Link to GitHub: https://github.com/911-Blajan-Denisa/FLCD.git

Symbol Table

I chose to make the Symbol Table as a Hash Table.
The symbol table contains a list and the size of the list, in which the elements will be stored.
The hash function is the sum of all the ASCII codes of the given key modulo the size of the table.
Using this representation, the conflicts are resolved such that when two elements having the same hash, they will both be added to the same list.

Operations

- contains(key) — checks if the key exists in the symbol table, it returns true if it does, false otherwise
- insert(key) — adds the given key into the symbol table
- remove(key) — deletes the given key from the symbol table
- hash(key) — returns a number that is the index of the symbol table list in which the element will be stored
- get_position(value) — gets the position from the symbol table

Program Internal Form

The Program Internal Form is defined as a list of pairs - (token, position), where position is -1 for reserved words, operators and separators.
The identifiers and constants will be defined like (hash from symbol table, position in the corresponding list).

Scanner

The algorithm splits each line of the program into tokens, builds the token character with character.
If it is s a constant or identifier, it looks up for its position in the ST, if it is a separator, an operator or a reserved word, its position is -1.
In the PIF we won't have the name of the identifier ot value of the constant, we will have "identifier" or "constant".
If the token is none of the above, that means there is a lexical error at that line and the error with its line will be displayed.

Example

PROBLEM:

```
main program
[
        | dek n, x := 1, i, p := 1 |
        | lasa n |
        | now i := 1, i <= n, i := i + 1 |
        [
                | p := p * x |
                | x := x + 2 |
        ]
        | skriva p |
]
```

ST:

```
0 -> ['x']
1 -> ['1']
2 -> ['n', '2']
3 -> ['i']
4 -> ['p']
5 -> []
```

PIF:

```
main -> -1
program -> -1
[ -> -1
| -> -1
dek -> -1
identifier -> (2, 0)
, -> -1
identifier -> (0, 0)
:= -> -1
constant -> (1, 0)
, -> -1
identifier -> (3, 0)
, -> -1
identifier -> (4, 0)
:= -> -1
constant -> (1, 0)
| -> -1
| -> -1
lasa -> -1
identifier -> (2, 0)
| -> -1
| -> -1
now -> -1
identifier -> (3, 0)
:= -> -1
constant -> (1, 0)
, -> -1
identifier -> (3, 0)
<= -> -1
identifier -> (2, 0)
, -> -1
identifier -> (3, 0)
:= -> -1
identifier -> (3, 0)
+ -> -1
constant -> (1, 0)
| -> -1
[ -> -1
| -> -1
identifier -> (4, 0)
:= -> -1
identifier -> (4, 0)
* -> -1
identifier -> (0, 0)
| -> -1
| -> -1
identifier -> (0, 0)
:= -> -1
identifier -> (0, 0)
+ -> -1
constant -> (2, 1)
| -> -1
] -> -1
| -> -1
skriva -> -1
identifier -> (4, 0)
| -> -1
] -> -1
```