MiniLang Programming Language — User Manual

MiniLang is a simple educational programming language designed for Compiler Design Lab projects.

It helps students understand how lexical analysis, syntax analysis, and semantic evaluation work in a real compiler.

This manual explains how to write MiniLang programs and run them using your MiniLang compiler built with Flex & Bison.

1. Getting Started

MiniLang programs are written in plain text files with the extension .ml or .minilang.

To run a MiniLang program:

./minilang program.ml

The compiler reads the program, parses it, checks for errors, and executes the statements.

MiniLang ignores extra whitespace (except inside string literals).
Each statement must end with a semicolon ; unless it is a block structure such as if, else, while.

2. Basic Syntax

MiniLang is case-sensitive and uses a clean, readable syntax.

Reserved Keywords

int, float, string, input, print, if, else, while

These keywords cannot be used as variable names.

Statement Ending

Most MiniLang statements end with a semicolon `;`.

Example:

```
a = 10;
print(a);
```

3. Variables

Variables in MiniLang must be declared before use.
Supported types:

- int →integer values

- float →decimal values
- `string` → text values

Variable Declaration

```
int a;
float b;
string name;
```

Declaration with Initialization

```
int x = 5;
float pi = 3.14;
string city = "Dhaka";
```

Reassigning Values

```
a = 20;
name = "Shakib";
b = b + 1.5;
```

4. Input and Output

MiniLang supports simple input/output operations.

Input Statement

```
input(a);
input(name);
```

The user enters a value which is stored into the variable.

Output Statement

```
print(a);
print("Hello");
print(name);
print(a + 10);
```

Multiple prints are allowed:

```
print("Value is:", a);
```

5. Comments

MiniLang supports two types of comments.

Single-line Comment

```
# This is a comment
```

Multi-line Comment

```
/*
   This is a
   multi-line comment
*/
```

Comments are ignored by the compiler.

6. Conditional Statements

MiniLang supports if–else for conditional execution.

Syntax

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

Example

```
if (age >= 18) {
    print("Adult");
} else {
    print("Minor");
}
```

The condition must evaluate to true (non-zero) or false (zero).

7. Loops

MiniLang supports looping using while.

While Loop

```
while (condition) {
    statements
}
```

Example

```
while (x < 5) {
    print(x);
    x = x + 1;
}
```

8. Expressions

Expressions can combine variables, literals, and operators.

Supported Operators

Arithmetic

+, -, *, /

Relational

==, !=, <, >, <=, >=

Logical

&&, ||, !

Example Expressions

```
a + 5
b * (a - 2)
x >= 10
!(a == b)
```

9. Example Program

Below is a complete MiniLang program demonstrating all features:

```
# MiniLang Example Program

int a;
float b;
string name;

a = 5;
b = 3.5;
name = "Shakib";

print("Initial values:");
print(a);
print(b);
print(name);

input(a);  # user input
print("New value of a:");
print(a);
```

```
if (a > 10) {
    print("Greater than 10");
} else {
    print("10 or smaller");
}

while (a < 15) {
    print("Loop value:", a);
    a = a + 1;
}
```

10. Conclusion

MiniLang is a lightweight programming language created for learning compiler design.
Using simple syntax and core programming concepts such as variables, I/O, conditions, and loops, MiniLang helps students understand each stage of compilation practically.This manual provides all necessary details to write and execute MiniLang programs effectively.
With this foundation, students can expand the language by adding functions, arrays, new operators, or code generation features.