VISÃO GERAL DA LINGUAGEM

A nossa linguagem, a Gambiarra++, é uma linguagem bem básica, que toma algumas inspirações em PHP e na linguagem exemplo apresentada pelo professor Maurício, tentando ser o mais simples de se ler o possível.

DEFINIÇÃO LÉXICA DA LINGUAGEM

Padrão	Tipo de Lexema	Sigla
Os próprios lexemas	Palavras-chave: DECLARACOES, ALGORITMO	3 primeiras letras
Os próprios lexemas	Operadores de Tipo: int, real, bool, string	TIPO
O próprio lexema	Retorno de dados: retorna	RET
O próprio lexema	Estrutura de repetição: enquanto, para	REP/PARA
O próprio lexema	Parênteses: (,)	AP/FP
O próprio lexema	Chave: {,}	AC/FC
O próprio lexema	Cifrão: $	CIF
O próprio lexema	Delimitador: ;	PTV
Sequência de caracteres iniciado com $ seguido de pelo menos uma letra ou número	VARIÁVEL	VAR
Pelo menos um número seguido de n outros e possivelmente ponto para decimais	NÚMERO	NUM
O próprio lexema	Operadores Aritmético: +, -, *, /	OP_ARIT
O próprio lexema	Operadores Relacionais: <, >, <=, >=, =	OP_RELA
O próprio lexema	Termos “ignoráveis”: , \r, \t, \n	WS

EXEMPLOS DE USO DA LINGUAGEM

A seguir temos prints da implementação dos algoritmos de Fatorial e Soma dos N Termos da Sequência de Fibonacci.

Algoritmo de fatorial

```
DECLARACOES
int $resultado
int $n

ALGORITMO
int fatorial($n){
    $resultado = 1;
    enquanto ($n > 1) {
        $resultado = $resultado * $n;
        $n = $n - 1;
    }
    retorna $resultado;
}
```

Algoritmo de Fibonacci

```
DECLARACOES
int $a;
int $b;
int $n;
int $i;
int $soma;
int $temp;

ALGORITMO
int somaFibonacci($n) {
    $a = 0;
    $b = 1;
    $soma = 0;
    para ($i = 0; $i < $n; $i++) {
        $soma = $soma + $a;
        $temp = a;
        $a = $b;
        $b = $temp + $b;
    }
    return $soma;
}

```

IMPLEMENTAÇÃO DO ANALISADOR LÉXICO

1. Criação da Gramática

Para a implementação do analisador léxico, foi utilizado o ANTLR, um gerador de analisadores léxicos e sintáticos que facilita a definição de gramáticas formais para linguagens específicas. A gramática foi definida em um arquivo com a extensão `.g4`, que especifica as regras léxicas (tokens) e, opcionalmente, regras sintáticas.

2. Uso de Geradores de Analisadores

Após a definição da gramática, o ANTLR gera automaticamente o analisador léxico (Lexer) e, se especificado, o analisador sintático (Parser). O comando para gerar os arquivos é:

```
antlr4 -Dlanguage=Java MinhaLinguagem.g4
```

Esse comando cria os arquivos `MinhaLinguagemLexer.java` e `MinhaLinguagemParser.java`, entre outros, que podem ser usados no código Java para executar o analisador.

3. Descrição do Programa Desenvolvido

O programa principal, escrito em Java, utiliza os arquivos gerados pelo ANTLR para processar um código-fonte de entrada. O código a seguir mostra um exemplo básico de como usar o Lexer para analisar um arquivo de entrada:

```java
import org.antlr.v4.runtime.CharStream;
import org.antlr.v4.runtime.CharStreams;
import org.antlr.v4.runtime.Token;

public class MainLexer {

    public static void main(String[] args) {
        String filename = "C:\\Users\\luis felipe\\UFLA 2024-2\\compiladores\\fibonacci.txt";
        try {
            CharStream input = CharStreams.fromFileName(filename);
            MinhaGramaticaLexer lexer = new MinhaGramaticaLexer(input);
            Token token;
            while(!lexer._hitEOF){
                token = lexer.nextToken();
                System.out.println("Token: "+ token.toString());
                System.out.println("   Lexema: "+ token.getText());
                System.out.println("   Classe: "+ lexer.getVocabulary().getDisplayName(token.getType()));
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4. Descrição dos Artefatos

- **Arquivo `.g4`**: Contém a definição da gramática utilizada pelo ANTLR.
- **Arquivos gerados (`Lexer` e `Parser`)**: Produzidos automaticamente pelo ANTLR, são utilizados pelo programa principal.
- **Código Java**: Implementa a lógica para leitura de arquivos, execução do analisador léxico e exibição dos tokens.

CASOS DE TESTE

A seguir temos prints do resultado da execução dos algoritmos de Fatorial e Soma dos N Termos da Sequência de Fibonacci:

Fatorial

```
Token: [@-1,0:10='DECLARACOES',<1>,1:0]
   Lexema: DECLARACOES
   Classe: 'DECLARACOES'
Token: [@-1,13:15='int',<3>,2:0]
   Lexema: int
   Classe: TIPO
Token: [@-1,17:26='$resultado',<13>,2:4]
   Lexema: $resultado
   Classe: VAR
Token: [@-1,29:31='int',<3>,3:0]
   Lexema: int
   Classe: TIPO
Token: [@-1,33:34='$n',<13>,3:4]
   Lexema: $n
   Classe: VAR
Token: [@-1,39:47='ALGORITMO',<2>,5:0]
   Lexema: ALGORITMO
   Classe: 'ALGORITMO'
Token: [@-1,50:52='int',<3>,6:0]
   Lexema: int
   Classe: TIPO
Token: [@-1,54:61='fatorial',<17>,6:4]
   Lexema: fatorial
   Classe: FUN
Token: [@-1,62:62='(',<9>,6:12]
   Lexema: (
   Classe: '('
Token: [@-1,63:64='$n',<13>,6:13]
   Lexema: $n
   Classe: VAR
Token: [@-1,65:65=')',<10>,6:15]
   Lexema: )
   Classe: ')'
```

```
Token: [@-1,66:66='{',<7>,6:16]
   Lexema: {
   Classe: '{'
Token: [@-1,73:82='$resultado',<13>,7:4]
   Lexema: $resultado
   Classe: VAR
Token: [@-1,84:84='=',<16>,7:15]
   Lexema: =
   Classe: OP_RELA
Token: [@-1,86:86='1',<14>,7:17]
   Lexema: 1
   Classe: NUM
Token: [@-1,87:87=';',<12>,7:18]
   Lexema: ;
   Classe: ';'
Token: [@-1,94:101='enquanto',<5>,8:4]
   Lexema: enquanto
   Classe: 'enquanto'
Token: [@-1,103:103='(',<9>,8:13]
   Lexema: (
   Classe: '('
Token: [@-1,104:105='$n',<13>,8:14]
   Lexema: $n
   Classe: VAR
Token: [@-1,107:107='>',<16>,8:17]
   Lexema: >
   Classe: OP_RELA
Token: [@-1,109:109='1',<14>,8:19]
   Lexema: 1
   Classe: NUM
Token: [@-1,110:110=')',<10>,8:20]
   Lexema: )
   Classe: ')'
Token: [@-1,112:112='{',<7>,8:22]
```

Fibonnaci

```
Token: [@-1,112:112='{',<7>,8:22]
   Lexema: {
   Classe: '{'
Token: [@-1,123:132='$resultado',<13>,9:8]
   Lexema: $resultado
   Classe: VAR
Token: [@-1,134:134='=',<16>,9:19]
   Lexema: =
   Classe: OP_RELA
Token: [@-1,136:145='$resultado',<13>,9:21]
   Lexema: $resultado
   Classe: VAR
Token: [@-1,147:147='*',<15>,9:32]
   Lexema: *
   Classe: OP_ARIT
Token: [@-1,149:150='$n',<13>,9:34]
   Lexema: $n
   Classe: VAR
Token: [@-1,151:151=';',<12>,9:36]
   Lexema: ;
   Classe: ';'
Token: [@-1,162:163='$n',<13>,10:8]
   Lexema: $n
   Classe: VAR
Token: [@-1,165:165='=',<16>,10:11]
   Lexema: =
   Classe: OP_RELA
Token: [@-1,167:168='$n',<13>,10:13]
   Lexema: $n
   Classe: VAR
Token: [@-1,170:170='-',<15>,10:16]
   Lexema: -
   Classe: OP_ARIT
Token: [@-1,172:172='1',<14>,10:18]
   Lexema: 1
   Classe: NUM
Token: [@-1,173:173=';',<12>,10:19]
   Lexema: ;
   Classe: ';'
Token: [@-1,180:180='}',<8>,11:4]
   Lexema: }
   Classe: '}'
Token: [@-1,187:193='retorna',<4>,12:4]
   Lexema: retorna
   Classe: 'retorna'
Token: [@-1,195:204='$resultado',<13>,12:12]
   Lexema: $resultado
   Classe: VAR
Token: [@-1,205:205=';',<12>,12:22]
   Lexema: ;
   Classe: ';'
Token: [@-1,208:208='}',<8>,13:0]
   Lexema: }
   Classe: '}'
```

```
Token: [@-1,0:10='DECLARACOES',<1>,1:0]
   Lexema: DECLARACOES
   Classe: 'DECLARACOES'
Token: [@-1,13:15='int',<3>,2:0]
   Lexema: int
   Classe: TIPO
Token: [@-1,17:18='$a',<13>,2:4]
   Lexema: $a
   Classe: VAR
Token: [@-1,19:19=';',<12>,2:6]
   Lexema: ;
   Classe: ';'
Token: [@-1,22:24='int',<3>,3:0]
   Lexema: int
   Classe: TIPO
Token: [@-1,26:27='$b',<13>,3:4]
   Lexema: $b
   Classe: VAR
Token: [@-1,28:28=';',<12>,3:6]
   Lexema: ;
   Classe: ';'
Token: [@-1,31:33='int',<3>,4:0]
   Lexema: int
   Classe: TIPO
Token: [@-1,35:36='$n',<13>,4:4]
   Lexema: $n
   Classe: VAR
Token: [@-1,37:37=';',<12>,4:6]
   Lexema: ;
   Classe: ';'
Token: [@-1,40:42='int',<3>,5:0]
   Lexema: int
   Classe: TIPO
Token: [@-1,44:45='$i',<13>,5:4]
   Lexema: $i
   Classe: VAR
Token: [@-1,46:46=';',<12>,5:6]
   Lexema: ;
   Classe: ';'
Token: [@-1,49:51='int',<3>,6:0]
   Lexema: int
   Classe: TIPO
Token: [@-1,53:57='$soma',<13>,6:4]
   Lexema: $soma
   Classe: VAR
Token: [@-1,58:58=';',<12>,6:9]
   Lexema: ;
   Classe: ';'
Token: [@-1,61:63='int',<3>,7:0]
   Lexema: int
   Classe: TIPO
Token: [@-1,65:69='$temp',<13>,7:4]
   Lexema: $temp
   Classe: VAR
Token: [@-1,70:70=';',<12>,7:9]
   Lexema: ;
   Classe: ';'
Token: [@-1,75:83='ALGORITMO',<2>,9:0]
   Lexema: ALGORITMO
   Classe: 'ALGORITMO'
Token: [@-1,86:88='int',<3>,10:0]
   Lexema: int
   Classe: TIPO
Token: [@-1,90:102='somaFibonacci',<17>,10:4]
   Lexema: somaFibonacci
   Classe: FUN
```

```
Token: [@-1,103:103='(',<9>,10:17]
   Lexema: (
   Classe: '('
Token: [@-1,104:105='$n',<13>,10:18]
   Lexema: $n
   Classe: VAR
Token: [@-1,106:106=')',<10>,10:20]
   Lexema: )
   Classe: ')'
Token: [@-1,108:108='{',<7>,10:22]
   Lexema: {
   Classe: '{'
Token: [@-1,115:116='$a',<13>,11:4]
   Lexema: $a
   Classe: VAR
Token: [@-1,118:118='=',<16>,11:7]
   Lexema: =
   Classe: OP_RELA
Token: [@-1,120:120='0',<14>,11:9]
   Lexema: 0
   Classe: NUM
Token: [@-1,121:121=';',<12>,11:10]
   Lexema: ;
   Classe: ';'
Token: [@-1,128:129='$b',<13>,12:4]
   Lexema: $b
   Classe: VAR
Token: [@-1,131:131='=',<16>,12:7]
   Lexema: =
   Classe: OP_RELA
Token: [@-1,133:133='1',<14>,12:9]
   Lexema: 1
   Classe: NUM
Token: [@-1,134:134=';',<12>,12:10]
   Lexema: ;
   Classe: ';'
Token: [@-1,141:145='$soma',<13>,13:4]
   Lexema: $soma
   Classe: VAR
Token: [@-1,147:147='=',<16>,13:10]
   Lexema: =
   Classe: OP_RELA
Token: [@-1,149:149='0',<14>,13:12]
   Lexema: 0
   Classe: NUM
Token: [@-1,150:150=';',<12>,13:13]
   Lexema: ;
   Classe: ';'
Token: [@-1,157:160='para',<6>,14:4]
   Lexema: para
   Classe: 'para'
Token: [@-1,162:162='(',<9>,14:9]
   Lexema: (
   Classe: '('
Token: [@-1,163:164='$i',<13>,14:10]
   Lexema: $i
   Classe: VAR
Token: [@-1,166:166='=',<16>,14:13]
   Lexema: =
   Classe: OP_RELA
Token: [@-1,168:168='0',<14>,14:15]
   Lexema: 0
   Classe: NUM
Token: [@-1,169:169=';',<12>,14:16]
   Lexema: ;
   Classe: ';'
```

```
Token: [@-1,171:172='$i',<13>,14:18]
   Lexema: $i
   Classe: VAR
Token: [@-1,174:174='<',<16>,14:21]
   Lexema: <
   Classe: OP_RELA
Token: [@-1,176:177='$n',<13>,14:23]
   Lexema: $n
   Classe: VAR
Token: [@-1,178:178=';',<12>,14:25]
   Lexema: ;
   Classe: ';'
Token: [@-1,180:181='$i',<13>,14:27]
   Lexema: $i
   Classe: VAR
Token: [@-1,182:182='+',<15>,14:29]
   Lexema: +
   Classe: OP_ARIT
Token: [@-1,183:183='+',<15>,14:30]
   Lexema: +
   Classe: OP_ARIT
Token: [@-1,184:184=')',<10>,14:31]
   Lexema: )
   Classe: ')'
Token: [@-1,186:186='{',<7>,14:33]
   Lexema: {
   Classe: '{'
Token: [@-1,197:201='$soma',<13>,15:8]
   Lexema: $soma
   Classe: VAR
Token: [@-1,203:203='=',<16>,15:14]
   Lexema: =
   Classe: OP_RELA
Token: [@-1,205:209='$soma',<13>,15:16]
   Lexema: $soma
   Classe: VAR
Token: [@-1,211:211='+',<15>,15:22]
   Lexema: +
   Classe: OP_ARIT
Token: [@-1,213:214='$a',<13>,15:24]
   Lexema: $a
   Classe: VAR
Token: [@-1,215:215=';',<12>,15:26]
   Lexema: ;
   Classe: ';'
Token: [@-1,226:230='$temp',<13>,16:8]
   Lexema: $temp
   Classe: VAR
Token: [@-1,232:232='=',<16>,16:14]
   Lexema: =
   Classe: OP_RELA
Token: [@-1,234:234='a',<17>,16:16]
   Lexema: a
   Classe: FUN
Token: [@-1,235:235=';',<12>,16:17]
   Lexema: ;
   Classe: ';'
Token: [@-1,246:247='$a',<13>,17:8]
   Lexema: $a
   Classe: VAR
Token: [@-1,249:249='=',<16>,17:11]
   Lexema: =
   Classe: OP_RELA
Token: [@-1,251:252='$b',<13>,17:13]
   Lexema: $b
   Classe: VAR
```

```
Token: [@-1,308:308='}',<8>,21:0]
   Lexema: }
   Classe: '}'
Token: [@-1,311:310='<EOF>',<-1>,22:0]
   Lexema: <EOF>
   Classe: EOF
```