https://github.com/StefanCsPurge/Formal-Languages-and-Compiler-Design

Documentation

Scanner (lexical analyzer) that uses a symbol table (ST) and a program internal form (PIF).

Input: Programs p1/p2/p3/p1err and token.in

Output: PIF.out, ST.out, message “lexically correct” or “lexical error + location”

Implementation: Python

Symbol Table

Based on: custom Hash Table, i.e., a Python list with a linked list on each position

Hash function: Sum of ASCII codes of chars of the elements **%** Hash table size (to have a result in the size boundary)

Conflicts are solved by the linked lists: when 2 elements have the same hash function result, the new one is added next to the old one in that list.

Operations of the Symbol Table:

- add(element) – add element and return it’s exact position
- contains(element) – return true or false if the element is/isn’t in the symbol table
- getPosition(element) – return the exact position of the element: tuple of list position and the position inside that list
- remove(element) – eliminate the given element from the symbol table

Scanner

Class which contains all the tokens, the symbol table and the PIF (as a list of <token, position> pairs). The position is also a pair that locates the token in the ST, or (-1,-1) if the token is a keyword/separator/operator. When an object of this class is created, it automatically reads the predefined tokens from the *token.in* file.

The scanning is started with the ***runScanner(programFile)*** method. It parses the program file line by line, eliminating unnecessary spaces and comments and getting the tokens (even if there is no space between them).

After detecting the tokens, it adds them to PIF and the ST accordingly, or prints a message in case of any ***Lexical error***.

After the scan is completed, the ST and the PIF are also saved in their corresponding files (ST.out and PIF.out).