

EasyLang Programming Language — User Manual

Welcome to the **EasyLang** programming language! EasyLang is an ultra-simple, educational programming language designed specifically for students learning compiler construction. It offers minimal syntax and essential features to help you understand the fundamental phases of compilation—lexical, syntax, and semantic analysis—with unnecessary complexity.

1. Getting Started

EasyLang programs are written in plain text files with the extension .easy. To run an EasyLang program, compile it using the EasyLang compiler built with Flex and Bison.

Example command:

```
easy.exe program.easy
```

2. Basic Syntax

EasyLang is case-sensitive and ignores whitespace (except within strings). Each statement typically occupies one line.

Keywords are reserved and cannot be used as variable names.

Reserved Keywords: var, print, input, if, repeat

3. Variables

Variables are declared with the var keyword. EasyLang uses **dynamic typing**—variables can hold either numbers or strings, and the type is determined at runtime.

Example:

```
var age = 25
```

```
var name = "Shipon"
```

```
var result
```

```
# Assigning values later:
```

```
age = age + 1
```

```
name = "Alice Johnson"
```

4. Input and Output

EasyLang provides simple I/O operations using print and input.

- print expr1, expr2, ... – Prints one or more expressions separated by spaces.
- input variable – Reads user input into a variable.

Example:

```
print "Enter your name:"
```

```
input name
```

```
print "Hello", name
```

5. Comments

Comments allow you to document your code.

- Single-line comments start with // and continue to the end of the line.

Example:

```
# This is a single-line comment
```

```
// This is a single-line comment
```

```
var x = 10 // This is also a comment
```

6. Conditional Statements

EasyLang supports conditional branching using if (condition) { ... }.

Syntax:

```
if (condition) {  
    statements  
}
```

Example:

```
if (age > 18) {  
    print "You are an adult."  
}
```

7. Loops

EasyLang provides one simple loop type for iteration:

- Counting loop: repeat expression { ... }

Example:

```
// Counting loop  
  
repeat 3 {  
    print "Hello!"  
}
```

```
// Loop with variable  
  
var count = 5  
  
repeat count {  
    print "Count:", count  
    count = count - 1  
}
```

8. Expressions

Expressions combine literals, variables, and operators to compute values.

Supported operators include:

- Arithmetic: +, -, *, /
- Relational: ==, !=, >, <, >=, <=
- Parentheses for grouping: ()

Example:

```
var a = 10
```

```
var b = 3
```

```
var result = (a + 5) * b
```

```
print "Result:", result
```

```
if (a > b) {
```

```
    print "a is greater than b"
```

```
}
```

9. Type System

EasyLang uses **dynamic typing** with two data types:

1. **Number** – Integer or floating-point values
2. **String** – Text enclosed in double quotes

Type examples:

```
var score = 95.5 // Number
```

```
var greeting = "Hi" // String
```

```
var total = score + 5 // Number operation
```

Important: EasyLang performs automatic type detection during input operations.

10. Example Program

Example 1: Basic Greeting Program

```
// Simple greeting program
```

```
var name
```

```
var age
```

```
print "What is your name?"
```

```
input name
```

```
print "How old are you?"
```

```
input age
```

```
print "Hello", name
```

```
print "You are", age, "years old"
```

```
if (age > 18) {
```

```
    print "You are an adult"
```

```
}
```

Example 2: Arithmetic Calculator

```
// Simple calculator
```

```
var num1
```

```
var num2
```

```
var operation
```

```
print "Enter first number:"
```

```

input num1
print "Enter second number:"
input num2
print "Enter operation (+, -, *, /):"
input operation

if (operation == "+") {
    print "Result:", num1 + num2
}

if (operation == "-") {
    print "Result:", num1 - num2
}

if (operation == "*") {
    print "Result:", num1 * num2
}

if (operation == "/") {
    if (num2 != 0) {
        print "Result:", num1 / num2
    } else {
        print "Error: Division by zero"
    }
}

```

Example 3: Counting Loop

```

// Counting program

var n = 5

var i = 1

```

```
repeat n {  
    print "Count", i, "of", n  
    i = i + 1  
}  
  
print "Counting complete!"
```

11. Error Messages

The EasyLang compiler provides helpful error messages:

1. Lexical Errors: Invalid characters or tokens

Error: Invalid character '@' at line 3

2. Syntax Errors: Incorrect grammar structure

Syntax error at line 5: unexpected token

3. Semantic Errors: Meaning-related issues

Error: Variable 'x' not declared

Error: Variable 'score' not initialized

Error: Division by zero

12. Conclusion

EasyLang is a minimalist programming language designed specifically for learning compiler construction principles. Through its simple syntax and limited feature set, EasyLang allows students to focus on the core concepts of lexical analysis, parsing, and semantic analysis without being overwhelmed by language complexity.