1 Value类的设计

使用的枚举类定义Json值的类型

```
enum class Type { Null, Boolean, Number, String, Array, Object };
```

定义Array和Object的实际类型

```
using Array = std::vector<Value>;
using Object = std::map<std::string, Value>;
```

```
Value();
Value(std::nullptr_t);
Value(bool b);
Value(int n);
Value(double n);
Value(const std::string& str);
Value(const char* c);
Value(const Array& arr);
Value(const Object& obj);
```

```
Value(Value&& other) noexcept;
Value& operator=(Value&& other) noexcept;
```

```
Value(const Value& other);
Value& operator=(const Value& other);
```

```
~Value(); // using tool: clear()
```

Value类的设计

Public	Private
构造函数 拷贝构造/赋值函数 移动构造/赋值函数 析构函数	类成员: Type type_ Union .....
	工具：清理函数
类判断函数	工具：拷贝函数
取值函数	比较运算符(==/!=)
数组访问	对象访问

存储值的类成员使用 union

```
switch (expression) {
    case constant1:
        // statement 1
        break;
    case constant2:
        // statement 2
        break;
    ...
    default:
        // optional others
        break;}
```

存储值的类成员使用 union

```
union {
        bool bool_value_;
        double number_value_;
        std::string* string_value_;
        Array* array_value_;
        Object* object_value_;};
```

- union（联合体） 是一种特殊的数据结构，所有成员共享同一块内存空间，最多只能存储其中一个成员的值，大小等于最大成员的大小。
- 使用指针：节省空间、动态分配、避免复杂对象的构造和析构

数组访问

- 重载 [] 运算符(const 和非 const) 注意：添加类型检查和边界检测!

```
Value& Value::operator[](size_t index) const {};
```

```
const Value& Value::operator[](size_t index) const{};
```

- 定义 size() 函数

对象访问

- 重载 [] 运算符(const 和非 const) 注意：添加类型检查和失败查找检测!

```
Value& Value::operator[](const std::string& key) {};
```

```
const Value& Value::operator[](const std::string& key) const{};
```

- 定义 has() 函数

移动构造/赋值函数格式

- 添加 noexcept 关键字
- 在重载 = 运算符时，需要添加 this != &other 的检查
- 在重载 = 运算符时，返回 *this 指向自己