练习项目(json-data-structure)

项目简介

设计JSON数据结构，方便读取成员值

JSON是一种树形数据结构。具体JSON格式可参考：

- w3school.com.cn
- json.org

一个JSON，包含一个JSON值类型。

JSON值是一种可变类型，可以是：BOOL值，数值，字符串，数组，对象。

数组的成员类型是JSON值，也就是说一个数组，可能包含0到n个元素，每个元素都是JSON值。

对象由1到n个键值对组成，键值对由键（字符串）和值（即JSON值类型）组成。

类型名	含义	说明	范例
JSON	JSON值	Value的别名	
Value	值类型	变体类型，可分为BOOL值，数值，字符串，数组，对象5种	`true`, `3.14`, `"200.200.0.1"`, `[1,2]`, `{"enable":true}`
Array	数组类型	`[]`括起来的就是数组	`[1,2,3]`, `[{"name":"huanan"}, {"name":"huabei", "ip": "200.0.0.254"}]`
Object	对象类型	`{}`括起来的就是对象	`{"enable": true, "ip": "200.200.3.61", "fd": -1, "dns": ["200.200.0.1"]}`
KeyValue	键值对	由`"Key": JSON`组成	`"enable": true`, `"ip": "200.200.0.1"`
BOOLValue	BOOL值类型	`true/false`	`true`, `false`
Num	数字类型	浮点数	`-1`, `133333333333`, `3.14`
Str	字符串类型	`""`括起来的就是字符串	`"huanan"`, `"200.200.0.1"`

JSON的EBNF定义：

```
JSON ::= Value
Value ::= BOOLValue | Num | Str | Array | Object
Array ::= '[' Value (',' Value)* ']'
Object ::= '{' KeyValue (',' KeyValue)* '}'
KeyValue ::= Key ':' Value
Str ::= '"' [^"]* '"'
```

练习要求

1. 支持接口

实现json.h中要求的接口，视需要可以增加或修改API。

接口支持对JSON值的增、删、改、查。

支持从JSON值中获取任意一个子孙成员的键值。

```
JSON *basic = json_get_member(json, "basic");     //basic对应的对象
JSON *dns = json_get_member(basic, "dns");        //dns对应的数组：["200.200.0.1",
"200.0.0.254"]
JSON *ip0 = json_get_element(dns, 0);             //数组中第0个IP地址："200.200.0.1"
JSON *ip1 = json_get_element(dns, 1);             //数组中第1个IP地址："200.0.0.254"
```

2. 输出结果

实现一个demo，该demo输出一个JSON值的YML格式文件。

需补充完善项目文件夹下的demo.c文件。

用最简单的写法（尽量少的代码）利用设计出来的API(json.h中的函数)构建一个复杂的JSON值。

完成后的demo.c，必须有完善的异常处理逻辑，

比如，能构建出如下JSON值：

```
{
    "basic": {
        "enable": true,
        "ip": "200.200.3.61",
        "port": 389,
        "timeout": 10,
        "basedn": "aaa",
        "fd": -1,
        "maxcnt": 133333333333,
        "dns": ["200.200.0.1", "200.0.0.254"]
    },
```

```
    "advance": {
        "dns": [
            {"name":"huanan", "ip": "200.200.0.1"},
            {"name":"huabei", "ip": "200.0.0.254"}],
        "portpool": [130,131,132],
        "url": "http://200.200.0.4/main",
        "path": "/etc/sinfors",
        "value": 3.14
    }
}
```

然后，将上述JSON值输出为YML格式文件。

3. 扩展能力

保留有序列化的扩展能力。

将来需要扩展序列化能力，比如支持从文件中读入JSON值，以及把API构建的JSON值输出为YML格式文件。

4. 设计要求

做到数据隐藏，除了API，不允许通过别的途径操作JSON值里面的内容，需要考虑设计怎么做好限制。

5. 单元测试

需完成每个API的单元测试，单元测试覆盖率不低于90%

注意事项

本练习项目分两个阶段，预写阶段，重构阶段。

1. 预写阶段要求

编码实训之前，学员先整体完成该项目，达成上述“练习要求”中所列的各项要求。

2. 重构阶段要求

编码实训过程中，学员花四天时间学习公司编码规范要求，基本流程方法，并对项目代码进行重构。

根据授课过程讲到的要求对预写阶段产出的代码进行重构。

2.1. 掌握基础设计方法：接口和数据结构的设计

本习题的目的是让大家掌握一种设计/实现通用数据结构，以及为模块设计API的方法。

2.2. 掌握编码六步法

掌握6步法这种基本的编程套路，减少BUG，提高工作效率。

2.3. 掌握编码风格的基本要求

需要考虑API的适用场景是什么，怎么让设计出来的API更好用，更容易调测，更容易扩展，更可靠。

2.4. 关注点

练习的主要目的，不是掌握调试BUG的技巧，而是掌握少出BUG，少调BUG的技巧。

完成练习的关键在于掌握基本的编码套路：API设计改进方法，编码、单测方法，以及利用单测进行调试的方法，而非里面的算法。

如果对算法不够清楚可以问，如果出现bug调试超过10分钟，赶紧从方法上下手改进。 比如编写单测案例，让每个单测涉及的代码量更少（意味着出错需要定位的代码范围小），检查更充分。 比如利用valgrind检查内存泄露和指针访问相关错误。 等等。