# Taller de Compiladores

*Ing. Edgar Oswaldo Hernández Barajas*
*UDN*

# Descripción general de un compilador

# Funciones Principales

- *Un compilador a grandes rasgos  es un programa (en cualesquiera lenguaje) que lee un programa escrito en un lenguaje(JAVA, C, C++, FORTRAN, .NET, etc.), el lenguaje fuente, y lo traduce a un programa equivalente en otro lenguaje, el lenguaje objeto. Como parte importante de este proceso de traducción, el compilador informa a su usuario de la presencia de errores en el programa fuente*

# Funciones Principales

- *A primera vista, la diversidad de compiladores puede parecer abrumadora. Hay miles de lenguajes fuente, desde los lenguajes de programación tradicionales, como C o JAVA, hasta los lenguajes especializados que han surgido virtualmente en todas las áreas de aplicación de la informática.*
- *Los lenguajes objeto son igualmente variados; un lenguaje objeto puede ser otro lenguaje de programación o el lenguaje de máquina de cualquier computador entre un microprocesador y un supercomputador.*

# Funciones Principales

- *A pesar de existir una aparente complejidad por la clasificación de los compiladores, como se vio en el tema anterior, las tareas básicas que debe realizar cualquier compilador son esencialmente las mismas.*
- *Al comprender tales tareas, se pueden construir compiladores para una gran diversidad de lenguajes fuente y máquinas objeto utilizando las mismas técnicas básicas.*

# Funciones Principales

- *Nuestro conocimiento sobre cómo organizar y escribir compiladores ha aumentado mucho desde que comenzaron a aparecer los primeros compiladores a principios de los años cincuenta.*
- *Es difícil dar una fecha exacta de la aparición del primer compilador, porque en un principio gran parte del trabajo de experimentación y aplicación se realizó de manera independiente por varios grupos.*

# Funciones Principales

- *Gran parte de los primeros trabajos de compilación estaba relacionada con la traducción de fórmulas aritméticas a código de máquina.*
- *En la década de 1950, se consideró a los compiladores como programas notablemente difíciles de escribir.*
- *EL primer compilador de FORTRAN, por ejemplo, necesitó para su implantación de 18 años de trabajo en grupo (Backus y otros [1975]).*

# Funciones Principales

- *Desde entonces, se han descubierto técnicas sistemáticas para manejar muchas de las importantes tareas que surgen en la compilación.*
- *También se han desarrollado buenos lenguajes de implantación, entornos de programación y herramientas de software.*
- *Con estos avances, puede hacerse un compilador real incluso como proyecto de estudio en un curso de un semestre sobre diseño sobre de compiladores.*

# Funciones Principales

- *Conceptualmente un compilador opera en fases.*
- *Cada una de las cuales transforma el programa fuente de una representación en otra.*
- *En la figura se muestra una descomposición típica de un compilador.*
- *En la práctica se pueden agrupar fases y las representaciones intermedias entres las fases agrupadas no necesitan ser construidas explícitamente.*

# Descomposición de un compilador

# Fases de un compilador

- *Administrador de la tabla de símbolos*
  - *Una función esencial de un compilador es registrar los identificadores utilizados en el programa fuente y reunir información sobre los distintos atributos de cada identificador.*
  - *Estos atributos pueden proporcionar información sobre la memoria asignada a un identificador, su tipo, su ámbito (la parte del programa donde tiene validez) y, en el caso de nombres de procedimientos, cosas como el número y tipos de sus argumentos, el método de pasar cada argumento (por ejemplo, por referencia) y el tipo que devuelve, si los hay.*

# Fases de un compilador

- *Administrador de la tabla de símbolos*
  - *Una tabla de símbolos es una estructura de datos que contiene un registro por cada identificador, con los campos para los atributos del identificador.*
  - *La estructura de datos permite encontrar rápidamente el registro de cada identificador y almacenar o consultar rápidamente datos en un registro.*

# Fases de un compilador

- *Administrador de la tabla de símbolos*
  - *Cuando el analizador léxico detecta un indentificador en el programa fuente, el identificador se introduce en la tabla de símbolos. Sin embargo, normalmente los atributos de un identificador no se pueden determinar durante el análisis léxico.*
  - *Por ejemplo, en una declaración en JAVA como int posicion, velociadad;*

# Fases de un compilador

- *Administrador de la tabla de símbolos*
  - *El tipo “int" no se conoce cuando el analizador léxico encuentra posición, inicial y velocidad.*
  - *Las fases restantes introducen información sobre los identificadores en la tabla de símbolos y después la utilizan de varias formas.*

# Fases de un compilador

- *Administrador de la tabla de símbolos*
  - *Por ejemplo, cuando se está haciendo el análisis semántico y la generación de código intermedio, se necesita saber cuáles son los tipos de los identificadores, para poder comprobar si el programa fuente los usa de una forma válida y así poder generar las operaciones apropiadas con ellos.*
  - *El generador de código, por lo general, introduce y utiliza información detallada sobre la memoria asignada a los identificadores.*

# Fases de un compilador

- *Detección e información de errores*
  - *Cada frase puede encontrar errores.*
  - *Sin embargo, después de detectar un error. Cada fase debe tratar de alguna forma ese error, para poder continuar la compilación, permitiendo la detección de más errores en el programa fuente.*
  - *Un compilador* ***que se detiene cuando encuentra el primer error****, no resulta tan útil como debiera.*

# Fases de un compilador

- *Detección e información de errores*
  - *Las fases de análisis sintáctico y semántico por lo general manejan una gran proporción de los errores detectables por el compilador.*
  - *La fase léxica puede detectar errores donde los caracteres restantes de la entrada no forman ningún componente léxico del lenguaje.*
  - *Los errores donde la cadena de componentes léxicos violan las reglas de estructura (sintaxis) del lenguaje son determinados por la fase del análisis sintáctico.*

# Fases de un compilador

- *Detección e información de errores*
  - *Durante el análisis semántico el compilador intenta detectar construcciones que tengan la estructura sintáctica correcta, pero que no tengan significado para la operación implicada, por ejemplo, si se intenta sumar dos identificadores.*
  - *Uno de los cuales es el nombre de una matriz, y el otro, el nombre de un procedimiento.*

# Fases de un compilador

- *Las fases del análisis.*
  - *Conforme avanza la traducción, la representación interna del programa fuente que tiene el compilador se modifica.*
  - *Para ilustrar esas representaciones, considérese la traducción de la proposición*

# Fases de un compilador

- *Las fases del análisis.*
  - *Posición = inicial + velocidad * 60*
  - *Id1 :=id2 +id3 *60*
  - *Genera la siguiente tabla de símbolos :*

# Fases de un compilador

- *1*
- *2*
- *3*
- *4 =*
- *id1*
- *id2*
- *id3 double*

- *60*
- *temp1= double (60)*
- *temp2=id3*temp1*
- *temp3=id2+temp2*
- *id1=temp3*
- *temp1=id3*60.0*

# Fases de un compilador

- *Representación de una proposición.*

- *La fase de análisis léxico lee los caracteres de un programa fuente y los agrupa en una cadena de componentes léxicos en los que cada componente representa una secuencia lógicamente coherente de caracteres, como un identificador, una palabra clave (if, while, etc), un carácter de puntuación, o un operador de varios caracteres, como ==.*

# Fases de un compilador

- *Representación de una proposición.*
  - *La secuencia de caracteres que forman un componente léxico se denomina lexema del componente.*
  - *A ciertos componentes léxicos se les agregará un "valor léxico".*
  - *Así, cuando se encuentra un identificador como **velocidad**, el analizador léxico no sólo genera un componente léxico, por ejemplo, **id**, sino que también introduce el lexema velocidad en la tabla de símbolos, **si aún no estaba allí**.*
  - *El valor léxico asociado con esta aparición de id señala la entrada de la tabla de símbolos correspondiente a velocidad.*

# Fases de un compilador

- *Representación de una proposición.*
  - *Usaremos id1 , id2 e id3 para posición, inicial y velocidad, respectivamente, para enfatizar que la representación interna de un identificador es diferente de la secuencia de caracteres que forman el identificador.*
  - *Por tanto, la representación después del análisis léxico queda sugerida por: id1 =id2 +id3*60*

# Analizador Léxico

*Ing. Edgar O. Hernández Barajas*

# Analizador Léxico

- *Se corresponden con las gramáticas de tipo 3 de la jerarquía de Chomsky.*
- *Las expresiones regulares son una forma de especificar patrones, entendiendo por patrón la forma de describir cadenas de caracteres.*
- *Es la forma de definir los **tokens** o **componentes léxicos**, cada patrón concuerda con una serie de cadenas. De esta forma, utilizamos las expresiones regulares para darle nombre a estos patrones*

# Analizador Léxico

- *El lenguaje que se reconoce mediante estas expresiones regulares (r), se denomina lenguaje generado por la expresión regular L(r).*

# Analizador Léxico

| Token (Componente léxico) | Lexema | Patrón |
|---|---|---|
| Identificador | a,valor,b | [a-zA-Z]+ |
| Numero | 5, 3.25, 56 | [0-9]+(\.[0-9]+)? |

# Analizador Léxico

- *Vemos que para describir un identificador que se define solo por letras utilizando el patrón [a-zA- Z]+, que reconoce cualquier letra mayúscula o minúscula seguido del símbolo +, estamos indicando que al menos tiene que haber una letra para describir un identificador, pero no hay límite para el número de caracteres que puede tener ese token (identificador).*

# Analizador Léxico

- *Una expresión regular se puede construir a partir de otras expresiones regulares más simples.*
- *Cuando definamos los símbolos mediante los que se especifican las expresiones regulares veremos ejemplos de estos, definiendo letras y dígitos y como un identificador es una combinación de ambos.*

# Analizador Léxico

- *Para definir las expresiones regulares usamos metacaracteres o metasímbolos que especifican las acciones que se pueden reconocer sobre un determinado carácter o símbolo.*
- *Algunos de estos metacaracteres son:* * , +, ?,|

# Analizador Léxico

- *Cuando queremos utilizar estos símbolos, como por ejemplo * con su significado normal, para la operación de multiplicar se utiliza un carácter de escape que anula el significado especial del metacarácter.*
- *En este ejemplo la forma correcta sería *, y así tenemos el símbolo de multiplicar en un patrón.*

# Analizador Léxico

| | |
|---|---|
| Selección entre alternativas | Se denota por el metacarácter \|. Ejemplo: a\|b, significa que puede ser a ó b. Esta operación equivale a la unión, puesto que tanto a como b valdrían como lexemas para este patrón (obsérvese que utilizamos patrón y expresión regular de forma indistinta). |
| Concatenación | Se construye poniendo un símbolo al lado del otro y no utiliza ningún metacarácter. Ejemplo: ab, significa que el lexema equivalente tiene que ser "ab", sin alternativa posible. |
| Repetición | También se la denomina cerradura de Kleene y se denota por el matacarácter *. Identifica una concatenación de símbolos incluyendo la cadena vacía, es decir "o o mas<br>instancias" del símbolo afectado. Ejemplo: a*, significa que los lexemas para este patrón podrían ser: , a, aa, aaa, aaaa,... |

# Analizador Léxico

- *Una o mas repeticiones:*
  - *Se denota le metacáracter +, también se le denomina cierre positivo.*
  - *Esta operación indica que el símbolo afectado tendrá una o mas instancias.*
  - *Ejemplo: a+, significa que los lexemas para esta expresión podrían ser: a, aa,aaa,aaaa, etc.*

# Analizador Léxico

- *Cero o una instancia:*
  - *Se denota por el metacarácter **?,** y significa cero o una ocurrencia del símbolo afectado.*
  - *Ejemplo: a?, significa que los lexemas que podrían ser validos son: a, Describe a como un símbolo opcional.*

# Analizador Léxico

- *Intervalo de carácteres:*
  - *[a-zA-Z]*
- *Clase de carácteres:*
  - *[abd], [eft],[aabcdp]*
- *Cualquier Caracter:*
  - *Se denota por un "." punto, sirve para expresar cualquier carácter encaja con la expresión regular, es común utilizar al final de una especificación de un analizador léxico*

# Analizador Léxico

- *Cualquier carácter que no pertenezca al conjunto:*
  - *Se denota por* **^** *o también por una tilde* **~** *y significa que cualquier carácter distinto a los que esta afectando el metacarácter.*
  - *Ejemplo: [^a] significa cualquier carácter que no sea a*

# Analizador Léxico

- *¿Que es un analizador léxico?*
- *Se encarga de buscar los componentes léxicos o palabras que componen el programa fuente, según unas reglas o patrones. La entrada del analizador léxico podemos definirla como una secuencia de caracteres.*

# Analizador Léxico

*Secuencia de Caracteres*

*Secuencia de Terminales*