PROCESADORES DE LENGUAJE

Ingeniería Informática
Especialidad de Computación
Tercer curso
Segundo cuatrimestre

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

Curso académico 2021 – 2022

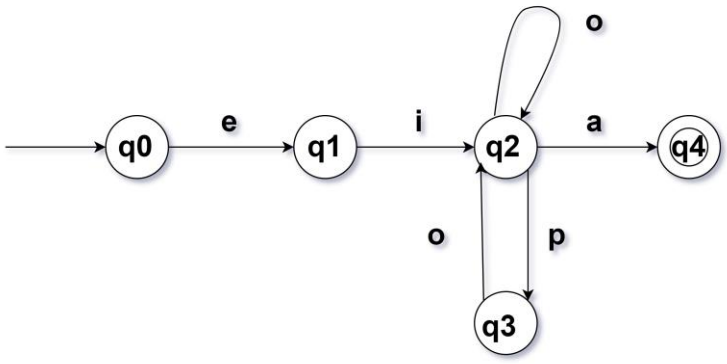
Hoja de ejercicios de FLEX

Nota previa

- En todos los ejercicios, se debe comprobar si el número de argumentos es correcto y si los ficheros existen o no, en su caso.

1. **Sustitución de una palabra**
   - Codifica un analizador léxico que reemplace una palabra por otra en un fichero de entrada.
   - Al final, se debe mostrar el número de palabras sustituidas.
   - Ejemplo de llamada al analizador léxico:
     ./sustituir.exe fichero.txt antigua nueva

2. **Simulación de un autómata finito determinista**



- El anterior autómata finito determinista simula el funcionamiento básico de un ordenador, desde que se enciende hasta que se apaga, donde
  - e: encender
  - i: iniciar sistema operativo
  - o: realizar alguna operación
  - p: cambia al modo de pausa

    - a: apagar
- Utiliza los **estados de flex** para codifica un analizador léxico que permita simular el funcionamiento de este autómata finito determinista.
- Comprueba si el analizador léxico reconoce las siguientes secuencias de acciones:
    - e i o o p o a
    - e i o o p p a

3. Analizador léxico de pseudocódigo

- Codifica un analizador léxico en flex que permita reconocer los componentes léxicos de un programa escrito en el lenguaje C.
- En particular, debe reconocer los componentes léxicos de un programa que ejecuta el algoritmo de ordenación de Shell (véase el código al final del ejercicio).
- El analizador léxico deberá reconocer, al menos, los siguientes componentes léxicos:
    - Palabras reservadas: ***#include***, ***void***, ***main***, ***if***, ***for***, …
    - Identificadores
    - Números
    - Cadenas de caracteres
    - Operadores
        - Asignación
        - Aritméticos
        - Relacionales
        - Lógicos
    - Comentarios
    - Otros componentes léxicos
        - Punto y coma:
        - Paréntesis izquierdo y derecho
        - Llaves izquierda y derecha
        - Etc.

shell.c

```
#include <stdio.h>
#include <stdlib.h>
#define MAXIMO 10
void shell (int *v, int n);

/*
  Este programa recibe un máximo de 10 números enteros
desde la línea de comandos y los ordena usando el
algoritmo de shell

*/
```

```
int main(int argc, char const *argv[])
{
     int numero[MAXIMO];
     int i;

     for (i=0; i<argc-1; i++)
    {
     numero[i] = atoi(argv[i+1]);
    }

    shell(numero,argc-1);

    for (i=0; i<argc-1; i++)
    {
     printf("%d ", numero[i]);
    }
    printf("\n");
    return 0;
}
void shell (int *v, int n)
{
int d, i, bandera;
int auxiliar;

     d = n;
     do
     {
          d = d / 2;
          do
          {
          bandera = 0;
          i = 0;
               do
               {
                    if (v[i] > v[i+d])
                    {
                         auxiliar = v[i+d];
                         v[i+d] = v[i];
                         v[i] = auxiliar;
                         bandera = 1;
                    }
                    i++;
               } while (i+d <= n-1);
          } while (bandera !=0);
     } while(d!=1);
}
```

4. Analizador léxico de pseudocódigo

- Codifica un analizador léxico que permita reconocer los componentes léxicos de un programa escrito en pseudocódigo.

- **Palabras reservadas**
  - *inicio, fin, leer, escribir, si, entonces, si_no, fin_si, mientras, hacer, fin_mientras, repetir, hasta_que, para, desde, hasta, paso, fin_para*
  - No se distinguirá entre mayúsculas ni minúsculas.
  - Las palabras reservadas no se podrán utilizar como identificadores.

- **Identificador**
  - 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 consecutivos.
    - No se distinguirá entre mayúsculas ni minúsculas.
  - Ejemplos
    - Identificadores válidos:
      dato, dato_1, dato_1_a
    - Identificadores **no** válidos:
      dato, dato, dato__1

- **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 formarán parte de la cadena.

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

- **Operadores aritméticos:**
  - SUMA: +

  - RESTA: -
  - PRODUCTO: *
  - DIVISIÓN: /
  - MÓDULO: #mod
  - DIVISIÓN ENTERA: #div
  - POTENCIA: **

- **Operador alfanumérico:**
  - CONCATENACIÓN: ||

- **Operadores relacionales de números y cadenas:**
  - MENOR_QUE: <
  - MENOR_IGUAL_QUE: <=
  - MAYOR_QUE: >
  - MAYOR_IGUAL_QUE: >=
  - IGUAL: ==
  - DISTINTO: <>
  - 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 << y >>

    << *ejemplo*
    de comentario
    de tres líneas >>

  - De una línea:
    - Todo lo que siga a los dos caracteres "!!" hasta el final de la línea.

      !! *ejemplo de comentario de una línea*

- **Otros componentes léxicos**
  - FIN_SENTENCIA: ;

  - Paréntesis
    - Izquierdo: (
    - Derecho:)

- **Control de errores**
  - El intérprete deberá controlar toda clase de errores:
    - Identificador mal escrito.
    - Números mal escritos.
    - Utilización de símbolos no permitidos.
    - Etc.

- **Prueba**
  - Se deberá comprobar el funcionamiento del analizador léxico usando tres ficheros:
    - Fichero denominado Newton.txt
    - ejemplo_1.txt: fichero original **sin** errores.
    - ejemplo_2.txt: fichero original **con** errores.
  - **Importante**
    - Se valorará que los ejemplos propuestos tengan un código de un algoritmo o tarea interesante.