COMPILADORES

Examen del primer parcial. 6 de febrero de 2009

Observaciones: 1. Las calificaciones del primer parcial se publicarán en la primera quincena de marzo, siendo la revisión al menos 2 días después. Las fechas exactas se avisarán en `http://www-lt.ls.fi.upm.es/compiladores`.
2. La duración de este examen será de 2 horas.
3. Cada ejercicio deberá entregarse en hojas separadas.

1. Un lenguaje tiene los siguientes elementos:

- Operadores aritméticos: `+`, `-`, `*`, `/`, `**`
- Operadores lógicos: `.AND.`, `.OR.`, `.NOT.`
- Identificadores que comienzan por una letra y van seguidos de un máximo de 7 letras o dígitos
- Palabras reservadas en mayúsculas, como `IF`, `THEN`, `FOR`, `WHILE`, `INTEGER`...
- Números reales según la expresión regular: $d^*.d^+$

Teniendo en cuenta que el lenguaje distingue mayúsculas de minúsculas, se pide construir un **Analizador Léxico** para este lenguaje (*tokens*, gramática, autómata finito determinista y acciones semánticas).

(4 puntos)

2. Sea la siguiente gramática:

E → T R
R → + T R | - T R | λ
T → id

Se pide:

a. Construir las tablas de un analizador sintáctico **SLR**, dando también el autómata en la plantilla adjunta (plantilla que contiene errores y está incompleta).

b. Construir la tabla de un analizador sintáctico **LL(1)**.

c. Contestar brevemente a las siguientes preguntas tanto para el SLR como para el LL según las respuestas de los apartados a y b. Si alguna de ellas no tiene sentido para alguno de los analizadores, explicar la razón:

1. ¿En qué casos podría haber conflictos? Estudiar cada uno de esos casos y decir si hay o no conflicto.
2. ¿Es válida la gramática para ese tipo de analizadores o aparece algún problema?
3. ¿Cómo se sabe en el estado I_2 cuál de las reglas de R hay que aplicar?
4. ¿Cómo se sabe si la regla R → + T R se aplica en el estado I_2, en el I_7 ó en el I_8?
5. ¿Hay problema con la recursividad de R → + T R y R → - T R? En caso afirmativo, decir cuál y si se puede resolver de alguna manera.
6. ¿En algún caso se puede aplicar la regla R → λ? En caso afirmativo, decir cuándo; en caso negativo, explicar cuál es el problema.
7. ¿Cómo sabe el analizador en un momento dado si tiene que aplicar la regla R → + T R o la regla R → - T R?
8. ¿Qué hace el analizador cuando R está en la cima de la pila e id en la entrada?

(6 puntos)

COMPILADORES

Examen final. Segundo parcial. 6 de febrero de 2009

Observaciones: 1. Se estima que se publicarán las notas del examen final a partir del 20 de febrero, siendo la revisión al menos 2 días después. Las fechas exactas se avisarán en `http://www-lt.ls.fi.upm.es/compiladores`.
2. La duración de este examen será de 2 horas.
3. Cada ejercicio deberá entregarse en hojas separadas.

1. De una gramática se han entresacado las siguientes reglas:

S → id := E
E → id (L) | id | E * E
L → E | λ

Las variables son enteras y reales, no hay conversión automática de tipos e id (L) puede representar una llamada a una función (con uno o ningún parámetro) o un acceso a un elemento de un vector (siendo necesario indicar el índice). Se pide construir una **Definición Dirigida por la Sintaxis** para realizar el **Análisis Semántico** y la **Generación de Código Intermedio** (explicando brevemente los atributos y funciones utilizadas).

(6 puntos)

2. Sea el siguiente programa:

```
Program Cálculo-día
  Type fecha: Record= día: integer, mes: integer, año: integer;
  Global   t, a: integer
           hoy: fecha
  Procedure P-año (Ref z: fecha) /* parámetro por referencia
  Begin P-año
       z.año:= z.año + 1
  End P-año
  Procedure P-mes (Ref b: integer; Ref x: fecha)   /* parámetros por referencia
       Procedure P-día ()
       Begin P-día
            While ((a / t) > 8) Do
                 a:= a - t
                 Call P-año (x)
            End While
            x.día:= x.día + 1;
       End P-día
  Begin P-mes
       If ((a / t) < x.mes)  Then x.mes:= a / t
                             Else Call P-día ()
  End P-mes
Begin Cálculo-día
  a:= 300
  t:= 30
  hoy.día:= 6
  hoy.mes:= 2
  hoy.año:= 2009
  Call P-mes (a, hoy)
  Imprime (hoy.día, hoy.mes, hoy.año)
End Cálculo-día
```

Teniendo en cuenta que los enteros y las direcciones ocupan 2 bytes, se pide:

a. Diseñar el **Registro de Activación** general para este lenguaje.

b. Realizar una **traza de ejecución** del programa representando el contenido completo de la pila e indicando el resultado de la sentencia "`Imprime`".

(4 puntos)

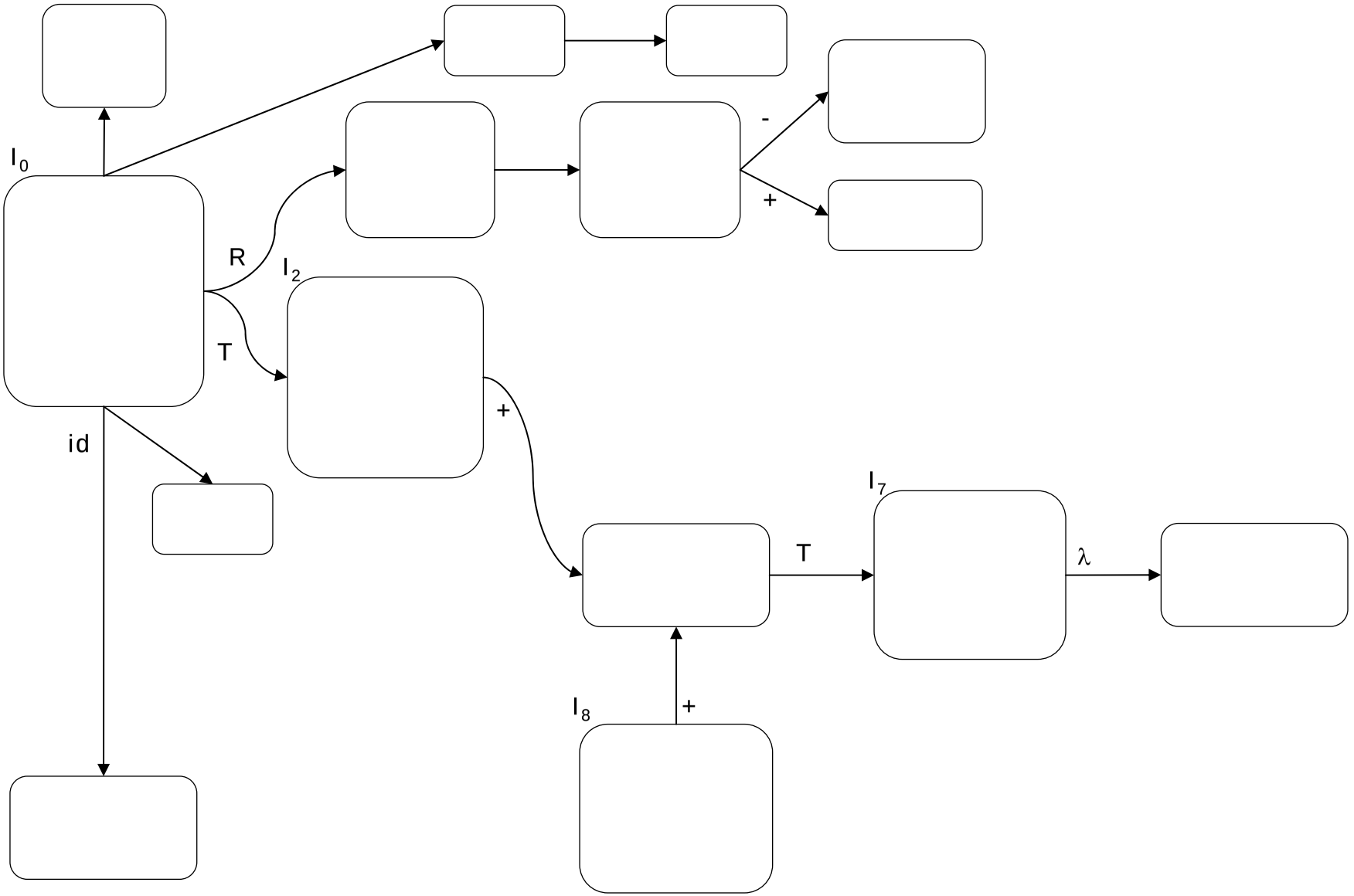
Compiladores

6-febrero-2009, Ejercicio 2.a

Hoja de Respuesta

Apellidos: .....

Nombre: .....



COMPILADORES
Segundo parcial, 22 de junio de 2009

Observaciones: 1. Fecha **estimada** de publicación de las calificaciones: 9 de julio.
2. Fecha **estimada** de la revisión: 13 de julio.
3. En http://www-lt.ls.fi.upm.es/compiladores 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.
5. Cada ejercicio deberá entregarse en hojas separadas.

1. De un lenguaje se ha extraído el siguiente fragmento de gramática:

S → id := E | if (E) then S | switch (E) { L } | S ; S
L → case C : S ; L | λ
E → C | id | E oprel E | E + E
C → cte | cte : cte : cte

Se pide diseñar el **Analizador Semántico** construyendo una **Definición Dirigida por la Sintaxis**. Téngase en cuenta que:

- El lenguaje define las constantes enteras y el tipo entero con 2 bytes
- La expresión de la sentencia if ha de ser lógica
- La máquina objeto dispone de enteros de 2 y 4 bytes
- La sentencia switch admite como expresión un entero o una hora
- Las constantes de tipo hora (cte : cte : cte) deben ser correctas
- Las horas se tienen que almacenar internamente como el número de segundos transcurridos desde la media noche
- No hay conversiones automáticas de tipos
- Se admiten operaciones de comparación y de suma entre enteros y horas.

(6 puntos)

2. Sea el siguiente fragmento de un programa:

```
Program Main
   Var  i: Integer
   Function A (Ref x: Integer)
      Procedure B (Ref y: Integer)
         Var j: integer
      Begin B
         If (y <= 0) Then Return
         Else Begin
                y:=y-2
                i:=y-1
              End
         j:=A (i)
      End B
   Begin A
      x:=x-1
      B (x)
      Return 2*x
   End A
Begin Main
   i:= 2
   i:= A (A (i))
   Imprime (i)
End Main
```

El compilador y el lenguaje tienen las siguientes características:

- El compilador no realiza ningún tipo de optimización
- El lenguaje tiene estructura de bloques
- El modo de paso de los parámetros es por referencia
- Las enteros y las direcciones ocupan 2 bytes

Se pide:

a. Diseñar el **Registro de Activación** general para este lenguaje.

b. Realizar una **traza de ejecución** del programa representando el contenido completo de la pila, e indicando el resultado de la sentencia "Imprime".

(4 puntos)

COMPILADORES

Primer parcial, 22 de junio de 2009

Observaciones: 1. Fecha **estimada** de publicación de las calificaciones: 9 de julio.
2. Fecha **estimada** de la revisión: 13 de julio.
3. En http://www-lt.ls.fi.upm.es/compiladores 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 1½ horas.
5. Cada ejercicio deberá entregarse en hojas separadas.
6. Los dos ejercicios tienen la misma puntuación.

1. Se tiene un lenguaje en el que hay identificadores, palabras reservadas, números enteros, operadores aritméticos de suma y resta y lo que se explica a continuación:

- Los identificadores están formados por letras y tienen una longitud máxima de 6 caracteres. Si empiezan por alguna de las letras `I, J, K, L, M, N` son enteros salvo que se declaren explícitamente de otro tipo.
- La declaración explícita de un identificador se construye con la palabra reservada que indica el tipo (`REAL`, `INTEGER` o `CHARACTER *long`), seguida de la lista de identificadores separados por comas. En la declaración de una variable de tipo carácter, se indica la longitud de la variable con un asterisco y un número entero de no más de 3 dígitos (debe ser una comprobación léxica).
- Las palabras reservadas están formadas por letras.
- Los números pueden ser positivos y negativos.

Se pide:

a. Construir el **Analizador Léxico** para este lenguaje (Gramática Regular, *tokens*, AFD, Acciones Semánticas e indicar 5 posibles casos de error), teniendo en cuenta que se debe introducir en la Tabla de Símbolos toda la información posible.

b. Si el lenguaje tuviera también la operación de producto, ¿qué cambios habría que realizar en el diseño de este Analizador Léxico?

2. Sea la siguiente gramática:

```
S      → while COND do CUERPO end  |  goto id  |  λ
COND   → id
CUERPO → begin REST end
REST   → S REST  |  λ
```

Se pide:

a. Completar en la plantilla adjunta el **Autómata Reconocedor de Prefijos Viables** (método SLR). La plantilla puede contener errores y/o omisiones.

b. Analizar la posible existencia de **conflictos** en el autómata anterior. Si hay algún conflicto, plantear una sentencia donde, caso de usarse dicho autómata, se llegaría a una situación de conflicto y explicar por qué se produce.

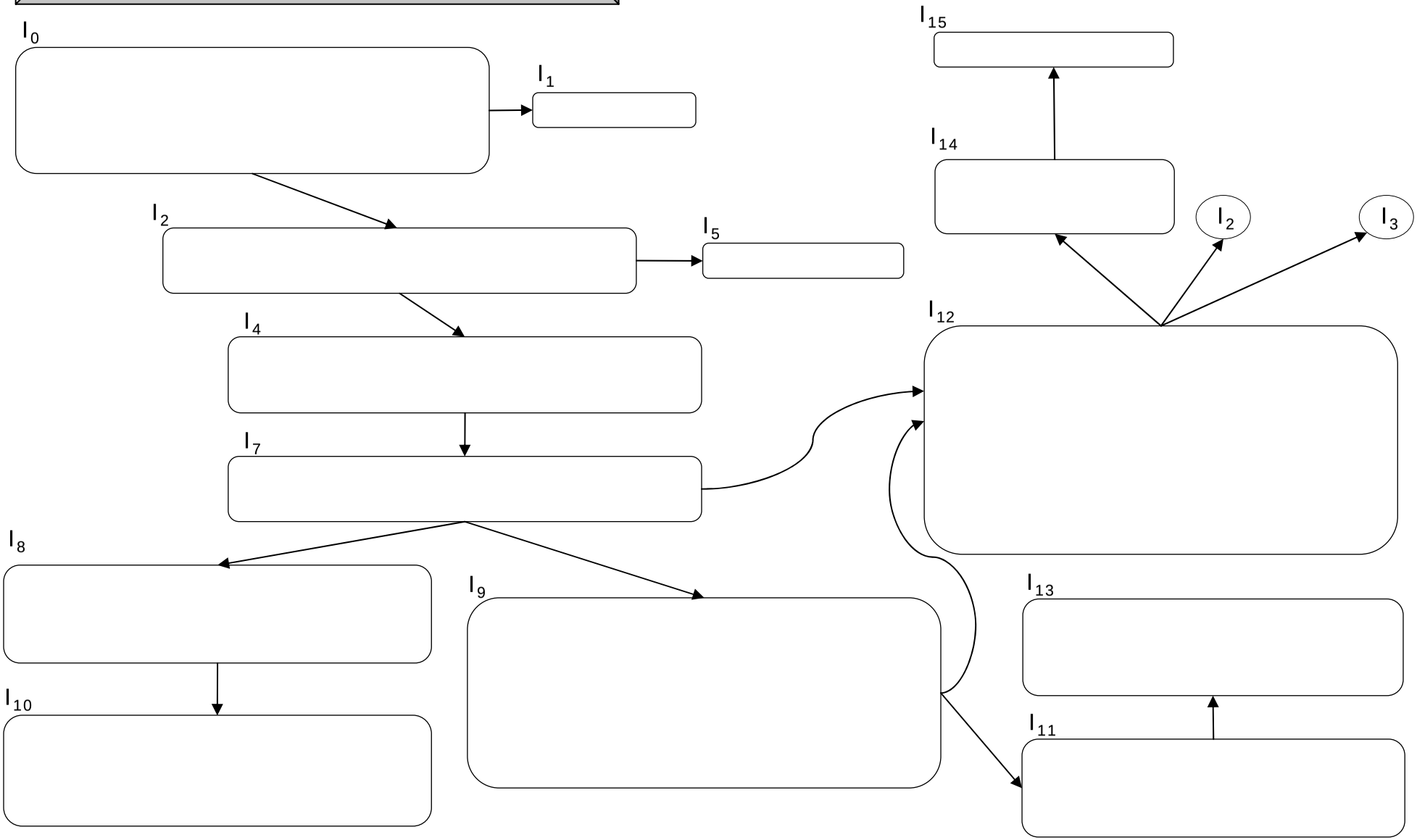
Compiladores

22-junio-2009, Ejercicio 2.a

Hoja de Respuesta

Apellidos: .....

Nombre: .....



COMPILADORES

Examen final, 1 de septiembre de 2009

Observaciones: 1. Fecha **estimada** de publicación de las calificaciones: 15 de septiembre. Fecha **estimada** de la revisión: 17 de septiembre. En http://www-lt.ls.fi.upm.es/compiladores 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 2¼ horas.
5. Cada ejercicio deberá entregarse en hojas separadas.

1. La descripción de un fragmento de un lenguaje de programación viene dada por: *(3 puntos)*

```
S → id = E | if ( E ) then S | switch ( E ) { L } | S ; S
E → C | id | E oprel E | E + E
C → cte | cte:cte:cte
cte → dígito dígitos
dígito → 0 | 1 | ... | 9
digitos → dígito dígitos | λ
id → letra letras
letra → a | b | c | ... | z | A | B | C | ... | Z
letras → letra letras | λ
L → case C = S ; L | λ
oprel → <> | =
```

Se pide diseñar un **Analizador Léxico** (gramática regular, *tokens*, autómata finito determinista, acciones semánticas) correspondiente a dicho lenguaje. El lenguaje presenta las siguientes características:

- El lenguaje define las constantes enteras y el tipo entero con 2 bytes
- La expresión de la sentencia if ha de ser lógica
- La máquina objeto dispone de enteros de 2 y 4 bytes
- La sentencia switch admite como expresión un entero o una hora
- Las constantes de tipo hora (cte:cte:cte) deben representar horas correctas (dando un error léxico en caso contrario), y cada parte de la hora debe ir escrita con dos dígitos
- Las horas tienen que almacenarse internamente como el número de segundos transcurridos desde la media noche
- Las palabras clave del lenguaje son reservadas
- Los elementos pueden ir separados por delimitadores
- No hay conversión automática de tipos
- Se admiten operaciones de comparación y de suma entre enteros y horas

2. Sea la siguiente gramática: *(3 puntos)*

```
S → begin DEF end
DEF → Deffuente DEF | Defvector DEF | Defestilo DEF | λ
Deffuente → fuente id
Defvector → vector integer : integer
Defestilo → estilo integer
```

a. Demostrar que se trata de una **gramática LL**, válida para construir un Analizador Descendente LL(1).
b. Construir la **tabla del Analizador Sintáctico Descendente** LL.
c. Definir una **gramática** equivalente a la anterior, pero que **no sea LL**, presentando al menos recursividad por la izquierda y no estando factorizada.
d. Construir la **tabla de Precedencia de Operador** y razonar si la siguiente gramática es válida o no para construir un Analizador Sintáctico con este método. *Para construir la tabla de precedencia, deben colocarse primero los operadores y a continuación las letras, según su orden de aparición en la gramática*:

```
S → A * B + B | B + A * A
A → a + a
B → b * b
```

3. Dado el siguiente fragmento de gramática de un lenguaje: *(4 puntos)*

```
S → id := E
E → id (L) | id
L → E, L | E
```

donde id(L) puede ser una llamada a función o el acceso a un elemento de un vector unidimensional de índices enteros. Teniendo en cuenta que no hay conversiones automáticas de tipos y que todo identificador tiene que estar declarado previamente, se pide diseñar una **Definición Dirigida por la Sintaxis** con el **Comprobador de Tipos** y el **Generador de Código Intermedio** (código de 3 direcciones).

COMPILADORES

Examen del primer parcial. 11 de febrero de 2010

Observaciones: 1. Las calificaciones del primer parcial se publicarán aproximadamente a mediados de marzo. Se avisará en `http://www-lt.ls.fi.upm.es/compiladores`.
2. La duración de este examen será de 1¾ horas.
3. Cada ejercicio deberá entregarse en hojas separadas.
4. Los dos ejercicios tienen la misma puntuación.

1. Se tiene un lenguaje con números, constantes de tipo cadena, variables y una serie de operadores y que no cuenta con conversión automática de tipos. Todos los elementos del lenguaje tienen que ir separados obligatoriamente por al menos un delimitador.

Los números pueden ser tanto enteros como reales. Los números enteros están formados por una secuencia de dígitos. Los números reales han de llevar un punto ("`.`") y pueden no tener parte entera, pero han de tener obligatoriamente parte real (son correctos, por ejemplo, "`.67`" y "`128.6`", pero no "`14.`").

Las constantes de tipo cadena van encerradas entre comillas dobles ("). Dentro de la cadena puede haber cualquier carácter excepto las propias comillas.

Las variables pueden ser de tipo cadena, vector, entero o real. El lexema de la variable estará formado por una combinación de letras y dígitos de cualquier longitud, pero siempre tendrá al menos una letra.

Los operadores del lenguaje son los siguientes:
- el operador de asignación ("="), válido sólo entre expresiones del mismo tipo.
- el operador de concatenación de cadenas ("`.`"), válido sólo para cadenas.
- el operador de comparación entre números ("="), válido sólo para números.
- el operador de comparación entre cadenas ("=="), válido sólo para cadenas.

Se pide construir un **Analizador Léxico** para este lenguaje (*tokens*, gramática, autómata finito determinista y acciones semánticas).

2. Sea la siguiente gramática en EBNF inspirada en la sintaxis del lenguaje COBOL:

S → procedure division . [DECLARACIONES] { CUERPO }$_1$
DECLARACIONES → declaratives . { STRUCT use PÁRRAFO }$_1$ end declaratives
CUERPO → id section . { PÁRRAFO }$_1$
STRUCT → id section number | λ
PÁRRAFO → number . SENTENCE
SENTENCE → goto id | evaluate id

Se pide:
- **a.** Realizar el **diagrama de transición** para las reglas de S, DECLARACIONES y SENTENCE.
- **b.** Definir el programa del **Analizador Sintáctico Descendente Predictivo Recursivo** para esta gramática, que es LL(1). Se deberá usar el siguiente procedimiento auxiliar:

```
ComprobarToken (t: token)
Begin
   If (t = SiguienteToken)
      Then Leer (SiguienteToken)
      Else Error (t)
End
```

Nota 1: En el programa se puede abreviar el nombre de la función `ComprobarToken` por `CT` y el nombre de la variable `SiguienteToken` por `ST`.

Nota 2: En la notación EBNF, el significado de los meta-símbolos es el siguiente:
- [ ]: Opcional
- { }$_1$: Repetitiva de 1 a *n* veces

COMPILADORES

Examen final. Segundo parcial. 11 de febrero de 2010

Observaciones: 1. Se estima que se publicarán las notas del examen final el 22 de febrero, siendo la revisión el 2 de marzo. Las fechas exactas se avisarán en `http://www-lt.ls.fi.upm.es/compiladores`.
2. La duración de este examen será de 2¼ horas.
3. Cada ejercicio deberá entregarse en hojas separadas.

1. De una gramática de un lenguaje se han entresacado las siguientes reglas:

```
P → D ; S
D → T : V | D ; D
T → integer | logical | time
V → id , V | id
S → id := E
E → C | id | E O E | E + E
C → cte | cte : cte : cte
O → < | > | = | <>
```

Se pide construir una **Definición Dirigida por la Sintaxis** para realizar el **Análisis Semántico** y la **Generación de Código Intermedio** (explicando brevemente los atributos y funciones utilizadas), teniendo en cuenta que:

- El lenguaje dispone de constantes enteras (`cte`) y constante de tipo hora (`cte:cte:cte`)
- El lenguaje tiene conversiones de tipo automáticas entre enteros, lógicos y horas
- Si los sumandos son de distinto tipo, el resultado de la suma es entero; si los sumandos son del mismo tipo, el resultado conserva el tipo de los sumandos
- Las constantes de tipo hora deben ser correctas y se almacenan internamente como un valor entero que representa el número de segundos transcurridos desde la media noche
- Los enteros se representan con 4 bytes y los lógicos con 1 byte
- En las expresiones lógicas, el valor 0 se considera como falso y cualquier otro valor como verdadero.

(6 puntos)

2. Sea el siguiente programa correcto:

```
Program QR
  Procedure Main
  Var  a:= 3, b:= 5: integer;
       Procedure Q;
       Var b: integer;
       BEGIN Q
            a:= a - 1;
            b:= a;
            IF b > 0 THEN R (a);
       END Q;
       Procedure R (b: integer);
       BEGIN R
            b:= b - a;
            Q;
       END R;
  BEGIN Main
       Q;
       PRINT (a);
       PRINT (b);
  END Main;
BEGIN QR
  Main;
END QR.
```

Teniendo en cuenta que se trata de un compilador de dos pasadas, que los enteros y las direcciones ocupan 2 bytes, que las expresiones lógicas se representan mediante la técnica de flujo de control, que se utiliza una estrategia de asignación de memoria mediante pila y que el lenguaje sigue las reglas de ámbito léxico, se pide:

a. Diseñar el **Registro de Activación** general para este lenguaje.

b. Realizar la **traza de ejecución** del programa representando el contenido completo de la memoria, indicando el resultado de las sentencias "`PRINT`". Se realizará la traza dos veces, considerando, primero, el paso de parámetros por referencia y, después, el paso de parámetros por valor.

(4 puntos)

COMPILADORES

Segundo parcial, 1 de julio de 2010

Observaciones:
1. Fecha **estimada** de publicación de las calificaciones: 8 de julio.
2. Fecha **estimada** de la revisión: 12 de julio.
3. En `http://www-lt.ls.fi.upm.es/compiladores` 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.
5. Cada ejercicio deberá entregarse en hojas separadas.
6. Los dos ejercicios tienen la misma puntuación.

1. De un lenguaje de programación se ha extraído el siguiente fragmento de gramática:

S → For id:= E To E Do S End For | id := E
E → id | núm | E + E

Se pide diseñar el **Analizador Semántico** y el **Generador de Código Intermedio** mediante un **Esquema de Traducción**, teniendo en cuenta que:

- El lenguaje tiene variables de tipo cadena, entero y lógico, y exige declaración previa de las mismas
- El lenguaje no realiza conversión de tipos
- La suma opera con enteros o con lógicos (en este último caso, se realiza una operación de OR)
- El funcionamiento del bucle for es como sigue:
  1. Se evalúan las dos expresiones
  2. Se inicializa la variable entera índice (id) con el valor de la primera expresión
  3. Si la variable índice es mayor al valor obtenido al evaluar la segunda expresión en el paso 1, se abandona la ejecución del bucle
  4. Se ejecutan las sentencias
  5. Se incrementa en una unidad el valor de la variable índice y se vuelve al paso 3.

2. Sea el siguiente fragmento de un programa:

```
Global a: int;
Program Fechas ()
  Type Record Fecha= {a, m, d: int};
  Var b: Fecha;
  Proced A (Var x:int, y:real, Var z:Fecha)
    Begin A
      If (x - y < 1) Then a:= a + 1;
      Case x:
        1: z.a:= z.a + a;
        2: z.m:= z.m + a;
        3: z.d:= z.d + a;
      End Case;
    End A;
  Proced BB (Var x:int, y:Fecha)
    Var i: int;
    Proced CCC (z:int)
      Var f: real;
      Begin CCC
        f:= z / 2.0;
        Call A (z, f, y);
      End CCC;
    Begin BB
      For i:= 1 To 2 Do
        Call CCC (i);
      End For;
      a:= Truncate(y.m / 4.0) + y.d;
    End BB;
Begin Fechas
  a:= 2;
  b.a:= 2000;
  b.m:= 6;
  b.d:= 15;
  Call BB (a, b);
  Print (a, b);
End Fechas;
```

El compilador y el lenguaje tienen las siguientes características:

- El bucle **For** funciona tal como se describe en el problema 1
- El compilador no realiza ningún tipo de optimización
- El lenguaje distingue las mayúsculas de las minúsculas
- El lenguaje realiza conversión automática de tipos
- El lenguaje tiene estructura de bloques
- La operación **Truncate** devuelve la parte entera, eliminando la parte decimal de su argumento
- Los enteros y las direcciones ocupan 2 bytes; los reales ocupan 6 bytes
- Los parámetros se pasan por valor, salvo que se indique la palabra **Var**, en cuyo caso se pasan por referencia

Se pide:

a. Diseñar el **Registro de Activación** general para este lenguaje.

b. Realizar una **traza de ejecución** del programa representando el contenido completo de la pila, e indicando el resultado de la sentencia "**Print**".

COMPILADORES
Primer parcial, 1 de julio de 2010

Observaciones: 1. Fecha **estimada** de publicación de las calificaciones: 8 de julio.
2. Fecha **estimada** de la revisión: 12 de julio.
3. En http://www-lt.ls.fi.upm.es/compiladores 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.
5. Cada ejercicio deberá entregarse en hojas separadas.

1. El directorio de personal de una organización está formado por el nombre y apellidos de los trabajadores, el número del despacho (un entero de hasta 3 dígitos), el nombre del departamento al que pertenecen (formado por una única palabra) y el número de extensión telefónica (formada por 5 dígitos, de tal forma que los dos primeros indican el código del departamento).

La organización dispone de 27 departamentos, cuyos nombres ordenados alfabéticamente se encuentran en una tabla, donde se indica también el código que sirve como prefijo de la extensión telefónica, comenzando por 01 ("Auditorías") hasta el 27 ("Ventas").

Se pide construir el **Analizador Léxico** para un lenguaje con estas características (gramática regular, *tokens*, AFD con el menor número de estados posible, acciones semánticas y errores), teniendo en cuenta que cada elemento del lenguaje puede ir separado por uno o varios espacios o saltos de línea.

(4 puntos)

2. Sea la siguiente gramática, donde A es el axioma:

1. $A \rightarrow x\ B\ y$
2. $A \rightarrow z\ C$
3. $A \rightarrow \lambda$
4. $B \rightarrow y\ B$
5. $B \rightarrow C$
6. $C \rightarrow A\ y$

Se pide:

a. Construir las **tablas completas** de un analizador por **Precedencia de Operador**, de un analizador **SLR** y de un analizador **LL**, independientemente de si la gramática es o no válida para construir esos analizadores.

b. Para cada uno de estos analizadores, razonar si la gramática es válida para utilizar dicho analizador. En caso negativo, explicar brevemente **todos** los **problemas** que presenta en cada caso.

(6 puntos)

COMPILADORES

Examen final, 2 de septiembre de 2010

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 16 de septiembre. Fecha **estimada** de la revisión: 20 de septiembre. En http://www-lt.ls.fi.upm.es/compiladores se avisará la fecha exacta de publicación de las calificaciones y la fecha y hora definitiva de la revisión. 2. La duración de este examen será de 2¾ horas. 3. Cada ejercicio deberá entregarse en hojas separadas.

1. En un sobre postal, una dirección puede contener los nombres y apellidos del destinatario o el nombre de una empresa, el tipo de calle ("c/", "Av.", "Pº", "Pza." o "Cº"), el nombre de la calle, el número de la calle (un número de 1 ó 2 cifras), la provincia de residencia (supóngase que las 52 provincias españolas están formadas por una única palabra) y el código postal (formado por 5 cifras, de tal forma que las dos primeras hacen referencia a la provincia y las tres últimas corresponden a zonas postales).

En España existen 52 provincias, cuyos nombres y códigos se encuentran en una tabla, comenzando por 01 ("Álava") hasta el 52 ("Melilla").

Se pide construir el **Analizador Léxico** para un lenguaje con estas características (gramática regular, *tokens*, AFD con el menor número de estados posible, acciones semánticas y errores), teniendo en cuenta que se desea transmitir la información de la provincia cuando se reconozca un código postal, que cada elemento del lenguaje va separado por uno o varios espacios o saltos de línea y que el lenguaje distingue mayúsculas y minúsculas.

(3 puntos)

2. Sea la siguiente gramática:

$S \rightarrow a\ B\ B\ a$
$B \rightarrow a\ C \mid \lambda$
$C \rightarrow D\ a\ E \mid E\ a\ D$
$D \rightarrow x\ y \mid y$
$E \rightarrow y\ x \mid x$

Se pide:

a. Completar en la plantilla adjunta el **Autómata Reconocedor de Prefijos Viables** (método SLR).

b. Analizar todos los estados que pudieran dar **conflictos** en el autómata anterior y justificar si la gramática es **SLR**.

c. Aplicar la **condición LL** a toda la gramática y justificar si es una gramática **LL**.

d. Justificar si es una gramática de **Precedencia de Operador**.

(3 puntos)

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

$D \rightarrow L : T\ I\ ;$
$L \rightarrow id \mid id\ ,\ L$
$I \rightarrow := V \mid \lambda$
$T \rightarrow integer \mid real \mid boolean$
$V \rightarrow cte\_entera \mid cte\_real \mid cte\_lógica$
$E \rightarrow id \mid E < E \mid E * E \mid V$
$S \rightarrow id := E$

Teniendo en cuenta que:

- Cuando en una proposición de declaración existe una inicialización, se inicializan todas las variables declaradas en dicha proposición
- El producto entre lógicos representa una operación AND
- Solamente se permite comparar valores numéricos con la operación '<'
- Los enteros ocupan 2 bytes, los reales 4 y los lógicos 1
- No hay conversiones automáticas de tipos
- Se quiere representar los valores lógicos por control de flujo.

Se pide diseñar un **Esquema de Traducción** con el **Analizador Semántico** y el **Generador de Código Intermedio** (código de 3 direcciones) para este lenguaje, explicando brevemente los atributos y funciones usadas.

(4 puntos)

ANÁLISIS LÉXICO

Primer examen. 25 de octubre de 2010

Observaciones: 1. Las calificaciones se publicarán hacia el 23 de noviembre.
2. La revisión será hacia el 26 de noviembre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen será de 40 minutos.

1. Un determinado lenguaje permite representar información mediante listas. Una lista está delimitada por paréntesis. Los elementos de una lista pueden ser cualquiera de los siguientes:

- Nombres: pueden contener letras, dígitos y guiones ('-'), empezando siempre con una letra o dos puntos (':'), y no pudiendo terminar por guión ni tener guiones seguidos.
- Números: están formados por una secuencia de dígitos, con una precisión máxima de 2 bytes.
- Expresiones: una expresión se representa siempre entre paréntesis y está formada por operandos (pueden ser nombres, números u otras expresiones) y operadores aritméticos. Los operadores pueden ser:
  - binarios: +, *, /
  - unarios: ++, **

Seguidamente, se muestran algunos ejemplos de listas válidas:

(hola esto es una lista con nombres) (en esta lista hay nombres-y-valores-enteros el :menor es 0 el :mayor es 65535) (:lista-mixta1 (9 + x1) (x1 +(i * val99++/:x) / (8)) :9 final) (a3 t5 pdl a1b2c3d4 (a+1+b+2+c+3+d+4) a-1-b-2-c-3-d-4) (8 88 8888)

Se pide diseñar un **Analizador Léxico** para este lenguaje (incluyendo los *tokens* completos, la gramática regular, el autómata finito determinista y las acciones semánticas).

TABLA DE SÍMBOLOS Y ANÁLISIS SINTÁCTICO

Segundo examen. 13 de diciembre de 2010

Observaciones: 1. Las calificaciones se publicarán hacia el 11 de enero.
2. La revisión será hacia el 14 de enero.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen será de 40 minutos.

Sea la siguiente gramática:

A → A L | λ
L → (B)
B → num B | id B | num op B | λ

Se pide:

a. Completar en la plantilla adjunta el **Autómata Reconocedor de Prefijos Viables** (correspondiente al método de análisis sintáctico SLR). La plantilla **no** contiene errores, aunque está incompleta.

b. Justificar la posible existencia de **conflictos** en los estados I_0, I_1, I_5 e I_6 del Autómata anterior.

c. Justificar si la gramática es **LL(1)**. En caso de que no lo sea, aplicar las transformaciones necesarias y demostrar que la nueva gramática cumple la **condición LL**.

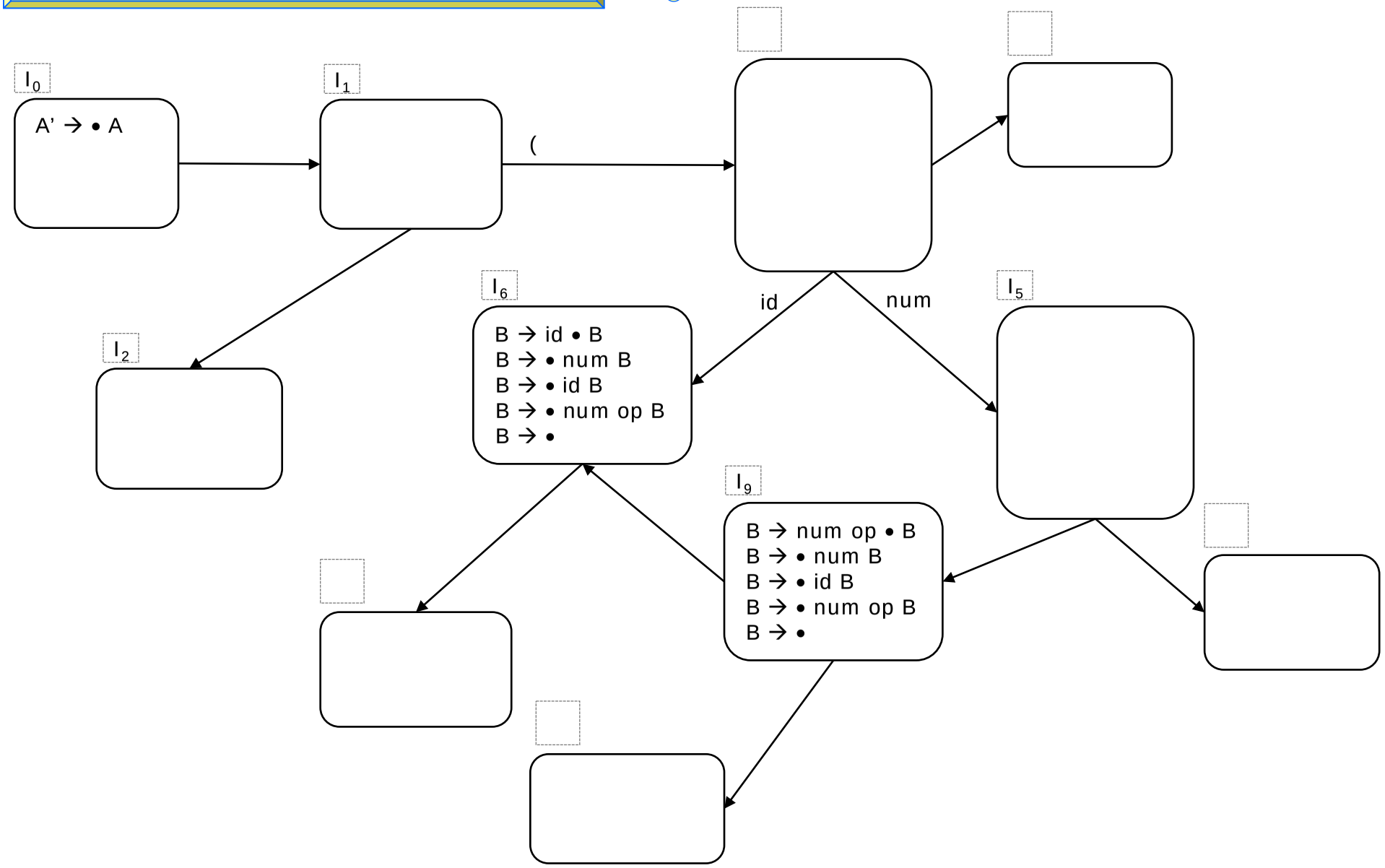
Análisis LR

13-diciembre-2010

Hoja de Respuesta

Apellidos: .....

Nombre: .....



TRADUCCIÓN DIRIGIDA POR LA SINTAXIS Y ANÁLISIS SEMÁNTICO

Tercer examen. 17 de enero de 2011

Observaciones:	1. Las calificaciones se publicarán el 24 de enero. 2. La revisión será el 26 de enero. 3. En la web se dará información sobre la publicación de notas y la revisión 4. La duración de este examen será de 40 minutos.

Sea el siguiente fragmento de la gramática de un lenguaje:

P' → P
P → D P | λ
D → function T id (A) begin B end | id : T
T → int | bool
A → id : T | A , A
B → id : T | id := E | B ; B
E → id | E and E | id (R)
R → E | R , R

Ténganse en cuenta las siguientes características:

- Los identificadores se han de declarar antes de ser usados.
- El lenguaje no realiza una conversión automática de tipos.
- El operador 'and' realiza una conjunción de lógicos.
- Un entero ocupa 2 bytes y un lógico 1 byte.
- El lenguaje dispone de funciones, que no se pueden anidar, y de llamadas a funciones mediante 'id (R)'.
- Los parámetros de la función se pasan por valor.
- El Analizador Léxico introduce los lexemas en la Tabla de Símbolos.

Se pide diseñar el **Analizador Semántico** que realice la comprobación de tipos mediante un Esquema de Traducción (EdT), teniendo en cuenta que:

- El EdT debe detallar todos los accesos a la Tabla de Símbolos.
- El EdT debe completar la Tabla de Símbolos con toda la información posible.
- Deben explicarse brevemente los atributos y funciones utilizadas en el EdT.

COMPILADORES

Examen primer parcial. 2 de febrero de 2011

Observaciones: 1. Las calificaciones se publicarán hacia el 18 de febrero.
2. La revisión será hacia el 22 de noviembre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen será de 2 horas.
5. Todos los ejercicios tienen la misma puntuación.

1. Las referencias bibliográficas de una determinada publicación pueden constar de los siguientes elementos:

- Apellido del autor: estará formado por una única palabra formada por letras minúsculas, excepto la primera que deberá ser mayúscula. Una referencia puede tener varios autores.
- Iniciales del autor: estará formada por una letra mayúscula seguida de un punto. Un autor puede tener más de una inicial.
- Título: estará formado por una serie de caracteres encerrados entre comillas dobles.
- Código ISBN: es un código numérico formado por 10 ó por 13 dígitos; entre dos dígitos puede haber guiones, aunque nunca dos guiones seguidos.
- Fecha: puede estar formada por el nombre completo de un mes en minúsculas y por el año escrito con 4 dígitos.

Además de los signos de puntuación que pueden verse en los ejemplos, los distintos elementos de una referencia pueden ir separados por uno o más blancos, y cada referencia debe ir escrita en un único párrafo y no puede haber dos referencias en el mismo párrafo.

Seguidamente, se muestran algunos ejemplos de referencias bibliográficas correctas:

> Aho, A.V., Lam, M. S., Sethi, R., Ullman, J. D., "Compilers. Principles, Techniques & Tools", 978-0-321-49169-5, abril, 2007
>
> Aho, A. V., Sethi, R., Ullman, J. D., "Compiladores. Principios, técnicas y herramientas", 0-201-62903-8, 1990
>
> Aho,A.V.,Ullman,J.D.,"The Theory of Parsing, Translation and Compiling. Vol. II: Compiling", 1973
>
> Aho, A. V., Ullman, J. D., "The Theory of Parsing, Translation and Compiling. Vol. I: Parsing", 1972
>
> Chomsky,N., "Three models for the description of language", febrero,1956
>
> Ellis,M.A.,Stroustrup,B.,"The Annotated C++ Reference Manual",0-201-51459-1,mayo, 1991
>
> Kernighan, B. W., Ritchie, D. M., "The C Programming Language", julio, 0131103628, 1988

Se pide diseñar un **Analizador Léxico** para este lenguaje (incluyendo los *tokens* completos, la gramática regular, el autómata finito determinista y las acciones semánticas).

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

Bloque → (block Sent Rest-Bloq)
Rest-Bloq → Sent Rest-Bloq | return | λ
Sent → Sent-Asig | Expresión | Sent-Control

Se sabe que:

- Sent-Asig, Expresión y Sent-Control son símbolos No-Terminales
- block y return son símbolos Terminales
- First (Sent-Asig) = First (Expresión) = {id}
- Después del "id" en Sent-Asig siempre viene un "op-asig" y un "id"
- Expresión comienza por un "id" seguido de un "op-arit" y después un "id"
- First (Sent-Control) = {If, While, For}

Se pide:

a. Sin transformar la gramática, construir la **tabla** del **Analizador Sintáctico LL(1)** y justificar si la gramática es LL.

b. Con la información que se proporciona, escribir las reglas correspondientes a Sent-Asig y a Expresión, de forma que la gramática resultante sea LL(1). A continuación, escribir el **Analizador Sintáctico Descendente Recursivo** para los símbolos Bloque, Rest-Bloq y Sent. Se deberá usar el siguiente procedimiento auxiliar:

```
Comprobar_Token (t: token)
Begin
  If (t = Siguiente_token)
      Then Leer (Siguiente_token)
      Else Error (t)
End
```

3. Sea el siguiente fragmento de una gramática de un lenguaje de programación:

P → D S
D → T : L | D ; D
T → integer | boolean | real | array
L → id | id [cte_entera] of T | L ; L
S → for id := E to E do S | repeat S until E | id := E | S ; S
E → E + E | id | cte_entera | true | id [E]

Se tienen las siguientes características:

- El lenguaje realiza conversión implícita de enteros a reales. Para el resto, no hay conversión implícita de tipos.
- El operador aritmético '+' permite realizar una suma de dos enteros o una operación *or* de dos lógicos.
- Las expresiones y el identificador de la instrucción for tienen que ser enteros.

Se pide diseñar un **Esquema de Traducción** para realizar el **Análisis Semántico**, explicando brevemente las funciones y atributos utilizados.

ANÁLISIS LÉXICO

Primer examen. 2 de marzo de 2011

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

1. En la representación informática del lenguaje Braille, los distintos elementos, que pueden ir separados por blancos, se representan de la siguiente forma:
   - Palabras: se forman con las letras del alfabeto escritas en minúsculas. Si se quiere indicar que una letra es mayúscula, hay que utilizar el prefijo '{'. Si se quiere indicar que toda la palabra va en mayúsculas, se duplica este prefijo al principio de la palabra.
   - Números enteros: se utilizan las letras de la 'a' a la 'j' precediendo todo el número del prefijo '#'. El valor numérico de las letras va en orden, de tal forma que 'a'=1, 'b'=2… 'h'=8, 'i'=9 y 'j'=0.
   - Signos de puntuación: se dispone de la coma ',' y del punto '.'.

   Seguidamente, se muestran algunos ejemplos válidos en este lenguaje indicando su representación en Braille y su equivalente en tinta:

Braille	**Tinta**
{procesadores de {lenguajes.	Procesadores de Lenguajes.
{martes, #bb de marzo de #bjaa.	Martes, 22 de marzo de 2011.
{{xhtml #a	XHTML 1
{java{script	JavaScript

   Se desea obtener la equivalencia del Braille en tinta, para lo que se pide diseñar un **Analizador Léxico** para analizar documentos escritos en este lenguaje (*tokens* completos, gramática, autómata y acciones semánticas).

TABLA DE SÍMBOLOS Y ANÁLISIS SINTÁCTICO

Segundo examen. 4 de mayo de 2011

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

a Sea el siguiente programa: *(3,5 puntos)*

```
Program F;
  Global   Mat: Array [1..8] of boolean;
           Var0: real;
  Procedure Cuerpo;
       Var var1= 10, var2= 20: integer;
       Procedure A;
            Var var2: integer;
            Procedure B (var2: integer);      /* Parámetro por valor
            BEGIN
                 var2:= var2 + var1;
                 A;
            END;
            Function C (REF var2: integer): integer; /* Parámetro por referencia
            BEGIN
                 var1:= var1 + 3;
                 var2:= var1 + 1;
                      /* Punto ☼
                 Return var2;
            END;
       BEGIN
            If var2 < 7 Then B(var1); Else var1:= var1 + C(Mat[5]);
       END;
  BEGIN
       var1:= 1;
       var2:= 2;
       A;
       var0:= var1;
  END;
BEGIN
  Cuerpo;
END.
```

Considerando que un Compilador ha procesado hasta el punto señalado con ☼, se pide:

1) Mostrar el **contenido de la Tabla de Símbolos** en ese instante.
2) Describir brevemente (5 líneas) la **estructura de la Tabla de Símbolos**.

b Sea la siguiente gramática: *(6,5 puntos)*

S → A B
A → 1 A | 2 B | λ
B → 3 A | 4 B | λ

1) Completar en la plantilla adjunta (que puede contener errores y omisiones) el **Autómata Reconocedor de Prefijos Viables** (Análisis Sintáctico LR).
2) Demostrar si la **gramática es LR** realizando el análisis de los posibles conflictos en el autómata.

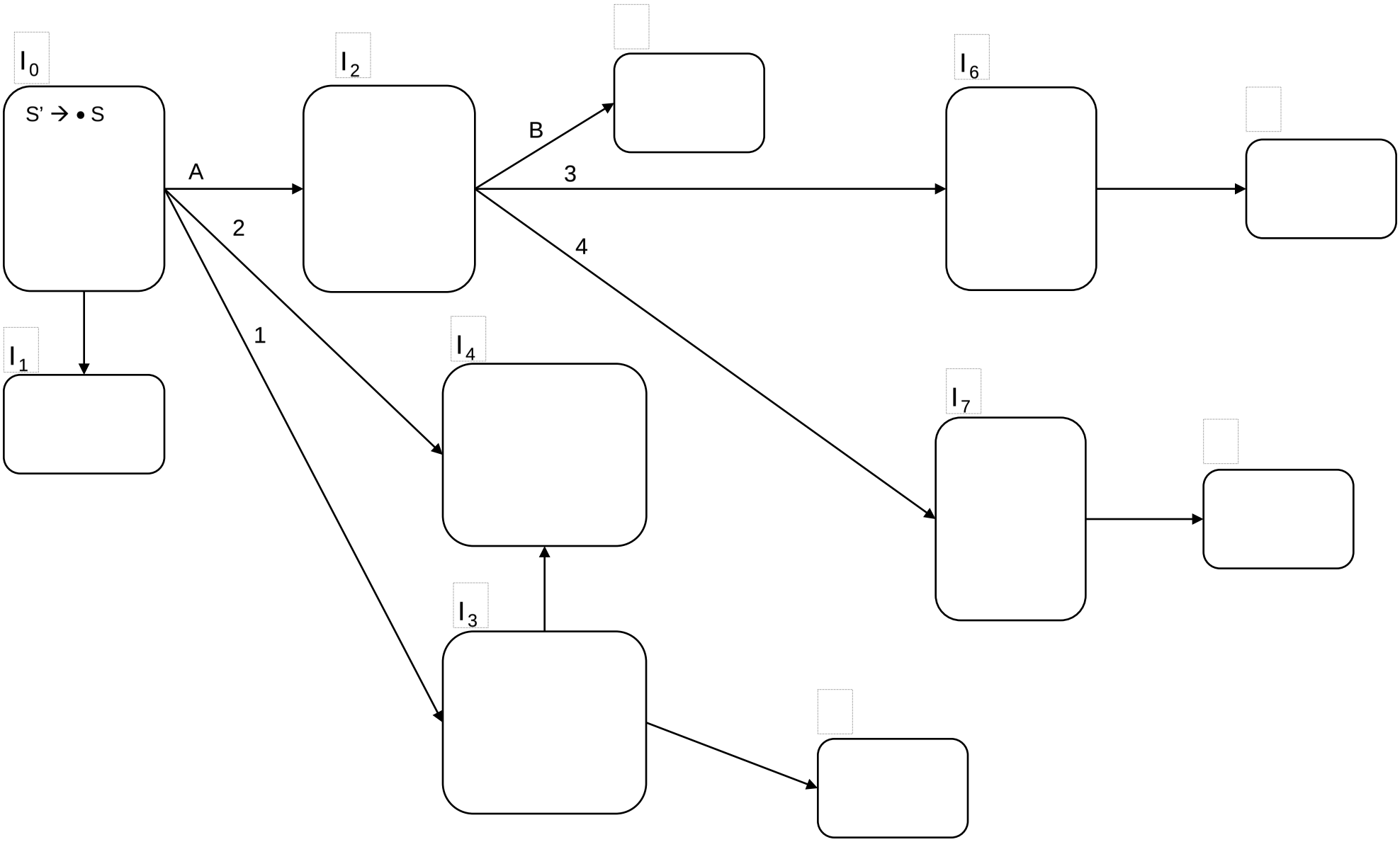
Procesadores de Lenguajes

4-mayo-2011

Hoja de Respuesta

Apellidos: .....

Nombre: .....



COMPILADORES

Primer parcial, 20 de junio de 2011

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 4 de julio. 2. Fecha **estimada** de la revisión: 6 de julio. 3. En `http://www-lt.ls.fi.upm.es/compiladores` 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. 5. Cada ejercicio deberá entregarse en hojas separadas. 6. Los dos ejercicios tienen la misma puntuación.

1. De un determinado lenguaje se ha extraído un fragmento con las siguientes características:

S → switch (E) do { L } | break ; | id := E ; | λ
L → case N : S L | default : S | λ
E → id | N | – E | (E)
N → cte | – cte
id → id l | id d | l
cte → cte d | d

a. Partiendo de la descripción del lenguaje y teniendo en cuenta todos los elementos que se muestran (identificadores, números enteros representados con 2 bytes, signos de puntuación, palabras reservadas…), se pide diseñar un **Analizador Léxico** para la parte léxica de este lenguaje (gramática, *tokens*, autómata finito determinista y acciones semánticas), que introduzca toda la información posible en la Tabla de Símbolos.

b. Partiendo de la descripción del lenguaje y teniendo en cuenta la parte sintáctica que se muestra, se pide extraer las reglas de la Gramática de Contexto Libre que refleja la sintaxis del lenguaje y, seguidamente, construir la tabla de **Análisis Sintáctico Descendente**. A la vista de la tabla obtenida, ¿qué se puede afirmar acerca de la gramática?

2. Sea el siguiente fragmento de gramática:

P → D S
D → type T var V
T → id = record C end ; T | λ /* *donde* ***id*** *será un nombre de tipo correspondiente a un registro*
C → id : X ; C | λ
V → id : X ; V | λ
X → int | bool | id /* *donde* ***id*** *se ha declarado como registro*
S → S ; S | id := E
E → E + E | E or E | id | id . id /* *donde* $id_1.id_2$ *refleja el acceso al campo* id_2 *del registro* id_1

Se pide diseñar el **Analizador Semántico** mediante un Esquema de Traducción (explicando el significado de todos los atributos y funciones utilizadas), teniendo en cuenta que:

- No hay en ningún caso conversión automática de tipos
- El identificador de la parte izquierda de la asignación puede ser de cualquiera de los tipos posibles del lenguaje
- El operador suma actúa entre enteros, que se representan con 2 bytes.
- El operador or actúa entre lógicos, que se representan con 1 byte.
- El analizador léxico introduce en la Tabla de Símbolos toda la información posible.

PROCESADORES DE LENGUAJES

Examen Final, 8 de julio de 2011

Observaciones: 1. La fecha estimada de publicación de las calificaciones es el 12 de julio.
2. La fecha estimada de la revisión es el 14 de julio.
3. La duración de este examen será de 2 horas.
4. Cada ejercicio deberá entregarse en hojas separadas.
5. Las tres preguntas tienen la misma puntuación.

1. Un lenguaje dispone, entre otros, de los siguientes elementos (se muestran ordenados por el número de prioridad):

1. Identificadores: pueden comenzar por una letra o un dígito y pueden contener cualquier cantidad de letras o dígitos, siempre que al menos tengan una letra.
2. Enteros: formados por una secuencia no vacía de dígitos.
3. Reales en notación normal: formados por una secuencia no vacía de dígitos, un punto y otra secuencia no vacía de dígitos.
4. Reales en notación científica: formados por una secuencia no vacía de dígitos, seguida, opcionalmente, de un punto y una secuencia no vacía de dígitos. A continuación, tras la letra E, irá la parte exponencial formada por un signo opcional y un entero.

En caso de duda con alguna palabra del lenguaje, el analizador tomará la que coincida con la descripción con número de prioridad mayor. La tabla resume algunos ejemplos válidos y erróneos de estos elementos:

identificador	entero	real, notación normal	real, notación científica	errores	
A	0	0.0	2E3	3.2E3F	-54
8E	88	44.3	4.102030E1	3E-5A	1.2E3.4
DC10	432	3.8888888888	8.77E+3	J8.3	6.
2E3F	9876	54321.0	6.023E-23	.5	5.E2

Se pide construir la gramática regular, *tokens*, autómata y acciones semánticas (indicando los accesos a la Tabla de Símbolos) de un **Analizador Léxico** para este lenguaje.

2. Sea la siguiente gramática:

P → λ | P S
S → < E >
E → λ | cte E | var E | cte + E

Se pide:

a. Completar en la plantilla adjunta el **Autómata Reconocedor de Prefijos Viables** (correspondiente al método de análisis sintáctico SLR). La plantilla NO contiene errores, aunque está incompleta.
b. Justificar la posible existencia de **conflictos** en los estados I_0, I_1, I_5 e I_6 del Autómata anterior.
c. Justificar si la gramática es **LL(1)**. En caso de que no lo sea, aplicar las transformaciones necesarias y demostrar que la nueva gramática cumple la **condición LL**.

3. Sea la siguiente gramática:

P → D S
L → id , L | id
I → id := E
A → id ++ | I
R → núm : S ; R | default S ; | λ
E → id | núm | E & E | E oprel E | E oplog E
S → S ; S | for (I ; E ; A) S | I | switch id of R end | λ
D → D L : T ; | λ
T → integer | boolean
oplog → and | or
oprel → > | < | = | !=

Se pide diseñar el **Analizador Semántico** mediante una Definición Dirigida por la Sintaxis (que introduzca toda la información posible en la Tabla de Símbolos y explicando brevemente las funciones y atributos utilizados) sabiendo que:

- Todas las variables han de estar declaradas previamente a su uso
- El Analizador Léxico mete los lexemas de los identificadores en la Tabla de Símbolos
- El lenguaje no tiene conversión de tipos
- El operador & actúa entre operandos del mismo tipo y devuelve un resultado de ese tipo
- El operador ++ actúa sobre enteros
- En el for, los identificadores de la inicialización (I) y la asignación (A) han de ser enteros, y la condición (E) tiene que ser lógica
- El switch funciona de la siguiente manera: se ejecuta solo la rama cuyo núm es igual que el valor del id; si ningún num coincide, se ejecuta la rama default si existe

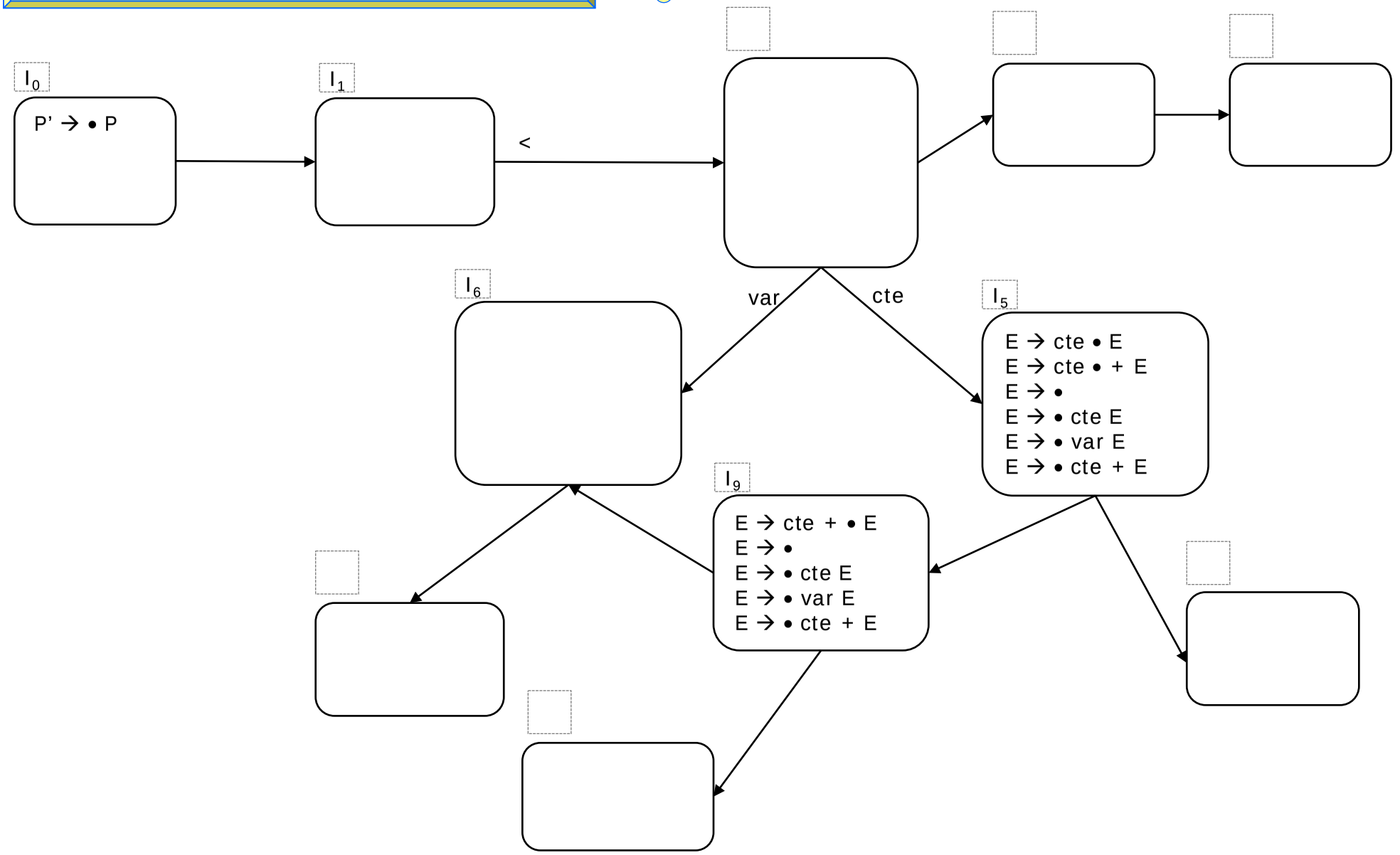
Procesadores de Lenguajes

8-julio-2011

Pregunta 2. Hoja de Respuesta

Apellidos: .....

Nombre: .....



COMPILADORES

Examen Final, 15 de septiembre de 2011

Observaciones: 1. La fecha estimada de publicación de las calificaciones es el 26 de septiembre.
2. La fecha estimada de la revisión es el 28 de septiembre.
3. La duración de este examen será de 2½ horas.
4. Cada ejercicio deberá entregarse en hojas separadas.
5. Las tres preguntas tienen la misma puntuación.

1. Un lenguaje dispone, entre otros, de los siguientes elementos (se muestran ordenados por el número de prioridad):

1. Identificadores: pueden comenzar por una letra o un dígito y pueden contener cualquier cantidad de letras o dígitos, siempre que al menos tengan una letra.
2. Enteros: formados por una secuencia no vacía de dígitos.
3. Reales en notación normal: formados por una secuencia no vacía de dígitos, un punto y otra secuencia no vacía de dígitos.
4. Reales en notación científica: formados por una secuencia no vacía de dígitos, seguida, opcionalmente, de un punto y una secuencia no vacía de dígitos. A continuación, tras la letra `E`, irá la parte exponencial formada por un signo opcional y un entero.

En caso de duda con alguna palabra del lenguaje, el analizador tomará la que coincida con la descripción con número de prioridad mayor. La tabla resume algunos ejemplos válidos y erróneos de estos elementos:

identificador	entero	real, notación normal	real, notación científica	errores	
`A`	`0`	`0.0`	`2E3`	`3.2E3F`	`-54`
`8E`	`88`	`44.3`	`4.102030E1`	`3E-5A`	`1.2E3.4`
`DC10`	`432`	`3.8888888888`	`8.77E+3`	`J8.3`	`6.`
`2E3F`	`9876`	`54321.0`	`6.023E-23`	`.5`	`5.E2`

Se pide construir la gramática regular, *tokens*, autómata y acciones semánticas (indicando los accesos a la Tabla de Símbolos) de un **Analizador Léxico** para este lenguaje.

2. Sea la siguiente gramática:

```
P → D S
L → id , L | id
I → id := E
A → id ++ | I
R → núm : S ; R | default S ; | λ
E → id | núm | E & E | E oprel E | E oplog E
S → S ; S | for ( I ; E ; A ) S | I | switch id of R end | λ
D → D L : T ; | λ
T → integer | boolean
oplog → and | or
oprel → > | < | = | !=
```

Se pide diseñar el **Analizador Semántico** mediante una Definición Dirigida por la Sintaxis (que introduzca toda la información posible en la Tabla de Símbolos y explicando brevemente las funciones y atributos utilizados) sabiendo que:

- Todas las variables han de estar declaradas previamente a su uso
- El Analizador Léxico mete los lexemas de los identificadores en la Tabla de Símbolos
- El lenguaje no tiene conversión de tipos
- El operador & actúa entre operandos del mismo tipo y devuelve un resultado de ese tipo
- El operador ++ actúa sobre enteros
- En el for, los identificadores de la inicialización (I) y la asignación (A) han de ser enteros, y la condición (E) tiene que ser lógica
- El switch funciona de la siguiente manera: se ejecuta solo la rama cuyo núm es igual que el valor del id; si ningún núm coincide, se ejecuta la rama default si existe.

ANÁLISIS LÉXICO

Primer examen. 7 de noviembre de 2011

Observaciones: 1. Las calificaciones se publicarán hacia el 22 de noviembre.
2. La revisión será hacia el 25 de noviembre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen será de 40 minutos.

1. Se tiene el siguiente lenguaje:

Dispone de variables con las siguientes características:

- Su nombre siempre debe comenzar con una letra.
- Después de la primera letra, pueden tener más letras o dígitos o subrayados (_).
- Aunque pueden tener cualquier longitud, solamente se tienen en cuenta los primeros 63 caracteres de su nombre (el resto será ignorado).
- No hay distinción entre mayúsculas y minúsculas.
- No pueden coincidir con ninguna de las 87 palabras reservadas del lenguaje (éstas estarán formadas sólo por letras).
- Las variables tienen que estar declaradas previamente a su uso.

Las variables pueden ser de tipo entero, carácter, lógico o cadena y se declaran utilizando las palabras reservadas `integer`, `char`, `boolean` o `string`, respectivamente. Para cada uno de estos tipos, el lenguaje dispone de las constantes correspondientes:

- Una constante de tipo entero está formada por una secuencia de dígitos y se almacenará en 4 bytes (por ejemplo: `0`, `7`, `11`, `1234567890`).
- Una constante de tipo carácter es un elemento del código ASCII encerrado entre apóstrofes (por ejemplo: `'a'`, `'B'`, `'2'`, `'+'`), y su valor es un número entre 0 y 127, correspondiente al código ASCII del elemento. Una constante de tipo carácter también se puede representar en el lenguaje fuente con un signo # seguido inmediatamente del valor numérico correspondiente (por ejemplo, el carácter `'A'` puede ser representado también como `#65`).
- Una constante de tipo lógico se representa mediante las palabras reservadas `true` y `false`.
- Una constante de tipo cadena se representa con una secuencia de caracteres, encerrados entre dobles comillas, con una longitud máxima de 255 caracteres (por ejemplo: `"Hola"`, `"Próximo Examen: 12/12/11"`, `"if (a>b) then go!"`).

Es posible acceder a un solo carácter de una cadena poniendo el nombre de la variable seguido entre corchetes de la posición del carácter al que se quiere acceder; dicha posición será una expresión entera (por ejemplo: `nombre[5]`).

Se pide diseñar un **Analizador Léxico** para este lenguaje (*tokens* completos, gramática regular, autómata finito determinista, acciones semánticas y errores).

TABLA DE SÍMBOLOS Y ANÁLISIS SINTÁCTICO

Segundo examen. 12 de diciembre de 2011

Observaciones: 1. Las calificaciones se publicarán hacia el 10 de enero.
2. La revisión será hacia el 12 de enero.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen será de 40 minutos.

Sea la siguiente gramática:

P → λ | P S
S → < E >
E → λ | cte E | var E | cte + E

Se pide:

a. Completar en la plantilla adjunta el **Autómata Reconocedor de Prefijos Viables** (correspondiente al método de análisis sintáctico SLR). La plantilla **no** contiene errores, aunque está incompleta.

b. Justificar la posible existencia de **conflictos** en los estados I_0, I_1, I_5 e I_6 del Autómata anterior.

c. Justificar si la gramática es **LL(1)**. En caso de que no lo sea, aplicar las transformaciones necesarias y demostrar que la nueva gramática cumple las **condiciones LL**.

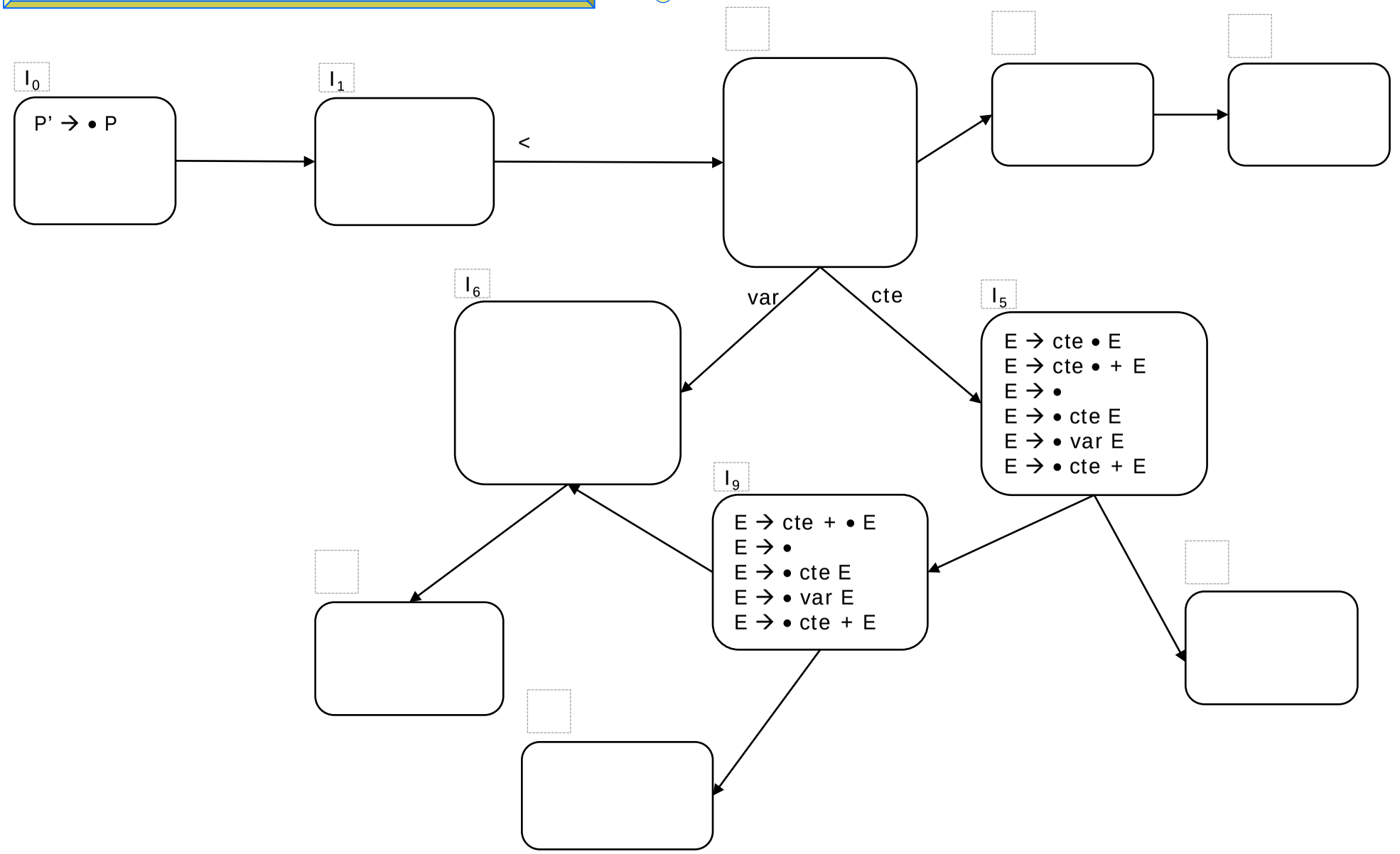
Análisis Sintáctico

12-diciembre-2011

Pregunta 1. Hoja de Respuesta

Apellidos: .....

Nombre: .....



Examen. 24 de enero de 2012

Observaciones: 1. Las calificaciones se publicarán hacia el 27 de febrero.
2. La revisión será hacia el 29 de febrero.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen será de 2 horas.
5. Todos los ejercicios tienen la misma puntuación.

1. Un intérprete de comandos de un Sistema Operativo dispone de instrucciones con los siguientes formatos:

`EDIT` $\{$*`fichero`*$\}_1^n$ `COPY` *`fichero fichero`* `DIRECTORY` [*`fichero`*]
`DELETE` *`fichero`* `RENAME` *`fichero fichero`*

El formato del nombre de un *`fichero`* es: *`nombre`*[`.`*`extensión`*][`;`*`versión`*], donde *`nombre`* y *`extensión`* pueden tener hasta quince letras o dígitos y la *`versión`* es un número entero sin signo inferior a 100. El nombre completo de un *`fichero`* no puede incluir blancos. Dentro del *`nombre`* o de la *`extensión`* de un fichero se pueden incluir caracteres comodines:

- Asterisco (`*`): representa ninguno, uno o varios caracteres. Si aparece, el asterisco será siempre el último carácter de ese *`nombre`* o de esa *`extensión`*.
- Interrogación (`?`): representa un carácter en la posición en que aparezca.

El número de *`versión`* también puede ser sustituido por el comodín asterisco.

Nota: El intérprete cuenta con una función (`FAT`) que cuando recibe el *`nombre`*, la *`extensión`* y la *`versión`*, devuelve un puntero al fichero o ficheros. Dicho puntero es la información que el Analizador Sintáctico necesita acerca de los ficheros.

Se pide diseñar el **Analizador Léxico** correspondiente a este intérprete de comandos, incluyendo la gramática regular en BNF, los *tokens* completos, un autómata finito determinista de tamaño reducido y las acciones semánticas.

2. Sea la siguiente gramática:

S → T V | V Z
T → a T | b T | h
V → λ | c Z h
Z → λ | d Z

Se pide:

a. Demostrar que la gramática cumple la **Condición LL(1)** detallando cada comprobación
b. Construir la **tabla** correspondiente al **Analizador Descendente** LL(1).
c. Diseñar los procedimientos correspondientes al **Analizador Descendente Predictivo Recursivo** para los símbolos `S` y `Z`.

3. De un lenguaje se ha extraído el siguiente fragmento de gramática:

```
F → function id (L): T; begin S end ;
L → id : T ; L | id : T
S → S ; S | return E | id += E
E → id | E != E
T → integer | boolean
```

Se pide diseñar el **Analizador Semántico** mediante un **Esquema de Traducción** teniendo en cuenta que:

- El lenguaje tiene variables enteras y lógicas y no realiza conversiones de tipos
- El cuerpo de la función tiene que tener al menos una sentencia `return`
- La sentencia `id += E` es equivalente a `id := id + E` (el operador `+` representa tanto la suma aritmética como el or logico)
- El `!=` es el operador distinto para comparar dos valores del mismo tipo
- Se debe detallar la gestión de la Tabla de Símbolos pero teniendo en cuenta que el Analizador Léxico introduce los lexemas en la Tabla de Símbolos
- Deben explicarse brevemente los atributos y funciones utilizadas

ANÁLISIS LÉXICO

12 de abril de 2012

Observaciones:	1. Las calificaciones se publicarán aproximadamente el 23 de abril. 2. La revisión será aproximadamente el 25 de abril. 3. En la web se dará información sobre la publicación de notas y la revisión. 4. La duración de este examen será de 40 minutos.

Una Federación deportiva necesita crear una aplicación para calcular los *rankings* de sus atletas en función de las clasificaciones de las distintas carreras en que participan. Una vez realizado el diseño de dicha aplicación, uno de los módulos que se debe crear es un procesador de los ficheros de clasificaciones que sirva de punto de entrada al generador de *rankings*. La tarea pedida es el diseño del Analizador Léxico de este procesador.

El formato del fichero de clasificaciones agrupa a los corredores por su categoría. Así, para cada una de las categorías, aparecerá en una línea el nombre de la categoría, formado por una letra (la inicial de Damas u Hombres), seguida de un guión, la edad (con dos dígitos) y, posiblemente, una letra más, como por ejemplo: D-14, H-21, H-20A.

A continuación, en cada línea, aparecerán los datos (separados por blancos) de cada corredor, que podrá incluir la siguiente información:

- El número de licencia es un número de 8 dígitos que identifica a cada corredor.
- Nombres y apellidos del corredor separados por blancos.
- Tiempo del corredor. Aparecerá en uno de dos posibles formatos: si el tiempo es inferior a una hora, aparecerá como *dd:dd* (36:08); en caso contrario, aparecerá como *d⁺:dd:dd* (201:24:45). La aplicación necesita manejar los tiempos como número de segundos para el cálculo posterior de las puntuaciones de cada corredor.
- Código que define el estado de la participación del corredor, que puede ser: 1, para prueba correcta; 2, para desclasificado; 3, indica fuera de control y 4, no toma la salida.

Se pide realizar el diseño del **Analizador Léxico** (*tokens*, gramática regular, autómata y acciones semánticas) para procesar estos ficheros de clasificaciones en este contexto.

Ejemplo de un fichero de clasificaciones correcto (se han impreso también los saltos de línea):

```
D-14 ↵
28001457 María Luisa Pérez Gómez 26:04     1     ↵
08906756 Inés    González 1:00:13 1↵
28973434           Sonia Béjar 3:13:13 3 ↵
H-14↵
23457876 Mario Soria        31:11:59          1↵
56245432  José González          4↵
```

PROCESADORES DE LENGUAJES

TABLA DE SÍMBOLOS Y ANÁLISIS SINTÁCTICO

Segundo examen. 10 de mayo de 2012

Observaciones: 1. Las calificaciones se publicarán hacia el 23 de mayo.
2. La revisión será hacia el 25 de mayo.
3. En la web se avisarán las fechas exactas.
4. La duración de este examen es de 40 minutos.

Sea el siguiente fragmento de una gramática:

E → A x | B y | C z
A → x | B x y | B x
B → x | E z
C → z | λ

Se pide:

a. Construir el **Autómata Reconocedor de Prefijos Viables** (Análisis Sintáctico LR). Estudiar y justificar la existencia de conflictos.

b. Independientemente de que la gramática sea o no LR, construir las **Tablas LR** (solamente las filas correspondientes a los estados que contengan ítems correspondientes a las reglas de B).

c. Independientemente de que la gramática sea o no LL, construir la **Tabla LL** (Análisis Sintáctico Descendente).

COMPILADORES Y PROCESADORES DE LENGUAJES
15 de junio de 2012

Observaciones: 1. Fecha aproximada de publicación de las calificaciones: 21-junio (Procesadores de Lenguajes) y 27-junio (Compiladores).
2. Fecha aproximada de la revisión: 25-junio (Procesadores de Lenguajes) y 29-junio (Compiladores).
3. Cada ejercicio debe entregarse en hojas separadas.
4. Cada ejercicio tiene la misma puntuación.
5. Los alumnos de "**Compiladores**" deben realizar los ejercicios 1, 2 y 3 y disponen de 2 horas.
6. Los alumnos de "**Procesadores de Lenguajes**" deben realizar únicamente el ejercicio 3 y disponen de 40 minutos.

1. Se pide diseñar el **Analizador Léxico** (gramática, *tokens*, AFD, acciones semánticas y errores) del siguiente fragmento de un lenguaje de programación:

- Un identificador válido es cualquier combinación de letras y dígitos que tenga al menos una letra. También puede contener el carácter subrayado ("_"), pero no puede ni empezar ni terminar con este carácter. Ejemplos: `111a`, `b`, `c25aF`, `11_1z`, `c__25v_f`
- Los comentarios empiezan por dos guiones seguidos ("`--`") y terminan con el fin de línea
- Las variables se declaran poniendo su nombre, el carácter dos puntos ("`:`"), el tipo ("**`integer`**", "**`boolean`**"…) y, finalmente, un punto y coma ("`;`"). Ejemplo: `a: integer;`
  - En una declaración se puede inicializar el identificador. Para ello, después del tipo, se pone el signo de asignación "`:=`" seguido de una expresión. Se termina con el punto y coma final de la declaración. Ejemplo: `a: integer:= 1;`
  - Los identificadores constantes se declaran añadiendo la palabra "**`const`**" justo delante del tipo. Ejemplo: `Ancho: const integer:= 31;`
- Las expresiones del lenguaje vienen definidas mediante las siguientes reglas:
  `E → id | núm_entero | true | false | E * E | E ** E | E and E | E or E`
- El lenguaje tiene palabras reservadas, como las que se han marcado en negrita en este enunciado, las correspondientes a las sentencias de control de flujo del lenguaje ("**`loop`**", "**`while`**", "**`repeat`**"…), etc.

2. a. Sea la siguiente gramática:

```
S → A d
A → A b | A c | B A
B → λ | A
```

Se pide obtener el estado inicial del **Autómata Reconocedor de Prefijos Viables** (método SLR), así como todos los estados a los que se puede transitar desde el estado inicial en un paso.

b. Sea la siguiente gramática:

```
S → case id is C A B | id := constante
A → C A | λ
B → other begin S end | λ
C → constante begin S end
```

Se pide diseñar los procedimientos del método de **Análisis Sintáctico Descendente Recursivo**, usando (si se desea) un procedimiento *Equipara* para el reconocimiento de símbolos terminales.

c. Sea la siguiente gramática:

```
S → d A
A → b A | b c B
B → λ | d B
```

Se pide construir la tabla del **Analizador Descendente LL**, modificando la gramática si fuera necesario para que cumpla la condición LL(1).

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

```
D → D D  |  function T id ( L ) begin S end
T → int  |  string  |  array (cte_entera) of T
L → T id  |  L , L
S → S S  |  id := E  |  T id
E → id  |  id ( A )
A → E  |  A , A
```

Se pide construir un **Analizador Semántico** para este lenguaje mediante un **Esquema de Traducción**, explicando brevemente todas las funciones y atributos utilizados, teniendo en cuenta que:

- El lenguaje exige declaración previa de identificadores y un identificador no puede declararse dos veces en un mismo ámbito.
- El lenguaje dispone de funciones (con al menos un parámetro), tipos simples (entero y cadena) y vectores unidimensionales de tipos simples.
- El lenguaje no realiza conversión automática de tipos.
- El lenguaje permite la declaración de variables locales intercaladas con las sentencias.
- Los índices de los vectores van de 1 hasta el valor indicado por la `cte_entera` en la declaración.
- La expresión `id(A)` puede representar un acceso a un elemento de un vector o la llamada a una función.

PROCESADORES DE LENGUAJES

Examen Final, 16 de julio de 2012

Observaciones:	1. La fecha estimada de publicación de las calificaciones es el 20 de julio. 2. La fecha estimada de la revisión es el 23 de julio. 3. La duración de este examen será de 1¾ horas. 4. Cada ejercicio deberá entregarse en hojas separadas. 5. Las dos preguntas tienen la misma puntuación.

1. La Dirección General de Tráfico recibe, de los responsables de tráfico de las distintas autonomías, un fichero que contiene las incidencias de tráfico en sus carreteras. Por cada incidencia, la información recibida va en una línea y consta (en este orden) de los siguientes elementos:

- Carretera: una o dos letras, seguidas de un guión y el código numérico de la carretera que está compuesto de hasta 4 dígitos, siendo el primero de ellos siempre un dígito entre 1 y 6.
- Puntos kilométricos inicial y final de la incidencia: vienen representados cada uno por un número de hasta 3 dígitos, que puede ir seguido de una coma y otro dígito. Si la incidencia no ocurre en un tramo, sino en un punto de la carretera, se proporciona un solo punto kilométrico.
- Nivel de incidencia: viene representado por una palabra que indica un color de los siguientes: Negro, Rojo, Amarillo, Verde, Gris.
- Municipio en donde se produce la incidencia: está formado por una palabra, y tiene que coincidir con uno de los municipios válidos que están almacenados en una tabla.
- Causa de la incidencia: la descripción es un texto que viene encerrado entre comillas y puede contener cualquier carácter (excepto las comillas). El texto de la causa puede estar vacío.

Algunos ejemplos de incidencias son:

```
B-20    14      12      Amarillo        Barcelona       "RETENCIÓN / CONGESTIÓN por AVERÍA"
N-323   67      77,8    Negro   Carchel "OBSTÁCULO FIJO por DESPRENDIMIENTOS"
AP-68   14 Gris Orozko ""
CA-3113         0 4,9   Rojo    PuertoReal "CARRETERA CERRADA EN AMBOS SENTIDOS por OBRAS"
N-1     308,5   307,3   Verde   Ameyugo         "CARRIL LENTO CERRADO por DESPRENDIMIENTOS"
```

Se pide construir la gramática regular, *tokens*, autómata y acciones semánticas (indicando los accesos a la Tabla de Símbolos) de un **Analizador Léxico** para este lenguaje.

2. Sea la siguiente gramática:

```
S → case E C otherwise S ; S  |  id := E ; S  |  λ
C → E ( S ) C  |  λ
E → id  |  cte_entera + E
```

El lenguaje presenta las siguientes características:

- No hay conversión automática de tipos
- Hay declaración previa de variables que pueden ser enteras, reales o lógicas
- La expresión `E` de la sentencia `case` puede ser entera o lógica. Las expresiones `E` de cada caso `C` deben ser todas del mismo tipo y coincidir con la expresión `E` de la correspondiente sentencia `case`.

Se pide:

a. Obtener la Tabla correspondiente al **Analizador Sintáctico Descendente por Tablas** (LL(1)). Razonar a la vista de la tabla si la gramática es LL(1) o no.

b. Diseñar el **Analizador Semántico** mediante un **Esquema de Traducción** para este fragmento de lenguaje, describiendo brevemente las funciones y atributos utilizados.

COMPILADORES

Examen Final, 10 de septiembre de 2012

Observaciones: 1. La fecha estimada de publicación de las calificaciones es el 19 de septiembre.
2. La fecha estimada de la revisión es el 21 de septiembre.
3. La duración de este examen es de 2½ horas.
4. Cada ejercicio deberá entregarse en hojas separadas.

1. La notación húngara para nombrar identificadores utiliza el primer carácter del nombre para indicar el tipo de la variable. Considérense únicamente los siguientes tipos:

`c`: carácter	`b`: lógico
`n`: entero	`s`: cadena
`f`: real	`l`: entero gran precisión

Se tiene un lenguaje con las siguientes características:

- Dispone de cadenas de caracteres encerradas entre comillas. La cadena puede estar vacía.
- Los elementos léxicos del lenguaje se encuentran separados entre sí por blancos o saltos de línea.
- Los identificadores tienen que ir en notación húngara, y pueden estar formados por letras, dígitos o subrayados, aunque no puede terminar por el carácter de subrayado. Los nombres de identificadores deben tener en total un mínimo de un carácter y un máximo de 64.
- Se diferencian las mayúsculas de las minúsculas.
- Un identificador puede aparecer en cualquier parte del programa y la primera aparición constituye su declaración. El primer carácter tiene que corresponder obligatoriamente con uno de los tipos válidos.

Se pide construir un **Analizador Léxico** (gramática regular, *tokens*, autómata, acciones semánticas y errores) que reconozca este lenguaje y rellene toda la información posible en la Tabla de Símbolos.

(3 puntos)

2. Sea la siguiente gramática, cuyo axioma es A:

A → B C | x D
B → y B | λ
C → z B | x D
D → B D | C A

Se pide:

a. Calcular los conjuntos **First** y **Follow** para todos los símbolos no terminales de la gramática.
b. Comprobar si se cumple la **condición LL(1)** para esta gramática.
c. Obtener la tabla correspondiente al **Analizador Sintáctico Descendente por Tablas** o LL(1) sin cambiar la gramática. Si la gramática no es LL(1), resaltar las celdas que demuestran que no es LL(1).
d. Escribir el procedimiento para el símbolo B correspondiente al método de **Análisis Sintáctico Descendente Recursivo**.
e. Calcular el estado del **Autómata Reconocedor de Prefijos Viables** (método Análisis Sintáctico Ascendente LR) que se genera a partir del ítem C → x • D.

(3 puntos)

3. Sea el siguiente fragmento de una gramática:

S → while E do S | exit E | id := E | S ; S
E → E nand E | id

Se pide diseñar el **Analizador Semántico** y el **Generador de Código Intermedio** (para obtener código de 3 direcciones) mediante una **Definición Dirigida por Sintaxis**, representando los lógicos por control de flujo y teniendo en cuenta que:

- Los identificadores pueden ser enteros o lógicos
- El lenguaje no realiza conversión de tipos
- Una expresión lógica nand es falsa si ambos operandos son ciertos
- Cuando la expresión de la instrucción exit E es cierta, se sale del bucle

(4 puntos)

ANÁLISIS LÉXICO Y TABLA DE SÍMBOLOS

Primer examen. 22 de octubre de 2012

Observaciones: 1. Las calificaciones se publicarán hacia el 14 de noviembre.
2. La revisión será hacia el 16 de noviembre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen es de 40 minutos.

Se tiene un lenguaje para representar los datos de alumnos en un fichero. Para cada alumno, el fichero contiene una línea con la siguiente información, llevando cada campo obligatoriamente al menos un blanco al final:

- Número de DNI (compuesto obligatoriamente por 8 dígitos seguidos de una letra)
- Número de matrícula. Existen tres formatos para representar el número de matrícula:
  - 6 dígitos
  - Una letra seguida de 4 dígitos
  - 2 dígitos, la letra 'M' y 3 dígitos
- Nombre y apellidos del alumno (formados únicamente con letras)
- La palabra 'ECTS:' seguida inmediatamente del número de créditos en los que está matriculado (número entero positivo mayor que cero y menor que 61). Si el alumno no está matriculado, este campo no aparecerá.

Se muestra a continuación como ejemplo un fragmento de cómo podría ser este fichero:

```
 ECTS:33 04422185K J0499 MARÍA JOSÉ    SÁNCHEZ DE MORA Y GÓMEZ DEL POSTIGO
87654321X   010088 ARNOLD   ALOIS SCHWARZENEGGER
  11M001  FÉLIX  PI     ECTS:15       60613579T
00578028S  101234  MARLENE JOSEPHINE GÓMEZ JUAN  ECTS:3
```

Se desea construir un procesador para este lenguaje con el objetivo de pasar la información de dicho fichero a una base de datos en la que se tendrán que almacenar: el nombre y apellidos, DNI y matrícula de cada alumno, así como el número de créditos en los que está matriculado (con el fin de poder contabilizar cuál es el número total de créditos de todos los alumnos).

Para ello, se pide diseñar un **Analizador Léxico** para este lenguaje (*tokens* completos, gramática regular, autómata finito determinista y acciones semánticas).

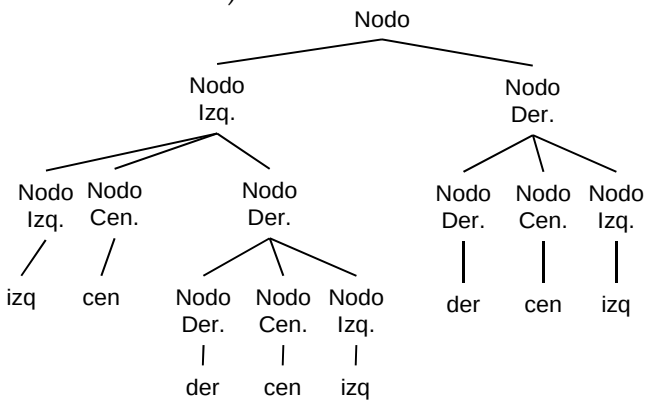
ANÁLISIS SINTÁCTICO

Segundo examen. 26 de noviembre de 2012

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

En un determinado entorno se generan árboles como el de la figura. Se desea construir un sistema capaz de validar la estructura de los árboles de acuerdo a las siguientes reglas (donde se han abreviado los nombres de los nodos):

N → I D
I → I C D
I → izq
D → D C I
D → der
C → cen



Se pide:

a. Construir el **Autómata Reconocedor de Prefijos Viables** (correspondiente al método de análisis sintáctico SLR).

b. Identifica todos los posibles **conflictos** en el Autómata anterior.

c. Una vez transformada la gramática anterior para eliminar la recursividad por la izquierda se obtiene:

N → I D
I → izq I'
I' → C D I' | λ
D → der D'
D' → C I D' | λ
C → cen

Comprobar si esta nueva gramática cumple la Condición **LL(1)**, detectando todos los posibles incumplimientos.

COMPILADORES

Examen primer parcial. 22 de enero de 2013

Observaciones: 1. Las calificaciones se publicarán hacia el 6 de febrero.
2. La revisión será hacia el 8 de febrero.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen será de 2 horas.
5. Todos los ejercicios tienen la misma puntuación.

1. Una descripción de una de las etiquetas de un lenguaje de marcado es:

<img *atributo=valor*>
La etiqueta *img* define una imagen en una página. Tiene dos atributos obligatorios: *src* y *alt*, y el resto de atributos son opcionales. Los atributos válidos son los siguientes:

Atributo	Descripción
accesskey	Especifica un atajo de teclado para activar el elemento
alt	Especifica un texto alternativo para la imagen
class	Especifica uno o más nombres de clase
dir	Especifica la dirección del texto
height	Especifica la altura de la imagen. El valor debe ser un número entero o un porcentaje entero
id	Especifica un identificador único asociado a la imagen
lang	Especifica el idioma en el que está la imagen
src	Especifica la URL donde está la imagen
style	Especifica el estilo para la imagen
tabindex	Especifica el orden de tabulación del elemento. El valor debe ser un número entero
title	Especifica información adicional sobre la imagen
width	Especifica el ancho de la imagen. El valor debe ser un número entero o un porcentaje entero

Los atributos se definen, en la forma en que se muestra en el ejemplo, con un valor que será una cadena, un número entero positivo o un porcentaje; la cadena puede contener cualquier carácter (excepto el salto de línea) en cualquier cantidad (incluso es válida la cadena vacía). Los elementos del lenguaje pueden ir separados por blancos, tabuladores o saltos de línea.

Ejemplo:

```
<img src="smiley.gif" alt="Smiley face" height=50 width=50% id="cara"><img
src="http://www.mynewsite.com/images/new/~very%20bigbuttonOK.jpg" alt="Aceptar" >
< img alt="Mapa que muestra la ruta al tesoro"       src="img/mapadeltesoro.jpg">
<img tabindex = 5 lang= "es" title ="Examen final" src="1.jpg" alt="Examen" width
= 88 height=31 style="vertical-align: middle; border: 4px;">
< img width=120% src="azul.png" alt="" class="fondo" id="fondo_azul_claro_3">
```

Se pide diseñar el **Analizador Léxico** correspondiente a este fragmento de lenguaje, incluyendo la gramática regular, los *tokens* completos, el autómata finito determinista y las acciones semánticas.

2. Sea la siguiente gramática LL(1):

S → A B | x y A
A → w x | λ | D z E
B → y z
D → z D
E → z E | λ

Se pide:

a. Calcular los conjuntos **First** y **Follow** para todos los símbolos no terminales.

b. Diseñar los procedimientos del **Analizador Sintáctico Descendente Recursivo** para esta gramática utilizando el procedimiento auxiliar que se define a continuación para comparar con el siguiente *token* y avanzar:

```
Procedure eq_tk (tok: token)
Begin
      If (preanálisis = tok)
            Then preanálisis:= scan ()
            Else error (tok)
End
```

c. Determinar si añadiendo las siguientes reglas a la gramática anterior se cumple o no la **condición LL(1)** y, en caso negativo, indicar todas las reglas que incumplen dicha condición.

B → λ
D → y

3. Sea el siguiente fragmento de gramática:

S → for (A ; E ; P) { S } | if (E) { S } | A ; | continue ; | S S
A → id = E
E → P | id
P → ++id

Se pide el diseño del **Analizador Semántico**, mediante una Definición Dirigida por Sintaxis, detallando todos los accesos a la Tabla de Símbolos y explicando brevemente los atributos y funciones utilizadas, teniendo en cuenta que:

- El lenguaje exige declaración previa de variables
- Los tipos del lenguaje son entero, cadena y lógico
- La sentencia for funciona de la siguiente manera: se inicializa la variable índice mediante la asignación A y, si la condición E se evalúa como cierta, se ejecuta el cuerpo del for (S); luego, se actualiza dicha variable índice (P) (exactamente la variable que se ha inicializado en A) y se vuelve a comprobar la condición para ver si hay que volver a ejecutar el cuerpo del for; el bucle termina cuando la condición sea falsa
- La sentencia continue hace que se termine la iteración en curso del for y se pase, automáticamente, a la siguiente iteración (que se ejecuta, como siempre, si la condición E se evalúa como cierta). Una sentencia continue no puede estar fuera del cuerpo de un for en un programa correcto
- No existe conversión automática de tipos
- El operador de preincremento aumenta en una unidad el valor de una variable entera o cambia el valor de una variable lógica (negación lógica)

PROCESADORES DE LENGUAJES

Primer Examen, 9 de abril de 2013

Observaciones:	1. Fecha estimada de publicación de las calificaciones: 23 de abril. 2. Fecha estimada de la revisión: 26 de abril. 3. Duración del examen: 40 minutos.

Un fragmento de un lenguaje tiene las siguientes características generales, de las cuales habrá que tener en cuenta las necesarias para construir un Analizador Léxico:

- Declaración obligatoria de variables numéricas (sin tipo), que pueden ser opcionalmente inicializadas. Las variables pueden almacenar enteros o reales, indistintamente, pudiendo variar el tipo durante la ejecución, por lo que su tipo dependerá del último valor asignado. Su sintaxis es:

  D→ var id I
  I → = cte_ent | = cte_real | λ

- Declaración de funciones. Las funciones devuelven siempre un entero. Pueden llevar cualquier número de parámetros (siempre de tipo entero), que se pasan siempre por valor. Las funciones se pueden anidar y pueden ser recursivas. Su sintaxis es:

  D→ function id (A) begin S end
  A→ id B | λ
  B→ , id B | λ

- Sentencias. Considérense solamente las sentencias de asignación y de retorno, con la siguiente sintaxis:

  S→ id = E | id = id (C) | return E
  C→ E D | λ
  D→ , E D | λ
  E→ cte_ent | cte_real | id | id (C)

- Identificadores. Su nombre comienza por una letra, que puede ir seguida de cualquier cantidad de letras o dígitos.
- Números. Los números reales tienen obligatoriamente parte entera y parte decimal (no se usa la notación científica para su representación). Los números enteros están formados por cualquier cantidad de dígitos.
- Delimitadores. Cada sentencia o declaración tiene que terminar obligatoriamente con un salto de línea. Los elementos léxicos del lenguaje pueden usar como delimitador el blanco.
- Palabras clave. Todas las palabras clave del lenguaje son reservadas.

Se pide:

a. Construir un A**nalizador Léxico** (gramática BNF regular, *tokens*, AFD y acciones semánticas) para todo el fragmento de lenguaje descrito y que introduzca el máximo de información posible en la tabla de símbolos.

b. Explicar la estructura general de la **Tabla de Símbolos** para este lenguaje. Detallar cómo iría cambiando la tabla de símbolos al realizar un análisis léxico, sintáctico y semántico del siguiente programa:

```
Var a= 1
Function p(a, b ,c)
   Function q(b)
   Begin
      Return p(a, b, 2)
   End
Var d=8
Var e
Begin
   d=8.2
   d=a
   b=q(c)
   Return b
End
a=p(3, a, 2)
```

PROCESADORES DE LENGUAJES

ANÁLISIS SINTÁCTICO

Segundo examen. 7 de mayo de 2013

Observaciones: 1. Las calificaciones se publicarán hacia el 20 de mayo. 2. La revisión será hacia el 22 de mayo. 3. En la web se avisarán las fechas exactas. 4. La duración de este examen es de 40 minutos.

Sea el siguiente fragmento de una gramática:

F → C I
C → a L | x L | L
L → e | λ
R → e | -
I → e | - O
O → e | L R

Se pide:

a. Utilizando la plantilla adjunta, construir el **Autómata Finito Determinista** reconocedor de los prefijos viables para realizar un **Analizador Sintáctico SLR (1)**. Nota: la plantilla puede contener errores y omisiones de todo tipo.

b. A la vista del Autómata Finito Determinista obtenido, estudiar todos los estados en los que puede haber **conflictos** y justificar si la gramática es o no SLR (1)

c. Construir la **tabla** para realizar un **Analizador Sintáctico LL (1)**.

d. A la vista de la tabla obtenida, justificar si la gramática es o no LL(1)

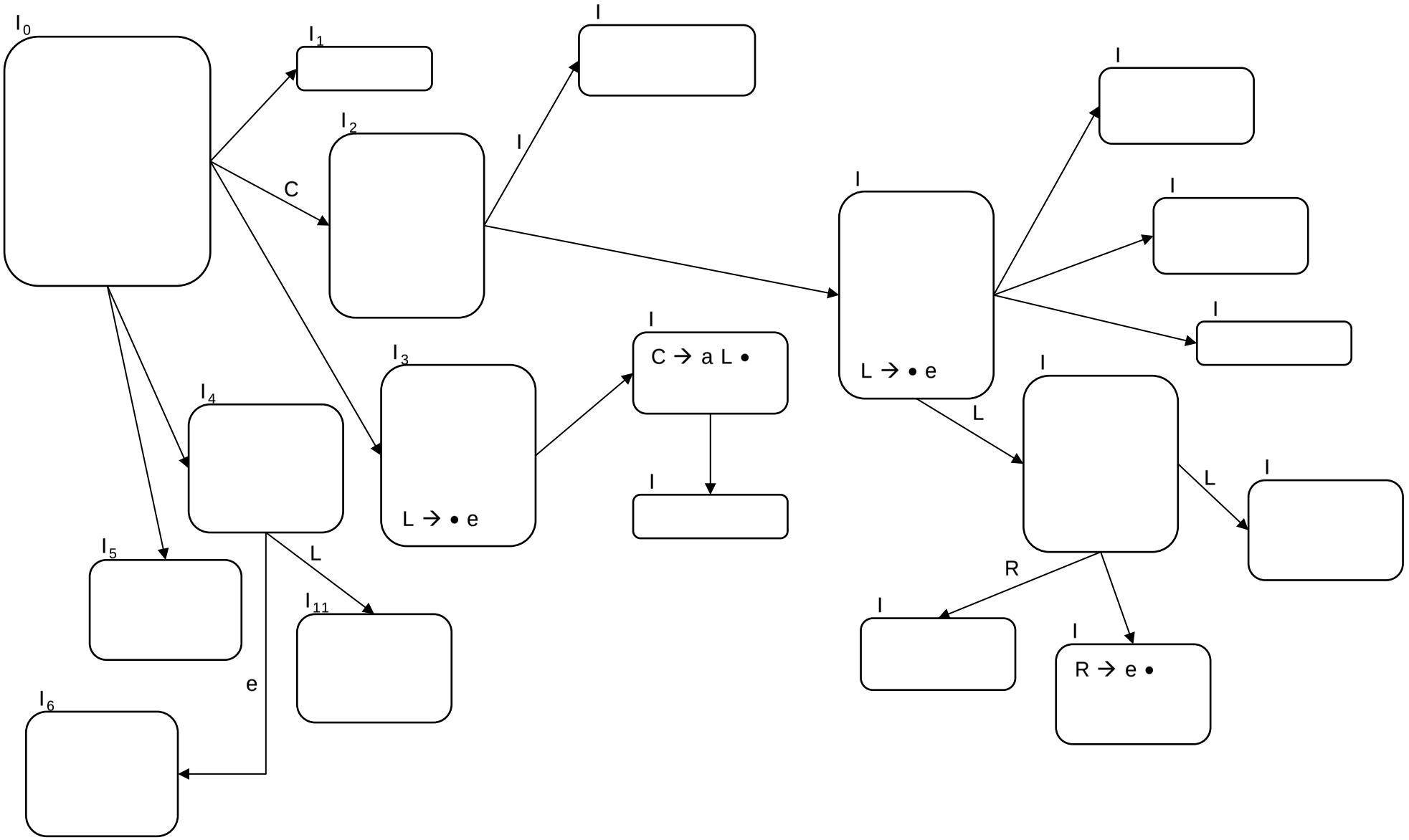
Analizador LR

7-mayo-2013

Hoja de Respuesta

Apellidos: .....

Nombre: .....



COMPILADORES Y PROCESADORES DE LENGUAJES
11 de junio de 2013

Observaciones: 1. Fecha aproximada de publicación de las calificaciones: 17-junio (Procesadores de Lenguajes) y 27-junio (Compiladores).
2. Fecha aproximada de la revisión: 19-junio (Procesadores de Lenguajes) y 2-julio (Compiladores).
3. Cada ejercicio debe entregarse en hojas separadas.
4. Cada ejercicio tiene la misma puntuación.
5. Los alumnos de "**Compiladores**" deben realizar los ejercicios 1, 2 y 3 y disponen de 2 horas.
6. Los alumnos de "**Procesadores de Lenguajes**" deben realizar únicamente el ejercicio 3 y disponen de 40 minutos.

1. Un lenguaje dispone, entre otros, de los siguientes elementos:

- Identificadores: pueden comenzar por una letra o un dígito y pueden contener cualquier cantidad de letras o dígitos, siempre que al menos tengan una letra.
- Enteros: formados por una secuencia no vacía de dígitos.
- Reales en notación normal: formados por una secuencia no vacía de dígitos, un punto y otra secuencia no vacía de dígitos.
- Reales en notación científica: formados por una secuencia no vacía de dígitos, seguida, opcionalmente, de un punto y una secuencia no vacía de dígitos. A continuación, tras la letra `E`, irá la parte exponencial formada por un signo opcional y un valor entero.

Los números enteros tienen que tener un valor menor a 2^{15}. Los números reales tienen que tener un valor menor a 10^{100}. En caso de ambigüedad entre el nombre de un identificador y un número real (por ejemplo, `5E3`), se considera siempre que es un número real.

La tabla resume algunos ejemplos válidos y erróneos de estos elementos:

identificador	entero	real, notación normal	real, notación científica	errores	
A	0	0.0	3E5	3.2E3F	-54
8E	88	44.3	4.102030E1	3E-5A	1.2E3.4
DC10	432	3.88888888	378E+3	J8.3	6.
2E3F4	9876	54321.0	6.023E-23	.5	5.E2
VARIABLES	22222	0.33333333333333	0.007E101	33E99	33333

Se pide diseñar la gramática regular, *tokens*, autómata, acciones semánticas (indicando los accesos a la Tabla de Símbolos) y errores de un **Analizador Léxico** para este lenguaje.

2. Sea la siguiente gramática que genera sentencias del tipo *while, if-then-else* e *if-then-elsif*:

S → while C do S | if C S else S | if C S elsif C S | λ
C → id

Se pide:

a. Se pide construir el **Autómata Reconocedor de Prefijos Viables** (método SLR).

b. Independientemente de si hay conflictos o no, construir las filas 1, 2, 3 y 6 de la **Tabla** de Análisis de dicho analizador.

3. Sea el siguiente fragmento de una gramática:

```
P → D S
D → id : T I D  |  λ
T → real  |  int  |  array [ cte-ent ] of T
I → λ  |  := C
C → A  |  { L }
A → cte-ent  |  cte-real
L → A , L  |  A
S → id := E  |  id [ E ] := E  |  S S
E → C  |  id  |  id [ E ]
```

Se pide diseñar el **Analizador Semántico** con una Traducción Dirigida por la Sintaxis para esta grámatica, teniendo en cuenta que:

- Los elementos del vector sólo pueden ser enteros o reales, nunca otros vectores.
- Todos los elementos de un vector han de ser del mismo tipo.
- No hay conversiones automáticas de tipos.
- Supóngase que los enteros ocupan 2 bytes y los reales 4 bytes.
- El Analizador Léxico inserta los identificadores en la Tabla de Símbolos.
- Se debe asumir que el índice de los vectores va desde 1 hasta el valor cte-ent indicado en la declaración, y siempre es entero.
- El lenguaje tiene declaración de variables (D), que pueden ser inicializadas (I).
- El lenguaje permite llenar un vector completo con una serie de valores constantes (L), pero se tienen que incluir exactamente tantos valores constantes como elementos tenga el vector y deben ser del tipo de los elementos del vector.
- Seguidamente se muestran, a modo de ejemplo, algunas sentencias válidas de este lenguaje:

```
a: int:= 2              // declaración e inicialización de una variable entera
b: real                 // declaración de una variable real
u: array [5] of int     // declaración de una variable vector de 5 enteros
v: array [5] of int
w: array [3] of real:={3.3, 7.7, 8.8}   // declaración e inicialización de una variable vector de 3 reales
u:= {0, 2, 4, 6, 8}     // asignación de valores a una variable vector de 5 enteros
v:= u                   // copia una variable vector a otra variable vector necesariamente del mismo tipo
v[1]:= 1
v[a]:= a
u[v[1]]:= u[a]
b:= w[a]
w[3]:= b
```

PROCESADORES DE LENGUAJES
10 de julio de 2013

Observaciones: 1. Fecha aproximada de publicación de las calificaciones: 16-julio.
2. Fecha aproximada de la revisión: 18-julio.
3. Cada ejercicio debe entregarse en hojas separadas.
4. Cada ejercicio tiene la misma puntuación.
5. La duración de este examen es de 2 horas

1. Una empresa de energía nuclear quiere gestionar de manera centralizada la información de sus almacenes que recibe diariamente en un fichero de texto. Para ello, desea construir un sistema informático que capture toda esta información para poderla tratar automáticamente. El formato del fichero es el siguiente:

El código de almacén (formado por una secuencia no vacía de hasta 8 letras o dígitos, pudiendo ser el primero una letra o un dígito y debiendo existir obligatoriamente al menos una letra) va seguido del número de bidones de residuos que están almacenados en dicho almacén (un valor entero menor que 10000); a continuación viene la temperatura media del almacén (un valor real inferior a 200, en el que tanto la parte entera como la parte decimal son obligatorias); seguidamente viene información sobre la velocidad de desintegración radiactiva (que se representa por un valor real en notación científica, es decir, una secuencia no vacía de dígitos, opcionalmente una coma seguida de otra secuencia no vacía de dígitos, la letra 'e', un signo opcional y una secuencia no vacía de dígitos).

Aunque éste es el orden habitual de los elementos del fichero, algunas veces se reciben los valores en otro orden o se ha omitido alguno de ellos.

A continuación se muestra un ejemplo correcto del formato de este fichero:

```
sprngfld 9999 199,999999 99e99  8emount    8,8  88      1,123456789e+1
director    0   22,0 2z4x6y8  0,3e-23 River10   33   0,33333333333333
     0,003e303  7e   0,0  9e0
```

A continuación se muestra un ejemplo de fichero en el que todos los elementos son incorrectos:

```
springfield 99999     299,999999 9,9e9i9 a_h 87654321 -5            8,
     0,1234Le+1  ,30 02z4x6y80  0h,0 0,e-23     200,0      0,003e3o3
```

Teniendo en cuenta que ningún almacén tiene un código formado por dígitos y una única letra 'e' en su interior (por ejemplo, `12e45`) y que todos los códigos de almacén se encuentran guardados en una tabla, se pide diseñar la gramática regular, *tokens*, autómata, acciones semánticas y errores de un **Analizador Léxico** para este lenguaje.

2. Sea la siguiente gramática G:

P → D S
D → T : id ; D | λ
T → real | int
S → if id then S else S ; S | forEach id in id S ; S | λ
E → id | id [E]

Se pide:

a. Construir una gramática G' equivalente sin recursividad a izquierdas y factorizada.
b. Calcular los conjuntos *First* y *Follow* de todos los símbolos no terminales de G'.
c. Construir la tabla de un **Analizador Sintáctico LL(1)** para G' y justificar si la gramática G' es LL(1).
d. Diseñar los procedimientos del **Analizador Sintáctico Descendente Recursivo** correspondientes a los símbolos D y S (puede utilizarse un procedimiento auxiliar para equiparar *tokens*).

3. Sea el fragmento de un lenguaje generado por la siguiente gramática:

```
S → for ( Cont ; E ; id ++ ) { S } | id := E | S ; S
Cont → int S
E → cte_entera | cte_real | true | false | id op_rel E | id ++ | id
op_rel → > | < | =
```

Se pide diseñar el **Analizador Semántico** mediante una Definición Dirigida por la Sintaxis para esta grámatica, teniendo en cuenta que:

- El lenguaje tiene, al menos, los tipos entero, real y lógico.
- No hay conversión automática de tipos.
- Los identificadores deben estar declarados previamente a su uso.
- La sentencia `for` tiene:
  - la declaración de un contador (`Cont`), que siempre será de tipo entero, y que se inicializa mediante una sentencia de asignación.
  - una condición de finalización (`E`), que viene dada por una expresión que comprueba si el identificador que se ha declarado como contador es igual, mayor o menor que una expresión entera.
  - un incremento, que se encarga de ir incrementando el identificador contador.
- El operador incremento (`++`) sólo se puede aplicar a datos enteros.
- Los operadores relacionales (`op_rel`) se pueden aplicar a datos enteros o reales.

COMPILADORES Y PROCESADORES DE LENGUAJES
11 de junio de 2013

Observaciones: 1. Fecha aproximada de publicación de las calificaciones: 17-junio (Procesadores de Lenguajes) y 27-junio (Compiladores).
2. Fecha aproximada de la revisión: 19-junio (Procesadores de Lenguajes) y 2-julio (Compiladores).
3. Cada ejercicio debe entregarse en hojas separadas.
4. Cada ejercicio tiene la misma puntuación.
5. Los alumnos de "**Compiladores**" deben realizar los ejercicios 1, 2 y 3 y disponen de 2 horas.
6. Los alumnos de "**Procesadores de Lenguajes**" deben realizar únicamente el ejercicio 3 y disponen de 40 minutos.

1. Un lenguaje dispone, entre otros, de los siguientes elementos:

- Identificadores: pueden comenzar por una letra o un dígito y pueden contener cualquier cantidad de letras o dígitos, siempre que al menos tengan una letra.
- Enteros: formados por una secuencia no vacía de dígitos.
- Reales en notación normal: formados por una secuencia no vacía de dígitos, un punto y otra secuencia no vacía de dígitos.
- Reales en notación científica: formados por una secuencia no vacía de dígitos, seguida, opcionalmente, de un punto y una secuencia no vacía de dígitos. A continuación, tras la letra `E`, irá la parte exponencial formada por un signo opcional y un valor entero.

Los números enteros tienen que tener un valor menor a 2^{15}. Los números reales tienen que tener un valor menor a 10^{100}. En caso de ambigüedad entre el nombre de un identificador y un número real (por ejemplo, `5E3`), se considera siempre que es un número real.

La tabla resume algunos ejemplos válidos y erróneos de estos elementos:

identificador	entero	real, notación normal	real, notación científica	errores	
A	0	0.0	3E5	3.2E3F	-54
8E	88	44.3	4.102030E1	3E-5A	1.2E3.4
DC10	432	3.88888888	378E+3	J8.3	6.
2E3F4	9876	54321.0	6.023E-23	.5	5.E2
VARIABLES	22222	0.33333333333333	0.007E101	33E99	33333

Se pide diseñar la gramática regular, *tokens*, autómata, acciones semánticas (indicando los accesos a la Tabla de Símbolos) y errores de un **Analizador Léxico** para este lenguaje.

2. Sea la siguiente gramática que genera sentencias del tipo *while, if-then-else* e *if-then-elsif*:

S → while C do S | if C S else S | if C S elsif C S | λ
C → id

Se pide:

a. Se pide construir el **Autómata Reconocedor de Prefijos Viables** (método SLR).

b. Independientemente de si hay conflictos o no, construir las filas 1, 2, 3 y 6 de la **Tabla** de Análisis de dicho analizador.

3. Sea el siguiente fragmento de una gramática:

P → D S
D → id : T I D | λ
T → real | int | array [cte-ent] of T
I → λ | := C
C → A | { L }
A → cte-ent | cte-real
L → A , L | A
S → id := E | id [E] := E | S S
E → C | id | id [E]

Se pide diseñar el **Analizador Semántico** con una Traducción Dirigida por la Sintaxis para esta grámatica, teniendo en cuenta que:

- Los elementos del vector sólo pueden ser enteros o reales, nunca otros vectores.
- Todos los elementos de un vector han de ser del mismo tipo.
- No hay conversiones automáticas de tipos.
- Supóngase que los enteros ocupan 2 bytes y los reales 4 bytes.
- El Analizador Léxico inserta los identificadores en la Tabla de Símbolos.
- Se debe asumir que el índice de los vectores va desde 1 hasta el valor cte-ent indicado en la declaración, y siempre es entero.
- El lenguaje tiene declaración de variables (D), que pueden ser inicializadas (I).
- El lenguaje permite llenar un vector completo con una serie de valores constantes (L), pero se tienen que incluir exactamente tantos valores constantes como elementos tenga el vector y deben ser del tipo de los elementos del vector.
- Seguidamente se muestran, a modo de ejemplo, algunas sentencias válidas de este lenguaje:

```
a: int:= 2              // declaración e inicialización de una variable entera
b: real                 // declaración de una variable real
u: array [5] of int     // declaración de una variable vector de 5 enteros
v: array [5] of int
w: array [3] of real:={3.3, 7.7, 8.8}   // declaración e inicialización de una variable vector de 3 reales
u:= {0, 2, 4, 6, 8}     // asignación de valores a una variable vector de 5 enteros
v:= u                   // copia una variable vector a otra variable vector necesariamente del mismo tipo
v[1]:= 1
v[a]:= a
u[v[1]]:= u[a]
b:= w[a]
w[3]:= b
```

PROCESADORES DE LENGUAJES
10 de julio de 2013

Observaciones: 1. Fecha aproximada de publicación de las calificaciones: 16-julio.
2. Fecha aproximada de la revisión: 18-julio.
3. Cada ejercicio debe entregarse en hojas separadas.
4. Cada ejercicio tiene la misma puntuación.
5. La duración de este examen es de 2 horas

1. Una empresa de energía nuclear quiere gestionar de manera centralizada la información de sus almacenes que recibe diariamente en un fichero de texto. Para ello, desea construir un sistema informático que capture toda esta información para poderla tratar automáticamente. El formato del fichero es el siguiente:

El código de almacén (formado por una secuencia no vacía de hasta 8 letras o dígitos, pudiendo ser el primero una letra o un dígito y debiendo existir obligatoriamente al menos una letra) va seguido del número de bidones de residuos que están almacenados en dicho almacén (un valor entero menor que 10000); a continuación viene la temperatura media del almacén (un valor real inferior a 200, en el que tanto la parte entera como la parte decimal son obligatorias); seguidamente viene información sobre la velocidad de desintegración radiactiva (que se representa por un valor real en notación científica, es decir, una secuencia no vacía de dígitos, opcionalmente una coma seguida de otra secuencia no vacía de dígitos, la letra 'e', un signo opcional y una secuencia no vacía de dígitos).

Aunque éste es el orden habitual de los elementos del fichero, algunas veces se reciben los valores en otro orden o se ha omitido alguno de ellos.

A continuación se muestra un ejemplo correcto del formato de este fichero:

```
sprngfld 9999 199,999999 99e99  8emount    8,8  88      1,123456789e+1
director    0    22,0 2z4x6y8  0,3e-23 River10   33    0,33333333333333
     0,003e303  7e    0,0   9e0
```

A continuación se muestra un ejemplo de fichero en el que todos los elementos son incorrectos:

```
springfield 99999     299,999999 9,9e9i9 a_h 87654321 -5             8,
     0,1234Le+1  ,30 02z4x6y80  0h,0 0,e-23     200,0      0,003e3o3
```

Teniendo en cuenta que ningún almacén tiene un código formado por dígitos y una única letra 'e' en su interior (por ejemplo, `12e45`) y que todos los códigos de almacén se encuentran guardados en una tabla, se pide diseñar la gramática regular, *tokens*, autómata, acciones semánticas y errores de un **Analizador Léxico** para este lenguaje.

2. Sea la siguiente gramática G:

P → D S
D → T : id ; D | λ
T → real | int
S → if id then S else S ; S | forEach id in id S ; S | λ
E → id | id [E]

Se pide:

a. Construir una gramática G' equivalente sin recursividad a izquierdas y factorizada.
b. Calcular los conjuntos *First* y *Follow* de todos los símbolos no terminales de G'.
c. Construir la tabla de un **Analizador Sintáctico LL(1)** para G' y justificar si la gramática G' es LL(1).
d. Diseñar los procedimientos del **Analizador Sintáctico Descendente Recursivo** correspondientes a los símbolos `D` y `S` (puede utilizarse un procedimiento auxiliar para equiparar *tokens*).

3. Sea el fragmento de un lenguaje generado por la siguiente gramática:

```
S → for ( Cont ; E ; id ++ ) { S } | id := E | S ; S
Cont → int S
E → cte_entera | cte_real | true | false | id op_rel E | id ++ | id
op_rel → > | < | =
```

Se pide diseñar el **Analizador Semántico** mediante una Definición Dirigida por la Sintaxis para esta grámatica, teniendo en cuenta que:

- El lenguaje tiene, al menos, los tipos entero, real y lógico.
- No hay conversión automática de tipos.
- Los identificadores deben estar declarados previamente a su uso.
- La sentencia `for` tiene:
  - la declaración de un contador (`Cont`), que siempre será de tipo entero, y que se inicializa mediante una sentencia de asignación.
  - una condición de finalización (`E`), que viene dada por una expresión que comprueba si el identificador que se ha declarado como contador es igual, mayor o menor que una expresión entera.
  - un incremento, que se encarga de ir incrementando el identificador contador.
- El operador incremento (`++`) sólo se puede aplicar a datos enteros.
- Los operadores relacionales (`op_rel`) se pueden aplicar a datos enteros o reales.

COMPILADORES

Examen Final, 11 de septiembre de 2013

Observaciones: 1. La fecha estimada de publicación de las calificaciones es el 19 de septiembre.
2. La fecha estimada de la revisión es el 24 de septiembre.
3. La duración de este examen es de 2½ horas.
4. Cada ejercicio deberá entregarse en hojas separadas.

1. Sea la siguiente gramática:

$M \rightarrow A\ G\ W$
$G \rightarrow E\ n\ F$
$E \rightarrow n\ c \mid \lambda$
$A \rightarrow a\ W \mid b\ W \mid a\ b\ W$
$W \rightarrow b \mid \lambda$
$F \rightarrow c \mid \lambda$

Se pide:

a. Construir el **Autómata Reconocedor de Prefijos Viables** (método Análisis Sintáctico Ascendente LR)

b. Calcular los conjuntos **First** y **Follow** para cada símbolo no terminal y analizar sobre el autómata todos los **conflictos** que pudieran existir.

c. Enumerar los **estados** por los que **transitaría** el Analizador LR para analizar la cadena `abbncn`. En el caso de que en algún estado usado se presente algún conflicto, deberán seguirse todas las alternativas posibles.

(3,5 puntos)

2. De un lenguaje se ha extraído el siguiente fragmento de gramática:

$S \rightarrow id := E \mid \text{if } E = \text{false then } S_1$
$E \rightarrow id \mid E_1 \leq E_2 \leq E_3$

Se pide diseñar un **Esquema de Traducción** con el **Analizador Semántico** y el **Generador de Código Intermedio** teniendo en cuenta que:

- El lenguaje tiene variables enteras, reales y lógicas y se exige su declaración.
- La expresión de comparación múltiple se evalúa como cierta si el valor de E_2 se encuentra entre los valores de E_1 y E_3, ambos inclusive. La operación puede realizarse entre expresiones de cualquiera de los tres tipos disponibles, siempre que las tres expresiones sean del mismo tipo.
- El lenguaje realiza conversiones automáticas de tipos en todos los casos excepto en la expresión de comparación múltiple.
- Los valores lógicos tienen que representarse numéricamente (0 es falso y cualquier otro valor es verdadero).
- Deben explicarse brevemente los atributos y funciones utilizadas.

(3,5 puntos)

ANÁLISIS LÉXICO Y TABLA DE SÍMBOLOS

Primer examen. 24 de octubre de 2013

Observaciones: 1. Las calificaciones se publicarán hacia el 18 de noviembre.
2. La revisión será hacia el 20 de noviembre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen es de 40 minutos.

Un determinado fragmento de un lenguaje dispone de palabras reservadas, variables, números enteros y números reales. Estos elementos, que pueden estar separados entre sí por blancos, tienen las siguientes características:

- Palabras reservadas: el lenguaje dispone de un total de 99 palabras reservadas distintas. El nombre de una palabra reservada está formado por una cantidad cualquiera de letras. Además, cualquier palabra reservada se puede escribir también empezando y terminando en punto.
- Variables: el nombre de una variable puede estar formado por un número cualquiera de letras y dígitos, pero siempre comenzando por una letra. No obstante, aunque la longitud del nombre puede ser cualquiera, internamente solamente se tienen en cuenta los primeros 16 caracteres (por ejemplo, se considera que las siguientes dos variables son la misma: `a2345678901234 56uno`, `a2345678901234 56dos`).
- Números enteros: están formados por una secuencia de dígitos y su valor se tiene que poder representar en 16 bits.
- Números reales: están formados por una parte entera y una parte decimal separadas por un punto. Tanto la parte entera como la parte decimal están formadas por secuencias de dígitos. En un número real se puede omitir la parte entera o la parte decimal, pero nunca ambas.

Se muestran a continuación algunos ejemplos de los elementos de este lenguaje:

Palabras reservadas	**Variables**	**Enteros**	**Reales**
`IF`	`a`	`0`	`9.0`
`.then.`	`a2B3c4D5e`	`1`	`123456.654321`
`then`	`abcdefghijklmnopqrst`	`65535`	`88.`
`RevampWhenGreaterThan`	`qwerty`	`1234`	`.99`
`.Else.`	`ZX81`	`999`	`0.0`

Se pide diseñar un **Analizador Léxico** para este lenguaje (*tokens* completos, gramática regular, autómata finito determinista, acciones semánticas y errores), que introduzca toda la información posible en la Tabla de Símbolos.

ANÁLISIS SINTÁCTICO

Segundo examen. 28 de noviembre de 2013

Observaciones: 1. Las calificaciones se publicarán hacia el 16 de diciembre.
2. La revisión será hacia el 18 de diciembre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen será de 40 minutos.

Dado el siguiente fragmento de una gramática:

S → id = E ; S | λ
E → E + T | T
T → id ++ | ++ id | id

Se pide:

a. Completar en la plantilla adjunta el **Autómata Reconocedor de Prefijos Viables** de un Analizador Sintáctico SLR. La plantilla puede contener errores o estar incompleta.

b. Realizar el estudio de todos los posibles **conflictos** sobre el Autómata anterior.

c. Completar, de las **tablas** del Analizador Sintáctico **SLR**, las filas correspondientes a los estados I_3, I_6 e I_8 (que ya vienen numerados en la plantilla).

d. Construir la **tabla** del Analizador Sintáctico **LL(1)** para esta gramática

e. ¿Cuál es la **principal diferencia** entre los analizadores **LR**(1) y los **LL**(1)?

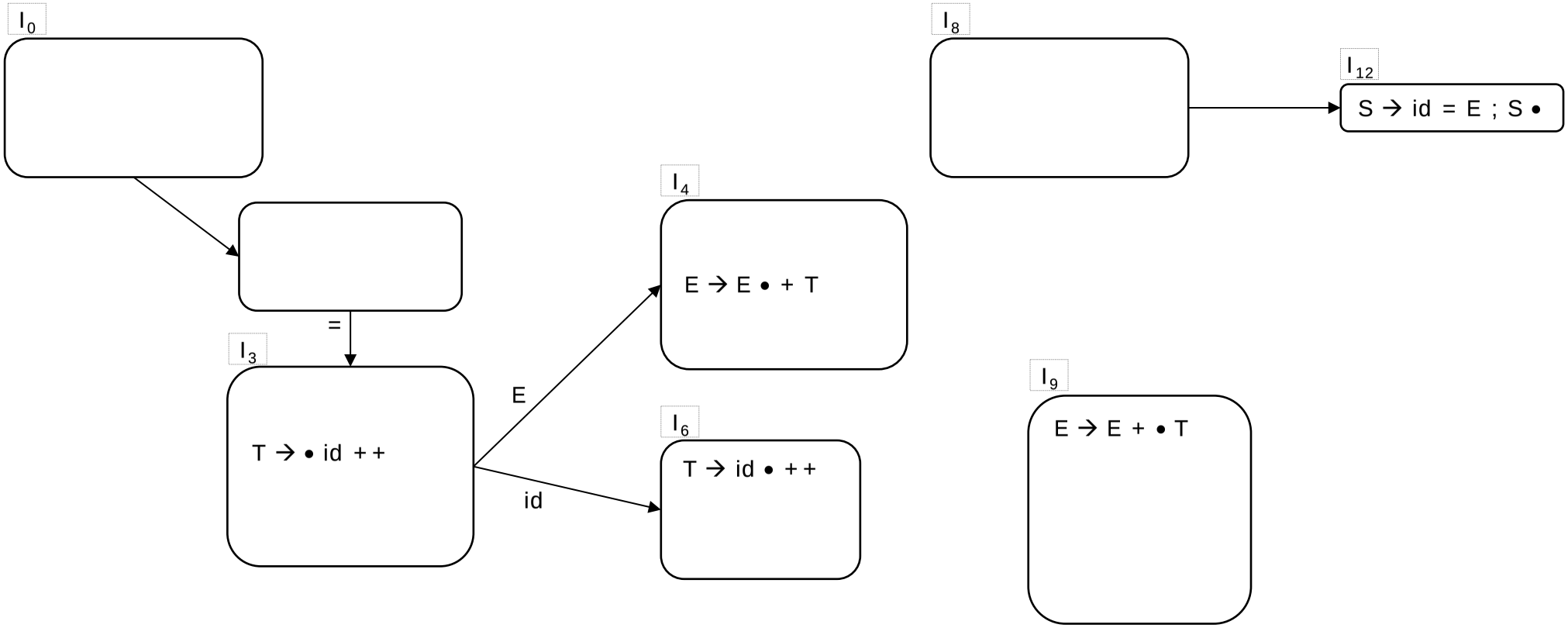
Análisis SLR

28-noviembre-2013

Hoja de Respuesta

Apellidos: .....

Nombre: .....



PROCESADORES DE LENGUAJES Y COMPILADORES

10 de enero de 2014

Observaciones: 1. Las calificaciones se publicarán hacia el 21 de enero.
2. La revisión será hacia el 23 de enero.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen es de 40 minutos por ejercicio.
5. Cada ejercicio deberá entregarse en hojas separadas.
6. Todos los ejercicios tienen la misma puntuación

1. Un determinado fragmento de un lenguaje de programación dispone de palabras reservadas, variables y números. El lenguaje distingue entre mayúsculas y minúsculas. Los elementos van siempre separados entre sí por blancos, tabuladores o saltos de línea. Estos elementos tienen las siguientes características:

- Palabras reservadas: el lenguaje dispone de un total de 111 palabras reservadas distintas. El nombre de una palabra reservada está formado por una cantidad cualquiera de letras. Una misma palabra reservada puede escribirse comenzando o no por un guión.
- Variables: el nombre de una variable puede estar formado por un máximo de 20 letras, dígitos o guiones, siempre comenzando por una letra o un guión, pero nunca terminando por guión.
- Números enteros: se representan en notación octal, mediante una secuencia de dígitos octales. Su valor se tiene que poder representar en 15 bits.
- Números reales: están formados por una parte entera obligatoria, un punto y una parte decimal opcional. Tanto la parte entera como la parte decimal están formadas por secuencias de dígitos. El valor de los números reales tiene que ser inferior a 2^{32}.

Se muestran a continuación algunos ejemplos de los elementos de este lenguaje:

Palabras reservadas	**Variables**	**Enteros**	**Reales**
`If`	`a`	`0`	`1.`
`-Then`	`-a2-B3-c4-D5-e6`	`1`	`4294967295.999999`
`Then`	`AbcdEfghIjklmnOpqrst`	`77777`	`77.7777`
`RevampWhenGreaterThan`	`-2`	`0246`	`0.987`
`Else`	`ZX-81`	`110`	`0.0`

Se pide diseñar un **Analizador Léxico** para este lenguaje (*tokens* completos, gramática regular, autómata finito determinista, acciones semánticas y errores), que introduzca toda la información posible en la Tabla de Símbolos.

2. Dado el siguiente fragmento de gramática:

P → D S
D → var id = R ; D | λ
R → int | bool
S → if (E) { S } else { S } ; S | id = E ; S | λ
E → T F
F → + T F | λ
T → id B | (E)
B → (H) | λ
H → E G
G → , E G | λ

Se pide:

a) Construir la tabla de un **Analizador Sintáctico LL(1)** (descendente tabular).
b) A la vista de la tabla, decir si la gramática dada es o no LL(1).
c) Realizar el análisis de la cadena `x = y` tal como lo haría el **Analizador Sintáctico LL(1)**.

3. Se pretende construir un Analizador para procesar programas dedicados al manejo de memorias flash. A grandes rasgos, estos programas constan de una serie de rutinas donde se especifican operaciones de borrado, escritura y lectura de la memoria flash. Sea el siguiente fragmento de gramática:

```
R → beginRutina id modo M S endRutina R
R → λ
M → borrado | escritura | lectura        / Modos de operación
S → λ
S → borrar E S                          / Operación
S → escribir E L S                      / Operación
S → leer E S                            / Operación
L → E , L
L → λ
E → id | id + E
```

Ejemplos de Programas:

```
beginRutina Rut modo escritura
   leer A
   borrar A  // Error
endRutina
```

```
beginRutina Rut modo borrado
   leer A
   escribir A B, C,
endRutina
```

Se deben tener en cuenta las siguientes características del dominio:

- Las variables (id) deben estar declaradas previamente a su uso y podrán ser del tipo entero, real o rutina.
- Los modos de operación tienen un orden de prioridad: borrado > escritura > lectura.
- Cada tipo de operación solo se puede realizar dentro de una rutina definida para su correspondiente modo de operación o para un modo de mayor prioridad.
- La expresión (E) de las sentencias de cada operación indica el bloque de datos de la memoria flash con el que se va a operar y deberá ser una variable entera cuyo valor esté comprendido en el intervalo [1, 32000].
- En la sentencia de escritura, L indica la secuencia de valores que se escribirán en la memoria y debe tener como mínimo un valor y como máximo 8.
- Dentro de cada rutina debe haber como mínimo una operación.

Se pide realizar un **Esquema de Traducción** con el diseño del **Analizador Semántico** para este lenguaje, y que proporcione mensajes de error adecuados a cada caso, explicando concisamente los atributos y funciones empleadas.

PROCESADORES DE LENGUAJES

Primer Examen, 31 de marzo de 2014

Observaciones: 1. Fecha estimada de publicación de las calificaciones: 22 de abril
2. Fecha estimada de la revisión: 24 de abril
3. Las fechas exactas se publicarán en: http://www-lt.ls.fi.upm.es/procesadores
4. Duración del examen: 40 minutos

Un Banco quiere crear una base de datos a partir de un fichero que contiene los nombres y apellidos, IBAN y saldo en cuenta corriente de sus clientes.

El IBAN es el Código internacional de Cuenta Bancaria (*International Bank Account Number*, en inglés) creado por el Comité Europeo de Normalización Bancaria (ECBS), en su norma ECBS 204. Es un código alfanumérico de entre 15 y 34 caracteres con la siguiente estructura[1]:

- Los 2 primeros caracteres son alfabéticos e indicativos del país (siguen la norma ISO 3166; por ejemplo, España es ES, la República Checa es CZ, Eslovaquia es SK, Suecia es SE, Túnez es TN, Noruega es NO, Malta es MT…).
- A continuación vienen 2 dígitos de control.
- Los restantes caracteres son numéricos.

Para saber si un IBAN es o no correcto, se aplica el siguiente procedimiento:

1. Convertir cada letra del IBAN en dos dígitos según la tabla:

A=10	B=11	C=12	D=13	E=14	F=15	G=16	H=17	I=18	J=19	K=20	L=21	M=22
N=23	O=24	P=25	Q=26	R=27	S=28	T=29	U=30	V=31	W=32	X=33	Y=34	Z=35

   Ej.: **ES**781234567891234567890 → **1428**781234567891234567890

2. Calcular el módulo 97 del número obtenido. Sólo si dicho resto es 1, el IBAN es correcto.

   Ej.: resto de la división entera: 142878123456789123456789 05 mod 97 = 1 ⇨ correcto

En cuanto al saldo en cuenta, vendrá expresado con una parte entera y, opcionalmente, una parte decimal (con una coma y 2 decimales) más un carácter correspondiente a la divisa. Ej.: 345€, 123,47$, 0¥, 100000p, 0,10£, 015R… Del saldo se quieren enviar la divisa y la cantidad.

Téngase en cuenta que los elementos del lenguaje están siempre delimitados por blanco, tabulador o salto de línea y que el lenguaje no distingue entre mayúsculas y minúsculas.

Se muestra a continuación, a modo de ejemplo, un fragmento de un fichero de entrada correcto:

```
ANA PI DE97979797979797979712  1000€      MARÍA JOSÉ    sánchez
DE ESPAÑA y GÓMEZ DEL Postigo ES7837837837837837837810 120,00$
ARNOLD ALOIS SCHWARZENEGGER AT756234567890756218 299,99£
```

Se muestra a continuación como ejemplo un fragmento de fichero erróneo (los errores están subrayados):

```
ANA PI DE979797979797          1000,0€     MARÍA JOSÉ    sánchez
DE ESPAÑA y GÓMEZ DEL3 Postigo ES1137837837837837837810 120,00
ARNOLD ALOIS SCHWARZENEGGER A756234567890756218  299,99£
```

Se pide diseñar el Analizador Léxico (*tokens*, gramática regular, autómata finito determinista, acciones semánticas y errores).

[1] Esta descripción y el método de cálculo indicado no corresponden exactamente al código IBAN real.

PROCESADORES DE LENGUAJES

ANÁLISIS SINTÁCTICO

Segundo examen. 12 de mayo de 2014

Observaciones: 1. Las calificaciones se publicarán hacia el 27 de mayo.
2. La revisión será hacia el 29 de mayo.
3. En la web se avisarán las fechas exactas.
4. La duración de este examen es de 40 minutos.

Sea el siguiente fragmento de una gramática:

R → begin id S end R
R → λ
S → escribe E L S
S → lee E S
S → λ
L → E , L
L → λ
E → id | id + E

Se pide:

a. Construir el **Autómata Finito Determinista** reconocedor de los prefijos viables para realizar un **Analizador Sintáctico SLR (1)**.
b. A la vista del Autómata Finito Determinista obtenido, indicar si los estados I_1 a I_9 presentan algún **conflicto**.
c. Construir el **Analizador Sintáctico Descendente Predictivo Recursivo** para las reglas de S, L y E (si es necesario, habrá que realizar los cambios pertinentes).

COMPILADORES
Primer parcial, 11 de junio de 2014

Observaciones: 1. Fecha aproximada de publicación de las calificaciones: 23-junio.
2. Fecha aproximada de la revisión: 25-junio.
3. Cada ejercicio debe entregarse en hojas separadas.
4. Cada ejercicio tiene la misma puntuación.
5. La duración del examen es de 2 horas.

1. Sea el siguiente fragmento de una gramática de tipo 2 (correspondiente a un Analizador Sintáctico) y un ejemplo de programa correcto correspondiente a la gramática:

R → begin id S end R | λ
S → escribe E L S | lee E S | λ
L → E , L | λ
E → id | id + cte_entera

```
Begin rutina14
     Escribe vartemporal posmemoria + 15678 ,
End
Begin r12utina
     Lee posmemoria + 1.285.354
End
```

Se pide diseñar el **Analizador Léxico** para este lenguaje:

a. Especificar el conjunto de ***tokens*** necesarios para dar servicio al analizador sintáctico. Especificar la regla-patrón para todos los *tokens*. Este patrón deberá ser el más sencillo que permita cubrir los casos del ejemplo. En el caso concreto de los números enteros, se deberán reconocer tanto números con puntos de millar correctamente colocados como sin puntos.
b. Diseñar la **gramática** regular.
c. Diseñar el **Autómata Finito Determinista**.
d. Diseñar las **acciones semánticas**.

2. Se tiene la siguiente gramática, correspondiente a un fragmento de un lenguaje de programación:

P → D S
D → var L T | λ
L → id | id , L
T → int
S → id := E
E → id | cte_ent

Se pide:

a. Construir el **Autómata Reconocedor de Prefijos Viables** (método SLR), completando la plantilla de la hoja de respuesta (que puede contener errores y omisiones).
b. Independientemente de si hay conflictos o no, construir las filas 0, 1, 3, 6, 7 y 10 (según la numeración de los estados del autómata en la plantilla) de las **Tablas** de Análisis de dicho analizador SLR(1).
c. Realizar el análisis de la cadena `var x int x := 7` utilizando solo las filas de la tabla obtenidas en el apartado anterior. Detener el análisis si se necesita una de las filas no indicadas.

3. Sea el siguiente fragmento de una gramática:

```
P → B S
B → D B  |  λ
D → T id I ;
I → = E  |  λ
T → int  |  float  |  struct { C }
C → D C  |  λ
S → id = E
E → cte_ent  |  cte_real  |  E . E  |  id
```

Se pide diseñar el **Analizador Semántico** con una Definición Dirigida por la Sintaxis para esta grámatica, teniendo en cuenta que:

- No hay conversiones automáticas de tipos.
- Supóngase que los enteros ocupan 2 bytes y los reales 4 bytes.
- El lenguaje tiene declaración de variables (D), que pueden ser inicializadas (I).
- El tipo struct representa un registro con una serie de campos de cualquier tipo
- Los campos de los registros no pueden inicializarse
- La expresión E.E representa el acceso a un campo de un registro

PROCESADORES DE LENGUAJES
3 de julio de 2014

Observaciones: 1. Fecha aproximada de publicación de las calificaciones: 11 ó 14-julio.
2. Fecha aproximada de la revisión: en la semana del 14 al 18 de julio.
3. Las fechas exactas de publicación y revisión se avisarán en la sección de avisos de la web: http://www-lt.ls.fi.upm.es/procesadores/Avisos.php
4. Cada ejercicio debe entregarse en hojas separadas.
5. Cada ejercicio tiene la misma puntuación.
6. La duración de este examen es de 2 horas

1. Un lenguaje tiene, entre otros, los siguientes tipos de elementos:

- Palabras reservadas: comienzan por una letra; a continuación, puede haber cualquier cantidad de letras o dígitos (o ninguno); el último carácter tiene que ser el símbolo del dólar ('$')
- Variables: comienzan por un subrayado ('_') o una letra; a continuación, puede haber cualquier cantidad de letras o dígitos (o ninguno); el último carácter tiene que ser una letra, un dígito o un subrayado ('_')
- Números: comienzan por un dígito o el símbolo de la almohadilla ('#'); a continuación, puede haber cualquier cantidad de dígitos (o ninguno); el último carácter tiene que ser un dígito. El símbolo de la almohadilla no afecta al valor del número
- Asignación: comienza por dos puntos (':') y termina por igual ('=')

Se pide diseñar la gramática regular, *tokens*, autómata, acciones semánticas y errores de un **Analizador Léxico** para este lenguaje, que introduzca toda la información posible en la Tabla de Símbolos y teniendo en cuenta que los distintos elementos pueden venir separados por blancos.

Se muestran a continuación algunos ejemplos válidos de elementos en este lenguaje:

Palabras reservadas	Variables	Números
`H$ abc123$ if$ a1b2c3$`	`A b8_ _ _8 _abc zx3`	`1 13246978 #0 #654 98`

2. Sea la siguiente gramática:

S → E F T | F
E → A B | C B
A → 1 F
B → 2 F
C → 3 S | λ
F → 4 | 5 A
T → C 2 | 5 S

Se pide:

a. Calcular los conjuntos **First** y **Follow** de todos los símbolos no terminales de la gramática.

b. Construir la **Tabla del Analizador Sintáctico Descendente** LL(1) y razonar si la gramática es LL(1)

c. Diseñar los procedimientos de los símbolos S y C correspondientes a un **Analizador Descendente Recursivo**.

3. Sea el fragmento de un lenguaje de programación generado por la siguiente gramática:

```
P → D S
D → var L T = I D  |  λ
L → id  |  id , L
T → int | float | boolean
I → E  |  E , I
S → if B < B then S  |  A
B → E  |  A
A → id := E
E → id  |  cte_ent |  E + E
```

Se pide diseñar el **Analizador Semántico** mediante una Definición Dirigida por la Sintaxis para esta grámatica, teniendo en cuenta que:

- En una declaración múltiple, el tipo se indica al final y se aplica a toda la lista; las inicializaciones tienen lugar en el mismo orden en que están las variables.
- Los enteros ocupan 2 bytes, los reales 4 y los lógicos 1 byte.
- Si la condición de la sentencia 'if' contiene alguna instrucción de asignación (A), dicha condición usa el valor de las variables de la parte izquierda de la asignación.
- La única conversión de tipos que tiene el lenguaje es de entero a real. No hay conversiones para el tipo lógico.
- El operador '<' solo es aplicable entre números.
- El operador '+' repesenta tanto la suma aritmética entre números como el OR lógico.
- Un ejemplo de un programa válido sería:

```
var x, y, z int = 3, 2, 9      // se inicializa x con 3, y con 2 y z con 9
var h float = 7                // se inicializa h con 7.0
var f, g int = x, 8+y          // se inicializa f con 3 y g con 10
if g < x:=5 then y:=f          // se asigna un 5 a x y se comprueba si g es menor que x
if h:=x < z:=7 then y:= g      // se asigna x a h, un 7 a z, y se comprueba si h es menor que z
```