**C++ 类中特殊成员变量（常量、静态、引用）的初始化方法**

2016年03月01日 14:33:38

阅读数：2542

有些成员变量的数据类型比较特别，它们的初始化方式和普通数据类型的成员变量有所不同。这些特殊类型的数据类型包括：

* 引用（&）
* 常量（const）
* 静态（static）   
  + 静态常量（static const）   
    - 整数类型（integral types：int、char、bool、）
    - 非整数类型

常量（const）和引用（&）必须听过**参数列表**进行初始化；

静态成员变量的初始化有些特别，是在类外初始化且不能在带有 static 关键字；

带有类内初始值设定项的成员必须为常量（且为integral types）；

所以，纵向来看，初始化方式主要有两种：

* 初始化参数列表：const，引用
* 类外初始化：static（且不再带有 static 关键字）
* 类内初始值设定：const+integral type

横向来看，不同数据类型的初始化方式分别为：

* 引用：初始化参数列表
* const   
  + 类内初始化值设定（如果为integral types）
  + 初始化参数列表
* static：   
  + 类外初始化，且不再带有 static 关键字
  + 类内初始化设置，必须为const，且integral
  + 不可通过构造函数的初始化参数列表进行初始化   
    - static 不属于某一特定的类实例，而是属于类，显然不能通过构造函数的初始化参数列表进行 static 变量的初始化

进行类内初始值设定的成员变量的数据类型必须满足：

* const
* integral types（int、char、bool）

class Test

{

public:

Test():i(1), ri(i), ci(2), cd(0.)

{}

// 对于一般常量型（非integral type）和引用型成员变量，必须通过初始化参数列表进行初始化

// 普通成员变量也可以放在函数体中，但本质已不是初始化，而是一种普通的赋值运算，效率极低

private:

int i;

// 普通成员变量

// 初始化参数列表，函数体

int& ri;

// 引用，初始化参数列表

const int ci;

// 常量

// 类内初始值设定

// 初始化参数列表

const double cd;

// 常量非integral types

// 初始化参数列表

static int si;

// 静态非常量

// 类外

static const int sci;

// 静态常量（integral type）

static const double scd;

// 静态常量（非integral ）

};

一般常量类型（未被static修饰）、引用类型，在类内部即可实现初始化，在类外实现初始化的必定是static类型（当然一部分的静态成员，也即同时又是const和integral type的，可在类内初始值设定，这一语法特性并非为所有的编译器所支持，所以一种推荐的做法，即是凡是static类型的，我们总在类外进行初始化，哪怕它在类内已进行初始值设定，只要在类外初始化时不修改其值）：

int Test::si = 0;

const int Test::sci = 1;

const double Test::scd = 99.0;

静态成员属于类作用域，但不属于类对象实例，和普通的static变量一样，程序一运行就分配内存并初始化，生命周期和程序一致；

静态成员其实和全局变量的地位是一致的，只不过编译器把它的使用限制在类作用域内。

**References**

[1] [C++ 类中特殊的成员变量（常变量、引用、静态）的初始化方法](http://boyishachang.blog.51cto.com/3485129/1285954)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/lanchunhui/article/details/50772363