутентифицирует клиента.

Проверяет логин и пароль клиента, используя хеширование MD5.

## Исключения

Error	Если аутентификация неуспешна.
-------	--------------------------------

## 4.5.3.2 calculateAndSendProduct()

```
void Server::calculateAndSendProduct ( )
```

Вычисляет произведения значений в векторе и отправляет их клиенту.

## Исключения

Error	Если возникает ошибка при чтении или отправке данных.
-------	---

## 4.5.3.3 getAddr()

```
string & Server::getAddr ( )
```

Возвращает адрес сервера.

Возвращает

std::string Адрес сервера.

#### 4.5.3.4 getData()

```
vector< pair< string, string > > & Server::getData ( )
```

Возвращает учетные данные для аутентификации.

Возвращает

`std::vector<std::pair<std::string, std::string>>` Вектор пар логин/пароль.

#### 4.5.3.5 getPort()

```
int & Server::getPort ( )
```

Возвращает порт сервера.

Возвращает

`int` Порт сервера.

#### 4.5.3.6 initializeServer()

```
void Server::initializeServer ( )
```

Инициализирует сервер.

Настраивает серверное соединение и начинает прослушивание порта.

Исключения

<b>Error</b>	Если возникает ошибка при создании, привязке или прослушивании сокета.
--------------	--

#### 4.5.3.7 terminateServer()

```
void Server::terminateServer ( )
```

Завершает работу сервера.

Закрывает все открытые соединения.

Исключения

Error	Если возникает ошибка при закрытии сокетов.
-------	---

#### 4.5.3.8 waitForClientConnection()

```
void Server::waitForClientConnection ( )
```

Ожидает подключения клиента.

Исключения

Error	Если возникает ошибка при принятии подключения клиента.
-------	---

Объявления и описания членов классов находятся в файлах:

- code/server.h
- code/server.cpp

# Глава 5

## Файлы

### 5.1 cli.h

```
1 #pragma once
2
3 #include <string>
4 #include "error.h"
5 #include <iostream>
6 #include <cstring>
7
8 using namespace std;
9
10 class CLI
11 {
12 public:
13     CLI();
14
15     void parseArgs(int argc, char *argv[]);
16
17     void showHelp() const;
18
19     string getAddr() const;
20
21     int getPort() const;
22
23     string getLogFile() const;
24
25     string getDataFile() const;
26
27 private:
28     string addr;
29     int port;
30     string logf;
31     string data;
32 };
```

### 5.2 dataworker.h

```
1 #pragma once
2
3 #include <string>
4 #include <fstream>
5 #include <vector>
6 #include "error.h"
7
8 using namespace std;
9
10 class DataWorker
11 {
12 public:
13     explicit DataWorker(const string &filePath);
14
15     vector<pair<string, string>> readCredentials() const;
16
17     string getFilePath() const;
18
19 private:
20     string filePath;
21 };
```

## 5.3 error.h

```

1 #pragma once
2
3 #include <stdexcept>
4 #include <string>
5 #include <ctime>
6
7 using namespace std;
8
9 class Error : public runtime_error
10 {
11 public:
12     explicit Error(const string &what_arg, bool is_critical = false);
13
14     explicit Error(const char *what_arg, bool is_critical = false);
15
16     bool isCritical() const;
17
18     string getTimestamp() const;
19
20 private:
21     bool critical;
22     string timestamp;
23 };

```

## 5.4 logger.h

```

1 #pragma once
2
3 #include "error.h"
4 #include <string>
5 #include <fstream>
6
7 using namespace std;
8
9 class Logger
10 {
11 public:
12     explicit Logger(const string &logFilePath);
13
14     void write(const Error &exception);
15
16     string getLogFilePath() const;
17
18 private:
19     string logFilePath;
20     ofstream logFile;
21 };

```

## 5.5 md5.h

```

1 #define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1
2 #pragma once
3
4 #include <string>
5 #include <sstream>
6 #include <cryptopp/hex.h>
7 #include <cryptopp/md5.h>
8 #include <cryptopp/osrng.h>
9 #include <cryptopp/filters.h>
10
11 using namespace std;
12 using namespace CryptoPP;
13 using namespace CryptoPP::Weak1;
14
15 std::string MD5(const std::string &data);

```

## 5.6 server.h

```

1 #pragma once
2
3 #include <string>
4 #include <vector>
5 #include <iostream>
6 #include <cstring>

```

```
7 #include <arpa/inet.h>
8 #include <unistd.h>
9 #include "error.h"
10 #include "md5.h"
11
12 using namespace std;
13
14 class Server
15 {
16 public:
17     Server(const std::string &addr, int port, const std::vector<std::pair<std::string, std::string>> &data);
18
19     void initializeServer();
20
21     void terminateServer();
22
23     std::string &getAddr();
24
25     int &getPort();
26
27     std::vector<std::pair<std::string, std::string>> &getData();
28
29     void waitForClientConnection();
30
31     void authenticateClient();
32
33     void calculateAndSendProduct();
34
35 private:
36     std::string addr;
37     int port;
38     std::vector<std::pair<std::string, std::string>> data;
39     int socket;
40     int client;
41 };
```





# Предметный указатель

authenticateClient  
Server, [14](#)

calculateAndSendProduct  
Server, [14](#)

CLI, [7](#)  
CLI, [7](#)  
getAddr, [8](#)  
getDataFile, [8](#)  
getLogFile, [8](#)  
getPort, [8](#)  
parseArgs, [9](#)  
code/cli.h, [17](#)  
code/dataworker.h, [17](#)  
code/error.h, [18](#)  
code/logger.h, [18](#)  
code/md5.h, [18](#)  
code/server.h, [18](#)

DataWorker, [9](#)  
DataWorker, [10](#)  
getFilePath, [10](#)  
readCredentials, [10](#)

Error, [11](#)

getAddr  
CLI, [8](#)  
Server, [14](#)  
getData  
Server, [14](#)  
getDataFile  
CLI, [8](#)  
getFilePath  
DataWorker, [10](#)  
getLogFile  
CLI, [8](#)  
getLogFilePath  
Logger, [12](#)  
getPort  
CLI, [8](#)  
Server, [15](#)

initializeServer  
Server, [15](#)

Logger, [11](#)  
getLogFilePath, [12](#)  
Logger, [11](#)  
write, [12](#)

parseArgs  
CLI, [9](#)

readCredentials  
DataWorker, [10](#)

Server, [13](#)  
authenticateClient, [14](#)  
calculateAndSendProduct, [14](#)  
getAddr, [14](#)  
getData, [14](#)  
getPort, [15](#)  
initializeServer, [15](#)  
Server, [13](#)  
terminateServer, [15](#)  
waitForClientConnection, [16](#)

terminateServer  
Server, [15](#)

waitForClientConnection  
Server, [16](#)

write  
Logger, [12](#)