Сервер

Создано системой Doxygen 1.9.4

Глава 1

Алфавитный указатель классов

1.1 Классы

Классы с их кратким описанием.

Глава 2

Список файлов

2.1 Файлы

Полный список файлов.

Глава 3

Классы

3.1 Класс Authenticator

Класс аутентификации

#include <Authenticator.h>

Открытые статические члены

- static std::string generateSalt ()
  Сгенерировать соль
- static std::string calculateSHA256 (const std::string &salt, const std::string &password)
  Вычислить SHA256 хеш от соли и пароля
- static bool verifyHash (const std::string &receivedHash, const std::string &salt, const std::string &storedPassword)
  Проверить хеш
- static bool isValidHexString (const std::string &hexString, size_t expectedLength)
  Проверить hex строку

3.1.1 Подробное описание

Класс аутентификации

3.1.2 Методы

3.1.2.1 calculateSHA256()

std::string Authenticator::calculateSHA256 (
const std::string & salt,
const std::string & password) [static]

Вычислить SHA256 хеш от соли и пароля

Аргументы

salt	Соль
password	Пароль в открытом виде

Возвращает

Хеш в hex формате (64 символа)

Граф вызова функции:

3.1.2.2 generateSalt()

std::string Authenticator::generateSalt ( ) [static]

Сгенерировать соль

Возвращает

Соль в hex формате (16 символов)

Граф вызова функции:

3.1.2.3 isValidHexString()

bool Authenticator::isValidHexString (
const std::string & hexString,
size_t expectedLength) [static]

Проверить hex строку

Аргументы

hexString	Строка
expectedLength	Ожидаемая длина

Возвращает

true - строка корректна

Граф вызова функции:

3.1.2.4 verifyHash()

bool Authenticator::verifyHash (
const std::string & receivedHash,

const std::string & salt,
const std::string & storedPassword) [static]

Проверить хеш

Аргументы

receivedHash	Хеш от клиента
salt	Соль
storedPassword	Пароль в открытом виде из базы

Возвращает

true - хеши совпадают

Граф вызовов: Граф вызова функции:

Объявления и описания членов классов находятся в файлах:

- src/Authenticator.h
- src/Authenticator.cpp

3.2 Класс Config

Класс конфигурации сервера

#include <Config.h>

Открытые члены

- Config ()
  Конструктор по умолчанию
- bool parseCommandLine (int argc, char **argv)
  Парсинг параметров командной строки
- void setDefaults ()
  Установка значений по умолчанию
- const std::string & getClientDbPath () const
- const std::string & getLogFilePath () const
- uint16_t getPort () const
- void showHelp (const char *programName)
  Показать справку
- void showVersion ()
  Показать информацию о версии программы

Закрытые данные

- std::string clientDbPath_
- std::string logFilePath_
- uint16_t port_

3.2.1 Подробное описание

Класс конфигурации сервера

3.2.2 Конструктор(ы)

3.2.2.1 Config()

```
Config::Config ( )
```

Конструктор по умолчанию

Граф вызовов:

3.2.3 Методы

3.2.3.1 getClientDbPath()

```
const std::string & Config::getClientDbPath ( ) const
```

Граф вызова функции:

3.2.3.2 getLogFilePath()

```
const std::string & Config::getLogFilePath ( ) const
```

Граф вызова функции:

3.2.3.3 getPort()

```
uint16_t Config::getPort ( ) const
```

Граф вызова функции:

3.2.3.4 parseCommandLine()

```
bool Config::parseCommandLine (
            int argc,
            char ** argv )
```

Парсинг параметров командной строки

Аргументы

argc	Количество аргументов
argv	Массив аргументов

Возвращает

true - успешный парсинг

Граф вызовов: Граф вызова функции:

3.2.3.5 setDefaults()

void Config::setDefaults ( )

Установка значений по умолчанию

Граф вызова функции:

3.2.3.6 showHelp()

void Config::showHelp (
const char * programName)

Показать справку

Аргументы

programName	Имя программы

Граф вызова функции:

3.2.3.7 showVersion()

void Config::showVersion ( )

Показать информацию о версии программы

Граф вызова функции:

3.2.4 Данные класса

3.2.4.1 clientDbPath_

std::string Config::clientDbPath_ [private]

3.2.4.2 logFilePath_

std::string Config::logFilePath_ [private]

3.2.4.3 port_

uint16_t Config::port_ [private]

Объявления и описания членов классов находятся в файлах:

- src/Config.h
- src/Config.cpp

3.3 Класс Database

Класс базы данных клиентов

#include <Database.h>

Открытые члены

- bool loadFromFile (const std::string &filename)
  Загрузить базу из файла
- bool userExists (const std::string &login) const
  Проверить существование пользователя
- std::string getPassword (const std::string &login) const
  Получить пароль пользователя
- size_t getClientCount () const
  Получить количество клиентов

Закрытые данные

- std::unordered_map< std::string, std::string > clients_

3.3.1 Подробное описание

Класс базы данных клиентов

3.3.2 Методы

3.3.2.1 getClientCount()

size_t Database::getClientCount ( ) const [inline]

Получить количество клиентов

Возвращает

Количество клиентов

Граф вызова функции:

3.3.2.2 getPassword()

std::string Database::getPassword (
const std::string & login) const

Получить пароль пользователя

Аргументы

login	Логин

Возвращает

Пароль в открытом виде

Граф вызова функции:

3.3.2.3 loadFromFile()

bool Database::loadFromFile (
const std::string & filename)

Загрузить базу из файла

Аргументы

filename	Имя файла

Возвращает

true - успешно

Граф вызова функции:

3.3.2.4 userExists()

bool Database::userExists (
const std::string & login) const

Проверить существование пользователя

Аргументы

login	Логин

Возвращает

true - существует

Граф вызова функции:

3.3.3 Данные класса

3.3.3.1 clients_

std::unordered_map<std::string, std::string> Database::clients_ [private]

Объявления и описания членов классов находятся в файлах:

- src/Database.h
- src/Database.cpp

3.4 Класс Logger

Класс логирования

#include <Logger.h>

Открытые члены

- Logger (const std::string &filePath)
  Конструктор
- ~Logger ()
  Деструктор
- void log (LogLevel level, const std::string &message, const std::string &details="")
  Записать сообщение в журнал
- void logSystemError (const std::string &context)
  Записать системную ошибку

Закрытые члены

- std::string levelToString (LogLevel level) const
- std::string getCurrentDateTime () const

Закрытые данные

- std::string logFilePath_
- std::ofstream logFile_

3.4.1 Подробное описание

Класс логирования

3.4.2 Конструктор(ы)

3.4.2.1 Logger()

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

Конструктор

Аргументы

filePath	Путь к файлу журнала

3.4.2.2 ~Logger()

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

Деструктор

3.4.3 Методы

3.4.3.1 getCurrentDateTime()

```
std::string Logger::getCurrentDateTime ( ) const    [private]
```

Граф вызова функции:

3.4.3.2 levelToString()

std::string Logger::levelToString (
LogLevel level) const [private]

Граф вызова функции:

3.4.3.3 log()

void Logger::log (
LogLevel level,
const std::string & message,
const std::string & details = "")

Записать сообщение в журнал

Аргументы

level	Уровень
message	Сообщение
details	Детали

Граф вызовов: Граф вызова функции:

3.4.3.4 logSystemError()

void Logger::logSystemError (
const std::string & context)

Записать системную ошибку

Аргументы

context	Контекст

Граф вызовов: Граф вызова функции:

3.4.4 Данные класса

3.4.4.1 logFile_

std::ofstream Logger::logFile_ [private]

3.4.4.2 logFilePath_

std::string Logger::logFilePath_ [private]

Объявления и описания членов классов находятся в файлах:

- src/Logger.h
- src/Logger.cpp

3.5 Класс Server

Главный класс сервера

#include <Server.h>

Граф связей класса Server:

Открытые члены

- Server (const Config &config)
  Конструктор
- ~Server ()
  Деструктор
- bool start ()
  Запустить сервер
- void stop ()
  Остановить сервер
- bool isRunning () const
  Проверить работает ли сервер

Закрытые члены

- bool initializeNetwork ()
  Инициализировать сетевое соединение
- void mainLoop ()
  Главный цикл сервера
- void handleClient (int clientSocket)
  Обработать клиента
- bool authenticateClient (int clientSocket, std::string &clientLogin)
  Аутентифицировать клиента
- void processVectorData (int clientSocket)
  Обработать векторные данные
- bool sendString (int socket, const std::string &str)
  Отправить строку клиенту
- bool recvString (int socket, std::string &str, size_t maxLength=1024)
  Получить строку от клиента
- bool sendAll (int socket, const void *data, size_t size)
  Отправить двоичные данные
- bool recvAll (int socket, void *data, size_t size)
  Получить двоичные данные
- void closeConnection (int socket)
  Закрыть соединение

Закрытые данные

- Config config_
- Database database_
- Logger logger_
- int serverSocket_
- bool running_

3.5.1 Подробное описание

Главный класс сервера

3.5.2 Конструктор(ы)

3.5.2.1 Server()

```
Server::Server (
            const Config & config )
```

Конструктор

Аргументы

config	Конфигурация

Граф вызовов:

3.5.2.2 ~Server()

```
Server::~Server ( )
```

Деструктор

Граф вызовов:

3.5.3 Методы

3.5.3.1 authenticateClient()

```
bool Server::authenticateClient (
            int clientSocket,
            std::string & clientLogin )    [private]
```

Аутентифицировать клиента

Аргументы

clientSocket	Сокет клиента
clientLogin	Логин клиента (выходной параметр)

Возвращает

true - аутентификация успешна

Граф вызовов: Граф вызова функции:

3.5.3.2 closeConnection()

```
void Server::closeConnection (
            int socket )   [private]
```

Закрыть соединение

Аргументы

socket	Сокет

Граф вызова функции:

3.5.3.3 handleClient()

```
void Server::handleClient (
            int clientSocket )   [private]
```

Обработать клиента

Аргументы

clientSocket	Сокет клиента

Граф вызовов: Граф вызова функции:

3.5.3.4 initializeNetwork()

```
bool Server::initializeNetwork ( )   [private]
```

Инициализировать сетевое соединение

Возвращает

true - успешно

Граф вызовов: Граф вызова функции:

3.5.3.5 isRunning()

bool Server::isRunning ( ) const [inline]

Проверить работает ли сервер

Возвращает

true - сервер работает

3.5.3.6 mainLoop()

void Server::mainLoop ( ) [private]

Главный цикл сервера

Граф вызовов: Граф вызова функции:

3.5.3.7 processVectorData()

void Server::processVectorData (
int clientSocket) [private]

Обработать векторные данные

Аргументы

clientSocket	Сокет клиента

Граф вызовов: Граф вызова функции:

3.5.3.8 recvAll()

bool Server::recvAll (
int socket,
void * data,
size_t size) [private]

Получить двоичные данные

Аргументы

socket	Сокет
data	Данные (выходной параметр)
size	Размер данных

Возвращает

true - успешно

Граф вызова функции:

3.5.3.9 recvString()

```
bool Server::recvString (
            int socket,
            std::string & str,
            size_t maxLength = 1024 )   [private]
```

Получить строку от клиента

Аргументы

socket	Сокет
str	Строка (выходной параметр)
maxLength	Максимальная длина

Возвращает

true - успешно

Граф вызова функции:

3.5.3.10 sendAll()

```
bool Server::sendAll (
            int socket,
            const void * data,
            size_t size )   [private]
```

Отправить двоичные данные

Аргументы

socket	Сокет
data	Данные
size	Размер данных

Возвращает

true - успешно

Граф вызова функции:

3.5.3.11 sendString()

```
bool Server::sendString (
                int socket,
                const std::string & str )   [private]
```

Отправить строку клиенту

Аргументы

socket	Сокет
str	Строка

Возвращает

true - успешно

Граф вызовов: Граф вызова функции:

3.5.3.12 start()

```
bool Server::start ( )
```

Запустить сервер

Возвращает

true - успешный запуск

Граф вызовов: Граф вызова функции:

3.5.3.13 stop()

```
void Server::stop ( )
```

Остановить сервер

Граф вызовов: Граф вызова функции:

3.5.4 Данные класса

3.5.4.1 config_

```
Config Server::config_   [private]
```

3.5.4.2 database_

Database Server::database_ [private]

3.5.4.3 logger_

Logger Server::logger_ [private]

3.5.4.4 running_

bool Server::running_ [private]

3.5.4.5 serverSocket_

int Server::serverSocket_ [private]

Объявления и описания членов классов находятся в файлах:

- src/Server.h
- src/Server.cpp

3.6 Класс VectorProcessor

Класс обработки векторов

#include <VectorProcessor.h>

Открытые статические члены

- static int32_t calculateSum (const std::vector< int32_t > &vector)
  Вычислить сумму вектора
- static std::vector< int32_t > processVectors (const std::vector< std::vector< int32_t > > &vectors)
  Обработать массив векторов

Закрытые статические члены

- static bool willOverflowAdd (int32_t a, int32_t b)
  Проверить переполнение

Закрытые статические данные

- static const int32_t MAX_INT32 = INT_MAX
- static const int32_t MIN_INT32 = INT_MIN

3.6.1 Подробное описание

Класс обработки векторов

3.6.2 Методы

3.6.2.1 calculateSum()

int32_t VectorProcessor::calculateSum (
const std::vector< int32_t > & vector) [static]

Вычислить сумму вектора

Аргументы

vector	Вектор для обработки

Возвращает

Сумма элементов

Граф вызовов: Граф вызова функции:

3.6.2.2 processVectors()

std::vector< int32_t > VectorProcessor::processVectors (
const std::vector< std::vector< int32_t > > & vectors) [static]

Обработать массив векторов

Аргументы

vectors	Массив векторов

Возвращает

Массив сумм

Граф вызовов:

3.6.2.3 willOverflowAdd()

```
bool VectorProcessor::willOverflowAdd (
                int32_t a,
                int32_t b )   [static], [private]
```

Проверить переполнение

Аргументы

a	Первое слагаемое
b	Второе слагаемое

Возвращает

true - будет переполнение

Граф вызова функции:

3.6.3 Данные класса

3.6.3.1 MAX_INT32

```
const int32_t VectorProcessor::MAX_INT32 = INT_MAX   [static], [private]
```

3.6.3.2 MIN_INT32

```
const int32_t VectorProcessor::MIN_INT32 = INT_MIN   [static], [private]
```

Объявления и описания членов классов находятся в файлах:

- src/VectorProcessor.h
- src/VectorProcessor.cpp

Глава 4

Файлы

4.1 Файл src/Authenticator.cpp

```
#include "Authenticator.h"
#include <sstream>
#include <iomanip>
#include <cstring>
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <algorithm>
#include <openssl/evp.h>
```

Граф включаемых заголовочных файлов для Authenticator.cpp:

4.2 Файл src/Authenticator.h

Аутентификация клиентов

```
#include <string>
```

Граф включаемых заголовочных файлов для Authenticator.h: Граф файлов, в которые включается этот файл:

Классы

- class Authenticator

  Класс аутентификации

4.2.1 Подробное описание

Аутентификация клиентов

Автор

Сударикov А.В.

Версия

1.0

Дата

2025

4.3 Authenticator.h

См. документацию.

```
1
9 #ifndef AUTHENTICATOR_H
10 #define AUTHENTICATOR_H
11
12 #include <string>
13
17 class Authenticator {
18 public:
23     static std::string generateSalt();
24
31     static std::string calculateSHA256(const std::string& salt,
32                                        const std::string& password);
33
41     static bool verifyHash(const std::string& receivedHash,
42                            const std::string& salt,
43                            const std::string& storedPassword);
44
51     static bool isValidHexString(const std::string& hexString,
52                                  size_t expectedLength);
53 };
54
55 #endif // AUTHENTICATOR_H
```

4.4 Файл src/Config.cpp

#include "Config.h"
#include <iostream>
#include <cstring>
#include <getopt.h>
#include <limits>
Граф включаемых заголовочных файлов для Config.cpp:

4.5 Файл src/Config.h

Конфигурация сервера

#include <string>
#include <cstdint>
Граф включаемых заголовочных файлов для Config.h: Граф файлов, в которые включается этот файл:

Классы

- class Config

  Класс конфигурации сервера

Перечисления

- enum class LogLevel { INFO , WARNING , ERROR , CRITICAL }

  Уровни логирования

4.5.1 Подробное описание

Конфигурация сервера

Автор

Судариков А.В.

Версия

1.0

Дата

2025

4.5.2 Перечисления

4.5.2.1 LogLevel

enum class LogLevel [strong]

Уровни логирования

Элементы перечислений

INFO	Информационное сообщение
WARNING	Предупреждение
ERROR	Ошибка
CRITICAL	Критическая ошибка

4.6 Config.h

См. документацию.

```
1
9 #ifndef CONFIG_H
10 #define CONFIG_H
11
12 #include <string>
13 #include <cstdint>
14
18 enum class LogLevel {
19     INFO,
20     WARNING,
21     ERROR,
22     CRITICAL
23 };
24
28 class Config {
29 private:
```

```
30     std::string clientDbPath_;
31     std::string logFilePath_;
32     uint16_t port_;
33
34 public:
38     Config();
39
46     bool parseCommandLine(int argc, char** argv);
47
51     void setDefaults();
52
53     // Геттеры (только объявления)
54     const std::string& getClientDbPath() const;
55     const std::string& getLogFilePath() const;
56     uint16_t getPort() const;
57
62     void showHelp(const char* programName);
63
67     void showVersion();
68 };
69
70 #endif // CONFIG_H
```

4.7 Файл src/Database.cpp

#include "Database.h"
#include <fstream>
#include <sstream>
#include <algorithm>
#include <iostream>
#include <cctype>
Граф включаемых заголовочных файлов для Database.cpp:

4.8 Файл src/Database.h

База данных клиентов

#include <string>
#include <unordered_map>
Граф включаемых заголовочных файлов для Database.h: Граф файлов, в которые включается этот файл:

Классы

- class Database

  Класс базы данных клиентов

4.8.1 Подробное описание

База данных клиентов

Автор

Судариков А.В.

Версия

1.0

Дата

2025

4.9 Database.h

См. документацию.

```
1
9 #ifndef DATABASE_H
10 #define DATABASE_H
11
12 #include <string>
13 #include <unordered_map>
14
18 class Database {
19 private:
20     std::unordered_map<std::string, std::string> clients_; // login -> password (open text)
21
22 public:
28     bool loadFromFile(const std::string& filename);
29
35     bool userExists(const std::string& login) const;
36
42     std::string getPassword(const std::string& login) const;
43
48     size_t getClientCount()const { return clients_.size(); }
49 };
50
51 #endif // DATABASE_H
```

4.10 Файл src/Logger.cpp

#include "Logger.h"
#include <iostream>
#include <iomanip>
#include <cstring>
Граф включаемых заголовочных файлов для Logger.cpp:

4.11 Файл src/Logger.h

Логирование сервера

#include <string>
#include <fstream>
#include <ctime>
#include "Config.h"
Граф включаемых заголовочных файлов для Logger.h: Граф файлов, в которые включается этот файл:

Классы

- class Logger

  Класс логирования

4.11.1 Подробное описание

Логирование сервера

Автор

Судариков А.В.

Версия

1.0

Дата

2025

4.12 Logger.h

См. документацию.

```
1
9 #ifndef LOGGER_H
10 #define LOGGER_H
11
12 #include <string>
13 #include <fstream>
14 #include <ctime>
15 #include "Config.h"
16
20 class Logger {
21 private:
22     std::string logFilePath_;
23     std::ofstream logFile_;
24
25     std::string levelToString(LogLevel level) const;
26     std::string getCurrentDateTime() const;
27
28 public:
33     Logger(const std::string& filePath);
34
38     ~Logger();
39
46     void log(LogLevel level, const std::string& message,
47              const std::string& details = "");
48
53     void logSystemError(const std::string& context);
54 };
55
56 #endif // LOGGER_H
```

4.13 Файл src/main.cpp

Главная функция сервера

#include "Config.h"
#include "Server.h"
#include <iostream>
#include <csignal>
#include <cstdlib>
Граф включаемых заголовочных файлов для main.cpp:

Функции

- void signalHandler (int signal)
  Обработчик сигнала Ctrl+C.
- int main (int argc, char **argv)

Переменные

- Server * serverInstance = nullptr

4.13.1 Подробное описание

Главная функция сервера

Автор

Судариков А.В.

Версия

1.0

Дата

2025

4.13.2 Функции

4.13.2.1 main()

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

Граф вызовов:

4.13.2.2 signalHandler()

```
void signalHandler (
            int signal )
```

Обработчик сигнала Ctrl+C.

Аргументы

signal	Номер сигнала

Граф вызовов: Граф вызова функции:

4.13.3 Переменные

4.13.3.1 serverInstance

Server* serverInstance = nullptr

4.14 Файл src/Server.cpp

#include "Server.h"
#include <iostream>
#include <cstring>
#include <cerrno>
#include <vector>
#include <arpa/inet.h>
Граф включаемых заголовочных файлов для Server.cpp:

Функции

- static uint32_t le32_to_host (uint32_t value)
  Конвертировать из little-endian в хостовый порядок
- static uint32_t host_to_le32 (uint32_t value)
  Конвертировать хостовый порядок в little-endian.
- static int32_t le32_to_host_int (int32_t value)
- static int32_t host_to_le32_int (int32_t value)

4.14.1 Функции

4.14.1.1 host_to_le32()

static uint32_t host_to_le32 (
uint32_t value) [static]

Конвертировать хостовый порядок в little-endian.

Граф вызова функции:

4.14.1.2 host_to_le32_int()

```
static int32_t host_to_le32_int (
                int32_t value )   [static]
```

Граф вызовов: Граф вызова функции:

4.14.1.3 le32_to_host()

```
static uint32_t le32_to_host (
                uint32_t value )   [static]
```

Конвертировать из little-endian в хостовый порядок

Граф вызова функции:

4.14.1.4 le32_to_host_int()

```
static int32_t le32_to_host_int (
                int32_t value )   [static]
```

Граф вызовов: Граф вызова функции:

4.15 Файл src/Server.h

Главный класс сервера

```
#include "Config.h"
#include "Database.h"
#include "Logger.h"
#include "Authenticator.h"
#include "VectorProcessor.h"
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <memory>
```

Граф включаемых заголовочных файлов для Server.h: Граф файлов, в которые включается этот файл:

Классы

- class Server

  Главный класс сервера

4.15.1 Подробное описание

Главный класс сервера

Автор

Судариков А.В.

Версия

1.0

Дата

2025

4.16 Server.h

См. документацию.

```
1
9 #ifndef SERVER_H
10 #define SERVER_H
11
12 #include "Config.h"
13 #include "Database.h"
14 #include "Logger.h"
15 #include "Authenticator.h"
16 #include "VectorProcessor.h"
17 #include <sys/socket.h>
18 #include <netinet/in.h>
19 #include <arpa/inet.h>
20 #include <unistd.h>
21 #include <memory>
22
26 class Server {
27 private:
28     Config config_;
29     Database database_;
30     Logger logger_;
31     int serverSocket_;
32     bool running_;
33
34 public:
39     Server(const Config& config);
40
44     ~Server();
45
50     bool start();
51
55     void stop();
56
61     bool isRunning()const { return running_; }
62
63 private:
68     bool initializeNetwork();
69
73     void mainLoop();
74
79     void handleClient(int clientSocket);
80
87     bool authenticateClient(int clientSocket, std::string& clientLogin);
88
93     void processVectorData(int clientSocket);
94
101     bool sendString(int socket, const std::string& str);
102
110     bool recvString(int socket, std::string& str, size_t maxLength = 1024);
111
119     bool sendAll(int socket, const void* data, size_t size);
120
128     bool recvAll(int socket, void* data, size_t size);
129
134     void closeConnection(int socket);
135 };
136
137 #endif // SERVER_H
```

4.17 Файл src/VectorProcessor.cpp

#include "VectorProcessor.h"
#include <iostream>
Граф включаемых заголовочных файлов для VectorProcessor.cpp:

4.18 Файл src/VectorProcessor.h

Обработка векторных данных

#include <cstdint>
#include <vector>
#include <climits>
Граф включаемых заголовочных файлов для VectorProcessor.h: Граф файлов, в которые включается этот файл:

Классы

- class VectorProcessor

  Класс обработки векторов

4.18.1 Подробное описание

Обработка векторных данных

Автор

Сударикov А.В.

Версия

1.0

Дата

2025

4.19 VectorProcessor.h

См. документацию.

```
1
9 #ifndef VECTORPROCESSOR_H
10 #define VECTORPROCESSOR_H
11
12 #include <cstdint>
13 #include <vector>
14 #include <climits>
15
19 class VectorProcessor {
20 public:
26     static int32_t calculateSum(const std::vector<int32_t>& vector);
27
33     static std::vector<int32_t> processVectors(
34         const std::vector<std::vector<int32_t»& vectors);
35
36 private:
37     static const int32_t MAX_INT32 = INT_MAX;      // 2^31-1
38     static const int32_t MIN_INT32 = INT_MIN;      // -2^31
39
46     static bool willOverflowAdd(int32_t a, int32_t b);
47 };
48
49 #endif // VECTORPROCESSOR_H
```

4.20 Файл tests/TestAuthenticator.cpp

Модульные тесты для класса Authenticator.

```
#include <UnitTest++/UnitTest++.h>
#include "../src/Authenticator.h"
#include <iostream>
#include <string>
#include <cstring>
#include <algorithm>
```

Граф включаемых заголовочных файлов для TestAuthenticator.cpp:

Функции

- TEST (Authenticator_GenerateSalt_Format)
- TEST (Authenticator_GenerateSalt_Unique)
- TEST (Authenticator_IsValidHexString_Correct64)
- TEST (Authenticator_IsValidHexString_Correct16)
- TEST (Authenticator_IsValidHexString_Lowercase)
- TEST (Authenticator_IsValidHexString_WrongLength)
- TEST (Authenticator_IsValidHexString_InvalidChars)
- TEST (Authenticator_IsValidHexString_Empty)
- TEST (Authenticator_CalculateSHA256_ValidInput)
- TEST (Authenticator_CalculateSHA256_Deterministic)
- TEST (Authenticator_CalculateSHA256_EmptyPassword)
- TEST (Authenticator_CalculateSHA256_LongPassword)
- TEST (Authenticator_VerifyHash_Correct)
- TEST (Authenticator_VerifyHash_WrongPassword)
- TEST (Authenticator_VerifyHash_WrongSalt)
- TEST (Authenticator_VerifyHash_InvalidHashFormat)
- TEST (Authenticator_VerifyHash_CaseInsensitive)
- TEST (Authenticator_VerifyHash_DifferentSaltsProduceDifferentHashes)
- TEST (Authenticator_FullAuthCycle)
- TEST (Authenticator_SpecialCharactersInPassword)
- int main ()

  Основная функция

4.20.1 Подробное описание

Модульные тесты для класса Authenticator.

Автор

Судариков А.В.

Версия

1.0

Дата

2025

4.20.2 Функции

4.20.2.1 main()

int main ( )

Основная функция

4.20.2.2 TEST() [1/20]

TEST (
Authenticator_CalculateSHA256_Deterministic)

Граф вызовов:

4.20.2.3 TEST() [2/20]

TEST (
Authenticator_CalculateSHA256_EmptyPassword)

Граф вызовов:

4.20.2.4 TEST() [3/20]

TEST (
Authenticator_CalculateSHA256_LongPassword)

Граф вызовов:

4.20.2.5 TEST() [4/20]

TEST (
Authenticator_CalculateSHA256_ValidInput)

Граф вызовов:

4.20.2.6 TEST() [5/20]

TEST (
Authenticator_FullAuthCycle)

Граф вызовов:

4.20.2.7 TEST() [6/20]

```
TEST (
        Authenticator_GenerateSalt_Format )
```

Граф вызовов:

4.20.2.8 TEST() [7/20]

```
TEST (
        Authenticator_GenerateSalt_Unique )
```

Граф вызовов:

4.20.2.9 TEST() [8/20]

```
TEST (
        Authenticator_IsValidHexString_Correct16 )
```

Граф вызовов:

4.20.2.10 TEST() [9/20]

```
TEST (
        Authenticator_IsValidHexString_Correct64 )
```

Граф вызовов:

4.20.2.11 TEST() [10/20]

```
TEST (
        Authenticator_IsValidHexString_Empty )
```

Граф вызовов:

4.20.2.12 TEST() [11/20]

```
TEST (
        Authenticator_IsValidHexString_InvalidChars )
```

Граф вызовов:

4.20.2.13 TEST() [12/20]

```
TEST (
        Authenticator_IsValidHexString_Lowercase )
```

Граф вызовов:

4.20.2.14 TEST() [13/20]

```
TEST (
        Authenticator_IsValidHexString_WrongLength )
```

Граф вызовов:

4.20.2.15 TEST() [14/20]

```
TEST (
        Authenticator_SpecialCharactersInPassword )
```

Граф вызовов:

4.20.2.16 TEST() [15/20]

```
TEST (
        Authenticator_VerifyHash_CaseInsensitive )
```

Граф вызовов:

4.20.2.17 TEST() [16/20]

```
TEST (
        Authenticator_VerifyHash_Correct )
```

Граф вызовов:

4.20.2.18 TEST() [17/20]

```
TEST (
        Authenticator_VerifyHash_DifferentSaltsProduceDifferentHashes )
```

Граф вызовов:

4.20.2.19 TEST() [18/20]

```
TEST (
        Authenticator_VerifyHash_InvalidHashFormat )
```

Граф вызовов:

4.20.2.20 TEST() [19/20]

```
TEST (
        Authenticator_VerifyHash_WrongPassword )
```

Граф вызовов:

4.20.2.21 TEST() [20/20]

```
TEST (
            Authenticator_VerifyHash_WrongSalt  )
```

Граф вызовов:

4.21 Файл tests/TestConfig.cpp

Модульные тесты для класса Config.

```
#include <UnitTest++/UnitTest++.h>
#include "../src/Config.h"
#include <iostream>
#include <vector>
#include <string>
#include <cstring>
```

Граф включаемых заголовочных файлов для TestConfig.cpp:

Функции

- void resetGetopt ()
- TEST (Config_Constructor_Defaults)
- TEST (Config_ParseCommandLine_ShortOptions)
- TEST (Config_ParseCommandLine_LongOptions)
- TEST (Config_ParseCommandLine_MixedOptions)
- TEST (Config_ParseCommandLine_Help)
- TEST (Config_ParseCommandLine_InvalidPortZero)
- TEST (Config_ParseCommandLine_InvalidPortText)
- TEST (Config_ParseCommandLine_NoOptions)
- TEST (Config_ParseCommandLine_UnknownOption)
- TEST (Config_ParseCommandLine_Partial)
- TEST (Config_SetDefaults)
- TEST (Config_ParseCommandLine_BoundaryPort)
- TEST (Config_ParseCommandLine_BoundaryPortMax)
- TEST (Config_ShowHelp)
- int main ()

  Основная функция

4.21.1 Подробное описание

Модульные тесты для класса Config.

Автор

Судариков А.В.

Версия

1.0

Дата

2025

4.21.2 Функции

4.21.2.1 main()

int main ( )

Основная функция

4.21.2.2 resetGetopt()

void resetGetopt ( )

Граф вызова функции:

4.21.2.3 TEST() [1/14]

TEST (
Config_Constructor_Defaults)

Граф вызовов:

4.21.2.4 TEST() [2/14]

TEST (
Config_ParseCommandLine_BoundaryPort)

Граф вызовов:

4.21.2.5 TEST() [3/14]

TEST (
Config_ParseCommandLine_BoundaryPortMax)

Граф вызовов:

4.21.2.6 TEST() [4/14]

TEST (
Config_ParseCommandLine_Help)

Граф вызовов:

4.21.2.7 TEST() [5/14]

```
TEST (
        Config_ParseCommandLine_InvalidPortText )
```

Граф вызовов:

4.21.2.8 TEST() [6/14]

```
TEST (
        Config_ParseCommandLine_InvalidPortZero )
```

Граф вызовов:

4.21.2.9 TEST() [7/14]

```
TEST (
        Config_ParseCommandLine_LongOptions )
```

Граф вызовов:

4.21.2.10 TEST() [8/14]

```
TEST (
        Config_ParseCommandLine_MixedOptions )
```

Граф вызовов:

4.21.2.11 TEST() [9/14]

```
TEST (
        Config_ParseCommandLine_NoOptions )
```

Граф вызовов:

4.21.2.12 TEST() [10/14]

```
TEST (
        Config_ParseCommandLine_Partial )
```

Граф вызовов:

4.21.2.13 TEST() [11/14]

```
TEST (
        Config_ParseCommandLine_ShortOptions )
```

Граф вызовов:

4.21.2.14 TEST() [12/14]

```
TEST (
        Config_ParseCommandLine_UnknownOption )
```

Граф вызовов:

4.21.2.15 TEST() [13/14]

```
TEST (
        Config_SetDefaults )
```

Граф вызовов:

4.21.2.16 TEST() [14/14]

```
TEST (
        Config_ShowHelp )
```

Граф вызовов:

4.22 Файл tests/TestDatabase.cpp

Модульные тесты для класса Database.

```
#include <UnitTest++/UnitTest++.h>
#include "../src/Database.h"
#include <iostream>
#include <fstream>
#include <cstdio>
#include <string>
```

Граф включаемых заголовочных файлов для TestDatabase.cpp:

Функции

- TEST (Database_LoadFromFile_ValidFile)
- TEST (Database_LoadFromFile_WithComments)
- TEST (Database_LoadFromFile_InvalidLines)
- TEST (Database_UserExists)
- TEST (Database_GetPassword)
- TEST (Database_LoadFromFile_EmptyFile)
- TEST (Database_LoadFromFile_NonexistentFile)
- TEST (Database_LoadFromFile_Reload)
- TEST (Database_LoadFromFile_TrimSpaces)
- TEST (Database_CaseSensitive)
- int main ()

  Основная функция

4.22.1 Подробное описание

Модульные тесты для класса Database.

Автор

Судариков А.В.

Версия

1.0

Дата

2025

4.22.2 Функции

4.22.2.1 main()

```
int main ( )
```

Основная функция

4.22.2.2 TEST() [1/10]

```
TEST (
        Database_CaseSensitive )
```

Граф вызовов:

4.22.2.3 TEST() [2/10]

```
TEST (
        Database_GetPassword )
```

Граф вызовов:

4.22.2.4 TEST() [3/10]

```
TEST (
        Database_LoadFromFile_EmptyFile )
```

Граф вызовов:

4.22.2.5 TEST() [4/10]

```
TEST (
        Database_LoadFromFile_InvalidLines )
```

Граф вызовов:

4.22.2.6 TEST() [5/10]

```
TEST (
        Database_LoadFromFile_NonexistentFile )
```

Граф вызовов:

4.22.2.7 TEST() [6/10]

```
TEST (
        Database_LoadFromFile_Reload )
```

Граф вызовов:

4.22.2.8 TEST() [7/10]

```
TEST (
        Database_LoadFromFile_TrimSpaces )
```

Граф вызовов:

4.22.2.9 TEST() [8/10]

```
TEST (
        Database_LoadFromFile_ValidFile )
```

Граф вызовов:

4.22.2.10 TEST() [9/10]

```
TEST (
        Database_LoadFromFile_WithComments )
```

Граф вызовов:

4.22.2.11 TEST() [10/10]

```
TEST (
        Database_UserExists )
```

Граф вызовов:

4.23 Файл tests/TestLogger.cpp

Модульные тесты для класса Logger.

```
#include <UnitTest++/UnitTest++.h>
#include "../src/Logger.h"
#include <iostream>
#include <fstream>
#include <cstdio>
#include <cstring>
#include <string>
```

Граф включаемых заголовочных файлов для TestLogger.cpp:

Функции

- TEST (Logger_Constructor_ValidFile)
- TEST (Logger_Log_DifferentLevels)
- TEST (Logger_Log_Format)
- TEST (Logger_Log_WithDetails)
- TEST (Logger_LogSystemError)
- TEST (Logger_Log_EmptyDetails)
- TEST (Logger_Log_CriticalToStderr)
- TEST (Logger_ConsoleOutput)
- TEST (Logger_Destructor)
- TEST (Logger_InvalidFile)
- int main ()

  Основная функция

4.23.1 Подробное описание

Модульные тесты для класса Logger.

Автор

Судариков А.В.

Версия

1.0

Дата

2025

4.23.2 Функции

4.23.2.1 main()

int main ( )

Основная функция

4.23.2.2 TEST() [1/10]

TEST (
Logger_ConsoleOutput)

Граф вызовов:

4.23.2.3 TEST() [2/10]

TEST (
Logger_Constructor_ValidFile)

Граф вызовов:

4.23.2.4 TEST() [3/10]

TEST (
Logger_Destructor)

Граф вызовов:

4.23.2.5 TEST() [4/10]

TEST (
Logger_InvalidFile)

Граф вызовов:

4.23.2.6 TEST() [5/10]

TEST (
Logger_Log_CriticalToStderr)

Граф вызовов:

4.23.2.7 TEST() [6/10]

TEST (
Logger_Log_DifferentLevels)

Граф вызовов:

4.23.2.8 TEST() [7/10]

```
TEST (
            Logger_Log_EmptyDetails )
```

Граф вызовов:

4.23.2.9 TEST() [8/10]

```
TEST (
            Logger_Log_Format )
```

Граф вызовов:

4.23.2.10 TEST() [9/10]

```
TEST (
            Logger_Log_WithDetails )
```

Граф вызовов:

4.23.2.11 TEST() [10/10]

```
TEST (
            Logger_LogSystemError )
```

Граф вызовов:

4.24 Файл tests/TestVectorProcessor.cpp

Модульные тесты для класса VectorProcessor.

```
#include "/usr/include/UnitTest++/UnitTest++.h"
#include "../src/VectorProcessor.h"
#include <iostream>
#include <vector>
#include <climits>
#include <cstdint>
```

Граф включаемых заголовочных файлов для TestVectorProcessor.cpp:

Функции

- TEST (CalculateSum_EmptyVector)
- TEST (CalculateSum_SingleElement)
- TEST (CalculateSum_MultipleElements)
- TEST (CalculateSum_NegativeNumbers)
- TEST (CalculateSum_MixedSigns)
- TEST (CalculateSum_MaxInt32)
- TEST (CalculateSum_MinInt32)
- TEST (CalculateSum_MaxPlusZero)
- TEST (CalculateSum_NearMaxNoOverflow)
- TEST (CalculateSum_OverflowPositive)
- TEST (CalculateSum_OverflowNegative)
- TEST (CalculateSum_CourseExample)
- TEST (CalculateSum_ZeroVector)
- int main ()

4.24.1 Подробное описание

Модульные тесты для класса VectorProcessor.

Автор

Судариков А.В.

Версия

1.0

Дата

2025

4.24.2 Функции

4.24.2.1 main()

```
int main ( )
```

4.24.2.2 TEST() [1/13]

```
TEST (
            CalculateSum_CourseExample  )
```

Граф вызовов:

4.24.2.3 TEST() [2/13]

```
TEST (
            CalculateSum_EmptyVector  )
```

Граф вызовов:

4.24.2.4 TEST() [3/13]

```
TEST (
            CalculateSum_MaxInt32  )
```

Граф вызовов:

4.24.2.5 TEST() [4/13]

```
TEST (
        CalculateSum_MaxPlusZero )
```

Граф вызовов:

4.24.2.6 TEST() [5/13]

```
TEST (
        CalculateSum_MinInt32 )
```

Граф вызовов:

4.24.2.7 TEST() [6/13]

```
TEST (
        CalculateSum_MixedSigns )
```

Граф вызовов:

4.24.2.8 TEST() [7/13]

```
TEST (
        CalculateSum_MultipleElements )
```

Граф вызовов:

4.24.2.9 TEST() [8/13]

```
TEST (
        CalculateSum_NearMaxNoOverflow )
```

Граф вызовов:

4.24.2.10 TEST() [9/13]

```
TEST (
        CalculateSum_NegativeNumbers )
```

Граф вызовов:

4.24.2.11 TEST() [10/13]

```
TEST (
        CalculateSum_OverflowNegative )
```

Граф вызовов:

4.24.2.12 TEST() [11/13]

```
TEST (
        CalculateSum_OverflowPositive )
```

Граф вызовов:

4.24.2.13 TEST() [12/13]

```
TEST (
        CalculateSum_SingleElement )
```

Граф вызовов:

4.24.2.14 TEST() [13/13]

```
TEST (
        CalculateSum_ZeroVector )
```

Граф вызовов:

Предметный указатель