Fiza Naz
L1F22BSCS0482
Compiler Construction

Assignment#2

Language Overview:

The purpose of this mini C++ language is to provide a simple, readable, and beginner-friendly programming language that demonstrates core programming concepts such as variable declaration, conditional statements, loops and output operations. It is designed mainly to understand lexical and syntax analysis in compiler construction.

Style of syntax:

The syntax is block-structured and inspired by C++ but uses custom Urdu-inspired keywords to make the language unique and expressive.

Curly braces { } are used to define blocks, and statements end with semicolon;.

Reason for Choosing Keywords:

The keywords are choosen from Roman Urdu wants to:

1. Make the language unique
2. Improve readability

③ Ensure originality and creativity.
④ Avoid similarity with standard programing languages.

Key words Used:

- shuru → program start
- agar → if
- agarbhi → else if
- warna → else
- dohrana → loop
- likho → output

Operatoons Used:

- (=) assignment
- (+) addition
- (= =) comparison
- (−) difference

Punctuation used

- { } (block)
- ; (statement terminator)

Grammer Definition (Context Free Grammer)

1- Non - Terminals
<Program>
< Block>
< Statement List>
< Statement>
< Declaration>
< Assignment>

<IfStatement>

<Loop Statement>

< OutputStatement>

< Expression>

< Condition >

Terminals

shuru, agar, agarbhi, warna, dohrana, likho,

IDENTIFIER, NUMBER

=, +, ==, <

{, }, (,), ;

Start symbol

Program

Grammer Rules (CFG)

→ Program Structure

Program → shuru Block

Block → { Statement List }

→ statement List

statementList → Statement StatementList

statementList → ε

Statement

statement → Declaration;

statement → Assignment;

statement → IfStatement

S Production Rules

statement → Loopstatement

Statement → Output Statement;

Variable Declaration:

Declaration → IDENTIFIER IDENTIFIER

Example:

Adad a;

Assignment Statement

Assignment → IDENTIFIER = Expression

Conditional Statement

If statement → agar (Condition) Block

IfStatement → agar (Condition) Block warna Block

Loop Statement

Loopstatement → dohrana (Assignment ; Condition) Block

Output statement

OutputStatement → likho Expression

Expression

Expression → Expression + Expression

Expression → NUMBER

Expression → IDENTIFIER.

Condition

Condition → Expression == Expression

Condition → Expression < Expression

Sample Production Rules

1- Program → shuru Block

2- Block → { statementList }

3- StatementList → Statement statementList.

4- Statement → ε

5- Statement → Declaration ;

6- Statement → Assignment;

7- Statement → If statement

8- Statement → Outputstatement;

9- Statement → Loop Statement

10- Declaration → IDENTIFIFR IDENTIFIER

11- Assignment → IDENTIFIER = Expression

12- IfStatement → agar(condition) Block warna Block

13- Loop statement → dohrana (Assignment; Condition) Block

14- Outputstatement → likho Expression

15- Expression → IDENTIFIER

16- Relop → ==

17- Relop → <

First and Follow sets

First and Follow for statement:

1- First set

FIRST of statement = {IDENTIFIER, agar, dohrana, likho}

Follow = { IDENTIFIER, agar, dohrana, likho }

Program:

FIRST (Program) = { shuru }

Block:

FIRST (Block) = { { }

Declaration:

FIRST (Declaration) = { IDENTIFIER }

Assignment:

FIRST(Assignment) ={ IDENTIFIER }

If statement:

FIRST (Ifstatement) = { agar }

Dohrana (Loopstatement):

FIRST (Loopstatement) = { dohrana }

OUTPUT Statement:

FIRST (OUTPUTstatement) = { likho }

Condition:

FIRST (Condition) = { Number, IDENIIFIER }

Expression:

FIRST (Expression) = { NUMBER, IDENTIFIER }

RelOP:

FIRST (RelOP) = { ==, < }

Follow Sets.

Program	{ $ }
Block	{warna, $}
StatementList	{ } }
Statement	{IDENTIFIER, agar, dohrana, likho, } }
Declaration	{ ; }
Assignment	{ ; }
IfStatement	{IDENTIFIER, agar, dohrana, likho, } }
Loopstatement	{IDENTIFIER, }, agar, dohrana, likho, }
Output statement	{ ; }
Condition	{) }
Expression	{ ; ,), = =, <, + }
RelOp	{number, IDENTIFIER}

Ambiquity Check

Is the grammer ambiguous?

Yes, the grammer can be ambiquous, enspecially in conditional statements (agar-warna) similar to the dangling else problem.

Ambiguous construct Example:

```
agar( a == b)
  agar( b == c)
      likho a;
warna
      likho b;
```

It is unclear which agar the warna belongs to.

Resolution:

Ambiguity can be resolved by.

1- Enforcing block usage {}

2- Associating warna with the nearest unmatched agar

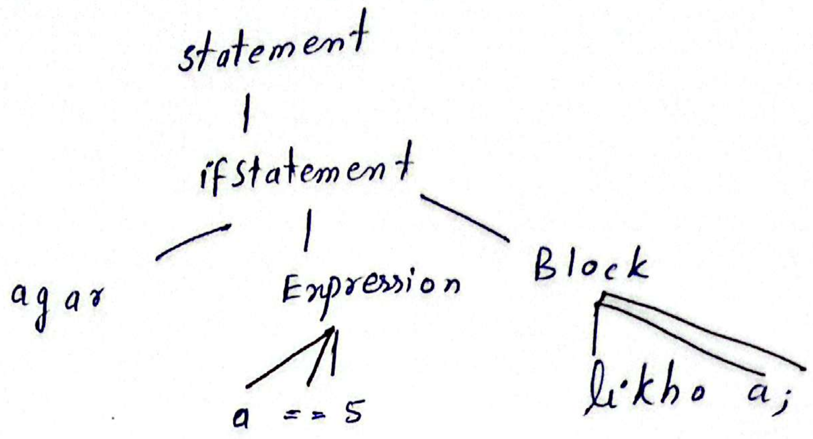
3- Handling ambiguity during parser implementation

Parse Tree Construction

Code.

```
      Adad a;
        a = 5;
        agar (a == 5) {
            likho a;
        }
```

Parse Tree :



Error Scenarios:

Example 1: (Error)

Line 3: a = ;

1) Error Line: 3
2) Violated Rule: Assignment → Identifiend = Expression
3) Expected Token: Number or IDENTIFIER

Example 2: Error

Line 5: agar(a == 5 {

1) Error Line: 5
2) Violated Rule: IfStatement → agar (Expr) Block
3) Expected Token:)