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

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

2- Gramática LL(1)

S = FunctionDeclaration S | Declaration ´;´ S | ε

FunctionDeclaration = ´function´ FunctionDeclarationAuxiliar

FunctionDeclarationAuxiliar = VoidFunction | OtherTypeFunction

VoidFunction = ´void´ VoidFunctionAuxiliar

VoidFunctionAuxiliar = MainFunction | Function

OtherTypeFunction = FunctionType Function

MainFunction = ´main´ ´{´ ´}´ ´[´ CommandsBlock ´]´

Function = ´Identificator´ ´{´ Parameters ´}´ ´[´ CommandsBlock ´]´

FunctionType = ´itg´ | ´dbl´ | ´chr´ | ´bool´ | ‘str´

Parameters = ParametersList | ε

ParametersList = Declaration ParametersListAuxiliar

ParametersListAuxiliar = ´,´ ParametersList | ε

CommandsBlock = Command CommandsBlock | ε

Command = IdentificatorCommand ´;´ | DeclarationCommand ´;´ | Condition | Loop | Output ´;´ | Input ´;´ | Return ´;´

IdentificatorCommand = ´Identificator´ IdentificatorCommandAuxiliar

IdentificatorCommandAuxiliar = FunctionCall | AttributionExpression | IdentificatorConcatenationExpression | AppendList | IncrementOperation | ListAcessAuxiliar

DeclarationCommand = ListDeclaration | VariableDeclaration DeclarationCommandAuxiliar

DeclarationCommandAuxiliar = AttributionExpression | ε

FunctionCall = ´{´ IdList ´}´

IdList = IdSequence | ε

IdSequence = ´Identificator´ IdSequenceAuxiliar

IdSequenceAuxiliar = ´,´ IdList | ε

AttributionExpression = ´=´ ArithmeticExpression

DeclarationAttributionExpression = VariableDeclaration ´=´ ArithmeticExpression

ArithmeticExpression = AddtiveExpression ArithmeticAuxiliar

ArithmeticAuxiliar = BitwiseOperation AddtiveExpression ArithmeticAuxiliar | ε

AddtiveExpression = MultiplicativeExpression AddtiveAuxiliar

AddtiveAuxiliar = AddtiveOperation MultiplicativeExpression AddtiveAuxiliar | ε

MultiplicativeExpression = ArithmeticValue MultiplicativeAuxiliar

MultiplicativeAuxiliar = MultiplicativeOperation ArithmeticValue
MultiplicativeAuxiliar | ε

ArithmeticValue = ArithmeticFactor ArithmeticValueAuxiliar

ArithmeticValueAuxiliar = IncrementOperation | ε

ArithmeticFactor = Constant | IdentificatorFactor

IdentificatorFactor = ´Identificator´ IdentificatorFactorAuxiliar

IdentificatorFactorAuxiliar = FunctionCall | ListAcesss | ε

ListAcesss = ´OpenPar´ ArithmeticExpression ´ClosePar´

BitwiseOperation = ´OperationXor´ | ´OperationOr´ | ´OperationAnd´

AddtiveOperation = ´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´ ´{´ ´Identificator´ 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 = ´Identificator´ | ´IntConst

ListAcessAuxiliar = ListAcess AttributionExpression