



Procesadores de lenguajes

Ingeniería Informática
Especialidad de Computación
Tercer curso, segundo cuatrimestre

Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

Curso académico: 2012 - 2013

TRABAJO DE PRÁCTICAS

1. Introducción

- Se debe utilizar **flex** y **bison** para elaborar un *intérprete de pseudocódigo en español*:
 - *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 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 6)
 - Fichero de flex
 - Fichero de bison
 - Ficheros de C (“*.c*”, “*.h*”)
 - 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: $__\text{mod}$
 - 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 comentario 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 operadores alfanuméricos

○ 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 control¹

- Sentencia condicional simple
si condición
entonces sentencias

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

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

- Bucle² “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. 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.

² Se valorará que se controlen los pasos con incrementos positivos y negativos del bucle “para”.

- **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

> ...

- 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.

6. 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
- **Tabla de símbolos**

- Descripción
- Análisis léxico
 - Expresiones regulares utilizadas para definir los componentes léxicos
- Análisis sintáctico:
 - 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
- 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 dos ejemplos
 - Fichero de ejemplo propuesto por el profesor.
 - Fichero de ejemplo propuesto por el autor o autores
 - Se valorará “fundamentalmente” la originalidad y complejidad de este ejemplo.
- Conclusiones:
 - Reflexión sobre el trabajo realizado
 - Puntos fuertes y puntos débiles del intérprete desarrollado.
- Bibliografía o referencias web
- Anexos

7. 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 6
 - El código elaborado deberá estar documentado.
 - 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.
- **Otros criterios de evaluación que podrán mejorar la calificación**
 - 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 alumnos podrán exponer el trabajo realizado al profesor tanto si ellos lo desean como si el profesor lo solicita.