МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

**«Исследование криптографических хеш-функций»**

Студент:

Септилко Анастасия Антоновна

Преподаватель:

Блинова Евгения Александровна

Минск 2020

**ЗАДАНИЯ:**

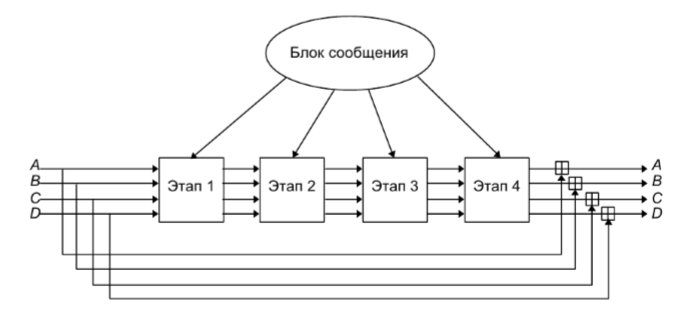
1. Разработать оконное приложение, реализующее один из алгоритмов хеширования из указанного преподавателем семейства (MD или SHA; или иного). При этом можно воспользоваться доступными готовыми библиотеками. Язык программирования – на свой выбор. Приложение должно обрабатывать входные сообщения, длина которых определяется спецификацией на реализуемый алгоритм.

2. Оценить быстродействие выбранного алгоритма хеширования.

3. Результаты оформить в виде отчета по установленным правилам.

Я реализовывала в лабораторной работе алгоритм для хеширования MD-5. Для начала немного теории, чтобы понимать суть алгоритма:

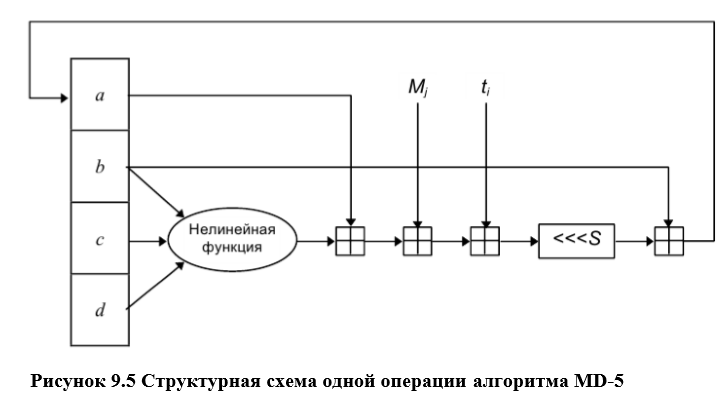
На рис. 9.4 приведена укрупненная структурная схема алгоритма MD-5, на рис. 9.5 – структурная схема одной операции (над одним 32разрадным подблоком).



Здесь знак «+» в квадратной фигуре соответствует операции сложения по модулю 2^32. Вспомним (см. ЛР №5):

A + B (mod 2^32) =

{A + B, если A + B<2^32 A + B – 2^32, если A + B≥ 2^32 }.



Главный модуль (рис. 9.4) состоит из четырех похожих этапов (у MD-4 было только три этапа). На каждом этапе 16 раз используются различные операции. Каждая операция представляет собой нелинейную функцию над тремя из a, b, c и d. Затем она добавляет этот результат к четвертой переменной, подблоку текста Мj и константе ti. Далее результат циклически сдвигается вправо на переменное число s бит и добавляет результат к одной из переменных a, b, c и d. Наконец результат заменяет одну из этих переменных (рис. 9.5).

Результатом хеширования, h, является конкатенация последних значений указанных переменных, т. е. 32\*4 = 128 бит.

На рис. 9.6 показана схема выполнения одной операции в алгоритме SHA-1. Цикл состоит из четырех этапов по 20 операций в каждом (в MD5 – 4 этапа по 16 операций в каждом). Каждая операция представляет собой нелинейную функцию над тремя из 5: a, b, c, d, e. Сдвиг и сложение – аналогично MD5.

В алгоритме используются следующие четыре константы:

Kt = 0x5a827999, при t = 0, …, 19,

Kt = 0x6ed9eba1 , при t = 20, …, 39,

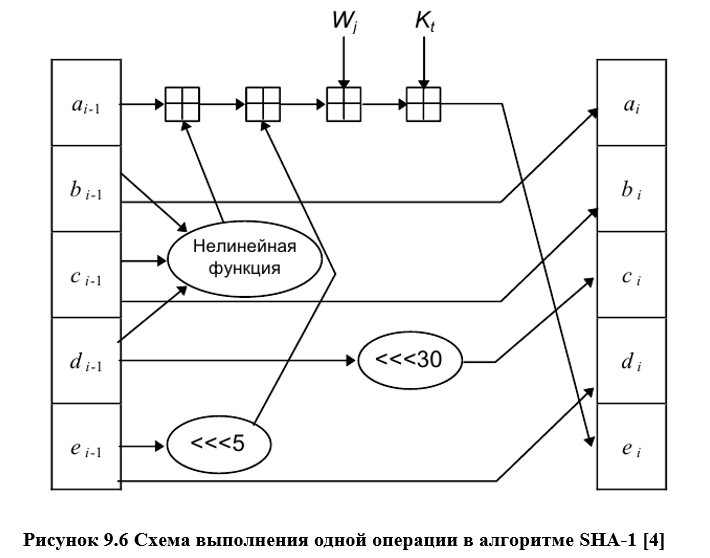
Kt = 0x8flbbcdc, при t = 40, …, 59,

Kt = 0xca62c1d6, при t = 60, …, 79

Блок сообщения трансформируется из 16-ти 32-битных слов (от M0 по M15) в 80-т 32-битных слов (W0, …, W79) с помощью следующего алгоритма:

Wt = Mt , при t = 0, …, 15,

Wt = (Wt-3 ⊕ Wt-8 ⊕ Wt-14 ⊕ Wt-16) <<< 1, при t = 16, …, 79



После обработки всех 512-битных блоков выходом является 160 битный дайджест сообщения в виде конкатенации последних значений переменных a, b, c, d, e.

Как мы заметили, в алгоритмах MD-5 SHA-1 результат текущего действия прибавляется к результату предыдущего. Это направлено на усиление лавинного эффекта. Этой же цели служит то обстоятельство, что значения циклического сдвига влево на каждом этапе были приближенно оптимизированы: четыре сдвига, используемые на каждом этапе, отличаются от значений, используемых на других этапах.

Как отмечает Б. Шнайер, SHA – это MD-4 с добавлением расширяющего преобразования, дополнительного этапа и с улучшенным лавинным эффектом. MD-5 – это MD-4 с улучшенным битовым хешированием, дополнительным этапом и улучшенным лавинным эффектом.

Первая стадия хеширования: расширение строки, чтобы недоставало 64 бита, чтобы длина была кратна 512 это показано на рисунке 1.1.

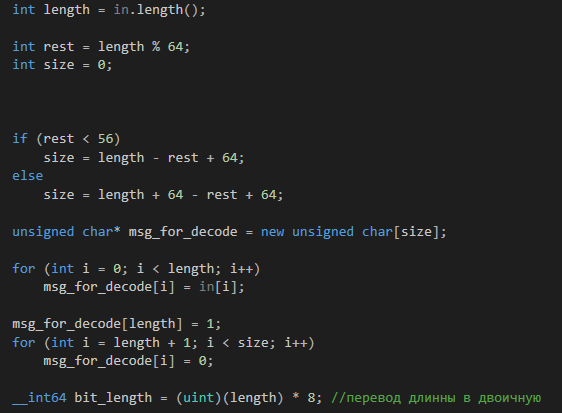


Рисунок 1.1 – Первая стадия

Вторая стадия: Разбивка расширенного сообщения на блоки. Продемонстрировано в листинге 1.1.

Листинг 1.1 – Стадия номер 2

for (int i = 0; i < 8; i++)

msg\_for\_decode[size - 8 + i] = (unsigned char)(bit\_length >> i \* 8);

Стадия третья: инициализация констант. Продемонстрировано в листинге 1.2.

Листинг 1.2 – Стадия номер 3

uint A = 0x67452301, B = 0xefcdab89, C = 0x98badcfe, D = 0x10325476;

uint \* X = (uint\*)(msg\_for\_decode);

uint T[64];

for (int i = 0; i < 64; i++)

T[i] = uint(pow(2, 32) \* fabs(sin(i + 1)));

Стадия четвертая: Обработка сообщений 512-битными блоками. Для нашего алгоритма хеширование необходимо 4 раунда. После этого необходимо произвести сумму.

Стадия пятая: Вывод результата.

На рисунке 1.2 продемонстрирована работа приложения:

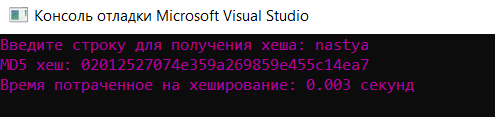


Рисунок 1.2 – Итог выполнения программы