**Algoritmos y programación III**

**Trabajo Práctico**

**TDA: Generación de archivos Json**

**Integrantes:**

* **Bellotti, Matias**

**Legajo: 35872**

**Mail:** [**matias.bellotti.35872@gmail.com**](mailto:matias.bellotti.35872@gmail.com)

* **Hansen, Nicolas**

**Legajo: 39448**

**Mail:** [**nicolas.han.sen@hotmail.com**](mailto:nicolas.han.sen@hotmail.com)

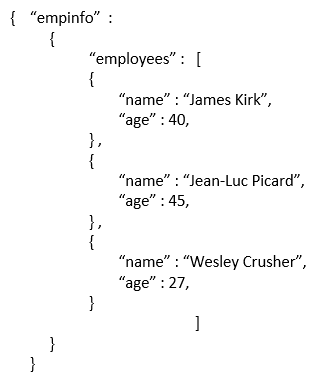
* **Juguera, Matias**

**Legajo: 40724**

**Mail:** [**mjuguera@hotmail.com**](mailto:mjuguera@hotmail.com)

*Introducción*

En el siguiente informe detallaremos el procedimiento llevado a cabo para realizar el trabajo práctico. La consigna de dicho trabajo es desarrollar un conjunto de TDAs que permitan operar con un conjunto de JSON, es decir, poder generar uno, modificarlo, imprimirlo y guardarlo. Un JSON (JavaScript Object Notation) es un formato de texto ligero para el intercambio de datos. Para llevar a cabo estas consignas, usamos el lenguaje de programación C aprendido durante la cursada, codeando en el IDE Eclipse.



Ejemplo de un Json

*Desarrollo*

Para el desarrollo del trabajo práctico, comenzamos a pensar al nJson como un TDA que tendría un nombre y un puntero a un nodo. Los nodos tendrían un nombre y una clave, cuya clave podría ser un array, un booleano, un float, un int, un string e incluso otro Json. Cada nodo, además, tendría una variable de tipo puntero a nodo, que señalaría a un “nodo siguiente”, de manera que cada nodo apuntaría, a su vez, al siguiente nodo de todo el Json. Esta idea principal fue más tarde descartada ya que nos encontramos con problemas al recorrer los nodos e intentar imprimirlos. La siguiente idea que tuvimos fue hacer una “lista” (implementada como un array) que contendría los nodos y se agrandaría con un “realloc” cada vez que sea necesario.

Los nodos son también un TDA, que contendría la clave y el valor del nodo, así como también la cantidad de elementos (uno en caso de ser un solo dato, o más en caso de ser un array) y un puntero a función. Dicho puntero a función apunta a un método que imprimirá solo el tipo de dato que se ingresó en el nodo. Quien ingrese un nodo en el nJson es responsable de indicar la función correcta al tipo de dato (es decir, si ingresamos un int, debemos ingresar la función de imprimir ints). Gracias a este puntero a función, nuestro programa sabrá cómo imprimir cada dato cuando sea necesario. Las funciones de impresión de cada tipo se encuentran en el main.

Para algunos tipos de datos tuvimos que utilizar algunas cosas extras. Por ejemplo, para poder usar booleanos correctamente, incluimos en nuestro programa la librería “stdbool.h”, que nos permite utilizar booleanos. Para los arrays, como se mencionó anteriormente, debemos indicar la cantidad de elementos que dicho array contendrá. Y para el caso de los strings, lo implementamos como un array de chars.

En el archivo Json.c, encontraremos otros métodos. Estos sirven para modificar un nodo existente, indicando la clave del nodo que se desea modificar y la nueva clave. Otros métodos funcionarán para liberar el Json y los nodos (utilizando la función free) que ya no necesitamos, para evitar goteo de memoria.

En el main encontraremos la implementación del ejemplo dado en la consigna del trabajo práctico, así como también un ejemplo de cada método para demostrar su correcto funcionamiento.

*Conclusión*

Durante el desarrollo de este trabajo práctico, pudimos consolidar conocimientos adquiridos durante la primera mitad de la materia. La solución de los problemas a los que tuvimos que enfrentarnos durante la realización, así como la programación del software en general nos sirvió también como una buena práctica para un lenguaje de programación nuevo para nosotros como es C.

***Directorio***: <https://github.com/Tps-UNTREF/TDA-generaci-n-de-archivos-JSON>

***Lista de clases***:

* main.c
* Json.c
* Json.h