Lab Assignment 3

Implementation of Symbol Table Using Data Structures

Nikitha Godavarthi.
AP19110010460
CSE-C

A. Symbol Table Implementation Using Stack Data Structure

Language Used: Python

Steps :

1. Create a stack data structure, an empty list to store the information of the identifiers, and a tuple to store the types of identifiers whose information can be stored in the symbol table.
2. A while loop surrounds the main part of the program, the user enters their choice if he/she wants to continue entering the identifier or not.
    a. If the user decides to continue, a menu will appear to show a list of accepted identifiers. User will choose one and start to provide information about the identifier.
        1. Variable - The user must provide the name, data type, scope of the variable.
        2. Function - The must should provide the name, return type, no. of parameters followed by the name and data type of the parameters, and the parameters passing technique of the Functions.
        3. Structure - The user must provide a name, no. of members followed by the names and data type of the members of the structure.
        4. Pointer - The user must provide the name, data type,

name of the identifier the pointer is holding.

5. Array - The user should provide the name, data type, no. of dimensions followed by the values of the dimensions of the array.

3. In each case, the information is pushed into the stack along with the address of these identifiers.
4. If an identifier with the same name is entered again, Then the output will display that the identifier already exists.
5. Once the identifier information is entered, the list item is added to the symbol table and the stack is cleared to push the values of the next Identifier.

Printing The Symbol Table :

1. Filters representing identifier type are set as the first element of each list item in symbol table.
2. A for loop is used to iterate through all the list items in the symbol table, if filters by 'variable' all the same identifiers will be printed and the loop iterates through the entries to print remaining identifiers

Output Screenshots :

```
**************CD-LAB-2-WEEK**************


Do you want to enter indentifier in the symbol table ?
Press Y or N please
 y
1 variable
2 function
3 structure
4 pointer
5 array
Enter your option : 1
Enter the name of the variable : X
Enter the datatype of the variable : int
Enter the scope of the variable (global, inside function, inside class): insid
e class
Information of X in the symbol table :
['variable', 'X', 'int', 'inside class', 2178612879856]
Do you want to enter indentifier in the symbol table ?
Press Y or N please
 N



Symbol Table


        Variables


name    datatype        scope           address
X       int             inside class            2178612879856
---------------------******----------------------


        Pointers
```

```
        Variables


name    datatype        scope           address
X       int             inside class            2178612879856
---------------------******----------------------


        Pointers


name    datatype        identifiersTable holding                value
                address
---------------------******----------------------


        Functions


name    returntype      no. of para     (name,datatype)                 passing method                          address
-----------------------------------------


        Structures


name    members (name,datatype)                         address
---------------------******----------------------


        Arrays


name    datatype        dim     dimvalue        address
---------------------******----------------------
```

B. Implementation Of Symbol Table Using Hash Table

Symbol tables are mainly implemented as hash tables, where the symbol from the source code itself is treated as a key to the hash function and the return value is the symbol information.

The following possible information about identifiers is stored in the symbol table:

Name (as string)

Attribute: Reserved word, Variable name, Type name, Procedure name, Constant name

The data type

The block level Its scope (global, local, or parameter)

Its offset from the base pointer (for local variables and parameters only)

A symbol table provides following operations:

insert() and lookup()

From our code: The entry of symbol table is

```
{
int add;
char label[10];
}sy[11];
```

A compiler maintains multiple block levels of symbol tables:

Level 0: A null hash table at level 0

Level 1: Keyword in the hash table at level 1

Level 2: Global symbol table which can be accessed by all the procedures

Level 4: Scope symbol tables that are created for each scope in the program

int create(int num) is used to create the symbol table,

The maximum number of values which can be given in our symbol table are 11.

int key;

key=num%11;

return key; -------> The source code to represent where given values will be stored in table.

void search()------> is used to search the required value given.

C. Symbol Table implementation using a Linear list data structure

Language can be used: C/C++.

It is the easiest way to implement the symbol table

Steps :

1. We should create an array, it can be a one dimensional or multidimensional array.
2. When we enter any information, it will be stored as a name and attribute in the array which we have created.
3. At the end of an array, there can be a pointer that indicates the availability to insert an element.
4. So, when we insert an entry, it will be stored in the available array position where the pointer indicates.
5. Search is like linear array search, it starts from the first entry and goes to the last element.
6. Insertion is fast O(1), but lookup is slow for large tables – O(n) on average.