PRÁCTICA DE PROCESADORES DEL LENGUAJE I

Curso 2022 – 2023

Entrega de Febrero

APELLIDOS Y NOMBRE: Isabel Manzaneque Núñez

IDENTIFICADOR: imanzaneq3

DNI: 53902577F

CENTRO ASOCIADO MATRICULADO: Londres

CENTRO ASOCIADO DE LA SESIÓN DE CONTROL: BARBASTRO, BERGARA, CANTABRIA, CENTROS EN EL EXTRANJERO (EUROPA), CERVERA, LA RIOJA, TORTOSA, TUDELA, VITORIA-GASTEIZ

MAIL DE CONTACTO: imanzaneq3@alumno.uned.es

TELÉFONO DE CONTACTO: +44 7762347351

El analizador léxico

Código de usuario

En esta sección no ha resultado necesario añadir nada para el correcto funcionamiento de la práctica.

Directivas JLex

Como se especifica que el lenguaje no es case sensitive, se ha añadido la directiva **ignorecase** que hará que el lenguaje no sea sensible a mayúsculas: dos identificadores compuestos de los mismos caracteres y que difieran únicamente en el uso de mayúsculas o minúsculas se 6 consideran iguales.

Se han añadido también las siguientes definiciones de Macros:

```
ESPACIO_BLANCO    =    [ \t\r\n\f]
fin               =    "fin"{ESPACIO_BLANCO}
LETTER            =    [a-zA-Z]
ESP_CHAR          =    [£$?!&%@áéíóúñç]
STRING            =    \".*\"
DIGIT             =    [0-9]
INT               =    {DIGIT}+
ID                =    ({LETTER}({LETTER}|{DIGIT})*)
WRONG_ID          =    (({INT}|{ESP_CHAR})+({ID}|{INT}|{ESP_CHAR})*) |
                       ({ID}{ESP_CHAR}+({ID}|{INT}|{ESP_CHAR})*)
SALTO_LINEA       =    [\r\n]
COMMENT           =    "--" ~{SALTO_LINEA}
```

No se ha definido ningún estado nuevo, tan solo se utiliza el estado inicial YYINITIAL definido por defecto.

Reglas patrón acción

A continuación se muestra una tabla con las reglas patrón acción que reconoce el lenguaje.

Palabra Reservada	Token
and	AND
begin	BEGIN
Boolean	BOOLEAN
constant	CONSTANT
else	ELSE
end	END
False	FALSE
function	FUNCTION
if	IF
integer	INTEGER
is	IS
loop	LOOP
out	OUT
procedure	PROCEDURE

Put_line	PUT_LINE
record	RECORD
return	RETURN
then	THEN
True	TRUE
type	TYPE
while	WHILE

Delimitador	Token
“	DOUBLEQ
(	LEFTBRACKET
)	RIGHTBRACKET
--	COMMENT
,	COMMA
;	SEMICOLON
:	COLON

Operador	Token
-	MINUS
*	MULT
>	GREATERTHAN
/=	NOTEQUAL
and	AND
:=	ASIGN
.	ACCESS

Patron	Token
ID	ID
STRING	STRING
INT	INTEGER
WRONG_ID	Lexical error
COMMENT	
ESPACIO_BLANCO	
fin	

Gestión de errores

No se produce recuperación de errores léxicos. Se ha modificado la llamada a la función lexicalErrorManager.lexicalError() en caso de que un identificador no sea correcto y en caso de que no haya coincidencia con ningún patrón de la siguiente manera, para que se proporcione más información del error:

lexicalErrorManager.lexicalFatalError ("Error en la linea " + error.getLine() + ", columna " + error.getColumn() +", Identificador no valido: " + yytext());

lexicalErrorManager.lexicalFatalError ("Error en la linea " + error.getLine() + ", columna " + error.getColumn() + ", Caracter no esperado: " + error.getLexema());

El analizador sintáctico

Paquetes e importaciones / Código de usuario

No se ha necesitado añadir nada en estas secciones para el correcto funcionamiento de la práctica

Declaración de símbolos gramaticales

Se han declarado 39 terminales, 79 no-terminales y 191 producciones que producen 427 estados diferentes. Todo esto se explora con más detalle en el apartado de especificación de la gramática

Asociatividad y precedencia

Se han empleado únicamente las reglas de asociatividad y precedencia especificadas en el enunciado de la practica:

```
precedence left     AND;
precedence left     NOTEQUAL;
precedence left     GREATERTHAN;
precedence left     MINUS;
precedence left     MULT;
precedence left     ACCESS, LEFTBRACKET, RIGHTBRACKET;
```

Gestión de errores

No se produce recuperación de errores sintácticos, si se encuentra una construcción sintácticamente incorrecta se aborta la compilación. Cuando se encuentra una construcción que no se equipara a ninguna de las construcciones correctas propuestas, se lanza un error del tipo:

error {:syntaxErrorManager.syntaxFatalError("Mensaje de error"); errorCounter ++;:};

Este error aborta la compilación, proporciona más información acerca del error que se ha producido y además aumenta el contador de errores en una unidad.

Conclusiones

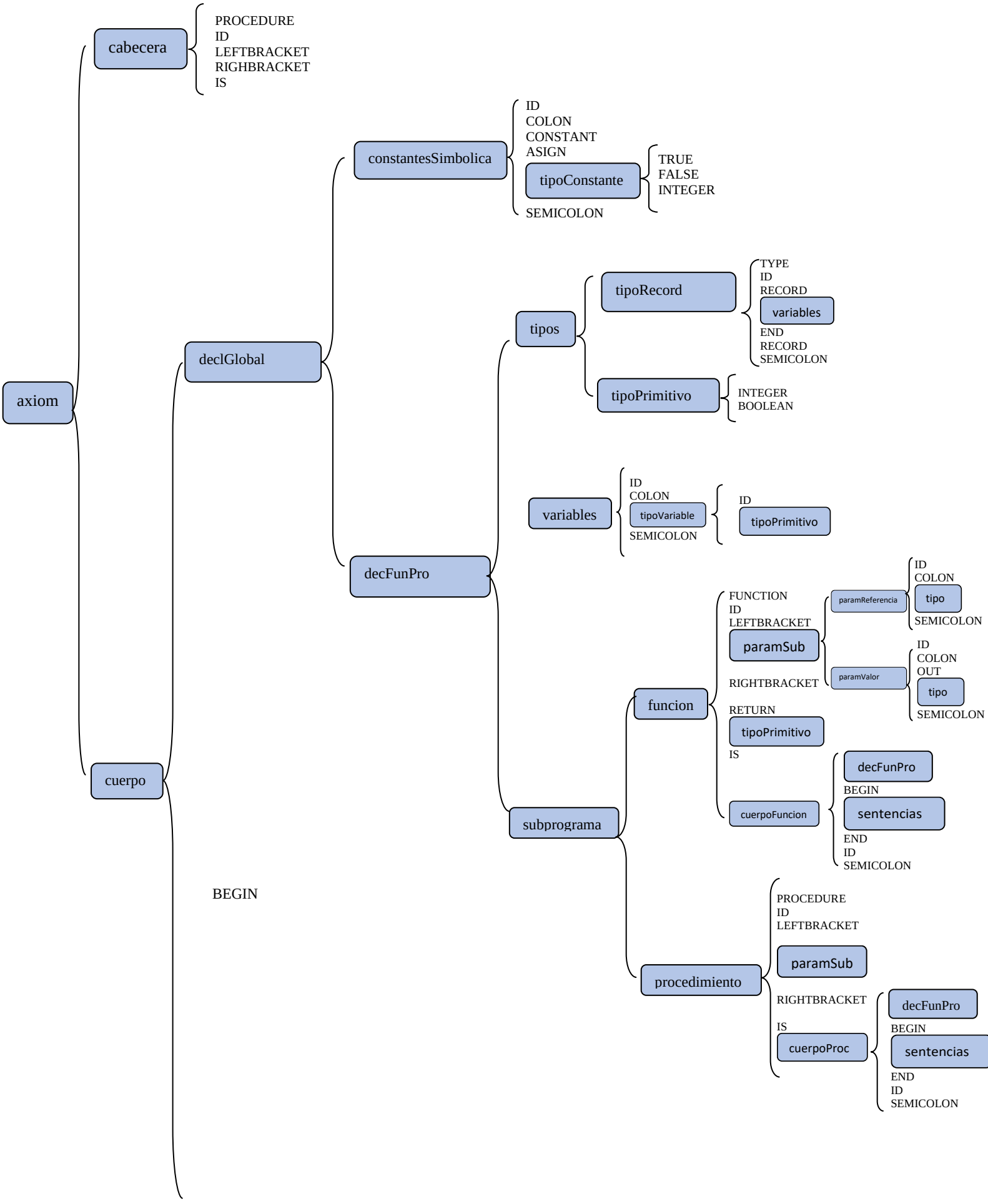
He disfrutado mucho en la realización de esta práctica, aunque en muchas ocasiones me ha resultado algo frustrante.

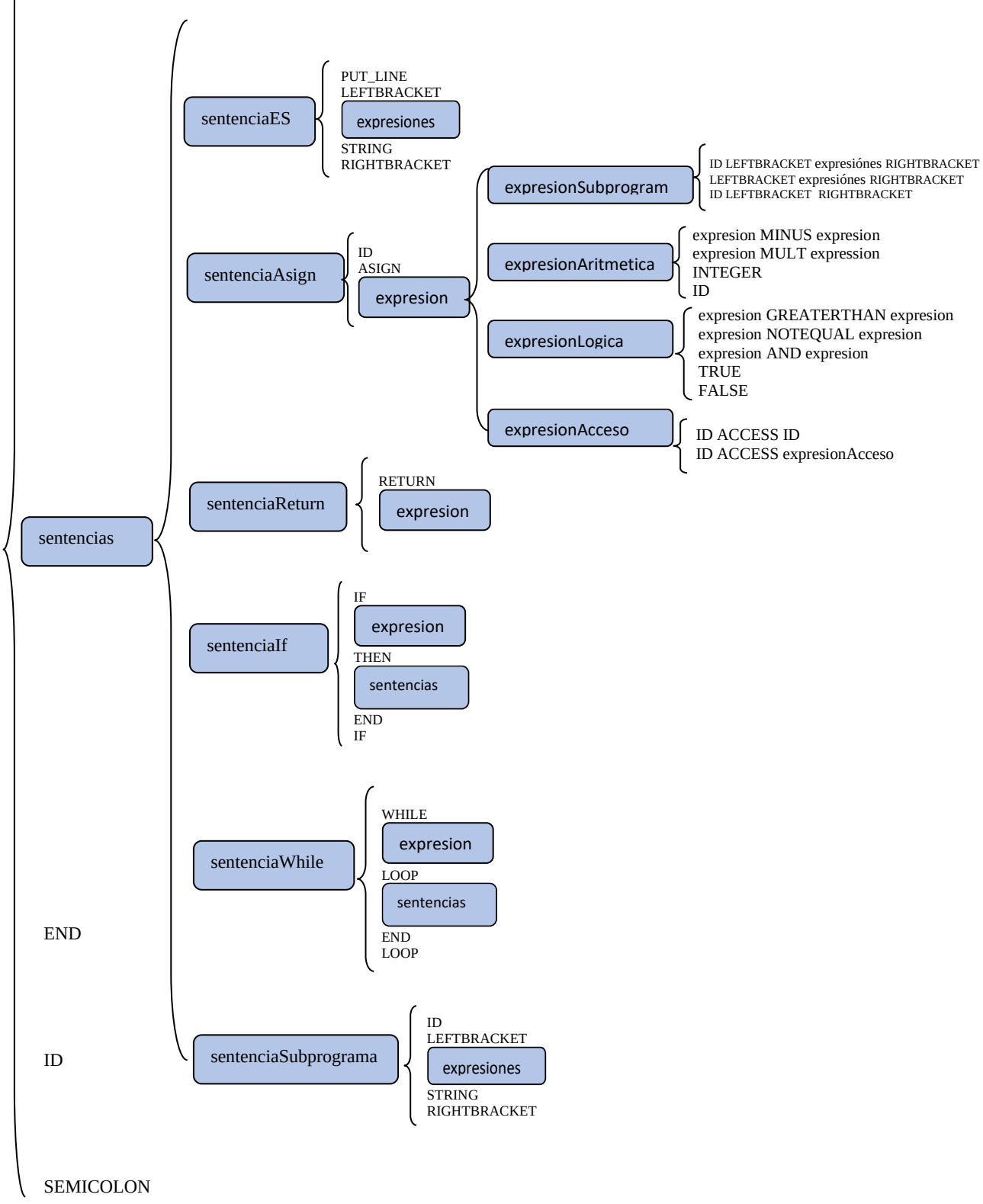
Comenzando con el analizador léxico, me resultó relativamente fácil y pude completarlo rápidamente. El analizador sintáctico me resultó más complicado y tuve que invertir bastante más tiempo en completarlo, hasta el punto de tener que empezar de 0 en una ocasión cuando me di cuenta de que el diseño de mi gramática solo iba a resultar en errores y conflictos

A pesar de mi encontronazo con el analizador sintáctico, la práctica me ha parecido muy interesante ya que me ha permitido ver como funcionan los lenguajes de programación “tras el telón”. Cuando estudié las gramáticas por primera vez en “Teoría de los lenguajes de programación” me resultaron interesantes, pero no podía realmente visualizar su utilización en un contexto relacionado con la informática; La realización de esta práctica me ha hecho atar cabos y ahora creo tener un mejor entendimiento de los fundamentos de los lenguajes de programación.

Por encima de todo una buena experiencia que me deja con ganas de comenzar a estudiar el analizador semántico.

Gramática


axiom
cabecera
PROCEDURE
ID
LEFTBRACKET
RIGHBRACKET
IS
cuerpo
declGlobal
constantesSimbolica
ID
COLON
CONSTANT
ASIGN
tipoConstante
TRUE
FALSE
INTEGER
SEMICOLON
decFunPro
tipos
tipoRecord
TYPE
ID
RECORD
variables
END
RECORD
SEMICOLON
tipoPrimitivo
INTEGER
BOOLEAN
variables
ID
COLON
tipoVariable
SEMICOLON
ID
tipoPrimitivo
subprograma
funcion
FUNCTION
ID
LEFTBRACKET
paramSub
paramReferencia
ID
COLON
tipo
SEMICOLON
paramValor
ID
COLON
OUT
tipo
SEMICOLON
RIGHTBRACKET
RETURN
tipoPrimitivo
IS
cuerpoFuncion
decFunPro
BEGIN
sentencias
END
ID
SEMICOLON
procedimiento
PROCEDURE
ID
LEFTBRACKET
paramSub
RIGHTBRACKET
IS
cuerpoProc
decFunPro
BEGIN
sentencias
END
ID
SEMICOLON
BEGIN


sentencias
sentenciaES
PUT_LINE
LEFTBRACKET
expresiones
STRING
RIGHTBRACKET
sentenciaAsign
ID
ASIGN
expresion
expresionSubprogram
ID LEFTBRACKET expresiónes RIGHTBRACKET
LEFTBRACKET expresiónes RIGHTBRACKET
ID LEFTBRACKET RIGHTBRACKET
expresionAritmetica
expresion MINUS expresion
expresion MULT expression
INTEGER
ID
expresionLogica
expresion GREATERTHAN expresion
expresion NOTEQUAL expresion
expresion AND expresion
TRUE
FALSE
expresionAcceso
ID ACCESS ID
ID ACCESS expresionAcceso
sentenciaReturn
RETURN
expresion
sentenciaIf
IF
expresion
THEN
sentencias
END
IF
sentenciaWhile
WHILE
expresion
LOOP
sentencias
END
LOOP
END
sentenciaSubprograma
ID
LEFTBRACKET
expresiones
STRING
RIGHTBRACKET
ID
SEMICOLON