INTEGRACIÓN DE PROGRAMAS ANTLR CON C++

Jorge Mario Castro C.
jmcc500@msn.com

Resumen: Como alternativa a la monolítica combinación FLEX/Bison para la generación de analizadores léxicos y sintácticos, ANTLR brinda una poderosa y amistosa herramienta de generación de compiladores. Esta es también multilenguaje, pues tiene la capacidad de generar código para diversos lenguajes populares de programación orientados a objetos, como son Java, C#, C++, Python, Ruby, Javascript, entre otros. Debido a estas características ANTLR resulta muy práctico, dado que además es multiplataforma, pues está desarrollado en java.

Por otra parte, C++ es un lenguaje multiparadigma, que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

En este artículo se presenta y describe la forma en que se integran los programas ANTLR con C++ dado que su combinación puede ser de gran utilidad en el desarrollo de aplicaciones poderosas que requieran alguna especie de intérprete o compilador.

Palabras Clave: Compiladores - ANTLR - Programas - C++ - integración

1. Introducción

Hoy día los compiladores son elementos fundamentales para el desarrollo y construcción de bases de datos, aplicaciones empresariales, navegadores web y entornos de desarrollo, entre otros sistemas. Todo compilador está hecho para traducir de un lenguaje a otro, a partir de la gramática definida para el lenguaje.

Existen diversas herramientas para la generación de compiladores, ANTLR es una de ellas. Esta proporciona un marco de trabajo para la construcción de reconocedores, intérpretes, compiladores y traductores de lenguajes a partir de gramáticas enriquecidas con acciones. Entre sus principales funcionalidades destacan: la construcción de analizadores léxicos, analizadores sintácticos, mecanismos de construcción y recorrido de árboles de sintaxis abstracta (AST), mecanismos de tratamiento de plantillas y de detección y recuperación de errores [1].

En cuanto a C++, es un exitoso lenguaje imperativo y orientado a objetos. Este nació a partir de C, solo que con nuevas funcionalidades de las que C carecía, como: clases, plantillas, sobrecarga de operadores, entre otras. Estas características hacen de C++ un lenguaje muy versátil, largo y complejo [5].

Debido a las anteriores características de C++ y la amplia funcionalidad de ANTLR, resulta útil su integración para el desarrollo de aplicaciones con un lenguaje propio o al menos utilización de comandos complejos que requieren de su respectiva interpretación y traducción. Tales programas gozarían además de la poderosa gama de posibilidades que brinda C++ para programar. A pesar de esto, ANTLR es una herramienta la cual aún no ha sido ampliamente difundida. Prueba de esto, son la cantidad tan limitada o nula de artículos o publicaciones que describan o al menos hagan referencia a esta aplicación.

Por lo tanto, la intención del presente artículo es brindar información útil sobre ANTLR, en la cual se describa la forma en que se logra la integración con C++ y las ventajas o desventajas que se pueden obtener. La integración de ANTLRcon C++ se facilita a través del entorno de desarrollo propio de ANTLR: ANTLRWorks. Esta plataforma permite la construcción de gramáticas proporcionando representaciones gráficas de las expresiones y árboles generados. Además presenta una opción en la barra de herramientas para generar código C++, creando archivos .c y .h. Estos archivos generados se pueden agregar a proyectos C++ y utilizar configurando ciertas opciones que se discutirán luego más a fondo.

La distribución del artículo será la siguiente: en la segunda sección se describirán brevemente las distintas funcionalidades de ANTLR, en la tercera sección se explicará detalladamente el procedimiento de integración con C++ a través de un ejemplo sencillo. Los posibles problemas o restricciones con tal integración se especificarán en la cuarta sección y finalmente se analizarán las principales ventajas o desventajas de la integración en la última sección de conclusiones.

2. Principales funcionalidades de ANTLR

ANTLR es capaz de generar un analizador léxico, sintáctico o semántico en varios lenguajes, a partir de unos ficheros escritos en un lenguaje propio. Dicho lenguaje es básicamente una serie de reglas de gramática y un conjunto de construcciones auxiliares [6].

Los ficheros con los que trabaja ANTLR tienen la terminación *.g y se denominan ficheros de gramática. Un fichero de gramática contiene la definición de uno o varios analizadores. Cada uno de estos analizadores se puede traducir a código nativo (java, C++ o C#) en forma de clases. Es decir, por cada analizador descrito en el fichero de gramáticas se generará una clase [6].

En cuanto a los flujos de información, se definen diferentes tipos entre los niveles que permite ANTLR (léxico, sintáctico y semántico). Para la entrada de datos del exterior hasta la primera fase del compilador, se maneja un flujo de caracteres. Este generalmente proviene de un fichero, opero pueden usarse otras fuentes como una página web [6].

El flujo de información entre el analizador léxico y sintáctico es denominado flujo de *tokens*. Estos *tokens* se representan en ANTLR a

través de la súper clase *antlr.Token* que declara los métodos y atributos, que heredan otras sub clases como *ANTLR.CommonToken*. "*De esta forma, a pesar de ser casi una interfaz, esta clase nos da una idea de cómo se tratan los tokens en ANTLR: un token es un "tipo" (un entero) un texto (una cadena) y una línea y columna (sendos Enteros)"* [6]. Entonces, el analizador sintáctico se comunica con el léxico utilizando la función *nextToken* del analizador léxico [6].
Los árboles de sintaxis abstracta (AST) manejan la información semántica de un código, de manera que pueden especificar la relación jerárquica entre los símbolos de una gramática. La forma normal de representar un árbol en ANTLR utiliza una notación basada en LISP. Por ejemplo: #(A B C) representa un árbol con A en la raíz y B y C como hijos [6].

Finalmente, en cuanto al autómata recursivo descendente en ANTLR, es posible pasarle parámetros a los métodos que representan las reglas. A estas reglas se le añaden parámetros utilizando los corchetes. Los parámetros de una regla pueden utilizarse tanto en las acciones como en los predicados semánticos [6].

3. Procedimiento de integración de ANTLR con C++

A continuación se describe el procedimiento de integración de ANTLR con C++ mediante un pequeño ejemplo de una calculadora. La integración de ANTLR con C++ se logra fácilmente con el entorno de desarrollo ANTLRWorks. Para descargarlo se accede a la página principal de ANTLR: *www.antlr.org.* Hecho esto, se corre el programa y se define el tipo de documento a crear, en este caso se selecciona gramática. Luego se escoge un nombre y un tipo para la gramática que se quiere producir. En el ejemplo actual, se escogerá tipo *lexer* para poder generar *tokens* a partir de lexemas.

ANTLR genera un archivo con varias expresiones regulares que representan los diferentes lexemas para identificadores, hileras, comentarios y otros. Al hacer click en el lexema se puede observar en la parte inferior de la ventana la representación gráfica del lexema. Entonces, una vez definidos los lexemas, se puede generar código utilizando la barra de herramientas superior. Simplemente se hace click sobre el botón *Generate* y se desplegarán varias opciones. Se escoge la opción de generar código C++, se especifica el nombre igual al de la gramática declarada y se elije la carpeta de destino.

Hecho esto, Se generan dos archivos en la carpeta de destino, un .g que contiene los lexemas y una carpeta que contiene el código generado. Dentro de la carpeta se produce un .h y .c, en los cuales se declaran los métodos necesarios para trabajar sobre lo deseado y además con documentación interna para facilitar su uso.

Para la utilización de estos archivos se puede usar cualquier IDE para C++, en este caso se usará Code Blocks. Entonces, se crea un proyecto y se incluyen los archivos necesarios, o sea, los que tienen las rutinas definidas propias del programador y los generados por ANTLR.

Sin embargo, para compilar el proyecto son necesarias ciertas librerías. Estas se encuentran disponibles en la página principal de ANTLR. Para accederlas se

ingresa a Descargas y se buscan las "runtime libraries" de C. Hecho esto se agregan las carpetas *include* y *source* del archivo comprimido descargado a la carpeta de trabajo.

Luego se incluyen las dependencias de las carpetas agregadas y se eliminan tanto de la carpeta *source* como de la de *include* los archivos de depuración. De esta forma, ya se puede compilar de manera correcta. Es normal que aparezcan muchas advertencias debido a que se utiliza C con C++.

Para el caso actual de una calculadora, se necesitan definir los siguientes *tokens*: número, paréntesis izquierdo, paréntesis derecho, espacio en blanco y las operaciones aritméticas. Hecho esto se siguen los pasos descritos anteriormente para la generación de código en C++ e integración correcta de los archivos generados. Luego, simplemente se declaran variables para obtener los *tokens* y lexemas y se ejecuta el procedimiento para obtenerlos.

Ahora bien, para leer una hilera por ejemplo, se declara un flujo de entrada para almacenar los datos, un *lexer* de la clase *lexer* que *tokeniza* el flujo de entrada y almacena los *tokens* en el flujo de *tokens*. De este flujo de *tokens* se piden los lexemas y los tokens uno por uno.

Luego se llaman los constructores del flujo de entrada, *lexer* y flujo de *tokens* en orden respectivo. El flujo de entrada ocupa un puntero al texto para almacenarlo. El *lexer* ocupa crearse utilizando de parámetro el flujo de entrada. Finalmente el flujo de *token* ocupa inicializarse usando el *lexer* de parámetro.

Para obtener los *tokens* del flujo de *tokens*, se puede utilizar uno temporal. De esta forma se puede tener un puntero al *token* inicial en cada ciclo y posteriormente sacarlo del flujo de *tokens* para obtener uno nuevo para el siguiente ciclo. Al *token* temporal se le pide el valor con la función *getType* y el lexema con *getText*. De esta manera, se logra la integración con C++.

4. Problemas o restricciones de la integración

Dentro de los aspectos negativos de la integración con C++, se encuentra el hecho de la gran cantidad de librerías necesarias para un simple *tokenizador*. Estas producen además muchos *warnings* al incluirse, que pueden ser un problema para compilar. Por esta razón, se debe desactivar la opción de tratar los *warnings* como errores en las opciones de compilador de CodeBlocks.

Además, para tratarse de un ejemplo sencillo como el expuesto de la calculadora, resulta compleja su utilización. Esto debido más que nada a la versatilidad y orientación de ANTLR hacia la creación de reconocedores más complejos.

En general los problemas y dificultades no son muchos y son fáciles de solucionar.

5. Conclusiones

La utilización de ANTLR es sencilla. Su integración con C++ resulta poco complicada para una persona con conocimiento en la materia. Además existe una buena documentación técnica sobre su uso y funcionalidades en la página principal de ANTLR.

El procedimiento para lograr su integración con C++ consiste básicamente en declarar en la gramática que se quiere generar un archivo tipo C e incluir las librerías de ejecución. De esta forma, se pueden crear programas en C++ que utilicen analizadores léxicos, sintácticos o semánticos.

Por último es importante destacar la integración de ANTLR y C++ provee muchos beneficios que se pueden aprovechar en diversas áreas. Además resulta práctico y fácil de usar en comparación a flex o Lex, al cual supera en algunos aspectos, como la compatibilidad con Unicode y la posibilidad de generar código para diversos lenguajes.

6. Referencias Bibliográficas:

[1] - Salvador Gómez, Implementación mediante ANTLR y C# de un compilador y una máquina virtual para un lenguaje de script sencillo, http://www.sgoliver.net/blog/wp-content/uploads/2008/12/doc-fkscript-01.zip, 2008, fecha de consulta: 28-10-2009.

[2]- Erick Giron, Tutorial ANTLR y Phyton, http://objektblog.wordpress.com/2009/03/29/tutorial-antlr-y-python-i-introduccion-e-instalacion/, marzo 29 2009, 26 octubre 2009.

[3]- Ian Kaplan, Building a C++ ANTLR Parser, http://www.bearcave.com/software/antlr/antlr_build.html, 19 de julio de 1999, fecha de consulta: 27-10-2009.

[4]- http://www.clubdesarrolladores.com/articulos/categoria/11-cpp

[5]- http://www.zator.com/Cpp/E1_2.htm

[6] - Enrique José García Cota, Guía Práctica de ANTLR, http://www.lsi.us.es/~troyano/documentos/guia.pdf, septiembre del 2003, fecha de consulta: 1-11-2009.