№1. Что такое **таблица**? Какие **операции** определены над таблицами?

Таблица — это набор элементов одинаковой организации, каждый из которых можно представить в виде двойки , где K — ключ, а V — тело (информационная часть) элемента. Ключ уникален для каждого элемента, т.е. в таблице нет двух элементов с одинаковыми ключами. Ключ используется для доступа к элементам при выполнении операций

Таблица — динамическая структура. Над таблицей определены следующие основные операции:

1. Инициализация.
2. Включение элемента.
3. Исключение элемента с заданным ключем.
4. Чтение элемента с заданным ключем.
5. Изменение элемента с заданным ключем.
6. Проверка пустоты таблицы.
7. Уничтожение таблицы.

№2. Как **классифицируются** таблицы в зависимости от способа размещения их элементов?

На физическом уровне таблица реализуется последовательной или связной схемой хранения. Располагаться таблица может в статической или динамической памяти. В зависимости от способа размещения элементов таблицы **классифицируются на неупорядоченные, упорядоченные и хештаблицы.**

№3. Определите порядок функции временной сложности операций **включения** и **исключения** элементов в **неупорядоченные** и **упорядоченные** таблицы.

На неупорядоченных, сложность **включения** O(n), так как необходимо выполнить линейный поиск, для проверки, не записан ли ранее заданный ключ, после чего включаем элемент на последнее место. Сложность **исключения** так же O(n), так как необходимо найти ключ. На ПЛС при исключении, последний элемент передвинется на позицию исключаемого элемента.

Для упорядоченных, возможны следующий реализации: ПЛС, ОЛС и дерево. В ПЛС для **включения** сначала найти позицию, на которую нужно включить элемент. Поиск можно осуществлять при помощи линейного, бинарного и блочного поисков. После чего необходимо подвинуть все элементы, расположенные правее на одну позицию вправо. В ОЛС перемещение не требуется, поэтому нужно будет лишь найти позицию включения.

№4. Как **исключить** элемент из упорядоченной таблицы, реализованной с использованием **бинарного дерев**а?

В БД **Включения** Слева меньшие, справа большие.

Алгоритм исключения элемента зависит от типа вершины, в которой находится исключаемый элемент таблицы. Возможны три типа вершин:

1) вершина не имеет потомков (лист). В этом случае родитель потеряет соответствующего сына.

2) вершина имеет одного потомка. В результате исключения такой вершины сыном родителя станет потомок удаляемой вершины.

3) вершина имеет двух потомков. Для исключения вершины нужно найти такую подходящую легко удаляемую вершину (типа 1 или 2), значение которой можно было бы переписать в исключаемую вершину без нарушения основного свойства. Такая вершина всегда существует: это либо самая правая вершина в левом поддереве, либо самая левая вершина в правом поддереве (объясните, почему).

№5. Что такое **хеш-таблица, хеш-функция, коллизия**?

Хеш-таблица — это таблица, в которой положение ai (адрес) элемента в памяти определяется с помощью некоторой функции H (хеш-функции), аргументом которой является значение ключа kj элемента. ~~Функция хеширования определяется как отображение~~

$H: K \to A$, где
$K=\{k_1,k_2,\ldots,k_m\}$ — множество значений ключа;
$A=\{a_1,a_2,\ldots,a_n\}$ — адресное пространство;
$m \le n$.

Коллизия происходит тогда, когда результат хеш функции может совпадать с результатом другого, уже занятого ключа, то есть когда на одну ячейку в памяти претендует два различных элемента.

№6. Какие существуют **методы разрешения коллизий**?

Наиболее часто используют для этого алгоритмы из двух основных групп, а именно: **открытая адресация** (методы двойного хеширования и линейного опробования) и **метод цепочек**.

В методе **открытой адресации** в качестве дополнительной СД используется массив, элементы которого могут содержать элементы таблицы. Для этого используется вторая хеш-функция (функция рехеширования):

1. Вычислить $a_i = H_1(k_j)$.

2. Если при включении в таблицу новой записи элемент массива a_i или содержал ранее элемент таблицы, но в настоящее время не содержит, а при исключении содержит элемент таблицы с ключем k_j, то поиск завершен. В противном случае перейти к следующему пункту.

3. Вычислить $a_i - H_2(a_i)$ и перейти к пункту 2.

В **методе цепочек** элемент массива содержит цепочку элементов массива. Для хранения таких цепочек может использоваться линейный список, либо для ускорения поиска элементов таблицы, может использоваться БД.

№7. При каком методе разрешения коллизий возможно **зацикливание** и как его избежать?

При включении новых элементов таблицы алгоритм остается конечным до тех пор, пока есть хотя бы один свободный элемент массива. При исключении элемента из таблицы алгоритм конечен, если таблица содержит элемент с заданным ключем. При невыполнении этих условий произойдет зацикливание, против которого нужно принимать специальные меры. Например, можно ввести счетчик числа просмотренных позиций, если это число станет больше n, то закончить алгоритм.

№8. Определите **порядок** функции временной сложности алгоритмов выполнения операций над хеш-таблицами.

Время выполнения операций над хеш-таблицей зависит от числа коллизий и времени вычисления значения хеш-функции, уменьшить которое можно путем использования «хорошей» хеш-функции.

Особенностью хеш-таблиц является то, что время поиска элемента с заданным ключем не зависит от количества элементов в таблице, а зависит только от заполненности массива.