Primer boletín de prácticas sobre MMPor

IG29: Compiladores e intérpretes

Sesiones segunda y tercera de prácticas

(1) Comprueba que el documento **Diseño inicial del lenguaje MMPor** se corresponde perfectamente con la calculadora con sumas, restas y multiplicaciones que se te ha proporcionado como recurso "MMPor, versión 0" en la *web* de la asignatura.

(2) Adapta ese diseño para modificar el funcionamiento de la calculadora en el siguiente sentido: la entrada consistirá en una expresión por línea (y no, como hasta ahora, en una única expresión que podía extenderse por diferentes líneas). Para ello, procede paso a paso:

- **a.** Modifica la especificación léxica para que el salto de línea pase, de ser un blanco que se omite, a suponer la emisión de un componente de la categoría léxica **salto**.
- **b.** Añade a la gramática una primera producción que indique que la entrada consistirá en una secuencia ⟨L⟩ de cero o más expresiones ⟨E⟩, cada una de ellas seguida por un terminador **salto**.
- **c.** Añade una nueva clase de nodo al diseño de los ASTs, Secuencia, con una lista de hijos expresión (un hijo expresión por cada línea de la entrada).
- **d.** Añade al esquema de traducción la nueva regla de la gramática con las acciones necesarias para, inicialmente, crear una lista vacía en una variable local *laux*; después, por cada expresión ⟨E⟩ analizada, añadir su correspondiente AST a la lista *laux*; y finalmente, asociar al símbolo inicial, como AST, un nodo Secuencia con *laux* como lista de hijos.

(3) Modifica la implementación proporcionada para ajustarla al nuevo diseño:

- **a.** Traslada las modificaciones de la especificación léxica a las primeras líneas de `mmpor.mc`.
- **b.** Crea en el módulo `AST.py` una nueva clase, `Secuencia`, que:
  - Acepte la correspondiente lista de hijos en su constructor.
  - Tenga un método `ejecuta` para, cada expresión, evaluarla e imprimir su resultado.
  - Tenga un método que permita que, de intentar imprimir un nodo Secuencia, éste se muestre como

    ```
    ( "Secuencia" ... )
    ```

    donde, en vez de puntos suspensivos, lo que debe mostrarse entre el nombre del nodo y el cierre de paréntesis es la representación de cada nodo hijo.
- **c.** Intenta incorporar al esquema de traducción de `mmpor.mc` las modificaciones introducidas en el diseño. Observa que el símbolo inicial auxiliar `<X>` deberá pasar a reescribirse como `<L>`, no como `<E>`.
- **d.** Modifica la definición de la función `main` para que se llame al método `ejecuta` del AST de la entrada (en vez de llamar a `evalua` e imprimir).

(4) Utiliza Metacomp para transformar `mmpor.mc` en un programa Python de nombre `mmpor` y pruébalo:

- **a.** Primero, lleva a cabo un par de pruebas informales, utilizando el teclado para introducir un par de expresiones sencillas en cada ocasión. ¿Proporciona los resultados esperados o hay que depurar errores obvios?
- **b.** Después, crea seis ficheros para probar con más cuidado tu implementación:
  - `p01.mmp`, un fichero vacío.
  - `p01.sal`, otro fichero vacío.
  - `p02.mmp`, un fichero con una expresión compleja donde aparezcan componentes de todas las categorías léxicas; de cada una (salvo de la categoría **salto**), al menos tres.
  - `p02.sal`, un fichero con el resultado de la expresión anterior.
  - `p03.mmp`, un fichero con diez expresiones de distinta complejidad.
  - `p03.sal`, un fichero de diez líneas con los resultados de las expresiones anteriores.

c. Ejecuta en el intérprete de órdenes de Linux lo siguiente:

```
for i in 1 2 3; do
    b=p0${i}
    echo "============> "${b}
    ./mmpor < ${b}.mmp | diff ${b}.sal -
done
```

Si no ha habido problemas, esto es todo lo que debería mostrarse:

```
============> p01
============> p02
============> p03
```

(5) Entrega en la correspondiente tarea de la *web* un paquete `entrega02.tgz` con los ficheros siguientes:

- `disenyo.txt`, tal como haya quedado después de la adaptación.
- `mmpor.mc` y `AST.py`, también después de la adaptación.
- `p01.mmp`, `p01.sal`, `p02.mmp`, `p02.sal`, `p03.mmp` y `p03.sal`.

(6) Puedes ir adelantando qué modificaciones serán necesarias para que tu calculadora pueda contar con variables; más concretamente, para que acepte lo siguiente:

- En cada línea, antes de la correspondiente expresión, deberá haber dos componentes adicionales: un identificador, formado por una secuencia de letras, y un símbolo formado por un carácter dos puntos y un igual. La línea, además de imprimir el identificador, un espacio en blanco y el resultado de la expresión, se entenderá como una asignación.
- En las expresiones, los operandos elementales podrán ser, aparte de literales, identificadores de variables inicializadas en alguna línea anterior.