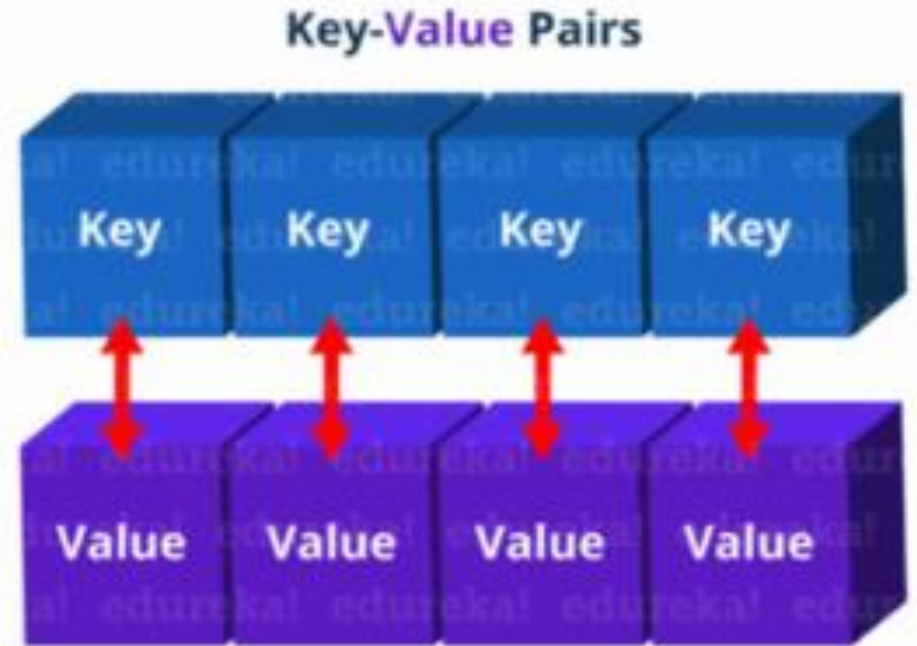


# HashStl

#MORE POWER TO C USERS.



## Reason behind choosing HashStl

→ **Curious to know**

How internally Hashmap works ?

How to handle Collision inside Hashmap ?

How internally STL functions works ?

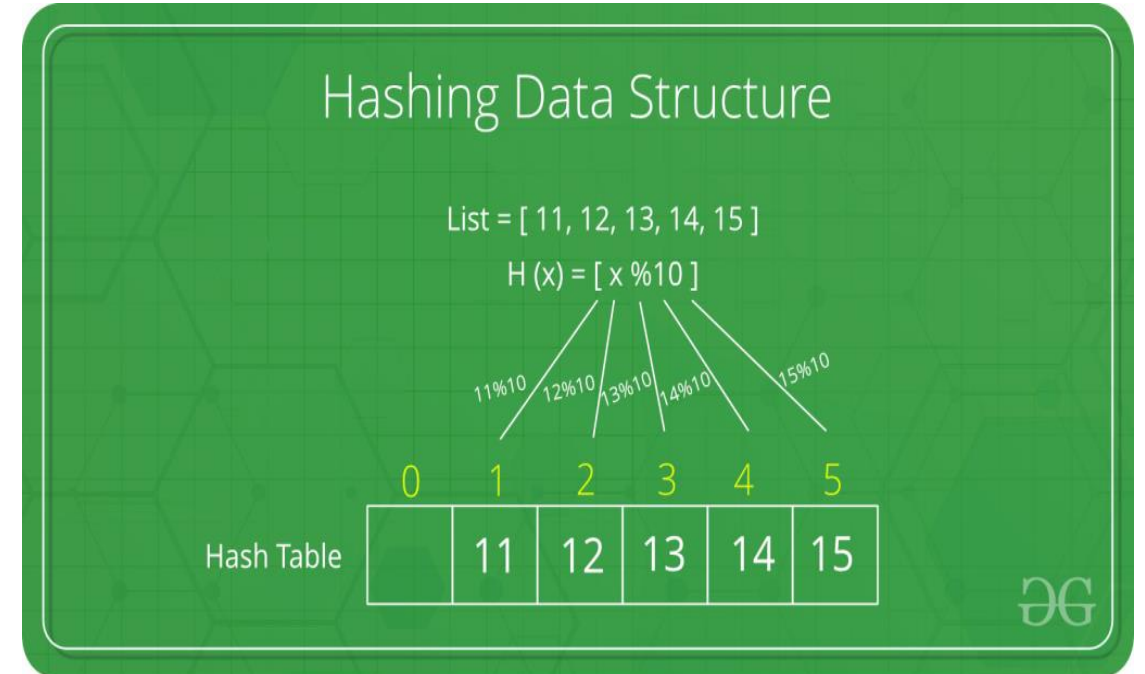


# What is Hashmap ?

Hashmap is an associated container that stores elements formed by the combination of key-value and a mapped value. The key value is used to uniquely identify the element and the mapped value is the content associated with the key.

# What is STL ?

The Standard Template Library (STL) is a set of C++ template classes to provide common programming function like upper bound, lower\_bound, sort etc.



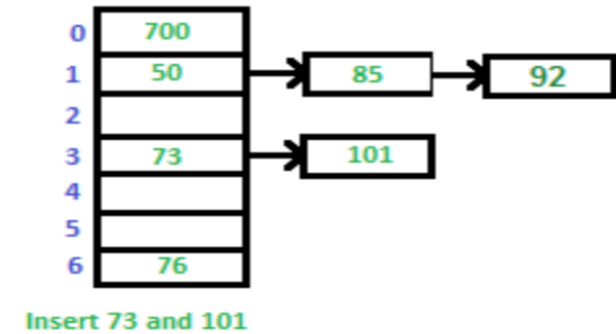
What will happen if Position in Hashmap already occupied ?



## Ways to handle collision :-

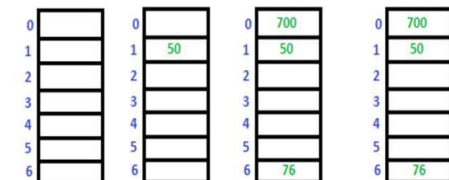


### Chaining



### Open Addressing

### Hashing with Open Addressing



## Reason to choose Open Addressing

---



Open Addressing provides better cache performance as everything stored in the same table.



Open Addressing does not required additional space while in case of Chaining additional space required for links.



In case of Chaining there might be Wastage of Space as Some Parts of hash table in chaining are never used . While in case of Open Addressing a slot can be used even if an input doesn't map to it.

➡ **Number of slots in hashmap  $\geq$  Number of keys inserted.**

# HashStl

#MORE POWER TO C USERS.

## Demo

```
vagrant@myvm19: ~/HashStl
vagrant@myvm19:~$ cd HashStl/
vagrant@myvm19:~/HashStl$ gcc Main.c hashmap.c hashmap.h stl.h stl.c
vagrant@myvm19:~/HashStl$ ./a.out

----- HashStl working Instructions -----
Press 'I' To Insert Key and Value into Hashmap . It Takes 2 Parameter 'Key' and 'Value'
Press 'D' along with key To Delete Key in Hashmap Data structure
Press 'P' To Display Hashmap Data structure
Press 'V' along key To Know Value corresponding key in Hashmap Data structure
Press 'S' along to key To Search Key Hashmap Data structure
Press 'C' To Clear Hashmap Data structure
Press 'R' To Know Max Capacity Of Hashmap Data structure
Press 'X' To Know Current size Of Hashmap Data structure
Press 'K' To check Hashmap Data structure empty or not
Press 'O' To get count of occurrence of key int Hashmap Data structure
Press 'A' To Display STL for INT Datatype
Press 'W' To Display STL for CHAR Datatype
Press 'E' To Exit

HashStl Execution begins
Enter Your desire operation
--> |
```

UCA\_TASKS/BitB.c at master · va · HashStl/hashmap.h at main · va · (1) Two Sum - LeetCode · (1) Two Sum - Submission Detail · +

leetcod.com/problems/two-sum/submissions/

LeetCode Explore Problems Interview Contest Discuss Store September LeetCode Challenge 2021 Premium

Success Details >

Runtime: 1503 ms, faster than 5.12% of C online submissions for Two Sum.

Memory Usage: 8.5 MB, less than 5.07% of C online submissions for Two Sum.

Next challenges: Two Sum III - Data structure design Two Sum Less Than K Count Good Meals

Show off your acceptance: f t in

Time Submitted	Status	Runtime	Memory	Language
09/10/2021 14:53	Accepted	1503 ms	8.5 MB	c
09/10/2021 14:53	Accepted	1068 ms	8.3 MB	c

```
50     int count_of_occurrence;
51 };
52
53 // creating an Hash_map_array of struct type.
54 struct Hash_map* Hash_map_array[SIZ];
55
56 // Function to return Hash_index ( O(1) Time complexity ).
57 int Hash_index(int val)
58 {
59     return (val + SIZ)%SIZ;
60 }
61
62 // Function to return maximum size of Hash_map_array.
63 void Max_Element_store()
64 {
65     printf("Map_default_size is %d\n",SIZ);
66 }
67
68 // Function to return current element count in Hash_map_array.
69 void Map_Size()
70 {
71     printf("Map_Element_count is %d\n",map_element_count);
72 }
73
74 // Function to return count of occurrence of key.
75 int get_count(int key)
76 {
77     struct Hash_map *temp=search(key,0);
78     if(temp==NULL)
79         return -1;
80 }
```

Problems Pick One < Prev 1/1998 Next > Console - Contribute i Run Code ^ Submit

Type here to search 22:10 09-09-2021 44% 28°C ENG

Type here to search 14:54 10-09-2021 99% 31°C ENG