Link to the github:https://github.com/Fabianozaur/FLCD/tree/main/lab3

## Statement:

Implement a scanner (lexical analyzer): Implement the scanning algorithm and use ST from lab 2 for the symbol table.

Input: Programs p1/p2/p3/p1err and token.in (see Lab 1a)

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

Deliverables: input, output, source code, documentation

Details:

ST.out should give information about the data structure used in representation

If there exists an error the program should give a description and the location (line and token)

## Scanner class

Using the function called tokenizeFunction the algorithm will go on each line of the code and separate all the tokens checking if they are part of  an operator or a separator and sends the tokens in main where they are added in the PIF on position (-1,-1) or if they are and idetifier or constat added in ST and then after in PIF on the positon recived from the Symbol Table.

The function checkIfIdentifier of checkIfConstant use regex to determine the constants and the identifiers.

## ProgramInternalForm class

It contains the function addToken to add the tokens on a specific given position in the pif and the function str for the printig of the class

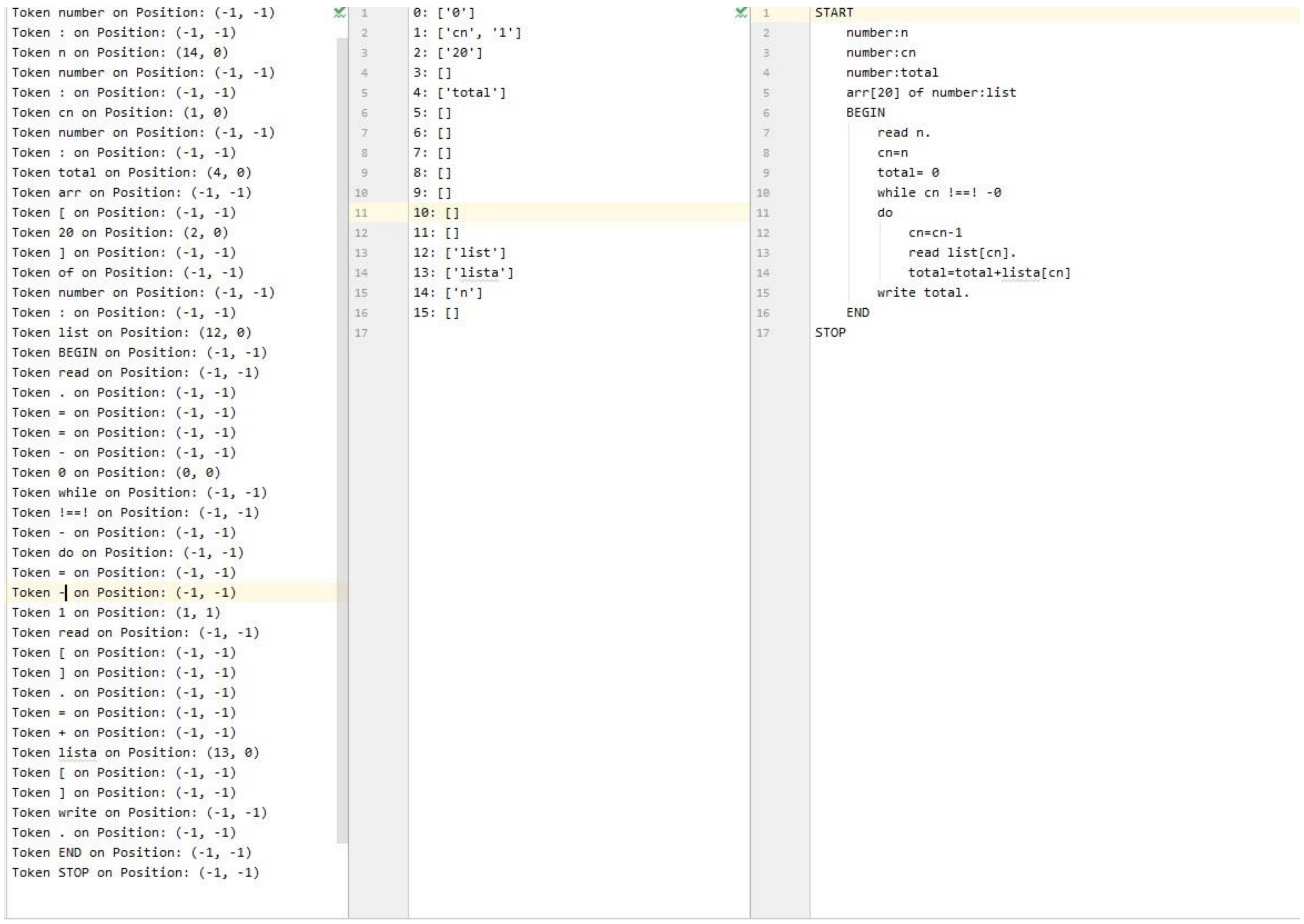
## SymbolTable class

The Symbol table class is implemented using a hash table with linked listed as a conflict resolution method.

The hash function taking the sum of the ASCII values of each character of the key and applying modulo by the capacity of the table.

The add function uses the hash function to add the index at a position in the linked list.

## Tests



```
Token number on Position: (-1, -1)
Token : on Position: (-1, -1)
Token n on Position: (14, 0)
Token number on Position: (-1, -1)
Token : on Position: (-1, -1)
Token cn on Position: (1, 0)
Token number on Position: (-1, -1)
Token : on Position: (-1, -1)
Token total on Position: (4, 0)
Token arr on Position: (-1, -1)
Token [ on Position: (-1, -1)
Token 20 on Position: (2, 0)
Token ] on Position: (-1, -1)
Token of on Position: (-1, -1)
Token number on Position: (-1, -1)
Token : on Position: (-1, -1)
Token list on Position: (12, 0)
Token BEGIN on Position: (-1, -1)
Token read on Position: (-1, -1)
Token . on Position: (-1, -1)
Token = on Position: (-1, -1)
Token = on Position: (-1, -1)
Token - on Position: (-1, -1)
Token 0 on Position: (0, 0)
Token while on Position: (-1, -1)
Token !==! on Position: (-1, -1)
Token - on Position: (-1, -1)
Token do on Position: (-1, -1)
Token = on Position: (-1, -1)
Token - on Position: (-1, -1)
Token 1 on Position: (1, 1)
Token read on Position: (-1, -1)
Token [ on Position: (-1, -1)
Token ] on Position: (-1, -1)
Token . on Position: (-1, -1)
Token = on Position: (-1, -1)
Token + on Position: (-1, -1)
Token lista on Position: (13, 0)
Token [ on Position: (-1, -1)
Token ] on Position: (-1, -1)
Token write on Position: (-1, -1)
Token . on Position: (-1, -1)
Token END on Position: (-1, -1)
Token STOP on Position: (-1, -1)
```

```
0: ['0']
1: ['cn', '1']
2: ['20']
3: []
4: ['total']
5: []
6: []
7: []
8: []
9: []
10: []
11: []
12: ['list']
13: ['lista']
14: ['n']
15: []
```

```
START
    number:n
    number:cn
    number:total
    arr[20] of number:list
    BEGIN
        read n.
        cn=n
        total= 0
        while cn !==! -0
        do
            cn=cn-1
            read list[cn].
            total=total+lista[cn]
        write total.
    END
STOP
```