📖FISHJSON使用手册

概览

FISHJSON是什么

FISHJSON是一个基于C语言开发的，用于处理C和JSON数据的开源库。

FISHJSON做什么

通过FISHJSON，你可以实现以下操作

1. 将JSON文本转化为C语言数据结构
2. 根据C语言数据结构生成JSON文本
3. 通过C语言对JSON数据进行修改

FISHJSON为了谁

FISHJSON主要为有处理JSON数据需求的C语言开发者提供便利

FISHJSON的使用环境

✅Visual Studio 2019

✅Win10 64

✅ECMA-404 JSON标准

🛌开始

🔍一、下载FISHJSON库

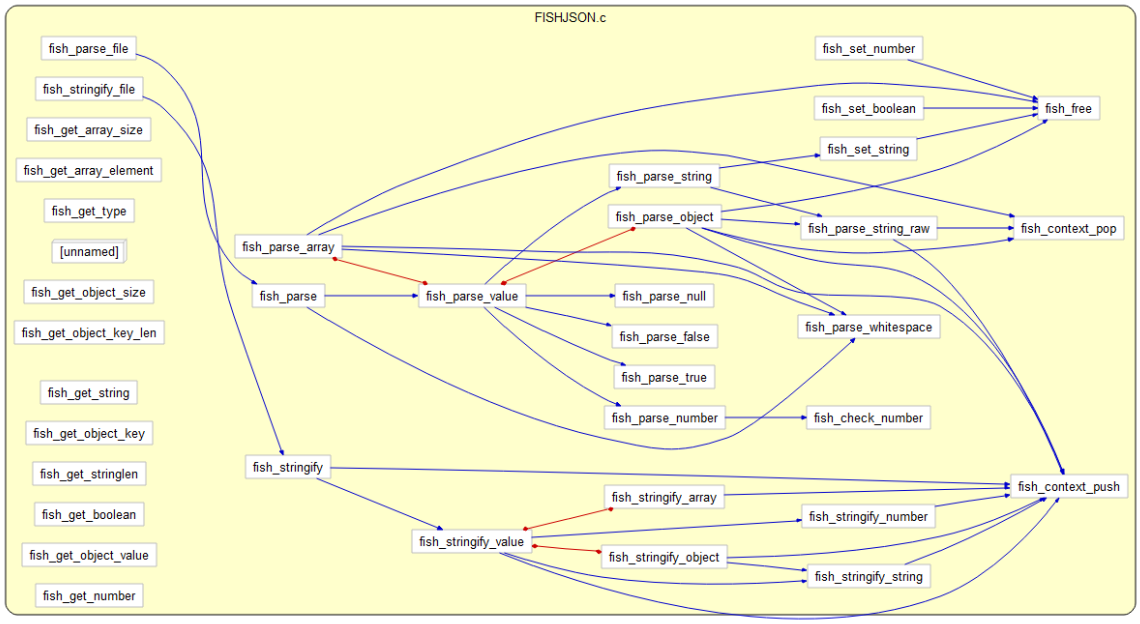
您可以在GITHUB上找到FISHJSON的开源项目并下载相关文件

❤FISHJSON

🗃二、熟悉各个文件

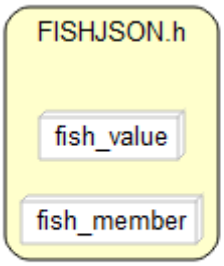
FISHJSON.c

主体库文件，包含了FISHJSON各种API的具体实现。



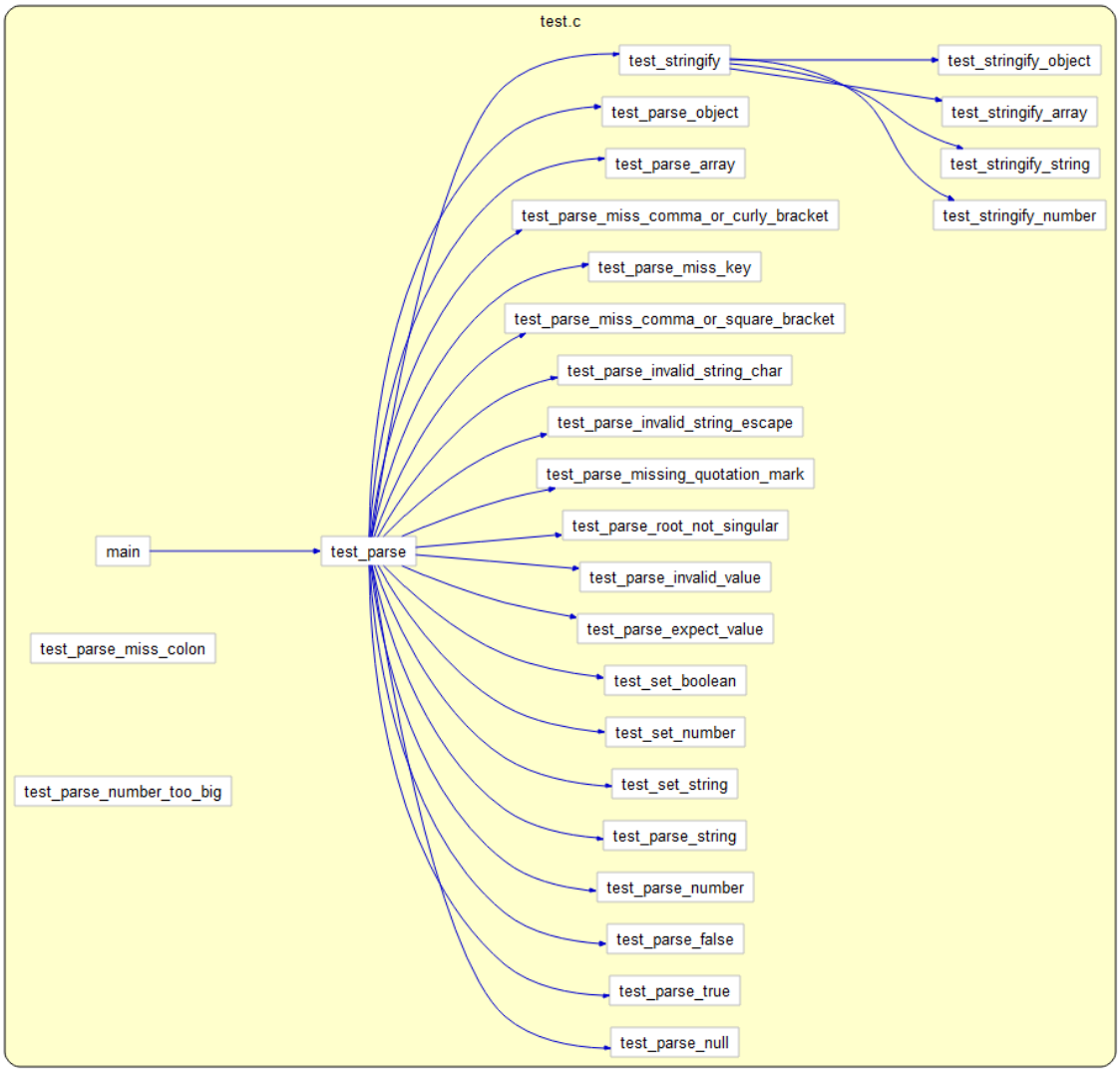
FISHJSON.h

头文件，包含了FISHJSON库的主要函数声明，以及FISHJSON的数据结构。



test.c

测试文件，包含了开发过程中的测试样例。同时您可以通过test文件学习每个API的使用方法



demo.c

演示文件，包含了一些API的使用样例

三、数据结构介绍

FISHJSON自定义了树状的数据结构来存储JSON文本数据

fish_value

```
typedef struct fish_value fish_value;
typedef struct fish_member fish_member;

struct fish_value{
    char* s; unsigned int len;              //string
    fish_value* e; unsigned int arr_size;   //array
    fish_member* m; unsigned int obj_size;  //object
    double n;
    fish_type type;
};
struct fish_member {
    char* key; unsigned int key_len;
    fish_value v;
};
```

四、API介绍

作为使用者，您不需要对每个API了如指掌，接下来将介绍主要的功能函数。

解析API

fish_parse()

```
int fish_parse(fish_value* v, const char* json);
```

将JSON文本解析成C语言数据结构

☒注意不支持解析含UTF-8的字符串

生成API

fish_stringify()

```
char* fish_stringify(const fish_value* v, unsigned int* length);
```

根据C语言数据结构生成JSON文本

☒注意JSON文本无美化

访问API

你可以通过这些访问函数获取JSON数据元素的值,或者将某些值设为您想要的数据。

```
double fish_get_number(const fish_value* v);
void fish_set_number(fish_value* v, double n);


int fish_get_boolean(const fish_value* v);
void fish_set_boolean(fish_value* v, int b);

const char* fish_get_string(const fish_value* v);
unsigned int fish_get_stringlen(const fish_value* v);
void fish_set_string(fish_value* v, const char* s, unsigned int length);

unsigned fish_get_array_size(const fish_value* v);
fish_value* fish_get_array_element(const fish_value* v, unsigned pos);

unsigned int fish_get_object_size(const fish_value* v);
const char* fish_get_object_key(const fish_value*, unsigned int pos);
unsigned int fish_get_object_key_len(const fish_value* v, unsigned int pos);
fish_value* fish_get_object_value(const fish_value* v, unsigned int pos);

fish_type fish_get_type(const fish_value* v);
```

文件API

fish_parse_file()

```
int fish_parse_file(fish_value* v,const FILE* fp);
```

fish_string_file()

```
void fish_stringify_file(const fish_value* v,FILE* fp);
```

1. fish_parse_file解析不支持UTF-8的string类型☒
2. fish_stringify_file生成JSON文本没所行和美化☒
3. 调整文件输入/输出大小
   您可以通过修改FISH_FILE_BUFFER来调整文件缓冲区大小

```
#ifndef FISH_FILE_BUFFER 1024
#define FISH_FILE_BUFFER 1024
#endif
```

内存管理API

fish_free()

您可以通过fish_free()来释放C数据结构的指针

```
void fish_free(fish_value* v);
```

五、返回值介绍

你可以根据这些返回值了解程序的运行情况

```
/*枚举解析的返回值*/
enum {
    FISH_PARSE_OK = 0,
    FISH_PARSE_EXPECT_VALUE,
    FISH_PARSE_INVALID_VALUE,
    FISH_PARSE_ROOT_NOT_SINGULAR,
    FISH_PARSE_NUMBER_TOO_BIG,
    FISH_PARSE_MISS_QUOTATION_MARK,
    FISH_PARSE_INVALID_STRING_ESCAPE,
    FISH_PARSE_INVALID_STRING_CHAR,
    FISH_PARSE_MISS_COMMA_OR_SQUARE_BRACKET,
    FISH_PARSE_MISS_KEY,
    FISH_PARSE_MISS_COLON,
    FISH_PARSE_MISS_COMMA_OR_CURLY_BRACKET
};
```

返回值	含义
FISH_PARSE_OK	解析成功
FISH_PARSE_EXPECT_VALUE	期待一个值
FISH_PARSE_INVALID_VALUE	非法的值
FISH_PARSE_ROOT_NOT_SINGULAR	部分解析
FISH_PARSE_NUMBER_TOO_BIG	数字过大
FISH_PARSE_MISS_QUOTATION_MARK	缺少引号
FISH_PARSE_INVALID_STRING_ESCAPE	string中非法空格
FISH_PARSE_INVALID_STRING_CHAR	string中非法字母
FISH_PARSE_MISS_COMMA_OR_SQUARE_BRACKET	缺少方括号或者逗号
FISH_PARSE_MISS_KEY	缺少键
FISH_PARSE_MISS_COLON	缺少冒号
FISH_PARSE_MISS_COMMA_OR_CURLY_BRACKET	缺少花括号或者逗号

反馈

FISHJSON起初仅仅作为一个简单的课程设计被开发出来，因此存在着一定的缺陷。如果您想反馈使用过程中的缺陷，可以直接在Github中的Issues中提出来，方便我们的更新与处理。