



UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO - IC
CIÊNCIA DA COMPUTAÇÃO

JOÃO VICTOR DE ALARCÃO AYALLA ALCÂNTARA
ASCANIO SAVIO DE ARAUJO NEVES
JACKSON BARBOSA DA SILVA

COMPILADORES
GRAMÁTICA LL(1) DA LINGUAGEM AJA++

Sumário

- 1. Tipo de analisador sintático implementado**
- 2. Gramática LL(1)**

1 - Tipo de analisador sintático implementado

O tipo de analisador sintático implementado foi o analisador descendente LL(1) preditivo recursivo.

2- Gramática LL(1)

$S = \text{FunctionDeclaration } S \mid \text{Declaration } ';' S \mid \epsilon$

$\text{FunctionDeclaration} = \text{'function' } \text{FunctionDeclarationAuxiliar}$

$\text{FunctionDeclarationAuxiliar} = \text{VoidFunction} \mid \text{OtherTypeFunction}$

$\text{VoidFunction} = \text{'void' } \text{VoidFunctionAuxiliar}$

$\text{VoidFunctionAuxiliar} = \text{MainFunction} \mid \text{Function}$

$\text{OtherTypeFunction} = \text{FunctionType } \text{Function}$

$\text{MainFunction} = \text{'main' } \text{'{' } \text{'}}' \text{'[' } \text{CommandsBlock } \text{'}]'$

$\text{Function} = \text{'Identificator' } \text{'{' } \text{Parameters } \text{'}}' \text{'[' } \text{CommandsBlock } \text{'}]'$

$\text{FunctionType} = \text{'itg' } \mid \text{'dbl' } \mid \text{'chr' } \mid \text{'bool' } \mid \text{'str'}$

$\text{Parameters} = \text{ParametersList} \mid \epsilon$

$\text{ParametersList} = \text{Declaration } \text{ParametersListAuxiliar}$

$\text{ParametersListAuxiliar} = \text{' ,' } \text{ParametersList} \mid \epsilon$

$\text{CommandsBlock} = \text{Command } \text{CommandsBlock} \mid \epsilon$

$\text{Command} = \text{IdentificatorCommand } ';' \mid \text{DeclarationCommand } ';' \mid \text{Condition} \mid \text{Loop} \mid \text{Output } ';' \mid \text{Input } ';' \mid \text{Return } ';' \mid \epsilon$

$\text{IdentificatorCommand} = \text{'Identificator' } \text{IdentificatorCommandAuxiliar}$

$\text{IdentificatorCommandAuxiliar} = \text{FunctionCall} \mid \text{AttributionExpression} \mid \text{ConcatenationExpression} \mid \text{AppendList} \mid \text{IncrementOperation} \mid \text{ListAccessAuxiliar}$

$\text{DeclarationCommand} = \text{ListDeclaration} \mid \text{VariableDeclaration} \mid \epsilon$

$\text{DeclarationCommandAuxiliar} = \text{AttributionExpression} \mid \epsilon$

FunctionCall = '{ ' IdList ' }

IdList = IdSequence | ϵ

IdSequence = 'Identifier' IdSequenceAuxiliar

IdSequenceAuxiliar = ', ' IdList | ϵ

AttributionExpression = '=' ArithmeticExpression

DeclarationAttributionExpression = VariableDeclaration '=' ArithmeticExpression

ArithmeticExpression = AdditiveExpression ArithmeticAuxiliar

ArithmeticAuxiliar = BitwiseOperation AdditiveExpression ArithmeticAuxiliar | ϵ

AdditiveExpression = MultiplicativeExpression AdditiveAuxiliar

AdditiveAuxiliar = AdditiveOperation MultiplicativeExpression AdditiveAuxiliar | ϵ

MultiplicativeExpression = ArithmeticValue MultiplicativeAuxiliar

MultiplicativeAuxiliar = MultiplicativeOperation ArithmeticValue
MultiplicativeAuxiliar | ϵ

ArithmeticValue = ArithmeticFactor ArithmeticValueAuxiliar

ArithmeticValueAuxiliar = IncrementOperation | ϵ

ArithmeticFactor = Constant | IdentifierFactor

IdentifierFactor = 'Identifier' IdentifierFactorAuxiliar

IdentifierFactorAuxiliar = FunctionCall | ListAccesss | ϵ

ListAccesss = 'OpenPar' ArithmeticExpression 'ClosePar'

BitwiseOperation = 'OperationXor' | 'OperationOr' | 'OperationAnd'

AdditiveOperation = 'OperationAdd' | 'OperationSub'

MultiplicativeOperation = 'OperationMult' | 'OperationDiv'

IncrementOperation = 'OperationInc' | 'OperationDec'

ConcatenationExpression = BooleanExpression ConcatenationAuxiliar

ConcatenationAuxiliar = 'OperationConc' BooleanExpression
ConcatenationAuxiliar | ϵ

BooleanExpression = BooleanTerm BooleanAuxiliar

BooleanAuxiliar = 'LogicOr' BooleanTerm BooleanAuxiliar | ϵ

BooleanTerm = BooleanFactor BooleanAuxiliarTerm

BooleanAuxiliarTerm = 'LogicAnd' BooleanFactor BooleanAuxiliarTerm | ϵ

BooleanFactor = 'LogicNot' BooleanRelation | BooleanRelation

BooleanRelation = ArithmeticExpression BooleanRelationAuxiliar

BooleanRelationAuxiliar = LogicalRelation ArithmeticExpression | ϵ

Declaration = VariableDeclaration | ListDeclaration

ListDeclaration = 'list' '(' Type ')' 'Identificator'

VariableDeclaration = Type 'Identificator'

Type = 'itg' | 'dbl' | 'chr' | 'bool' | 'str'

Condition = IfCommand ConditionAuxiliar

ElseCondition = ElseIfCommand ConditionAuxiliar | ElseCommand

ConditionAuxiliar = ElseCondition | ϵ

IfCommand = 'if' '{' BooleanExpression '}' '[' CommandsBlock ']'

ElseIfCommand = 'elseif' '{' BooleanExpression '}' '[' CommandsBlock ']'

ElseCommand = 'else' '[' CommandsBlock ']'

Loop = ForStatement | WhileStatement

ForStatement = 'for' '{' 'Identifier' ForStatmentAuxiliar

ForStatmentAuxiliar = LogicForStatement | CounterForStatement

LogicForStatement = AttributionExpression ';' BooleanExpression ';' ArithmeticExpression '}' '[' CommandsBlock ']'

CounterForStatement = ',' IntValue ',' IntValue ',' IntValue

CounterForStatementAuxiliar

CounterForStatementAuxiliar = '}' '[' CommandsBlock ']' | ',' IntValue '}' '['

CommandsBlock ']'

WhileStatement = 'while' '{' BooleanExpression '}' '[' CommandsBlock ']'

LogicalRelation = 'RelationEqual' | 'RelationNotEqual' | 'RelationGreater' |

'RelationLower' | 'RelationGreaterEqual' | 'RelationLowerEqual'

Output = 'write' '{' OutputValues '}'

OutputValues = ArithmeticExpression OutputValuesAuxiliar

OutputValuesAuxiliar = ',' OutputValues | ϵ

Input = 'read' '{' IdSequence '}'

AppendList = '.' 'append' '{' ArithmeticExpression '}'

Return = 'return' ArithmeticExpression

Constant = 'CharConst' | 'StringConst' | 'DoubleConst' | 'IntConst' | 'BooleanConst'

IntValue = 'Identifier' | 'IntConst'

ListAccessAuxiliar = ListAccess AttributionExpression