НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №1**

з дисципліни

«Основи проектування трансляторів»

Тема: **«Розробка лексичного аналізатора»**

Виконав: студент 3 курсу ФПМ

групи КВ-02

Костюков С. В.

Перевірив: Коваленко О. П.

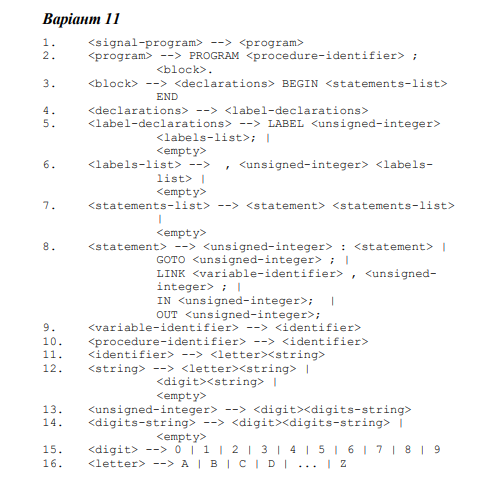
Київ – 2023

**Постановка задачі**

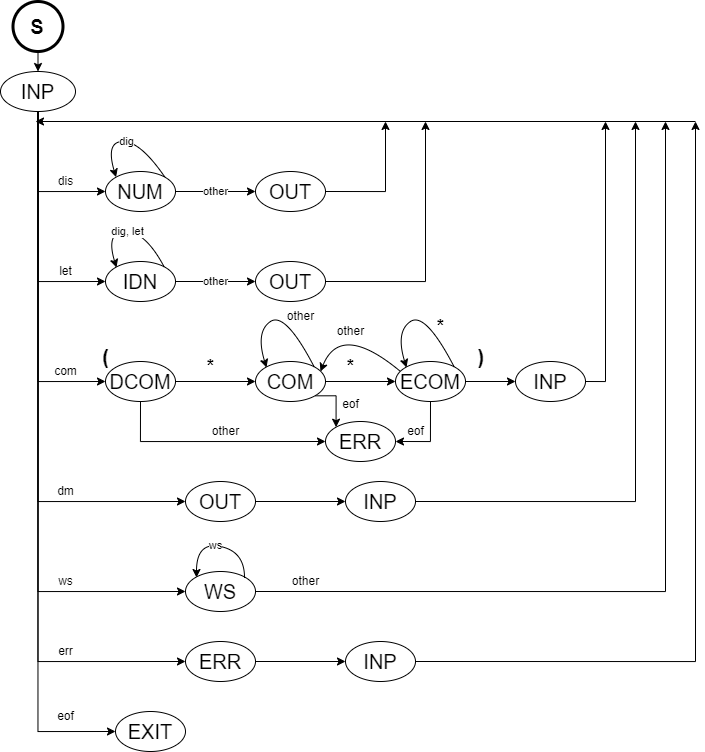
Розробити програму лексичного аналізатора (ЛА) для підмножини мови програмування SIGNAL.

Програма має забезпечувати наступні дії:

* видалення (пропускання) пробільних символів; пробіл, повернення каретки; перехід на новий рядок, горизонтальна та вертикальна табуляція, перехід на нову сторінку;
* згортання ключових слів;
* згортання багато-символьних роздільників;
* згортання констант із занесенням до таблиці значення та типу константи;
* згортання ідентифікаторів;
* видалення коментарів, заданих у вигляді (\*<текст коментарю>\*)
* формування рядка лексем з інформацією про позиції лексем;
* заповнення таблиць ідентифікаторів та констант інформацією, отриманою під час згортки лексем;
* виведення повідомлень про помилки.



**Граф автомату, що визначає алгоритм ЛА:**

****

**Лістинг коду програми ЛА:**

#include <iostream>

#include <fstream>

#include <map>

#include <vector>

#include <iomanip>

**using** std::cout;

**using** std::endl;

**struct** lex {

lex(**int** x, **int** y, **int** z, **const** std::string &n) {

code = x;

line = y;

column = z;

lexem = n;

}

**int** code;

**int** line;

**int** column;

std::string lexem;

};

**bool** is\_digit(**const** **char** &symbol) {

// check is it symbol from 0 to 9

**return** (**int**(symbol) >= 48 && **int**(symbol) <= 57);

}

**bool** is\_alpha(**const** **char** &symbol) {

// check is it symbol from A to Z

**return** (**int**(symbol) >= 65 && **int**(symbol) <= 90);

}

**bool** is\_separator(**const** **char** &symbol) {

// check is it one of symbols: ',', ':', ';', '.'

**return** (**int**(symbol) == 44 || **int**(symbol) == 58 || **int**(symbol) == 59 || **int**(symbol) == 46);

}

**bool** is\_white\_space(**const** **char** &symbol) {

// check is it one of white space symbols

**return** (**int**(symbol) == 32 || **int**(symbol) == 8 || **int**(symbol) == 9 || **int**(symbol) == 10 || **int**(symbol) == 13);

}

**int** Code\_identifier(**const** std::string &buf, std::map<std::string, **int**> &KeyW, std::map<std::string, **int**> &ident,

**int** &curr\_code) {

**if** (KeyW.contains(buf)) {

**return** KeyW[buf];

} **else** **if** (ident.contains(buf)) {

**return** ident[buf];

} **else** {

ident[buf] = curr\_code++;

**return** ident[buf];

}

}

**void** Show\_Table(**const** std::map<std::string, **int**> &data) {

cout << "Key words table: " << endl;

**if** (data.empty()) {

cout << "Empty" << endl;

}

**for** (**auto** &i: data) {

cout << std::setw(10) << std::left << i.first << " | " << i.second << endl;

}

cout << endl;

}

**void** Show\_data(**const** std::vector<lex> &data) {

cout << "Lexemes line: " << endl;

**if** (data.empty()) {

cout << "Empty" << endl;

}

**for** (**auto** &i: data) {

cout << std::setw(3) << std::left << i.line << " | " << std::setw(3) << i.column << " | " << std::setw(5)

<< i.code << " | " << std::setw(10) << i.lexem << endl;

}

cout << endl;

}

**void** file\_parsing(**const** std::string &name) {

std::ifstream file(name);

**if** (!file.is\_open()) {

cout << "Open error" << endl;

} **else** {

**int** count\_line = 1, count\_position = 1;

**int** identifier\_count = 401, labels\_count = 501;

**int** buff\_pos{}, buff\_code{};

std::map<std::string, **int**> KeyWords{{"PROGRAM", 301},

{"BEGIN", 302},

{"END", 303},

{"GOTO", 304},

{"LINK", 305},

{"IN", 306},

{"OUT", 307},

{"LABEL", 308}

};

std::map<std::string, **int**> identifiers, labels;

std::vector<lex> lexemes;

**char** symbol;

std::string buf;

file.get(symbol);

**while** (!file.eof()) {

**if** (is\_digit(symbol)) {

buff\_pos = count\_position;

**do** {

buf += symbol;

file.get(symbol);

count\_position++;

} **while** (is\_digit(symbol));

**if** (!labels.contains(buf)) {

labels[buf] = labels\_count++;

}

lexemes.emplace\_back(lex(labels[buf], count\_line, buff\_pos, buf));

buf.clear();

} **else** **if** (is\_alpha(symbol)) {

buff\_pos = count\_position;

**do** {

buf += symbol;

file.get(symbol);

count\_position++;

} **while** (is\_digit(symbol) || is\_alpha(symbol));

buff\_code = Code\_identifier(buf, KeyWords, identifiers, identifier\_count);

lexemes.emplace\_back(lex(buff\_code, count\_line, buff\_pos, buf));

buf.clear();

} **else** **if** (symbol == '(') {

file.get(symbol);

count\_position++;

**if** (symbol == '\*') {

**while** (file.get(symbol)) {

count\_position++;

**if** (symbol == '\*') {

file.get(symbol);

count\_position++;

**if** (symbol == ')') {

file.get(symbol);

count\_position++;

**break**;

}

**if** (file.eof()) {

cout << "Lexer: Error(line " << count\_line << ", column " << count\_position

<< "): Unlocked comment" << endl;

}

}

**if** (symbol == 10) {

count\_line++;

count\_position = 0;

}

}

**if** (file.eof()) {

cout << "Lexer: Error(line " << count\_line << ", column " << count\_position

<< "): Unlocked comment" << endl;

}

} **else** {

cout << "Lexer: Error(line " << count\_line << ", column " << count\_position

<< "): Incorrect symbol in comment" << endl;

}

} **else** **if** (is\_separator(symbol)) {

buf += symbol;

lexemes.emplace\_back(lex(**int**(symbol), count\_line, count\_position, buf));

buf.clear();

count\_position++;

file.get(symbol);

} **else** **if** (is\_white\_space(symbol)) {

**do** {

**if** (symbol == 10) {

count\_line++;

count\_position = 0;

}

count\_position++;

file.get(symbol);

} **while** (is\_white\_space(symbol));

} **else** {

cout << "Lexer: Error(line " << count\_line << ", column " << count\_position

<< "): Incorrect symbol '" << symbol << "'" << endl;

file.get(symbol);

count\_position++;

}

}

Show\_Table(KeyWords);

Show\_Table(identifiers);

Show\_Table(labels);

Show\_data(lexemes);

}

file.close();

}

**int** main() {

file\_parsing("Program.txt");

**return** 0;

}

**Тестування програми:**

*True-тестування:*

***Текст програми:***

*Program.txt*

PROGRAM PROG;

(\*jqwj\*)

LABEL 12, 13;

(\*

\*\*\*\*ewfwef\*\*\*\*

\*)

BEGIN

12 : GOTO 13 ;

LINK VERIDENT, 13 ;

13 : IN 12 ;

OUT 12 ;

END.

***Отриманий результат:***

Key words table:

BEGIN | 302

END | 303

GOTO | 304

IN | 306

LABEL | 308

LINK | 305

OUT | 307

PROGRAM | 301

Key words table:

PROG | 401

VERIDENT | 402

Key words table:

12 | 501

13 | 502

Lexemes line:

1 | 1 | 301 | PROGRAM

1 | 9 | 401 | PROG

1 | 13 | 59 | ;

3 | 1 | 308 | LABEL

3 | 7 | 501 | 12

3 | 9 | 44 | ,

3 | 11 | 502 | 13

3 | 13 | 59 | ;

9 | 1 | 302 | BEGIN

10 | 1 | 501 | 12

10 | 4 | 58 | :

10 | 6 | 304 | GOTO

10 | 11 | 502 | 13

10 | 14 | 59 | ;

11 | 4 | 305 | LINK

11 | 9 | 402 | VERIDENT

11 | 17 | 44 | ,

11 | 19 | 502 | 13

11 | 22 | 59 | ;

12 | 1 | 502 | 13

12 | 4 | 58 | :

12 | 6 | 306 | IN

12 | 9 | 501 | 12

12 | 12 | 59 | ;

13 | 5 | 307 | OUT

13 | 9 | 501 | 12

13 | 12 | 59 | ;

14 | 1 | 303 | END

14 | 4 | 46 | .

*False-тестування:*

***Текст програми:***

PROGRAM PROG;

(\*jqwj\*)

LABEL 12, 13;

BEGIN /

12 : GOTO 13 ;

LINK VERIDENT, 13 ;

13 : IN 12 ; ?

OUT 12 ;

END.

***Отриманий результат:***

Lexer: Error(line 5, column 7): Incorrect symbol '/'

Lexer: Error(line 8, column 14): Incorrect symbol '?'

Key words table:

BEGIN | 302

END | 303

GOTO | 304

IN | 306

LABEL | 308

LINK | 305

OUT | 307

PROGRAM | 301

Key words table:

PROG | 401

VERIDENT | 402

Key words table:

12 | 501

13 | 502

Lexemes line:

1 | 1 | 301 | PROGRAM

1 | 9 | 401 | PROG

1 | 13 | 59 | ;

3 | 1 | 308 | LABEL

3 | 7 | 501 | 12

3 | 9 | 44 | ,

3 | 11 | 502 | 13

3 | 13 | 59 | ;

5 | 1 | 302 | BEGIN

6 | 1 | 501 | 12

6 | 4 | 58 | :

6 | 6 | 304 | GOTO

6 | 11 | 502 | 13

6 | 14 | 59 | ;

7 | 4 | 305 | LINK

7 | 9 | 402 | VERIDENT

7 | 17 | 44 | ,

7 | 19 | 502 | 13

7 | 22 | 59 | ;

8 | 1 | 502 | 13

8 | 4 | 58 | :

8 | 6 | 306 | IN

8 | 9 | 501 | 12

8 | 12 | 59 | ;

9 | 5 | 307 | OUT

9 | 9 | 501 | 12

9 | 12 | 59 | ;

10 | 1 | 303 | END

10 | 4 | 46 | .