# 基本定义

左值：非常量限定的，可以出现在赋值表达式的左侧

消亡值：即将到期的右值

右值(纯右值)：仅仅出现在赋值表达式右侧的非消亡值的表达式。

通常意义上的右值包括prvalue(纯右值)和xvalue(消亡值)。

左值与消亡值则可以称作泛左值(glvalue)

**注意：**

左值和消亡值可以是不完全类型，但是右值(纯右值)必须是完全类型(不完全类型是尚未指定其成员的结构或联合类型)。

类的右值可以被常量-易变量修饰(cv-qualified),但是非类右值不能常量或者易变量修饰。

不完全类型指“函数之外、类型的大小不能被确定的类型”，只能以有限方式使用。不能定义该类型的对象。不完全类型只能用于定义指向该类型的指针及引用，或者用于声明使用该类型作为形参类型或者返回值类型，比如：

char str[] //不完全类型

char str[10] //完全类型

左值不一定运行修改其对象，比如下列情况：

1. 数组类型的左值
2. 不完全类型
3. 常量限定的左值
4. 一个结构体或者联合体，其中至少有一个成员被常量限定

以下几种情况是消亡值(临时值)：

1. 函数的返回值，其结果是一个右值引用
2. 类型转换(cast)为右值引用(比如std::move?)
3. 通过消亡值表达式访问的非引用类型的非静态数据成员
4. 指向成员访问表达式的指针，其中第一个操作数是消亡值表达式，第二个操作数是指向成员类型的指针。

lvalue的范围：

1. 变量、函数都是左值
2. 内建类型和组合类型都是左值
3. 前置自增是左值，可以对其取地址(&++a)
4. 内建指针是左值
5. 内建下标运算符，除非数组类型是右值数组
6. 指向成员的函数指针，只要指向的成员为数据，则表达式为左值
7. 内建逗号表达式，此时表达式的最右值是左值
8. C++规定字符串是左值

简单来说，能取地址的变量一定是左值，至于为什么不能把=号左边的都看成左值，是因为C++中等号是可以重载的，等号完全可以重载成为等号左边为右值的形式。

prvalue(纯右值)：

1. 字面值常量，包括空指针，bool值
2. 返回值是非引用的表达式都是右值
3. 后自增的表达式时纯右值
4. 取地址表达式也是一个右值，毕竟地址本身就是纯右值
5. 枚举或者成员函数
6. this指针也是纯右值，因为this也是一个地址
7. lambda表达式时纯右值

注意，纯右值不能是动态类型，不能被声明为常量或者易变量、不能是不完全类型。

简单来说，纯右值是传统右值的一部分，其是表达式产生的中间值，不能取地址。

xvalue(消亡值)：

1. 返回右值引用的表达式是消亡值(注意，返回的是右值引用)，而不是纯右值

本质上，消亡连接右值表达式，可以是动态类型，但是非class类型不能被常量或者易变量限定。

glvalue(泛左值)：

glvalue可以隐式转换为prvalue，左值转换为rvalue，数组转换为指针，函数转换为隐式指针转换。

glvalue可以是多态的。

rvalue(右值)：

1. 和左值不同，右值不能取地址
2. 右值不能在内建等号的左边，但是左值可以在内建等号的右边(简单来说，右值不能当左值用，但是左值可以当右值用)
3. 在C++11中，如果存在一个函数有两个重载，一个重载右值引用，一个重载const 左值引用，此时传入右值的话，将会匹配到右值引用(虽然右值引用可以绑定const左值引用，但是匹配右值引用更加精确)