Aluno: Pedro Balen

O processo de análise em um compilador é onde acontece a tradução do código-fonte em um programa executável, ou seja tem grande importância na computação, e podemos dividir essa análise em três etapas sequenciais: análise léxica, sintática e a semântica. Cada fase constrói sobre o resultado da fase anterior.

Análise léxica: primeira etapa do processo. Sua função é ler o código fonte original, caractere por caractere. Conforme a análise vai sendo executada, esses caracteres são agrupados em unidades chamadas tokens, que representam algo da linguagem, como palavra chave (tipo de variável), identificadores (nome da variável), operadores e números. Por exemplo, no código float media = 9.5; o analisador léxico não vê a linha inteira, mas identifica float como uma palavra chave, media como identificador, = como um operador de atribuição, 9.5 como número e ; como um delimitador. No final, ao invés de termos apenas um arquivo de texto, temos uma sequência de tokens, essa é a saída da análise léxica, geralmente acompanhada de uma tabela de símbolos, que indica o que representa cada token.

Essa sequência de tokens é então passada como entrada para a segunda etapa, a análise sintática.

Análise sintática: aqui acontece a verificação dos tokens na ordem que aparecem. O analisador sintático agrupa os tokens de acordo com as regras gramaticais da linguagem. Por exemplo, garante que um comando if seja seguido por uma condição entre parênteses ou que, em uma atribuição, exista um identificador válido à esquerda do operador. Um erro será detectado se o código for 9.5 = media;. A saída gerada por esta fase é uma estrutura em formato de árvore, que representa a hierarquia e a estrutura do programa.

Análise semântica: Vai utilizar a árvore gerada pela fase anterior para analisar o “significado” ou a lógica do código. Mesmo que a sintaxe esteja correta, o código pode não fazer sentido lógico. O analisador semântico percorre toda árvore e realiza verificações, como a verificação de tipos, como por exemplo não permitir a soma de um texto com um booleano, verificação de escopo, que garante que variáveis e funções foram declaradas antes de serem usadas e a verificação de duplicidade, que verifica se há declarações duplicadas. Por exemplo, se o código for int valor = “teste”; a sintaxe estaria correta, mas a análise semântica detectaria erro de tipo.

No final dessa etapa, o esperado é ter a árvore sintática validada, confirmando que o programa é sintático e semanticamente correto.