УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе № 3

по предмету «Компиляторные технологии»

Вариант 9

|  |  |  |
| --- | --- | --- |
| Выполнил  Студент гр. 351004 |  | А.И. Бражалович |
| Проверил |  | С.В. Болтак |

Минск 2024

**Задание**

Для заданного вариантом языка разработать:

* лексический анализатор, распознающий необходимые лексемы;
* грамматику языка, включающую в себя базовые конструкции языка;
* программное средство, проверяющее исходный код программы на соответствие грамматике.

Отчёт по выполненной работе

**Заданный язык:**

| Номер варианта | Язык/формат | Примечания |
| --- | --- | --- |
| 9 | PHP |  |

**Лексический анализатор, распознающий необходимые лексемы:**

%{

#include <stdio.h>

int lineno = 1;

int column = 1;

%}

%option noyywrap

ECH echo

SMCOL ;

COND [0-9]

NEXT \n|\t|\r\n

LIT \".\*\"

%%

{NEXT} { if (yytext[0] == '\n') { lineno++; column = 1; } else { column += yyleng; } printf("0 %d %d\n", lineno, column); }

[ ] { column += yyleng; printf("1 %d %d\n", lineno, column); }

"<?" { column += yyleng; printf("2 %d %d\n", lineno, column); }

"?>" { column += yyleng; printf("3 %d %d\n", lineno, column); }

php { column += yyleng; printf("4 %d %d\n", lineno, column); }

[(] { column += yyleng; printf("5 %d %d\n", lineno, column); }

[)] { column += yyleng; printf("6 %d %d\n", lineno, column); }

[{] { column += yyleng; printf("7 %d %d\n", lineno, column); }

[}] { column += yyleng; printf("8 %d %d\n", lineno, column); }

{ECH} { column += yyleng; printf("9 %d %d\n", lineno, column); }

if { column += yyleng; printf("10 %d %d\n", lineno, column); }

{COND} { column += yyleng; printf("11 %d %d\n", lineno, column); }

else { column += yyleng; printf("12 %d %d\n", lineno, column); }

while { column += yyleng; printf("13 %d %d\n", lineno, column); }

{LIT} { column += yyleng; printf("14 %d %d\n", lineno, column); }

{SMCOL} { column += yyleng; printf("15 %d %d\n", lineno, column); }

%%

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

yylex();

return 0;

}

**Грамматика языка, включающая в себя базовые конструкции языка:**

**StartOperator** -> STPROG PHP

**Operator** -> **OperatorIf** / **OperatorEcho** / **OperatorWhile / OperatorBlock**

**OperatorIf** -> IF OPEN\_BRACKET CONDITION CLOSE\_BRACKET **OperatorBlock** / IF OPEN\_BRACKET CONDITION CLOSE\_BRACKET **OperatorBlock** ELSE **OperatorBlock**

**OperatorEcho** -> ECHO LITTERAL SEMICOLON

**OperatorWhile** -> WHILE OPEN\_BRACKET CONDITION CLOSE\_BRACKET **OperatorBlock**

**OperatorBlock** -> OPEN\_FIGURE **Operator** CLOSE\_FIGURE

**EndOperator** -> ENDPROG

Исходный код программного средства, **проверяющее исходный код программы на соответствие грамматике:**

#include <stdio.h>

#include <stdbool.h>

#define MAX 500

int lineno[MAX];

int column[MAX];

bool operator();

bool echo();

bool o\_if();

bool o\_while();

bool block();

typedef enum Types {

NEXT = 0,

SPACE,

STPROG,

ENDPROG,

PHP,

OPEN\_BRACKET,

CLOSE\_BRACKET,

OPEN\_FIGURE,

CLOSE\_FIGURE,

ECHO,

IF,

CONDITION,

ELSE,

WHILE,

LITTERAL,

SEMICOLON,

IDLE

} lex;

const char \*TypeStrings[] = { "NEXT", "SPACE", "STPROG", "ENDPROG", "PHP", "OPEN\_BRACKET", "CLOSE\_BRACKET", "OPEN\_FIGURE", "CLOSE\_FIGURE", "ECHO", "IF", "CONDITION", "ELSE", "WHILE", "LITTERAL", "SEMICOLON" };

int next = 0, count = 0;

int token[MAX];

int error\_line = 0;

int error\_column = 0;

char\* error\_token = NULL;

void save\_error\_position() {

error\_line = lineno[next];

error\_column = column[next];

error\_token = TypeStrings[token[next]];

}

bool term(const lex expected) {

if (token[next] != expected)

return false;

if ((error\_line < lineno[next]) || (error\_line == lineno[next] && error\_column < column[next])) {

save\_error\_position();

}

next++;

return true;

}

bool current(const lex expected) {

return token[next] == expected;

}

bool start() {

bool res = true;

if (!term(STPROG))

res = false;

if (!term(PHP))

res = false;

while (!current(ENDPROG) && res)

if (!operator())

res = false;

if (!term(ENDPROG))

res = false;

if (!res)

printf("Error at line %d, column %d after %s\n", error\_line, error\_column, error\_token);

return res;

}

bool operator() {

int save = next;

if (echo())

return true;

next = save;

if (o\_if())

return true;

next = save;

if (o\_while())

return true;

next = save;

if (block())

return true;

return false;

}

bool echo() {

return term(ECHO) &&

term(LITTERAL) &&

term(SEMICOLON);

}

bool o\_if() {

int save = next;

return (next = save, term(IF) && term(OPEN\_BRACKET) && term(CONDITION) && term (CLOSE\_BRACKET) && block() && term(ELSE) && block()) ||

(next = save, term(IF) && term(OPEN\_BRACKET) && term(CONDITION) && term (CLOSE\_BRACKET) && block());

}

bool o\_while() {

int save = next;

return (next = save, term(WHILE) && term(OPEN\_BRACKET) && term(CONDITION) && term (CLOSE\_BRACKET) && block());

}

bool block() {

bool res = true;

if (!term(OPEN\_FIGURE))

res = false;

else {

while (!current(CLOSE\_FIGURE) && res)

if (!operator())

res = false;

if (!term(CLOSE\_FIGURE))

res = false;

}

return res;

}

void read() {

int temp, line, col;

char\* path = "D:\\BSUIR\\Compiler Technologies\\Lab3\\SecTry\\init.txt";

FILE \*file = fopen(path, "r+t");

fseek(file, 0, SEEK\_SET);

if (file == NULL) {

perror("Ошибка при открытии файла");

}

while (fscanf(file, "%d %d %d", &temp, &line, &col) == 3) {

if ((temp != SPACE) && (temp != NEXT)) {

token[count] = temp;

lineno[count] = line;

column[count++] = col;

}

}

fclose(file);

}

int main(void)

{

read();

printf(start() ? "Accept\n" : "Reject\n");

return 0;

}

Тестирование программного средства**:**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Теста** | **Вид тестирования** | **Предполагаемый результат** | **Результат** |
| **1** | <?php  if (0) {  if (3) {  while (3) {  echo "fd";  }  }  } else {  echo "df";  }  ?> | **Успешная работа программы и вывод дерева в файл с результатом** | **Результат соответствует предполагаемым** |
| **2** | <?php  if (0) {  if (3) {  while (3) {  echo "fd"; ыва  }  }  } else {  echo "df";  }  ?> | **Неверно. Вывод сообщения об ошибке.** | Error at line 5, column 23 after SEMICOLON |
| **3** | <?php  if (0) {  if (3) {  while () {  echo "fd";  }  }  } else {  echo "df";  }  ?> | **Неверно. Вывод сообщения об ошибке.** | Error at line 4, column 16 after OPEN\_BRACKET |

Дерево разбора**:**

\*дерево разбора построено на примере первого теста\*

