

Ingeniería en Sistemas de Información

Cátedra Sintaxis y Semántica del Lenguaje

Curso K2054

TRABAJO PRÁCTICO N° 1

Web Scraping

Alumno: Martin Tomkinson-170863-6

Docente: Pablo Méndez

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fecha de entrega: | 16/07/2020 | Nota: |  | Fecha Aprobación: |  |

|  |
| --- |
| Comentarios: |

Tabla de contenido

Contenido

[Introducción 1](#_Toc44010043)

[Desarrollo 1](#_Toc44010044)

[Estrategias 1](#_Toc44010045)

[Problemas 1](#_Toc44010046)

[Soluciones 2](#_Toc44010047)

[Conclusiones 2](#_Toc44010048)

## Introducción

En el presente informe se detallarán las estrategias tomadas para el desarrollo del programa, el cual, tiene como objetivo analizar una web en tiempo real y extraer los datos necesarios luego, los mismos, serán exportados a un archivo o mostrados por pantalla según corresponda. También, se detallan los problemas encontrados durante el desarrollo y la solución por la cual se optó.

## Desarrollo del Trabajo práctico

### Estrategias

Al principio del desarrollo se realizó un análisis para identificar los distintos módulos que eran necesarios para el programa, los cuales son:

* Lectura
* Escritura
* Análisis de Tags HTML
* Condiciones de listado y formato
* Menús y elegir opciones

Para simplificar las primeras partes del desarrollo se decidió descargar la página web brindada para analizarla en profundidad y ver cómo está distribuido su contenido. Una vez realizado, se empezó a confeccionar el módulo de los tags HTML para poder reconocer las filas y columnas de manera adecuada.

A continuación, se procedió a los módulos de lectura y escritura para esto se decidió crear una variable con la capacidad de poder almacenar todos los campos de una fila, de esta manera se pueden verificar las condiciones que se piden para realizar los listados.

Luego, se desarrollaron las funciones que verifican, cuales, son los datos para listar y que formato deben tener.

Por último, se desarrolló la parte del menú y los distintos estados en los que puede estar el programa durante su ejecución, para así, permitir realizar varias consultas sin necesidad de volver a ejecutar.

Una cosa que siempre se tuvo en cuenta a la hora del desarrollo es el de utilizar Macros para evitar el *Hardcode* logrando una mejor mantenibilidad del programa. También se tuvo en cuenta que en el futuro puede ser necesario una nueva forma de exportar los archivos, uno de los problemas ocurridos está relacionado con la fácil actualización y modificación del programa.

### Problemas

Durante el desarrollo del programa se presentaron algunos problemas, los cuales, se listan a continuación:

* Error al leer caracteres con tildes
* Error al convertir números con coma a float
* Repetir código innecesariamente
* La lectura del campo se “llevaba” el ‘<’ del próximo tag HTML

### Soluciones

Al realizar la lectura, se compara lo leído (en caso de la fila que posee los encabezados) con los encabezados buscados para verificar que columnas son las que se necesitan listar y si es la tabla buscada. Al observar que las palabras con tilde fallaban continuamente en la comparación, se decidió imprimir de a un carácter, pero, mostrando su valor entero asociado a la tabla ASCII. Al realizar las pruebas se llegó al resultado que los caracteres con tilde leídos poseían los siguientes valores.

* -102 = Ú
* -77 = ó
* -95 = á
* -83 = í

Por esta razón, es que fue necesario crear una función, la cual, al leer uno de estos caracteres lo reemplace por la letra asociada, pero sin la tilde, cabe destacar que este valor no se correspondía a lo informado en la tabla ASCII, debido a que los mismos pertenecen al ASCII extendido. De esta manera se pudieron eliminar los acentos y, por consecuente, las fallas en las comparaciones de cadenas de caracteres.

Otro problema relacionado a formato fue el ocasionado por la diferencia entre el sistema de numeración, en el cual, los números racionales se escriben con punto a diferencia de Argentina donde se suelen escribir con coma. Es por esto, que fue necesario, convertir los números al formato esperado por C, para realizar la conversión a float, y así, hacer las comparaciones necesarias entre campos, una vez realizada la comparación se vuelve a convertir al formato original.

Una vez habiendo “finalizado” la construcción de todos los módulos, se observó una gran cantidad de lógica, y código repetido en las funciones principales. El mismo, se encuentra al final del main.c comentado. Para realizar la corrección, fue necesario utilizar punteros a funciones, para así, poder pasarlas como parámetro a otra función, de esta manera es más sencillo crear nuevas formas de exportar la información, siempre y cuando se respete el formato adecuado. También fue necesario operar el *stdout*, la cual es la consola, como un archivo utilizando *fprintf*, por ende, no era necesario independizar la función de impresión por pantalla de las otras.

El último problema encontrado fue que la lectura se llevaba el ‘<’ de inicio del siguiente tag, el cual siempre debía ser </td>, es por esto, que se decidió que la función de lectura de cada campo limpie el </td> en su totalidad, para así, en la próxima lectura leer el próximo tag.

El programa fue compilado con los siguientes flags:

-Wall -g -pedantic-errors -Wall -std=c99 -c

## Conclusiones

A la hora de realizar un programa, es necesario, conocer sus características y módulos principales, de esta manera, se realizará el desarrollo de la forma más eficiente centrándose en lo más importante.

Durante todo el desarrollo es de vital importancia no realizar *Hardcode* y evitar código repetido, puesto que esto ocasionará problemas a futuro a la hora de entenderlo y modificarlo, todo desarrollo de software se verá en la posición de actualizarse o morir, producto de los grandes cambios a nivel tecnológico, y nuevas necesidades por parte de los usuarios, que ocurren día a día.