Jiunn Siow

Project 4 Report

1. In this program, I used an open hash table with a doubly-linked lists. Because of the large number of items that the program must process, an open hash table is the most efficient way to search through the items. I made a Node for each “bucket” of the hash table so that all the items in the hash table may be linked together.
2. The following time complexities of the functions:
   1. enterScope is linear in the number of identifiers going into the scope.
   2. exitScope is linear in the number of identifiers exiting the scope
   3. declare is linear in the number of identifiers in the scope
   4. find is constant in the number of identifiers it can access
3. Pseudocode for non-trivial functions
   1. int hashFunc(const string& id)

{

Unsigned int value is equal to 0

For loop while i is less than id.size()

{

Value is equal to value plus (i + 1) times id[i]

}

Value is equal to value % size

Return value

* 1. bool hashing::declare(string id, int lineNum)

{

Int key is equal to hashFunc(id)

hashNode\*\* ptr is equal to &table[key]

while loop ptr does not reach nullptr

{

If the id of \*ptr is equal to id inputted into function

Return false

Else

Ptr is equal to &((\*ptr)->next)

}

\*ptr is equal to new hashNode(id, lineNum)

Return true

}

* 1. int hashing::search(string id)

{

Int key is equal to hashFunc(id)

hashNode\* ptr is equal to table[key]

while loop ptr is not equal to nullptr

{

if the id of ptr is equal to the inputted id

int num is equal to the lineNum of ptr

return num

}

Ptr equals ptr’s next node

Return -1

}

1. Some bugs that I ran into whilst creating the program involved exception errors, in which the linked list of each bucket was not properly handling the large amount of test values. Another trouble I had was implementing the hash table, as initially my code only ran my linked lists and the code written for my hash table did nothing. I fixed this by creating a separate class for my hash table and calling functions from it through my symbol implementation.