Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет (институт) Информационных технологий

Кафедра Прикладная математика

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Е.Н.Крючкова

(подпись преподавателя) (инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

Отчет

по лабораторной (практической) работе №\_2-3\_

\_\_\_\_\_\_\_\_\_\_\_«Написание программы сканера»\_\_\_\_\_\_\_\_\_\_\_\_

(название лабораторной (практической) работы)

по дисциплине Методы анализа в теории формальных языков

(наименование дисциплины)

ЛР 09.03.04.05.000 ОТ

(обозначение документа)

Студент группы ПИ-42 Г.Гю Глушков

(инициалы, фамилия)

Преподаватель профессор, доцент Е.Н.Крючкова

(должность, ученое звание) (инициалы, фамилия)

Барнаул 2017

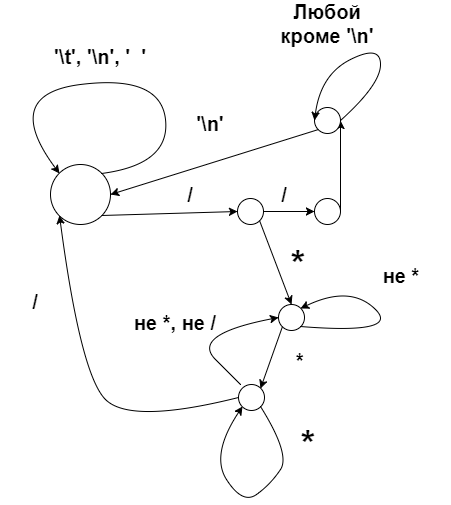
Задание: Лексика. Написать программу сканера, используя предварительно построенные

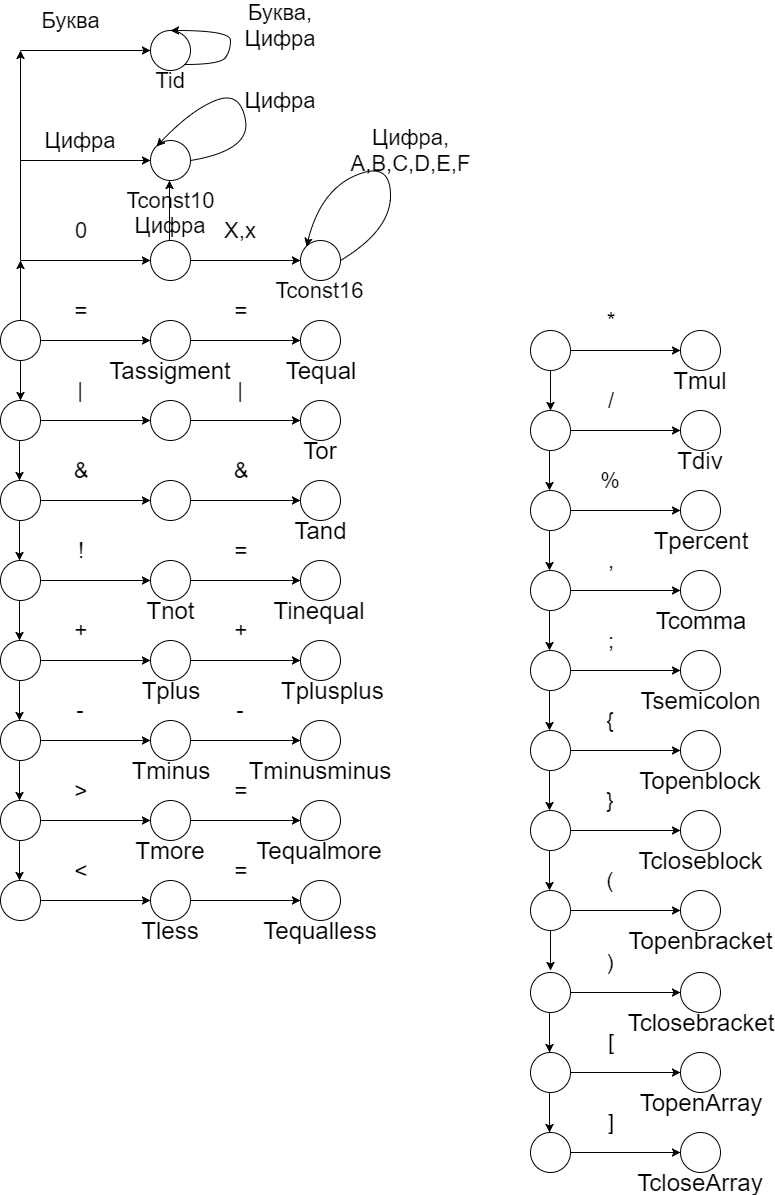
таблицу лексем и автоматы лексики.

Лексемы:

|  |  |  |  |
| --- | --- | --- | --- |
| Лексема | Тип | Id | Символ ограничения |
| Идентификатор | Tid | 1 | Не цифра, не буква, не \_ |
| Константы: |  |  |  |
| целое 10 c/c | Tconst10 | 21 | Не цифра |
| целое 16 c/c | Tconst16 | 22 | Любой символ кроме цифр и букв A, B, C, D, E, F |
| Ключевые слова: |  |  |  |
| void | Tvoid | 31 | Не цифра, не буква |
| int64 | Tint64 | 32 | Не цифра, не буква |
| int | Tint | 33 | Не цифра, не буква |
| for | Tfor | 34 | Не цифра, не буква |
| return | Treturn | 35 | Не цифра, не буква |
| main | Tmain | 36 | Не цифра, не буква |
| Операции: |  |  |  |
| = | Tassignment | 41 | не = |
| || | Tor | 42 | Любой символ |
| && | Tand | 43 | Любой символ |
| == | Tequal | 44 | Любой символ |
| != | Tinequal | 45 | Любой символ |
| > | Tmore | 46 | не = |
| < | Tless | 47 | не = |
| >= | Tequalmore | 48 | Любой символ |
| <= | Tequalless | 49 | Любой символ |
| + | Tplus | 50 | не + |
| - | Tminus | 51 | не - |
| \* | Tmul | 52 | Любой символ |
| / | Tdiv | 53 | Любой символ |
| % | Tpercent | 54 | Любой символ |
| ++ | Tplusplus | 55 | Любой символ |
| -- | Tminusminus | 56 | Любой символ |
| ! | Tnot | 57 | не = |
| Разделители: |  |  |  |
| , | Tcomma | 61 | Любой символ |
| ; | Tsemicolon | 62 | Любой символ |
| Скобки: |  |  |  |
| { | Topenblock | 71 | Любой символ |
| } | Tcloseblock | 72 | Любой символ |
| ( | Topenbracket | 73 | Любой символ |
| ) | Tclosebracket | 74 | Любой символ |
| [ | TopenArray | 75 | Любой символ |
| ] | TcloseArray | 76 | Любой символ |
| Конец исходного модуля | Tend | 101 |  |
| Ошибочная лексема | Terror | 102 |  |

Конечный автомат лексики, конечный автомат игнорируемых символов:





Код:

h файл

#pragma once

#include "defs.h"

class TScaner

{

public:

TScaner(string);

~TScaner(void);

void loadFromFile(string);

int scaner(LEX);

void paintError(char\*,char\*);

int GetUK();

void SetUK(int \_uk);

private:

Module  text;

int uk;

int pos;

int line;

};

cpp файл:

#include "TScaner.h"

TScaner::TScaner(string path)

{

loadFromFile(path);

uk = 0;

line = 1;

pos = 1;

}

LEX keywords[MAXKEY] = { "void", "int", "longlongint",

"for","return", "main" };

int indexkeywords[MAXKEY] = { Tvoid, Tint,

Tlonglongint, Tfor,Treturn ,Tmain };

TScaner::~TScaner(void)

{

}

int TScaner::GetUK()

{

return uk;

}

void TScaner::SetUK(int \_uk)

{

uk = \_uk;

}

void TScaner::loadFromFile(string path)

{

ifstream inp(path);

if (!inp.is\_open()) paintError(FileIsNotOpen, "");

int i;

for (i = 0; i < MAXTEXT && !inp.eof(); i++)

{

text[i] = inp.get();

}

text[i - 1] = '\0';

inp.close();

}

void TScaner::paintError(char\* error, char\* l)

{

if (l[0] == '\0')

cout << "Ошибка: " << error << endl;

else

cout << "Ошибка: " << error << ". Неверный символ: " << l << endl;

}

int TScaner::scaner(LEX lex)

{

int len;

for (len = 0; len < MAXLEX; len++) lex[len] = 0;

len = 0;

start:

//Пропуск незначимых символов

while (text[uk] == ' ' || text[uk] == '\n' || text[uk] == '\t')

{

if (text[uk] == '\n')

{

line++;

pos = 1;

}

else

pos++;

uk++;

};

//Пропуск однострочного комментария

if (text[uk] == '/' && text[uk + 1] == '/')

{

uk += 2;

pos += 2;

while (text[uk] != '\n'&&text[uk] != '\0')

{

pos++;

uk++;

}

if (text[uk] == '\n')

{

pos = 1;

line++;

uk++;

}

goto start;

}

//Пропуск многострочного комментария

if (text[uk] == '/' && text[uk + 1] == '\*') {

uk += 2;

while (text[uk] != '\0') {

if (text[uk] == '\*' && text[uk + 1] == '/') {

uk += 2;

goto start;

}

uk++;

}

goto start;

}

//Конец модуля

if (text[uk] == 0)

return Tend;

//Идентификатор

if (text[uk] >= 'a' && text[uk] <= 'z' ||

text[uk] >= 'A' &&  text[uk] <= 'Z')

{

while (text[uk] >= 'a' && text[uk] <= 'z' ||

text[uk] >= 'A' &&  text[uk] <= 'Z' ||

text[uk] >= '0' && text[uk] <= '9' ||

text[uk] == '\_')

if (len < MAXLEX - 1)

{

lex[len++] = text[uk++];

pos++;

}

else

{

uk++;

pos++;

};

for (int i = 0; i < MAXKEY; i++)

{

if (strcmp(keywords[i], lex) == 0)

return indexkeywords[i];

}

return Tid;

}

//Константы

if (text[uk] >= '1' && text[uk] <= '9')

{

number: while (text[uk] >= '0' && text[uk] <= '9')

{

if (len < MAXLEX - 1)

{

lex[len++] = text[uk++];

pos++;

}

else

{

paintError(LenthOfConstant, lex);

while (text[uk] >= '1' && text[uk] <= '9') { uk++; pos++; }

return Terror;

}

}

return Tconst10;

}

if (text[uk] == '0')

{

lex[len++] = text[uk++];

pos++;

if (text[uk] >= '0' && text[uk] <= '9') goto number;

if (text[uk] == 'x' || text[uk] == 'X') {

lex[len++] = text[uk++];

pos++;

while (text[uk] >= '0' && text[uk] <= '9' ||

text[uk] >= 'A' && text[uk] <= 'F')

{

if (len < MAXLEX - 1)

{

lex[len++] = text[uk++];

pos++;

}

else

{

uk++;

pos++;

}

}

if (len >= 9) return Terror;

return Tconst16;

}

}

if (text[uk] == '=')

{

lex[len++] = text[uk++];

pos++;

if (text[uk] == '=')

{

lex[len++] = text[uk++];

pos++;

return Tassignment;

}

return Tassignment;

}

if (text[uk] == '!'&&text[uk + 1] == '=')

{

lex[len++] = text[uk++];

lex[len++] = text[uk++];

pos += 2;

return Tinequal;

}

if (text[uk] == '>')

{

lex[len++] = text[uk++];

pos++;

if (text[uk] == '=')

{

lex[len++] = text[uk++];

pos++;

return Tmore;

}

return Tmore;

}

if (text[uk] == '<')

{

lex[len++] = text[uk++];

pos++;

if (text[uk] == '=')

{

lex[len++] = text[uk++];

pos++;

return Tequalless;

}

return Tless;

}

if (text[uk] == '+')

{

lex[len++] = text[uk++];

pos++;

if (text[uk] == '+')

{

lex[len++] = text[uk++];

pos++;

return Tplusplus;

}

return Tplus;

}

if (text[uk] == '-')

{

lex[len++] = text[uk++];

pos++;

if (text[uk] == '-')

{

lex[len++] = text[uk++];

pos++;

return Tminusminus;

}

return Tminus;

}

if (text[uk] == '/')

{

lex[len++] = text[uk++];

pos++;

return Tdiv;

}

if (text[uk] == '\*')

{

lex[len++] = text[uk++];

pos++;

return Tmul;

}

if (text[uk] == '%')

{

lex[len++] = text[uk++];

pos++;

return Tpercent;

}

if (text[uk] == '|')

{

lex[len++] = text[uk++];

pos++;

if (text[uk] == '|')

{

lex[len++] = text[uk++];

pos++;

return Tor;

}

//return Terror;

}

if (text[uk] == '&')

{

lex[len++] = text[uk++];

pos++;

if (text[uk] == '&')

{

lex[len++] = text[uk++];

pos++;

return Tand;

}

//return Terror;

}

if (text[uk] == ',')

{

lex[len++] = text[uk++];

pos++;

return Tcomma;

}

if (text[uk] == ';')

{

lex[len++] = text[uk++];

pos++;

return Tsemicolon;

}

if (text[uk] == '{')

{

lex[len++] = text[uk++];

pos++;

return Topenblock;

}

if (text[uk] == '}')

{

lex[len++] = text[uk++];

pos++;

return Tcloseblock;

}

if (text[uk] == '(')

{

lex[len++] = text[uk++];

pos++;

return Topenbracket;

}

if (text[uk] == ')')

{

lex[len++] = text[uk++];

pos++;

return Tclosebracket;

}

if (text[uk] == '[')

{

lex[len++] = text[uk++];

pos++;

return TopenArray;

}

if (text[uk] == ']')

{

lex[len++] = text[uk++];

pos++;

return TcloseArray;

}

lex[len++] = text[uk++];

paintError(InvalidCharacter, lex);

pos++;

return Terror;

}