嵌入式系统工程师

类与对象

课程内容

- 类、对象概念
- 构造函数与析构函数
- const修饰的成员函数

课程内容

- 类、对象概念
- 构造函数与析构函数
- const修饰的成员函数

结构与类

- C结构：结构体成员是数据
- C++中的结构：结构的成员是数据和函数
  成员的访问权限有：
  1）私有成员：private，它只能被该结构中的其它成员访问
  2）公有成员：public，既可以被结构中的其它成员访问，又可以被结构外的其它部分访问
  - 把结构保留字struct换为class，既成为类的定义

类的声明

- 结构与类的区别是：默认访问级别不同
- 类使用注意：
  - 类的声明中的private和public两个关键字可以按任意顺序出现任意次。为了使程序更加清晰，把所有私有成员和公有成员归类放在一起
  - 除了private和public之外，还有protected(保护性成员)关键字
  - 数据成员可以是任何数据类型，但不能用auto、register或extern说明
  - 不能在类的声明中给数据成员进行初始化

类与对象

- 类与对象的关系
  - 类—自定义数据类型　对象(实例)—变量
- 对象的定义
  - 声明类的同时定义对象：这是全局的对象
  - 在使用对象时再定义：这是局部对象
  - 说明：
    - 定义对象之后，会为对象分配存储空间
    - 全局的对象在任何函数内均可使用，而局部对象只能在定义对象的函数中使用

成员函数的定义

- 类中有两种成份，即数据成员和成员函数

定义成员函数的方式:

- 在类声明中只给出成员函数的原型，其具体定义在类的外部。一般格式为:

```
返回类型   类名::函数名(参数表)
{       //函数体
}
```

- **在类内直接定义:**

```
class  class_name{
//.....
返回类型   函数名（参数表）
{
        //函数体
 }
};
```

类的作用域

- 类的作用域就是指在类声明时一对花括号所形成的作用域。
  - 一个类的所有成员都在该类的作用域内。
  - C++把类的所有成员都作为一个整体的相关部分（无定义的先后顺序之分）
  - 一个类的任何成员可以不受限制的引用该类的其它成员。
- 类作用域之外
  - 只有公有成员才可被外部函数引用，而私有成员是不允许被外部函数引用的。这体现了类的封装功能

对象的定义及引用

对象的引用

是指对对象成员的引用。不论是数据成员，还是成员函数，只要是公有的，就可以被外部函数直接引用：

- 对象名.数据成员名
- 对象名.成员函数名(实参表)

对象的定义及引用

- 说明:
  - op.setPoint(1,2)实际上是op.point::setPoint(1,2)的缩写形式，两者等价
  - 外部函数不能引用对象的私有成员。如:

```
int   main()
{
        Point  op;
        op.x  =  5;              //错误
        …
 }
```

对象的定义及引用

- 对象赋值语句

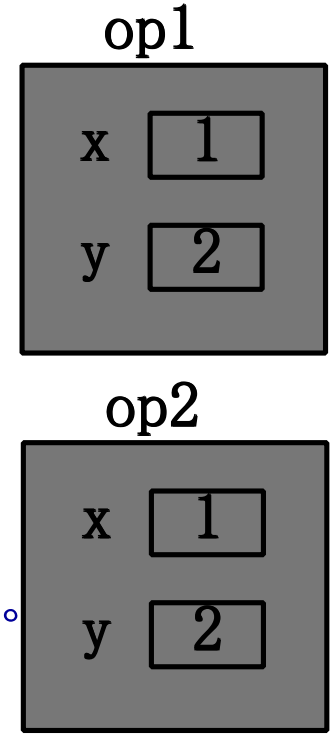
同一种类的对象可以相互赋值。如:

```
Point op1,op2;
op1.setPoint(1,2);
op2 = op1;
```

此时是将所有数据成员逐个地进行拷贝。

```
cout<<op2.getx();          //结果为1
```

例2.1 对象赋值



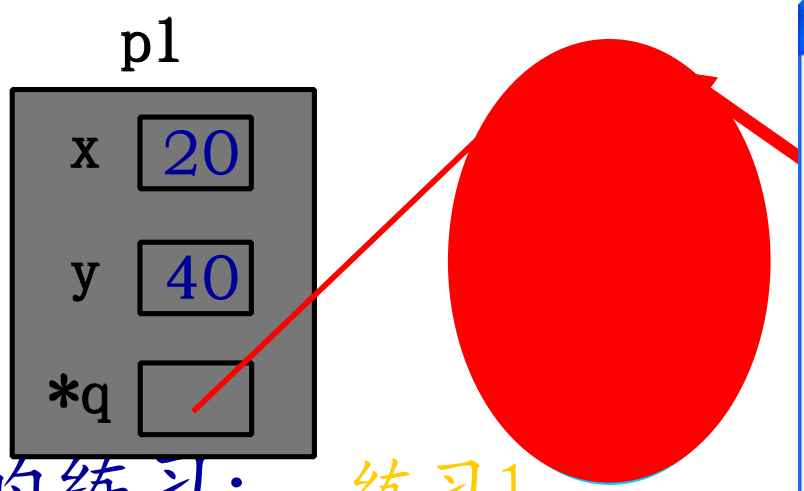
对象的定义及引用

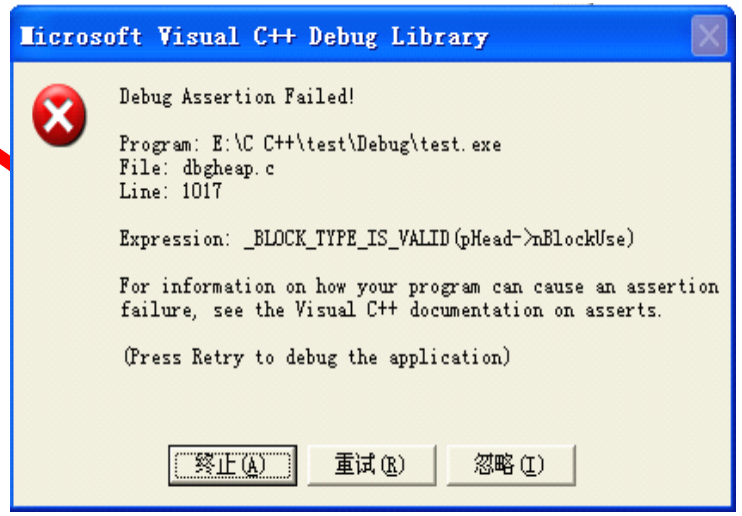
- 说明：
  - 在使用对象赋值时，两个对象的类型必须相同
  - 两个对象之间的赋值，仅仅使这些对象中的数据相同，而两个对象仍是独立的对象，它们有各自的数据成员

对象的定义及引用

- 将一个对象的值赋给另一个对象时，多数情况下都是成功的，但当类中存在指针时，可能会产生错误

p2 = p1





➢ 类的练习：练习1

课程内容

- 类、对象概念
- 构造函数与析构函数
  - 构造函数
  - 析构函数
  - 拷贝构造函数
- const修饰的成员函数

构造函数与析构函数

- 当定义一个类对象时，编译程序需要为对象分配存储空间，进行必要的初始化，初始化的工作随着类的不同而不同。在C++中，可以由构造函数来完成这初始化工作
- 与构造函数对应的是析构函数，当撤消类对象时，析构函数就回收存储空间，并做一些善后工作。
- 构造函数和析构函数都属于某一个类，它们可以由用户提供，也可以由系统自动生成

课程内容

- 类、对象概念
- 构造函数与析构函数
  - 构造函数
  - 析构函数
  - 拷贝构造函数
- const修饰的成员函数

构造函数

- 构造函数是一种特殊的成员函数，它主要用于初始化对象
- 每一个类都必须有一个构造函数。如果没有给类定义构造函数，则编译系统自动地生成一个空的构造函数（参数和函数体均为空）

  形如：Complex(){ }
- 如果用户自定义了构造函数，那么C++就不提供默认构造函数

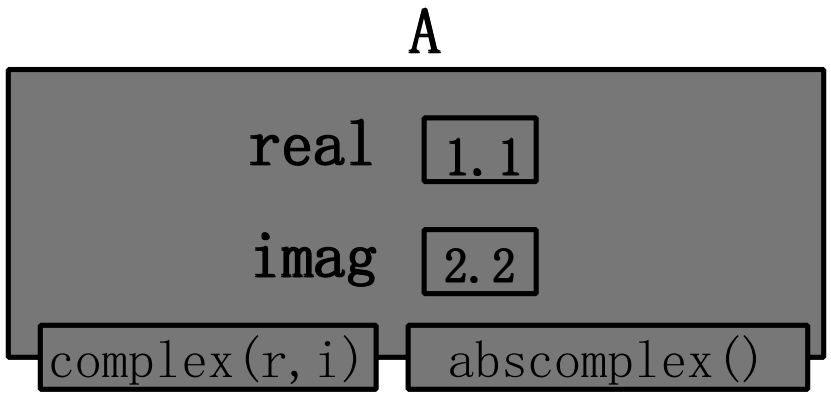
构造函数

- 构造函数具有一些特殊的性质
  - 构造函数的名字必须与类名相同
  - 定义对象时被系统自动调用
  - 可以有任意类型的参数（也可以不带参数），但不能有返回值，void也不可以
  - 被定义为公有的，但其它时间无法被调用

构造函数

➢ 例2.2

```
class Complex{
private:
   double real, imag;
public:
   Complex(double r,double i){
       real=r; imag=i; }
   double absComplex(){
     double t;
     t = real*real+imag*imag;
      return sqrt(t);
   }
};
int main(){
     Complex A(1.1,2.2);      //定义类的对象A时调用构造函数
     cout<<" abs of complex A="<<A.absComplex()<<endl;
}
```



构造函数

- 调用构造函数的条件
  - 1.定义对象直接调用

    ```
    Complex A(1.1,2.2);
    ```

  - 2.动态分配对象空间时

    ```
    Complex *p = new Complex(3.0,4.0);
    ```

  - 3.定义无名对象(没有名字的对象)

    ```
    Complex (2,4);
    new Complex (4,8);
    ```

重载构造函数

- 构造函数也可以带缺省参数，
- 构造函数也可以被重载，以适应不同的场合。
- 注意：

  一个类中既有重载构造函数，又有缺省参数构造函数，有可能产生二义性

课程内容

- 类、对象概念
- 构造函数与析构函数
  - 构造函数
  - 析构函数
  - 拷贝构造函数
- const修饰的成员函数

析构函数

- 析构函数也是一种特殊的成员函数. 它执行与构造函数相反的操作, 通常用于执行一些清理任务, 如释放分配给对象的内存空间等
- 析构函数有以下一些特点:
  - 析构函数与构造函数名字相同, 但它前面必须加一个波浪号(~)
  - 析构函数没有参数, 也没有返回值, 而且不能重载, 因此在一个类中只能有一个析构函数
  - 当撤消对象时, 编译系统会自动地调用析构函数

析构函数

例2.3 运行分析

```
class Complex{
      double real, imag;
public:
Complex(double r=0.0,double i=0.0){
      cout<<"construction..."<<endl;
      real=r; imag=i;
}
~Complex(){cout<<"destruction..."<<endl
double absComplex(){
      return sqrt(real*real+imag*imag);
}
};
int main(){
   Complex A(1.1,2.2);
   cout<<"abs of complex A="<<A.absComplex()<<endl;
}
```

结果：

construction…
abs of complex A=2.45967
destruction…

析构函数

- 调用析构函数的条件
  - 1.对象自动退出生命周期

    比如：全局对象、局部对象

```
{ Complex A(1.1,2.2); }
void fun(Complex p){ };
```

  - 2.程序员手动释放对象指针

```
Complex *p = new Complex(5,6);
delete p;
```

析构函数

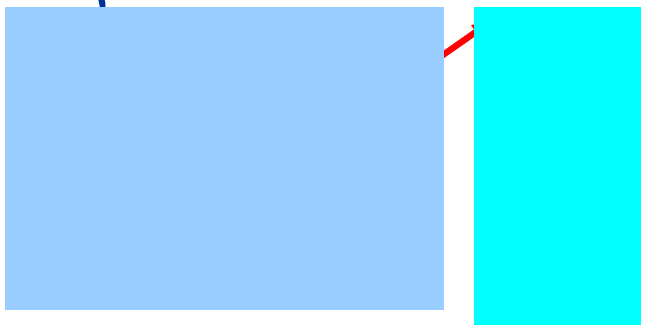
- 说明:
  - 每个类必须有一个析构函数。若没有显式地为一个类定义析构函数，编译系统会自动地生成一个空的析构函数。如:

```
~Complex(){}
```

  - 对于大多数类而言，缺省的析构函数已经足够了。但是，如果在一个对象完成其操作之前需要做一些内部处理，则应该显式地定义析构函数，例如:

```
class StringData{
private:  char *str;
public:
    StringData(char *s){
   str=new char[strlen(s)+1];
       strcpy(str,s);
~string_data() { delete str; }
};
```

```
int main(){
   StringData  x("abc");
     //…
}
```

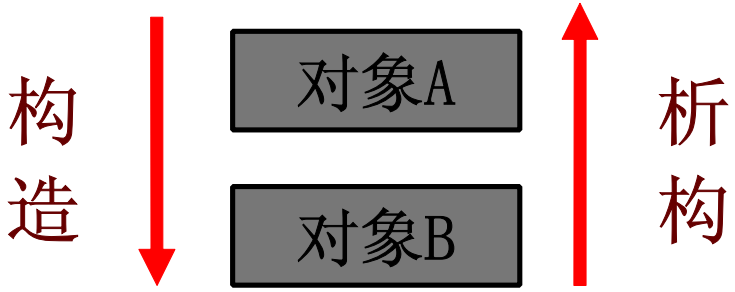


析构函数

在同一作用域内类对象的构造和析构的执行顺序：先构造的后析构

```
class XYZ{
    ......
};
int main(){
    XYZ  A,B;
    ......
}
```

例2.4



课程内容

- 类、对象概念
- 构造函数与析构函数
  - 构造函数
  - 析构函数
  - 拷贝构造函数
- const修饰的成员函数

拷贝构造函数

- 拷贝构造函数是一种特殊的构造函数。它用于依据已存在的对象建立一个新对象。典型的情况是，将参数代表的对象逐域拷贝到新创建的对象中
- 每个类都有一个构造函数，它可以由用户根据需要自己定义，或者系统也可以为类产生一个缺省的拷贝构造函数

拷贝构造函数

- 缺省的拷贝构造函数

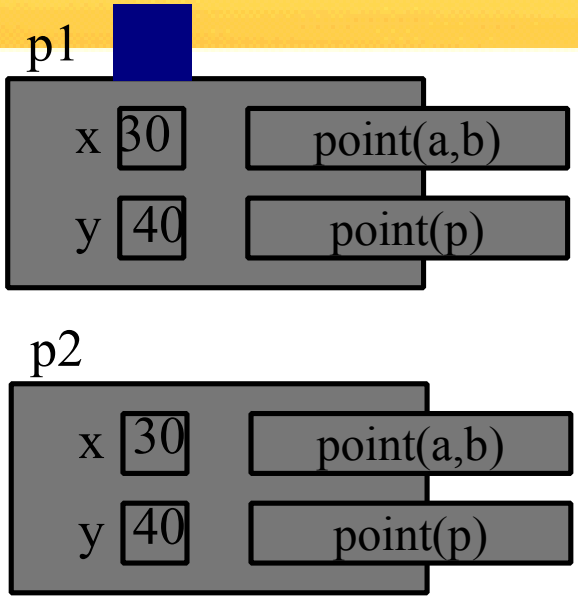
```
classname(const classname &ob)
{
        //按数据成员的顺序逐一赋值
}
```

- 自定义拷贝构造函数

```
classname(const classname &ob)
{
        //自定义拷贝构造函数的函数体
}
```

其中ob是用来初始另一个对象的对象的引用

```
class Point{                例2.5
        int x,y;
public:
        Point(int a,int b)    //构造函数
        { x=a; y=b; }
        Point(const Point &p)  //拷贝构造函数
        { x=p.x; y=p.y;}
        //……
};
int main(){
        Point p1(30,40);  //定义对象p1
        Point p2=p1;      //显式调用拷贝构造函数,通过对象p1创建对象p2
        ……
}
```

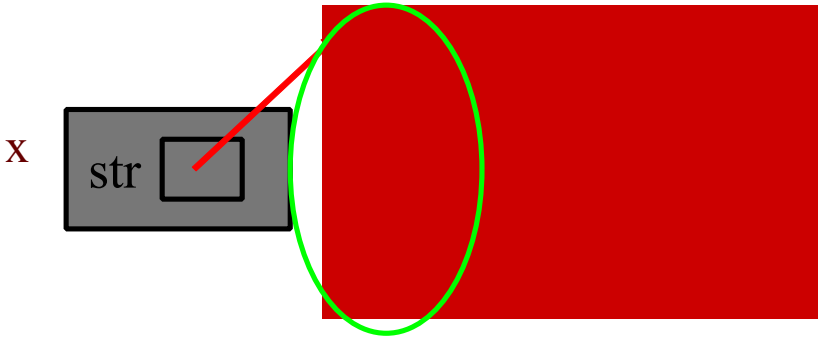


其中Point p2(p1)也可以写成赋值形式的调用。

- 说明:
  - 通常缺省的拷贝构造函数是能够胜任工作的，但若类中有指针类型时，按成员复制的方法有时会产生错误

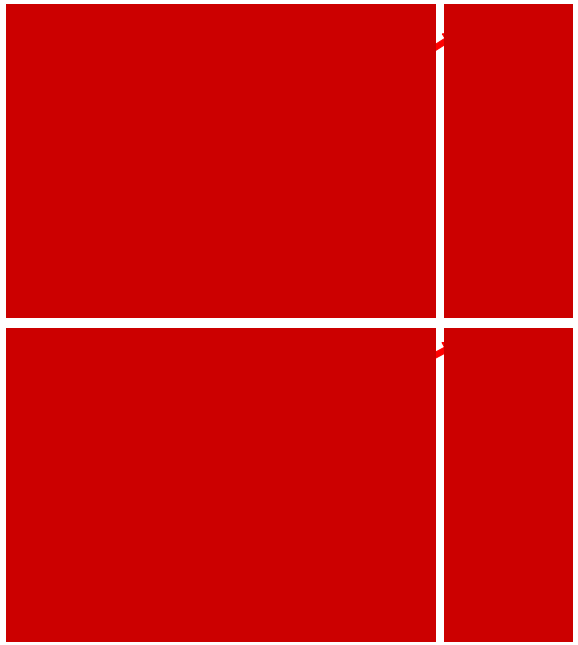
```
class StringData{
private:  char *str;
publie:
     StringData(char *s){
          str = new char[strlen(s)+1];
          strcpy(str,s);
     }
     ~StringData() { delete str; }
     //…
};
```

```
int main(){
     StringData  x("abc");
     StringData  y(x);
}
```



当类中有指针类型时，正确的处理方法是增加自己的拷贝构造函数：例2.6

```
class StringData{
private:  char *str;
public:
        StringData(char *s){
            str=new char[strlen(s)+1];
            strcpy(str,s);
        }
        StringData(const StringData &p){
            str=new char[strlen(p.str)+1];
            strcpy(str,p.str);
        }
        ~StringData() { delete str; }
        //…
};
int main(){
    StringData  x(“abc”);
    StringData  y(x);
}
```



构造函数

- 调用拷贝构造函数的条件
  - 1.定义对象时

```
Point p1(30,40);
Point p2(p1);// Point p2 = p1;
```

  - 2.函数的参数是对象时

```
void test(Point p);
test(p1);
```

  - 3.函数的返回值是对象时

```
Point test();
test();
```

构造函数

- 提示：利用无名对象初始化对象系统不会调用拷贝构造函数。
- 例如：

```
Point A = Point(4,5);
Point test(Point p){retrun p;}
test(Point(4,5));
```

课程内容

- 类、对象概念
- 构造函数与析构函数
  - 构造函数
  - 析构函数
  - 拷贝构造函数
- const修饰的成员函数

const修饰的成员函数

- 由于成员函数可以任意访问类内的任何数据成员，但是当我们不愿意让其修改数据成员时，我们可以用const修饰类的成员函数，一般形式为：
  - Class class_name{

```
    private:
            ......
    public:
            (type) function_name(...)const
            {
                    ......
            }
};
```

const修饰的成员函数

- 用const修饰的成员函数时，成员函数体内不可以修改本结构体内的任何数据成员
- 但有一种情况是例外的就是当数据成员类型符前用mutable修饰时，在const修饰的成员函数体内该数据成员是可以改变的
- 例：2.7_fun_const.cpp

小结

- 本章主要讲了C++中类与对象的一些基础知识：
  - 构造函数：主要用于在定义对象时进行初始化工作；与类名相同，没有返回值（即使是void也不可以），可以没有参数（也可以带有多个参数）
  - 析构函数：主要用于在对象的声明周期结束时，进行善后工作（或者进行一些结束时的操作）；与构造函数名字相同，因为与构造函数进行相反的造作所以前面加上“~”，不可以有返回值，不可以有参数
  - 拷贝构造：主要用于当类中有指针数据成员时需要自己定义拷贝构造函数，拷贝构造函数与构造函数名字相同，同样也不可以有返回值，形参为该类的引用

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400 705 9666 . Email: edu@sunplusapp.com . BBS: bbs.sunplusedu.com