|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**Лабораторная работа № 6**

**«Порождение лексического анализатора с помощью flex»**

***по курсу «Конструирование компиляторов»***

Студент *Ионов Т.Р. 61Б*

Преподаватель *Коновалов А.B.*

*Москва, 2022 г.*

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc106224508)

[2 Индивидуальное задание 3](#_Toc106224509)

[3 Реализация 3](#_Toc106224510)

[4 Тестирование 4](#_Toc106224511)

[Вывод 4](#_Toc106224512)

# 1 Цель работы

Целью данной работы является изучение генератора лексических анализаторов flex.

# 2 Индивидуальное задание

Целые числа одинарной длины: последовательности десятичных цифр. Целые числа двойной длины: последовательности десятичных цифр с точкой на конце. Слова: любые последовательности непробельных символов, не являющихся числами.

3 Реализация

Листинг 1 −lexer.

%option noyywrap bison-bridge bison-locations

%{

#include <stdio.h>

#include <stdlib.h>

#define TAG\_EOF 0

#define TAG\_FLOAT 1

#define TAG\_DOUBLE 2

#define TAG\_WORD 3

#define TAG\_SPACE 4

#define TAG\_EOL 5

#define TAG\_ERROR 6

#include <float.h>

#include <limits.h>

const char \*tag\_names [] = {

"EOF", "FLOAT", "DOUBLE", "WORD", "SPACE", "EOL", "ERROR"

};

struct Position {

int line, pos, index;

};

void print\_pos(struct Position \*p) {

printf("(%d,%d)", p->line, p->pos);

}

struct Fragment {

struct Position starting, following;

};

typedef struct Fragment YYLTYPE;

void print\_frag(struct Fragment \*f) {

print\_pos(&(f->starting));

printf("-");

print\_pos(&(f->following));

}

union Token {

char \*word;

float float\_value;

double double\_value;

};

typedef union Token YYSTYPE;

int continued;

struct Position cur;

#define YY\_USER\_ACTION \

{ \

int i; \

if (!continued) \

yylloc->starting = cur; \

continued = 0; \

\

for (i = 0; i < yyleng; i++) { \

if (yytext[i] == '\n') { \

cur.line++; \

cur.pos = 1; \

} else { \

cur.pos++; \

} \

cur.index++; \

} \

yylloc->following = cur; \

}

long size;

void init\_scanner(char \*program) {

continued = 0;

cur.line = 1;

cur.pos = 1;

cur.index = 0;

yy\_scan\_string(program);

}

void err(const char \*msg) {

printf("ERROR");

print\_pos(&cur);

printf(": %s\n", msg);

}

%}

SPACE [ \t\r]+

EOL [\n]+

DIGIT [0-9]

FLOAT {DIGIT}+

DOUBLE {DIGIT}+\.

WORD [^ \t\r\n]+

%%

{EOL} {

yylval->word = (char\*)("");

return TAG\_EOL;

}

{SPACE} {

yylval->word = (char\*)(" ");

return TAG\_SPACE;

}

{FLOAT} {

char \* endptr;

float value = strtof(yytext, &endptr);

if (value > FLT\_MAX) {

return TAG\_ERROR;

}

yylval->float\_value = value;

return TAG\_FLOAT;

}

{DOUBLE} {

char \* endptr;

double value = strtod(yytext, &endptr);

if (value > DBL\_MAX) {

return TAG\_ERROR;

}

yylval->double\_value = value;

return TAG\_DOUBLE;

}

{WORD} {

size += yyleng;

yylval->word = yytext;

return TAG\_WORD;

}

. {

err("");

yylval->word = (char\*)("");

return TAG\_ERROR;

}

%%

int main(int argc, const char \*\*argv) {

int tag;

YYSTYPE token;

YYLTYPE coords;

long size\_str;

char \*str;

FILE \*input;

input = fopen(argv[1], "rb");

if (input == NULL) {

fputs("cannot open file", stderr);

exit(1);

}

fseek(input, 0, SEEK\_END);

size\_str = ftell(input);

rewind(input);

str = (char\*)malloc(sizeof(char)\*(size\_str+1));

if (str == NULL) {

fputs("malloc error", stderr);

exit(1);

}

fread(str, sizeof(char), size\_str, input);

str[size\_str] = '\0';

fclose(input);

init\_scanner(str);

do {

tag = yylex(&token, &coords);

printf("%s ", tag\_names[tag]);

print\_frag(&coords);

if (tag == TAG\_WORD || tag == TAG\_SPACE || tag == TAG\_EOL) {

printf(": %s\n", token.word);

}

else if (tag == TAG\_FLOAT) {

printf(": %f\n", token.float\_value);

}

else if (tag == TAG\_DOUBLE) {

printf(": %f\n", token.double\_value);

}

else {

printf("\n");

}

} while (tag != TAG\_EOF);

printf("\n");

free(str);

return 0;

}

# 4 Тестирование

Вход:

4242

1881.

w0rd 12e3.

1111340282346638528859811704183484516925440 // for overflow

Результат:

FLOAT (1,1)-(1,5): 4242.000000

EOL (1,5)-(2,1):

DOUBLE (2,1)-(2,6): 1881.000000

EOL (2,6)-(3,1):

WORD (3,1)-(3,5): w0rd

SPACE (3,5)-(3,9):

WORD (3,9)-(3,14): 12e3.

EOL (3,14)-(4,1):

ERROR (4,1)-(4,44)

EOL (4,44)-(5,1):

EOF (4,44)-(5,1)

# Вывод

В результате выполнения данной лабораторной работы был изученен генератор лексических анализаторов flex.