**C++学习笔记——左值、右值、引用及与const限定符的关系**

**1. 左值**

既可以出现在赋值运算符左边，也可以出现在赋值运算符右边的表达式。

特点：

- 可以取地址

- 可读写

- 总是具有名称

- 生命周期通常为当前代码块；全局变量/常量的生命周期等于程序生命周期

**2. 右值**

只能出现在赋值运算符右边的表达式。

特点：

- 不能取地址

- 只读

- 没有名称

- 默认生命周期为当前表达式，表达式执行完毕即被销毁。使用右值引用可延长右值的生命周期

**3. const左值引用**

示例：

文本

描述已自动生成

- 普通（非const）左值引用不能用右值初始化，这会导致权限放大——右值可读不可写，而左值可读可写。

- const左值引用是可读不可写的，与右值权限完全相同，所以const左值引用可以用右值来初始化。

- const左值引用也可以用左值来初始化，这会导致权限缩小——左值可读可写，而const左值可读不可写。

- 综上所述，const左值引用，可以用左值或右值初始化；而普通（非const）左值引用，只能用左值初始化。

**4. 右值引用**

右值引用是一种特殊的左值，它只能用右值初始化。

引入右值引用的目的：延长右值生命周期；减少对象构造过程中的复制开销。

右值引用是现代C++很多特性的基础。

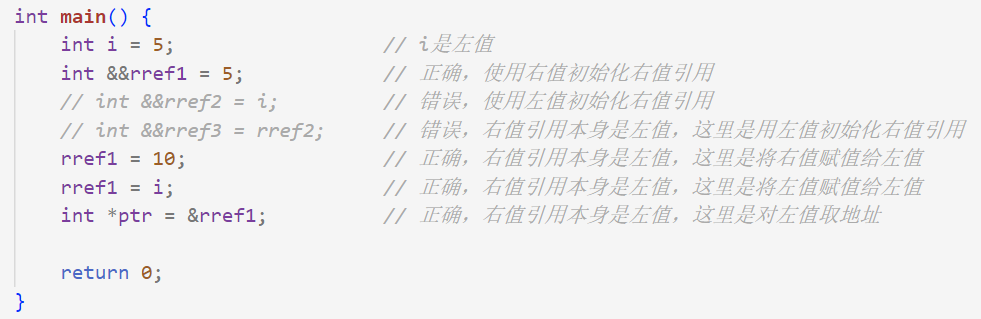
特点：

- 只能用右值初始化

- 可读写

- 右值引用本身是左值——实际上，右值引用在初始化时，会将用来初始化它的右值，复制到栈空间某一内存区域。现在，右值引用就是左值：它所指向的栈空间内存区域具有名称（即右值引用变量名），可读写，并且生命周期持续到当前代码块（而非当前表达式）结束

示例：



**5. const右值引用**

缩小右值引用的权限，使其由可读写变为只读。

示例：

文本, 信件

描述已自动生成