

基本语言特征

第1章结构化绑定

相当于对结构体进行解耦，分别用不同的变量去表示结构体的成员变量。

图形用户界面, 应用程序

中度可信度描述已自动生成

1.1结构化绑定详细信息

在使用结构化绑定时，存在一个隐藏的匿名变量（原始的副本）。变量为这个副本的引用。干脆就是原本的副本。互不干扰

当对函数返回值使用结构化绑定时，同样会创建一个匿名变量，用函数的返回值来初始化它，结构化绑定绑定到这个匿名变量的成员，而不是直接绑定到函数的返回值。

生命周期是独立的作用域。

限定符



图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

限定符是应用于新的匿名实体的

以 const auto& [u, v] = ms; 为例，匿名实体 e 是对 ms 的 const 引用，但 u 和 v 不是引用类型，它们只是具有 ms 成员的类型，并且由于匿名实体是 const 的，所以 u 和 v 也具有 const 属性。

表格

低可信度描述已自动生成

文本

描述已自动生成

**3. “有效但值未指定” 状态的含义**

“有效但值未指定”（valid but unspecified state）是 C++ 中移动语义里一个重要的概念，具体解释如下：

**3.1 有效**

“有效” 意味着对象仍然处于一个合法的、可操作的状态。也就是说，虽然对象的资源被移动走了，但它仍然可以安全地被销毁、重新赋值或者进行其他不会依赖其原有值的操作。例如，对于一个 std::string 对象，即使它的资源被移动走了，我们仍然可以对它进行析构操作，或者给它赋一个新的字符串值。

**3.2 值未指定**

“值未指定” 表示对象当前的值是不确定的，我们不能对其值做任何假设。这是因为在移动操作中，对象的资源被转移到了其他对象中，其内部状态可能已经被改变。例如，对于 std::string 类型，在移动操作后，它可能是空字符串，也可能包含一些随机的字符，具体取决于实现。因此，我们不应该依赖这个对象的值来进行后续的逻辑判断。

1.2可以使用结构化绑定的地方

* **类元组对象**：对于任何类型，只要提供了特定的类元组 API，就可以使用结构化绑定。

所有非静态数据成员必须属于同一个类定义（也就是说，它们必须是该类型的直接成员，或者是同一个无歧义的公共基类的成员）：

1.3为结构化绑定提供类似元组的API

非类型模板参数（Non-type template parameter）要求在实例化模板时，传入的必须是一个编译时常量表达式，且该表达式的值的类型要与非类型模板参数所声明的类型相匹配

文本

描述已自动生成