**Звіт**

**до лабораторної роботи №3**

Оформив студент групи ТТП-31

Казмірчук Володимир

Викладач: Шишацький Андрій

Постановка задачі

Реалізувати лексичний аналізатор мови програмування C. Розфарбувати код написаний цією мовою, вивести пари (лексема, тип лексеми), вивести лексеми, що зустрілися в програмі, розбитими на класи.

Розв’язок

Для вирішення цієї задачі було написано наступний код на мові C++:

#include <iostream>

#include <fstream>

#include <regex>

#include <set>

#include <map>

#include <windows.h>

using namespace std;

// Lexem type

enum type { NONE, STRING, COMMENT, PREPR, RESERVED, NUMBER, OPERATOR };

struct chunk {

    int start, end;

    mutable type type;

};

// Overloading operator for custom sorting

bool operator < (const chunk& a, const chunk& b) { return a.start < b.start; }

// Search given lexem type in string

void parse (const string& str, const string& sreg, set < chunk >& chunks, type type) {

    regex exp = regex (sreg);

    smatch res;

    auto it = chunks.begin ();

    int start = 0, end;

    while (start < str.size ()) {

        while (it != chunks.end () && start == it -> start) {

            start = it -> end;

            it ++;

        }

        if (it == chunks.end ()) { end = str.size (); }

        else { end = it -> start; }

        // Skipalready searched chunks

        while (regex\_search (str.begin () + start, str.begin () + end, res, exp)) {

            int pos = res.suffix ().first - str.begin ();

            chunks.insert ((chunk) { pos - (int) res.str ().size (), pos, type });

            start = pos;

        }

        start = end;

    }

}

int main () {

    // Open and read file

    ifstream fin;

    while (true) {

        cout << "Enter file name: ";

        string fname;

        cin >> fname;

        fin.open (fname);

        if (fin) {

            cout << "The file was opened successfully" << endl;

            break;

        }

        else { cout << "Couldn't open the file!" << endl; }

    }

    string str ((istreambuf\_iterator <char> (fin)), istreambuf\_iterator <char> ());

    fin.close ();

    //cout << str << endl;

    // Perform analysis

    set < chunk> chunks;

    parse (str, R"(\"(?:\\.|[^\"])\*\"|\/\/[^\n]\*|\/\\*[\s\S]\*?\\*\/)", chunks, type::STRING);

    for (auto& it : chunks) {

        if (str [it.start] == '/') { it.type = type::COMMENT; }

    }

    parse (str, R"(^\s\*#.\*$)", chunks, type::PREPR);

    parse (str, R"(\b(?:auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|int|long|register|return|short|signed|sizeof|static|struct|switch|typedef|union|unsigned|void|volatile|while)\b)", chunks, type::RESERVED);

    parse (str, R"(\b(?:0[bB][01]+|0[1-7]+|0[xX][0-9a-fA-F]+|\d+(?:\.\d+)?(?:[eE][+-]?\d+)?)\b)", chunks, type::NUMBER);

    parse (str, R"(\b(?:\+|\-|\\*|=|\!=|\<|\<=|\>|>=|\&|\||\^|\~|\<<|\>>|\&=|\|=|\^=|\?|\:|\,|\-\>)\b)", chunks, type::OPERATOR);

    // Print all found lexems and their locations

    map < int, pair < string, int > > types = {

        {type::NONE, {"TEXT", 7}},

        {type::STRING, {"STRING", 6}},

        {type::COMMENT, {"COMMENT", 2}},

        {type::PREPR, {"PREPROCESSOR", 8}},

        {type::RESERVED, {"RESERVED", 3}},

        {type::NUMBER, {"NUMBER", 5}},

        {type::OPERATOR, {"OPERATOR", 8}}

    };

    for (auto it : chunks) {

        cout << types [it.type].first << ": \t from " <<  it.start << " to " << it.end << endl;

    }

    cout << endl;

    // Print file to console highlighting detected lexems

    auto it = chunks.begin ();

    int start = 0, end;

    HANDLE con = GetStdHandle (STD\_OUTPUT\_HANDLE);

    SetConsoleTextAttribute (con, types [type::NONE].second);

    while (start < str.size ()) {

        while (it != chunks.end () && start == it -> start) {

            SetConsoleTextAttribute (con, types [it -> type].second);

            cout << string (str, start, it -> end - start);

            cout << flush;

            start = it -> end;

            it ++;

        }

        if (it == chunks.end ()) { end = str.size (); }

        else { end = it -> start; }

        SetConsoleTextAttribute (con, types [type::NONE].second);

        cout << string (str, start, end - start);

        cout << flush;

        start = end;

    }

    SetConsoleTextAttribute (con, types [type::NONE].second);

    cout << endl;

    return 0;

}

Цей код виконує аналіз вмісту файлу програми і виділяє різні лексеми, такі як рядки, коментарі, директиви препроцесора, ключові слова, числа та оператори. Потім він виводить результати аналізу, вказуючи тип кожної лексеми та її місце в тексті.

Основні етапи програми:

1. Відкриття файлу:

Користувачеві пропонується ввести ім'я файлу. Потім програма відкриває вказаний файл і зчитує його вміст.

2. Аналіз лексем:

Використовується функція *parse*, яка за допомогою регулярних виразів знаходить вказані типи лексем (рядки, коментарі, директиви препроцесора, ключові слова, числа та оператори) у тексті програми та зберігає їх інформацію в множині *chunks*.

3. Визначення типів лексем:

Створюється мапа *types*, яка містить інформацію про типи лексем, їх назви та кольорове відзначення при виведенні.

4. Вивід результатів:

Для кожної лексеми виводиться інформація про її тип та місце в тексті. Потім виводиться весь файл з виділеними лексемами на консоль.

Загалом, програма слугує для лексичного аналізу та відображення різних частин програмного коду, зазначаючи їхні типи та місцезнаходження у вихідному тексті – чого і вимагає технічне завдання. Важливе зауваження: використання кольорів в консолі залежить від платформи. У даному випадку був використаний Windows API, тому програма буде працювати виключно з системами, що підтримують цю бібліотеку.

Приклади роботи програми

Приклад №1

Вхідний файл:

#include "stdio.h" //comment

int main () {

printf ("Hello, world!\n");

return 0;

}

Вивід:

A computer screen shot of a computer code

Description automatically generated

Приклад №2

Вхідний файл:

#include "stdio.h" // awesome include!

#include "math.h"

const int a = NULL;

int main (int argc, char\* argv []) {

char hw [] = "Hello, world!";

printf ("%s\n", hw);

// This is a string with an escaped quote: "This is an escaped quote: \""

char esc [] = "This is an escaped quote: \"";

char com [] = "String with // a comment and a /\* multiline \*/ comment";

/\* This is a multi-line comment with a string:

"This should not be matched" \*/

int numbers [] = {0, 01, 0b1, 0x1, 0xA, 1, 2, 2.5, 2.25, 2e5, 007, 09, 0b2, 0xZ};

return 0;

}

A black screen with white text

Description automatically generatedВивід:

A screenshot of a computer code

Description automatically generated