Лексически анализ

1. Лексика на входния език

Най-малката структурна единица в езика се нарича лексема. Тя се състои от двойка елементи:

- тип на лексемата (напр. ключова дума, оператор, разделител, константа)
- стойност на лексемата, представляваща символ или символен низ (напр. if, >, {, 100)

В езика са дефинирани следните типове лексеми:

- идентификатори (имена на променливи и функции)

- ключови думи (напр. while, if, else, print, read, return)

- целочислени, булеви, символни и низови константи (напр. 10, true, '1', "5")

- едно и двусимволни оператори и разделители (напр. =, >, <, ==, >=, <=, {})

Лексемите на езика се формират от азбуката му по определни правила.

Азбуката на езика включва следните символи:

- малки и главни букви ('a'..'z', 'A'..'Z')

- цифри ('0'..'9')

- специални символи (например '+', '-', '%', ';')

Лексиката на езика се описва чрез граматика. Граматиката се състои от правила, всяко от които има следният вид:

лява страна : дясна страна

Входният език на учебния компилатор има следната лексическа граматика:

Ключови думи

INT : 'int'

CHAR : 'char'

VOID : 'void'

BOOLEAN : 'boolean'

WHILE : 'while'

RETURN : 'return'

IF : 'if'

ELSE : 'else'

TRUE : 'true'

FALSE : 'false'

LENGTH : 'length'

PROGRAM : 'program'

Оператори

PLUS : '+'

MINUS : '-'

MUL : '*'

DIV : '/'

MOD : '%'

BECOMES : '='

Релационни оператори

NOT : '!'

EQUALS : '=='

NOTEQUALS : '!='

GREATER : '>'

LESS : '<'

GREATER_EQ : '>='

LESS_EQ : '<='

Логически оператори

AND : '&&'

OR : '||'

Специални символи

LSQUARE	:	'['
RSQUARE	:	']'
LBRACKET	:	'{'
RBRACKET	:	'}'
LPAREN	:	'('
RPAREN	:	')'
SEMICOLON	:	';'
COMMA	:	','
SINGLE_QUOTE	:	'''
DOUBLE_QUOTES	:	'"'
ARROW	:	'->'
AT	:	'@'

Целочислени, булеви и символни константи

INT LITERAL	:	'0' | [1-9]{1}[0-9]*
CHAR_LITERAL	:	any ascii character
BOOLEAN_LITERAL	:	'true' | 'false'

Идентификатор

IDENTIFIER	:	[A-Za-z]{1}[A-Za-z0-9]*

2. Работа на лексическия анализатор

Лексическият анализатор разпознава следните типове лексеми:

- идентификатори (имена на променливи и функции)
- ключови думи (напр. while, if, else, print, read, return)
- целочислени, булеви, символни и низови константи (напр. 10, true, '1', "5")
- едно и двусимволни оператори и разделители (напр. =, >, <, ==, >=, <=, {)

Всяка лексема има тип, символно име, номер на ред и позиция.

Алгоритъм на лексическия анализатор:

```
 while (currentChar != Source.EOF)
 {
     switch  (currentChar)  {
         case ‘ ’: case ‘\t’:
                  /* Премахване на незначещ символ интервал или табулация */
         break;
         case  ‘a’: case ‘b’: … case ‘z’ :
                 /* Разпознаване на идентификатор или ключова дума */
                  break;
         case ‘0’: case ‘1’ : /* . . . */ case ‘9’:
                  /* Разпознаване на целочислена константа */
                   break;
         case ‘\’’:
                  /* Разпознаване на символна константа */
                   break;
         case ‘“’:
                  /* Разпознаване на низова константа (символен низ) */
                   break;
          case ‘-’: case ‘=’: case ‘>’: case ‘<’: case ‘!’: case ‘&’: case ‘|’: case ‘/’:
                 /* Разпознаване на двусимволни оператори */
                   break;
          case ‘+’: case ‘[’: case ‘]’: case ‘{’: case ‘}’: case ‘(’: case ‘)’: case ‘;’: case
‘*’’: case ‘%’: case ‘,’: case ‘@’:
                  /* Разпознаване на едносимволни оператори и разделители */
                  break;
};
```

Лексемите са описани от интерфейса Token.

```
public interface Token<Type> {
    public Type getTokenType();
    public String getText();
    public int getPosition();
    public int getLine();
}
```

Работата на лексическия анализатор се състои в последователно прочитане на всички символи от входния поток и причисляване на поредния прочетен символ към текущо формирана лексема. Тази задача се изпълнява от метода nextToken().

Задачи:

1. Да се довършат фрагментите за разпознаване на лексемите. Те са маркирани с коментари /* ToDo */ във файловете LexerImpl.java и TokenType.java.

2. В директория resources на проекта е дадена примерна входна програма (Fib.txt). Да се провери в изхода на лексическия анализатор дали всички лексеми от програмата са правилно разпознати.

3. Да се променя входната програма и да се проверява работата на лексическия анализатор с различни входни програми.