COMPILER CONSTRUCTION PROJECT – PHASE 02

Syntax Analyzer (Parser) Implementation

Student Name: Inshrah Alam

Roll No: L1F22BSCS0384

Section: G4

---

1. Project Objective

The objective of Phase 02 is to design and implement a **Syntax Analyzer** using YACC/Bison that validates the grammatical structure of the custom "Mini C++" language defined in Phase 01. This phase integrates the Lexical Analyzer (Scanner) with a Context-Free Grammar (CFG) to ensure that programs written in this language follow valid syntactic rules.

2. Grammar Design (Context-Free Grammar)

The following production rules define the structure of the language. The grammar is designed to enforce that all code resides within a specific start and end block (`AAGHAZ` and `IKHTITAM`).

Terminal Symbols: `AAGHAZ`, `IKHTITAM`, `AMAL`, `PUNJI`, `MEEZAN`, `SHART`, `WARNAH`, `FARZ`, `CHALA`, `id`, `num`, `;`, `(`, `)`, `{`, `}`.

Production Rules:

$$\begin{aligned}
\text{Program} &\rightarrow \text{AAGHAZ FunctionList IKHTITAM} \\
\text{FunctionList} &\rightarrow \text{Function} \mid \text{Function FunctionList} \\
\text{Function} &\rightarrow \text{AMAL id () \{ StmtList \}} \\
\text{StmtList} &\rightarrow \text{Stmt} \mid \text{Stmt StmtList} \\
\text{Stmt} &\rightarrow \text{Declaration} \mid \text{Assignment} \mid \text{IfStmt} \mid \text{WhileStmt} \mid \text{ForStmt} \mid \text{IOStmt} \mid \text{ReturnStmt} \\
\text{Declaration} &\rightarrow \text{Type id ;} \mid \text{Type id = Expr ;} \\
\text{Type} &\rightarrow \text{PUNJI} \mid \text{MEEZAN} \mid \text{LISAN} \\
\text{Assignment} &\rightarrow \text{id = Expr ;} \mid \text{id :? Expr ; (Cond Assign)} \\
\text{IfStmt} &\rightarrow \text{SHART (Expr) \{ StmtList \} [WARNAH \{ StmtList \}]} \\
\text{WhileStmt} &\rightarrow \text{CHALA (Expr) \{ StmtList \}} \\
\text{IOStmt} &\rightarrow \text{NIKAL => Expr ;} \mid \text{DAKHAL => id ;}
\end{aligned}$$

---

3. FIRST and FOLLOW Sets

Below are the calculated FIRST and FOLLOW sets for the primary non-terminals `Program` and `Stmt` (Statement).

Non-Terminal: Program

- **FIRST(Program)** = `{ AAGHAZ }`
  - *Reasoning:* The grammar rule for Program explicitly starts with the token `AAGHAZ`.
- **FOLLOW(Program)** = `{ $ }` (End of Input)
  - *Reasoning:* The Program represents the entire file; nothing follows it except the end-of-file marker.

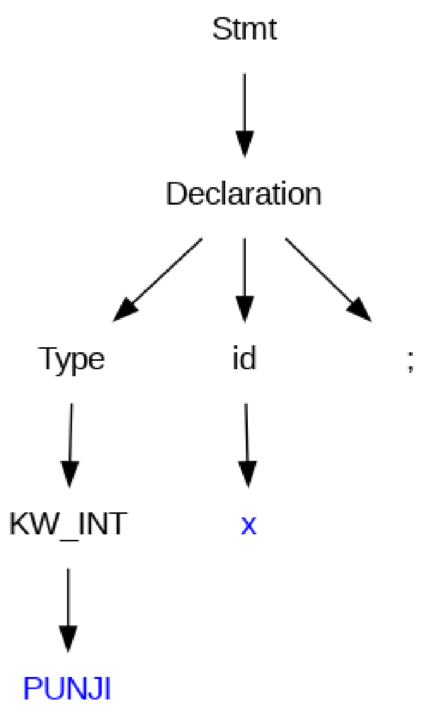
Non-Terminal: Stmt (Statement)

- **FIRST(Stmt)** = `{ PUNJI, MEEZAN, LISAN, IDENTIFIER, SHART, CHALA, FARZ, NIKAL, DAKHAL, VAPIS }`
  - *Reasoning:* A statement can begin with a data type (declaration), an identifier (assignment), or any control flow keyword (if, while, etc.).
- **FOLLOW(Stmt)** = `{ PUNJI, MEEZAN, ..., IDENTIFIER, SHART, ..., RBRACE ('}') }`
  - *Reasoning:* A statement is followed by the start of the next statement or the closing brace of the current block.

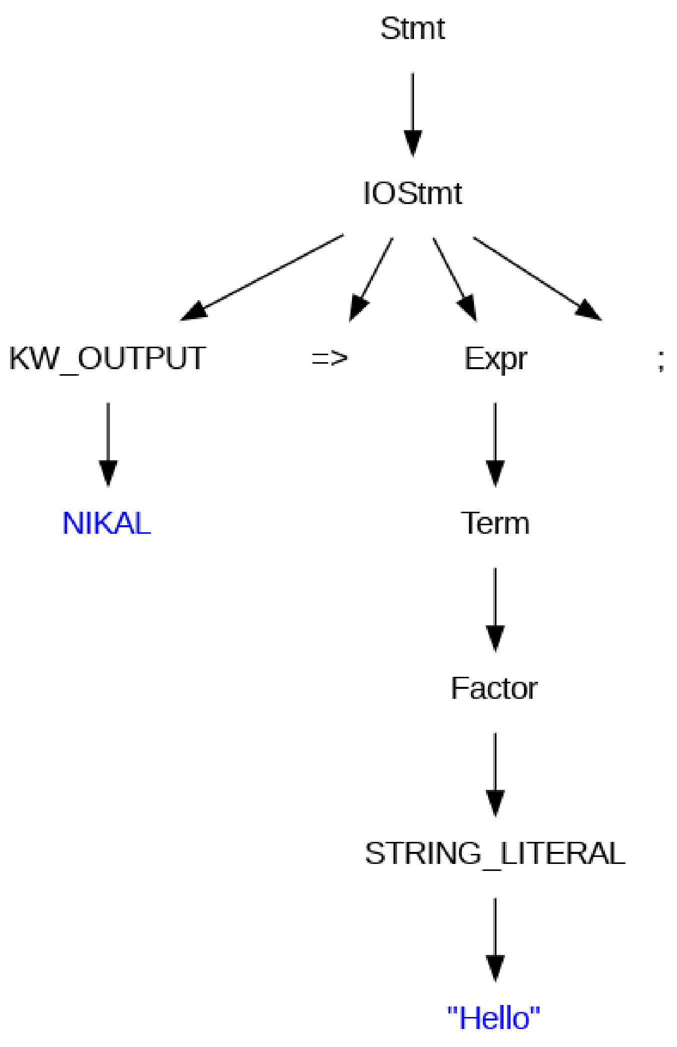
---

4. Parse Tree Visualization

Example Code Segment: `PUNJI x;` (Integer Declaration)



Example Code Segment:`NIKAL => "Hello";` (Output Statement)



---

5. Phase 01 Integration Strategy

The Lexical Analyzer from Phase 01 was integrated as follows:

1. **Token Reuse:** All regex patterns defined in Phase 01 (`scanner.l`) were preserved.
2. **Return Mechanism:** The `scanner.l` actions were modified from `fprintf` (file writing) to `return TOKEN_ID`. This allows the YACC parser to consume tokens one by one.
3. **Shared Header:** The `y.tab.h` file generated by YACC is included in the scanner, ensuring that both the scanner and parser agree on the integer values of tokens like `KW_START` or `KW_INT`.

---

6. Test Cases

6.1 Valid Program (`test.mc`)

This program tests declarations, loops, and conditions using the custom Urdu keywords.

C++

```
AAGHAZ

AMAL main() {
    PUNJI x;
    MEEZAN y = 5.5;

    NIKAL => "Starting Analysis...";

    x = 10;
    x ++++;  /* Custom operator test */

    SHART (x > 5) {
        NIKAL => "Value is valid";
    } WARNAH {
        NIKAL => "Value is too low";
    }

    CHALA (x > 0) {
        x = x - 1;
    }

    VAPIS 0;
}

IKHTITAM
```

6.2 Invalid Program (`error.mc`)

This program contains two errors: missing the required `AAGHAZ` block start and a missing semicolon.

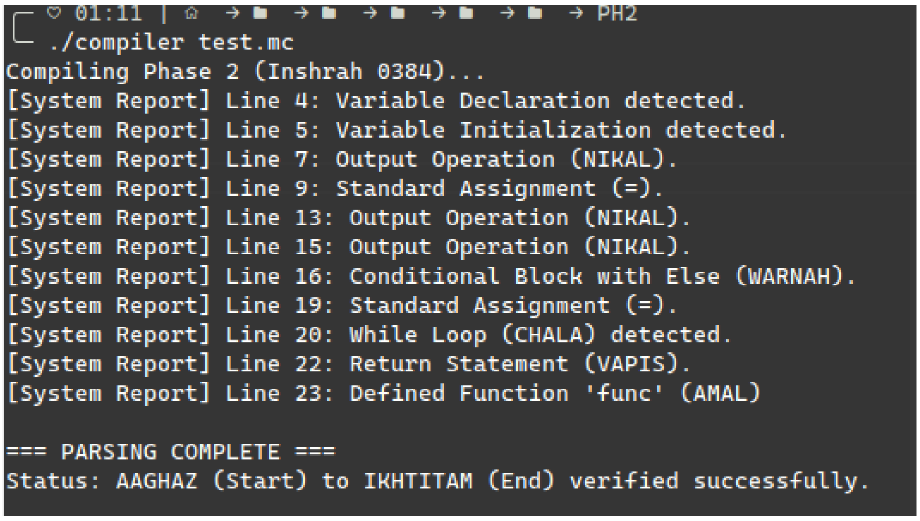
C++

```
/* Error 1: Missing AAGHAZ at start */

AMAL main() {
    PUNJI x
    x = 10; /* Error 2: Missing semicolon on previous line */
}

IKHTITAM
```

---

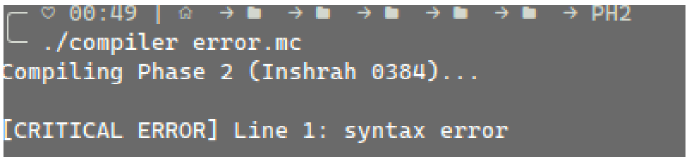
7. Execution Outputs

success output:



```
┌─ ♡ 01:11 | ⌂  → ■  → ■  → ■  → ■  → ■  → PH2
└─ ./compiler test.mc
Compiling Phase 2 (Inshrah 0384)...
[System Report] Line 4: Variable Declaration detected.
[System Report] Line 5: Variable Initialization detected.
[System Report] Line 7: Output Operation (NIKAL).
[System Report] Line 9: Standard Assignment (=).
[System Report] Line 13: Output Operation (NIKAL).
[System Report] Line 15: Output Operation (NIKAL).
[System Report] Line 16: Conditional Block with Else (WARNAH).
[System Report] Line 19: Standard Assignment (=).
[System Report] Line 20: While Loop (CHALA) detected.
[System Report] Line 22: Return Statement (VAPIS).
[System Report] Line 23: Defined Function 'func' (AMAL)

=== PARSING COMPLETE ===
Status: AAGHAZ (Start) to IKHTITAM (End) verified successfully.
```

Error Output:



```
┌─ ♡ 00:49 | ⌂  → ■  → ■  → ■  → ■  → ■  → PH2
└─ ./compiler error.mc
Compiling Phase 2 (Inshrah 0384)...

[CRITICAL ERROR] Line 1: syntax error
```