UNIVERSIDAD DE OVIEDO

Departamento de Ingeniería Eléctrica,
Electrónica, de Computadores y Sistemas

Guía de Prácticas

ASIGNATURA:	Informática Industrial y Comunicaciones		
CENTRO:	Escuela Politécnica de Ingeniería de Gijón		
ESTUDIOS:	Grado en Ingeniería Electrónica y Automática		
CURSO:	3º	CUATRIMESTRE:	1
CARÁCTER:	Obligatoria	CRÉDITOS ECTS:	6
PROFESORADO:	Ignacio Alvarez, José Mª Enguita, Borja Millán		

PRACTICA 9: Procesamiento de cadenas de caracteres, estados

1. En el sistema de control de motor simulado, realizar un programa que espere por una cadena de caracteres (función gets) y la procese para determinar el modo de control, según los comandos siguientes:

   >> POS=POT (se desea control de posición con consigna obtenida del potenciómetro manual).

   >> POS=valor_en_grados (se desea control de posición con la consigna indicada en valor, independientemente del potenciómetro manual).

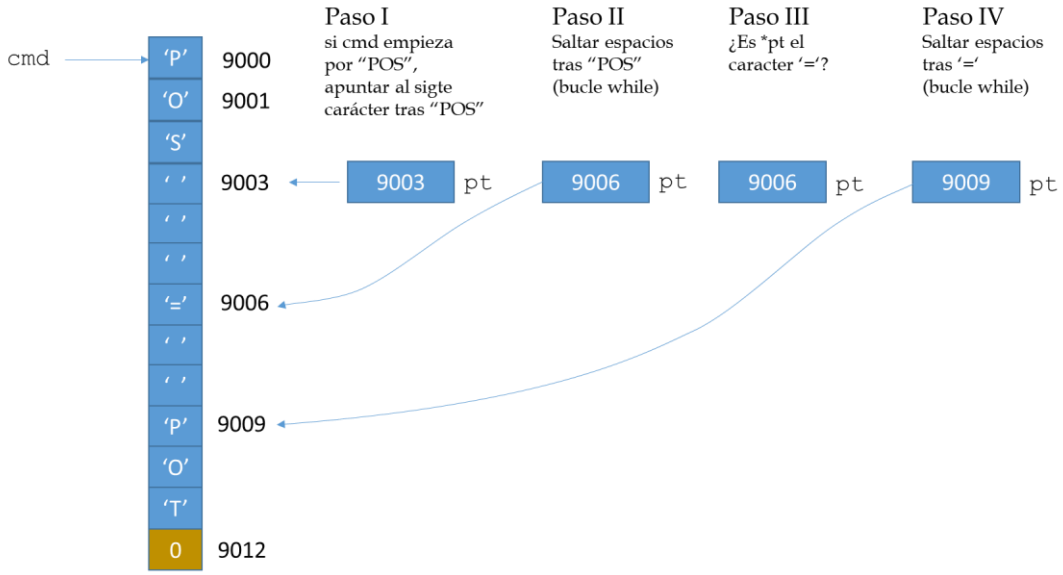
   >> TENSION=valor_en_volt (se indica el valor de tensión a aplicar en control todo/nada).

   >> VEL=POT (se desea control de velocidad con consigna obtenida del potenciómetro manual).

   >> VEL=valor_en_rpm (se desea control de velocidad con consigna indicada en valor, independientemente del potenciómetro manual).

   En todos los casos, se debe procesar la cadena teniendo en cuenta que el usuario podría introducir espacios en blanco a ambos lados del igual

   1.a) Realizar un programa que pide por teclado una cadena, comprueba si empieza por **POS**, salta espacios en blanco, comprueba que el siguiente carácter es **=** , salta espacios en blanco, escribe en pantalla el contenido tras los mismos.



1.b) Mover el código común a todas las opciones a una función del estilo:

```
const char* GetValorComando(const char* comando,const char* busca);
```

que procese la cadena **comando** y devuelva:

- puntero al texto que está tras los espacios posteriores al '=' (valor de **pt** en el paso IV de la figura anterior) si se cumplen las condiciones 1.a). Sólo hay que sustituir **"POS"** por **busca,** y el valor constante **3** (longitud de "POS") por **strlen(busca)** .
- NULL si no se cumple alguna de las condiciones

1.c) Modificar el **main( )** realizado en la práctica anterior para que detecte los 5 comandos, y actualice las variables necesarias en función de su contenido.

```
// Las diferentes opciones para un estado/selección se guardan en enteros
#define MODO_REF_POTENCIOM   1  // Referencia por potenciómetro (POS/VEL=POT)
#define MODO_REF_TECLADO     2  // Referencia por teclado (POS/VEL=valor)

#define MODO_CONTROL_POS     1 // Modo control posición (POS=POT/valor)
#define MODO_CONTROL_VEL     2 // Modo control velocidad (VEL=POT/valor)
…
main()
{
   int modo_ref=MODO_REF_POTENCIOM,modo_control=MODO_CONTROL_POS;
   float valor_ref_teclado;
   float tension_cte_teclado=2.0f;
   char cmd[80];
   const char* pt_value;
   Resto de vbles necesarias para control de pos y de vel

   Conectar con simulador, inicializar LCD
   Escribir prompt y pedir cmd con gets()

   pt_value=GetValorComando(cmd,"POS");
   if (pt_value!=NULL)
   {
      modo_control=MODO_CONTROL_POS;
      Actualizar modo_ref y valor_ref_teclado según contenido a partir de pt_value
   }
   pt_value=GetValorComando(cmd,"VEL");
   if (pt_value!=NULL)
   {
      modo_control=MODO_CONTROL_VEL;
      Actualizar modo_ref y valor_ref_teclado según contenido a partir de pt_value
   }
   pt_value=GetValorComando(cmd,"TENSION");
   if (pt_value!=NULL)
   {
      Actualizar tension_cte_teclado a partir de pt_value
   }

   while (1)
   {
       Desplazar Tablas Temporales
       Leer valores más recientes de sensores
       Calcular ek[0] según modo_ref, modo_control y valor_ref_teclado
       Calcular uk[0] según estado de SW y modo_control
       AplicarTensionMotor(uk[0]);
       Actualizar LEDs y escribir en LCD
       Retardo TM_ms;
   }
}
```

2. Añadir LEDS de indicación de modo de funcionamiento actual:

- Si estamos en control de velocidad, **parpadear** cada 0.5 seg el LED de peso 4.
- Si estamos en referencia por teclado, **parpadear** cada 0.5 seg el LED de peso 5

Para parpadear, usar un contador que sature en 5 (0.5 seg = 5 * TM_ms), y modificar el estado del bit con el operador ^ (XOR a nivel de bit).

AMPLIACIONES

3. Añadir servicio de comando de cadena abierta:

   OPEN LOOP = valor_en_V

   Ante la recepción de este comando, se establece un nuevo modo de control MODO_OPEN_LOOP, en el que el programa aplica directamente al motor la tensión indicada, sin ningún cálculo adicional en el lazo de control.

4. Añadir servicio de comando de espera:

   SLEEP = tiempo_en_ms

   Ante la recepción de este comando, el programa simplemente obtiene el valor de tiempo_en_ms y realiza por este orden: Activar LED peso 0 ; retardo(tiempo); Desactivar LED peso 0 (por tanto, deja de recibir comandos durante ese tiempo).

5. Dar servicio al comando:

   RZ = [b_0 b_1 …. b_m]/[a_0 a_1 …. a_n]

que permita cambiar el tamaño y coeficientes del regulador, con las condiciones siguientes:

- Reasignar memoria dinámica para tablas afectadas.
- El separador entre los valores de los polinomios puede ser espacio(s) y/o coma.

5.1) Se realizará algo similar para un polinomio en una PA próxima, se puede esperar a su resolución para realizar esta ampliación.

5.2) Crear inicialmente una función:

const char* ContarElementosEnTabla(const char* cadena,int* n_els);

Esta función debe devolver 2 valores:

- El número de reales entre **[** y **]** que se encuentren separados por espacios o comas en la variable apuntada por n_els.
- El puntero al carácter tras el **]** como valor retornado.

El nº de elementos será 0 y el puntero devuelto NULL si no se encuentra una tabla con el formato adecuado.

Ejemplo para prueba:

```
char texto[]="[ 0.14 , -0.25 , 0.76 ]";
```

Secuencia de código y resultados previstos:

5.2.a) `pt=cadena`. Comprueba `*pt == '['` .

```
pt
↓
[ 0.14 , -0.25 , 0.76 ]
```

Si ok: `pt++, cuenta=0`, empezamos bucle

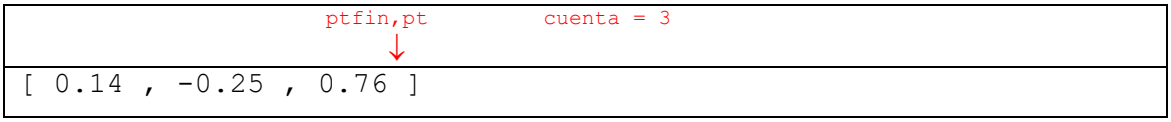
5.2.b) `valor=strtod(pt,&ptfin);`

```
 pt     ptfin                              cuenta = 0
 ↓      ↓
[ 0.14 , -0.25 , 0.76 ]
```

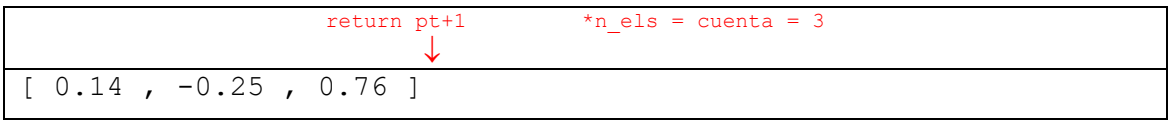
5.2.c) Si conversión Ok: `cuenta++; pt=SaltarEspaciosYComas(ptfin);`

```
      ptfin   pt                          cuenta = 1
        ↓     ↓
[ 0.14 , -0.25 , 0.76 ]
```

5.2.d) Repetir 5.2.b) y 5.2c) (bucle) mientras conversión Ok (ptfin!=pt)



5.2.e) Saltar espacios, comprobar `*pt==']'`, Si Ok: `pt++, *n_els=cuenta`, retornar pt+1



5.3) Usando la anterior, crear una función:

const char* RellenarTabla(const char* cadena,float tabla,int* n_els);**

que, tras obtener el número de elementos, asigne memoria y rellene el contenido de la tabla.

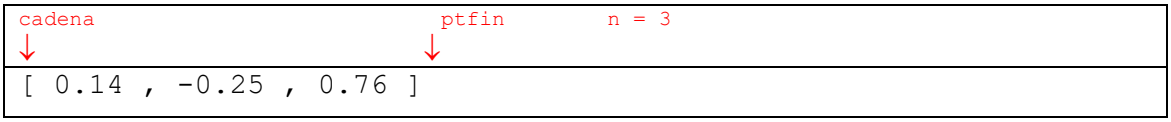
Ejemplo para prueba:

```
char texto[]="[ 0.14 , -0.25 , 0.76 ]";
```

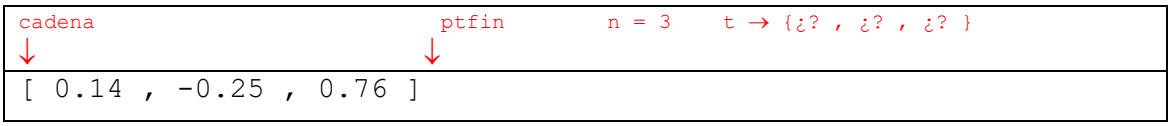
Secuencia de código y resultados previstos:

```
float* t;
int n;
```

5.3.a) `ptfin=ContarElementosEnTabla(cadena,&n);`

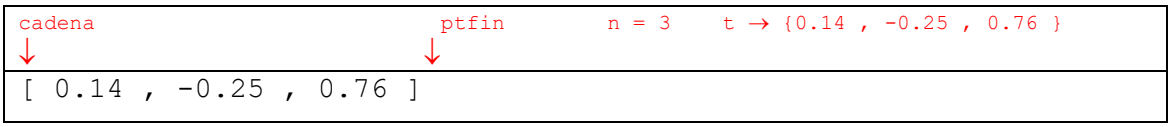


5.3.b) Si ok: `t=(float*) malloc(n_els*sizeof(float));`



5.3.c) Si ok: `pt=cadena+1, i=0`, hacemos bucle similar a 5.2.b) … 5.2.d), sólo añadiendo:

`tabla[i]=valor;`



5.3.d) Tras fin de bucle, asignar valores devueltos:

```
*tabla=t;
*n_els=n;
return ptfin;
```

5.4) Ahora hacer el comando RZ completo:

Ejemplo para prueba:

```
char cmd[]="RZ =    [ 0.14 , -0.25 , 0.76 ]      / [1    -0.28 ]";
```

Secuencia de código y resultados previstos:

```
float* tabla;
int n_els;
```

5.4.a) `pt = ObtenerValorComando(txt,"RZ");`

```
     pt
     ↓
RZ =    [ 0.14 , -0.25 , 0.76 ]      / [1    -0.28 ]
```

5.4.b) Si todo Ok: `pt = SaltarEspacios(pt);`

```
        pt
        ↓
RZ =    [ 0.14 , -0.25 , 0.76 ]      / [1    -0.28 ]
```

5.4.c) Si todo Ok: `pt = RellenarTabla(pt,&tabla,&n_els);` //

```
tabla → {0.14,-0.25,0.76}  n_els:3
                                        pt
                                        ↓
RZ =    [ 0.14 , -0.25 , 0.76 ]      / [1    -0.28 ]
```

Si todo ok y estamos en modo de control POS (para VEL algo similar):

```
free(_rz_pos.b);
_rz_pos.b=tabla;
_rz_pos.m=n_els-1;
Reasignar de la misma forma _ek_pos y poner a 0
```

5.4.d) pt = Saltar espacios, comprobar /, saltar espacios

```
                                        pt
                                        ↓
RZ = [ 0.14 , -0.25 , 0.76 ]      / [1    -0.28 ]
```

5.4.e) Si todo Ok, lo mismo pero ahora aplicado a _rz_pos.a , _rz_pos.n y _uk_pos (o VEL)