Disciplina: Fundamentos de Compiladores

Data: 31/11/2021

Alunos: Filipe Silva, Jéssica Cardoso, Reinilson Bispo e Vitor Manoel.

Trabalho (II Unidade)

Descrição: Projeto do Analisador Léxico/Sintático para uma Calculadora com Funções Avançadas.

Tabela de Tokens/Lexemas

Token	Padrão	Lexema	Descrição
<number, >	\d* ([.][\d]*) ?	1, 1.51, 21.9	Números inteiro e flutuante
<l1o, >	([+] ,[-])	+,-	Operadores de grau de precedência 1
<l2o, >	([*], [/], [%])	*,/,%	Operadores de grau de precedência 2
<l3o, >	[^]	^	Operadores de grau de precedência 3
<, >	[,]	,	vírgula
<cos, >	[c,C][o,O][s,S]	Cos, cos, COS, ...	Palavra reservada
<sin, >	[s,S][i,I][n,N]	Sin, sin, SIN, ...	Palavra reservada
<round, >	[r,R][o,O][u,U][n,N][d,D]	Round, round, ROUND, ...	Palavra reservada
<abs, >	[a,A][b,B][s,S]	Abs, abs, ABS, ...	Palavra reservada
<sqrt, >	[s,S][q,Q][r,R][t,T]	Sqrt, sqrt, SQRT, ...	Palavra reservada
<tan, >	[t,T][a,A][n,N]	Tan, tan, TAN, ...	Palavra reservada
<log, >	[l,L][o,O][g,G]	Log, log, LOG ...	Palavra reservada
<pi, >	[p,P][i,I]	pi,Pi,PI	Palavra reservada
<e, >	[e,E]	e,E	Palavra reservada

<(, >	[( ]	(	Delimitador Início
<), >	[ )]	)	Delimitador Fim

Descrição das Funções:
A calculadora avançada fornece um conjunto de 6 funções internas:
abs: Calcula o valor absoluto de um número ou resultado de uma expressão matemática. Recebe como parâmetro um número ou expressão matemática.
round: Calcula o valor inteiro aproximado de um número ou resultado de uma expressão matemática. Recebe como parâmetro um número ou expressão matemática.
sin: Calcula o seno de um ângulo em graus. Recebe como parâmetro um número ou expressão matemática.
cos: Calcula o cosseno de um ângulo em graus. Recebe como parâmetro um número ou expressão matemática.
tan: Calcula a tangente de um ângulo em graus. Recebe como parâmetro um número ou expressão matemática.
sqrt: Calcula a raiz quadrada de um número ou resultado de uma expressão matemática. Recebe como parâmetro um número ou expressão matemática.
log: Calcula o logaritmo de um número em determinada base, Recebe como parâmetros um número ou expressão matemática como base e um número ou expressão matemática como logaritmando.

Gramática Livre de Contexto (GLC):

G = {EXP, TERM, OPER1, OPER2, OPER3, NUM}, {<number>, <l1o>, <l2o>, <l3o>, <,>, <round>, <sin>, <tan>, <sqrt>, <abs>, <log>, <pi>, <e>, <(>, <)>, ƛ}, P, EXP

P:

EXP -> NUM TERM | <(> EXP <)> TERM | <l1o> EXP
TERM -> OPER1 EXP | ƛ
OPER1 -> <l1o> | OPER2
OPER2 -> <l2o> | OPER3
OPER3 -> <l3o>
NUM -> <number> | <pi> | <e> | FUNC
FUNC -> FUNC1 <(> EXP <)> | FUNC2 <(> EXP <,> EXP <)>
FUNC1 -> <cos> | <round> | <sin> | <tan> | <sqrt> | <abs>
FUNC2 -> <log>

Conjuntos First e Follow:

	FIRST	FOLLOW
EXP	<number>,<pi>,<e>,<cos>,<round>, <sin>,<tan>,<sqrt>,<abs>,<log>,<(>,<l 1o>	<,>,<)>,$
TERM	<l1o>,<l2o>,<l3o>,<ƛ>	<,>,<)>,$
OPER1	<l1o>,<l2o>,<l3o>	<number>,<pi>,<e>,<round>,<cos > <sin>,<tan>,<sqrt>,<abs>,<log>,<(>,<l1o>
OPER2	<l2o>,<l3o>	<number>,<pi>,<e>,<round>, <cos><sin>,<tan>,<sqrt>,<abs>,<l og>,<(>,<l1o>
OPER3	<l3o>	<number>,<pi>,<e>,<round>, <cos>,<sin>,<tan>,<sqrt>,<abs>,<l og>,<(>,<l1o>
NUM	<number>,<pi>,<e>,<round>, <sin>,<tan>,<sqrt>,<abs>, <log>	<,>,<l1o>,<l2o>,<l3o>,),$
FUNC	<cos>,<round>, <sin>,<tan>,<sqrt>,<abs>,<log>	<,>,<l1o>,<l2o>,<l3o>,),$
FUNC1	<cos>,<round>, <sin>,<tan>,<sqrt>,<abs>	<(>
FUNC2	<log>	<(>

Parser Table Parte 1

	<log>	**<abs>**	**<sqrt>**	**<tan>**	**<sin>**	**<round>**	**<cos>**	**<)>**
EXP	EXP → NUM TERM	EXP → NUM TERM	EXP → NUM TERM	EXP → NUM TERM	EXP → NUM TERM	EXP → NUM TERM	EXP → NUM TERM	
TERM								TERM → ƛ
OPER1								
OPER2								
OPER3								
NUM	NUM → FUNC	NUM → FUNC	NUM → FUNC	NUM → FUNC	NUM → FUNC	NUM → FUNC	NUM → FUNC	
FUNC	FUNC → FUNC2 <(> EXP <,> EXP <)>	FUNC → FUNC1 <(> EXP <)>	FUNC → FUNC1 <(> EXP <)>	FUNC → FUNC1 <(> EXP <)>	FUNC → FUNC1 <(> EXP <)>	FUNC → FUNC1 <(> EXP <)>	FUNC → FUNC1 <(> EXP <)>	
FUNC1		FUNC1 → <abs>	FUNC1 → <sqrt>	FUNC1 → <tan>	FUNC1 → <sin>	FUNC1 → <round>	FUNC1 → <cos>	
FUNC2	FUNC2 → <log>							

Parser Table Parte 2

	<,>	**<(>**	**<e>**	**<pi>**	**<number>**	**<l3o>**	**<l2o>**	**<l1o>**	**$**
EXP		EXP → <(> EXP <)> TERM	EXP → NUM TERM	EXP → NUM TERM	EXP → NUM TERM			EXP → <l1o> EXP	
TERM	TERM → ǎ					TERM → OPER1 EXP	TERM → OPER1 EXP	TERM → OPER1 EXP	TERM → ǎ
OPER1						OPER1 → OPER2	OPER1 → OPER2	OPER1 → <l1o>	
OPER2						OPER2 → OPER3	OPER2 → <l2o>		
OPER3						OPER3 → <l3o>			
NUM			NUM → <e>	NUM → <pi>	NUM → <number>				
FUNC									
FUNC1									
FUNC2									