Lab 4 – Andreica Daniel-Vladut

GitHub repository: https://github.com/Andreica-Daniel/FLCD

Language Specification

Alphabet:

a. Upper (A-Z) and lower case letters (a-z) of the English alphabet
b. Decimal digits (0-9);
c. Underline character '_';

Lexic:

a.Special symbols, representing:
- operators + - * / % == < <= = >=
- separators {} () [] space
- reserved words: str elif else if int bool while print input for and or in not

b.identifiers
-a sequence of letters and digits, such that the first character is a letter; the rule is:
 identifier ::= letter | letter{letter}{digit}
 letter ::= "A" | "B" |...| "Z"|"a"|"b"|...|"z"
 digit ::= "0"|non_zero_digit
 non_zero_digit ::= "1" |...| "9"

c.constants
1.integer
 number ::= non_zero_digit{digit}|digit
 integer ::= [sign] number|zero
 sign::=+|-
 zero::=0

2.string
 string::=’{letter|digit}’

Syntax:

The words - predefined tokens are specified between " and ":
Sintactical rules:
type ::= "bool" | "str" | "int"
cmpdstmt ::= "{" stmtlist "}"
stmtlist ::= stmt | (stmt stmtlist)
stmt ::= simplstmt | structstmt
simplstmt ::= assignstmt | iostmt
assignstmt ::= identifier "=" expression
expression ::= expression ("+" | "-") term | term

term ::= term "*" factor | factor
factor ::= "(" expression ")" | identifier
iostmt ::= "input" "(" [text] ")" | "print" "(" identifier ")"
text ::= {string [space]}
structstmt ::= cmpstmt | ifstmt | whilestmt | forstmt
ifstmt ::= "if" condition ":" stmt ["elif" condition ":" stmt]["else ":" stmt"]
whilestmt ::= "while" condition ":" stmt
forstmt ::= "for" stmt ":"
condition ::= expression relation expression
relation ::= "<" | "<=" | "==" | "!=" | ">=" | ">"

Classes and files structure

Main file:

- Initializes the Scanner, gets every token from the given file and classify it and then checks if there is a lexical error or not.
- Write the content in “ST.out” and “PIF.out”.

Pair class

- Pair(key, value)
- Getters and setters for key and value attributes.

SymbolTable class

- SymbolTable()
- Methods:

```
        - checkIfElementExists(self, pair):
            '''
            Check if a key exists in the sorted symbol table.
            input: pair - Pair(key, value)
            return: position of the key, if it already exists
                   -1, otherwise
            '''

      - addElement(self, token):
            '''
            Adds a new element in the symbol table.
            input: token - the token to be added
            return: None, if that key already exist
            '''

      - getList(self):
            '''
            Returns the sorted symbol table.
            input: -
            return: sorted symbol table
            '''
```

Scanner class

- Scanner(excercise)
- Methods:
  - addToPIF(self, classification, token, index): creates a Pair and adds it to the PIF
  - getPIF(self): returns the PIF
  - nextToken(self): returns the next token
  - codifyToken(self, currentToken): codifies the token if it is an identifier/constant
  - classifyCodification(self, code): classifies the code (identifier/constant/reserved-word/separator/operator) and returns 'lexical error' if none of them is true

Test Cases

Input file contents:

```
a = input ( )
b = input ( )
c = input ( )
if a < b and a < c :
{ smallest = a }
elif b < a and b < c :
{ smallest = b }
else :
{ smallest = c }
print ( smallest )
```

PIF output file:

```
identifier a 1
operator = -1
reserved-word input -1
separator ( -1
separator ) -1
identifier b 2
operator = -1
reserved-word input -1
separator ( -1
separator ) -1
identifier c 3
operator = -1
reserved-word input -1
separator ( -1
separator ) -1
reserved-word if -1
identifier a 1
operator < -1
```

```
identifier b 2
reserved-word and -1
identifier a 1
operator < -1
identifier c 3
separator : -1
separator { -1
identifier smallest 4
operator = -1
identifier a 1
separator } -1
reserved-word elif -1
identifier b 2
operator < -1
identifier a 1
reserved-word and -1
identifier b 2
operator < -1
identifier c 3
separator : -1
separator { -1
identifier smallest 4
operator = -1
identifier b 2
separator } -1
reserved-word else -1
separator : -1
separator { -1
identifier smallest 4
operator = -1
identifier c 3
separator } -1
reserved-word print -1
separator ( -1
identifier smallest 4
separator ) -1
```