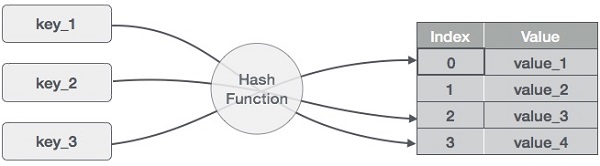
**Hash Table**

std::unordered\_map\multimap, std::unordered\_set\multiset all use hash table.





Hash table is a data structure that uses an array of keys.  A hash table uses a [hash function](https://en.wikipedia.org/wiki/Hash_function) to compute an *index* into an array of *slots (or “buckets”)*, from which the desired value can be found.

Given a key, the algorithm of hash function computes an *index* that suggests where the entry can be found.

It works like this:

Input: Key

Operation Hash(Key) is called.

Output: Index

Every time you enter the same key, you will get the original index. Thanks to that, we get O(1) lookup time!

**Hash function:**

A basic requirement is that the function should provide a uniform distribution of hash values. A non-uniform distribution increases the number of collisions and the cost of resolving them.

The hash function might be simply *h*(*key*) = *key* % *SIZE* where SIZE is the size of an array.

**Hash collisions**

Hash collisions happen when hash function deduces the same index for different keys.

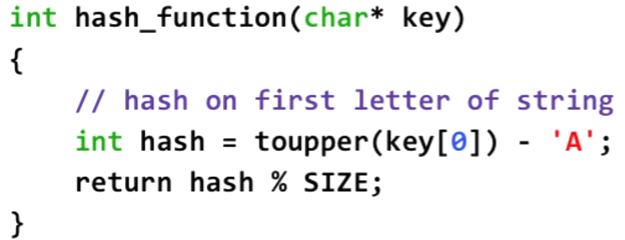
All hash table implementations have some kind of collision resolution strategy to handle such events.

#### The most popular is called “Separate chaining with linked lists”:

When index in array is already taken, we create linked list for this index.

Then, we push the colliding key onto the list.

For example, with hash function implemented like this:



Elements starting with the same letter would be placed in hash table like this:

