UNIVERSIDADE FEDERAL DO PARANÁ

DEPARTAMENTO DE INFORMÁTICA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CI211 - CONSTRUÇÃO DE COMPILADORES

Trabalho 1 - Analisador Léxico

Nome: Henrique Colodetti Escanferla - GRR20135427

DEFINIÇÕES REGULARES DAS REGRAS FORMADORAS DE TOKENS

Alfabeto aceito pelo analisador léxico:

{ [a-zA-Z0-9] + - \* / & | \ = > < ! , ; ( ) { } [ ] " . \s \n \t \x00 } .

# delimita as regras. \ é caractere de escape, \s é espaço, \n é "pula linha", \t é a tecla tab, \x00 é o caractere nulo, \Q delimita o início de uma seqüência de caracteres literais e \E delimita o fim de uma seqüência de caracteres literais.

letra => [a-zA-Z] # digito => [0-9] # dígitos => digito+ # STR => ["][^"]\*["] #

CH => ['][^']['] # NUM => dígitos.(dígitos | €) # ID => letra.(letra | digito)\* #

TYPE => [i][n][t] | [c][h][a][r] | [f][l][o][a][t] | [s][t][r][i][n][g] # if => [i][f] #

const => [c][o][n][s][t] # then => [t][h][e][n] # else => [e][l][s][e] #

while => [w][h][i][l][e] # do => [d][o] #

OP\_ART => [+\-\*/] # = => [=] # OP\_REL => ([><] ([=] | €)) | ([=] | [!]) [=] #

INC => [+][+] | [-][-] # OP\_LOG => [&][&] | [|][|] | [!] # SEMI\_COLON => [;] #

COMMA => [,] # LPARENT => [(] # RPARENT => [)] # LBRACK => [\[] #

RBRACK => [\]] # LBRACE => [{] # RBRACE => [}] #

comment => [/][/][^\n\x00]\* |

[/][\*] qualquer conjunto de caracteres que não forme o padrão "\*/" [\*][/]

Obs.: \n \s \t e \x00 pertencem ao alfabeto mas são jogados fora exceto dentro de STR e CH. Estes tokens aceitam qualquer caractere dentro de seus delimitadores.

IMPLEMENTAÇÃO DO ANALISADOR LÉXICO

Java foi a linguagem escolhida para que o programa seja orientado a objetos na intenção de implementar um autômato, seus estados e suas transições. Além disso, é fácil adicionar novos tokens com novos estados e transições no código com poucas linhas novas. Implementar este código aumentou a intimidade com Java que é uma plataforma de programação tão requerida profissionalmente.

A classe LexicAnaliser contém o método main. Ela envia comandos para métodos das classes Automaton e State afim de gerenciar toda a execução do código. O main consiste de alguns comandos de inicialização para as outras classes e um "for loop" que executa o método que lê um token do arquivo de entrada "entry.code". Tal loop termina quando certos estados são executados.

A classe abstrata Automaton guarda uma lista dos objetos da classe State e é responsável por instanciá-los, criar transições entre estes e define o método para encontrar um token de "entry.code".

A classe State guarda uma porção de variáveis instanciáveis e não instanciáveis. Algumas poderiam estar na classe Automaton, principalmente as não instanciáveis mas já que a execução do estado necessita acessá-las constantemente foram colocadas em State. Os estados são tipados pois há diferentes padrões de execução de um estado e serve para decidir o que fazer. Esta classe contém um método para a execução de qualquer estado do autômato e por esta razão contém inúmeros outros métodos que dependem da tipagem do estado:

1) Ler caractere de "entry.code" e **incluir na leitura do token**, se não for ignoreChar, contando '\n' para imprimir a linha atual em caso de erro léxico. Estados que reconhecem comentários, delimitadores de char/string e o estado inicial que lê espaços, '\n' e '\t' ignora-os.

2) Executar transição de acordo com o caractere lido. O que se faz após leitura depende do novo estado após a transição.

3) Enviar várias variáveis para que a classe Automaton controle sua própria execução 4) Decidir se o token é palavra reservada/tipo/identificador

5) Popular tabela de identificadores 6) Imprimir tabela de identificadores

7) Escrever no arquivo de saida "saida.tokens" o token encontrado quando for final, imprimir mensagem de erro caso erro e termina a execução do código quando EOS (End of String => fim do arquivo "entry.code"). Tal método imprime '\n' entre os tokens para delimitá-los melhor além de '<' e '>' pois '>' pode significar o operador relacional "maior que" ou o delimitador final de um token.