**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Российский химико-технологический университет имени Д.И. Менделеева»**

**Кафедра информационных компьютерных технологий**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9**

Выполнил студент группы КС-30 Лобачев Дмитрий Сергеевич

Ссылка на репозиторий: https://github.com/MUCTR-IKT-CPP/DSLobachev\_30/blob/main/Algorithms/Laba9/Laba9.cpp

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: 24.04.2023

**Оглавление**

[Описание задачи 3](#_Toc133180614)

[Описание метода/модели 4](#_Toc133180615)

[Выполнение задачи. 5](#_Toc133180616)

[Результат работы 8](#_Toc133180617)

[Заключение 10](#_Toc133180618)

# Описание задачи

В рамках лабораторной работы необходимо реализовать алгоритм SHA2.

Для реализованной хеш функции провести следующие тесты:

1. Провести генерацию 1000 пар строк длинной 128 символов, отличающихся друг от друга 1,2,4,8,16 символов и сравнить хеши для пар между собой, проведя поиск одинаковых последовательностей символов в хешах и подсчитав максимальную длину такой последовательности. Результаты для каждого количества отличий нанести на график, где по оси х кол-во отличий, а по оси y максимальная длина одинаковой последовательности.
2. Провести N = 10^i(i от 2 до 6) генерацию хешей для случайно сгенерированных строк длиной 256 символов, и выполнить поиск одинаковых хешей в итоговом наборе данных, результаты привести в таблице где первая колонка это N генераций, а вторая таблица наличие и кол-во одинаковых хешей, если такие были.
3. Провести по 1000 генераций хеша для строк длинной n (64, 128, 256, 512, 1024, 2048, 4096, 8192)(строки генерировать случайно для каждой серии), подсчитать среднее время и построить зависимость скорости расчета хеша от размера входных данных

# Описание метода/модели

SHA-2 (англ. Secure Hash Algorithm Version 2 — безопасный алгоритм хеширования, версия 2) — семейство криптографических алгоритмов — однонаправленных хеш-функций, включающее в себя алгоритмы SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256 и SHA-512/224.

Хеш-функции предназначены для создания «отпечатков» или «дайджестов» для сообщений произвольной длины. Применяются в различных приложениях или компонентах, связанных с защитой информации.

Хеш-функции SHA-2 разработаны Агентством национальной безопасности США и опубликованы Национальным институтом стандартов и технологий в федеральном стандарте обработки информации FIPS PUB 180-2 в августе 2002 года.

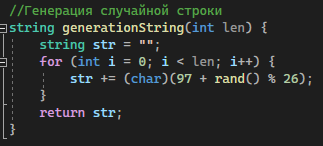
Хеш-функции семейства SHA-2 построены на основе структуры Меркла — Дамгора.

Исходное сообщение после дополнения разбивается на блоки, каждый блок — на 16 слов. Алгоритм пропускает каждый блок сообщения через цикл с 64 или 80 итерациями (раундами). На каждой итерации 2 слова преобразуются, функцию преобразования задают остальные слова. Результаты обработки каждого блока складываются, сумма является значением хеш-функции. Тем не менее, инициализация внутреннего состояния производится результатом обработки предыдущего блока. Поэтому независимо обрабатывать блоки и складывать результаты нельзя.

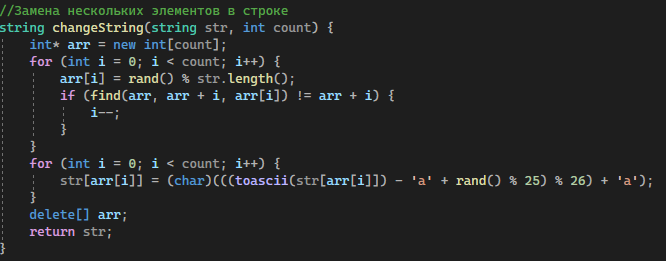
# Выполнение задачи.

Для реализации данного метода сортировки использовался язык программирования C++ и алгоритм SHA-256.

1. Функция генерации случайной строки generationString()

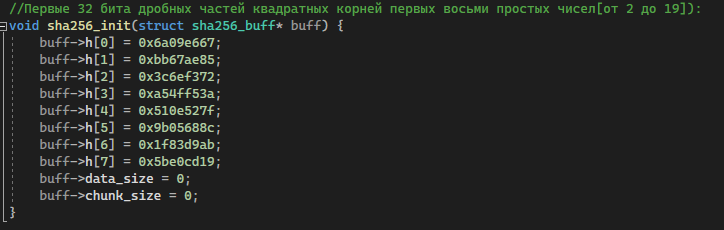


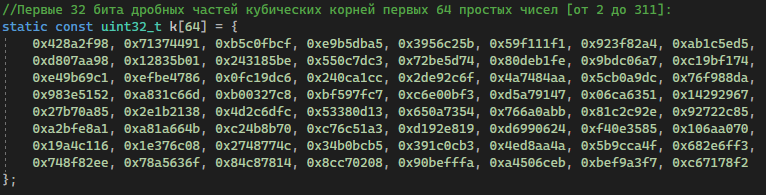
1. Функция изменения элементов в строке changeString()



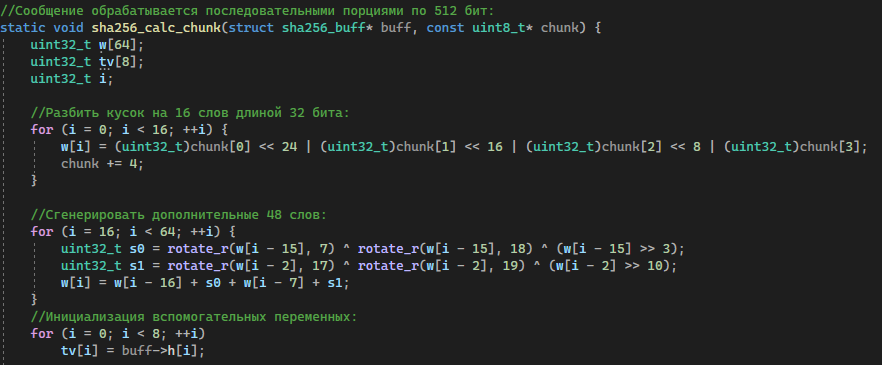
**Хэш-функция**

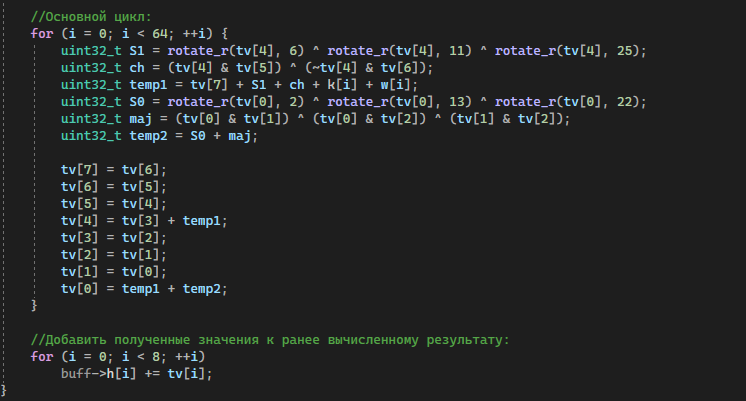
1. Функция переустановки начальных значений хэша:



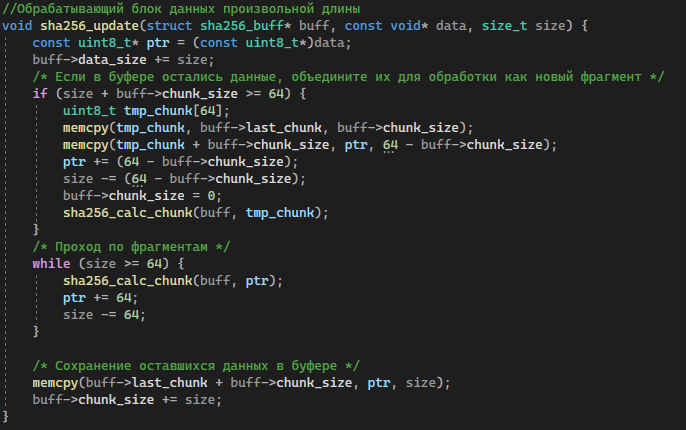


1. Функция хэширования:

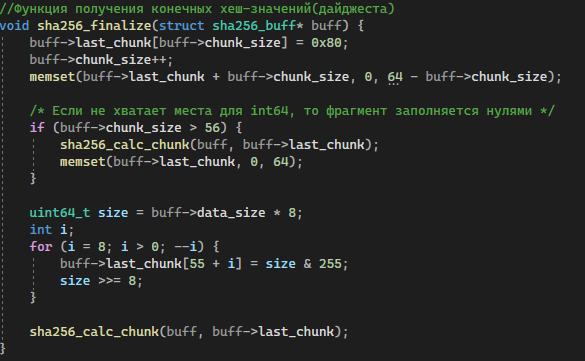




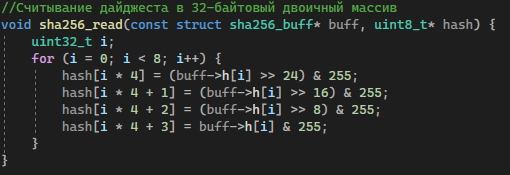
1. Обрабатывающий блок:



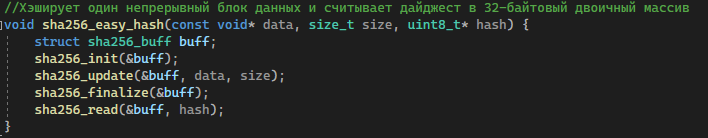
1. Функция получения конечных хеш-значений:



1. Функция считывания дайджеста в 32-байтовый массив



1. Функция хеширования:



1. Функция rotate\_r



# Результат работы

Поиск максимальной последовательности одинаковых символов:

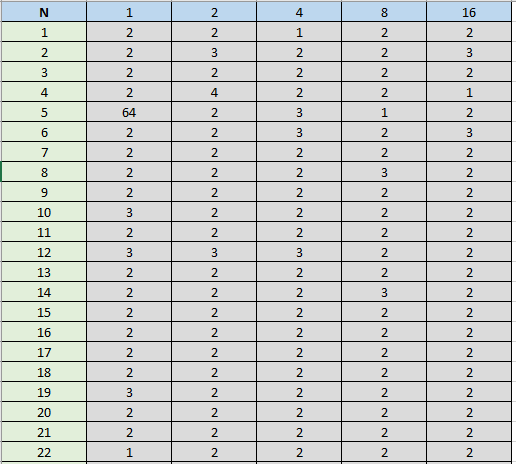
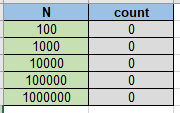


График последовательностей повторений:

График среднего значения максимальной последовательности:

Коллизии:



Среднее время хэширования:



График среднего времени хэширования:

# Заключение

По результатам работы можно сделать вывод, что хэширование данным методом происходит достаточно быстро. В процессе хэширования несколько раз проявлялась коллизия, однако количество таких случаев достаточно низко и возникает только на сообщениях малой длинны. Такое малое количество коллизий объясняется тем, что при хешировании практически одинаковых строк возникает лавинный эффект, из-за которого хеш значения похожих строк будут кардинально отличаться.

Проанализировав графики можно сделать вывод, что с увеличением длинны строки, увеличивается и время хеширования.