Práctica 3: Lex

El programa que he implementado sirve para comprobar el precio de una carta del juego Magic: The Gathering en sus distintas ediciones, según la lista de precios de Metropolis Center la cual es una de las principales tiendas a nivel nacional. (Página de inicio -> http://www.mtgmetropolis.com/metropolistienda/index_tienda.asp).

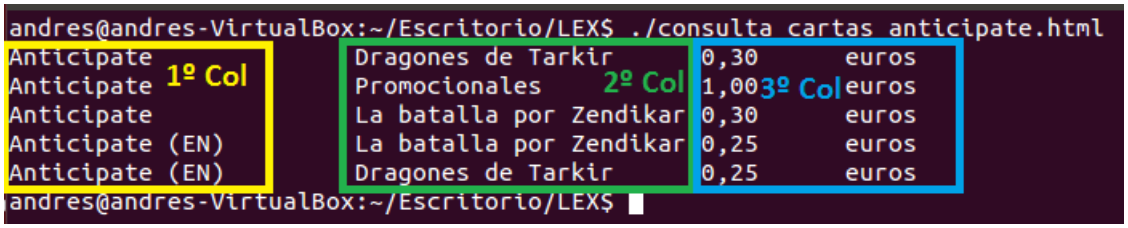
Aunque sería mejor que el programa recibiese el nombre de una carta y le hiciese un wget a la página buscando en la URL la carta pasada como argumento, no he podido hacerlo porque el tipo de URL que utiliza la página es .asp, y cuando buscamos una carta la URL sigue siendo .asp por lo que no puedo insertar la búsqueda directamente en la URL.

De modo que sobre lo que he trabajado para hacer el programa ha sido sobre HTMLs descargados con el navegador, es decir, busco una carta en la página, y la página que me muestra la descargo con el navegador (Firefox) dándole a guardar como…De esta manera consigo el HTML que me interesa, en el cual hay que buscar el nombre de la carta, la edición y el precio correspondiente a esa edición.

Ahora que ya sabemos cómo he conseguido el texto sobre el que trabajar, decir que la salida del programa es una tabla de 3 columnas que indican los siguiente:

- **1ª columna** -> el nombre de la carta en inglés, aunque a no ser que entre paréntesis ponga EN, la carta está en español. También pueden venir otras indicaciones como el estado de la carta, es decir, si ha sido jugada PLAYED, o ligeramente jugada SP, en caso de no venir estado, significa que la carta está nueva. Otra indicación sería si es FOIL o no, por defecto si no viene especificado significa que no lo es.
- **2ª columna** -> nombre de la edición de la carta. Esto se debe a que una carta puede estar publicada en varias ediciones distintas, y dependiendo de la edición a la que hagamos referencia el precio de la carta puede variar.
- **3º columna** -> precio de la carta en euros para el estado y edición que indica esa fila.

A continuación, vemos una ilustración que es un ejemplo de salida que daría el programa al buscar la carta llamada anticipar.



Como vemos nos encontramos que hay 5 tipos distintos de anticipar dependiendo de la edición y del idioma de la carta:

- La primera está en español, es de la edición Dragones de Tarkir y cuesta 0,3 euros.
- La segunda está en español, es de la edición de Promocionales y cuesta 1 euro.
- La tercera está en español, es de la edición La batalla por Zendikar y cuesta 0,3 euros.
- La cuarta está en inglés, es de la edición La batalla por Zendikar y cuesta 0,25 euros.
- La quinta está en inglés, es de la edición Dragones de Tarkir y cuesta 0,25 euros.

Todas las salidas son análogas a esta analizada.

En mi repositorio de GitHub se pueden encontrar varios HTMLs para probar el programa y darán una salida similar con sus datos correspondientes. Además, se puede ir a la página que he enlazado anteriormente, buscar cualquier carta de Magic, bajarse el HTML, pasárselo al programa como argumento y este tiene que dar un resultado correcto.

Las reglas definidas para encontrar lo que me interesa del HTML son las siguiente:

- style=.({especial}|{letra}|{digito})+.>[\b a-zA-Z(),]+<.a>
 { bool copia = false; string nombre; for(int i=0; i<yyleng; i++){ if(copia){nombre += yytext[i];} if(isspace(yytext[i])){copia = true;} } nombre.erase(nombre.end()-9,nombre.end()); nombres.push_back(nombre); } => sirve para encontrar la primera columna de la tabla (nombre, idioma y estado) meterlo en un string y añadir el string a un vector en el que iran todos los nombres escontrados.

- style=.({especial}|{letra}|{digito})+.>[\b a-zA-Z0-9]+<.a>\n **{ bool copia = false; string edicion; for(int i=0; i<yyleng; i++){ if(copia){edicion += yytext[i];} if(yytext[i] == '>'){copia = true;} } edicion.erase(edicion.end()-5,edicion.end()); ediciones.push_back(edicion); }** => sirve para encontrar la segunda columna (edición) meter la edición en un string y este en un vector de strings donde irán todas las ediciones en las que la carta está editada.

- align=.{letra}+.>[\.,0-9]+ **{ bool copia = false; string precio; int coincidencia = 0; for(int i=0; i<yyleng; i++){ if(copia && coincidencia>1){precio += yytext[i];} if(yytext[i] == '>'){copia = true; coincidencia++;} } precios.push_back(precio); }** => con esta encontramos el precio, lo introducimos en un string, este en un vector de strings donde irán todos los precios que tiene la carta con sus distintas características.

Además de estas reglas he tenido que dejar las que había en la plantilla de la práctica porque no me funcionaba lo de sobrescribir la regla por defecto, así que lo que he hecho ha sido sobrescribir las de la plantilla diciéndoles que no hagan nada. Así tendría las siguientes 3 reglas más:

- [^ \t\n]+ { }
- [\t]+ { }

- \n { }

Como vemos en las primeras 3 reglas, también he definido 3 alias, lo cuales son:

- letra [a-zA-Z]
- digito [0-9]
- especial [:#;]

Para finalizar adjunto el código completo del programa, para que pueda ser probado, además en mi repositorio de GitHub (https://github.com/Andresmag/Modelos_de_Computacion_UGR) podemos encontrarlo junto con varios archivos HTML para probarlo.

```
     /*----- Seccion de Declaraciones --------------*/
%{
#include <iostream>
#include <iomanip>
#include <vector>
#include <string>
using namespace std;

vector<string> nombres;
vector<string> ediciones;
vector<string> precios;

%}

digito     [0-9]
letra      [a-zA-Z]
especial   [:#;]

%%
     /*----- Seccion de Reglas ----------------*/
style=.({especial}|{letra}|{digito})+.>[\b a-zA-Z(),]+<.a><br>
          { bool copia = false; string nombre; for(int i=0;
i<yyleng;     i++){     if(copia){nombre     +=     yytext[i];}
if(isspace(yytext[i])){copia          =          true;}          }
nombre.erase(nombre.end()-9,nombre.end());
nombres.push_back(nombre); }

style=.({especial}|{letra}|{digito})+.>[\b a-zA-Z0-9]+<.a>\n
     { bool copia = false; string edicion; for(int i=0; i<yyleng;
i++){ if(copia){edicion += yytext[i];} if(yytext[i] == '>'){copia
=    true;}    }    edicion.erase(edicion.end()-5,edicion.end());
ediciones.push_back(edicion); }

align=.{letra}+.><strong>[\.,0-9]+                    {      bool
copia = false; string precio; int coincidencia = 0; for(int i=0;
i<yyleng; i++){ if(copia && coincidencia>1){precio += yytext[i];}
```

```
if(yytext[i]   ==   '>'){copia   =   true;   coincidencia++;}   }
precios.push_back(precio); }

[^ \t\n]+   { }
[ \t]+     { }
\n      { }

%%
      /*----- Seccion de Procedimientos --------*/
int main(int argc, char *argv[]){
      if(argc == 2){
            yyin = fopen (argv[1], "rt");
            if (yyin == NULL){
                  printf("El  fichero  %s  no  se  puede  abrir\n",
argv[1]);
                  exit (-1);
            }
      }
      else yyin = stdin;

      yylex();
      for(int i=0; i<nombres.size(); i++){
            cout << left << setw(20) << nombres[i] << "\t" << left
<<  setw(23)  <<  ediciones[i]  <<  "\t"  <<  left  <<  setw(10)  <<
precios[i] << "euros" << endl;
      }

      return 0;
}
```