

第12周 对象的构造和析构

准备知识

- 函数的缺省参数
- 函数重载

函数的缺省参数

- 函数在声明时可以预先给出缺省的形参值，调用时如给出实参，则采用实参值，否则采用预先给出的缺省形参值。缺省值在声明或定义处给出均可，但不能重复给出，一般在声明时给出。

```
int add(int x = 5, int y = 6);  
int add(int x, int y) {  
    return x + y;  
}
```

OR

```
int add(int x = 5, int y = 6){  
    return x + y;  
}
```

```
int main(int argc, const char* argv[]) {  
    cout << add(3, 7) << endl;  
    cout << add(7) << endl;  
    cout << add() << endl;  
    return 0;  
}
```


函数的缺省参数

- 有缺省参数的形参必须连续存在与形参列表的最右，也就是说缺省形参值的右面不能有无缺省值的参数。因为调用时实参与形参的结合是从左向右的顺序。

```
int add(int x, int y = 5, int z = 6); // 正确?
```

```
int add(int x = 1, int y = 5, int z); // 正确?
```

```
int add(int x = 1, int y, int z = 6); // 正确?
```

函数的缺省参数

成员函数的缺省参数

- 类成员函数的默认值，**应该**写在类定义中，而不**应该**写在类定义之外的函数实现中。

```
class cTime{
public:
    void Set(int newH = 0, int newM = 0, int newS = 0);
    void Show();
private:
    int hour;
    int minute;
    int second;
};
```

函数重载

- C++允许功能相近的函数在相同的作用域内以相同函数名声明，从而形成重载。方便使用，便于记忆。

```
int add(int x, int y);  
float add(float x, float y);
```

形参类型不同

```
int add(int x, int y);  
int add(int x, int y, int z);
```


形参个数不同

函数重载


➤ 注意事项:

- 重载函数的形参必须不同: **个数**不同或**类型**不同。
- 编译程序将根据实参和形参的类型及个数的最佳匹配来选择调用哪一个函数。

```
int add(int x,int y);  
int add(int a,int b);  
编译器不以形参名来区分
```



```
int add(int x,int y);  
void add(int x,int y);  
编译器不以返回值来区分
```



- 不要将不同功能的函数声明为重载函数, 以免出现调用结果的误解、混淆。这样不好:

```
int add(int x, int y)  
{ return x + y; }
```

```
float add(float x,float y)  
{ return x - y; }
```

函数重载

范老师郑重强调：
函数参数默认值相当于
一组函数重载

函数名：**相同**

形参表：**不能完全相同**

返回值：**I don't care !!!**

const：**可以在前两者完全相同时构成重载**

对象的构造与析构

构造函数

- 构造函数的作用是在对象被创建时使用特定的值构造对象，或者说将对象**初始化**为一个特定的状态
- 构造函数特定的语法要求：
 - 函数名与类名相同，且不允许有任何返回值
 - 允许为**内联函数**、**重载函数**、**带默认形参值的函数**
- 在对象创建时**由系统自动调用**
- **不需要任何参数**(无参数或参数都有缺省值)**进行调用**的构造函数称为**缺省构造函数**(也称为**默认构造函数**)，创建对象如果不提供任何初始化参数进行，则系统会调用缺省构造函数。
- 如果程序中未声明构造函数，则系统自动产生出一个参数列表为空的**隐含构造函数**(它主要负责基类对象和成员对象的构造函数的调用，在会讲)

注意，隐含构造函数也是一种缺省构造函数

范：是否是缺省构造函数主要看能否无参数调用。所有参数都有默认值的构造函数也是缺省构造函数

对象的构造与析构

构造函数举例

- 举例1: 构造函数特定的语法要求
 - 与类同名
 - 不能有返回值类型!!!
 - 可以重载但不能同时存在无参数构造函数和全部参数都有缺省值的构造函数
 - 函数体内没有return表达式
 - 构造函数会自动被调用
- 举例2: 系统只在没有编写任何构造函数的情况下, 才会自动添加隐含构造函数。
- 举例3: 对象构造的方式, 局部或new都会触发构造函数

对象的构造与析构

拷贝构造函数

- 拷贝构造函数是一种特殊的构造函数，其形参为本类的对象引用。

```
class 类名
```

```
{
```

```
public :
```

```
    类名 ( 形参 ); //构造函数
```

```
    类名 ( const 类名 &对象名 ); //拷贝构造函数
```

```
    ...
```

```
};
```

```
类名::类 ( const 类名 &对象名 ) //拷贝构造函数的实现
```

```
{
```

```
    函数体
```

```
}
```


对象的构造与析构

拷贝构造函数

```
class Point    //Point 类的定义
{
public:
    Point(int xx=0, int yy=0) { x = xx; y = yy; }    //构造函数，实际不允许写在类声明内部
    Point(const Point& p); //拷贝构造函数
    int getX() { return x; }
    int getY() { return y; }
private:
    int x, y; //私有数据
};

//成员函数的实现
Point::Point (const Point& p)
{
    x = p.x;
    y = p.y;
    cout << "Calling the copy constructor " << endl;
}
```

范：通过对象a的成员函数去操作对象b的数据成员，不合理，但向现实低头了。

对象的构造与析构

拷贝构造函数举例

- 举例1: 实参到形参, 等价于形参拷贝构造函数(实参)
- 举例2: return表达式到返回值, 等价于返回值拷贝构造函数 (return后的表达式)
- 举例3: 用对象a初始化对象b, 调用b的拷贝构造函数
- 举例4: 用对象a赋值给对象b, 不会调用拷贝构造函数, 调用的是operator=函数

对象的构造与析构

隐含的拷贝构造函数

- 如果程序员没有为类声明拷贝构造函数，则编译器自己生成一个**隐含的拷贝构造函数**。
- 这个构造函数执行的功能是：用作为初始值的对象的每个数据成员的值，初始化将要建立的对象的对应数据成员。（如果对象的数据成员也是对象类型，则继续深入这一过程。）
- 既然有隐含的拷贝构造函数，程序员还有必要自己编写拷贝构造函数吗？（有必要。因为有时候隐含的拷贝构造函数不能满足要求，如**浅拷贝与深拷贝**的问题。但上例中的拷贝构造函数其实没必要。）

对象的构造与析构

析构函数

- 完成对象被删除前的一些清理工作。
- 在对象的生存期结束的时刻**系统自动调用它**，然后再释放此对象所属的空间。
- 如果程序中未声明析构函数，编译器将自动产生一个**隐含析构函数**（它主要负责基类对象和成员对象的析构函数的调用）。
- 函数名：~类名
- 返回值：不能有！
- 参数：不能有！

对象的构造与析构

构造函数和析构函数举例

```
#include <iostream>
using namespace std;

class Point
{
public:
    Point(int xx,int yy);
    ~Point();
    //...其他函数原型
private:
    int x, y;
};
```

```
Point::Point(int xx,int yy)
{
    x = xx;
    y = yy;
}
```

```
Point::~~Point()
{
}
//...其他函数的实现略
```

析构函数名由“~”加类名构成，析构函数没有返回值和参数。

对象的构造与析构

程序实例

- 整型数组类
- 默认构造函数设定Length为10，元素初始化为0;
- 带参数的构造函数根据参数设定Length和所有元素初值（深拷贝）
- 拷贝构造函数调整Length和元素值
- operator= 实现深拷贝
- 析构函数销毁开辟的空间