TIPS Language Grammar Guide — Part 2

MSU CSE 4714/6714 — Theory and Implementation of Programming Languages

This guide is designed to help you understand and implement the grammar and parser structure for Part 2 of the TIPS language compiler project. It introduces variables, declarations, compound statements, the abstract Statement hierarchy, and the new 'value' nonterminal. It’s written to be friendly and easy to follow, and serves as both a learning resource and a debugging reference.

PROGRAM

EBNF: program = *PROGRAM IDENT SEMICOLON* **block** *TOK_EOF*
First Tokens: { *PROGRAM* }
Explanation: A TIPS program must start with *PROGRAM*, followed by an identifier name, a semicolon, then a **block**, and end with *TOK_EOF*.

BLOCK

EBNF: block = [*VAR IDENT COLON* (*INTEGER* | *REAL*) *SEMICOLON* (*IDENT COLON* (*INTEGER* | *REAL*) *SEMICOLON*)*] **compound**
First Tokens: { *VAR*, *TOK_BEGIN* }
Explanation: The block may begin with an optional variable declaration section. Each declaration must specify a variable name, type, and end with a semicolon. Multiple declarations are allowed. The block always ends with a **compound**.

COMPOUND

EBNF: compound = *TOK_BEGIN* **statement** (*SEMICOLON* **statement**)* *END*
First Tokens: { *TOK_BEGIN* }
Explanation: A compound statement groups one or more statements. Each statement is separated by a semicolon. Do **not** place a semicolon immediately before *END*.

STATEMENT

EBNF: statement = **assignment** | **compound** | **read** | **write**
First Tokens: { *IDENT*, *TOK_BEGIN*, *READ*, *WRITE* }
Explanation: Statement is an **abstract base class** in the AST. Its concrete subclasses are: **AssignmentStmt**, **CompoundStmt**, **ReadStmt**, and **WriteStmt**. Each subclass implements print_tree() and interpret().

ASSIGNMENT

EBNF: assignment = *IDENT ASSIGN* **value**
First Tokens: { *IDENT* }
Explanation: Assigns a **value** to a variable. The variable must be declared first.

READ

EBNF: read = *READ OPENPAREN IDENT CLOSEPAREN*
First Tokens: { *READ* }
Explanation: Reads a single identifier from input. Lists are not supported in Part 2.

WRITE

EBNF: write = *WRITE OPENPAREN* (*IDENT* | *STRINGLIT*) *CLOSEPAREN*
First Tokens: { *WRITE* }
Explanation: Writes either a variable value or a string literal. Multiple arguments and concatenation are not supported.

VALUE

EBNF: value = *INTLIT* | *FLOATLIT* | *IDENT*
First Tokens: { *INTLIT*, *FLOATLIT*, *IDENT* }
Explanation: A value can be a numeric literal or a variable. Expressions will be introduced in Part 3.

■ Token Reference Table

Below is a summary of the most important tokens defined in lexer.h. Use this table when interpreting grammar rules, writing test programs, or debugging lexer and parser output.

Token Name	Meaning	Example Lexeme
PROGRAM	Start of program	PROGRAM
TOK_BEGIN	Begin compound	BEGIN
END	End compound	END
VAR	Variable declaration	VAR
INTEGER / REAL	Data types	INTEGER / REAL
IDENT	Identifier	myVar
INTLIT / FLOATLIT	Numeric literals	42 / 3.14
STRINGLIT	String literal	'Hello'
ASSIGN	Assignment operator	:=
SEMICOLON	Statement separator	;
COLON	Type separator	:
OPENPAREN / CLOSEPAREN	Parentheses	( )
READ / WRITE	I/O operations	READ / WRITE
TOK_EOF	End of file	EOF