Symbol Table

Role of Symbol Table

- Essential data structure for compiler
- Used for storing information about identifiers appearing in a source program
- Lexical Analyzer and Parser fill up symbol table
- Code generator and optimizer make use of symbol table
- Entities stored in a symbol table
  - Variables, procedures, functions, defined constants, labels, structures, file identifications, compiler generated temporaries

Information in symbol table

- Name: May be stored directly in the table or the entry may point to another character string, possibly in an associated string table
- Type: type of identifier
  - Whether variable / function / procedure name
  - For variables, identify whether integer / real / array …
- Location: offset in the program where the identifier is defined
- Scope: identifies the region of the program in which the current symbol definition is valid
- Other attributes: Array limits, record fields / parameters / return values of functions

Usage of symbol table information

- Semantic Analysis: check correct semantic usage of language constructs
  - May need checking types of identifiers
- Code generation: All program variables and temporaries need to be allocated some memory locations
  - Symbol table provides information regarding memory size required for identifiers by their types
- Error Detection: Leave variables undefined
- Optimization: To reduce the total number of variables used in a program we need to reuse the temporaries generated by the compiler

Features of symbol tables

- Insert
- Delete
- Lookup
- Modify

Symbol Table Design

- Format of symbol table entries
  - Linear Lists / arrays / trees….
- Access mechanism
  - Linear search / hashing / binary search ..
- Location of storage
  - Primary Memory (RAM) / Secondary Memory (large symbol table)
- Scope issues
  - Simple symbol table with nested scope
  - Scoped symbol table with nested scopes

Simple symbol table: Operations

- Enter a new symbol into the table
- Lookup for a symbol
- Modify information about a symbol stored earlier

Simple Symbol Table Formats

- Linear Table
- Ordered List
- Tree
- Hash table

Linear Table

int x, y;

float z;

…..

procedure abc

…..

L1:…

…..

Linear Table

Name	Type	Location
x	int	Offset of x
y	int	Offset of y
z	float	Offset of y
Abc	procedure	Offset of abc
L1	label	Offset of L1

- Insert, Lookup and modify operations take O (n) time, n being the number of identifiers
- Insertion can be made in O (1) be remembering the pointer to the next free position

Ordered List

- Variation of linear table
- List may be sorted and a binary search may be used for access in O (log n)
- Insertion needs to be done at proper place to preserve the sorted nature
- Self-organizing list: Dynamically re-arrange list based on recency of reference

Tree Symbol Table

- Each entry is used represented as a node in a tree
- Based on string comparison of names, entries lesser than a reference node are kept in the left subtree and entries greater than a reference node are kept in the right subtree
- Average lookup time is O (log n)
- AVL trees may be used

Tree symbol table example