B-Lang Compiler — User Manual

1. Introduction

B-Lang (Bengali Language) is an educational, mini-programming language designed to demonstrate the fundamental phases of compilation (Lexical, Syntax, and Semantic Analysis). The B-Lang project is built using **Flex**, **Bison**, and a **C** backend.

B-Lang's primary feature is its syntax, which utilizes Bengali keywords in an "Avro" or "Banglish" style. Its syntax is C-inspired, where each statement must end with a semicolon (`;`), code blocks use curly braces (`{}`), and whitespace is ignored (except inside strings).

B-Lang source files use the extension: `.bl`

To compile and run a program (in the terminal):

Bash
```
./compiler.exe program.bl
```

2. Keywords

The following identifiers are reserved and cannot be used as variable names:

- **Data Types:** `shongkha`, `doshomik`, `lekha`
- **Control Flow:** `jodi`, `nahole`, `jotokhhon`
- **Input/Output:** `poro`, `dekhaw`

3. Variables and Types

B-Lang supports three primitive data types:

- **`shongkha`** — Integer values (e.g., `10`, `-5`, `0`)
- **`doshomik`** — Floating-point values (e.g., `3.14`, `0.5`)
- **`lekha`** — String or text values (e.g., `"Hello"`)

3.1 Variable Declaration

Variables must be declared with their type before use. The language does not support Type Inference.

Syntax:

type_keyword variable_name;

Example:

Code snippet
```
shongkha a;
doshomik b;
lekha myString;
```

4. Assignments

After declaration, variables can be assigned a value. The semantic analyzer checks if the value's type matches the variable's declared type.

Example:
Code snippet
```
shongkha age;
age = 21;

lekha name;
name = "B-Lang";

cc This will cause a Type Mismatch Error:
cc age = "A Mistake";
```

5. Literals

B-Lang supports the following literal forms:

- **Integer:** `10`, `0`, `-50`
- **Float:** `2.5`, `0.0`, `3.1416`
- **String:** Delimited by double quotes (`" "`), e.g., `"Amar Sonar Bangla"`

6. Input and Output

B-Lang provides simple I/O functions using `poro` and `dekhaw`.

6.1 Input — `poro`

Reads user input from the console and stores it in a variable. The language performs a type check based on the variable's declared type.

Syntax:

poro variable_name;

Example:
Code snippet
```
shongkha year;
poro year;
```

6.2 Output — `dekhaw`

Prints an expression or value to the console.

Syntax:

dekhaw expression;

Example:
Code snippet
```
doshomik pi = 3.14;
dekhaw pi;
dekhaw "Hello World";
```

7. Expressions

Currently, B-Lang expressions support type checking based on the symbol table. It supports the following forms:

- **Literals:** e.g., `10`, `3.14`, `"Hello"`
- **Variables:** e.g., `age`, `name`

(Full arithmetic or relational expressions, such as `a + b` or `a > 10`, are planned for future enhancement).

8. Control Flow

B-Lang supports two main control flow structures.

8.1 IF / ELSE (jodi / nahole)

Executes code blocks based on a condition.

Syntax:
Code snippet
```
jodi (condition_expression) {
    ... statements ...
} nahole {
    ... statements ...
}
```

Example:
Code snippet
```
shongkha a = 10;
jodi (a) { cc will run if 'a' is non-zero
    dekhaw "Hello";
} nahole {
    dekhaw "World";
```

```
}
```

8.2 WHILE Loop (jotokhhon)

Repeatedly executes a code block as long as a condition is true (non-zero).

Syntax:
Code snippet
```
jotokhhon (condition_expression) {
    ... statements ...
}
```

Example:
Code snippet
```
shongkha i = 5;
jotokhhon (i) { cc will run as long as 'i' is not 0
    dekhaw i;
    cc i = i - 1; (Future plan)
}
```

9. Comments

B-Lang supports two types of comments for code documentation:

- Single-line Comment: Begins with cc.

  cc This is a single-line comment

- **Multi-line Comment:** Begins with `mcc` and ends with `mcc`.

  Code snippet
  ```
  mcc
    This is a
    multi-line comment block
  mcc
  ```

10. Example Program (Full)

Below is a complete B-Lang program demonstrating various features:

Code snippet
```
cc Program: User Greeting
cc This program takes the user's name and age as input

lekha name;
shongkha age;
doshomik height;

dekhaw "--- User Registration ---";
```

```
dekhaw "Enter your name:";
poro name;

dekhaw "Enter your age (number):";
poro age;

dekhaw "Enter your height (decimal):";
poro height;

cc Now displaying the inputs
dekhaw "--- Your Information ---";
dekhaw "Name: ";
dekhaw name;
dekhaw "Age: ";
dekhaw age;
dekhaw "Height: ";
dekhaw height;
```

11. Future Enhancements

The following features are planned for future releases to make B-Lang more robust:

- **Arithmetic Expressions:** Support for `+`, `-`, `*`, `/` operators (e.g., `a = b + c;`)
- **Relational Expressions:** Use of relational operators (`<`, `>`, `==`) in control flow (e.g., `jodi (a > 10)`)
- **Logical Operators:** `and`, `or`, `not`
- **Functions:** User-defined functions (`func`)
- **Arrays:** Support for collections or arrays.

12. Conclusion

The B-Lang Compiler project presents the complex process of compiler construction in a simple and understandable way. Its unique Bengali syntax makes programming language design more engaging for new developers. This project serves as a solid foundation for building semantic analysis, type checking, and a basic interpreter.