C++中，判断左值和右值典型的判别方法：出现在赋值表达式左边的就是“左值”，右边的是“右值”。例如：

a = b + c;

a是左值，b + c就是右值。

更严谨的方法：有名字的变量，可以取地址的就是左值，反之就是右值。

在C++11中，右值由两个概念构成，一个是将亡值（expiring value），另一个是纯右值(pure value)。

纯右值就是C++98标准右值，主要是临时变量和一些不跟对象关联的值，如函数返回的临时变量值，一些运算表达式产生的临时变量值都是纯右值。不跟对象关联的字面常量，如2, ‘c’, true也是右值。此外，类型转换函数的返回值，lambda表达式也是右值。

C++11中，将亡值是跟右值引用相关的表达式，如返回右值引用T&&的函数返回值，std::move()的返回值，转换为T&&类转换函数的返回值。

C++98中的引用称为“左值引用”，无论是左值引用，还是右值引用都必须初始化。因为引用本身并不拥有所绑定对象的内存。左值引用是具名变量的别名，右值引用时匿名变量的别名。

右值引用不能绑定到任何的左值。

int c = 0;

int&& c1 = 0;

int&& d = c; // error

int&& e = c1; // error，c1是左值

左值引用一般也不能绑定到右值，只有常量左值引用可以绑定到右值

bool& judge = true; // error

const bool& judge = true; // ok，常量左值引用

const bool judge = true; // 表达式结束后，右值就会销毁

const bool& judge = true; // 直接使用右值并为其“续命”

如果判断不出来表达式是右值引用还是左值引用，可以考虑使用stl模板库中的模板类：

#include <type\_traits>

is\_rvalue\_reference

is\_lvalue\_reference

is\_reference