UTN.BA UNIVERSIDAD TECNOLÓGICA NACIONAL FACULTAD REGIONAL BUENOS AIRES	**DPTO DE INGENIERIA en Sistemas de la Información** **Sintaxis y Semántica de los lenguajes**	**2021**	2° Cuatrimestre
	Trabajo Práctico de Laboratorio N° 2: Archivos HTML		

Trabajo Práctico de Laboratorio N° 2: Archivos HTML

Docentes	Mendez, Pablo Damián		
	Alumnos	**Legajo**	
	López, Axel Javier	166876-6	
Fecha	**07/10/21**		**Aprobación**

***Manual del usuario**: sistema operativo UNIX (**wget instalado**) ó Windows (**wget instalado**)*

*Trabajo realizado y compilado bajo el sistema operativo **macOS Big Sur.***

Estrategia de Resolución

Reporte B)

Comencé por este reporte ya que intuí por el tipo de datos que se solicitaba que al terminarlo tendría ya armado el "esqueleto principal" del algoritmo a utilizar en los reportes restantes.

En primer lugar utilicé el código base provisto por el profesor de la cátedra. En el mismo se abría el archivo HTML, se lo guardaba en una variable de tipo file, se comenzaba a leer el contenido por línea y, si la misma cumplía con las características requeridas, se comenzaba con su lectura carácter por carácter.

```
if (html = popen("wget -q -O - https://bolsar.info/lideres.php --no-check-certificate","r"))
{
    punto2 = fopen("punto2.csv","w");
    fprintf(punto2, "Especie;Precio de compra;Precio de venta;Apertura;Precio Mínimo;Precio Máximo\n");

    while (fgets(buffer,2048,html)) // Lee una línea y guarda en buffer.
    {
        if ((pchar = strstr(buffer, "data-order")) && strstr(buffer, "Cdo.</td>")) // Verifica que en la línea existan data order y celda al contado
        {
```

Aquí el mayor desafío consistió en definir qué anclaje utilizar para splitear la línea, y obtener así los datos requeridos, y con qué herramienta splitear dicha línea. De manera que se me ocurrió utilizar el **strtok** y splitear la línea anclándome en el **'/'**, carácter que supone el fin de una celda.

A partir de ahí debía identificar cuándo comenzar a leer el dato requerido con el substring obtenido con **strok**. Así fue cómo identifiqué que exactamente cada dos **'<'** el dato de las tablas estaba a mi alcance. De esta manera, simplemente con una estructura cíclica y contando los caracteres **'<'**, logré guardar los datos que cada reporte requería. (OBS: la variación requería de contar tres caracteres **'<'** para ser obtenida).

Una vez conseguido esto se recorría el string utilizando aritmética de punteros y se iba escribiendo en el archivo de salida los datos requeridos.

```
/*Escribamos el Vencimiento */

substring = strtok(NULL, "/");
while (i != 2){
    if (*substring == '>')
    {
        i++;
    }
    substring++;
}
while (*substring != '<'){
    //fprintf(punto2, "%c", *substring);
    substring++;
}
```

Esta estructura simplemente se repetía para cada dato (columna) de la tabla.

OBS: al tener que crear un archivo CSV teniendo que guardar elementos numéricos con el carácter decimal simbolizado con coma, seleccioné como carácter separador de celdas el punto y coma.

Reporte A)

Como señalé al principio de la descripción del reporte anterior, la mayor parte del problema ya estaba resuelta, solo restaba cambiar ciertas partes del código para conseguir el nuevo reporte.

En este caso ahora debíamos insertar una estructura condicional para solo escribir en nuestro archivo de salida las variaciones con valor negativo. De manera que solo cuando el substring del dato comenzara con un carácter **'-'** el dato sería escrito en el archivo de salida.

```
if (*substring == '-')
{
    fprintf(punto1, "%s;", especie);
    while (*substring != '<')
    {
    fprintf(punto1,"%c", *substring);
    substring++;
    }
    fprintf(punto1, "\n");
}
```

Reporte C)

La mayor dificultad aquí apareció a la hora de tener que guardar los datos de compra y venta para poder ser luego comparados con el dato de la variación (el hecho de tener que armar un HTML era simplemente agregar la estructura de la tabla a los fprintf, no suponían mayor dificultad). La investigación, y la ayuda de los compañeros de cursada, me llevó a encontrarme con la función **atof()**, la cual convierte un string en un dato de tipo double. La dificultad extra que suponía esta función es que necesitaba que el string presente como símbolo decimal el punto. Entonces, debí agregar al código un paso extra en el que se reemplace en el buffer de tipo string la coma leída por un punto:

```
int c = 0;
while (*substring != '<'){
    bcompra[c] = *substring;
    if (bcompra[c] == ',') // Reemplazo coma por punto para poder convertir el string en float.
    {
        bcompra[c] = '.';
    }
    c++;
    substring++;
}
```

Una vez guardadas las tres variables a comparar, creé una estructura condicional que marcaba lo siguiente:

1) Solo escribía en el archivo de salida si la variación era negativa.
2) Si la venta y la compra eran menores que la apertura la fila se escribía en verde.
3) Si la venta y la compra no eran menores que la apertura se escribía sin colorear.

```
if (variacion < 0)
{
    if (venta < apertura && compra < apertura)
    {
    fprintf(punto3, "<tr>\n");
    fprintf(punto3, "<td> <span style=\"color:#10da10;\"> %s </span></td>", especie);
    fprintf(punto3, "<td> <span style=\"color:#10da10;\"> %f </span></td>\n", variacion);
    fprintf(punto3, "</tr>\n");
    }
    else
    {
    fprintf(punto3, "<tr>\n");
    fprintf(punto3, "<td> <span> %s </span></td>", especie);
    fprintf(punto3, "<td> <span> %f </span></td>\n", variacion);
    fprintf(punto3, "</tr>\n");
    }
}
```

OBS: para colorear las filas en verde reutilicé el código de color de la variación provisto por el HTML de prueba.