Universidad de Alcalá

Práctica 3: Semántica y generación de código. El compilador completo.

Prof. Marçal Mora Cantallops, Universidad de Alcalá 17 de noviembre de 2025

Objetivos de la práctica

- Construir un sistema completo de compilación, desde el análisis léxico hasta la ejecución del código.
- Ser capaz de generar un analizador léxico y sintáctico que reconozca un lenguaje en concreto.
- Ser capaz de generar código intermedio que transforme el lenguaje dado.
- Ser capaz de añadir mejoras y funciones adicionales al proceso.
- Profundizar en el concepto de la tabla de símbolos y la gestión de errores.

Consideraciones generales

- La práctica se realizará en grupos de hasta tres personas, siguiendo lo planteado en la PL2.
- Para la realización de la práctica se deberá usar ANTLR. Está permitido el uso de plugins para Intellij, NetBeans, Eclipse, Visual Studio Code, Visual Studio IDE, y jEdit.
- Para el segundo caso, se usará Jasmin como assembler del código intermedio generado (que, por lo tanto, deberá seguir su sintaxis).
- El entregable será tanto el código generado como una memoria explicativa del proceso seguido; dicha memoria será el principal factor en la valoración final (previa comprobación de que el código funciona adecuadamente).

Enunciado

Esta práctica consta de tres partes separadas.

Primera parte: del CSV al JSON (Básico, 2,5 puntos)

Implemente, mediante el uso de visitors, un programa que use la gramática generada para reconocer CSV y traduzca su contenido a formato JSON.

Por ejemplo, en el caso de la práctica anterior:

```
Nombre; Frase; Tipo; Lanzamiento
Aatrox; the Darkin Blade; Juggernaut; 2013-06-13
Ahri; the Nine-Tailed Fox; Burst; 2011-12-14
Akali; the Rogue Assassin; Assassin; 2010-05-11
```

Debería convertirse en salida en:

```
[ { Nombre: 'Aatrox',
    Frase: 'the Darkin Blade',
    Tipo: 'Juggernaut',
    Lanzamiento: '2013-06-13' },
  { Nombre: 'Ahri',
    Frase: 'the Nine-Tailed Fox',
    Tipo: 'Burst',
    Lanzamiento: '2011-12-14' },
  { Nombre: 'Akali',
    Frase: 'the Rogue Assassin',
    Tipo: 'Assassin',
    Lanzamiento: '2010-05-11' } ]
```

Segunda parte: generando código intermedio en Jasmin (Básico/Intermedio, 2,5 puntos)

Esta primera parte es independiente de ANTLR y del lenguaje dado, aunque lo podéis usar de referencia.

Se proporciona una lista de ejemplos y se pide, para cada uno de ellos, realizar las siguientes tareas:

- Entender el ejemplo a ejecutar y proponer un código que cumpla con lo pedido.
- Traducir el código propuesto ("manualmente") a código intermedio Jasmin.
- Mostrar el resultado de la ejecución del código intermedio generado.

Los ejemplos a realizar son los siguientes:

1. Ser capaz de generar un programa principal vacío.
2. Ser capaz de generar un programa principal que muestre por pantalla una cadena de texto.

3. Ser capaz de generar un programa principal que multiplique dos números enteros y lo muestre por pantalla.
4. Ser capaz de generar un programa principal que muestre por pantalla el resultado de una operación lógica.
5. Ser capaz de generar un programa principal que muestre por pantalla la concatenación de una cadena de texto y un número.
6. Ser capaz de generar un programa principal que realice uno o varios "if" anidados, mostrando el resultado de cada uno.
7. Ser capaz de generar un bucle for.
8. Ser capaz de generar un bucle while.
9. Ser capaz de llamar a una función sin parámetros ni devolución de resultado.
10. Ser capaz de llamar a una función devolviendo un resultado en un entero.
11. Ser capaz de llamar a una función pasando un parámetro.
12. Ser capaz de llamar a una función pasando varios parámetros.

Los dos primeros ejemplos se proporcionan. El resto se valorarán a 0,25 por ejemplo correcto.

Tercera parte: compilando E++ (5 puntos)

Esta parte es una extensión sobre el lenguaje analizado en la práctica 2 y comprende la ejecución de las siguientes tareas:

Nivel básico (1,5 puntos):

- Extender el lenguaje para que pueda:
  - Realizar el resto de operaciones aritméticas comunes.
  - Tener bucles tipo for. Podéis llamarlos como queráis.
  - Tener bucles tipo while. Idem.
  - Admitir el tipo booleano, y, por lo tanto, sus operaciones.
- Adaptad lo que sea necesario en el lenguaje y justificad las decisiones de diseño tomadas.

Nivel intermedio (3,5 puntos):
Implementar y explicar adecuadamente en la memoria las decisiones tomadas al respecto:

- Implementar una estructura de tabla de símbolos, que mantenga las variables encontradas y sus valores/tipos cuando corresponda.

- Implementar la gestión de errores (incluyendo errores semánticos) para los errores que consideréis necesarios.
- Generación de código intermedio Jasmin para al menos un ejemplo de cada una de las funciones del lenguaje (es decir, para una serie de ejemplos representativos a modo de conjunto de pruebas). Estos ejemplos deben incluir también posibles casos de error y mostrar cómo el compilador lanza su correspondiente mensaje al usuario.
- Ejecución del código intermedio generado para comprobar su correcto funcionamiento.

Para todos ellos debéis asumir que el programa entero estará en un solo fichero (es decir, no hay *includes* o similar).

Secuencia recomendada

Seguir el orden establecido en la propia práctica.

Defensa

En la defensa de la práctica se deberá exponer y explicar los distintos componentes que formen el sistema programado por el alumno, respondiendo a las preguntas del profesor, si corresponde. Adicionalmente, se probarán ficheros de entrada en formato texto que se deberán poder ejecutar (nota: esos ficheros, en caso de incluir funcionalidades extendidas, se adaptarán a la sintaxis especificada por los alumnos).

Ejemplos

Se encuentran como materiales anexos.

Material Adicional

Tenéis también varios vídeos explicativos del proceso completo en la plataforma.