As Três Fases da Análise

1. Análise Léxica

Transforma o código-fonte em tokens (unidades básicas como palavras reservadas, identificadores, operadores e símbolos). Essa fase utiliza um autômato finito determinístico (AFD) para reconhecer os padrões válidos, gerando uma tabela de símbolos contendo o tipo de token, sua posição no código (linha e coluna), e seu valor (ex: nome de variável, operador, etc.).
Assim verificando se todas as palavras e símbolos do código sejam reconhecidos e válidos para a linguagem.

Exemplo:

Código: int x;
Tokens gerados:
1. int → palavra reservada
2. x → variável
3. ; → Ponto e virgula

2. Análise Sintática

Verifica a estrutura e a ordem dos tokens gerados na análise léxica, garantindo que o código siga as regras gramaticais da linguagem de programação. Gera uma árvore sintática (ou árvore de derivação) representando a estrutura hierárquica das instruções, é esperado que os comandos tenham sentido estrutural, ou seja, estejam na ordem correta (por exemplo, tipo → nome da variavel → ponto e vírgula).

Exemplo:

Para o trecho 'int x;', a análise sintática confirma que a sequência 'tipo → identificador → ;' está correta. Mas em 'int ; x' seria detectado um erro de sintaxe.

3. Análise Semântica

Verifica se o código faz sentido lógico. Confere se os tipos são compatíveis, se variáveis foram declaradas antes do uso, e se os retornos de funções são coerentes. Caso verifique que contenha alguma variável não declarada gera Mensagens de erro ou avisos sobre inconsistências semânticas (como tipos incompatíveis). Espera-se que o código seja coerente e válido logicamente, sem erros de tipo ou uso incorreto de variáveis e funções.

Exemplo:

Se tivermos:
int x;
x = "texto";
A análise semântica acusará erro, pois a variável x é do tipo inteiro e está recebendo uma string.