# Язык Hash Query

Этот язык представляет собой простой декларативный LINQ (Language Integrated Queries) или SQL подобный язык. Целью создания языка было желание предоставить единый, унифицированный интерфейс для работы с криптографическими хэшами строк и файлов с некоторыми дополнительными плюшками вроде восстановления строк по хэшам, поиск файлов по хэшам и пр.

## Основы синтаксиса

Все задания на работу с хэшами в языке представляют собой предложения разделенные точкой с запятой «**;**». Они могут передаваться компилятору или через командную строку (опция **-q**) или быть записанными в простом текстовом файле, который потом скармливается компилятору (опция **-f**). В файле может быть произвольное количество запросов, но не более 10 тыс. (по соображениям производительности).

Пробельные символы в запросах (переводы строк, табуляторы и пр.) не учитываются, за исключением тех, что находятся собственно в данных (строках в кавычках или апострофах).

### Общий вид запросов

Все запросы имеют вид:

**for** … **do** …;

После for и do идут инструкции специфичные для каждой из категорий запроса (см. далее).

Точка с запятой в конце запроса обязательна, даже в случае единичного запроса.

В инструкциях после **for**, но до **do** могут встречаться подвыражения **let** и **where** не являющиеся обязательными.

### Подвыражение let

Подвыражение **let** не является обязательным и может быть опущено. Также, могут быть опущены отдельные части выражения.

Подвыражение **let** содержит список инструкций с разделителем запятой, порядок инструкций не имеет значения. Инструкция имеет следующий вид:

<переменная>.<имя атрибута> = <значение атрибута>

Переменная — это любая последовательность латинских букв и цифр, но начинаться она должна **обязательно** с буквы. Переменная должна быть задано до **let**.

Имя атрибута является одним из зарезервированных имен.

Значением атрибута является либо строка (в апострофах или кавычках), либо целое число.

### Подвыражение where

Подвыражение **where** используется для задания условий, например для фильтрации файлов по некоторым признакам.

**where** cостоит из выражений вида:

<переменная>.<имя атрибута> <условный оператор> <значение>

Соединенных операторами конъюнкции (логическое И) — **and** и дизъюнкции (логическое ИЛИ) — **or**.

Переменная — это любая последовательность латинских букв и цифр, но начинаться она должна **обязательно** с буквы. Переменная должна быть задано до **where**.

Условный оператор — это один из следующий операторов:

* **==** (равно)
* **!=** (не равно)
* **>** (больше)
* **<** (меньше)
* **>=** (больше или равно)
* **<=** (меньше или равно)
* **~** (соответствует регулярному выражению)
* **!~** (не соответствует регулярному выражению)

Для группировки частей выражения (изменения порядка выполнения условных операторов) могут быть использованы круглые скобки.

### Категории запросов

Все запросы можно разделить на 4 категории:

1. Запросы на вычисление хэша строки
2. Запросы на восстановление строки по её хэшу
3. Запросы на вычисление хэша отдельных файлов
4. Запросы на вычисление хэша файлов в заданном каталоге

#### Запросы на вычисление хэша строки

Запросы имеют следующий вид:

**for** **string** '<строка>' **do** <хэш>;

Строка может быть как в апострофах (одиночных кавычках), так и в обычных кавычках (двойных кавычках). В запросах передаваемых через командную строку должны использоваться апострофы (понятно почему), в запросах из файлов можно использовать любые кавычки. Хэш может принимать одно из следующих значений (регистр имеет значение):

* md5
* md4
* sha1
* sha256
* sha384
* sha512
* whirlpool
* crc32

#### Запросы на восстановление строки по её хэшу

Общий вид запроса такой:

**for string** <переменная> **from hash** '<значение хэша>' [**let** <переменная>.dict = '0123456789', <переменная>.min = 2, <переменная>.max = 6] **do crack** <хэш>**;**

Здесь хэш является одним из поддерживаемых типов хэшей (см. предыдущий раздел), значение хэша.

Подвыражение **let** используется для управления параметрами восстановления строки, а именно:

1. словарь (атрибут **dict**)
2. минимальная длина (атрибут **min**)
3. максимальная длина (атрибут **max**)