Week-2 Compiler design Lab Report

AP20110010628

Aileni Ashish Reddy

**Symbol Table:**

The compiler creates and maintains a data structure to store information about the occurrence of various entities such as variable and function names, objects and classes is known as Symbol table.

Items stored in symbol table:

Variable names and constants, Procedure and function names, Literal constants and strings, Compiler generated temporaries, Labels in source languages.

Symbol table can be implemented in one of the following ways:

Linear (sorted or unsorted) list

Binary Search Tree

Hash table

And other ways.

**Implementation of symbol table using Linear List:**

This code is for creating Symbol Table using linear list.

Linear list is a simplest and easiest to implement data structure. We use a single array to store names and their associated information. New names are added to the list in the order in which they are encountered. To insert a new name, we must scan down the list to make sure that it is not already there. When the name is located, the associated information can be found in words following next. Searching of names is done in order pointed by the link of the link field. A pointer “First” is maintained to point to the first record of the symbol table. Insertion is fast O(1), but lookup is slow for large tables – O(n) on average.

In these symbol table program. I am taking an expression as “a=d\*c+b-3”. Based on these in expression I am going to separate the expression into the lexemes. And based on these lexemes the code separates them and identifies which one is identifier or operator or c

constant or an Invalid token.

Here this program considers any alphabet as the identifier, any number from 0 to 9 as the constant and the symbols “+, -, \*, /, =” are considered as Operators.

Here this code stores the expression in expr variable which is initialized as an array of size 100.

Now the code identifies the lexemes type and stores in the Symbol Table, here is the Output of the Code.

