Procesadores de lenguajes

Ingeniería Informática
Primero curso de Segundo ciclo
Segundo cuatrimestre

Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

Curso académico: 2012 - 2013

TRABAJO DE PRÁCTICAS

1. Introducción

- Se debe utilizar **ANTLR** y **Java** para desarrollar dos actividades
  a) Elaboración de un ***i**ntérprete de **p**seudocódigo en **e**spañol*:
     - *ipe*.
  b) Ampliación del intérprete "ipe" para que permita simular la ejecución de sentencias predefinidas de un entorno o juego
     - Por ejemplo: extensión del Mundo Wumpus

- Descripción de los apartados:
  2) Elaboración y entrega del trabajo
  3) Características del lenguaje de pseudocódigo
  4) Propuestas de entornos o juegos de simulación
  5) Control de errores
  6) Modos de ejecución del intérprete
  7) Documentación del trabajo
  8) Criterios de evaluación

2. Elaboración y entrega

- Modo de realización del trabajo
  - El trabajo se podrá realizar de
    - forma individual
    - o en grupo compuesto por un máximo de tres personas

- Modo de entrega
  - Un fichero comprimido deberá ser "subido" a la tarea de la plataforma de "moodle"
  - Dicho fichero deberá contener los siguientes ficheros
    - Documentación del trabajo (véase el apartado nº 7)
    - Ficheros de ANTLR

        - Ficheros de Java
        - Fichero makefile
        - Al menos dos ficheros de ejemplo con la extensión ".e"
            - Uno de los ejemplos deberá ser el fichero "ejemplo-1.e", proporcionado por el profesor

- **Plazo de entrega**
    - Hasta las 9:00 horas del viernes 7 de junio de 2013.

3. Características de lenguaje de pseudocódigo

a) Componentes léxicos o *tokens*

- **Palabras reservadas**
    - *__mod, __o, __y, __no, leer, leer_cadena, escribir, escribir_cadena, si, entonces, si_no, fin_si, mientras, hacer, fin_mientras, repetir, hasta, para, desde, paso, fin_para, borrar, lugar*
    - No se distinguirá entre mayúsculas ni minúsculas.
    - Las palabras reservadas no se podrán utilizar como identificadores.

- **Identificadores de variables**
    - Características
        - Estarán compuestos por una serie de letras, dígitos y el subrayado;
        - Deben comenzar por una letra,
        - No podrán acabar con el símbolo de subrayado, ni tener dos subrayados seguidos.
    - Identificadores válidos:
        - *dato, dato_1, dato_1_a*
    - Identificadores **no** válidos:
        - *_dato, dato_, dato__1*
    - No se distinguirá entre mayúsculas ni minúsculas.

- **Número**:
    - Se utilizarán números enteros, reales de punto fijo y reales con notación científica.
    - Todos ellos serán tratados conjuntamente como números.

- **Cadena**:
    - Estará compuesta por una serie de caracteres delimitados por comillas simples:
      'Ejemplo de cadena'
    - Deberá permitir la inclusión de la comilla simple utilizando la barra (\):
      'Ejemplo de cadena con \' comillas\' simples'.

- **Nota**:
  - Las comillas exteriores no se almacenarán como parte de la cadena.

- **Operadores aritméticos:**
  - suma: +
    - Unario: +2
    - Binario: 2+3
  - resta: -
    - Unario: -2
    - Binario: 2-3
  - producto: *
  - división: /
  - módulo: __mod
  - potencia: **

- **Operador alfanumérico:**
  - concatenación: ||

- **Operadores relacionales de números y cadenas:**
  - menor que: <
  - menor o igual que: <=
  - mayor que: >
  - mayor o igual: >=
  - igual que: ==
  - distinto que: <>
  - Por ejemplo:
    - si *A* es una variable numérica y *control* una variable alfanumérica, se pueden generar las siguientes expresiones relacionales:
      (A >= 0)
      (control <> 'stop')

- **Operadores lógicos:**
  - disyunción lógica: __*o*
  - conjunción lógica: __*y*
  - negación lógica: __*no*
    - Por ejemplo:
      (A >= 0) __y __no (control <> 'stop')

- **Comentarios**
  - De varias líneas: delimitados por llaves

    { ejemplo maravilloso
    de comentario
    de tres líneas }

  - De una línea: todo lo que siga al carácter # hasta el final de la línea.

 # ejemplo espectacular de cometario de una línea

- **Punto y coma**
  - Se utilizará para indicar el fin de una sentencia.

b) Sentencias

- **Asignación**
  - *identificador = expresión numérica*
    - Declara a *identificador* como una variable numérica y le asigna el valor de la expresión numérica.
    - Las expresiones numéricas se formarán con números, variables numéricas y operadores numéricos.
    - Ejemplo:
      Edad = 12;

  - *identificador = expresión alfanumérica*
    - Declara a *identificador* como una variable alfanumérica y le asigna el valor de la expresión alfanumérica.
    - Las expresiones alfanuméricas se formarán con cadenas, variables alfanuméricas y el operador alfanumérico de concatenación
    - Ejemplo
      nombre = 'Ana' + ' '+ 'Luna';

- **Lectura**
  - ***Leer** (identificador)*
    - Declara a *identificador* como variable numérica y le asigna el número leído.

  - ***Leer_cadena** (identificador)*
    - Declara a *identificador* como variable alfanumérica y le asigna la cadena leída (sin comillas).

- **Escritura**
  - ***Escribir** (expresión numérica)*
    - El valor de la expresión numérica es escrito en la pantalla.

  - ***Escribir_cadena** (expresión alfanumérica)*
    - *La cadena (sin comillas exteriores) es escrita en la pantalla.*

- *Se debe permitir la interpretación de comandos de saltos de línea (\n) y tabuladores (\t) que puedan aparecer* en la expresión alfanumérica.

- **Sentencias de control**[1]
  - 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[2] "*para*"

    para *identificador*
    desde *expresión numérica 1*
    hasta *expresión numérica 2*
    paso *expresión numérica 3*
    hacer
    sentencias
    fin_para

- **Comandos especiales**
  - ***Borrar***: borra la pantalla
  - ***Lugar***(*expresión numérica1, expresión numérica2*)
    - Coloca el cursor de la pantalla en las coordenadas indicadas por los valores de las expresiones numéricas.

4. Entornos o juegos de simulación

- Cada grupo deberá definir el conjunto de sentencias predefinidas que permitan simular un entorno o juego.

[1] Una *condición* será una *expresión relacional* o una *expresión lógica compuesta*.
[2] Se valorará que se controlen los pasos con incrementos positivos y negativos del bucle "*para*".

- A modo de ejemplo, se propone una **extensión del Mundo Wumpus** (véase el documento adjunto), que deberá incorporar las siguientes **novedades**
    - El aventurero tendrá un número inicial de "vidas". Por defecto, el valor inicial será 1.
    - En el tablero habrá nuevos elementos
        - Mina
            - Si el aventurero accede a una celda con una mina, perderá una vida; si no le quedan vidas, morirá
        - Flecha
            - Si accede a una celda con una flecha, la recogerá.
        - Ambrosía
            - Habrá celdas que tengan ambrosía (elixir de la vida)
            - Si el aventurero accede a una de estas celdas, tomará la ambrosía e incrementará en uno su número de vidas
        - En una celda, solamente podrá haber un elemento: pozo, mina, flecha o tesoro.
    - El Wumpus podrá moverse
        - Si accede a una celda con una flecha, la romperá
        - Si accede a una celda con ambrosía, se la tomará
        - Es inmune a las minas.
        - Se le impedirá acceder a las celdas que contengan un pozo
        - No se dará cuenta de si el tesoro está en una celda
        - Etc.
    - Se deberán simular sentencias que permitan:
        - La configuración del tablero
            - Tamaño
            - Ubicación de los elementos
                - Wumpus
                - Tesoro
                - Pozos, minas
                - Flechas, ambrosía
            - Entrada y salida
            - Número de flechas del aventurero
            - Número de vidas del aventurero
            - Las acciones del aventurero
                - Moverse
                - Disparar la flecha, si tiene
                - Recoger el tesoro, una flecha o ambrosía
            - Etc.

- También se pueden simular **otros entornos alternativos**, como, por ejemplo:

    - Simulador de un juego de mesa
        - *Ajedrez*
        - *Damas*

        - *Go*
        - *Reversi*
        - *Hundir la flota*
        - *Papel, piedra o tijera*
            - En un tablero se colocan tres objetos: papel, piedra o tijera
            - Por turnos, se van colocados estos objetos de forma que
            - Si una piedra es delimitada por dos papeles (en diagonal, vertical u horizontal) entonces se convierte en papel (el papel "envuelve" a la piedra).
            - Si una papel es delimitada por dos tijeras (en diagonal, vertical u horizontal) entonces se convierte en tijera (la tijera "corta" al papel).
            - Si una tijera es delimitada por dos piedras (en diagonal, vertical u horizontal) entonces se convierte en piedra (la piedra "rompe" las tijeras).
            - Cuando el tablero está lleno, gana el objeto que tenga más elementos
    - El "encuentro"
        - En un tablero se colocan dos personajes, por ejemplo, un/a elfo/a y un hombre/mujer de las montañas.
        - Cada personaje se mueve por el tablero en busca del otro personaje y, si lo encuentra, gana la partida.
        - En el tablero hay minas, pozos, etc., que pueden provocar la muerte.
        - Etc.
    - *Canal de Panamá*
        - Simulador de navegación de barcos por un canal como el de Panamá, que tiene esclusas
        - Un canal tiene dos puntos de entrada y salida
        - Los barcos pueden entrar o salir por cada punto de entrada y salida
        - El canal está dividido en esclusas, que se pueden abrir o cerrar.
        - Si la esclusa "n" está abierta entonces las esclusas n-1 y n +1 deben estar cerradas. Se deben tener en cuenta las esclusas iniciales o finales.
        - Etc.
    - *Cámaras de seguridad*
        - Simulador de un sistema de control con cámaras de un recinto dividido en zonas.
        - Las cámaras se pueden encender, apagar y girar (arriba – abajo, izquierda - derecha)
    - *Editor gráfico de figuras geométricas:*
        - Polígonos, círculos, elipses, etc.
    - *Simulador de una carrera por etapas*
        - Carrera ciclista

        - Rally
    - *Gestor de contenidos*
        - Consigna de documentos
        - Mensajería electrónica
    - *Gestor de una ruta de senderismo*
        - Puntos de salida y llegada, etapas, senderistas, etc.

- **Importante**
    - Además, cada grupo podrá **proponer** y desarrollar la simulación de otro entorno o juego alternativo si cuenta con el visto bueno del profesor**.**

5. Control de errores

El intérprete deberá controlar toda clase de errores:

- **Léxicos**:
    - Identificador mal escrito
    - Utilización de símbolos no permitidos
    - Etc.
- **Sintácticos:**
    - Sentencias de control más escritas.
    - Sentencias con argumentos incompatibles.
    - Etc.
    - Observación
    - Se valorará la utilización de producciones de error.
- **Semánticos**
    - Argumentos u operandos incompatibles
- **De ejecución**:
    - Sentencia "para" que pueda generar un bucle infinito.
    - Fichero de entrada inexistente o con una extensión incorrecta.
    - Etc.

- **Observación**
    - Debe mostrarse toda la información que sea posible
        - Número de línea, error y causas posibles

6. Modos de ejecución del intérprete

El intérprete se podrá ejecutar de dos formas diferentes:

- Modo interactivo:
    - Se ejecutarán las instrucciones tecleadas desde un terminal de texto

      ipe.exe
      > …

- Interpretando las sentencias de un fichero
    - El fichero deberá tener la extensión ".e"

      ipe.exe ejemplo.e

- **Observaciones**
  - El intérprete deberá funcionar correctamente en "ThinStation" de la Universidad de Córdoba
  - La gramática **no** deberá tener ningún conflicto.

7. Documentación del trabajo

- Portada
  - Título del trabajo desarrollado
  - Nombre y apellidos de las personas que forman el grupo
  - Nombre de la asignatura: Procesadores de lenguaje
  - Nombre de la Titulación: Ingeniería informática
  - Curso: tercer curso
  - Curso académico: 2012 – 2013
  - Escuela Politécnica Superior de Córdoba
  - Universidad de Córdoba
  - Fecha
- Índice
  - Las páginas deberán estar numeradas.
- Introducción
  - Breve descripción del trabajo realizado y de las partes del documento
- Lenguaje de pseudocódigo
  - Componentes léxicos
  - Sentencias
- Lenguaje del entorno o juego de simulación elegido
- Tabla de símbolos
  - Descripción
- Análisis léxico
  - Expresiones regulares utilizadas para definir los componentes léxicos
- Esquema de traducción
  - Símbolos terminales (componentes léxicos)
  - Símbolos no terminales
  - Reglas de producción de la gramática
  - Atributos heredados o sintetizados
  - Acciones semánticas:
    - Se deberán describir las acciones semánticas de las producciones que generan las sentencias de control y especialmente las diseñadas para los bucles "repetir" y "para".
- Elementos auxiliares para la simulación del entorno elegido
- Modo de obtención del intérprete
  - Nombre y descripción de cada fichero utilizado
  - Descripción del fichero makefile
- Modo de ejecución del intérprete:
  - Interactiva

    - A partir de un fichero
- Ejemplos:
    - Se valorará la cantidad de ejemplos propuestos
    - Al menos, se deberán proponer tres ejemplos
        - Fichero de ejemplo propuesto por el profesor para simular sentencias del lenguaje de pseudocódigo
        - Dos ficheros de ejemplo propuestos por el autor o autores
            - Uno de los ejemplo solamente tendrá sentencias del lenguaje de pseudocódigo.
            - El otro ejemplo tendrá además sentencias del entorno o juego de simulación elegido (por ejemplo: extensión del Mundo Wumpus).
        - Se valorará "fundamentalmente" la originalidad y complejidad de estos ejemplos.
- Conclusiones:
    - Reflexión sobre el trabajo realizado
    - Puntos fuertes y puntos débiles del intérprete desarrollado.
- Bibliografía o referencias web
- Anexos

8. Criterios de evaluación

- **Informes de evaluación (10 %)**
    - Cada semana, el profesor irá elaborando un "**informe de evaluación**" que servirá para conocer los progresos de cada grupo:
        - Partes desarrolladas
        - Documentación elaborada
        - Dificultades encontradas y soluciones adoptadas
        - Etc.
    - Este informe permitirá conocer, revisar y sugerir propuestas de mejora del trabajo.
    - **Observación**
        - La asistencia a las clases de práctica se tendrá en cuenta en los informes de evaluación.

- **Documentación**: 40 %
    - Se tendrá en cuenta lo indicado en el apartado nº 7
    - El código elaborado deberá estar documentado con **Javadoc.**
    - Se valorará la inclusión de gráficos o figuras.
    - También se valorará la corrección ortográfica y la calidad en la redacción.

- **Funcionamiento del intérprete (software):** 50 %
    - La gramática diseñada **no** deberá tener ningún conflicto.
    - El intérprete deberá funcionar correctamente en **ThinStation** tanto de forma interactiva como ejecutando la instrucciones de un fichero.

- Los ejemplos deberán funcionar correctamente.
- Tipo de simulación del entorno o juego
  - La simulación podrá ser en modo texto
  - La simulación también podrá ser en modo gráfico (**opcional**)
    - En dos dimensiones (panel o frame: Flash, JAVA2D)
    - o bien en tres dimensiones (VRML/X3D, JAVA3D)

- **Otros criterios de evaluación que podrán mejorar la calificación**
  - Utilización de árboles de sintaxis abstracta: **AST**
  - Originalidad
  - Diseño del lenguaje y la gramática.
  - Completitud del trabajo realizado.
  - Ampliación del lenguaje de pseudocódigo.
  - Soluciones a dificultades encontradas durante la elaboración del trabajo que hayan sido convenientemente documentadas.
  - Aportaciones propias del grupo.
  - Número y complejidad de los ejemplos propuestos

- **Observación**
  - Los/as alumnos/as podrán exponer el trabajo realizado al profesor tanto si ellos/as lo desean como si el profesor lo solicita.