Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

Дисциплина «Теория языков программирования и методы трансляции»

Отчет к лабораторной работе № 4

«Таблично управляемый синтаксический разбор сверху вниз»

Выполнил:

Василения Иван Валерьевич

Мельниченко Артём Олегович

Принял:

Гордиенко А. П.

Орёл, 2024г

Грамматика:

goal -> 'CREATE' + 'TABLE' + name + '(' + attr\_init + pkey + fkey + ')'

name -> STRING

names\_list -> name + names\_list'

names\_list' -> comma + name + names\_list'

names\_list' -> э

comma -> ','

attr\_init -> attr + attr\_init

attr\_init -> э

attr -> name + type + if\_null + comma

if\_null -> ('not' | э) + 'null'

type -> ('int' | 'char' + '(' + NUM + ')' )

pkey -> 'PRIMARY' + 'KEY' + '(' + names\_list + ')'

fkey -> comma + 'FOREIGN' + 'KEY' + '(' + names\_list + ')' + 'REFERENCES' + name + '(' + names\_list + ')' + fkey

fkey -> э

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| символ | шаг | | | first |
| 1 | 2 | 3 |
| CREATE | {CREATE} |  |  | {'CREATE'} |
| TABLE | {TABLE} |  |  | {'TABLE'} |
| LBRACKET | {LBRACKET} |  |  | {'('} |
| RBRACKET | {RBRACKET} |  |  | {')'} |
| COMMA | {COMMA} |  |  | {','} |
| INT | {INT} |  |  | {'INT'} |
| CHAR | {CHAR} |  |  | {'CHAR'} |
| NUM | {NUM} |  |  | {NUM} |
| NOT | {NOT} |  |  | {'NOT'} |
| NULL | {NULL} |  |  | {'NULL'} |
| PRIMARY | {PRIMARY} |  |  | {'PRIMARY'} |
| FOREIGN | {FOREIGN} |  |  | {'FOREIGN'} |
| KEY | {KEY} |  |  | {'KEY'} |
| goal |  |  | {'CREATE'} | {'CREATE'} |
| name |  |  | {STRING} | {STRING} |
| names\_list |  |  | {STRING} | {STRING} |
| names\_list' |  | э | {','} | {',', э} |
| attr\_init |  | э | {STRING} | {STRING, э} |
| attr |  |  | {STRING} | {STRING} |
| if\_null |  |  | {'NOT', 'NULL'} | {'NOT', 'NULL'} |
| type |  |  | {'INT', 'CHAR'} | {'INT', 'CHAR'} |
| pkey |  |  | {'PRIMARY'} | {'PRIMARY'} |
| fkey |  | э | {','} | {',', э} |

Рисунок 1 - Таблица FIRST

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Нетерминал | шаги | | | | follow |
|  | 1 | 2 | 3 | 4 |  |
| goal | {$} |  |  |  | {$} |
| names\_list |  | ) |  |  | {)} |
| names\_list' |  |  | ) |  | {)} |
| attr\_init |  | PRIMARY |  |  | {PRIMARY} |
| attr |  | string |  | PRIMARY | {stringPRIMARY} |
| if\_null |  | "," |  |  | {","} |
| type |  | not , null |  |  | {not , null} |
| pkey |  | "," |  |  | {","} |
| fkey |  | ) | "," |  | {)","} |

Рисунок 2 – Таблица FOLLOW

Листинг анализатора:

#ifndef ANALYZER\_H

#define ANALYZER\_H

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include "stack.h"

#include "parse\_tables.h"

extern int yylex();

extern char\* yytext;

stack\* parse\_stack;

#define DEBUG\_MODE true

void debug\_info(char\* text) {

if (DEBUG\_MODE) {

printf("\033[36m%s\033[0m\n", text);

}

}

void show\_stack(stack\* s) {

if (!DEBUG\_MODE) return;

printf("Stack content: ");

for (int i = 0; i <= s->head\_num; i++) {

printf("\033[35m%d, \033[0m", s->values[i]);

}

printf("\n");

}

void alloc\_stack\_init() {

parse\_stack = stack\_alloc();

}

void drop\_parse\_error(int token) {

fprintf(stderr, "\033[31mParse error! token: %d | text: %s\033[0m\n", token, yytext);

exit(EXIT\_FAILURE);

}

void parse\_stack\_substitute(int value) {

for (int i = 0; i < TODO\_TABLE\_COLS; i++) {

if (todo\_table[value][i] >= 0)

printf("val: %d, ", todo\_table[value][i]);

stack\_push(parse\_stack, todo\_table[value][i]);

}

printf("\n");

}

void parse\_start() {

alloc\_stack\_init();

tables\_init();

int curr\_token = yylex();

stack\_push(parse\_stack, GOAL);

while (parse\_stack->head\_num >= 0)

{

show\_stack(parse\_stack);

if (stack\_top(parse\_stack) >= GOAL) {

debug\_info("nonterm detected");

int top = stack\_pop(parse\_stack);

int index = top - GOAL;

int action = parse\_table[index][curr\_token];

printf("Action: %d\n | Row: %d, Col: %d\n", action, index, curr\_token);

if (action < 0) {

drop\_parse\_error(curr\_token);

}

else {

parse\_stack\_substitute(action);

}

}

else {

debug\_info("term detected");

int top = stack\_pop(parse\_stack);

if (top < 0 || top != curr\_token)

drop\_parse\_error(curr\_token);

debug\_info(yytext);

curr\_token = yylex();

}

}

printf("Выражение принадлежит языку\n");

}

#endif