Language Specification of BISAYA++ Programming Language

Introduction

Bisaya++ is a strongly–typed high–level interpreted Cebuano-based programming language developed to teach Cebuanos the basics of programming. Its simple syntax and native keywords make programming easy to learn.

Sample Program:

```
-- this is a sample program in Bisaya++
SUGOD
        MUGNA NUMERO x, y, z=5
        MUGNA LETRA a_1='n'
        MUGNA TINUOD t="OO"
        x=y=4
        a_1='c'
        -- this is a comment
        IPAKITA: x & t & z & $ & a_1 & [&] & "last"
KATAPUSAN
```

Output of the sample program:

```
4OO5
c&last
```

Language Grammar

Program Structure:

- all codes are placed inside SUGOD and KATAPUSAN
- all variable declaration starts with MUGNA
- all variable names are case sensitive and starts with letter or an underscore (_) and followed by a letter, underscore or digits.
- every line contains a single statement
- comments starts with double minus sign(--) and it can be placed anywhere in the program
- all reserved words are in capital letters and cannot be used as variable names
- dollar sign($) signifies next line or carriage return
- ampersand(&) serves as a concatenator
- the square braces([]) are as escape code

Data Types:

1. NUMERO – an ordinary number with no decimal part. It occupies 4 bytes in the memory.
2. LETRA – a single symbol.
3. TINUOD – represents the literals true or false.
4. TIPIK – a number with decimal part.

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>)

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

Boolean values (enclosed with a double quote)

OO	- TRUE
DILI	- FALSE

Unary operator

+	- positive
-	- negative
++	- increment
--	- decrement

Sample Programs

1. A program with arithmetic operation

```
SUGOD
        MUGNA NUMERO xyz, abc=100
        xyz= ((abc *5)/10 + 10) * -1
        IPAKITA: [[] & xyz & []]
KATAPUSAN
```

Output of the sample program:

```
[-60]
```

2. A program with logical operation

```
SUGOD
        MUGNA NUMERO a=100, b=200, c=300
        MUGNA TINUOD d="DILI"
        d = (a < b UG c <>200)
        IPAKITA: d
KATAPUSAN
```

Output of the sample program:

```
OO
```

Code output statement:

IPAKITA - writes formatted output to the output device

Code input statement:

DAWAT – allow the user to input a value to a data type.

Syntax:

```
        DAWAT: <variableName>[,<variableName>]*
```

Sample use:

```
         DAWAT: x, y
```

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

CODE control flow structures:

1. Conditional

a. KUNG (if selection)

```
KUNG (<BOOL expression>)
PUNDOK{
        <statement>
                ...
        <statement>
}
```

b. KUNG-KUNG WALA (if-else selection)

```
KUNG (<BOOL expression>)
PUNDOK{
        <statement>
        ...
        <statement>
}
KUNG WALA
PUNDOK{
        <statement>
        ...
        <statement>
}
```

c. KUNG-KUNG DILI (if-else with multiple alternatives)

```
KUNG (<BOOL expression>)
PUNDOK{
        <statement>
        ...
        <statement>
}
KUNG DILI (<BOOL expression>)
PUNDOK{
        <statement>
        ...
        <statement>
}
```

```
KUNG WALA
PUNDOK{
          <statement>
          ...
          <statement>
}
```

PUNDOK{ } – group a block of codes. Statements inside conditions and loops are enclosed **PUNDOK{ }**.

2. **Loop Control Flow Structures**
   a. **ALANG SA (initialization, condition, update) - (FOR LOOP)**

```
PUNDOK{
          <statement>
          ...
          <statement>
}
```

Example:

```
ALANG SA (ctr=1, ctr<=10, ctr++)
PUNDOK{
          IPAKITA: ctr & ` `
}
```

Output:

```
1 2 3 4 5 6 7 8 9 10
```

Note: You may use any language to implement the interpreter except Python and Javascript.