1st Milestone

Compiler project

(System Programming)

Hassan El-seoudy [ID : 3780]
Ahmed Tarek Kilani [ID : 3750]
Merna Adel Rgab [ID : 4168]

1. Regular Expression

Regular expressions have the capability to express finite languages by defining a pattern for finite strings of symbols. The grammar defined by regular expressions is known as regular grammar. The language defined by regular grammar is known as regular language.

Regular expression is an important notation for specifying patterns. Each pattern matches a set of strings, so regular expressions serve as names for a set of strings. Programming language tokens can be described by regular languages. The specification of regular expressions is an example of a recursive definition. Regular languages are easy to understand and have efficient implementation.

Keywords: “HashMap”

if, while, for, . . .

Identifiers:

“[a-zA-Z][a-zA-Z 0-9]*”

Integers:

“[+-]?[0-9]+”

Floats:

“[+-]?(([0-9]+ (.[0-9]*)?|.[0-9]+)([eE][+-]?[0-9]+)?)”

String constants:

“([a-zA-Z0-9]|\[a-zA-Z])*”

Read

("\\s*READ\\(([a-zA-Z_][a-zA-Z0-9_]*(?:\\,[a-zA-Z_][a-zA-Z0-9_]*)*)\\)\\s*")

Write

("\\s*WRITE\\(([a-zA-Z_][a-zA-Z0-9_]*(?:\\,[a-zA-Z_][a-zA-Z0-9_]*)*)\\)\\s*")

Operation

("\\s*([a-zA-Z0-9_\\(\\)\\,\\;\\:\\s\\=\\+*]*)\\s*")

Correct program expression

“\\s*PROGRAM\\s*([a-zA-Z_][a-zA-Z0-9_]*)\\s*VAR\\s*([a-zA-Z_][a-zA-Z0-9_]*(?:\\,[a-zA-Z_][a-zA-Z0-9_]*)*)\\s*BEGIN\\s*([a-zA-Z0-9_\\(\\)\\,\\;\\:\\s\\=\\+*]*)\\s*END.”

2. Lexical Analysis (Principle)

A language is considered as a finite set of strings over some finite set of alphabets. Computer languages are considered as finite sets, and mathematically set operations can be performed on them. Finite languages can be described by means of regular expressions.

When the lexical analyzer read the source-code, it scans the code letter by letter; and when it encounters a whitespace, operator symbol, or special symbols, it decides that a word is completed.

--

The lexical grammar of a programming language is a set of formal rules that govern how valid lexemes in that programming language are constructed. For example, the rules can state that a string is any sequence of characters enclosed in double-quotes or that an identifier may not start with a digit. The rules in the lexical grammar are often expressed with a set of **regular definitions**.

--

- Goals of the lexical analysis
    01. Divide the character stream into meaningful sequences called lexemes.
    02. Label each lexeme with a token that is passed to the parser (syntax analysis).
    03. Update the symbol tables with all identifiers (and numbers).
    04. Remove non-significant blanks and comments.
    05. Provide the interface between the source program and the parser.

3. Lexical Analysis (Output -> example)

```
1       PROGRAM 1
1       BASICS  17
2       VAR     2
3       X       17
3       Y       17
3       A       17
3       B       17
3       C       17
3       Z       17
4       BEGIN   3
5       READ    7
5       (       15
5       X       17
5       Y       17
5       Z       17
5       B       17
5       )       16
6       A       17
6       =       12
6       X       17
6       +       13
6       B       17
6       ;       11
7       C       17
7       =       12
7       X       17
7       +       13
7       Z       17
7       ;       11
8       C       17
8       =       12
8       C       17
8       *       18
8       B       17
8       ;       11
9       Z       17
9       =       12
9       A       17
```

```
9       +       13
9       B       17
9       +       13
9       C       17
9       +       13
9       Y       17
9       ;       11
10      WRITE   8
10      (       15
10      A       17
10      C       17
10      Z       17
10      )       16
11      END.    5
```