Consideraciones sobre la entrega 3

Eduardo Amaya Espinosa y Pablo Sanz Sanz

Junio de 2020

Junto con este PDF se entregan los ficheros fuente del compilador para nuestro lenguaje. También se entregan unos cuantos ejemplos de programas en nuestro lenguaje que pueden servir de pruebas. Estos ejemplos se pueden encontrar en el directorio `src/main/resources/examples`. Para usar el compilador se debe ejecutar la clase `Compiler`, que recibe como parámetro de línea de comandos el fichero principal a compilar. Si este fichero importara otros, estos deberán encontrarse en la ruta relativa al principal especificada en la orden `import`. Por ejemplo, si compilamos un archivo `File.bbl` en `/home/user/` contiene la línea `import imports/Import;` entonces debe existir el archivo `/home/user/imports/Import.bbl`, pero no hace falta que se le indique nada al compilador. De todos los archivos importados exactamente uno de ellos debe definir el método `main()`, independientemente de que sea este el que se ha usado como principal para compilar.

Asimismo, durante el desarrollo de esta tercera fase de la práctica hemos hecho ligeras modificaciones (o aclaraciones) sobre nuestro lenguaje que conviene comentar, a continuación se listan.

1. El tipo primitivo `Real` se traduce directamente al entero que representa su código binario. Esto es debido a que la Máquina-P que estamos usando no tiene soporte para números en coma flotante. Así que, en caso de usar este tipo ocurrirán resultados inesperados. En cualquier caso, sigue siendo un tipo distinto de `Int`.

2. Debido a la imposibilidad de determinar su tipo, las listas vacías, antes `[]`, ahora deben indicarlo explícitamente: `[Int]`.

3. Se permite la sobrecarga del operador `...` en la clase `T` siempre que el argumento recibido sea de tipo `T` y el tipo devuelto sea `Array<T>`. Así, se puede iterar en un bucle `for` "normal" un tipo creador por el usuario:

   ```
   for Point x in p ... q
   ```

   Igualmente, si usamos este operador en una expresión, no en un bucle `for`, en el caso de los tipos primitivos `Int`, `Char` y `Real` devolverá también un array conteniendo todos los valores entre el operando izquierdo y el derecho, ambos incluidos. En los bucles no se llega a crear el array y simplemente se recorren estos valores.

4. Ya no es necesario usar la palabra reservada `this` para acceder a atributos y métodos de una clase dentro de esta. Sólo será necesario cuando haya conflicto de nombres. Por ejemplo,

   ```
   class Bar { Int x = 0; Void foo() { x = 1; } }
   ```

   es válido, pero en el siguiente caso sí que hay que usar `this` para acceder al atributo `x` de la clase:

   ```
   class Bar { Int x = 0; Void foo(Int x) { this.x = x; } }
   ```

5. Como ya habíamos especificado, los diferentes casos de un `switch` solo pueden ser constantes. Para nosotros los números negativos no son constantes, sino el operador `-` con una constante, por tanto no se pueden utilizar como casos en un `switch`.

6. El constructor genérico de `Array` ahora únicamente recibe como parámetro el tamaño de la primera dimensión. Si se quiere inicializar un array multidimensional donde todas las dimensiones tengan el mismo tamaño debe hacerse dimensión a dimensión con un bucle, por ejemplo. Además tambien se debe especificar en el constructor, utilizando el operador diamante `<>`, el tipo del `Array`.

7. Se ha eliminado el operador `==` de los objetos anónimos `Form` por la impracticabilidad de generar su código.

Además, añadimos otra lista con errores que creemos que pueden aparecer y, por tanto, aconsejamos evitar este tipo de prácticas.

1. No se puede crear una lista vacía de un tipo en mayúsculas: `[TYPE]` se considera una lista de un elemento formado por la constante `TYPE`. De todas formas esto es equivalente a `Array<TYPE>(0)`, con lo cual se recomienda usar esta expresión en lugar de la anterior.

2. Puede haber un bucle en declaraciones de variables globales de la siguiente forma.

   ```
   Foo x = Foo(); Int y = 100; class Foo { Int x = y; constructor() {} }
   ```

   Quedará algo de la forma `x = Foo[x = ?]`, `y = 100`, porque `y` aún no se habría inicializado al crear `Foo`. La interrogación será generalmente `null` aunque podría contener cualquier otro dato.