For blockchain.hpp and blockchain.cpp

|  |  |
| --- | --- |
| **void** Blockchain::adding\_new\_block ( std::vector <transaction> &tr) | Добавление нового блока в цепочку |
| std::ifstream input(**"electronic\_signature.txt"**); | Открытие файла на чтение |
| input.close(); | Закрытие файла |
| **bool** Blockchain::checking\_signature () | Проверка валидности цифровой подписи |
| **bool** Blockchain::valid\_chain() | Проверка валидности цепочки |
| Block Blockchain::get\_latest\_block() | Получение последнего блока из цепочки |
| **const char** \* Blockchain::chain\_to\_file () | Запись цепочки в файл |

For block.hpp and block.cpp

|  |  |
| --- | --- |
| std::string Block::calc\_hash() | Создание хэша из данных |
| std::string Block::get\_hash\_of\_trans(**const** std::vector < transaction > &tr) | Получение хэша транзакций |
| std::string Block::get\_hash\_of\_trans () | Получение хэша транзакций |
| Block Block::create\_gen\_blok() | Создание генезисного блока в цепочке |
| **int** Block::get\_index() | Возращение индекса блока |
| std::string Block::get\_time() | Возращение времени блока |
| std::string Block::get\_hash() | Возращение хэша блока |
| hugeint Block::get\_address() | Возращение адреса(подписи) блока |
| std::string Block::get\_prev\_hash() | Возращение хэша предыдущего блока |
| hugeint Block::get\_part\_adr1()  hugeint Block::get\_part\_adr2()  hugeint Block::get\_part\_adr3() | Возращение частей подписи блока |
| std::vector <transaction> Block::get\_trans() | Возращение вектора транзакций блока |

For trusted\_comp.hpp and trusted\_comp.hpp

|  |  |
| --- | --- |
| cryptkey trusted\_node::get\_privkey() | ВОзвращение приватного ключа подписи |
| cryptkey trusted\_node::get\_publkey () | ВОзвращение публичного ключа подписи |
| hugeint trusted\_node::from\_file(**const char** \*out1) | ВОзвращение числа из файла |
| **void** trusted\_node::write\_to\_file (**const char** \*in1 , **const** std::string &str\_h) | Запись даных в файл |

For rsa.hpp ans rsa.hpp

|  |  |
| --- | --- |
| **void static** genKeys(cryptkey &pbkey, cryptkey &prkey); | Генерация публичнгого и закрытого ключа |
| **void static** encryptTxtFile(**const char** \*in, **const char** \*out, cryptkey &key); | Зашифровка данных и запись в файл |
| std::string **static** decrypt (hugeint &data, cryptkey &key); | Расшифровка подписи |

For main.cpp

|  |  |
| --- | --- |
| std::vector <transaction> from\_file\_to\_struct\_trans (**const char** \* filename\_serv ) | Из данных файла в нужный тип |
| **int** listening = socket(**AF\_INET**, **SOCK\_STREAM**, 0 ); | Создание сокета с доменом ,типом и протоколом |
| inet\_pton(**AF\_INET**, ip\_adr.c\_str(), &address.sin\_addr); | Данная функция преобразует строку символов ipAddress в структуру сетевого адреса сетевого семейства адресов af, а затем копирует полученную структуру по адресу |
| **int** a = bind(listening,(sockaddr\*)&address,**sizeof**(address)); | Связываем созданный сокет и айпи и порт |
| listen(listening, **SOMAXCONN**); | Создание очереди для соединений |
| Int nsockfd = accept(listening, (sockaddr \*) &addr\_remote, &sin\_size) | Настройка сокета на принятие данных |
| FILE \*fr = fopen(filename, **"a"**); | Функция возвращает указатель на открытый файл |
| bzero(revbuf, 512); | Функция заполняет пустыми байтами refbuf |
| Int fr\_block\_sz = recv(nsockfd, revbuf, 512, 0) | Чтение данных, отправленных клиентом |
| **int** write\_sz = fwrite(revbuf, **sizeof**(**char**), fr\_block\_sz, fr); | Запись в файл принятых данных сокетом |

Sha256(std::string ) - получение хэша из данных