

Introducción a la programación

con Java

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

4 de octubre de 2020



Programa

- 1 Programa
- 2 Traductores
- 3 Análisis del código
- 4 Bibliografía

Temas

- 1 Programa
 - Definición

Programa

Definición (Programa)

Un *programa* es una secuencia de instrucciones que le indica a la computadora cómo resolver un problema.

- Los programas implementan algoritmos.
- Los programas reciben *datos de entrada* y devuelven los resultados como *datos de salida*.

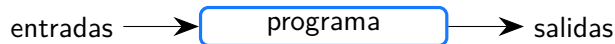
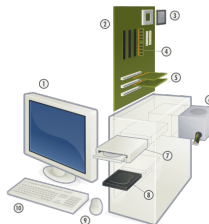


Figura: Diagrama de bloques de un *programa ejecutable*.

Lenguaje máquina

- Para que un programa sea ejecutado por una computadora, es necesario que esté escrito en un código que entienda la máquina.
- Este lenguaje es específico para cada modelo de computadora.
- A este código se le llama *lenguaje de máquina*.
- Consiste en una serie de comandos codificados en código binario.



Lenguajes de alto nivel

- Son más cercanos al lenguaje natural utilizado por los seres humanos (Ej: inglés), pero su definición es más formal y rígida.
- Por ello son lenguajes de *alto nivel de abstracción*.
- La computadora no puede ejecutarlos directamente.
- Se requiere traducirlos a lenguaje máquina.
- Se utilizan programas para realizar estas traducciones automáticamente.

```
1 public class Hola {  
2     public static void main(String[] args) {  
3         System.out.println("!Hola!");  
4     }  
5 }
```

Traductores

- 1 Programa
- 2 Traductores
- 3 Análisis del código
- 4 Bibliografía

Temas

2 Traductores

- Compiladores
- Compiladores en más fases
- Intérpretes

Compiladores

Definición (Compilador)

Un *compilador* es un programa que lee un programa escrito en un lenguaje, el lenguaje *fuentes*, y lo traduce a un programa equivalente en otro lenguaje, el lenguaje *objetivo*.

Aho y col. 2007

```
1 $ javac Hola.java
2 $ ls
3 Hola.java    Hola.class
```

- El lenguaje objetivo del compilador de Java es otro lenguaje llamado bytecode.

- Como parte importante de este proceso de traducción, el compilador informa a su usuario de la presencia de [algunos] errores en el programa fuente.
- Si el programa objetivo es un programa ejecutable en lenguaje de máquina, puede ser invocado por el usuario para su ejecución.



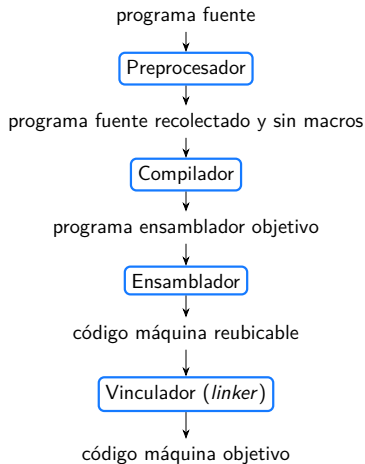
Figura: Diagrama de bloques del concepto *compilador*.

Temas

2 Traductores

- Compiladores
- **Compiladores en más fases**
- Intérpretes

Compiladores en más fases



Temas

- 2 Traductores
 - Compiladores
 - Compiladores en más fases
 - Intérpretes

Intérprete

Definición (Intérprete)

Un intérprete es un programa que, una vez cargado en la memoria de una computadora y al ejecutarse, procede como sigue:

- 1 Toma un enunciado del programa en lenguaje de alto nivel, llamado código fuente.
- 2 Traduce ese enunciado y lo ejecuta.
- 3 Repite estas dos acciones hasta que alguna instrucción le indique que pare, o bien tenga un error fatal en la ejecución. Viso y Peláez V. 2012

- Mientras que un *compilador* sólo traduce, un *intérprete* va traduciendo y ejecutando.

```
1 $ java Hola
2 ¡Hola!
```

- A los *intérpretes* se les conoce también como *máquinas virtuales*, porque una vez que están cargados en una máquina, se comportan como si fueran *otra* computadora, aquella cuyo lenguaje de máquina es el que se está traduciendo y ejecutando.

Análisis del código

- 1 Programa
- 2 Traductores
- 3 Análisis del código**
- 4 Bibliografía

Temas

- 3 Análisis del código
 - Análisis léxico, sintáctico y semántico
 - Tipos de errores
 - Tiempo de compilación y tiempo de ejecución

Análisis

- Un compilador analiza la estructura del código para poder generar su versión en otro lenguaje.
- Las tres primeras fases de este análisis son:
Análisis
 - 1 Léxico
 - 2 Sintáctico
 - 3 Semántico
- Posteriormente vendría la generación del código en el nuevo lenguaje.
 - 4 Generación de código intermedio
 - 5 Optimización del código
 - 6 Generación del código

Análisis léxico

El *análisis léxico*:

- Es la primer fase de análisis de un traductor.
- Recibe la secuencia de caracteres que forman el texto de un programa
- Agrupa estos caracteres en *lexemas*.
- Cada lexema es un conjunto de símbolos que tiene un significado unitario, por ejemplo:
 - identificadores
 - operadores
 - números
 - caracteres que marcan fin de línea o bloques.

Ejemplo

En la expresión:

```
1 float pi = 3.14f;
2 int r = 2;
3 int area = pi * r ^ 2;
```

Se encuentran los siguientes:

Tabla: Símbolos

Índice	símbolo
1	pi
2	r
3	área

Tabla: Lexemas

<tipo de lexema, valor>

<tipo, float>

<id, 1>

<=>

<3.14>

<;>

...

<id, 3>

<=>

<id, 1>

<*>

<id, 2>

...

Análisis sintáctico

El *análisis sintáctico* (*parsing*):

- Genera una representación intermedia que revela la estructura gramatical del código.

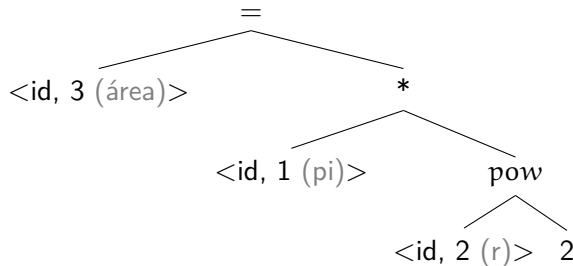


Figura: Árbol sintáctico.

Análisis semántico

El *análisis semántico*:

- Utiliza la información sintáctica y los símbolos identificados para verificar la consistencia semántica, según la definición del lenguaje fuente.
- Realiza la *verificación de tipos*, donde el traductor verifica que cada operación sólo se aplique sobre los tipos correctos de operandos.
- Algunos lenguajes permiten conversiones *coercitivas*, donde el programador indica que un dato de un tipo se debe **reinterpretar** como otro tipo para aplicarle una operación.

Ejemplo: consistencia semántica y verificación de tipos.

```
1 float pi = 3.14f;  
2 int r = 2;  
3 int area = pi * r ^ 2;
```

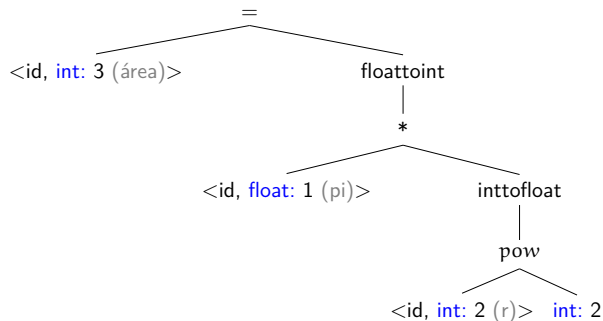


Figura: Árbol sintáctico con información semántica.

Temas

- 3 Análisis del código
 - Análisis léxico, sintáctico y semántico
 - Tipos de errores
 - Tiempo de compilación y tiempo de ejecución

Tipos de errores

Sintácticos Son las “faltas de ortografía”.

```
1 float pi = 3.14f           // Falta el ;  
2 flot  num = 0.11f;        // le falta la 'a' a float
```

Semánticos El significado escrito no corresponde con la intención del programador.

```
1 int x = "Número" + 8;    // Cadena + número no es un número
```

Lógicos ¡Bugs! El programa se ejecuta, pero no llega a los resultados esperados.

```
1 if (x != 0)  
2     y = 1/x;  
3     System.out.println("x es distinto de 0");    // Siempre se imprime
```

Temas

- 3 Análisis del código
 - Análisis léxico, sintáctico y semántico
 - Tipos de errores
 - Tiempo de compilación y tiempo de ejecución

Compilación vs ejecución



Cuando estamos programando distinguimos entre dos tiempos:

- *Tiempo de compilación*: Incluye los análisis llevados a cabo por el compilador. Durante la ejecución del compilador se pueden detectar los errores sintácticos y varios de los semánticos.
- *Tiempo de ejecución*: Es cuando el programa, ya traducido, está siendo utilizado. En este tiempo pueden ocurrir los errores lógicos si el algoritmo implementado no es correcto o si la programación fue errónea.

Bibliografía

- 1 Programa
- 2 Traductores
- 3 Análisis del código
- 4 Bibliografía**

Bibliografía I

-  Aho, Alfred V. y col. (2007). *Compilers, Principles, Techniques and Tools*. Addison Wesley.
-  Viso, Elisa y Canek Peláez V. (2012). *Introducción a las ciencias de la computación con Java*. 2a. Temas de computación. Las prensas de ciencias. 571 págs. ISBN: 978-607-02-3345-6.

Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual

