Правительство Российской Федерации ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ» (НИУ ВШЭ)

Московский институт электроники и математики им. А. Н. Тихонова

ОТЧЕТ

О ПРАКТИЧЕСКОЙ РАБОТЕ № 6

по дисциплине «Криптографические методы защиты информации»

Программа алгоритма консенсуса блокчейн «Доказательство выполнения работы»

(Proof of Work, PoW)

|  |
| --- |
| Студент гр. БИБ222  Малыгин Данила Дмитриевич  «02» июня 2024 г. |
| Старший преподаватель  Доктор технических наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. Г. Шапошников  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г. |

Москва 2024

**СОДЕРЖАНИЕ**

[1. Цель работы 3](#_Toc169202438)

[2. Задание 4](#_Toc169202439)

[3. Краткая теоретическая часть 5](#_Toc169202440)

[3.1. Общие сведения о блокчейн 5](#_Toc169202441)

[3.2. Хэш-функции 6](#_Toc169202442)

[3.3. Алгоритм «дерево Меркла» подсчёта хэш-функции в блокчейн биткоин 6](#_Toc169202443)

[3.4. Алгоритмы консенсуса публичных блокчейн 7](#_Toc169202444)

[3.5. Общие сведения о блокчейн криптовалют 8](#_Toc169202445)

[3.6. Состав блока блокчейн биткоин 8](#_Toc169202446)

[3.7. Алгоритм создания блоков в сети криптовалюты Bitcoin (майнинга) 9](#_Toc169202447)

[3.8. Атака двойной траты (двойного расходования) или атака 51% 10](#_Toc169202448)

[4. Хэш-функция SHA-256 11](#_Toc169202449)

[4.1. Достоинства 12](#_Toc169202450)

[4.2. Недостатки 12](#_Toc169202451)

[7. Описание программной реализации 14](#_Toc169202452)

[8. Пример работы программы 18](#_Toc169202453)

[9. Выводы о проделанной работе 20](#_Toc169202454)

[10. Список использованных источников 21](#_Toc169202455)

[11. Ссылка на репозиторий с программой 22](#_Toc169202456)

# Цель работы

Целью данной работы является приобретение навыков программной реализации алгоритма консенсуса PoW, аналогичного алгоритму PoW в Bitcoin (майнинга).

# Задание

1. Написать программную реализацию алгоритмов хэш-функции SHA-256;
2. Для четырёх транзакций (файлов размера 226 байт каждый) подсчитать хэш корня Меркла с помощью алгоритма SHA-256;

3. Создать заголовок блока, состоящий из следующих полей: размер блока (4 байта); хэш заголовка предыдущего блока (берётся случайное 256 битное число); хэш корня Меркла транзакций (256 бит); метка времени (4 байта); nonce (32 бита);

4. Последовательно изменяя значение nonce, найти хэш заголовка блока с четырьмя первыми нулями.

5. Подготовить отчет о выполнении работы.

Программа должна обладать следующей функциональностью:

1. Принимать на вход файлы, содержащие транзакции;

2. Принимать на вход файл, содержащий хэш заголовка предыдущего блока;

3. Осуществлять вычисление корня Меркла для транзакций;

4. Осуществлять вычисления хэш заголовка до получения требуемого результата, последовательно изменяя nonce;

5. Сохранять созданный блок в файле.

Отчет должен содержать следующие составные части:

1. Раздел с заданием;

2. Раздел с краткой теоретической частью;

3. Раздел с описанием программной реализации с учетом особенностей выбранной среды разработки и языка программирования;

4. Раздел с результатами работы программы;

5. Раздел с выводами о проделанной работе.

# Краткая теоретическая часть

**3.1. Общие сведения о блокчейн**

Блокчейн (blockchain) — это распределённая база данных, защищённая от изменений, где устройства хранения не подключены к общему серверу. Блокчейн представляет собой цепочку блоков информации, каждый из которых содержит транзакции и ссылку на предыдущий блок. Благодаря хэш-суммам, блокчейн защищён от изменений.

Концепцию цепочек блоков предложил Сатоши Накамото в 2008 году и впервые реализовал в 2009 году как часть биткоина. Однако идеи, подобные блокчейну, обсуждались ещё с 1982 года в работах криптографа Дэвида Чаума и других учёных.

Основные характеристики блокчейна:

– Пиринговый: все участники сети равноправны, без центрального контроля.

– Распределённый реестр: каждый участник имеет полную копию базы данных.

– Только добавление: блоки можно только добавлять, изменение данных практически невозможно.

– Обновляемый по соглашению: обновления происходят через консенсус участников.

Существует три основных типа блокчейна:

1. Публичный: доступен всем, любой может участвовать в его работе.
2. Приватный: доступ ограничен, только по приглашению администраторов.
3. Консорциум: управляется группой равноправных узлов, члены консорциума совместно валидируют транзакции.

Блокчейн используется для передачи информации без участия третьей стороны, что делает его важным для финансовых транзакций, прав собственности, умных контрактов и других сфер. Он также применяется в банковском секторе, земельных реестрах, медицине и других областях.

Основные требования к криптографическим хэш-функциям включают:

1. Необратимость – невозможно вычислить исходный блок данных по заданному значению хэш-функции.
2. Стойкость к коллизиям первого рода – невозможно подобрать другое сообщение с тем же хэш-значением.
3. Стойкость к коллизиям второго рода – невозможно найти пару сообщений с одинаковым хэш-значением.

**3.2. Хэш-функции**

Безопасность блокчейна основывается на криптографических хэш-функциях. Хэширование преобразует входные данные произвольной длины в фиксированную битовую строку. Хэш-функция называется криптографической, если она необратима и устойчиво к нахождению коллизий — сообщений с одинаковыми хэш-значениями.

Хэш-функции в блокчейне защищают данные от изменений. Сложность нахождения коллизий называется стойкостью хэш-функции. Наиболее известный метод нахождения коллизий — атака "дней рождения", где вероятность успеха поиска коллизии связана с числом возможных хэш-значений.

В блокчейне биткоина используется алгоритм SHA-256, который генерирует 256-битные хэши. Алгоритм обрабатывает данные блоками по 512 бит через 64 цикла, используя 32-битные слова и константы. В результате обработки всех блоков получается итоговый 256-битный хэш.

Для корректной работы исходное сообщение дополняется до длины, кратной 512 битам, добавлением единицы, нулей и длины сообщения в последние 64 бита. Начальное значение хэша заполняется константами.

Изображение выглядит как текст, диаграмма, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1 - Схема работы SHA-256

**3.3. Алгоритм «дерево Меркла» подсчёта хэш-функции в блокчейн биткоин**

Подсчет хэша блока в блокчейне обычно выполняется с помощью алгоритма «дерево Меркла». Это двоичное дерево, где листовые вершины содержат хэши данных, а внутренние вершины — хэши дочерних вершин. Корневой узел дерева содержит хэш всего набора данных.

Дерево Меркла упрощает вычисление хэша блока при добавлении новой транзакции и позволяет эффективно проверять подлинность транзакций. Если количество транзакций нечетное, последняя транзакция дублируется.

Использование хэш-дерева позволяет быстро доказать принадлежность блока данных к множеству, проверяя только один блок, без пересчета хэшей остальных узлов дерева.

Изображение выглядит как снимок экрана, Прямоугольник, прямоугольный, клавиатура

Автоматически созданное описание

Рисунок 2 - Дерево Меркла

**3.4. Алгоритмы консенсуса публичных блокчейн**

Консенсус — это процесс согласования данных в блокчейне между узлами. Алгоритм консенсуса должен обеспечивать согласие всех «честных» узлов и работать при наличии неисправных или вредоносных узлов. Он также должен защищать от атак, таких как атака Сивиллы, где один пользователь создает множество ложных узлов для захвата сети.

В блокчейне существуют два основных алгоритма консенсуса: Proof of Work (PoW) и Proof of Stake (PoS).

Proof of Work (PoW) – участники (майнеры) решают сложные вычислительные задачи, чтобы предложить блок для сети. Первый, кто решает задачу, получает вознаграждение. Пример: алгоритм HashCash в биткоине. Сложность задач регулируется для поддержания средней скорости создания блоков.

Proof of Stake (PoS) – участники (валидаторы) показывают, что у них есть доля в системе (например, фиксированное количество криптовалюты). Валидаторы выбираются случайным образом для создания и подтверждения блоков. Пример: Ethereum перешел на PoS в 2022 году, сократив потребление энергии на 99%. Валидаторы должны заморозить 32 ETH для участия.

PoS не требует майнеров и специального оборудования, что делает его более энергоэффективным. Стейкинг (заморозка криптовалюты) используется для обеспечения участия в консенсусе PoS.

**3.5. Общие сведения о блокчейн криптовалют**

Криптовалюта — это цифровая валюта, работающая через компьютерную сеть без центрального органа, такого как правительство или банк. Монеты (например, Bitcoin, BTC) хранятся в блокчейне как транзакции, подписанные секретным ключом. Транзакции включают открытый ключ для проверки подписи и адреса отправителя и получателя. Адреса, часто представляющие собой хэшированный открытый ключ, могут изменяться для повышения анонимности.

Баланс пользователя проверяется через программное обеспечение, используя закрытые ключи, которые хранятся в кошельке. Кошелек — это программа (например, Bitcoin Core), позволяющая пользователю отправлять и получать монеты.

Сеть криптовалюты является пиринговой (P2P), узлы обмениваются информацией о транзакциях и блоках. Полные узлы хранят блокчейн и проверяют новые блоки, а легковесные узлы выполняют только функции кошелька и сетевой маршрутизации.

Биткоин, первая криптовалюта, был выпущен в 2009 году. К июню 2023 года существовало более 25 000 криптовалют, из которых более 40 имели рыночную капитализацию свыше 1 миллиарда долларов. Среди них крупнейшие — Bitcoin и Ethereum.

**3.6. Состав блока блокчейн биткоин**

Блок в блокчейне биткоина состоит из следующих полей: размер блока (4 байта), заголовок блока (80 байт), счётчик транзакций (переменной длины) и сами транзакции (переменной длины). Размер блока не должен превышать 1 МБ.

Транзакции представляют собой криптографически защищенный обмен активами между участниками сети. Первая транзакция в любом блоке — coinbase транзакция — создаётся майнером для получения вознаграждения за найденный блок.

Заголовок блока включает: версию блока (4 байта), хэш предыдущего блока (32 байта), хэш корня Меркла (32 байта), метку времени (4 байта), текущую цель сложности сети (4 байта) и nonce (4 байта).

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 3 - Цепочка блоков

Цель сложности сети — это число, меньше которого должен быть хэш заголовка. Если 32-битного nonce недостаточно для нахождения нужного хэша, используется extraNonce в транзакции coinbase. При этом пересчитывается только одна "ветка" дерева Меркла.

**3.7. Алгоритм создания блоков в сети криптовалюты Bitcoin (майнинга)**

После подписания владельцем биткойнов транзакция передается в сеть, где каждый узел проверяет подпись и историю транзакций, чтобы подтвердить наличие средств. После подтверждения транзакция попадает в мемпул (локальный пул неподтвержденных транзакций). Майнеры выбирают транзакции из мемпула для формирования блока.

Майнеры получают вознаграждение за создание блока, включающее субсидию и комиссии за транзакции. Субсидия — это фиксированное количество новых биткойнов за каждый добавленный блок, которое уменьшается вдвое каждые 210,000 блоков (процесс халвинга). Комиссии за транзакции добавляются майнерами в блок, причем приоритет отдается транзакциям с более высокими комиссиями.

Сеть биткойн использует алгоритм Proof of Work (PoW), при котором майнеры решают сложные вычислительные задачи, чтобы найти хэш, удовлетворяющий текущей цели сложности. Если 32-битного nonce недостаточно, используется поле extraNonce, что требует пересчета корня дерева Меркла.

По мере роста хэшрейта сети, сложность задач автоматически регулируется. Первоначальная субсидия за блок составляла 50 BTC, и она уменьшалась в 2012, 2016 и 2020 годах. Ожидается, что последний биткойн будет добыт около 2140 года. После этого майнеры будут получать вознаграждение только за счет комиссий за транзакции.

**3.8. Атака двойной траты (двойного расходования) или атака 51%**

Злоумышленник-майнер создает скрытую от сети побочную цепочку блокчейна и тратит свои биткойны в основной цепочке, не включая эти транзакции в изолированную версию. Затем он пытается добавлять блоки в свою цепочку быстрее, чем другие майнеры в основной цепочке. Когда его цепочка становится длиннее, он транслирует ее в сеть, заставляя всех переключиться на подменную версию, возвращая себе потраченные биткойны.

Для успешной атаки злоумышленнику нужны мощности, превышающие мощность всей сети, что практически невозможно. Однако такие атаки возможны при ошибках в протоколе блокчейна.

# Хэш-функция SHA-256

SHA-256 (Secure Hash Algorithm 256-bit) — это криптографическая хэш-функция, разработанная Национальным институтом стандартов и технологий США (NIST), и является частью семейства SHA-2. Она широко используется для обеспечения целостности и безопасности данных в различных областях, включая блокчейн-технологии.

Принципы работы:

1. Входные данные:

– Принимает данные произвольной длины.

– Обрабатывает данные по блокам размером 512 бит.

1. Подготовка данных:

– Дополняет данные до длины, кратной 512 битам.

– К исходным данным добавляется бит "1", за которым следует набор битов "0".

– В последние 64 бита добавляется длина исходного сообщения.

1. Инициализация:

– Инициализация восьми 32-битных слов начальных значений, которые используются для хранения промежуточных хэш-значений.

1. Основной процесс:

– Сообщение разбивается на блоки по 512 бит.

– Каждый блок обрабатывается функцией сжатия, состоящей из 64 раундов, в которых используются побитовые операции, сдвиги и логические функции.

– В каждом раунде используются разные константы и изменяемые данные, основанные на текущем состоянии хэша и обрабатываемом блоке.

1. Итерации:

– Промежуточные хэш-значения обновляются после обработки каждого блока.

– Итоговый хэш получается после обработки всех блоков сообщения.

1. Выходные данные:

– Результатом является 256-битный (32-байтовый) хэш, уникальный для каждого набора входных данных.

Связь с блокчейном:

– В блокчейне биткоина SHA-256 используется для создания хэшей транзакций и блоков, что обеспечивает целостность данных.

– Хэш каждого блока содержит хэш предыдущего блока, создавая непрерывную цепочку.

– Майнеры решают задачи поиска хэша, который соответствует определенным условиям (целевой сложности).

– Для нахождения подходящего хэша майнеры изменяют параметр nonce и вычисляют новый хэш блока.

– Процесс требует значительных вычислительных ресурсов, обеспечивая безопасность сети.

* 1. **Достоинства**

– Хэш-функция SHA-256 является односторонней, что означает, что из хэша невозможно восстановить исходные данные. Это обеспечивает высокий уровень безопасности данных.

– Вероятность того, что две разные последовательности данных дадут одинаковый хэш (коллизия), крайне мала. Это делает SHA-256 надежным для проверки целостности данных.

– SHA-256 разработан с учетом защиты от различных криптографических атак, таких как атаки на коллизии и атаки на прообраз.

– Алгоритм SHA-256 достаточно быстрый и может обрабатываться эффективно как программными, так и аппаратными средствами.

– Благодаря своей структуре, SHA-256 может быть оптимизирован для выполнения на различных платформах и устройствах.

– SHA-256 является частью криптографического стандарта SHA-2 и широко используется в различных индустриях, включая блокчейн, цифровую подпись, SSL/TLS и другие криптографические протоколы.

– Использование SHA-256 гарантирует совместимость с большинством современных систем безопасности и криптографических приложений.

* 1. **Недостатки**

– В контексте блокчейна, особенно в случае биткоина, процесс майнинга с использованием SHA-256 требует значительных вычислительных ресурсов и, следовательно, большой затраты энергии. Это приводит к экологическим и экономическим проблемам.

– Размер хэша в 256 бит может быть избыточным для некоторых приложений, где требуется меньший размер хэша. Это увеличивает объем данных, которые нужно хранить и передавать.

– С развитием квантовых компьютеров алгоритмы, основанные на традиционной криптографии, такие как SHA-256, могут стать уязвимыми к квантовым атакам. Хотя на данный момент это не является непосредственной угрозой, будущее развитие квантовых технологий может потребовать новых хэш-функций, устойчивых к квантовым атакам.

– Хотя SHA-256 эффективен на современных платформах, его производительность может значительно варьироваться в зависимости от аппаратного обеспечения. Некоторые устройства могут испытывать трудности с выполнением вычислений на приемлемой скорости.

# Описание программной реализации

Программа написана на языке Python с использованием стандартных библиотек argparse, os, struct и time. Я использовал текстовый редактор Visual Studio Code и Python версии 3.10 для разработки программы. Программа реализует алгоритм блокчейн-майнинга, включая вычисление корня дерева Меркла и создание заголовка блока.

Программа позволяет пользователю указать список файлов транзакций и файл с хэшем предыдущего блока через аргументы командной строки. После выполнения майнинга, программа сохраняет заголовок блока в файл mined\_block.bin и выводит сообщение о завершении операции.

Подробное описание работы программы

Программа состоит из нескольких файлов, каждый из которых выполняет свои задачи:

– main.py — Основная программа.

– block\_header.py — Определение заголовка блока.

– merkle\_root.py — Вычисление корня дерева Меркла.

– sha256.py — Реализация SHA-256 хэширования.

– utils.py — Вспомогательные функции для чтения транзакций.

1. **Main.py:**

Основная программа отвечает за координацию всех шагов, необходимых для майнинга нового блока.

Шаги программы:

– Чтение транзакций из указанных файлов.

– Чтение хэша предыдущего блока из файла.

– Вычисление корня дерева Меркла для транзакций.

– Создание заголовка блока.

– Поиск подходящего nonce (майнинг блока).

– Сохранение заголовка блока в файл.

Использование библиотек:

– **argparse** – Используется для обработки аргументов командной строки. Позволяет пользователю указать файлы транзакций и файл с хэшем предыдущего блока при запуске программы.

– **time** – Используется для получения текущего времени (временной метки), которое добавляется в заголовок блока.

– **os** – Используется для проверки существования файла с хэшем предыдущего блока и для чтения этого файла.

1. **Block\_header.py:**

Этот файл определяет класс BlockHeader, который представляет заголовок блока.

– \_\_init\_\_ – инициализация заголовка блока.

– serialize – сериализация заголовка блока в байтовый формат.

– hash – вычисление SHA-256 хэша заголовка блока.

– mine – поиск подходящего nonce, чтобы хэш блока начинался с '0000'.

1. **Merkle\_root.py:**

Этот файл определяет функцию для вычисления корня дерева Меркла.

– merkle\_root – рекурсивная функция для вычисления корня дерева Меркла. Объединяет пары хэшей и вычисляет их хэш до тех пор, пока не останется один хэш. Используется для вычисления корня дерева Меркла на основе списка транзакций. Корень дерева Меркла позволяет эффективно проверять целостность и аутентичность всех транзакций в блоке. В блокчейне корень дерева Меркла включается в заголовок блока, обеспечивая надежный механизм для проверки целостности данных без необходимости повторной обработки всех транзакций.

1. **Sha256.py:**

Этот файл реализует функцию SHA-256 хэширования.

А) Импорт библиотеки **struct** – используется для упаковки и распаковки данных в бинарном формате.

Б) Константы SHA-256 – K — массив из 64 констант, используемых в алгоритме SHA-256. Эти константы представляют собой первые 32 бита дробных частей кубических корней первых 64 простых чисел.

В) Функция right\_rotate – выполняет правое вращение (циклический сдвиг) 32-битного числа на заданное количество бит.

*Параметры*:

– value – 32-битное число, которое нужно сдвинуть.

– count – количество бит для сдвига.

Возвращаемое значение: Результат циклического сдвига.

Г) Функция sha256 – реализует алгоритм SHA-256 для хэширования данных.

*Параметры*:

– data – входные данные для хэширования.

Возвращаемое значение: Хэш данных в виде шестнадцатеричной строки.

– Основная функция для генерации SHA-256 хеша.

Этапы работы функции sha256:

1. Начальные значения хэша:

– H — массив из 8 начальных значений хэша. Эти значения представляют собой первые 32 бита дробных частей квадратных корней первых 8 простых чисел.

1. Предобработка:

– Входные данные преобразуются в байтовый массив (bytearray).

– Длина исходных данных в битах сохраняется в переменной orig\_len\_in\_bits.

– К данным добавляется бит '1' (0x80 в шестнадцатеричном формате).

– К данным добавляются нулевые биты, пока их длина не станет кратной 512 минус 64 (т. е. 448 бит).

– В конце добавляется исходная длина данных в битах в виде 64-битного числа (8 байт).

1. Основной цикл:

– Данные обрабатываются в блоках по 512 бит (64 байта).

– Каждый блок разбивается на 16 32-битных слов (всего 64 слова), и каждое слово преобразуется в целое число.

– Затем для каждого блока вычисляются дополнительные 48 слов с использованием функции right\_rotate.

1. Инициализация рабочих переменных:

– Рабочие переменные (a, b, c, d, e, f, g, h) инициализируются значениями массива H.

1. Основной цикл компрессии:

– Для каждого из 64 раундов выполняются математические операции (включая побитовые операции, сложение по модулю 2^32 и функции сдвига), чтобы обновить рабочие переменные.

– В конце каждого раунда рабочие переменные обновляются новыми значениями.

1. Обновление значений хэша:

– Значения рабочего массива (H) обновляются с использованием текущих рабочих переменных.

1. Возвращение результата:

– Возвращается конечный хэш данных в виде шестнадцатеричной строки.

1. **Utils:**

Cсодержит функцию для чтения транзакций из файлов – **read\_transactions** используется в основной программе для автоматического чтения транзакций из нескольких файлов. Это позволяет пользователю просто указать список файлов, а программа автоматически обработает их содержимое, считывая транзакции для дальнейшей обработки.

Эта функция упрощает процесс работы с транзакциями, так как пользователю не нужно вручную открывать и читать каждый файл. Вместо этого, достаточно передать список файлов функции read\_transactions, и она вернет все транзакции в виде единого списка.

1. **Папка files:**

Содержит файлы транзакций, а также файл с хэшем предыдущего блока.

1. **Шаги для запуска программы через терминал:**
2. Откройте терминал (Command Prompt, PowerShell, Terminal на MacOS или Linux).
3. Используйте команду cd, чтобы перейти в директорию, где находятся ваши файлы.
4. Используйте следующую команду для запуска программы:

python main.py --transactions *transaction1* *transaction2* *transaction3* *transaction4* --prev-block-hash *prev\_block\_hash*

--transactions – список файлов с транзакциями. Здесь вы указываете все файлы, которые содержат транзакции для текущего блока.

--prev-block-hash – файл с хэшем предыдущего блока.

1. Содержимое файлов:

– prev\_block\_hash – например:

3d2e0a9e7f3b2a4a8e6f20b9678b423f1f2b4b7e3b5d6f7898e1b5c3d1e4a3c2

– transaction1, transaction2, transaction3, transaction4: любые файлы по 226 байт.

1. После успешного запуска программы вы должны увидеть сообщение о завершении майнинга, включая хэш нового блока и найденное значение nonce. Также будет создан файл mined\_block.bin, содержащий заголовок нового блока.

# Пример работы программы

Для запускаем программы, используем упомянутую ранее команду. Для этого я создал 4 файла с транзакциями (по 226 байт), а также файл содержащий хэш предыдущего блока.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 4 – Запуск программы

Результат выполнения программы. После успешного запуска программы мы видим сообщение о завершении майнинга, включая хэш нового блока и найденное значение **nonce**. Также был создан файл mined\_block.bin, содержащий заголовок нового блока.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 5 - Результат выполнения программы

Кроме того, программа отлично справляется и с другими файлами по 226 байт, например, **pdf** и **jpg**.

Команда: python main.py --transactions ../files/transaction1.pdf ../files/transaction2.pdf ../files/transaction3.pdf ../files/transaction4.pdf --prev-block-hash ../files/prev\_block\_hash.txt

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 6 - Результат выполнения программы с pdf

Команда: python main.py --transactions ../files/transaction1.jpg ../files/transaction2.jpg ../files/transaction3.jpg ../files/transaction4.jpg --prev-block-hash ../files/prev\_block\_hash.txt

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 7 - Результат выполнения программы с jpg

# Выводы о проделанной работе

В результате выполнения практической работы была создана программа, реализующая основные аспекты алгоритма консенсуса Proof of Work (PoW), аналогичного используемому в блокчейн сети Bitcoin. Программа включает в себя создание хэшей транзакций, вычисление корня дерева Меркла, создание заголовка блока и майнинг блока с заданным условием сложности.

Таким образом, цели работы были достигнуты, и необходимые навыки программной реализации алгоритма консенсуса PoW были успешно приобретены.

# Список использованных источников

1. Как это работает: Деревья Меркла в биткойн сети – URL: [https://habr.com/ru/companies/bitfury/articles/346398/](https://en.wikipedia.org/wiki/Modular_exponentiation).

2. Что такое SHA-256 – URL: [https://crypto.ru/algoritm-heshirovaniya-sha-256/](https://en.wikipedia.org/wiki/Modular_exponentiation).

3. Block Headers – <https://developer.bitcoin.org/reference/block_chain.html>.

# Ссылка на репозиторий с программой

https://github.com/fedyarays/blockchain