Projeto compiladores

Grupo Rai

João Victor Varella Siqueira - 820142349

Raí Joia Miquilino Valencio - 82318841

Matheus Henrique Oliveira Chuang - 823165173

Henryk Bagdanovicius Roza - 823135401

Bruno Godoy Dias - 82311358

Introdução

Durante o semestre foram lançados 3 casos relacionados ao tema de compiladores, cada caso teve seu respectivo problema e nível de complexidade.

Case 3 Analisador Léxico

- Analisador léxico.
- O analisador deve ser capaz de fazer a análise léxica de sequências de comandos (pedaços de códigos) e mostrar tokens, tabela símbolos, eliminação de caracteres em branca, comentários.
- Ao ser executado, deve-se exibir o código fonte de entrada e a análise léxica

Análise Léxica

- A análise léxica também conhecida como scanner ou leitura é a primeira fase de um processo de compilação e sua função é fazer a leitura do programa fonte, caractere a caractere, agrupar os caracteres em lexemas e produzir uma sequência de símbolos léxicos conhecidos como tokens.
- A sequência de tokens é enviada para ser processada pela análise sintática que é a próxima fase do processo de compilação .
- O analisador léxico deve interagir com a tabela de símbolos inserindo informações de alguns tokens, como por exemplo os identificadores.

Visão geral

- A análise léxica pode ser dividida em duas etapas, a primeira chamada de escondimento que é uma simples varredura removendo comentários e espaços em branco, e a segunda etapa, a análise léxica propriamente dita onde o texto é quebrado em lexemas.

- Podemos definir três termos relacionados a implementação de um analisador léxico:

  Padrão: é a forma que os lexemas de uma cadeia de caracteres podem assumir
  Lexema: é uma sequência de caracteres reconhecidos por um padrão.
  Token: é um par constituído de um nome e um valor de atributo esse último opcional.

Tabela de símbolos

	A	B	C
1	Código	Tipo	Função
2	abstract	Palavra-chave	Indica que uma classe ou método é abstrato, isto é, não possui uma implementação concreta.
3	assert	Palavra-chave	Usado para testar expressões booleanas durante o desenvolvimento.
4	boolean	Tipo de Dados	Representa um tipo de dado que pode ter apenas dois valores: verdadeiro (true) ou falso (false).
5	break	Palavra-chave	Utilizado para sair de loops ou switches.
6	byte	Tipo de Dados	Representa um tipo de dado inteiro de 8 bits.
7	case	Palavra-chave	Define uma condição em uma instrução switch.
8	catch	Palavra-chave	Captura exceções em blocos try-catch.
9	char	Tipo de Dados	Representa um único caractere Unicode.
10	class	Palavra-chave	Define uma classe em Java.
11	const	Palavra-chave	Não utilizada em Java desde o Java 1.4.
12	continue	Palavra-chave	Pula a iteração atual em um loop e continua com a próxima iteração.
13	default	Palavra-chave	Define o bloco padrão em uma instrução switch.
14	do	Palavra-chave	Inicia um loop do-while.
15	double	Tipo de Dados	Representa um número de ponto flutuante de precisão dupla.
16	else	Palavra-chave	Define o bloco de código a ser executado se a condição de um if não for verdadeira.
17	enum	Palavra-chave	Define um tipo de dados enumeração.
18	extends	Palavra-chave	Indica que uma classe herda de outra classe.
19	final	Palavra-chave	Indica que uma variável, método ou classe não pode ser alterada ou estendida.
20	finally	Palavra-chave	Define um bloco de código a ser executado após um bloco try-catch, independentemente do resultado.
21	float	Tipo de Dados	Representa um número de ponto flutuante de precisão simples.
22	for	Palavra-chave	Inicia um loop for.
23	goto	Palavra-chave	Não é utilizado em Java.
24	if	Palavra-chave	Indica uma instrução condicional.
25	implements	Palavra-chave	Indica que uma classe implementa uma ou mais interfaces.
26	import	Palavra-chave	Importa pacotes ou classes para usar em um programa Java.
27	instanceof	Palavra-chave	Verifica se um objeto é uma instância de uma determinada classe ou interface.
28	int	Tipo de Dados	Representa um número inteiro de 32 bits.
29	interface	Palavra-chave	Define uma interface em Java.
30	long	Tipo de Dados	Representa um número inteiro de 64 bits.
31	native	Palavra-chave	Indica que um método é implementado em código nativo, geralmente escrito em outra linguagem.
32	new	Palavra-chave	Cria uma nova instância de uma classe.
33	package	Palavra-chave	Define um pacote em Java.
34	private	Modificador	Indica que um membro é acessível apenas dentro da própria classe.
35	protected	Modificador	Indica que um membro é acessível dentro da própria classe e subclasses.
36	public	Modificador	Indica que um membro é acessível em qualquer lugar.
37	return	Palavra-chave	Retorna um valor de um método.
38	short	Tipo de Dados	Representa um número inteiro de 16 bits.
39	static	Modificador	Indica que um membro pertence à classe em vez de instâncias individuais.
40	strictfp	Palavra-chave	Restringe o comportamento de ponto flutuante para garantir a portabilidade de ponto flutuante.
41	super	Palavra-chave	Referencia a superclasse imediata de um objeto.
42	switch	Palavra-chave	Seleciona uma das várias opções de acordo com o valor de uma expressão.
43	synchronized	Modificador	Indica que um método só pode ser acessado por uma thread de cada vez.
44	this	Palavra-chave	Referencia o objeto atual.
45	throw	Palavra-chave	Lança uma exceção.
46	throws	Palavra-chave	Declara exceções que podem ser lançadas por um método.
47	transient	Modificador	Indica que um membro não deve ser serializado durante a serialização de um objeto.
48	try	Palavra-chave	Define um bloco de código que pode lançar exceções.
49	void	Tipo de Dados	Indica que um método não retorna nenhum valor.
50	volatile	Modificador	Indica que uma variável pode ser modificada por diferentes threads.
51	while	Palavra-chave	Inicia um loop while.
52	System	Classe	Fornece acesso ao ambiente de tempo de execução do sistema.
53	out	Objeto	Um objeto de saída padrão.
54	println	Método	Imprime uma linha no console.
55	print	Método	Imprime uma mensagem no console.
56	printf	Método	Formata e imprime uma mensagem no console.
57	scanf	Método	Não é uma função padrão em Java.
58	next	Método	Lê a próxima entrada de dados.
59	nextInt	Método	Lê o próximo valor inteiro da entrada.
60	nextDouble	Método	Lê o próximo valor de ponto flutuante da entrada.
61	nextLine	Método	Lê a próxima linha de texto da entrada.

Tokens

- Os tokens podem ser divididos em dois grupos:

Tokens simples: são tokens que não têm valor associado pois a classe do token já a descreve. Exemplo: palavras reservadas, operadores, delimitadores: `<if,>`, `<else>`, `<+,>`

Tokens com argumento: são tokens que têm valor associado e correspondem a elementos da linguagem definidos pelo programador. Exemplo: identificadores, constantes numéricas - `<id, 3>`, `<número, 10>`, `<literal, Olá Mundo>` .

Erros léxicos

- A análise léxica é muito prematura para identificar alguns erros de compilação, veja o exemplo abaixo:
- fi (a == “123”) ...

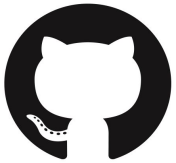
- O analisador léxico não consegue identificar o erro da instrução listada acima, pois ele não consegue identificar que em determinada posição deve ser declarado a palavra reservada if ao invés de fi.
- Essa verificação somente é possível ser feita na análise sintática.

Expressões regulares

- Expressões regulares ou **regex** são uma forma simples e flexível de identificar cadeias de caracteres em palavras.
- As expressões regulares não validam dados, apenas verificam se um texto está em uma determinado padrão.

- Elas são escritas em uma linguagem formal que pode ser interpretada por um processador de expressão regular que examina o texto e identifica partes que casam com a especificação dada, são muito utilizadas para validar entradas de dados, fazer buscas, e extrair informações de textos.

GitHub



HTML CSS JS

Projeto: https://1zod.github.io/Maquina-de-Doces