Київський національний університет імені Тараса Шевченка

Факультет комп'ютерних наук та кібернетики

Лабораторна робота №3

З дисципліни “Системне програмування”

на тему

“Lexical analysis”

Виконав:

студент 3-го курсу групи ТТП-32

ОПП «Інформатика»

Черненко Євгеній

Київ – 2023

Лабораторна робота №3

Варіант 9: Мова програмування Assembler

Реалізувати лексичний аналізатор мови програмування. Для зберігання класів лексем організувати таблиці. Вивести вміст таблиць після обробки тексту програми.

Розрізняти принаймні такі класи лексем:

- числа (десяткові, з плаваючою крапкою, шістнадцяткові),
- рядкові та символьні константи,
- директиви препроцесора,
- коментарі,
- зарезервовані слова,
- оператори,
- розділові знаки,
- ідентифікатори.

Позначати ситуації з помилками (наприклад, нерозпізнавані символи).

Реалізовано даний вивід результату роботи програми:

- вивід пар < лексема, тип_лексеми >

Лабораторну роботу виконано на мові програмування C++.

Робота програми:

- Програма починає роботу із виведення базової інформації

```
Lab task: code the analyzer to divide the code into lexemes and determine their type
Chernenko Yevhenii, TTP-32
```

- Згодом програма запитує ім’я файлу, якщо зчитування пройшло невдало, то програма пропонує користувачу знову ввести ім’я файлу, або пропонує команду для виходу з програми

```
Please enter the file name:
asd
Cannot read a file, please enter the file name again. Also, you can type 'exit' and stop a program
example_code.txt
```

- Потім програма відкриває файл та зчитує весь текст в змінну code, розбиває отриманий код на лексеми та аналізує до якої категорії відноситься кожна лексема та виводить вектор розгаданих лексем

```
string code( beg: (istreambuf_iterator<char>( &: file)), end: istreambuf_iterator<char>());
```

```
vector<pair<string, string>> lexemes;
regex wordsRegex( p: ":.*|\\d*\\.\\d+|\\w+|[\\.,:\\[\\]{}()]|\"[^\"]*\"|'.'|#[a-zA-Z_][a-zA-Z0-9_]*\\b");
smatch match;

string::const_iterator start(code.cbegin());

while (regex_search( s: start, e: code.cend(), &: match, re: wordsRegex)) {
    string lexeme = match[0];
    string lexemeType = "Error";

    for (const auto& pattern : pair<...> const & : patterns) {
        if (regex_match( s: lexeme, re: pattern.first)) {
            lexemeType = pattern.second;
            break;
        }
    }
    lexemes.emplace_back( a: lexeme, b: lexemeType);
    start = match.suffix().first;
}
return lexemes;
```

```
for (const auto& lexeme : pair<...> const & : lexemes) {
    cout << "< " << lexeme.first << " - " << lexeme.second << " >" << endl;
}
```

Приклад роботи програми:

Вхідний файл:

```
; Example of comment
MOV adsa00, 1
ADD b, a
DIV a, 0x10
#directive
SUB c, 21.2
MOV c, 'a'
DIV d, "asd"
```

Результат роботи програми:

```
< ; Example of comment - Comment >
< MOV - Reserved word >
< adsa00 - Identifier >
< , - Separator >
< 1 - Dec number >
< ADD - Reserved word >
< b - Identifier >
< , - Separator >
< a - Identifier >
< DIV - Reserved word >
< a - Identifier >
< , - Separator >
< 0x10 - Hex number >
< #directive - Preprocessor >
< SUB - Reserved word >
< c - Identifier >
< , - Separator >
< 21.2 - Float number >
< MOV - Reserved word >
< c - Identifier >
< , - Separator >
< 'a' - Char >
< DIV - Reserved word >
< d - Identifier >
< , - Separator >
< "asd" - String >
```

Додаток. Код програми

```
#include <iostream>
#include <string>
#include <fstream>
#include <utility>
#include <regex>

using namespace std;

vector<pair<string, string>> lexicalAnalyzer(ifstream &file) {

    regex numberFloatRegex("^\\d*\\.\\d+$");
    regex numberDecimalRegex("^\\d+$");
    regex numberHexadecimalRegex("^0[xX][0-9a-fA-F]+$");
    regex stringConstantRegex("\"[^\"]*\"");
    regex charConstantRegex("'.'");
    regex preprocessorRegex("^#[a-zA-Z_]\\w*$");
    regex commentRegex("^;.*$");
    regex reservedWordRegex("^(MOV|ADD|SUB|DIV|MUL)$");
    regex operatorRegex("^(AND|OR|XOR)$");
    regex separatorRegex("^[\\.,;:\\[\\]{}()]$");
    regex identifierRegex("^[a-zA-Z_][a-zA-Z0-9_]*$");

    string code((istreambuf_iterator<char>(file)),
istreambuf_iterator<char>());

    vector<pair<regex, string>> patterns = {
            {numberFloatRegex, "Float number"},
            {numberDecimalRegex, "Dec number"},
            {numberHexadecimalRegex, "Hex number"},
            {stringConstantRegex, "String"},
            {charConstantRegex, "Char"},
            {preprocessorRegex, "Preprocessor"},
            {commentRegex, "Comment"},
            {reservedWordRegex, "Reserved word"},
            {operatorRegex, "Operator"},
            {separatorRegex, "Separator"},
            {identifierRegex, "Identifier"}
    };

    vector<pair<string, string>> lexemes;
    regex
wordsRegex(";.*|\\d*\\.\\d+|\\w+|[\\.,;:\\[\\]{}()]|\"[^\"]*\"|'.'|#[a-zA-
Z_][a-zA-Z0-9_]*\\b");
    smatch match;

    string::const_iterator start(code.cbegin());

    while (regex_search(start, code.cend(), match, wordsRegex)) {
        string lexeme = match[0];
        string lexemeType = "Error";

        for (const auto& pattern: patterns) {
            if (regex_match(lexeme, pattern.first)) {
                lexemeType = pattern.second;
                break;
            }
        }
        lexemes.emplace_back(lexeme, lexemeType);
        start = match.suffix().first;
    }
    return lexemes;
```

```
}

int main() {

    cout << "Lab task: code the analyzer to divide the code into lexemes and
determine their type" << endl;
    cout << "Chernenko Yevhenii, TTP-32" << endl;

    string filename;
    ifstream file;

    cout << "Please enter the file name:" << endl;
    cin >> filename;

    do {
        file.open(filename);

        if(file.is_open()) {
            break;
        }

        cout << "Cannot read a file, please enter the file name again. Also,
you can type 'exit' and stop a program" << endl;
        cin >> filename;

        if(filename == "exit"){
            return 1;
        }

    } while(true);

    vector<pair<string, string>> lexemes = lexicalAnalyzer(file);

    for (const auto& lexeme : lexemes) {
        cout << "< " << lexeme.first << " - " << lexeme.second << " >" <<
endl;
    }

    file.close();

    return 0;
}
```