A3 COMPILADOR

Breno Cesar Mol de Olivera / RA 323213875
Daniel Macedo Silveira / RA: 12725236217
Hiago da Silva Lins Gonçalves / RA: 12525163385
Leonardo de Paula do Prado / RA: 323214126
Luiza Cavalcante Quina de Siqueira / RA: 824118304
Pedro Henrique moura de Lima / RA:12725174579
Rayssa de Lima Ribeiro Assis / RA:13525123217

1. Visão Geral e Escopo

A Linguagem X é uma *Domain-Specific Language* (DSL) imperativa, desenvolvida com foco em demonstrar as etapas de Análise Léxica, Sintática e Semântica utilizando a ferramenta ANTLR 4. Optamos por uma sintaxe familiar ao C/Java, mas com palavras-chave em português para clareza no escopo do projeto. A gramática garante o tratamento de precedência e suporta todas as estruturas de controle obrigatórias.

2. Tipos de Dados e Declaração

A Linguagem X é estaticamente e fortemente tipada. Implementamos os três tipos primitivos exigidos:

Palavra-chave	**Descrição**	**Token (Gramática)**
inteiro	Armazena números inteiros (ex: 10, 0, -5).	TIPO_INT
real	Armazena números de ponto flutuante (decimais).	TIPO_REAL
texto	Armazena sequências de caracteres delimitadas por aspas duplas.	TIPO_TEXTO

Sintaxe da Declaração:

<TIPO> <ID> ;

Exemplo: inteiro idade;

3. Estruturas de Controle

Para delimitar escopo e blocos de comandos, utilizamos a notação padrão em chaves ({ e }).

3.1. Condicional (se...senao)

O *if-else* é mapeado para se e senao. A expressão lógica de condição é obrigatória e deve estar entre parênteses:

se (expressao_logica) {

    comando*

} senao { // Opcional

    comando*

}

3.2. Estruturas de Repetição

Implementamos as duas estruturas de repetição exigidas: enquanto (While) e para (For).

- **enquanto (While):** Ideal para repetições que dependem puramente de uma condição de parada:
- enquanto (expressao_logica) {
- comando*
- }
- **para (For):** Segue o modelo tripartido, facilitando laços controlados por contadores:
- para (atribuicao_inicial; expressao_logica; atribuicao_incremento;) {
- comando*
- }

4. Comandos de I/O e Atribuição

4.1. Atribuição

Utilizamos o operador de atribuição padrão (=).

Sintaxe: ID = expressao;

Na Análise Semântica, é neste ponto que verificamos a **declaração prévia da variável** de destino (diferencial) e, futuramente, a compatibilidade de tipos.

4.2. Entrada e Saída (I/O)

Mapeamos os comandos de I/O para as palavras-chave leia (entrada, equivalente ao scanf) e escreva (saída, equivalente ao printf).

- **Entrada:** leia aceita apenas um identificador de variável como argumento: leia(variavel_destino);

- **Saída:** escreva aceita qualquer expressão válida (incluindo variáveis, literais ou cálculos): escreva(expressao);

5. Precedência de Expressões

A gramática foi desenhada para garantir a precedência dos operadores (Multiplicação e Divisão > Soma e Subtração) através da estruturação das regras expressao e termo.

Prioridade	**Operadores**	**Token**
Alta (1)	* (Multiplicação), / (Divisão)	MULT, DIV
Média (2)	+ (Soma), - (Subtração)	SOMA, SUB
Baixa (3)	==, !=, >, <, >=, <= (Relacionais)	OP_REL

O uso de parênteses () sobrepõe a precedência natural, forçando a avaliação antecipada, como de costume.

6. Considerações Léxicas

No Lexer, garantimos que:

1. **Números Decimais:** O token NUM_DEC reconhece literais com ponto decimal, cumprindo o requisito.
2. **Tokens Ignorados:** A regra WS : [\t\r\n]+ -> skip; ignora todos os espaços em branco, *tabs* e quebras de linha. Isso assegura que o Lexer forneça ao Parser apenas os tokens relevantes, simplificando a análise sintática.
3. **Identificadores:** ID é definido para aceitar letras, seguido por letras ou números, evitando conflito com palavras-chave reservadas.

7. Comprovação da Funcionalidade

Teste Semântico: Para ver erro semântico no código

```
PS C:\Users\hiago\Downloads\CompiladorA3> python main.py teste_sucesso_decl.lx
--- Iniciando Compilador ---
Análise Sintática OK. Iniciando Análise Semântica...

--- COMPILAÇÃO FALHOU (Erros Semânticos) ---
ERRO SEMÂNTICO na linha 3: Variável 'b' utilizada sem ser declarada.
```

Teste Sintático: Para ver erro sintático no código

```
PS C:\Users\hiago\Downloads\CompiladorA3> python main.py teste_erro_sintatico.lx
--- Iniciando Compilador ---

--- COMPILAÇÃO FALHOU (Erros Sintáticos) ---
ERRO SINTÁTICO na linha 2, coluna 0: Token inesperado 'contador'
```

Teste Sucesso do Código: Para ver se o código está funcionando