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](#_Toc70068262)

[Recuperación de errores: 3](#_Toc70068263)

[Trabajo realizado: 3](#_Toc70068264)

[Casos de prueba: 4](#_Toc70068265)

[Cabeceras: 4](#_Toc70068266)

[Siguientes: 5](#_Toc70068267)

[Directores: 5](#_Toc70068268)

# 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.

# Recuperación de errores:

Mañana

# 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:

Añadir al terminar

# 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 para corregir la recursividad por la izquierda, para resolver la factorización, deberíamos crear unos nuevos 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::= "<=") = {"<="}