Universidade Federal de Alagoas
Instituto de Computação
Bacharelado em Ciência da Computação

Especificação da Gramática - Isengard++

Márcio Henrique Vieira de Oliveira
Michael Miller Rodrigues Cardoso

Maceió - AL, 2021

Sumário

1. Tipo de Analisador Sintático

O tipo de analisador sintático escolhido para o presente trabalho foi o analisador descendente LL(1) preditivo recursivo.

2. Gramática livre de contexto

S = DcMain | DcFun S | DcId S | ε

DcMain = ‘Funct’ ‘Int’ ‘Main’ ‘(‘ ‘)’ BlockDc

DcFun = ‘Funct’ FunType ‘id’ ‘(‘ Param ‘)’ BlockDc

FunType = ‘Void’ | VarType

VarType = ‘Int’ | ‘Float’ | ‘Char’ | ‘Str’ | ‘Bool’

Param = ParamDc | ε

ParamDc = VarType ‘id’ Vet ‘,’ ParamDc | VarType ‘id’ Vet

BlockDc = ‘Begin’ Instructions ‘End’

Vet = ‘[’ ‘]’ | ε

Instructions = DcId Instructions | Command Instructions | CommandIO Instructions | FunCall Instructions | AtrDir Instructions | Return Instructions | ε

DcId = VarType DcIdAtr ‘;’

DcIdAtr = Id | Id ‘,’ DcIdAtr | Atr | Atr ‘,’ DcIdAtr

Id = ‘id’ ‘[‘ Ea ‘]’ | ‘id’

Atr = ‘id’ ‘[‘ Ea ‘]’ ‘=’ ‘[‘ AtrVet ‘]’ | ‘id’ ‘=’ Ec

AtrVet = Ec ‘,’ AtrVet | Ec

AtrDir = ‘id’ ‘[‘ Ea ‘]’ ‘=’ Ec ‘;’ | ‘id’ ’=’ Ec ‘;’

Command = IfElse | While | For

CommandIO = Input | Output

FunCall = ‘id’ ‘(‘ ParamFun ‘)’ ‘;’

ParamFun = ParamFun ‘,’ Ec | Ec | ε

Return = ‘Return’ Ec ‘;’

IfElse = ‘If’ ‘(’ Eb ‘)’ BlockDc | ‘If’ ‘(’ Eb ‘)’ BlockDc ‘Else’ BlockDc

While = ‘While’ ‘(’ Eb ‘)’ BlockDc

For = ‘For’ ‘(‘ DcInt ‘,’ IntValue ‘,’ IntValue ‘)’ BlockDc | ‘For’ ‘(‘ DcInt ‘,’ IntValue ‘)’ BlockDc

DcInt = ‘Int’ ‘id’ | ‘id’

IntValue = ‘id’ | ‘CT_INT’

Input = ‘Input’ ‘(’ InputParam ‘)’ ‘;’

InputParam = ‘id’ | ‘id’ ‘,’ InputParam

Output = ‘Output’ ‘(’ OutputParam ‘)’ ‘;’ | ‘OutputIn’ ‘(’ OutputParam ‘)’ ‘;’

OutputParam = Ec | Ec ‘,’ OutputParam

Ec = Ec ‘OP_CONCAT’ Eb | Eb

Eb = Eb ‘PR_OR’ Tb | Eb ‘PR_AND’ Tb | Tb

Tb = Tb ‘PR_NOT’ Ra | Ra

Ra = Ra Rel Rb | Rb

Rb = Rb Ops Ea | Ea

Ea = Ea ‘OP_AD’ Ta | Ea ‘OP_SUB’ Ta | Ta

Ta = Ta ‘OP_MULT’ Fa | Ta ‘OP_DIV’ Fa | Ta ‘OP_RES’ Fa | Fa

Fa = ‘(‘ Ec ‘)’ | Id | FunCall | ‘CT_INT’ | ‘CT_FLOAT’ | ‘PR_TRUE’ | ‘PR_FALSE’ | ‘CT_CHAR’ | ‘CT_STR’ | ‘OP_NOTUNI’ ‘id’ | ‘OP_SIZE’ ‘id’

Rel = ‘OP_RELEQUAL’ | ‘OP_RELDIF’

Ops = ‘OP_GREATER’ | ‘OP_LESS’ | ‘OP_GREATERT’ | ‘OP_LESST’

3. Gramática LL(1)

S = ‘Funct’ DcFun S | DcId S | ε

DcFun = FunTypeMinusInt ‘id’ ‘(‘ Param ‘)’ BlockDc | ‘Int’ DcIntFunMain

FunTypeMinusInt = ‘Void’ | ‘Float’ | ‘Char’ | ‘Str’ | ‘Bool’

DcIntFunMain = ‘Main’ ‘(‘ ‘)’ BlockDc | ‘id’ ‘(‘ Param ‘)’ BlockDc

VarType = ‘Int’ | ‘Float’ | ‘Char’ | ‘Str’ | ‘Bool’

Param = ParamDc | ε

ParamDc = VarType ‘id’ Vet ParamDcFat

ParamDcFat = ‘,’ ParamDc | ε

BlockDc = ‘Begin’ Instructions ‘End’

Vet = ‘[’ ‘]’ | ε

Instructions = DcId Instructions | Command Instructions | CommandIO Instructions | ‘id’ AtrDirFunCall Instructions | Return Instructions | ε

DcId = VarType DcIdAtr ‘;’

DcIdAtr = ‘id’ Id Atr DcIdAtrFat

DcIdAtrFat = ‘,’ DcIdAtr | ε

Id = ‘[‘ Ea ‘]’ | ε

Atr = ‘=’ AtrFat | ε

AtrFat = ‘[‘ AtrVet ‘]’ | Ec

AtrVet = Ec AtrVetFat

AtrVetFat = ‘,’ AtrVet | ε

Command = IfElse | While | For

CommandIO = Input | Output

AtrDirFunCall = AtrDir | FunCall

AtrDir = ‘[‘ Ea ‘]’ ‘=’ Ec ‘;’ | ’=’ Ec ‘;’

FunCall = ‘(‘ ParamFun ‘)’ ‘;’

IdFunCall = Id | FunCall

ParamFun = Ec ParamFunLL

ParamFunLL = ‘,’ ParamFun | ε

Return = ‘Return’ Ec ‘;’

IfElse = ‘If’ ‘(’ Eb ‘)’ BlockDc IfElseFat

IfElseFat = ‘Else’ BlockDc | ε

While = ‘While’ ‘(’ Eb ‘)’ BlockDc

For = ‘For’ ‘(‘ DcInt ‘,’ IntValue ForFat

ForFat = ‘,’ IntValue ‘)’ BlockDc | ‘)’ BlockDc

DcInt = ‘Int’ ‘id’ ‘=’ IntValue | ‘id’ AtrInt

IntValue = ‘id’ | ‘CT_INT’

AtrInt = ‘=’ IntValue | ε

Input = ‘Input’ ‘(’ InputParam ‘)’ ‘;’

InputParam = ‘id’ Id InputParamFat

InputParamFat = ‘,’ InputParam | ε

Output = ‘Output’ ‘(’ OutputParam ‘)’ ‘;’ | ‘OutputIn’ ‘(’ OutputParam ‘)’ ‘;’

OutputParam = Ec OutputParamFat

OutputParamFat = ‘,’ OutputParam | ε

Ec = Eb EcLL

EcLL = ‘OP_CONCAT’ Eb EcLL | ε

Eb = Tb EbLL

EbLL = ‘PR_OR’ Tb EbLL | ‘PR_AND’ Tb EbLL | ε

Tb = Ra TbLL

TbLL = ‘PR_NOT’ Ra TbLL | ε

Ra = Rb RaLL

RaLL = Rel Rb RaLL | ε

Rb = Ea RbLL

RbLL = Ops Ea RbLL | ε

Ea = Ta EaLL

EaLL = ‘OP_AD’ Ta EaLL | ‘OP_SUB’ Ta EaLL | ε

Ta = Fa TaLL

TaLL = ‘OP_MULT’ Fa TaLL | ‘OP_DIV’ Fa TaLL | ‘OP_RES’ Fa TaLL | ε

Fa = ‘(‘ Ec ‘)’ | ‘id’ IdFunCall | ‘CT_INT’ | ‘CT_FLOAT’ | ‘PR_TRUE’ | ‘PR_FALSE’ | ‘CT_CHAR’ | ‘CT_STR’ | ‘OP_NOTUNI’ ‘id’ | ‘OP_SIZE’ ‘id’ | ε

Rel = ‘OP_RELEQUAL’ | ‘OP_RELDIF’

Ops = ‘OP_GREATER’ | ‘OP_LESS’ | ‘OP_GREATERT’ | ‘OP_LESST’