Republic of the Philippines
POLYTECHNIC UNIVERSITY OF THE PHILIPPINES
College of Computer and Information Sciences
Department of Computer Science

BAL: Basic and Adaptable Language

In Partial Fulfillment of the Requirements for the

COSC 30053

Automata and Language Theory

Submitted by:

Balmadrid, Vince Earl D.

Campo, Jeslen C.

Gero, Michaella P.

Pagcanlungan, Evitha A.

Reyes, Jewel Anne A.

Velarde, Mia Jamille E.

BSCS 3-1N

February 2023

TABLE OF CONTENTS

I. Introduction

1. Description

Basic & Adaptable Language (BAL) is an Object-Oriented Programming Language inspired from Java Language. It aims to further improve exception handling, garbage collection, and memory allocation of the Java Language in pursuit for a new robust language.

The Java Language was known to offer limited string functions. With this, the Basic & Adaptable Language aims to use and expand these string-related functions to provide a large array of libraries.

BAL is best for Mobile application, Desktop GUI application, Web-based application, Enterprise application, Scientific application, Gaming application, Big data application, Business application, Distributed applications, and Cloud-based applications as it provides great and huge array of libraries, allows to perform many tasks at the same time, and it does not need any interpreter that will translate the source code into machine-independent bytecode.

2. Brief History

Basic & Adaptable Language or BAL was developed by a group of 3rd-year students from Polytechnic University of the Philippines under the program Bachelor of Science in Computer Science. The group created this project in partial fulfillment of their subject Principles of Programming Languages. The name BAL was inspired by the term “Balbal” which means “slang” in the English Language. Balbal is an informal language used in everyday conversations.

In creating this program, the group decided to use Balbal because in general, Filipinos are fond of inventing words and borrowing foreign terms, so applying this in programming can help novice Filipino programmers to enjoy coding. By using this kind of language, the group can also help beginners to not be overwhelmed by the complexity of the programming. They also decided to use Balbal because they believe that we can easily familiarize ourselves with programming concepts if we associate them with words that we usually encounter in our daily lives.

II. Syntactic Elements of Language

1. Character Set

a. char = {alphabeti, digitz, espesyal_char}
b. alphabeti = {daks, smol}
c. daks = {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}
d. smol = {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z}
e. digitz = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
f. espesyal_char = {~ , ! , # , $, % , ^ , & , * , (,) , _ , +, | , \ , ‘ , - , = , { , } , [,] , : , “ , ; , < , > , ? , (,) , (.) , / , <space>}

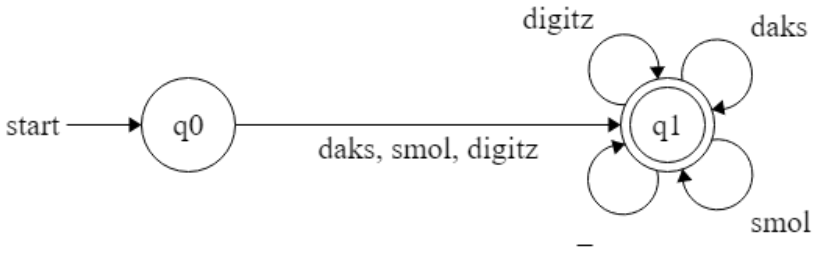
2. Identifier

Rules in Naming Identifiers:

1. A maximum of 26 characters are allowed.
2. Underscore (_) is only allowed as a special character.
3. It must not be a keyword or a reserved word.
4. It must be started with a letter followed by a digit, underscore, or another letter.
5. BAL is case-sensitive. Uppercase and lowercase are not considered as one.
6. There should not be any space.
7. Apostrophe symbol (‘) is not allowed.

Acceptable Values:	Not Acceptable Values:
FullName	_FullName
average_grade	Item’sID
Address123	1st Visit
fullname	height-weight
Telephone_123	143ily

Finite State Machine of Identifier



3. Operational Symbols

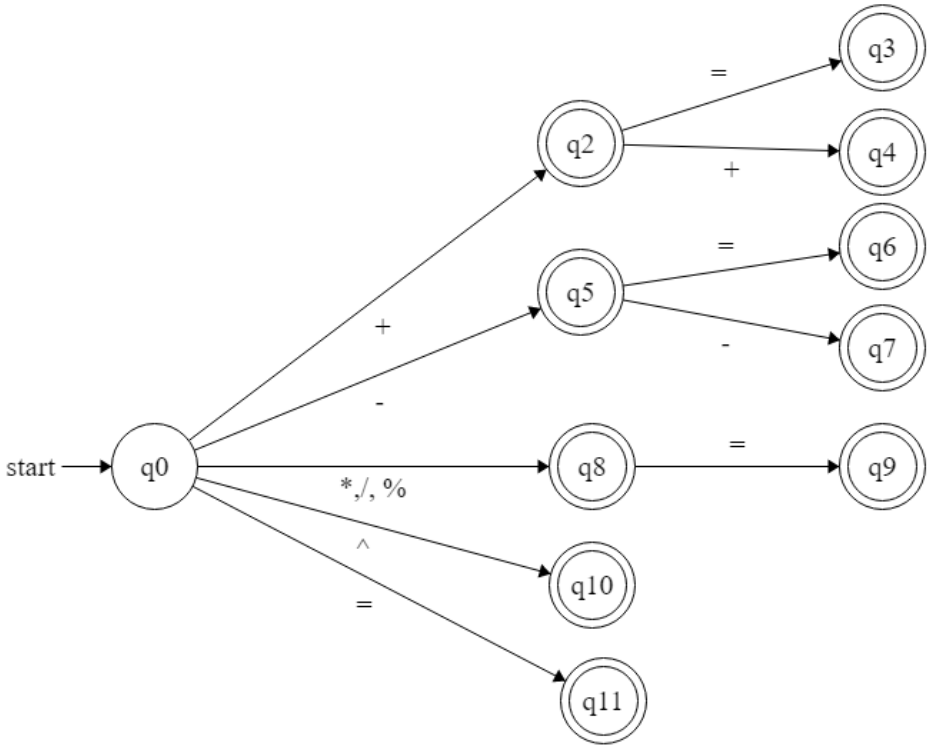
a. Arithmetic Operators

Operator	Description
+	Adds two operands
-	Subtracts second operand from the first
*	Multiples both operands
/	Divides numerator and denominator
%	Modulus operator finds the remainder after division of one number by another
^	Exponent
++	Increment operator increases integer value by one
--	Decrement operator decreases integer value by one

b. Assignment Operators

Operator	Description
=	Used to assign the value of the expression on the right-hand side to the variable on the left-hand side.
+=	Adds the value on the right-hand side to the variable on the left-hand side, and then assigns the result to the variable on the left-hand side.
-=	Subtracts the value on the right-hand side from the variable on the left-hand side, and then assigns the result to the variable on the left-hand side.
*=	Multiplies the value on the right-hand side with the variable on the left-hand side, and then assigns the result to the variable on the left-hand side.
/=	Divides the variable on the left-hand side by the value on the right-hand side, and then assigns the result to the variable on the left-hand side.
%=	Divides the variable on the left-hand side by the value on the right-hand side and returns the remainder, and then assigns the remainder to the variable on the left-hand side.

Finite State Machine of Arithmetic and Assignment Operator



c. Relational Operators

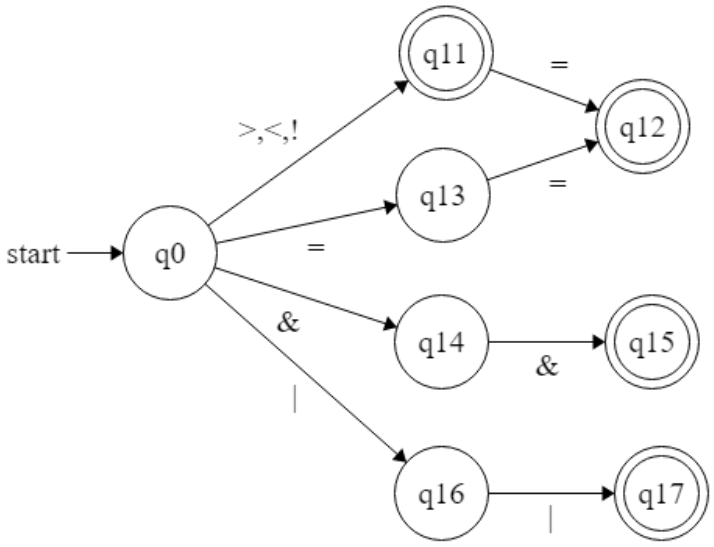
Operator	Description
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.
!=	Checks if the values of two operands are equal

	or not, if values are not equal then the condition becomes true.

d. Logical Operators

Operator	Description
!	It is called Logical NOT Operator. Use to reverse the logical state of its operand. If a condition is true, then the Logical NOT operator will make false.
||	It is called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.
&&	It is called a Logical AND operator. If both the operands are non-zero, then the condition becomes true.

Finite State Machine of Relational and Logical Operator



4. Keywords and Reserved Words

KEYWORDS	
Keywords	**Definition**
char	It is used to store character constants in memory. BAL provides a character data type called *char* whose type consumes two bytes but can only hold a single character.
booly	BAL provides a boolean data type called *booly* which is used to test a particular condition during program execution. Its variable can take either true or false and usually consumes one byte of storage.
dobol	BAL provides a double data type called *dobol* which is used to declare a variable that can hold 64-bit floating-point number.
giginteger	BAL provides an integer data type called *giginteger* which can hold whole numbers without any fractional point such as 143, -352, 913, etc.
istring	BAL provides a string data type called *istring* which is used to represent a sequence of char values.
	BAL provides a float data type called *mcfloat* which is used to declare a

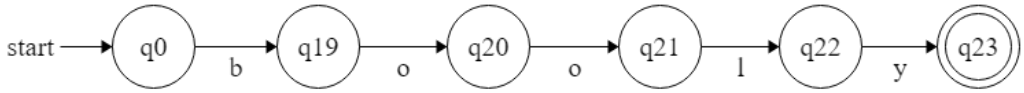
mcfloat	variable that can hold a 32-bit floating-point number.

RESERVED WORDS	
Keywords	**Definition**
getchi	It is used to get user data.
bounce	It is used to display data.
paano_kung	BAL provides an if keyword called *paano_kung* to test the condition. It executes the *paano_kung block* if the condition is true.
elsa	BAL provides an else keyword called *elsa* which is used to indicate the alternative branches in a *paano_kung* statement.
elsa_paano_kung	BAL provides an else if keyword called *elsa_paano_kung* used to test the condition. It executes the *pano_kung block* if condition is true otherwise *elsa block* is executed.
waylalu	BAL provides a while keyword called *waylalu* which is used to start a *waylalu* loop. This loop iterates a part

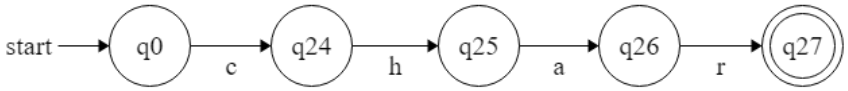
	of the program several times. If the number of iteration is not fixed, it is recommended to use the *waylalu* looping.
nugagawen	BAL provides a do keyword called *nugagawen* which is used in the control statement to declare a loop. It can iterate a part of the program several times.

Finite State Machine of Keywords

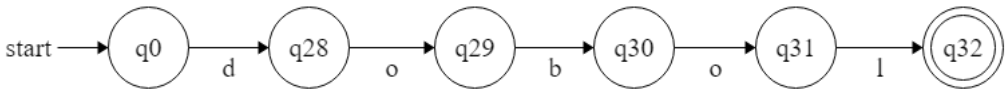
- booly



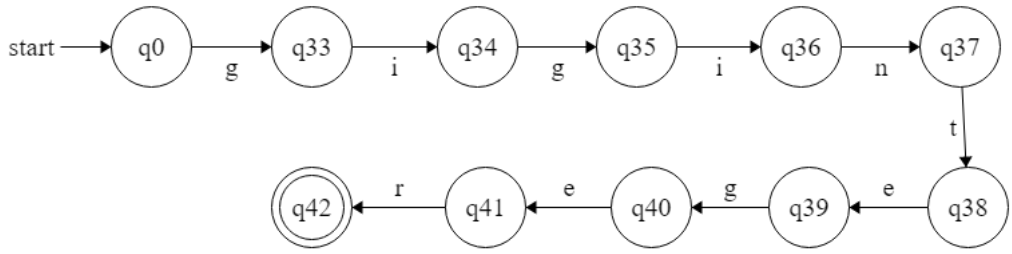
- char



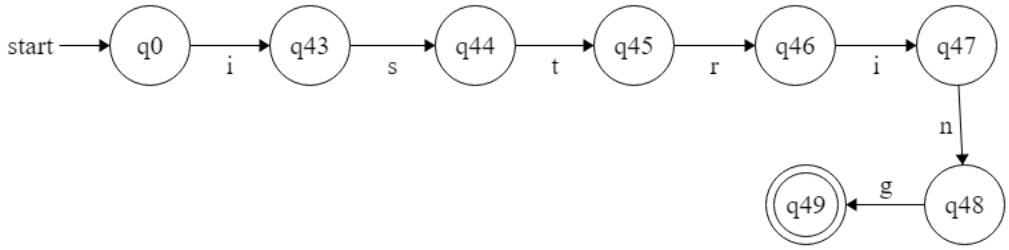
- dobol



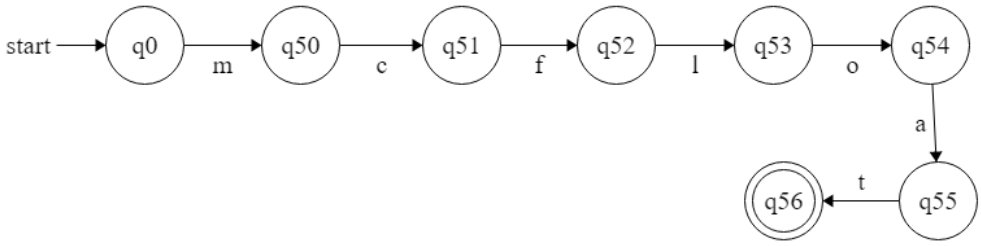
- giginteger



- istring

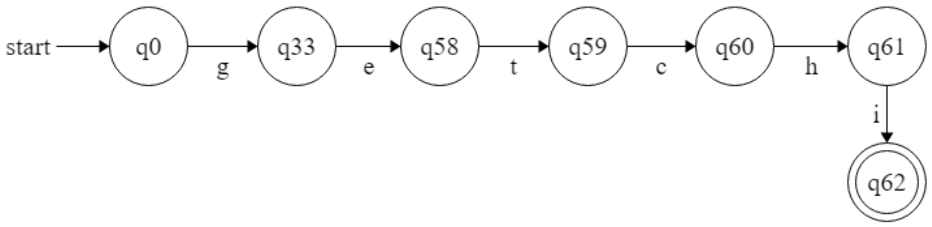


- mcfloat

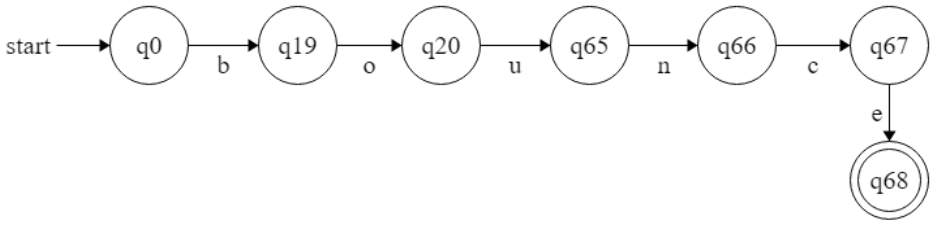


Finite State Machine of Reserved Words

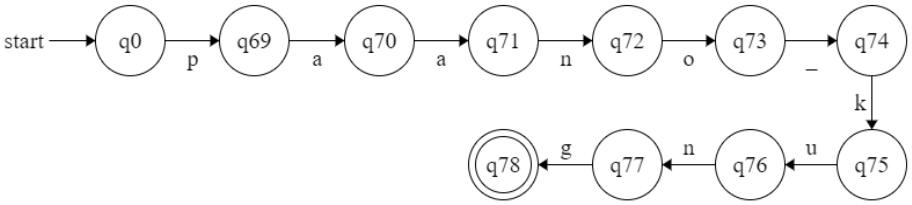
- getchi



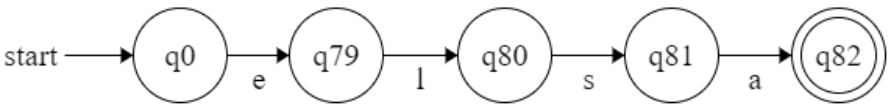
- bounce



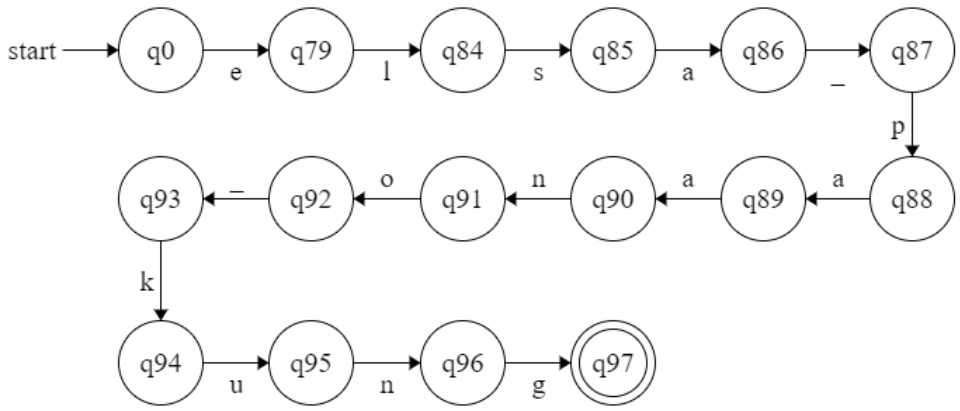
- paano_kung



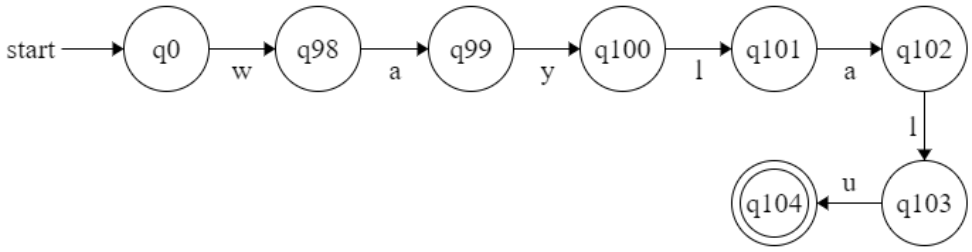
- elsa



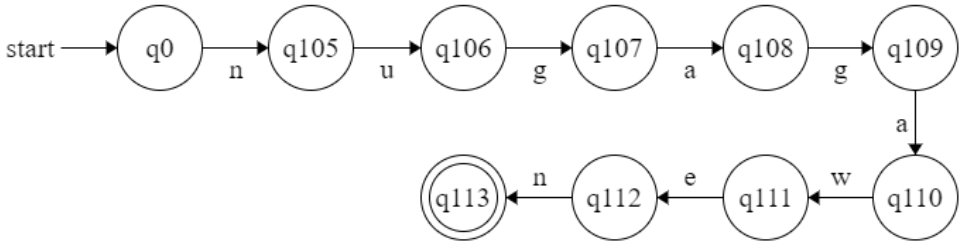
- elsa_paano_kung



- waylalu



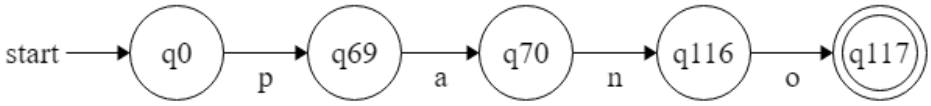
- nugagawen



5. Noise Words

Noise Word	Description
pano	It is a noise word that is used in conjunction with the reserved word "elsa" to create the conditional statement "elsa_pano". The purpose of "pano" in this context is to establish the conditions for the conditional statement, which would be executed if the conditions are met.

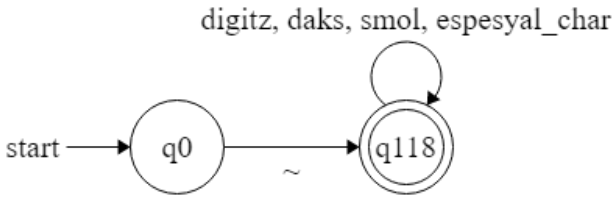
Finite State Machine of Noise Word



6. Comments

Symbol	Definition
~	Single line comment

Finite State Machine of Comments



7. Blanks/Spaces

Basic & Adaptable Language (BAL) allows single/one space for separating the character, word, expressions, and statements.

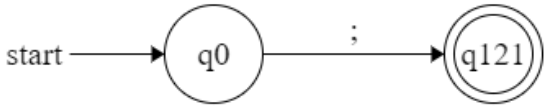
Blanks	Description
\n	New line
\t	New tab
‘ ‘	Single space

8. Delimiters and Brackets

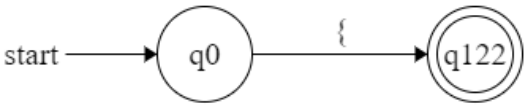
Symbol		Definition
Semicolon	;	Used to terminate and is required after every variable or statement.
Opening Curly Bracket	{	Used to enclose groups of statements.
Closing Curly Bracket	}	
Opening Square Bracket	[	Used to index elements in arrays and also Strings.
Closing Square Bracket	]	
Whitespace	“ “	Used to separate texts for readability.
Comma	,	Used to create compound expressions in which multiple expressions are evaluated.
Pipes	|	A bitwise OR operator.
Ampersand	&	A bitwise AND operator.
Forward Slash	/	Is a normal character.
Backward Slash	\	Used as an escape sequence character.
Single Quotation	‘	Used to enclose a character.
Double Quotation	“	Used to enclose a string.

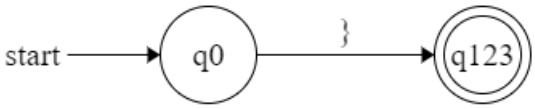
Finite State Machine of Delimiters and Brackets

- **Semicolon**

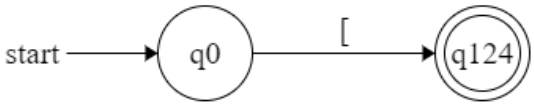


- **Opening Curly Bracket and Closing Curly Bracket**

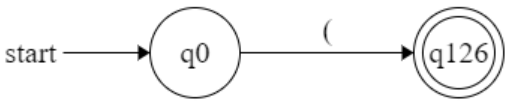


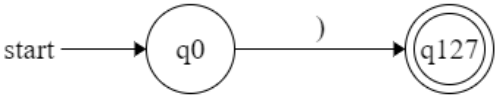


- **Opening Square Bracket and Closing Square Bracket**

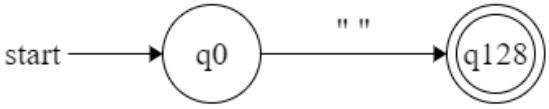


- **Opening Parenthesis and Closing Parenthesis**

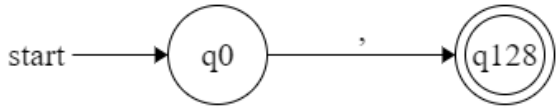




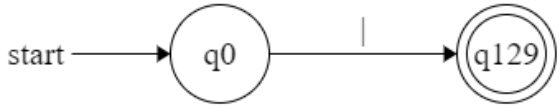
- **Whitespace**



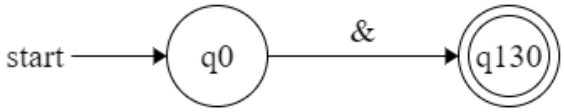
• **Comma**



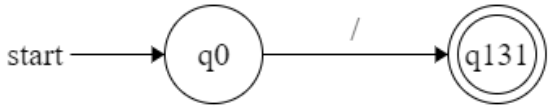
• **Pipes**



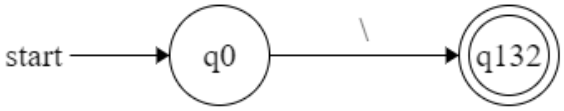
• **Ampersand**



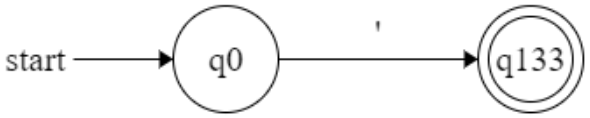
• **Forward Slash**



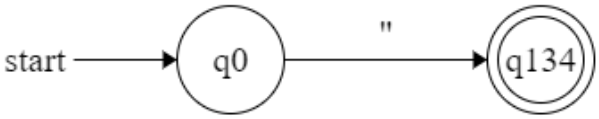
• **Backward Slash**



• **Single Quotation**



• Double Quotation



9. Free-and-fixed Formats

Since the inspiration and basis of this language, Java, is a free field format language. BAL is also a free field format programming language. It doesn't follow a strict format in terms of coding the language.

10. Expressions

a. Arithmetic Expressions

The syntax for an arithmetic expression is "variable = variable (operator) variable. In accordance with mathematical conventions, arithmetic operations are performed in a specific order of precedence. Exponentiation, multiplication, and division are evaluated first, followed by addition and subtraction.

Symbol	Definition
X = A + B	Adds B to the current value of A and returns the sum.
X = A – B	Substracts B to the current value of A and returns the difference.
X = A * B	Multiplies B to the current value of A and returns the product.
X = A / B	Divides B to the current value of A and returns the quotient.
X = A % B	Finds the remainder when A is divided by B.

X = A ^ B	Indicates the number of times A needs to be multiplied by itself.

b. Relational Expressions

The syntax for a relational expression is “variable (operator) variable”. Relational operators follow a hierarchy of precedence, with the operators on the left side of the expression taking the highest priority. The order of operations is: "equal to" (==), "not equal to" (!=), "less than" (<), "less than or equal to" (<=), "greater than" (>), and "greater than or equal to" (>=).

Symbol	Definition
X > Y	Checks if the current value of X is greater than the current value of Y then returns true, else returns false.
X >= Y	Checks if the current value of X is greater than or equal to the current value of Y then returns true, else returns false.
X < Y	Checks if the current value of X is less than the current value of Y then returns true, else returns false.
X <= Y	Checks if the current value X is less than or equal to the current value of Y then returns true, else returns false.
X == Y	Checks if the current value of X and Y are equal then returns true, else returns false.
X != Y	Checks if the value of X and Y are not equal then returns true, else returns false.

c. Logical Expressions

The syntax of the logical expression are: variable (operator) variable (for AND and OR) and (operator) variable (for NOT). The hierarchy of the logical operations follows AND (&&) as the highest and OR (||) as the lowest.

Symbol	Definition
!X	Returns the opposite value of the expression.
X > 0 || X < 1	Returns a value true if one or more statements in a condition are true, else returns false.
X > 0 && X < 1	Returns a value true if both statements in a condition are true, else returns false.

11. Statements

a. Declaration Statements

giginteger a = 10;

giginteger b, c, sum;

mcfloat grades;

dobol score = 1.25;

char MiddleInitial = ‘ A ’;

istring greetings = “Hello World!”;

booly answer1 = true, answer2 = false;

b. Output Statement

giginteger a = 0;

bounce("hello" + a);

c. Input Statement

```
giginteger b = 0;
getchi(b);
getchi(x);
```

d. Assignment Statement

```
a = sold + profit;
b = 100.91;
word = “Basic Adaptable Language”;
sym = ‘&’;
c* = b;
d = a - c;
```

e. Conditional Statement

```
paano_kung (a == 1) {
        bounce(“hello!”); }
elsa_paano_kung (y == 16 || x == 20){
        bounce(“My age is:” + y); }
elsa pano {
        bounce(“Welcome!”); }
elsa {
        bounce(“You’re Welcome!”); }
```

f. Iterative Statement

```
nugagawen{
        a++;
```

```
        b–;
}waylalu(a != 3);
nugagawen{
        z++;
}waylalu(z <= 3);
```