Analisa Léxica da Linguagem T++

Humberto Moreira Gonçalves

August 31, 2016

1 Introdução

Este documento descreve como foi implementado o analisador léxico da linguagem T++ proposta na disciplina de Compiladores.

2 Linguagem T++

Esse projeto consiste na implementação da análise léxica de um compilador para a linguagem de programação T++. Será demonstrado as palavras reservadas e expressões regulares da análise léxica. Um compilador é um programa que possui a capacidade de converter a implementação de uma determinada linguagem para código de máquina, muito compiladores geram código em Assembly, dado que é uma linguagem mais próxima do binário utilizado no código de máquina.

T++ é uma linguagem de programação simples que foi construída para intuito de aprendizado de compiladores e para realizações de operações simples de matemática, possuindo operadores matemáticos, como adição, subtração, multiplicação e divisão, e operadores lógicos, como maior, menor, etc. Fornecendo número inteiros e de ponto flutuante.

3 Análise Léxica

Essa análise se baseia em um conjunto de tokens de palavras reservadas e símbolos. Também possuindo número inteiro ou ponto flutuante

Com base na tabela 1, foi criado o automâto para cada token, da seguinte forma:

Desta forma, a implementação dos tokens na tabela 1, ficou como a figura 2:

Para cada token é necessário a utilização de de expressões regulares, para que possam ser reconhecidos na passada de leitura do código, como na figura 4:

Palavras reservadas	Símbolos	Outros
se	+ soma	número
então	- subtração	identificador
senão	* multiplicação	comentário
fim	/ divisão	nova linha
repita	= igualdade	
flutuante	, vírgula	
retorna	:= atribuição	
até	< menor	
leia	> maior	
escreve	<= menor-igual	
inteiro	>= maior-igual	
	(abre-par	
	) fecha-par	
	: dois-pontos	

Table 1: Classes de tokens e suas definições

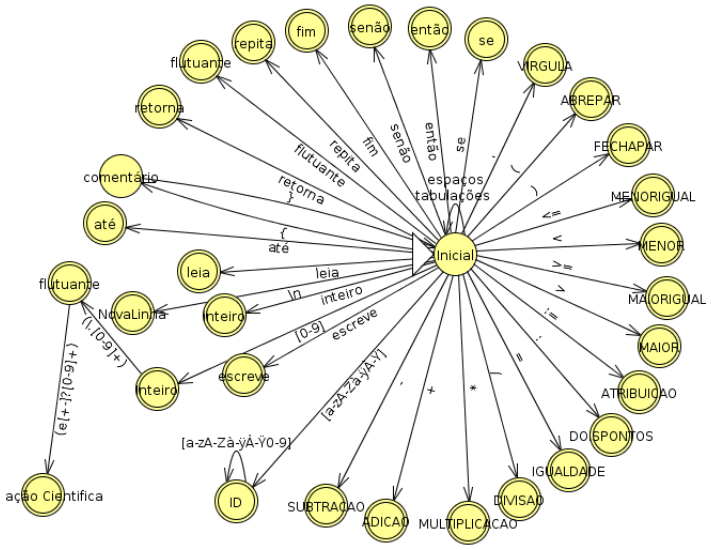

Figure 1: Automato dos tokens

```
# Dicionario reservadas
keywords = {
    u'se': 'SE',
    u'então': 'ENTAO',
    u'senão': 'SENAO',
    u'fim': 'FIM',
    u'repita': 'REPITA',
    u'flutuante': 'FLUTUANTE',
    u'retorna': 'RETORNA',
    u'até': 'ATE',
    u'leia': 'LEIA',
    u'escreve': 'ESCREVE',
    u'inteiro': 'INTEIRO',
    u'principal': 'PRINCIPAL',
    }
```

Figure 2: Implementação das palavras reservadas

```
# Lista de Tokens
tokens = ['ADICAO', 'SUBTRACAO', 'MULTIPLICACAO', 'DIVISAO', 'IGUALDADE',
          'VIRGULA', 'ATRIBUICAO', 'MENOR', 'MAIOR', 'MENORIGUAL', 'MAIORIGUAL',
           'ABREPAR', 'FECHAPAR', 'DOISPONTOS','NUMERO', 'ID'] + list(keywords.values())
```

Figure 3: Implementação dos tokens

```
# Expressões simples
t_ADICAO = r'\+'
t_SUBTRACAO = r'\-'
t_MULTIPLICACAO = r'\*'
t_DIVISAO = r'/'
t_IGUALDADE = r'='
t_VIRGULA = r','
t_ATRIBUICAO = r':='
t_MENOR  = r'<'
t_MAIOR = r'>'
t_MENORIGUAL = r'<='
t_MAIORIGUAL = r'>='
t_ABREPAR = r'\('
t_FECHAPAR = r'\)'
t_DOISPONTOS = r':'
t_NUMERO = r'[0-9]+(\.[0-9]+)?'


def t_ID(self, t):
    r'[a-zA-Zà-ú][a-zA-Zà-úÀ-Ú0-9]*'
    t.type = self.keywords.get(t.value, 'ID')
    return t

def t_COMMENT(self, t):
    r'\{[^\{^\}]*\}'
    pass

def t_NEWLINE(self, t):
    r'\n+'
    t.lexer.lineno += len(t.value)
#Ignora tabs e espacos
t_ignore = ' \t'
```

Figure 4: Automato dos tokens