Bangladesh University of Engineering & Technology
Department of Computer Science & Engineering
CSE 310: Compiler Sessional
Session January 2018

Assignment 1
Implementation of Symbol Table

April 1, 2018

1 Introduction

The purpose of this course is to construct a simple compiler. In the first step to do so, we are going to implement a symbol-table. A *symbol-table* is a data structure maintained by compilers in order to store information about the occurrence of various entities such as identifiers, objects, function names etc. Information of different entities may include type, value, scope etc. At the starting phase of constructing a compiler, we will construct a symbol-table which maintains a list of hash tables where each hash table contains information of symbols encountered in a scope of the source program.

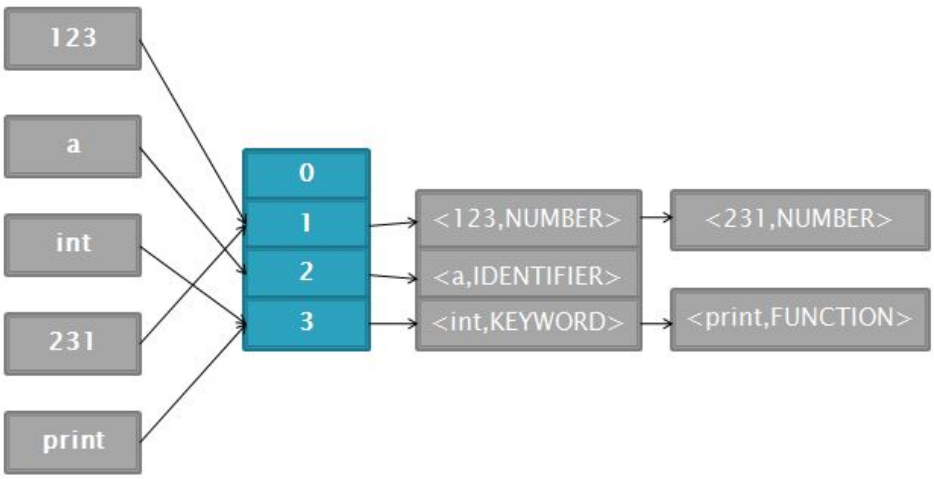

Figure 1: Symbol Table using hashing.

2 Symbol Table

In this assignment we will construct a symbol table that can store type and scope information of a symbol found in the source program. If the source program consists of a single scope then we can use a hash table to store information of different symbols. Hashing is done using the symbol as key. Figure 1 illustrates such a symbol table. Now consider the follwoing C code.

```
1. int a,b,c;
2. int func(int x){
3.    int t=0;
4.    if(x==1){
5.        int a=0;
6.        t=1;
7.    }
8.    return t;
9. }
10.int main(){
11.    int x=2;
12.    func(x);
13.    return 0;
14.}
```

To successfully compile this program, we need to store the scope information. Suppose that we are currently dealing with line no. 5. In that case, global variables a,b and c, function parameter x , function func's local variable t and the variable a declared within the if block are visible. Moreover the variable a declared within the if block hides the global variable a. How we can store symbols in Symbol Table which can help us to handle scope easily? One way is to maintain a separate hash table for each scope. We call each hash table a Scope Table. In our symbol table, we maintain a list of scope table. You can also think the list of scope table as a stack of scope table. Whenever we enter in a new block, we create a new scope table and insert it in the top of the list. Whenever we found a new symbol within that block of the source code, we insert it in the newest scope table i.e. the scope table at the top of the stack. When we need to get the information of a symbol, at first we will search at the topmost scope table. If we could not find it there, we would search in the next scope table and so on. When a block ends we simply pop the topmost scope table. Suppose we give a unique number to each block, 1 to global scope, 2 to the function func, 3 to the if block and 4 to the main function. Figure 2 illustrates the state of symbol table when we are in line no 6 of the given code.

3 Tasks

You have to implement following three classes.

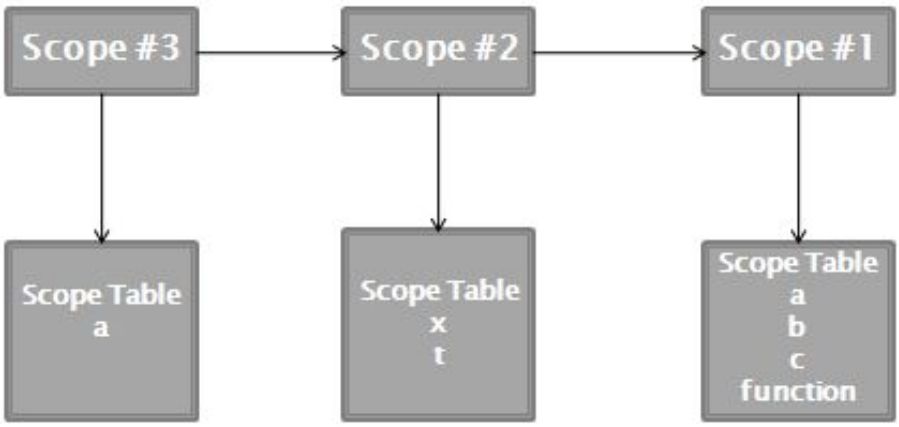

Figure 2: Symbol Table with scope handling.

- **SymbolInfo:** This class contains the information regarding a symbol faced in the source program. In the first step we will limit ourselves to only two members. One is for the **Name** of the symbol and other is the **Type** of the symbol. You can use C++ stl string. These two member variables will be private and you should write getter setter method as needed. This class will also contain a pointer to a SymbolInfo object as you need to implement chaining mechanism to resolve collision in hash table. Also keen in mind that you may have to extend this class as we progress to develop our compiler.

- **ScopeTable:** This class implements a hash table. You may need an array (array of pointers of SymbolInfo type) and a hash function (a decent one). You will also need a pointer of ScopeTable type object named *parentScope* as a member variable so that you can maintain a list of scope tables in the symbol table. Also give each each table a unique id. Then you need to add the following functionalities.

  - **Insert:** Insert into symbol table if already not inserted in this scope table. Return type of this function should be boolean indicating whether insertion is successful or not.
  - **Look up:** Search the hash table for particular symbol. Return a SymbolInfo pointer.
  - **Delete:** Delete an entry from the symbol table. Return true in case of successful deletion and false otherwise.
  - **Print:** Print the scope table in the console.

  You should also write a constructor that takes an integer n as a parameter and allocates n buckets for the hash table. You should also write a destructor to deallocate memory.

- **SymbolTable:** This class implements a list of scope tables. The class should have a

pointer of ScopeTable type which indicates the current scopetable. This class should contain the following functionalities:

- **Enter Scope:** Create a new ScopeTable and make it current one. Also make the previous current table as its parentScopdeTable.
- **Exit Scope:** Remove the current ScopeTable.
- **Insert:** Insert a symbol in current ScopeTable. Return true or false.
- **Remove:** Remove a symbol from current ScopeTable. Return true or false.
- **Look up:** Look up a symbol in the ScopeTable. At first search in the current ScopeTable, if not found then search in its parent ScopeTable. Return a pointer of SymbolInfo type.
- **Print Current ScopeTable:** Print the current ScopeTable.
- **Print All ScopeTable:** Print all the ScopeTables currently in the SymbolTable.

4 Input

The first line of the input is a number indicating number of buckets in each hash table. Each of the following line will start with a code letter indicating the operation you want to perform. The letters will be among 'I', 'L', 'D', 'P', 'S' and 'E' . 'I' stands for insert which is followed by two space separated strings where the first one is symbol name and the second one is symbol type. As you already know, the symbol name will be the key of the record to be stored in the symbol-table. 'L' means lookup which is followed by a string containing the symbol to be looked up in the table. 'D' stands for delete which is also followed by a string to be deleted. 'P' stands for print the symbol table which wil be followed by another code letter which is either 'A' or 'C'. If 'A' is followed by 'P' then you will need to print all the scope tables otherwise you will print only the current scope table. Finally, 'S' is for entering into a new scope and 'E' is for exiting the current scope. You should also check the sample input file.

5 Output

Check the sample output file.

6 Important Notes

Please try to follow the instructions listed below while implementing your assignment:

- Implement using C++ programming language

- Avoid hard coding
- Use dynamic memory allocation
- Take input from file. You may output both in console and file.
- Try to get accustomed to a Linux platform

7 Rules

- You have to submit all your source codes via moodle. All the file name will be in following format

  <your 7 digit student id>_<additional name>

  For example, the submitted file name would look like 1505999_symboltable.cpp if it is submitted by a student having 1505999 as student id. Put all your source files in a folder (even if you put all your codes in only one file) named after your 7 digit student id and create a **zipped** archive of the folder. Then submit the zip file in moodle.

- Any type of plagiarism is strongly forbidden. -100% marks will be given to the students who will be found to be involved in plagiarism. It does not matter who is the server and who is the client. Another important note is, in a 0.75 credit course, a single penalty can make a huge impact.
- Prepare for an online evaluation.

8 Deadline

Deadline is set at 11:55 pm, April 15, 2018 for all lab groups.