C**ourse:** Principles of Programming Languages **Course ID:** IT092IU

**INTERNATIONAL UNIVERSITY - VIETNAM NATIONAL UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

# **PPL Lab 1**

# **LexerExercise**

| **NAME** | **STUDENT ID** |
| --- | --- |
| Vũ Nhật Duy | ITITIU17047 |

**Write lexer rules in “BKIT.g4” that can accept the following tokens.**

**Example:**

Int:

All integer numbers

Integer numbers can start with 0

Id:

Containing only lower-case alphabets

**Exercise 1:**

Int:

All integer numbers

Integer numbers do not start with 0

Float:

Example: 1.0, 1., 12e2, 9e-2

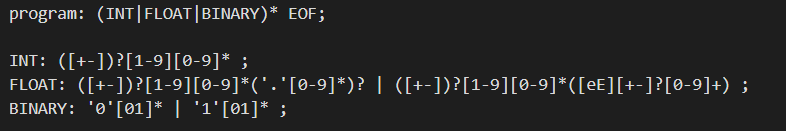
Float numbers do not start with 0

Binary:

Example: 001, 101, 0010, 1001

**Solution:**

With the provided Lexer Code files, changing the program part to meet the requirements in Exercise1.g4.



INT: ([+-]?)[1-9][0-9]\*  
 ([+-]?): All integer numbers including positive and negative integer numbers

[1-9][0-9]\*: Integer numbers do not start with 0

FLOAT: ([+-])?[1-9][0-9]\*('.'[0-9]\*)? | ([+-])?[1-9][0-9]\*([eE][+-]?[0-9]+)

([+-]?): All float numbers including positive and negative float numbers

([+-]?)[1-9][0-9]\*('.'[0-9]\*)?: All float numbers do not start with 0

([+-])?[1-9][0-9]\*([eE][+-]?[0-9]+): All float numbers that contain e or E

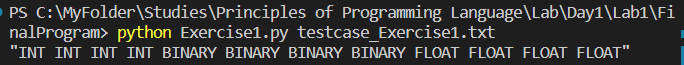
BINARY: '0'[01]\* | '1'[01]\*

'0'[01]\* | '1'[01]\*: Binary that only contains 0 or 1

Then after completing, running the code to compile and test with the testcase\_Exercise1.txt

+3 -45 28 2 0001 0101 0010 01001 1.0 1. 12E2 9e-2





**Exercise 2:**

Id:

Starting with a lower-case alphabet

Can contain alphabets, numbers and ‘\_’

Int, Float, Binary:

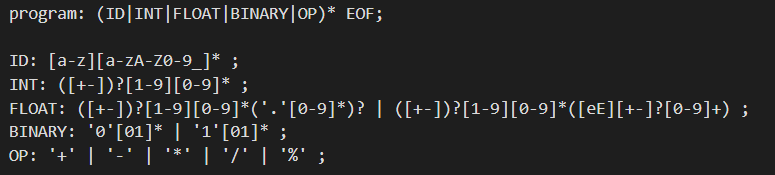
The same requirements as Exercise 1

Operator:

Example: +, -, \*, /, %

**Solution:**

With the provided Lexer Code files, changing the program part to meet the requirements in Exercise2.g4.



ID: [a-z][a-zA-Z0-9\_]\*

[a-z]: IDs that start with a lower-case alphabet

[a-zA-Z0-9\_]\*: IDs that can contain alphabets, numbers and ‘\_’

INT: ([+-]?)[1-9][0-9]\*  
 ([+-]?): All integer numbers including positive and negative integer numbers

[1-9][0-9]\*: Integer numbers do not start with 0

FLOAT: ([+-])?[1-9][0-9]\*('.'[0-9]\*)? | ([+-])?[1-9][0-9]\*([eE][+-]?[0-9]+)

([+-]?): All float numbers including positive and negative float numbers

([+-]?)[1-9][0-9]\*('.'[0-9]\*)?: All float numbers do not start with 0

([+-])?[1-9][0-9]\*([eE][+-]?[0-9]+): All float numbers that contain e or E

BINARY: '0'[01]\* | '1'[01]\*

'0'[01]\* | '1'[01]\*: Binary that only contains 0 or 1

OP: '+' | '-' | '\*' | '/' | '%'

'+' | '-' | '\*' | '/' | '%': operators that contain +, -, \*, /, %

Then after completing, running the code to compile and test with the testcase\_Exercise2.txt

3 45 28 2 001 0101 0010 01001 1.0 1. 12e2 9e-2 myVariable x alpha\_123 vuNhatDuy + \* / - %



