Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Кафедра интеллектуально-информационных технологий

Лабораторная работа №2

По дисциплине «Языковые процессы интеллектуальных систем»

Выполнил:

студент 4 курса

группы ИИ-22

Копанчук Е. Р.

Проверил:

Монтик Н. С.

Брест-2024

**Цель работы:** изучить основных понятий теории регулярных грамматик, ознакомиться с назначением и принципами работы лексических анализаторов (сканеров), получить навыки построения сканера на примере заданного простейшего входного языка.

Для выполнения лабораторной работы требуется написать программу, которая выполняет лексический анализ входного текста в соответствии с заданием и порождает таблицу лексем с указанием их типов и значений. Текст на входном языке задается в виде символьного (текстового) файла. Программа должна выдавать сообщения о наличие во входном тексте ошибок, которые могут быть обнаружены на этапе лексического анализа.

Длину идентификаторов и строковых констант считать ограниченной 32 символами. Программа должна допускать наличие комментариев неограниченной длины во входном файле. Форму организации комментариев выбрать самостоятельно.

**Задание**

9. Входной язык содержит последовательность описаний массивов в соответствии со спецификацией языка Паскаль, разделенных символами ; (точка с запятой). Считать, что массивы могут содержать только элементы скалярных типов integer, real, byte, word и char.

**Код программы:**

OPERATORS = ['+', '-', '\*', '/', '=', '==', '!', '&&', '||', '(', ')', '[', ']', '{', '}', ',', ';', '.']

WHITESPACE = [' ', '\n', '\t']

DIGITS = '0123456789'

LETTERS = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZабвгдеёжзийклмнопрстуфхцчшщъыьэюяАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ'

ALLOWED\_CHARS = LETTERS + DIGITS + '\_'

def determine\_token\_type(token):

# Проверка для целых чисел

if token.isdigit():

return 'Integer'

elif '.' in token and all(part.isdigit() for part in token.split('.')) and len(token.split('.')) == 2:

return 'Real'

elif len(token) == 3 and (token.startswith("'") or token.startswith('"')) and token[1] in LETTERS:

return 'Char'

elif token[0] in LETTERS or token[0] == '\_':

if all(char in ALLOWED\_CHARS for char in token):

return 'Identifier'

elif token in OPERATORS:

return 'Operator'

elif token.startswith('//'):

return 'Comment'

else:

return 'Unknown'

def lexical\_analysis(text):

token\_table = []

errors = []

token = ""

i = 0

while i < len(text):

char = text[i]

if char in WHITESPACE:

if token:

token\_type = determine\_token\_type(token)

if token\_type == 'Unknown':

errors.append(f"Unknown token: '{token}'")

else:

token\_table.append((token, token\_type))

token = ""

i += 1

continue

if char in OPERATORS:

if token:

token\_type = determine\_token\_type(token)

if token\_type == 'Unknown':

errors.append(f"Unknown token: '{token}'")

else:

token\_table.append((token, token\_type))

token = ""

if i + 1 < len(text) and text[i:i + 2] in OPERATORS:

token\_table.append((text[i:i + 2], 'Operator'))

i += 2

else:

token\_table.append((char, 'Operator'))

i += 1

continue

if char == '/' and i + 1 < len(text) and text[i + 1] == '/':

if token:

token\_type = determine\_token\_type(token)

if token\_type == 'Unknown':

errors.append(f"Unknown token: '{token}'")

else:

token\_table.append((token, token\_type))

token = ""

comment = text[i:].split('\n', 1)[0]

token\_table.append((comment, 'Comment'))

i += len(comment)

continue

token += char

i += 1

if token:

token\_type = determine\_token\_type(token)

if token\_type == 'Unknown':

errors.append(f"Unknown token: '{token}'")

else:

token\_table.append((token, token\_type))

return token\_table, errors

file\_path = "D:/7 семестр/ЯПИС лабы/ЯПИС лаба №2/Массивы.txt"

with open(file\_path, 'r', encoding='utf-8') as file:

file\_data = file.read()

token\_table, errors = lexical\_analysis(file\_data)

print("Таблица лексем:")

for token, token\_type in token\_table:

print(f"Токен: '{token}', Тип: {token\_type}")

if errors:

print("\nОшибки:")

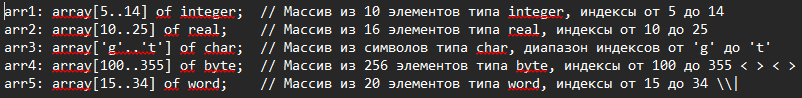
for error in errors:

print(error)

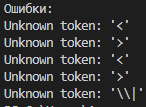
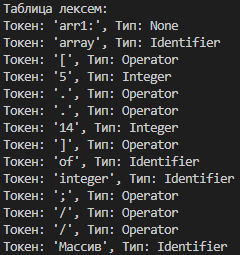
else:

print("\nОшибок не обнаружено.")

**Содержимое текстового файла:**



**Вывод программы:**



**Вывод:** изучил основные понятия теории регулярных грамматик, ознакомился с назначением и принципами работы лексических анализаторов (сканеров), получил навыки построения сканера на примере заданного простейшего входного языка.