CSDN 学院

C++语言基础

迁者 - 贺利坚

http://blog.csdn.net/sxhelijian/

http://edu.csdn.net

本节主题：

静态成员——静态数据成员和静态成员函数

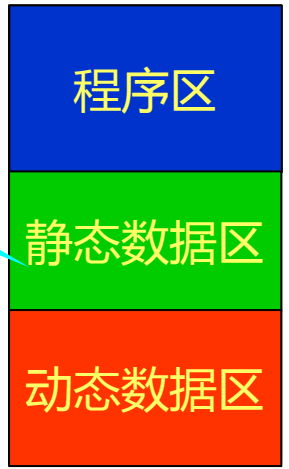
回顾：静态局部变量

- 用static声明静态局部变量

```
int f(int a)
{
   static int c=3;
   ……
}
```

- 性质
  - 局部特性：作用范围仅限于本函数
  - 静态特性：存储在静态区，函数调用结束后不消失而保留原值，在下一次调用时，保留上一次调用结束时的值。

静态
局部变量
存储在
静态存储区

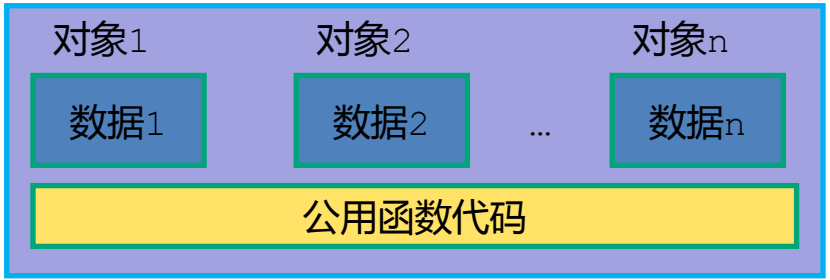


问题的提出

- 现状：n个同类的对象，每一个对象都分别有自己的数据成员，各自有值，互不相干。
- 期望：希望有某一个或几个数据成员为某类所有对象所共有，以实现数据共享。
- 方案：用全局变量

```
#include<iostream>
using namespace std;
int N = 0;
class Class
{
private:
  int a;
public:
  Class(){N++;a=0;}
  void add(){N++;}
};
```

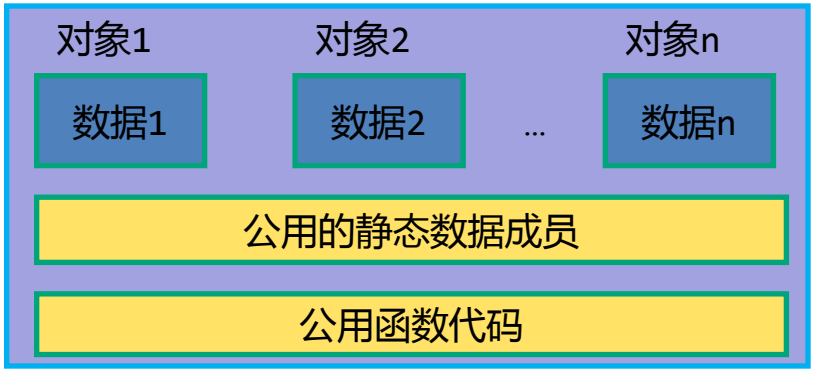
```
int main( )
{
   Class c1, c2;
   N=300;
   c1.add();
   cout<<N<<endl;
   return 0;
}
```



在程序各处都可以自由地修改全局变量的值——不安全

新方案：用静态数据成员

- 静态数据成员被所有对象共享，在所有对象之外单独开辟空间存储；
- 静态数据成员所占空间并不随某个对象的撤消而释放；
- 用静态的数据成员：可以在同类的多个对象之间实现数据共享。




```
class Box
{
public:
    int volume( );
 private:
    static int height;  //静态的数据成员
    int width;
    int length;
};
```

例 引用静态数据成员

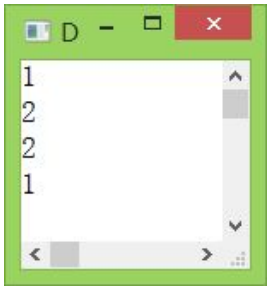
```
#include <iostream>
using namespace std;
class Student
{
public:
   Student(int n,string nam, int a):
         num(n),name(nam),age(a) { ++count; }
   ~Student() { --count; }
   int getCount()   { return count; }
private:
   static int count;
   int num;
   string name;
   int age;
};
int Student::count=0;
```

只能类外初始化

静态数据成员是“大家”的

- 静态数据成员不属于某对象，而是属于类的所有对象，不过，用类的对象可以引用它。
- 如果静态数据成员被定义为私有的，则不能在类外直接引用，而必须通过公用的成员函数引用。
- 静态数据成员实现了各对象之间的数据共享，同时避免了使用全局变量破坏了封装的原则。

```
int main( )
{
   Student stu1(1001,"He",40);
   cout<<stu1.getCount()<<endl;
   Student *pt=new Student(1001,"You",20);
   cout<<pt->getCount()<<endl;
   cout<<stu1.getCount()<<endl;
   delete pt;
   cout<<stu1.getCount()<<endl;
   return 0;
}
```



用类名访问数据成员

- 静态数据成员既可以通过对象名引用，也可以通过类名来引用。
- 在作用域内，通过类名和域运算符“::”引用静态数据成员时，不用考虑该类是否有对象存在。

```
#include <iostream>
using namespace std;
class Student
{
public:
   Student(int n,string nam, int a):
         num(n),name(nam),age(a) { ++count; }
   ~Student() { --count; }
   int getCount()   { return count; }
   static int count;
private:
   int num;
   string name;
   int age;
};
int Student::count=0;
```

count为public型

```
int main( )
{
   cout<<Student::count<<endl;
   Student *pt=new Student(1001,"You",20);
   cout<<pt->getCount()<<endl;
   delete pt;
   cout<<Student::count<<endl;
   return 0;
}
```

为什么出错？

```
#include <iostream>
using namespace std;
class Student
{
public:
   Student(int n,string nam, int a):
         num(n),name(nam),age(a) { ++count; }
   ~Student() { --count; }
   int getCount()   { return count; }
private:
   static int count;
   int num;
   string name;
   int age;
};
int Student::count=0;
```

count重为private

```
int main( )
{
   cout<<Student::getCount()<<endl;
   cout<<Student::count<<endl;
   Student *pt=new Student(1001,"You",20);
   cout<<pt->getCount()<<endl;
   delete pt;
   return 0;
}
```

error: cannot call member function 'int Student::getCount()' without object

error: 'int Student::count' is private

静态成员函数

- 成员函数也可以定义为静态的
- 方法：类声明函数的前面加static

  static int getCount( );
- 静态成员函数的作用就是为了能处理静态数据成员
- 本质：不需要this指针访问的成员！

```
#include <iostream>
using namespace std;
class Student
{
public:
   Student(int n,string nam, int a):
         num(n),name(nam),age(a) { ++count; }
   ~Student() { --count; }
   static int getCount()   { return count; }
private:
   static int count;
   int num;
   string name;
   int age;
};
int Student::count=0;
```

```
int main( )
{
   cout<<Student::getCount()<<endl;
   Student *pt=new Student(1001,"You",20);
   cout<<pt->getCount()<<endl;
   delete pt;
   return 0;
}
```

静态成员函数不能处理非静态数据成员

```
#include <iostream>
using namespace std;
class Student
{
public:
   Student(int n,string nam, int a):
         num(n),name(nam),age(a) { ++count; }
   ~Student() { --count; }
   static int getCount()
      { age++; return count; }
private:
   static int count;
   int num;
   string name;
   int age;
};
int Student::count=0;
```

```
int main( )
{
   cout<<Student::getCount()<<endl;
   Student *pt=new Student(1001,"You",20);
   cout<<pt->getCount()<<endl;
   delete pt;
   return 0;
}
```

error: invalid use of member 'Student::age' in static member function

error: from this location

静态成员函数没有this形参，故不能访问非静态成员

非静态成员函数可以处理静态成员，但不提倡

```
#include <iostream>
using namespace std;
class Student
{
public:
   Student(int n,string nam, int a):
         num(n),name(nam),age(a) { ++count; }
   ~Student() { --count; }
   void showCount()
       {++count;cout<<count<<endl;}
   static int getCount() {return count; }
private:
   static int count;
   int num;
   string name;
   int age;
};
int Student::count=0;
```

小结：

- 静态成员的本质特征：是类中所有对象的“公共元素”
- 静态成员的语法特征：通过类名和域运算符“::”引用，而不只是通过对象引用

```
int main( )
{
   cout<<Student::getCount()<<endl;
   Student *pt=new Student(1001,"You",20);
   cout<<pt->getCount()<<endl;
   pt->showCount();
   delete pt;
   return 0;
}
```

例 静态成员函数的使用

```
#include <iostream>
using namespace std;
class Student
{
public:
   Student(int n,int a,float s):num(n),age(a),score(s) { }
   void total( );
   static float average( );
private:
   int num;
   int age;
   float score;
   static float sum;
   static int count;
};
```

```
void Student::total( )
{
   sum+=score;
   count++;
}
float Student::average( )
{
   return(sum/count);
}
```

```
float Student::sum=0;
int Student::count=0;
int main( )
{
   Student stud[3]=
   {
      Student(1001,18,70),
      Student(1002,19,78),
      Student(1005,20,98)
   };
   for(int i=0; i<3; i++)
      stud[i].total( );
   cout<<"the average score of 3 students is "
      <<Student::average( )<<endl;
   return 0;
}
```

静态成员应用实例

```
class Date{
    int d,m,y;
    static Date default_date;
public:
    Date(int dd=0,int mm=0,int yy=0);
    //...
    static void set_default(int,int,int);
};
Date::Date(int dd,int mm,int yy){
    d=dd?dd:default_date.d;
    m=mm?mm:default_date.m;
    y=yy?yy:default_date.y;
}
```

CSDN 学院

THANKS

本课程由 迁者-贺利坚 提供

CSDN网站：www.csdn.net
企业服务：http://ems.csdn.net/
人才服务：http://job.csdn.net/
CTO俱乐部：http://cto.csdn.net/
高校俱乐部：http://student.csdn.net/
程序员杂志：http://programmer.csdn.net/

CODE平台：https://code.csdn.net/
项目外包：http://www.csto.com/
CSDN博客：http://blog.csdn.net/
CSDN论坛：http://bbs.csdn.net/
CSDN下载：http://download.csdn.net/