Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

Дисциплина «Теория языков программирования и методы трансляции»

Отчет к лабораторной работе № 7

«Построение лексического анализатора с помощью YACC»

Выполнил:

Василения Иван Валерьевич

Мельниченко Артём Олегович

Принял:

Гордиенко А. П.

Орёл, 2024г

Листинг файла grammar.y  
%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "attr\_checker.h"

%}

%union {

char\* str;

int num;

}

%token <str> SELECT FROM WHERE AND OR COMMA \_STRING COMPARISON

%token <num> NUM

%%

goal: SELECT attr\_list FROM \_STRING where\_case

{

//printf("%s\n", $4);

if (find\_table\_name($4)) {

printf("Такая строка есть\n");

}

else {

printf("Такой строки нет\n");

}

printf("\033[32mВыражение принадлежит языку\033[0m\n");

}

;

attr\_list: \_STRING

{

if (find\_table\_attr($1)) {

printf("Такой атрибут есть\n");

}

else {

printf("Такого атрибута нет\n");

}

//printf("%s\n", $1);

}

| \_STRING COMMA attr\_list

{

if (find\_table\_attr($1)) {

printf("Такой атрибут есть\n");

}

else {

printf("Такого атрибута нет\n");

}

//printf("%s\n", $1);

}

;

where\_case: WHERE condition\_list

| /\* epsilon \*/ ;

condition\_list: condition

| condition AND condition\_list

| condition OR condition\_list;

condition: \_STRING COMPARISON NUM;

%%

int main (void) {

/\*

file = fopen("table.db", "r");

if (file == NULL) {

perror("Error opening file");

return -1;

}

char buffer[256];

\*/

start\_scan();

print\_attrs();

return yyparse();

}

void yyerror(const char\* s) {

fprintf(stderr, "Ошибка: %s\n", s);

}

Листинг файла flex.l

%{

#include <stdio.h>

#include <stdlib.h>

#include "y.tab.h"

void yyerror(const char \*s);

%}

delim [ \t\n\r]+

ws {delim}+

letter [a-zA-Z]

digit [0-9]

integer {digit}+

decimal {digit}+[.,]{digit}+

id {letter}+"[A-Za-z0-9]"\*

operator [+\-\*/]

comp [<>]=|[<>]|=

%option noyywrap

%%

{ws} { /\* Игнорируем пробелы \*/ }

{integer} { yylval.num = atoi(yytext); return NUM; }

"SELECT" { return SELECT; }

"FROM" { return FROM; }

"WHERE" { return WHERE; }

"OR" { return OR; }

"AND" { return AND; }

"," { return COMMA; }

{comp} { return COMPARISON; }

[a-zA-Z]\* { yylval.str = strdup(yytext); return \_STRING; }

<<EOF>> { return 0;}

. { printf("Ошибка: неожиданный символ: %d %s\n", (int)yytext[0], yytext); exit(1); }

%%

Листинг файла attr\_checker.h

#ifndef ATTR\_CHECKER\_H

#define ATTR\_CHECKER\_H

#include <stdbool.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#define ATTRS\_INIT\_VALUE 20

FILE\* file;

char\* attrs[ATTRS\_INIT\_VALUE];

int attrs\_count;

bool find\_table\_name(char\* name) {

printf("%s <=> %s\n", attrs[0], name);

if(strcmp(name, attrs[0]) == 0) {

return true;

}

else

return false;

}

bool find\_table\_attr(char\* name) {

if (attrs\_count < 2 ) return false;

for (int i = 1; i < attrs\_count; i++) {

if (attrs[i] == NULL) return false;

printf("%s <=> %s\n", attrs[i], name);

if (strcmp(name, attrs[i]) == 0)

return true;

}

return false;

}

bool open\_file() {

file = fopen("table.db", "r");

if (file == NULL) {

return false;

}

return true;

}

void scan\_attrs() {

attrs\_count = 0;

char buffer[32];

while (fgets(buffer, sizeof(buffer), file) != NULL) {

printf("Buffer: %s", buffer);

attrs[attrs\_count] = (char \*)malloc(strlen(buffer) + 1);

if (attrs[attrs\_count] == NULL) {

printf("Ошибка при выделении памяти");

return;

}

size\_t len = strlen(buffer);

buffer[len-2] = '\0';

strcpy(attrs[attrs\_count], buffer);

attrs\_count++;

}

printf("\nAttr\_count: %d\n", attrs\_count);

}

void start\_scan() {

open\_file();

scan\_attrs();

}

void print\_attrs() {

for (int i = 0; i < attrs\_count; i++) {

printf("%s", attrs[i]);

}

int size = sizeof(attrs) / sizeof(attrs[0]);

printf("\nArray size: %d elements\n", size);

}

#endif