面向对象程序设计(OOP)

C++(第3章：类和对象)

福州大学·软件学院·软件工程系
王灿辉(wangcanhui@fzu.edu.cn)

类和对象：概述

```
class <类名>{
  [private:]
    [<私有数据成员和成员函数>]
  protected:
    [<保护数据成员和成员函数>]
  public:
    [<公有数据成员和成员函数>]
} [对象列表];
```

第3章：类和对象

复习：结构的定义

```
#include <stdio.h>
#define PRINT(x) printf(#x"=%d\n",x)
struct Time {   //时
    int hour;   //时
    int minute; //分
    int second; //秒
};
int main() { //……
}
```

第3章：类和对象

复习：结构的定义

```
int main() {
   struct Time t;
   t.hour=t.minute=t.second=5;
   PRINT(t.hour);
   PRINT(t.minute);
   PRINT(t.second);
   return 0;
}                                    //完整的程序
```

第3章：类和对象

C++的结构和方法成员

```
#include <stdio.h>
#define PRINT(x,y,z)
    printf(#x"=%d, "#y"=%d, "#z"=%d\n", x,y,z)
//C++结构不仅可以包含数据，还可以包含函数。
struct Time {
  int hour,minute,second; //时,分,秒
  void init(int h=0,int m=0,int s=0) {
     hour=h;minute=m;second=s;
  }
};
```

第3章：类和对象

C++的结构和方法成员

```
int main() {
    Time t; //(C++中)Time前面不需要加struct
    t.hour=t.minute=t.second=5;
    PRINT(t.hour,t.minute,t.second);
    t.init();PRINT(t.hour,t.minute,t.second);
    t.init(11); //引用方法和数据成员一样
    PRINT(t.hour,t.minute,t.second);
    t.init(11,11);
    PRINT(t.hour,t.minute,t.second);
    t.init(11,11,11);
    PRINT(t.hour,t.minute,t.second);
}                                    //完整的程序
```

第3章：类和对象

C++结构(类)的访问控制

```
#include <iostream>
//(C++中)可以(对类中的数据、函数)进行访问控制
struct Time {
private:  //(私有)访问说明符
   int hour,minute,second; //时,分,秒
public:  //(公共)访问说明符
   void initialize(int h=0,int m=0,int s=0) {
     hour=h;minute=m;second=s;}
   void showTime() {
   std::cout<<hour<<':'<<minute<<':'<<second;}
};
```

第3章：类和对象

C++结构(类)的访问控制

```
int main() {
    Time t;
    //t.hour=t.minute=t.second=5;
    //不允许访问结构(类)的私有变量！
    t.showTime(); //没有初始化，输出随机值！
    t.initialize();          t.showTime();
    t.initialize(12);        t.showTime();
    t.initialize(12,30);     t.showTime();
    t.initialize(12,30,10);t.showTime();
    return 0;
}                                    //完整的程序
```

第3章：类和对象

C++新引入的类(class)

```
#include <iostream>
class Time {//C++的类(class)默认为私有(访问)，结构默认为公有(为了和C兼容)
    int hour,minute,second; //时,分,秒，私有
public://没有该限定符特默认为私有(访问权限)
    void initialize(int h=0,int m=0,int s=0) {
       hour=h;minute=m;second=s; }
    void showTime() {
    std::cout<<hour<<':'<<minute<<':'<<second;
    }
};
```

第3章：类和对象

C++新引入的类(class)

```
int main() {
    Time t;
    //t.hour=t.minute=t.second=5;
    //不允许访问结构(类)的私有变量！
    t.showTime(); //没有初始化，输出随机值！
    t.initialize();          t.showTime();
    t.initialize(12);        t.showTime();
    t.initialize(12,30);     t.showTime();
    t.initialize(12,30,10);t.showTime();
    return 0;
}                                    //完整的程序
```

第3章：类和对象

C++的数据类型

- 整型[1](short、int、long、long long)，实型(float、double、long double)，布尔型(bool)、字符型[1](char)，空类型(void)，枚举型(enum)、联合(共用体union)、结构(struct)、类(class)，指针(*)、引用(&)、数组([])。～好像都已讲完
- 注：[1]可以加signed,unsigned修饰符。

第3章：类和对象

类和对象概述

- 类是C++语言封装的基本单位，用来创建对象。
- 对象（Object）是类（Class）的实例。
- 类（Class）又称对象类是指一组具有相同属性和运算的对象的抽象，一组具有相同数据结构和相同操作的对象的集合。在一个类中，每个对象都是类的实例(instance)，它们都可以使用类中提供的函数。

第3章：类和对象

类和对象概述

- 类具有**属性**，用数据结构来描述类的属性，类具有**操作**，它是对象的行为的抽象，用操作名和实现该操作的方法（method），即操作实现的过程(函数)来描述。
- **非常简单的类可能只有操作或只含有数据**。
- 对象是指一个属性（数据）集及其操作（行为）的封装（encapsulation)体。

第3章：类和对象

类和对象概述

- 对象（Object)是系统中用来描述客观事物的一个实体，是构成系统的一个基本单位，由一组属性和对这些属性进行操作的一组服务构成。
- **对象的属性是指描述对象的数据，对象的属性值的集合成为对象的状态。对象的行为是定义在对象属性上的一组操作方法的集合。**

类的声明

```
class <类名>{ //默认私有(访问)
   [private:]   //可以多次出现
      [<私有数据成员和成员函数>]
   protected:   //可以多次出现
      [<保护数据成员和成员函数>]
   public:      //可以多次出现
      [<公有数据成员和成员函数>]
} [对象列表];
```

结构的声明

```
struct <类名>{ //默认公有(访问)
   private:   //可以多次出现
     [<私有数据成员和成员函数>]
   protected:  //可以多次出现
     [<保护数据成员和成员函数>]
   [public]:   //可以多次出现
     [<公有数据成员和成员函数>]
} [对象列表];
```

第3章：类和对象

概述：类的声明

- 声明类的一般形式如下：

```
class 〈类名〉{   //类名后不能加括号()!!
  [private:]     //可以多次出现
    [〈私有数据成员和成员函数〉]
  protected:     //可以多次出现
    [〈保护数据成员和成员函数〉]
  public:        //可以多次出现
    [〈公有数据成员和成员函数〉]//外部接口
} [对象列表];    //类定义后的;不能少！
```

第3章：类和对象

概述：类的声明实例

```
//运载工具类，只含公有数据，与结构类似！
class Vehicle {  //类名Vehicle
public:          //公有数据成员
   int p;              //载客数
   int f;              //油箱容积(g)
   int mpg;            //行驶里程(km)/g
};
```

第3章：类和对象

概述：声明实例变量和成员引用

```
Vehicle m; //创建类Vehicle的对象
//给对象的各个数据成员赋值
m.p = 7;
m.f = 16;
m.mpg = 21;
//计算最大可行驶里程
int range = m.f * m.mpg;
```

第3章：类和对象

概述：一个简单的实例

- 一个简单的实例

输出结果：

```
Minivan can carry 7 with a range of 336
```


- 创建两个对象的实例

输出结果：

```
Minivan can carry 7 with a range of 336
Sportscar can carry 2 with a range of 168
```

第3章：类和对象

概述：类的声明实例

```
//声明Vehicle类(增加一个成员函数)
class Vehicle {
public:
   int passengers;
   int fuelcap;
   int mpg;
   int range();//计算并返回最大可行驶里程
};
```

声明range()成员函数(方法)：与原型定义类似

第3章：类和对象

概述：类的声明实例

```
//成员函数range的实现
int Vehicle::range() {
    return mpg * fuelcap;
}
```


类的名称

::作用域解析运算符

实例变量不需要用类名限定

- 调用成员函数：

  对象名.成员函数名（……） //不是类名！

第3章：类和对象

概述：类的声明实例

```
void main() {
  Vehicle m; //创建对象(声明变量)
  //……
  cout << "Minivan can carry "
       << m.passengers
       << " with a range of "
       << m.range() << "\n";
}
```

第3章：类和对象

概述：类的声明实例

- 一个完整的实例(运载工具类)

输出结果：

```
Minivan can carry 7 with a range of 336

Sportscar can carry 2 with a range of 168
```

- 另一个完整的实例：Point(点)类

第3章：类和对象

概述：如何给对象赋初值？

- int x=10,y=20;

```
class Vehicle{
public:
   int passengers=4; //可以吗？
   //……
};
Vehicle m1,m2;
```

第3章：类和对象

概述：如何给对象赋初值？

```
class Vehicle {
public:
   int passengers;
   int fuelcap;
   int mpg;
};
Vehicle m1(7,16,21),m2(2,14,12);
```

构造函数和析构函数

- 默认构造函数为：

  类名 () {}

- 默认析构函数为：

  ~类名 () {}

- 类的数据成员一般定义为私有的，而函数成员一般定义为公有的。

第3章：类和对象

构造函数和析构函数

- 构造函数用于为由类定义的实例变量赋初始值，或者其他任务，格式如下：

  类名()**//没有返回值，可以有参数，也可重载**

  {//构造函数的代码}

- 析构函数**(一个类只能有一个)**用于执行对象被销毁时执行的一系列动作，格式如下：

  ~类名()　　**//没有参数也没有返回值！！**

  {//析构函数的代码}

- **构造函数和析构函数正常需定义为public**

第3章：类和对象

构造函数和析构函数

- 一个最简单的构造函数和析构函数实例
- 输出结果：


```
0 0
Destructing...
Destructing...
```

- 默认构造函数为：　　默认析构函数为：

```
类名() {}　　~类名() {}
```

第3章：类和对象

带参数的构造函数

- 一个带参数的构造函数实例
- 输出结果：

```
5 19 23
Destructing object whose x value is 23
Destructing object whose x value is 19
Destructing object whose x value is 5
```

- 对象析构的顺序与其创建时的顺序相反，即最后创建的对象最先被析构

第3章：类和对象

带参数的构造函数

- 利用类的构造函数进行对象的初始化：

  MyClass t1(5); //第一种方法(最常用)

  MyClass t3 = MyClass(23); //第二种方法

  MyClass t2 = 10;//第三种方法(仅当构造函数只有一个参数时才能用这种方法初始化)

- MyClass t4;该语句仅在下述情况下正确：

  1、没有构造函数时(使用默认构造函数)；

  2、定义有无参的构造函数时；

  3、构造函数允许所有参数全部为默认值时。

第3章：类和对象

构造函数和析构函数

- 带构造函数的Vehicle类
- 输出结果：

```
Minivan can carry 7 with a range of 336

Sportscar can carry 2 with a range of 168
```

- 带构造函数的Point(点)类

第3章：类和对象

完整的一个类的定义

- 对类Vehicle进行改写：

  1、构造函数、析构函数、和range（）都在类的内部定义（内联函数，inline）

  2、将类的三个字段全部设置为私有变量

  3、添加存取函数用于获得三个私有变量的值

- 类的数据成员一般定义为私有的，而函数成员一般定义为公有的。

第3章：类和对象

完整的一个类的定义

- 完整的Vehicle类
- 输出结果：

```
Minivan can carry 7 with a range of 336
Sportscar can carry 2 with a range of 168
```

- 在类内部定义的成员函数（一般应为简单函数）自动成为内联函数，并且函数定义前不需要使用inline关键字

第3章：类和对象

又一个类的定义实例

- 时钟类　　输出结果为：4230561:0:4230351
  10:20:30
- 修改后的时钟类
  输出结果：
  First time set(default) and output:
  0:0:0
  Second time set and output:
  10:20:30

第3章：类和对象

又一个类的定义实例

- 修改后带构造函数的时钟类

输出结果：


```
First time(Constructor) output:
0:0:0
Second time set and output:
10:20:30
Three time set(default) and output:
11:30:0
```

对象指针(类和对象)

- 可以说明对象的指针。例如：

```
Myclass obj;//改为:Myclass obj();错误
Myclass *pObj=&obj;
```


- 也可以利用对象指针访问其成员，例如：

```
pObj->data  //等价于：obj.data
//也等价于：(*pObj).data，但括号不能少
pObj->func(10); //执行成员函数
```

this指针(类和对象)

- 每次调用成员函数时，系统都会将一个指针this自动传递给它，这个this指针指向调用此函数的对象。
- this指针是所有成员函数的隐含参数。因此在成员函数内部可以直接使用this指针引用调用对象。
- this指针使用实例

第3章：类和对象

this关键字

- 如：设有类Example，其定义如下：

```
class Example {
    int m;
public:
    void setvalue(int arg1) {m=arg1;}
};
```

- 说明Example s;调用s.setvalue(100);

第3章：类和对象

this关键字

- 实际上成员函数的定义是：

```
void setvalue(Example *this,int arg1)
   {this->m=arg1;}
```


- s.setvalue(100);实际上相当于：

```
s.setvalue(&s,100);
```

成员函数setvalue实际上得到了对象s的地址，并将其传递给该函数的隐含指针变量this，函数中通过this指针给对象s的数据成员m赋值。

- 类中成员函数有了隐含指针变量this后，就可保证用不同的对象调用成员函数是对不同对象的操作。

第3章：类和对象

this关键字

- 一个常规的成员函数声明描述了三件在逻辑上相互不同的事情：
  1、该函数能访问类声明的私有成员
  2、该函数位于类的作用域之中
  3、该函数必须经由一个对象去激活(有一个this指针)
- 后面要介绍的类的静态成员函数(static)只具有前两个性质，类的友元(friend)函数只具有第一个性质。

本部分内容讲授到此结束！

福州大学·软件学院·软件工程系
王灿辉(wangcanhui@fzu.edu.cn)