



PROCESADORES DE LENGUAJE

Ingeniería Informática
Primer curso de segundo ciclo



Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba

Curso académico 2009 - 2010

Hoja de ejercicios nº 1: ANÁLISIS LÉXICO

1. Indica el alfabeto de cada uno de los siguientes lenguajes de programación: C++ y Java.
2. Muestra algunas de las expresiones regulares que se pueden definir sobre los siguientes alfabetos:
 - a) $\Sigma = \{0, 1\}$
 - b) $\Sigma = \{a, b, c\}$
 - c) $\Sigma = \{a, b, c, 0, 1\}$
 - d) $\Sigma = \{\text{int, float, if, else, while}\}$
3. Describe de manera informal los lenguajes denotados por las siguientes expresiones regulares definidas sobre el alfabeto $\Sigma = \{a, b, c\}$:
 - a) $a(b+c)^*a^*$
 - b) $a(b+c)^*a$
 - c) $a(b^*+c^*)a$
 - d) $a(b+c^*)a$
 - e) $a^*(b+c)a$
 - f) $a(b+c)a^*$
4. Considérense las siguientes expresiones regulares:
 - $\alpha = a^* + b^*$
 - $\beta = a b^* + b a^* + b^* a + (a^* b)^*$
 - a) Indica una palabra que pertenezca al lenguaje denotado por α pero que no pertenezca al denotado por β . $X \in L(\alpha) - L(\beta)$
 - b) Indica una palabra que pertenezca al lenguaje denotado por β pero que no pertenezca al denotado por α . $X \in L(\beta) - L(\alpha)$
 - c) Encuentra una palabra que pertenezca al lenguaje denotado por α y al denotado por β . $X \in L(\alpha) \cap L(\beta)$
 - d) Encuentra una palabra que no pertenezca a ninguno de los dos lenguajes denotados por las expresiones regulares. $X \in \Sigma^* - (L(\alpha) \cap L(\beta))$
5. Define expresiones regulares que denoten los siguientes lenguajes definidos recursivamente:
 - a) $x \in L$. Si $x \in L$, entonces aax y xb son palabras de L . Sólo están en L las palabras obtenidas mediante las premisas anteriores.

- b) $x \in L$. Si $x \in L$, entonces abx , bax , aax y bbx son palabras de L . Sólo están en L las palabras obtenidas mediante las premisas anteriores.
6. Escribe expresiones regulares que denoten los siguientes lenguajes definidos sobre $\Sigma = \{a, b, c\}$
- Cadenas que comienzan y finalizan con la letra "a".
 - Cadenas que comienzan o finalizan con la letra "a" (o ambas posibilidades).
 - Cadenas en las que la "a", si aparece, siempre precede a la "b".
 - Cadenas que tengan un número impar de aes.
7. Escribe expresiones regulares que denoten los siguientes lenguajes definidos sobre $\Sigma = \{0, 1, \dots, 9, +, -, e, E, .\}$
- Números naturales
 - Números naturales que no contengan dos o más ceros al principio: 0, 10, 121,...
 - Números pares.
 - Números impares.
 - Números reales con formato de punto fijo: 2.343, -12.0, + 12.02, 0.0001, 505.00,...
 - Números reales con formato de punto flotante: -323.3 e -7, + 0.3 E - 12,...
 - Números reales con formato de punto fijo o con formato de punto flotante.
 - Números reales con formato de punto fijo o con formato de punto flotante pero que no tengan ceros superfluos, es decir,
 - son permitidos los números del tipo 0.0, 132.0, 0.526, 1203.0494,
 - pero no son permitidos los números de la forma 00.12, 124.000, 001.727, 52.700.
8. Escribe las expresiones regulares de los siguientes componentes léxicos ("tokens") del lenguaje C:
- Identificadores
 - Números
 - Caracteres
 - Cadenas de caracteres
 - Operadores aritméticos
 - Operadores relacionales
 - Operadores lógicos
 - Palabras reservadas: if, else, while, for, main, switch, case, #include, #define, etc.
 - Signos de puntuación:[,], {, }, ;, etc.
 - Comentarios
9. Dado el siguiente código escrito en lenguaje C que implementa el método de ordenación de Shell:
- Indica los tipos de componentes léxicos o "tokens" que generaría el analizador léxico.
 - Define las expresiones regulares que denotan los tipos de componentes léxicos indicados en el apartado anterior.

```
#include <stdio.h>
#include <malloc.h>
```

```

#include "macros.h"
/* Longitud maxima -1 de los nombres */
#define NUMERO_CARACTERES 20

void shell (struct ficha_persona *dato, int n)
{
    int d,i, bandera;
    struct ficha_persona auxiliar;

    d = n;
    do {
        d = d / 2;
        do{
            bandera = 0;
            i = 0;
            do {
                if (dato[i].edad > dato[i+d].edad)
                {
                    strcpy(auxiliar.nombre, (dato+i)->nombre);
                    auxiliar.edad = (dato+i)->edad;
                    strcpy((dato+i)->nombre, (dato+i+d)->nombre);
                    (dato+i)->edad = (dato+i+d)->edad;
                    strcpy((dato+i+d)->nombre, auxiliar.nombre);
                    (dato+i+d)->edad = auxiliar.edad;
                    bandera = 1;
                }
                i++;
            } while (i+d <= n-1);
        } while (bandera !=0);
    } while(d!=1);
}

```

10. Considera el siguiente fragmento de código erróneo escrito en C

```

foor (i = N ; > i 1.0.0 ; i--)
{
    factorial = factorial * $n;
}

printf " Factorial = %d " /n, $factorial);

```

- Indica los errores que **puede** detectar "el analizador léxico" y los que **no** puede detectar y "**por qué**".
- Indica los componentes léxicos que reconocería el "analizador léxico".
- Escribe las expresiones regulares correspondientes a los diferentes tipos de componentes léxicos reconocidos en el fragmento anterior.

11. Considera el siguiente código escrito en el lenguaje FORTRAN

```

INTEGER I, J
REAL Vector(10), Matriz(5,5)

```

```

PRINT *, 'INTRODUCE LAS COMPONENTES IMPARES DEL VECTOR'
DO 10 I = 1, 9, 2
    PRINT *, 'COMPONENTE ', I, ' -->'
    READ *, Vector (I)
10 CONTINUE

PRINT *, 'INTRODUCE LAS COMPONENTES DE LA DIAGONAL PRINCIPAL'
DO 20 I = 1, 5
    PRINT *, 'COMPONENTE (', I, ', ', I, ') -->'
    READ *, Matriz (I,I)
20 CONTINUE

```

- Indica los diferentes tipos de componentes léxicos que reconocería el analizador léxico.
- Escribe las expresiones regulares correspondientes a dichos tipos de componentes léxicos.

12. Define las expresiones regulares que denoten los siguientes componentes léxicos de FORTRAN:

- Identificadores: han de comenzar por una letra y estar seguidos por cero o más letras o dígitos.
- Operadores relacionales: .LE., .LT., .GE., .GT., EQ., .EN., .le., .lt., etc.
- Operadores lógicos: .OR., .AND., .NOT., .XOR. .or., .and., etc.

13. Se desea diseñar un lenguaje de programación en pseudocódigo cuyos componentes léxicos tengan las siguientes características:

- Identificadores:
 - Podrán estar compuestos por letras, números y el símbolo \$.
 - Deberán comenzar por una letra.
 - El símbolo \$ sólo podrá aparecer al final.
- Números:
 - Se podrán definir números enteros (19), reales de punto fijo (19.75) o con notación exponencial (0.19e+2).
- Cadenas de caracteres:
 - Estarán compuestas por cualquier carácter excepto las comillas simples de apertura (') y cierre ('), que deberán aparecer al principio y al final, respectivamente.
 - Se utilizará la barra invertida \ para poder introducir las comillas simples dentro de las cadenas.
- Palabras reservadas o claves:
 - Se podrán escribir con letras mayúsculas o minúsculas o ambas
 - Deberán comenzar por el símbolo de subrayado.
- Operadores:
 - Asignación: se utilizará el operador de Pascal (:=)
 - Lógicos: estarán delimitados por dos símbolos de subrayado (por ejemplo: _no_)
 - Aritméticos: se podrán utilizar operadores para la suma, resta, multiplicación, división, división entera, resto de la división entera y potencia.

- o Relacionales: se utilizarán los símbolos empleados por el lenguaje C, excepto los operadores "igual" y "distinto" que se utilizarán los empleados por Pascal (=, <>).
- a) Define las expresiones regulares que permitan denotar los componentes léxicos descritos.

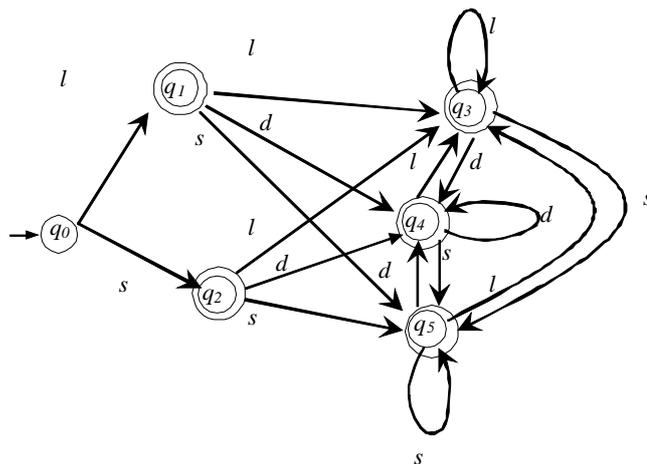
14. Dado el siguiente autómata finito no determinista

δ	a	b	c	ϵ
$\rightarrow q_0$	$\{q_1\}$	$\{q_2, q_3\}$	ϕ	$\{q_2\}$
q_1	ϕ	$\{q_2, q_3, q_4\}$	$\{q_1, q_3\}$	ϕ
q_2	$\{q_3\}$	$\{q_1, q_2\}$	ϕ	$\{q_3\}$
$\leftarrow q_3$	$\{q_4\}$	$\{q_2\}$	$\{q_0\}$	ϕ
$\leftarrow q_4$	ϕ	$\{q_2, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_1, q_2\}$

Nota: sólo se muestran las transiciones épsilon no triviales

- a) Dibuja su representación gráfica.
- b) Comprueba si reconoce la cadena $x = ababc$
- c) Utiliza el algoritmo de construcción de subconjuntos para obtener el autómata finito determinista equivalente.
- d) Comprueba si el autómata finito determinista reconoce la cadena $x = ababc$
- e) Minimiza el autómata finito determinista construido en el apartado c
- f) Comprueba si el autómata finito determinista minimizado reconoce la cadena $x = ababc$

15. Dado el siguiente autómata finito determinista:



donde el significado de l: letra, d: dígito y s: subrayado.

- a) Comprueba si reconoce o no la cadena $x = lds$

- b) Minimiza el autómata finito determinista.
- c) Comprueba si el autómata minimizado reconoce o no la cadena $x = \text{ldsl}$

16. Dadas las siguientes expresiones regulares

- $\alpha_1 = a^* (b + c)^* d$
- $\alpha_2 = a (b^* + c^*) d$
- $\alpha_3 = (\text{letra} + \text{subrayado}) (\text{letra} + \text{subrayado} + \text{dígito})^*$

- a) Utiliza el **algoritmo de Thompson** para construir los autómatas finitos no deterministas equivalentes a dichas expresiones regulares
- b) Utiliza el **algoritmo de construcción de subconjuntos** para obtener el autómata finito determinista equivalente a los autómatas no deterministas obtenidos en el apartado anterior.
- c) **Minimiza** los autómatas finitos deterministas.

17. Dadas las siguientes expresiones regulares:

- **dígito (punto + ϵ) + dígito* punto dígito dígito***
donde **dígito** $\in \{0,1,2,3,4,5,6,7,8,9\}$ y **punto** representa el punto decimal "."
- **c (l + d + e + b c)* c**
donde **l** = letra, **d** = dígito, **c** = comillas dobles, **e** = espacio en blanco y **b**= barra invertida ("\")

- a) Utiliza el **algoritmo de Thompson** para obtener los autómatas finitos no deterministas equivalentes a las expresiones regulares.
- b) Utiliza el **algoritmo de Construcción de subconjuntos** para obtener los autómatas finitos deterministas equivalentes a los autómatas no deterministas del apartado anterior.
- c) **Minimiza**, si es necesario, los autómatas finitos deterministas obtenidos en el apartado anterior.
- d) Comprueba si los últimos autómatas obtenidos reconocen las siguiente palabras, respectivamente:
 - i. 54.123
 - ii. "Hola \"Juan\""