Implementation of Symbol table

Following are commonly used data structure for implementing symbol table:-

1. **Hash Table**

- A hash table is an array with index range: 0 to Table Size – 1
- Most commonly used data structure to implement symbol tables
- Insertion and lookup can be made very fast – O(1)
- A hash function maps an identifier name into a table index
- A hash function, h(name), should depend solely on name
- h(name) should be computed quickly
- h should be uniform and randomizing in distributing names
- All table indices should be mapped with equal probability
- Similar names should not cluster to the same table index

2. **List**

- In this method, an array is used to store names and associated information.

- A pointer "available" is maintained at end of all stored records and new names are added in the order as they arrive
- To search for a name we start from beginning of list till available pointer and if not found we get an error "use of undeclared name"
- While inserting a new name we must ensure that it is not already present otherwise error occurs i.e. "Multiple defined name"
- Insertion is fast O(1), but lookup is slow for large tables – O(n) on average
- Advantage is that it takes minimum amount of space.

3. **Linked List**

- This implementation is using linked list. A link field is added to each record.
- Searching of names is done in order pointed by link of link field.
- A pointer "First" is maintained to point to first record of symbol table.
- Insertion is fast O(1), but lookup is slow for large tables – O(n) on average

4. **Binary Search Tree**

- Another approach to implement symbol table is to use binary search tree, we add two link fields i.e. left and right child.

- All names are created as child of root node that always follows the property of binary search tree.
- Insertion and lookup are O ($\log_2$ n) on average.