

Удалов В.В. 23ПТ1

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 Calculator	7
4.1.1 Подробное описание	7
4.1.2 Конструктор(ы)	7
4.1.2.1 Calculator()	7
4.1.3 Методы	8
4.1.3.1 send_res()	8
4.2 Класс Communicate	8
4.2.1 Подробное описание	8
4.2.2 Методы	9
4.2.2.1 connection()	9
4.2.2.2 generate_salt()	9
4.2.2.3 sha256()	9
4.3 Класс Connector	10
4.3.1 Подробное описание	10
4.3.2 Методы	10
4.3.2.1 connect()	10
4.3.2.2 get_data()	11
4.4 Класс crit_err	11
4.4.1 Подробное описание	12
4.4.2 Конструктор(ы)	12
4.4.2.1 crit_err()	12
4.5 Класс Interface	12
4.5.1 Подробное описание	12
4.5.2 Методы	13
4.5.2.1 comm_proc()	13
4.6 Класс Logger	13
4.6.1 Подробное описание	13
4.6.2 Конструктор(ы)	14
4.6.2.1 Logger()	14
4.6.3 Методы	14
4.6.3.1 set_path()	14
4.6.3.2 writelog()	14
4.7 Класс no_crit_err	15
4.7.1 Подробное описание	16

---

4.7.2 Конструктор(ы) . . . . .	16
4.7.2.1 no_crit_err() . . . . .	16
5 Файлы . . . . .	17
5.1 Файл Calculator.h . . . . .	17
5.1.1 Подробное описание . . . . .	18
5.2 Calculator.h . . . . .	18
5.3 Файл Communicate.h . . . . .	18
5.3.1 Подробное описание . . . . .	19
5.4 Communicate.h . . . . .	19
5.5 Файл Connector.h . . . . .	20
5.5.1 Подробное описание . . . . .	21
5.6 Connector.h . . . . .	21
5.7 Файл Errors.h . . . . .	21
5.7.1 Подробное описание . . . . .	22
5.8 Errors.h . . . . .	22
5.9 Includer.h . . . . .	23
5.10 Файл Interface.h . . . . .	23
5.10.1 Подробное описание . . . . .	24
5.11 Interface.h . . . . .	24
5.12 Файл Logger.h . . . . .	24
5.12.1 Подробное описание . . . . .	25
5.13 Logger.h . . . . .	26
5.14 Файл main.cpp . . . . .	26
5.14.1 Подробное описание . . . . .	27
5.14.2 Функции . . . . .	27
5.14.2.1 main() . . . . .	27
Предметный указатель . . . . .	29

# Глава 1

## Иерархический список классов

### 1.1 Иерархия классов

Иерархия классов.

Calculator . . . . .	7
Communicate . . . . .	8
Connector . . . . .	10
Interface . . . . .	12
Logger . . . . .	13
std::runtime_error	
crit_err . . . . .	11
no_crit_err . . . . .	15



## Глава 2

# Алфавитный указатель классов

### 2.1 Классы

Классы с их кратким описанием.

<a href="#">Calculator</a>	Выполняет вычисления на векторе целых чисел . . . . .	7
<a href="#">Communicate</a>	Обрабатывает обмен данными между сервером и клиентом . . . . .	8
<a href="#">Connector</a>	Управляет подключениями к базе данных . . . . .	10
<a href="#">crit_err</a>	Представляет критическую ошибку . . . . .	11
<a href="#">Interface</a>	Управляет процессом обмена сообщениями с клиентами . . . . .	12
<a href="#">Logger</a>	Класс для обработки записи логов . . . . .	13
<a href="#">no_crit_err</a>	Представляет некритическую ошибку . . . . .	15





## Глава 3

# Список файлов

### 3.1 Файлы

Полный список документированных файлов.

<a href="#">Calculator.h</a>	
Объявление класса <a href="#">Calculator</a> для векторных вычислений . . . . .	17
<a href="#">Communicate.h</a>	
Объявление класса <a href="#">Communicate</a> для серверного обмена данными . . . . .	18
<a href="#">Connector.h</a>	
Объявление класса <a href="#">Connector</a> для управления подключениями к базе данных . .	20
<a href="#">Errors.h</a>	
Определяет классы исключений для обработки ошибок . . . . .	21
<a href="#">Includer.h</a> . . . . .	??
<a href="#">Interface.h</a>	
Объявление класса <a href="#">Interface</a> для обработки взаимодействий с пользователем . . .	23
<a href="#">Logger.h</a>	
Объявление класса <a href="#">Logger</a> для ведения логов . . . . .	24
<a href="#">main.cpp</a>	
Главная точка входа для приложения . . . . .	26



## Глава 4

# Классы

### 4.1 Класс Calculator

Выполняет вычисления на векторе целых чисел.

```
#include <Calculator.h>
```

Открытые члены

- [Calculator](#) (std::vector< uint16\_t > input\_data)  
Конструктор, инициализирующий вычисления с входными данными.
- uint16\_t [send\\_res](#) ()  
Возвращает результат вычисления.

#### 4.1.1 Подробное описание

Выполняет вычисления на векторе целых чисел.

Обрабатывает массив целых чисел и вычисляет произведение.

#### 4.1.2 Конструктор(ы)

##### 4.1.2.1 Calculator()

```
Calculator::Calculator (  
    std::vector< uint16_t > input_data )
```

Конструктор, инициализирующий вычисления с входными данными.

## Аргументы

input_data	Вектор целых чисел для вычисления.
------------	------------------------------------

## 4.1.3 Методы

## 4.1.3.1 send\_res()

```
uint16_t Calculator::send_res ( )
```

Возвращает результат вычисления.

Возвращает

uint16\_t Результат вычисления.

Объявления и описания членов классов находятся в файлах:

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

## 4.2 Класс Communicate

Обрабатывает обмен данными между сервером и клиентом.

```
#include <Communicate.h>
```

## Открытые члены

- int [connection](#) (int port, std::map< std::string, std::string > database, [Logger](#) \*l1)  
Устанавливает соединение с клиентом.

## Открытые статические члены

- static std::string [sha256](#) (std::string input\_str)  
Вычисляет SHA256-хэш строки.
- static std::string [generate\\_salt](#) ()  
Генерирует случайный соль.

## 4.2.1 Подробное описание

Обрабатывает обмен данными между сервером и клиентом.

### 4.2.2 Методы

#### 4.2.2.1 connection()

```
int Communicate::connection (
    int port,
    std::map< std::string, std::string > database,
    Logger * l1 )
```

Устанавливает соединение с клиентом.

Аргументы

port	Порт сервера.
database	База данных с учетными данными пользователей.
l1	Экземпляр <a href="#">Logger</a> для ведения журнала.

Возвращает

int Код состояния.

#### 4.2.2.2 generate\_salt()

```
std::string Communicate::generate_salt ( ) [static]
```

Генерирует случайный соль.

Возвращает

std::string Сгенерированная соль в шестнадцатеричном формате.

#### 4.2.2.3 sha256()

```
std::string Communicate::sha256 (
    std::string input_str ) [static]
```

Вычисляет SHA256-хэш строки.

Аргументы

input_str	Входная строка.
-----------	-----------------

Возвращает

std::string SHA256-хэш входной строки.

Объявления и описания членов классов находятся в файлах:

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

## 4.3 Класс Connector

Управляет подключениями к базе данных.

```
#include <Connector.h>
```

Открытые члены

- int [connect](#) (std::string base\_file="/home/stud/Server/test\_files/auth.txt")  
Подключается к базе данных.
- std::map< std::string, std::string > [get\\_data](#) ()  
Получает данные из базы данных.

### 4.3.1 Подробное описание

Управляет подключениями к базе данных.

Загружает данные пользователей из файла и хранит их в словаре.

### 4.3.2 Методы

#### 4.3.2.1 connect()

```
int Connector::connect (  
    std::string base_file = "/home/stud/Server/test_files/auth.txt" )
```

Подключается к базе данных.

Аргументы

base_file	Путь к файлу базы данных.
-----------	---------------------------

Возвращает

int Код состояния.

#### 4.3.2.2 get\_data()

```
std::map< std::string, std::string > Connector::get_data ( )
```

Получает данные из базы данных.

Возвращает

std::map<std::string, std::string> Словарь записей в базе данных.

Объявления и описания членов классов находятся в файлах:

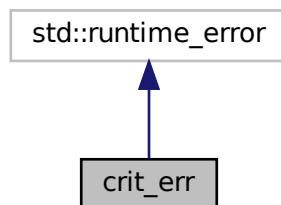
- [Connector.h](#)
- Connector.cpp

## 4.4 Класс crit\_err

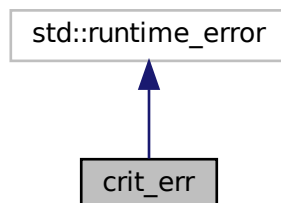
Представляет критическую ошибку.

```
#include <Errors.h>
```

Граф наследования:crit\_err:



Граф связей класса crit\_err:



## Открытые члены

- [crit\\_err](#) (const std::string &s)  
Конструктор для [crit\\_err](#).

### 4.4.1 Подробное описание

Представляет критическую ошибку.

### 4.4.2 Конструктор(ы)

#### 4.4.2.1 crit\_err()

```
crit_err::crit_err (
    const std::string & s ) [inline]
```

Конструктор для [crit\\_err](#).

Аргументы

s	Описание ошибки.
---	------------------

Объявления и описания членов класса находятся в файле:

- [Errors.h](#)

## 4.5 Класс Interface

Управляет процессом обмена сообщениями с клиентами.

```
#include <Interface.h>
```

## Открытые члены

- Interface ()  
Конструктор.
- int [comm\\_proc](#) (int argc, const char \*\*argv)  
Обрабатывает аргументы командной строки и инициализирует обмен сообщениями.

### 4.5.1 Подробное описание

Управляет процессом обмена сообщениями с клиентами.

Обрабатывает аргументы командной строки и настраивает логирование.



### 4.5.2 Методы

#### 4.5.2.1 comm\_proc()

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

Обрабатывает аргументы командной строки и инициализирует обмен сообщениями.

Аргументы

argc	Количество аргументов командной строки.
argv	Массив аргументов командной строки.

Возвращает

int Код состояния.

Объявления и описания членов классов находятся в файлах:

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

## 4.6 Класс Logger

Класс для обработки записи логов.

```
#include <Logger.h>
```

Открытые члены

- int [writelog](#) (std::string s)  
Записывает строку в лог.
- int [set\\_path](#) (std::string path\_file)  
Устанавливает путь к файлу лога.
- [Logger](#) ()  
Конструктор по умолчанию.
- [Logger](#) (std::string s)  
Конструктор с указанием пути к файлу.

### 4.6.1 Подробное описание

Класс для обработки записи логов.

Позволяет записывать и управлять файлами логов.

## 4.6.2 Конструктор(ы)

### 4.6.2.1 Logger()

```
Logger::Logger (  
    std::string s )
```

Конструктор с указанием пути к файлу.

Аргументы

s	Путь к файлу лога.
---	--------------------

## 4.6.3 Методы

### 4.6.3.1 set\_path()

```
int Logger::set_path (  
    std::string path_file )
```

Устанавливает путь к файлу лога.

Аргументы

path_file	Путь к файлу лога.
-----------	--------------------

Возвращает

int Код состояния.

### 4.6.3.2 writelog()

```
int Logger::writelog (  
    std::string s )
```

Записывает строку в лог.

Аргументы

s	Строка для записи.
---	--------------------

Возвращает

int Код состояния.

Объявления и описания членов классов находятся в файлах:

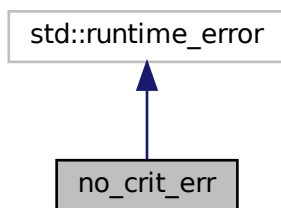
- [Logger.h](#)
- [Logger.cpp](#)

## 4.7 Класс no\_crit\_err

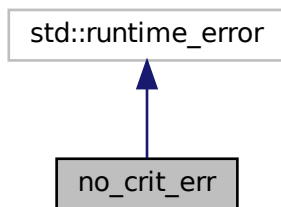
Представляет не критическую ошибку.

```
#include <Errors.h>
```

Граф наследования: no\_crit\_err:



Граф связей класса no\_crit\_err:



Открытые члены

- [no\\_crit\\_err](#) (const std::string s)  
Конструктор для [no\\_crit\\_err](#).

### 4.7.1 Подробное описание

Представляет не критическую ошибку.

### 4.7.2 Конструктор(ы)

#### 4.7.2.1 no\_crit\_err()

```
no_crit_err::no_crit_err (  
    const std::string s )    [inline]
```

Конструктор для [no\\_crit\\_err](#).

Аргументы

s	Описание ошибки.
---	------------------

Объявления и описания членов класса находятся в файле:

- [Errors.h](#)

## Глава 5

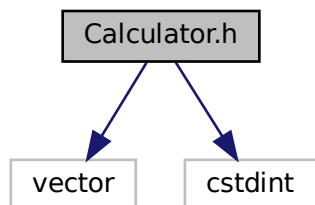
# Файлы

### 5.1 Файл Calculator.h

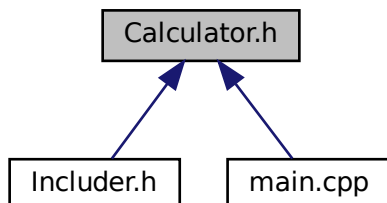
Объявление класса `Calculator` для векторных вычислений.

```
#include <vector>
#include <cstdlib>
```

Граф включаемых заголовочных файлов для Calculator.h:



Граф файлов, в которые включается этот файл:



## Классы

- class [Calculator](#)

Выполняет вычисления на векторе целых чисел.

### 5.1.1 Подробное описание

Объявление класса [Calculator](#) для векторных вычислений.

Версия

1.0

Дата

20.11.2024

Предупреждения

Это учебный пример

## 5.2 Calculator.h

[См. документацию.](#)

```
1
8 #pragma once
9 #include <vector>
10 #include <cstdint>
11
15 class Calculator {
16     uint16_t results;
18 public:
23     Calculator(std::vector<uint16_t> input_data);
24
29     uint16_t send_res();
30 };
```

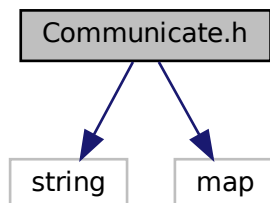
## 5.3 Файл Communicate.h

Объявление класса [Communicate](#) для серверного обмена данными.

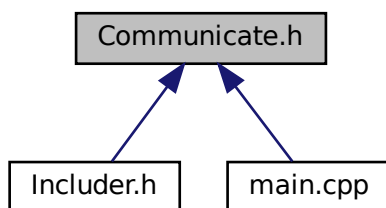
```
#include <string>
```

```
#include <map>
```

Граф включаемых заголовочных файлов для Communicate.h:



Граф файлов, в которые включается этот файл:



## Классы

- class [Communicate](#)

Обрабатывает обмен данными между сервером и клиентом.

### 5.3.1 Подробное описание

Объявление класса [Communicate](#) для серверного обмена данными.

## Версия

1.0

## Дата

20.11.2024

## Предупреждения

Это учебный пример

## 5.4 Communicate.h

[См. документацию.](#)

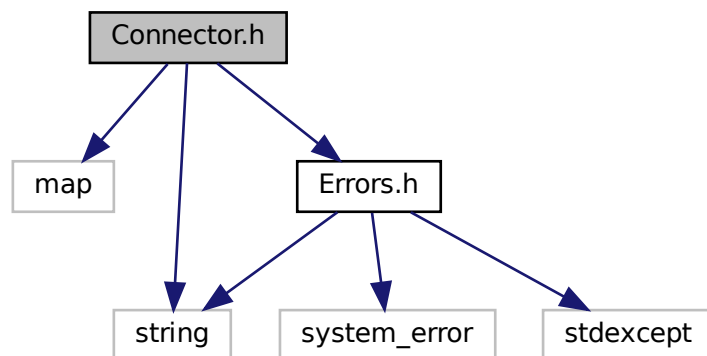
```
1
2 #pragma once
3 #include <string>
4 #include <map>
5
6 class Logger;
7
8 class Communicate {
9 public:
10     int connection(int port, std::map<std::string, std::string> database, Logger* l1);
11
12     static std::string sha256(std::string input_str);
13
14     static std::string generate_salt();
15 };
```

## 5.5 Файл Connector.h

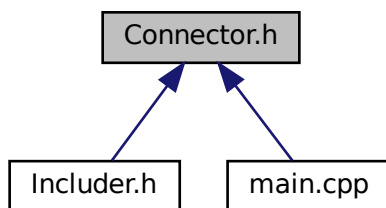
Объявление класса `Connector` для управления подключениями к базе данных.

```
#include <map>
#include <string>
#include "Errors.h"
```

Граф включаемых заголовочных файлов для Connector.h:



Граф файлов, в которые включается этот файл:



### Классы

- class `Connector`

Управляет подключениями к базе данных.



### 5.5.1 Подробное описание

Объявление класса [Connector](#) для управления подключениями к базе данных.

Версия

1.0

Дата

20.11.2024

Предупреждения

Это учебный пример

## 5.6 Connector.h

[См. документацию.](#)

```
1
8 #pragma once
9 #include <map>
10 #include <string>
11 #include "Errors.h"
12
16 class Connector {
17 private:
18     std::map<std::string, std::string> data_base;
20 public:
26     int connect(std::string base_file = "/home/stud/Server/test_files/auth.txt");
27
32     std::map<std::string, std::string> get_data();
33 };
```

## 5.7 Файл Errors.h

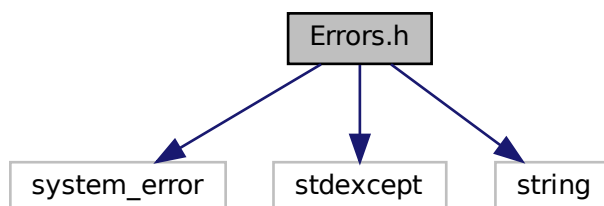
Определяет классы исключений для обработки ошибок.

```
#include <system_error>
```

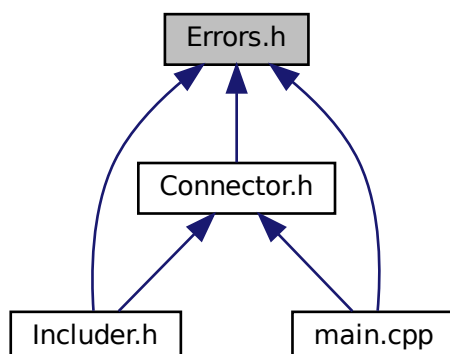
```
#include <stdexcept>
```

```
#include <string>
```

Граф включаемых заголовочных файлов для Errors.h:



Граф файлов, в которые включается этот файл:



## Классы

- class `crit_err`  
Представляет критическую ошибку.
- class `no_crit_err`  
Представляет некритическую ошибку.

### 5.7.1 Подробное описание

Определяет классы исключений для обработки ошибок.

Версия

1.0

Дата

20.11.2024

Предупреждения

Это учебный пример

## 5.8 Errors.h

[См. документацию.](#)

```

1
2
3
4
5
6
7
8 #pragma once
9 #include <system_error>
10 #include <stdexcept>
11 #include <string>
12
13
14
15 class crit_err: public std::runtime_error {
16 public:
17     crit_err(const std::string& s): std::runtime_error(s) {}
18 };
19
20
21
22
23
24
25
26 class no_crit_err: public std::runtime_error {
27 public:
28     no_crit_err(const std::string s): std::runtime_error(s) {}
29 };
30
31
32
33
  
```

## 5.9 Includer.h

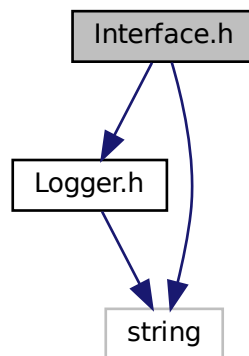
```
1 #include "Connector.h"
2 #include "Communicate.h"
3 #include "Calculator.h"
4 #include "Interface.h"
5 #include "Logger.h"
6 #include "Errors.h"
```

## 5.10 Файл Interface.h

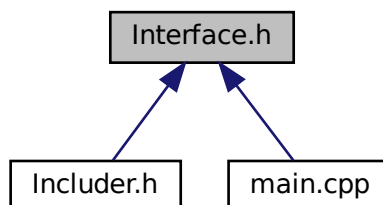
Объявление класса `Interface` для обработки взаимодействий с пользователем.

```
#include "Logger.h"
#include <string>
```

Граф включаемых заголовочных файлов для `Interface.h`:



Граф файлов, в которые включается этот файл:



## Классы

- class `Interface`

Управляет процессом обмена сообщениями с клиентами.

### 5.10.1 Подробное описание

Объявление класса [Interface](#) для обработки взаимодействий с пользователем.

Версия

1.0

Дата

20.11.2024

Предупреждения

Это учебный пример

## 5.11 Interface.h

[См. документацию.](#)

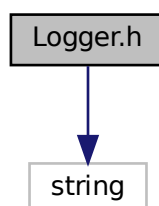
```
1
8 #pragma once
9 #include "Logger.h"
10 #include <string>
11
15 class Interface {
16     int PORT;
18 public:
22     Interface() {}
23
30     int comm_proc(int argc, const char** argv);
31 };
```

## 5.12 Файл Logger.h

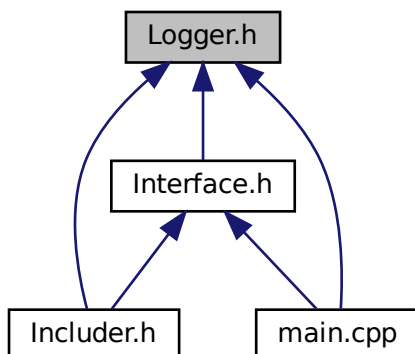
Объявление класса [Logger](#) для ведения логов.

```
#include <string>
```

Граф включаемых заголовочных файлов для Logger.h:



Граф файлов, в которые включается этот файл:



## Классы

- class `Logger`

Класс для обработки записи логов.

### 5.12.1 Подробное описание

Объявление класса `Logger` для ведения логов.

Автор

Удалов В.В.

Версия

1.0

Дата

20.11.2024

Авторство

ИБСТ ПГУ

Предупреждения

Это учебный пример

## 5.13 Logger.h

См. документацию.

```

1
10 #pragma once
11 #include <string>
12
13 class Logger {
14     static std::string getCurrentDateTime(std::string s);
15     std::string path_to_logfile;
16 public:
17     int writelog(std::string s);
18     int set_path(std::string path_file);
19     Logger();
20     Logger(std::string s);
21 };

```

## 5.14 Файл main.cpp

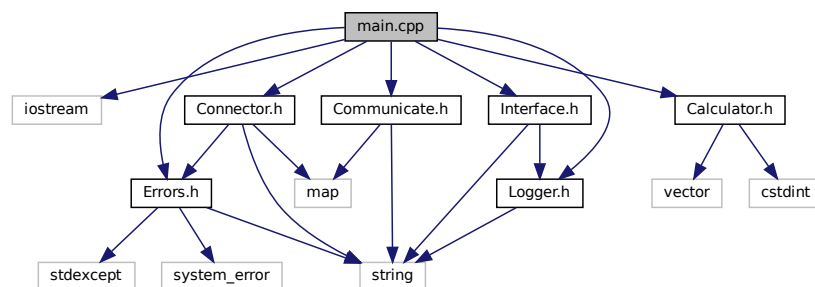
Главная точка входа для приложения.

```

#include <iostream>
#include "Connector.h"
#include "Interface.h"
#include "Communicate.h"
#include "Calculator.h"
#include "Errors.h"
#include "Logger.h"

```

Граф включаемых заголовочных файлов для main.cpp:



## Функции

- int `main` (int argc, const char \*\*argv)

Главная функция для инициализации интерфейса и запуска процесса обмена сообщениями.

### 5.14.1 Подробное описание

Главная точка входа для приложения.

Автор

Удалов В.В.

Версия

1.0

Дата

20.11.2024

Авторство

ИБСТ ПГУ

Предупреждения

Это учебный пример

### 5.14.2 Функции

#### 5.14.2.1 main()

```
int main (  
    int argc,  
    const char ** argv )
```

Главная функция для инициализации интерфейса и запуска процесса обмена сообщениями.

Аргументы

argc	Количество аргументов командной строки.
argv	Массив аргументов командной строки.

Возвращает

int Код завершения программы.





# Предметный указатель

- Calculator, [7](#)
  - Calculator, [7](#)
  - send\_res, [8](#)
- Calculator.h, [17](#)
- comm\_proc
  - Interface, [13](#)
- Communicate, [8](#)
  - connection, [9](#)
  - generate\_salt, [9](#)
  - sha256, [9](#)
- Communicate.h, [18](#)
- connect
  - Connector, [10](#)
- connection
  - Communicate, [9](#)
- Connector, [10](#)
  - connect, [10](#)
  - get\_data, [11](#)
- Connector.h, [20](#)
- crit\_err, [11](#)
  - crit\_err, [12](#)
- Errors.h, [21](#)
- generate\_salt
  - Communicate, [9](#)
- get\_data
  - Connector, [11](#)
- Interface, [12](#)
  - comm\_proc, [13](#)
- Interface.h, [23](#)
- Logger, [13](#)
  - Logger, [14](#)
  - set\_path, [14](#)
  - writelog, [14](#)
- Logger.h, [24](#)
- main
  - main.cpp, [27](#)
- main.cpp, [26](#)
  - main, [27](#)
- no\_crit\_err, [15](#)
  - no\_crit\_err, [16](#)
- send\_res
  - Calculator, [8](#)
- set\_path
  - Logger, [14](#)