Fase 2: Desarrollo de analizador sintáctico para Tiny(0) y Tiny

G01
Esther Babon Arcauz
Pablo Campo Gómez
Claudia López-Mingo Moreno
José Antonio Ruiz Heredia

Índice:

1. Desarrollo manual de un analizador sintáctico para Tiny(0).

1.1. Especificación sintáctica (gramática) para Tiny(0).

programa → bloque EOF

bloque → { declaraciones_opt instrucciones_opt }
declaraciones_opt → declaraciones &&
declaraciones_opt → ε
istrucciones_opt → instrucciones
instrucciones_opt → ε

declaraciones → declaracion_var ; declaraciones
declaraciones → declaracion_var

declaracion_var → tipo identificador
tipo → int
tipo → real
tipo → bool

instrucciones → instrucciones ; instruccion
instrucciones → instruccion

instruccion → @ E0

E0 → E1 = E0
E0 → E1
E1 → E1 op1 E2
E1 → E2
E2 → E2 + E3
E2 → E3 - E3
E2 → E3
E3 → E4 op3
E3 → E4
E4 → E4 op4 E5
E4 → E5
E5 → op5 E5
E5 → literalEntero
E5 → literalReal
E5 → literalBool
E5 → variable
E5 → (E0)
op1 → >

op1 → >=
op1 → <
op1 → <=
op1 → ==
op1 → !=
op3 → and E3
op3 → or E4
op4 → *
op4 → /
op5 → -
op5 → not

1.2. Acondicionamiento de la gramática para permitir la implementación de un analizador sintáctico descendente predictivo recursivo.

Original	Acondicionado
programa → bloque EOF	
bloque → { declaraciones_opt instrucciones_opt }	
declaraciones_opt → declaraciones && declaraciones_opt → ε	
instrucciones_opt → instrucciones instrucciones_opt → ε	
declaraciones → declaracion_var ; declaraciones declaraciones → declaracion_var	declaraciones → declaracion_var restodec restodec → ; declaraciones restodec → ε
declaracion_var → tipo identificador	
tipo → int tipo → real tipo → bool	
instrucciones → instrucciones ; instruccion instrucciones → instruccion	Instrucciones → Instruccion restoInst restoInst → ; Instruccion restoInst restoInst → ε
instruccion → @ E0	
E0 → E1 = E0	E0 → E1 RE0

E0 → E1	RE0 → = E0 RE0 → ε
E1 → E1 op1 E2 E1 → E2	E1 → E2 RE1 RE1 → op1 E2 RE1 RE1 → ε
E2 → E2 + E3 E2 → E3 - E3 E2 → E3	E2 → E3 RE2 E2' RE2 → - E3 RE2 → ε E2' → + E3 E2' E2' → ε
E3 → E4 op3 E3 → E4	E3 → E4 RE3 RE3 → op3 RE3 → ε
E4 → E4 op4 E5 E4 → E5	E4 → E5 RE4 RE4 → op4 E5 RE4 RE4 → ε
E5 → op5 E5 E5 → literalEntero E5 → literalReal E5 → literalBool E5 → variable E5 → (E0)	
op1 → > op1 → >= op1 → < op1 → <= op1 → == op1 → != op3 → and E3 op3 → or E4 op4 → * op4 → / op5 → - op5 → not	

1.3. Directores de cada regla de la gramática acondicionada.

programa → bloque EOF
- Director: **{**

bloque → { declaraciones_opt instrucciones_opt }
- Director: **{**

declaraciones_opt → declaraciones &&
- Dir: **int**, **real**, **bool**

declaraciones_opt → ε
- Dir: ∅

instrucciones_opt → instrucciones
- Dir: **@**

instrucciones_opt → ε
- Dir: ∅

declaraciones → declaracion_var restodec
- Dir: **int**, **real**, **bool**

restodec → ; declaraciones
- Dir: **;**

restodec → ε
- Dir: ∅

declaracion_var → tipo identificador
- Dir: **int**, **real**, **bool**

tipo → int
- Dir: **int**

tipo → real
- Dir: **real**

tipo → bool
- Dir: **bool**

Instrucciones → Instruccion restoInst

- Dir: **@**

restoInst → ; Instruccion restoInst
- Dir: **;**

restoInst → ε
- Dir: ∅

instruccion → @ E0
- Director: **@**

E0 → E1 RE0
- Dir: **-,not, literalEntero, literalReal, literalBool, variable, (**

RE0 → = E0
- Dir: **=**

RE0 → ε
- Dir: ∅

E1 → E2 RE1
- Dir: **-,not, literalEntero, literalReal, literalBool, variable, (**

RE1 → op1 E2 RE1
- Dir: **>, >=, <, <=, ==, !=**

RE1 → ε
- Dir: ∅

E2 → E3 RE2 E2'
- Dir: **-, not, literalEntero, literalReal, literalBool, variable, (**

RE2 → - E3
- Dir: **-**

RE2 → ε
- Dir: ∅

E2' → + E3 E2'
- Dir: **+**

E2' → ε
- Dir: ∅

E3 → E4 RE3
- Dir: **-, not, literalEntero, literalReal, literalBool, variable, (**

RE3 → op3
- Dir: **and, or**

RE3 → ε
- Dir: ∅

E4 → E5 RE4
- Dir: **-, not, literalEntero, literalReal, literalBool, variable, (**

RE4 → op4 E5 RE4
- Dir: ***, /**

RE4 → ε
- Dir: ∅

E5 → op5 E5
- Dir: **-, not**

E5 → literalEntero
- Dir: **literalEntero**

E5 → literalReal
- Dir: **literalReal**

E5 → literalBool
- Dir: **literalBool**

E5 → variable
- Dir: **variable**

E5 → (E0)
- Dir: **(**

op1 → >
- Dir: **>**

op1 → >=
- Dir: **>=**

op1 →<
- Dir: **<**

op1 →**<=**

- Dir: **<=**

op1 → **==**

- Dir: **==**

op1 → **!=**

- Dir: **!=**

op3 → and E3

- Dir: **and**

op3 → or E4

- Dir: **or**

op4 → *

- Dir: *****

op4 → /

- Dir: **/**

op5 → -

- Dir: **-**

op5 → not:

- Dir: **not**

2. Desarrollo de analizadores sintácticos descendentes y ascendentes para Tiny.

2.1. Especificación sintáctica (gramática) para Tiny.

programa → bloque EOF

bloque → { declaraciones_opt instrucciones_opt }
declaraciones_opt → declaraciones &&
declaraciones_opt → ε

istrucciones_opt → instrucciones
instrucciones_opt → ε

declaraciones → declaraciones ; declaracion
declaraciones → declaracion

declaracion → declaracion_var
declaracion → declaracion_tipo
declaracion → declaracion_proc

declaracion_var → tipo identificador
declaracion_tipo → type tipo identificador
declaracion_proc → proc identificador (parametros_formales_opt) bloque

parametros_formales_opt → parametros_formales
parametros_formales_opt → ε

parametros_formales → parametros_formales , parametro_formal
parametros_formales → parametro_formal

parametro_formal → tipo and_opt identificador
and_opt → &
and_opt → ε

tipo → tipo [literalEntero]
tipo → tipo1
tipo1 → ^ tipo1
tipo1 → tipo2
tipo2 → identificador
tipo2 → struct { campos }
tipo2 → int
tipo2 → real
tipo2 → bool
tipo2 → string

campos → campos , campo
campos → campo

campo → tipo identificador

instrucciones → instrucciones ; instruccion
instrucciones → instruccion

instruccion → @ E0
instruccion → if E0 bloque
instruccion → if E0 bloque else bloque
instruccion → while E0 bloque
instruccion → read E0
instruccion → write E0
instruccion → nl
instruccion → new E0
instruccion → delete E0
instruccion → call identificador (parametros_reales_opt)
instruccion → bloque

parametros_reales_opt -> parametros_reales
parametros_reales_opt → ε

parametros_reales -> parametros_reales , E0
parametros_reales -> E0

E0 → E1 = E0
E0 → E1
E1 → E1 op1 E2
E1 → E2
E2 → E2 + E3
E2 → E3 - E3
E2 → E3
E3 → E4 op3
E3 → E4
E4 → E4 op4 E5
E4 → E5
E5 → op5 E5
E5 → E6
E6 → E6 op6
E6 → literalEntero
E6 → literalReal
E6 → literalBool
E6 → literalCadena
E6 → identificador
E6 → null
E6 → (E0)
op1 → >
op1 → >=
op1 → <
op1 → <=

op1 → ==
op1 → !=
op3 → and E3
op3 → or E4
op4 → *
op4 → /
op4 → %
op5 → -
op5 → not
op6 → [E0]
op6 → .identificador
op6 → ^

2.2. Acondicionamiento de la gramática para permitir la implementación de un analizador sintáctico descendente predictivo recursivo.

Original	Acondicionado
programa → bloque EOF	
bloque → { declaraciones_opt instrucciones_opt } declaraciones_opt → declaraciones && declaraciones_opt → ε instrucciones_opt → instrucciones instrucciones_opt → ε	
declaraciones → declaraciones ; declaracion declaraciones → declaracion declaracion → declaracion_var declaracion → declaracion_tipo declaracion → declaracion_proc	declaraciones → declaracion declaracion_extra declaracion_extra → ; declaraciones declaracion_extra → ε declaracion → declaracion_var declaracion → declaracion_tipo declaracion → declaracion_proc
declaracion_var → tipo identificador declaracion_tipo → type tipo identificador declaracion_proc → proc identificador (parametros_formales_opt) bloque	
parametros_formales_opt → parametros_formales parametros_formales_opt → ε	

parametros_formales → parametros_formales , parametro_formal parametros_formales → parametro_formal	parametros_formales → parametro_formal parametros_formales' parametros_formales' → , parametro_formal parametros_formales' parametros_formales' →ε
parametro_formal → tipo and_opt identificador and_opt → & and_opt → ε	
tipo → tipo [literalEntero] tipo → tipo1 tipo1 → ^ tipo1 tipo1 → tipo2 tipo2 → identificador tipo2 → struct { campos } tipo2 → int tipo2 → real tipo2 → bool tipo2 → string	tipo → tipo1 tipo' tipo' → [literalEntero] tipo' tipo' → ε tipo1 → ^ tipo1 tipo1 → tipo2 tipo2 → identificador tipo2 → struct { campos } tipo2 → int tipo2 → real tipo2 → bool tipo2 → string
campos → campos , campo campos → campo	campos → campo campos' campos' → , campo campos' campos' → ε
campo → tipo identificador	
Instrucciones → instrucciones ; instruccion Instrucciones → instruccion	Instrucciones → instruccion instrucciones' instrucciones' → ; instruccion instrucciones' instrucciones' → ε
instruccion → @ E0 instruccion → if E0 bloque instruccion → if E0 bloque else bloque instruccion → while E0 bloque instruccion → read E0 instruccion → write E0 instruccion → nl instruccion → new E0 instruccion → delete E0	instruccion → if E0 bloque if_select if_select → else bloque if_else → ε instruccion → @ E0 instruccion → while E0 bloque instruccion → read E0 instruccion → write E0 instruccion → nl instruccion → new E0 instruccion → delete E0

instruccion → call identificador (parametros_reales_opt) instruccion → bloque	instruccion → call identificador (parametros_reales_opt) instruccion → bloque
parametros_reales_opt → parametros_reales parametros_reales_opt → ε	
parametros_reales → parametros_reales , E0 parametros_reales → E0	parametros_reales → E0 parametros_reales' parametros_reales' → , E0 parametros_reales' parametros_reales' → ε
E0 → E1 = E0 E0 → E1	E0 → E1 RE0 RE0 → = E0 RE0 → ε
E1 → E1 op1 E2 E1 → E2	E1 → E2 RE1 RE1 → op1 E2 RE1 RE1 → ε
E2 → E2 + E3 E2 → E3 - E3 E2 → E3	E2 → E3 RE2 E2' RE2 → - E3 RE2 → ε E2' → + E3 E2' E2' → ε
E3 → E4 and E3 E3 → E4 or E4 E3 → E4	E3 → E4 RE3 RE3 → op3 RE3 → ε
E4 → E4 op4 E5 E4 → E5	E4 → E5 RE4 RE4 → op4 E5 RE4 RE4 → ε
E5 → op5 E5 E5 → E6	
E6 → E6 op6 E6 → literalEntero E6 → literalReal E6 → literalBool E6 → literalCadena E6 → identificador E6 → null E6 → (E0)	E6 → RE6 E6' E6' → op6 E6' E6' → ε RE6 → literalEntero RE6 → literalReal RE6 → literalBool RE6 → literalCadena RE6 → identificador

	RE6 → null RE6 → (E0)
op1 → > op1 → >= op1 → < op1 → <= op1 → == op1 → != op3 → and E3 op3 → or E4 op4 → * op4 → / op4 → % op5 → - op5 → not op6 → [E0] op6 → . identificador op6 → ^	