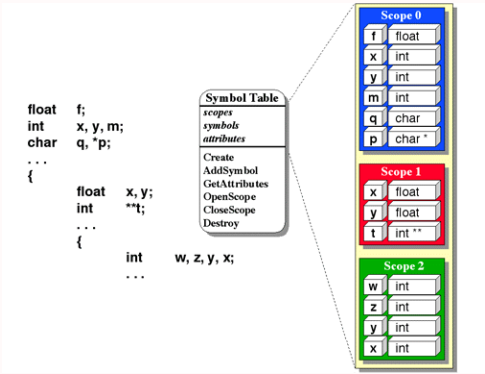
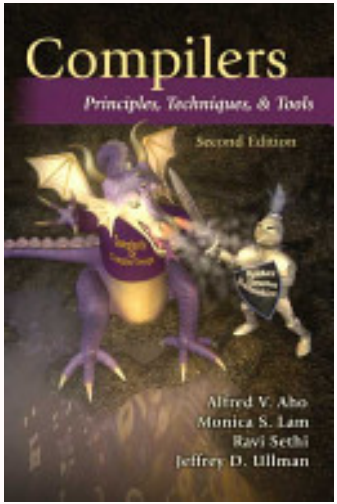
Symbol Tables in Language Translation

Brian A. Malloy
November 28, 2016



1. Resources


Compilers
Principles, Techniques, & Tools
Second Edition
Alfred V. Aho
Monica S. Lam
Ravi Sethi
Jeffrey D. Ullman

2. Symbol Table Definition

Data structure used by compilers to hold information about source-program constructs.[1]

1. Information is collected incrementally during analysis phase.
2. Symbol table entries contain information about an identifier: string, type, storage location, ...
3. Typically need to support multiple declarations of the same identifier.

[1] Compilers Principles, Techniques, & Tools, Aho, Lam, Sethi, Ullman, 2007

3. Who Creates an Entry

- The parser knows about the semantic structure of a program, and is in better position to distinguish different decls of an id.
- In some cases, a scanner can create a symbol table entry as soon as it sees the name.
- Only parser knows about type & scope.

4. Scope

The scope of a decl is the portion of a program to which the decl applies.[2]

- Typical implementation: separate symbol table for each scope.
- A program block has its own symbol table, with an entry for each decl in the block.
- A class has its own symbol table, with an entry for each data attribute and method.

[2] Compilers Principles, Techniques, & Tools, Aho, Lam, Sethi, Ullman, 2007

4.1. Block Structure

Consider the following code segment:
`{ int x; char y; { bool y; x; y; } x; y; }`

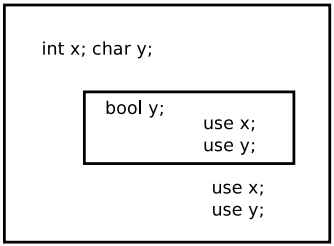
Let's rewrite the above segment with decls removed and each use is an identifier followed by its type:

```
{ { x:int; y:bool; } x:int; y:char; }
```

- This example illustrates nested scopes: the same identifier can appear in different scopes.
- Common names, such as i & x, can have multiple uses; subclasses can override method names.

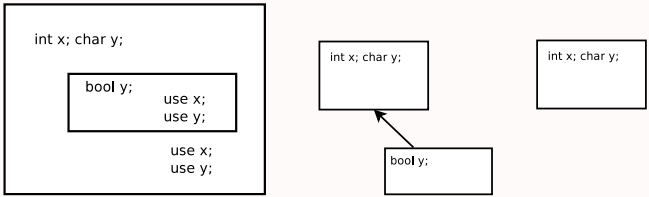
4.2. Symbol Tables & Scopes

- Symbol tables must permit multiple variable use
- Most closely nested rule: an id x is in the scope of the most-closely nested decl of x.
- The contour diagram below illustrates this nesting

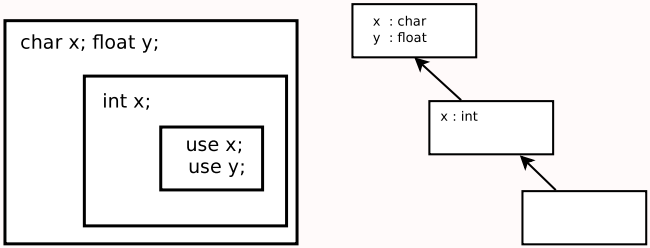


5. Chained Symbol Tables

```
{ int x; char y; { bool y; x; y; } x; y; }
```

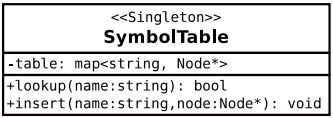


5.1. Another example of contour diagram and chain of symbol tables

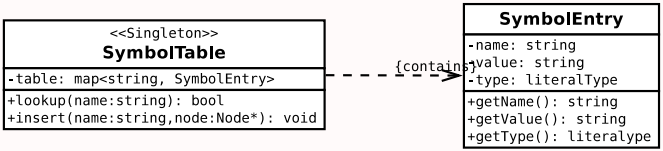


6. Implementation

Implementation of symbol table: `map<string, Node*>`



Alternative implementation of symbol table: `map<string, SymbolEntry>`



7. Use of Symbol Tables

- Role is to pass info from decl to use
- A semantic action puts info about an identifier into the symbol table
- A production such as $\mathsf{factor} \rightarrow \mathsf{id}$ gets information about an *id* from the symbol table
- Translation of E_1 op E_2 may depend on type of E_1 and E_2.