面向对象程序设计
420420

构造函数

- 1、什么是构造函数
- 2、构造函数的定义

27.1 什么是构造函数

- 建立一个对象时，通常最需要立即做的工作是初始化对象，如对数据成员赋初值。

- 构造函数就是用来在创建对象时初始化对象，为对象数据成员赋初始值的。

- 类的数据成员是不能在类定义时初始化的，例如：

```
class  Point  {  //Point类
       int  x=0,  y=0;  //错误，不能在类定义中对数据成员初始化
       …//其他成员
}
```

- 原因是类定义并没有产生一个实体，而是给出了一个数据类型，不占用存储空间，因而也无处容纳数据。

- 如果一个类中所有的数据成员是公有的，则可以在定义对象时对数据成员进行初始化，例如：

```
class   Point   {     //Point类定义
public:
        int   x,   y;  //数据成员声明
        …              //其他成员
};
Point   one={10,10};  //对象初始化
Point   A[3]={{10,10},{20,20},{30,30}};  //对象数组初始化
```

- 如果类中的数据成员是私有的，如private的或protected的，就不能用这种方法初始化，因为外部不能直接访问私有的数据成员。

27.2 构造函数的定义

- C++提供了构造函数（constructor）来处理对象的初始化问题。构造函数是类的一种特殊成员函数，不需要人为调用，而是在建立对象时自动被执行。

- 换言之，在建立对象时构造函数被自动执行了，程序员因此有机会在这里进行对象的初始化工作。

- 1. 定义构造函数
- C++规定构造函数的名字与类的名字相同，并且不能指定返回类型。定义形式为：

```
类名(形式参数列表)
{
        函数体
}
```

- 构造函数可以没有形参，有如下两种形式：

```
类名()
{
        函数体
}
```

```
类名(void)
{
        函数体
}
```

27.2 构造函数的定义

- 与其他任何函数一样，构造函数可以声明为内联的。
- 只要创建类类型的新对象，都要执行构造函数。因此，构造函数的主要用途是初始化类的数据成员。

- 对于有参数的构造函数，定义对象的一般形式为：

```
类名  对象名1(实参列表), 对象名2(实参列表), ......;
```

- 对于无参数的构造函数，定义对象的一般形式为：

```
类名  对象名1, 对象名2, ......;
```

【例27.1】构造函数举例

```
1  #include <iostream>
2  using namespace std;
3  class Cuboid { //Cuboid类表示长方体
4  public:
5      Cuboid(int l,int h, int d); //构造函数
6      int volumn() { return length*height*depth; } //计算体积
7  private:
8      int length,height,depth; //长、高、深
9  };
```

27.2 构造函数的定义

```
10  Cuboid::Cuboid(int  l,int  h,int  d)  //外部定义的构造函数
11  {
12      length=l,  height=h,  depth=d;  //初始化数据成员
13      cout<<"Cuboid:  "<<"L="<<l<<",H="<<h<<",D="<<d<<endl;
14  }
15
16
17  int  main() {
18      Cuboid  a(1,2,3);  //定义长方体对象a，调用构造函数初始化
19      cout<<"volumn="<<a.volumn()<<endl;  //输出体积
20      Cuboid  b(10,20,30);  //定义长方体对象b，调用构造函数初始化
21      cout<<"volumn="<<b.volumn()<<endl;  //输出体积
22      return  0;
23  }
```

输出结果：

```
Cuboid:  L=1,H=2,D=3
volumn=6
Cuboid:  L=10,H=20,D=30
volumn=6000
```

- 关于构造函数的说明：
- （1）构造函数是在创建对象时自动执行的，而且只执行一次，并先于其他成员函数执行。构造函数不需要人为调用，也不能被人为调用。
- （2）构造函数一般声明为公有的（public），因为创建对象通常是在类的外部进行的。如果构造函数声明为保护的（protected）或私有的（private），那就意味着在类外部创建对象（并调用构造函数）是错误的。换言之，这样的类是不能由外部实例化，只能由类内部实例化，这种情况不是通常的做法。

- （3）在构造函数的函数体中不仅可以对数据成员初始化，而且可以包含任意其他功能的语句，例如分配动态内存等，但是一般不提倡在构造函数中加入与初始化无关的内容。

- （4）每个构造函数应该为每个数据成员提供初始化。否则将使那些数据成员处于未定义的状态。而使用一个未定义的成员是错误的。

- （5）带参数的构造函数中的形参，是在定义对象时由对应的实参给定的，用这种方法可以方便地实现对不同对象进行不同的初始化。需要注意，实参必须与构造函数的形参的个数、次序、类型一致。

- 2．构造函数初始化列表
- 与普通函数一样，构造函数具有函数名、形参列表和函数体。与其他函数不同的是，构造函数可以包含一个构造函数初始化列表，一般形式为：

```
类名(形式参数列表): 构造函数初始化列表
{
        函数体
}
```

- 与其他的成员函数一样，构造函数可以定义在类的内部或外部，但构造函数初始化列表只在构造函数的定义中而不是函数原型声明中指定。
- 从初始化角度来看，可以认为构造函数分两个阶段执行：①初始化阶段；②普通的计算阶段。初始化阶段由构造函数初始化列表组成，计算阶段由构造函数函数体的所有语句组成，初始化阶段先于普通的计算阶段。即

```
类名(形式参数列表):  初始化阶段
{
        普通的计算阶段
}
```

【例27.2】构造函数初始化列表举例

```
1   #include <iostream>
2   using namespace std;
3   class Cuboid { //Cuboid类表示长方体
4   public:
5         Cuboid(int l,int h, int d); //构造函数
6         int volumn() { return length*height*depth; }; //计算体积
7   private:
8         int length,height,depth; //长、高、深
9   };
10  Cuboid::Cuboid(int l,int h,int d) :length(l),height(h),depth(d)
      //带构造函数初始化列表的构造函数
11  {
12        cout<<"Cuboid: "<<"L="<<l<<",H="<<h<<",D="<<d<<endl;
13  }
```

- 关于构造函数初始化列表的说明。
- （1）有时必须用构造函数初始化列表。
- 如果没有为类类型的数据成员提供初始化列表，编译器会隐式地使用该成员的默认构造函数。如果那个类没有默认构造函数，则编译器会报告错误。在这种情况下，为了初始化类类型的数据成员，必须提供初始化列表。

- 一般地，没有默认构造函数的成员，以及const或引用类型的成员，都必须在构造函数初始化列表中进行初始化。

27.2 构造函数的定义

```
class  point
{
private:
        int  x;  int  y;
public:
        point(int  i,int  j){  x=i,y=j;  } //没有默认构造函数
        void  print(){    cout<<x<<','<<y<<endl;  }
};
class  pointTest
{
private:
        point  a;   //私有数据成员a是point类型，即pointTest类的数据成员a是point类的对象
public:
        pointTest(int  i,int  j):a(i,j){  } //只能在初始化列表里对成员a初始化
};
```

- （2）成员初始化的次序

- 每个成员在构造函数初始化列表中只能指定一次，但构造函数初始化列表仅指定用于初始化的数据成员的值，并不指定这些初始化执行的次序。数据成员被初始化的次序就是数据成员的声明次序。第1个成员首先被初始化，然后是第2个，依次类推。

- 一般地，按照与成员声明一致的次序编写构造函数初始化列表，并且尽可能避免使用成员来初始化其他成员。

- （3）初始化式可以是任意表达式。

构造函数

- 3、构造函数的重载
- 4、带默认参数的构造函数

- 在一个类中可以定义多个构造函数版本，即构造函数允许被重载，只要每个构造函数的形参列表是唯一的。一个类的构造函数数量是没有限制的。一般地，不同的构造函数允许建立对象时用不同的方式来初始化数据成员。

【例27.3】构造函数重载举例

```
1  #include <iostream>
2  using namespace std;
3  class Point { //Point类表示平面上的点
4  public:
5      Point() { x=y=0; } //无参数的构造函数
6      Point(int a, int b) : x(a), y(b) { } //有参数的构造函数
7      void display() { cout<<"x="<<x<<",y="<<y<<endl; }
8  private:
9      int x,y; //私有数据成员，坐标值
10 };
11 int main()
12 {
13     Point m; //定义Point对象m，m没有实参，调用无参数的构造函数进行初始化
14     m.display(); //显示m坐标
15     Point n(1,2); //定义Point对象n，n有实参, 调用有参数的构造函数初始化
16     n.display(); //显示n坐标
17     return 0;
18 }
```

输出结果：

x = 0 , y = 0

x = 1 , y = 2

27.3 构造函数的重载

- 尽管在一个类中可以包含多个构造函数，但是对于每一个对象来说，建立对象时只执行其中一个，并非每个构造函数都被执行。

27.4 带默认参数的构造函数

- 构造函数的参数允许使用默认值。对类的设计者来说，使用默认参数可以减少代码重复；对类的使用者者来说，使用默认参数可以方便地用适当的参数进行初始化。

27.4 带默认参数的构造函数

【例27.4】带默认参数的构造函数举例

```
1  #include  <iostream>
2  using  namespace  std;
3  class  Point  {  //Point类表示平面上的点
4  public:
5      Point(int  a=0,int  b=0)  :  x(a),y(b)  {  }//带默认参数的构造函数
6      void  display()  {  cout<<"x="<<x<<",y="<<y<<endl;  }
7  private:
8      int  x,y;  //坐标值
9  };
10 int  main()
11 {
12     Point  k,m(1),n(1,2);  //定义Point对象k,m,n，调用构造函数初始化
13     k.display();  m.display();  n.display();  //显示坐标
14     return  0;
15 }
```

输出结果：

```
x = 0 , y = 0
x = 1 , y = 0
x = 1 , y = 2
```

27.4 带默认参数的构造函数

- 关于构造函数默认参数的说明：
- （1）必须在类的内部指定构造函数的默认参数，不能在类外部指定默认参数。

```
class Point {
public:
        Point(int a,int b);    //构造函数
        void display() { cout<<"x="<<x<<",y="<<y<<endl; }
  private:
        int x,y; //坐标值
};
Point::Point(int a=0,int b=0) //错误，不能在类外指定默认参数
{
        x=a; y=0;
}
```

- （2）如果构造函数的全部参数都指定了默认值，则在定义对象时可以给一个或几个实参，也可以不给出实参。这时，就与无参数的构造函数有歧义。

```
class  Point  {
public:
        Point(){x=y=0;}                                          //无参数的构造函数
        Point(int  a=1,int  b=1):x(a),y(b)  {  };        //带2个参数的构造函数
        void  display()  {  cout<<"x="<<x<<",y="<<y<<endl;  }
  private:
        int  x,y;  //坐标值
};
int  main()
{
        Point  k;  //出现歧义，既可以x=0，y=0；也可以x=1，y=1
        return  0;
}
```

- （3）在一个类中定义了带默认参数的构造函数后，不能再定义与之有冲突的重载构造函数。

- 一般地，不应同时使用构造函数的重载和带默认参数的构造函数。