面向对象程序设计

Object Oriented Programming

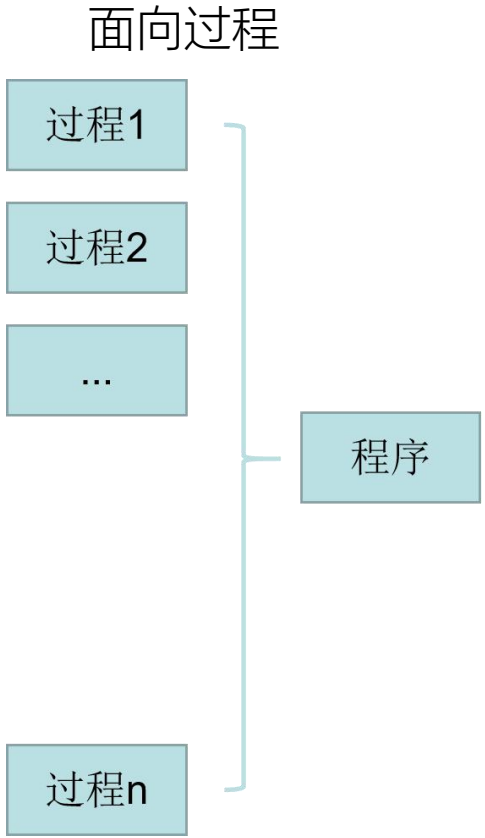
主 讲 教 师：陈洪刚
开 课 时 间：2021年
honggang_chen@yeah. net

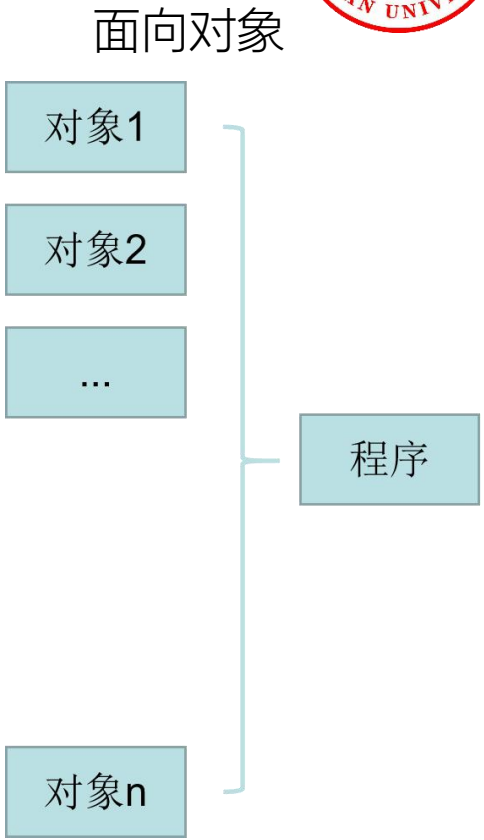
CHAPTER 2

CONTENTS

2.1 面向对象程序设计方法概述

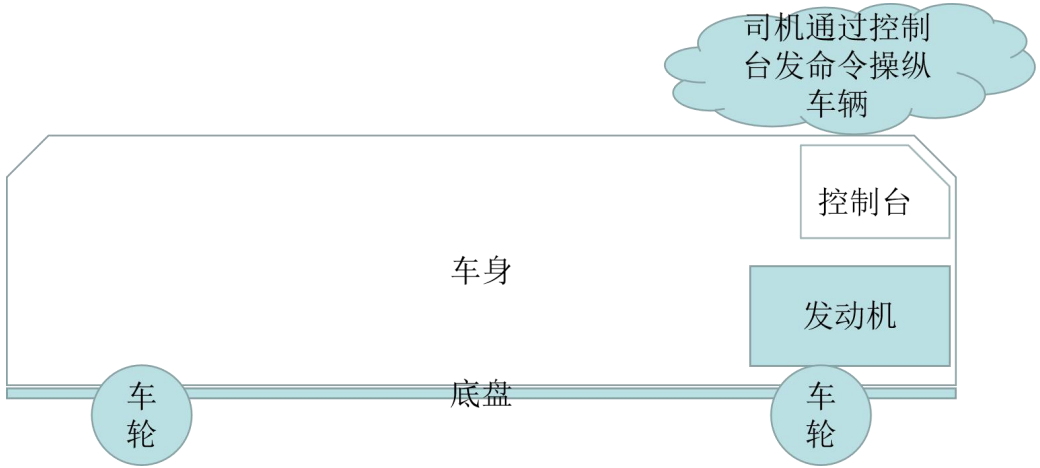
- 对于规模比较小的程序，程序员可以直接编写出一个面向过程的程序，能详细地描述每一瞬间的数据结构和对其操作的过程。
- 但是当程序规模越来越大，面向过程的程序语言就显得力不从心，面向对象程序设计语言就应运而生，C++ 就是其中的一个成员。





2.1.1 什么是面向对象的程序设计

- 面向对象的程序设计的思路与人们日常生活中处理问题的思路是相似的。一个复杂的事物总是由许多部分(对象)组成的。例如：汽车



- 对象：客观世界中任何一个事物都可以看成一个对象。对象可以是自然物体，也可以是社会中的一种组织结构（班级、系、学校）甚至一个图形、一项计划等都可以看成对象。复杂的对象由简单的对象组成。对象是构成系统的基本单位。**任何一个对象都具有静态和动态的特征。**

2.1.1 对象

班级对象

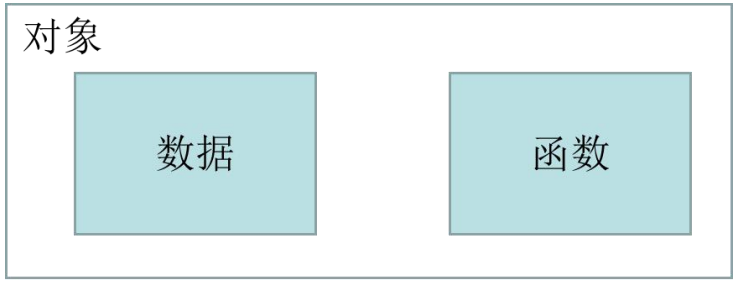
静态特征(属性)：	动态特征(行为)：
系: string department 专业: string major 学生人数: int number 所在教室: string classroom	学习: study() 开会: party() 体育比赛: match()

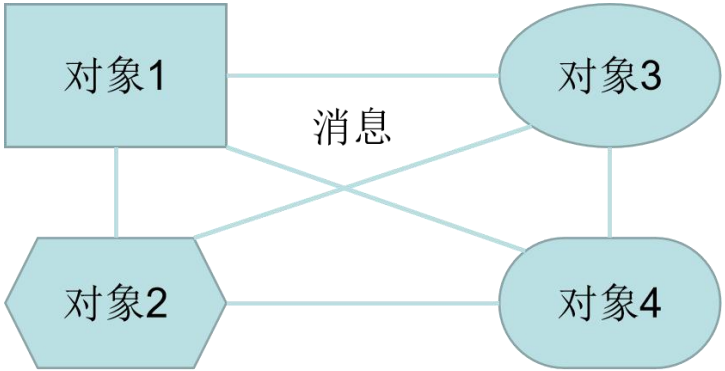
电视机对象

静态特征(属性)：	动态特征(行为)：
生产厂家: string sccj 品牌: string pp 屏幕尺寸: int width, height 隐含属性： int bOpen, nChannel, Volume, nColor;	开机: open() 关机: close() 选择频道: selectchannel() 调节音量: adjustvolume() 调节彩色: adjustcolor()

- 班级这个对象具有从属一个系和专业、学生人数、组建时间等静态特征，它还具有出操、学习、参加运动会等动态特征。静态特征称为属性，动态特征称为行为，外界给对象发出的信息一般称作消息。
- 一个对象往往包含一组属性和一组行为。如电视机包括生产厂家、品牌、屏幕尺寸等是静态特征，开机、关机、选择频道、调节音量、调节彩色等是动态特征。

2.1.1 对象





- 面向对象的程序设计在设计一个系统时，首先要确定系统中包括哪些对象，要分别设计这些对象。
- 在C++ 中，每个对象由数据和函数（操作代码）两部分组成。数据代表了属性，函数是对数据操作的代码，代表了行为。
- 例如三条边长是三角形的属性，利用三条边长计算三角形面积、输出计算结果，实现这些操作的代码就是对象的行为，在程序设计中又称方法。调用对象中的函数就是向对象传递一个消息，要求对象执行某个操作。

2.1.1 对象

三角形

属性(数据)：
```
double _a, _b, _c;
double _area;
```

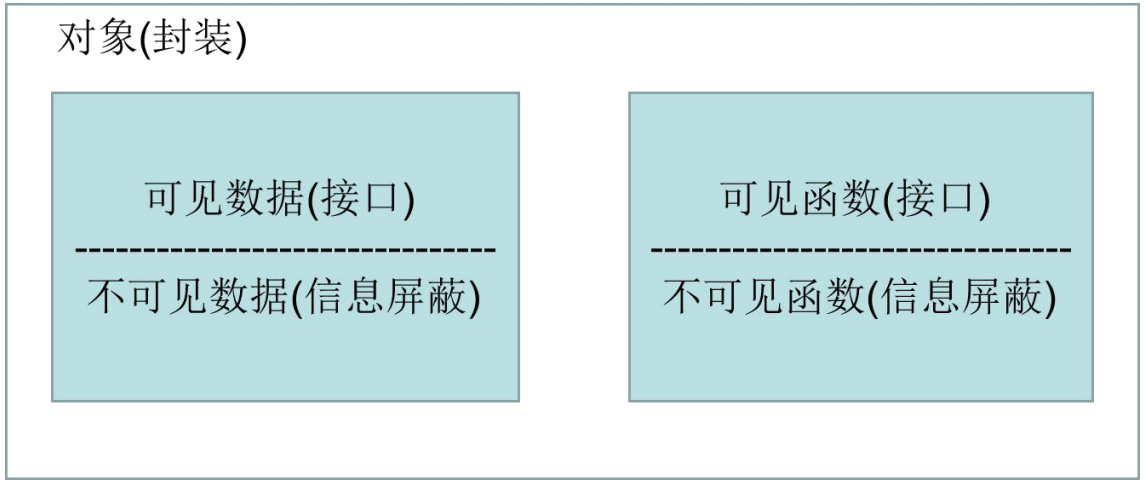
行为(函数)：
```
void set(double a, double b,
double c)
{ _a = a; _b = b; _c = c; }
void calc_area();
void display_area()
{ cout << _area << endl; }
bool is_triangle()
{
  if (_a+_b>_c &&
_a+_c>_b && _b+_c>_a)
    return true;
  return false;
}
```

```
int main() {
  三角形 a;  a.set(3, 4, 5);
  if (a.is_triangle()) { a.calc_area(); display_area(); }
  return 0;
}
```

- 面向对象的程序设计在设计一个系统时，首先要确定系统中包括哪些对象，要分别设计这些对象。
- 在C++ 中，每个对象由数据和函数（操作代码）两部分组成。数据代表了属性，函数是对数据操作的代码，代表了行为。
- 例如三条边长是三角形的属性，利用三条边长计算三角形面积、输出计算结果，实现这些操作的代码就是对象的行为，在程序设计中又称方法。调用对象中的函数就是向对象传递一个消息，要求对象执行某个操作。

2.1.1 封装与信息隐蔽

- 对一个对象进行封装处理，把它的一部分属性和功能向外界屏蔽，从外面看不到这些属性和功能。例如录像机里有电路板和机械控制部件，由于有机壳的保护，在外面只看到一个黑箱子，在它的表面有若干个按键。人们只需知道这些按键的功能即可操作录像机。而不必知道录象机的工作原理和系统结构。



2.1.1 封装与信息隐蔽

- 在设计一个对象时，要周密地考虑如何进行封装，把不必让外界知道的部分隐蔽起来。也就是说，把对象的内部实现和外部行为分隔开来。

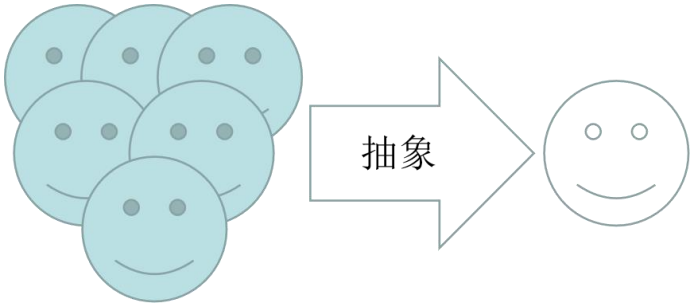
- 封装性是面向对象程序设计的一个重要特点，封装在此有两个含义：

（1）把有关的数据和操作代码封装在一个对象中，形成程序中的一个基本单位，各个对象之间相互独立，互不干扰。

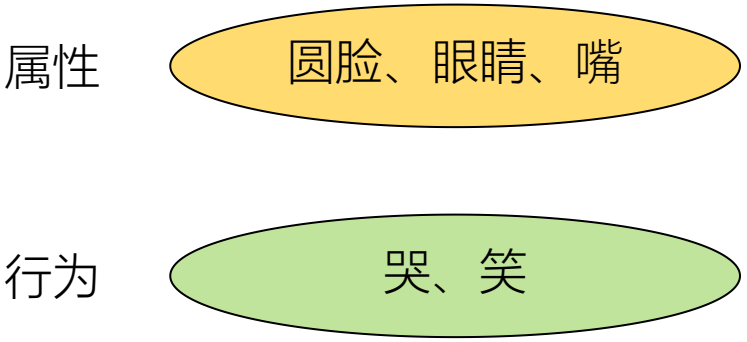
（2）把对象中的某些部分对外隐蔽，只留下与外界联系的接口接收外界的消息，这种对外界隐蔽的做法称为信息屏蔽。封装把对象内部实现与外部分隔开，外界不了解对象内部的具体细节，其实外界也不需要了解。外界需要真正了解的是对象的对外接口。C++ 对象中的公有函数就是对象的对外接口。外界通过调用公有函数，访问对象中的数据成员，完成指定的操作。

2.1.1 抽象

- 抽象是表示同一类事物本质的方法，它关注事物本质特征，对象是具体的，可以将一组同类对象的共同特征抽象出来，从而形成类的概念。类是对象的抽象，而对象是类的具体实例。



对象　　类

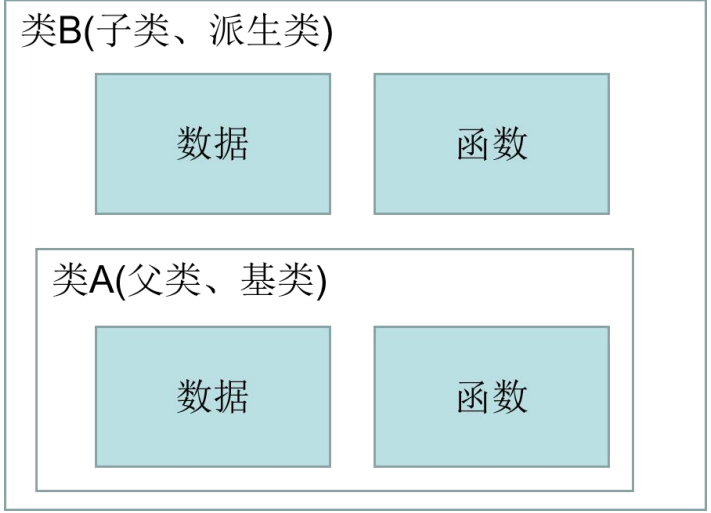


共同特征

2.1.1 继承与重用

- 如果已经建立名为A 的类，现要建立一个名为 B 的类，而B 类与 A 类的内容基本相同，只是在 A 类基础上增加了一些属性和行为，这样只需在 A 类的基础上添加新内容即可，这就是面向对象程序设计中的继承机制。

- C++也具有继承机制，利用这个机制可以在一个已有的类的基础上建立一个新类，这也是软件重用思想，不仅可以利用自己过去建立的类，而且还可以利用其他人放在类库中的类建立类，这就大大缩短了软件开发周期。



- 在已有类的基础上，扩充或改写其某些属性及方法，生成新的类，称为原有类的子类(派生类)，原有类称为新类的父类(基类)。

- 子类包含父类所有的公共属性和行为。

- 优点：减少了编程的代码量和代码的重复；继承体现了面向对象的方法对现实世界人们有组织的抽象思维特点的模拟。

2.1.1 多态性

- 多个相似而不完全相同的对象，收到外界给的同一个消息时，它们各自执行不同的操作，这种现象叫多态现象。

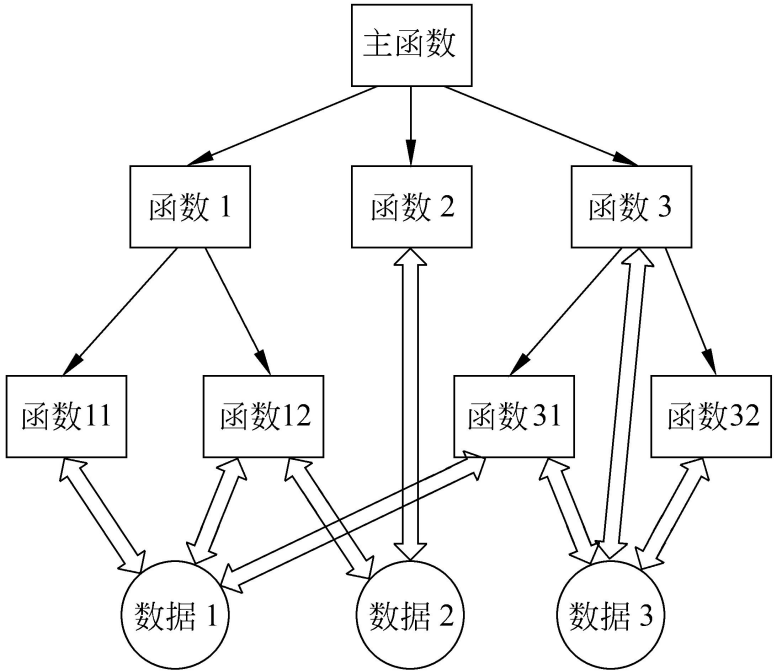
- 在C++中所谓多态性是指用相同的名字定义、调用不同的方法：
  子类对父类方法的重载（用类名区分）
  对本类中同名方法的重载（用参数区分）

- 优点：提高程序的抽象程度
  降低程序的维护工作量，降低代码量

2.1.2 面向对象程序设计的特点

- 传统的面向过程程序设计是围绕功能进行的，用一个函数实现一个功能，一个函数可以使用任何一组数据，一组数据可以被多个函数使用。

- 当程序规模越来越大，数据越来越多，操作越来越复杂，这种方法容易出现错误，程序员往往感到难以应付。



2.1.2 面向对象程序设计的特点

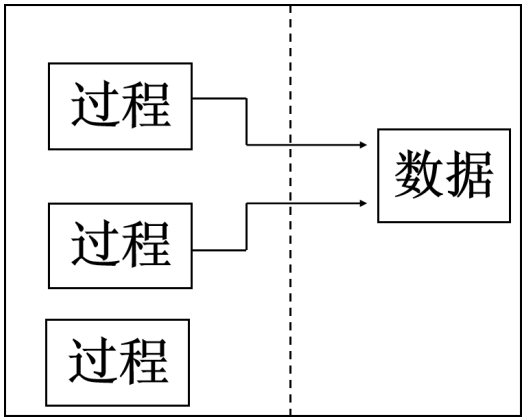
- 面向对象程序设计采用的是另外一种思路。它面对的是一个个对象。实际上每组数据都有特定的用途，它是某种操作的对象。也就是说，一组操作调用一组数据。

- 例如三角形的三条边长 a、b、c 它们只与计算该三角形的面积、三角形的周长等操作有关，与其他操作无关。

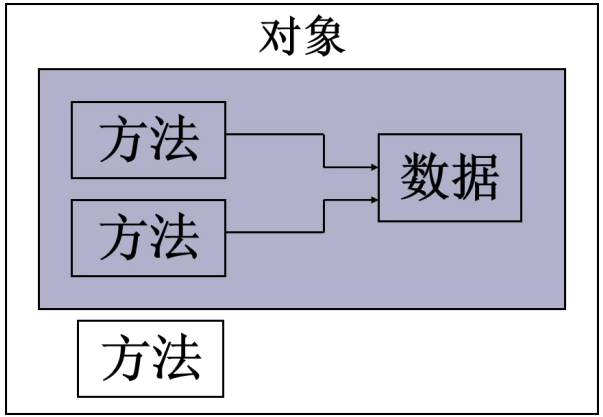
- 我们把这三个数据和计算三角形面积、周长等操作的代码放在一起，封装成一个对象，与外界相对分隔。这也符合客观世界的本来面目。

2.1.2 面向对象程序设计的特点

面向过程的程序



面向对象的程序



- 在面向过程的程序中，数据和过程之间没有逻辑和组织上的联系，这是面向过程程序的主要缺陷
- 在面向对象的程序中，方法和数据被封装在一起形成一个集合，这就是对象，对象就是对现实世界中事物的抽象

2.1.3 类和对象的作用

- 类是C++中十分重要的概念，它是实现面向对象程序设计的基础。C++对C改进，其中最重要的就是增加了类这样一种类型。类是所有面向对象的语言的共同特征，所有面向对象的语言都提供了这种类型。一个有一定规模的C++程序是由许多类构成的。可以说类是C++的灵魂。

- C++支持面向过程的程序设计，也支持基于对象和面向对象的程序设计。从本章到第四章介绍基于对象的程序设计。包括类和对象的概念、类的机制和声明、对象的定义与使用等。这是面向对象程序设计的基础。

2.1.3 类和对象的作用

- 基于对象就是基于类，基于对象的程序是以类和对象为基础的，程序的操作是围绕对象进行的。在此基础上利用继承机制和多态性，就成为面向对象的程序设计。

- 基于对象程序设计所面临的是一个个对象。所有的数据分别属于不同的对象。面向过程的程序设计中数据可以是公用的或者说是共享的，是缺乏保护的。缺乏保护的数据容易导致程序失败。其实程序中一组数据是为一种操作准备的，也就是说一组数据与一种操作对应。因此把一组数据和相关的操作放在一起，这就是面向对象程序设计中的对象。

2.1.3 类和对象的作用

- 在面向过程的结构化程序设计中，人们用下面的公式描述程序：
  程序=数据结构+算法

- 算法和数据结构两者是相互独立的，是分开设计的。面向过程的程序设计是以数据结构为基础的算法设计。在实践中人们逐渐认识到算法和数据结构是互相紧密联系不可分的。应当以一个算法对应一组数据结构，而不宜提倡一个算法对应多组数据结构，以及以一组数据结构对应多个算法。

- 基于对象和面向对象程序设计就是把一个算法和一组数据结构封装在一个对象中，形成了新的观念：
  对象=数据结构+算法
  程序=（对象+对象+...+对象）+消息

- 消息的作用就是对对象进行控制。程序设计的关键是设计好每个对象，以及确定向这些对象发出的命令，让各个对象完成相应的任务。

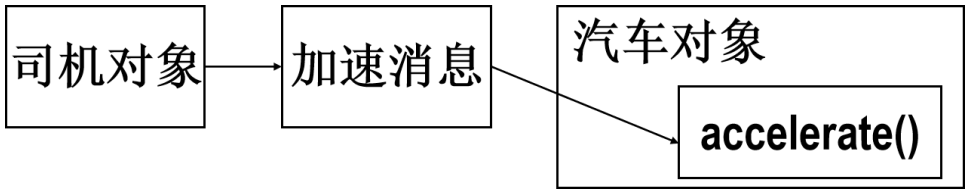
2.1.3 消息

- 消息 — 对象之间相互请求或相互协作的途径，是要求某个对象执行某项操作的规格说明
- 消息内容 — 通常包含接收方及请求接收方完成的功能信息
- 发送方 — 发出消息，请求接收方响应
- 接收方 — 收到消息后，经过解释，激活方法，予以响应

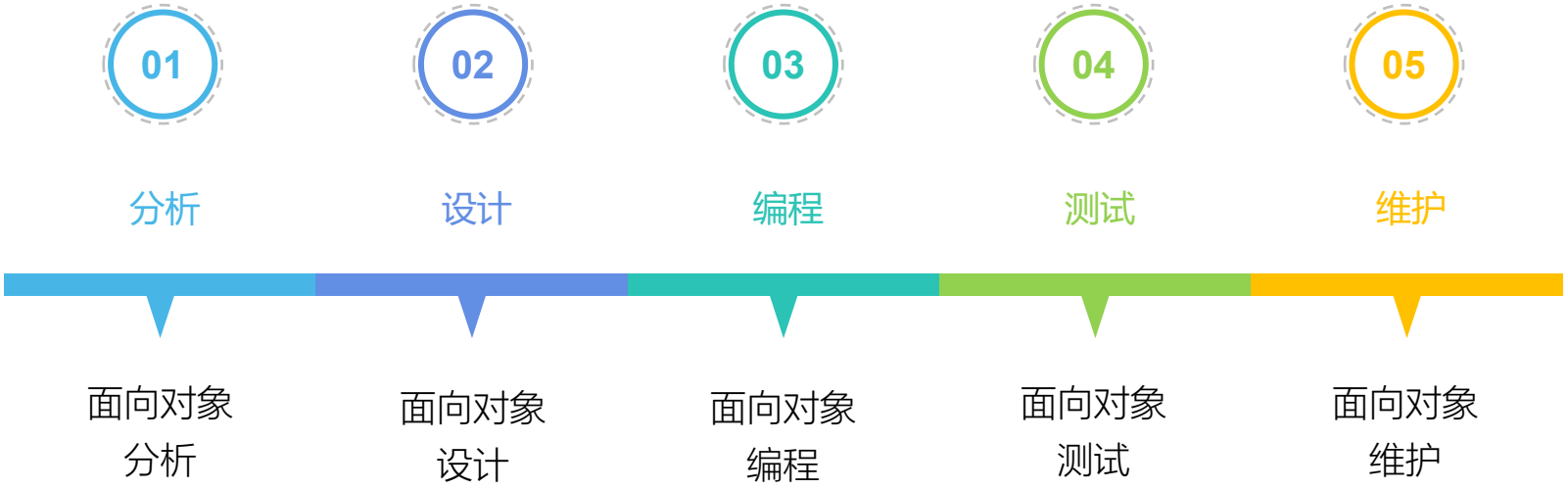
2.1.3 消息

- 孤立的对象动作没有太大的用处，一个对象需要对其他对象做些有实际意义的事情：司机对象向汽车对象发送“加速”消息，这个过程实际上就 是司机对象通过调用汽车对象的accelerate()方法向汽车对象发送“加速”消息。

- 一条消息由三件事组成：
  - 接收消息的对象 (小汽车)
  - 执行的动作 (加速)
  - 该操作所需要的参数 (+15mph)



2.1.4 面向对象的软件开发



2.1.4 面向对象的软件开发

- 面向对象分析：要按照面向对象的概念和方法，在对任务的分析中，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：根据面向对象分析阶段形成的需求模型，对每一部分分别进行具体的设计，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：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++、VB...

- 面向对象测试：写完程序交付用户使用前，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：任何软件产品在使用过程中，可能用户需要改进软件的性能，这就需要修改程序。由于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。

2.2 类的声明和对象的定义

- 在面向对象程序设计中，要使用大量的对象，要程序员对问题进行分析抽象出类的概念，在设计程序时首先要声明类类型，然后再定义类类型变量即对象。

类和对象的关系

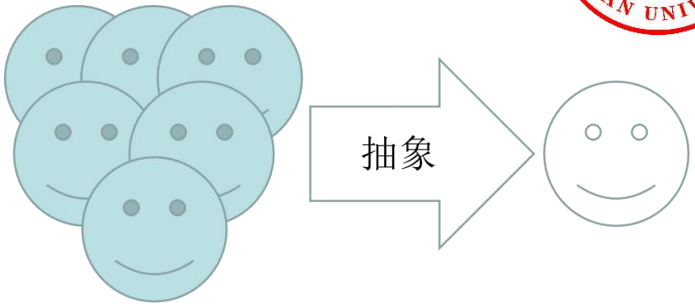
声明类的类型

类和结构体类型的异同

定义对象的方法

2.2.1 类和对象的关系

- C++中对象的类型称为类（class）。类代表了某一批对象的共性和特征。
- 类是对象的抽象，而对象是类的具体实例。
- 一个类可以定义多个对象，每个对象包含类中定义的各个属性的存储空间，他们共享类中定义的行为特征。



- 类与对象的关系类似于整数类型int与整数变量i的关系：类和整数类型int代表的是一般的概念，不占存储空间，类中的属性不能赋值、存取、运算；对象和整型变量代表的是具体的东西，需要占用存储空间，对象中的数据可以赋值、存取、运算。
- int x;　int *y;　Screen x;　Screen *y;

2.2.2 声明类的类型

- 类是要用户自己定义的类型，如果程序里要用类类型，程序员必须自己根据需要声明，也可以使用别人已设计好的类。

- 声明一个类类型与声明一个结构体类型相似。结构体是一种构造数据类型，用途：把不同类型的数据组合成一个整体——自定义数据类型。

结构体类型定义格式：

```
struct <结构体名>
{
  类型标识符    成员名;
  类型标识符    成员名;
  ……
};
<结构体名> 变量名表;
```

```
// 定义结构体类型
struct  student
{
  int  num;            // 学号
  char  name[20];      // 姓名
  char  sex;           // 性别
};
student  st1,st2;      // 定义结构体变量
```

2.2.2 声明类的类型

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

// 定义结构体类型
struct  student {
  int  num;              // 学号
  char  name[20];     // 姓名
  char  sex;          // 性别
};
student  st1,st2;  // 定义结构体变量
```

```
void  setdata(student *p) {  // 设置
  cin >> p->num;
  cin >> p->name;
  cin >> p->sex;
}
void display(student s) {    // 显示
  cout << s.num << endl;
  cout << s.name << endl;
  cout << s.sex << endl;
}
```

```
int main()
{
  setdata(&st1);
  setdata(&st2);
  display(st1);
  display(st2);
  return 0;
}
```

- 对于结构体变量，可以通过点操作符"."来访问结构中的成员。例如：st1.num = 1001;
- 对于结构体变量的指针，可以通过箭头操作符"->"来访问结构中的成员。例如：student *p = &st2; p->num = 1002;
- 结构体只解决了把数据封装在一起，而没有把处理数据的操作(函数)包含进去，造成了数据和操作分离。那么类是如何定义的呢？

2.2.2 声明类的类型

```
类定义格式:
    class    类名
    {            private:
                    私有成员声明
                 public:
                    公有成员声明
                 protected:
                    保护成员声明
    };
```

- class是保留字，声明类类型
- 类名：按标识符取名

- private、public、protected也是保留字，是成员访问限定符，其后必须跟冒号。在定义类时，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。

- 在类声明的 } 后如不直接定义对象就必须跟分号。

- private：只能被本类中的成员函数访问，类外（除友元外）不能访问。
- public：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。

2.2.2 声明类的类型

```
class    student
{ private:              // 从此句开始以下为私有成员
    int   num;
    string  name;
    char sex;
  public:              // 从此句开始以下为公有成员
    void  setdata()
   {    cin >> num;   cin >> name;  c in >> sex;
   }
   void display()
   {    cout << num << endl;  cout << name << endl;  cout << sex << endl;
   }
};
student   st1,st2;
```

2.2.3 定义对象的方法

- 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;

- 在声明类类型的同时定义对象

```
class 类名
        { private:
            ...
          public:
            ...
        } 对象名表;
```

```
class   student
    {
          int   num;
        public:
          void  setdata()
             {
               cin >> num;
             }
    } st1,st2;
```

2.2.3 定义对象的方法

- 不出现类名，直接定义对象
- 虽然是合法的，但不提倡使用

```
class
        {
        private:
            ...
        public:
            ...
        } 对象名表;
```

- 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。
- 在实际程序开发中一般采用第一种方法，在小型程序中或声明的类只用于本程序时，也可采用第二种方法。
- 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。

2.2.3 类和结构体类型的异同

- C++允许用struct定义一个类类型，为什么C++ 要这样做?
- 这是设计C++ 语言时规定的一项原则：C++必须兼容C，让C程序不用修改就能在C++环境中使用。
- 两种定义方法还是有区别的：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。

```
class  student {
  // 默认私有
  int  num;              // 学号
  char  name[20];        // 姓名
  char  sex;             // 性别
};
```

```
struct  student {
// 默认公用
  int  num;              // 学号
  char  name[20];        // 姓名
  char  sex;             // 性别
};
```

2.3 类的成员函数

- 在声明类时，必须定义了访问类数据成员的函数，称之类的成员函数

成员函数的性质

在类外定义成员函数

成员函数的存储方式

内置成员函数
(inline成员函数)

2.3.1 成员函数的性质

- 类成员函数是函数中的一种，成员函数可以访问本类中的所有成员。
- 如果成员函数的成员访问限定符是public，则对象可以通过它访问类的其他成员；如果是private或protected，则对象不能用来访问类的其他成员。
- 一般方法是把需要与外界沟通的成员函数指定为public，作为类与外界的接口。

```
class    student
{
 private:
// 从此句开始以下为私有成员
    int   num;
  public:
// 从此句开始以下为公有成员
    void  setdata()
   {    cin >> num;      }
   void display()
   {    cout << num << endl;  }
};
student   st1,st2;
```

2.3.2 在类外定义成员函数

- C++允许在类内声明成员函数的原型，然后在类外定义成员函数
- 格式： 类型 类名::函数名（形参表）{ …. }
- 类外定义成员函数时，必须在函数中增加类名，用于限定它属于哪个类，::是作用域限定符或作用域运算符。
- 右侧的例子中如果函数不用成员访问限定符，函数就成了全局作用域中的setdata函数而不是成员函数。
- 如果在::前不带类名，或函数名前既无类名又无作用域运算符::，表示该函数是全局函数。

```
class    student
{        int   num;
         string  name;
         char sex;
   public:
         void   setdata();
};

 void  student :: setdata()
{          cin >> num;
           cin >> name;
           cin >> sex;
}
```

2.3.3 inline成员函数

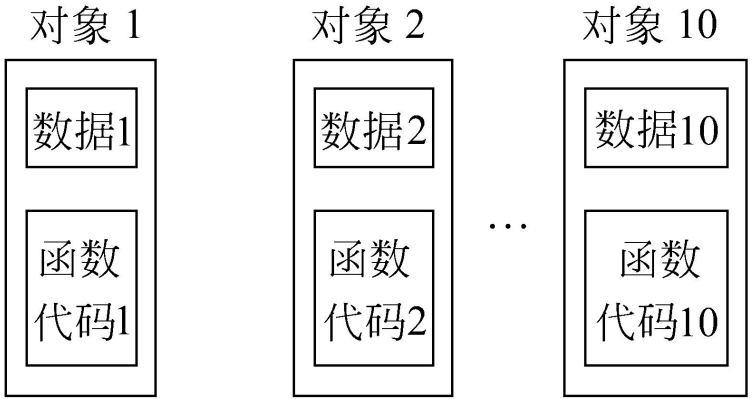
- C++默认在类内定义的成员函数是inline函数，如果在类外定义成员函数时，系统默认不把它当作inline成员函数，此时必须在声明函数时在函数前加inline关键字。
- 在右侧的例子中在函数声明里，函数头未加inline，C++将把它当作非inline函数。
- 如果要把它作为inline函数，只需在函数声明时，在函数头增加关键字inline。

```
class    student
{        int   num;
         string  name;
         char sex;
   public:
         void   setdata();
// inline  void   setdata();
};

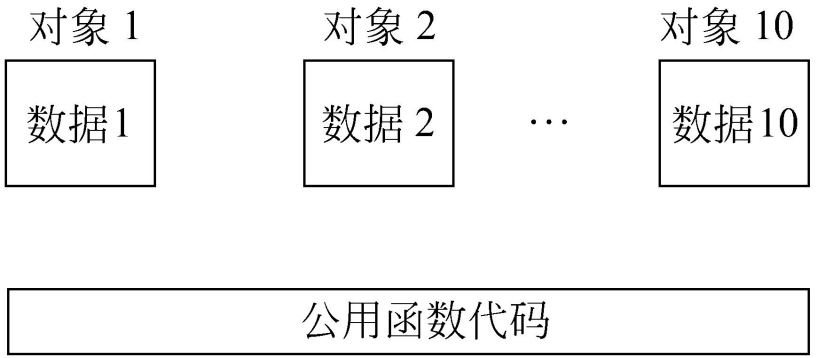
 void  student :: setdata()
{           cin >> num;
            cin >> name;
            cin >> sex;
}
```

2.3.4 成员函数的存储方式

- 用类定义对象时，系统为每个对象分配内存空间，同一类对象的成员函数是一样的如果每个对象成员函数都分配内存空间，会造成大量浪费。



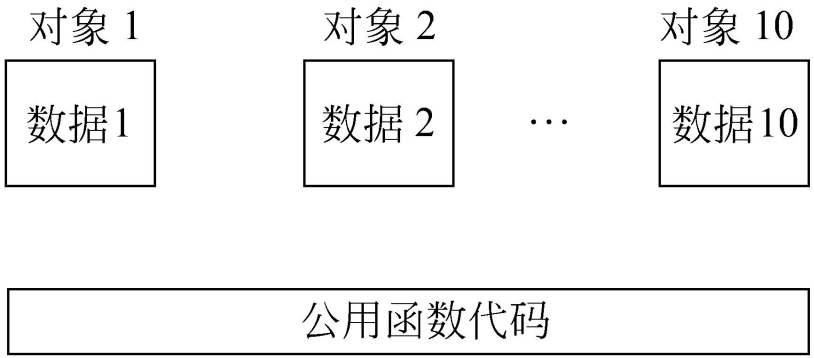
- C++ 处理的方法是，只为对象的数据成员分配内存空间，一个类的所有对象共享一个成员函数空间。



2.3.4 成员函数的存储方式

- 可以用下面的语句计算该类对象占用的字节数
  cout<< sizeof(Time) <<endl;

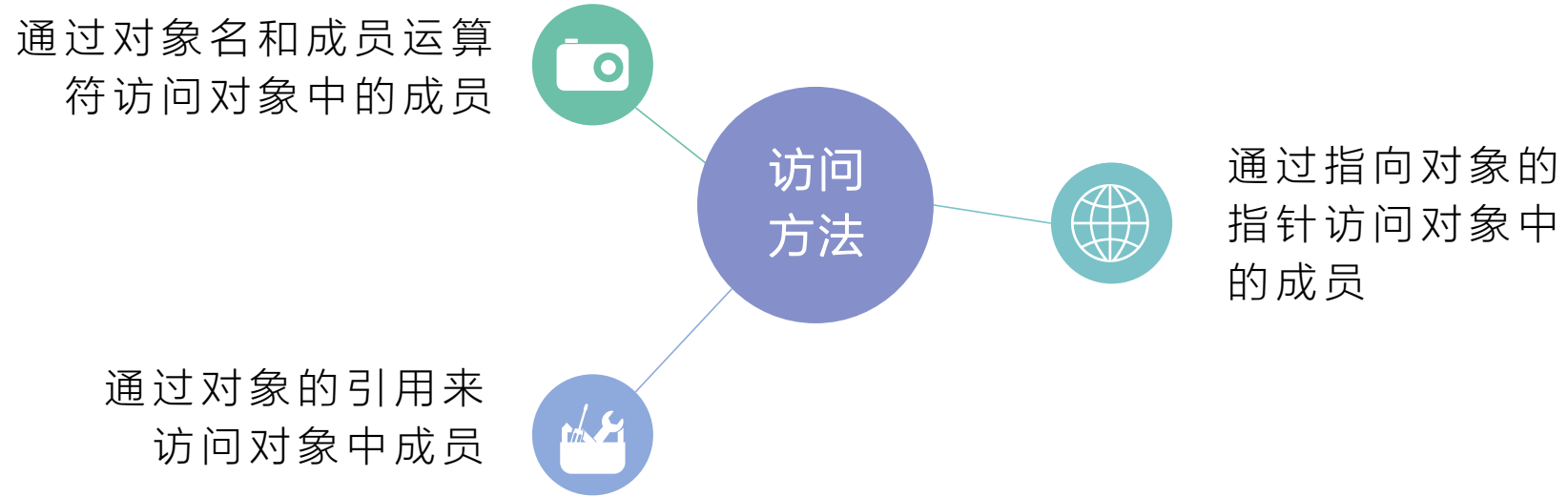
- 结果输出值是12，这是数据成员所占的空间尺寸，这就证明一个对象占用的空间其实是它的数据成员占据的内存空间。




```
class Time
{ public:
  int hour;
// sizeof(int)=4
  int minute;
// sizeof(int)=4
  int sec;
// sizeof(int)=4
  void set()
  { cin >> hour ; }
};
```

2.4 对象成员的引用

- 定义了对象后，在程序中可以直接访问对象中的公有成员，它们可以是数据成员，也可以是成员函数。
- 在程序中访问对象成员有三种方法



2.4.1 通过对象名和成员运算符访问对象中的成员

- 格式：对象名.成员名
- 例：st1.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：
  st1.num = 10101;

```
class    student
{        int   num;
         string  name;
         char sex;
   public:
         void   setdata();
         void   display();
}st1, st2;

 void  student :: setdata()
{           cin >> num;
            cin >> name;
            cin >> sex;
}
```

2.4.2 通过指向对象的指针访问对象中的成员

```
class Time
{public:
    int hour;
    int minute;
};

int main()
{
Time  t, *p;
p= &t;
cout<< p->hour<<endl;
return 0;
}
```

- 可以通过指针访问对象中的成员。
- p->hour表示p当前指向对象t中的成员hour，此时(*p).hour也代表对象t中的成员hour
- 在这个例子中，p->hour、(*p).hour、t.hour三种表示是一个意思

2.4.3 通过对象的引用来访问对象中的成员

- 如果为一个对象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;
  ```

- 这里t2是t1的别名，所以访问t2.hour就是访问t1.hour。

```
class Time
{public:
    int hour;
    int minute;
};

int main()
{
Time  t1;
Time & t2=t1;
cout<< t2.hour<<endl;
return 0;
}
```
```

2.5 类和对象的简单应用举例

例2.1 时间类

```
#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.mi
nute<<":"<<t2.sec<<endl;
  return 0;
}
```

- 必须在引用成员之前加对象名而不是类名。
- 数据成员必须有初始值后才能访问，否则它们的值是不可预知的。

2.5 类和对象的简单应用举例

例2.2 引用多个对象成员：只有两个对象就显得有些繁琐，为了解决这个问题，可以在主函数中定义输入输出函数进行输入和输出

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

class Time
{ public:
  int hour;
  int minute;
  int sec;
};
int main()
{ Time t1;
  cin>>t1.hour >>t1.minute>>t1.sec;
  cout<<t1.hour<<":"<<t1.minute<<":"<<t1.sec;
  cout<<endl;
  Time t2;
  cin>>t2.hour>>t2.minute>>t2.sec;
  cout<<t2.hour<<":"<<t2.minute<<":"<<t2.sec;
  cout<<endl;
  return 0; }
```

2.5 类和对象的简单应用举例

例2.2 引用多个对象成员：主函数中定义输入输出函数-引用形参

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

class Time
{ public:
  int hour;
  int minute;
  int sec;
};

int main()
 {
 //  使用引用形参
 void set_time( Time& ) ;
 void show_time( Time& );
 Time t1;
 set_time(t1);
 show_time(t1);
 Time t2;
 set_time(t2);
 show_time(t2);
  return 0;}

void set_time(Time& t)
 {
   cin>>t.hour;
   cin>>t.minute;
   cin>>t.sec;
}

void show_time(Time& t)
{
cout<<t.hour<<":"<<t.minute
<<":"<<t.sec<<endl;
}
```

2.5 类和对象的简单应用举例

例2.3 用成员函数处理输入输出

```
#include <iostream>
using namespace std;
class Time
{ public:
  void set_time() ;
  void show_time();
private:
  int hour;
  int minute;
  int sec;
 };
```

```
int main()
 {
  Time t1;
  t1.set_time();
  t1.show_time();
  Time t2;
  t2.set_time();
  t2.show_time();
  return 0;
 }
```

```
void Time::set_time()
{
  cin>>hour;
  cin>>minute;
  cin>>sec;
}

void Time::show_time()
 {
cout<<hour<<":"<<minute<
<":"<<sec<<endl;
}
```

2.5 类和对象的简单应用举例

例2.4 在整型数组中找最大值

```
#include <iostream>
using namespace std;
class Array_max
{public:
   void set_value();
   void max_value();
   void show_value();
 private:
   int array[10];
   int max;
};
void Array_max::set_value()
{ int i;
  for (i=0;i<10;i++)
    cin>>array[i];}

void Array_max::max_value()
{int i;
 max=array[0];
 for (i=1;i<10;i++)
      if(array[i]>max) max=array[i];}

void Array_max::show_value()
  {cout<<"max="<<max; }
int main()
 {
  Array_max  arrmax;
  arrmax.set_value();
  arrmax.max_value();
  arrmax.show_value();
  return 0;
 }
```

2.6 类的封装性和信息隐蔽

面向对象程序设计
中的几个名词

实现公用接口与
私有的分离

类声明和成员函
数定义分离

2.6.1 公用接口与私有实现的分离

- C++ 通过类实现封装性，类的作用就是把数据和关于操作数据的算法封装在类类型中。
- 在声明类时，一般将数据成员指定为私有，使它们与外界隔离，把让外界调用的成员函数指定为公有，外界通过公有函数实现对数据成员的访问。
- 外界与对象的唯一联系就是调用公有成员函数。公有成员函数是用户使用类的公用接口。
- 用户可以调用公有成员函数实现某些功能，用户也只要了解每个公有成员函数的功能，不必了解这些功能是怎样实现的，这就是接口与实现分离。
- 为了防止用户任意修改公有成员函数，从而改变对数据的操作，往往不让用户看到公有成员函数源代码。类中的数据是私有的，实现数据操作的细节对用户是隐蔽的这种实现称为私有实现。这种类的公有接口与私有实现的分离形成了信息隐蔽。

2.6.1 公用接口与私有实现的分离

- 信息隐蔽的长处包括：

- 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。例如，在2.2.3节声明的student类中增加数据成员年龄，可以修改为右侧代码。

- 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。

- 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。这就使得程序（尤其是大程序）的设计、修改和调试变得方便和简单。

```
class    student
{  int   num;
   string  name;
   int   age;    // 新增加的数据
   char sex;
public:  void   setdata();
            void  display(); };

void  student :: setdata()
{cin >> num >> name;
 cin >> age>> sex; // 新增加的}
void student :: display()
{ ...}
```

2.6.2 类声明和成员函数定义分离

- 面向对象程序开发时，一般将类的声明（包括成员函数的声明）放在指定的头文件中，在程序中只要把有关的头文件包含进来即可。不必在程序中重复写类的声明。还有一种方法是把类成员函数不定义在头文件中，而另外放在一个文件中。

```
//  student.h
//这是头文件在此声明类
class    student
{  int   num;
   char  name[10];
   char sex;
public:
    void  setdata();
    void  display();
};
```

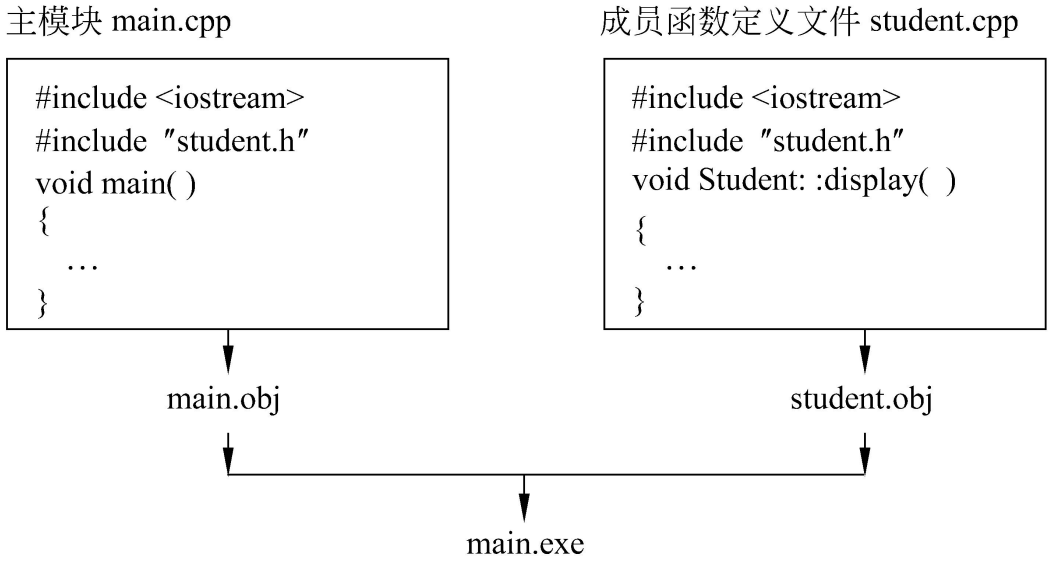
```
//  student.cpp
#include <iostream>
#include "student.h"
using namespace std;
void  student :: setdata()
{cin >> num ;
}
void student :: display()
{   cout<< num  <<endl;
}
```

```
//  main.cpp
#include <iostream>
#include "student.h"
using namespace std;
int main()
{           student st1;
            st1.setdata();
            st1.display();
            return 0;
 }
```

2.6.2 类声明和成员函数定义分离

- 这个程序包含了三个文件，一个是主模块main.cpp，一个是student.cpp。在预编译时会将头文件student.h中内容取代#include “student.h”行。

- 可以按对多个文件程序的编译和运行方法对程序进行编译和连接。C++编译系统对两个.cpp文件分别进行编译，得到两个.obj文件，然后将它们和其他系统资源连接起来，形成可执行文件main.exe。



2.6.2 类声明和成员函数定义分离

- 可以不必每次对student.cpp进行编译，把第一次编译后形成的目标文件保存起来，以后在需要时把它调出来直接与程序的目标文件相连即可。这和使用函数库中的函数是类似的。
- 在实际中，将若干个常用的功能相近的类声明集中在一起，形成类库。类库包括C++编译系统提供的标准类库，用户的类库。
- 类库有两个组成部分：（1）类声明头文件;（2）经过编译的成员函数的定义，它是目标文件。
- 用户只要把类库装入C++编译系统所在的子目录中，并在程序中用include命令把类声明的头文件包括到程序中，就能使用这些类和其中的成员函数。

2.6.3 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();
- 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。

面向对象程序设计

Object Oriented Programming

2021

谢谢大家