
## Department of Computer Science and Engineering

**Course Code: CSE 430 Course Title: Compiler Design Lab**

**Lab Exercise 2**

*Read the instructions carefully and submit the exercise within time*

# Task: Construct a simple hash-based symbol table (data-dictionary) based on chaining.

Symbol table is an important data structure created and maintained by compilers in order to store information about the occurrence of various entities such as variable names, function names, objects, classes, interfaces etc. The information is collected by the analysis phase of the compiler and used by dash synthesis phase to generate target code.

## Usage of symbol table by all the phases of a compiler

i. Lexical analysis: Creates new entries for each new identifiers.
ii. Syntax Analysis: Adds information regarding attributes like type, scope, dimension, line of reference and line of use.
iii. Semantic Analysis: adds information regarding attributes like type, scope, dimension, line of reference and line of usage.
iv. Intermediate Code Generation: information in symbol table helps to add temporary variable's information.
v. Code Optimization: information in symbol table used in machine-dependent optimization by considering address and variable information.
vi. Target Code Generation: generates the code by using the address information of identifiers.

## Symbol Table Entries

each entry in the symbol table is associated with "attributes" that support the compiler in different phases the attributes are:

- Name
- Size
- Dimension (used if it is an array)
- Type
- Line of declaration (where the variable is declared to generate errors)
- Line of usage (link list to keep track of multiple usage of a variable)
- Address

## Example

| Name | Type | Size | Dimension | Line of code | Line of usage | Address |
|---|---|---|---|---|---|---|
| John | Char | 4 | 1 | 5 | 12 | 0x6dfed4 |
| Age | Int | 2 | 0 | 3 | 5 | 0x7ffdd8747 |

## Symbol Table Data Structure

To store data in a symbol table we can use various types of data structures such as:

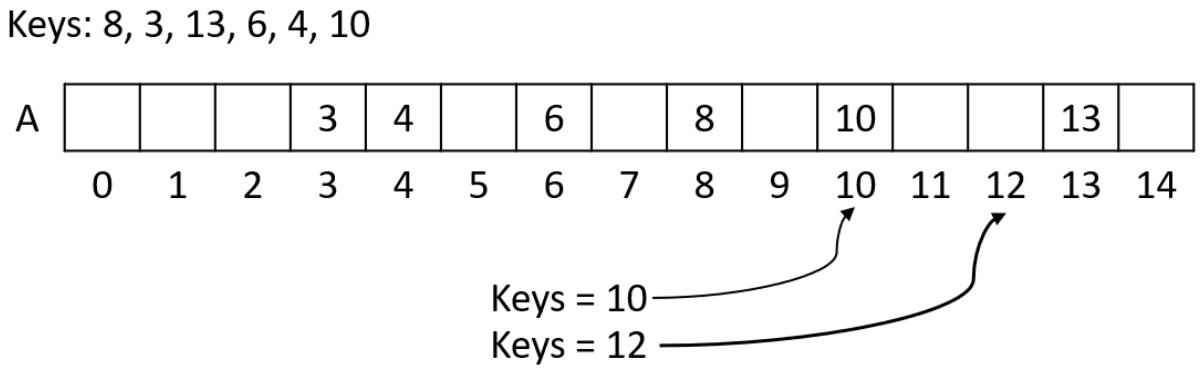
| Data Structure | Complexity |
|---|---|
| Linear List (Linked List) | O(n) |
| Binary Search Tree | O(logn) |
| Hash Table | O(1) |

## Hash Table

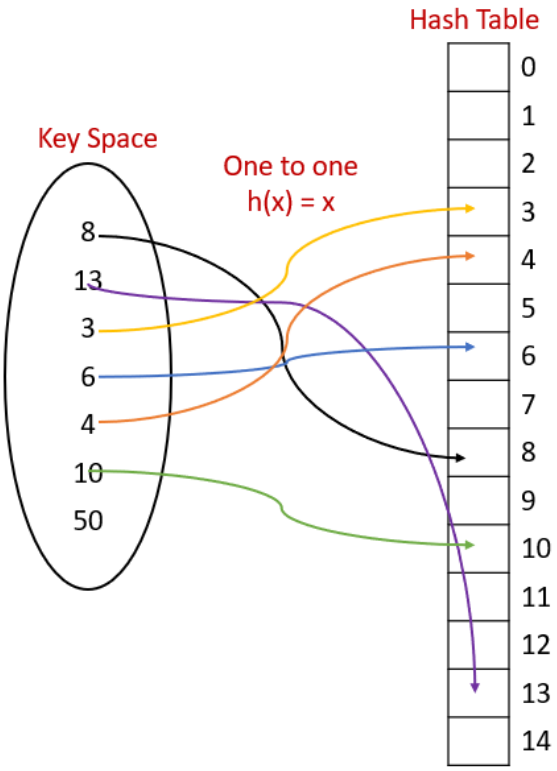
We will use hash table as a data structure for our symbol table. A hash table allows very fast retrieval of data. It is widely used in database indexing, caching and error checking. At the high level, a hash table is a key value lookup.

In linear search, the elements are arranged in an array randomly. And for searching we start from left to right. This is the reason of the time complexity O(n). In binary search the elements are sorted and kept in an array. Binary search improves the search method and makes it faster to (O)(logn). But, sorting the elements will take time. This is the reason hashing technique is used as it takes constant time O(1).

In hashing technique if list of elements are given and array space is given where we have to store the list of elements then we store the element upon its own index. We take the value of an element itself as the index and we store it at same index.



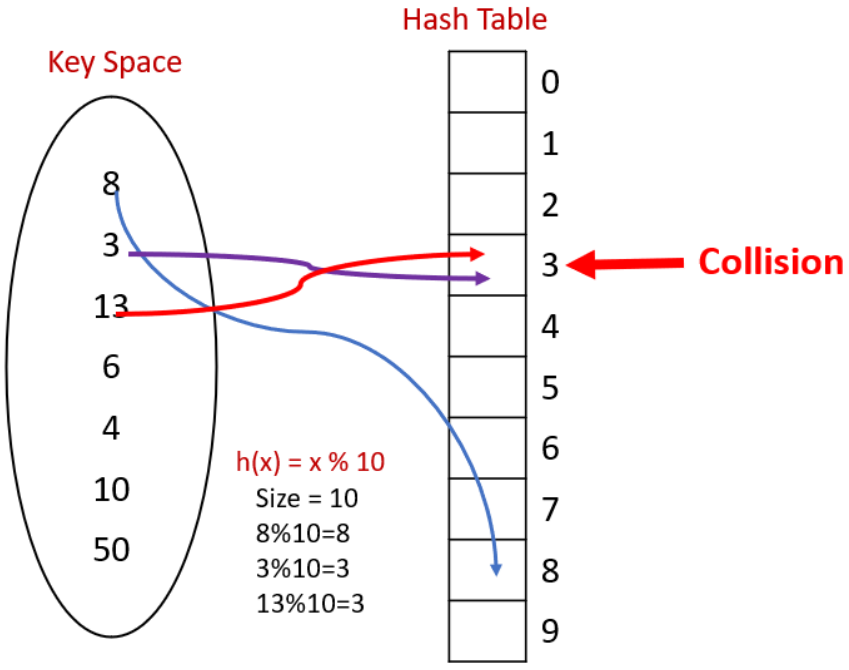

Here, keys 8, 3, 13, 6, 4, 10 are stored on the index 8, 3, 13, 6, 4, 10 itself. But, problem will occur if the key is 50 then we will have to store it on index 50 and a lot of space will be wasted. To improve this technique, we adapt a mathematical model for hashing based on functions.




The above diagram shows the use of the hash function h(x) = x and here we have the same space problem. So, we modify the function as:

h(x) = x % 10 where, 10 is the size of hash table and x is the key

As the modified hash function is used, we are able to solve the space problem at some extend.



We take the key 8 and store it on h(8)=8%10=8 index. Similarly, we store 3 as 3%10=3 index. But, while we try to store key 13, 13%10=3 which is overlapping with already existing key 3. This scenario is called ‘collision’.

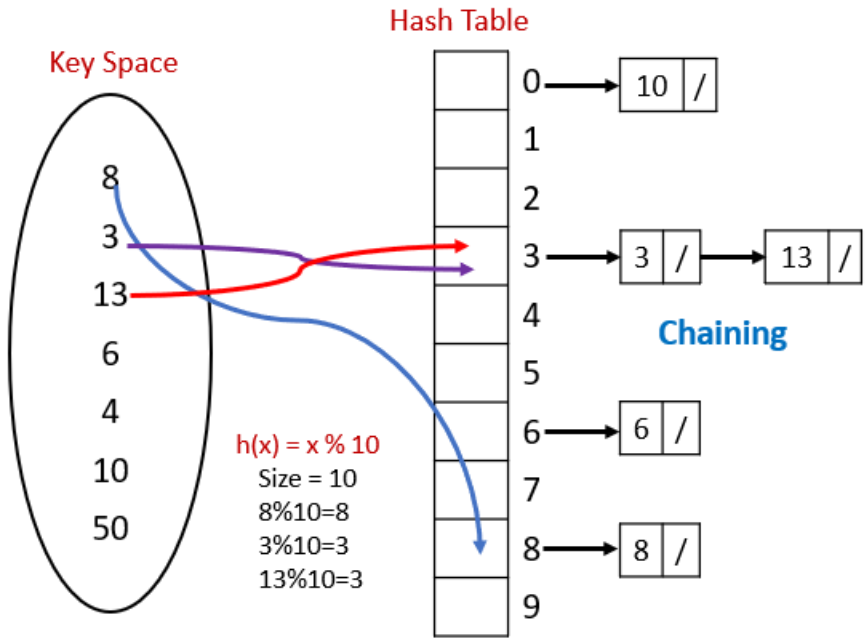
## Chaining

Chaining is an open hashing collision resolution technique which uses the same modified hash function as shown above. But, here the element will not be stored directly at the index. A chain of linked list is formed to store the elements. So, we can use a distinct hash table size as per as our choice and we can store as much as elements in the hash table we want.

For searching a key in the chaining based hash table, we use the same hash function

h(x) = x % 10 where, 10 is the size of hash table and x is the key

So, if we need to search for the element 13, 13%10=3, we search the chain at index 3 for the key 13. When an element is not found in the chain, it is considered as a unsuccessful search.




## Sample Inputs and Outputs

The input to your program will be a sequence of six tuples . They are:

- Name
- Type
- Size
- Dimension
- Line of Code
- Address

Where each element in each tuple is a string

An example of input sequence is given below:

```
x, ID, 2, 1, 5, 0x6dfed4
```

The symbol table should be able to store multiple rows and the symbol table should have the following functionalities:

1. ***Insert*** a new symbol/name along with its type into the symbol table

2. ***Search*** a symbol/name along with its type from the symbol table

3. ***Delete*** a symbol/name along with its type into the symbol table

4. ***Show*** the contents of the symbol table in the console

5. ***Update*** an already existing entry in the symbol table

6. ***getHashKey()*** function is used to show the hash value