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´ ´{´ ParametersList ´}´ ´[´ CommandsBlock ´]´

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

ParametersList = Declaration ParametersListAuxiliar

ParametersListAuxiliar = ´,´ ParametersList | ε

CommandsBlock = Command CommandsBlockAuxiliar

CommandsBlockAuxiliar = CommandsBlock | ε

Command = FunctionCall | AttributionExpression | ConcatenationExpression | Declaration | Condition | Loop | Output | Input | AppendList | Return

FunctionCall = ´Identificator´ ´{´ IdList ´}´

IdList = ´Identificator´ IdListAuxiliar

IdListAuxiliar = ´,´ IdList | ε

AttributionExpression = ´Identificator´ ´=´ AhritimeticExpression | VariableDeclaration ´=´ AhritimeticExpression

AhritimeticExpression = AddtiveExpression AhritimeticAuxiliar

AhritimeticAuxiliar = BitwiseOperation AddtiveExpression AhritimeticAuxiliar | ε

AddtiveExpression = MultiplicativeExpression AddtiveAuxiliar

AddtiveAuxiliar = AddtiveOperation MultiplicativeExpression AddtiveAuxiliar | ε

MultiplicativeExpression = AhritimeticValue MultiplicativeAuxiliar

MultiplicativeAuxiliar = MultiplicativeOperation AhritimeticValue MultiplicativeAuxiliar | ε

AhritimeticValue = AhritimeticFactor AhritimeticValueAuxiliar

AhritimeticValueAuxiliar = IncrementOperation | ε

AhritimeticFactor = Value | FunctionCall

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 = AhritimeticExpression BooleanRelationAuxiliar

BooleanRelationAuxiliar = LogicalRelation AhritimeticExpression | 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 = LogicForStatement | CounterForStatement | WhileStatement

LogicForStatement = ´for´ ´{´ AttributionExpression ´;´ BooleanExpression ´;´
AhritimeticExpression ´}´ ´[´ CommandsBlock ´]´

CounterForStatement = ´for´ ´{´ ´Identificator´ ´,´ IntValue ´,´ IntValue ´,´ IntValue
CounterForStatementAuxiliar

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

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

LogicalRelation = `RelationEqual´ | ´RelationNotEqual´ | ´RelationGreater´ |
´RelationLower´ | ´RelationGreaterEqual´ | ´RelationLowerEqual´

Output = ´write´ ´{´ OutputValues ´}´

OutputValues = Value OutputValuesAuxiliar

OutputValuesAuxiliar = ´,´ OutputValues | ε

Input = ´read´ ´{´ IdList ´}´

AppendList = ´Identificator´ ´.´ ´append´ ´{´ Value ´}´

Return = ‘return’ Value

Value = ´Identificator´ | ´CharConst´ | ´StringConst´ | ´DoubleConst´ | ´IntConst´ | ´BooleanConst´

IntValue = ´Identificator´ | ´IntConst