Programming Language Specification Document

Language Name: Fourward

Introduction:

Fourward is a simple, beginner friendly programming language made for learning programming concepts while keeping things simple. The language focuses on readability and straightforward syntax, making it great for first time programmers.

Structure of Statements:

Statements in Fourward are terminated by a semicolon (;). Statements can include variable declarations, arithmetic operations, conditionals, and loops. Indentation is not needed but can make code easier to read

Example:

let x = 5;

if (x > 3) {

print("x is greater than 3");

}

Reserved Words:

The following keywords are reserved in Fourward:

let, const, if, else, while, for, function, return, print, input, true, false, null

Data Types:

Fourward has the following data types:

• Integer (int): Whole numbers (5, -3)

• Float (float): Decimal numbers (3.14, -0.5)

• String (str): Text enclosed in double quotes ("Hello")

• Boolean (bool): Logical values (true, false)

• Null (null): Represents the absence of a value

Arithmetic Operations:

Fourward supports basic arithmetic operations:

• Addition (+)

• Subtraction (-)

• Multiplication (\*)

• Division (/)

• Modulus (%)

Comparative Operators:

• Equal to (==)

• Not equal to (!=)

• Greater than (>)

• Less than (<)

• Greater than or equal to (>=)

• Less than or equal to (<=)

Selection Sequences (Control Flow):

Fourward supports conditional statements using if, else if, and else. Example:

if (x > 10) {

print("x is large");

} else {

print("x is small");

}

Repetition Sequences (Loops):

• while loop: Repeats as long as a condition is true.

• for loop: Iterates over a specified range or collection.

Example:

for (let i = 0; i < 5; i++) {

print(i);

}

Procedures, Functions, and Methods:

Functions are defined using the function keyword and can return values using the return statement. Example:

function add(a, b) {

return a + b;

}

Token Identification:

Tokens are identified by their line number and position within the line. The tokenizer scans the input stream and assigns token types and locations, which are stored in a symbol table for later reference.

Example Token Format:

Line 1, Col 5: let (keyword)

Line 1, Col 9: x (identifier)

Line 1, Col 11: = (operator)

Line 1, Col 13: 5 (integer)