Symbol Tables
Article #12

From theoretical point of view, a ***symbol table*** is a mechanism that associates *attributes* with names. Because these attributes are a representation of the meaning (or semantics) of the names with which they are associated, a symbol table is sometimes called dictionary. A symbol table is a necessary component of a compiler because the introduction of a name appears in only one place in a program, its declaration or definition, whereas the name may be used in any number of places within the program text. Each time the name is encountered, the symbol table provides access to the information collected about the name during the compilation process.

From implementation point of view, a symbol table is a specialized database containing records for names and associated attribute, usually one record per name. A database consists of two parts: Database Manager (Symbol Table Manager) and Database Record Structure. The STM provides services to the client and separates and hides the particular implementation of the database record structure from the client.

Typical Symbol Table Manager services are:

- create a symbol table record structure
- install or insert a name record
- find or lookup for a name record
- update a record
- delete a record
- destroy the symbol table
- other auxiliary services: sort, print, pack.

Typical implementation of the symbol table record structure is:

- unordered or ordered liner structures: arrays and linked lists
- hierarchical structures: binary trees and hash tables
- combinations of linear and hierarchical structures

The design choice of a specific implementation depends mainly on the characteristics of the source language the compiler has to translate and to some extend on the compiler implementation language. In all cases, the two important leading criteria determining the design of the symbol table are space and speed. The speed is usually determined by the time require for two frequently used operations: adding a record and finding a record.