DIPL Final Project

Introduction

Below is the specificatin of a simple programming language. It is a strongly typed language and intended as final requirement for students enrolled in DIPL. It aims to train them on how to build a pure interpreter.
Sample Program:

```
~ my first program
    VAR abc, b, c AS INT
    VAR x, w_23='w' AS CHAR
    VAR t="TRUE" AS BOOL
    START
        abc=b=10
        w_23='a'
        ~ this is a comment
        OUTPUT: abc & "hi" & b & "#" & w_23 & "[#]"
    STOP
```

Output of the sample program:

```
10hi10
a#
```

Language Grammar

Program Structure:

- every line contains a single statement
- all variable declaration is found on top of the program
- a line that starts with tilde(~) is considered as a comment and comment can be found in any part of the program
- executable code should be found inside the START and STOP block
- all reserved words are in capital letters
- sharp sign(#) signifies next line or carriage return
- ampersand(&) serves as a concatenator
- the square braces([]) are as escape code

Data Types:

1. INT – an ordinary number with no decimal part. It uses 32 bits. It can be positive or negative.
2. CHAR – a single symbol. It uses UNICODE.
3. BOOL – represents the literals true or false.
4. FLOAT – a number with decimal part. It uses 64 bits.

Operators:

Arithmetic operators

( )	- parenthesis
*, /, %	- multiplication, division, modulo
+, -	- addition, subtraction
>, <	- greater than, lesser than
>=, <=	- greater than or equal to, lesser than or equal to
==, <>	- equal, not equal

Logical operators (<BOOL expression><LogicalOperator><BOOL expression>)

AND	- needs the two BOOL expression to be true to result to true, else false
OR	- if one of the BOOL expressions evaluates to true, returns true, else false
NOT	- the reverse value of the BOOL value

Unary operator

+	- positive
-	- negative

Control structures:

Conditional Statements

- IF (<BOOL expression>)
  START
  <statement>
  …
  <statement>
  STOP
- IF (<BOOL expression>)
  START
  <statement>
  …
  <statement>
  STOP
  ELSE
  START
  <statement>
  …
  <statement>
  STOP

Repetition Statement

- WHILE (<BOOL expression>)
  START
  <statement>
  …
  <statement>
  STOP

Command-line statements:

OUTPUT - allows the user to output values to the console.

Syntax:

OUTPUT: (<immediateValue>|< variableName)>[& (<immediateValue>|< variableName)]*

Sample use:

OUTPUT: abc & “hi”

~ abc could be of any data type…

INPUT – allows the user to input a value to a data type.

Syntax:

INPUT: <variableName>[,<variableName>]*

Sample use:

INPUT: x, y

- means in the screen you have to input two values separated by comma(,)

Sample Programs

1. ~A program with arithmetic operation

```
VAR xyz, abc=100 AS INT
START
        xyz= ((abc *5)/10 + 10) * -1
        ~ xyz should have the value -60
        OUTPUT: "[[]" & xyz & "[]]"
STOP
```

Output of the sample program:

[-60]

2. ~A program with logical operation

```
VAR a=100, b=200, c=300 AS INT
VAR d="FALSE" AS BOOL
START
        d = (a < b AND c <>200)
        OUTPUT: d
STOP
```

Output of the sample program:

TRUE

3. ~A program with conditional statement

```
VAR a=0 AS INT
START
        OUTPUT: "Input a number:"
        INPUT: a
        IF (a<=0)
                START
                        OUTPUT: "#Negative or Zero..."
                STOP
        ELSE
                START
                        OUTPUT: "#Positive"
                STOP

STOP
```

Sample Trace:

```
        Input a number: -1
        Negative or Zero...
```

4. ~A program with looping statement

```
VAR a=0 AS INT
START
        OUTPUT: "Input a number:"
        INPUT: a
        WHILE (a<=10)
                START
                        OUTPUT: a & "#"
                        a = a + 1
                STOP
STOP
```

Sample Trace:

Input a number: 5
5
6
7
8
9
10