

Федеральное государственное автономное  
образовательное учреждение  
высшего образования  
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий  
институт

Кафедра информатики  
кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

Разработка лексического анализа компилятора  
тема

Преподаватель

А. С. Кузнецов

Студент

КИ19-04-1М, 031943329

номер группы, зачетной книжки

подпись, дата

подпись, дата

инициалы, фамилия

И. С. Байкалов

инициалы, фамилия

Красноярск 2020

## СОДЕРЖАНИЕ

1 Цель.....	3
2 Ход работы .....	3
3 Вывод .....	7

## 1 Цель

Изучить методы лексического анализа и их программную реализацию.

## 2 Ход работы

Файл flex делится на 3 части, а именно:

- объявление переменных, констант;
- правила;
- вспомогательные функции.

В ходе работы была реализован код, представленный в таблице 1.

Таблица 1 – листинг кода tokens.l

```
%{
    #include <stdio.h>
    int chars = 0;
}%

%option yylineno
CHAR      [^"\n]
DIGIT     [0-9]
%%

\n                                { chars = 0; }

[ \t\r]                            { chars++; }

{DIGIT}+"."{DIGIT}*                { printf("%s\t\t->\t FLOAT\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
{DIGIT}+                            { printf("%s\t\t->\t
INTEGER\t\t(%d:%d)\n", yytext, yylineno, ++chars); chars +=
strlen(yytext) - 1; }
\'{CHAR}{1}\'                      { printf("%s\t\t->\t CHAR\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\"{CHAR}*\"                          { printf("%s\t\t->\t
STRING\t\t(%d:%d)\n", yytext, yylineno, ++chars); chars +=
strlen(yytext) - 1; }

integer|float|string|char          { printf("%s\t\t->\t DATA
TYPE\t\t(%d:%d)\n", yytext, yylineno, ++chars); chars += strlen(yytext) -
1; }
if|else|elif|while                  { printf("%s\t\t->\t
KEYWORD\t\t(%d:%d)\n", yytext, yylineno, ++chars); chars +=
strlen(yytext) - 1; }

[a-zA-Z][a-zA-Z0-9]*               { printf("%s\t\t->\t
```

```

IDENTIFIER\t(%d:%d)\n", yytext, yylineno, ++chars); chars +=
strlen(yytext) - 1; }

\+ { printf("%s\t\t->\t PLUS\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\= { printf("%s\t\t->\t MINUS\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\/ { printf("%s\t\t->\t DIVISION\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\* { printf("%s\t\t->\t MULTIPLY\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\= { printf("%s\t\t->\t ASSIGN\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\=\= { printf("%s\t\t->\t EQUALS\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\< { printf("%s\t\t->\t LESS\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\> { printf("%s\t\t->\t MORE\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }

\{ { printf("%s\t\t->\t OPENPAREN\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\} { printf("%s\t\t->\t CLOSEPAREN\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\{ { printf("%s\t\t->\t OPENBRACE\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }
\} { printf("%s\t\t->\t CLOSEBRACE\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }

\. { printf("%s\t\t->\t DOT\t\t(%d:%d)\n", yytext,
yylineno, ++chars); chars += strlen(yytext) - 1; }
\, { printf("%s\t\t->\t COMMA\t\t(%d:%d)\n", yytext,
yylineno, ++chars); chars += strlen(yytext) - 1; }
\: { printf("%s\t\t->\t COLON\t\t(%d:%d)\n", yytext,
yylineno, ++chars); chars += strlen(yytext) - 1; }
\; { printf("%s\t\t->\t SEMICOLON\t\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }

\\/\\S*[a-zA-Z][a-zA-Z0-9]* { printf("%s\t->\t COMMENT\t(%d:%d)\n",
yytext, yylineno, ++chars); chars += strlen(yytext) - 1; }

\"{CHAR}*${ printf("%s\t\t->\t STRING
ERROR\t\t(%d:%d)\n", yytext, yylineno, ++chars); chars +=
strlen(yytext) - 1; }

. { chars++; printf("Error in line
%d:%d\n", yylineno, chars); exit(1); }

%%

int yywrap(){ return 0; }

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

    char filename [255];

    FILE*file;

```

```
do {
    printf ("path to file: ");
    scanf ("%s", filename);
    file = fopen(filename, "r");
    if (file == NULL) {
        printf ("Can't open file \n");
    }
} while(file == NULL);

yyin = file;

yylex();
fclose(file);
printf("\nOlmeca is good\n");
}
```

Далее, после написания кода на flex, исходный код нужно пропустить через компилятор flex командой `flex tokens.l` и получить файл `lex.yy.c`.

После этого файл `lex.yy.c` нужно пропустить через компилятор C командой `cc lex.yy.c` и получить файл `a.out`, который уже можно запустить в терминале.

Для тестирования использовались 3 файла `1.olm`, `2.olm` и `3.olm`.

На рисунках 4 - 6 можно увидеть результат выполнения работы.

```
Терминал
EnotBoris:tests ilyabaikalow$ ../a.out
path to file: 1.olm
integer      -> DATA TYPE      (1:1)
cash         -> IDENTIFIER      (1:9)
;            -> SEMICOLON      (1:13)
string       -> DATA TYPE      (2:1)
name         -> IDENTIFIER      (2:8)
=            -> ASSIGN          (2:13)
"Ilya"       -> STRING          (2:15)
;            -> SEMICOLON      (2:21)
cash         -> IDENTIFIER      (4:1)
=            -> ASSIGN          (4:6)
1403         -> INTEGER         (4:8)
if           -> KEYWORD         (6:1)
(            -> OPENPAREN        (6:4)
cash         -> IDENTIFIER      (6:5)
>            -> MORE            (6:10)
1203         -> INTEGER         (6:12)
)            -> CLOSEPAREN       (6:16)
{            -> OPENBRACE        (6:18)
cash         -> IDENTIFIER      (7:3)
=            -> ASSIGN          (7:8)
cash         -> IDENTIFIER      (7:10)
-            -> MINUS           (7:15)
1203         -> INTEGER         (7:17)
;            -> SEMICOLON      (7:21)
}            -> CLOSEBRACE       (8:1)

Olmeca is good
EnotBoris:tests ilyabaikalow$ |
```

Рисунок 4 – Таблица токенов для 1.olm

```
Терминал
EnotBoris:tests ilyabaikalow$ ../a.out
path to file: 2.olm
integer      -> DATA TYPE      (1:1)
cash         -> IDENTIFIER      (1:9)
;            -> SEMICOLON      (1:13)
char         -> DATA TYPE      (2:1)
ch           -> IDENTIFIER      (2:6)
=            -> ASSIGN          (2:9)
'e'          -> CHAR            (2:11)
string       -> DATA TYPE      (3:1)
name         -> IDENTIFIER      (3:8)
;            -> SEMICOLON      (3:12)
name         -> IDENTIFIER      (5:1)
=            -> ASSIGN          (5:6)
"Ilya"       -> STRING          (5:8)
;            -> SEMICOLON      (5:14)
cash         -> IDENTIFIER      (6:1)
=            -> ASSIGN          (6:6)
1403         -> INTEGER         (6:8)
if           -> KEYWORD         (8:1)
(            -> OPENPAREN        (8:4)
cash         -> IDENTIFIER      (8:5)
>            -> MORE            (8:10)
1203         -> INTEGER         (8:12)
)            -> CLOSEPAREN       (8:16)
{            -> OPENBRACE        (8:18)
cash         -> IDENTIFIER      (9:3)
=            -> ASSIGN          (9:8)
cash         -> IDENTIFIER      (9:10)
-            -> MINUS           (9:15)
1203         -> INTEGER         (9:17)
;            -> SEMICOLON      (9:21)
}            -> CLOSEBRACE       (10:1)
else         -> KEYWORD         (10:3)
{            -> OPENBRACE        (10:8)
cash         -> IDENTIFIER      (11:3)
=            -> ASSIGN          (11:8)
cash         -> IDENTIFIER      (11:10)
+            -> PLUS            (11:15)
1203         -> INTEGER         (11:17)
}            -> CLOSEBRACE       (12:1)

Olmeca is good
EnotBoris:tests ilyabaikalow$
```

Рисунок 5 – Таблица токенов для 2.olm

```
Терминал
EnotBoris:tests ilyabaikalow$ ../a.out
path to file: 3.olm
//variables -> COMMENT (1:1)
integer -> DATA TYPE (2:1)
cash -> IDENTIFIER (2:9)
; -> SEMICOLON (2:13)
string -> DATA TYPE (3:1)
name -> IDENTIFIER (3:8)
; -> SEMICOLON (3:12)
name -> IDENTIFIER (5:1)
= -> ASSIGN (5:6)
"Ilya" -> STRING (5:8)
; -> SEMICOLON (5:14)
cash -> IDENTIFIER (6:1)
= -> ASSIGN (6:6)
1403 -> INTEGER (6:8)
while -> KEYWORD (8:1)
( -> OPENPAREN (8:7)
cash -> IDENTIFIER (8:8)
> -> MORE (8:13)
1203 -> INTEGER (8:15)
) -> CLOSEPAREN (8:19)
{ -> OPENBRACE (8:21)
if -> KEYWORD (9:3)
( -> OPENPAREN (9:6)
name -> IDENTIFIER (9:7)
== -> EQUALS (9:12)
"Ilya" -> STRING (9:15)
) -> CLOSEPAREN (9:21)
{ -> OPENBRACE (9:23)
cash -> IDENTIFIER (10:5)
= -> ASSIGN (10:10)
cash -> IDENTIFIER (10:12)
- -> MINUS (10:17)
10 -> INTEGER (10:19)
; -> SEMICOLON (10:21)
} -> CLOSEBRACE (11:3)
} -> CLOSEBRACE (12:1)

Olmeca is good
```

Рисунок 6 – Таблица токенов для 3.olm

### 3 Вывод

В ходе лабораторной работы бел реализован лексический анализатор с использованием flex. Анализатор выводит в консоль таблицу токенов, также показывает номер строки и позицию в строке.