**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЯ ОБРАЗОВАНИЯ**

**“ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ ЕВФРОСИНИИ ПОЛОЦКОЙ”**

Факультет информационных технологий

Кафедра технологий программирования

**Отчёт по Лабораторной работе №1**

**«Лексический анализатор»**

Выполнил студент 2 курса, группа 21-ИТ-1 Макеёнок Д.И.

Проверил Сыцевич Д.

Полоцк, 2022 г.

**Цель работы:** ознакомиться с лексическим анализатором. написать программу, которая выполняет лексический анализ входного текста в соответствии с заданием и порождает таблицу лексем с указанием их типов.

#include<iostream>

#include<string>

#include<list>

using namespace std;

enum tok\_names { semocolon, ident, num, asgn, parentheses, OR, XOR, AND, NOT, boolean };

struct token

{

enum tok\_names token\_name;

string token\_names[10] = { "Semocolon", "Ident", "Number", "Assign", "Parentheses", "OR", "XOR", "AND", "NOT", "Boolean" };

string token\_value;

};

list<token> table;

token add\_token(tok\_names a, string b) {

token tok;

tok.token\_name = a;

tok.token\_value = b;

return tok;

}

list<token> lexer(string str)

{

list<token> table;

int i = 0; int value = 0;

while (i < str.size())

{

if (str[i] == 'F') {

table.push\_back(add\_token(boolean, "F"));

i++;

}

if (str[i] == 'T') {

table.push\_back(add\_token(boolean, "T"));

i++;

}

if (str[i] == ':' and str[i + 1] == '=') {

table.push\_back(add\_token(asgn, ":="));

}

if (str[i] == ';') {

table.push\_back(add\_token(semocolon, ";"));

}

if (str[i] == '(') {

table.push\_back(add\_token(parentheses, "("));

}

if (str[i] == ')') {

table.push\_back(add\_token(parentheses, ")"));

}

if (str[i] == 'n' and str[i + 1] == 'o' and str[i + 2] == 't') {

table.push\_back(add\_token(NOT, "not"));

i += 3;

}

if (str[i] == 'a' and str[i + 1] == 'n' and str[i + 2] == 'd') {

table.push\_back(add\_token(AND, "and"));

i += 3;

}

if (str[i] == 'x' and str[i + 1] == 'o' and str[i + 2] == 'r') {

table.push\_back(add\_token(XOR, "xor"));

i += 3;

}

if (str[i] == 'o' and str[i + 1] == 'r') {

table.push\_back(add\_token(OR, "or"));

i += 2;

}

if ((str[i] >= 'a' and str[i] <= 'z') or (str[i] >= 'A' and str[i] <= 'Z')) {

string str2 = "";

while ((str[i] >= 'a' and str[i] <= 'z') or (str[i] >= 'A' and str[i] <= 'Z'))

{

str2 += str[i];

i++;

}

i--;

table.push\_back(add\_token(ident, str2));

}

if ((str[i] >= '0' and str[i] <= '9')) {

string str2 = "";

while ((str[i] >= '0' and str[i] <= '9'))

{

str2 += str[i];

i++;

}

i--;

table.push\_back(add\_token(num, str2));

}

i++;

}

return table;

}

int main()

{

token tok;

string str = "(T or F aa := xor or and b := 321)";

list<token> table = lexer(str);

while (table.empty() == 0) {

tok = table.front();

cout << "name:" << tok.token\_names[tok.token\_name] << ", " << "value:" << tok.token\_value << endl;

table.pop\_front();

}

}

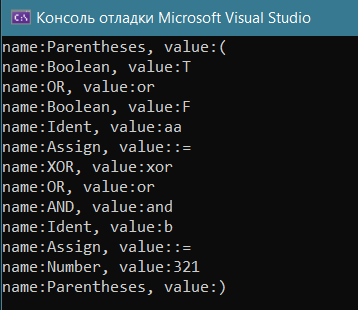


Рисунок 1 – Результат работы программы

**Вывод :** Я ознакомился с лексическим анализатором. написал программу, которая выполняет лексический анализ входного текста в соответствии с заданием и порождает таблицу лексем с указанием их типов.