Procesadores de lenguajes

Ingeniería Informática
Especialidad de Computación
Tercer curso, segundo cuatrimestre

Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

Curso académico: 2015 - 2016

TRABAJO DE PRÁCTICAS

1. Introducción

- **Competencias**
  - El presente trabajo de prácticas pretende desarrollar las siguientes "competencias de la asignatura":
    - CU1. Acreditar el uso y dominio de una lengua extranjera.
    - CTEC2. Capacidad para **conocer** los fundamentos teóricos de los lenguajes de programación y las técnicas de procesamiento léxico, sintáctico y semántico asociadas, y saber **aplicarlas para la creación, diseño y procesamiento de lenguajes.**
- **Objetivo**
  - Se debe utilizar **flex** y **bison** para elaborar un ***i**ntérprete de **p**seudocódigo en **e**spañol*:
    - *ipe.exe*

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

2. Elaboración y entrega

- Modo de realización del trabajo
  - El trabajo se podrá realizar de forma individual o por parejas.

- Modo de entrega
  - Un fichero comprimido deberá ser "subido" a la tarea de la plataforma de "*moodle*".
  - Dicho fichero comprimido deberá contener:
    - Documentación del trabajo (véase el apartado nº 6)
    - Fichero de flex

- Fichero de bison
- Ficheros de C (".c", ".h")
- Fichero makefile
- Ficheros de ejemplo de pseudocódigo con la extensión ".e"

- **Plazo de entrega**
  - Hasta las 9:00 horas del lunes 6 de junio de 2016.

3. Características de lenguaje de pseudocódigo

a) Componentes léxicos o *tokens*

- **Palabras reservadas**
  - ✓ *_mod, _div*
  - ✓ *_o, _y, _no,*
  - ✓ *leer, leer_cadena*
  - ✓ *escribir, escribir_cadena,*
  - ✓ *si, entonces, si_no, fin_si*
  - ✓ *mientras, hacer, fin_mientras*
  - ✓ *repetir, hasta*
  - ✓ *para, desde, hasta, paso, fin_para*
  - ✓ *_borrar, _lugar*
- **Observaciones**
  - ✓ No se distinguirá entre mayúsculas ni minúsculas.
  - ✓ Las palabras reservadas no se podrán utilizar como identificadores.

- **Identificadores**
  - 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'
    'Ejemplo de cadena con salto de línea \n y tabulador \t'
  - 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.

- **Operador de asignación**
  - asignación: :=

- **Operadores aritméticos**
  - suma: +
    - ✓ Unario: + 2
    - ✓ Binario: 2 + 3
  - resta: -
    - ✓ Unario: - 2
    - ✓ Binario: 2 - 3
  - producto: *
  - división: /
  - división entera: _div
  - 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 el símbolos #

    # 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.
  - ***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 (||).

- **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.

      escribir_cadena('\t Introduzca el dato \n');

- **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

[1] Una *condición* será una *expresión relacional* o una *expresión lógica compuesta*.

- 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.

- **Observación**
  - Se debe permitir que una variable pueda cambiar de tipo durante la ejecución del intérprete.
    - ✓ Ejemplo

```
@ la variable dato es numérica
dato := 10;
escribir(dato);
…
@ la variable dato se convierte en alfanumérica
leer_cadena(dato);
escribir_cadena(dato);
```

- **Se valorará la inclusión de nuevos operadores o sentencias**
  - **Ejemplos**
    - ✓ Operadores unarios: ++, --
    - ✓ Operadores aritméticos y de asignación: +:=, -:=, etc.
    - ✓ Sentencia "según"

```
segun (expresión)
    valor v1:  …
```

[2] Se valorará que se controlen los pasos con incrementos positivos y negativos del bucle "*para*".

valor v2: …

…

defecto: …

fin_segun

- ✓ Etc.

4. 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 "reglas de producción de control de errores" que no generen conflictos.

- **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.

5. 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

    > …
  - Se utilizará el carácter de fin de fichero para terminar la ejecución: Control + D

- **Ejecución desde un fichero**
  - Se interpretarán las sentencias de un fichero pasado como argumento desde la línea de comandos
  - El fichero deberá tener la extensión ".e"

    ipe.exe ejemplo.e

6. Documentación del trabajo

Se deberá elaborar un documento de texto con las siguientes características:

- **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
  - Especialidad: Computación
  - Tercer curso
  - Segundo cuatrimestre
  - Curso académico: 2015 – 2016
  - Escuela Politécnica Superior de Córdoba
  - Universidad de Córdoba
  - Lugar y fecha

- **Índice**
  - Las páginas deberán estar numeradas.

- **Introducción**
  - Breve descripción del trabajo realizado y de las partes del documento.

- **Lenguaje de pseudocódigo**
  - Se corresponde con el apartado nº 3 de este documento
    - Componentes léxicos
    - Sentencias
  - **Observación**
    - Si se ha ampliado el lenguaje de pseudocódigo entonces se deberá indicar en este apartado.

- **Tabla de símbolos**
  - Descripción
  - Se valorará que se utilice una implementación eficiente de la tabla de símbolos: lista ordenada, árbol binario, etc.

- **Análisis léxico**
  - Descripción del fichero de **flex** utilizado para definir y reconocer los componentes léxicos.

- **Análisis sintáctico:**
  - Descripción del fichero de **bison** utilizado para definir la gramática de contexto libre
    - Símbolos de la gramática
      - ✓ Símbolos terminales (componentes léxicos)
      - ✓ Símbolos no terminales

        - Reglas de producción de la gramática
        - 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"**.
            - ✓ Se valorará la inclusión de gráficos explicativos.

- **Funciones auxiliares**
    - Se deben indicar y describir las funciones auxiliares que se hayan codificado.

- **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**
    - Al menos se deben proporcionar dos ejemplos.
    - Se valorará la cantidad, originalidad y complejidad de los ejemplos propuestos.
    - También se puede incluir el ejemplo propuesto por el profesor.

- **Conclusiones**:
    - Reflexión sobre el trabajo realizado.
    - Puntos fuertes y puntos débiles del intérprete desarrollado.

- **Bibliografía o referencias web**
    - Se recomienda consultar el documento elaborado por el personal de la biblioteca de la Universidad de Córdoba
        - ¿Cómo citar bibliografía en un trabajo académico?

          http://www.uco.es/servicios/biblioteca/CursosP/referenciasbibliograficas.pdf

- **Anexos**
    - Se podrían incluir aquellos anexos que se consideren oportunos para mejora la calidad de la documentación

7. Criterios de evaluación

- **Documentación**: 40 %
    - Se tendrá en cuenta lo indicado en el apartado nº 6.
    - El código elaborado deberá estar documentado.

- Se valorará la inclusión de gráficos o figuras.
- Se valorará la cantidad, originalidad y **complejidad** de los ejemplos propuestos.
- También se valorará
  - la acentuación,
  - la corrección ortográfica
  - y la calidad y claridad de la redacción.

- **Funcionamiento del intérprete (software):** 60 %
  - La gramática diseñada **no** podrá conflictos.
    - Esta condición es **imprescindible** para aprobar el trabajo de prácticas.
  - El intérprete deberá
    - funcionar correctamente en el entorno de **ThinStation** tanto de forma interactiva como ejecutando la instrucciones de los ficheros de ejemplo
    - en particular, deberá ejecutar correctamente el ejemplo propuesto por el profesor y los ejemplos propuestos por los autores del trabajo.

  - Se valorará
    - la completitud del lenguaje de pseudocódigo.
    - La calidad en el diseño del lenguaje y la gramática.
    - El control de errores.
    - La ampliación del lenguaje de pseudocódigo.

  - **Observación**:
    - Además, se valorará la asistencia a clase de prácticas y la resolución de dificultades encontradas durante la elaboración del trabajo.