PRÁCTICA PL: SEGUNDA FASE II

Tiny(1)

Integrantes:

David Davó Laviña

Ela Katherine Shepherd Arévalo Grupo 12

PASO 1: Especificación sintáctica

Los nombres subrayados son clases léxicas determinadas en la especificación léxica. El símbolo inicial o raíz de la gramática es `Programa.`

```
Programa -> Decs Insts
Decs -> ε
Decs -> LDecs &&
LDecs -> Dec
LDecs -> LDecs ; Dec
Dec -> var TypN Identificador
Dec -> type TypN Identificador
Dec -> proc Identificador ParamsF Bloq
ParamsF -> ( LParamFOpc )
LParamFOpc -> ε
LParamFOpc -> LParamF
LParamF -> ParamF
LParamF -> LParamF , ParamF
ParamF -> TypN & Identificador
ParamF -> TypN Identificador
TypN -> BaseType
TypN -> array [ LitEnt ] of BaseType
TypN -> record { LCampos }
TypN -> pointer BaseType
LCampos -> Campo
LCampos -> LCampos ; Campo
Campo -> TypN Identificador
BaseType -> BasicType
BaseType -> Identificador
BasicType -> int
BasicType -> realw
BasicType -> bool
BasicType -> string
Insts -> Inst
Insts -> Insts ; Inst
Inst -> E0 = E0
Inst -> if E0 then LInsts endif
Inst -> if E0 then LInsts else Linsts endif
Inst -> while E0 do LInsts endwhile
Inst -> read E0
Inst -> write E0
Inst-> nl
Inst -> new E0
Inst -> delete E0
Inst -> call Identificador ParamsR
```

```
Inst -> Bloq
LInsts -> ε
LInsts -> Insts
ParamsR -> ( LParamROpc )
LParamROpc -> ε
LParamROpc -> LParamR
LParamR -> E0
LParamR -> LParamR , E0
Bloq -> { BloqOpc }
BloqOpc -> ε
BloqOpc -> Programa
E0 -> E1 OpIn0AsocD E0
E0 -> E1 OpIn0NoAsoc E1
E0 -> E1
E1 -> E1 OpIn1AsocI E2
E1 -> E2
E2 -> E2 OpIn2AsocI E3
E2 -> E3
E3 -> E4 OpIn3NoAsoc E4
E3 -> E4
E4 -> OpPre4NoAsoc E5
E4 -> OpPre4Asoc E4
E4 -> E5
E5 -> E5 OpPos5Asoc
E5 -> E6
E6 -> OpPre6Asoc E6
E6 -> E7
E7 -> ( E0 )
E7 -> LitEnt
E7 -> LitReal
E7 -> LitCad
E7 -> true
E7 -> false
E7 -> Identificador
E7 -> null
OpIn0AsocD -> +
OpIn0NoAsoc-> -
OpIn1AsocI -> and
OpIn1AsocI -> or
OpIn2AsocI -> <
OpIn2AsocI -> <=
OpIn2AsocI -> >
OpIn2AsocI -> >=
OpIn2AsocI -> ==
OpIn2AsocI -> !=
OpIn3NoAsoc -> *
OpIn3NoAsoc -> /
OpIn3NoAsoc -> %
OpPre4NoAsoc -> -
```

```
OpPre4Asoc -> not
OpPos5Asoc -> [ E0 ]
OpPos5Asoc -> .Identificador
OpPos5Asoc -> ->Identificador
OpPre6Asoc -> *
```

PASO 2: Acondicionamiento de la gramática

En el paso anterior, las reglas en negrita tienen recursión a izquierdas y las reglas en cursiva tienen un factor común.

```
Programa -> Decs Insts
Decs -> ε
Decs -> LDecs &&
LDecs -> Dec RLDecs
RLDecs -> ; Dec RLDecs
RLDecs -> ε
Dec -> var TypN Identificador
Dec -> type TypN Identificador
Dec -> proc Identificador ParamsF Bloq
ParamsF -> ( LParamFOpc )
LParamFOpc -> ε
LParamFOpc -> LParamF
LParamF -> ParamF RLParamF
RLParamF -> ε
RLParamF -> , ParamF RLParamF
ParamF -> TypN RParamF
RParamF -> & Identificador
RParamF -> Identificador
TypN -> BaseType
TypN -> array [ LitEnt ] of BaseType
TypN -> record { LCampos }
TypN -> pointer BaseType
LCampos -> Campo RLCampos
RLCampos -> ε
RLCampos -> ; Campo RLCampos
Campo -> TypN Identificador
BaseType -> BasicType
BaseType -> Identificador
BasicType -> int
BasicType -> realw
BasicType -> bool
BasicType -> string
Insts -> Inst RInsts
RInsts -> ; Inst RInsts
RInsts -> ε
```

```
Inst -> E0 = E0
Inst -> if E0 then LInsts RInst
RInst -> endif
RInst -> else LInsts endif
Inst -> while E0 do LInsts endwhile
Inst -> read E0
Inst -> write E0
Inst-> nl
Inst -> new E0
Inst -> delete E0
Inst -> call Identificador ParamsR
Inst -> Bloq
LInsts -> ε
LInsts -> Insts
ParamsR -> ( LParamROpc )
LParamROpc -> ε
LParamROpc -> LParamR
LParamR -> E0 RLParamR
RLParamR -> ε
RLParamR -> , E0 RLParamR
Bloq -> { BloqOpc }
BloqOpc -> ε
BloqOpc -> Programa
E0 -> E1 RE0
RE0 -> OpIn0AsocD E0
RE0 -> OpIn0NoAsoc E1
RE0 -> ε
E1 -> E2 RE1
RE1 -> OpIn1AsocI E2 RE1
RE1 -> ε
E2 -> E3 RE2
RE2 -> OpIn2AsocI E3 RE2
RE2 -> ε
E3 -> E4 RE3
RE3 -> OpIn3NoAsoc E4
RE3 -> ε
E4 -> OpPre4NoAsoc E5
E4 -> OpPre4Asoc E4
E4 -> E5
E5 -> E6 RE5
RE5 -> OpPos5Asoc RE5
RE5 -> ε
E6 -> OpPre6Asoc E6
E6 -> E7
E7 -> ( E0 )
E7 -> LitEnt
E7 -> LitReal
E7 -> LitCad
E7 -> true
```

```
E7 -> false
E7 -> Identificador
E7 -> null
OpIn0AsocD -> +
OpIn0NoAsoc-> -
OpIn1AsocI -> and
OpIn1AsocI -> or
OpIn2AsocI -> <
OpIn2AsocI -> <=
OpIn2AsocI -> >
OpIn2AsocI -> >=
OpIn2AsocI -> ==
OpIn2AsocI -> !=
OpIn3NoAsoc -> *
OpIn3NoAsoc -> /
OpIn3NoAsoc -> %
OpPre4NoAsoc -> -
OpPre4Asoc -> not
OpPos5Asoc -> [ E0 ]
OpPos5Asoc -> .Identificador
OpPos5Asoc -> ->Identificador
OpPre6Asoc -> *
```

La clase léxica de la palabra reservada “real” tiene nombre realw(de realword) porque ya existe una definición auxiliar llamada “real” en la especificación léxica.