

Лабораторная работа №3

Lex.py

```
class Lex:
```

```
    def __init__(self, name: str = "", value: str = ""):
        self.name = name
        self.value = value
```

```
    def __str__(self) -> str:
        if self.value == "":
            return f'{self.name}'
        else:
            return f'{self.name}: {self.value}'
```

consts.py

```
static_words = {  
    'void': 'VOID',  
    'switch': 'SWITCH',  
    'else': 'ELSE',  
    'int64': 'INT64',  
    'int': 'INT',  
    'long': 'LONG',  
    'break': 'BREAK',  
    'default': 'DEFAULT',  
    'case': 'CASE'  
}
```

```
one_symbols = {  
    '(': 'ROUND_LEFT',  
    ')': 'ROUND_RIGHT',  
    '{': 'CURLY_LEFT',  
    '}': 'CURLY_RIGHT',  
    ';': 'SEMICOLON',  
    '=': 'ASSIGN',  
    '+': 'PLUS',  
    '-': 'MINUS',  
    '*': 'STAR',  
    '/': 'SLASH',  
    '%': 'PERCENT',  
    ',': 'COMMA',  
    '<': 'LESS',  
    '>': 'GREATER',  
    '::': 'COLON'  
}
```

```
two_symbols = {  
    '>>': 'R_SHIFT',  
    '<<': 'L_SHIFT',  
    '==': 'EQ',  
    '!=': 'NOT_EQ',  
    '<=': 'LESS_EQ',  
    '>=': 'GREATER_EQ'  
}
```

file_reader.py

```
def read_file(path: str):  
    f = open(path, 'r')  
    t = "".join(f.readlines())  
    f.close()  
    return t + '\n\0'
```

is_functions.py

```
def is_digit_16(s: str):  
    return len(s) == 1 and s in '0123456789abcdefABCDEF'
```

```
def is_digit_16_not_zero(s: str):  
    return len(s) == 1 and s in '0123456789abcdefABCDEF'
```

```
def is_digit(s: str):  
    return len(s) == 1 and s in '0123456789'
```

```
def is_digit_not_zero(s: str):  
    return len(s) == 1 and s in '123456789'
```

```
def is_not_digit(s: str):  
    return len(s) == 1 and s in  
'abcdefghijklmnopqrstuvwxyz_ABCDIFJHIJKLMNOPQRSTUVWXYZ'
```

scanner.py

```
from utils.Lex import Lex
from utils.consts import static_words, one_symbols, two_symbols
from utils.file_reader import read_file
from utils.is_functions import is_not_digit, is_digit, is_digit_not_zero,
is_digit_16_not_zero, is_digit_16
```

```
i = 0
col = 1
row = 1
text = "
```

```
def up_i():
    global i, col, row
    if i < len(text) - 1:
        if text[i] == '\n':
            row += 1
            col = 0
        else:
            col += 1
        i += 1
```

```
def skip_white_symbols_and_comments():
    global i, text
    has = True
    while has:
        has = False
        if text[i:i + 2] == '//':
            has = True
            while text[i] != '\n':
                up_i()
            up_i()
```

```
if text[i] in '\t\n ':  
    has = True  
    up_i()
```

```
def find_two_symbols():  
    global i, text  
    s = text[i:i + 2]  
    if s in two_symbols.keys():  
        name = two_symbols[s]  
        up_i()  
        up_i()  
        return Lex(name), True  
    return None, False
```

```
def find_one_symbols():  
    global i, text  
    s = text[i]  
    if s in one_symbols.keys():  
        name = one_symbols[s]  
        up_i()  
        return Lex(name), True  
    return None, False
```

```
def find_id_or_static_words():  
    global i, text  
    if is_not_digit(text[i]):  
        s = text[i]  
        up_i()  
    while is_digit(text[i]) or is_not_digit(text[i]):  
        s += text[i]  
        up_i()
```

```
    if s in static_words.keys():
        return Lex(static_words[s]), True
    else:
        return Lex('ID', s), True
return None, False
```

```
def find_consts():
    global i, text
    if is_digit_not_zero(text[i]):
        s = text[i]
        up_i()
        while is_digit(text[i]):
            s += text[i]
            up_i()
        return Lex('DEC', s), True
    return None, False
```

```
def find_consts_hex():
    global i, text
    if text[i:i + 2] == '0x':
        s = text[i:i + 2]
        up_i()
        up_i()
        if is_digit_16_not_zero(text[i]):
            s += text[i]
            up_i()
            while is_digit_16(text[i]):
                s += text[i]
                up_i()
            return Lex('HEX', s), True
    else:
        return Lex('HEX', '0x0'), True
```

```
return None, False
```

```
def load_file(path: str):  
    global i, col, row, text  
    i = 0  
    col = 1  
    row = 1  
    text = read_file(path)
```

```
def next_lex():  
    skip_white_symbols_and_comments()  
    lex, ok = find_two_symbols()  
    if ok:  
        return lex  
    lex, ok = find_one_symbols()  
    if ok:  
        return lex  
    lex, ok = find_id_or_static_words()  
    if ok:  
        return lex  
    lex, ok = find_consts()  
    if ok:  
        return lex  
    lex, ok = find_consts_hex()  
    if ok:  
        return lex  
    return Lex('EOF')
```