

Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики

Лабораторна робота №3

З дисципліни “Системне програмування”

за темою “Лексичний аналізатор мови програмування”

Виконала:

студентка 3-го курсу групи ТТП-32

спеціальності "Інформатика"

Черечеча Катерина Сергіївна

Київ – 2023

Варіант 5. Лексичний аналізатор для мови C#

Приклад коду C#:

```
using System;

class Program {
    // Single line comment
    static void Main(string[] args) {
        Console.WriteLine("Hello, world!");

        int hexNumber = 0x1a3;
        int number = 10;
        int numberNext = 5;
        if (number < numberNext){
            Console.WriteLine(number);
        }
        /*
            Multi-line comment
        */
    }
}
```

Результат роботи програми:

| | |
|----------------------------------|------------------------------|
| using - Keyword | = - Operator |
| System - Identifier | 10 - Numeric Constant |
| ; - Delimiter | ; - Delimiter |
| class - Keyword | int - Keyword |
| Program - Identifier | numberNext - Identifier |
| { - Delimiter | = - Operator |
| // Single line comment - Comment | 5 - Numeric Constant |
| static - Keyword | ; - Delimiter |
| void - Keyword | #if - Preprocessor Directive |
| Main - Identifier | (- Delimiter |
| (- Delimiter | number - Identifier |
| string - Keyword | < - Operator |
| [- Delimiter | numberNext - Identifier |
|] - Delimiter |) - Delimiter |
| args - Identifier | { - Delimiter |
|) - Delimiter | int - Keyword |
| { - Delimiter | dec - Identifier |
| Console.WriteLine - Unknown | = - Operator |
| (- Delimiter | 10.5 - Decimal Number |
| Hello, world!" - String Constant | ; - Delimiter |
|) - Delimiter | } - Delimiter |
| ; - Delimiter | } - Delimiter |
| int - Keyword | /* |
| hexNumber - Identifier | |
| = - Operator | Multi-line comment |
| 0x1a3 - Hexadecimal Number | */ - Comment |
| ; - Delimiter | |
| int - Keyword | } - Delimiter |
| number - Identifier | } - Delimiter |
| | \$ - Unknown |

Реалізація програми:

```
#include <iostream>
#include <vector>
#include <regex>
#include <fstream>
#include <sstream>

enum TokenType {
    Keyword,
    Identifier,
    String,
    Number,
    HexadecimalNumber,
    DecimalNumber,
    Punctuation,
    Operator,
    Comment,
    PreprocessorDirective,
    Unknown
};

struct Token {
    std::string value;
    TokenType type;
};
```

Підключення всіх необхідних бібліотек, TokenType - всі можливі типи токенів

Функції, представлені на наступних скріншотах визначають всі можливі типи лексем за допомогою регулярних виразів та `std::regex_match` для перевірки того, чи відповідає рядок заданому виразу.

isKeyword(const std::string& token): визначає, чи є переданий токен ключовим словом мови C# за допомогою функції `std::find` для пошуку токenu серед заданих ключових слів.

isIdentifier(const std::string& token): визначає, чи є переданий токен ідентифікатором в мові C# за допомогою регулярного виразу для перевірки його формату.

isStringConstant(const std::string& token): визначає, чи є переданий токен стрінгом використовуючи регулярний вираз для перевірки формату рядкової константи. ("text")

isHexadecimalNumber(const std::string& token): визначає, чи є переданий токен шістнадцятковим числом за допомогою регулярного виразу.

isDecimalNumber(const std::string& token): визначає, чи є переданий токен десятковим числом з плаваючою точкою за допомогою регулярного виразу.

isNumericConstant(const std::string& token): визначає, чи є переданий токен числовою константою в мові C# за допомогою регулярного виразу.

isOperator(const std::string& token): визначає, чи є переданий токен оператором в мові C# за допомогою функції std::find для пошуку токена серед заданих операторів.

isDelimiter(const std::string& token): визначає, чи є переданий токен роздільником в мові C# за допомогою функції std::find для пошуку токена серед заданих роздільників.

isPreprocessorDirective(const std::string& token): визначає, чи є переданий токен директивою препроцесора в мові C# за допомогою регулярного виразу. В мові C# директиви препроцесора виглядають наступним чином: #if, #else, #endif і т.д.


```

std::vector<Token> tokenize(const std::string& code) {
    std::vector<Token> tokens;
    std::string token;
    bool isString = false;
    bool isSingleLineComment = false;
    bool isMultiLineComment = false;

    // Iterate over the code character by character
    for (size_t i = 0; i < code.length(); ++i) {
        char c = code[i];

        // Handle comments
        if (c == '/' && !isString && !isMultiLineComment) {
            if (i + 1 < code.length()) {
                if (code[i + 1] == '/') isSingleLineComment = true;
                else if (code[i + 1] == '*') isMultiLineComment = true;
            }
        }

        // Handle end of comments
        if (isSingleLineComment && c == '\n') {
            isSingleLineComment = false;
            tokens.push_back({token, .type: Comment});
            token.clear();
        }
        if (isMultiLineComment && c == '*' && i + 1 < code.length() && code[i + 1] == '/') {
            isMultiLineComment = false;
            token += "*/";
            tokens.push_back({token, .type: Comment});
            token.clear();
            ++i;
            continue;
        }
    }
}

```

```

if (isSingleLineComment || isMultiLineComment) {
    token += c;
    continue;
}

// Handle strings
if (c == '\"') {
    if (isString) {
        token += c;
        tokens.push_back({token, .type: String });
        token.clear();
    }
    else {
        if (!token.empty()) {
            tokens.push_back({token, .type: Unknown });
            token.clear();
        }
    }
    isString = !isString;
}
else if (isString) {
    token += c;
}
else if (std::isspace(c)) {
    if (!token.empty()) {
        tokens.push_back({token, .type: Unknown });
        token.clear();
    }
}
else if (std::ispunct(c) && c != '.') {
    if (!token.empty()) {
        tokens.push_back({token, .type: Unknown });
        token.clear();
    }
}

```

```

    token.clear();
}
tokens.push_back({ .value: std::string( n, c), .type: Unknown });
}
else {
    token += c;
}
}

if (!token.empty()) {
    tokens.push_back({token, .type: Unknown });
}

for (Token &t: tokens) {
    if (isKeyword(t.value)) t.type = Keyword;
    else if (isIdentifier(t.value)) t.type = Identifier;
    else if (isHexadecimalNumber(t.value)) t.type = HexadecimalNumber;
    else if (isDecimalNumber(t.value)) t.type = DecimalNumber;
    else if (isNumericConstant(t.value)) t.type = Number;
    else if (isOperator(t.value)) t.type = Operator;
    else if (isDelimiter(t.value)) t.type = Punctuation;
    else if (isPreprocessorDirective(t.value)) t.type = PreprocessorDirective;
    else if (isStringConstant(t.value)) t.type = String;
}

return tokens;
}

```

Функція `displayTokens` забезпечує вивід результату роботи програми в консоль у вигляді пар <лексема, тип лексеми>

```
void displayTokens(const std::vector<Token>& tokens) {
    for (const Token& token : tokens) {
        std::cout << token.value << " - ";
        switch (token.type) {
            case Keyword: std::cout << "Keyword"; break;
            case Identifier: std::cout << "Identifier"; break;
            case String: std::cout << "String Constant"; break;
            case Number: std::cout << "Numeric Constant"; break;
            case HexadecimalNumber: std::cout << "Hexadecimal Number"; break;
            case DecimalNumber: std::cout << "Decimal Number"; break;
            case Operator: std::cout << "Operator"; break;
            case Punctuation: std::cout << "Delimiter"; break;
            case PreprocessorDirective: std::cout << "Preprocessor Directive"; break;
            case Comment: std::cout << "Comment"; break;
            case Unknown: std::cout << "Unknown"; break;
        }
        std::cout << std::endl;
    }
}
```

В мейні відбувається відкриття та зчитування файлу, і, звичайно, виклик вище перелічених функцій

```
int main() {
    std::ifstream inputFile("s: ../../test.txt");

    if (!inputFile.is_open())
    {
        std::cerr << "Не вдалося відкрити файл" << std::endl;
        return 1;
    }

    std::stringstream buffer;
    buffer << inputFile.rdbuf();
    std::string code = buffer.str();

    inputFile.close();

    std::vector<Token> tokens = tokenize(code);
    displayTokens(tokens);
    return 0;
}
```

[Посилання на GitHub](#)