PROCESADORES DE

LENGUAJES

Práctica Obligatoria

Primera entrega

GRADO EN INGENIERÍA INFORMÁTICA

ETSII



Grupo 30

Gonzalo de la Cruz Cepeda

Álvaro Brea Arjona

Contenido

[Introducción: 3](#_Toc70161675)

[Recuperación de errores: 3](#_Toc70161676)

[Trabajo realizado: 3](#_Toc70161677)

[Casos de prueba: 4](#_Toc70161678)

[Caso1A: 4](#_Toc70161679)

[Caso1B: 5](#_Toc70161680)

[Caso2A: 6](#_Toc70161681)

[Caso2B: 7](#_Toc70161682)

[Caso3A: 7](#_Toc70161683)

[Caso3B: 8](#_Toc70161684)

[Caso4A: 9](#_Toc70161685)

[Caso4B: 10](#_Toc70161686)

[Anexo: 11](#_Toc70161687)

[Cabeceras: 11](#_Toc70161688)

[Siguientes: 12](#_Toc70161689)

[Directores: 12](#_Toc70161690)

# Introducción:

La práctica consiste en implementación de un visualizador de código fuente para un lenguaje de programación.

Esto se llevará a cabo partiendo de un fichero de texto de entrada con el código, y terminará con un fichero HTML en el que aparecerá dicho código, mostrado de forma correcta, añadiendo tabulaciones y haciendo uso de las reglas de diseño en la programación.

Para llevar a cabo dicha tarea, haremos uso del lenguaje ANTLR v4, Java y HTML.

# Gestión de errores:

Para este apartado, hemos añadido una nueva clase llamada errorListener.java para sustituir el errorListener que actuaba por defecto. En él hemos seguidos los pasos explicados en el libro de ANTLR recomendado por la asignatura. Hemos añadido un par de métodos, uno que nos señalase la ubicación del error, otro que nos tradujese los cuatro tipos de errores mencionados en el libro.

En cuanto a la recuperación de errores, no hemos modificado ni añadido nada ya que el propio ANTLR nos proporciona estas herramientas. Aun así, hay diversos errores de los que no se recupera con tanta facilidad, los cuales trataremos de solventar en la siguiente entrega.

# Trabajo realizado:

Para facilitar el trabajo en grupo y más de forma remota, hemos hecho uso de Github, y a continuación se encuentra una imagen del control de versiones del proyecto, en el que se han ido añadiendo las distintas partes del proyecto. En primer lugar, desarrollamos el analizador léxico para que nuestra gramática sea capaz de reconocer los identificadores, las constantes etc.

A continuación, añadimos la parte del analizador sintáctico que nos fue facilitado en el enunciado de la práctica. Empezando por la parte obligatoria seguida de la parte opcional. Este fragmento de código no cumplía las propiedades necesarias como para ser una gramática LL(1) ya que tenía recursividad por la izquierda y también requería de factorización. Así que con ayuda de las cabeceras y los directores que calculamos y que se incluyen más adelante en el anexo, hicimos de esta gramática, una gramática LL(1).

Mas adelante corregimos ciertos elementos que eran innecesarios y de esta manera se simplificaba el código.

Captura de pantalla de computadora

Descripción generada automáticamente

# Casos de prueba:

Pasa estos 8 casos de prueba, estarán separados por la mitad en la que una mitad estará dedicada a los casos de correcta estructura y funcionamiento, y los otros cuatro, estarán dedicados a mostrar los errores.

Cada caso estará tematizado para mostrar las distintas funcionalidades de nuestro programa.

## Caso1A:

Este primer caso de prueba estará dedicado a la declaración de variables.

Texto

Descripción generada automáticamente

## Caso1B:

Este caso de prueba consistirá en los errores partiendo del caso 1A.

Texto

Descripción generada automáticamente

## Caso2A:

En este segundo caso de prueba, se mostrarán ejemplos de operaciones y asignaciones.

Texto

Descripción generada automáticamente

## Caso2B:

Este caso de prueba consistirá en los errores partiendo del caso 2A.

Texto

Descripción generada automáticamente

## Caso3A:

Este caso de prueba consistirá en mostrar las distintas opciones del sent.

Texto

Descripción generada automáticamente

## Caso3B:

Este caso de prueba consistirá en los errores partiendo del caso 3A.

Texto

Descripción generada automáticamente

## Caso4A:

En este último caso de prueba, se dedicará a mostrar los comentarios.

Texto

Descripción generada automáticamente

## Caso4B:

Este caso de prueba consistirá en los errores partiendo del caso 4A.

Texto

Descripción generada automáticamente

# Anexo:

## Cabeceras:

CAB(program) = CAB'(part) U CAB'(part) = {"function"}  
  
CAB(part) = {"function", "procedimiento"}  
  
CAB(restpart) = {IDENTIFICADOR}  
  
CAB(listparam) = ̶C̶A̶B̶'̶(̶l̶i̶s̶t̶p̶a̶r̶a̶m̶)̶ U CAB'(type) = {"entero", "real", "caracter"}  
  
CAB(type) = {"entero", "real", "caracter"}  
  
CAB(blq) = {"inicio"}  
  
CAB(sentlist) = ̶C̶A̶B̶'̶(̶s̶e̶n̶t̶l̶i̶s̶t̶)̶ U CAB'(sent) =  
 {"entero", "real", "caracter", "inicio, "return", "bifurcacion", "buclepara", "buclemientras", "bucle"}  
  
CAB(sent) = CAB'(type) U CAB'(blq) U {IDENTIFICADOR} U {"return"} U {"bifurcacion"} U {"buclepara"} U {"buclemientras"} U {"bucle"} =  
 = {"entero", "real", "caracter", "inicio, "return", "bifurcacion", "buclepara", "buclemientras", "bucle"}  
  
CAB(lid) = {IDENTIFICADOR}  
  
CAB(asig) = {"="} U {"+="} U {"-="} U {"\*="} U {"/="} =  
 = {"=", "+=", "-=", "\*=", "/="}  
  
CAB(exp) = C̶A̶B̶'̶(̶e̶x̶p̶)̶ U {IDENTIFICADOR} U {"("} U {CONSTENTERO} U {CONSTREAL} U {CONSTLIT} =  
 {IDENTIFICADOR, "(", CONSTENTERO, CONSTREAL, CONSTLIT}  
  
CAB(op) = {"+"} U {"-"} U {"\*"} U {"/"} =  
 = {"+", "-", "\*", "/"}  
  
CAB(lcond) = ̶C̶A̶B̶'̶(̶l̶c̶o̶n̶d̶)̶ U CAB'(cond) U {"no"} =  
 = {IDENTIFICADOR, "(", CONSTENTERO, CONSTREAL, CONSTLIT} U {"no"} =  
 = {IDENTIFICADOR, "(", CONSTENTERO, CONSTREAL, CONSTLIT, "no"}  
  
CAB(cond) = CAB'(exp) U {"cierto"} U {"falso"} = {IDENTIFICADOR} U {"("} U {CONSTENTERO} U {CONSTREAL} U {CONSTLIT} =  
 = {IDENTIFICADOR, "(", CONSTENTERO, CONSTREAL, CONSTLIT}  
  
CAB(opl) = {"y"} U {"o"} = {"y", "o"}  
  
CAB(opr) = {"=="} U {"<>"} U {"<"} U {">"} U {">="} U {"<="} =  
 = {"==", "<>", "<", ">", ">=", "<="}

## Siguientes:

En cuanto a los siguientes, en este caso, no han hecho falta calcularlos, ya que, para la creación de los directores, al no haber λ, no serán necesarios.

## Directores:

Estos directores están hechos antes de corregir la recursividad por la izquierda, para resolver la factorización cuando había más de un símbolo de anticipación. También deberíamos haber hecho los directores de las partes en las que se ha arreglado la recursividad, pero no hizo falta ya que eran visibles de forma clara.

DIR(program ::= part program) = CAB'(part) = {"function", "procedimiento"}  
DIR(program ::= part) = CAB'(part) = {"function", "procedimiento"}  
  
DIR(part ::= "funcion" type restpart) = {function}  
DIR(part ::= "procedimiento" restpart) = {procedimiento}  
  
DIR(restpart ::= IDENTIFICADOR "(" listparam ")" blq) = {IDENTIFICADOR}  
DIR(restpart ::= IDENTIFICADOR "(" ")" blq) = {IDENTIFICADOR}  
  
DIR(listparam ::= listparam "," type IDENTIFICADOR) = {"entero", "real", "caracter"}  
DIR(listparam ::= type IDENTIFICADOR ) = {"entero", "real", "caracter"}  
  
DIR(type ::= "entero") = {"entero"}  
DIR(type ::= "real") = {"real"}  
DIR(type ::= "caracter") = {"caracter"}  
  
DIR(blq ::= "inicio" sentlist "fin") = {"inicio"}  
  
DIR(sentlist ::= sentlist sent) = {"entero", "real", "caracter", "inicio, "return", "bifurcacion", "buclepara", "buclemientras", "bucle"}  
DIR(sentlist ::= sent) = {"entero", "real", "caracter", "inicio, "return", "bifurcacion", "buclepara", "buclemientras", "bucle"}  
  
DIR(sent ::= type lid ";") = {"entero", "real", "caracter"}  
DIR(sent ::= IDENTIFICADOR asig exp ";") = {IDENTIFICADOR}  
DIR(sent ::= "return" exp ";") = {"return"}  
DIR(sent ::= IDENTIFICADOR "(" lid ")" ";") = {IDENTIFICADOR}  
DIR(sent ::= IDENTIFICADOR "(" ")" ";") = {IDENTIFICADOR}  
DIR(sent ::= "bifurcacion" "(" lcond ")" "entonces" blq "sino" blq) = {"bifurcacion"}  
DIR(sent ::= "buclepara" "(" IDENTIFICADOR asig exp ";" lcond ";" IDENTIFICADOR asig exp ")" blq ) = {"buclepara"}  
DIR(sent ::= "buclemientras" "(" lcond ")" blq ) = {"buclemientras"}  
DIR(sent ::= "bucle" blq "hasta" "(" lcond ")") = {"bucle"}  
DIR(sent ::= blq) = {"inicio"}  
  
DIR(lid ::= IDENTIFICADOR) = {IDENTIFICADOR}  
DIR(lid ::= IDENTIFICADOR "," lid) = {IDENTIFICADOR}  
  
DIR(asig ::= "=") = {"="}  
DIR(asig ::= "+=") = {"+="}  
DIR(asig ::= "-=") = {"-="}  
DIR(asig ::= "\*=") = {"\*="}  
DIR(asig ::= "/=") = {"/="}  
  
DIR(exp ::= exp op exp) = {IDENTIFICADOR, "(", CONSTENTERO, CONSTREAL, CONSTLIT}  
DIR(exp ::= IDENTIFICADOR "(" lid ")") = {IDENTIFICADOR}  
DIR(exp ::= "(" exp ")") = {"("}  
DIR(exp ::= IDENTIFICADOR) = {IDENTIFICADOR}  
DIR(exp ::= CONSTENTERO) = {CONSTENTERO}  
DIR(exp ::= CONSTREAL) = {CONSTREAL}  
DIR(exp ::= CONSTLIT) = {CONSTLIT}  
  
DIR(op ::= "+") = {"+"}  
DIR(op ::= "-") = {"-"}  
DIR(op ::= "\*") = {"\*"}  
DIR(op ::= "/") = {"/"}  
  
DIR(lcond ::= lcond opl lcond) = {IDENTIFICADOR, "(", CONSTENTERO, CONSTREAL, CONSTLIT, "no"}  
DIR(lcond ::= lcond) = {IDENTIFICADOR, "(", CONSTENTERO, CONSTREAL, CONSTLIT, "no"}  
DIR(lcond ::= "no" cond) = {"no"}  
  
DIR(cond ::= exp opr exp) = {IDENTIFICADOR, "(", CONSTENTERO, CONSTREAL, CONSTLIT}  
DIR(cond ::= "cierto") = {"cierto"}  
DIR(cond ::= "falso") = {"falso"}  
  
DIR(opl ::= "y") = {"y"}  
DIR(opl ::= "o") = {"o"}  
  
DIR(opr::= "==") = {"=="}  
DIR(opr::= "<>") = {"<>""}  
DIR(opr::= "<") = {"<"}  
DIR(opr::= ">") = {">"}  
DIR(opr::= ">=") = {">="}  
DIR(opr::= "<=") = {"<="}