Symbol table

Symbol table is an important data structure created and maintained by compilers in order to store information about the occurrence of various entities such as variable names, function names, objects, classes, interfaces, etc.

Symbol table is used by both the analysis and the synthesis parts of a compiler.

A symbol table may serve the following purposes

- To store the names of all entities in a structured form at one place.
- To verify if a variable has been declared.
- To implement type checking, by verifying assignments and expressions in the source code are semantically correct.
- To determine the scope of a name (scope resolution).

Implementation:

If a compiler is to handle a small amount of data, then the symbol table can be implemented as an unordered list, which is easy to code, but it is only suitable for small tables only.

A symbol table can be implemented in one of the following ways:

- Linear (sorted or unsorted) list
- Binary Search Tree
- Hash table

Among all, symbol tables are mostly implemented as hash tables, where the source code symbol itself is treated as a key for the hash function and the return value is the information about the symbol.

Operations:

A symbol table, either linear or hash, should provide the following operations.

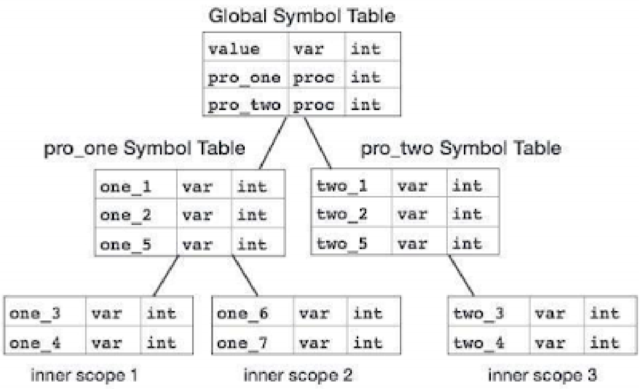
- Allocate: to allocate a new empty symbol table.
- Free: to remove all entries and free the storage of a symbol table.
- Insert: to insert a name in a symbol table and return a pointer to its entry.
- lookup: to search for a name and return a pointer to its entry.
- set_ attribute: to associate an attribute with a given entry.
- get _attribute: to get an attribute associated with a given entry.

Scope Management:

A compiler maintains two types of symbol tables:

1. Global symbol table: which can be accessed by all the procedures .
2. Scope symbol tables: that are created for each scope in the program.

To determine the scope of a name, symbol tables are arranged in hierarchical structure as shown in the example below:



Global Symbol Table

value	var	int
pro_one	proc	int
pro_two	proc	int

pro_one Symbol Table

one_1	var	int
one_2	var	int
one_5	var	int

pro_two Symbol Table

two_1	var	int
two_2	var	int
two_5	var	int

one_3	var	int
one_4	var	int

inner scope 1

one_6	var	int
one_7	var	int

inner scope 2

two_3	var	int
two_4	var	int

inner scope 3

This symbol table data structure hierarchy is stored in the semantic analyzer and whenever a name needs to be searched in a symbol table, it is searched using the following algorithm:

1. First a symbol will be searched in the current scope, i.e. current symbol table.
2. If a name is found, then search is completed, else it will be searched in the parent symbol table until.
3. Either the name is found or global symbol table has been searched for the name.