SERVER

Создано системой Doxygen 1.9.1

Глава 1

Иерархический список классов

1.1 Иерархия классов

Иерархия классов.

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

Глава 3

Классы

3.1 Класс authorization

Класс для аутентификации клиента на сервере

#include <authorization.h>

Открытые члены

- authorization (string Id, string Password)
  Конструктор для установки идентификатора и пароля клиента
- void generate_salt ()
  Генерация случайной соли для вычисления хэша
- bool compare_hash (string client_hash)
  Сравнение хэша, присылаемого клиентом и хэша, вычисляемого внутри метода
- string get_salt ()
- std::string get_id ()
- std::string get_password ()
- std::string get_hash ()

Открытые атрибуты

- char err [3] = {'E','R','R'}
  Сообщение, отсылаемое клиенту при ошибке его обрабоки
- char ok [2] = {'O','K'}
  Сообщение, отсылаемое клиенту при успешной авторизации

3.1.1 Подробное описание

Класс для аутентификации клиента на сервере

Функция вычисления хэша MD5

3.1.2 Конструктор(ы)

3.1.2.1 authorization()

```
authorization::authorization (
            string Id,
            string Password )
```

Конструктор для установки идентификатора и пароля клиента

Аргументы

in	Id,идентификатор	клиента, string.
in	Password,пароль	клиента, string.

3.1.3 Методы

3.1.3.1 compare_hash()

```
bool authorization::compare_hash (
            string client_hash )
```

Сравнение хэша, присылаемого клиентом и хэша, вычисляемого внутри метода

Аргументы

in	client_hash,хэш	клиента, tring

3.1.3.2 generate_salt()

```
void authorization::generate_salt ( )
```

Генерация случайной соли для вычисления хэша

Аргументы

in	salt,соль,string.	
in	size,размер	соли, size_t.

Объявления и описания членов классов находятся в файлах:

- authorization.h
- authorization.cpp

3.2 Класс base

Класс для работы с базой данных

#include <read_base.h>

Открытые члены

- base (string base_name)
  Конструктор, считывающий базу данных и сохраняющий её в словарь
- bool check_user (string user)
  Проверка наличия идентификатора клиента в базе данных

Открытые атрибуты

- map< string, string > DataBase
  Словарь с парами идентификатор:пароль

3.2.1 Подробное описание

Класс для работы с базой данных

3.2.2 Конструктор(ы)

3.2.2.1 base()

```
base::base (
            string base_name )
```

Конструктор, считывающий базу данных и сохраняющий её в словарь

Аргументы

in	base_name,путь	к файлу с базой данных, string.

3.2.3 Методы

3.2.3.1 check_user()

```
bool base::check_user (
                string user )
```

Проверка наличия идентификатора клиента в базе данных

Аргументы

in	user,идентификатора	клиента, string

Объявления и описания членов классов находятся в файлах:

- read_base.h
- read_base.cpp

3.3 Класс calc

Класс для вычислений по вектору

#include <calculation.h>

Открытые члены

- int16_t * calculation (vector< int16_t > vecto)
  Конструктор без параметров

3.3.1 Подробное описание

Класс для вычислений по вектору

Операция - произведение

3.3.2 Методы

3.3.2.1 calculation()

```
int16_t* calc::calculation (
                vector< int16_t > vecto )   [inline]
```

Конструктор без параметров

Вычисляет произведение вектоов

Аргументы

in	vector,вектор,vector<int16_t>	

Объявления и описания членов класса находятся в файле:

- calculation.h

3.4 Класс connection

Класс соединения с клиентом

#include <connection.h>

Открытые члены

- connection (int port)
  Конструктор
- int Bind ()
  Привязка сокета к адресу
- void Listen ()
  Установка сокета в пассивный режим ожидания
- int Accept ()
  Приём соединения
- int Recv (int s, void *buf, int size)
  Приём данных от клиента
- void Send (int s, void *buf, int sizeb)
  Отправка данных клиенту
- void Close ()
  Закрытие сокета

3.4.1 Подробное описание

Класс соединения с клиентом

Для соединения использованы сокеты

3.4.2 Конструктор(ы)

3.4.2.1 connection()

```
connection::connection (
            int port )
```

Конструктор

Устанавливает порт, инициализирует основной сокет и структуру sockaddr_in

Аргументы

in	port,порт,на	котором работает сервер, int.

3.4.3 Методы

3.4.3.1 Recv()

```
int connection::Recv (
                int s,
                void * buf,
                int size )
```

Приём данных от клиента

Аргументы

in	sock,сокет,int	
in	buf,буфер	для данных, void*
in	size,размер	буфера, int

3.4.3.2 Send()

```
void connection::Send (
                int s,
                void * buf,
                int sizeb )
```

Отправка данных клиенту

Аргументы

in	sock,сокет,int	
in	buf,буфер	с данными, void*
in	sizeb,количество	отправляемых байт, int

Объявления и описания членов классов находятся в файлах:

- connection.h
- connection.cpp

3.5 Класс error_processing

Класс для обработки ошибок и записи в журнал

#include <error_processing.h>

Открытые члены

- string get_log_path (string log_path)
  Конструктор без параметров
- void write_log (string why, bool exit)
  Функция, выдающая путь к файлу с журналом ошибок

3.5.1 Подробное описание

Класс для обработки ошибок и записи в журнал

3.5.2 Методы

3.5.2.1 write_log()

```
void error_processing::write_log (
                string why,
                bool exit )
```

Функция, выдающая путь к файлу с журналом ошибок

Функция записи ошибки в журнал

Функция записывает время, суть и критичность ошибки

Аргументы

in	why,суть	ошибки, string
in	exit,критичность	ошибки (Критическая - true, Некритическая - false), string

Объявления и описания членов классов находятся в файлах:

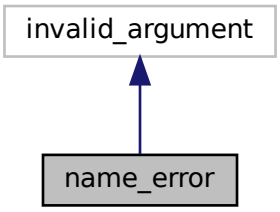
- error_processing.h
- error_processing.cpp

3.6 Класс name_error

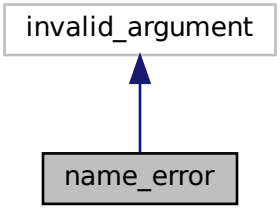
Класс для обработки ошибок

#include <error_processing.h>

Граф наследования:name_error:



Граф связей класса name_error:



Открытые члены

- name_error (const string &whyy, bool exitt=false)
  Конструктор ошибок с строкой в качестве параметра
- name_error (const char *whyy, bool exitt=false)
  Конструктор ошибок с си-строкой в качестве параметра
- bool get_status () const

3.6.1 Подробное описание

Класс для обработки ошибок

Наследует от класса invalid_argument

3.6.2 Конструктор(ы)

3.6.2.1 name_error() [1/2]

name_error::name_error (
const string & whyy,
bool exitt = false) [inline], [explicit]

Конструктор ошибок с строкой в качестве параметра

Аргументы

in	whyy,тип	ошибки, const string.
in	exitt,критическа	ошибка - true, некритическая ошибка - false, bool

3.6.2.2 name_error() [2/2]

name_error::name_error (
const char * whyy,
bool exitt = false) [inline], [explicit]

Конструктор ошибок с си-строкой в качестве параметра

Аргументы

in	whyy,тип	ошибки, const char*.
in	exitt,критическа	ошибка - true, штатная - false, bool

3.6.3 Методы

3.6.3.1 get_status()

bool name_error::get_status ( ) const [inline]

<Возвращает статус критичности ошибки

Объявления и описания членов класса находятся в файле:

- error_processing.h

3.7 Класс server

Класс для получения параметров коммандной строки

#include <server.h>

Открытые члены

- server (int argc, char **argv)

  Конструктор, внутри которого считываются параметры командной строки
- string get_base ()
- string get_log ()
- int get_port ()

3.7.1 Подробное описание

Класс для получения параметров коммандной строки

3.7.2 Конструктор(ы)

3.7.2.1 server()

```
server::server (
                int argc,
                char ** argv )
```

Конструктор, внутри которого считываются параметры командной строки

Параметры командной строки: 1)-b Путь к файлу с базой данных, необязательный 2)-l Путь к файлу для записи логов, необязательный 3)-p Порт, на котором работает сервер, необязательный 4)-h вызов подсказки 5) ? вызов подсказки При ошибках в параметрах или запуске без них вызывается справка и программа завершает работу

Аргументы

in	int	argc
in	char	**argv

Объявления и описания членов классов находятся в файлах:

- server.h
- server.cpp

Предметный указатель