Explicação: Gramática Livre de Contexto - LL(1)

Este documento explica a estrutura gramatical definida no arquivo, projetada para um analisador sintático do tipo LL(1). O arquivo atua como um 'Livro de Regras' que ensina o compilador a validar o código fonte.

1. Definição Formal

O arquivo define os componentes básicos da linguagem:
- S (Símbolo Inicial): A análise começa sempre pela regra 'Programa'.
- T (Terminais / Tokens): Palavras que o compilador reconhece (ex: if, while, int, {, }).
- V (Não-Terminais): Conceitos abstratos (ex: IfStmt, Declaracao).

2. Estrutura Geral

A regra <Programa> define o esqueleto obrigatório:
- Todo código deve iniciar com 'int main()'.
- O corpo do código deve estar delimitado por chaves '{ ... }'.

3. Comandos

Dentro do bloco principal, o programador pode utilizar:
- Declarações: int x; ou float y;
- Atribuições: x = 10;
- Controle de Fluxo: Estruturas if, while, do-while e for.

4. Lógica Booleana

A gramática divide a lógica em três níveis para garantir a ordem correta:
1. <CondicaoL> (OU / ||): Menor prioridade.
2. <CondTermoL> (E / &&): Prioridade média.
3. <CondFator> (NÃO / !): Maior prioridade.
Isso garante que 'A || B && C' seja interpretado como 'A || (B && C)'.

5. Expressões Aritméticas

A matemática segue a hierarquia de precedência padrão:
1. <ExpL> (Soma/Subtração): Menor prioridade.
2. <TermoL> (Multiplicação/Divisão): Maior prioridade.
3. <Fator>: Unidade básica (números, variáveis).
Isso garante que '2 + 3 * 4' resulte em 14 e não 20.

6. O Conceito LL(1)

Explicação: Gramática Livre de Contexto - LL(1)

A gramática é classificada como LL(1), o que significa:

- Left-to-right: Lê da esquerda para a direita.
- (1): Precisa olhar apenas 1 token à frente para decidir a regra.

A gramática usa estruturas auxiliares (como ExpL) para eliminar a recursão à esquerda.