Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра вычислительных технологий

**ОТЧЕТ**

о выполнении лабораторной работы № 1

по дисциплине «Методы разработки трансляторов»

Тема: «Построениелексического анализатора»

Выполнил: ст. гр. 39/1

Козлов Э.Д.

Проверил: Преподаватель

Усов П.Е

Краснодар

2025

Ход работы:

Постановка задачи: Требуется разработать лексический анализатор для определения лексем языка С.

В языке программирования C лексемы можно разделить на несколько категорий:

1. Служебные слова (ключевые слова)

Это зарезервированные слова, определяющие синтаксис языка:

• int, char, float, double, void

• if, else, switch, case, default

• for, while, do, break, continue

• return, goto, sizeof, static, struct, union, typedef

• const, volatile, extern, register, auto

• signed, unsigned, long, short, enum

2. Разделители

Символы, используемые для структурирования кода:

• Скобки: (), {}, []

• Запятые и точки с запятой: ,, ;

• Операторы доступа и указателей: ., ->

3. Операции

• Арифметические: +, -, \*, /, %

• Логические: &&, ||, !

• Побитовые: &, |, ^, ~, <<, >>

• Операции присваивания: =, +=, -=, \*=, /=, %=

• Операции сравнения: ==, !=, <, >, <=, >=

4. Константы

• Числовые: 42, 3.14, 0xFF, 075

• Символьные: 'A', '\n', '\x41'

• Строковые: "Hello, World!"

5. Идентификаторы

• Пользовательские имена переменных, функций, структур, типов:

• x, volume, y

Эта классификация используется сканером для разбиения кода на отдельные токены.

Перенеся эти данные в программу, можно разбивать на лексемы код языка С.

Проведя тесты программы, мы убедились в правильности работы сканера.

Исходный код на языке С:

int main() {  
 int x = 10;  
 int y = 192  
 printf("Hello, World!\n");  
 if (x+1 == 2) return 1;  
 return 0;  
}

Ожидаемый результат:

W1 W27 R1 R2 R3

W1 I1 O17 N1 R8

W1 I2 O17 N2

W28 R1 C1 R2 R8

W6 R1 I1 O12 N3 O1 N4 R2 W5 N5 R8

W5 N6 R8

W29 R1 I2 R2

W29 R1 I1 R2

R4

Фактический результат:

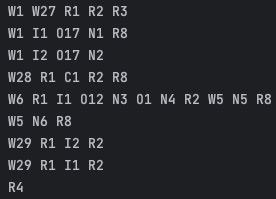


Рисунок 1 – Фактический результат

Словари лексем можно обновлять при необходимости, что делает программу гибкой для изменений.

**Алгоритм работы кода:**

1. **Определение классов лексем** – задаются служебные слова, операторы, разделители, числа и константы.

2. **Создание таблиц лексем** – ключевые слова, операторы и разделители заносятся в словари с уникальными кодами.

3. **Формирование регулярных выражений** для распознавания различных типов токенов (идентификаторы, числа, строки и др.).

4. **Чтение кода C из файла** – программа загружает содержимое test.c.

5. **Токенизация** – разбивает код на строки и анализирует их:

• Пропускает пробелы и комментарии.

• Определяет тип лексемы по регулярным выражениям.

• Назначает уникальный код для идентификаторов, чисел и констант.

• Подставляет код для известных ключевых слов, операторов и разделителей.

6. **Вывод токенизированного кода** – каждая строка представляется в виде последовательности кодов лексем.

Листинг программы

import re  
  
LEXEME\_CLASSES = {  
 "W": "Служебное слово",  
 "I": "Идентификатор",  
 "O": "Операция",  
 "R": "Разделитель",  
 "N": "Число",  
 "C": "Константа"  
}  
  
KEYWORDS = {  
 kw: f"W{idx}" for idx, kw in enumerate([  
 "int", "char", "float", "double", "return", "if", "else",  
 "for", "while", "do", "switch", "case", "break", "continue",  
 "default", "void", "static", "struct", "typedef", "union",  
 "unsigned", "signed", "long", "short", "goto", "sizeof",  
 "main", "printf", "scanf"  
 ], start=1)  
}  
  
OPERATORS = {  
 op: f"O{idx}" for idx, op in enumerate([  
 "==", "!=", "<=", ">=", "&&", "||", "+=", "-=", "\*=", "/=", "%=",  
 "+", "-", "\*", "/", "%", "=", "<", ">"  
 ], start=1)  
}  
  
DELIMITERS = {  
 delim: f"R{idx}" for idx, delim in enumerate([  
 "(", ")", "{", "}", "[", "]", ",", ";", "."  
 ], start=1)  
}  
  
TOKEN\_PATTERNS = [  
 (r'//.\*', 'COMMENT'),  
 (r'/\\*.\*?\\*/', 'COMMENT', re.DOTALL),  
 (r'#\s\*include\s\*<.\*?>', 'W'),  
 (r'\b(' + '|'.join(KEYWORDS.keys()) + r')\b', 'W'),  
 (r'[a-zA-Z\_]\w\*', 'I'),  
 (r'\d+\.\d+|\d+', 'N'),  
 (r'".\*?"', 'C'),  
 (r'(' + '|'.join(map(re.escape, OPERATORS.keys())) + r')', 'O'),  
 (r'(' + '|'.join(map(re.escape, DELIMITERS.keys())) + r')', 'R')  
]  
  
IDENTIFIERS = {}  
  
ident\_counter = 1  
num\_counter = 1  
const\_counter = 1  
def tokenize(code):  
 global ident\_counter, num\_counter, const\_counter  
 lines = code.split("\n")  
 tokenized\_lines = []  
  
 for line in lines:  
 tokens = []  
 code = line.strip()  
 while code:  
 code = code.lstrip()  
 match = None  
 for pattern, lexeme\_type, \*flags in TOKEN\_PATTERNS:  
 regex\_flags = flags[0] if flags else 0  
 match = re.match(pattern, code, regex\_flags)  
 if match:  
 value = match.group(0)  
 if lexeme\_type == "I":  
 if value in IDENTIFIERS:  
 lexeme\_code = IDENTIFIERS[value]  
 else:  
 lexeme\_code = f"I{ident\_counter}"  
 IDENTIFIERS[value] = lexeme\_code  
 ident\_counter += 1  
 elif lexeme\_type == "N":  
 lexeme\_code = f"N{num\_counter}"  
 num\_counter += 1  
 elif lexeme\_type == "C":  
 lexeme\_code = f"C{const\_counter}"  
 const\_counter += 1  
 else:  
 lexeme\_code = (  
 KEYWORDS.get(value, None) or  
 OPERATORS.get(value, None) or  
 DELIMITERS.get(value, None)  
 )  
  
 if lexeme\_code:  
 tokens.append(lexeme\_code)  
 code = code[len(value):]  
 break  
 if not match:  
 raise SyntaxError(f"Неизвестный символ: {code[:10]}")  
  
 if tokens:  
 tokenized\_lines.append(" ".join(tokens))  
  
 return tokenized\_lines  
  
file\_path = "test.c"  
  
with open(file\_path, "r", encoding="utf-8") as f:  
 code\_c = f.read()  
  
tokenized\_lines = tokenize(code\_c)  
  
for line in tokenized\_lines:

print(line)

Словарь лексем

|  |  |  |
| --- | --- | --- |
| Код | Лексема | Тип |
| W1 | int | Служебное слово |
| W2 | char | Служебное слово |
| W3 | float | Служебное слово |
| W4 | double | Служебное слово |
| W5 | return | Служебное слово |
| W6 | if | Служебное слово |
| W7 | else | Служебное слово |
| W8 | for | Служебное слово |
| W9 | while | Служебное слово |
| W10 | do | Служебное слово |
| W11 | switch | Служебное слово |
| W12 | case | Служебное слово |
| W13 | break | Служебное слово |
| W14 | continue | Служебное слово |
| W15 | default | Служебное слово |
| W16 | void | Служебное слово |
| W17 | static | Служебное слово |
| W18 | struct | Служебное слово |
| W19 | typedef | Служебное слово |
| W20 | union | Служебное слово |
| W21 | unsigned | Служебное слово |
| W22 | signed | Служебное слово |
| W23 | long | Служебное слово |
| W24 | short | Служебное слово |
| W25 | goto | Служебное слово |
| W26 | sizeof | Служебное слово |
| W27 | main | Служебное слово |
| W28 | printf | Служебное слово |
| W29 | scanf | Служебное слово |
| O1 | == | Операция |
| O2 | != | Операция |
| O3 | <= | Операция |
| O4 | >= | Операция |
| O5 | && | Операция |
| O6 | || | Операция |
| O7 | += | Операция |
| O8 | -= | Операция |
| O9 | \*= | Операция |
| O10 | /= | Операция |
| O11 | %= | Операция |
| O12 | + | Операция |
| O13 | - | Операция |
| O14 | \* | Операция |
| O15 | / | Операция |
| O16 | % | Операция |
| O17 | = | Операция |
| O18 | < | Операция |
| O19 | > | Операция |
| R1 | ( | Разделитель |
| R2 | ) | Разделитель |
| R3 | { | Разделитель |
| R4 | } | Разделитель |
| R5 | [ | Разделитель |
| R6 | ] | Разделитель |
| R7 | , | Разделитель |
| R8 | ; | Разделитель |
| R9 | . | Разделитель |