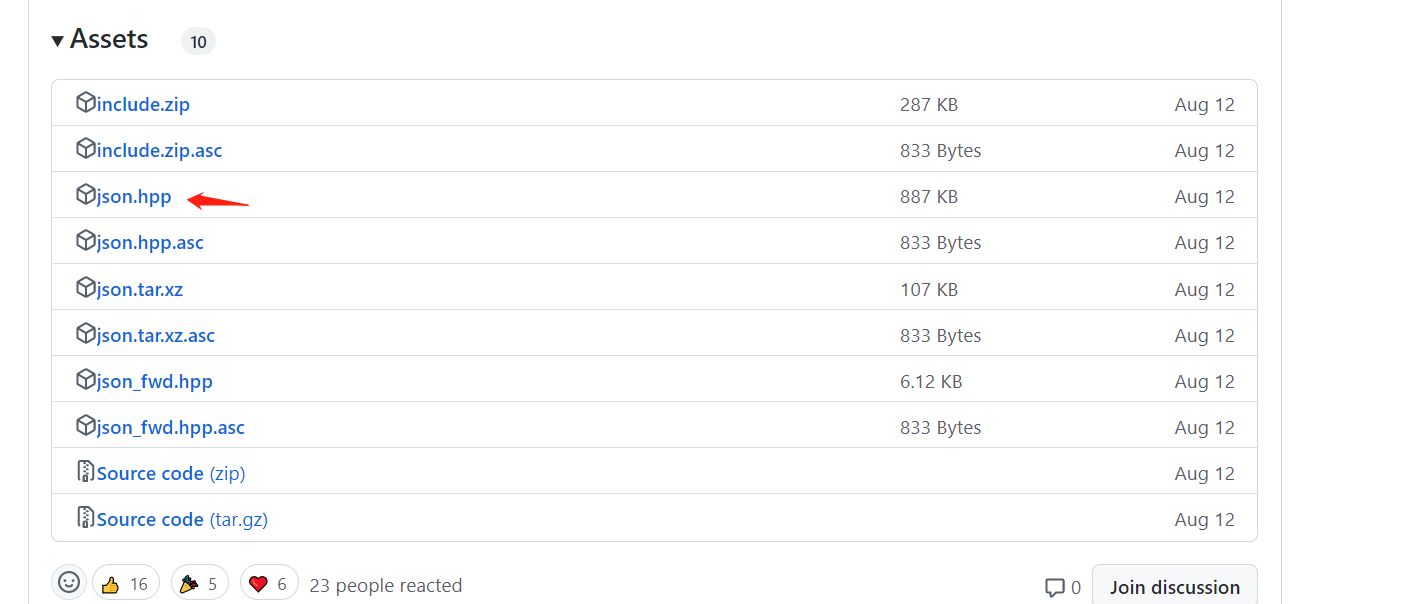
C++ json库用得比较多的是nlohmann::json库，支持现代C++版本，使用方便，只需要引入一个头文件。

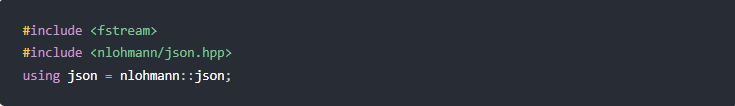
# 下载

库地址：<https://github.com/nlohmann/json/releases>

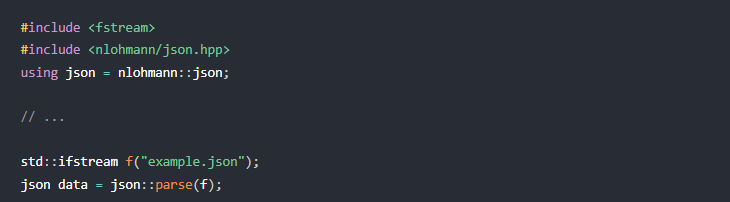


下载json.hpp放在工程里

# 引入代码

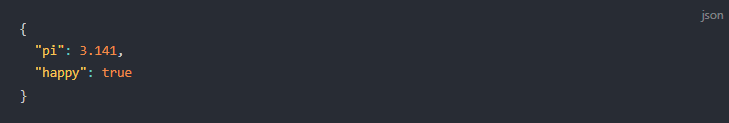


# 读取JSON文件

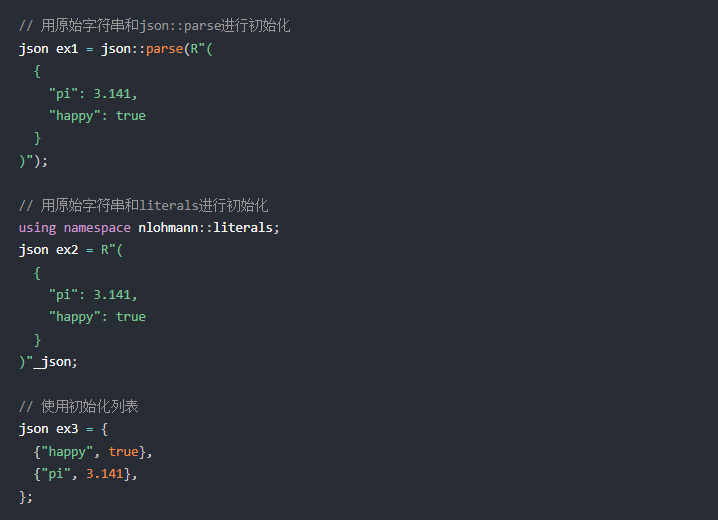


# 从JSON文本创建JSON对象

假设您要在文件中创建此文本 JSON 值作为对象：



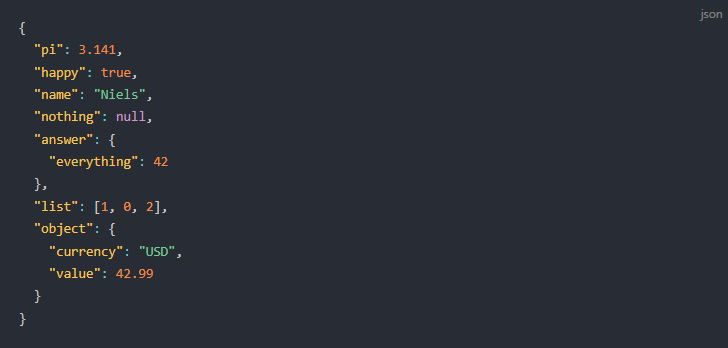
有多种选择：



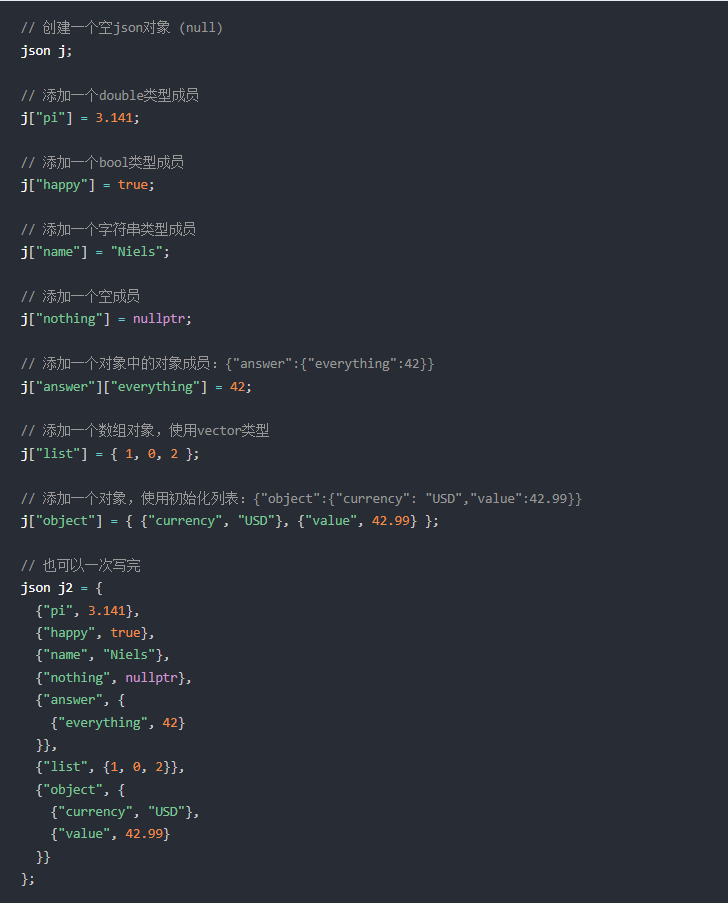
# JSON 作为第一类数据类型

下面是一些示例，可让您了解如何使用该类。

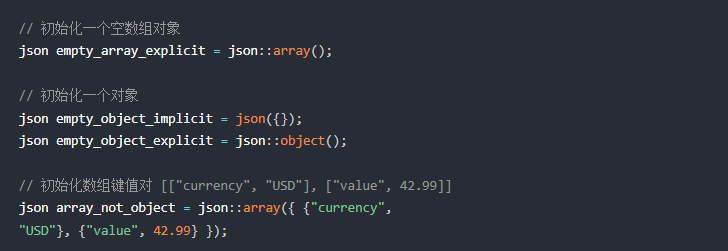
假设您要创建 JSON 对象



使用此库，您可以编写：

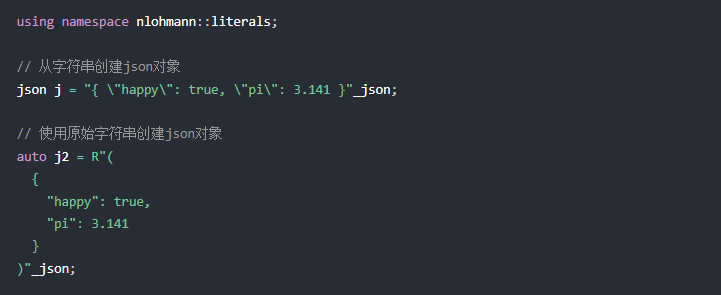


请注意，在所有这些情况下，您永远不需要“告诉”编译器要使用哪种 JSON 值类型。如果你想明确或表达一些边缘情况，函数json::array（）和json::object（）会有所帮助：



# 序列化/反序列化

## JSON对象和string互转

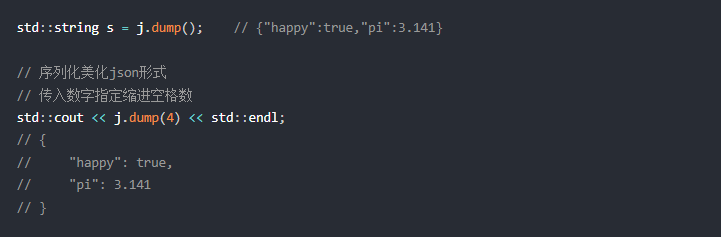


请注意，如果不附加后缀，则传递的字符串文字不会被解析，而只是用作 JSON 字符串 价值。也就是说，只会存储字符串而不是解析实际对象。\_jsonjson j = "{ \"happy\": true, \"pi\": 3.141 }""{ "happy": true, "pi": 3.141 }"

上面的例子也可以用json::parse()显式表示：

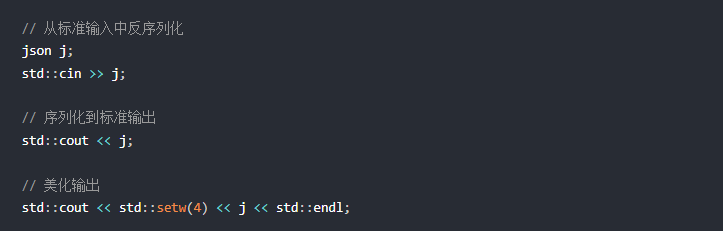


获取 JSON 字符串（序列化）：

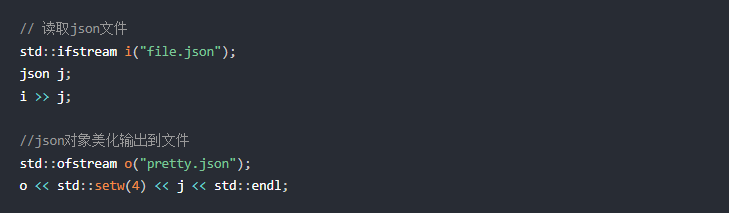


## 流输入输出（例如文件读写、字符串流）

您还可以使用流来序列化和反序列化：

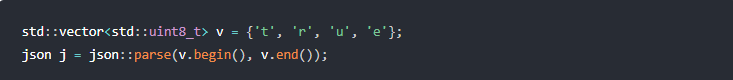


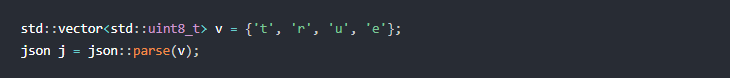
文件读写json：std::istream，std::ostream



从迭代器范围读取json

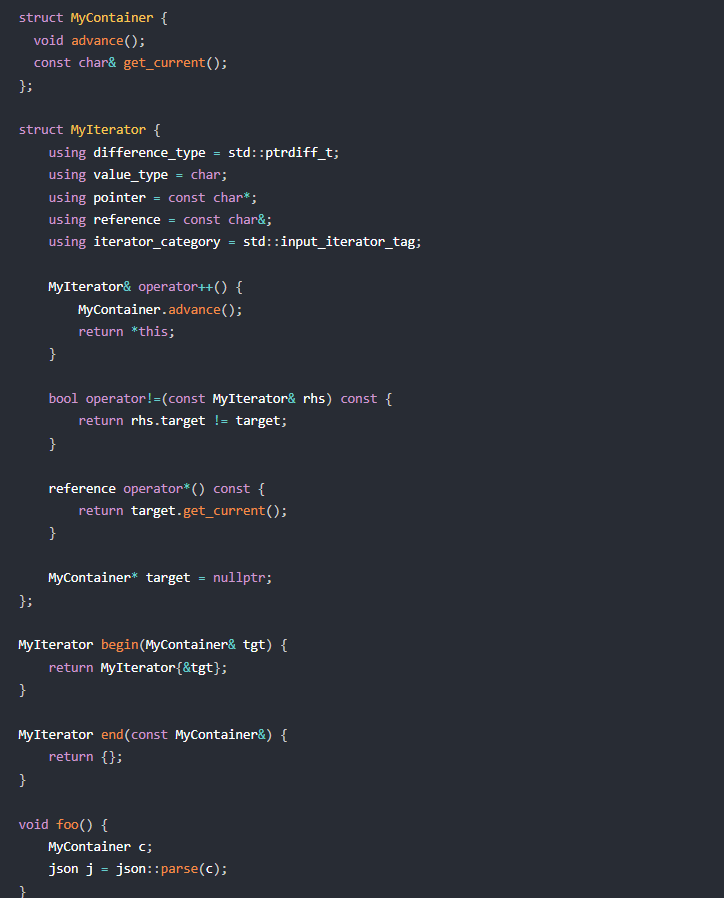
可以从迭代器范围解析 JSON;也就是说，从迭代器可访问的任何容器中，迭代器是 1、2 或 4 个字节的整数类型，将分别解释为 UTF-8、UTF-16 和 UTF-32。例如，std::vector<std::uint8\_t>,std::list<std::uint16\_t>





## 自定义数据源

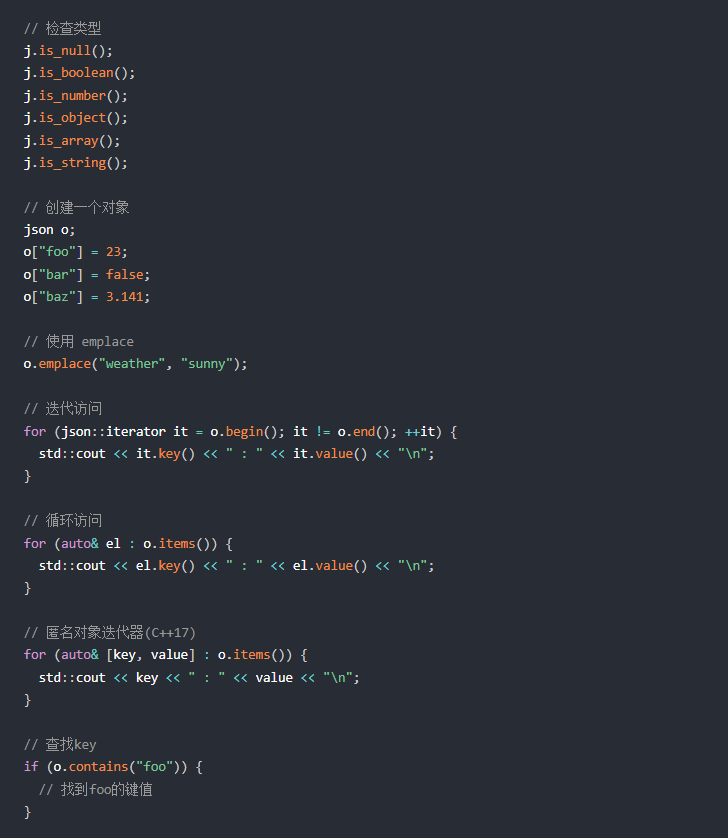
由于 parse 函数接受任意迭代器范围，因此您可以通过实现概念来提供自己的数据源。

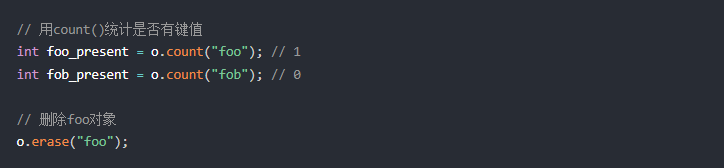


## 类似 STL 的访问

我们将 JSON 类设计为类似于 STL 容器的行为。事实上，它满足可逆容器的要求。







## 从 STL 容器转换

从STL容器转换到json，std::list转json，std::vector转json



std::map转json、std::unordered转json

