Práctica PL

Elena Kaloyanova Popova y Álvaro Borja Velasco García

2018

Índice general

Capítulo 1

Introducción

Esta práctica consistirá en el desarrollo de un procesador de lenguajes sobre el siguiente lenguaje:

Capítulo 2

Fase 1: Analizador léxico

2.1. Clases Léxicas

Todo programa consta de dos secciones: una para las declaraciones y otra para las instrucciones, separadas por un token «&&». La sección de declaraciones está formada por una serie de declaraciones compuestas por el nombre de tipo y el de variable y separadas por un punto y coma. La sección de instrucciones, por su parte, consta de una serie de asignaciones (variable=expresión), separadas también por un punto y coma. Las clases léxicas que hemos considerado para representar los tokens del lenguaje son las siguientes:

- **SEC:** Representa el seccionador de las dos partes del programa («&&»).
- **TIPO:** Representa el nombre del tipo. Puede ser únicamente o «num» o «bool».
- **NUM:** Palabra reservada «num».
- **BOOL:** Palabra reservada «bool».
- **VAR:** Representa el nombre de la variable. Comienza necesariamente por una letra, seguida por una secuencia de cero o más letras, dígitos o el símbolo «_».
- **ASIG:** Representa el signo igual de las asignaciones.
- **EXPR:** Representa una expresión. Puede ser básica o compuesta, según si aparece un único elemento o dos acompañados de un operador.

- **BASICA:** Se corresponde a una expresión básica. Puede ser un número real con o sin signo, o bien un booleano,«true» o «false». También puede ser el nombre de una variable que se correspondería a un número.
- **TRUE:** Palabra reservada «true».
- **FALSE:** Palabra reservada «false».
- **COMP:** Se corresponde a una expresión compuesta que en el fondo no es más que dos básicas separadas por un operador.
- **NUM:** Representa un número real. Puede empezar opcionalmente con un signo seguido de una secuencia de uno o más digitos cualesquiera, pudiendo poner ceros no significativos a la izquierda. Puede opcionalmente estar seguido por una parte decimal y/o una parte exponencial.
- **SIGNO:** Puede ser el signo + o el signo -.
- **DIG:** Cualquier dígito del 0 al 9.
- **DEC** Parte decimal que consta de un punto seguido por una sucesión de uno o más dígitos.
- **EX:** Parte exponencial que consta de un letra E mayúscula o minúscula seguida, opcionalmente, por un signo + o -, y obligatoriamente por una parte entera.
- **OP:** Representa un operador que puede ser aritmético, lógico o relacional.
- **OPAR:** Operador aritmético, es decir: un signo (+ o -), * o /.
- **OPLOG:** Operador lógico, es decir: «and», «or» o «not».
- **AND:** Palabra reservada «and».
- **OR:** Palabra reservada «or».
- **NOT:** Palabra reservada «not».
- **OPREL:** Operador relacional, es decir: <,>,<=,>=,== o !=.
- **PAP:** Signo de apertura de paréntesis.
- **PCI:** Signo de cierre de paréntesis.

- **SEP:** Separadores: espacio en blanco, tabulador, retorno de carro, salto de línea y backspace.

2.2. Especificación Formal

Las definiciones regulares que definen el lenguaje de forma general son las siguientes:

- **PROGRAMA:** DECL.SEC.INSTR
- **DECLARACION:** (TIPO.SEP.VAR;)*|(TIPO.SEP.VAR)
- **INSTRUCCION:** VAR=EXPR

Las definiciones regulares correspondientes a las clases léxicas definidas son:

- **SEC:** &&
- **TIPO:** (NUM|BOOL)
- **NUM:** ([n][u][m])
- **BOOL:** ([b][o][o][l])
- **VAR:** ([a-z,A-Z][a-z,A-Z,0-9,_]*)
- **EXPR:** (BASICA.OP.BASICA | BASICA.OP.COMP | COMP.OP.COMP | COMP.OP.BASICA)
- **BASICA:** [VAR,NUM,true,false]
- **TRUE:** ([t][r][u][e])
- **FALSE:** ([f][a][l][s][e])
- **COMP:** BASICA.OP.BASICA
- **NUM:** SIGNO?(DIG+(DEC)?(EXP)?)
- **SIGNO:** [+,-]
- **DIG:** [0-9]
- **DEC:** .(DIG+)

- **EX:** [e|E](SIGNO?DIG+(DEC)?)
- **OP:** [OPAR,OPLOG,OPREL]
- **OPAR:** [SIGNO,*,/]
- **OPLOG:** (AND|OR|NOT)
- **AND:** ([a][n][d])
- **OR:** ([o][r])
- **NOT:** ([n][o][t])
- **OPREL:** [<,>,<=,>=,==,!=]
- **PAP:** («(»)
- **PCIERRE:** («)»)
- **SEP:** [BLANCO,TAB,ENDL,CARRO,BACK]

2.3. Diseño

El autómata que reconocería el lenguaje es el siguiente: