1ª Guía Compiladores

Nombre: Diaz Medina Jesús Kaimorts
Grupo: 3CM Fecha: 17 de septiembre del 2018

Defina compilador.

- Es un programa que lee un programa escrito en lenguaje fuente, y lo traduce a un programa equivalente en un lenguaje objeto.

Cuáles son las dos partes de la compilación

1. Análisis: *Divide al programa fuente en sus elementos, componentes* y *crea una representación intermedia* del programa fuente.

2. Síntesis: *Construye el programa objeto* deseado a *partir de la representación intermedia*.

Describa las 6 fases de un compilador.

Análisis Léxico o Análisis Lineal	• En este el programa fuente se lee de izquierda a derecha y se agrupa en **componentes léxicos,** que son secuencias de caracteres con significado colectivo. A la secuencia de caracteres que forman a un componente léxico se denomina **lexema.** • En esta etapa se va a "cortar" la cadena de entrada en cuanto vea un token. Es decir, divide la cadena de entrada en token (terminales)
Análisis Sintáctico	• Implica agrupara jerárquicamente los componentes léxicos del programa fuente en frases gramaticales, que el compilador usa para sintetizar la salida. Por lo general las frases gramaticales del programa fuente se representan mediante un árbol de análisis sintáctico. • Se encarga de **verificar el orden de los tokens**. A. Si no viene "en cierto orden", se debe marcar error. • Se encarga de **crear el A.A.S (Árbol de Análisis Sintáctico)** de forma implícita o explicita. • Además, verifica si la cadena ∈ (pertenece) al lenguaje generado por la gramática.
Análisis Semántico	• Revisa el programa fuente para tratar de encontrar errores semánticos y reúne información sobre los tipos para la fase posterior de **generación de código.** • Tiene que ver con el significado. • Se da cuenta que **los operadores correspondan con los operadores de expresiones y proposiciones.**
Generador de código intermedio	• Generan una representación intermedia explícita del programa fuente. Esta representación es como un programa para una máquina abstracta, la cual tiene dos propiedades importantes: ser fácil de producir y fácil de traducir al programa objeto. • También es conocido como ***"código de 3 direcciones"*** porque puede tener **máximo dos operadores y tres operandos.** • Uno de los operadores es el de asignación; ":= "

Optimizador de código	• Trata de **mejorar el código intermedio**, de modo que resulte **un código máquina más rápido de ejecutar**. • Se encarga de reducir tiempo y espacio. Reducir instrucciones.
Generación de código	• **Genera el código objeto**: Un código máquina relocalizable o código ensamblador. • Las posiciones de memoria se seleccionan para cada una de las variables usadas por el programa. • Las instrucciones intermedias se traducen a la secuencia de instrucciones máquina, que ejecuta la misma tarea. • **Aspecto decisivo: asignación de variables a registros.**

Cuáles son los 8 módulos de un compilador

TABLA DE SÍMBOLOS	MANEJO DE ERRORES

- **Analizador Léxico.**
- **Analizador Sintáctico.**
- **Analizador Semántico.**
- **Generación de código intermedio.**
- **Optimización de código.**
- **Generación de código.**

En cada uno de los módulos aparecen tanto la tabla de símbolos como el manejo de errores.

A partir de hoc4 se usan dos etapas en hoc. ¿Cuáles son y que hacen?

1.-

2.-

Para que sirve el Análisis Léxico

a) Para generar el código en lenguaje objeto b) Nos dice si una cadena pertenece al lenguaje generado por una gramática (C)

c) Para dividir una cadena en tokens d) Los compiladores no lo necesitan nunca

El _____ comprueba que el orden en que el analizador léxico le va entregando los tokens es válido.

a) analizador semántico b) analizador sintáctico c) optimizador d) generador de código (B)

Es una *gramática que* tiene cuatro componentes:

1. Un conjunto de componentes léxicos.

2. Un conjunto de no terminales.

3. Un conjunto de producciones, en el que cada producción consta de un no terminal, llamado *lado izquierdo* de la producción, una flecha y una secuencia de componentes léxicos y no terminales, o ambos, llamado *lado derecho* de la producción.

4.. La denominación de uno de los no terminales como símbolo *inicial.*

a) Gramática Asociativa por la izquierda *b) Gramática recursiva* (C)

c) Gramática libre de contexto d) Gramática *ambigua*

¿Cuál de las sigs. opciones no es sinónimo de las otras?

a) Componente léxico b) no terminal c) token d) Símbolo gramatical (B)

Es una gramática donde existe una cadena que tiene más de un árbol de análisis sintáctico.

a) Gramática recursiva por la izquierda *b) Gramática recursiva* (D)

c) Gramática libre de contexto d) Gramática *ambigua*

Si Una gramática contiene una regla de producción de la forma A → A α entonces es una

a) Gramática recursiva por la izquierda *b) Gramática ambigua* (A)

c) Gramática libre de contexto d) ninguna de las anteriores

Falso o verdadero (F/V)

1-Análisis sintáctico ***descendente*** *es donde la construcción del árbol de análisis sintáctico se inicia en las hojas y avanza hacia la raíz* _F_.

2-Análisis sintáctico **a*scendente*** *es donde la construcción del árbol de análisis sintáctico se inicia en las hojas y avanza hacia la raíz* __V__.

3-Las variables en HOC son de tipo entero ___F_____.

4.-yylex llama a yyparse ___F_____.

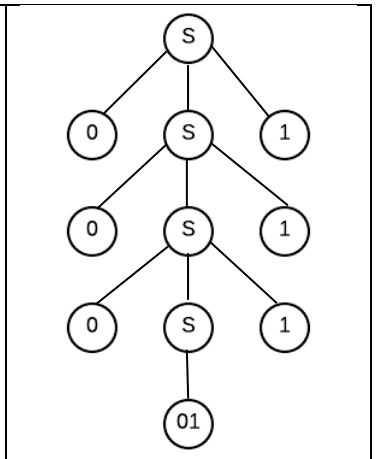
Considere la siguiente gramática

*S-> **0** S **1** | **01***

a) Mostrar una derivación de ***00001111***

b) Dibuje el árbol de análisis sintáctico para la entrada ***00001111***

$S \rightarrow 0S1$
$S \rightarrow 00S11$
$S \rightarrow 000S111$
$S \rightarrow 00001111$



Considere la siguiente gramática

S → **b**A

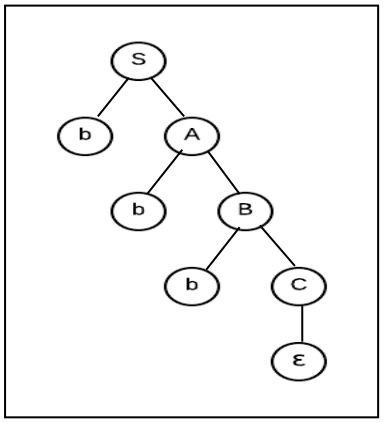
A → **b**B

B → **b**C

C → **ε**

a) Mostrar una derivación de **bbb**

b) Dibuje el árbol de análisis sintáctico para la entrada **bbb**



a)

$S \rightarrow bA$
$S \rightarrow bbB$
$S \rightarrow bbbC$
$S \rightarrow bbb$

Considere la siguiente gramática
$S \rightarrow A$
$A \rightarrow A+A \mid B++$
$B \rightarrow y$
a) Mostrar una derivación de **y + + + y + +**
b) Dibuje el árbol de análisis sintáctico para la entrada **y + + + y + +**

Considere la siguiente gramática
$l \rightarrow l, d \mid d$
$d \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

a) Mostrar una derivación de ***9,8,7,6,5,4,3,2,1,0***
b) Dibuje el árbol de análisis sintáctico para la entrada ***9,8,7,6,5,4,3,2,1,0***

Dada la gramática
T={**a**, **b**, +, -, *, /, (,)}, N={E, T, F} S={E}
P={ E->T | E+T | E-T
T->F | T*F | T/F
F-> **a** | **b** | (E) }
y la cadena **(a+b)/b**
a) Obtenga una derivación de dicha cadena
b) Dibuje el árbol de análisis sintáctico que corresponde a la cadena mencionada

Análisis sintáctico predictivo descendente recursivo
Considere la siguiente gramática
S →**a** | **(** S **)**
Escriba el analizador sintáctico predictivo descendente recursivo

Ambigüedad
Demostrar que la siguiente gramática es ambigua

A →A **x** B | **x**
B → **x** B | **x**

usando la cadena **xxxxx**

Demostrar que la siguiente gramática es ambigua

S→ **a** S **b** S | **b** S **a** S | **ε**

usando la cadena **abab**

Verificar si las siguientes gramáticas son ambiguas
S → S + S | S - S | **a**
S → S S + | S S - | **a**

Recursividad por la **izquierda**
Para eliminar la recursividad por la izquierda
A → A**a** | **b**

se transforma en
A → **b** | **b**R

R → **a**R | **ε**

Ahora considere la siguiente gramática

S→ **(** L **)** | **a**

L→ L**,** S | S

Elimine la recursividad por la izquierda de dicha gramática.
Escriba el analizador sintáctico predictivo descendente recursivo
Escriba la sección de reglas de la especificación de YACC para dicha gramática

Definiciones dirigidas por la sintaxis

PRODUCCIÓN	REGLA SEMÁNTICA
sec ⟶ **comienza**	*sec.x = 0* *sec.y = 0*
sec ⟶ *sec₁ instr*	*sec.x = sec₁.x + instr.dx* *sec.y = sec₁.y + instr.dy*
instr ⟶ **este**	*instr.dx = 1* *instr.dy = 0*
instr ⟶ **norte**	*instr.dx = 0* *instr.dy = 1*
instr ⟶ **oeste**	*instr.dx = -1* *instr.dy = 0*
instr ⟶ **sur**	*instr.dx = 0* *instr.dy = -1*

Dibuje el árbol de análisis sintáctico con anotaciones para la sig cadena
c n e ss oo nnn eee ssss oooo

Escribir la sección de reglas de la especificación de yacc para calcular la posición final del robot.

```
%{
    Struct cord{
        Int x, y,dx,dy;
    }
    ;
    Typedef struct cord cordenada;
    define struct cord cordenada
    #define YYSTYPE struct cord
 %}

%token comienza este oeste norte sur
%%
    sec:    comienza {$$.x = 0; $$.y = 0;}
       |    sec instr {$$.x = $1.x + $2.dx;   $$.y = $1.y +  $2.dy;}

    instr:  este {$1.dx = 1; $1.dy = 0;}
       |   oeste{$1.dx = -1; $1.dy = 0;}
       |    norte{$1.dx = 0; $1.dy = 1; }
       |    sur{$1.dx = 0; $1.dy = -1;}
       ;
%%
```

Escriba una definición dirigida por la sintaxis para evaluar expresiones booleanas.

PRODUCCIÓN	REGLA SEMÁNTICA
expr → **expr1 '|' expr2**	*expr.t = expr1.t || expr2.t*
expr → **expr1 '&' expr2**	*expr.t = expr1.t && expr2.t*
expr → **'!' expr1**	*expr.t = !expr1.t*
expr → **termino**	*expr.t = termino.t*

Esquemas de traducción

Escriba un esquema de traducción para convertir una expresión en:

infijo a postfijo

```
expr + termino { print (' + ')}
expr -> expr + termino { print (' - ')}
expr -> termino
termino -> 0          { print (' 0 ')}
termino -> 1          { print (' 1')}
termino -> 9          { print (' 9')}
```

postfijo a infijo

```
expr  termino +   { print (' + ');}
expr ->-  expr  termino  -   { print (' - ');}
expr -> termino
termino -> 0          { print (' 0 ')}
termino -> 1          { print (' 1')}
termino -> 9          { print (' 9')}
```

infijo a prefijo

```
expr  termino +   { print ('+'expr , termino}
expr -> expr  termino  -   { print ('-' exp, termino) }
expr -> termino
termino -> 0          { print (' 0 ')}
termino -> 1          { print (' 1')}
termino -> 9          { print (' 9')}
```

prefijo a infijo

```
expr  termino    { print (expr , '+', termino}
expr -> - expr  termino     { print (exp, '-', termino) }
expr -> termino
termino -> 0          { print (' 0 ')}
termino -> 1          { print (' 1')}
termino -> 9          { print (' 9')}
```

Escriba un esquema de traducción para evaluar expresiones booleanas

Para cada esquema de traducción de arriba escriba la sección de reglas de la especificación de YACC

Escritura de Gramaticas

Escribir una gramática que genere todas las cadenas de longitud 4 formadas con los símbolos del alfabeto {a,b,c}

```
T = {a,b,c}
N = {A,S}
S = {S}
P = { S-> AAAA
A-> a|b|c}
```

Escribir una gramática que sirva para generar las siguientes cadenas

Especie perro	**Especie gato**	**Especie perro**	**Especie gato**
Edad 1	**Edad 2**	**Edad 2**	**Edad 2**
Sexo macho	**Sexo macho**	**Sexo hembra**	**Sexo macho**
Tamaño grande	**Tamaño mediano**	**Tamaño pequeño**	**Tamaño grande**
Colores negro , blanco	**Colores negro , blanco , café**	**Colores canela , gris**	**Colores blanco**
Soy rápido , activo, alegre	**Soy tranquilo , sociable**	**Soy fuerte , alegre, activo.**	**Soy listo , obediente**
Aficiones correr, comer	**Aficiones dormir, parrandear, comer**	**Aficiones aullar**	**Aficiones jugar, haraganear**

12.-Escribir una gramática que sirva para generar las siguientes cadenas

Etiquetado Nerd Nivel Junior Sexo Hombre Lenguajes Java , C , Logo Aficiones programar, videogames, comics, hackear, googlear	Etiquetado Geek Nivel Senior Sexo Mujer Lenguajes Pascal , Prolog , SQL Aficiones chatear, videogames, programar	Etiquetado Nerd Nivel Junior Sexo Mujer Lenguajes PHP , Perl, Java Aficiones hackear, googlear, gotcha, dormir	Etiquetado Freak Nivel Senior Sexo Hombre Lenguajes Ensamblador, C Aficiones gotcha, dormir, chatear, comics

YACC

1.-Los %% se usan para indicar

a) inicio de la sección de declaraciones b) inicio de la sección de reglas (b)
c)precedencia de los operadores d)fin del código de soporte

2.-%token sirve para indicar

a)inicio de la sección de declaraciones d)los no terminales de la gramática (d)
c)precedencia de los operadores d)los terminales de la gramática

.-$$ sirve para indicar:

.-$n sirve para indicar:

3.-Como le indica el analizador léxico (yylex) al analizador sintáctico (yyparse) que ya no hay mas tokens en la entrada

a) retornando cero b) retornando -1 (a)
c) almacenando -1 en yylval d) almacenando 0 en yylval

4.-Una acción gramatical debe ir entre

a) comillas b) paréntesis c) corchetes d) llaves (d)

5.-Considere la producción

S : S ' a' S 'b'

$4 a cual de los miembros del lado derecho de la producción se refiere?

a) la 'a' b) la 1er S (d)
c)la segunda S d) la 'b'

Si el codigo de yylex es el siguiente

```
int yylex() { return getchar(); }
```

de cuantos caracteres son los tokens

a) 0 b) 1 c) 2 d) la cantidad de caracteres del token varia (b)

Considere la siguiente gramática (los terminales se indican en negritas)

L-> L**,** D | D
D-> **0** | **1**

Escriba la sección de reglas de la especificación de yacc para dicha gramática

```
X||
%%
L:  L ',' D
|   D
;
```

```
D:  0
| 1
;
%%
```

Escriba la especificación de yacc para la gramática

S → U | V
U → T**a**U | T**a**T
V → T**b**V | T**b**T
T → **a**T**b**T | **b**T**a**T | **ε**

Escriba las acciones gramaticales para que imprima el numero de b's en la cadena de entrada

```
%%
S:  U
 |   V
 ;
U:  T 'a' U
 |  T 'a' T
 ;
V:  T 'b' V
 |  T 'b' T
 ;
 T  /*nada*/
 |'a' T 'b' T
 |  'b' T 'a' T
 ;
%%
```

```
%{
/*escriba el tipo de los elementos en la pila de yacc */
#define YYSTYPE
%}

%%
S : '(' B ')' { $$ = $$2; }
;
B : '(' B ')' { $$ = $$2; }
  | D { $$=$1; }
;
D : { }
  | 'b' D { $$.numb ++; $$=$2; }
;
%%
```

Considere la siguiente gramática (los terminales se indican en negritas)
lista->lista **,** figura | figura
figura-> triangulo | cuadrilatero
triangulo-> **lado lado lado**
cuadrilatero-> **lado lado lado lado**

Escriba la sección de reglas de la especificación de yacc para dicha gramática y las acciones semánticas respectivas para que se imprima si un triangulo es equilátero y si un cuadrilátero es un cuadrado

```
%%
lista:  lista ',' figura
     |  figura
     ;
figura: triangulo
     |  cuadrilátero
     ;
triangulo:  lado lado lado {if($1==$2 && $2==$3) printf("Equilatero");}
     ;
cuadrilátero:  lado lado lado lado {if($1 == $2 && $2 == $3 && $3 == $4)
printf("Cuadrialtero");}
     ;
%%
```

Análisis Sintáctico Predictivo no Recursivo

-Para las siguientes GLC construya la tabla Análisis Sintáctico Predictivo no Recursivo
-Use dicho análisis para analizar las cadenas propuestas:
-Muestre el contenido de la pila, la entrada y la acción a realizar

Problema 1.-Considere la gramática para generar paréntesis anidados

1) A → **(A)**	2) A → **a**

Cadenas propuestas:
(a)
((a))
(((a)))
((((a))))

➔ Tabla de Análisis Sintáctico Predictivo No Recursivo

	(	)	a	$
A	**A → (A)**		**A → a**	

➔ (a)

Pila	Entrada	Acción
$A	(a)$	**A → (A)**
$)A(	(a)$	
$)A	a)$	**A → a**
$)a	a)$	
$)	)$	
$	$	

➔ ((a))

Pila	Entrada	Acción
$A	((a))$	**A → (A)**
$)A(	((a))$	
$)A	(a))$	**A → (A)**
$))A(	(a))$	
$))A	a))$	**A → a**
$))a	a))$	
$))	))$	
$)	)$	
$	$	

➔ (((a)))

Pila	Entrada	Acción
$A	(((a)))$	**A → (A)**
$)A(	(((a)))$	
$)A	((a)))$	**A → (A)**
$))A(	((a)))$	
$))A	(a)))$	**A → (A)**
$)))A(	(a)))$	
$)))A	a)))$	**A → a**
$)))a	a)))$	
$)))	)))$	
$))	))$	
$)	)$	
$	$	

Problema 2.-Considere la siguiente gramática :

1) S → **a**	2) S → **(** S R	3) R → **,** S R	4) R → **)**

Cadenas propuestas:
(a)
(a , a)
(a , a, a)
(a , a, a, a)

➔ Tabla de análisis Sintáctico Predictivo No Recursivo

	a	(	,	)	$
S	**S → a**	**S → (S R**			
R			**R → , S R**	**R →)**	

➔ (a)

Pila	Entrada	Acción
$S	(a)$	**S → (S R**
$RS(	(a)$	
$RS	a)$	**S → a**
$Ra	a)$	
$R	)$	**R →)**
$)	)$	
$	$	

➔ (a,a)

Pila	Entrada	Acción
$S	(a,a)$	**S → (S R**
$RS(	(a,a)$	
$RS	a,a)$	**S → a**
$Ra	a,a)$	
$R	,a)$	**R → , S R**
$RS,	,a)$	
$RS	a)$	**S → a**
$Ra	a)$	
$R	)$	**R →)**
$)	)$	
$	$	

➔ (a,a,a)

Pila	Entrada	Acción
$S	(a,a,a)$	**S → (S R**
$RS(	(a,a,a)$	
$RS	a,a,a)$	**S → a**
$Ra	a,a,a)$	
$R	,a,a)$	**R → , S R**
$RS,	,a,a)$	
$RS	a,a)$	**S → a**
$Ra	a,a)$	
$R	,a)$	**R → , S R**
$RS,	,a)$	
$RS	a)$	**S → a**
$Ra	a)$	
$R	)$	**R →)**
$)	)$	
$	$	

➔ (a, a, a, a)

Pila	Entrada	Acción
$S	(a,a,a,a)$	**S → (S R**
$RS(	(a,a,a,a)$	
$RS	a,a,a,a)$	**S → a**
$Ra	a,a,a,a)$	
$R	,a,a,a)$	**R → , S R**
$RS,	,a,a,a)$	
$RS	a,a,a)$	**S → a**
$Ra	a,a,a)$	
$R	,a,a)$	**R → , S R**
$RS,	,a,a)$	
$RS	a,a)$	**S → a**
$Ra	a,a)$	
$R	,a)$	**R → , S R**
$RS,	,a)$	
$RS	a)$	**S → a**
$Ra	a)$	
$R	)$	**R →)**
$)	)$	

Problema 3.-Considere la siguiente gramática :

1) S → A**a**A**b**	2) S → B**b**B**a**	3) A → **ε**	4) B → **ε**

Cadenas propuestas:
ab y **ba**

Problema 4.-Considere la siguiente gramática :
S → A
A → **ε**
A → **bb**A

Cadena propuesta:
bbbb