PROCESADORES DE LENGUAJES

Primer examen. 17 de octubre de 2017

Observaciones: 1. Las calificaciones se publicarán hacia el 31 de octubre.
2. La revisión será hacia el 3 de noviembre.
3. En la web se avisarán las fechas exactas.
4. La duración de este examen es de 40 minutos.

Un nuevo lenguaje de programación tiene las siguientes características:

- Dispone de números enteros en octal formados por dígitos octales. Ejemplos: `246`, `0`, `77`. Los enteros se representan internamente con 4 bytes.
- Dispone de variables cuyos nombres están formados por cualquier cantidad de letras y dígitos, pero al menos debe tener una letra. Ejemplos: `hola`, `3E`, `c123`, `98w89`.
- Dispone de los operadores aritméticos siguientes: `+`, `-`, `++`, `--`.
- Dispone de cadenas de caracteres, delimitadas mediante comillas. Si se quiere introducir unas comillas dentro de la cadena, se puede realizar duplicando dicho carácter. Una cadena no puede tener más de 80 caracteres.

Teniendo en cuenta que los distintos elementos del lenguaje pueden ir separados por blancos, tabuladores o saltos de línea y que no hay distinción entre mayúsculas y minúsculas, se pide diseñar un **Analizador Léxico** para este lenguaje (*Tokens*, Gramática, Autómata y Acciones Semánticas), que introduzca toda la información posible en la Tabla de Símbolos.

Ejemplo de un fragmento de fichero correcto en este lenguaje:

```
1ab3cd+  3d  1  --  0
 "hola"++
+ "Adiós"
-3141   13579L
Hola -- "3,1416 (""pi"")"
-Begin 333-333+ 0110 ++ 3D
```

CC BY NC SA

PROCESADORES DE LENGUAJES

ANÁLISIS SINTÁCTICO

14 de noviembre de 2017

Observaciones: 1. Las calificaciones se publicarán hacia el 28 de noviembre.
2. La revisión será hacia el 30 de noviembre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen es de 40 minutos.

Sea la siguiente gramática G_1:

$F \rightarrow$ fun A C

$A \rightarrow$ T id A | ref

$T \rightarrow$ int

$C \rightarrow$ núm

a. En relación con el método de Análisis Sintáctico Ascendente LR(1), construye el **Autómata Reconocedor de Prefijos Viables** para la gramática G_1:

b. Detalla las filas de la **Tabla LR** correspondiente a todos los estados que contienen el ítem $A \rightarrow T\ id \bullet A$ o el ítem $A \rightarrow ref \bullet$.

CC BY NC SA

Sea la siguiente gramática G_2 (relativa a la definición de funciones en un lenguaje de programación):

$F \rightarrow$ fun id A C val : T

$A \rightarrow$ T id A | ref T id A | λ

$T \rightarrow$ int | real

$C \rightarrow$ id = núm | λ

c. Calcula los conjuntos **FIRST** y **FOLLOW** de todos los símbolos no terminales de G_2.

d. Comprueba si las reglas de A de la gramática G_2 cumplen la **condición LL(1)**.

e. Diseña el procedimiento del método **Descendente Recursivo** correspondiente al símbolo A (se puede usar el método `EqT (t)` – Equipara-Token) de G_2.

f. Señala en la tabla siguiente los errores u omisiones presentes (método de Análisis Sintáctico **Descendente LL(1)**).

	fun	**val**	**:**	**id**	**ref**	**int**	**real**	**λ**
F	F→fun id A C val : T							
A								A→λ
T					A→ref T id A	T→int	T→real	
C				C→id = núm				C→λ

CC BY NC SA

PROCESADORES DE LENGUAJES

10 de enero de 2018

Observaciones: 1. Las calificaciones se publicarán hacia el 22 de enero.
2. La revisión será hacia el 24 de enero.
3. En la web se avisarán las fechas exactas.
4. La duración de este examen es de 40 minutos.

3. Sea un lenguaje con las siguientes características:

- Tiene los tipos int, long, byte, void, float, boolean y char. Esto quiere decir que existen reglas para los tipos (T → int | long | byte | void | float | boolean | char).
- Tiene expresiones (E → id | E op-ar E | E op-rel E | E op-lóg E | (E) | id [E] | cte_entera | cte_real | …).
- Exige declaración previa de variables
- Las variables y los vectores pueden ser de cualquier tipo del lenguaje, salvo void.
- Los tipos int, long y byte son variantes del tipo "entero" y son compatibles entre sí. Para el resto de tipos, el lenguaje no tiene conversión automática entre ellos.
- Al asignar valores a un vector completo (id={L}), para cada elemento del vector tiene que haber uno y solo un valor.
- La variable índice (id) de la sentencia from se declara en la propia sentencia y ha de ser de alguno de los tres tipos "entero".
- La sentencia from funciona de la siguiente manera: se realiza la asignación id=E; si la expresión del when es cierta, se ejecuta el cuerpo del from y se vuelve al inicio; si es falsa, se sale del from.

Se pide diseñar el Analizador Semántico mediante un Esquema de Traducción únicamente para las siguientes reglas de la gramática (usando el espacio proporcionado para cada regla):

D → T id * /* declaración de una variable */*

D → T id [1 .. cte_entera] * /* declaración de un vector de "cte_entera" elementos */*

S → id = E * /* asignación de un valor a una variable */*

CC BY NC SA

S → id [cte_entera] = E ** asignación de un valor a un elemento del vector */*

S → id = { L } ** asignación de valores a un vector completo */*

L → E ** una expresión */*

L → E , L_1 ** una lista de expresiones */*

S → from T id = E_1 when E_2 { S_1 } ** sentencia repetitiva */*

CC BY NC SA

PROCESADORES DE LENGUAJES

Examen Final. 10 de enero de 2018

Observaciones: 1. Las calificaciones se publicarán hacia el 22 de enero.
2. La revisión será hacia el 24 de enero.
3. En la web se avisarán las fechas exactas.
4. Los 3 ejercicios tienen la misma puntuación.
5. La duración total de este examen es de 120 minutos.

1. Un lenguaje de programación tiene las siguientes características:

- Dispone de variables cuyos nombres están formados por un máximo de 80 letras y dígitos, teniendo al menos una letra. Ejemplos: `hola`, `3E`, `c123`, `98w89`.
- Dispone de los operadores relacionales siguientes: `>`, `<`, `>>`, `<<`.
- Dispone de números enteros en hexadecimal formados por dígitos hexadecimales (del `0` al `9` y de la `A` a la `F`), terminando con el carácter `#`. Los enteros se representan internamente con 6 bytes. Ejemplos: `C9#`, `0#`, `88#`.
- Dispone de cadenas de caracteres, delimitadas mediante comillas simples. Si se quiere introducir una comilla dentro de la cadena, se puede realizar duplicando dicho carácter.

Sabiendo que los distintos elementos del lenguaje pueden ir separados por blancos, tabuladores o saltos de línea y que no hay distinción entre mayúsculas y minúsculas, se pide diseñar un **Analizador Léxico** para este lenguaje (*Tokens*, Gramática, Autómata y Acciones Semánticas), que introduzca toda la información posible en la Tabla de Símbolos.

Ejemplo de un fragmento de fichero correcto en este lenguaje:

```
1ab3cd>  3d  1a#  <<  B0# 'hola'>>< 'Adiós' >
    'tiempo: 3h25''.' <3141#   13579L     Hola <<
'3,1416 (''pi'')'   <Begin FFF#<fff#> 010# >> 3D
```

2. Sea la gramática G cuyo conjunto de reglas es:

P = { S → X Y Z
X → λ | X *
Y → Z Y | +
Z → – }

Se pide:

a. Construir el **Autómata Reconocedor de Prefijos Viables** para esta gramática.

b. Analizar los posibles **conflictos** a la hora de generar un Analizador LR para esta gramática.

c. Justificar si esta gramática es **LL** o no. En caso negativo, transformarla para que lo sea (cambiando exclusivamente las reglas del No Terminal implicado).

d. Construir la **Tabla del Analizador Sintáctico Descendente** para la gramática LL (siguiendo el mismo orden de los símbolos).

CC BY NC SA

PROCESADORES DE LENGUAJES

Examen Final, 27 de junio de 2018

Observaciones:	1. Fecha estimada de publicación de las calificaciones: 10 de julio. Fecha estimada de la revisión: 12 de julio. Las fechas exactas se avisarán en la web 2. La duración de este examen es de 2 horas. 3. Cada ejercicio deberá entregarse en hojas separadas. 4. Todos los ejercicios tienen la misma puntuación.

1. Un fragmento de un lenguaje dispone de identificadores (pueden estar formados hasta por 256 letras o dígitos y comienzan por una letra), operadores relacionales ('=', '<>', '<', '>', '<=' y '>=') y comentarios (comienzan por '<<' y terminan por '>>', pudiendo contener cualquier carácter en su interior). Así mismo, el lenguaje puede usar como delimitadores el espacio en blanco, el tabulador y el salto de línea.

Se pide construir un **Analizador Léxico** (gramática BNF regular, *tokens*, autómata de tamaño reducido y acciones semánticas) para este fragmento de lenguaje, indicando todos los accesos a la Tabla de Símbolos y detallando todos los mensajes de error.

2. En relación con el Análisis Sintáctico, responder a las preguntas en la plantilla de respuesta.

3. Dado el siguiente fragmento de la gramática de un lenguaje de programación:

F → function id (R) { S }
R → var T id U
U → , R | λ
T→ bool | int | real | chars
E → id
S → S_1 ; S_2 | call id (P) | id A E
P → E | E , P_1
A → += | %=

Teniendo en cuenta que:

- Los identificadores son lógicos (1 *byte*), enteros (2 *bytes*), reales (4 *bytes*) y cadena (10 *bytes*). El lenguaje no tiene conversiones automáticas de tipos (excepto para la operación +, como se detallada a continuación).
- La sentencia call llama a la función id que recibe parámetros y no devuelve nada.
- El operador += equivale a id = id + E. Se aplica a números y a cadenas de la siguiente manera:
  - realiza una suma con asignación (id = id + E) entre enteros o reales, convirtiendo el resultado al tipo del identificador del lado izquierdo
  - realiza una concatenación y asignación entre cadenas.
- El operador id %= E equivale a id = id % E. El operador lógico % devuelve cierto si el primer operando es falso y el segundo es cierto; en caso contrario devuelve falso.

Se pide diseñar, en la plantilla de respuesta, el **Analizador Semántico** mediante una **Definición Dirigida por Sintaxis**, detallando todos los accesos a la Tabla de Símbolos.

CC BY NC SA

APELLIDOS:……NOMBRE:……..

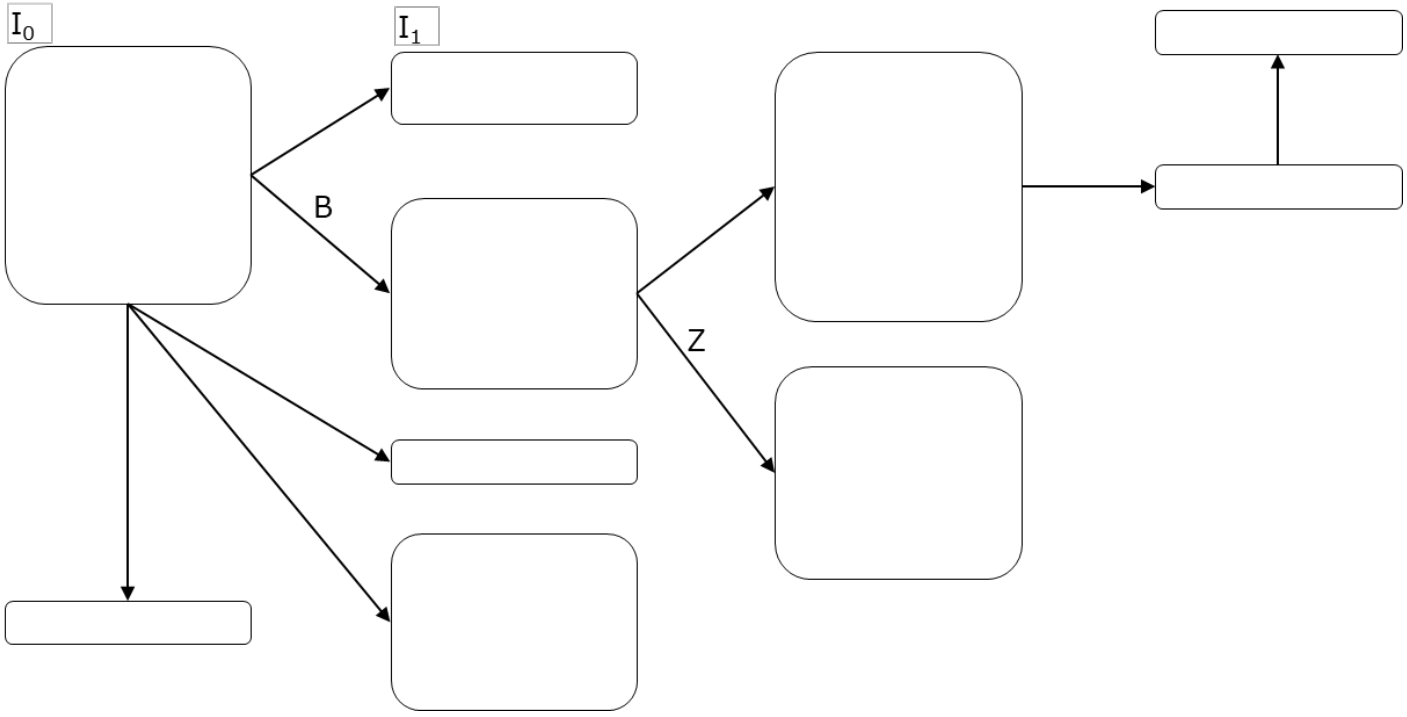
PROCESADORES DE LENGUAJES

Plantilla de Respuesta - Análisis Sintáctico. 27 de junio de 2018

a. En relación con el método de Análisis Sintáctico Ascendente (LR), dada la gramática:

S → B Z S | fin
B → var id | λ
Z → 1 Z | λ

Se pide construir el **Autómata Reconocedor de Prefijos Viables** para esta gramática, usando la siguiente plantilla, que puede contener errores u omisiones:



b. Se pide realizar el análisis de todos los posibles **conflictos** en el Autómata resultante del apartado **a**.

CC BY NC SA

c. En relación con el método de **Análisis Sintáctico Descendente LL(1)**, dada la gramática:

A → 1 B | 2 C | D C
B → 1 C
C → 3 D | 4 C
D → 5 A | 6 | λ

Se pide corregir los errores u omisiones de la siguiente **tabla LL(1)**:

	1	**2**	**3**	**4**	**5**	**6**	**$**	**λ**
A	A→ 1 B	A→ 2 C			A→ D C	A→ D C	A→ D C	
B	B→ 1 C							
C								
D					D→ 5 A	D→ 6		D→ λ

d. Se pide escribir los procedimientos correspondientes a los símbolos B y D pertenecientes a un **Analizador Sintáctico Descendente Predictivo Recursivo**, dada la gramática del apartado **c**.

CC BY NC SA

APELLIDOS:.....NOMBRE:.....

PROCESADORES DE LENGUAJES

Plantilla de Respuesta - Análisis Semántico. 27 de junio de 2018

$F \rightarrow$ function id (R) { S }

$R \rightarrow$ var T id U

$U \rightarrow$, R

$U \rightarrow \lambda$

$T \rightarrow$ bool

$T \rightarrow$ chars

$T \rightarrow$ int

$T \rightarrow$ real

$E \rightarrow$ id

$S \rightarrow S_1 ; S_2$

CC BY NC SA

$S \rightarrow$ call id (P)

$S \rightarrow$ id A E

$P \rightarrow E$

$P \rightarrow E , P_1$

$A \rightarrow +=$

$A \rightarrow \%=$

CC BY NC SA

PROCESADORES DE LENGUAJES

ANÁLISIS LÉXICO

3 de octubre de 2018

Observaciones: 1. Las calificaciones se publicarán hacia el 17 de octubre.
2. La revisión será hacia el 19 de octubre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen es de 40 minutos.

Una casa de apuestas está en la última fase del proyecto de automatización de la gestión de apuestas. En esta etapa, se abordará la gestión de las apuestas recibidas por teléfono. En estos casos, cada teleoperador escribe cada apuesta en un fichero de texto (con una apuesta por línea) que, una vez unificados por día, se vuelcan a la Base de Datos donde ya están las apuestas recibidas por la web y aplicaciones móviles.

Cuando se recibe una llamada telefónica, los teleoperadores anotan en el fichero los siguientes elementos en cualquier orden, pudiendo separarlos por blancos o tabuladores:

- Código de Apuesta: consta del año de la apuesta (un año del siglo actual escrito con 4 dígitos) seguido de un guion y otra serie de dígitos (al menos uno). Por ej.: `2018-123`.
- Cantidad apostada: el servicio de teleoperador está disponible en la Unión Europea, Reino Unido y Estados Unidos, por lo que el teleoperador anota la moneda en la que se hace la apuesta (obligatoriamente con parte entera y dos decimales). Por ej.: `7.99€`, `28.05£`, `45.90$`.
- Código de Usuario: empieza por la letra '`U`' y va seguida de 6 dígitos. Por ej.: `U123456`.
- Hora de la Apuesta: se escribe con horas y minutos en el formato `dd:dd`. Hay que tener en cuenta que las apuestas por teleoperador solo pueden realizarse desde las *08:00* hasta las *21:59*. Por ej.: `14:21`, `09:00`.

La Base de Datos que almacena la información tiene los siguientes campos:

- Código de Apuesta. Por ej.: `2018-123`.
- Cantidad apostada. Se almacena siempre el importe en euros. Por ej.: `7.99`, `31.57`, `39.55`.
- Código de Usuario: Por ej.: `U123456`.
- Hora de la Apuesta: se almacena como el minuto del día en que se ha recibido la apuesta. Por ej.: `861`, `540`.

Ejemplo de fichero de apuestas <u>correcto</u>:

```
2019-34 37.00£  U435678   12:35
  5442.90$2018-85233852      21:59U564896
```

Ejemplo de fichero de apuestas con todos los elementos <u>incorrectos</u>:

```
2209-34 37.0£  U435   22:35
  .90¥  201885233852  08:67     V564896
```

Se pide realizar el Diseño del **Analizador Léxico**, que ayude en el proceso de volcar los ficheros de texto a la Base de Datos, en todas sus fases: *Tokens*, Gramática Regular, Autómata Finito, Acciones Semánticas y Errores.

Nota: Se dispone de acceso al servicio del Banco de España para la tasa de cambio de monedas, con el formato *cambio-diario (importe, moneda)*, que devuelve la tasa de cambio en euros en formato de número real. Por ej.: *cambio-diario (45.90, $)* → *39.55*.

CC BY NC SA

APELLIDOS:..... NOMBRE:.....

PROCESADORES DE LENGUAJES

ANÁLISIS SINTÁCTICO

14 de noviembre de 2018

Observaciones: 1. Las calificaciones se publicarán hacia el 28 de noviembre. 2. La revisión será hacia el 30 de noviembre. 3. En la web se avisará de las fechas exactas. 4. La duración de este examen es de 40 minutos.

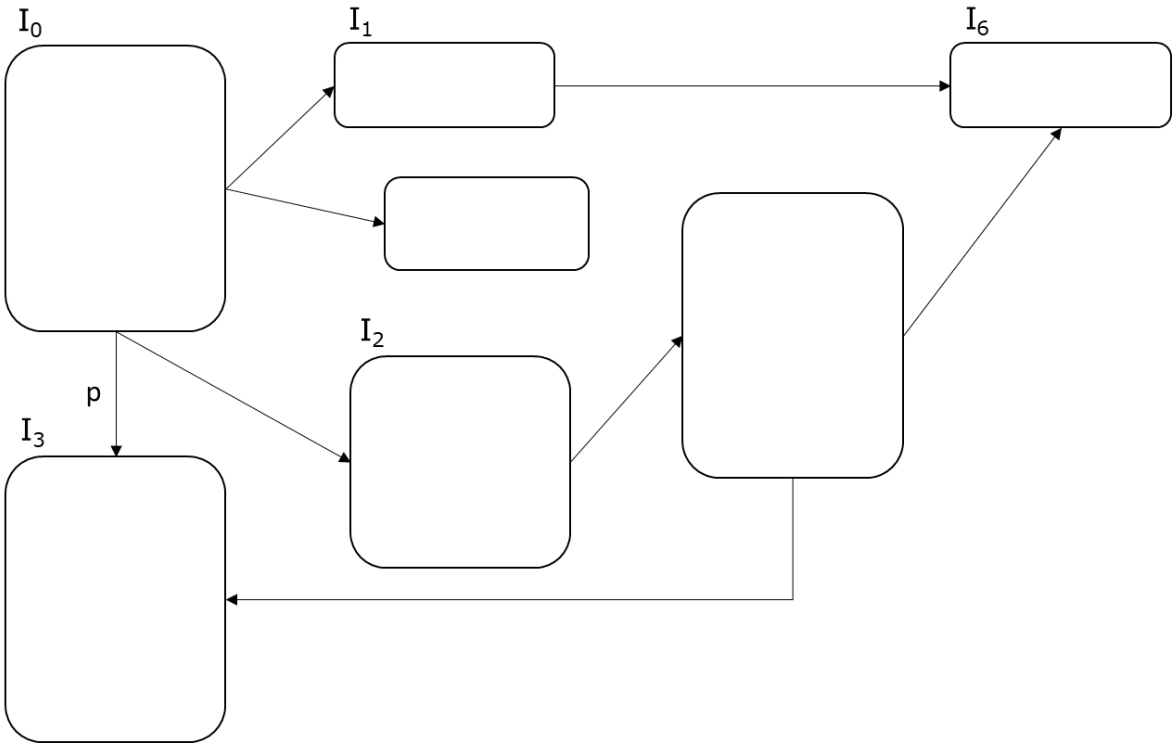
Sea la gramática G_1:

$S \rightarrow A + B$

$A \rightarrow p\ S \mid \lambda$

$B \rightarrow rec \mid A\ B$

a. Construye el **Autómata** Reconocedor de los Prefijos Viables de un **Analizador Sintáctico Ascendente** (LR) para la gramática G_1:



b. Detalla el estudio de todos los posibles **conflictos** y responde, razonadamente, si la gramática G_1 **es válida o no** para este método

CC BY NC SA

Sea la gramática G_2:

S → A + B
A → p S + | λ
B → rec | A B it C
C → A C | it | λ

c. Para la gramática G_2, calcula los conjuntos **First** y **Follow** de todos los símbolos no terminales:

FIRST	**S**		**FOLLOW**	**S**	
	A			**A**	
	B			**B**	
	C			**C**	

d. Construye la tabla de un **Analizador Sintáctico Descendente** para la gramática G_2 y responde razonadamente si dicha gramática **es válida o no** para ese método.

	+	p	rec	it		

e. Representa la **Tabla de Símbolos** completa y rellénala para el siguiente fragmento de programa correcto (teniendo en cuenta que las cadenas se representan con 128 bytes, los enteros con 2 bytes, los reales con 4 bytes y las direcciones de memoria con 8 bytes):

```
var int a= 8;
Function int f (string a, float b)     // paso de parámetros por valor
{   var float v[10], x= 3.2;     // declaración de vector de 10 reales y variable real
    v[3]= b + x;                 // el lenguaje no permite anidamiento de funciones
    a= "hola"; return f (a, v[3]); }
```

f. Indica qué **módulo del procesador** introduce cada **atributo** en la Tabla de Símbolos.

cc BY NC SA

PROCESADORES DE LENGUAJES

23 de enero de 2019

Observaciones: 1. Las calificaciones se publicarán hacia el 4 de febrero.
2. La revisión será hacia el 6 de febrero.
3. En la web se avisarán las fechas exactas.
4. La duración de este examen es de 40 minutos.

3. Sea la siguiente gramática de contexto libre que representa un fragmento de un lenguaje:

P → D ; S
D → D ; D | var T : V
T → int | real | bool | array [C] of T
V → id A
A → , V | λ
S → S ; S | id := E
E → E + E | id | id [E] | C
C → cte_entera | cte_real

Se pide construir un **Analizador Semántico** para este lenguaje mediante un **Esquema de Traducción**, detallando todos los accesos a la Tabla de Símbolos. Explicar brevemente todas las funciones y atributos utilizados.

Notas:

- El lenguaje no exige declaración previa de variables. Cualquier variable no declarada se considera de tipo real.
- No existe ningún tipo de conversiones de tipos.
- Los enteros ocupan 2 bytes, los reales 4 y los lógicos 1.
- La operación + sólo puede usarse para sumar valores numéricos.
- El tamaño y el índice de un array deben ser enteros. Los índices de un array van desde 1 hasta el tamaño indicado por C.

CC BY NC SA

PROCESADORES DE LENGUAJES

Examen Final. 23 de enero de 2019

Observaciones: 1. Las calificaciones se publicarán hacia el 4 de febrero
2. La revisión será hacia el 6 de febrero
3. En la web se avisarán las fechas exactas
4. Cada ejercicio tiene la misma puntuación y deberá entregarse en hojas separadas
5. El tiempo para realizar cada ejercicio es de 40 minutos

1. Se tiene un lenguaje que consta de identificadores de cuenta (*login*), números y *passwords*:

- Los números pueden ser enteros (representados con 4 bytes) o reales (representados en coma flotante). Los reales llevan coma y deben tener parte entera y parte decimal.
- Los *login* están formados por letras o dígitos, y han de empezar por letra. Cada vez que se encuentre un *login* nuevo, deberá almacenarse en la tabla de usuarios.
- Las *passwords* deben tener al menos 8 caracteres entre letras, dígitos y caracteres especiales (@, #, %), debiendo tener obligatoriamente al menos uno de cada tipo.

Teniendo en cuenta que los elementos del lenguaje van separados por delimitadores, se pide construir un **Analizador Léxico** para este lenguaje (*tokens*, gramática regular, autómata finito, acciones semánticas y errores).

2. Dado el siguiente fragmento de una gramática, G_1:

```
S → id = E          /* asignación de una expresión a un identificador */
E → id              /* identificador */
E → id <= E         /* identificador menor o igual que expresión */
```

Se pide:

a. Con la gramática G_1 (o realizando en ella los cambios mínimos necesarios), construir el **autómata reconocedor de prefijos viables**, estudiar detalladamente los posibles conflictos sobre el autómata y construir la **tabla** de un **Analizador Sintáctico Ascendente** SLR, mostrando todos los pasos seguidos.
b. Con la gramática G_1 (o realizando en ella los cambios mínimos necesarios para que sea LL(1)), construir la **tabla** de un **Analizador Sintáctico Descendente** no recursivo, mostrando los pasos seguidos.

Dado el siguiente fragmento de una gramática, G_2, donde A es el axioma:

```
A → B C g | a A
B → b B b | λ
C → e C | d C e | λ
```

Se pide:

c. Calcular todos los **First** y **Follow** de los símbolos no terminales de G_2.
d. Demostrar si G_2 es **LL(1)** comprobando la Condición **LL(1)** sobre todos los símbolos no terminarles de G_2.

CC BY NC SA

PROCESADORES DE LENGUAJES

12 de julio de 2019

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 22 de julio. 2. Fecha **estimada** de la revisión: 24 de julio. 3. En la web se avisará la fecha exacta de publicación de las calificaciones y la fecha y hora definitiva de la revisión. 4. La duración de este examen será de 2 horas.

3. Sea el siguiente fragmento de una gramática de contexto libre que representa una parte de un lenguaje:

```
S → S ; S  |  id := E  |  var T : V
T → int  |  real  |  bool  |  matrix [ C , C ] of T
V → id A
A → , V  |  λ
E → E ≈ E  |  id  |  id [ E , E ]  |  C
C → cte_entera  |  cte_real
```

Se pide construir un **Analizador Semántico** para este lenguaje mediante una **Definición Dirigida por la Sintaxis**, detallando todos los accesos a la Tabla de Símbolos. Explicar brevemente todas las funciones y atributos utilizados.

Notas:

- El lenguaje exige declaración previa de variables (`var T : V`).
- Existen conversiones de tipos entre números.
- Los lógicos ocupan 1 byte, los números enteros 2 y los números reales 4.
- La operación ≈ se usa para comparar valores numéricos y da cierto solo si los números son aproximadamente iguales (se diferencian menos de 0.5).
- Los tamaños de una `matrix` deben ser enteros. Los índices de una `matrix` van desde 0 hasta el tamaño indicado por `C`. No se permiten matrices de matrices.

CC BY NC SA

1. Se pretende realizar unos Diseños de **Analizadores Léxicos** correspondientes a los siguientes supuestos:

A. Una aplicación recibe como entrada un fichero de configuración donde se establecen los parámetros que definen cómo se ejecutará la aplicación. El fichero es de tipo CSV, por lo que todos los elementos en el fichero (parámetros y sus valores) siempre finalizan con ";" que es por lo tanto *el único delimitador*.

En cada línea del fichero aparece un único parámetro y su valor o valores:

- Los nombres de parámetros se forman con letras, opcionalmente separadas por guiones, y terminando en una letra o en un dígito.
- Para cada parámetro, pueden aparecer diferentes tipos de valores:
  - números enteros ([0-65535])
  - fechas en formato `mm/aaaa` ([01-12]/[0000-2019]). La fecha se representará internamente como el número de meses transcurridos desde el mes 01 del año 0000.

El fichero también puede tener comentarios, encabezados por `#`. El comentario ocupa toda la línea, por lo que no termina hasta finalizar la línea.

Ejemplo de fichero de configuración:

```
Mes-inicio;07/2018;
#Esto, es un comentario válido;
Lista--valores;2;3;4;
Sistema-Pun-tuacion2;01;12;234;3456;45678;
```

Se pide el diseño del Analizador Léxico correspondiente: **Tokens, Autómata Finito Determinista y Acciones Semánticas/Errores**.

B. Un lenguaje tiene los siguientes *tokens* (delimitados por al menos un espacio en blanco):

- Identificadores: empiezan por letra, seguida de cualquier cantidad de caracteres alfanuméricos y guiones (`-`), pero no pueden acabar en guion (ej.: `a12-sd--34`, `uno`, `i-1`)
- Números enteros en el intervalo [0-65535] (ej.: `014`, `1256`)
- Números enteros con exponente (ej.: `014E0`, `1256E12`, `2E023`). Al menos debe haber un dígito a cada lado de la "E".
- Códigos de departamento: 3 dígitos terminando en la letra "D" (ej.: `000D`, `056D`, `769D`)

Se pide escribir la **Gramática Regular** que pueda usarse para construir directamente un Analizador Léxico.

CC BY NC SA

APELLIDOS:_____ NOMBRE:_____

Análisis Sintáctico. HOJA DE RESPUESTAS

2. Dada la siguiente gramática:

1. A → B D c
2. A → a
3. A → λ
4. B → b D
5. B → λ
6. D → x D
7. D → y B

Se pide:

a) Calcular los siguientes conjuntos:

FIRST (A)=	FOLLOW (A)=
FIRST (B)=	FOLLOW (B)=
FIRST (D)=	FOLLOW (D)=

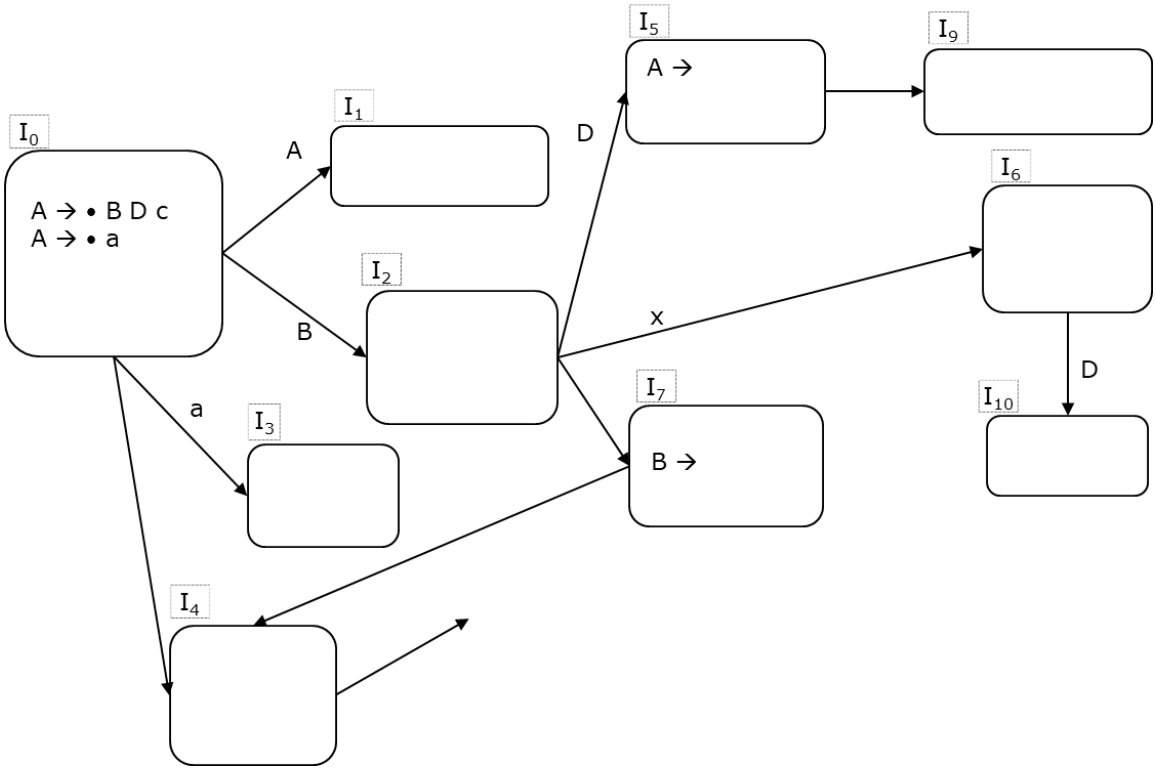
b) Demostrar si la gramática cumple la **condición LL(1)**, mostrando todas las comprobaciones necesarias.

c) Construir la tabla del **Analizador Sintáctico LL(1)**.

	c	a	b	x	y		

CC BY NC SA

d) Completar el **AFD reconocedor de los prefijo viables de un LR(1)** (sobre la plantilla adjunta, que está incompleta pero no contiene ningún error).



e) Rellenar las filas indicadas de la **tabla de un Analizador LR(1)**.

	c	a	b	x	y				
0									
1									
2									
3									
5									
7									
9									
10									

f) Mostrar el **funcionamiento** paso a paso del **análisis sintáctico LR(1)** en el reconocimiento de la cadena correcta "a", dando además el *parse* y el árbol.

CC BY NC SA

PROCESADORES DE LENGUAJES

ANÁLISIS LÉXICO Y TABLA DE SÍMBOLOS

23 de octubre de 2019

Observaciones: 1. Las calificaciones se publicarán hacia el 6 de noviembre.
2. La revisión será hacia el 8 de noviembre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen es de 40 minutos.

Un supermercado en línea quiere tener cada día un registro de todos los productos comprados por sus clientes para controlar las existencias de sus productos. Para ello, la aplicación de la tienda proporciona al final del día un fichero que contiene, en este orden, la siguiente información (cada elemento puede ir separado por delimitadores):

- Nombre completo del comprador: está formado por letras, aunque también existen nombres compuestos (con un único guion en su interior para separar ambos nombres). Ej.: `Juan Moreno-Romero`. (Si un comprador adquiere varios productos, solamente aparecerá su nombre una vez delante de todos sus productos.)
- Código EAN (*European Article Number*) del producto: el código de barras de cada artículo se corresponde con una secuencia de 13 dígitos, donde los 3 primeros dígitos identifican el país del producto (solo hay 675 prefijos válidos y cada prefijo con su país se encuentran accesibles en una tabla). Ej.: `8412042502367` (el prefijo 841 corresponde a España), `7702993035023` (el prefijo 770 corresponde a Colombia), `1997531598520` (199 es uno de los prefijos que no corresponde a ningún país, por lo que es un EAN erróneo).

- Descripción del producto: viene en una o varias cadenas encerradas entre comillas simples ('). Ej.: `'Agua Mineral Natural' '500 ml.' 'Pontevedra (España)'`. (Una cadena no puede tener más de 128 caracteres.)
- Precio del producto: es un número que puede tener una coma seguida de uno o dos decimales. Ej.: `11,99 103 2,5`. (El supermercado no tiene productos que cuesten más de 5000€.)

Ejemplo de fichero de compras correcto:

```
   Juan Luis Moreno-Romero y Alonso 8412042502367    'agua
mineral -natural-'   '500ml.' 'Pontevedra    (España)' 1,99
7702993035023'12 Barritas de coco con chocolate, 12 un.'3,5 Peter
Navarro0692771022044  '' 'American block 120 hojas, 6*4.3” 4º'  10
```

Ejemplo de fichero de compras con elementos incorrectos:

```
Juan-Luis-Pedro Moreno- Fdez. 12345678 'abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz'   ,99
1997515389521 '(Madrid) 8,
```

Se pide realizar el Diseño del **Analizador Léxico**, en todas sus fases: *Tokens*, Gramática Regular, Autómata Finito, Acciones Semánticas y Errores.

CC BY NC SA

Apellidos: …… Nombre: ……

PROCESADORES DE LENGUAJES

ANÁLISIS SINTÁCTICO

20 de noviembre de 2019

Observaciones:
1. Las calificaciones se publicarán hacia el 2 de diciembre.
2. La revisión será hacia el 4 de diciembre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen es de 40 minutos.

Se pide, dada la siguiente gramática:

$V \rightarrow id\ L\ ;\ V \mid L \mid \lambda$ $\quad$ $L \rightarrow cte\ R$ $\quad$ $R \rightarrow ,\ cte\ R \mid \lambda$

a. Para el método de **Análisis Sintáctico Ascendente LR**, construir el **estado inicial** del **Autómata** Reconocedor de Prefijos Viables, y todos los que se obtienen directamente desde él con la función *goto* sobre dicho estado inicial.

b. Detallar el estudio de los posibles **conflictos** para los estados del apartado a.

c. Detallar todo el contenido de la **Tabla LR** que se pueda rellenar con el fragmento de autómata del apartado a.

CC BY NC SA

PROCESADORES DE LENGUAJES

23 de enero de 2020

Observaciones: 1. Las calificaciones se publicarán hacia el 7 de febrero.
2. La revisión será hacia el 11 de febrero.
3. En la web se avisará de las fechas exactas.
4. Las tres preguntas tienen la misma puntuación.
5. La duración de este examen es de 120 minutos.

Análisis Léxico

1. Una empresa quiere tener cada día un registro de toda la actividad de sus empleados en sus distintas oficinas para poder calcular adecuadamente su nómina. Para ello, el sistema de control horario proporciona al final del día un fichero que contiene, en este orden, la siguiente información (cada elemento tiene que ir separado por algún delimitador):

- Nombre completo del trabajador: tendrá varios nombres formados por letras, aunque también existen nombres compuestos (con un único guion en su interior para separar ambos nombres). Ej.: `Pedro Romero-Moreno`.
- Código postal donde se ubica la oficina: el código postal de cada oficina se corresponde con una secuencia de 5 dígitos, donde los 2 primeros dígitos identifican la provincia de la oficina (solo hay 52 provincias válidas, desde el 01 hasta 52). Ej.: `28660` (el prefijo 28 corresponde a Madrid), `08080` (el prefijo 08 corresponde a Barcelona), `55080` (55 no es un prefijo que corresponda a ninguna provincia, por lo que es un código postal erróneo).
- Horas trabajadas en la jornada: es un número que puede tener una coma seguida de uno o dos decimales. Ej.: `4,75 8 10,5`. (Evidentemente, un trabajador no puede trabajar más de 24 horas al día.)
- Tareas desempeñadas por el trabajador: están en una o varias cadenas encerradas entre comillas angulares («»). Ej.: `«Venta de 50 neveras.» «Redacción informe 99/2011» «Pedido de la pieza H-12321 (a Burgos)»`. (Una cadena no puede tener más de 64 caracteres ni estar vacía.)

Ejemplo de fichero de actividad correcto:

```
  Luis  Felipe  y  Alonso   28020 05,90 «pedido agua -pura- en garrafa»
«500Kl. de depuración» «Envío a Sofía   (Bulgaria)» Xon Moreno-Romero
01310 3,5 «Enriquecimiento de Uranio, 0,75% 238U» Juan-Jo
Juan 52901  22 «American DDT 10100, 6*4.5” nivel 4º»
```

Ejemplo de fichero de actividad con todos los elementos erróneos:

```
Luis-Pedro-Juan 123456  ,75 «abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz»  Glez. 98765
24,525  «»  Pedro--Luis Moreno-  2468 3,  «(España)
```

Se pide realizar el Diseño completo del **Analizador Léxico**: *Tokens*, Gramática Regular, Autómata Finito, Acciones Semánticas y Errores.

CC BY NC SA

2.Sea la siguiente gramática (siendo L el axioma):

L → id L C | λ
C → E & C | λ
E → id | E + E | ++ id | id ++

Se pide:

a. Construir el estado inicial del **Autómata Reconocedor de Prefijos Viables**.

b. Sabiendo que el estado I_3 está formado por los ítems que se indican, obtener todos los **estados** que se obtienen directamente desde I_3 en un paso.

I_3

L → id L ▪ C
C → ▪ E & C
C → ▪
E → ▪ id
E → ▪ E + E
E → ▪ ++ id
E → ▪ id ++

c. Detallar el estudio de los posibles **conflictos** en todos los estados del apartado b (incluido I_3).

d. Detallar el contenido de las filas de la **Tabla LR** correspondientes a los estados del apartado b. Se deberá incluir solamente la información que se recoge en dichos estados.

e. Comprobar detalladamente, para todas las reglas de la gramática, si cumplen la **condición LL(1)**.

f. Si alguna de las reglas de E no cumple la condición LL(1), indicar **cómo** se tendrían que **modificar** esas reglas para que cumplan la condición LL(1).

g. Diseñar el procedimiento del método **Descendente Recursivo** correspondiente al símbolo C.

CC BY NC SA

d. Demostrar si las reglas de la gramática cumplen la **condición LL(1)**.

e. Diseñar el procedimiento del método de **Análisis Sintáctico Descendente Recursivo** correspondiente al símbolo V (se puede usar el método `EqT (t)` para equiparar el *token* `t`).

f. Terminar de rellenar la **tabla** que se indica a continuación.

	id					
V	V → id L ; V					
L						
R						

g. En relación con la tabla del apartado f:

¿Qué método de análisis sintáctico utiliza esa tabla?

¿Qué tipo de gramática es válida para construir dicha tabla?

CC BY NC SA

Apellidos: _____ **Nombre:** _____

PROCESADORES DE LENGUAJES

ANÁLISIS SEMÁNTICO

23 de enero de 2020

Observaciones: 1. Las calificaciones se publicarán hacia el 7 de febrero.
2. La revisión será hacia el 11 de febrero.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen es de 40 minutos.

3.Sea un lenguaje de programación del cual se ha extraído el siguiente subconjunto de reglas:

T → int | float | bool
F → function T id (L) begin C end
L → T id L | λ
C → E ; C
C → λ
E → id | E + E | EsInt (id)

Dicho lenguaje exige la declaración previa de variables (en otras reglas que no se muestran), admitiendo los tipos de datos entero, real y lógico (que ocupan, respectivamente, 1, 2 y 1 palabras). No se permite ninguna conversión de tipos.

El operador + se aplica a operandos numéricos y el operador EsInt devuelve un valor lógico que indica si su argumento es o no un entero. Además de las comprobaciones habituales, el lenguaje obliga a que todos los argumentos de una función sean del mismo tipo, que debe ser, además, del tipo del valor devuelto por dicha función (por lo que cada función deberá tener siempre un argumento como mínimo). La función devuelve el valor de evaluar la última expresión que aparezca en el cuerpo C (siempre ha de haber al menos una expresión).

Ejemplo de programa válido en el que no se muestra la declaración de variables previa (`var1` es entera y `var2` es lógica); la función devolverá el resultado de la expresión "`a+var1+var1`":

```
function int Lista (int a int b) begin
  var1;   EsInt(var2);   a + var1 + var1;
end
```

Se pide diseñar el Analizador Semántico para este lenguaje mediante un Esquema de Traducción para las reglas que aparecen debajo y solo en el espacio reservado. Se valorará positivamente incluir mensajes de error significativos:

T → int | float | bool *(Solamente hay que hacer una de las 3 reglas)*

E → id

CC BY NC SA

E → E + E

E → EsInt (id)

F → function T id (L) begin C end

L → T id L

L → λ

C → E ; C

C → λ

CC BY NC SA

Apellidos: _____ Nombre: _____

PROCESADORES DE LENGUAJES

ANÁLISIS SEMÁNTICO

22 de enero de 2021

Observaciones:	1. Las calificaciones se publicarán hacia el 8 de febrero. La revisión será hacia el 10 de febrero. En la web se avisará de las fechas exactas. 2. La duración de este examen es de 40 minutos. 3. No se responderán preguntas sobre el enunciado. Deberá describirse cualquier decisión de diseño asumida.

3.Sea un lenguaje de programación del cual se ha extraído el siguiente subconjunto de reglas:

T → int | float | bool | string | array [C] of T
F → function T id (L) begin S end
L → T id L | λ
S → S ; S ; | let T id | id = E
E → id G | E + E | E — E | E ∈ id | C | id
G → (A) | [A]
A → E A | λ
C → cte_ent | cte_real | cte_lóg | cte_cadena

Este lenguaje tiene las siguientes características:

- Exige declaración previa de variables, permite la recursividad y no hay anidamiento de funciones.
- Dispone de los tipos básicos de datos entero, real, cadena y lógico (que ocupan, respectivamente, 1, 2, 64 y 1 palabras).
- Dispone de un tipo compuesto de datos: vectores. Sus elementos solo pueden ser de un tipo básico y los índices (enteros) van desde 1 hasta la dimensión indicada. Ejemplo: `let array[5] of int v`.
- No realiza ninguna conversión de tipos.
- La instrucción let permite declarar una variable de cualquiera de los tipos del lenguaje.
- Las operaciones de suma (+) y resta (—) solamente se pueden realizar con datos numéricos.
- La operación E∈id indica si el valor de la expresión E es uno de los elementos del vector id.
- En la regla G, los paréntesis ( ) indican la llamada a una función y los corchetes [ ] el acceso a un elemento de un vector.

Se pide diseñar un **Esquema de Traducción** del **Analizador Semántico** de este lenguaje únicamente para las siguientes reglas en el espacio reservado (sin utilizar atributos heredados):

T → int | float | bool | string *(Solamente hay que hacer una de las 4 reglas)*

$T \rightarrow \text{array [C] of } T_1$

$E \rightarrow E_1 + E_2$

E → id

cc BY NC SA

$E \rightarrow E_1 \in id$

$F \rightarrow$ function T id (L) begin S end

$L \rightarrow T\ id\ L_1$

$L \rightarrow \lambda$

$E \rightarrow id\ G$

$G \rightarrow (\ A\ )$

$G \rightarrow [\ A\ ]$

$A \rightarrow E\ A_1$

$A \rightarrow \lambda$

CC BY NC SA

Apellidos: _____ Nombre: _____

PROCESADORES DE LENGUAJES

EXAMEN FINAL - 24 DE JUNIO DE 2021

Observaciones:	1. Las calificaciones se publicarán hacia el 8 de julio. La revisión será hacia el 12 de julio. 2. La duración del examen completo es de 2 horas. 3. Cada una de las tres preguntas se entregarán por separado 4. No se responderán preguntas sobre el enunciado debido a la situación sanitaria. Deberá describirse cualquier decisión de diseño asumida no especificada en el enunciado.

1.La Federación Española de Modalidades Atléticas (FEMA) quiere diseñar un procesador para el tratamiento automático de los resultados de los *meetings* de atletismo con vistas a cargar los resultados de cada atleta en su Base de Datos (BD). La organización de cada *meeting* enviará a la FEMA un fichero de texto donde aparecerá en cada línea el resultado de un único atleta en una prueba. Para definir cada resultado aparecerán, en cualquier orden, los siguientes elementos (cada uno de dichos elementos ha de ir seguido de al menos un espacio en blanco):

- Código de Atleta: número de licencia del atleta, que es una secuencia de 8 dígitos.
- Prueba: un código de prueba está formado por una letra mayúscula (`C`: carreras, `S`: saltos o `L`: lanzamientos), un guion y una secuencia alfanumérica no vacía en minúsculas. Al código de prueba se le añade una letra mayúscula al final para indicar el sexo (`F` o `M`). Por ejemplo, `C-100vF`, `C-4x400M`, `S-longitudM`, `L-discoF`. Se asume que estos códigos ya se han validado antes de volcarlos al fichero de texto. Se ha de tener en cuenta que en la BD de la FEMA existen los campos sexo y código de prueba por separado, por lo que el procesador deberá proporcionar la información necesaria para rellenarlos, cumpliendo obviamente las restricciones de todo Analizador Léxico.
- Marca realizada: aparece siempre como un número real que indica los segundos o los metros de la marca, siempre con dos decimales.
- Posta: solo para las pruebas de carreras de relevos, aparecerá también el número de posta realizado por el atleta, que es un entero del 1 al 4 con un solo dígito.

Ejemplos correctos:	`C-5000F 03004589 624,37`
	`L-martilloM 65,34 28500733`
	`21000456 C-4x100F 1 011,27`

Ejemplos no correctos:	`451000 C-100T F 6 10,8`

Se pide diseñar los *tokens*, gramática regular, autómata finito determinista y acciones semánticas/errores de un **Analizador Léxico** para este lenguaje, utilizando el espacio reservado para cada apartado:

a. *Tokens*

b. Gramática Regular

CC BY NC SA

c. Autómata finito determinista

d. Acciones semánticas y Errores

CC BY NC SA

Apellidos: _____ Nombre: _____

PROCESADORES DE LENGUAJES

EXAMEN FINAL - 24 DE JUNIO DE 2021

Observaciones:	1. Las calificaciones se publicarán hacia el 8 de julio. La revisión será hacia el 12 de julio. 2. La duración del examen completo es de 2 horas. 3. No se responderán preguntas sobre el enunciado debido a la situación sanitaria. Deberá describirse cualquier decisión de diseño asumida no especificada en el enunciado.

3.Sea un lenguaje de programación del cual se ha extraído el siguiente subconjunto de reglas:

T → int | real | string | boolean | λ
F → function id (L) : T begin S end ;
L → T id , L | λ
S → S S | var T id V ; | id = E ;
V → [C] | λ
E → id P | E + E | C
P → (A) | [A] | λ
A → E , A | λ
C → cte_ent | cte_real | cte_lóg

Este lenguaje tiene las siguientes características:

- Dispone de los tipos básicos de datos entero, real, cadena y lógico (que ocupan, respectivamente, 2, 4, 128 y 1 palabras).
- Dispone de un tipo compuesto de datos: vectores. Sus elementos serán de uno de los tipos básicos. Los índices (enteros) van desde 0 hasta la dimensión indicada. Ejemplo: `var int v[8];`.
- La instrucción var declara una variable local de un tipo básico o compuesto.
- El lenguaje exige declaración previa de variables con su tipo, permite la recursividad, no hay anidamiento de funciones y no tiene conversión de tipos.
- La operación de suma (+) solamente se pueden realizar entre datos numéricos o entre vectores iguales (devolviendo otro vector en el que cada elemento es la suma del elemento de esa posición de ambos vectores)
- En la regla P, los paréntesis ( ) indican la llamada a una función. Ejemplo, `f(8,)`, `g()`, `h(8+b,g(),a,)`. Y los corchetes [ ] indican el acceso a un elemento de un vector. Ejemplo, `v[3,]`, `w[a,]`, `u[8+v[0,],]`.

Se pide diseñar una **Definición Dirigida por la Sintaxis** del **Analizador Semántico** de este lenguaje únicamente para las siguientes reglas en el espacio reservado:

T → string

T → λ

S → var T id V ;

CC BY NC SA

$V \rightarrow [\ C\ ]$

$V \rightarrow \lambda$

$L \rightarrow T\ id\ ,\ L_1$

$F \rightarrow function\ id\ (\ L\ )\ :\ T\ begin\ S\ end\ ;$

$E \rightarrow E_1 + E_2$

$E \rightarrow id\ P$

$P \rightarrow (\ A\ )$

$P \rightarrow [\ A\ ]$

$A \rightarrow E\ ,\ A_1$

$A \rightarrow \lambda$

cc BY NC SA

Apellidos: _____ Nombre: _____

PROCESADORES DE LENGUAJES

EXAMEN FINAL - 24 DE JUNIO DE 2021

Observaciones:	1. Las calificaciones se publicarán hacia el 8 de julio. La revisión será hacia el 12 de julio. En la web se avisará de las fechas exactas. 2. La duración del examen completo es de 2 horas. 3. Cada una de las tres preguntas se entregarán por separado 4. No se responderán preguntas sobre el enunciado debido a la situación sanitaria. Deberá describirse cualquier decisión de diseño asumida no especificada en el enunciado.

2.Sea el siguiente fragmento de gramática de un lenguaje de programación:

$S \rightarrow$ T id H S | λ | id = E
$T \rightarrow$ number | boolean
$H \rightarrow \lambda$ | = E | (L) S end | [C]
$E \rightarrow$ id P | C
$P \rightarrow$ (A) | [A] | λ
$A \rightarrow$ E , A | λ
$C \rightarrow$ cte

Se pide:

a. Comprobar si se cumple la **condición LL(1)** para las reglas de S, H, E y A.
b. Construir, de la tabla de un **Analizador Sintáctico Descendente**, las filas correspondientes a S, H y A.
c. Construir, del autómata reconocedor de los prefijos viables de un **Analizador Sintáctico Ascendente**, el estado inicial y los inmediatos siguientes (es decir, los que se obtienen desde el inicial con una sola transición).
d. Analizar los **conflictos** posibles del estado inicial del autómata del apartado anterior.
e. Construir, de la tabla de un **Analizador Sintáctico Ascendente**, las filas correspondientes a los estados I_0 e I_1 (es decir, los estados inicial y final del analizador).

CC BY NC SA

PROCESADORES DE LENGUAJES

ANÁLISIS LÉXICO, TABLA DE SÍMBOLOS Y ANÁLISIS SINTÁCTICO

22 de noviembre de 2021

Observaciones: 1. Las calificaciones se publicarán hacia el 3 de diciembre. La revisión será hacia el 9 de diciembre. En la web se avisará de las fechas exactas.
2. La duración de este examen es de 40 minutos.
3. No se responderán preguntas sobre el enunciado. Deberá describirse cualquier decisión de diseño que no esté especificada.
4. Las dos preguntas tienen la misma puntuación.

Un lenguaje de programación tiene la siguiente estructura:

$S \rightarrow$ id := E | id (L)
$L \rightarrow$ E A L | λ
$A \rightarrow$:= E | λ
$E \rightarrow$ H — E | id
$H \rightarrow$ — H | entero

Dicho lenguaje tiene las siguientes características:

- Los nombres de los identificadores deben comenzar por letra o subrayado y pueden ir seguidos de cualquier cantidad de dígitos o letras, pudiendo terminar por dígito, letra o subrayado. El lenguaje no obliga a realizar la declaración previa de variables.
- Los números enteros se tienen que poder representar con 2 palabras de 16 bits cada una.
- Los elementos del lenguaje pueden ir separados por delimitadores.

Se pide contestar cada una de las dos preguntas siguientes en **hojas separadas:**

1. Realizar el Diseño completo del **Analizador Léxico** para el lenguaje descrito (se deben incluir todos los *tokens* que se observan en la gramática dada), detallando todos los accesos a la Tabla de Símbolos.

2. Contestar las siguientes preguntas en relación con el **Análisis Sintáctico**, teniendo en cuenta que las tablas que se piden en los apartados b) y c) hay que rellenarlas en las **plantillas** que aparecen al reverso de esta hoja (se pueden añadir filas o columnas si se necesitaran):
   a. Para diseñar un **Analizador Ascendente**, construir el fragmento de autómata que se obtiene con los siguientes conjuntos de ítems: I_0, I_1=Goto (I_0, S), I_2=Goto (I_0, id) y todos los Goto posibles de I_2.
   b. Completar solo la parte de la tabla ACCIÓN de un **Analizador Ascendente** LR(1) que se obtiene a partir del fragmento de autómata del apartado anterior.
   c. Factorizar la gramática y colocar todas las reglas correspondientes a los símbolos S, L y A en la tabla de un **Analizador Descendente Predictivo Tabular**.
   d. Completar la función correspondiente a E para un **Analizador Descendente Predictivo Recursivo**.

CC BY NC SA

Apellidos: …….. **Nombre:** ……

	id	:=	(	)	–	entero	$	
0								
1								
2								
3								
...								
...								
...								

	id	:=	(	)	–	entero	$	
S								
L								
A								

CC BY NC SA

PROCESADORES DE LENGUAJES

EXAMEN FINAL

19 de enero de 2022

Observaciones: 1. Las calificaciones se publicarán hacia el 3 de febrero. La revisión será hacia el 7 de febrero. En la web se avisará de las fechas exactas.
2. La duración de este examen es de 40 minutos por cada pregunta.
3. No se responderán preguntas sobre el enunciado. Deberá describirse cualquier decisión de diseño que no esté especificada.
4. Las tres preguntas tienen la misma puntuación y se entregan por separado.

1. Realizar el Diseño completo del **Analizador Léxico** para el fragmento de lenguaje descrito a continuación (se deben incluir todos los *tokens* que se observan en la gramática dada), detallando todos los accesos a la Tabla de Símbolos. El lenguaje de programación tiene la siguiente estructura:

```
S → id := E  |  id ( L )
L → E A L  |  λ
A → := E  |  λ
E → E + E  |  + E  |  id  |  ++ id  |  real
```

Dicho lenguaje tiene las siguientes características:

- Los elementos del lenguaje pueden ir separados por delimitadores.
- Los nombres de los identificadores pueden comenzar por letra o dígito y pueden ir seguidos de cualquier cantidad de dígitos, letras o subrayados, no pudiendo terminar por subrayado. El lenguaje no obliga a realizar la declaración previa de variables. Ejemplos: `z314`; `19`; `a_b`; `9_A_9`.
- Los números reales utilizan la coma (`,`) como separador de la parte entera y decimal (la parte entera puede omitirse). Ejemplos: `3,1416`; `19,80`; `0,1`; `,999`.
- Todo número real debe ser inferior a 10^6.

2. Sea la siguiente gramática:

```
S → A B C  |  T p S  |  λ
A → q T p  |  λ
B → z  |  λ
T → x  |  λ
C → p B
```

Se pide:

a. Comprobar detalladamente la **condición LL(1)** e indicar si la gramática es LL(1).
b. Construir la **tabla** del **Analizador Descendente LL(1)**, independientemente del resultado obtenido en el apartado anterior.
c. Diseñar la función correspondiente a `A` para un **Analizador Descendente Recursivo**.
d. Construir los estados I_0 hasta I_6 del **autómata reconocedor de los prefijos viables** de un Analizador LR(1).
e. Construir las filas de la **tabla de un Analizador LR(1)** que se obtengan de los estados del apartado anterior (incluyendo toda la información que se pueda obtener de dichos estados, aunque esté incompleta).

CC BY NC SA

PROCESADORES DE LENGUAJES

EXAMEN FINAL

28 de junio de 2022

Observaciones: 1. Las calificaciones se publicarán hacia el 4 de julio. La revisión será hacia el 6 de julio. En la web se avisará de las fechas exactas.
2. La duración de este examen es de 2 horas.
3. No se responderán preguntas sobre el enunciado. Deberá describirse cualquier decisión de diseño que no esté especificada.
4. Las tres preguntas tienen la misma puntuación y se entregan por separado.

1. Realizar el Diseño completo del **Analizador Léxico** para el fragmento de lenguaje descrito a continuación (se deben incluir todos los *tokens* que se observan en la gramática dada), detallando todos los accesos a la Tabla de Símbolos. El lenguaje de programación tiene la siguiente estructura:

```
S → id = E
E → id | E * E | * E | E ** E | cte_real | id [ L ]
L → E , L | E
```

Dicho lenguaje tiene las siguientes características:

- Los elementos del lenguaje pueden ir separados por delimitadores.
- El operador binario `*` representa el producto, el operador unario `*` representa el acceso a puntero y el operador binario `**` representa la potencia. Los corchetes `[ ]` indican el acceso a un elemento de una matriz.
- Los nombres de las variables pueden comenzar por letra o subrayado y pueden ir seguidos de cualquier cantidad de dígitos, letras o subrayados, no pudiendo tener dos subrayados consecutivos. Un nombre no puede tener más de 32 caracteres. El lenguaje no obliga a realizar la declaración previa de variables. Ejemplos: `z314`; `L9`; `a_b`; `p`; `_0_`.
- Los números reales utilizan el punto (`.`) como separador de la parte entera y decimal (la parte decimal puede omitirse). El valor de un número real no puede ser superior a diez millones. Ejemplos: `3.1416`; `19.80`; `1.`; `0.99`.

2. Sea la siguiente gramática:

```
S → a D A b | C | b A D a
A → a A | b
C → c D | λ
D → b D b | λ
```

Se pide:

a. Construir la **tabla** completa de un **Analizador Descendente** LL(1).
b. A la vista de la tabla LL(1), justificar razonadamente si la gramática es o no **LL(1)**.
c. Diseñar la función correspondiente a `S` para un **Analizador Descendente Recursivo**.
d. Construir el estado inicial (I_0) del **autómata reconocedor de los prefijos viables** de un Analizador LR(1). Añadir todos los estados que se obtienen directamente de I_0 mediante una operación *Goto*, así como todos los estados que contengan el ítem `D → · b D b`.
e. Realizar el estudio de los **conflictos** sobre los estados obtenidos en el apartado anterior.

CC BY NC SA

3. Sea un lenguaje de programación del cual se muestran algunas reglas de la gramática que lo genera:

P → S P | λ
S → A := Q | for id in [L]: S | S ; S
A → id | A , A
Q → E | Q , Q
L → E | L , L
E → id | cte_entera | true | E + E

El lenguaje presenta las siguientes características:

- El lenguaje permite tipos de datos lógico, entero y real. El tipo lógico ocupa 2 bytes, el entero 4 y el real 8.
- No existe la declaración previa explícita de variables. Las variables se declaran implícitamente al aparecer como variable índice en un for o al asignarles por primera vez un valor en la sentencia de asignación; su tipo se modifica cada vez que se le asigna un valor de un tipo distinto. En el lado derecho de las asignaciones o en cualquier otra sentencia (salvo la variable índice del for) solamente pueden aparecer variables que hayan sido previamente declaradas implícitamente (variables a las que se les haya asignado previamente un valor).
- La sentencia de asignación es múltiple: asigna el primer valor del lado derecho (Q) a la primera variable del lado izquierdo (A) y así sucesivamente. Debe haber el mismo número de elementos en ambos lados.
- La sentencia for recibe una variable índice, que no necesita estar declarada previamente, y una lista (L). Todos los elementos de la lista deben ser del mismo tipo. Durante las iteraciones del for, se asigna sucesivamente a la variable índice cada uno de los valores de la lista.
- El lenguaje no realiza ninguna conversión de tipos.

Ejemplo de programa válido:

```
a, b:= 12, true;
for i in [a, 9, 7+a]:
      a:= a + i
```

Teniendo en cuenta que el Analizador Léxico se encarga de insertar los lexemas en la Tabla de Símbolos y que NO hay que realizar en esta solución la gestión del desplazamiento de las variables, se pide:

Diseñar el **Analizador Semántico** para este lenguaje mediante un **Esquema de Traducción SOLO** para las reglas que aparecen debajo y solo en el espacio reservado. Se valorará positivamente incluir mensajes de error significativos. Si hace falta realizar alguna suposición no deducible del enunciado, se deberá especificar.

¿En qué regla se haría la creación de la tabla de símbolos? Define la regla semántica:

S → A := Q

A → id

cc BY NC SA

A → A , A

Q → E

Q → Q , Q

S → for id in [L]: S;

L → E

L → L, L

S → S ; S

E → id

E → true

CC BY NC SA

PROCESADORES DE LENGUAJES

13 de enero de 2023

Observaciones: 1. Las calificaciones se publicarán hacia el 27 de enero. La revisión será hacia el 1 de febrero. En la web se avisará de las fechas exactas.
2. La duración de este examen es de 2 horas.
3. Las tres preguntas tienen la misma puntuación y se entregan por separado.

1. La Dirección General de Carreteras (DGT) quiere automatizar la gestión de su fichero diario de incidencias en las carreteras del estado. Quiere diseñar un Analizador Léxico que lea el fichero de incidencias y cargue la información en una Base de Datos.

El fichero de incidencias tiene, en cada línea, toda la información de las incidencias de una carretera. En la línea constan el código de la carretera (una letra, un guion opcional y 2 dígitos) y los puntos kilométricos donde ha habido incidencias (un real siempre con parte entera y 1 único digito decimal) seguidos de cadenas de caracteres, con una descripción adicional, encerrada entre comillas ("…"). Estos elementos están separados por uno o más delimitadores.

En el espacio reservado, se pide realizar el diseño del **Analizador Léxico** (solamente los *tokens* completos, la Gramática y el Autómata, minimizando la necesidad de acciones semánticas):

Ejemplo de fichero válido:
```
N-53     123,2 "ok"  225,2   "congestión en Villanueva"
                C23     55,0        "Problema resuelto: 14h"
```

a. ***Tokens* completos**

b. **Gramática**

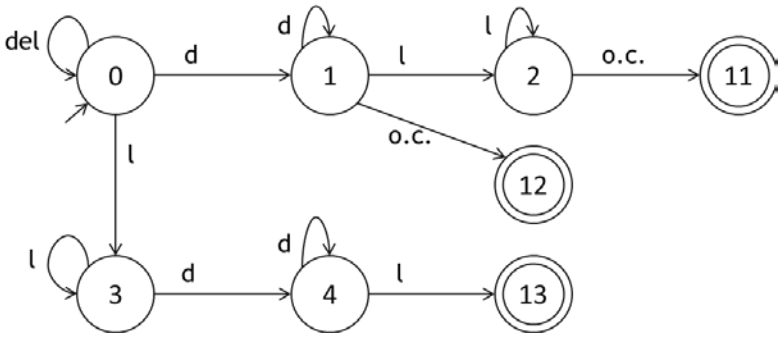
c. **Autómata**

CC BY NC SA

d. Para otro problema (lectura de matrículas), la DGT recibe un fichero con la matrícula de los coches que pasan en tramos con detección automática de matrícula. Dichos lectores producen ficheros donde consta el código de 7 dígitos del aparato lector de matrículas y, a continuación, la lista de matrículas detectadas. Las matrículas tienen dos formatos posibles:
   - Cuatro dígitos seguidos por tres letras: `1234MCB`, `0909BCF`.
   - Una o dos letras seguidas por cuatro dígitos, finalizando con una letra: `M3456F`, `AV1999A`. Para este formato se deberá comprobar que las letras del principio corresponden a una provincia correcta. Se dispone de una tabla con todas las provincias (`Tabla-Prov`).

   El formato de los *tokens* que se debe emitir es:
   <lector, código>, <matrícula, código> // *ejemplos:* <*lector, "0012345"*>, <*matrícula, "M3456F"*>

   Se han realizado ya los pasos previos del diseño del Analizador Léxico y se pide diseñar las **Acciones Semánticas** (sólo las correspondientes a las transiciones indicadas, usando los espacios reservados) del siguiente autómata para que se produzca el comportamiento descrito.



del: blanco, tab, EOL
l: letras mayúsculas
d: dígitos 0..9

0:0	
0:1	
1:1	
1:2	
2:2	
2:11	
1:12	
0:3	
4:13	

CC BY NC SA

PROCESADORES DE LENGUAJES

13 de enero de 2023

2. Responde cada apartado en el hueco correspondiente.

a. **Factorizar** la siguiente gramática (G_1):

S → while id { C } else { C } S → while id do S S → do S while id S → id = id C → id [cte_ent] = id C → S ; C C → λ	

Sea la siguiente gramática (G_2):

1) S → if id : R
2) S → λ
3) R → S ;
4) R → { C } M L
5) L → else : { C }
6) L → λ
7) C → S ; C
8) C → λ
9) M → elif id : { C } M
10) M → λ

b. Calcular los conjuntos **First** y **Follow** de G_2:

	FIRST		FOLLOW
S		S	
R		R	
L		L	
C		C	
M		M	

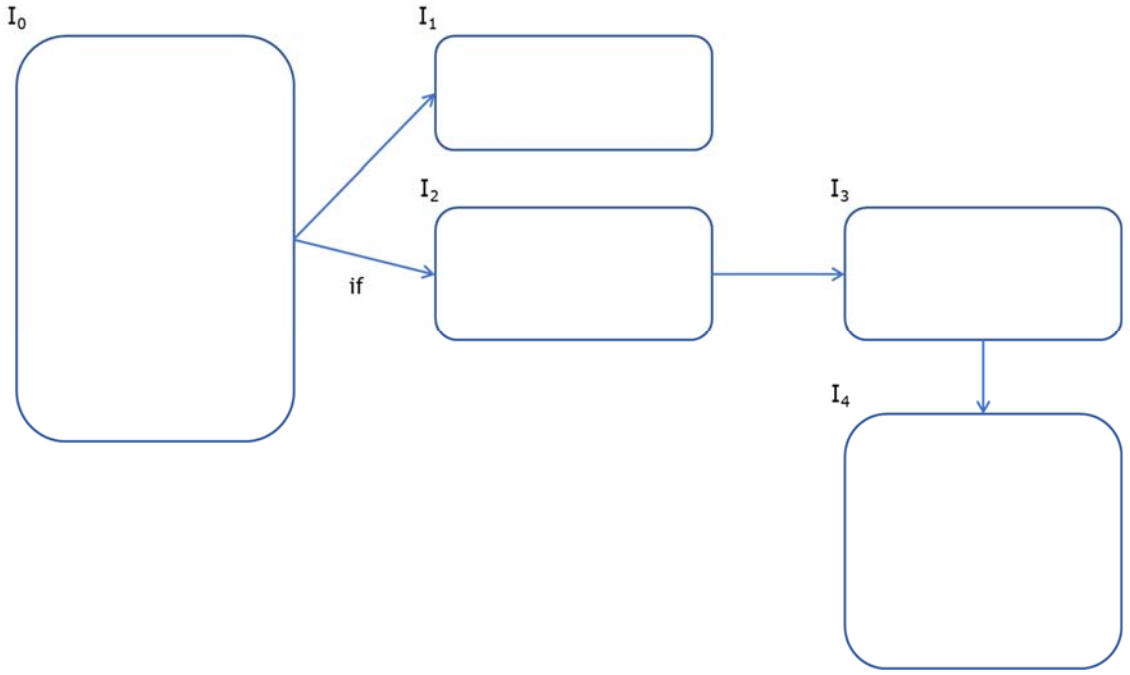
c. Construir la **tabla** de un **Analizador LL(1)** para G_2:

	if	id	:	;	{	}	else	elif	$
S									
R									
L									
C									
M									

Explicar, a la vista de la tabla, si la gramática G_2 es o no LL(1):

CC BY NC SA

d. Completar la plantilla con los primeros estados del **autómata** reconocedor de prefijos viables de un **Analizador Ascendente** para G_2, hasta tener todos los estados a los que se transita directamente desde el estado I_4:



e. Con los estados obtenidos del autómata, completar todo lo que se pueda en las filas en blanco de la **tabla** del **Analizador Ascendente**:

	if	id	:	;	{	}	else	elif	$	S	R	C	M
0													
1													
2													
3													
4													
5													
6				d 8									
7													
8				r 3					r 3				

f. Detallar el **reconocimiento** de la cadena "`if id : ;`" por el Analizador Ascendente, indicando el *parse*, si la cadena pertenece al lenguaje, o un mensaje de error adecuado, si no pertenece.

CC BY NC SA

PROCESADORES DE LENGUAJES

13 de enero de 2023

3. Sea un lenguaje de programación del cual se muestran algunas reglas de la gramática que lo genera:

$P \rightarrow D\ S\ .$
$D \rightarrow D\ D \mid \text{var}\ T : V\ ;$
$V \rightarrow \text{id} \mid \text{id}\ ,\ V$
$T \rightarrow \text{int} \mid \text{real} \mid \text{char} \mid \text{array}\ [\ C\ ]\ \text{of}\ T$
$S \rightarrow S\ S \mid \text{id} := E\ ; \mid \text{break}\ ; \mid \text{switch}\ E\ \text{begin}\ L\ \text{end}\ ;$
$L \rightarrow \text{case}\ C : S\ L \mid \lambda$
$E \rightarrow E \oplus E \mid \text{id} \mid E\ [\ E\ ] \mid C$
$C \rightarrow \text{cte\_entera} \mid \text{cte\_real} \mid \text{cte\_carácter}$

El lenguaje presenta las siguientes características:

- El lenguaje tiene un único ámbito y exige declaración previa de variables, que pueden ser enteras, reales, carácter o vectores. El tipo entero ocupa 2 bytes, el real 4 y el carácter 1. Los índices de los vectores, que deben ser siempre enteros, van desde 1 hasta el valor de su dimensión.
- La sentencia switch recibe una expresión (E) que solo puede ser entera o carácter. Todos los tipos de las constantes (C) de los case deben coincidir con el tipo de la expresión de su switch.
- La sentencia break solamente puede utilizarse dentro de una sentencia switch.
- El lenguaje no dispone de ninguna conversión de tipos.
- La operación suma de vectores (⊕) permite obtener un nuevo vector cuyos elementos son la suma dos a dos de los elementos de dos vectores iguales (mismo tamaño y tipo de elementos), siempre que sus elementos sean numéricos.

Se pide diseñar el **Analizador Semántico** para este lenguaje mediante una **Definición Dirigida por la Sintaxis** solo para las reglas que aparecen debajo y solo en el espacio reservado.

$P \rightarrow D\ S\ .$

$D \rightarrow \text{var}\ T : V\ ;$

$V \rightarrow \text{id}\ ,\ V$

$T \rightarrow \text{int}$

$T \rightarrow \text{array}\ [\ C\ ]\ \text{of}\ T$

CC BY NC SA

$S \rightarrow$ switch E begin L end ;

$L \rightarrow$ case C : S L

$L \rightarrow \lambda$

$S \rightarrow$ break ;

$S \rightarrow S\ S$

$S \rightarrow$ id := E ;

$E \rightarrow E \oplus E$

$E \rightarrow E\ [\ E\ ]$

Indicar un mensaje de error adecuado para uno de los errores detectados en cada una de las siguientes reglas:

- $S \rightarrow$ switch E begin L end ;
- $E \rightarrow E\ [\ E\ ]$
- $E \rightarrow E \oplus E$

CC BY NC SA