

Especificação da Fase 1

Bacharelado em Ciência da Computação
Universidade Federal de São Carlos, Campus Sorocaba
Compiladores

1 Primeira Fase

Nesta primeira fase vocês realizarão a análise léxica e sintática para a linguagem descrita pela gramática da Seção 3.

2 Entrega

Data: 31/05/19.

Entregue um arquivo compactado .zip com o formato de nome: F01_Nome1_Nome2_Nome3_Nome4.zip, contendo apenas os arquivos de extensão .java e testes realizados, respeitando a seguinte estrutura:

- F01_Nome1_Nome2_Nome3_Nome4/AST: Classes da árvore sintática.
- F01_Nome1_Nome2_Nome3_Nome4/Lexer: Classes do analisador léxico.
- F01_Nome1_Nome2_Nome3_Nome4/Testes: Arquivos de teste (Opcional).
- F01_Nome1_Nome2_Nome3_Nome4/Compiler.java: Arquivo principal e analisador sintático.
- F01_Nome1_Nome2_Nome3_Nome4/Main.java: Arquivo que contém a função void main().

Em nome poder ser NomeSobrenome, para diferenciar alunos com nomes iguais.

Use o arquivo Main.java do compilador 9 (da apostila do Zé), que toma como argumento um arquivo com o código a ser analisado e imprime a saída para tela.

Seu Compiler.java deve implementar um método compile(), chamado em Main.java.

Adicione um cabeçalho com nome e RA dos integrantes do grupo em todos os arquivos *.java que forem entregues.

Deve ser possível compilar esta estrutura via linha de comando com javac.

3 Gramática

Na gramática, são terminais:

- Id, um identificador;
- LiteralInt, um número inteiro entre 0 e 2147483647;
- LiteralString, uma sequência de caracteres entre " e ".

Há algumas produções não fatoradas, como **AssignExprStat** (não se sabe se teremos uma atribuição ou apenas uma expressão) e **ExprPrimary** (não se sabe se teremos apenas um Id, uma variável, ou Id seguido de "(", uma chamada de função).

```
Program ::= Func {Func}  
Func ::= "function" Id [ "(" ParamList ")" ] ["->" Type ] StatList  
ParamList ::= ParamDec {", " ParamDec}  
ParamDec ::= Id ":" Type
```

```

Type ::= "Int" | "Boolean" | "String"
StatList ::= "{" {Stat} "}"
Stat ::= AssignExprStat | ReturnStat | VarDecStat | IfStat | WhileStat
AssignExprStat ::= Expr [ "=" Expr ] ";"
ReturnStat ::= "return" Expr ";"
VarDecStat ::= "var" Id ":" Type ";"
IfStat ::= "if" Expr StatList [ "else" StatList ]
WhileStat ::= "while" Expr StatList
Expr ::= ExprAnd {"or" ExprAnd}
ExprAnd ::= ExprRel {"and" ExprRel}
ExprRel ::= ExprAdd [ RelOp ExprAdd ]
ExprAdd ::= ExprMult {"+" | "-"} ExprMult
ExprMult ::= ExprUnary {"*" | "/" } ExprUnary
ExprUnary ::= [ "(" "+" | "-" ) ] ExprPrimary
ExprPrimary ::= Id | FuncCall | ExprLiteral
ExprLiteral ::= LiteralInt | LiteralBoolean | LiteralString
LiteralBoolean ::= "true" | "false"
FuncCall ::= Id "(" [ Expr {"," Expr} ] ")"

```

4 Exemplo

```

function fatorial(n : Int) -> int {
    if n <= 0 {
        return 1;
    }
    else {
        return n*fatorial(n-1);
    }
}

function imprima(before: Int, valor: Int, after: String) {
    var Int i;
    i = 0;
    while i < before {
        write("*");
        i = i + 1;
    }
    writeln("");
    writeln(valor);
    writeln(after);
}

function main {
    imprima(50, fatorial(5)*2*fatorial(3), "    fim");
}

```