Symbol Table:

A symbol table stores tokens identified in lexical analysis. It adds type and scope information during syntactical and semantical analysis. It is used for live analysis in optimization.

A symbol table is a significant data structure used in a compiler that correlates characteristics with program identifiers. The analysis and synthesis stages employ symbol table information to verify that used identifiers have been specified, to validate that expressions and assignments are semantically accurate, and to build intermediate or target code.

Possible Implementations Techniques

There are three data structures used to implement symbol table:

1.Linear List

2.Binary Search Tree

3.Hash Table

Here we have used the Hash Table and Linear List to implement symbol table.

1. Linear List:

It is the simplest and most straightforward method of implementing data structures. To store the information about the token that has been identified, we create an empty list. New names are added to the list in the order that they appear. Here we have used the class symbol and default constructor to store the data of the symbol that we have entered and stored it using the attributes of the class. To display the data we want, we have used the display function and for searching the data that we want search function is being used here.

- The time required to search an element using a linear list algorithm depends on the size of the list.
- The time complexity of a linear list is O(n).

2. Hash Function:

The hash function is the mapping between an item and the place in the hash table where that item resides. The hash function takes a collection item and returns an integer in the range of slot names from 0 to m-1.

Here we have used nested lists and each index is given by hashing. Similar to linear list, we have used the class named "Table" and used a constructor to insert and assign the data using the attributes of the class label, address, datatype.

For searching, hashing function is used since it matches the same index number in the "inp" list to the keylabel hashing. And the last one is to display.

- The time complexity of a hashing is O(1).