Conceptos previos

- Hemos visto que al usar una notación BNF (o cualquiera de sus variantes) para documentar un Lenguaje de programación, podemos:
  - Determinar con precisión las categorías léxicas (Tokens) que lo componen
  - Determinar las categorías sintácticas del lenguaje, pero con algunas restricciones
- Estas restricciones se deben a:
  - No complicar la gramática en demasía
  - Limitaciones de las GIC, ya que hay reglas que realmente corresponden a una GSC

Semántica estática

- A las restricciones no representadas en la BNF pero aclaradas en la documentación del lenguaje, las llamamos semántica estática si su incumplimiento puede ser detectado al compilar.
  - Los errores de este tipo reportados por el compilador los clasificaremos como errores semánticos.
- Casos de semántica estática (En particular en lenguaje C)
  - Compatibilidad de tipos entre operadores y operandos.
  - Que los identificadores hayan sido declarados (problemas de ámbito y alcance).
  - Que no se declare más de una vez un identificador (redeclaración) dentro de un mismo espacio de nombre.
  - Que las funciones se llamen con la cantidad y tipo correcto de parámetros.

Semántica

- La semántica (significado) de un constructo sintáctico es "que hace", es decir que implica en la ejecución del programa.
  - Por eso algunos la llaman semántica dinámica.
  - En muchos casos se usa el concepto de una máquinas virtual para explicar el funcionamiento (por ejemplo: el estándar de C)
- Hay varios enfoques para definir la semántica, el comentado en el párrafo anterior, que es el que usaremos, se lo conoce como semántica operacional, es decir, describe la operación del programa
- Otros enfoques recurren a modelos matemáticos, como la semántica denotacional que suele usar funciones, o la semántica axiomática que se basa más en relaciones.

Semántica

- La semántica es la que guía el código que se va a generar.
- No solo hablaremos de la semántica de sentencias, podemos hablar, por ejemplo, de la semántica de la definición de variables.
  - Zona de memoria donde se alojará
  - Valor inicial por defecto
  - Interpretación del patrón de bits
- El estándar de C define la semántica y habla de comportamiento para definirla operacionalmente. Sin embargo hay casos donde el mismo no queda totalmente definido.

Ejemplo del Estándar de C

6.5.5 Multiplicative operators

Syntax

```
multiplicative-expression:
              cast-expression
              multiplicative-expression * cast-expression
              multiplicative-expression / cast-expression
              multiplicative-expression % cast-expression
```

Constraints

Each of the operands shall have arithmetic type.
The operands of the % operator shall have integer type.

Semantics

The result of the binary * operator is the product of the operands.

Comportamiento

- Comportamiento:
  - Apariencia externa de la acción
- Comportamiento no especificado:
  - El estándar da un conjunto de posibilidades y la implementación elige una, o varias de ellas en diferentes partes del programa. Ejemplo: el orden de evaluación de los argumentos de una función. Esta "omisión" por parte del estándar tiene por objetivo permitir hacer optimizaciones al compilador.
- Comportamiento dependiente de la implementación:
  - Comportamiento no especificado en el estándar que cada implementación debe decidir y documentar, haciendo luego siempre lo mismo. Ejemplos: tamaño de un int, desplazamiento de bits a derecha en enteros con signo.

Comportamiento

- Comportamiento específico local (o regional)
  - Es el que depende de la configuración regional, por ejemplo la función islower puede devolver verdadero para algunos caracteres no pertenecientes al ascii básico.
- Comportamiento no definido:
  - El estándar no lo define y en general puede ocurrir que el compilador lo resuelva de alguna manera, compile y de una advertencia. O bien que de un error de compilación. O más comúnmente, compila (ignorando la situación) pero en ejecución el resultado es impredecible, pudiendo producir un error de ejecución que cuelgue el programa. Ejemplos: el uso de una variable automática antes de asignarle valor, o el uso de un puntero a una zona de memoria que ya fue liberada.

Sintaxis y Semántica (estática)

- Se considera que es un **error sintáctico**, cuando la cadena analizada no puede ser derivada de la BNF
- Si una cadena es **derivable** de la BNF decimos que es **sintácticamente correcta**. Pero si no cumple con las restricciones (semántica estática) del lenguaje: es un **error semántico**.
- Ejemplo en C:

```
if (var) i++;
```

  - Es sintácticamente correcto
  - Semánticamente correcto si var fue definida con un tipo escalar (numérico o puntero).

Distinta Sintaxis e Igual Semántica

- La misma semántica se puede implementar en diferentes lenguajes con sintáxis distintas. Por ejemplo la asignación, en todos los ejemplos siguientes se asigna el valor 15 a la variables nro:
  - `nro = 15;` En lenguajes C,C++,Java y otros
  - `nro := 15;` En lenguajes Pascal, Algol y otros
  - `nro <- 15` En lenguajes R, S, quizás otros

Igual Sintaxis y Distinta Semántica

- C y C++ divergieron con el tiempo, un ejemplo:
  - `sizeof('a')`
    - El valor numérico en C++ es 1 y en C depende de la implementación, ya que el tipo de dato de una constante de carácter es tipo char para C++ pero tipo int en C.
  - ++a (donde a es tipo int)
    - En C devuelve un valorR y en C++ una valorL (una referencia).

Licencia