Bangla Programming Language

User Manual

Welcome to the Bangla programming language. The Bangla Programming Language Compiler is a custom-designed educational compiler that allows users to write programs using simple Bangla keywords such as dhoro, deo, dekhao, jodi, cholo, and more. Its goal is to make programming easier for beginners by using familiar language terms instead of English keywords.

1. Getting Started

Bangla programs are written in plain text files with the extension .ruby. To run a Bangla program, compile it using your Bangla compiler built with Flex and Bison.

Example command:

```
./bangla_compiler example.ruby
```

Creating our first program:

First of all create a file named hello.ruby and we have to write our code. The Bangla compiler will parse and execute your program.

2. Basic Syntax

Bangla Programming Language is case-sensitive and ignores whitespace except within strings. Each statement ends with a newline. Keywords are reserved and cannot be used as variable names.

Reserved Keywords:

Bengali Keyword	English Equivalent	Purpose
dhoro	let	Variable declaration
deo	scan	Read user input
dekhao	print	Display output
jodi	if	Conditional branching
tahole	then	Conditional block
nahole	else	Alternative branch
koro	do	Loop execution
jotokhon	while	Loop condition
sesh	end	Block termination

3. Data Types

Bangla Programming Language supports three fundamental data types for storing different kinds of information.

Supported Types:

- int → Integer values (whole numbers)
- float → Floating-point values (decimal numbers)
- string → Text and character sequences

Type Characteristics:

- Integer operations: Addition, subtraction, multiplication, division, modulo
- Float operations: All arithmetic operations with decimal precision
- String operations: Concatenation and basic manipulation

4. Variables

Variables are declared with the dhoro keyword followed by a type, identifier, and optional initializer. Variables store data that can be modified throughout program execution.

Declaration Syntax:

```
dhoro int variable_name = initial_value
dhoro float variable_name = initial_value
dhoro string variable_name = "initial_value"
```

Examples:

```
dhoro int age = 21
dhoro float pi = 3.14159
dhoro string name = "Jubayer"
```

Assigning values later

```
age = age + 1
name = name + " Hossain"
```

Variable Naming Rules:

- Must start with a letter or underscore
- Can contain letters, numbers, and underscores
- Case-sensitive (age, Age, AGE are different)
- Cannot be reserved keywords

5. Input and Output

Bangla Programming Language provides simple I/O functions using `deo` for reading input and `dekhao` for printing output.

Output with `dekhao`:

- `dekhao expr1, expr2, ...` → Prints one or more expressions separated by spaces

Input with `deo`:

- `deo variable` → Reads user input into a variable (type must match)

Examples:
dekhao "Enter your name:"
deo name
dekhao "Hello, ", name
dekhao "Your age is: ", age

6. Comments

Comments allows to document our code and are ignored by the compiler.

Single-line comments:

- Start with `#` and go to the end of the line

Multi-line comments:

- Enclosed between `/*` and `*/`

Examples:

This is a single-line comment

/*
This is a
multi-line comment
for documentation
*/

7. Conditional Statements

Bangla Programming Language supports conditional branching using `jodi ... tahole ... nahole ... sesh` syntax. The `nahole` (else) block is optional.

Conditional Syntax:
jodi (condition) tahole
# statements to execute if true
nahole
# statements to execute if false
sesh

Examples:

```
jodi age >= 18 tahole
dekhao "You are an adult."
nahole
dekhao "You are a minor."
sesh
```

```
jodi score > 80 tahole
dekhao "Grade: A"
nahole
jodi score > 60 tahole
dekhao "Grade: B"
nahole
dekhao "Grade: C"
sesh
sesh
```

Comparison Operators:

- `==` --- Equal to
- `!=` --- Not equal to
- `<` --- Less than
- `<=` --- Less than or equal to
- `>` --- Greater than
- `>=` --- Greater than or equal to

8. Loops

Our Language provides loop constructs for iteration and repeated execution of code blocks.

While Loop Syntax:

```
jotokhon (condition) koro
# statements to repeat
sesh
```

Do-While Loop Syntax:

```
koro
# statements to execute
jotokhon (condition)
```

Loop Examples:

While loop - counting down

```
dhoro int x = 5
jotokhon x > 0 koro
dekhao "x = ", x
x = x - 1
sesh
```

Do-while loop - execute at least once

```
dhoro int counter = 0
koro
dekhao "Counter: ", counter
counter = counter + 1
jotokhon counter < 5
```

Infinite loop

```
jotokhon 1 koro
# statements (use break to exit)
sesh
```

9. Expressions

Expressions combine literals, variables, and operators to compute values. Expressions can be used in assignments, conditions, and output statements.

Arithmetic Operators:

- `+` → Addition
- `-` → Subtraction
- `*` → Multiplication
- `/` → Division
- `%` → Modulo (remainder)

Logical Operators:

- `and` → Logical AND
- `or` → Logical OR
- `not` → Logical NOT

Expression Examples:

```
dhoro int a = 10
dhoro int b = 3
dhoro float result = (a + b) * 2.5

dekhao a + b
dekhao a - b
dekhao a * b
dekhao a / b
dekhao a % b

dhoro int x = 5
jodi x > 3 and x < 10 tahole
dekhao "x is between 3 and 10"
sesh
```

Complete Example Program

Program: Student Grade Calculator

Determine grade based on score

```
jodi score >= 90 tahole
grade = "A"
nahole
jodi score >= 80 tahole
grade = "B"
nahole
jodi score >= 70 tahole
grade = "C"
nahole
jodi score >= 60 tahole
grade = "D"
nahole
grade = "F"
sesh
sesh
sesh
sesh

dekhao "--- Results ---"
dekhao "Student: ", student_name
dekhao "Score: ", score
dekhao "Grade: ", grade
```

Loop to show performance analysis

```
dhoro int performance = 0
jotokhon performance < 3 koro
dekhao "Analysis iteration: ", performance + 1
performance = performance + 1
sesh

dekhao "Calculation complete!"
```

10. Conclusion

The Bangla Programming Language Compiler is designed to make programming easier and more familiar for beginners by using Bangla keywords and simple syntax. Its goal is to help new learners understand core programming concepts—such as variables, input/output, conditions, loops, and expressions—in a language they are comfortable with.

With features like string and numeric handling, keyboard input, if–else, loops, and comments, this compiler can be used to build small logical programs while learning the foundations of coding.