**UNTREF**

# Algoritmos y Programación 3

*Trabajo Práctico Final: TDA generación de archivos JSON*

1er cuatrimestre, 2018

(Trabajo Grupal)

Alumnos:

|  |  |  |  |
| --- | --- | --- | --- |
| Nro | Nombre | Legajo | Mail |
| 1 | Maximiliano Andreoli | 39439 | andreolixam@gmail.com |
| 2 | Franco De Alesandro | 38583 | francodealesandro@@gmail.com |
| 3 | Andrés Ramos | 39412 | andresgabriel.92@gmail.com |

## Fechas de entrega programadas

* 1er Entrega : 5/5/2018
* 2da Entrega : 19/05/2018
* 3er Entrega : 9/6/2018
* 4ta y última Entrega : 30/06/2018

***Nota Final***:

# Introducción

## Objetivo del trabajo

Aplicar los conceptos enseñados en la materia a la resolución de un problema así como también la aplicación de buenas prácticas, trabajando en forma grupal y utilizando un lenguaje de tipado estático de bajo nivel (Lenguaje C)

## Consigna general

Desarrollar un conjunto de TDAs que permitan operar con un conjunto de JSON (JavaScript object notaton)) Una vez creado este conjunto de datos utlizarlo en la generación de un archivo JSON) La aplicación deberá ser acompañada con la apropiada documentación de diseño) En la siguiente sección se describe la aplicación a desarrollar)

Descripción de la aplicación a desarrollar

Definir un conjunto de TDAs que permitan generar el siguiente archivo JSON:

|  |
| --- |
| {  "size": "0 bytes",  "hash": "37eb1ba1849d4b0fb0b28caf7ef3af52",  "bytes": 0,  "thumb\_exists": false,  "rev": "714f029684fe",  "modified": "Wed, 27 Apr 2011 22:18:51 +0000",  "path": "/Photos",  "is\_dir": true,  "icon": "folder",  "root": "dropbox",  "contents": [  {  "size": "2.3 MB",  "rev": "38af1b183490",  "thumb\_exists": true,  "bytes": 2453963,  "modified": "Mon, 07 Apr 2014 23:13:16 +0000",  "client\_mtime": "Thu, 29 Aug 2013 01:12:02 +0000",  "path": "/Photos/flower.jpg",  "photo\_info": {  "lat\_long": [  37.77256666666666, -122.45934166666667  ],  "time\_taken": "Wed, 28 Aug 2013 18:12:02 +0000"  },  "is\_dir": false,  "icon": "page\_white\_picture",  "root": "dropbox",  "mime\_type": "image/jpeg",  "revision": 14511  }  ],  "revision": 29007  } |

*Consignas para el programa principal:*

Luego de definir apropiadamente las estructuras crear el programa principal que generará este archivo) Dicho programa podrá recibir como parámetro:

* **Ningún parámetro:** En cuyo caso el JSON se imprimirá en la consola (standard output)
* **Parámetro: -f <archivo\_destino:** En cuyo caso el JSON generado deberá escribirse en un archivo llamado *archivo\_destino*)

*La API para manejar JSON deberá proveer funcionalidad para realizar las siguientes tareas:*

1. Crear un nuevo *nodo JSON (en adelante nJson) asignándoles nombre y valor a nJSON.*

2) Cambiarle el contenido\* al nJson 3) Agregar nuevo nJson a un nJson existente

*\*Respecto a los contenidos posibles:*

1. nJson)
2. Cualquier tipo de dato punto flotante)
3. Cualquier tipo de dato entero)
4. Strings)
5. Booleans)
6. Arrays

a) De cualquier tipo entre 1-5 (solo un tipo por vez))

**NOTA**:

Los contenidos del nJson en ocasiones pueden ser Números enteros / Flotantes / Booleanos / Strings / etc) En estos casos el usuario de la API deberá poder operar con los tpos de datos natvos provistos por C que mejor se adapten al dato en cuestón)

## Pautas de calidad al momento de resolver el TP

* Los atributos del TDA deberán accederse mediante funciones provistas por la API **exclusivamente** (Applicaton Programming Interface))
* Dentro de las funciones que operan sobre el TDA/s evitar los llamados a funciones que impriman en las salidas estándar de errores o en la salida estándar (print, fprint, etc))
* Cada uno de los TDAs definidos deberán estar apropiadamente documentas, tanto sus campos como todas sus primitvas)
* Elegir un standard de codificación y seguirlo a lo largo del proyecto)
* Toda la memoria reservada dentro del TDA será responsabilidad de la API, es decir, si el TDA reserva memoria deberá proveer las herramientas necesarias para liberarla apropiadamente)
* Si necesita utilizar números mágicos, utilice #define y documéntelos apropiadamente)
* Los arrays utilizados dentro del TP deberán poder ser de una longitud indefinida (sin restricciones en su cantidad de elementos))
* La estructura del proyecto deberá ser:

o Archivo main)c con el ejemplo de uso de el/los TDA/s o Archivo <declaración del tda>)h o Archivo <definición del tda>)c

* El proyecto deberá ser implementado utilizando la interfase Eclipse CDT)

## Formas de entrega

**Habrá** 4 entregas formales. **Las mismas tendrán una calificación de** APROBADO o NO APROBADO **en el momento de la entrega. Cada uno de las entregas deberá contar con:**

* **Estructura de proyecto tal como lo crea el entorno Eclipse CDT.**
* **Este informe en el formato que lo distribuyó el docente.**
* **El informe debe contar con los datos de los integrantes del grupo.**
* **Cada uno de los requerimientos de la entrega deberán estar agrupados en un contexto (scope) independiente.**
* **La entrega se realiza digitalmente vía mail. Nombrar el archivo como**

**TP\_<apellido\_0>\_<apellido\_1>\_...\_<apellido\_n>.zip**

1er Entrega:

* Estructura completa del proyecto)
* Creación del main)c y la lectura de parámetros que provean la posibilidad de ejecutar la aplicación en los dos modos)

2da Entrega:

Tener la posibilidad de:

1. Entrega 1,
2. Crear un nJson
3. Poder asignarle cualquier contenido (salvo array) para todos los tipos primitivos de C) Agregar la posibilidad de poder imprimir el nJson)

3ra Entrega:

Tener la posibilidad de:

1. Entrega 2,
2. Cambiarle el contenido\* al nJson
3. Agregar nuevo nJson a un nJson existente
4. Poder asignar como contenido de un njson un array
5. Completar la salida por pantalla)
6. Crear otro JSON y mostrarlo en la salida utilizando el parámetro -C)

4ta y última Entrega:

Trabajo Práctico completo, funcionando, con el informe y cumpliendo todas las normas de calidad)

IMPORTANTE:

Si el TP no se puede ejecutar por cualquier tipo de error (acceso inválido de memoria, goteo de memoria, etc) el TP estará **desaprobado.**

## Fechas de entrega programadas

* 1er Entrega: 5/5/2018
* 2da Entrega: 19/05/2018
* 3er Entrega: 9/6/2018
* 4ta y última Entrega: 30/06/2018

Informe

**Supuestos:**

A los fines de resolver la consigna, partiremos del supuesto de que en el contexto de nuestro programa existirá un gestor de archivos json (en adelante simplemente "json") que estará representado por un modelo TDA. Este gestor poseerá la capacidad de crear, leer y almacenar nodos json así como también de generar archivos de salida (o bien impresiones en consola) que se deriven de la lectura de dichos nodos almacenados. Los nodos json (en adelante, "nJson") también estarán representados por un TDA y se ocuparan de almacenar datos (ya sean primarios como integers, chars, etc. u otros jsons).

## Modelo de dominio:

Nuestro diseño está compuesto de dos TDAs a saber: "JSON" y "nJSON". El primero, se define/declara en los archivos Json.h y Json.c mientras que el segundo lo hace en los archivos nJson.h y nJson.c. En cuanto a las responsabilidades de cada uno, como mencionamos en los supuestos, el TDA "JSON" se ocupará de almacenar nodos json y leer los datos contenidos en ellos. También, tendrá la atribución de imprimirlos todos juntos en consola o bien volcar su contenido en un archivo de salida con el nombre que desee el usuario. Por otra parte, el TDA "nJSON" se ocupará de retener información respetando un formato del estilo "nombre/clave: valor" y de la impresión y/o modificación de su contenido (funcionalidades que luego serán requeridas por el TDA "JSON").

## Detalles de implementación:

Para almacenamiento de los nodos, el TDA "JSON" se vale de un arreglo de punteros a nodos json cuya longitud se especifica al inicio. Posteriormente, con cada adición de un nuevo nodo, se efectúa el correspondiente realocamiento de memoria lo que permite que la dimensión del arreglo se modifique dinámicamente a medida que se va utilizando el programa. En lo que respecta a la lectura de los nodos, lo que se hizo fue recorrer el arreglo de punteros a nodos json y derreferenciar las direcciones de memoria de uno o de todos ellos según la operación que se desea realizar. Si lo que se quiere es modificar el contenido de un nodo, solamente se derreferenciará aquel nodo cuya clave coincida con la ingresada por el usuario. Luego, el TDA "JSON" invocará a la primitiva "njson\_set\_value\_by\_key" del nodo en cuestión. Por otra parte, si el objetivo es imprimir, los nodos se irán derreferenciando e imprimiendo uno a uno mediante la invocación de la primitiva "njson\_print" de cada uno de ellos. Para la impresión de nodos que poseen como valor otro nodos, el programa se vale de llamados recursivos a la primitiva imprimir haciendo la tratativa que corresponda según el tipo de dato adjudicado a los sucesivos valores que se van detectando en cada llamado.

Nota: véase en el código, que por razones de prolijidad y claridad se definió un enumerable que tipifica a los valores "false" y "true" como valores propios del tipo “bool”.

## Checklist de corrección

Esta sección es para uso exclusivo de los docentes, por favor no modificar)

### Carpeta

#### Generalidades

* ¿Son correctos los supuestos y extensiones?
* ¿Es prolija la presentación? (hojas del mismo tamaño, numeradas y con tipografía uniforme) **Modelo**
* ¿Está completo? ¿Contempla la totalidad del problema?
* ¿Respeta encapsulamiento?
* ¿Pierde memoria?
* ¿Cumple con las buenas prácticas?

## Código

### Generalidades

* ¿Respeta estándares de codificación?
* ¿Está correctamente documentado?
* ¿Define magic numbers? ¿Están documentados?  ¿Respeta estructura del proyecto?