SYMBOL TABLE DESIGN

WHO CREATES SYMBOL TABLE??

- Identifiers and attributes are entered by the analysis phases when processing a definition (declaration) of an identifier
- In simple languages with only global variables and implicit declarations:
  - ✓ The scanner can enter an identifier into a symbol table if it is not already there
- In block-structured languages with scopes and explicit declarations:
  - ✓ The parser and/or semantic analyzer enter identifiers and corresponding attributes

USE OF SYMBOL TABLE

- Symbol table information is used by the analysis and synthesis phases
- To verify that used identifiers have been defined (declared)
- To verify that expressions and assignments are semantically correct – type checking
- To generate intermediate or target code

SYMBOL TABLE DATA STRUCTURES

- Issues to consider : Operations required
    - Insert
        - Add symbol to symbol table
    - Look UP
        - Find symbol in the symbol table (and get its attributes)
- Insertion is done only once
- Look Up is done many times
- Need Fast Look Up
- The data structure should be designed to allow the compiler to find the record for each name quickly and to store or retrieve data from that record quickly.

LINKED LIST

- A linear list of records is the easiest way to implement symbol table.
- The new names are added to the symbol table in the order they arrive.
- Whenever a new name is to be added to be added it is first searched linearly or sequentially to check if or the name is already present in the table or not and if not , it is added accordingly.
- Time complexity – O(n)
- Advantage – less space , additions are simple
- Disadvantages - higher access time.

UNSORTED LIST

```
01  PROGRAM Main
02    GLOBAL a,b
03    PROCEDURE P (PARAMETER x)
04            LOCAL a
05    BEGIN {P}
06            …a…
07            …b…
08            …x…
09    END  {P}
10  BEGIN{Main}
11    Call P(a)
12 END {Main}
```

Look up Complexity $O(n)$

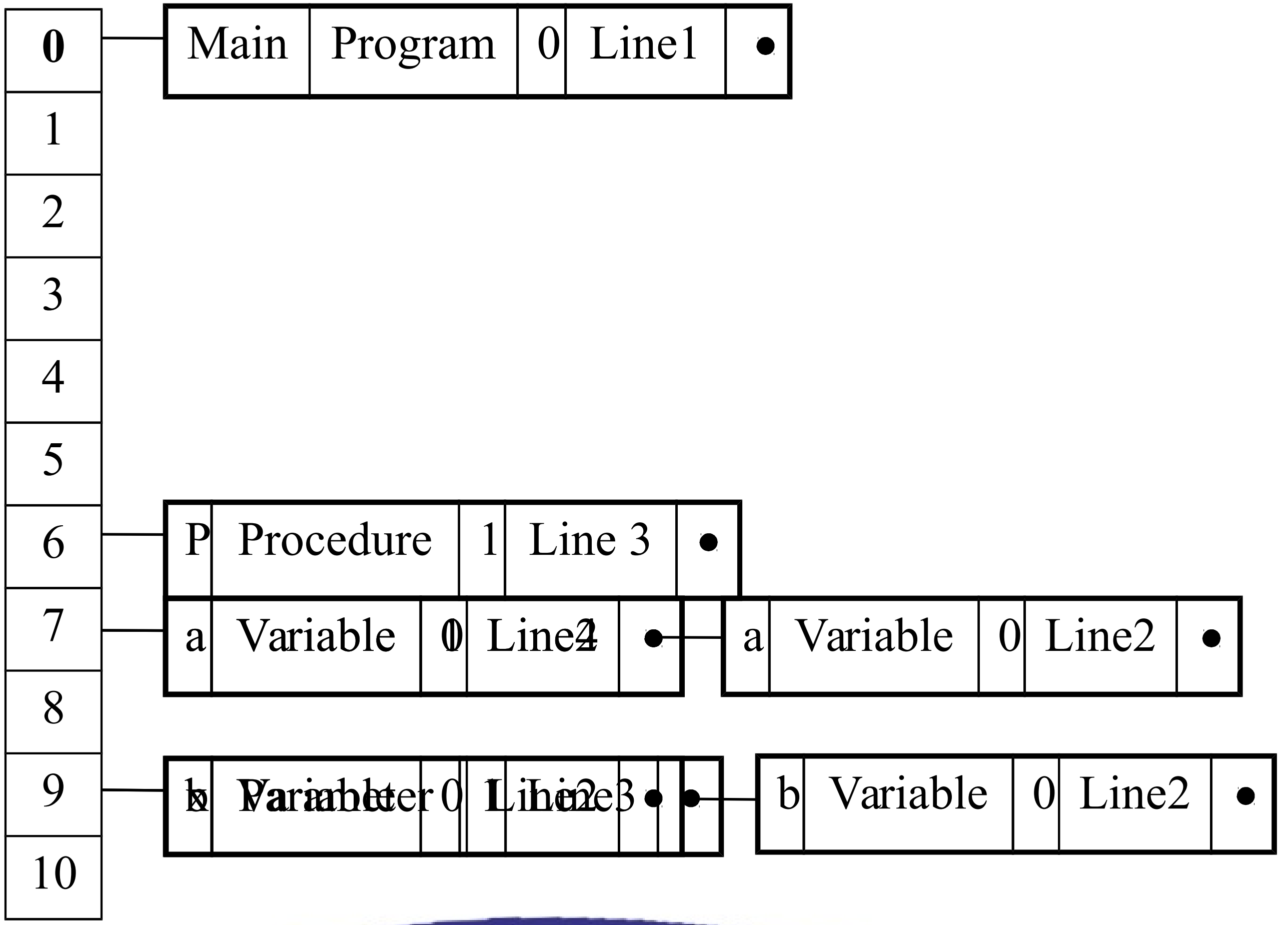
Name	**Characteristic Class**			**Scope**	**Other Attributes**
		Declared		**Referenced**	**Other**
Main	Program	0	Line 1		
a	Variable	0	Line 2	Line 11	
b	Variable	0	Line 2	Line 7	
P	Procedure	0	Line 3	Line 11	1, parameter, x
x	Parameter	1	Line 3	Line 8	
a	Variable	1	Line 4	Line 6	

HASH TABLE

- Table of k pointers numbered from zero to k-1 that points to the symbol table and a record within the symbol table.
- To enter a name in to the symbol table we found out the hash value of the name by applying a suitable hash function.
- The hash function maps the name into an integer between zero and k-1 and using this value as an index in the hash table.

HASH TABLE - EXAMPLE

M	n	a	b	P	x
77	110	97	98	80	120



0 → Main | Program | 0 | Line1 | ●

1

2

3

4

5

6 → P | Procedure | 1 | Line 3 | ●

7 → a | Variable | 1 | Line4 | ● → a | Variable | 0 | Line2 | ●

8

9 → x | Parameter | 1 | Line3 | ● / b | Variable | 0 | Line2 | ● → b | Variable | 0 | Line2 | ●

10

```
PROGRAM Main
GLOBAL a,b
PROCEDURE
P(PARAMETER x)
LOCAL a
BEGIN (P)
...a...
...b...
...x...
END (P)
BEGIN  (Main)
Call P(a)
End (Main)
```

H(Id) = (# of first letter + # of last letter) mod 11

REFERENCES

- *O.G.Kakde - Compiler design*
- *Compilers Principles, Techniques & Tools - Second Edition : Alfred V Aho , Monica S Lam, Ravi Sethi, Jeffery D Ullman*

THANK YOU…