Lynne Coblammers
EECS 665
Lab 7 Report
2015.11.01

This lab focused on symbol tables. Several stub functions were filled in to build and print a symbol table from a parser for a c-like language. This lexer and parser for this lab were provided. Three functions that needed to be filled in were located in sem.c, including enterblock, leaveblock, and dump required definitions. The other functions that needed to be defined were in sem_sym.c, and included fname, ftail, blockdcl, and btail.

At a high level, the symbol tables work by entering a new block every time a new scope is entered. New scopes included the function header, the function body, and blocks delimited by curly braces, such as the body of while, for, or if statements. All variables and names declared within that scope are added to the symbol table with the current level. When a scope is exited, the symbols defined in that scope are printed to the screen and removed from the symbol table. In this way, the symbol table can keep track of what symbols have been defined and are within the current scope for later use.

Much of the code for the symbol tables was provided. One function defined for this lab was enterblock. When a new scope is entered, the enterblock function is called. This function was defined to increment the global level variable, which essentially keeps track of nested scopes, and then calls the new_block function, which saves the previous stack top and marks a new stack top. Since enterblock must be called at the beginning of each new scope, it is called in several places. First, it is called from the main function of the parser to allow globally available names, such as do, while, return, etc. to be entered into the symbol table before entering any other scopes. However, as mentioned, it also needs to be called when a function header is written or the beginning of a block is seen. When these happen, the parser calls fname and blockdcl, respectively. Therefore, those functions were defined to call enterblock().

Another important function was the leaveblock function. This is called as a scope is exited, such as at the end of a function or block. This function was defined to call dump, which prints the symbols defined in the scope that is being left. Next, the actual symbols needed to be removed from the table and the memory that was allocated for them in the table was freed. Finally, exit_block is called, which frees semantic records and moves the stack top pointer, and the level is decremented. Since the leaveblock function needs to be called at the end of a function or a block, a call to it was added to the definitions of ftail and btail, which are called by the parser at the close of a function or a block. The function ftail actually needed to call leaveblock twice since it is both exiting the function body scope and the scope of the function parameters. It is also called directly by the parser's main function once parsing has been completed. One challenge encountered in defining leave block was determining how to update the symbol table. Initially, it was left alone, which meant that the old symbols were still present and printed after the scopes they had been defined in had already been left. Therefore, it was determined those records needed to be deleted and the next in the link needed to be the new entry for that index of the table.

The final function to implement was dump. This function was defined to iterate through the members of the symbol table and print any that had a level greater than or equal to the current level (i.e. symbols that were falling out of scope). This was the most challenging function to implement, but the function sdump, which prints the string table, was a good reference to figure out how to traverse the symbol table using pointer arithmetic. However, since the symbol table was implemented as a hash table, not every entry in the table had legitimate data, so dereferencing null pointers had to be avoided by checking the value of the current entry before trying to access it.

Once the all the functions were filled in with definitions, the program was tested using the provided test files. Both worked as expected, and the symbol tables printed correctly.