Topic 14: Crafting A Compiler – Chapter 8: Symbol Tables

Jiri Mikulka, David Rusek
{xmikul39,xrusek02}@stud.fit.vutbr.cz

Abstract

Symbol tables are *data structures* that are used by compilers to hold information about source program constructs. A symbol table is a necessary component of a compiler because the definition of a name appears in only one place in a source program, its declaration, whereas the name may be used in any number of places within the source program. It is a mechanism that associates values (*attributes*) with *names*, therefore a symbol table is sometimes called a *dictionary*.

Two aspects of symbol tables are of interest to us: the operations associated with a symbol table, which are visible to other components of the compiler, and the implementation of those operations. The operations associated with a symbol table are:

- *search* (whether a name has been used),
- *insert* (add a new name),
- *delete* (remove a name when its scope is closed).

This talk is mainly concerned with implementation issues of a symbol table and its operations. Depending on the number of names we wish to accommodate and the performance we desire, a wide variety of implementations is possible:

- *unordered list*
  - good for a very small set of variables (but bad performance for a large number of variables)
  - easy coding
- *ordered list*
  - use binary search (but insertion and deletion are expensive operations)
  - relatively easy coding
- *binary search trees*
  - $O(\log n)$ time per operation (search, insert, delete) for n variables
  - relatively difficult coding
- *hash tables*
  - most commonly used (but performance may be bad if unlucky or the table is saturated)
  - very efficient memory allocation (the memory space is adequately larger then the number of variables)
  - not too difficult coding

In the talk we will show and explain the best practises for implementing symbol tables using hash tables.

References

[1] Charles N. Fischer and Ron K. Cytron and Richard J. LeBlanc Jr., *Crafting A Compiler*. Addison-Wesley, Massachusetts, 1st Edition, 2009.