María José Moreno Serrano
morenoserranomariajose@gmail.com

Basic Implementation Techniques of Symbol Tables

The first consideration of symbol table implementation is how to find a free space and store a data in it. Depending on the number of names we wish to accommodate and the performance we desire, a wide variety of implementations is possible:

Unordered List : is the simplest possible storage mechanism. The only data structure required is an array, with insertions being performed by adding new names in the next available location.

Ordered List :If a list of names in an array is kept ordered, it may be searched using a binary search, which requires O(log(*n*)) time for a list of *n* entries. Insertion in an ordered array is a relatively expensive operation, in general. Thus ordered lists are typically used only when the entire set of names in a table is known in advance.

Binary Search Trees: are a data structure designed to combine the size flexibility and insertion efficiency of a linked data structure with the search speed provided by a binary search. On average, entering or searching for a name in a binary search tree built from random inputs requires O(log(*n*)) time.

Hash Tables*:* are probably the most common means of implementing symbol tables in production compilers and other system software. With a large enough table, a good hash function, and the appropriate collision-handling technique, searching can be done in essentially constant time regardless of the number of entries in the table.