

NatLang Language Design

Here's a **language design summary** includes description covers **syntax, structure, supported features, and operator rules.**

NatLang is a beginner-friendly, English-inspired programming language designed to make logical thinking and programming approachable through natural-language-like syntax. It supports variables, expressions, control flow, and output using intuitive keywords.

Program Structure

A complete NatLang program begins with:Hi!

And ends with: Bye!

In between, the program consists of **statements**, each separated by a "." and optionally followed by a newline (\n).

Supported Statement Types

1. Variable Declarations

LetsSay x is 10.

LetsSay msg is "Hello!".

LetsSay a is true.

2. Aliases

LetsSay alias isAlso original.

This creates a reference to the value of another variable.

3. Assignment

x is x plus 1.

msg is "Updated!".

4. Output

Show x.

Show "Done!".

5. If-Else Conditionals

When x IsGreaterThan 5

Then

Show "Big".

Otherwise

Show "Small".

ThenStop

6. Ternary Conditional (Single-line)

LetsSay msg is When x IsEqualTo 10 Then "Match" Otherwise "No Match" ThenStop.

7. For Loops

ForAll item in numbers:

Show item.

StopNow

8. Until Loops

Until x IsEqualTo 10:

x is x plus 1.

NowStop

Expressions

Expressions support:

- Constants: number, string, boolean
- Variables
- Arithmetic and logic operations
- Parentheses for grouping

Examples:

x plus y.

(2 plus 3) times 4.

a AsWellAs b.

Arithmetic Operators (with Precedence)

Operator	Symbol	Precedence
Parentheses	(...)	Highest
Multiplicative	times, dividedBy	High
Additive	plus, minus	Medium
Logical	AsWellAs, EitherOr	Low

Comparison Operators

Used in When or Until conditions:

- IsEqualTo
- IsNotEqualTo
- IsGreaterThan
- IsLessThan
- IsAtLeast
- IsAtMost
- IsNot

Values

- **Number:** 5, 3.14
- **String:** "Hello!" (must be in double quotes)
- **Boolean:** true, false
- **List (optional):** [1, 2, 3] (if supported by evaluator)

Special Tokens

Token Meaning

Hi! Program start

Bye! Program end

. Statement terminator

\n Newline (optional)

Grammar in BNF form

%Program Structure

program ::= "Hi!" NEWLINE statements "Bye!" NEWLINE

%Statements

statements ::= statement NEWLINE statements
| ϵ

statement ::= var_decl "."
| assignment "."
| output_stmt "."
| if_stmt
| for_loop
| until_loop
| ternary_stmt "."

%Variable

var_decl ::= "LetsSay" IDENTIFIER "is" value
| "LetsSay" IDENTIFIER "isAlso" IDENTIFIER

assignment ::= IDENTIFIER "is" expression

%Output

output_stmt ::= "Show" expression

%Control Structures

if_stmt ::= "When" condition NEWLINE
"Then" NEWLINE statements
"Otherwise" NEWLINE statements
"ThenStop"

for_loop ::= "ForAll" IDENTIFIER "in" IDENTIFIER ":" NEWLINE
statements
"StopNow"

until_loop ::= "Until" condition ":" NEWLINE
statements
"NowStop"

ternary_stmt ::= "When" condition "Then" statement "Otherwise" statement "ThenStop"

%Expressions with Precedence

expression ::= logical_or

logical_or ::= logical_and
| logical_and "EitherOr" logical_or

logical_and ::= comparison
| comparison "AsWellAs" logical_and

comparison ::= additive
| additive comparison_operator additive

additive ::= multiplicative
| multiplicative "plus" additive
| multiplicative "minus" additive

multiplicative ::= primary
| primary "times" multiplicative
| primary "dividedBy" multiplicative

primary ::= NUMBER
| STRING
| BOOLEAN
| IDENTIFIER
| "(" expression ")"

%Values

value ::= NUMBER
| STRING
| BOOLEAN
| LIST

%Conditions

condition ::= expression comparison_operator expression

%Operators

comparison_operator ::= "IsEqualTo"
| "IsNotEqualTo"
| "IsGreaterThan"
| "IsLessThan"
| "IsAtLeast"
| "IsAtMost"
| "IsNot"

%Terminals

IDENTIFIER ::= [a-zA-Z][a-zA-Z0-9]*
NUMBER ::= [0-9]+ ("." [0-9]+)?
STRING ::= "\"" chars "\""
chars ::= char | char chars
char ::= [a-zA-Z0-9]
BOOLEAN ::= "true" | "false"
LIST ::= "[" (value ("," value)*)? "]"
NEWLINE ::= "\n"