Міністерство освіти і науки, молоді та спорту України

Національний Технічний Університет України

“Київський Політехнічний Інститут”

*Факультет прикладної математики*

*Кафедра СПіСКС*

**Лабораторна робота № 1**

*з дисципліни*

“Інженерія програмного забезпечення. Основи проектування трансляторів”

Тема: “Розробка лексичного аналізатора (ЛА)”

Виконав:

cтудент групи КВ-42

Сахнік Іван

Перевірив:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ 2017 р.

**Постановка завдання**

Розробити програму лексичного аналізатора (ЛА) для підмножини мови програмування SIGNAL.

Програма має забезпечувати наступне (якщо це передбачається граматикою варіанту):

* згортання ідентифікаторів;
* згортання ключових слів;
* згортання цілих десяткових констант;
* згортання дійсних десяткових констант;
* згортання строкових констант, якщо вони визначені в заданій мові;

Також у всіх варіантах необхідно забезпечити:

* видалення коментарів, заданих у вигляді (\*<текст коментарю>\*)

Для кодування лексем необхідно використовувати числові діапазони.

**Входом** ЛА має бути наступне:

* вихідна програма, написана підмножиною мови SIGNAL відповідно до варіанту;
* таблиця кодів ASCII з атрибутами для визначення токенів;
* таблиця багато символьних роздільників;
* таблиця ідентифікаторів, в яку попередньо занесені ключові слова з атрибутом ключового слова;

**Вихід** ЛА має бути наступним:

* закодований рядок лексем;
* таблиці ідентифікаторів, числових, символьних та рядкових констант, сформовані для конкретної програми;

***Варіант 15***

1.<signal-program> --> <program>

2.<program> --> PROGRAM <procedure-identifier> ;

<block> ;

3.<block> --> BEGIN <statements-list> END

4.<statements-list> --> <statement> <statements-list> | <empty>

5.<statement> --> <unsigned-integer> : <statement>|

<variable-identifier> := <unsigned-in-teger> ; |

<procedure-identifier> <actual-arguments> ; |

GOTO <unsigned-integer> ; |

LINK <variable-identifier> , <unsigned-integer> ; |

IN <unsigned-integer>; |OUT <unsigned-integer>; |

RETURN ; |; |

($ <assembly-insert-file-identifier> $)

6.<actual-arguments> --> ( <variable-identifier>

<actual-arguments-list> ) |

<empty>

7.<actual-arguments-list> --> ,<variable-identi-fier>

<actual-arguments-list> | <empty>

8.<variable-identifier> --> <identifier>

9.<procedure-identifier> --> <identifier>

10.<assembly-insert-file-identifier> --> <identi-fier>

11.<identifier> --> <letter><string>

12.<string> --> <letter><string> |

<digit><| 8 | 9string> | <empty>

13.<unsigned-integer> --> <digit><digits-string>

14.<digits-string> --> <digit><digits-string> | <empty>

15.<digit> --> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

16.<letter> --> A | B | C | D | ... | Z

**Код програми**

#lexical analyzer 2017

separetors\_dictionary={";":59, ":":58, ",":44, "(":40, ")":41}

separetors\_comp\_dictionary={":=":301, "($":302, "$)":303}

key\_words\_dicrionary={"PROGRAM":401, "BEGIN":402, "END":403, "GOTO":404,

"LINK":405, "IN":406, "OUT":407, "RETURN":408}

const\_dictionary={}

identificators\_dictionary={}

def lexer(file\_name):

white\_space=[32, 13, 10, 9, 11, 12]

digits=[i for i in range(48, 57)]

chars=[i for i in range(65, 90)]

separetors=[i for i in separetors\_dictionary.keys()]

separetors\_comp=[i for i in separetors\_comp\_dictionary.keys()]

key\_words=[i for i in key\_words\_dicrionary.keys()]

tmp=''

list\_out=[]

ident\_counter=1001

const\_counter=501

file=open(file\_name)

ch=file.read(1)

while ch:

if ord(ch) in white\_space:

ch=file.read(1)

elif ch==':':

tmp+=ch

ch=file.read(1)

if ch=='=':

tmp+=ch

list\_out.append(separetors\_comp\_dictionary[tmp])

ch=file.read(1)

else:

list\_out.append(separetors\_dictionary[tmp])

tmp=''

elif ch=='(':

tmp+=ch

ch=file.read(1)

if ch=='$':

tmp+=ch

list\_out.append(separetors\_comp\_dictionary[tmp])

ch=file.read(1)

elif ch=="\*":

ch=file.read(1)

while ch:

if ch=="\*":

ch=file.read(1)

if ch==")":

ch=file.read(1)

break

else:

ch=file.read(1)

else:

list\_out.append(separetors\_dictionary[tmp])

tmp=''

elif ch=='$':

tmp+=ch

ch=file.read(1)

if ch==')':

tmp+=ch

list\_out.append(separetors\_comp\_dictionary[tmp])

tmp=''

ch=file.read(1)

else:

list\_out.append("\*\*\*")

elif ch in separetors:

tmp+=ch

ch=file.read(1)

list\_out.append(separetors\_dictionary[tmp])

tmp=''

elif ord(ch) in digits:

tmp+=ch

ch=file.read(1)

while ch and ord(ch) in digits:

tmp+=ch

ch=file.read(1)

if tmp in const\_dictionary.keys():

list\_out.append(const\_dictionary[tmp])

tmp=''

else:

const\_dictionary[tmp]=const\_counter

list\_out.append(const\_counter)

const\_counter+=1

tmp=''

elif ord(ch) in chars:

tmp+=ch

ch=file.read(1)

while ch and (ord(ch) in chars or ord(ch) in digits):

tmp+=ch

ch=file.read(1)

if tmp in key\_words:

list\_out.append(key\_words\_dicrionary[tmp])

tmp=''

else:

if tmp in identificators\_dictionary.keys():

list\_out.append(identificators\_dictionary[tmp])

tmp=''

else:

identificators\_dictionary[tmp]=ident\_counter

list\_out.append(ident\_counter)

ident\_counter+=1

tmp=''

else:

list\_out.append("\*\*\*")

ch=file.read(1)

file.close()

print(list\_out)

if \_\_name\_\_ == '\_\_main\_\_':

lexer("TEST.txt")

**Стандартний вміст таблиць**

**Таблиця роздільників**

|  |  |
| --- | --- |
| **Код** | **Значення** |
| 59 | ; |
| 58 | : |
| 44 | , |
| 40 | ( |
| 41 | ) |
| 301 | := |
| 302 | ($ |
| 303 | $) |

**Таблиця ключових слів**

|  |  |
| --- | --- |
| **Код** | **Значення** |
| 401 | PROGRAM |
| 402 | BEGIN |
| 403 | END |
| 404 | GOTO |
| 405 | LINK |
| 406 | IN |
| 407 | OUT |
| 408 | RETURN |