Sintaxis y semántica de los lenguajes
Trabajo Práctico Nro. 2

WEB SCRAPING FINANCIERO (AGOSTO 2021)

Comisión: Ing. Pablo Damián Mendez.

Fecha de entrega: 07/10/21

Ciclo Lectivo: 2021

Alumno: Guido Reboredo

Legajo: 159-078.9

Parte 1: Informe Digital

INTRODUCCIÓN:

Para realizar la práctica de Web Scraping utilizando código en lenguaje “C” sin utilizar librerías existentes a tal fin, se realizó la siguiente estrategia:

En principio se generó la siguiente estructura de archivos en la carpeta “src”:

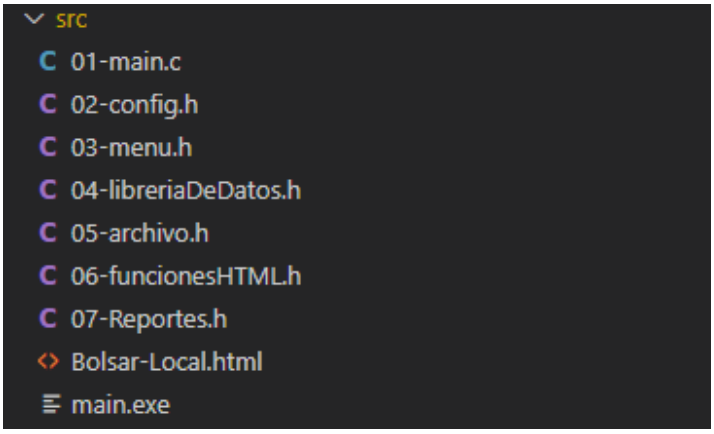

Imagen 1

01-main.c: Contiene la función “main” del código y un loop infinito que permanecerá iterando indefinidamente realizando la ejecución del MENÚ. En este archivo además se encuentra el include de las principales librerías estándar y del resto de los archivos “.h” listados en la *imagen 1*:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
```

02-config.h: Este archivo contiene la declaración de constantes y variables globales que se utilizan a lo largo del código y que pueden ser modificadas en caso de que así se lo requiera. Es importante mencionar las siguientes configuraciones:

```
#define GC_ARCHIVO "Bolsar-Local.html" //Archivo HTML Local enviado
por el docente

char CSV_PATH[]="cotizaciones.csv"; //Archivo de salida del reporte B

char HTML_PATH[] = "listaDeEspecies.html"; //Archivo de salida del reporte C

bool FILTER_CDO = true; //Indica si se desa o no filtrar solo las acciones que
contengan el Vto. = Cdo. (Contado Inmediato)
```

03-menu.h: En él se encuentra el switch que selecciona, en función del input de usuario, cual será el reporte que se mostrará en pantalla. Siendo las opciones:

```
------------------------------------------------------------
Para Comenzar, ingrese una opcion de acuerdo al siguiente menu y luego oprima enter:

1   -   Mostrar tabla de Acciones Completa
2   -   Reporte A (Especies con variacion negativa)
3   -   Reporte B (Generar tabla CSV)
4   -   Reporte C (Exportar tabla A en HTML)
5   -   Salir
```

Imagen 2

Cada opción corresponde a un punto del enunciado del archivo con el agregado de la opción 1: “Mostrar toda la tabla” y la opción 5: “Salir del Programa”.

04-libreriaDeDatos.h: La librería de datos contiene el modelo que se creo a fin de guardar cada una de las filas de la tabla de acciones. Para ello se creo una estructura con el nombre “accion” que representará una entrada de la tabla, cada entrada formará luego parte de un nodo en una lista simplemente enlazada por el campo **“accion *next”**.

Los campos de la estructura y su tipo de datos se obtuvieron de la observación de la tabla HTML del archivo fuente. Quedando de la siguiente manera:

```
struct accion
{
    char especie[6];
    char vto[4];
    int compra_cantidadNominal;
    float compra_precio;
    float venta_precio;
    int venta_cantidadNominal;
    float ultimo;
    char variacion[10];
    float apertura;
    float min;
    float max;
    float cierreAnterior;
    float volumen;
    int monto;
    char hora[8];
    accion *next;
};
```

A su vez se crearon las funciones killerListaAcciones, replace_char, deleteDot, myAtof y myAtoi con los siguientes objetivos:

killerListaAcciones: Limpia la memoria de la lista pasada como parámetro de entrada.

replace_char: Se recorre el string enviado como primer parámetro y se reemplaza el carácter enviado en el segundo parámetro por el que se envía en el parámetro número tres.

Esta función fue necesaria ya que los números en el archivo fuente tenían el punto decimal representado por "," (coma) pero las funciones estándar atoi y atof utilizadas para convertir los string a enteros o float respectivamente requieren que el separador sea un "." (punto).

deleteDot: Elimina los signos "."(punto) que se encuentran en el string enviado por parámetro. Esto fue necesario ya que los números del archivo fuente contenían puntos como separadores de millares.

myAtof y myAtoi: Estas funciones simplemente llamaban a las funciones estándares atoi y atof para convertir los strings a enteros o floats respectivamente. Pero antes se formateaba el string utilizando las funciones deleteDot y replace_char mencionadas anteriormente.

05-Archivo.h: Este archivo contiene las funciones que se utilizarán para abrir los 3 archivos necesarios durante la ejecución del programa principal, el HTML de entrada, el CSV de salida del punto B y el HTML de salida del punto C.

Cabe destacar que, en la primera función, que se encarga del HTML de entrada, se utiliza la constante global **GC_TIPO** (config.h) para definir si se abrirá el archivo de modo local con fopen o si se utilizara wget para obtenerlo de la URL definida en el archivo config.h . Así si GC_TIPO es igual a 1 se abrirá de modo local y si GC_TIPO = 2 se abrirá a través de la URL.

Además, puede revisarse el apartado de **config.h** para ver las constantes que definen los nombres de los archivos y sus rutas.

06-funcionesHTML.h: Este archivo contiene quizá las funciones más importantes del práctico, que se encargan de buscar y procesar el archivo HTML de entrada, así como también generar una lista enlazada con los valores de cada fila de la tabla de entrada.

Primero nos encontramos con la función **init()** que simplemente llama a HTMLtable2List() y retorna la lista generada por dicha función.

La función **HTMLTable2List()** abre el archivo HTML y avanza hasta la tabla utilizando la función **goToTableTag()** que utiliza la función estandar fgets para avanzar en el archivo hasta encontrar una línea que contenga el substring **"<thead>"**. Esta comparación se realiza en la función **findTableTag()** que compara utilizando la función estandar **strstr().** Una vez que llegue al tag <thead> avanza dos veces mas para estar posicionados en el comienzo de las líneas de la tabla.

Una vez que se llegó al comienzo de la tabla, la función **getTableRow()** será la encargada de crear un nodo del tipo accion ya mencionado anteriormente. Para esto primero que nada se realiza la asignación de memoria con la siguiente sentencia:

```
accion *nodo = malloc(sizeof(accion));
```

Luego se utiliza una combinación de la función strtok (estandar) para separar las líneas en los caracteres **“<”** y **“>”.** Estos son caracteres de apertura y cierre de tags html. Para eso se creo la función **findRowValue()** que utiliza strtok para obtener los valores encerrados únicamente entre dos tags **“<td>”** y **“</td>”.**

Los nodos se crean de forma iterativa mientras se recorre el archivo HTML, hasta que se encuentre el nodo “</table>” que indica el final de nuestra tabla.

Siempre que se termina de crear un nodo nuevo, se enlaza a la lista con la sentencia:

```
        if (nuevo != NULL)
        {
            aux1->next = nuevo;
            aux1 = nuevo;
        }
```

Al finalizar la función con éxito se cierra el archivo HTML y se retorna la Lista obtenida.

07-reportes.h: El archivo reportes contiene las funciones que se ejecutarán para cada opción del menú principal.

En él, se puede ver un apartado de funciones generales que permiten por ejemplo imprimir el título del reporte seleccionado (**printTitle()**), por ejemplo:

```
**********************************
REPORTE: 1
Descripcion:
Tabla de Especies
**********************************
```

Imagen 3

Otra que permite imprimir el encabezado de la tabla HTML de forma ordenada en la consola cmd (**printHeader()**)**.** Y, por último, otra función que permite imprimir un nodo de la estructura “accion” de forma ordenada y alineada con el header (**printNode()**).

Luego, cada reporte tiene su sección en la que se ejecutan los pasos para obtener la salida correspondiente a cada punto solicitado en el enunciado del práctico. Utilizando todas las funciones mencionadas hasta el momento.