**Lexic.txt**

1. Alphabet:
   1. Upper (A-Z) and lower case letters (a-z) of the English alphabet
   2. Underline character '\_';
   3. Decimal digits (0-9);
2. Seperators: - (), [], {}, “”, ‘’, ``, white space
3. Not include preprocessor directives
4. Comment - //
5. Operators: ["+", "-", "%", "/", "\*", "=", ">", "<", ">=", "<=", "==", "||", "&&"]
6. Reserved words: [int, float, boolean, print, array, main, return, for, Add, while, if, else if, else]
7. Identifiers: Only letters, digits and underline (\_) are allowed. Always starts with a letter. Case sensitive.

identifier ::= letter | letter{letter}{digit}{underline}

    letter ::= "A" | "B" | . ..| "Z"

    digit ::= "0" | "1" |...| "9"

underline = “\_”

1. Constants:

Integer: only digits

Character: ‘letter’|’digit’

String: char{string}

1. A backslash is used(\) to do some special actions depends on what character is going after.

\n – new line

\t – creates a horizontal tab

\\ - Inserts a backslash character (\)

\” - Inserts a double quote character

**Sytax.txt**

1. declaration : Type identifier
2. Start: main() function
3. Data type : “int” | “string” | “char” | “boolean” | “float”
4. Array decleration: “array” identifier “[number]” “of” type
5. Assignment: identifier ”=” expression
6. expression = expression "+" term | term
7. term ::= term "\*" factor | factor
8. factor ::= "(" expression ")" | IDENTIFIER
9. structstmt ::= cmpdstmt | ifstmt | whilestmt | forstmt
10. if statement : "if" “(“ condition”)” "{" stmt “}” [“else if” “(“ condition“)” “{” etmt”}” ]["else " “{”stmt “}”]
11. condition ::= expression RELATION expression
12. Relation: “||” | “&&” | “== “| “>= “| “>” | “<” | “<=”
13. iostmt ::= "input" | "print" "(" IDENTIFIER ")"
14. forstmt: “for” “(” condition “)” “{“ statement“}”
15. whilestmt: “while” “(” condition “)” “{” statement“}”
16. Variable name cannot be started with number
17. Reserved word Add to import library
18. Function: type identifier ”(” arguments “)” {statement “return ” identifier | type}

import re

from tkinter import W

input\_program = open("C:/Users/user/Documents/Semester#7/FLACT/LAB2/program.txt", "r")

tokens = open("C:/Users/user/Documents/Semester#7/FLACT/LAB2/tokens.txt", "w")

dataTypes = {"int", "string", "float", "const", "char"}

keywords = {"for", "if", "else", "Add", "return", "print", "while", "main", "array"}

operators = {"+", "-", "%", "/", "\*", "=", ">", "<", ">=", "<=", "==", "||", "&&"}

separators = {";", "[", "]", "{", "}", "(", ")", ","}

letters = "AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz"

digits = "1234567890"

special\_quote = {"t", "n", "\\", '"'}

keywords1 = {}

operators1 = {}

separators1 = {}

identifiers = {}

constantas = []

errors = []

backslash  = []

comments = []

for line in input\_program:

    temp = False

    isConst = False

    temp\_arr = []

    find\_string\_const1 = False

    find\_string\_const2 = False

    temp\_str = ""

    if(line.find("//") != -1):

        comments.append(line[line.find("//"):-1])

    line = line[:line.find("//")]

    if(line):

        for l in line:

            if (l == "'"):

                if(find\_string\_const1):

                    find\_string\_const1 = False

                    temp\_str += l

                    temp\_arr.append(temp\_str)

                    line = line.replace(temp\_str, '')

                    temp\_str = ''

                elif(find\_string\_const2 == False):

                    find\_string\_const1 = True

            if (l == '"'):

                if(find\_string\_const2):

                    find\_string\_const2 = False

                    temp\_str += l

                    temp\_arr.append(temp\_str)

                    line = line.replace(temp\_str, '')

                    temp\_str = ''

                elif(find\_string\_const1 == False):

                    find\_string\_const2 = True

            if (find\_string\_const1 or find\_string\_const2):

                temp\_str += l

                continue

    words = re.split("\s", line)

    words = filter(lambda x: x != "", words)

    for word in list(words):

        str = ""

        d = ''

        for w in word:

            if (w in separators and w != '' ):

                if(str != ''):

                    temp\_arr.append(str)

                temp\_arr.append(w)

                w = ''

                str = ""

            if (w in operators and w != ''):

                if(str != ''):

                    temp\_arr.append(str)

                temp\_arr.append(w)

                w = ''

                str = ""

            str += w

        temp\_arr.append(str)

    for word in temp\_arr:

        if (word in dataTypes):

            if(word in keywords1):

                keywords1[word] += 1

            else:

                keywords1[word] = 1

            temp = True

            continue

        if (word in keywords):

            if(word in keywords1):

                keywords1[word] += 1

            else:

                keywords1[word] = 1

            continue

        if(word in separators):

            if(word in separators1):

                separators1[word] += 1

            else:

                separators1[word] = 1

            continue

        if(word in operators):

            isConst = True

            if(word in operators1):

                operators1[word] += 1

            else:

                operators1[word] = 1

            continue

        yes = True

        for w in word:

            if ((w in digits or w in letters or w == "\_")and word[0] in letters):

                continue

            yes = False

        if(yes and temp == True and word != ''):

            identifiers[word] = word

        elif(yes and temp == False and word != ''):

            if(word in identifiers and word != ''):

                good = 0

            else:

                errors.append([word, 'What is that, a variable? '])

        elif(temp and yes == False and len(word) != 1 and isConst == False):

            errors.append([word, 'Wrong name for variable, what is that'])

        elif(word and yes == False):

            if(re.match('^[0-9]\*$', word) and word != ''):

                constantas.append(word)

            elif(word[1] == "\\"[0]):

                if(word[len(word)-2] in special\_quote and len(word) == 4):

                    backslash.append(word)

                else:

                    errors.append([word, "Wrong usage of blackslash"])

            elif(word[0] == '"' or word[0] == "'"):

                constantas.append(word)

            else:

                errors.append([word, 'Whats that?'])

        else:

            if(word != ''):

                errors.append([word, 'Wrong symbol, check'])

print("Keywords: ", keywords1)

print("Separators: ", separators1)

print("Operators: ", operators1)

print('Const: ', constantas)

print('Identifiers: ', identifiers)

print('Errors: ', errors)

print('Blackslash: ', backslash)

print('Comments: ', comments)

Analyzer code:

Example:

Изображение выглядит как текст

Автоматически созданное описание

Output:

Изображение выглядит как текст

Автоматически созданное описание

Libraries not included yet ☹

Here we have 2 lexical errors: 4ac and \&. And library thing (