Министерство образования и науки российской федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ПГТУ»)

Кафедра ИиСП

ОТЧЁТ

По лексическому анализатору

Разработала: студентка ПС-32

Амерханова Елена Рафаиловна

Йошкар-Ола

2021 год

Лексический анализатор (сканер) читает поток символов, составляющих исходную программу, и группирует эти символы в значащие последовательности, называющиеся лексемами.

Лексема – это структурная единица языка, которая состоит из элементарных символов языка и не содержит в своём составе других структурных единиц языка. Лексемами языков программирования являются идентификаторы, константы, ключевые слова языка, знаки операций и т.п.

На вход подаются названия входного файла (код программы) и выходного (где будет список токенов).

Список типов токенов:

* Identifier - идентификатор
* Key word - ключевое слово
* Integer – целое число
* Float - число с плавающей точкой
* Char - символ
* String - строка
* Arithmetic operator – арифметический оператор
* Comparison operator – оператор сравнения
* Assignment - присвоение
* Separator - разделитель
* Bracket - скобка
* Logical operator – логический оператор
* Comment – комментарий
* Error – ошибка
* EOF –конец файла

Лексемы

1) ключевые слова:

* if
* else
* while
* for
* read
* write
* return
* main
* void
* int
* float
* char
* string
* bool
* true
* false

2) скобки:

* (
* )
* {
* }

3) разделители: ';', ',', ':'

4) комментарии

* //строчные
* /\*многострочные\*/

5) операторы сравнения: ==, !=, >, <, >=, <=

6) логические операторы: &&, ||, !

7) присваивание =

8) арифметические операторы: +, -, \*, /

9) идентификатор - буквы, “\_”, “.”,цифры. Не может начинаться с цифры

Пример работы:

|  |  |
| --- | --- |
| bool someFunc3000(string str) {  if (str == "true") {  write("true");  return true;  }  else {  write("false");  return false; //some comment  }  }  int main() {  int number = 0;  while (number < 100.5 && !(number + 5 > 100)) {  for (int i = 0; i <= 10; i=i+1) {  number = number + 1;  }  number = number \* 1.5E+10;  }  /\*another  comment  now  multiline\*/    string s = "false";  someFunc3000(s);    return 0;  } | KEY\_WORD bool (1:1)  IDENTIFIER someFunc3000 (1:6)  BRACKET ( (1:18)  KEY\_WORD string (1:19)  IDENTIFIER str (1:26)  BRACKET ) (1:29)  BRACKET { (1:31)  KEY\_WORD if (2:2)  BRACKET ( (2:5)  IDENTIFIER str (2:6)  COMPARISON\_OPERATOR == (2:11)  STRING "true" (2:13)  BRACKET ) (2:19)  BRACKET { (2:21)  KEY\_WORD write (3:3)  BRACKET ( (3:8)  STRING "true" (3:9)  BRACKET ) (3:15)  SEPARATOR ; (3:16)  KEY\_WORD return (4:3)  KEY\_WORD true (4:10)  SEPARATOR ; (4:14)  BRACKET } (5:2)  KEY\_WORD else (6:2)  BRACKET { (6:7)  KEY\_WORD write (7:3)  BRACKET ( (7:8)  STRING "false" (7:9)  BRACKET ) (7:16)  SEPARATOR ; (7:17)  KEY\_WORD return (8:3)  KEY\_WORD false (8:10)  SEPARATOR ; (8:15)  BRACKET } (9:2)  BRACKET } (10:1)  KEY\_WORD int (12:1)  KEY\_WORD main (12:5)  BRACKET ( (12:9)  BRACKET ) (12:10)  BRACKET { (12:12)  KEY\_WORD int (13:2)  IDENTIFIER number (13:6)  ASSIGNMENT = (13:13)  INTEGER 0 (13:15)  SEPARATOR ; (13:16)  KEY\_WORD while (14:2)  BRACKET ( (14:8)  IDENTIFIER number (14:9)  COMPARISON\_OPERATOR < (14:16)  FLOAT 100.5 (14:18)  LOGIC\_OPERATOR && (14:24)  LOGIC\_OPERATOR ! (14:27)  BRACKET ( (14:28)  IDENTIFIER number (14:29)  ARITHMETIC\_OPERATOR + (14:36)  INTEGER 5 (14:38)  COMPARISON\_OPERATOR > (14:40)  INTEGER 100 (14:42)  BRACKET ) (14:45)  BRACKET ) (14:46)  BRACKET { (14:48)  KEY\_WORD for (15:3)  BRACKET ( (15:7)  KEY\_WORD int (15:8)  IDENTIFIER i (15:12)  ASSIGNMENT = (15:14)  INTEGER 0 (15:16)  SEPARATOR ; (15:17)  IDENTIFIER i (15:19)  COMPARISON\_OPERATOR < (15:21)  ASSIGNMENT = (15:22)  INTEGER 10 (15:24)  SEPARATOR ; (15:26)  IDENTIFIER i (15:28)  ASSIGNMENT = (15:29)  IDENTIFIER i (15:30)  ARITHMETIC\_OPERATOR + (15:31)  INTEGER 1 (15:32)  BRACKET ) (15:33)  BRACKET { (15:35)  IDENTIFIER number (16:4)  ASSIGNMENT = (16:11)  IDENTIFIER number (16:13)  ARITHMETIC\_OPERATOR + (16:20)  INTEGER 1 (16:22)  SEPARATOR ; (16:23)  BRACKET } (17:3)  IDENTIFIER number (18:3)  ASSIGNMENT = (18:10)  IDENTIFIER number (18:12)  ARITHMETIC\_OPERATOR \* (18:19)  FLOAT 1.5E+10 (18:21)  SEPARATOR ; (18:28)  BRACKET } (19:2)  KEY\_WORD string (26:2)  IDENTIFIER s (26:9)  ASSIGNMENT = (26:11)  STRING "false" (26:13)  SEPARATOR ; (26:20)  IDENTIFIER someFunc3000 (27:2)  BRACKET ( (27:14)  IDENTIFIER s (27:15)  BRACKET ) (27:16)  SEPARATOR ; (27:17)  KEY\_WORD return (29:2)  INTEGER 0 (29:9)  SEPARATOR ; (29:10)  BRACKET } (30:1)  END\_OF\_FILE EoF (30:1) |