University of Asia Pacific

Department of Computer Science & Engineering

CSE 430: Compiler Design

Lab2: Symbol Table

Introduction

We will build a symbol-table in this assignment. At this initial stage, we will omit many details regarding an actual symbol-table and we will simply adhere to the basic concept that "a symbol-table is an efficient data-dictionary for the symbols used in a program". Thus, our focus in this assignment is to construct a simple hash-based data-dictionary based on chaining.

Symbol table attributes:

- **NAME**
- **TYPE**
- **SIZE**
- **DIMENSION**
- **Line of Code**
- **Address**

Inputs

Name will be the name of the element

Type will be the type of the element

Size will be the size of the type.

Dimension will be dimension of the type**.**

Line of Code will be inserted randomly, you can insert it as u wish.

Address will be also random, you can insert it as u wish.

The input to your program will be a sequence of six tuples, where each element in each tuple is a string.
An example of input sequence is given below:

- x, ID, 2, 1, 5, 0x0FF

The first element of each tuple will be the name of the record to be stored in the symbol table. Hence, you have to apply the hash function on the first element of each tuple that is the name.

Tasks

Implement the following functionalities:

1. ***Insert*** a new symbol/name along with its type into the symbol table
2. ***Search*** a symbol/name along with its type from the symbol table
3. ***Delete*** a symbol/name along with its type into the symbol table
4. ***Show*** the contents of symbol table in the console
5. ***Update*** an already existing entry in the symbol table

Implementation Details

To implement the tasks mentioned in the previous section, you need to do the followings:

1. Declare the followings:
    (a) *struct SymbolInfo* : The definition of this structure will grow gradually throughout the development of this project. For this assignment, we simply need two members, one for storing the symbol (e.g. \x") and another for storing the type of the symbol (e.g. IDENTIFIER).
    (b) *SymbolInfo *SymbolTable[MAX]* : Since our symboltable will be a hashtable based on chaining, we will have to start with an array of pointers where each pointer points to a list of nodes of type *struct SymbolInfo*. *SymbolTable* is such an array of pointers. For this assignment, the choice of the size of this array, MAX as well as of the hash function is left upto you.
2. In addition to this array of pointers, define the following global functions:
    (a) insert()
    (b) search()
    (c) delete()
    (d) show()
    (e) update()
    (f) getHashKey()

Additional Task (If you can, not mandatory): You can try linking the exercise of Lexical Analyzer with the symbol table where tokens generated by the lexical analyzer will be stored in the symbol table.