Seminário Final
Compiladores 2

Grupo: 13

Componentes:
Alexandre Lara , 587117
Lucas Callegari , 551996
Alessandra Camargo , 552038
Tiago Avellar , 551910

Apresentação Geral da Linguagem

- **Nome**
  - Linguagem Matemática
- **O que é ?**
  - um projeto de linguagem de programação desenvolvida com propósito de trabalhar com integração e derivação de funções matemáticas na forma de equações diferenciais ordinárias
- **Objetivo**
  - construir uma ferramenta computacional capaz de oferecer recursos para definição de funções matemáticas e para definição e cálculo de integrações numéricas, além de ter capacidade de derivação de funções.
- **Potencial**
  - resolver qualquer problema real que seja modelado por uma função e resolvido através da integração ou derivação de equações/funções.

Esboço da gramática

```
grammar matematica;
programa
 : bloco EOF
 ;
bloco
 : (declaracao)* (retorno)?
 ;
retorno returns [String tipo]
 : 'retorne' expressao { $tipo = "expressao"; }
 | 'retorne' String    { $tipo = "string";    }
 ;
declaracao returns [String tipoDecl]
 : atribuicao { $tipoDecl = "atribuicao"; }
 | funcao     { $tipoDecl = "funcao";     }
 | print      { $tipoDecl = "print";      }
 ;
```

```
print returns [String tipoPrint]
 : 'print' '('(expressao     { $tipoPrint = "expressao";     }
             | String        { $tipoPrint = "string";        }) ')';
atribuicao
 : Identificador  '=' expressao
 ;
funcao returns [String tipo]
 : (relacao)? identificadorF '=' seno       { $tipo = "seno";     }
 | (relacao)? identificadorF '=' cosseno    { $tipo = "cosseno";  }
 | (relacao)? identificadorF '=' polinomio  { $tipo = "polinomio";}
 | (relacao)? identificadorF '=' expressao  { $tipo = "derivada"; }
 | identificadorF
 ;
```

Esboço da gramática

```
expressao returns [ String tipo ]
 : '-' expressao           { $tipo = "unario";         }
 | '+' expressao           { $tipo = "unarioSoma";     }
 | integral                { $tipo = "integral";       }
 | derivada                { $tipo = "derivada";       }
 | expressao '^' expressao { $tipo = "potencia";       }
 | expressao '*' expressao { $tipo = "multiplicacao";  }
 | expressao '/' expressao { $tipo = "divisao";        }
 | expressao '+' expressao { $tipo = "soma";           }
 | expressao '-' expressao { $tipo = "subtracao";      }
 | '(' expressao ')'       { $tipo = "parenteses";     }
 | valor                   { $tipo = "valor";          }
 | identificadorF          { $tipo = "identificadorF"; }
 | X                       { $tipo = "X";              }
 | Identificador           { $tipo = "identificador";  }
 | seno                    { $tipo = "seno";           }
 | cosseno                 { $tipo = "cosseno";        }
 ;
```

```
integral
 : 'integre' expressao 'd' X intervaloIntegracao
 ;
 derivada
 : 'derive' expressao (pontoDerivacao)?
 ;
pontoDerivacao
 : 'em' expressao
 ;
polinomio
 :
 | monomio+
 ;
monomio
 : coeficiente? incognita expoente?
 | coef2 = coeficiente
 ;
incognita
 : X
 ;
```

Esboço da gramática

```
coeficiente returns [ String tipo ]
 : numeroComSinal { $tipo = "numero";}
 | expressao      { $tipo = "expressao";}
 ;
expoente
 : '^' numeroComSinal
 ;
intervaloIntegracao
 : 'de' l1 = limiteIntegracao 'a' l2 = limiteIntegracao
 ;
limiteIntegracao
 : expressao
 ;
identificadorF
 : ID1 = Identificador '('(ID2 = X | expressao)')'
 ;
relacao
 : '{' dominio'|'contradominio'}'
 ;
```

```
dominio
 : ('N' | 'Z' | 'Q' | 'R') (intervalo)?
 ;
contradominio
 : ('N' | 'Z' | 'Q' | 'R') (intervalo)?
 ;
intervalo
 : '['valor '<->' valor']'
 ;
seno
 : 'sen''('expressao')'
 ;
cosseno
 : 'cos''('expressao')'
 ;
valor returns [ String tipo ]
 : Numero    { $tipo = "numero";    }
 | constante { $tipo = "constante"; }
 ;
```

Esboço da gramática

```
constante
 :  'pi'
 |  'e'   //euler
 |  '+infinito'
 |  '-infinito'
 ;
numeroComSinal
 : Numero
 | '+'Numero
 ;
X
 : 'x'
 ;
Identificador
 : ('a'..'c'|'e'..'z'|'A'..'Z'|'_')
('a'..'z'|'A'..'Z'|'0'..'9'|'_')*
 ;
```

```
Numero
 : Int ('.' Digito*)?
 | '-'Int ('.' Digito*)?
 ;
String
 : '"' ~('\r'|'\n')*? '"'
     {
      String s = getText();
      s = s.substring(1, s.length() - 1);
      setText(s);
     }
 ;
Comentario
 : ('//' ~[\r\n]* | '/*' .*? '*/') -> skip
 ;
Espaco
 : [ \t\r\n\u000C] -> skip
 ;
```

```
Letra
 :'a'..'z'
 |'A'..'Z'
 ;
fragment Int
 : [1-9] Digito*
 | '0'
 ;
fragment Digito
 : '0'..'9'
 ;
```

Descrição da análise semântica

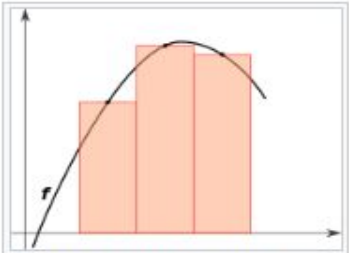
```
public Double visitRetorno
    if (ctx.tipo.equals("expressao"))
    else if (ctx.tipo.equals("string"))
    System.out.println  ("A linguagem não aceita esse tipo de retorno.");

public Double visitPrint
    if (ctx.tipoPrint.equals("expressao")) {
    if (ctx.expressao().tipo.equals("identificadorF")
    else if(etds.cosseno == null)
    else System.out.println("cos(x)");
    else if(ctx.tipoPrint.equals("string")){


public Double visitAtribuicao(matematicaParser.AtribuicaoContext ctx) {
```

Descrição da análise semântica

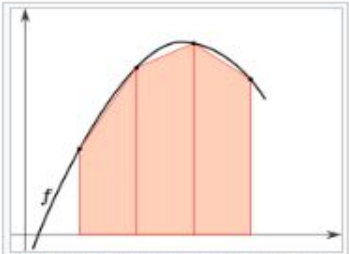
```
public Double visitFuncao(matematicaParser.FuncaoContext ctx) {
        switch (ctx.tipo) {
            case "polinomio":
            case "cosseno":
            case "seno":
            case "derivada":
public Double visitExpressao(matematicaParser.ExpressaoContext ctx) {
        switch (ctx.tipo) {
            case "identificador": //retornar o valor de um identificador
            case "identificadorF"://retornar o valor de uma funcao calculada em um ponto
            case "seno":
            case "cosseno":
            case "soma":
            case "subtracao":
            case "divisao":
            case "multiplicacao":
```

Descrição da análise semântica

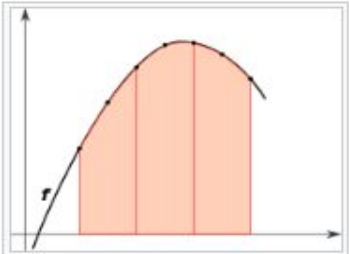
```
        case "potencia":
        case "integral": SimpsonIntegrator(relativeAccuracy, absoluteAccuracy,
                         minimalIterationCount, maximalIterationCount);
        case "derivada":
        case "parenteses":
        case "unario":
        case "valor":
public Double visitConstante(matematicaParser.ConstanteContext ctx) {
   if(ctx.getText().equals("+infinito")){
   else if(ctx.getText().equals("-infinito")){
   if(ctx.getText().equals("pi")){
   return Math.E; // euler
public Double visitLimiteIntegracao(matematicaParser.LimiteIntegracaoContext ctx) {
   else if(ctx.expressao().Identificador() != null) {
```



Superfície em vermelho representa o valor estimado da integral pelo método do ponto médio ou dos retângulos



Superfície em vermelho representa o valor estimado da integral pelo método dos trapézios



Superfície em vermelho representa o valor estimado da integral pelo método Simpson

Exemplos

```
print("5) - Testes de atribuição e avaliação de
funções polinomiais e trigonométricas")
g(x) = 1x^2 -2x +3
k(x) = -2x +3 -1x^12

print("Exibindo funcoes:")
print(g(x))
print(k(x))

print("Calculando composições de funcoes do tipo
f(g(X)) e g(f(x))")
print( g(k(1)) )
print( k(cos(-1)) )
```

```
print("6) - Testes de atribuição e avaliação de
funções trigonométricas")
c1(x) = cos(x)
s1(x) = sen(x)
print(c1(pi))
print(s1(pi))
tg_PI = s1(pi)/c1(pi)
print("tangente de pi")
print(tg_PI)
```

```
/usr/lib/jvm/java-8-oracle/bin/java ...
5) - Testes de atribuição e avaliação de funções polinomiais e trigonométricas
Exibindo funcoes:
3 - 2 x + x^2
3 - 2 x - x^12
Calculando composições de funcoes do tipo f(g(X)) e g(f(x))
3.0
1.9187764578102526
6) - Testes de atribuição e avaliação de funções trigonométricas
-1.0
1.2246467991473532E-16
tangente de pi
-1.2246467991473532E-16
```

Outros exemplos

```
print("7) - Testes de integração de funções ")
print(integre g(x)dx  de 0 a e       )
print(integre k(x)dx  de -1 a pi     )
print(integre c1(x)dx de 0 a pi      )
print(integre s1(x)dx de 0 a 2*pi    )

print("8) - Testes de derivação de funções")
gLinha(x) = derive g(x)
print(gLinha(x))
print(derive k(x) em 17)

senoLinha(x) = derive s1(x)
print(senoLinha(x))
print(derive s1(x) em pi)
```

```
print("1)Programa para o cálculo de distâncias a partir
da velocidade e aceleração.")
a1 = 1
c = 0.5
Distancia(x) = 1x^2 - (4*a1*c)
print("Distância em quilômetros: ")
print(Distancia(12))
```

```
7) - Testes de integração de funções
7.460968360842375
-223356.5511831164
1.2246467991473532E-16
0.0
8) - Testes de derivação de funções
0
-4.112627556915 98E14
0
-1.0
1)Programa para o cálculo de distâncias a partir da velocidade e aceleração
Distância em quilômetros:
142.0
```

Outros exemplos

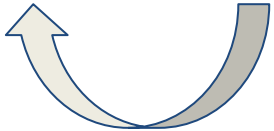
```
print("2)Programa para o cálculo de área delimitada por um gráfico de uma f(x) qualquer.")
{R|R}
f(x) = (1/4)x^3 +4
Area = integre f(x)dx de 1 a 10
print("Area total em m²:")
print(Area)

print("3)Programa para o cálculo de volume de sólidos de revolução.")
{R|R}
g(x) = cos(x)
A = -2
B = 8
Vn = pi*(integre g(x)dx de A a B)
print("Volume em m³: ")
print(Vn)
```

```
print("4)Programa para o cálculo de Momentos de Inércia de Sólidos.")
r = 7.5
V = 2
Iz = integre r^2 dx de 0 a V
print("Momento de inércia do sólido: ")
print(Iz)
```

```
2)Programa para o cálculo de área delimitada por um gráfico de uma f(x) qualquer.
Area total em m²:
660.9375
3)Programa para o cálculo de volume de sólidos de revolução.
Volume em m³:
5.964802715404159
4)Programa para o cálculo de Momentos de Inércia de Sólidos.
Momento de inércia do sólido:
112.5
RETORNO: sucesso

Process finished with exit code 0
```

Dificuldades encontradas

- Escrever o pom.xml para não precisar rodar na linha de comando
- Definir um polinômio, fazer o parser, armazenar na TDS
- Aprender a usar a Apache Commons Math
- Testar  debugar



Referências

- Manual da linguagem Matemática
  - https://drive.google.com/open?id=1Bhgjh5dFc81ozY33lEqTiORoUconQaWp1RgAKHIOdN4