# TABLE OF CONTENTS

# std::optional

在标头 <optional> 定义

```
template< class T >
class optional;
```
(C++17 起)

类模板 std::optional 管理一个*可选*的所含值，即既可以存在也可以不存在的值。

一种常见的 optional 使用情况是作为可能失败的函数的返回值。与如 std::pair<T, bool> 等其他手段相比，optional 可以很好地处理构造开销高昂的对象，并更加可读，因为它明确表达了意图。

optional<T> 的任何实例在任意给定时间点要么*含值*，要么*不含值*。

如果一个 optional<T> *含值*，那么保证该值内嵌于 optional 对象，这表示不会发生动态内存分配。因此，optional 对象模拟的是对象而非指针，尽管定义了 operator*() 和 operator->() 运算符。

当一个 optional<T> 类型的对象被按语境转换到 bool 时，若对象*含值*则转换返回 true，若它*不含值*则返回 false。

optional 对象在下列条件下*含值*：

- 对象被以 T 类型的值或另一*含值*的 optional 初始化/赋值。

对象在下列条件下*不含值*：

- 对象被默认初始化。
- 对象被以 std::nullopt_t 类型的值或*不含值*的 optional 对象初始化/赋值。
- 调用了成员函数 reset()。

optional 对象是一种 view，当其*含值*时包含一个元素，当其*不含值*时包含零个元素。所言元素的生存期与对象绑定。 (C++26 起)

不存在可选的引用、函数、数组或 *cv* void：如果以这些类型实例化 optional，那么程序非良构。另外，如果以（可有 cv 限定的）标签类型 std::nullopt_t 或 std::in_place_t 实例化 optional，那么程序非良构。

## 模板形参

**T** - 要为之管理初始化状态的值的类型。该类型必须满足*可析构 (Destructible)* 的要求。（特别是不允许数组和引用类型）

## 成员类型

| 成员名称 | 定义 |
|---|---|
| value_type | T |
| iterator (C++26 起) | 由实现定义的*老式随机访问迭代器 (LegacyRandomAccessIterator)*、*常量表达式迭代器 (ConstexprIterator)* 和 contiguous_iterator，其 value_type 和 reference 分别为 std::remove_cv_t<T> 和 T&。 |
| const_iterator (C++26 起) | 由实现定义的*老式随机访问迭代器 (LegacyRandomAccessIterator)*、*常量表达式迭代器 (ConstexprIterator)* 和 contiguous_iterator，其 value_type 和 reference 分别为 std::remove_cv_t<T> 和 const T&。 |

针对*容器 (Container)* 的迭代器的要求同样适用于 optional 的 iterator 类型。

## 成员函数

| | |
|---|---|
| (构造函数) | 构造 optional 对象<br>(公开成员函数) |
| (析构函数) | 销毁容纳的值（如果存在）<br>(公开成员函数) |
| **operator=** | 对内容赋值<br>(公开成员函数) |
| **迭代器** | |
| **begin** (C++26) | 返回指向起始的迭代器<br>(公开成员函数) |
| **end** (C++26) | 返回指向末尾的迭代器<br>(公开成员函数) |
| **观察器** | |
| **operator->**<br>**operator*** | 访问所含值<br>(公开成员函数) |

| | |
|---|---|
| operator bool<br>has_value | 检查对象是否含值<br>(公开成员函数) |
| value | 返回所含值<br>(公开成员函数) |
| value_or | 在所含值可用时返回它，否则返回另一个值<br>(公开成员函数) |

**单子式操作**

| | |
|---|---|
| and_then (C++23) | 在所含值存在时返回对其应用给定的函数的结果，否则返回空的 optional<br>(公开成员函数) |
| transform (C++23) | 在所含值存在时返回含有变换后的所含值的 optional，否则返回空的 optional<br>(公开成员函数) |
| or_else (C++23) | 在 optional 含值时返回自身，否则返回给定函数的结果<br>(公开成员函数) |

**修改器**

| | |
|---|---|
| swap | 交换内容<br>(公开成员函数) |
| reset | 销毁任何所含值<br>(公开成员函数) |
| emplace | 原位构造所含值<br>(公开成员函数) |

## 非成员函数

| | |
|---|---|
| operator== (C++17)<br>operator!= (C++17)<br>operator< (C++17)<br>operator<= (C++17)<br>operator> (C++17)<br>operator>= (C++17)<br>operator<=> (C++20) | 比较 optional 对象<br>(函数模板) |
| make_optional (C++17) | 创建一个 optional 对象<br>(函数模板) |
| std::swap(std::optional) (C++17) | 特化 std::swap 算法<br>(函数模板) |

## 辅助类

| | |
|---|---|
| std::hash<std::optional> (C++17) | **std::optional** 的散列支持<br>(类模板特化) |
| nullopt_t (C++17) | 不含值的 std::optional 的指示器<br>(类) |
| bad_optional_access (C++17) | 指示进行了到不含值的 optional 的有检查访问的异常<br>(类) |

## 辅助对象

## 辅助特化

| | | |
|---|---|---|
| nullopt (C++17) | | nullopt_t 类型的对象<br>(常量) |
| in_place<br>in_place_type<br>in_place_index<br>in_place_t<br>in_place_type_t<br>in_place_index_t | (C++17) | 就地构造的标签<br>(标签) |

```
template< class T >
constexpr bool ranges::enable_view<std::optional<T>> = true;
```
(C++26 起)

这个 ranges::enable_view 特化使得 optional 满足 view。

```
template< class T >
constexpr auto format_kind<std::optional<T>> = range_format::disabled;
```
(C++26 起)

这个 format_kind 特化禁用 optional 的范围格式化支持。

## 推导指引

### 注解

| 功能特性测试宏 | 值 | 标准 | 功能特性 |
| --- | --- | --- | --- |
| __cpp_lib_optional | 201606L | (C++17) | std::optional |
| | 202106L | (C++20)<br>(DR20) | 完全 constexpr |
| | 202110L | (C++23) | 单子式操作 |
| __cpp_lib_optional_range_support | 202406L | (C++26) | std::optional 的格式化支持 |

### 示例

运行此代码

```
#include <iostream>
#include <optional>
#include <string>

// optional 可用作可能失败的工厂的返回类型
std::optional<std::string> create(bool b)
{
    if (b)
        return "Godzilla";
    return {};
}

// 能用 std::nullopt 创建任何（空的）std::optional
auto create2(bool b)
{
    return b ? std::optional<std::string>{"Godzilla"} : std::nullopt;
}

int main()
{
    std::cout << "create(false) 返回 "
              << create(false).value_or("empty") << '\n';

    // 返回 optional 的工厂函数可用作 while 和 if 的条件
    if (auto str = create2(true))
        std::cout << "create2(true) 返回 " << *str << '\n';
}
```

输出：

```
create(false) 返回 empty
create2(true) 返回 Godzilla
```

### 参阅

| | |
| --- | --- |
| **variant** (C++17) | 类型安全的可辨识联合体<br>(类模板) |
| **any** (C++17) | 可保有任何*可复制构造 (CopyConstructible)* 类型的实例的对象。<br>(类) |
| **expected** (C++23) | 含有一个预期值或错误值的包装器<br>(类模板) |
| **ranges::single_view**<br>**views::single** (C++20) | 含有具有指定值的单个元素的 view<br>(类模板) (定制点对象) |
| **ranges::empty_view**<br>**views::empty** (C++20) | 无元素的空 view<br>(类模板) (变量模板) |