# **Sintáctico-Semántico**

¿Cómo se comunican el Analizador Sintáctico y el Analizador Semántico?

**Respuesta seleccionada:** Cada vez que el Analizador Sintáctico usa una regla, le envía al Analizador Semántico el número de dicha regla

  La respuesta seleccionada fue correcta

**Explicación:**

-Así es como funciona. Ese número de la regla forma parte del *parse* correspondiente al análisis de la cadena

**Respuesta seleccionada:** Cada vez que el Analizador Sintáctico usa una regla, le envía al Analizador Semántico el fragmento de árbol correspondiente a dicha regla

  La respuesta seleccionada fue correcta

**Explicación:**

Así es como funciona. Ese fragmento de árbol se envía representándolo con el número de la regla, que forma parte del *parse* correspondiente al análisis de la cadena

# **Analizador Semántico: declaraciones simples**

Dado el siguiente fragmento de gramática de un lenguaje (en el que es obligatoria la declaración previa de variables), correspondiente a la declaración de variables enteras, reales, lógicas o vector, indicar cuáles de las siguientes reglas semánticas serían correctas y completas al diseñar un Analizador Semántico mediante un Esquema de Traducción:

D → var T id

T → integer | boolean | float | T [cte\_ent]

El atributo .pos refleja la entrada en la Tabla de Símbolos del elemento.

**Respuestas seleccionadas:**

**3:** T → T1 [cte\_ent] {T.tipo:= array(1..cte\_ent.valor, T1.tipo); T.ancho:= T1.ancho \* cte\_ent.valor}

  Esta respuesta es correcta.

**Explicación:**

Se obtiene el tipo de T como un vector cuyos elementos son del tipo de T1 y que tiene tantos elementos como indica el índice; también se calcula lo que ocupa

**4:** T → T1 [cte\_ent] {T.ancho:= T1.ancho \* cte\_ent.valor; T.tipo:= array(1..cte\_ent.valor, T1.tipo)}

  Esta respuesta es correcta.

**Explicación:**

Se obtiene el tipo de T como un vector cuyos elementos son del tipo de T1 y que tiene tantos elementos como indica el índice; también se calcula lo que ocupa

**Además, te ha faltado por seleccionar:**

**Respuesta:**

  T → integer {T.tipo:= entero; T.ancho:= 4}

**Explicación:**

Se está obteniendo el tipo entero y lo que ocupa

**1:**T → boolean {T.tipo:= lógico; T.ancho:= 1}

  Esta respuesta es correcta.

**2:**T → T1[cte\_ent] {T.tipo:= array(1..cte\_ent.valor, T1.tipo); T.ancho:= T1.ancho \* cte\_ent.valor}

  Esta respuesta es correcta.

**3:**T → T1[cte\_ent] {T.ancho:= T1.ancho \* cte\_ent.valor; T.tipo:= array(1..cte\_ent.valor, T1.tipo)}

  Esta respuesta es correcta.

**2:** D → var T id {AñadeTipoTS (id.pos, T.tipo); AñadeDesplTS (id.pos, despl); despl:= despl + T.ancho}

# **Semántica de la asignación**

Dada la regla semántica que aparece a continuación, señala la respuesta correcta relativa al lenguaje de programación que se está analizando:

S → id:= E {if (BuscaTipoTS (id.pos)=E.tipo) then S.tipo:=tipoOK else S.tipo:=tipoError}

**Respuesta seleccionada:** Los atributos tipo de E y de S son sintetizados

  La respuesta seleccionada fue correcta

**Respuesta seleccionada:** El lenguaje obliga a que los tipos en la asignación sean iguales

  La respuesta seleccionada fue correcta

**Respuesta seleccionada:** El lenguaje no realiza conversión automática de tipos

  La respuesta seleccionada fue correcta

**Explicación:**

No la realiza, dado que está obligando a que ambos operandos sean del mismo tipo

# **Analizador Semántico: declaraciones múltiples**

Dado el siguiente fragmento de gramática de un lenguaje (en el que es obligatoria la declaración previa de variables y en el que se pueden declarar varias variables en la misma instrucción de declaración), correspondiente a la declaración de variables enteras y lógicas, se pide diseñar un Analizador Semántico mediante un Esquema de Traducción:

D → var T id | var T L

T → integer | boolean

L → id R

R → , id R1 | λ

El atributo entrada indica la posición en la Tabla de Símbolos donde se encuentra el elemento

**Respuestas seleccionadas:**

**1:** T → integer {T.tipo:= entero; T.ancho:= 4}

  Esta respuesta es correcta.

**Explicación:**

Se guarda el tipo y el ancho del tipo

**2:** T → boolean {T.tipo:= lógico; T.ancho:= 1}

  Esta respuesta es correcta.

**Explicación:**

Se guarda el tipo y el ancho del tipo

**2:** D → var T { L.ancho:= T.ancho; L.tipo:= T.tipo } L

  Esta respuesta es correcta.

**Explicación:**

T tiene los valores de tipo y ancho y se los pasa a L (atributo heredado de L), para que después se puedan ir rellenando los tipos de cada uno de los id que aparecen en las reglas L → id R y R → , id R1

**Respuesta:**

  D → var T { L.tipo:= T.tipo; L.ancho:= T.ancho } L

**Explicación:**

T tiene los valores de tipo y ancho y se los pasa a L (atributo heredado de L), para que después se puedan ir rellenando los tipos de cada uno de los id que aparecen en las reglas L → id R y R → , id R1

**2:** D → var T id {AñadeTipoTS (id.pos, T.tipo); AñadeDesplTS (id.pos, despl); despl:= despl + T.ancho}

  Esta respuesta es correcta.

**Explicación:**

Se almacena la información sobre tipo y desplazamiento del id en la Tabla de Símbolos, y se actualiza el desplazamiento en función del ancho del tipo

4: R → , id { AñadeTipoyDesplTS (id.entrada, R.tipo, desp); despl:= despl + R.ancho; R1.tipo:= R.tipo; R1.ancho:= R.ancho } R1

**Explicación:**

Se crea correctamente la información del id en la Tabla de símbolos y se hereda para que las reglas de R puedan seguir funcionando

# **Análisis Semántico de expresiones**

Sean las siguientes reglas pertenecientes a la gramática de un lenguaje de programación que no tiene conversión automática de tipos, que exige la declaración previa de variables y que admite los tipos básicos de datos (entero, real, lógico, carácter y cadena). ¿Cuáles de las siguientes reglas semánticas es correcta (obviando las situaciones de error)?

E → id

E → id(A) // Llamada a una función

E → id[id] // Acceso a un elemento de un vector

A → E

A → A,A

El atributo .ent indica la posición en la Tabla de Símbolos donde se encuentra el elemento

**Respuestas seleccionadas:**

**1:** A→E {A.tipo:= E.tipo}

  Esta respuesta es correcta.

**Explicación:**

El parámetro tendrá el tipo de la expresión

**Respuesta:**

  E→id(A) {IF buscatipoTS(id.ent)=A.tipo→t THEN E.tipo:= t}

**Explicación:**

Primero se comprueba que efectivamente id se corresponde con una función previamente declarada y después se asigna el tipo devuelto por la función

**1:**E→id1[id2] {IF buscatipoTS(id1.ent)=array(i,t) AND buscatipoTS(id2.ent)=entero THEN E.tipo:= t}

  Esta respuesta es correcta.

**1:** E→id {E.tipo:= buscatipoTS(id.ent)}

  Esta respuesta es correcta.

**Explicación:**

El tipo de la expresión será el tipo previamente declarado para el identificador

# **Declaración múltiple**

¿Cuál de las siguientes afirmaciones es correcta, para el subconjunto de reglas sintácticas que se muestran y en relación con la regla semántica que aparece en la primera regla:

D → var T : {V.tipo:= T.tipo; V.ancho:= T.ancho} V

V → id | id , V

**Respuesta seleccionada:** Esta regla semántica calcula el valor de 2 atributos que podrán utilizarse para declarar varias variables del mismo tipo en una misma declaración

  La respuesta seleccionada fue correcta

**Explicación:**

Eso es precisamente lo que hace. Definir los atributos heredados tipo y ancho de V para que las reglas de V puedan usar esa información para la declaración

**Respuesta seleccionada:** Esta regla semántica define dos atributos heredados

  La respuesta seleccionada fue correcta

**Explicación:**

Define los atributos heredados tipo y ancho de V para que las reglas de V puedan usar esa información para la declaración

# **Análisis Semántico de expresiones con operandos especiales**

Sean las siguientes reglas pertenecientes a la gramática de un lenguaje de programación que no tiene conversión automática de tipos, que exige la declaración previa de variables y que admite los tipos básicos de datos (entero, real y cadena). Solo se pueden multiplicar números. La operación 'más' aplicada a números devuelve la suma y aplicada a cadenas devuelve la concatenación. ¿Cuales de las siguientes reglas semánticas es correcta?

S → id += E

S → id \*= E

E → id++ // suma 1 al id

E → id

**Respuestas seleccionadas:**

**2:** E→id++ {id.tipo:= buscatipoTS(id.pos); E.tipo:= if (id.tipo∈{ent,real}) then id.tipo else tipoError}

  Esta respuesta es correcta.

**Explicación:**

El tipo debe ser numérico y es el tipo que tendrá la expresión

**3:** E→id++ {E.tipo:= if (buscatipoTS(id.pos)∈{ent,real}) then buscatipoTS(id.pos) else tipoError}

  Esta respuesta es correcta.

**Explicación:**

El tipo debe ser numérico y es el tipo que tendrá la expresión

**1:**E→id {E.tipo:= buscatipoTS(id.pos)}

  Esta respuesta es correcta.

**2:**S→id+=E {S.tipo:= if (buscatipoTS(id.pos)=E.tipo) then tipoOK else tipoError}

  Esta respuesta es correcta.

**3:**S→id+=E {S.tipo:= if ((buscatipoTS(id.pos)=E.tipo) and (E.tipo∈{ent,real})) then tipoOK else if ((buscatipoTS(id.pos)=E.tipo) and (E.tipo=cadena})) then tipoOK else tipoError}

  Esta respuesta es correcta.

**2: S→id\*=E {S.tipo:= if ((buscatipoTS(id.pos)=E.tipo) and (E.tipo∈{ent,real})) then tipoOK else tipoError}**

**Esta respuesta es correcta.**

**Explicación:**

**La sentencia es correcta si se multiplican números**

# **Declaraciones**

Para el subconjunto de reglas sintácticas que se muestran y en relación con la regla semántica que aparece en la primera regla

D → var T : {V.tipo:= T.tipo; V.ancho:= T.ancho} V

V → id | id , V

¿Cuál de las siguiente afirmaciones es correcta?

**La respuesta correcta era:**

  Se podría eliminar dicha regla semántica si hiciéramos que V sintetizara una lista de posiciones de los identificadores y al final de D recorrer dichas posiciones para guardar la información necesaria en la Tabla de Símbolos

**Explicación:**

Sería otra forma de escribir el Esquema de Traducción sin utilizar atributos heredados

**Respuesta seleccionada:** Esta regla semántica calcula el valor de 2 atributos para que las reglas de V inserten el tipo y el desplazamiento en las entradas correspondientes a los id en la Tabla de Símbolos

  La respuesta seleccionada fue correcta

**Explicación:**

Exactamente esto es lo que permite esta regla semántica

**Respuesta seleccionada:** Esta regla semántica calcula el valor de 2 atributos heredados que podrán utilizarse para declarar varias variables del mismo tipo en una misma declaración

  La respuesta seleccionada fue correcta

**Explicación:**

Exactamente esto es lo que permite esta regla semántica

# **Errores**

¿Cuáles de los siguientes errores puede emitir un compilador durante el procesado de un fichero fuente?

**Respuestas seleccionadas:**

**1:** Errores semánticos

  Esta respuesta es correcta.

**Explicación:**

Son los detectados por el Analizador Semántico

**1:**Errores sintácticos

  Esta respuesta es correcta.

**1:** Errores léxicos

# **Analizador Semántico: declaración sin heredar**

Dado el siguiente fragmento de gramática de un lenguaje (en el que es obligatoria la declaración previa de variables y en el que se pueden declarar varias variables en la misma instrucción de declaración), correspondiente a la declaración de variables enteras y lógicas, se quiere diseñar un Analizador Semántico mediante un Esquema de Traducción que no utilice atributos heredados. ¿Cuáles de las siguientes acciones semánticas habría que emplear?:

D → var T id | var T L

T → integer | boolean

L → id R

R → , id R1 | λ

**Respuestas seleccionadas:**

**1:** T → integer {T.tipo:= entero; T.ancho:= 4}

  Esta respuesta es correcta.

**Explicación:**

Se guarda el tipo y el ancho del tipo

**2:** L → id R { L.entrada:= id.entrada ⊕ R.entrada; L.long:= R.long + 1 }

  Esta respuesta es correcta.

**Explicación:**

Se sintetiza una lista de entradas de identificadores para subirla por el árbol y se cuenta el número de identificadores que hay

T → integer { T.ancho:= 4; T.tipo:= entero }

  Esta respuesta es correcta.

**Explicación:**

Se guarda el tipo y lo que ocupa el tipo

**Además, te ha faltado por seleccionar:**

**Respuesta:**

  D → var T L { for i:= 1 to L.long do { AñadeTipoTS (L.entrada[i], T.tipo, despl); despl:= despl + T.ancho } }

**Explicación:**

Se ha subido hasta L la lista completa de los identificadores que se están declarando y se ha puesto un bucle for para completar tipo y desplazamiento de todos ellos en la Tabla de Símbolos

**1:**T → boolean {T.tipo:= lógico; T.ancho:= 1}

  Esta respuesta es correcta.

**Explicación:**

Se guarda el tipo y el ancho del tipo

**2:**D → var T id {AñadeTipoTS (id.entrada, T.tipo); AñadeDesplTS (id.entrada, despl); despl:= despl + T.ancho}

  Esta respuesta es correcta.

**Explicación:**

Se almacena la información sobre tipo y desplazamiento del id en la Tabla de Símbolos, y se actualiza el desplazamiento en función del ancho del tipo

D → var T id {AñadeTipoyDesplTS (id.entrada, T.tipo, despl); despl:= despl + T.ancho}

**Explicación:**

Se almacena la información sobre tipo y desplazamiento del id en la Tabla de Símbolos, y se actualiza el desplazamiento en función del ancho del tipo

**Además, te ha faltado por seleccionar:**

**Respuesta:**

  R → λ { R.entrada:= ∅; R.long:= 0 }

**Explicación:**

Como hay que sintetizar la lista de identificadores para subirla por el árbol, se inicializa la lista a vacía y el contador de variables a cero.

**Respuesta:**

  R → , id R1{ R.entrada:= id.entrada ⊕ R1.entrada; R.long:= R1.long + 1 }

**Explicación:**

Se sintetiza una lista de entradas de identificadores para subirla por el árbol y el contador de variables se incrementa por tener una variable más

# **Semántico y Errores**

El Analizador Semántico se encarga de...

**Respuesta seleccionada:** ...comprobar si los parámetros de una llamada a una función son correctos en número y tipo

  La respuesta seleccionada fue correcta

**Explicación:**

El Analizador Semántico tiene que comprobar que las llamadas a las funciones se produzcan de forma adecuada con los parámetros tal como se definió la función

**Respuesta seleccionada:** ...comprobar si los tipos en una asignación son válidos

  La respuesta seleccionada fue correcta

**Respuesta seleccionada:** ...comprobar el flujo del control (por ejemplo, si el return aparece dentro de una función con el tipo adecuado, o el break en un case, o el continue en una sentencia iterativa...)

  La respuesta seleccionada fue correcta

**Explicación:**

La comprobación del flujo del control es una de las tareas que realiza el Analizador Semántico

**Respuesta seleccionada:** ...comprobar si hay errores de tipos en las instrucciones del programa

  La respuesta seleccionada fue correcta

**Explicación:**

La comprobación de tipos es una de las principales tareas del Analizador Semántico

# **Entrada del Semántico**

¿Qué es lo que recibe el Analizador Semántico del Analizador Sintáctico para poder realizar su labor?

**Respuesta seleccionada:** El Analizador Semántico recibe el equivalente al fragmento del árbol de análisis sintáctico correspondiente a la regla que ha acaba de encontrar el Sintáctico

  La respuesta seleccionada fue correcta

**Explicación:**

La comunicación entre ambos módulos se limita al parse; en cada envío, se manda el número correspondiente a una regla (que equivale al fragmento del árbol de análisis sintáctico)

**Respuesta seleccionada:** El Analizador Semántico recibe del Sintáctico el parse

  La respuesta seleccionada fue correcta

**Respuesta:**

El Analizador Semántico recibe del Sintáctico el número de las reglas utilizadas

**Explicación:**

La comunicación entre ambos módulos se limita al parse

## 

## 

## 

## 

## **Análisis Semántico de expresiones y sentencias**

Sean las siguientes reglas pertenecientes a la gramática de un lenguaje de programación que no tiene conversión automática de tipos, que exige la declaración previa de variables y que admite los tipos básicos de datos (entero, real, carácter y cadena). Solo se pueden multiplicar números. La operación más aplicadas a números devuelve la suma y aplicada a cadenas devuelve la concatenación. ¿Cuáles de las siguientes reglas semánticas es correcta (obviando las situaciones de error)?

S → id = E

S → ++id // sentencia equivalente a id = id + 1

E → id

E → E + E

E → E \* E

**1:** S→id=E {id.tipo:=buscaTipoTS(id.pos); S.tipo:= if (id.tipo=E.tipo) then tipoOK else tipoError}

  Esta respuesta es correcta.

**Explicación:**

La asignación es correcta si ambos tipos son iguales

**2:** S→++id {id.tipo:= buscatipoTS(id.pos); S.tipo:= if (id.tipo∈{ent,real}) then tipoOK else tipoError}

  Esta respuesta es correcta.

**Explicación:**

El tipo debe ser numérico para que la sentencia sea correcta

**1:** S→++id {S.tipo:= if (buscatipoTS(id.pos)∈{ent,real}) then tipoOK else tipoError}

  Esta respuesta es correcta.

**Explicación:**

El tipo debe ser numérico para que la sentencia sea correcta

**2:** E→E1\*E2 {E.tipo:= if (E1.tipo=E2.tipo∈{ent,real}) then E1.tipo else tipoError}

  Esta respuesta es correcta.

**Explicación:**

Se pueden multiplicar números y el resultado es el tipo de dichos números

E→E1\*E2 {E.tipo:= if ((E1.tipo=E2.tipo) and (E1.tipo=ent)) then ent else if ((E1.tipo=E2.tipo) and (E1.tipo=real)) then real else tipoError}

**Respuesta:**

S→id=E {S.tipo:= if (buscaTipoTS(id.pos)=E.tipo) then tipoOK else tipoError}

**1:** S→++id {S.tipo:= if (buscatipoTS(id.pos)∈{ent,real}) then tipoOK else tipoError}

**2:** E→id {E.tipo:= buscatipoTS(id.pos)}

  Esta respuesta es correcta.

## **Errores de año**

Has terminado la actividad:

Un lenguaje tiene diferentes tipos de datos (entero, real, cadena, lógico, etc.) sin conversión de tipos, siendo todos ellos válidos para una instrucción de entrada por teclado. Se está analizando un programa que calcula automáticamente la edad de un usuario. Para ello, el programa pide que el usuario introduzca su año de nacimiento y lo resta del año actual, con cuyo valor se ha inicializado una variable entera (aunque se sabe que el programador se ha confundido de año al dar valor a la variable "año\_actual"). Si el usuario, además, introduce un número erróneo, ¿cuál de las siguientes afirmaciones es correcta?

**La respuesta correcta era:**

**Respuesta:**

El Procesador detectará un error semántico si el valor con que se ha inicializado la variable es un número real

**Explicación:**

Has terminado la actividad:

Un lenguaje tiene diferentes tipos de datos (entero, real, cadena, lógico, etc.) sin conversión de tipos, siendo todos ellos válidos para una instrucción de entrada por teclado. Se está analizando un programa que calcula automáticamente la edad de un usuario. Para ello, el programa pide que el usuario introduzca su año de nacimiento y lo resta del año actual, con cuyo valor se ha inicializado una variable entera (aunque se sabe que el programador se ha confundido de año al dar valor a la variable "año\_actual"). Si el usuario, además, introduce un número erróneo, ¿cuál de las siguientes afirmaciones es correcta?Será un error semántico

**Respuesta:**

El Procesador no detectará ningún error porque ya no está funcionando cuando ocurre ese hecho

**Explicación:**

Es algo que ocurre durante la ejecución del programa. Por tanto, si se está ejecutando es que era correcto y se ha compilado completamente

**Respuesta:**

El Procesador no detectará ningún error porque el programa es correcto

Esta respuesta es correcta.

**Explicación:**

Es algo que ocurre durante la ejecución del programa. Por tanto, si se está ejecutando es que era correcto y se ha compilado completamente

## **Análisis Semántico de declaración de funciones**

Has terminado la actividad:

Sea la siguiente regla perteneciente a la gramática de un lenguaje de programación con recursividad que no tiene conversión automática de tipos, que exige la declaración previa de identificadores y que admite como tipos básicos de datos: entero, real, lógico y cadena. Cada respuesta dada contiene una única acción semántica que podría pertenecer al conjunto completo de reglas semánticas de la regla gramatical. ¿Cuáles de las siguientes acciones semánticas deberían estar en un Esquema de Traducción?

F → TIPO function id ( L ) CUERPO

El atributo ent indica la posición en la Tabla de Símbolos donde se encuentra el elemento

**Respuestas Seleccionadas:**

**1:** F → TIPO function id {TSLocal:=CrearTS()} ( L ) CUERPO

  Esta respuesta es correcta.

**Explicación:**

Esta regla semántica crea una nueva Tabla de Símbolos para el ámbito definido por la función. La Tabla de Símbolos se crea después de recibir el *token* correspondiente al nombre de la función, para que el Analizador Léxico pueda introducirlo correctamente en la Tabla Global y no en la Local

**Además, te ha faltado por seleccionar:**

**Respuesta:**

F → TIPO function id ( L ) {InsertarTipoTS (id.ent, L.tipo⇾TIPO.tipo} CUERPO

**Explicación:**

Esta regla semántica inserta el tipo de la función en la Tabla de Símbolos. Lo hace una vez terminado de generar el subárbol de L, para poder obtener el valor del tipo de los parámetros de la función, así como el tipo devuelto

**1:** F → TIPO function id ( L {InsertarTipoTS (id.ent, L.tipo⇾TIPO.tipo} ) CUERPO

  Esta respuesta es correcta.

**Explicación:**

Esta regla semántica inserta el tipo de la función en la Tabla de Símbolos. Lo hace una vez terminado de generar el subárbol de L, para poder obtener el valor del tipo de los parámetros de la función, así como el tipo devuelto

**2:** F → TIPO function id ( L {InsertarTipoTS (id.ent, L.tipo⇾TIPO.tipo} ) CUERPO

  Esta respuesta es correcta.

**Explicación:**

Esta regla semántica inserta el tipo de la función en la Tabla de Símbolos. Lo hace una vez terminado de generar el subárbol de L, para poder obtener el valor del tipo de los parámetros de la función, así como el tipo devuelto

**Además, te ha faltado por seleccionar:**

**Respuesta:**

F → TIPO function id ( L ) {InsertarTipoTS (id.ent, L.tipo⇾TIPO.tipo} CUERPO

**1:**F → TIPO function id ( L ) CUERPO {If (CUERPO.return=false) Then error(“Función sin Return”)} /\* Si el lenguaje obliga a la existencia de al menos una sentencia Return dentro del cuerpo de la función, esta regla permite comprobarlo \*/

  Esta respuesta es correcta.

**Explicación:**

Este atributo sintetizado permitiría saber que existe al menos una sentencia Return dentro de la función, si se define S.Return:=true en la sentencia Return y se transmite adecuadamente hasta CUERPO

**2:**F → TIPO function id {TSLocal:=CrearTS()} ( L ) CUERPO

  Esta respuesta es correcta.

**Explicación:**

Esta regla semántica crea una nueva Tabla de Símbolos para el ámbito definido por la función. La Tabla de Símbolos se crea después de recibir el *token*correspondiente al nombre de la función, para que el Analizador Léxico pueda introducirlo correctamente en la Tabla Global y no en la Local

**Además, te ha faltado por seleccionar:**

**Respuesta:**

F → TIPO function id ( L ) {CUERPO.fun:=true} CUERPO /\* Si el lenguaje no permite poner Return fuera de las funciones, este atributo permite comprobar que toda sentencia Return está dentro del cuerpo de una función \*/

**Explicación:**

Permite que el CUERPO herede que se está en una función, para que la instrucción Return verifique que está bien ubicada. Se podría hacer con la regla semántica: S → Return {If (S.fun=false) Then error (“Return fuera de función”)

**F → TIPO function id ( L ) {InsertarTipoTS (id.ent, L.tipo⇾TIPO.tipo} CUERPO**

## **Análisis Semántico de expresiones con conversión de tipos**

Sean las siguientes reglas pertenecientes a la gramática de un lenguaje de programación que SÍ tiene conversión automática de tipos, que exige la declaración previa de variables y que admite los tipos básicos de datos (entero, real y cadena). La operación + aplicada sobre 2 cadenas devuelve la concatenación de ambas cadenas. ¿Cuáles de las siguientes reglas semánticas es correcta (obviando las situaciones de error)?

S → id = E

E → id

E → E + E

E → E \* E

**Respuesta:**

E→E1\*E2 {E.tipo:= if (E1.tipo=E2.tipo∈{ent,real}) then E1.tipo else if ((E1.tipo=ent and E2.tipo=real) or (E2.tipo=ent and E1.tipo=real)) then real else tipoError}

**Explicación:**

El producto mixto es un real y el de enteros es un entero

RESPUESTA: S id = E {S.tipo:= if (BuscaTipo(id.pos)=E.tipo)

then tipoOK

else if (BuscaTipoTS(id.pos) € {entero , real} and E.tipo € {entero, real})

then tipoOK else tipoError}

Explicacion: La asignacion es posible si ambos son del mismo tipo o de tipos compatibles.

S→id=E {id.tipo:=buscaTipoTS(id.pos); S.tipo:= if (id.tipo=E.tipo) then tipoOK else if (id.tipo∈{ent,real} and E.tipo∈{ent,real}) then tipoOK else tipoError}

**Explicación:**

La asignación es posible si ambos son del mismo tipo o de tipos compatibles

E→E1+E2{E.tipo:= if (E1.tipo=E2.tipo∈{ent,real,cadena}) then E1.tipo else if ((E1.tipo=ent and E2.tipo=real) or (E2.tipo=ent and E1.tipo=real)) then real else tipoError}

  Esta respuesta es correcta.

**Explicación:**

Los dos tipos tienen que ser numéricos para que la sentencia sea correcta o ambos deben ser de tipo cadena

**2:**E→E1\*E2{E.tipo:= if (E1.tipo=E2.tipo∈{ent,real}) then E1.tipo else if ((E1.tipo=ent and E2.tipo=real) or (E2.tipo=ent and E1.tipo=real)) then real else tipoError}

  Esta respuesta es correcta.

**Explicación:**

El producto mixto es un real y el de enteros es un entero

**Respuesta:**

E→id {E.tipo:= buscatipoTS(id.pos)}

**Explicación:**

El tipo de la expresión será el tipo previamente declarado para el identificador

**Sin Coerción**

Sean las siguientes reglas pertenecientes a la gramática de un lenguaje de programación que no tiene conversión automática de tipos, que exige la declaración previa de variables y que admite los tipos básicos de datos (entero, real, lógico, carácter y cadena). ¿Cuáles de las siguientes reglas semánticas son correctas (obviando las situaciones de error)?

E → id

E → id(A) // Llamada a una función

E → id[id] // Acceso a un elemento de un vector

A → E

A → A, A

**Respuesta:**

E → id {E.tipo:=buscaTipoTS(id.pos)}

Esta respuesta es correcta.

**Explicación:**

El tipo de la expresión será el tipo del identificador

**Respuesta:**

A → A1, A2 {A.tipo:=A1.tipo x A2.tipo}

**Explicación:**

Se acumula en A el tipo de los argumentos para después compararlos en la llamada a la función

**Respuesta:**

E → id1[id2] {IF buscatipoTS(id1.pos)=array(i,t) AND buscatipoTS(id2.pos)=entero THEN E.tipo:= t ELSE E.tipo:=tipo\_error}

Esta respuesta es correcta.

**Explicación:**

El tipo de la expresión será el tipo de los elementos del vector

# Errores del Procesador

En el listado se muestran distintos tipos de errores que puede detectar un Procesador de un lenguaje concreto, junto con el tipo de error, que se corresponde con el módulo que lo detecta. Indica si el tipo de error indicado realmente es correcto en cada caso.

**Respuesta:**

Error Sintáctico: Llamada a función sin el paréntesis de cierre f (9

**RESPUESTA :**   
ERROR SEMANTICO: Una suma de un entero con una cadena.

**Respuesta:**

**Error Léxico: La constante 3.14.15 con dos puntos decimales**

**Esta respuesta es correcta.**

**Explicación:**

**Es un error léxico**

**Respuesta:**

Error Léxico: cadena no cerrada "hola mundo

Esta respuesta es correcta.

**Explicación:**

Es un error léxico

## **Función y asignación**

Si se declara la función mi\_funcion, que recibe dos parámetros enteros y uno real, pasados por valor, y que devuelve un valor lógico, ¿qué resultado dará el Procesador al analizar la sentencia de asignación mi\_funcion:= true;?

**Respuesta:**

Es un error semántico porque el uso que hay que hacer de un identificador que es una función es con los paréntesis y los parámetros

Esta respuesta es correcta.

**Explicación:**

Y, además, no se puede asignar un valor al nombre de una función

**Respuesta:** Un error semantico porque mi\_funcion no es variable de tipo lógico

**Explicacion**: No se puede asignar un valor al nombre de una funcion y eso lo detecta el Analizador Semantico.

**Respuesta:**

Es un error semántico porque el uso que hay que hacer de un identificador que es una función es con los paréntesis y los parámetros

**Explicación:**

Y, además, no se puede asignar un valor al nombre de una función

## **Semántica del operador condicional**

Has terminado la actividad:

Sean las siguientes reglas extraídas de una gramática de un lenguaje de programación que no tiene conversión automática de tipos, que exige la declaración previa de variables y que admite los tipos básicos de datos (entero, real y lógico). El operador condicional (? :) devuelve el valor de la segunda expresión si la primera expresión es cierta y el valor de la tercera expresión si la primera expresión es falsa. ¿Cuáles de las siguientes reglas semánticas son correctas en una Definición Dirigida por la Sintaxis de este lenguaje?

S → id = E // *asignación*

E → id

E → E1 ? E2 : E3

**Respuestas Seleccionadas:**

**1:** S→id=E | id.tipo:=buscaTipoTS(id.pos); S.tipo:= if (id.tipo=E.tipo) then tipoOK else tipoError

  Esta respuesta es correcta.

**Explicación:**

La asignación es correcta si ambos tipos son iguales

**Respuesta:**

S→id=E | S.tipo:= if (buscaTipoTS(id.pos)=E.tipo) then tipoOK else tipoError

**Explicación:**

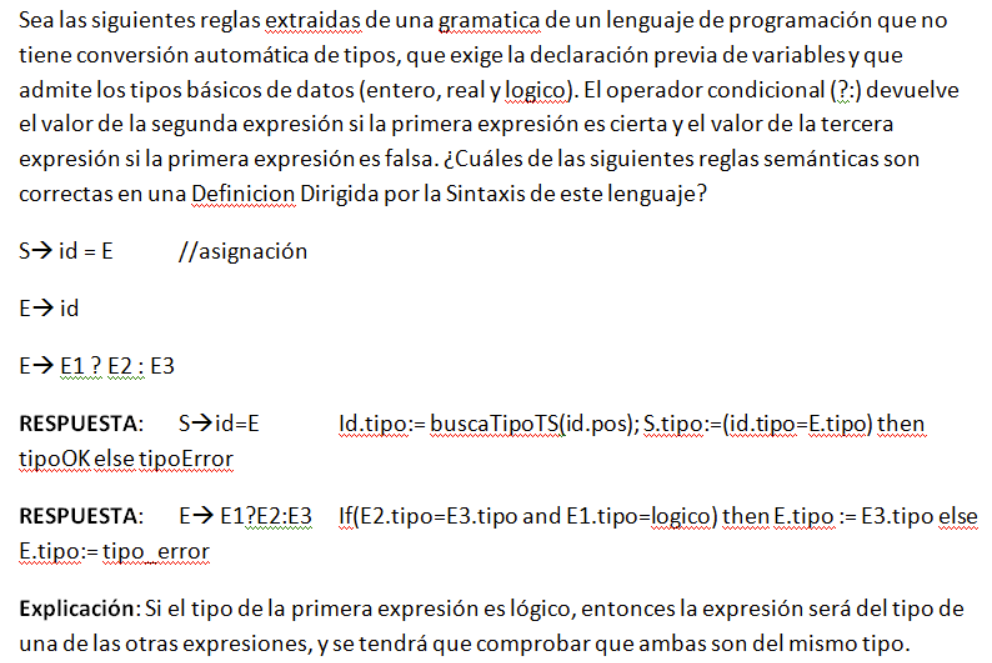
La asignación es correcta si ambos tipos son iguales

**Respuesta:**

E→E1?E2:E3 | E.tipo:= if (E1.tipo=lógico and E2.tipo=E3.tipo) then E2.tipo else tipo\_error

**Explicación:**

Si el tipo de la primera expresión es lógico, entonces la expresión será del tipo de una de las otras expresiones, y se tendrá que comprobar que ambas son del mismo tipo



**Respuesta:**

S→id=E | id.tipo:=buscaTipoTS(id.pos); S.tipo:= if (id.tipo=E.tipo) then tipoOK else tipoError

**1:** E→E1?E2:E3 | if (E2.tipo=E3.tipo and E1.tipo=lógico) then E.tipo:=E3.tipo else E.tipo:=tipo\_error

  Esta respuesta es correcta.

**Explicación:**

Si el tipo de la primera expresión es lógico, entonces la expresión será del tipo de una de las otras expresiones, y se tendrá que comprobar que ambas son del mismo tipo

**3:** E→id | E.tipo:= buscatipoTS(id.pos)

  Esta respuesta es correcta.

**1:** S→id=E | S.tipo:= if (buscaTipoTS(id.pos)=E.tipo) then tipoOK else tipoError

  Esta respuesta es correcta.

**Explicación:**

La asignación es correcta si ambos tipos son iguales

**2:** E→E1?E2:E3 | E.tipo:= if (E1.tipo=lógico and E2.tipo=E3.tipo) then E2.tipo else tipo\_error

  Esta respuesta es correcta.

**Explicación:**

Si el tipo de la primera expresión es lógico, entonces la expresión será del tipo de una de las otras expresiones, y se tendrá que comprobar que ambas son del mismo tipo

**3:** E→E1?E2:E3 | if (E2.tipo=E3.tipo and E1.tipo=lógico) then E.tipo:=E3.tipo else E.tipo:=tipo\_error

  Esta respuesta es correcta.

**Explicación:**

Si el tipo de la primera expresión es lógico, entonces la expresión será del tipo de una de las otras expresiones, y se tendrá que comprobar que ambas son del mismo tipo

## **Análisis Semántico: expresiones y operadores**

Has terminado la actividad:

Dadas las siguientes reglas pertenecientes a la gramática de un lenguaje de programación que no tiene conversión automática de tipos, que exige la declaración previa de variables y que admite como tipos básicos de datos integer, float, boolean y string), indicar cuáles de las siguientes partes de un Esquema de Traducción serían correctas:

E → E + T | T // + Se aplica sobre datos numéricos

T → T \* F | F // \* Se aplica sobre datos numéricos

F → F < G | F > G | G // <, > Se aplican sobre datos numéricos

G → G AND H | H // AND Se aplica sobre datos lógicos

H → id | cte\_entera | cte\_real | true | cadena

**Respuestas Seleccionadas:**

**1:** H→cte\_real {H.tipo:=cte\_real.tipo}

  Esta respuesta es incorrecta.

**Explicación:**

El *token* de cte\_real no lleva el atributo .tipo sino .valor. El tipo de esta expresión será siempre real

**2:** H→cadena {H.tipo:=cadena}

  Esta respuesta es correcta.

**Explicación:**

Las constantes cadena son por definición cadenas

**3:** E→E1 + T {IF E1.tipo=entero and T.tipo=entero THEN E.tipo:=entero ELSE IF E1.tipo=real or T.tipo=real THEN E.tipo:=real ELSE E.tipo:=tipo\_error}

  Esta respuesta es incorrecta.

**Explicación:**

El lenguaje no tiene conversión de tipos, por lo que no se pueden operar datos de distintos tipos entre sí

**1:** G→G1 AND H {IF G1.tipo=H.tipo and G1.tipo=lógico THEN G.tipo:=lógico ELSE G.tipo:=tipo\_error}

  Esta respuesta es correcta.

**Explicación:**

Los operadores lógicos se aplican sobre datos de tipo lógico y devuelven un valor de tipo lógico

**2:** H→cadena {H.tipo:=cadena}

  Esta respuesta es correcta.

**Explicación:**

Las constantes cadena son por definición cadena

## **Semántica de la sentencia de selección múltiple**

Has terminado la actividad:

Sean las siguientes reglas pertenecientes a la gramática de un fragmento de un lenguaje de programación:

S → Switch E C otherwise S // E admite cualquier expresión. S define sentencias

C → Case E (S) C // Define los diferentes casos que pueden ocurrir

C → λ

Este lenguaje no tiene conversión automática de tipos, exige la declaración previa de variables y admite entero, real y lógico como tipos básicos de datos. La expresión E del Switch puede ser entera o lógica. Las expresiones E de los casos C deben ser todas del mismo tipo para un mismo Switch, y este tipo debe coincidir con el de la expresión del Switch. Asumiendo que la estrategia a seguir es recoger el tipo de las expresiones en cada cláusula Case, e ir subiéndolo por el árbol sintáctico hasta comprobar que coincide con el tipo de la expresión del Switch, tratando de detectar errores lo antes posible, ¿cuál de las siguientes reglas semánticas de un Esquema de Traducción es correcta?:

**Respuesta:**

C→Case E (S) C1 {If (E.tipo=C1.tipo Or C1.tipo=vacío) And E.tipo∈{entero,lógico} And S.tipo=tipo\_ok Then C.tipo:=E.tipo Else C.tipo:=tipo\_error (“Tipo incorrecto en clausula Case”)}

Esta respuesta es correcta.

**Explicación:**

Se comprueba que los tipos de E y los demás C1 coincidan, excepto si estamos examinando la última cláusula Case y se comprueba que el tipo de E es válido

**Respuesta:**

S→Switch E C otherwise S1 {If (E.tipo=C.tipo Or C.tipo=vacio) And E.tipo∈{entero,lógico} Then S.tipo:=S1.tipo Else S.tipo:=tipo\_error(“tipos erróneos en Switch”)}

Esta respuesta es correcta.

**Explicación:**

Se comprueba que el tipo de la Expresión coincide con el tipo común de las cláusulas Case (si hay alguna). La comprobación adicional que se trata de uno de los tipos válidos puede ser redundante si ya se hizo en las cláusulas Case

## **Reto Semántico - Inicialización**

Has terminado la actividad:

Se tiene la siguiente gramática:

1. D → var T L I
2. T → integer
3. T → boolean
4. L → id R
5. R → , id R
6. R → λ
7. I → = V
8. I → λ
9. V → C X
10. X → , C X
11. X → λ
12. C → cte\_int
13. C → true
14. C → false

Como se puede observar, la gramática genera una declaración de variables con la siguiente sintaxis: se declaran una o más variables, de un tipo (entero o lógico) y, opcionalmente, se pueden inicializar cada una de esas variables con una constante. El número de constantes debe coincidir con el número de variables. El analizador semántico se ha de encargar de insertar el tipo de las variables en la tabla de símbolos y sus desplazamientos (se quiere hacer heredando información por el árbol), y deberá comprobar que, en los casos en que se realice inicialización, se estén usando el número y tipo de valores correctos (es decir, una constante por cada variable, y del mismo tipo que la variable). Teniendo como planteamiento de base que las comprobaciones sobre la inicialización se quieren realizar obligatoriamente en la regla 1, ¿cuál de las siguientes afirmaciones es correcta para el diseño, bajo estas condiciones?

**Respuesta:**

La variable despl se inicializará a cero en la regla 1, antes del símbolo L

.**Explicación:**

Debe inicializarse la variable desplazamiento antes de que empiece la declaración de las variables, no después

**Respuesta:**

La acción semántica {L.tipo:= T.tipo} estaría en la regla 1

**Explicación:**

El tipo de L tiene que heredar del tipo de T

**La respuesta correcta era:**

**Respuesta:**

La acción semántica {R.tipo:= R1.tipo} no es una acción semántica correcta en este diseño

**Explicación:**

Efectivamente, no encaja en ninguna regla pues no se sintetiza el tipo de R en ningún momento

**Respuesta:**

La acción semántica {InsertatipoTS(R.pos, R1.tipo} no es una acción semántica correcta en este diseño

**Explicación:**

No tiene sentido utilizar el atributo posición en R

**Respuesta:**

La acción semántica de la regla 12 sería {C.tipo:= entero}

**Explicación:**

El tipo de la constante será siempre entero

## **Análisis Semántico de Sentencias**

Indicar cuáles de las siguientes reglas semánticas son correctas, para un lenguaje con las siguientes características y el fragmento de gramática dado:

* · el lenguaje exige declaración previa de variables
* · tiene los tipos entero, real y lógico
* · dispone de conversión automática de tipos entre enteros y reales y viceversa
* · la sentencia continue se utiliza para detener la iteración de un bucle for
* · el bucle for termina cuando el valor del identificador id (que debe ser entero y empieza tomando el valor de la primera expresión) es igual o mayor que la segunda expresión
* · la condición del while debe ser lógica
* · las expresiones (E), que no se encuentran detalladas en este fragmento de gramática, pueden ser de cualquiera de los tipos del lenguaje

S → while E do S

S → S ; S

S → for id = E to E do S

S → continue

## 

**Respuestas Seleccionadas:**

**1:** S → continue {S.tipo:= if S.for then tipo\_ok else tipo\_error}

  Esta respuesta es correcta.

**Explicación:**

La sentencia continue es correcta solo si aparece dentro de un bucle for, y eso es justamente lo que indica el atributo S.for

**Además, te ha faltado por seleccionar:**

**Respuesta:**

S → {S1.for:= S.for} S1 ; {S2.for:= S.for} S2 {S.tipo:= if S1.tipo = tipo\_ok then S2.tipo else tipo\_error;}

**Respuesta:**

S → for id = E1 to E2 do {S1.for:=true} S1{S.tipo:=if BuscaTipoTS(id.entrada) = entero AND E1.tipo pertenece {entero, real} AND E2.tipo pertenece {entero, real} AND S1.tipo = tipo\_ok then tipo ok else tipo\_error)}  
  
RESPUESTA:

S → while E do {S1.for:= S.for} S1 {S.tipo:= if E.tipo = lógico then S1.tipo else tipo\_error}

  Esta respuesta es correcta.

**Explicación:**

Se utilizan dos atributos para S: tipo almacena el tipo de la sentencia (correcta o errónea) y for almacena un valor lógico heredado que indica si la sentencia está o no dentro de un bucle for}}

**1:** S → while E do {S1.for:= S.for} S1 {S.tipo:= if E.tipo = lógico AND S1.tipo = tipo\_ok then tipo\_ok else tipo\_error}

  Esta respuesta es correcta.

**Explicación:**

Se utilizan dos atributos para S: tipo almacena el tipo de la sentencia (correcta o errónea) y for almacena un valor lógico heredado que indica si la sentencia está o no dentro de un bucle for

**Además, te ha faltado por seleccionar:**

**Respuesta:**

S → for id = E1 to E2 do {S1.for:= true} S1 {S.tipo:= if BuscaTipoTS(id.entrada) = entero AND E1.tipo ∈ {entero, real} AND E2.tipo ∈ {entero, real} then S1.tipo else tipo\_error)}

**Explicación:**

Se utilizan dos atributos para S: tipo (sintetizado) almacena el tipo de la sentencia (correcta o errónea) teniendo en cuenta que hay conversión de tipos, y for (heredado) almacena un valor lógico que indica si la sentencia está o no dentro de un bucle for

## **Análisis Semántico: expresiones y operadores**

Dadas las siguientes reglas pertenecientes a la gramática de un lenguaje de programación que no tiene conversión automática de tipos, que exige la declaración previa de variables y que admite como tipos básicos de datos integer, float, boolean y string), indicar cuáles de las siguientes partes de un Esquema de Traducción serían correctas:

E → E + T | T // + Se aplica sobre datos numéricos

T → T \* F | F // \* Se aplica sobre datos numéricos

F → F < G | F > G | G // <, > Se aplican sobre datos numéricos

G → G AND H | H // AND Se aplica sobre datos lógicos

H → id | cte\_entera | cte\_real | true | cadena

**1:** G→H {G.tipo:=H.tipo}

  Esta respuesta es correcta.

**Explicación:**

Solo es necesario pasar el tipo proveniente de operaciones tratadas en otras reglas

**2:** H→cte\_entera {H.tipo:=entero}

  Esta respuesta es correcta.

**Explicación:**

Las constantes enteras son por definición enteras

**3:** G→G1 AND H {IF G1.tipo=H.tipo and G1.tipo=lógico THEN G.tipo:=lógico ELSE G.tipo:=tipo\_error}

  Esta respuesta es correcta.

**Explicación:**

Los operadores lógicos se aplican sobre datos de tipo lógico y devuelven un valor de tipo lógico

**Además, te ha faltado por seleccionar:**

**Respuesta:**

T→T1 \* F {IF T1.tipo=F.tipo and F.tipo∈(entero, real) THEN T.tipo:=F.tipo ELSE T.tipo:=tipo\_error}

**Explicación:**

Si los operandos son del mismo tipo y numéricos, la operación de producto devuelve el tipo de los operandos, sino devuelve error

**Respuesta:**

F→F > G {IF F1.tipo=G.tipo and F1.tipo∈(entero, real) THEN F.tipo:=lógico ELSE F.tipo:=tipo\_error}

**Explicación:**

Los operadores relacionales se aplican sobre datos numéricos y devuelven un valor de tipo lógico

**Respuesta:**

E→T {E.tipo:=T.tipo}

**Explicación:**

Solo es necesario pasar el tipo proveniente de operaciones tratadas en otras reglas

**1:**H→cte\_entera {H.tipo:=entero}

  Esta respuesta es correcta.

**Explicación:**

Las constantes enteras son por definición enteras

**2:**E→T {E.tipo:=T.tipo}

  Esta respuesta es correcta.

**Explicación:**

Solo es necesario pasar el tipo proveniente de operaciones tratadas en otras reglas

**3:**H→true {H.tipo:=lógico}

  Esta respuesta es correcta.

**Explicación:**

Las constantes lógicas son por definición lógicas

**4:**E→E1+ T {IF E1.tipo=T.tipo and T.tipo∈(entero, real) THEN E.tipo:=T.tipo ELSE E.tipo:=tipo\_error}

  Esta respuesta es correcta.

**Explicación:**

Si los operandos son del mismo tipo y numéricos, la operación + devuelve el tipo de los operandos, sino devuelve error

**5:**E→E1+ T {IF E1.tipo=T.tipo and E1.tipo∈(entero, real) THEN E.tipo:=E1.tipo ELSE E.tipo:=tipo\_error}

  Esta respuesta es correcta.

**Explicación:**

Si los operandos son del mismo tipo y numéricos, la operación + devuelve el tipo de los operandos, sino devuelve error

H→id {H.tipo:=BuscaTipoTS (id.ent)}

  Esta respuesta es correcta.

**Explicación:**

Se busca en la Tabla de Símbolos el tipo de la variable

H→cte\_real {H.tipo:=real}

  Esta respuesta es correcta.

**Explicación:**

Las constantes reales son por definición reales

Respuesta:

F→F < G {IF F1.tipo=G.tipo and F1.tipo∈(entero, real) THEN F.tipo:=logico ELSE F.tipo:=tipo\_error}

  Esta respuesta es correcta.

**3:** T→T1 \* F {IF T1.tipo=F.tipo and F.tipo∈(entero, real) THEN T.tipo:=F.tipo ELSE T.tipo:=tipo\_error}

  Esta respuesta es correcta.

**Explicación:**

Si los operandos son del mismo tipo y numéricos, la operación de producto devuelve el tipo de los operandos, sino devuelve error

## **Errores semánticos**

Has terminado la actividad:

¿Cuál de los siguientes errores es de tipo Semántico?

**Respuesta:**

Usar una variable sin declarar, cuando el lenguaje exige declaración previa de variables

**Respuesta:**

Tener declarada dos veces la misma variable en el mismo ámbito con dos tipos distintos

Esta respuesta es correcta.

**Respuesta:**

Llamar a una función con un número de parámetros erróneo

Esta respuesta es correcta.

**Explicación:**

Es un error semántico

**Explicación:**

Es un error semántico

## **TDS**

Has terminado la actividad:

En relación con el tema que estamos estudiando actualmente, ¿cuáles de las siguientes afirmaciones son correctas?

**Respuestas Seleccionadas:**

**1:**DDS significa Diseño Dirigido por la Sintaxis

  Esta respuesta es incorrecta.

**Explicación:**

DDS significa Definición Dirigida por la Sintaxis

**2:**Los atributos heredados son aquéllos cuyo valor en un nodo del árbol de análisis sintáctico se obtiene a partir de los valores de los atributos de los hermanos o padre de dicho nodo

  Esta respuesta es correcta.

**Explicación:**

La información fluye hacia abajo u horizontalmente en el árbol

**3:**Los EdT son especificaciones de un relativo bajo nivel que permiten definir una TDS

  Esta respuesta es correcta.

**Explicación:**

Los Esquemas de Traducción son una notación de bajo nivel que especifican el funcionamiento de una Traducción Dirigida por la Sintaxis

**Además, te ha faltado por seleccionar:**

**Respuesta:**

Un EdT consiste en una gramática de tipo 2 que tiene acciones semánticas intercaladas entre los símbolos de las reglas

**Explicación:**

Un Esquema de Traducción es una gramática de tipo 2 con acciones semánticas intercaladas entre los símbolos de las reglas