C++语言程序设计

第二章: 类和对象

宋霜

哈尔滨工业大学（深圳）

机电工程与自动化学院

邮箱: songshuang@hit.edu.cn

第二章：类和对象

引子

- 整数 （int）
  - 属性
  - 操作

第二章：类和对象

引子

- 整数 （int）
  - 属性
  - 操作
- 复数 （complex）
  - 属性
  - 操作

第二章：类和对象

面向对象的程序设计

- 复杂的事物总是由许多部分组成
- 研究各部分之间的功能与联系

第二章：类和对象

面向对象的程序设计

- 复杂的事物总是由许多部分组成
- 研究各部分之间的功能与联系

对象

- 万物皆对象
- 静态特征：属性
- 动态特征：行为
- 外界交互：消息

第二章：类和对象

面向对象的程序设计

- 确定系统由哪些对象组成
- 确定每个对象的属性与行为
- 研究对象之间的联系

第二章：类和对象

面向对象的程序设计

- 确定系统由哪些对象组成
- 确定每个对象的属性与行为
- 研究对象之间的联系

C++的对象

- 数据
- 函数
- 消息

第二章：类和对象

面向对象

- 抽象
- 封装
- 继承
- 多态

第二章：类和对象

面向对象

- 抽象
  - 抽象的作用是表示同一类事物的本质
  - 类是对象的抽象，对象是类的具体表现形式
- 封装
- 继承
- 多态

第二章：类和对象

面向对象

- 抽象
- 封装
  - 把客观事物封装成抽象的类，并且把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。
  - 封装是面向对象的特征之一，是对象和类概念的主要特性。
  - 一个类就是一个封装了数据以及操作这些数据的代码的逻辑实体
- 继承
- 多态

第二章：类和对象

面向对象

- 抽象
- 封装
- 继承
  - 某个类型的对象获得另一个类型的对象的属性的方法
  - 可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展
- 多态

第二章：类和对象

面向对象

- 抽象
- 封装
- 继承
- 多态
  - 一个类实例的相同方法在不同情形有不同表现形式。
  - 多态机制使具有不同内部结构的对象可以共享相同的外部接口

第二章：类和对象

面向对象

- 面向过程：
  - 程序=数据结构+算法
- 面向对象：
  - 对象=数据结构+算法
  - 程序=（对象+对象+...）+消息

第二章：类和对象

面向对象的软件开发

- 面向对象分析
- 面向对象设计
- 面向对象编程
- 面向对象测试
- 面向对象维护

第二章：类和对象

面向对象的软件开发

- 面向对象分析
- 面向对象设计
- 面向对象编程
- 面向对象测试
- 面向对象维护

第二章：类和对象

类的声明与对象的定义

- 类是对象的抽象
- 对象是类的具体实例（instance）
- 对象的类型称为类

第二章：类和对象

类的声明与对象的定义

- 类是对象的抽象
- 对象是类的具体实例（instance）
- 对象的类型称为类

```
struct Student
{
    int num;
    char name;
    char sex;
};
Student st1,st2;
```

第二章：类和对象

类的声明与对象的定义

- 类是对象的抽象
- 对象是类的具体实例（instance）
- 对象的类型称为类

```
struct Student
{
    int num;
    char name;
    char sex;
};
Student st1,st2;
```

```
class Student
{
    int num;
    char name;
    char sex;
};
Student st1,st2;
```

第二章：类和对象

类的声明与对象的定义

- 类是对象的抽象
- 对象是类的具体实例（instance）
- 对象的类型称为类
- 结构体的成员默认为公用（public）
- 类的成员默认为私有（private）：封装与信息隐藏

类的声明与对象的定义

```
class Student_CPP
{
public:
    void set_stu(int stu_num, char stu_name, char stu_sex)
    {
        num=stu_num;
        name=stu_name;
        sex=stu_sex;
    }
    void display_stu()
    {
        cout<<num<<" "<<name<<" "<<sex<<endl;
    }
private:
    int num;
    char name;
    char sex;
};
Student_CPP st2;
```

类的声明与对象的定义

```
class Student_CPP
{
public:
    void set_stu(int stu_num, char stu_name, char stu_sex)
    {
        num=stu_num;
        name=stu_name;
        sex=stu_sex;
    }
    void display_stu()
    {
        cout<<num<<" "<<name<<" "<<sex<<endl;
    }
private:
    int num;
    char name;
    char sex;
};
Student_CPP st2;
```

成员函数

成员变量

类的声明与对象的定义

```
st2.set_stu(2,'b','F');
st2.display_stu();
```

类的声明与对象的定义

```
st2.set_stu(2,'b','F');
st2.display_stu();
```

```
Student_CPP *st3=&st2;
st3->display_stu();
(*st3).display_stu();
```

类的成员

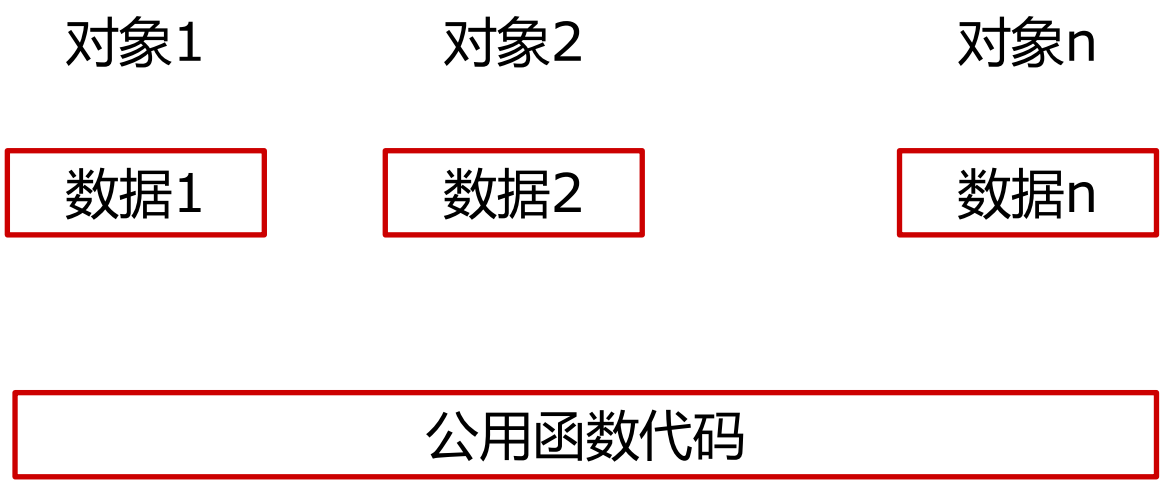
```
class Student_CPP
{
public:
    void set_stu(int stu_num, char stu_name, char stu_sex)
    {
        num=stu_num;
        name=stu_name;
        sex=stu_sex;
    }
    void display_stu()
    {
        cout<<num<<" "<<name<<" "<<sex<<endl;
    }
private:
    int num;
    char name;
    char sex;
};
Student_CPP st2;
```

成员函数

成员变量

类的成员

同一类的不同对象分配变量空间，共享函数代码空间。



第二章：类和对象

类的成员

- 属于类的成员
- private，public，protected
- 希望被使用者调用的函数 public：对外接口

第二章：类和对象

类的成员

- 属于类的成员
- private，public，protected
- 希望被使用者调用的函数 public：对外接口
- 使用者并不关心接口的实现方法，而是提供了何种接口

类的成员

- 属于类的成员
- private，public，protected
- 希望被使用者调用的函数 public：对外接口
- 使用者并不关心接口的实现方法，而是提供了何种接口

```
public:
    void set_stu(int stu_num, char stu_name, char stu_sex)
    {
        num=stu_num;
        name=stu_name;
        sex=stu_sex;
    }
    void display_stu()
    {
        cout<<num<<" "<<name<<" "<<sex<<endl;
    }
```

第二章：类和对象

类的成员

- 属于类的成员
- private，public，protected
- 希望被使用者调用的函数 public：对外接口
- 使用者并不关心接口的实现方法，而是提供了何种接口
- 接口与实现分离

类的成员

- 接口与实现分离

```
class Student_CPP
{
public:
    void set_stu(int stu_num, char stu_name, char stu_sex);
    void display_stu();
private:
    int num;
    char name;
    char sex;
};

void Student_CPP::set_stu(int stu_num, char stu_name, char stu_sex)
{
    num=stu_num;
    name=stu_name;
    sex=stu_sex;
}
void Student_CPP::display_stu()
{
    cout<<num<<" "<<name<<" "<<sex<<endl;
}
```

类的成员

- 接口与实现分离
- Student.h：接口
- Student.cpp：实现
- Main.h：使用者