



Project Proposal

Course Code: CSE314

Course Title: Compiler Design Lab

Project Title: Icarus: The Banglisch Programming Language

Submitted To:

Daffodil International University

Name: Tamanna Sultana, Lecturer

Department: Computer Science and Engineering
Daffodil International University.

Submitted By:

Nabid Rezuan (0242310005101144)

Raiyan Jahan (0242310005101599)

Rizwan Karim Lam (0242310005101608)

Ahsanur Rahman Nakib (0242310005101611)

Fahim Montasir (0242310005101748)

Section: 64_N1

Department: Computer Science and Engineering

Daffodil International University.

Submission Date: 23-07-2025

Introduction:

This project seeks to develop a simple interpreter for a miniature programming language using Flex and Bison. The language supports features like variable declarations, arithmetic operations, control structures (such as if-else and loops), and basic input/output. It offers hands-on experience with essential compiler design concepts, including lexical analysis, syntax analysis, parsing, and symbol table handling.

Objectives:

- Define Language Syntax and Semantics: Create clear and concise rules for statements, expressions, and control flow structures.
- Lexical Analysis with Flex: Develop a lexer to tokenize the source code into components like keywords, identifiers, operators, and numbers.
- Syntax Analysis with Bison: Implement grammar rules to parse token sequences and generate a parse tree.
- Symbol Table Management: Maintain variable information, ensure proper declarations, and handle type checking.
- Interpretation or Output Generation: Execute the parsed code or produce the appropriate output.
- Basic Error Handling: Identify and report syntax and semantic errors during compilation or interpretation.

Scope:

- Data Types: Support integer variables.
- Operators: Implement arithmetic (+, -, *, /, %), assignment (=), comparisons (==, !=, <, >, <=, >=).
- Control Structures: Parse and execute `jodi`, `nahole`, `jokhon`, and `ejonne` statements.
- Input/Output: Support `por()` for input and `lekh()` for output.
- Comments: Single-line comments with `**`.
- Variable Features: Declaration, assignment, and use in expressions, with enforcement of declaration-before-use.

Tools and Technologies:

- Flex: Generates the lexical analyzer (tokenizer).
- Bison: Generates the parser (syntax analyzer, grammar rules).
- C/C++: Main implementation language interfacing with generated parsers.
- GCC: Compiles and links the project components.

Methodology:

1. Language Design:
 - Define syntax rules and grammar for statements, expressions, and control structures.
2. Lexical Analysis with Flex:
 - Write regular expressions to tokenize reserved words (e.g., `jodi`, `jokhon`, `lekh`), variables, numbers, and symbols.
3. Parsing with Bison:
 - Write context-free grammar rules for valid statement and expression formation.
 - Build a parse tree and perform expression evaluation during parsing.
4. Symbol Table Management:
 - Implement data structures for variable names, types, and values.
 - Support type checking and enforce constraint rules.
5. Execution/Interpretation:
 - Either execute statements as parsed or generate simple intermediate code.
 - Display or process results for `lekh()` and handle input via `por()`.
6. Testing and Validation:
 - Devise a suite of test programs to check handling of all syntax and semantic rules, as well as error cases.

Expected Output:

- A working interpreter or compiler for the custom language, Icarus.
- Capability to correctly parse and run valid source code files.
- Meaningful error messages for syntax or semantic errors.

Conclusion:

This project provides practical experience in developing a compiler/interpreter by building a simple but functional programming language from the ground up, covering lexical analysis, parsing, symbol management, and code execution.