图片包含 表格

描述已自动生成

图形用户界面, 应用程序

中度可信度描述已自动生成

**4.1 扩展聚合初始化的动机**

第5章强制复制省略或传递非实体化对象

**2. C++17 强制复制省略**

* **规则改变**：C++17 引入了强制复制省略规则。当使用临时对象（纯右值）来初始化新对象时，编译器必须省略复制或移动操作。实际上，我们传递的是未具现化对象的值，直接用这个值来具现化新对象，而不是先创建临时对象再进行复制或移动。

c++17用临时对象初始化的时候，不依赖复制和移动构造函数了，直接在构造对象的内存位置，初始化，但是值返回类型的省略临时对象依然需要复制或移动构造函数

5.2临时文件强制副本省略的好处

工厂函数可以返回没有复制移动构造的类了。

5.3明确的价值类别

作为初始化新对象时要求临时对象复制省略的建议更改的副作用，对值类别进行了一些调整。

5.3.1价值分类

图示

描述已自动生成

C++11开始，

将亡值的示例包括：

* 通过右值引用（type&&，特别是由 std::move() 返回）返回的函数的结果。
* 转换为对象类型的右值引用的强制类型转换。

文本, 信件

描述已自动生成

图示

描述已自动生成

将亡值（xvalue）则被视为一种特殊的位置，代表一个其资源可以被重用的对象（通常是因为该对象接近其生命周期的结束）。

C++17 随后引入了一个新术语，即（临时对象的）具现化（materialization），用于描述纯右值变成临时对象的时刻。因此，临时对象具现化转换是一种纯右值到将亡值的转换。

文本, 信件

描述已自动生成C++17 引入了临时对象具现化的概念，指的是纯右值转变为临时对象的过程。这是一种纯右值到将亡值的转换。当一个纯右值出现在期望泛左值（左值或将亡值）的位置时，就会触发临时对象具现化。

* **复制省略更合理**：在 C++17 之前，复制省略是可选的优化，并且在某些情况下需要对象可复制或可移动。而 C++17 中，纯右值不再被视为对象，而是用于初始化的表达式。这使得强制复制省略更加合理，因为使用纯右值初始化变量时，不需要进行复制或移动操作，只需要将纯右值具现化到目标对象的位置即可。

5.4非实体化返回值传递

Chapter 6

Lambda Extensions

从 C++17 开始，如果条件允许，Lambda 表达式会隐式地成为常量表达式（constexpr）。也就是说，只要 Lambda 表达式所使用的特性在编译时上下文是有效的（例如，仅使用字面量类型、没有静态变量、没有虚函数、没有 try/catch、没有 new/delete），那么任何 Lambda 表达式都可以在编译时上下文中使用。

为了同时保证 Lambda 表达式可以在编译时调用和闭包对象在编译时初始化，建议使用 constexpr auto squared = [](auto val) constexpr { ... }; 的声明方式。

第7章新属性和属性特征

[[nodiscard]] 通常用于在返回值未被使用时表明存在不当行为。这种不当行为可能包括：

* **内存泄漏**：例如未使用返回的已分配内存。
* **意外或非直观的行为**：例如，不使用返回值时会出现不同的、意外的行为。
* **不必要的开销**：例如，若不使用返回值，调用的操作就相当于空操作

以下是一些使用该属性很有用的示例：

* 那些分配了需要通过另一个函数调用释放的资源的函数，应该标记为 [[nodiscard]]。一个典型的例子是分配内存的函数，比如 malloc() 或分配器的成员函数 allocate()。

第8章其他语言特征



图形用户界面, 文本, 应用程序, Teams

描述已自动生成