República Bolivariana de Venezuela

Ministerio del Poder Popular para la Educación

Universidad José Antonio Páez

Facultad de Ingeniería

Escuela de Computación

Manual de Usuario del Compilador BCJ#

Traductores e Interpretadores

Integrantes:

-Carlos Cantero
- Bryan Porras
- José Castro

Introducción

Un compilador es simplemente un programa que traduce otros programas. Los compiladores clásicos traducen código fuente a código máquina ejecutable que tu ordenador puede entender.

Un compilador es un programa que traduce código escrito en un lenguaje de programación (llamado fuente) a otro lenguaje (conocido como objeto). En este tipo de traductor el lenguaje fuente es generalmente un lenguaje de alto nivel y el objeto un lenguaje de bajo nivel, como assembly o código máquina. Este proceso de traducción se conoce como compilación.

Los principios y técnicas que se usan en la escritura de compiladores se pueden emplear en muchas otras áreas. Se basan en los conceptos de teoría de autómatas y lenguajes formales que se están exponiendo en la parte teórica, y constituyen un campo de aplicación práctica bastante directo.

Nosotros desarrollaremos un compilador en el lenguaje de programación Java, tendrá como nombre "BCJ#", implementando el análisis léxico, sintáctico y semántico mediante las librerías JFlex y CompilerTools. Así mismo, veremos como ejecutar el código que inventamos para nuestro lenguaje.

Librerías

Jflex (Para el análisis léxico)

Es una librería escrita en java que permite generar analizadores léxicos. Es una reimplementación de JLex con más funcionalidades y mayor compatibilidad con algunos analizadores sintácticos. Toma como entrada reglas de análisis léxico y devuelve como salida el código fuente en java implementando un autómata finito que reconoce los lenguajes regulares.

Objetivos de diseño

- Compatibilidad con Unicode.
- Escáneres generados rápidamente.
- Rápida generación de escáneres.
- Sintaxis de especificación conveniente.
- Independencia de la plataforma.
- Compatibilidad JLex.

CompilerTools (Para el análisis sintáctico y semántico):

El paquete javax.tools proporciona interfaces y clases para trabajar con el compilador de Java y se puede invocar desde un programa durante la ejecución.

Proporciona un marco que permite a los clientes ubicar y ejecutar compiladores desde su propio código de aplicación. También proporciona una interfaz de proveedor de servicios para acceso estructurado a diagnósticos y abstracción de archivos para anular el acceso a archivos.

Características clave:

- Implementa un autocompletado de código en un JTextComponent.
- Extrae los bloques de código del lenguaje inventado para su posterior ejecución.
- Cuenta con una clase para la manipulación de archivos, soportando operaciones como Nuevo, Abrir, Guardar y Guardar Como.
- Se pueden crear errores ya sean Léxicos, Sintácticos, Semánticos o Lógicos.
- Tiene diversas funciones automatizadas que realizan algunas operaciones complejas para la programación del compilador.
- Se pueden crear gramáticas mediante la agrupación de Tokens a través de Producciones.
- Implementa un contador de número de línea en un JTextComponent.
- Cuenta con una función para el coloreo de palabras en un JTextComponent.
- Implementa el almacenamiento de tokens y sus atributos básicos, los cuales son lexema, componente léxico, línea y columna.

Variables básicas de comentarios y espacios

Nombre	Ejemplo
TerminadorDeLinea	\r |\n |\r\n
EntradaDeCaracter	[^\r\n]
EspacioEnBlanco	{TerminadorDeLinea} | [\t\f]
ComentarioTradicional	"/*" [^*] ~"*/" | "/*" "*"+ "/"
FinDeLineaComentario	"//"{EntradaDeCaracter}*{TerminadorDeLinea}?
ContenidoComentario	([^*] | *+ [^/*]) *
ComentarioDeDocumentacion	"/**" {ContenidoComentario} "*"+ "/"

Comentario

Nombre	Ejemplo
Comentario	{ComentarioTradicional} | {FinDeLineaComentario} | {ComentarioDeDocumentacion}
Comentarios o espacios en blanco	{Comentario} | {EspacioEnBlanco} { /*Ignorar*/ }

Tipo de datos

Nombre	Ejemplo
Letra	[A-Za-zÑñ_ÁÉÍÓÚáéíóúÜü]
Digito	[0-9]
Identificador	{Letra} ({Letra} | {Digito})* \${Identificador}
Numero	0 | [1-9] [0-9]* {Numero}
Colores	#[{Letra}{Digito}]{6}
Operadores de agrupación	"(" , ")" , "{" , "}"
Signos de puntuación	"," ";"

Funciones

Métodos	Funciones
Movimiento ()	adelante atrás izquierda derecha norte sur este oeste
Pintar()	pintar
Detener pintar()	detenerpintar
Tomar()	tomar poner
Lanzar Moneda()	lanzarmoneda
Ver()	izquierdaEsObstáculo | izquierdaEsClaro | izquierdaEsBaliza | izquierdaEsBlanco | izquierdaEsNegro | frenteEsObstáculo | frenteEsClaro | frenteEsBaliza | frenteEsBlanco | frenteEsNegro | derechaEsObstáculo | derechaEsClaro | derechaEsBaliza | derechaEsBlanco | derechaEsNegro

Operadores Lógicos

BCJ#	Java
&	&& (AND)
|	! (NOT)
→	== (Igual)

Estructuras de control

BCJ#	Java
Repetir	While
repetirMientras	While
Interrumpir	Break
Si	If
Sino	If
bcj	Java

Final y posibles errores

Nombre	Ejemplo
Final	Final()
Posibles Errores	Número erróneo 0 {Numero}+ Identificador sin $ {Identificador} { "ERROR_2"}.{ "ERROR"}

Interfaz gráfica

Componentes

Área de texto: en esta área se escribe el código en lenguaje bcj que se desea compilar en caso de que se desee generar un archivo nuevo directamente desde el programa.

Consola: consola que muestra los mensajes del proceso de compilación (fin del proceso y errores).

Tabla de símbolo: Tabla donde se muestra los componentes léxicos con su respectivo lexema

Componente léxico	Lexema	[Línea, Columna]

Botones

Botón nuevo: botón que permite generar un nuevo archivo bcj.

Botón abrir: botón que permite abrir un archivo bcj ya existente.

Botón guardar: botón que guarda en el archivo lo escrito en el área de texto.

Botón guardar como: botón que guarda lo escrito en el área de texto con otro nombre de archivo.

Botón compilar: botón para compilar el código del archivo.

Botón ejecutar: botón que ejecuta el código compilado en JAVA.

Botón de cerrar: botón que permite cerrar en programa.

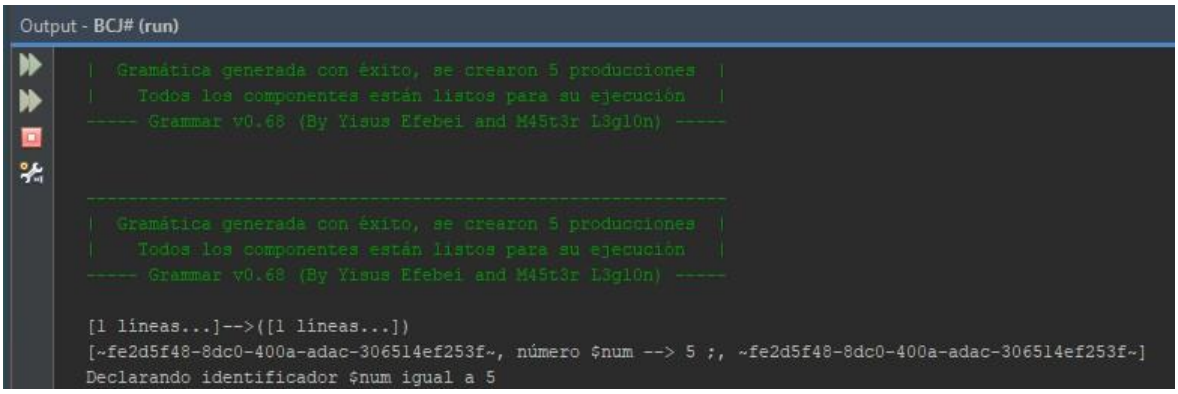
Como usar el compilador

1) Abrir el programa y seleccionar el archivo que desea compilar con el botón abrir, en caso de no tener un archivo crear y pegar el código que desea compilar en el área de texto.

2) Una vez listo el archivo a compilar, pulsar la opción compilar para iniciar el proceso de compilación.

3) Tras compilar el programa enviara un mensaje de culminación si no encuentra ningún error léxico, sintáctico o semántico y muestra la tabla de símbolos.

4) Una vez compilado el código podemos ejecutar en java con el botón ejecutar y que se muestre en la consola el código.


Output - BCJ# (run)
| Gramática generada con éxito, se crearon 5 producciones |
| Todos los componentes están listos para su ejecución |
----- Grammar v0.68 (By Yisus Efebei and M45t3r L3g10n) -----
| Gramática generada con éxito, se crearon 5 producciones |
| Todos los componentes están listos para su ejecución |
----- Grammar v0.68 (By Yisus Efebei and M45t3r L3g10n) -----
[1 lineas...]-->([1 lineas...])
[~fe2d5f48-8dc0-400a-adac-306514ef253f~, número $num --> 5 ;, ~fe2d5f48-8dc0-400a-adac-306514ef253f~]
Declarando identificador $num igual a 5