Tutorial de construcción de un compilador

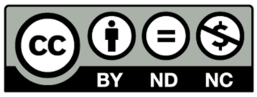
Versión 3.0

Contenido

Raúl Izquierdo Castanedo
raul@uniovi.es

Área de Lenguajes y Sistemas Informáticos
Departamento de Informática

Escuela de Ingeniería Informática de Oviedo
Universidad de Oviedo



Objetivo

El objetivo de este tutorial es mostrar el proceso básico de construcción de un traductor de un lenguaje de programación. Concretamente se implementará un compilador. Para ello se ha elegido un lenguaje de programación sencillo que permita centrarse en el proceso de su construcción en vez de en las características particulares del mismo.

En este documento se describe únicamente:

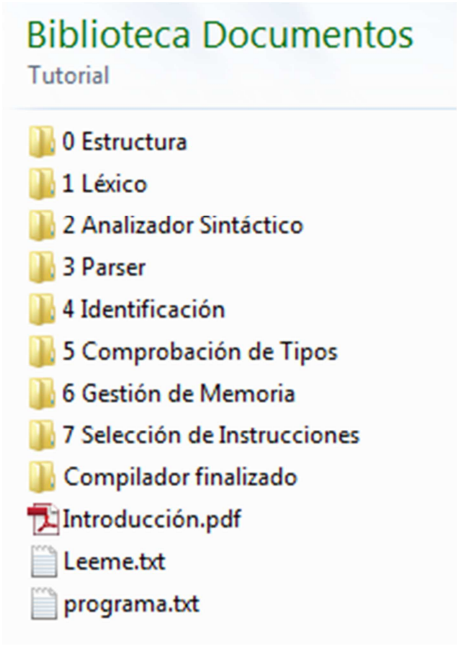
1. La estructura común que tiene cada uno de los capítulos del tutorial.
2. El lenguaje del cual se implementará un compilador a lo largo del tutorial.

No es objetivo de este tutorial sustituir a la parte teórica de la asignatura sino ser un complemento de las mismas. Por tanto no se explicarán aquí los metalenguajes ni las herramientas utilizadas. Se aconseja realizar cada capítulo del tutorial después de la clase teórica correspondiente a cada módulo del compilador.

Estructura del Tutorial

En la carpeta principal del tutorial hay dos documentos. Uno es el documento actual (*Introducción.pdf*). El otro documento, *programa.txt*, es un programa de ejemplo en el lenguaje para el cual se implementará el compilador.

El tutorial contiene además nueve carpetas. El contenido de estas carpetas es:



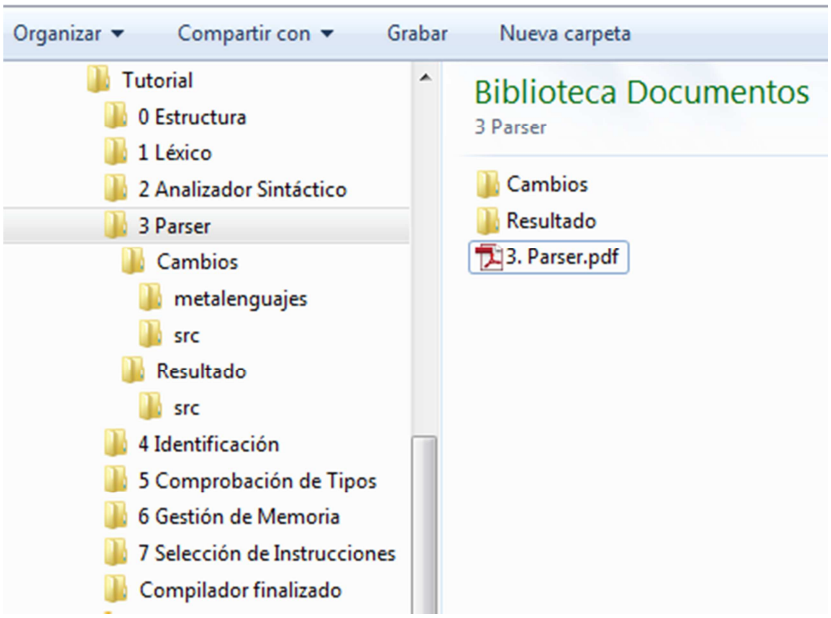
- La carpeta *"0 Estructura"* contiene el esqueleto de un traductor genérico el cual incluye un método *main*, unas clases auxiliares y unos paquetes (*packages*) ya creados donde ir añadiendo los fuentes que se implementen en el resto de los capítulos. Es el punto de partida del tutorial.
- A continuación aparece el tutorial propiamente dicho dividido en 7 capítulos (de *"1 Léxico"* a *"7 Selección de Instrucciones"*), cada uno de los cuales se corresponde con una de las fases del proceso de construcción y añade la parte del traductor

correspondiente. La estructura de estos capítulos se describe en el apartado siguiente.

- Por último aparece la carpeta *"Compilador finalizado"* donde se encuentra el código completo del traductor listo para ser compilado y ejecutado (por ejemplo sobre "*programa.txt*").

Estructura de cada Capítulo del Tutorial

Las carpetas del tutorial (numeradas de 1 a 7) tienen todas la misma estructura. Por ejemplo en el capítulo 3 se tiene el siguiente contenido:



- Un documento PDF (en este caso *"3. Parser.pdf"*) que indica qué se hará en dicho capítulo.
- Una carpeta "*Cambios*" con únicamente aquellos ficheros que se hayan modificado o añadido en esta fase. Es una forma de poder comprobar rápidamente lo que se ha hecho en este capítulo. Esta carpeta tendrá a su vez dos subcarpetas:
  - "*metalenguajes*" con la especificación de la fase en el metalenguaje correspondiente.
  - "*src*" con los ficheros *Java* modificados o añadidos.
- Una carpeta "*Resultado*" con la fase del traductor sobre la que trate dicho capítulo ya implementada (compilable y ejecutable). El contenido de dicha carpeta se ha obtenido en cada caso copiando la carpeta "*Resultado*" *del capítulo anterior* y añadiendo los ficheros de "*Cambios*" de este capítulo.

La forma recomendada de seguir este tutorial es entrar en los capítulos por orden y, en cada uno de ellos, seguir los siguientes pasos:

1. Abrir en el documento PDF del capítulo y leer el apartado *Objetivo* (pero no leer aún el apartado *Solución*). En dicho apartado se indica qué debe hacer el nuevo módulo del compilador sobre el que trata el capítulo.

2. Intentar hacer el diseño y la implementación de dicha fase partiendo de los fuentes de la carpeta *Resultado* de la fase *anterior*. De esta manera se puede implementar cada capítulo partiendo de una base de código estable aunque no se hayan hecho los capítulos anteriores.
3. Leer el resto del PDF del capítulo (diseño de la solución y su implementación) y comparar con la solución propuesta por el alumno. Los ficheros con la solución de esta fase se encuentran en la en la carpeta "*Cambios*" .

Descripción del lenguaje de entrada

Programa de Ejemplo

En *programa.txt* se tiene un programa de ejemplo que muestra el léxico y la sintaxis del lenguaje de este tutorial:

```
DATA
	float precio;
	int ancho;
	int alto;
	float total;

CODE
	precio = 9.95;
	total = (precio – 3.0) * 1.18;
	print total;

	ancho = 10; alto = 20;
	print 0 - ancho * alto / 2;
```

Especificación del Lenguaje

A continuación se describen las características del lenguaje de programación mediante lenguaje natural. A lo largo del desarrollo del traductor se irán expresando dichas características de manera formal con distintos metalenguajes.

Características del lenguaje. Aspectos generales

- La sección *DATA* deberá aparecer obligatoriamente antes de la sección *CODE* y cada una de ellas solo puede aparecer una vez.
- Podrán aparecer comentarios en cualquier parte del programa.

Características del lenguaje. Sección de Datos

- En la sección *DATA* se realizan las definiciones de variables (no puede haber definiciones en la sección *CODE*).
- No es obligatorio que se definan variables pero en cualquier caso tiene que aparecer la palabra reservada *DATA*.
- Las variables solo pueden ser de tipo entero o tipo real (*float*). Las variables de tipo entero ocupan 2 bytes y las reales 4.
- En cada definición habrá una sola variable (por ejemplo no se permite "*int a,b;*")

Características del lenguaje. Sección de Código

- En la sección *CODE* aparecen las sentencias del programa (no puede haber sentencias en la sección *DATA*).
- Un programa puede no tener ninguna sentencia, pero en cualquier caso es obligatorio que aparezca la palabra reservada *CODE*.
- En la sección *CODE* puede haber dos tipos de sentencias: escritura y asignación.
- La escritura (*print*) puede ser tanto de expresiones enteras como reales.
- Las expresiones podrán estar formadas por literales enteros, literales reales y variables combinados mediante operadores aritméticos (suma, resta, multiplicación y división).
- Se podrán agrupar expresiones mediante paréntesis.
- Las operaciones aritméticas deberán ser entre operandos del mismo tipo, no habiendo conversiones implícitas. Esto quiere decir que, por ejemplo, no se puede sumar un entero con un real.
- En las asignaciones, el tipo de la variable a asignar también deberá coincidir con el tipo del valor a ser asignado. Por tanto no se puede asignar un valor real a una variable entera ni a la inversa.