## Практическая работа №5 - Хеш - функция

**Цель работы:**

Исследование структуры алгоритмов хеширования: *MD5, MD6, CRC, HAVAL, SHA-2, ГОСТ Р 34.11-94*.

**Задание:**

- реализовать одну из существующих функций хеширования: *MD5, MD6, CRC, HAVAL, SHA-2, ГОСТ Р 34.11-94*;

- реализовать алгоритм работы функции хеширования на любом языке программирования.

**Методика выполнения Практической работы**

Студентами рассматриваются одна из функций хеширования. Изучается алгоритм работы выбранной функции и после чего, реализуется на каком либо языке программирования.

**Дополнительная информация к выполнению Практической работы**

Хеширование применяется для сравнения данных: если у двух массивов хеш-функции разные, массивы гарантированно различаются; если одинаковые — массивы, скорее всего, одинаковы. В общем случае однозначного соответствия между исходными данными и хеш-кодом нет в силу того, что количество значений хеш-функций меньше чем вариантов входного массива; существует множество массивов, дающих одинаковые хеш-коды — так называемые [коллизии](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D1%8F_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8). Вероятность возникновения коллизий играет немаловажную роль в оценке качества хеш-функций.

Существует множество алгоритмов хеширования с различными характеристиками ([разрядность](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D1%8F%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D1%8C), [вычислительная сложность](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C), [криптостойкость](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D1%82%D0%BE%D0%B9%D0%BA%D0%BE%D1%81%D1%82%D1%8C) и т.п.). Выбор той или иной хеш-функции определяется спецификой решаемой задачи. Простейшими примерами хеш-функций могут служить [контрольная сумма](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%83%D0%BC%D0%BC%D0%B0) или [CRC](http://ru.wikipedia.org/wiki/CRC). Более детальное рассмотрение требует алгоритм хеширования MD5.

MD5 - 128-битный алгоритм [хеширования](http://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), разработанный профессором [Рональдом Л. Ривестом](http://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D0%B2%D0%B5%D1%81%D1%82,_%D0%A0%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%B4) из [Массачусетского технологического института](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B0%D1%87%D1%83%D1%81%D0%B5%D1%82%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82) в [1991 году](http://ru.wikipedia.org/wiki/1991_%D0%B3%D0%BE%D0%B4). Предназначен для создания «отпечатков» или «дайджестов» сообщений произвольной длины. Является улучшенной в плане безопасности версией [MD4](http://ru.wikipedia.org/wiki/MD4). Зная MD5, невозможно восстановить входное сообщение, так как одному MD5 могут соответствовать разные сообщения. Используется для проверки подлинности опубликованных сообщений путём сравнения дайджеста сообщения с опубликованным.[5]

На вход алгоритма поступает входной поток данных, хеш которого необходимо найти. Длина сообщения может быть любой (в том числе нулевой). L – длина сообщения. Это число целое и неотрицательное. Кратность каким-либо числам необязательна. После поступления данных идёт процесс подготовки потока к вычислениям. На рис. 5 Изображена схема работы алгоритма MD5.

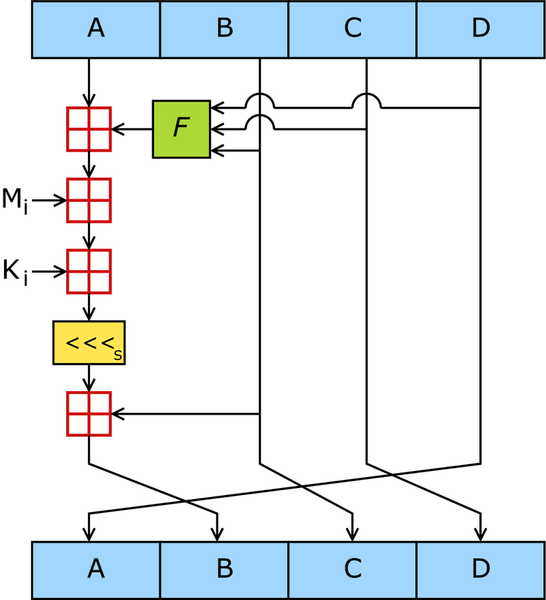


Рис. 5. Схема алгоритма MD5

Шаг 0. Выравнивание потока.

Входные данные выравниваются так, чтобы их размер был сравним с 448 по модулю 512 (L’ = 512 × N + 448). Сначала дописывают единичный бит в конец потока, затем необходимое число нулевых бит.

Шаг 1. Добавление длины сообщения.

В оставшиеся 64 бита дописывают 64-битное представление длины данных (количество бит в сообщении) до выравнивания. Если длина превосходит 264 − 1, то дописывают только младшие биты. После этого длина потока станет кратной 512. Вычисления будут основываться на представлении этого потока данных в виде массива слов по 512 бит.

Шаг 2. Инициализация буфера.

Для вычислений инициализируются 4 переменных сцепления размером по 32 бита и задаются начальные значения шестнадцатеричными числами.

uint A = 0x01234567;

uint B = 0x89ABCDEF;

uint C = 0xFEDCBA98;

uint D = 0x76543210;

В этих переменных будут храниться результаты промежуточных вычислений. Начальное состояние ABCD называется инициализирующим вектором.[5]

Далее определяются функции и константы, которые понадобятся для вычислений.

Требуется 4 нелинейных функции для четырёх операций. Вводятся дополнительные функции от трёх параметров - слов, результатом также будет слово.

private uint funF(uint X, uint Y, uint Z)

{

return (X & Y) | ((~X) & Z);

}

private uint funG(uint X, uint Y, uint Z)

{

return (X & Z) | (Y & (~Z));

}

private uint funH(uint X, uint Y, uint Z)

{

return X ^ Y ^ Z;

}

private uint funI(uint X, uint Y, uint Z)

{

return Y ^ (X | (~Z));

}

Эти функции спроектированы так, чтобы, если соответствующие биты X,Y,Z независимы и несмещены, каждый бит результата также был бы независимым и несмещенным.

Определим таблицу констант T[1..64] - 64-элементная таблица данных, построенная следующим образом: *T*[*i*] = *int*(4294967296 \* | *sin*(*i*) | ), где 4294967296 = 232.

Выровненные данные разбиваются на блоки (слова) по 32 бита, и каждый блок проходит 4 раунда из 16 операторов. Все операторы однотипны и имеют вид: [abcd k s i], определяемый как *a* = *b* + ((*a* + *Fun*(*b*,*c*,*d*) + *X*[*k*] + *T*[*i*]) < < < *s*), где X — блок данных. X[k] = M [n \* 16 + k], где k - номер 32-битного слова из n-го 512-битного блока сообщения, и s - циклический сдвиг влево на s бит полученного 32-битного аргумента.

Шаг 3. Вычисление в цикле.

Заносим в блок данных элемент n из массива. Сохраняются значения A, B, C и D, оставшиеся после операций над предыдущими блоками (или их начальные значения, если блок первый).

AA = A

BB = B

CC = C

DD = D

*Этап 1*

//[abcd k s i] a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s).

[ABCD 0 7 1][DABC 1 12 2][CDAB 2 17 3][BCDA 3 22 4]

[ABCD 4 7 5][DABC 5 12 6][CDAB 6 17 7][BCDA 7 22 8]

[ABCD 8 7 9][DABC 9 12 10][CDAB 10 17 11][BCDA 11 22 12]

[ABCD 12 7 13][DABC 13 12 14][CDAB 14 17 15][BCDA 15 22 16]

*Этап 2*

//[abcd k s i] a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s).

[ABCD 1 5 17][DABC 6 9 18][CDAB 11 14 19][BCDA 0 20 20]

[ABCD 5 5 21][DABC 10 9 22][CDAB 15 14 23][BCDA 4 20 24]

[ABCD 9 5 25][DABC 14 9 26][CDAB 3 14 27][BCDA 8 20 28]

[ABCD 13 5 29][DABC 2 9 30][CDAB 7 14 31][BCDA 12 20 32]

*Этап 3*

//[abcd k s i] a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s).

[ABCD 5 4 33][DABC 8 11 34][CDAB 11 16 35][BCDA 14 23 36]

[ABCD 1 4 37][DABC 4 11 38][CDAB 7 16 39][BCDA 10 23 40]

[ABCD 13 4 41][DABC 0 11 42][CDAB 3 16 43][BCDA 6 23 44]

[ABCD 9 4 45][DABC 12 11 46][CDAB 15 16 47][BCDA 2 23 48]

*Этап 4*

//[abcd k s i] a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s).

[ABCD 0 6 49][DABC 7 10 50][CDAB 14 15 51][BCDA 5 21 52]

[ABCD 12 6 53][DABC 3 10 54][CDAB 10 15 55][BCDA 1 21 56]

[ABCD 8 6 57][DABC 15 10 58][CDAB 6 15 59][BCDA 13 21 60]

[ABCD 4 6 61][DABC 11 10 62][CDAB 2 15 63][BCDA 9 21 64]

Суммируем с результатом предыдущего цикла:

A = AA + A

B = BB + B

C = CC + C

D = DD + D

После окончания цикла необходимо проверить, есть ли ещё блоки для вычислений. Если да, то изменяем номер элемента массива (n++) и переходим в начало цикла.

Шаг 4. Результат вычислений.

Результат вычислений находится в буфере ABCD, это и есть хеш. Если вывести слова в обратном порядке DCBA, то получится MD5 хеш.

На рис. 6 представлен результат работы хеш-функции основанной на алгоритме MD5. Хеш код строки «Хеш-функция» равен шестнадцатеричному значению «cb29836cc3e488123dcfbb54adbb9222».

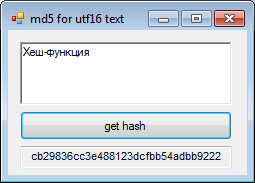


Рис. 6. Форма хеш – функция MD5

**Требования к предъявляемой работе:**

- Программа шифрования должны быть наделена понятным и удобным пользовательским интерфейсом.

- Полная реализация выбранной функции хеширования.