Conceptos y Paradigmas de los Lenguajes de Programación

Ulises J. Cornejo Fandos

Abril 2017

Práctica 3

1. ¿Que define la semantica?

   La semántica describe el significado de los símbolos, palabras y frases de un lenguaje ya sea lenguaje natural o lenguaje informático. La semantica se puede catalogar de dos formas:

   - Semantica Estatica
     Es la semantica que puede ser analizada al momento de compilación, esta corrobora que no haya variables sin declarar, verifica los tipos y atributos de las mismas, etc...
   - Semantica Dinamica
     Este tipo de semantica se analiza en ejecución, debido a que en determinados lenguajes no es posible realizar siempre un analisis total mediante la semantica estatica.

2. (a) ¿Qué significa compilar un programa?
   El proceso de compilación es aquel que traduce un programa escrito en un lenguaje de alto nivel a codigo maquina, indicando errores sintacticos, semanticos y optimizando parte del programa.

   (b) Describa brevemente cada uno de los pasos necesarios para compilar un programa.

   - Pre Procesado
     Este proceso solo se realiza en algunos lenguajes como C, lo que hace es reemplazar todas las Macros o constantes definidas en todo el codigo. También es el proceso encargado de incluir las librerias utilizadas.
   - Compilado
     Encargado de traducir el codigo de alto nivel en un lenguaje objeto (lenguaje de maquina, assembler o de cercano bajo nivel).
   - Ensamblado (Assembler)
     Encargado de traducir el codigo compilado en codigo maquina (ceros y unos).
   - Linkeado (Link-editor)
     Encargado de linkear librerias de sistema y reubicar el codigo de entrada con todos sus modulos.
   - Loader
     Encargado de cargar el programa a memoria, esto hace al mismo ejecutable, en codigo maquina.

   (c) ¿En qué paso interviene la semántica y cual es su importancia dentro de la compilación?

   La semantica es analizada en la etapa de compilado, esta es indispensable a la hora de prevenir futuros errores tales como variables no declaradas, errores de tipos y en 'casteos', etc...

3. Con respecto al punto anterior ¿es lo mismo compilar un programa que interpretarlo? Justifique su respuesta mostrando las diferencias básicas, ventajas y desventajas de cada uno.

No, si bien tanto la compilación como la interpretación de un codigo de alto nivel tienen pasos similares, estos no son los mismos, sus principales diferencias son:

- Tiempo de ejecución
  En la interpretación, por cada sentencia se realiza el proceso de decodificación para determinar las operaciones a ejecutar y sus operandos, si la sentencia está en un proceso iterativo, se realizará la tarea tantas veces como sea requerido,la velocidad de proceso se puede ver afectada durante la ejecución. Mientras que en la compilación no se repiten lazos, el codigo de alto nivel se decodifica una unica vez.
- Eficiencia
  Al tener que decodificar linea por linea, le interpretación es mas lenta a la hora de ejecutarse, mientras que, desde el punto de vista del hardware, la compilación es mas eficiente.
- Espacio ocupado
  Como cada sentencia al ser compilada puede abarcar miles de sentencias de lenguaje maquina, el codigo compilable ocupa mas espacio, mientras que el interprete, al mantener sus sentencias en su forma original, optimiza este campo.
- Detección de errores
  Al momento de la detección de errores la interpretación puede ser mas precisa dependiendo determinados casos, el unico inconveniente es que al ser codigo dinamico, el cual no siempre sigue el mismo 'camino', puede que no nos enteremos de que parte del codigo tiene errores.

4. Explique claramente la diferencia entre un error sintáctico y uno semántico. Ejemplifique cada caso.

   - Un error sintactico hace referencia a los errores que ocurren cuando una sentencia no esta escrita en un formato aceptable, este formato suele estar definido en un documento BNF/EBNF e indica como se definen dichas sentencias, ejemplos de esto pueden ser en un lenguaje como C:

     x int = 2;
     (en C seria int x = 2;)

     x++
     (le falta el ';')

   - Un error semantico hace referencia a errores de tipos, variables indeclaradas, etc... Casos comunes suelen ser (nuevamente en C):

     int x = 2;
     x = 3.5;
     (error de tipo)

5. Sean los siguientes ejemplos de programas. Analice y diga qué tipo de error se produce (Semántico o Sintáctico) y en qué momento se detectan dichos errores (Compilación o Ejecución). Aclaración: Los valores de la ayuda pueden ser mayores.

   Pascal

```
Program P
var 5: integer;
var a:char;
Begin
for i:=5 to 10 do begin
write(a);
a=a+1;
end;
End.
// Ayuda: sintacticos 2, semanticos 3
```

Java

```
public String tabla(int numero, arrayList<Boolean> listado)
{
String result = null;
for(i = 1; i < 11; i--) {
result += numero + "x" + i + "=" + (i*numero) + "\n";
listado.get(listado.size()-1)=(BOOLEAN)numero>i;
}
return true;
}
```

Ayuda:

```
// Sintacticos 4, Semanticos 3, Logicos 1
```

C

```c
# include <stdio.h>
int suma; /* Esta es una variable global */
int main()
{ int indice;
encabezado;
for (indice = 1 ; indice <= 7 ; indice ++)
cuadrado (indice);
final(); Llama a la funcion final */
return 0;
}
cuadrado (numero)
int numero;
{ int numero_cuadrado;
numero_cuadrado == numero * numero;
suma += numero_cuadrado;
printf("El cuadrado de %d es %d\n",
numero, numero_cuadrado);
}

// Ayuda: Sintacticos 2, Semanticos 6
```

Python

```python
#!/usr/bin/python
print "\nDEFINICION DE NUMEROS PRIMOS"
r = 1
while r = True:
N = input("\nDame el numero a analizar: ")
i = 3
fact = 0
if (N mod 2 == 0) and (N != 2):
print "\nEl numero %d NO es primo\n" % N
else:
while i <= (N^0.5):
if (N % i) == 0:
mensaje="\nEl numero ingresado NO es primo\n" % N
msg = mensaje[4:6]
print msg
fact = 1
i+=2
if fact == 0:
print "\nEl numero %d SI es primo\n" % N
r = input("Consultar otro numero? SI (1) o NO (0)--->> ")

// Ayuda: Sintacticos 2, Semanticos 3
```

Ruby

```ruby
def ej1
Puts 'Hola, Cual es tu nombre?'
nom = gets.chomp
puts 'Mi nombre es ', + nom
puts 'Mi sobrenombre es 'Juan''
puts 'Tengo 10 agnos'
meses = edad*12
dias = 'meses' *30
hs= 'dias * 24'
```

```
puts 'Eso es: meses + ' meses o ' + dias + ' dias o ' + hs + ' horas
puts 'vos cuantos agnos tenes'
edad2 = gets.chomp
edad = edad + edad2.to_i
puts 'entre ambos tenemos ' + edad + ' agnos'
puts 'Sabes que hay ' + name.length.to_s + ' caracteres en tu nombre, ' + name
end
```

6. Dado el siguiente código escrito en pascal. Transcriba la misma funcionalidad de acuerdo al lenguaje que haya cursado en años anteriores. Defina brevemente la sintaxis (sin hacer la gramática) y semántica para la utilización de arreglos y estructuras de control del ejemplo.

```
Procedure ordenarArreglo(var arreglo:
arregloDeCaracteres; cont:integer);
var
i:integer; ordenado:boolean;
aux:char;
begin
repeat
ordenado:=true;
for i:=1 to cont-1 do
if ord(arreglo[i]) > ord(arreglo[i+1])
then begin
aux := arreglo[i];
arreglo[i] := arreglo[i+1];
arreglo[i+1] := aux; ordenado := false
end;
until ordenado;
end;
```

Observación: Aquí sólo se debe definir la instrucción y qué es lo que hace cada una; detallando alguna particularidad del lenguaje respecto de ella. Por ejemplo el for de java necesita definir una variable entera, una condición y un incremento para dicha variable.

7. Explique cuál es la semántica para las variables predefinidas en lenguaje Ruby 'self' y 'nil'. ¿Que valor toman? ¿Cómo son usadas por el lenguaje?

   En Ruby 'self' hace referencia al objeto que se esta ejecutando en el momento, la referencia 'self' puede ser utilizada en cualquier momento del programa y tendra el contexto semantico que tiene el objeto al que referencie en ese momento de ejecución. Por otro lado, 'nil', hace referencia al valor null de otros lenguajes de programación, suele utilizarse para verificar si hay variables que no hayan sido inicializadas, u otras funciones de control, su contexto semantico siempre debe ser utilizado en relación a una variable o expresión lógica.

8. Determine la semántica de la sentencia 'break' en C, PHP, javascript y Ruby. Cíte las características más importantes de esta sentencia para cada lenguaje.

   - 'break' en C

     Se utiliza para la 'ruptura' de bucles tales como la sentencia 'for', 'while' y también para realizar una salida de estructuras como 'swich', la sentencia 'break' debe estar anidado en alguna de dichas estructuras, este es su contexto semantico.

   - 'break' en PHP

     Dicha sentencia tiene el mismo contexto semantico que en C (debido a que este lenguaje deriba del mismo).

   - 'break' en javascript

     Sigue la semantica de C.

   - 'break' en Ruby

     Ruby utiliza 'break' al igual que C y los demás lenguajes mencionados, su diferencia es que funciones como 'switch' no existen (existen equivalencias como 'case'), mas allá de los nombres de las mismas, el contexto semantico es el mismo.

9. Defina el concepto de ligadura y su importancia respecto de la semántica de un programa. ¿Qué diferencias hay entre ligadura estática y dinámica? Cite ejemplos (proponer casos sencillos).

   La ligadura hace referencia a la especificación exacta de la naturaleza de un atributo, ya sea su tipo o clase perteneciente. Esta se puede definir como:

   - Ligadura estatica

     Este tipo de ligadura se establece antes de la ejecución del programa y no es modificable durante el tiempo de vida del mismo.

     ```
     int x; // Liga el tipo int al atributo tipo de la variable x.
     ```

   - Ligadura dinamica

     Se establece en el momento de la ejecución y puede cambiarse durante la misma, siguiendo las reglas que especifique cada lenguaje de programación.

     ```
     x = 2 // Liga el tipo int al atributo tipo de la variable x.
     x = 'c' // Cambia el atributo tipo de la variable x por char.
     ```