UNIVERSIDADE SÃO JUDAS TADEU
MOOCA

PESQUISA – CASE 3 – ANALISADOR LÉXICO

LUISA DA CRUZ COELHO – RA: 82312510
MURILO BRIGATTI DA SILVA – RA: 823158112
MURILO FERNANDES CORSO – RA: 823129492
MURILO FREDERICO GARCEZ – RA: 823153149

SÃO PAULO

05/06/2024

1. ANALISADOR LEXICO

Analisadores Léxicos são componentes essenciais de compiladores e interpretadores em linguagens de programação, que utilizam autômatos finitos. Eles são responsáveis por converter sequências de caracteres em tokens, que são os blocos básicos de construção para a análise sintática e semântica.

1.1 - COMPILAÇÃO X INTERPRETAÇÃO

O compilador é o programa responsável por realizar a tradução. O processo de compilação envolve a conversão de um código escrito em uma linguagem de programação de alto nível para a linguagem da máquina, ao ler o código fonte é feito a divisão em tokens significativos. Ele possui dois módulos principais: o front-end e o back-end. No front-end, o compilador verifica erros de digitação, sintaxe e tipos de dados no código-fonte, enquanto no back-end, o compilador aloca espaço de memória e gera um arquivo de código-objeto, que é o código traduzido para a linguagem desejada.

Em algumas implementações de compiladores, o analisador léxico pode realizar otimizações lexicais para melhorar o desempenho do processo de compilação.

Já na interpretação, não há geração de arquivos executáveis. O código-fonte é traduzido diretamente pela máquina de destino durante a execução, não há um passo de compilação prévio, ao ler o código fonte conforme necessário ele produz os tokens para a interpretação imediata.

Exemplos de linguagens interpretadas incluem: Python, Ruby, JavaScript e PHP.

Devido a isso a interpretação geralmente oferece mais flexibilidade do que a compilação, pois permite a execução de código diretamente a partir do código fonte, facilitando a depuração e a prototipagem rápida, além de garantir que o código fonte seja interpretado corretamente em tokens significativos.

1.2 - ANÁLISE LÉXICA

A análise léxica, também conhecida como scanner ou leitura, é a primeira fase de um processo de compilação ou interpretação de um programa. Sua função é ler o programa fonte, caractere a caractere, agrupar os caracteres em lexemas e produzir uma sequência de símbolos léxicos conhecidos como tokens, neles incluem palavras-chave, identificadores, operadores, números e símbolos especiais. Esses tokens são enviados para serem processados pela análise sintática, que é a próxima fase do processo de compilação.

Ao realizar a Análise eles percorrem o código fonte caractere por caractere e combinam esses caracteres em tokens de acordo com as regras de sintaxe da linguagem, durante esse processo, os analisadores léxicos também lidam com espaços em branco, comentários e outros caracteres que não são relevantes para a estrutura da linguagem. A principal tarefa é reconhecer os tokens do código fonte, como palavras-chave, identificadores e símbolos especiais, além de descarte de Informações Irrelevantes como espaços em branco e comentários e identificar erros léxicos que compõem os tokens inválidos ou malformados.

1.3 – ANALISADORES LÉXICOS

Os analisadores léxicos desempenham um papel vital no processo de compilação e interpretação de linguagens de programação. Eles são responsáveis por examinar o código-fonte e identificar elementos básicos como identificadores, palavras-chave e símbolos, convertendo-os em unidades significativas chamadas tags. Para realizar essa tarefa, os analisadores lexicais utilizam uma variedade de mecanismos, incluindo autômatos finitos determinísticos (DFA) que representam padrões de entrada com eficiência, expressões regulares que descrevem padrões de texto e autômatos de estados finitos não determinísticos (AFN) que fornecem flexibilidade ao especificar padrões.

Além disso, em linguagens mais complexas, como aquelas que suportam recursão e estruturas aninhadas, pode ser necessário recorrer a máquinas de estados finitos empilhadas. A escolha do mecanismo adequado depende das características da linguagem e dos requisitos de desempenho do sistema em questão.

A implementação de um analisador léxico requer uma descrição dos lexemas (sequência de caracteres reconhecidos por um padrão), permitindo que o código identifique a ocorrência de cada lexema e associe-o ao padrão correto. Assim, o analisador léxico desempenha um papel fundamental na análise de linguagens de programação e outras sequências de caracteres.

1.4 TABELA DE SÍMBOLOS

Também conhecida como “tokens aceitos”, a tabela de símbolos funciona como uma base de consulta para o compilador saber se aquele identificador (palavra digitada) é uma palavra reservada, função, classe entre outros. A tabela de símbolo é utilizada durante todo o processo de análise léxica, sintática e as outras fases do processo de compilação e identificação.

As principais funções da Tabela de Símbolos:

1 – Armazenamento: Conseguimos armazenar, tipos das variáveis, valor das variáveis, funções, classes, escopos etc.

2 – Armazena o Escopo da variável: Identifica se aquele valor é em escopo global ou local (atuando dentro da função).

3 – Armazena tipo de dados: Durante a compilação a tabela de Símbolos armazena o tipo dos dados, por exemplo, int, float, Strings.

Exemplo de como a tabela de Símbolo funciona:

Código –

```
def mensg():

    mensagem = 'Hello, World'

    print(mensagem)

mensg()
```

Tabela de Símbolos:

Nome	Tipo	Escopo	Valor
mensg	funcao	Global	-
mensagem	Str	Dentro de mensg	“Hello, World”
print	Função		-