# static 与非 static 成员（函数）

通常，非 static 数据成员存在于类类型的每个对象中。然而，static 数据成员独立于该类的任意对象而存在；
每个 static 数据成员是与类关联的对象，而不是与该类的对象相关联。

类也可以定义 static 成员函数。static 成员函数没有 this 形参，它可以直接访问所属类的 static 成员，但不能直接使用非 static 成员。
注意：类的非 static 成员函数是可以直接访问类的 static 和非 static 成员，而不用作用域操作符。

使用 static 成员的优点：
（1）避免命名冲突：static 成员的名字在类的作用域中，因此可以避免与其他类的成员或全局对象名字冲突。
（2）可以实施封装：static 成员可以是私有成员，而全局对象不可以。
（3）易读性：static 成员是与特定类关联的，可显示程序员的意图。

static 成员 与 非 static 成员调用方法：
非 static 成员通过对象调用。
static 成员通过作用域操作符（直接调用）、对象、引用、指向该类类型对象的指针（间接调用）

```
class Lunais{

static double zty();
double zzz;

};
Lunais z;
Lunais *t = &z;
double zty;

zty = Lunais::zty();   //static 成员通过作用域操作符（直接调用）
zty = z.zty();          //static 成员通过对象（简接调用）
zty = t->zty();        //static 成员通过指向该类类型对象的指针（简接调用）
```

static 数据成员定义：
1、一般情况下，static 数据成员是类内声明，类外定义；
2、static 成员不通过类构造函数初始化，而是在定义时进行初始化；
3、一个例外：初始化式为常量表达式，整型 static const 数据成员（static const int） 可以在类的定义体内进行初始化：

```
class Lunais{
```

**static const int zty = 30;**

**}**

**值得注意的是：const static 数据成员在类的定义体中出始化时，该数据成员仍必须在类的定义体外定义，只是不再指定初始值：**

**const int Lunais::zty;**

**常实型 static const 数据成员不可在类内初始化。一个好的解决方法是使用宏定义：**

**#define zty 5421.5421**

常整型静态数据成员可以在类中直接初始化，而常实型静态数据成员不可以

```
 class circle
{
int a;                                                  // 普通变量，不能在类中初始化
static int b;                                           // 静态变量，不能在类中初始化
static const int c=2;                                   // 静态常整型变量，可以在类中初始化
static const double PI=3.1416;//error C2864:     //只有静态常量整型数据成员才可以在类中初始化
} ;
```

**const int cicle::c ;  //const static 数据成员在类的定义体中出始化时，该数据成员仍必须在类的定义体外定义，只是不再指定初始值**

b 可以在类外进行初始化，且所有对象共享一个 b 的值：

int circle::b = 2;

**double circle::PI = 3.1416;**