TEORÍA DE AUTÓMATAS Y LENGUAJES FORMALES

Ingeniería Técnica en Informática de Sistemas
Segundo curso

Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba
Curso académico 2012 – 2013

TRABAJO DE PRÁCTICAS

Convocatoria de diciembre de 2012

- **Observación**
  - El enunciado del trabajo de prácticas coincide con el propuesto en el último curso en el que la asignatura tuvo docencia:
    - Curso académico 2010 – 2011
  - El trabajo se podrá hacer de forma individual o por parejas
  - Entrega:
    - Fecha:
      - ✓ Hasta las 14:00 horas del lunes 3 de diciembre de 2012
    - Modo de entrega
      - ✓ Subida de un fichero comprimido "zip" a la tarea de moodle

---

- El objetivo de la práctica de esta asignatura es diseñar e implementar un "**traductor" de pseudocódigo** utilizando las herramientas **LEX y YACC**.

- EL trabajo práctico se dividirá en cuatro partes o tareas:
  1. Especificación del lenguaje fuente
  2. Construcción del analizador léxico
  3. Construcción del analizador sintáctico
  4. Generación de código

1. Especificación del lenguaje fuente

- La primera tarea consistirá en **redactar** una especificación del pseudocódigo que se va a utilizar.
- Como ejemplo se puede utilizar la especificación detallada en http://pseint.sourceforge.net/pseudocodigo.php, aunque los estudiantes podrá modificarla como consideren más oportuno.

- Para **aprobar** la práctica, el pseudocódigo deberá contener los siguientes elementos, **como mínimo**:

    - Tipos de datos:
        - Numéricos
        - Constantes de cadena
    - Operadores:
        - Aritméticos básicos:
            - ✓ suma (+) , resta (-), multiplicación (*), división (/), módulo (%) y potencia (^)
        - Lógicos:
            - ✓ y, o, no
        - Relacionales:
            - ✓ Mayor que, menor que, distinto

    - Sentencias:
        - Asignación, que almacena el valor de una expresión en una variable
        - Condicional simple: si … entonces …
        - Condicional compuesta: si … entonces …. si_no …
        - Bucles:
            - ✓ mientras
            - ✓ para
            - ✓ repetir … hasta que
        - Entrada/ salida:
            - ✓ funciones LEER y ESCRIBIR que leen desde el teclado y escriben por la pantalla el valor de una variable o constante.

- Para optar a una calificación de **notable**, además de los elementos anteriores, la práctica deberá contener:
    - Tipos de datos: arrays unidimensionales.
    - Variables de cadena
    - Operadores para concatenar cadenas y concatenar valores numéricos a cadenas.
    - Sentencia condicional de selección múltiple

- Para optar a una calificación de **sobresaliente**, además de los elementos anteriores, la práctica deberá contener
    - Tipos de datos: arrays n-dimensionales.

- Permitir la existencia de funciones o sub-programas, a los que se les puedan pasar parámetros (al menos por "valor") y puedan devolver un resultado.

- El resultado de la especificación será un **documento** en el que los estudiantes describirán de manera detallada los elementos del lenguaje para el que va a construir el traductor.
- Este documento se deberá denominar
  - **pseudocodigo.pdf**

2. Construcción del analizador léxico

- Una vez realizada la especificación, la siguiente tarea consistirá en construir un analizador léxico que permita reconocer todos los TOKENS o componentes léxicos del lenguaje que anteriormente se ha definido.
- Para ello se utilizará la herramienta LEX, que permite construir analizadores léxicos a partir de definiciones de expresiones regulares.
- El resultado de esta fase será un fichero con la especificación del analizador léxico en formato lex.
- Este documento se deberá denominar
  - **lexico.l**

3. Construcción del analizador sintáctico

- Se debe escribir la gramática del lenguaje especificado.
- Esta gramática es el eje de la práctica y debe estar cuidadosamente diseñada, abarcando todas las posibles sentencias que pueden aparecer en un programa fuente.
- Los errores en el diseño de esta gramática obligarán a volver sobre ella más adelante, lo que conlleva pérdidas de tiempo y esfuerzo.
- Es conveniente diseñar la gramática primero con "lápiz y papel" para posteriormente implementarla en YACC.
- La implementación de la gramática utilizando YACC se hace especificando cada una de las reglas de producción de dicha gramática.
- EL resultado de dicha fase será un fichero fuente de YACC con dicha especificación.
- Este documento se deberá denominar
  - **gramatica.y**

4. Generación de código

- En esta fase se dotará a la gramática previamente desarrollada de una "semántica", asociando una **acción** a cada una de las reglas de producción de la gramática.
- Dicha acción está escrita en **lenguaje C** y esencialmente consistirá en generar un fragmento del programa objeto.

- El programa objeto, resultado de la compilación podrá estar escrito en cualquier lenguaje de programación.
- En principio el resultado será un programa en **ANSI-C**, compilable con el compilador GNU de c (gcc), aunque los estudiantes puede optar por cualquier otro lenguaje de alto nivel (java, C++, Pascal, etc.) o de nivel de ensamblador (en este caso se recomienda utilizar algún ensamblador "sencillo" como ENS2001[1] o JASMIN[2]).
- EL resultado de dicha fase será un fichero fuente de YACC con la de las acciones semánticas de cada regla de producción de la gramática
- Este documento se deberá denominar
    - gramatica_ampliada.y

- Además, se deberá redactar un fichero "**makefile**" que permita genera el "traductor de pseudocógido" utilizando los ficheros
    - lexico.l
    - gramatica_ampliada.y
    - y cualquier otro fichero auxiliar que sea necesario

Entrega

- Se deberá subir a la tarea de *moodle* un fichero comprimido "zip" que contenga los siguientes ficheros:
    - Leeme.txt:
        - Contendrá el nombre o nombres de los autores del trabajo de prácticas
        - Nombre y descripción de los ficheros contenidos en el fichero comprimido "zip"
        - También se indicará la forma de generar y ejecutar el traductor.
    - pseudocódigo.pdf
    - léxico.l
    - gramatica.y
    - gramatica_ampliada.y
    - makefile
    - **Al menos** un fichero de ejemplo que contenga un programa escrito en pseudocódigo
        - ejemplo.p (o la extensión que se desee)
    - Un fichero de ejemplo que contenga la salida generada por el traductor al recibir el fichero *ejemplo.p*
        - ejemplo.c (o la extenxión que corresponda)
    - y cualquier otro fichero auxiliar que sea necesario

[1] http://usuarios.multimania.es/ens2001/
[2] http://jasmin.sourceforge.net/

Evaluación

- La evaluación de la práctica se hará mediante
  - Una presentación del trabajo de prácticas ante el profesor.
  - La corrección del trabajo realizado:
    - Documentación elaborada
      - ✓ Pseudocódigo.pdf
      - ✓ lexico.l
      - ✓ gramatica.y
      - ✓ gramatica_amplicada.y
      - ✓ Ficheros con los ejemplos
        - Se valorará que el código esté debidamente documentado
    - Funcionamiento del traductor de pseudocódigo

- La presentación podrá ser calificada cualitativamente como "mal", "bien" y "muy bien", dependiendo del conocimiento que demuestren los estudiantes de la práctica que han realizado.

- Para poder aprobar el trabajo de prácticas es necesario
  - Obtener una calificación cualitativa de la presentación de "bien o muy bien"
  - Haber realizado al menos la funcionalidad mínima exigida al traductor de pseudocódigo

- Importante
  - Aquellas prácticas que a juicio del profesor se consideren copiadas (sean de "origen" o de "destino" de copia) se calificarán con "suspenso", lo que implica el suspenso en la asignatura.