UNIVERSIDAD DE CARABOBO
Facultad Experimental de Ciencias y Tecnología
Departamento de Computación
Unidad Académica de Algoritmos y Programación
CAO403: Programación II

Profesor: Álvaro Espinoza

PROYECTO
(Valor: 10%)

1. Instrucciones

a. El ejercicio propuesto requiere la lectura de datos USANDO ARCHIVOS.
b. Para el desarrollo del proyecto debe utilizar el lenguaje de programación C++, junto con sus librerías estándar. Para la compilación de sus códigos fuentes, debe realizarla por medio de un archivo **makefile**, el cual deberá entregar junto con sus códigos fuentes. De no poseer makefile la entrega, el taller no será revisado.
c. Su código debe estar debidamente comentado.
d. Debe utilizar lowerCamelCase para las variables, y UpperCamelCase para la creación de tipos de datos (en caso de necesitarlos).
e. Puede usar Estructuras Lineales, Jerarquicas o Multienlazadas para resolver el problema.
f. Debe modularizar su código de tal forma que se evite el "código spaghetti".
g. El archivo makefile debe generar un ejecutable con el nombre "proyecto", con el fin de facilitar las labores de corrección de su taller, el no hacerlo generará puntos menos en su nota final del taller.
h. La fecha de entrega máxima es el día lunes 10 de Octubre de 2021 a las 11:59pm. Se restará un punto por cada hora de retraso.
i. Debe enviar un archivo comprimido .zip con los archivos necesarios y un documento .txt con su nombre y cédula al correo: aespinoza3@protonmail.com

2. Enunciado

Wikipedia define **JSON** (acrónimo de **JavaScript Object Notation**, 'notación de objeto de JavaScript') como un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera un formato independiente del lenguaje.

Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de

texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

Sintaxis

Los tipos de datos disponibles con JSON son:

- Números: Se permiten números negativos y opcionalmente pueden contener parte fraccional separada por puntos. Ejemplo: 123.456
- Cadenas: Representan secuencias de cero o más caracteres. Se ponen entre doble comilla y se permiten cadenas de escape. Ejemplo: `"Hola"`
- Booleanos: Representan valores booleanos y pueden tener dos valores: `true` y `false`
- null: Representan el valor nulo.
- Array: Representa una lista ordenada de cero o más valores los cuales pueden ser de cualquier tipo. Los valores se separan por comas y el vector se mete entre corchetes. Ejemplo `["juan","pedro","jacinto"]`
- Objetos: Son colecciones no ordenadas de pares de la forma **<nombre>:<valor>** separados por comas y puestas entre llaves. El nombre tiene que ser una cadena entre comillas dobles. El valor puede ser de cualquier tipo.

Ejemplo de JSON:

```
{
  "departamento":8,
  "nombredepto":"Ventas",
  "director": "Juan Rodríguez",
  "empleados":[
    {
      "nombre":"Pedro",
      "apellido":"Fernández"
    },{
      "nombre":"Jacinto",
      "apellido":"Benavente"
    }
  ]
}
```

Es importante resaltar que el JSON en general es enviado entre componentes en formato de cadena de caracteres, pero respetando el formato mencionado arriba. Por lo que en la actualidad los lenguajes de programacion poseen “parsers” que básicamente son programas que procesan el texto JSON y lo convierten en objetos propios del lenguaje.

¿Es posible parsear JSON y manipularlos en C++?

La respuesta es que si, dependiendo de la versión de C++ habrán versiones más sofisticadas, en el caso de este proyecto se RECOMIENDA la siguiente librería:

https://github.com/gregjesl/simpleson

Su uso es sumamente sencillo, solo se debe copiar los archivos json.cpp y json.h y ya puede ser importada sencillamente, aquí un ejemplo de juguete:

```
#include <iostream>
#include "json.h"
using namespace std;

int main()
{
    // variables declaration
    string input = "{ \"hello\": \"world\" }";
    json::jobject result = json::jobject::parse(input);

    //code
    string value = result.get("hello");
    cout << value;
}
```

En este caso el ejemplo sencillo lo que imprimirá será la cadena “World”. SE RECOMIENDA leer la documentación de uso en el link, y también revisar el código fuente con el fin de entender nombres de métodos que hayan podido ser cambiados.

¿Qué se pide?

Para una pequeña práctica del uso de JSON se pide leer un archivo “.json” jerárquico y realizar los recorridos PREORDEN, POSTORDEN, INORDEN y POR NIVELES del árbol asociado a la estructura JSON jerárquica.

3. Formato de Entrada

Se requerirán dos archivos, el primero se llamará “entrada.json” y será un JSON BIEN FORMADO jerárquico, en donde tendremos datos de personas y sus subordinados, por ejemplo:

```
{
    "nombre": "Alvaro",
    "apellido": "Espinoza",
    "cedula": "24500719",
    "subordinados": [
        {
            "nombre": "Giuliana",
            "apellido": "Belli",
            "cedula": "5468339912",
            "subordinados": []
        },
        {
            "nombre": "Martin",
            "apellido": "Rodriguez",
            "cedula": "1698972",
```

```
            "subordinados": [
                {
                    "nombre": "Juan",
                    "apellido": "Cano",
                    "cedula": "16989723",
                    "subordinados": []
                }
            ]
        },
        {
            "nombre": "Angel",
            "apellido": "Oropeza",
            "cedula": "78982",
            "subordinados": []
        }
    ]
}
```

El segundo será sencillo, se llamará "entrada.in" y contendra sólo una de estas dos palabras: NOMBRE ó CEDULA. Si la palabra es NOMBRE entonces por cada nodo se imprime es la concatenación sin espacio del campo "nombre" y el campo "apellido", por ejemplo, en el caso del nodo principal del JSON de arriba sería: AlvaroEspinoza. Y en caso de ser CEDULA entonces se imprime el campo "cedula"

4. **Formato de Salida**

La salida serán cuatro (4) líneas con cada recorrido del arbol descrito en "entrada.json" mostrando el campo escrito en "entrada.in" para cada nodo, precedido por el nombre del recorrido en mayúscula seguido de dos puntos, cada palabra entre comillas, separadas por coma y espacio y el recorrido en sí debe estar entre corchetes. Por ejemplo, en el caso del JSON anterior con la palabra NOMBRE:

PREORDEN: ["AlvaroEspinoza", "GiulianaBelli", "MartinRodriguez", "JuanCano", "AngelOropeza"]
INORDEN: ["GiulianaBelli", "AlvaroEspinoza", "JuanCano", "MartinRodriguez", "AngelOropeza"]
POSTORDEN: ["GiulianaBelli", "JuanCano", "AngelOropeza", "MartinRodriguez", "AlvaroEspinoza"]
NIVELES: ["AlvaroEspinoza", "GiulianaBelli", "MartinRodriguez", "AngelOropeza", "JuanCano"]