Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти й газу

Кафедра комп’ютерних систем і мереж

Лабораторна робота №6

на тему:

**“**ПРОЕКТУВАННЯ ЛЕКСИЧНОГО АНАЛІЗАТОРА **”**

Виконав:

студент групи КІ-21-1К

Шиманський Б.В.

Перевірив: викладач

Гарасимів В. М.

**Мета:** вивчення основних понять теорії регулярних граматик, ознайомлення з призначенням і принципами роботи лексичних аналізаторів (сканерів), отримання практичних навиків побудови сканера на прикладі заданої простої вхідної мови.

**Завдання**

Для виконання лабораторної роботи потрібно написати програму, яка виконує лексичний аналіз вхідного тексту відповідно до завдання і породжує таблицю лексем з вказівкою їх типів і значень. Текст на вхідній мові задається у вигляді символьного (текстового) файлу. Програма повинна видавати повідомлення про наявність у вхідному тексті помилок, які можуть бути виявлені на етапі лексичного аналізу. Довжину ідентифікаторів і рядкових констант вважати за обмежену 32 символами. Програма повинна допускати наявність коментарів необмеженої довжини у вхідному файлі. Форму організації коментарів пропонується вибрати самостійно.

Вхідна мова містить логічні вирази, розділені символом ;(крапка з комою). Логічні вирази складаються з ідентифікаторів, шіснадцяткових чисел, знаку привласнення (:=), знаків операцій or, xor, and, not і круглих дужок.

**Хід роботи**

**Код програми**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <unordered\_map>

using namespace std;

enum TokenType {

IDENTIFIER,

HEX\_NUMBER,

ASSIGNMENT,

OR\_OPERATOR,

XOR\_OPERATOR,

AND\_OPERATOR,

NOT\_OPERATOR,

LEFT\_PAREN,

RIGHT\_PAREN,

SEMICOLON,

INVALID

};

struct Token {

TokenType type;

string value;

};

bool is\_hex\_digit(char c) {

return ((c >= '0' && c <= '9') || (c >= 'a' && c <= 'f') || (c >= 'A' && c <= 'F'));

}

bool is\_valid\_identifier\_char(char c) {

return ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9'));

}

Token get\_next\_token(string& input, size\_t& pos) {

while (pos < input.size() && input[pos] == ' ') {

pos++;

}

if (pos == input.size()) {

return { SEMICOLON, "" };

}

if (is\_valid\_identifier\_char(input[pos])) {

string identifier = "";

while (pos < input.size() && is\_valid\_identifier\_char(input[pos])) {

identifier += input[pos];

pos++;

}

return { IDENTIFIER, identifier };

}

if (input[pos] == '0' && pos + 1 < input.size() && input[pos + 1] == 'x') {

pos += 2;

string hex\_num = "";

while (pos < input.size() && is\_hex\_digit(input[pos])) {

hex\_num += input[pos];

pos++;

}

return { HEX\_NUMBER, hex\_num };

}

if (input[pos] == ':' && pos + 1 < input.size() && input[pos + 1] == '=') {

pos += 2;

return { ASSIGNMENT, ":=" };

}

if (input[pos] == '|') {

pos++;

if (pos < input.size() && input[pos] == '|') {

pos++;

return { OR\_OPERATOR, "||" };

}

return { INVALID, "" };

}

if (input[pos] == '^') {

pos++;

return { XOR\_OPERATOR, "^" };

}

if (input[pos] == '&') {

pos++;

if (pos < input.size() && input[pos] == '&') {

pos++;

return { AND\_OPERATOR, "&&" };

}

return { INVALID, "" };

}

if (input[pos] == '!') {

pos++;

return { NOT\_OPERATOR, "!" };

}

if (input[pos] == '(') {

pos++;

return { LEFT\_PAREN, "(" };

}

if (input[pos] == ')') {

pos++;

return { RIGHT\_PAREN, ")" };

}

if (input[pos] == ';') {

pos++;

return { SEMICOLON, ";" };

}

return { INVALID, "" };

}

void print\_token(Token token) {

unordered\_map<TokenType, string> type\_to\_str = {

{ IDENTIFIER, "Identifier" },

{ HEX\_NUMBER, "Hex Number" },

{ ASSIGNMENT, "Assignment" },

{ OR\_OPERATOR, "Or Operator"},

{ XOR\_OPERATOR, "Xor Operator" },

{ AND\_OPERATOR, "And Operator" },

{ NOT\_OPERATOR, "Not Operator" },

{ LEFT\_PAREN, "Left Parenthesis" },

{ RIGHT\_PAREN, "Right Parenthesis" },

{ SEMICOLON, "Semicolon" },

{ INVALID, "Invalid" }

};

cout << type\_to\_str[token.type] << " - " << token.value << endl;

}

int main() {

ifstream input\_file("input.txt");

if (!input\_file.is\_open()) {

cerr << "Error opening file: " << endl;

return 1;

}

string input((istreambuf\_iterator<char>(input\_file)), istreambuf\_iterator<char>());

input\_file.close();

vector<Token> tokens;

size\_t pos = 0;

while (pos < input.size()) {

Token token = get\_next\_token(input, pos);

if (token.type != INVALID) {

tokens.push\_back(token);

}

else {

cerr << "Invalid token found at position " << pos << endl;

return 1;

}

}

for (Token token : tokens) {

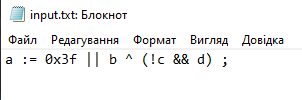
print\_token(token);

}

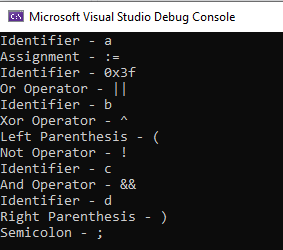
return 0;

}

**Вміст файлу**



**Результат виконання програми**



GitHub репозиторій з виконаними лабораторними роботами:

https://github.com/Rocker7000/labs

**Висновки:** на даній лабораторній роботі я вивчив основні поняття теорії регулярних граматик, ознайомився з призначенням і принципами роботи лексичних аналізаторів (сканерів), отримав практичні навикі побудови сканера на прикладі заданої простої вхідної мови.