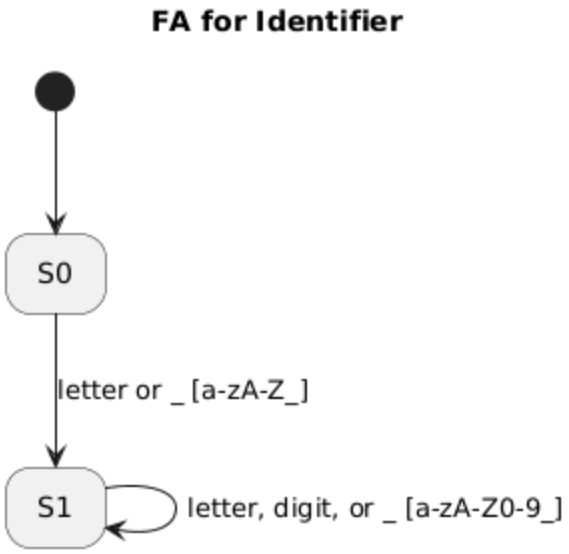
CoreLang: Language Specification Document

1. Regex Token Definitions

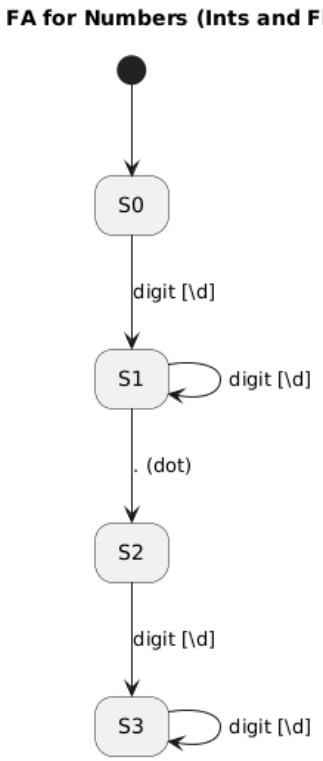
Token Type	Regular Expression	Description
Keywords	"begin_main", "end_main", etc.	The 15 literal keywords (e.g., if_cond, integer). They are matched as-is.
Identifier	[a-zA-Z_][a-zA-Z0-9_]*	Must start with a letter or underscore, followed by letters, numbers, or underscores.
Float	\d+\.\d+	A sequence of one or more digits, a decimal point, and one or more digits.
Integer	\d+	A sequence of one or more digits.
Operator	`":="	"+"
Punctuation	`";"	"("
Whitespace	[\t]+	Matches spaces or tabs. This token is ignored by the scanner.
Newline	\n	Matches the newline character. This is used to increment the line counter.
Error	.	

2. Finite Automata (FA) Diagrams

FA for Identifier



FA for Numbers (Integers and Floats)



3. Explanation of Keyword Choices

Theme: Readability and clarity over brevity.

Core Idea: Make the code read like a set of instructions.

Keyword Groups & Rationale

1. **Program Structure:** begin_main, end_main
   - **Why:** Unlike C++'s { and }, these keywords create a very clear, unmissable "block" that defines where the program's execution starts and stops.
2. **Data Types:** integer, real, text
   - **Why:** These are plain English words.
   - real (borrowed from Pascal) is used instead of float as it's a more intuitive mathematical term.
   - text is used instead of string as it feels more fundamental.
3. **Variable Declaration:** constant
   - **Why:** constant is more descriptive than const. It clearly states the *purpose* of the variable: it is a constant value.
4. **Control Flow:** if_cond, else_cond, repeat_while
   - **Why:** This is a key design choice.
   - if_cond (if condition) and else_cond (else condition) are more explicit than just if and else.
   - repeat_while is clearer than just while. It reads like an action: "Repeat this block *while* this condition is true."
5. **Functions/Procedures:** proc, return_val
   - **Why:** proc is short for "procedure," a classic term for a function that doesn't return a value.
   - return_val is more explicit than just return. It clarifies that an action is happening: a *value* is being *returned*.
6. **Input/Output:** print, read
   - **Why:** These are simple, universally understood verbs. They are much clearer to a beginner than cout or cin.
7. **Boolean Values:** true, false
   - **Why:** These are the standard, non-negotiable keywords for boolean logic.