C/C++程序设计案例实战
——包裹数量的秘密

华中农业大学信息学院　翟瑞芳

包裹数量的秘密：静态成员

包裹数量的秘密：静态成员

```
class Package
{
    public:
      int m_nCount;
    private:
      int m_nID;
};

int main( )
{
    Package A(1), B(2);
    Package *p = new Package(3);
    …
    delete p;
    …
    return 0;
}
```

弊端：

1.数据的冗余度大。

2.数据的一致性保证费时、费力、甚至难以实现。

包裹数量的秘密：静态成员

```
class Package
{
    public:
      int m_nCount;
    private:
      int m_nID;
};
int main( )
{
    Package A(1), B(2);
    Package *p = new Package(3);
    …
    delete p;
    …
    return 0;
}
```

m_nID:
每个对象特有的属性，占对象的数据空间；

M_nCount:
类的属性，属于所有对象共有，仅需要一份而且与对象分离的数据空间。

包裹数量的秘密：静态数据成员

```
class Package
{
    public:
      static int m_nCount;
    private:
      int m_nID;
};

int main( )
{
    Package A(1), B(2);
    Package *p = new Package(3);
    …
    delete p;
    …
    return 0;
}
```

静态数据成员

包裹数量的秘密：静态数据成员

```
#include<iostream>
using namespace std;
class Package
{public:
    Package (int nID)
    {
        m_nID = nID;
        m_nCount++;
    }
    Package(Package & p);
    int GetID() {return m_nID;}
    void GetCount()
    {
        cout<<"I got " <<
    m_nCount <<"package(s)!\n";
    }
private:
    int m_nID;
    static int m_nCount;
};
```

```
Package::Package(Package & p)
{
    m_nID = p.m_nID;
    m_nCount++;
}

int Package::m_nCount=0;

int main()
{
  Package A(1);
  cout<<A.GetID()<<endl;
  A.GetCount();
  Package B(2);
  cout<<B.GetID()<<endl;
  B.GetCount();
  return 0;
}
```

包裹数量的秘密：静态数据成员

静态数据成员的定义及初始化

1．在类体内声明；

2．在类体外部初始化，初始化形式为：

数据类型 类名::静态数据成员名 = 初始化值；

或

数据类型 类名::静态数据成员名(初始化值)；

```
int Package::m_nCount=0;
int Package::m_nCount(0);
```

包裹数量的秘密：静态数据成员

静态数据成员的访问

1．通过类的某个对象访问；

```
Package A(1);
A.m_nCount (m_nCount的访问控制属性是共有)
```

2.直接访问静态数据成员；

```
类名::静态数据成员名;
Package::m_nCount (m_nCount的访问控制属性是共有)
```

包裹数量的秘密：静态数据成员

静态数据成员的生命期是全局的

在main函数执行前创建，直到main函数返回后销毁。

包裹数量的秘密：静态成员函数

```
#include<iostream>
using namespace std;
class Package
{public:
    Package (int nID)
    {
        m_nID = nID;
        m_nCount++;
    }
    Package(Package & p);
    int GetID() {return m_nID;}
    static int GetCount()
    {
        return m_nCount;
    }
private:
    int m_nID;
    static int m_nCount;
};
```

包裹数量的秘密：静态成员函数

```
#include<iostream>
using namespace std;
class Package
{public:
    Package (int nID)
    {
        m_nID = nID;
        m_nCount++;
    }
    Package(Package & p);
    int GetID() {return m_nID;}
    static int GetCount()
    {
        return m_nCount;
    }
private:
    int m_nID;
    static int m_nCount;
};
```

```
Package::Package(Package & p)
{
    m_nID = p.m_nID;
    m_nCount++;
}

int Package::m_nCount=0;

int main()
{
  Package A(1);
  cout<<A.GetCount()<<endl;

  Package B(2);
  cout<< Package::GetCount();
<<endl;
  return 0;
}
```

包裹数量的秘密：静态成员函数

```
#include<iostream>
using namespace std;
class Package
{public:
    Package (int nID)
    {
        m_nID = nID;
        m_nCount++;
    }
    Package(Package & p);
    int GetID() {return m_nID;}
    static int GetCount()
    {
        return m_nCount;
    }
private:
    int m_nID;
    static int m_nCount;
};
```

静态成员函数的声明

static 返回类型 静态成员函数名（形式参数表）；

```
static int GetCount();
```

包裹数量的秘密：静态成员函数

静态成员函数的访问

- 通过类的某个对象访问：

对象名.静态成员函数名(实参)

A.GetCount()

- 直接访问静态数据成员；

类名::静态数据成员名(实参)

Package::GetCount ()

```
Package::Package(Package & p)
{
    m_nID = p.m_nID;
    m_nCount++;
}

int Package::m_nCount=0;

int main()
{
  Package A(1);
  cout<<A.GetCount()<<endl;

  Package B(2);
  cout<< Package::GetCount();
<<endl;
  return 0;
}
```

包裹数量的秘密：静态成员函数

```
#include<iostream>
using namespace std;
class Package
{public:
    Package (int nID)
    {
        m_nID = nID;
        m_nCount++;
    }
    Package(Package & p);
    int GetID() {return m_nID;}
    static int GetCount()
    {
        cout<<m_nID<<endl;
        return m_nCount;
    }
private:
    int m_nID;
    static int m_nCount;
};
```

注意

- 静态成员函数不能对非静态成员进行默认访问。

包裹数量的秘密：静态成员函数

```
#include<iostream>
using namespace std;
class Package
{public:
    Package (int nID)
    {
        m_nID = nID;
        m_nCount++;
    }
    Package(Package & p);
    int GetID() {return m_nID;}
    static int GetCount()
    {
        return m_nCount;
    }
private:
    int m_nID;
    static int m_nCount;
};
```

- 静态成员函数没有this指针，而非静态成员有一个指向当前对象的this指针。

小结

为什么需要静态成员

静态数据成员的声明和初始化

静态数据成员的访问

静态成员函数的声明和访问

延伸

针对快递包裹案例，增加静态成员，表示包裹的运送地址，并对包裹地址输出。请编写代码实现。