

**软件开发综合训练**

**实验报告**

|  |  |
| --- | --- |
| 实验名称： | Json解析器设计 |
| 学 号： | 21281123 |
| 姓 名： | 曾伯文 |
| 学 院： | 计算机与信息技术学院 |
| 日 期： | 2023.7.5 |

## 一、需求分析

### 1.问题描述

Json（JavaScript Object Notation）是一种轻量级的数据交换格式，广泛应用于各种应用程序中。为了方便处理Json数据，我们计划开发一个Json工具，提供Json文件的读取、解析、格式化、校验和生成等功能。

开发一个简单易用的Json工具，满足以下功能需求：

1. 读取Json文件并将其解析为内部数据结构

2. 将内部数据结构转换为格式化的Json字符串

3. 校验Json数据的合法性

4. 根据给定的数据生成Json文件

### 2.功能需求

#### 2.1 读取Json文件

（1）用户能够选择一个Json文件进行读取

（2）工具能够将Json文件的内容解析为内部数据结构

#### 2.2 格式化Json字符

（1）用户能够将内部数据结构转换为格式化的Json字符串

（2）工具能够保持良好的缩进、换行和对齐格式

#### 2.3 校验Json数据

（1）用户能够输入Json字符串进行校验

（2）工具能够检查Json数据的语法和结构是否合法

（3）错误信息能够清晰地提示给用户

#### 2.4 生成Json文件

（1）用户能够根据给定的数据生成Json文件

（2）工具能够将内部数据结构转换为符合Json规范的文件内容

（3）用户能够指定生成的文件路径和文件名

### 3.非功能需求

#### 3.1 用户友好性

（1）工具应该提供直观、易于理解的用户界面

（2）错误处理应该友好，给出明确的错误提示

#### 3.2 性能

（1）工具应具备较高的解析和生成速度

（2）对于大型Json文件，工具应能够高效处理

#### 3.3 可扩展性

工具应具备良好的可扩展性，方便后续添加更多功能和特性

### 4.开发环境

开发语言：推荐使用C++，因其性能和跨平台特性

功能实现：使用Visual Studio 2019开发功能实现代码

用户界面：使用Qt creater框架，该平台能够提供丰富的图形界面组件和跨平台支持

本工具不使用Json第三方库，通过自定义Json类来实现功能

### 5.开发过程

**（1）需求分析和设计**

在需求分析阶段，我们明确了Json工具的功能需求和非功能需求。根据这些需求，我们进行了设计，包括用户界面设计、数据结构设计和功能模块设计。

**（2）Json类的设计和开发**

设计并实现了自定义的Json类，用于处理Json文件的读取、解析、格式化、校验和生成功能。Json类内部包含了类型定义、数值定义、运算符重载、类型判断、类型转换等一些私有成员变量和方法，用于存储和操作Json数据。

**（3）Json文件的读取和解析功能开发**

在Json类中实现了读取Json文件的功能，通过读取文件内容将Json数据存储到内部数据结构中。同时，实现了解析Json数据的功能，将Json数据解析为内部数据结构。

**（4）格式化Json字符串功能开发**

在Json类中实现了将内部数据结构转换为格式化的Json字符串的功能。通过递归遍历内部数据结构，将数据按照一定的格式转换为字符串，并添加缩进、换行等格式化信息。

**（5）校验功能开发**

在Json类中实现了Json数据的校验功能。通过递归遍历内部数据结构，检查Json数据的语法和结构是否合法，并将校验结果返回给用户。

**（6）生成Json文件功能开发**

在Json类中实现了根据给定的数据生成Json文件的功能。通过将内部数据结构转换为符合Json规范的字符串，将字符串写入到文件中。

**（7）界面设计和开发**

使用Qt creater框架进行界面设计和开发，根据需求设计了主窗口和相应的控件，如文件选择按钮、文本框、校验按钮等。实现了界面的布局和交互逻辑。

**（8）测试和修复漏洞**

进行了单元测试和集成测试，发现了一些bug和漏洞，并进行了修复。确保工具的稳定性和可靠性。

（9）文档编写和发布

编写了开发文档，包括需求分析文档和开发过程文档。准备发布Json工具，提供用户文档和使用指南。

## 二、概要设计

### 1.总体设计

我们将自定义一个Json类来实现Json文件的读取、解析、格式化、校验和生成功能。Json类将包含以下功能模块：

读取功能：从Json文件中读取内容并返回。

解析功能：将Json文件内容解析为Json对象。

格式化功能：将Json对象格式化为易读的Json字符串。

校验功能：检查Json对象的合法性和完整性。

生成功能：根据Json对象生成新的Json文件。

**1.1 读取功能**

输入：Json文件路径

输出：Json文件内容

实现：通过文件读取操作，将Json文件内容读取到内存中。

**1.2 解析功能**

输入：Json文件内容

输出：Json对象

实现：根据Json的语法规则，通过递归解析Json文件内容，构建Json对象。

**1.3 格式化功能**

输入：Json对象

输出：格式化后的Json字符串

实现：通过递归遍历Json对象，按照一定的缩进和换行规则，将Json对象转换为格式化后的Json字符串。

**1.4 校验功能**

输入：Json对象

输出：校验结果（合法/不合法）

实现：根据Json的语法规则，逐个检查Json对象的键值对，判断是否符合规范。

**1.5 生成功能**

输入：Json对象

输出：生成的Json文件

实现：通过递归遍历Json对象，将键值对按照Json的语法规则转换为字符串，并写入文件中。

### 2.类的定义

#### 2.1 Json类

**属性：**

Type m\_type;

Value m\_value;

m\_type是自定义的枚举类Type类型，用于记录元素的数值类型。m\_value是自定义的联合类Value类型，用于存储元素的值。Type类和Value类的定义如下：

**Type类：**

|  |
| --- |
| enum Type  {  json\_null = 0, //空类型  json\_bool, //布尔类型  json\_int, //整型  json\_double, //浮点型  json\_string, //字符串  json\_array, //数组类型  json\_object //对象类型  }; |

**Value类：**

|  |
| --- |
| union Value  {  bool m\_bool; //布尔值  int m\_int; //整型值  double m\_double; //浮点值  std::string\* m\_string; //字符串值  std::vector<Json>\* m\_array; //数组值，使用vector容器实现  std::map<string, Json>\* m\_object; //对象值，使用map实现  }; |

**方法：**

|  |
| --- |
| //返回当前元素的类型  Type type() const;  //判断元素类型  bool is\_null() const;  bool is\_bool() const;  bool is\_int() const;  bool is\_double() const;  bool is\_string() const;  bool is\_array() const;  bool is\_object() const;  //返回对应元素类型的值  bool as\_bool() const;  int as\_int() const;  double as\_double() const;  string as\_string() const;  //获取当前数组或对象的元素个数  int size() const;  //如果为空数组、空对象、NULL，返回true  bool empty() const;  //清空json变量的内容  void clear();  //判断索引对应的值是否存在  bool has(int index) const;  bool has(const char\* key) const;  bool has(const string& key) const;  //获取对应索引的值  Json get(int index) const;  Json get(const char\* key) const;  Json get(const string& key) const;  //移除对应索引的值  void remove(int index);  void remove(const char\* key);  void remove(const string& key);  //在数组末尾添加元素  void append(const Json& value);  void append(Json&& value);  //运算符重载  Json& operator = (const Json& other);  Json& operator = (Json&& other);  bool operator == (const Json& other);  bool operator != (const Json& other);  Json& operator [] (int index);  Json& operator [] (const char\* key);  Json& operator [] (const string& key);  operator bool();  operator int();  operator double();  operator string();  operator string() const;  //Json解析函数  void parse(const string& str);  //Json格式化函数  string str\_p() const;//包含字体颜色  string str\_f() const;//不包含字体颜色  //Json对象的复制  void copy(const Json& other);  //Json对象与另一个对象交换值  void swap(Json& other); |

#### 2.2 Parser类

**属性：**

string m\_str;

size\_t m\_idx;

m\_str用于存储待解析的Json文件字符串。m\_idx用于记录当前解析字符串的位置。

**方法：**

|  |
| --- |
| //将str的值赋给m\_str  void load(const string& str);  //解析函数  Json parse();  //跳过无用字符  void skip\_white\_space();  //获取当前字符，并指向下一个字符  char get\_next\_token();  //各类型单独解析  Json parse\_null();  Json parse\_bool();  Json parse\_number();  string parse\_string();  Json parse\_array();  Json parse\_object(); |

### 3.接口设计

**3.1 parse(const string& str)**

输入：从Json文件中读取的字符串

输出：解析后的Json类对象

功能：将Json文件的读取内容解析到一个Json类对象。

**3.2 str\_p()**

输入：无

输出：Json数据格式化结果

功能：将Json文件的解析结果按照固定格式（包含字体颜色）输出到字符串流当中

**3.3 str\_f()**

输入：无

输出：Json数据格式化结果

功能：将Json文件的解析结果按照固定格式（不包含字体颜色）输出到字符串流当中

**3.4 on\_pushButton\_clicked()**

输入：图形界面输入的Json文件字符串

输出：图形界面输出框的格式化结果

功能：对图形界面输入的Json文件字符串进行解析、校验、格式化，并将格式化结果输出到窗口的输出框中。

**3.5 on\_pushButton\_3\_clicked()**

输入：图形界面输出框的格式化结果

输出：格式化结果文件

功能：读取输出框的格式化结果，将其存入新的Json文件

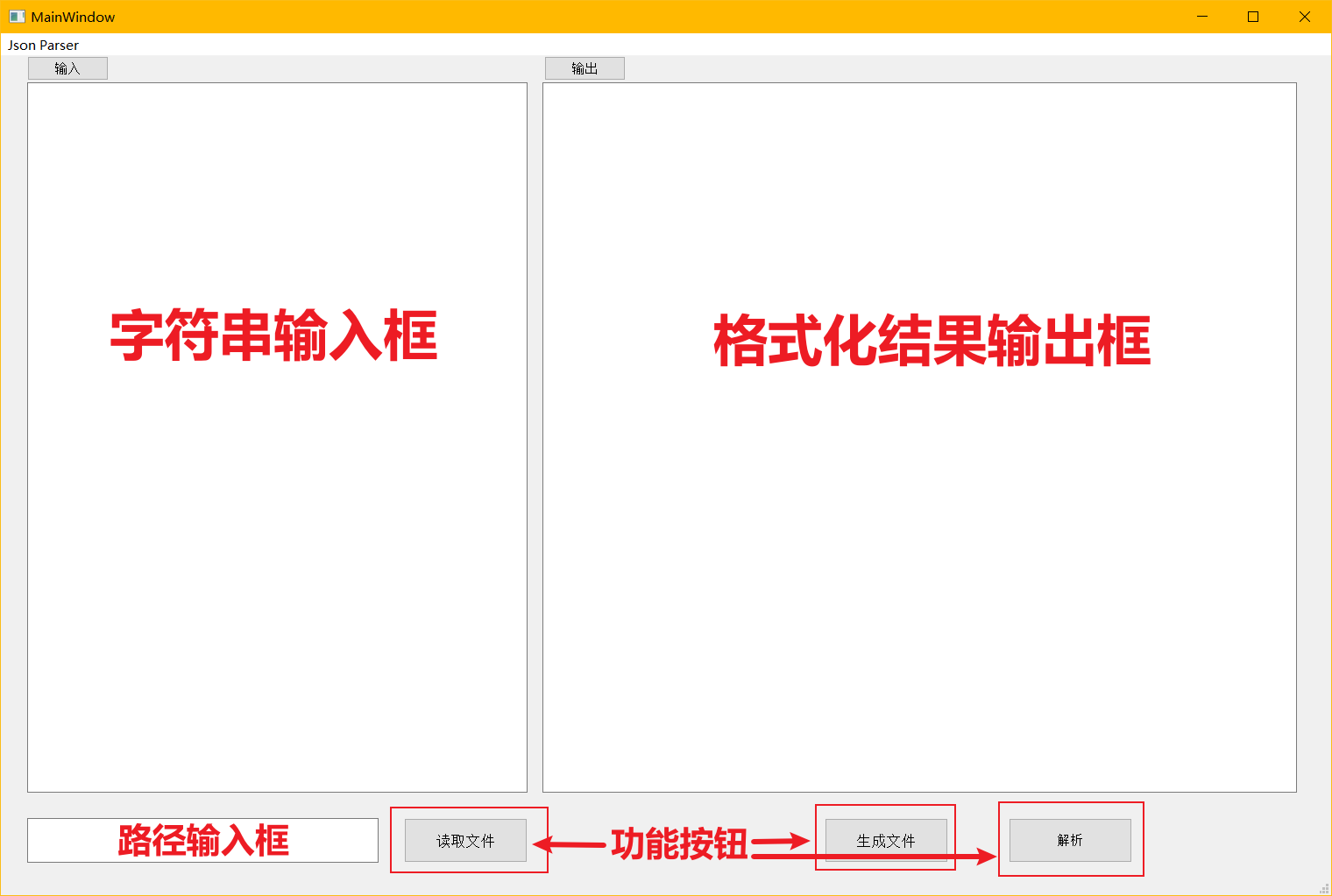
**3.6 on\_pushButton\_5\_clicked()**

输入：Json文件路径

输出：图形界面输出框的格式化结果

功能：对指定路径的Json文件字符串进行解析、校验、格式化，并将格式化结果输出到窗口的输出框中。

### 4.运行界面设计



运行界面包括3个文本框，分别为字符串输入框、格式化结果输出框和路径输入框，其中字符串输入框和路径输入框可编辑，格式化结果输出框不可编辑，读取文件、生成文件功能分别由“读取文件”按钮和“生成文件”按钮控制，“解析”按钮控制解析、校验、格式化功能。

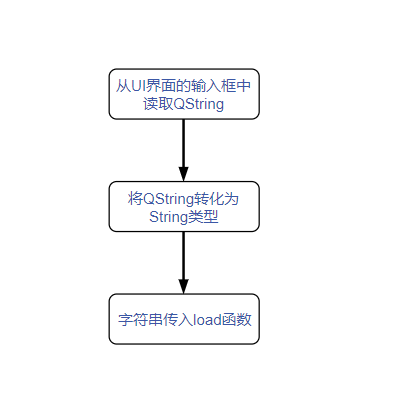
## 三、详细设计

### 读取模块：

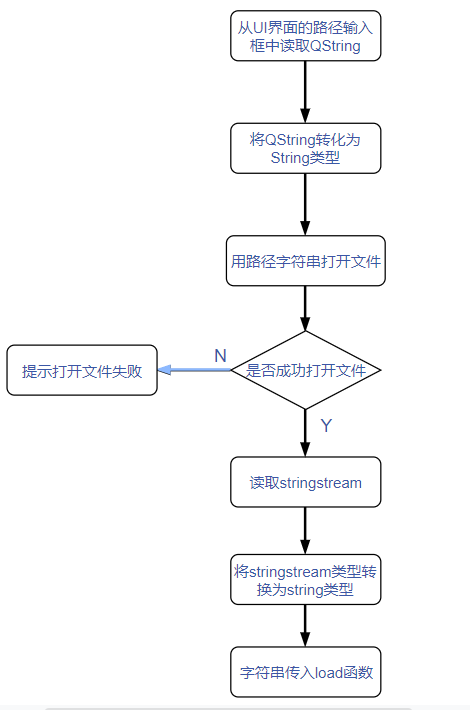
**1.装载函数load(const string& str)**

将从外部读入的Json文件字符串存入Parser类的m\_str中，供后续的解析使用

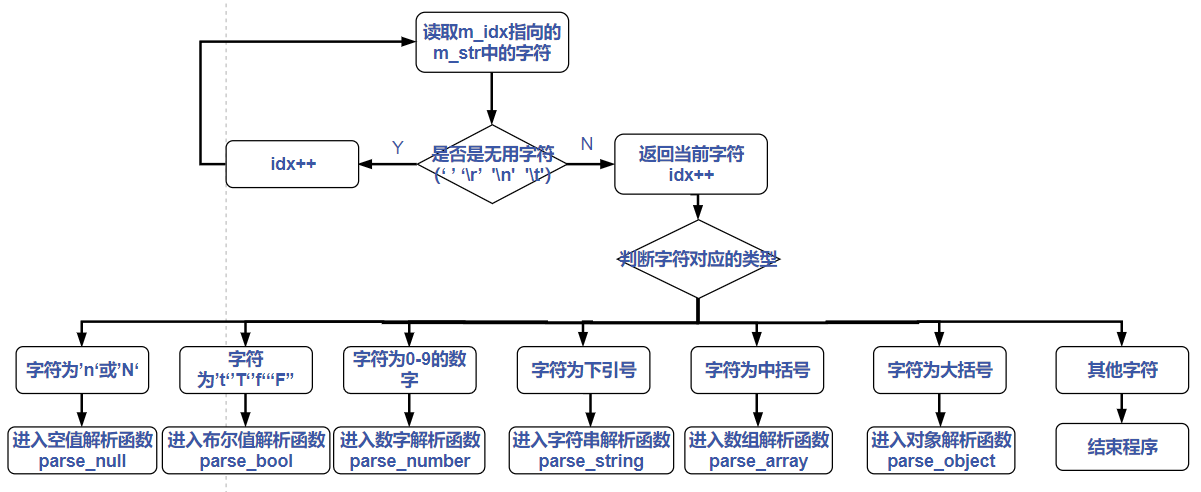
**2,输入框读取**



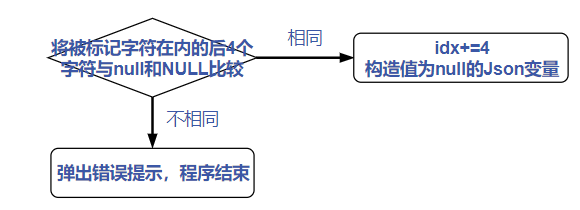
**3.文件读取**



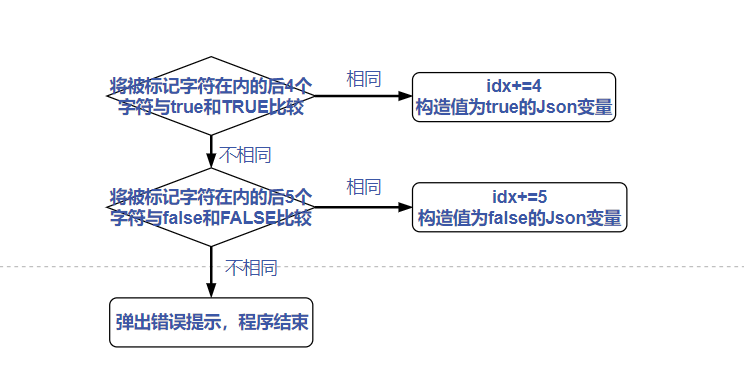
### 解析校验模块：

**解析函数流程图：**

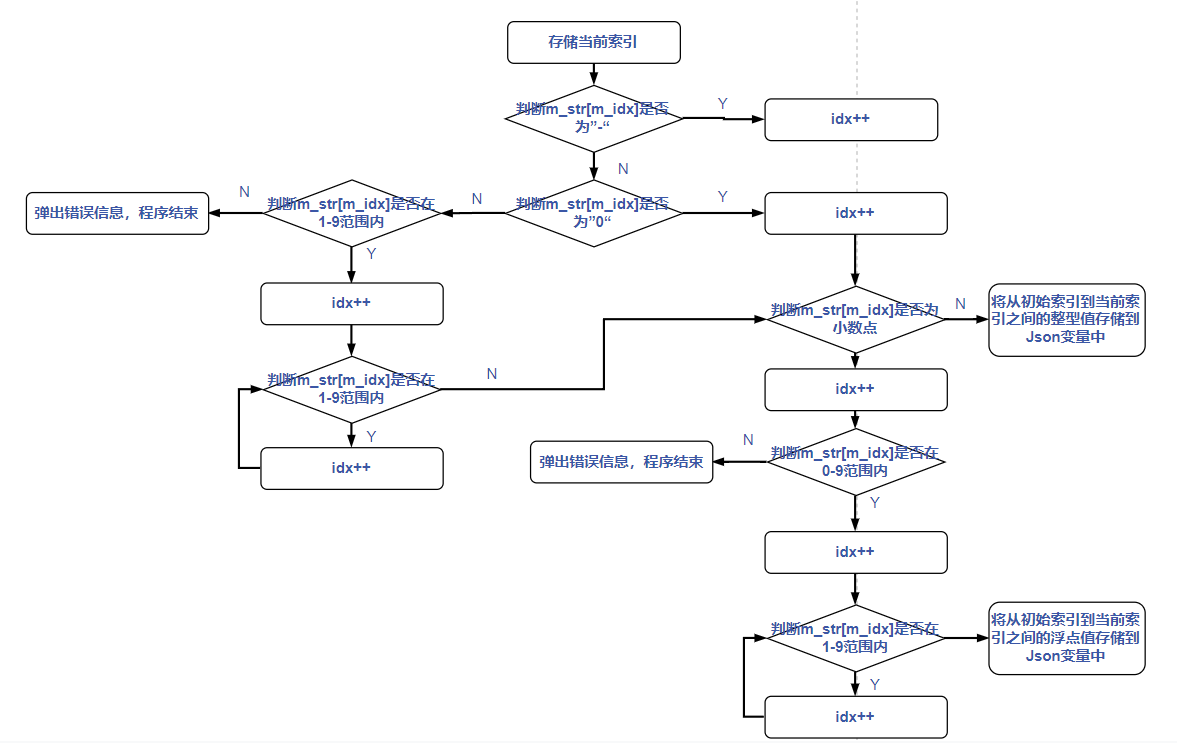
**空值解析函数parse\_null()：**



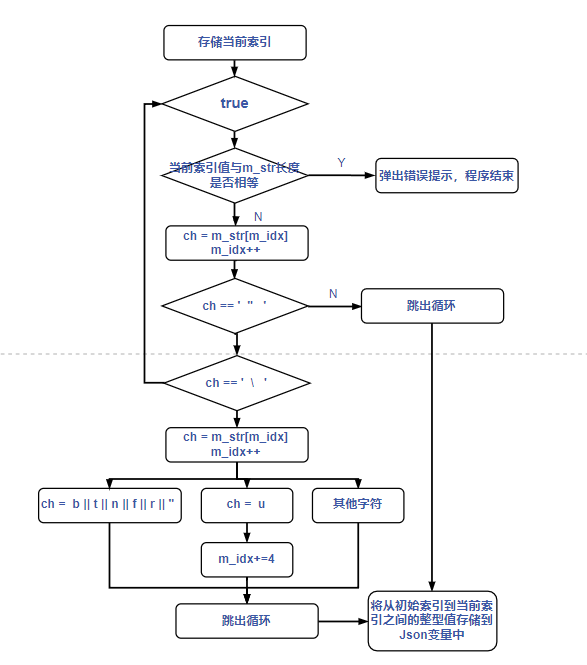
**布尔值解析函数parse\_bool():**



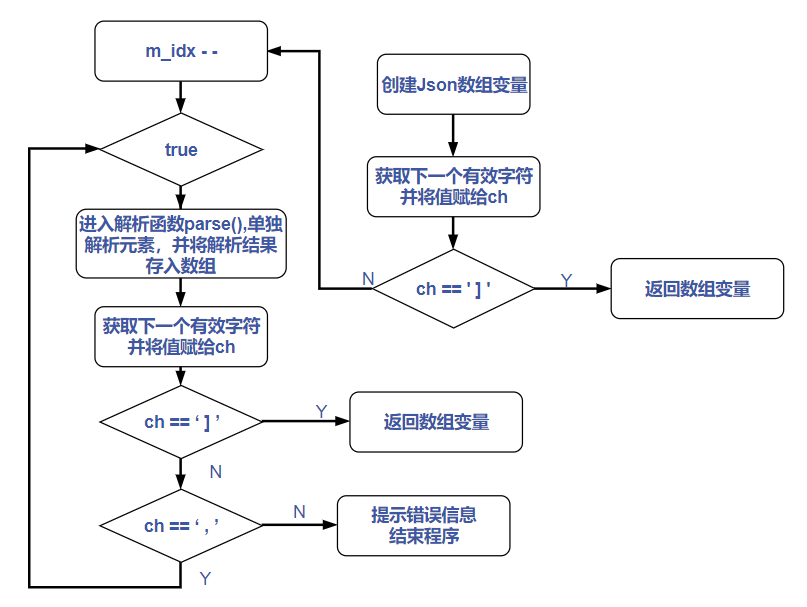
**数字解析函数parse\_number():**



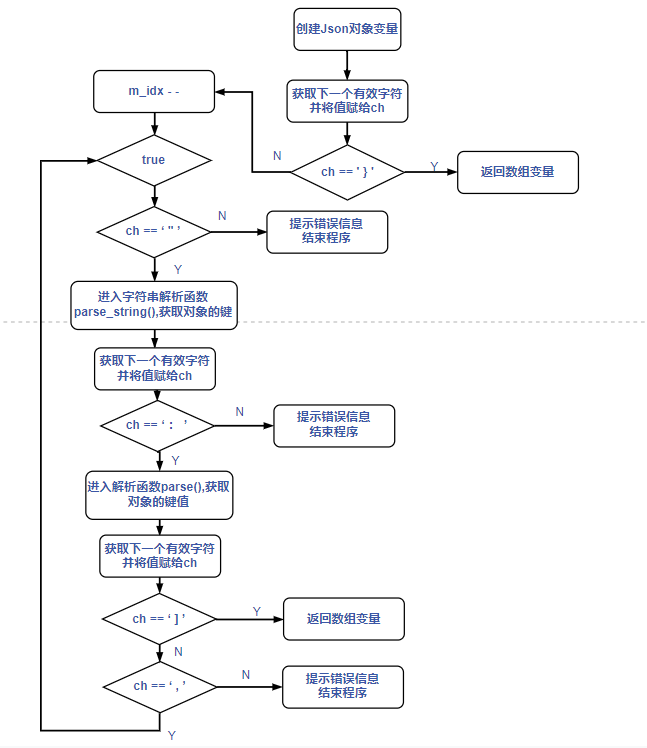
**字符串解析函数parse\_string():**



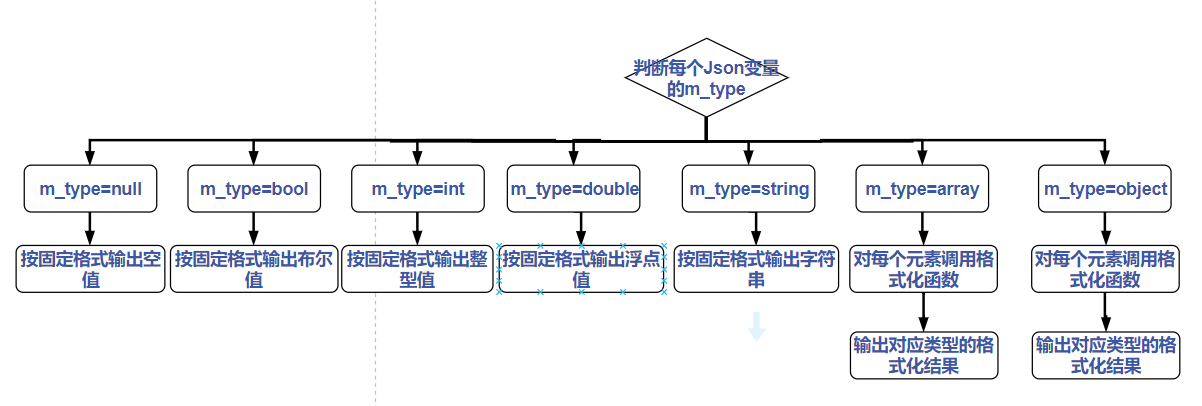
**数组解析函数parse\_array():**



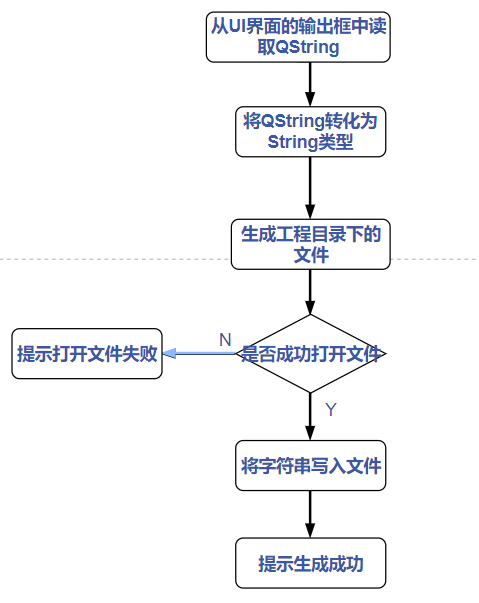
**对象解析函数parse\_object():**



### 格式化模块：



### 生成文件模块：



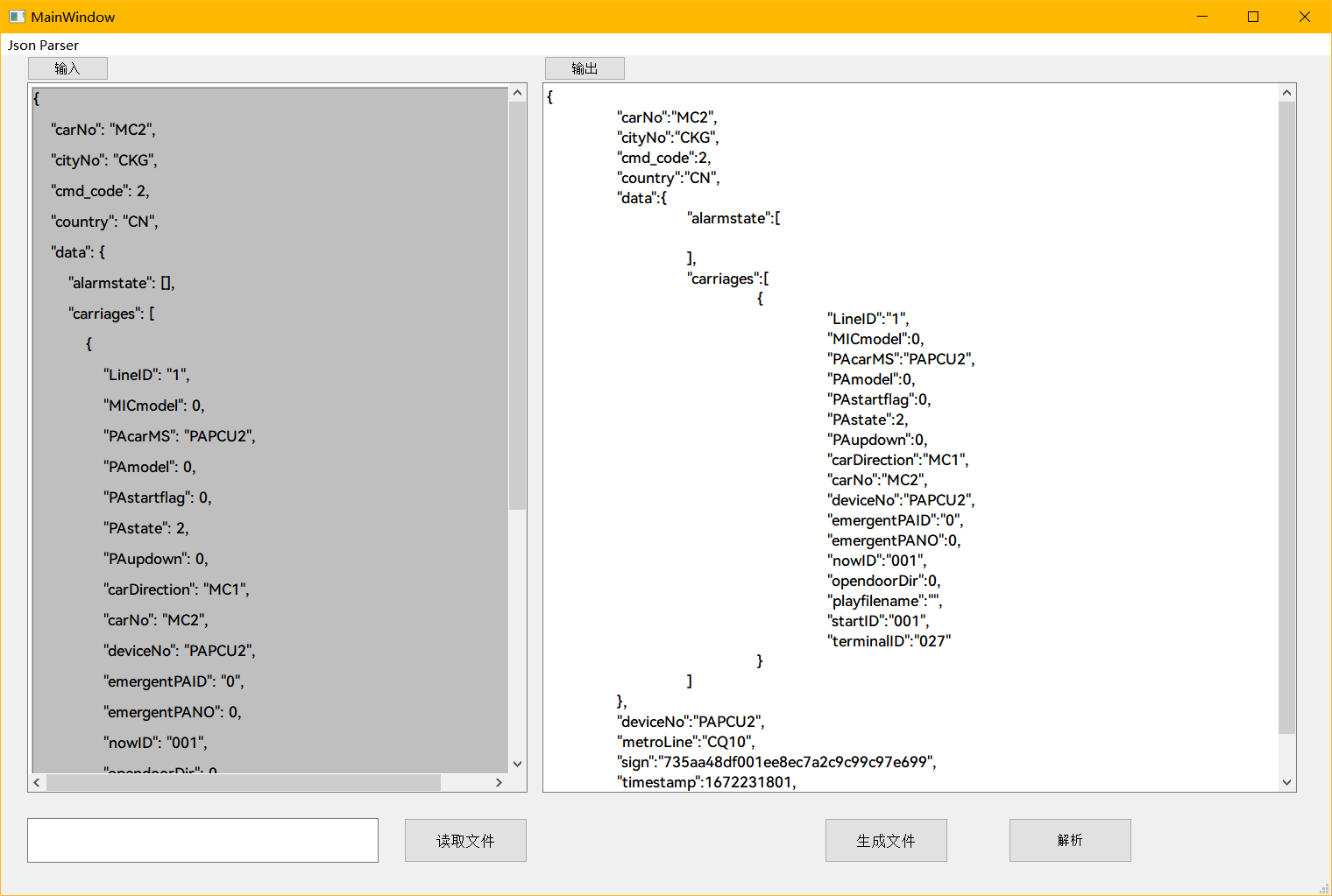
## 四、测试分析

### 功能测试

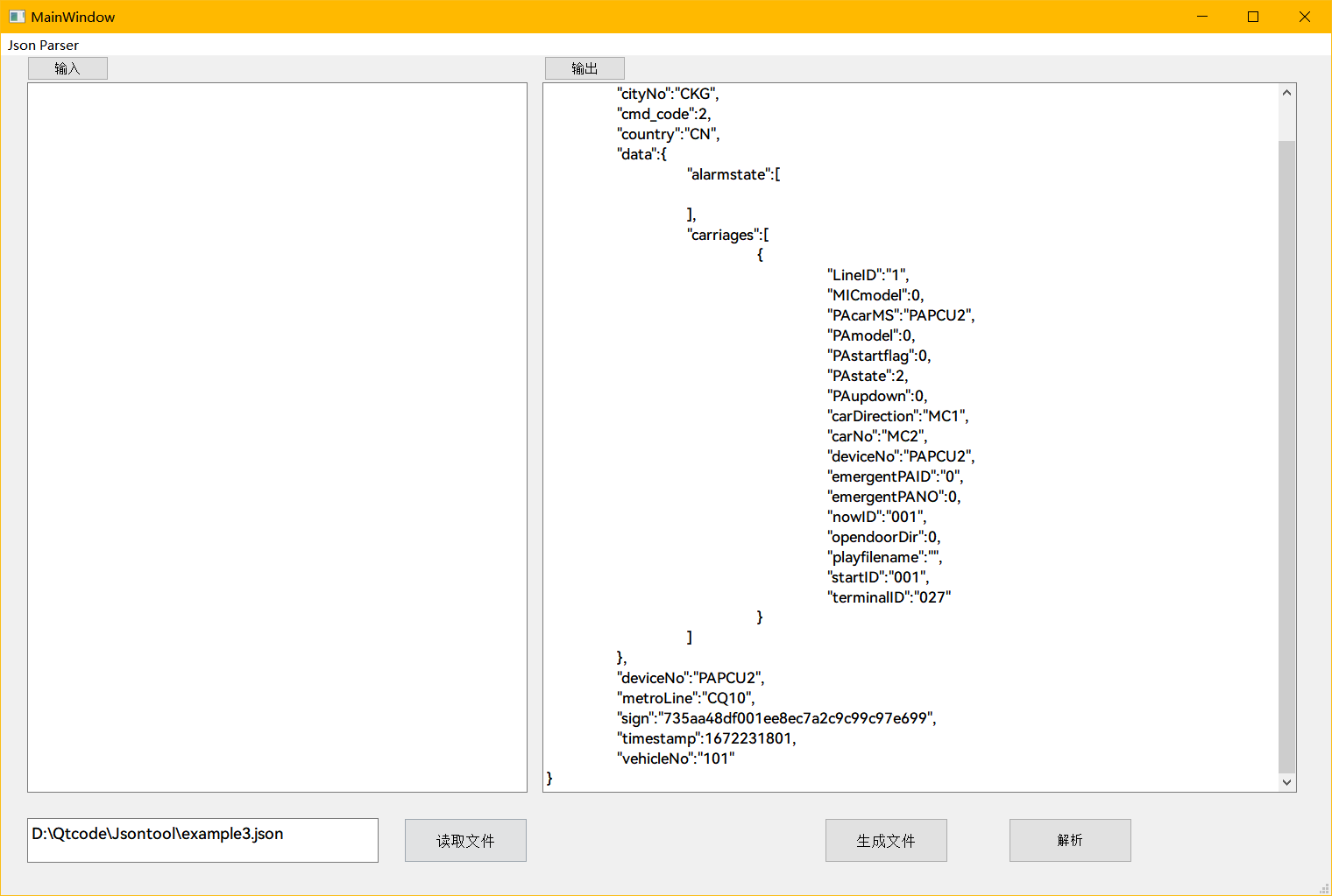
**测试样例：**

|  |
| --- |
| {  "carNo": "MC2",  "cityNo": "CKG",  "cmd\_code": 2,  "country": "CN",  "data": {  "alarmstate": [],  "carriages": [  {  "LineID": "1",  "MICmodel": 0,  "PAcarMS": "PAPCU2",  "PAmodel": 0,  "PAstartflag": 0,  "PAstate": 2,  "PAupdown": 0,  "carDirection": "MC1",  "carNo": "MC2",  "deviceNo": "PAPCU2",  "emergentPAID": "0",  "emergentPANO": 0,  "nowID": "001",  "opendoorDir": 0,  "playfilename": "",  "startID": "001",  "terminalID": "027"  }  ]  },  "deviceNo": "PAPCU2",  "metroLine": "CQ10",  "sign": "735aa48df001ee8ec7a2c9c99c97e699",  "timestamp": 1672231801,  "vehicleNo": "101"  } |

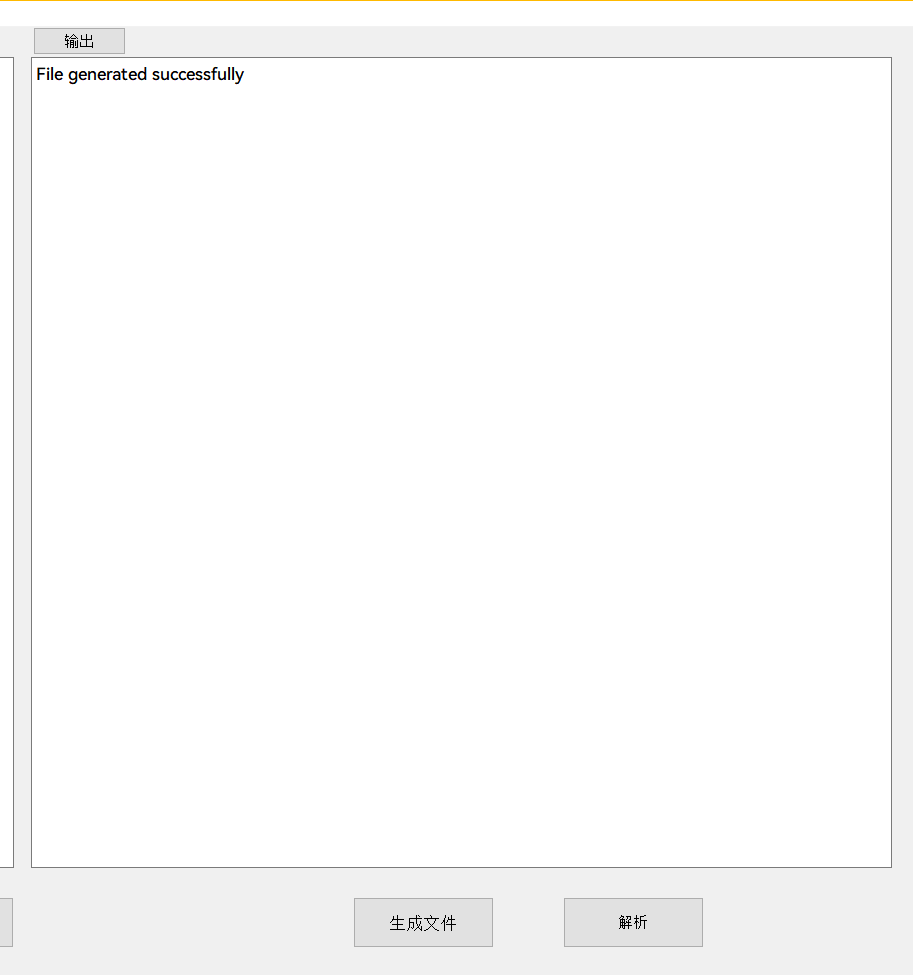
**1.输入框读取、解析、格式化功能测试**

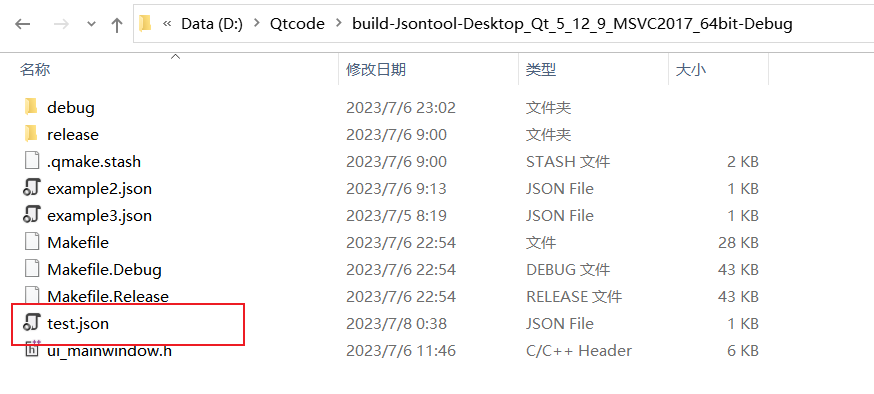


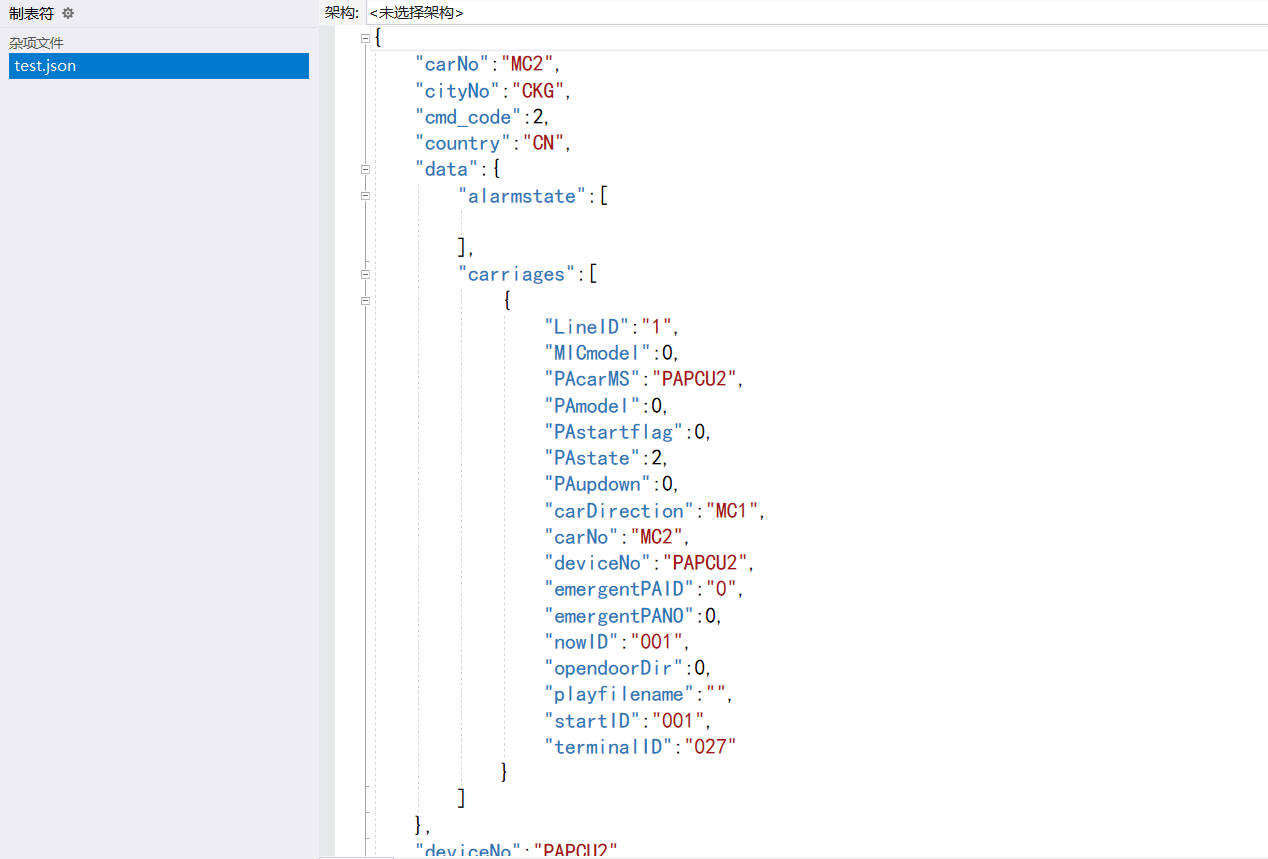
**2.文件读取、解析、格式化功能测试**



**3.文件生成功能**





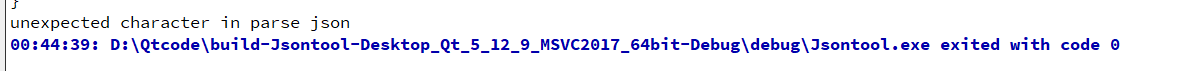


### 健壮性测试：

**测试样例1**

|  |
| --- |
| [  {  "[": ]  }  ] |

**测试结果：**



**测试样例2**

|  |
| --- |
| [  {  "[" ]  }  ] |

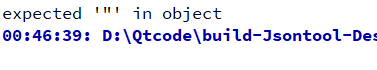
**测试结果**

****

**测试样例3**

|  |
| --- |
| {  []  } |

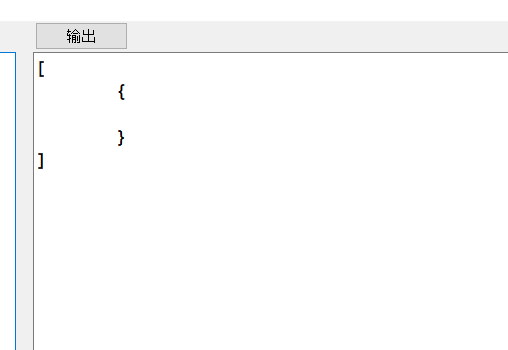
**测试结果**



**测试样例4**

|  |
| --- |
| [  {}  ] |

**测试结果**



## 五、总结和体会

1. 通过此次综合训练，我不仅了解到了Json的语法规范，还深入理解了Json的数据结构和组织方式。这使我对Json格式有了更深入的认识，提升了我的编程技能。

2. Json工具需要实现读取和处理Json文件的功能，这提高了我的文件处理能力。我学会了如何有效地读取文件内容，并将其转化为可操作的数据结构。

3. Json工具需要将Json数据解析为应用程序可以理解的格式，这要求我熟悉数据解析和处理的技巧，如使用递归算法遍历复杂的Json结构。

4. 在进行校验功能设计时，需要确保其有效性。这使我学会了如何编写校验逻辑，并处理各种可能的错误情况，提高了我的错误处理能力。

5. 开发Json工具器需要编写结构清晰、模块化的代码，以便于维护和扩展。使我更加注重代码的可读性和可维护性，学会了使用设计模式和良好的编码习惯。

6. 在开发Json工具时，我遇到了各种各样的问题，例如解析错误或格式不正确的Json数据。这促使我加强了调试和测试技能，学会了使用调试工具和编写有效的单元测试，为今后的学习和开发打下了良好的基础。