Lab Five

Brandon Litwin

Brandon.Litwin1@Marist.edu

March 26, 2019

1 Crafting A Compiler

8.1 What are the advantages and disadvantages of using a binary search tree and a hash table for scope tables?
An advantage of a binary search tree is its simple, widely used implementation. Each node of the tree is a stack of currently active scopes. If the tree is balanced, a symbol can be found or inserted in O(log n) time. The disadvantage of a binary search tree is it is slower than a hash table. The hash table's advantage is that it is faster than a binary search tree with search and insert operations taking O(1) time. The hash table's disadvantage is it's much more difficult to implement than a binary search tree. It requires a good hash function and proper collision-handling techniques.

8.3 Describe two alternative approaches to handling multiple scopes in a symbol table. (314 in book) List the actions required to open and close scopes in each alternative. Trace the sequence of actions that would be performed for each alternative during compilation of program in figure 8.1.

Option 1: Individual Table for Each Scope
In this approach, each scope as its own table. Every time a new scope is found, a new table is created. Scopes are opened and closed in a last-in, first-out stack, so the innermost scope is at the top of the stack. When a new scope is opened, it is pushed onto the stack, and when a scope is closed, the stack is popped.

Symbol Table Sequence

Scope 0: f(float,float,float), g(int)
Creates new scope
Scope 1: w = int, x = int
Creates new scope
Scope 2: x = float (not the same as x in scope 1), z = float
f(x,w,z) uses the x and z declared in scope 2 and the w declared in scope 1
Scope 2 is closed
g(x) uses the x declared in scope 1
Scope 1 is closed

Scope 0 is closed

Option 2: One Scope Table
In a single scope table, each variable is assigned a depth value that is equal to the level of scope it is in. When a new scope is opened, it is kept track by assigning the depth level to it, which increases each time a new scope is opened. When a scope is closed, each variable at that scope is removed from the hash table and the depth decreases by 1.
Symbol Table Sequence
Depth 0: f(float,float,float), g(int) added to hash table
Creates new scope
Depth 1: w = int, x = int added to hash table
Creates new scope
Depth 2: x = float (not the same as x in scope 1), z = float. X is assigned depth 2 and hashed differently from the x at scope 1.
f(x,w,z) uses the x and z declared in scope 2 and the w declared in scope 1
Scope 2 is closed and all depth 2 variables are removed from the hash table
g(x) uses the x declared in scope 1
Scope 1 is closed and all depth 1 variables are removed from the has table
Scope 0 is closed and all depth 0 variables are removed from the hash table

The program:
```
import f(float, float, float)
import g(int)
{
  int w,x
  {
    float x,z
    f(x,w,z)
  }
  g(x)
}
```