



Procesadores de Lenguajes

Ingeniería Informática

Primer curso de segundo ciclo
Segundo cuatrimestre



Departamento de informática y análisis numérico
Escuela Politécnica Superior
Universidad de Córdoba.
Curso académico: 2011-2012

Trabajo de prácticas

1. Objetivo

- Desarrollar un **intérprete** con ANTLR y Java que permita la simulación de un **entorno definido por el usuario**
 - A título de ejemplo, se propone el entorno del **Mundo Wumpus** (véase el documento adjunto).
 - Sin embargo, cada **grupo** podrá proponer y desarrollar la simulación del **entorno** que **desea**
- El intérprete habrá de incluir análisis léxico, análisis sintáctico y análisis semántico, así como la correspondiente ejecución de las diferentes órdenes interpretadas.
- La simulación podrá ser en modo texto o bien en modo gráfico
 - En dos dimensiones (panel o frame: Flash, JAVA2D)
 - o bien en tres dimensiones (VRML/X3D, JAVA3D)

2. Consideraciones generales

- El trabajo se podrá realizar en grupos compuestos por un máximo de tres personas.
- El trabajo se deberá “subir” a la plataforma de moodle” antes de las 9:00 horas el miércoles 6 de junio de 2012.

3. Características generales del intérprete

- El intérprete deberá permitir
 - La **simulación del entorno propuesto** por cada grupo
 - En el caso del **Mundo Wumpus**,
 - La configuración del tablero
 - Tamaño
 - Ubicación de los elementos
 - Tesoro
 - Wumpus
 - Pozos
 - Entrada y salida
 - Número de flechas del aventurero
 - Las acciones del aventurero
 - Moverse

- Disparar la flecha, si tiene
 - Recoger el tesoro
 - Etc.
- La ejecución de sentencias de un **lenguaje de programación en pseudocódigo** que permita controlar las acciones del entorno simulado
 - Este lenguaje de programación en pseudocódigo es descrito en el apartado 4.
- **Control de errores**
 - La comunicación de errores es imprescindible para que el usuario perciba tanto la correcta como errónea evolución del intérprete.
 - Debe mostrarse toda la información que sea posible
 - número de línea, error y causas posibles
 - Debe recuperarse del error en la medida de lo posible.
 - El intérprete deberá controlar los errores que sea capaz detectar
 - Léxicos:
 - Identificador mal escrito
 - Utilización de símbolos no permitidos
 - Etc.
 - Sintácticos:
 - Sentencias de control más escritas.
 - Sentencias con argumentos incompatibles.
 - Etc.
 - Semánticos o lógicos:
 - Sentencia “para” que pueda generar un bucle infinito.
 - Etc.
- **Observaciones**
 - **AST**
 - Se valorará la utilización de árboles de sintaxis abstracta AST, pero no será obligatorio.
 - **Makefile**
 - Se deberá crear un fichero makefile que permita generar el intérprete.
- **Formas de ejecución del intérprete**
 - Interactiva:
 - Se evaluarán “una a una” las sentencias que se introduzcan por el teclado
 - A partir de un fichero de entrada:
 - El fichero debe estar compuesto por las sentencias que se deben ejecutar

4. Lenguaje de programación en pseudocódigo

- **Componentes léxicos o tokens**
 - **Comentarios**
 - De una línea: comenzará con el símbolo #
 - Ejemplo: *# Comentario de una línea*
 - De varias líneas
 - Estará delimitado por llaves { ... }

- Ejemplo

{

*Ejemplo maravilloso
de comentarios
con varias líneas*

 }

- **Número:**
 - Desde enteros a reales con notación científica
 - Todos serán tratados como si fueran del mismo tipo: número
 - Ejemplos: 2, 2.5, 2.5e+7, 0.3e-4, 5.2e5
- **Cadenas de caracteres**
 - Se escribirán con comillas dobles, pero las comillas no se almacenarán como parte de la cadena.
 - Ejemplos: "cadena \"maravillosa\"", "a", "102"
- **Identificadores:**
 - Podrán estar compuestas por letras, subrayados y dígitos, pero deberán comenzar por un letra, no podrá haber dos subrayados seguidos y no podrán terminar en subrayado.
 - No se distinguirá entre mayúsculas y minúsculas
 - Ejemplos:
 - iva, IVA, Iva (todos serán considerados iguales)
 - Fin_de_mes
 - Saldo_total
 - **Importante**
 - Los identificadores deberán ser declarados antes de su utilización.
 - Cada identificador podrá ser de un único tipo
 - Número
 - Cadena
- **Palabras reservadas**
 - Las que sean necesarias para definir las reglas gramaticales del análisis sintáctico
 - No se distinguirá entre mayúsculas y minúsculas
 - Las palabras reservadas no se podrán utilizar como identificadores.
 - Ejemplo: Para, PARA, para (todos serán considerados iguales)
- **Operadores lógicos:**
 - negación lógica: _no
 - conjunción lógica: _y
 - disyunción lógica: _o
- **Operadores relacionales**
 - igual: ==
 - mayor: >
 - mayor que: >=
 - menor que: <
 - menor igual: <=
 - distinto: <>
- **Operador de asignación**
 - :=
- **Operadores ariméticos**

- suma:
 - binario: $2 + 3$
 - unario: $+2$
 - resta
 - binario: $2 - 3$
 - unario: -2
 - Producto: $*$
 - División: $/$
 - Potencia: $^$ (asociatividad por la derecha y máxima precedencia)
- **Operadores de manejo de cadenas**
 - Concatenación: $+$
 - Ejemplo:
 - *“Hola ” + nombre*
 - donde *nombre* es una variable de tipo cadena
- **Sentencias**
 - **Observación**
 - Todas las sentencias deben terminar en punto y coma “;”
 - **Declaración de variables o identificadores**
 - **Número**
 - Ejemplo
 - *Numero a, b, c;*
 - **Cadena**
 - Ejemplo
 - *Cadena nombre, apellido1, apellido2;*
 - **Sentencias de lectura**
 - leer (identificador):
 - permite leer un número y asignárselo al identificador de tipo numérico
 - Ejemplo
 - *leer (edad);*
 - leer(identificador_cadena):
 - permite leer una cadena “sin comillas” y asignársela al identificador de tipo cadena
 - Ejemplo
 - *leer(nombre);*
 - **Sentencias de escritura**
 - escribir: permite escribir un número numérica
 - Ejemplo:
 - *escribir(2*a)*
 - donde “a” es una variable numérica
 - escribir: permite escribir una expresión alfanumérica
 - Ejemplo
 - *escribir(“Giro a la”+ movimiento)*
 - donde “movimiento” es una variable de tipo cadena
 - Observación:
 - Se valorará la interpretación de comandos de saltos de línea ($\backslash n$) y tabuladores ($\backslash t$) que puedan aparecer en la expresión alfanumérica.

- **Asignación:**
 - `identificador_numérico := expresión numérica`
 - Ejemplo:
 - `edad := 12;`
 - `identificador_cadena := expresión alfanumérica`
 - `nombre := "Juan Pablo" + " " + "Luna" + " " + "Aguirre";`
 - **Importante:**
 - Las variables o identificadores deberán haber sido declarados previamente del tipo correspondiente.
- **Sentencias de control**
 - **Sentencia condicional simple**
si condición
entonces sentencias
fin_si;
 - **Sentencia condicional compuesta**
si condición
entonces sentencias
si_no sentencias
fin_si;
 - **Bucle “mientras”**
mientras condición hacer
sentencias
fin_mientras;
 - **Bucle “repetir”**
repetir
sentencias
hasta condición;
 - **Bucle “para”**
para identificador
desde expresión numérica 1
hasta expresión numérica 2
paso expresión numérica 3
hacer
sentencias
fin_para;

5. Documentación

- **Aspectos formales**
 - Portada
 - Título del documento
 - Número de documento
 - Nombre y apellidos de las personas que forman el grupo
 - Nombre de la asignatura: Procesadores de lenguaje

- Nombre de la Titulación: Ingeniería informática
 - Curso: primer curso de segundo ciclo
 - Curso académico: 2011 – 2012
 - Escuela Politécnica Superior de Córdoba
 - Universidad de Córdoba
 - Fecha
 - Las páginas deberán estar numeradas.
 - Se valorará la corrección ortográfica y la calidad en la redacción.
- **Contenido de la documentación**
- Portada
 - Índice
 - Introducción
 - Descripción del entorno que se va a simular con el intérprete
 - Fases del proceso de generación del intérprete
 - Definición del lenguaje diseñado
 - Sentencias o instrucciones para simular el entorno elegido
 - Sentencias del lenguaje de programación de pseudocódigo
 - Descripción de la gramática asociada al lenguaje definido.
 - Descripción del intérprete ANTLR construido para la gramática.
 - Analizador léxico: componentes léxicos
 - Analizador sintáctico:
 - Símbolos no terminales
 - Reglas de producción de la gramática
 - Análisis semántico
 - Atributos: heredados y sintetizados
 - Funciones auxiliares
 - Tabla de símbolos
 - Elementos auxiliares para la simulación del entorno diseñado
 - Etc.
 - Modo de obtención del intérprete: descripción del fichero makefile
 - Nombre y descripción de cada uno de los ficheros utilizados.
 - Modo de generación de intérprete.
 - Modo de ejecución del intérprete:
 - Interactiva
 - A partir de un fichero
 - Ejemplos:
 - Se valorará la cantidad de ejemplos propuestos
 - Al menos, se deberán proponer cinco ejemplos
 - Conclusiones:
 - Reflexión sobre el trabajo realizado
 - Puntos fuertes y puntos débiles del simulador desarrollado.
 - Bibliografía o referencias web
 - Anexos
 - Al menos, se debe incluir el código desarrollado
 - Se valorará que se haya documentado el código con **javadoc**

6. Evaluación del trabajo

○ Condiciones básicas

- La **correcta** realización del trabajo de prácticas es **imprescindible** para poder aprobar la asignatura.
- El intérprete deberá funcionar correctamente bajo cualquier plataforma, pues se debe utilizar la versión estándar de Java.
- La gramática diseñada no deberá tener ningún conflicto.
- Los alumnos podrán exponer el trabajo realizado al profesor tanto si ellos lo desean como si el profesor lo solicita.

○ Evaluación continua:

- Cada grupo deberá presentar un informe **quincenal** indicando la evaluación del trabajo realizado.
- Este informe permitirá conocer, revisar y sugerir propuestas de mejora del trabajo .
- En dicho informe, se deberá indicar
 - Portada
 - Índice
 - Las tareas desarrolladas.
 - Las dificultades encontradas y las soluciones adoptadas, en su caso.
 - Las tareas pendientes.
 - Etc.

○ Evaluación final

- Se subirá a la plataforma de “*moodle*”, antes de las 9:00 horas del miércoles 6 de junio de 2012, la siguiente información:
 - Código del intérprete desarrollado
 - Documentación final (descrita en el apartado 5)

○ Criterios de evaluación

- La calificación final se obtendrá a partir de
 - La documentación: 40 %
 - Se tendrán en cuenta los documentos presentados durante la evaluación continua y la documentación final.
 - Se valorará la realización de esquemas o gráficos de elaboración propia que faciliten la comprensión de la documentación.
 - Software (60 %)
 - El intérprete deberá funcionar correctamente
 - Además, se tendrán en cuenta
 - Originalidad
 - Completitud y complejidad
 - Diseño del lenguaje y la gramática.
 - Dificultades en la elaboración del trabajo que hayan sido convenientemente documentadas.
 - Aportaciones propias del grupo tanto en contenidos como en la gramática.
 - Número de ejemplos propuestos
 - Uso de AST

- Documentación del código con javadoc