The open hash table algorithm present in SymbolTable.cpp consisted of a set number (HASH\_TABLE\_SIZE) of buckets, each with a pointer to a Data object. Each Data object is characterized by an ID, a line number, a scope number, and a pointer to another Data object to account for collisions. That way, Data objects with the same hashed id could be stored in a linked list of Data objects, with the very last one having a null Data pointer. The hash table itself contains an array of HASH\_TABLE\_SIZE buckets, along with a hash function to convert a string id into a positive integer that is less than HASH\_TABLE\_SIZE using the modulus operator and the STL functional algorithm. The SymbolTableImpl class, aside from the necessary functions, contains a pointer to the hash table being used, an int m\_scope private variable to keep track of the scope count, and a vector containing pointers to each Data pointer in the hash table.

The time complexity of Find is on average O(logN) and at worst O(N). The time complexity of Declare was on average O(logN) and at worst O(N). Both enterScope and exitScope are linear.

**Pseudocode:**

**Insert:**

*Get hash id*

*Make pointer to that position on the array*

*If pointing at an empty space, add the new data*

*While the space isn’t empty…*

*Move down the linked list*

*Add new data*

**Search:**

*Get hash id*

*Make pointer to that position in the array*

*While not pointing at an empty space*

*If found, save line and scope*

*Go to the next item*

*If nothing found, return false*

*Return true*

**Enter scope** is trivial – when a scope is entered, push back a nullptr onto the vector of pointers to Data pointers and increment the scope.

**Exit scope:**

*Get the size of the vector*

*While the size is greater than zero…*

*Decrease the size*

*If the value of the vector at that point is nullptr, break*

*Delete the value there and set it equal to nullptr*

*Pop back the vector*

*If the vector is empty, return false*

*Pop back the vector (removing the scope entry marker)*

*Decrease scope*

*Return true*