Symbol Table

Link github: https://github.com/914-Mathe-Andrei/lftc/tree/main/lab2/src

I. Introduction

The entire implementation is written in python v3.10.

I decided to implement two different symbol tables for identifiers and constants, respectively. Both of them use a hash table as a data structure, therefore I created two python modules: SymbolTable.py, HashTable.py. There is an additional python module named Type.py where all the possible types that exists in the programming language are defined.

II. Details

a) Type.py

The module contains an enum named *Type* that contains two types supported by the programming language: *INT* and *STR*. The enum is used to easily perform the business logic and store the type of the identifiers and constants within the program.

b) HashTable.py

The module defines a general purpose *HashTable* class that implements the hast table data structure supporting all the CRUD operations. It takes advantage of the python's magic methods like *__setitem__*, *__getitem__*, *__delitem__* and *__contains__*. The hash function used by the data structure utilizes the built-in hash function implemented by the python's *object* class applied over the key modulo the size of the array of the hash table. The class also overrides the *__str__* function to print and debug the hash table with ease.

The error handling is implemented in the *HashTable* class.

c) SymbolTable.py

The module defines two auxiliary data structures, *Identifier* and *Const*, and the two types of symbol tables, *IdentiferSymbolTable* and *ConstSymbolTable*. The auxiliary data structures are internal representations of the identifiers and constants, respectively. The symbols tables are similar, they both use the *HashTable* class as a

data structure and implement the insert, get, delete and contains operations. The symbol table for the identifiers defines an additional update operation because the values of the identifiers can be changed.

The *IdentifierSymbolTable* represents a mapping from the identifier, which is a string, to an *Identifier* instance. The *ConstSymbolTable* represents a mapping from the constant, being it an integer or a string, to a *Const* instance.

The two symbol tables have the __*str*__ method overriden and implement error handling.