Trabajo Práctico N.º 1

Un escáner elemental

Programaremos un autómata finito determinístico (AFD) que reconozca tres lenguajes, el de las constantes enteras, el de los identificadores y el de numeral.

Lenguaje de las constantes enteras: son las cadenas formadas solamente por dígitos decimales
Lenguaje de los identificadores: son las cadenas que comienzan con una letra (no importa si mayúscula o minúscula) y siguen luego con letras o dígitos.
Lenguaje numeral: es un lenguaje formado solamente por un símbolo numeral

En el archivo a analizar pueden venir varios lexemas de cada uno de estos tres lenguajes. Los espacios en blanco simplemente se ignoran, sirven para dar legibilidad al separar los lexemas. Por espacios entendemos: espacios, tabuladores y salto de línea.

Pueden venir también caracteres no contemplados, por ejemplo asteriscos, operadores aritméticos, etc. En caso de encontrar un cadena formado por estos caracteres se considera que se encontró un error. El autómata debe informar error luego de recorrer todo el lexema y quedar listo para un nuevo escaneo. Es decir, en varios aspectos el error es similar a tener otro lenguaje más a reconocer.

El programa debe analizar la entrada (usaremos standar input y redirigiremos el archivo a analizar) e informar que va encontrando. Al final debe informar los totales encontrados de cada categoría léxica.

Ejemplo, dada la entrada:

```
total 123 Mesa34 @?¡<>@89640 SILLA56 *+%
,#    bien 571##$%
```

La salida del programa debiera ser:

```
identificador
constante entera
identificador
error
constante entera
identificador
error
error
Numeral
identificador
constante entera
Numeral
Numeral
error
----
Totales:
Identificadores 4
Constantes enteras 3
Numerales 3
Errores 4
```

Nota:
Para que quede claro, los lexemas detectados son:
Identificadores: `total, Mesa34, SILLA56, bien`
Constantes enteras: `123, 89640, 571`
Errores: `@?¡<>@  *+%  ,  $%`

Observaciones

Debe dividir el programa en al menos 3 archivos: `main.c, scanner.c y scanner.h`
`main.c` invoca al scanner , informa que token encontró y acumula las cantidades que informa al final.
`scanner.c` implementa el autómata devolviendo que token (categoría léxica) encontró. Pude abrirse en otros fuentes si quiere, por ejemplo para la función que calcula que columna del autómata se debe usar.

DEBEN documentar la tabla de transición a utilizar, en un documento de texto o planilla de cálculo, indicando claramente que es cada estado y como agruparon los caracteres en las columnas.

Pueden entregar un proyecto de CodeBlocks, antes de enviarlo compacte la carpeta del proyecto luego de **HABER BORRADO** las carpetas **bin** y **ojb** para no incluir ni ejecutables. En caso de usar otro entorno de desarrollo, solo envíe los fuentes.
En cualquier caso el programa debe compilar correctamente con el estandar C11 / C17 y debe apegarse estrictamente al estándar, sin usar nada que depende de un sistema operativo en particular.