**Summary:**

The types of tokens I will have in my language are:

-Integer literals

-Character literals

-Operators

-Identifiers/names

-Keywords

-Delimeters

**Integer Literals:**

This can be described by the following:

**IntLit** -> **T** | **H** | **O**

**T** -> [1-9][0-9]\*

**H** -> 0[xX][0-9a-fA-F]+

**O** -> 0[0-7]\*

where **T** is base ten, **H** is hexadecimal, and **O** is octal.

**Character Literals:**

A character literal can be described by the following (quotes are meant to distinguish an individual character:

**CharLit** -> “ ‘ “ **Ch** “ ‘ “

**Ch** -> **NonPrintASCII** | “\” **Escape**

**Escape** -> [a b f n r t v 0 \ ’ “ ?]

**Operators:**

These are the following possible operators (no bitwise operators yet, and organized by the class of operators we can have):

**Assignment:**

|  |  |
| --- | --- |
| **Operator Symbol** | **Attribute/Classification** |
| = | ASSIGN |

**Arithmetic:**

|  |  |
| --- | --- |
| **Operator Token** | **Attribute/Classification** |
| **+** | ADD |
| **-** | SUBT |
| **\*** | MULT |
| **/** | DIV |
| % | MOD |

**Boolean:**

|  |  |
| --- | --- |
| **Operator Token** | **Attribute/Classification** |
| < | LT |
| > | GT |
| <= | LEQ |
| >= | GEQ |
| == | EQ |
| || | LOR |
| && | LAND |

**Identifiers:**

The following is a simple RegExp describing identifiers:

**Ident** -> [a-zA-Z\_][0-9a-zA-Z\_]\*

**Keywords:**

Here are the keywords in our language:

|  |
| --- |
| **if** |
| **else** |
| **for** |
| **int** |
| **char** |
| **print** |

**Delimeters:**

Here are the delimeters in our language:

|  |
| --- |
| **(** |
| **)** |
| **{** |
| **}** |
| **;** |
| **,** |