LogicScript Programming Language— User Manual

1. Introduction and Objectives

Welcome to the **LogicScript** programming language! LogicScript is a simple, educational language designed for the Compiler Design Laboratory project. The objective is to practically apply the theoretical concepts of compiler design by implementing a simple language that supports essential programming constructs.

2. Basic Syntax and Keywords

LogicScript is case-sensitive, and each statement ends with a semicolon (;).

Reserved Keywords (LogicScript Unique)

Category	**Keywords**	**Purpose**
Data Types	WHOLE, SCALAR, PHRASE	Integer, Float, and String declaration[4].
I/O	GRAB, SHOUT	Input and Output operations[5].
Conditional	IF_SO, THEN, OTHER_CASE, END_IF	Conditional branching structure[6].
Looping	CYCLE, WHILE, DO, END_CYCLE	Condition-based repetition[7].
Literals	TRUE, FALSE	Boolean values.

3. Variables and Declaration

Variables must be declared before use and support integer, float, and string types. The declaration uses the category-based keywords followed by assignment (=).

Data Type	Keyword	Example
Integer	WHOLE	WHOLE counter = 10;
Float	SCALAR	SCALAR price = 49.99;
String	PHRASE	PHRASE message = "Success";

4. Input and Output Operations

LogicScript uses specific keywords for handling I/O.

- **Input:** GRAB var; — Reads user input into the specified variable.
- **Output:** SHOUT expr1, expr2, ...; — Prints one or more expressions or variables.

Example:

```
PHRASE user_name = "";
SHOUT "Please enter your name:";
GRAB user_name;
SHOUT "Welcome, ", user_name, " to LogicScript.";
```

5. Comments

Comments are supported for code documentation.

- **Single-line comments:** Start with the hash symbol (#) and continue to the end of the line.
- **Multi-line comments:** Enclosed between {# and #}.

Example Comments:

```
# This is a single-line comment describing the next block.

{#

This block handles the primary logic and complex calculations.

#}

WHOLE result = 0;
```

6. Conditional Statements

LogicScript uses the IF_SO ... OTHER_CASE ... END_IF structure for conditional branching. The OTHER_CASE block is optional.

Syntax:

```
IF_SO (condition) THEN

        // Code block if condition is

TRUE OTHER_CASE

        // Code block if condition is

FALSE END_IF;
```

Example:

```
IF_SO (age >= 18) THEN

         SHOUT "Adult rate applies.";

OTHER_CASE SHOUT "Child rate applies.";

        END_IF;
```

7. Loops

LogicScript provides a condition-based loop using the CYCLE WHILE structure.

Syntax:

```
CYCLE WHILE (condition) DO
        // Code block inside the loop
END_CYCLE;
```

Example:

```
WHOLE count = 5;
 CYCLE WHILE (count > 0) DO
        SHOUT "Count is: ", count;
        count = count - 1;
END_CYCLE;
```

8. Expressions and Operators

Expressions combine literals, variables, and operators to compute values

Category	Operator	Meaning
Arithmetic	+, -, *, /	Standard math operations.
Relational	==, !=, <, <=, >, >=	Used within conditions.
Assignment	=	Assigns a value to a variable.

9. Example LogicScript Program

This program demonstrates variable declaration, I/O, a conditional statement, and a loop.

```
# LogicScript Sample Program for Compiler Demo
PHRASE user_name = "";
WHOLE limit = 0;
```

```
SHOUT "Enter your name and a loop limit:";
GRAB user_name;
GRAB limit;

 IF_SO (limit > 10) THEN
        SHOUT user_name, " - That's a large loop!";
 OTHER_CASE
         SHOUT user_name, " - Starting loop...";
END_IF;

WHOLE i = 0;
CYCLE WHILE (i < limit) DO
        SHOUT "Iteration: ", i;
        i = i + 1;
END_CYCLE;

SHOUT "Program completed successfully.";
```

10. Conclusion

The LogicScript Programming Language serves its primary objective as a foundational tool for the **Compiler Design Laboratory Project**. This manual has documented the complete specification of LogicScript, providing a clear and comprehensive blueprint for its implementation.