MaxLang Language Specification v1.0

Complete Language Reference Manual

Table of Contents

---

1. Introduction

1.1 Purpose

MaxLang is a domain-specific language designed for finding maximum values between integers. It provides a simple, intuitive syntax for variable manipulation and maximum value computation.

1.2 Design Philosophy

- **Simplicity**: Minimal syntax, easy to learn
- **Safety**: Strong type checking, compile-time error detection
- **Clarity**: Readable code with explicit operations

1.3 Scope

MaxLang supports:

- Integer arithmetic
- Variable assignment
- Maximum value computation
- Output operations

2. Lexical Structure

2.1 Character Set

MaxLang uses ASCII character encoding.

2.2 Tokens

2.2.1 Keywords

```
print    - Output statement
max      - Maximum function
```

Keywords are reserved and cannot be used as identifiers.

2.2.2 Identifiers

ebnf

```
identifier ::= letter (letter | digit | '_')*
letter     ::= 'a'..'z' | 'A'..'Z' | '_'
digit      ::= '0'..'9'
```

Rules:

- Must start with a letter or underscore
- Case-sensitive
- Cannot be a keyword

Valid Examples:

```
x, y, result, num1, _temp, myVariable
```

Invalid Examples:

```
1num      // Starts with digit
my-var    // Contains hyphen
print     // Reserved keyword
```

2.2.3 Integer Literals

ebnf

```
number ::= digit+
digit  ::= '0'..'9'
```

Range: Standard 32-bit signed integers (-2,147,483,648 to 2,147,483,647)

Examples:

0, 42, 999, 2147483647

2.2.4 Operators

=	Assignment
+	Addition
-	Subtraction
*	Multiplication
/	Division

2.2.5 Delimiters

;	Statement terminator
(	Left parenthesis
)	Right parenthesis
,	Comma (function argument separator)

2.2.6 Comments

//	Single-line comment (to end of line)

Example:

maxlang
```
// This is a comment
x = 5;  // Inline comment
```

2.2.7 Whitespace

Spaces, tabs, and newlines are ignored except as token separators.

---

3. Syntax

3.1 Program Structure

ebnf
```
program ::= statement*
```

A MaxLang program consists of zero or more statements.

3.2 Statements

3.2.1 Assignment Statement

ebnf

```
assignment ::= identifier '=' expression ';'
```

Semantics: Assigns the value of expression to identifier. Creates variable if it doesn't exist.

Examples:

maxlang

```
x = 5;
result = 10 + 20;
y = max(a, b);
```

3.2.2 Print Statement

ebnf

```
print_stmt ::= 'print' expression ';'
```

Semantics: Evaluates expression and outputs the result to standard output.

Examples:

maxlang

```
print 42;
print x;
print max(a, b);
```

3.3 Expressions

3.3.1 Expression Grammar

ebnf

```
expression     ::= additive
additive       ::= multiplicative (('+' | '-') multiplicative)*
multiplicative ::= primary (('*' | '/') primary)*
primary        ::= number
                 | identifier
                 | max_call
                 | '(' expression ')'
```

3.3.2 Operator Precedence (Highest to Lowest)

1. Parentheses ( )
2. Multiplication *, Division / (left-associative)
3. Addition +, Subtraction - (left-associative)

3.3.3 Expression Examples

```maxlang
5                  // Literal
x                  // Variable
x + 5              // Addition
2 * 3 + 4          // Mixed operators: (2 * 3) + 4 = 10
(2 + 3) * 4        // Parenthesized: 5 * 4 = 20
max(x, y)          // Function call
max(a + 1, b * 2)  // Function with expressions
```

3.4 Function Calls

3.4.1 Max Function

```ebnf
max_call ::= 'max' '(' expression ',' expression ')'
```

Semantics: Returns the maximum of two integer values.

Examples:

```maxlang
max(5, 3)          // Returns 5
max(x, y)          // Returns larger of x or y
max(max(a, b), c)  // Nested call
max(x + 1, y - 1)  // With expressions
```

---

4. Semantics

4.1 Variable Semantics

4.1.1 Declaration

Variables are implicitly declared on first assignment. No explicit declaration is required.

```maxlang
x = 10;  // x is created and initialized
```

4.1.2 Scope

MaxLang v1.0 uses global scope for all variables.

4.1.3 Initialization

Variables must be initialized before use in expressions.

Valid:

```maxlang
x = 5;
print x;  // OK: x is initialized
```

Invalid:

```maxlang
print x;  // Error: x not initialized
```

4.2 Expression Evaluation

4.2.1 Arithmetic Operations

- **Addition**: `a + b`
- **Subtraction**: `a - b`
- **Multiplication**: `a * b`
- **Division**: `a / b` (integer division, truncates toward zero)

4.2.2 Division by Zero

Division by zero is a runtime error and terminates execution.

```maxlang
x = 10 / 0;  // Runtime error
```

4.2.3 Overflow

Integer overflow follows C++ signed integer overflow semantics (undefined behavior).

4.3 Execution Model

Programs execute sequentially, statement by statement, from top to bottom.

maxlang

```
x = 5;        // Execute first
y = 10;       // Execute second
print x + y;  // Execute third
```

---

5. Type System

5.1 Type

MaxLang v1.0 supports a single type: **integer** (32-bit signed).

5.2 Type Checking

All type checking is performed at compile time (semantic analysis phase).

5.3 Type Errors

The following are type errors:

- Using undefined variables
- Using uninitialized variables

---

6. Built-in Functions

6.1 max(a, b)

Signature: `max(int, int) -> int`

Description: Returns the maximum of two integer values.

Parameters:

- `a`: First integer expression
- `b`: Second integer expression

Returns: The larger of `a` or `b`. If equal, returns `a`.

Examples:

maxlang

```
max(5, 3)           // Returns 5
max(-5, -10)        // Returns -5
max(0, 0)           // Returns 0
max(x, y)           // Returns larger variable
max(a + b, c * d)   // Returns larger expression result
```

Nesting:

```maxlang
max(max(a, b), c)      // Find max of three values
max(max(a, b), max(c, d))  // Find max of four values
```

---

7. Examples

7.1 Hello Maximum

```maxlang
// Simple maximum finding
x = 5;
y = 3;
print max(x, y);
// Output: 5
```

7.2 Arithmetic Expression Maximum

```maxlang
a = 10;
b = 20;
result = max(a + 5, b - 3);
print result;
// Output: 17
```

7.3 Multiple Comparisons

```maxlang
num1 = 42;
num2 = 17;
num3 = 85;

max_of_two = max(num1, num2);
max_of_all = max(max_of_two, num3);
```

```
print max_of_all;
```

```
// Output: 85
```

7.4 Complex Expressions

```maxlang
x = 2 * 3 + 4;      // x = 10
y = 10 - 2 * 2;    // y = 6
z = (5 + 3) * 2;   // z = 16

result = max(max(x, y), z);
print result;
```

```
// Output: 16
```

7.5 Temperature Comparison

```maxlang
// Find maximum temperature
monday = 75;
tuesday = 82;
wednesday = 78;
thursday = 85;
friday = 80;

week_max = max(max(monday, tuesday), max(wednesday, max(thursday, friday)));
print week_max;
```

```
// Output: 85
```

7.6 Score Evaluation

```maxlang
// Find best score
quiz1 = 85;
quiz2 = 92;
quiz3 = 88;

best_score = max(quiz1, max(quiz2, quiz3));
print best_score;
```

```
// Output: 92
```

---

8. Error Reference

8.1 Lexical Errors

E001: Invalid Character

```
Error: Invalid character '&' at line 1, column 5
```

Cause: Character not in MaxLang alphabet
Fix: Remove or replace with valid character

E002: Invalid Token

```
Error: Invalid token '@var' at line 2, column 1
```

Cause: Malformed identifier or token
Fix: Follow identifier naming rules

8.2 Syntax Errors

E101: Missing Semicolon

```
Parse Error at line 1, column 5: Expected ';' after assignment (found 'y')
```

Cause: Statement not terminated with semicolon
Fix: Add semicolon at end of statement

E102: Missing Parenthesis

```
Parse Error at line 3, column 10: Expected ')' after max arguments (found ';')
```

Cause: Unmatched parentheses
Fix: Balance parentheses

E103: Missing Comma

```
Parse Error at line 2, column 14: Expected ',' between max arguments (found 'y')
```

Cause: Missing comma in function call
Fix: Add comma between arguments

E104: Unexpected Token

```
Parse Error at line 1, column 1: Expected statement (found ')')
```

Cause: Invalid statement structure
Fix: Check statement syntax

8.3 Semantic Errors

E201: Undefined Variable

```
Semantic Error: Variable 'x' not declared
```

Cause: Using variable before assignment
Fix: Assign value before use

E202: Uninitialized Variable

```
Warning: Variable 'y' may be uninitialized
```

Cause: Variable declared but not initialized
Fix: Initialize before use

8.4 Runtime Errors

E301: Division by Zero

```
Runtime Error: Division by zero
```

Cause: Dividing by zero
Fix: Ensure divisor is non-zero

---

9. Language Evolution

9.1 Version History

- **v1.0** (2024): Initial release with basic functionality

9.2 Future Extensions

Planned for v1.1

- `min(a, b)` function
- Comparison operators (`<`, `>`, `==`, `!=`, `<=`, `>=`)
- Boolean type

Planned for v2.0

- Conditional statements (`if-else`)
- Loop constructs (`while`, `for`)
- User-defined functions
- Local scope

Planned for v3.0

- Arrays
- String type
- File I/O
- Module system

---

10. Grammar Summary

Complete EBNF Grammar

```ebnf
(* MaxLang v1.0 Complete Grammar *)

program       ::= statement*

statement     ::= assignment | print_stmt

assignment    ::= identifier '=' expression ';'

print_stmt    ::= 'print' expression ';'

expression    ::= additive

additive      ::= multiplicative (('+' | '-') multiplicative)*

multiplicative ::= primary (('*' | '/') primary)*

primary       ::= number
                | identifier
                | max_call
                | '(' expression ')'

max_call      ::= 'max' '(' expression ',' expression ')'

identifier    ::= letter (letter | digit | '_')*

number        ::= digit+

letter        ::= 'a'..'z' | 'A'..'Z' | '_'

digit         ::= '0'..'9'
```

```
comment       ::= '//' .* '\n'
```

---

11. Standard Library

11.1 Built-in Functions

Function	Signature	Description
`max(a, b)`	`(int, int) -> int`	Returns maximum of two integers

11.2 Future Built-ins

- `min(a, b)`: Minimum of two integers
- `abs(x)`: Absolute value
- `pow(x, y)`: Power function

---

12. Best Practices

12.1 Code Style

```maxlang
// Good: Clear variable names
temperature_max = max(monday_temp, tuesday_temp);

// Bad: Unclear names

x = max(a, b);
```

12.2 Comments

```maxlang
// Good: Explain intent
// Find the highest score from three tests
best = max(test1, max(test2, test3));

// Bad: State the obvious

x = 5; // Assign 5 to x
```

12.3 Expression Clarity

```maxlang
// Good: Use parentheses for clarity
result = (a + b) * (c - d);

// Acceptable but less clear
result = a + b * c - d;
```

---

13. Conformance

A conforming MaxLang implementation must:

1. Implement all language features as specified
2. Report all syntax and semantic errors
3. Follow the evaluation semantics precisely
4. Support the complete grammar

---

14. References

- Compiler Construction: Principles and Practice
- The C Programming Language (for operator semantics)
- Formal Language Theory