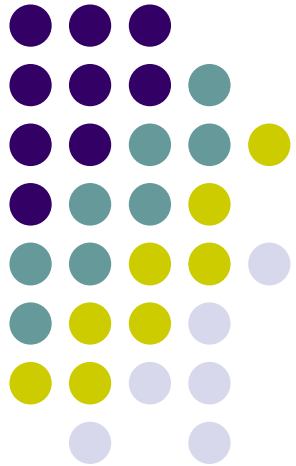
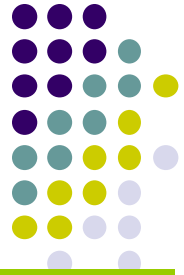
第六章 类和对象

C++程序设计

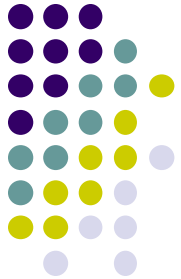


类和对象

——数据成员



```
1.  class  Point
2.  {
3.  private:
4.       int x,y;
5.  public：
6.       Point( ){ }                    //默认构造函数
7.       Point( int a=0, int b=0){x=a;y=b;}//带参数构
8.                                          造函数
9.       Point(Point &a);          //拷贝构造函数
10.       ~Point( ){ }                //析构函数
11.   ……
12.  };    Point b1; Point b2(5,6); Point b3(b2);
```



对象创建　　分配对象空间

构造函数（数据空间初始化） Student(…){…}

Student s(10,”zhang”);

使用

析构函数（主要是堆清理）~Student(){…}

对象销毁　　释放对象空间

类和

1.

2.

3.

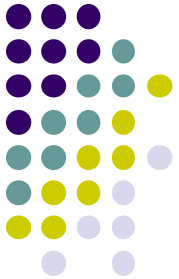
```
Point fun2()
{    Point A(1,2);
     return A; //调用拷贝构造函数
}
int main()
{    Point B;
     B=fun2();
}
{  cout<<p.GetX()<<endl;}
void main()
{
    Point A(1,2);
    fun1(A); //调用拷贝构造函数
}
```

```
class Student
{   static int count;//类的所有对象共享这块数据空间
    int id;          //对象自己的空间
    char name[10]; //对象自己的空间
public:
    static int GetCount(){return count;}
    //静态函数成员访问静态数据成员
    static int SetCount(){count++;}
    //静态函数成员访问静态数据成员
};
```

静态成员在类外初始化：
int Student::count=0;

静态成员的访问方式（类名，对象）
数据成员：类名：cout<<Student::count; //输出初始值0
对象：Student s1,s2;
s1.count=9;
cout<<s2.count; //输出9
函数成员：类名： Student::GetCount( );
对象： s1.GetCount( );

146页 6.4(2)



答:11
12
13
13
13
12
11

6.5 编程题

1．按下列要求编程：

（1）定义一个描述矩形的类Rectangle，包括的数据成员有宽（width）和长（length）；

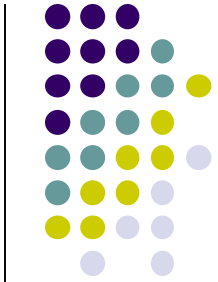
（2）计算矩形周长；

（3）计算矩形面积；

（4）改变矩形大小。

通过实例验证其正确性。

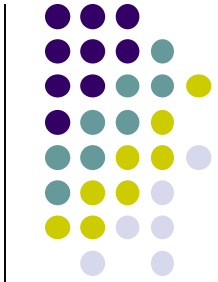
```
#include <iostream.h>
class Rectangle
{
public:
        Rectangle (float x,float y)
          {
                width=x;
                length=y;
          }
         void SetDate (float x,float y)
          {
                width=x;
                length=y;
          }
          void Area()
          {
                s=width*length;
                l=2*(width+length);
          }
          void Print()
          {
                cout<<"area="<<s<<endl;
                cout<<"circumference="<<l<<endl;
          }
private:
          float width,length,s,l;
};
```



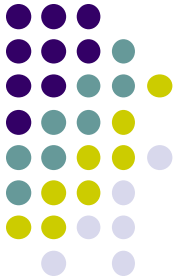
```
void main()
{
        float n,m;
        cout<<"Please input the width and length:"<<endl;
        cin>>n>>m;
        Rectangle d(n,m);
        d.Area();
        d.Print();
        cout<<endl;

        cout<<"Please input the other width and length:"<<endl;
        cin>>n>>m;
        d.SetDate(n,m);
        d.Area();
        d.Print();
}
```

177 页6.5(3)编一个关于求某个多门功课总分和平均分的程序（1）每个学生信息包括姓名和某门功课成绩（2）假设5个学生（3）使用静态成员计算5个学生总成绩和平均分。

```
#include <iostream.h>
#include <string.h>
const int N=5;
class Student
{
public:
    Student(char str[ ]="0",double x=0) //用构造函数进行初始化
        {
            strcpy(name,str);
            score=x;
        }
        void total()  //求总分
        {
            sum=sum+score;  count++;
```



```
static double ave()  //用静态成员函数输出平均分
        {
                return sum/count;
        }
static double sumall()  //用静态成员函数输出总分
        {
                return sum;
        }
private:
        char name[20];
        double score;
        static double sum,count;
};
double Student::sum=0;
double Student::count=0;  //静态数据成员初始化
```



```
void main()
{
        Student stu[N];
        char str[20];
        double x;
        int i;
        for (i=0;i<N;i++)  //以下循环为用构造函数进行初始化
        {
        cout<<"Please input the name and the score:"<<endl;
                cin>>str;
                cin>>x;
                stu[i]=Student(str,x);
        }
        for (i=0;i<N;i++)
                stu[i].total();  //求总分
        cout<<"sum="<<Student::sumall()<<endl;
        cout<<"average="<<Student::ave()<<endl;
}
```

常成员

- 常数据成员
  - 常数据成员即只读变量。
  - 常数据成员初始化是通过采用构造函数的成员初始列表来实现的,中间用逗号分开。

```
class A
{
public:
    A(int i, int j): a(i)
    {  b = j;  }
    A():a(100);
    A(A &i):a(i.a);
private:
    const int a;
    int b;
};
```

这个称为初始化列表
const变量只能这样初始化！

§6.5 常成员

- 【例6.9】分析下列程序的输出结果，学会常数据成员的用法。

```
#include <iostream.h>
#include <string.h>
class A
{
public:
  A(int i);
  void Print()
  {  cout<<a<<','<<b<<','<<r<<endl;  }
  const int &r;
private:
  const int a;
  static const int b;
};
const int A::b=15;
A::A(int i):a(i),r(a)
{   }
```

```
void main()
{
   A a1(10),a2(20);
   a1.Print();
   a2.Print();
}
```

程序输出：

```
10,15,10
20,15,20
```

常成员

- 常成员函数
  - 常成员函数保证不修改任何外部变量。
  - 常对象只能调用它的常成员函数。

```
class B
{
public:
    B(int i, int j)
    {  b1 = i; b2 = j; }
    void Print()
    { cout << b1 << ':'<< b2 << endl; }
    void Print() const
    { cout << b1 << ':'<< b2 << endl; }
private:
    int b1,b2;
};
……
```

```
void main()
{
    B b1(5,10);
    b1.Print();
    const B b2(2,8);
    b2.Print();
}
```

程序输出：

```
5:10
2:8
```

声明常对象的同时必须被初始化，并不能修改对象的数据成员，常对象只能调用类的常成员函数。

常成员函数可以引用任何数据成员，但不可改变外部的变量。
const函数内不能调用非const函数

```
class Stack
{
public:
   void Push(int elem);
   int Pop(void);
   int GetCount(void) const; // const 成员函数
private:
    int m_num;
    int m_data[100];
};

int Stack::GetCount(void) const
{
     ++ m_num;
     Pop();
     return m_num;
}
```

常成员函数可以引用任何数据成员，但不可改变外部的变量。
const函数内不能调用非const函数

```
class Stack
{
public:
   void Push(int elem);
   int Pop(void);
   int GetCount(void) const; // const 成员函数
private:
    int m_num;
    int m_data[100];
};

int Stack::GetCount(void) const
{
    ++ m_num; // 编译错误，企图修改数据成员m_num
    Pop(); // 编译错误，企图调用非const 函数
    return m_num;
}
```

下列关于常成员的描述中，错误的是（C）。

A. 常成员是用关键字const说明的

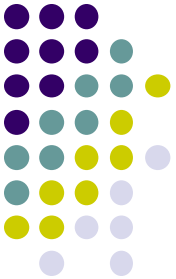
B. 常成员有常数据成员和常成员函数两种

C. 常数据成员的初始化是在类体内定义它时进行的

D. 常数据成员的值是不可以改变的

类和对象

——友员



- 友员提供了类的成员函数与一般函数之间，不同类或对象的成员函数之间，进行数据共享的机制；
- 如果没有友员机制只能将类的数据声明为公共的，但这样就破坏了数据的封装性；
- 友员可以实现通过友员对象名访问类的私有和保护成员。

友元函数和友元类

- 友元函数
  - 友元函数是说明在类体内的一般函数，它不是这个类中的成员函数，但是它访问该类所有成员。
  - 友元函数举例如下（例6.11）：

```
class Point
{
public:
    Point(double i,double j)
    {  x=i;y=j;  }
    void Print()
    {  cout<<'('<<x<<','<<y<<')'<<endl;  }
    friend double Distance(Point a,Point b);
private:
    double x,y;
};
double Distance(Point a,Point b)
{  double dx=a.x-b.x; double dy=a.y-b.y;
    return sqrt(dx*dx+dy*dy);
}
```

友元函数的说明

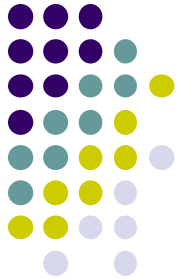
友元函数的实现
可访问Point中私有成员

§6.6 友元函数和友元类

```
void main()
{
    double d1=3, d2=4, d3=6, d4=8;
    Point p1(d1,d2), p2(d3,d4);
    p1.Print();
    p2.Print();
    double d = Distance(p1,p2);
    cout << "Distance=" << d << endl;
}
```

友元函数与普通函数的调用方式相同

书147页 6.4(3)



答： 2005/10/1
2005/12/9

§6.6 友元函数和友元类

- 友元类
  - 将一个类作为另一个类的友元，则该类称为友元类。友元类中的所有成员函数都是这个类的友元函数。

```
class A
{
friend class B;
public:
     A(int i, int j): a(i)
     {  b = j;  }
private:
     const int a;
     int b;
};
……
```

class B中任何函数可不受限制的访问class A中任何元素。

友元类举例

```
class A
{    friend class B;
     public:
       void Display()
        {cout<<x<<endl;}
     private:
       int x;
}
class B
{    public:
       void Set(int i);
       void Display();
     private:
       A a;
};
```

```
void B::Set(int i)
{
    a.x=i;
}
void B::Display()
{
    a.Display();
}
```

友元关系是单向的

如果声明B类是A类的友元，B类的成员函数就可以访问A类的私有和保护数据，但A类的成员函数却不能访问B类的私有、保护数据。

下面关于友元的描述正确的是
（ A ）

A. 友元函数不是类的成员，但可以访问所在类的任何成员。

B. 友元函数中能访问所在类的私有成员但不能访问类的保护成员。

C. 如果一个类成为另一个类的友元类，则两个类可以互相访问对方的私有成员。

友元函数和友元类

- 【例6. 12】分析下列程序的输出结果，熟悉友元类的用法。

```
#include <iostream.h>
class X
{
friend class Y;
public:
  X(int i)
   {  x=i;  }
    void Print()
   {  cout<<"x="<<x<<','<<"s="<<s<<endl;  }
private:
  int x;
  static int s;
};
```

友元类

友元函数和友元类

```
int X::s=5;
class Y
{
public:
  Y(int i)
  {  y = i;  }
  void Print(X &r)
  {  cout<<"x="<<r.x<<','<<"y="<<y<<endl;  }
private:
   int y;
};
void main()
{
  X m(2);
  m.Print();
  Y n(8);
  n.Print(m);
}
```

访问X中任何成员

x=2,s=5

x=2,y=8

其中访问类X中私有成员

例：定义Boat与Car两个类，二者都有weight属性，
定义二者的一个友元函数totalWeight()，计算二者的重量和。

```
#include <iostream.h>
class Boat;
class Car
{
    private:
        int weight;
    public:
        Car(int j)    {weight = j;}
        friend int totalWeight(Car &aCar, Boat &aBoat);
};
class Boat
{
    private:
        int weight;
    public:
        Boat(int j)  {weight = j;}
        friend int totalWeight(Car &aCar,Boat &aBoat);
};
```

```
int totalWeight(Car &aCar, Boat &aBoat)
{
        return aCar.weight + aBoat.weight;
}

void main()
{
        Car c1(4);
        Boat b1(5);
        cout << totalWeight(c1，  b1) << endl;
}
```

阅读下面有关友元函数的程序，请按要求补写适当代码。

```
#include <iostream.h>
class two;
class one
{
        int x;
public:
        one(int a) {x = a;}
        void func(two&);
        int getX() {return x;}
};

class two
{
        int y;
public:
        two(int a) {y = a;}
        int getY() {return y;}
        _____;
// 将类one中函数func()声明为two的友元函数
};
```

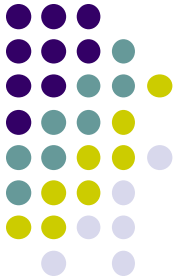
```
_____          // 类one中的函数func的实现
{
        x = t.y;
}

void main()
{
        one obj1(10);
        two obj2(20);
        cout<<obj1.getX()<<endl;
        cout<<obj2.getY()<<endl;
        _____ ;
// 调用obj1中func函数获取obj2中y的值。
        cout<<obj1.getX()<<endl;
}
```

firend void one::func(two&) ;

void one::func(two& t)

obj1.func(obj2) ;

20171115 作业



试建立一个类Worker用于描述职工，具体要求如下：
（1）私有数据成员
unsigned int id: 职工号。
char name[11]:姓名
float salary: 工资。（有一符号常量为工资最低值，设为2000）
int level: 技术等级（1~20级），1级为最低，每个等级差别200元
（2）公有成员函数
Worker ( )：构造函数，初始化数据成员为默认值（自动生成职工号，姓名为空，工资默认最低）。
Worker (...)：构造函数(自己定义参数)，用参数初始化数据成员。参数有各个数据成员。
void setName(...): 设置职工姓名
void infoList( ): 输出职工的各项信息。
void setLevel(...): 修改技术等级level的值。每增加一个等级工作增加200元。
（3）static变量total计算工资总数和函数Average()计算平均工资。
（4）在主程序中定义N(=5)个Worker对象作为测试数据，完成对Worker类和程序的测试。
（5）在主程序中对其中某个工人发放奖金500元。