Procesadores de lenguajes

Ingeniería Informática
Especialidad de Computación
Tercer curso, segundo cuatrimestre

Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

Curso académico: 2014 - 2015

TRABAJO DE PRÁCTICAS

1. Introducción

- Se debe utilizar **flex** y **bison** para elaborar un ***i****ntérprete de* ***p****seudocódigo en* ***e****spañol con* ***notación prefija***:
  - *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.

- **Plazo de entrega**
  - Hasta las 9:00 horas del lunes 8 de junio de 2015.

- 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
    - Dos o más ficheros de ejemplo con la extensión ".e"

3. Características de lenguaje de pseudocódigo

a) Componentes léxicos o *tokens*

- **Palabras reservadas**
  - **Palabras claves**
    - ✓ *#mod*
    - ✓ *#o, #y, #no*
    - ✓ *leer, leer_cadena*
    - ✓ *escribir, escribir_cadena*
    - ✓ *si, entonces, si_no*
    - ✓ *mientras*
    - ✓ *hacer*
    - ✓ *repetir*
    - ✓ *hasta*
    - ✓ *para*
    - ✓ *desde*
    - ✓ *paso*
    - ✓ *borrar*
    - ✓ *lugar*
  - **Constantes y funciones predefinidas**
    - ✓ PI, E, …
    - ✓ seno, coseno, raíz, …
  - **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'
  - 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 con notación prefija**
  - suma: +
    - ✓ Unario: *(+ 2)*
    - ✓ Binario: *(+ dato 2)*
  - resta: -
    - ✓ Unario: *(- 2)*
    - ✓ Binario: *(- dato 2)*
  - producto: *
    (2 dato)*
  - división: /
    (/ dato 2)
  - módulo: **#mod**
    ***(#mod** dato 2)*
  - potencia: ^
    (^ a 3)

- **Expresión numérica**
  - Número:
    - ✓ Ejemplos
      10
      12.5
      0.5e-7
  - Variable que tenga un valor numérico

dato

- Expresión compuesta por operadores numéricos y constantes o funciones predefinidas
  - ✓ Ejemplo

```
(raiz
        (+
                (* a a)
                (* b b)
        )
)
```

- **Operador alfanumérico**
  - concatenación: ::
    - ✓ Ejemplo

      (:: 'Hola ' nombre)

      donde nombre es una variable con un valor alfanumérico

- **Expresión alfanumérica**
  - Cadena:
    'Hola'
  - Variable con un valor alfanumérico
    nombre
  - Expresión compuesta por el operador de concatenación

    (:: 'Hola ' nombre)

- **Operadores relacionales de números y cadenas con notación prefija**
  - menor que: <

    (< dato 0)

    (< nombre 'Antonio')
  - menor o igual que: <=

    (<= dato 0)

    (<= nombre 'Antonio)
  - mayor que: >

    (> dato 0)

    (> nombre 'Antonio')
  - mayor o igual: >=

    (>= dato 0)

    (>= nombre 'Antonio')
  - igual que: =

    (= dato 0)

    (= nombre 'Antonio')

- distinto que: <>

  (<> dato 0)

  (<> nombre 'Antonio')

- **Operadores lógicos**
  - disyunción lógica: #o

    (#o (> a 5) (> b 5))
  - conjunción lógica: #y

    (#y (> a 0) (< a 10))
  - negación lógica: #no

    (#no (<> control 'stop'))

- **Condición**
  - **Simple**

    (= dato 7)
  - **Compuesta**

    (#y (> a 0) (< a 10))

- **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 maravilloso de cometario de una línea

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

      (:= dato 12.5)

- ***(:= 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 'Alejandro')

- **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*
    sentencias
)

  - Sentencia *condicional* compuesta

    (***si** condición*
    ***entonces** sentencias*
    ***si_no** sentencias*
)

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

- Bucle "*mientras*"
  *(**mientras** condición*
  *****hacer*****
  sentencias
)

- Bucle "*repetir*"
  (***repetir***
  sentencias
  hasta
  condición
)

- Bucle[2] "*para*"
  *(**para** identificador*
  ***desde** expresión_numérica*
  ***hasta** expresión numérica*
  ***paso** expresión numérica*
  hacer
  sentencias
)

- **Comandos especiales**
  - Borrar la pantalla
    (Borrar)
  - Posicionar el curso en pantalla
    (Lugar** expresión numérica expresión_numérica**)
    - ✓ Coloca el cursor de la pantalla en las coordenadas indicadas por los valores de las expresiones numéricas.

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 error" que no generen conflictos.
- **Semánticos**

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

- 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**
  - Cada grupo deberá **decidir** si permite o no cambiar el tipo de una variable durante la ejecución del intérprete.
    - Si permite cambiar de tipo entonces debe controlar la ejecución de las sentencias de asignación, lectura y escritura.
      - ✓ Ejemplo

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

    - Si no se permite cambiar el tipo de una variable entonces debe controlar los errores que se pueden producir en las sentencias de asignación, lectura o escritura.
      - ✓ Ejemplo

```
@ la variable dato es numérica
(=  dato)
(escribir dato)
…
{ Se produce un error al intentar convertir
la variable numérica dato
en  variable alfanumérica }
(leer_cadena dato)
```

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
    > …

- **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

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

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: 2014 – 2015
  - 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**
  - 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
- **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
    - 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"**.
- **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**
  - Se valorará la cantidad, la originalidad y complejidad de los ejemplos propuestos.
- **Conclusiones**:
  - Reflexión sobre el trabajo realizado
  - Puntos fuertes y puntos débiles del intérprete desarrollado.
- **Bibliografía o referencias web**
- **Anexos**

7. Criterios de evaluación

- **Documentación: 40 %**
  - Se tendrá en cuenta lo indicado en el apartado nº 6
  - El código elaborado deberá estar documentado.
  - Calidad en la descripción del lenguaje y la gramática.
  - Descripción de los ejemplos
  - 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 %**
  - **Importante**
    - La gramática diseñada **no** deberá tener ningún conflicto.
    - Esta condición es **imprescindible** para poder aprobar el trabajo de prácticas
  - Completitud del lenguaje de pseudocódigo.
  - Control de errores.
  - El intérprete deberá funcionar correctamente en **ThinStation**
    - tanto de forma interactiva
    - como ejecutando la instrucciones de un fichero.
  - Los ejemplos deberán funcionar correctamente.
    - En particular, deben funcionar correctamente los ejemplos propuestos por el profesor.
    - Nuevos ejemplos
      - Cantidad, originalidad y complejidad de los ejemplos propuestos por los estudiantes
      - Se valorarán los ejemplos que prueben comandos no utilizados en los ejemplos propuestos por el profesor: potencia (^), concatenación (::), etc.

- **Otros criterios de evaluación: 10 %**
  - Ampliación del lenguaje de pseudocódigo
    - Por ejemplo: operadores incremento (++) y decremento (--), etc.
  - Soluciones a dificultades encontradas durante la elaboración del trabajo que hayan sido convenientemente documentadas.
  - Asistencia a clase de prácticas.