C++语言程序设计
第二章 类和对象

王焦乐

http://faculty.hitsz.edu.cn/jlwang

哈尔滨工业大学（深圳）

机电工程与自动化学院

C++语言程序设计...

群号：599477959

扫一扫二维码，加入群聊。

QQ

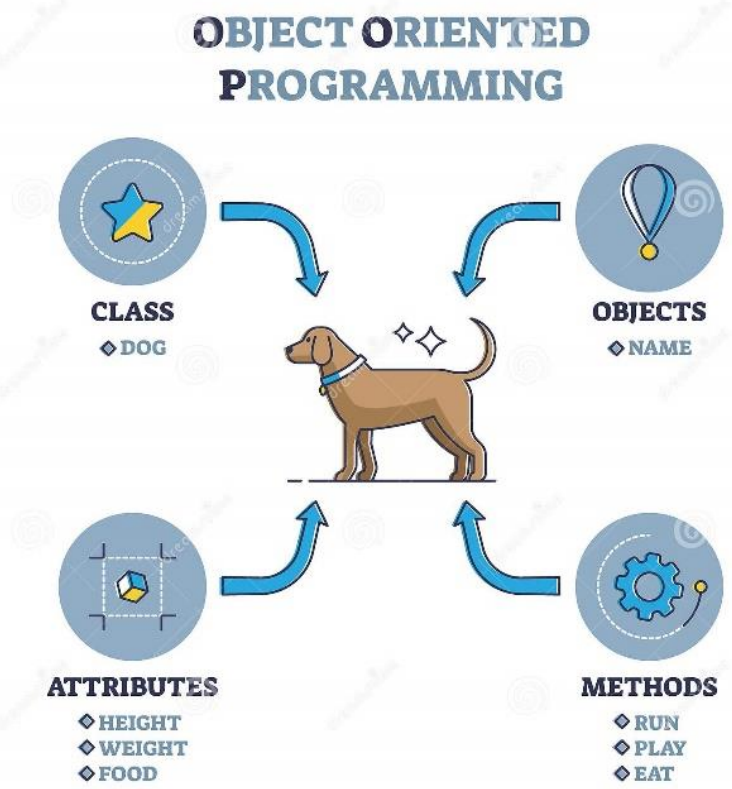
本章主要内容

- 面向对象程序设计方法概述
- 类的申明和对象的定义
- 类的成员函数与对象成员
- 类的封装性和信息隐蔽

0、面向对象程序设计方法概述

□ 对象 Object

- 万物皆对象
- 属性 attribute：静态特征
  - 班级所属系、专业、教室
- 行为 behavior：动态特征
  - 学习、开会、体育比赛
- 消息 message：外界交互
  - 听到打铃就下课
- 一个对象：一组属性、一组行为

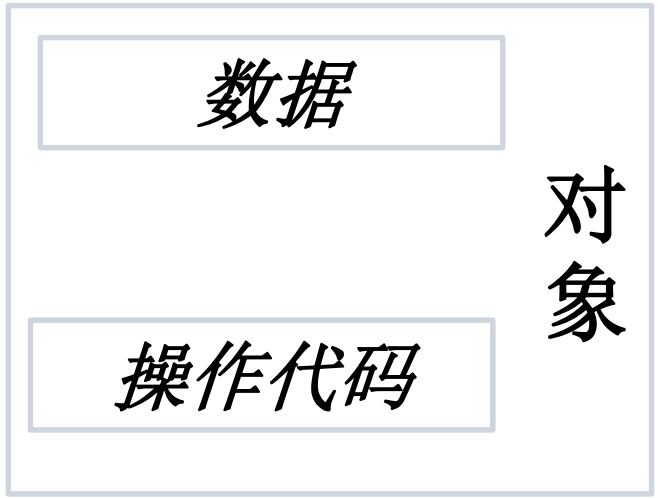


- 整数 （int）
  - 属性
  - 操作
- 复数 （complex）
  - 属性
  - 操作

0、面向对象程序设计方法概述

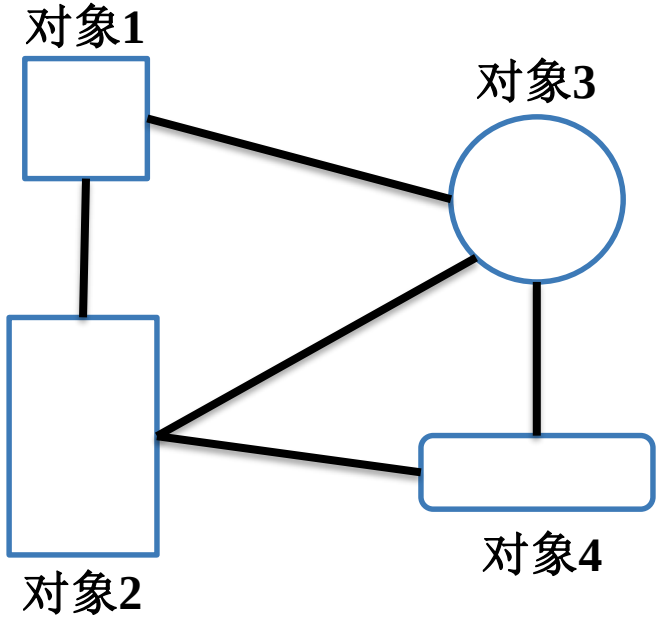
面向对象的程序设计

- 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 → 属性 attribute
- 函数 → 行为 behavior
- 函数调用 → 消息 message



0、面向对象程序设计方法概述

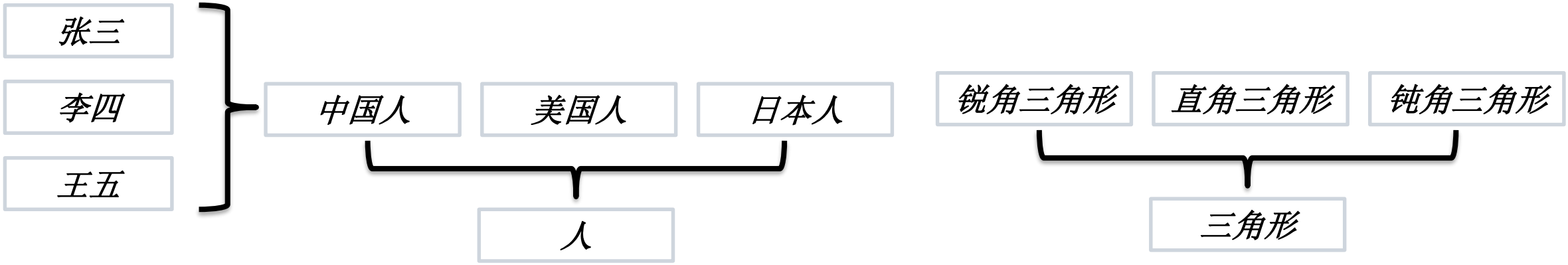
面向对象的几个概念

- 抽象（abstraction）
- 封装（encapsulation）与信息隐蔽（information hiding）
- 继承（inheritance）与重用（software reusability）
- 多态性（polymorphism）

0、面向对象程序设计方法概述

面向对象的几个概念

- 抽象（abstraction）
  - 抽象的作用：表示同一事物的本质
  - 类（class）是对象的抽象
  - 对象是类的特例、类的具体表现形式（instantiation, instance）

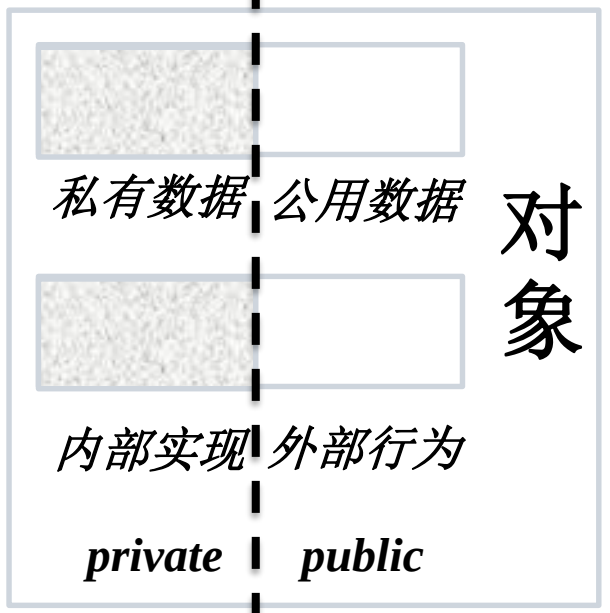


- 封装（encapsulation）与信息隐蔽（information hiding）
- 继承（inheritance）与重用（software reusability）
- 多态性（polymorphism）

0、面向对象程序设计方法概述

面向对象的几个概念

- 抽象（abstraction）
- 封装（encapsulation）与信息隐蔽（information hiding）
  - 封装是面向对象的特征之一，是对象和类概念的主要特性。
  - 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。
  - 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
  - 数据与操作代码封装于一个对象，对象间相互独立
  - 隐藏对象内部细节，只留接口接收外部消息

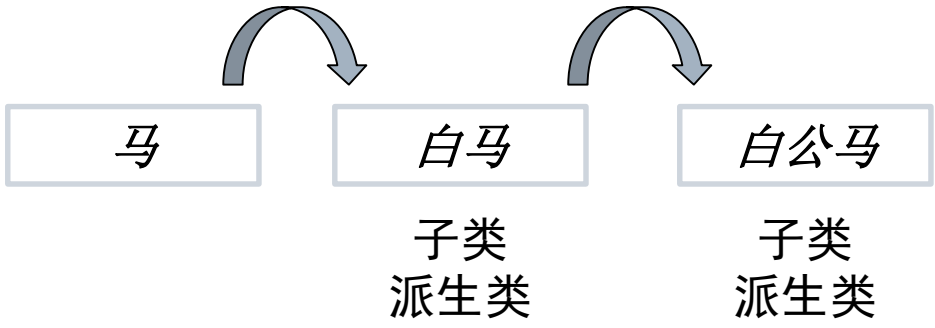


- 继承（inheritance）与重用（software reusability）
- 多态性（polymorphism）

0、面向对象程序设计方法概述

□ 面向对象的几个概念

- 抽象（abstraction）
- 封装（encapsulation）与信息隐蔽（information hiding）
- 继承（inheritance）与重用（software reusability）
  - 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
  - 利用已有的类，建立一个新类，重用已有软件中的一部分甚至大部分，节省了编程工作量（软件重用）



- 多态性（polymorphism）

0、面向对象程序设计方法概述

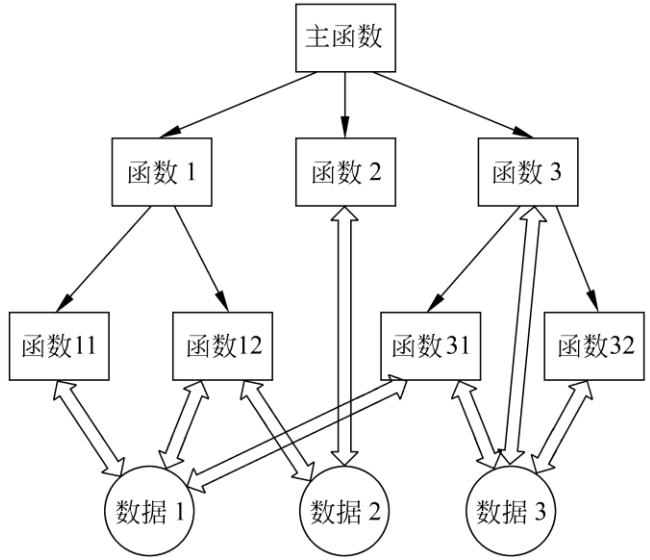
□ 面向对象的几个概念

- 抽象（abstraction）
- 封装（encapsulation）与信息隐蔽（information hiding）
- 继承（inheritance）与重用（software reusability）
- 多态性（polymorphism）
  - 多态现象：甲乙丙三个高二学生，同时听到上课铃，走入三个不同教室
  - 多态性：由继承而产生的相关的不同的类，其对象对同一消息会作出不同的响应。
  - 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

0、面向对象程序设计方法概述

面向对象的程序设计的特点

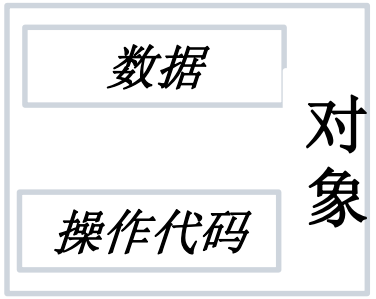
- 面向过程：
  - 程序=数据结构+算法

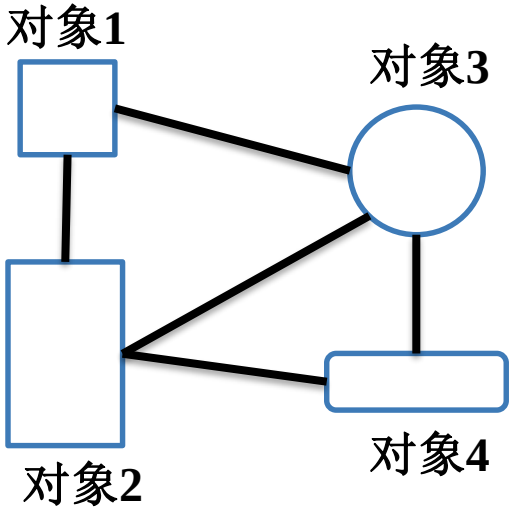


- 面向对象：
  - 对象=数据结构+算法
  - 程序=（对象+对象+...）+消息
  - 面向对象程序设计包括两个方面：

①设计所需的各种类和对象，即决定把哪些数据和操作封装在一起。

②考虑怎样向对象发送消息（调用对象的成员函数）。





0、面向对象程序设计方法概述

□ 类和对象的作用

- 类是所有面向对象的语言具有的共同特征
- 不包含类的计算机语言不能称为面向对象的语言
- C++早期被称为“带类的C”
- 类是C++的灵魂，不掌握类就不能真正掌握C++

- C++兼容三种程序设计：
  - 面向过程
  - 基于对象：类和对象的概念、类的机制和声明、类对象的定义与使用
  - 面向对象：继承机制和多态

0、面向对象程序设计方法概述

面向对象的软件开发

- 面向对象分析（OOA）
  - 对任务的分析中，从客观事物和事物之间的关系归纳出有关对象（对象的属性和行为）以及对象之间的联系。并将具有相同属性和行为的对象用一个类来表示。
- 面向对象设计（OOD）
- 面向对象编程（OOP）
- 面向对象测试（OOT）
- 面向对象维护（OOSM）

0、面向对象程序设计方法概述

面向对象的软件开发

- 面向对象分析（OOA）
- 面向对象设计（OOD）
  - 首先是进行类的设计，类的设计可能包含多个层次（利用继承和派生机制）。
  - 然后以这些类为基础提出程序设计的思路和方法，包括了算法的设计。
  - 在此设计阶段，并不牵涉某一具体的计算机语言。
- 面向对象编程（OOP）
- 面向对象测试（OOT）
- 面向对象维护（OOSM）

0、面向对象程序设计方法概述

面向对象的软件开发

- 面向对象分析（OOA）
- 面向对象设计（OOD）
- 面向对象编程（OOP）
  - 根据面向对象设计的结果，用一种计算机语言把它写成程序。
  - C++、Dephi、VB、Java
- 面向对象测试（OOT）
- 面向对象维护（OOSM）

0、面向对象程序设计方法概述

面向对象的软件开发

- 面向对象分析（OOA）
- 面向对象设计（OOD）
- 面向对象编程（OOP）
- 面向对象测试（OOT）
  - 交付用户使用前，必须对程序进行严格的调试，如果发现错误，要及时改正。
  - 面向对象测试，是以类作为测试的基本单元用面向对象的方法实施测试。
- 面向对象维护（OOSM）

0、面向对象程序设计方法概述

面向对象的软件开发

- 面向对象分析（OOA）
- 面向对象设计（OOD）
- 面向对象编程（OOP）
- 面向对象测试（OOT）
- 面向对象维护（OOSM）
  - 采用了面向对象的方法，方便了维护程序。因为类的封装性，修改一个类对其他类（非子类）影响很小，极大提高了程序维护的效率。

1、类的申明和对象的定义

类和对象的关系

- 对象的类型称为类
- 类（class）是对象的抽象（abstraction），不占用内存，无法直接存储数据或执行操作
- 对象是类的具体实例（instance），占用内存，可以存储数据，进行操作

- 结构体类型（struct）→ 结构体变量
- 类 → 对象
- 需要声明类的类型

1、类的申明和对象的定义

声明类的类型

- 类头：class 类名
- 类体：类的成员表（class member list）
- class默认private，struct默认public

```
class 类名
{
  private:
    私有的数据和成员函数;
  public:
    公用的数据和成员函数;
  protected:
    受保护的数据和成员函数;
};
```

```
struct Student
{
  int num;
  char name[20];
  char sex;
};
Student stud1, stud2;
```

定义对象 →

```
class Student
{
  int num;
  char name[20];
  char sex;
};
Student stud1, stud2;
```

类头 class head

类体 class body

```
class Student
{
  private:
    int num;
    char name[20];
    char sex;
  public:
    void display()
    {
      cout<<"num:"<<num<<endl;
      cout<<"name:"<<name<<endl;
      cout<<"sex:"<<sex<<endl;
    }
};
Student stud1, stud2;
```

成员访问限定符 Member access specifier

1、类的申明和对象的定义

- 声明类的类型
  - 类头：class 类名
  - 类体：类的成员表（class member list）
  - class默认private，struct默认public
- **private**：只能被本类中的成员函数访问，类外不能访问。
- **public**：公有成员可以被本类的成员函数访问，也能在类的作用域范围内的其他函数访问。
- **protected**：受保护成员可由本类的成员函数访问，也能由派生类的成员函数访问。
- 定义类时，这三类成员不分前后顺序，也可以重复出现。一般推荐最多出现一次。

1、类的申明和对象的定义

□ 对象的定义

1. 先声明类类型，然后再定义对象

   在声明类类型后，像定义变量一样定义对象。

   （1） class 类名 对象名表　　例： class student st1, st2;

   （2）类名 对象名表　　例： student st1, st2;

2. 在声明类类型的同时定义对象
3. 不出现类名直接定义对象
4. 在面向对象程序设计和C++程序中，类的声明和类的使用是分开的，类并不只为一个程序服务，人们常把一些常用的功能封装成类，并放在类库中。一般采用第一种方法。
5. 在定义对象后，编译程序在编译时会为对象分配内存空间，存放对象的成员。

1、类的申明和对象的定义

类和结构体的异同

- C++允许用struct定义一个类类型（兼容C，让C程序不用修改就能在C++环境中使用）
- 用class声明的类如果不带成员访问限定符，所有成员默认限定为private；
- 用struct声明的类如果不带成员访问限定符，所有成员默认限定为public

2、类的成员函数

成员函数的性质

- 返回值、函数类型
- 属于类的成员、出现在类体中
- 指定为：private、public、protected
- private：只能被类中其他成员函数调用

  工具函数（utility function）
- public：类的对外接口

```
class Student
{
  int num;
  char name[20];
  char sex;
  void display()
  {
    cout<<"num:"<<num<<endl;
    cout<<"name:"<<name<<endl;
    cout<<"sex:"<<sex<<endl;
  }
};
Student stud1, stud2;
```

2、类的成员函数

在类外定义成员函数

- 必须现在类体中作原型声明

类型　函数名（形参表）；

类型　类名::函数名（形参表）

{

成员声明;

}

- 实现接口和实现细节的分离

- “::”作用域限定符
  - Student::display()
  - ::display()
  - display()

```
class Student
{
  public:
    void display();        //公用成员函数原型声明
  private:
    int num;
    string name;
    char sex;
};

void Student::display()  //在类外定义display类函数
{
  cout<<"num:"<<num<<endl;
  cout<<"name:"<<name<<endl;
  cout<<“sex:"<<sex<<endl;
}
```

2、类的成员函数

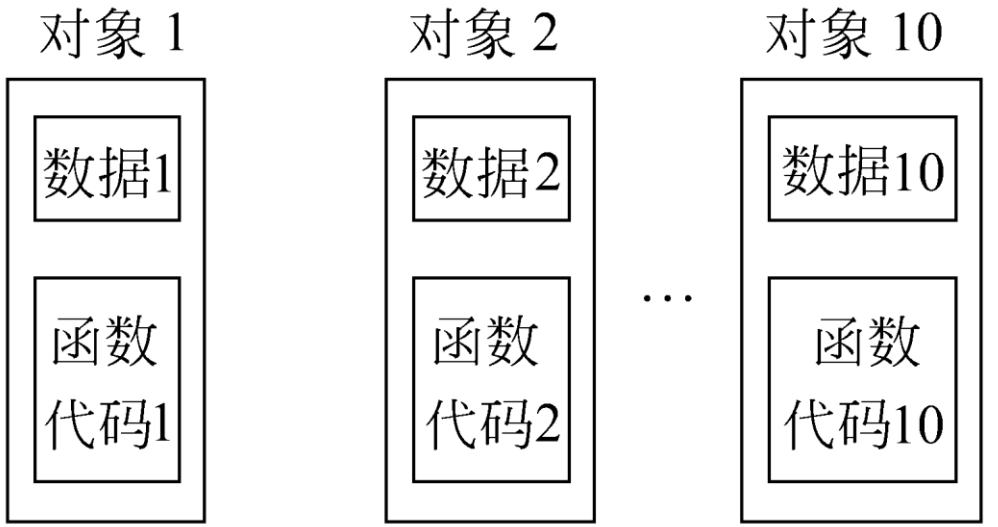
内置成员函数（inline成员函数）

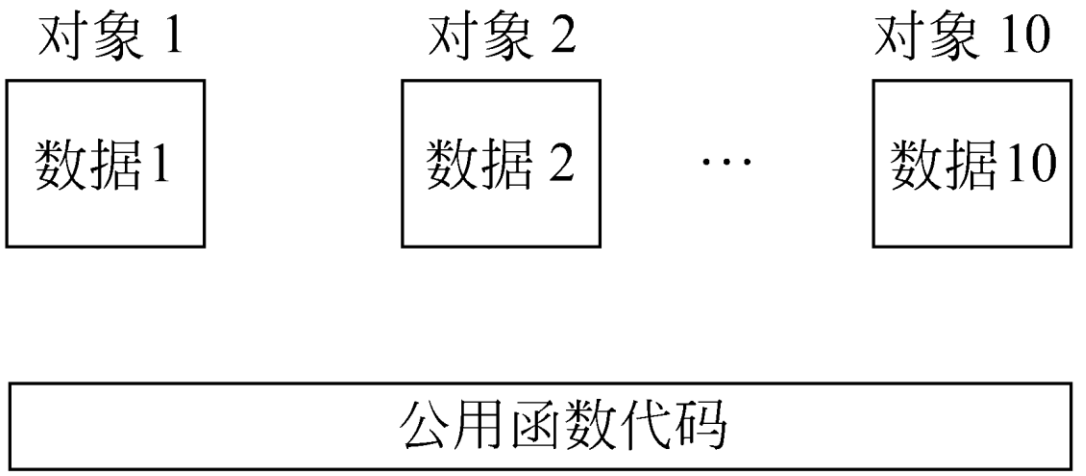
- 编译时进行置换（代码拷贝）
- 类体中定义的成员函数，不包含循环等控制结构，自动默认为inline
- 类体外定义需要作显示声明
- 类体外定义inline函数：
  - 声明与定义在同一文件中（头文件或源文件）
  - 提高效率，适用于规模小、使用频率高的函数

```
class Student
{
  public:
    //声明此成员函数为内置函数
    inline void display();
  private:
    int num;
    string name;
    char sex;
};
//在类外定义display函数为内置函数
inline void Student::display()
{
  cout<<"num:"<<num<<endl;
  cout<<"name:"<<name<<endl;
  cout<<"sex:"<<sex<<endl;
}
```

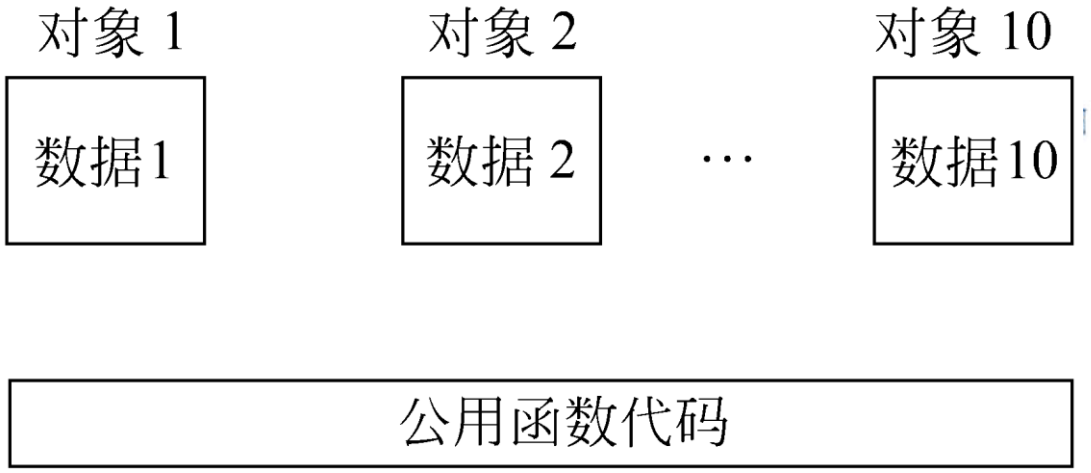
3、成员函数的存储方式

- 对象不同，数据不同，函数相同
- 从逻辑的角度：
  - 每个对象封装了数据和函数，只允许该对象的成员函数访问私有数据
- 从物理的角度：
  - 每个对象只有数据占用内存，对应同一个函数代码段
- this指针
  - this指向stud1，之后stud2





3、成员函数的存储方式



- 对象不同，数据不同，函数相同
- 从逻辑的角度：
  - 每个对象封装了数据和函数，只允许该对象的成员函数访问私有数据
- 从物理的角度：
  - 每个对象只有数据占用内存，对应同一个函数代码段
- this指针
  - this指向stud1，之后stud2

```
// 如果声明一个类：
class Time
{
  public:
    int hour;
    int minute;
    int sec;
    void set()
    {cin >> hour >> minute >> sec ;}
};
// 可以用下面的语句计算该类对象占用的字节数
cout<< sizeof(Time) <<endl;
// 结果输出值是12
// 一个对象占用的空间是其数据成员占据的内存空间。
```

4、访问对象的成员

- 通过对象名和成员运算符

格式：**对象名.成员名**

例：stud1.display();　　**// 调用成员函数**

　　display();　　**// 调用普通函数**

注意：只有成员函数可以访问类中的所有成员，而在类外只能访问公有成员。

如果在类外面用下面的语句是错误的：

　　stud1.num = 10101

4、访问对象的成员

通过指向对象的指针

- p->hour表示p当前指向对象t中的成员hour
- (*p).hour也代表对象t中的成员hour
- p->hour、(*p).hour、t.hour三种表示是一个意思。

```
class Time
{
  public:
    int hour;
    int minute;
};
Time t, *p;
p= &t;

cout << p->hour << endl;
cout << (*p).hour << endl;
cout << t.hour << endl;
```

4、访问对象的成员

□ 通过对象的引用

对象A定义一个引用B，B是对象A的别名，A和B都是一个对象，所以完全可以通过引用访问对象中的成员。

```
Time  t1;
Time & t2=t1;
cout << t2.hour << endl;
cout << t1.hour << endl;
```

这里t2是t1的别名，所以访问t2.hour就是访问t1.hour。

5、类和对象的简单应用举例

- 类外必指定对象
- 不要把对象名写成类名
- 不赋初值的成员变量不可预知

```
#include <iostream>
using namespace std;
class Time
 {
   public:
    int hour;
    int minute;
    int sec;
 };

int main()
 {
   Time t1;
   Time &t2=t1;
   cin>>t2.hour;
   cin>>t2.minute;
   cin>>t1.sec;
   cout<<t1.hour<<":"<<t1.minute<<":"<<t2.sec<<endl;
 }
```

6、类的封装和信息隐蔽

公用接口与私有实现的分离

- 类的功能实现：通过成员函数对数据成员进行操作
- 公用接口（public interface）：公用成员函数
- 私有实现（private implementation）：类的功能实现细节对用户隐蔽
- 信息隐蔽（information hiding）：公用接口与私有实现的分离

```
class Student
{
  private:
    int num;
    string name;
    int age;    //新增
    char sex;
  public:
    void display()
    {
      cout<<"num:"<<num<<endl;
      cout<<"name:"<<name<<endl;
      cout<<"age:"<<age<<endl;  //新增
      cout<<"sex:"<<sex<<endl;
    }
};
Student stud;
```

6、类的封装和信息隐蔽

□ 公用接口与私有实现的分离

■信息隐蔽（information hiding）

1. 如果想修改或扩充类的功能，只需修改类中有关的数据成员和成员函数，类外的部分不用修改。
2. 当接口与实现（对数据的操作）分离时，只要类的接口没有改变，对私有实现的修改不会影响程序的其他部分。
3. 如果在编译时发现类中的数据读写有错，不必检查整个程序，只需检查本类中访问这些数据的成员函数。

```
class Student
{
  private:
    int num;
    string name;
    int age;     //新增
    char sex;
  public:
    void display()
    {
      cout<<"num:"<<num<<endl;
      cout<<"name:"<<name<<endl;
      cout<<"age:"<<age<<endl;  //新增
      cout<<"sex:"<<sex<<endl;
    }
};
Student stud;
```

6、类的封装和信息隐蔽

□ 类声明和成员函数定义的分离

- C++程序包含3各部分：

类声明头文件（后缀为.h或无后缀）

　　用户使用类库的有效方法和公用接口

类实现文件（后缀为.cpp），包括成员函数的定义

类的使用文件（后缀为.cpp），即主文件

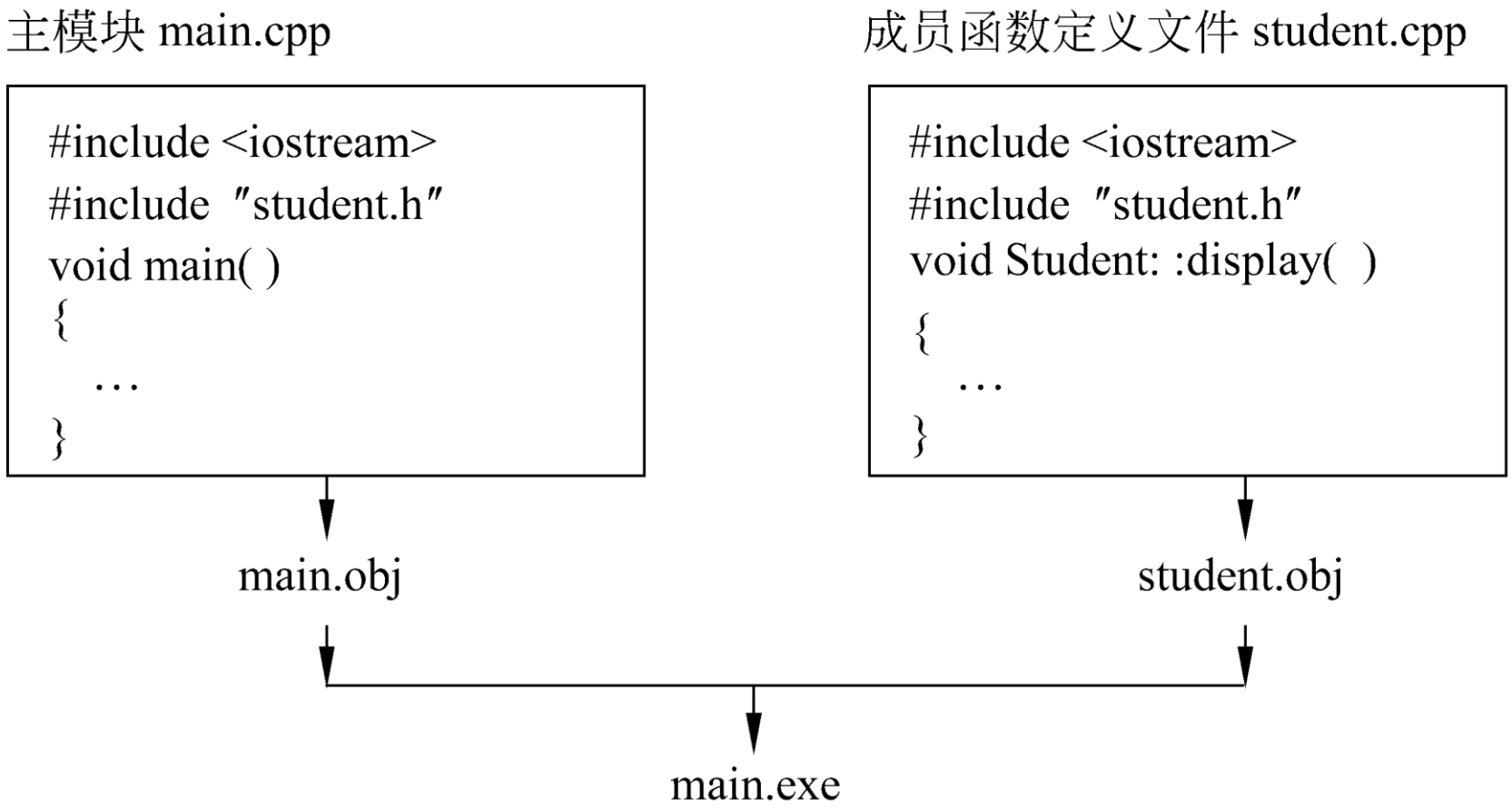
```cpp
// main.cpp 主文件
#include <iostream>
#include "student.h"
using namespace std;
int main()
{
  Student stud;
  stud.setdata();
  stud.display();
  return 0;
}
```

```cpp
// student.h 类声明头文件
class Student
{
  private:
    int num;
    string name;
    char sex;
  public:
    void display();
};
```

```cpp
// student.cpp 类实现文件
#include <iostream>
#include "student.h"
using namespace std;
void Student::display()
{
  cout<<"num:"<<num<<endl;
  cout<<"name:"<<name<<endl;
  cout<<"sex:"<<sex<<endl;
}
```

6、类的封装和信息隐蔽

- 类库：①类声明头文件；②已编译的成员函数的定义
- 类声明头文件是用户使用类库的有效方法和公用接口



- **面向对象程序设计中的几个名词**
- 类的成员函数在面向对象程序理论中又称为方法，方法是指对数据的操作。
- 只有被声明为公有的方法（成员函数）才能被对象外界所激活。
- 外界是通过发消息激活有关的方法。所谓消息其实就是一条命令，由程序语句实现。如
- st1.display();
- 是向对象st1发出一个消息，让它执行display方法，这里，st1是**对象**，display()是**方法**，语句st1.display();是**消息**。