高级语言程序设计II

四川大学计算机学院
四川大学计算机学院
陈良银
cly_6666@sina.com
http://cs.scu.edu.cn/~chenliangyin

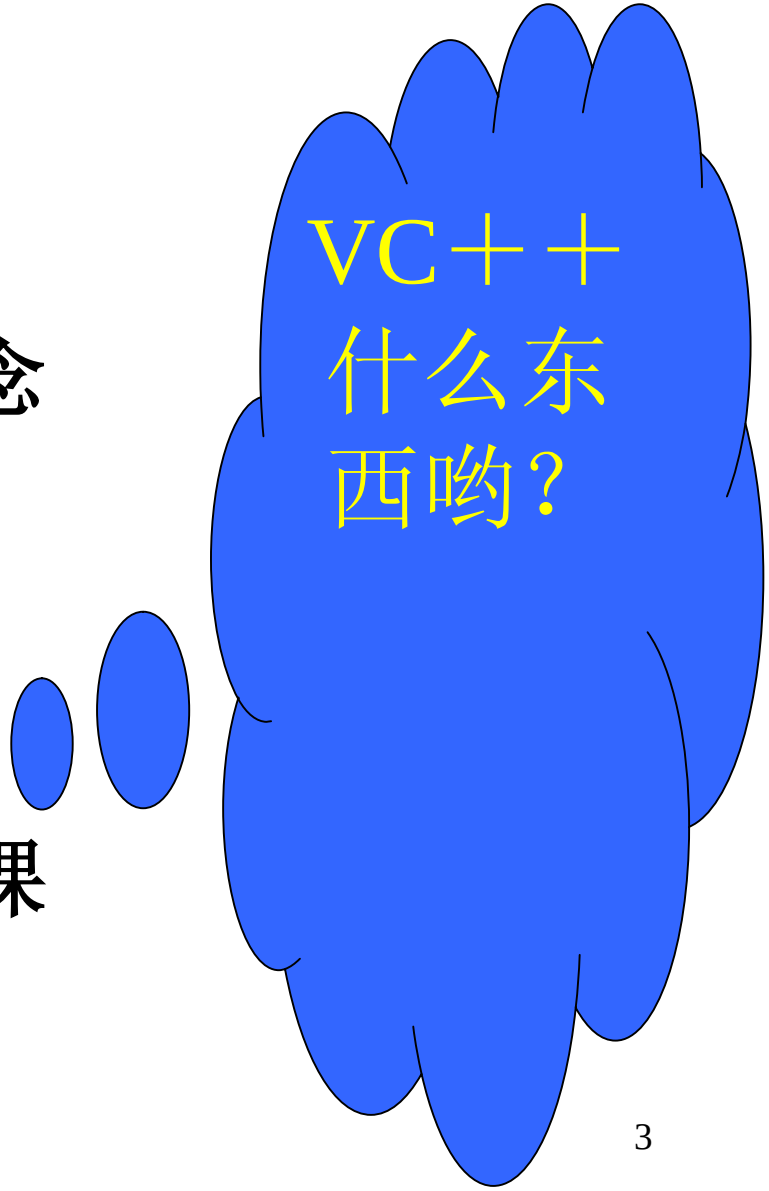
星期一、第4大节、N1-A509。

教学目的

- 1. 掌握面向对象程序设计的基本理论和基本知识；
- 2. 掌握面向对象程序设计基本思想与方法；
- 3. 培养面向对象的思维习惯和思维方式。

教学要求

- **1. 能比较熟练地用C++语言进行面向对象的程序设计。**
- **2.了解面向对象的基本概念和使用面向对象技术进行程序设计的基本思想。**
- **3. 初步掌握面向对象编程环境Visual C++的使用。**
- **4.为学习计算机系统高级课程做准备。**



教学时间、内容安排

- 2学分，34 学时，1－17周上课。
- 1学分，20学时，实验（时间协调）。

一　引论（2学时)

二　C＋＋：一个更好的C（6学时）

三　类类型（8学时）

四　运算符重载（7学时）

五　派生类（7学时）

六　流库（1学时）

七　模板（1学时）

使用教材

- C＋＋语言设计
- (第二版）

- 高等教育出版社

主要参考资料

- 1、面向对象程序设计基础　李师贤等著　高等教育出版社 1998 （无C基础选用）
- 2、面向对象程序设计语言C++　陈文宇编著　机械工业出版社　2004
- 3、C++面向对象程序设计简明教程　廉师友编著　西安电子科技大学出版社　1998
- 4、C++程序设计　谭浩强编著　清华大学出版社 2004　(无C基础选用）
- http://cs.scu.edu.cn/~chenliangyin

考核办法

- 《高级语言程序设计II》采用闭卷考试方式（占2学分）

  ——作业占？？%，期末考试占？%。
- 计算机实践操作采用上机考核方式（占1学分）

  ——学生提供实验报告或者上机实际操作考核方式。

面向对象程序设计语言C++

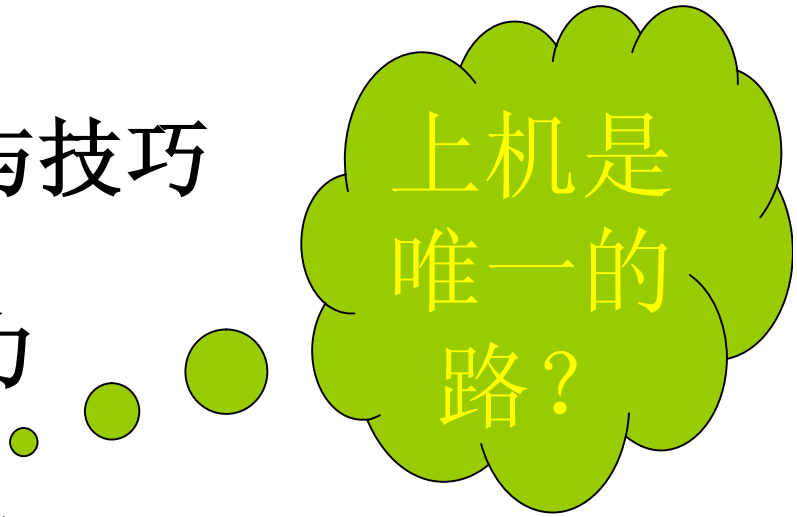
- 第1章　引论
- 第2章　C++语言与C语言的不同
- 第3章　类类型
- 第4章　运算符重载
- 第5章　派生类
- 第6章　流库
- 第7章　模板
- 第8章　面向对象设计技术（自学）
- 第9章　命名空间和例外处理（自学）

第一章　引论

- 面向对象的目标
- 面向对象语言的核心概念
- 按对象方式思维
- 面向对象的思想和方法
- 类属
- 面向对象的程序设计语言

学习方法

- 多练习，掌握基本概念
- 多读程序，学习编程方法与技巧
- 多上机实践，加强动手能力
- 多剖析范例，积累编程经验



面向对象的由来和发展

- 机器语言（二进制码）
- 汇编语言
- 20世纪50年代中期，**FORTRAN**语言

在计算机语言发展史上具有划时代的意义，引入了许多程序设计概念。如变量、数组、循环、分支等。

- 20世纪50年代中期，**Algol**语言

提出块(Begin…End)思想，对数据进行保护，是一种初级封装。

面向对象的由来和发展

- 20世纪60年代中期，Simula 67语言

面向对象语言的鼻祖，提出了对象、类的概念，并支持类的继承。

- 20世纪70年代中期，Ada语言

支持数据抽象类型的最重要的语言之一，但不完全支持继承。

- 20世纪70年代中期，Smalltalk语言

最有影响的面向对象语言之一，丰富了面向对象的概念。

- 20世纪80年代中期后出现C++等多种面向对象语言

C++的起源和特点

一、C++的起源

- 在C语言基础上为支持面向对象的程序设计研制的一个通用目的的程序设计语言；
- 由AT&T贝尔实验室Bjarne Stroustrup博士开发；

二、C++的特点

- 与C兼容，既保持了C的简洁、高效和接近汇编的特点，又比C更安全，结构化程度更高；
- 既支持面向过程的程序设计，又支持面向对象的程序设计；

注意其两面性

面向对象的由来和发展

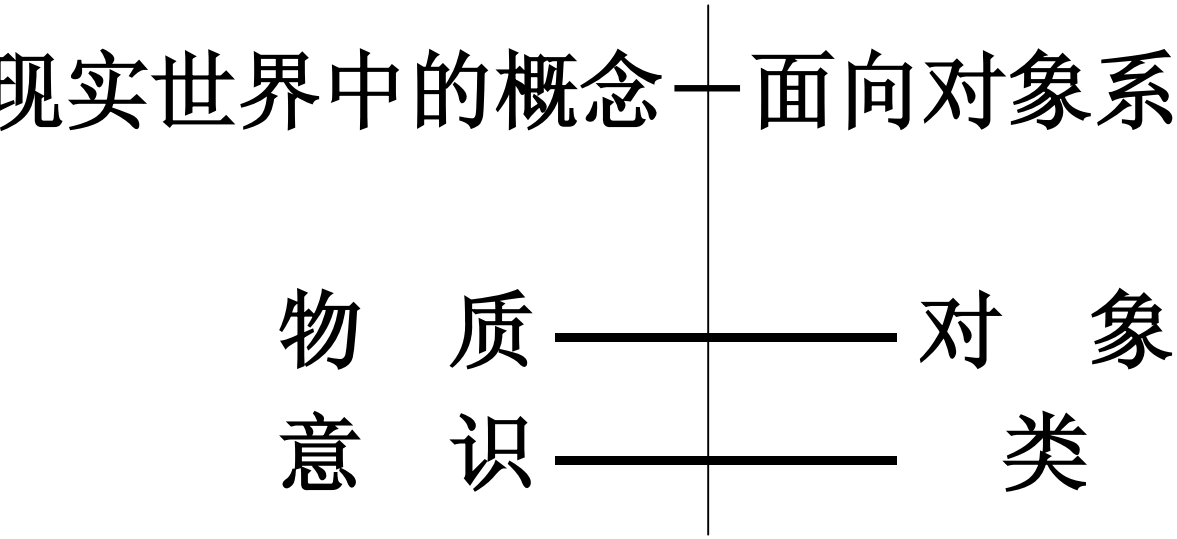
- 对象：表示现实世界中的具体事物。

- 面向对象技术追求的是软件系统对现实世界的直接模拟，尽量实现将现实世界中的事物直接映射到软件系统的解空间

太过分了，居然什么都可以是对象。

面向对象的由来和发展

- 现实世界中的概念——面向对象系统中的概念



- 物质：表达具体的事物
- 意识：描述抽象的概念

面向对象的由来和发展

• 现实问题空间	面向对象解空间
• 物　质： • 一辆白色自行车 • 意　识： • 自行车	具体事物： 一个对象—自行车的实例 抽象概念： 自行车类

面向对象的由来和发展

• 现实问题空间	• 面向对象解空间
• 客观世界的实体+相互间的联系。	• 把要构造的系统表示为对象的集合。对象之间通过消息联系。 • 每个实体对应一个数据结构。 • 直观模型化。 • 信息局部化。

面向对象的由来和发展

- 二、面向对象程序设计：
- 通过增加软件可扩充性和可重用性来改善并提高程序员的生产能力，并控制维护软件的复杂性和软件维护的开销

面向对象的由来和发展

- 在结构化程序设计中，结构化模块的缺点
- 缺点：把数据和过程分离为相互独立的实体
- 恶果：1.可重用性不好
- 不同数据作相同处理
- 相同数据作不同处理
- 需编写不同的程序

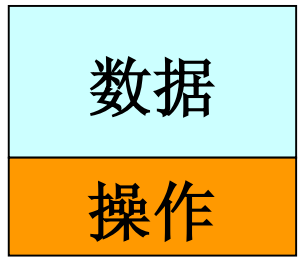
面向对象的由来和发展

- **2.要求数据与程序始终保持相容**
-

面向对象的由来和发展

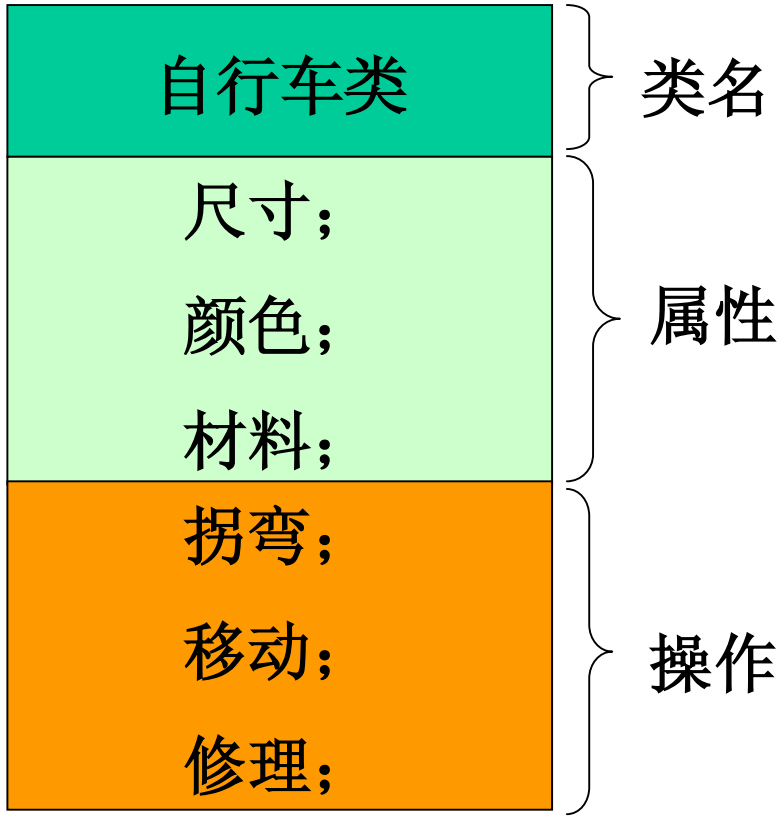
- **OOP**则具有结构化程序设计的一切优点
- 　　将数据和对数据的操作放在一起，作为一个相互依存、不可分割的一个整体，抽象成一种新的数据类型—类

- 两大技术（两个重要原理）
- 　　数据抽象
- 　　信息隐藏



面向对象的由来和发展

- 自行车类：
- （1）数据：尺寸、颜色、材料等（属 性）
- （2）操作：转弯、移动、修理等（操作）
- 每一辆具体的自行车就是属于自行车类的一个对象。



UML中的类

面向对象的由来和发展

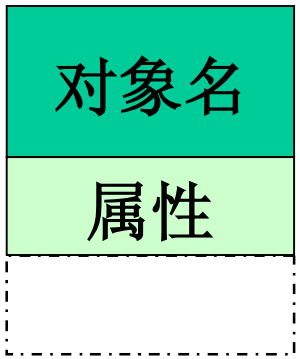
- **OOP支持的软件开发策略**

- 编写可重用代码
- 编写可维护的代码
- 共享代码
- 精化已有的代码

面向对象的五大基本概念

- 对象、类、消息、方法、继承
- **1．对象（Object）：数据及可以对这些数据施加的操作结合在一起所构成的独立实体的总称。包括具体事物和抽象概念。**

  对象：属性＋行为。



- 成员变量（数据）：表明对象的状态
- 成员方法（操作）：表明对象的行为
- 一组成员变量和相关的方法集合。
- ——对象即是实例。

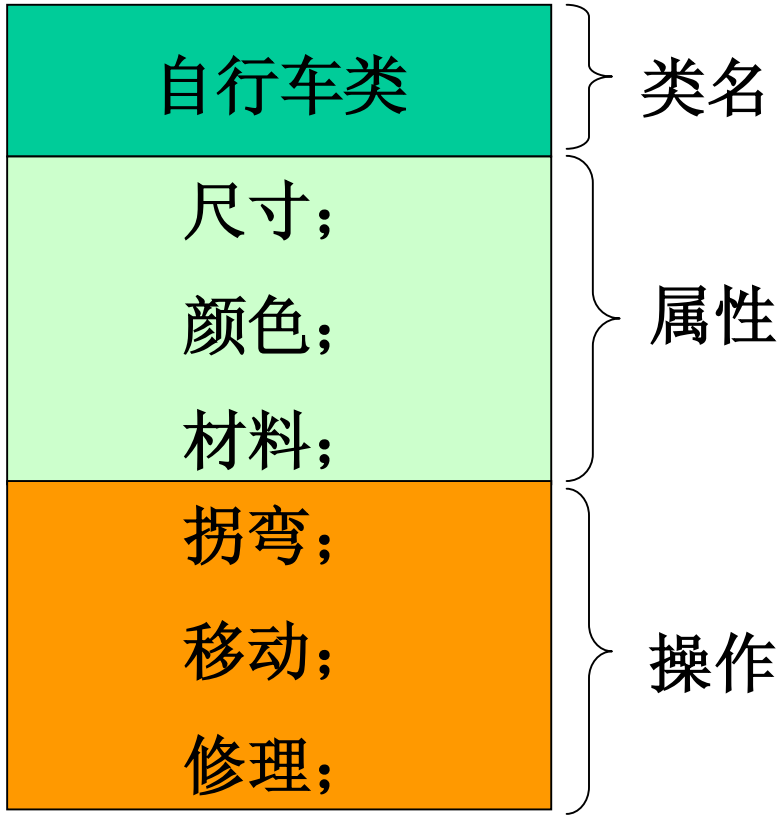
面向对象的五大基本概念

- **2．类（Class）：对一组具有相同数据和相同操作的对象的描述（定义），依据共同的行为将有关对象进行的一种分组。它是数据和作用在其上的操作构成的整体。**
- 类中所有对象共享

  ——类中所有共同的特性。
- 类是对象的抽象

——程序中：只有类
- 对象是类的实例

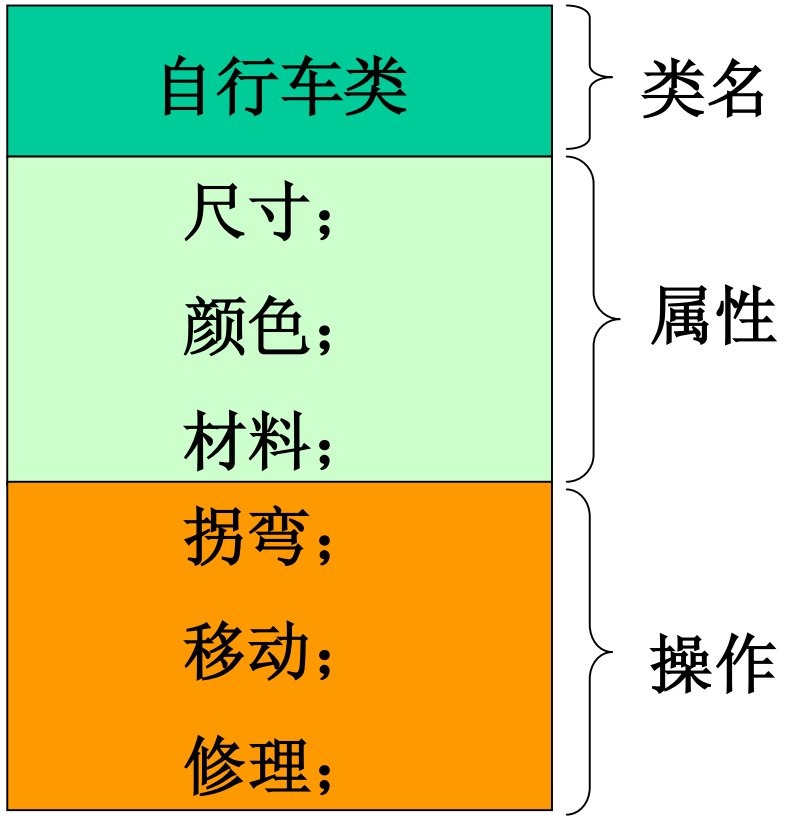
——运行时：只有对象



UML中的类

面向对象的五大基本概念

3．消息（Message）：要求某个对象执行类中所定义的某个操作的规格说明。就是要求对象进行某种活动的信息。消息是对象间合作的途径。

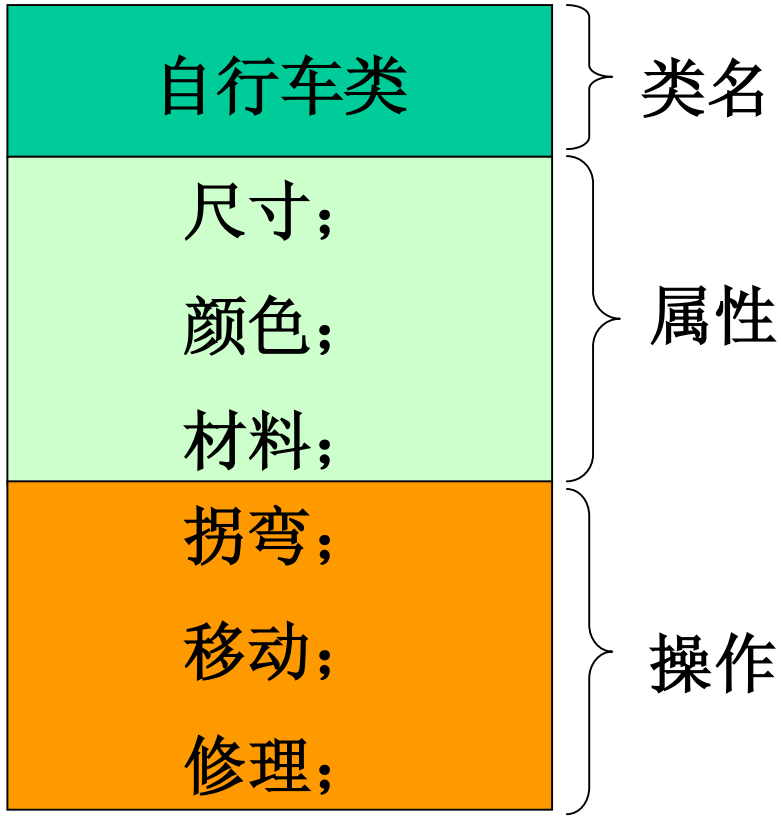


UML中的类

面向对象的五大基本概念

- **4．方法（Method）：**对象所能执行的操作

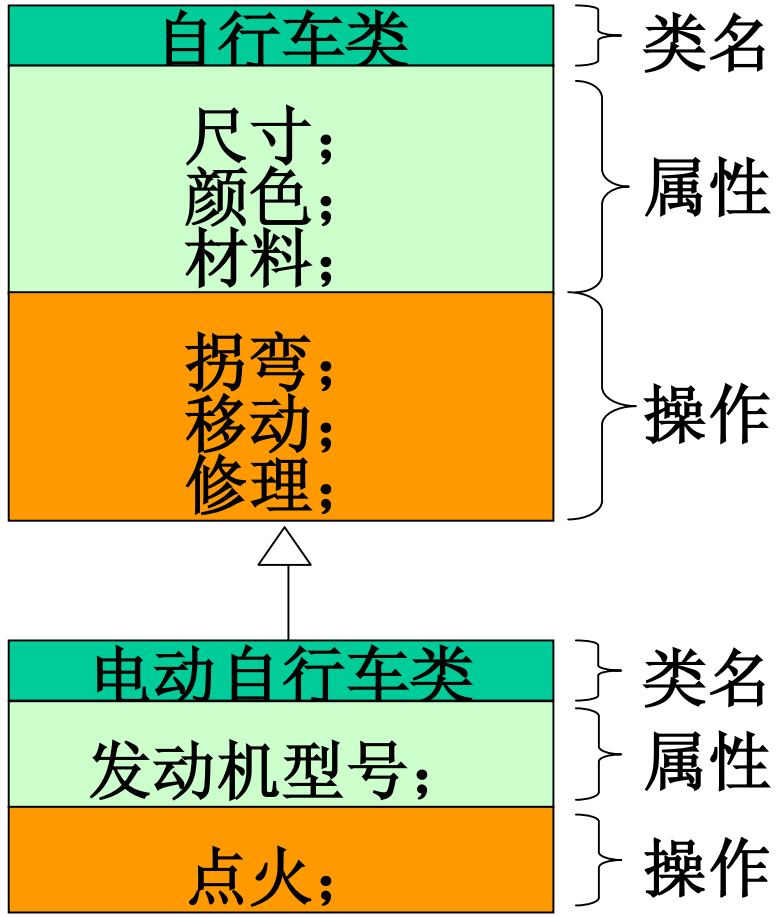
- 它是类中定义的函数，描述对象执行操作的算法，即响应消息的方法。方法包括定义和实现两部分。



UML中的类

面向对象的五大基本概念

- 5．继承：一个类向上可能有父类，向下可能有子类。子类就从父类那里继承了其属性和行为。凡父类中的数据和操作也都是子类的数据和操作。
- 另外，对象和类的实例是同义词



类的继承关系

面向对象方法学的四大要点

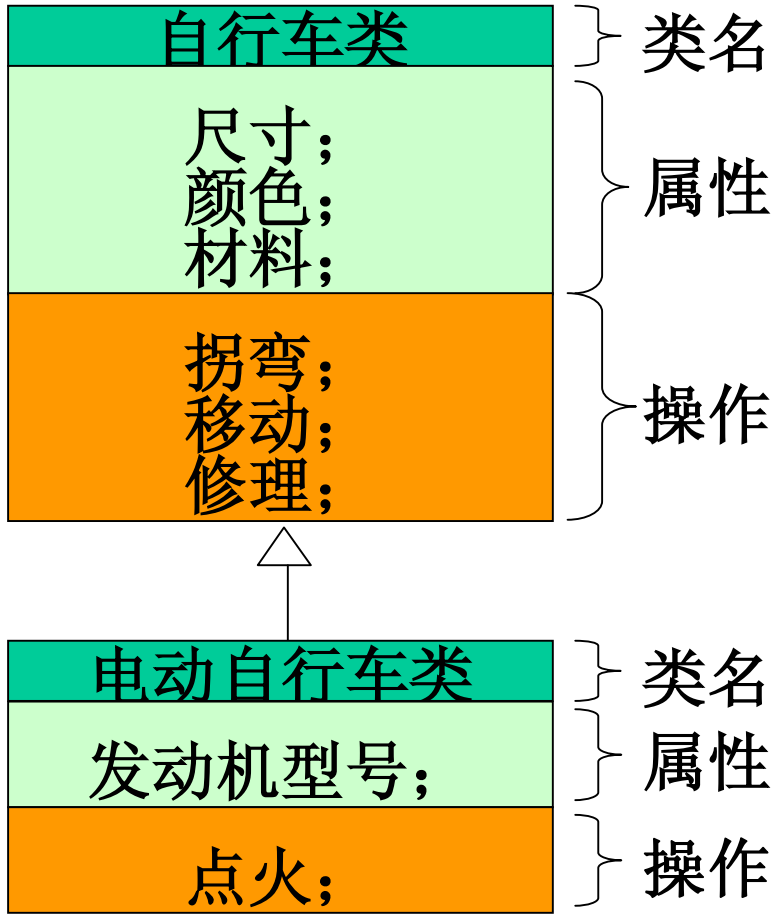
- 世界由对象组成、对象都属于某个类、对象间只能够通过消息联系、类具有层次结构。
- **1．认为世界由各种对象（object）组成，任何事物都是对象，是某个对象类（class）的实例（instance）。**
- **2．把所有对象都划分成各种对象类，每个对象类都定义了一组方法（method），即允许施加于该类对象上的各种操作。**

面向对象方法学的四大要点

- **3．对象之间除了互相传递消息的联系之外，再没有其他联系。**
- **4．对象类将按照“类”、“子类”与“父类”的关系构成一个层次结构的系统。**

面向对象的三大核心特征

- 封装性（Encapsulation）
- 继承性（Inheritance）
- 多态性（Polymorphism）



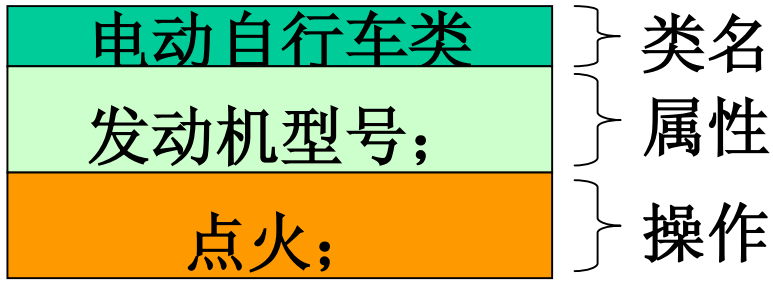
类的继承关系

面向对象的三大核心特征

- 1．封装性：数据和加工处理该数据的方法紧密结合在一起构成黑匣子的整体。

  封装就是将一组数据和这组数据的有关操作组装在一起形成一个对象。

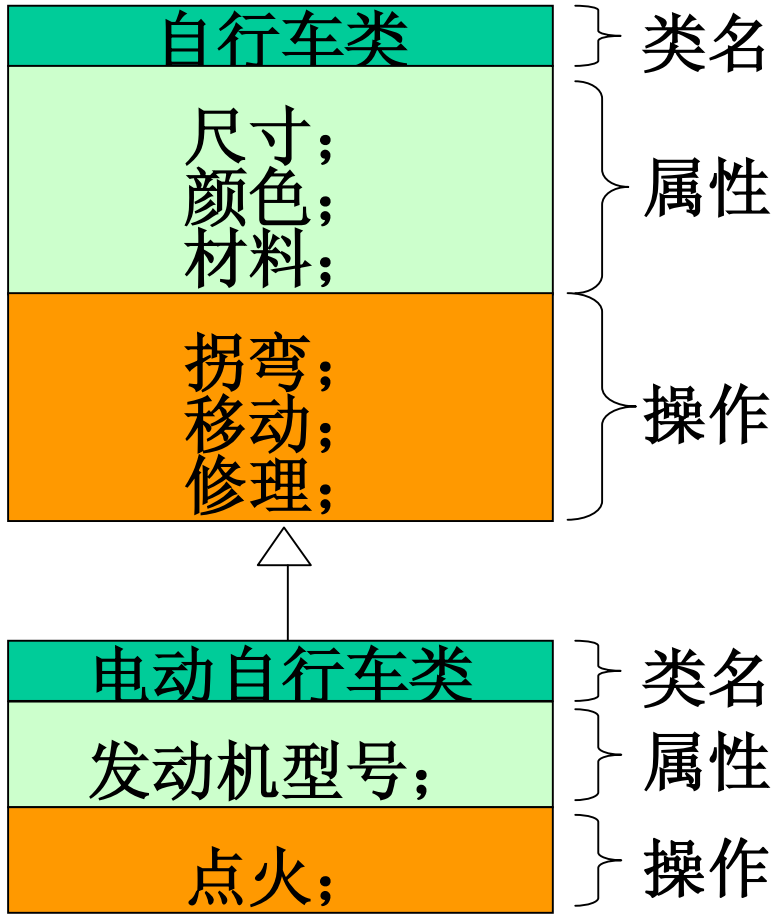
  封装实现了信息隐藏。



- 对私有数据不能执行该对象的成员函数之外的任何其它函数。

面向对象的三大核心特征

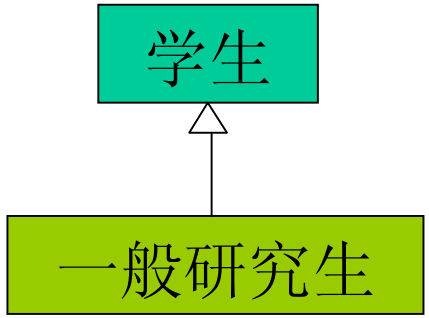
- 2．继承性：一个类直接继承其父类的全部描述（数据和函数）
- 继承具有传递性：
- 若类C继承类B，类B继承类A，则类C继承类A。
- 类实际上继承了类等级中在其上层的全部基类（父类）的所有描述。



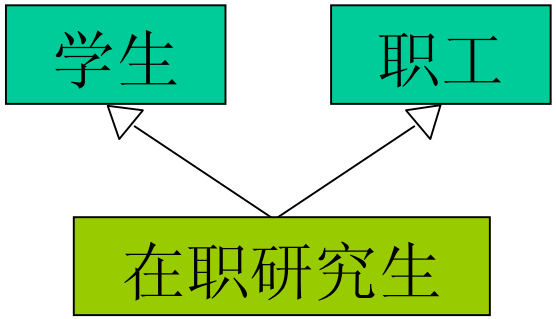
类的继承关系

面向对象的三大核心特征

- 继承方式：
- 单继承——
  一个类只有
  一个父类时
  （树结构）

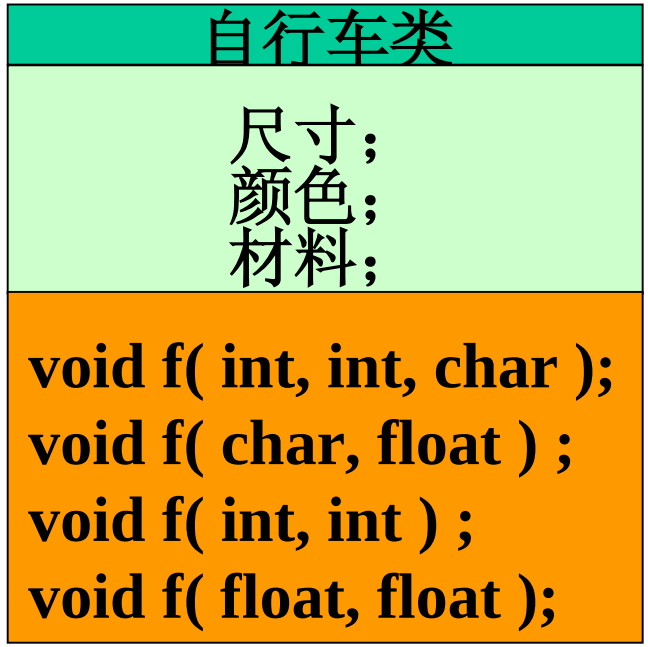


- 多继承——
  一个类可有
  多个父类时
  （图结构）



面向对象的三大核心特征

- 3．多态性：在类等级的各层中共享（公用）一个行为（函数）的名字，而类等级中的每个类却各自按照自己的需要来实现这个行为。
- 实际就是指一个名字具有多种含义。
- 一个名字，多种语义
- 相同界面，多种实现



面向对象的三大核心特征

- 函数重载表达了最简单的多态性
- **void f(int, int, char);**
- **void f(char, float) ;**
- **void f(int, int) ;**
- **void f(float, float);**

- 参数数量、类型、顺序不同，函数体也可以完全不同

面向对象的两个重要原理

- 数据抽象、行为共享
- 1．数据抽象：通过从特定的实例中抽取共同性质以形成一般化的概念的过程。
- ——获取共性
- 强调部分特性（用户所关心的特性）
- 忽略其他特性（用户不关心的特性）
- 用户：只关心做什么，不关心怎么做
- OOP技术比任何一种编程技术都更强调抽象在软件开发中的重要性。

面向对象的两个重要原理

- 2．行为共享：
- （1）实体（模块）的外部接口称为行为。
- （2）行为共享允许多个实体（模块）具有相同的接口集。——接口一致。
- （3）行为共享增强系统的灵活性
- （4）行为共享增强系统的抽象

面向对象与软件IC

软件IC——一种可重用模块。

集成电路。

类概念支持软件IC。

面向对象的思想和方法

必须先研究事物，而后才能研究过程。必须先知道一个事物是什么，而后才能觉察这个事物中所发生的变化。

《路德维希·费尔巴哈和德国古典文学的终结》

恩格斯

面向过程与面向对象程序设计

例1.1：输入任意短语或句子，计算该短语包含多少个字(word)和字符(character)。

注意：输入短语时，必须在字和字之间只空一格。

面向过程与面向对象程序设计

面向过程的方法

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char ch;
    int wdcount,chcount;
    wdcount=1;
    chcount=0;
    printf("***:Please input any phrase…\n");
    while((ch=getche())!='\r')
    {
        chcount++;
```

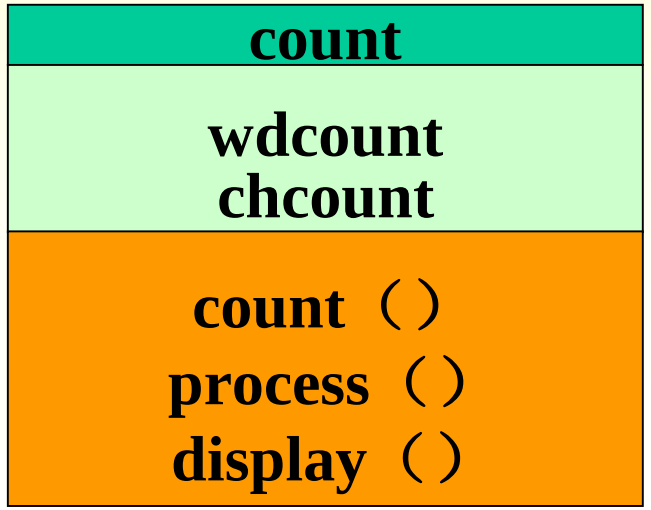
```
        if(ch==' ')
            wdcount++;
    }
    printf("\n***:The number of word is %d\n",wdcount);
    printf("\n***:The number of char is %d\n",chcount);
}
```

面向对象的方法

```
#include <iostream.h>
#include <conio.h>
class count     ←——定义类
{
public:         ←——公有成员
```

面向过程与面向对象程序设计

```
    count();                    ←——构造函数
    void process();             ←——成员函数
    void display();
private:                        ←——私有成员
    int wdcount,chcount;        ←——数据成员
};
count::count()
{
    wdcount=1;
    chcount=0;
}
void count::process()
{
```



count类

```
    char ch;
    cout<<"***:Please input any phrase…"<<endl;
    while((ch=getche())!='\r')
    {
        chcount++;
        if(ch==' ')
            wdcount++;
    }
    cout<<endl;
}
void count::display()
{
    printf("\n***:The number of word is %d\n",wdcount);
```

面向过程与面向对象程序设计

```
 printf("\n***:The number of char is %d\n",chcount);
}
void main()
{
    count A;
    A.process();
    A.display();
}
```

←——定义对象

←——调用公有成员函数

运行结果

```
***Please input any phrase...
I am a teacher
***:The number of word is 4
***:The number of char is 15
```

面向过程与面向对象程序设计

面向过程程序设计

- 是一种数学思维或计算机思维方法，与人们认识世界的方法不同
- 以不稳定的、多变的“过程”和“操作”为中心来构造系统
- 可重用性较差
- 功能抽象和模块性
- 程序＝过程＋调用

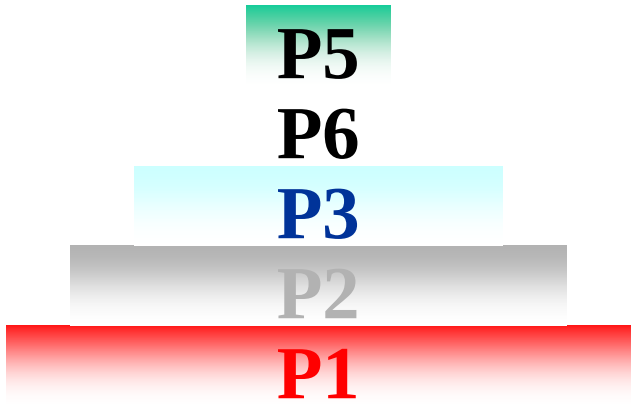
面向对象程序设计

- 是建立在认识方法学基础上的一项技术科学，比较自然地模拟了人类认识客观世界的方式；数据抽象。
- 以相对稳定的“对象”和“数据结构”为中心来构造系统
- 可重用性较好
- 程序＝对象＋消息

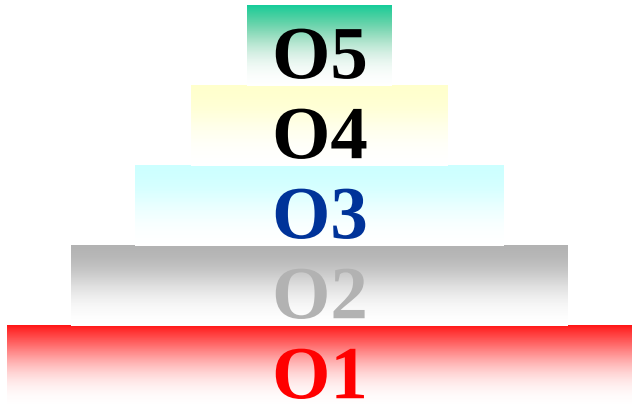
面向过程与面向对象程序设计

面向过程程序设计

功能抽象和模块性，逐步细化。



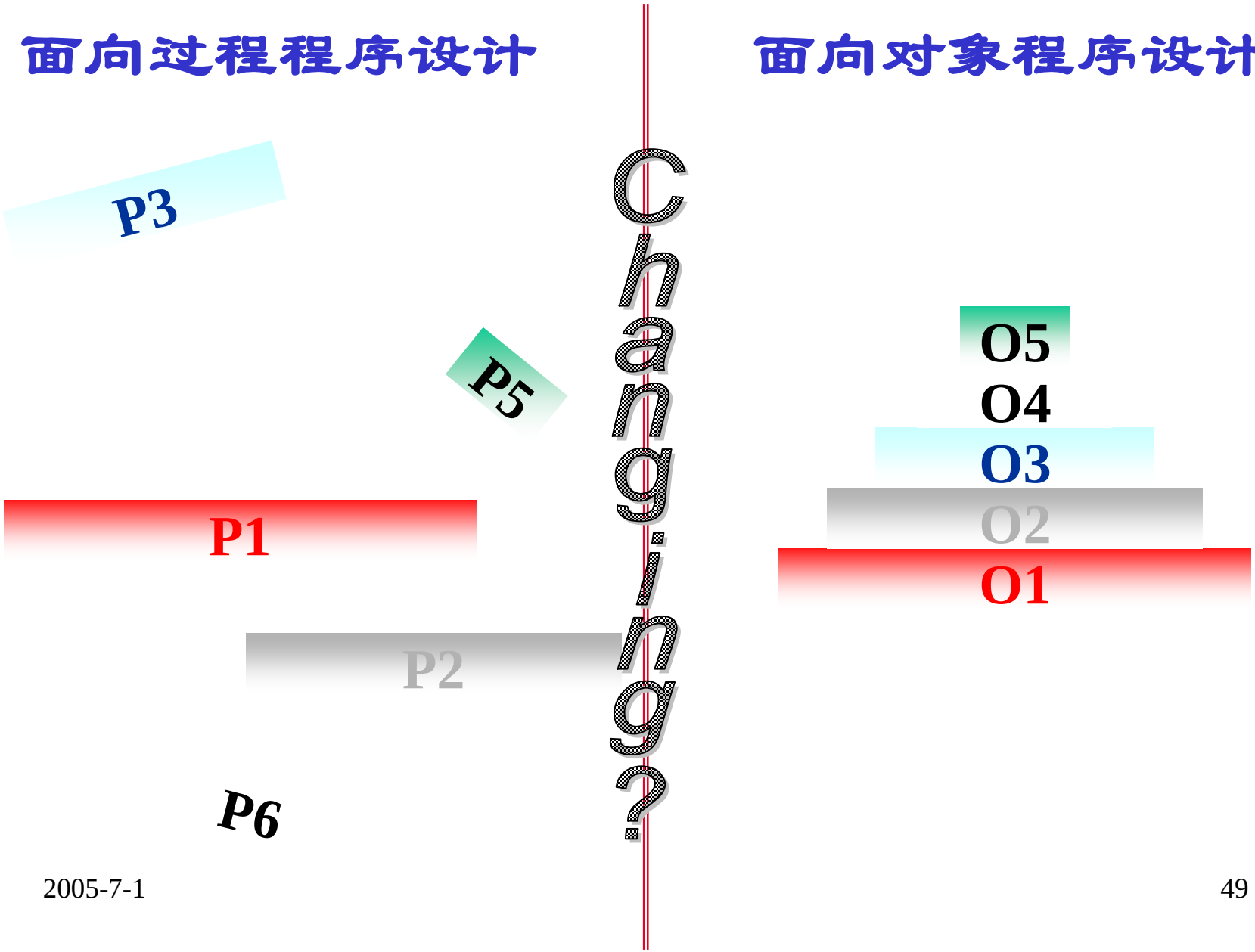
面向对象程序设计



功能抽象和数据抽象。以数据结构为中心。

Changing?

面向过程与面向对象程序设计



按对象方式思维

CRC方法帮助设计者分散责任，直到设计的最后阶段才考虑问题的总体。

Class name（对象取名）；

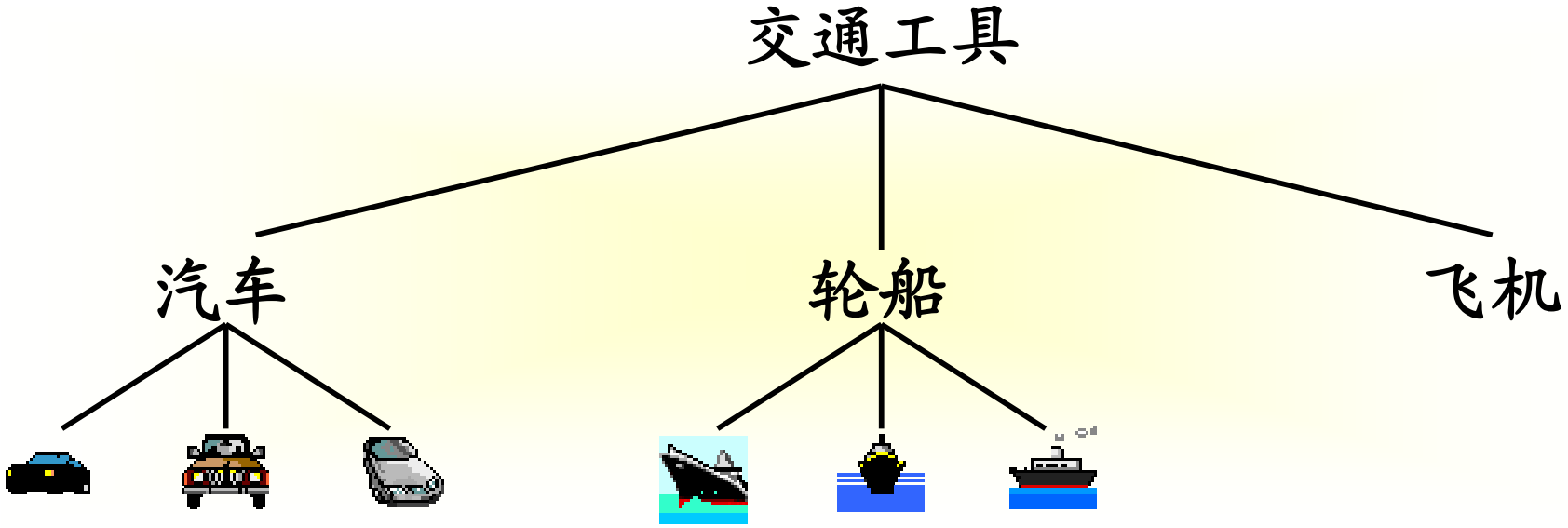
Responsibilities（责任）；

Collaborators（合作者）；

抽象在面向对象中的作用

一、抽象的概念

- 抽象代表着一个对象的本质特征，这个特征将这个对象与所有其他种类的对象区别开来；
- 抽象是通过从特定的实例中抽取共同的性质以形成一般化的概念的过程；
- 抽象具有层次；



抽象在面向对象中的作用

二、面向对象抽象的原理（面向对象计算的本质）

数据抽象、行为共享、进化、确定性

1、数据抽象

为程序员提供了一种对数据和为操作这些数据所需要的算法的抽象；**是面向对象方法的核心**，包括：

类

公有成员

模块化： 构成了面向对象计算的本质；

信息隐藏： 将一个模块的细节部分对用户隐藏起来，用户只能通过一个**受保护的接口**来访问某个模块，而不能直接访问一个模块内部的细节；

抽象在面向对象中的作用

2、行为共享

对象　　公有成员 函数名

- 行为是由实体的外部接口定义的
- 行为共享指许多实体具有相同的接口，可增加系统的灵活性；
- 支持行为共享的方式
  - k 分类与层次分类
  - k 多态与继承

抽象在面向对象中的作用

3、进化

- 需求进化（虚函数）
- 进化式的问题求解（继承的构造函数）

4、确定性

- 确保每个行为项都有一个正确的解释，系统不会因不能响应某一行为而失败；
- 确定性与类型的正确性有关；

面向对象计算的基本特征

面向对象系统的三要素：对象、类和继承；

一、对象

1、概念上

对象是代表着正在创建的系统中的一个实体；

2、实现形式上

对象是一个状态和操作（或方法）的封装体；

3、对象的定义

一个对象具有状态、行为和标识。

状态：对象的状态由这个对象的属性和这些属性的当前值决定。属性是静态的，当前值是动态的；

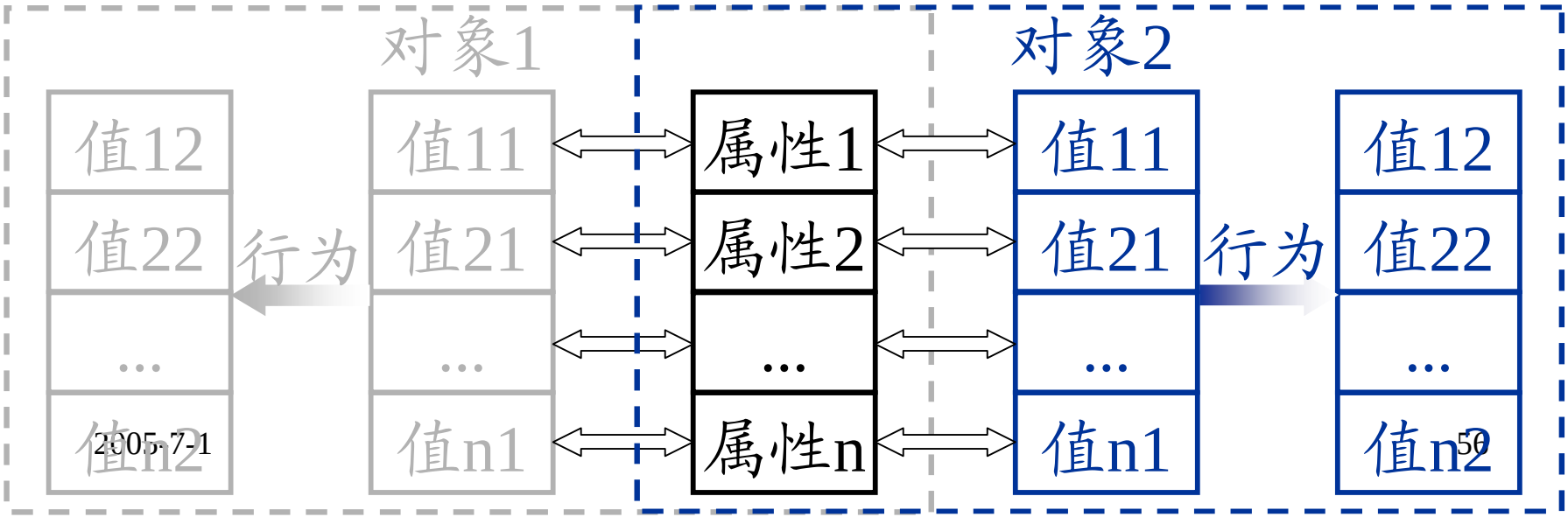
面向对象计算的基本特征

行为：一个对象如何以状态变化和消息传递的形式进行作用和对外界进行反应。

一个对象的行为代表了这个对象的外部可见的和可测试的活动；

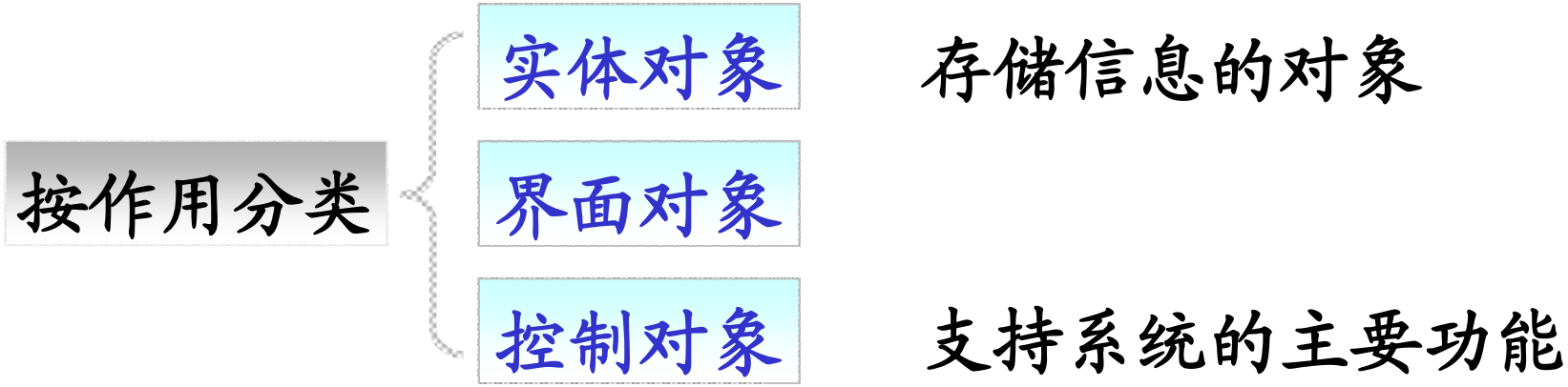
一个对象的状态代表着它的行为的累积结果；

标识：标识是一个对象固有的一种特性，该特性将这个对象与其他对象区别开来；



面向对象计算的基本特征

4、对象的分类



5、对象的确认

- 发现对象：主要是实体对象或界面对象；
- 发明对象：主要是控制对象；

二、类

1、什么是类

面向对象计算的基本特征

类是创建对象的样板，它包含着所创建对象的状态描述和方法的定义。类的完整描述包含了外部接口和内部算法以及数据结构的形式;

2、类是对象的抽象及描述

类中包含生成对象的具体方法，由一个类所创建的对象称为该类的实例;

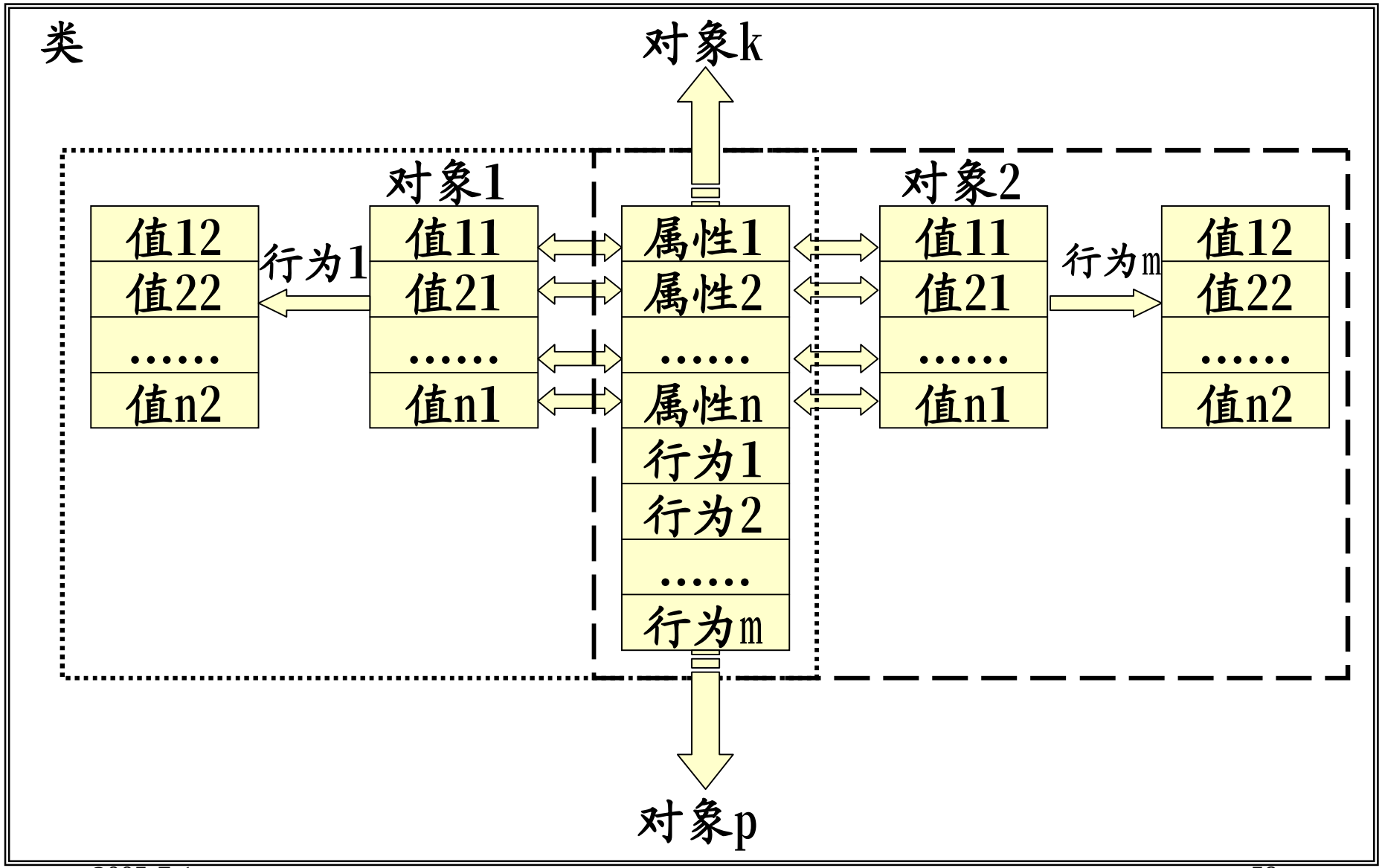
3、类是抽象数据类型的实现

类是所有对象的共同的行为和不同的状态的集合体;

三、继承

继承提供了创建新类的一种方法，它的本质特征是行为共享;

面向对象计算的基本特征



类与对象的关系

C++语言中的抽象支持

- 控制抽象：用于排列任意动作的顺序的一种方法；

  三种语句控制结构：顺序、循环、分支

- 过程抽象（面向过程）：对一组输入数据的一个计算动作和产生的输出结果；

- 数据抽象（面向对象）：类是实现抽象数据类型的工具；

C++对面向对象程序设计方法的支持

- C++支持数据封装（数据抽象）

C++中，类是支持数据封装的工具，对象则是数据封装的实现;

- C++中包含有私有、公有和保护成员
- C++中通过发送消息来处理对象

每个可能的消息对应一个相应的方法，方法通过函数来定义;

- C++中允许友元破坏封装性
- C++中允许函数名和运算符重载
- C++支持继承性
- C++支持动态联编

C++对C语言的改进

- 增加了新的运算符：::，new，delete等；
- 改进了类型系统，增加了安全性；
- 引进了引用概念；
- 允许函数重载，允许设置缺省参数，提高了编程的灵活性；
- 引进了内联函数，提高了程序的效率；
- 可以根据需要随时对变量进行说明；

C++程序的编辑、编译和运行

一、编辑：源文件的扩展名为.cpp

二、编译

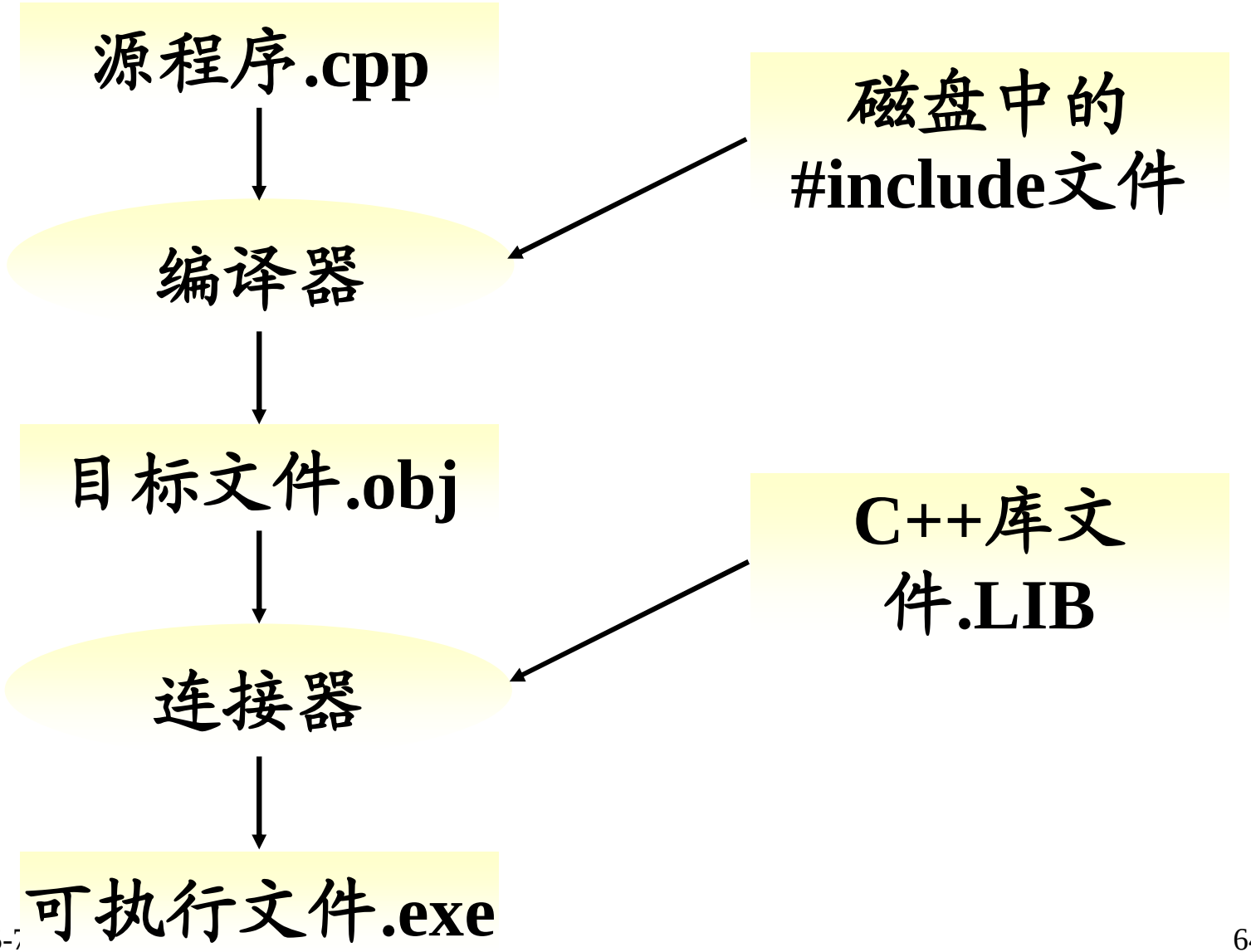
1、预处理过程

2、编译过程
- 词法分析：单词
- 语法分析：构造程序的格式
- 符号表：程序中的各种符号及其属性
- 错误处理程序：
- 生成目标代码：目标文件扩展名.obj

3、连接过程：可执行文件扩展名.exe

三、运行

C++程序的编辑、编译和运行



C++的字符集

- 大小写的英文字母：a～z，A～Z
- 数字字符：0～9
- 特殊字符

空格	！	#	%	^	&	*	_
+	=	-	~	<	>	/	\
,	“	;	.	,	()	[]	

词法记号

1、关键字（保留字）

表1-1　C++的关键字

auto	bool	break	case	catch	char	class
const	const_cast	continue	default	delete	do	double
dynamic_cast	else	enum	explicit	extern	false	float
for	friend	goto	if	inline	int	long
mutable	new	operator	private	protected	public	register
reinterpret_cast		return	short	signed	sizeof	static
static_cast	struct	switch	template	this	throw	true
try	typedef	typeid	typename	union	unsigned	virtual
void	volatile	while				

表1-2　C++的标准保留字

asm	auto	break	case	catch	char
class	const	continue	default	delete	do
double	else	enum	extern	float	for
friend	goto	if	inline	int	long
new	operator	overload	private	protected	public
register	return	short	signed	sizeof	static
struct	switch	this	template	throw	try
typedef	union	unsigned	virtual	void	volatile
while					

词法记号

2、标识符

组成规则：以字母或下划线开始，其后跟零个或多个字母、数字或下划线；

不能以数字开始

正确标识符：Result, DoubleList, _first, first_

错误标识符：1first

标识符的长度任意（受编译器限制）；

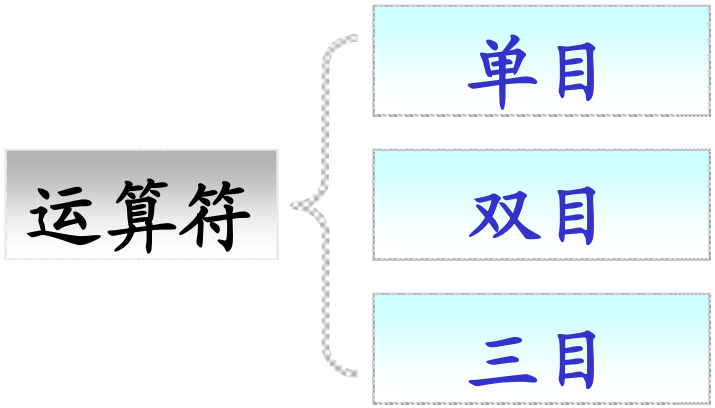
区分字母的大小写；

例如：ADD, Add, add

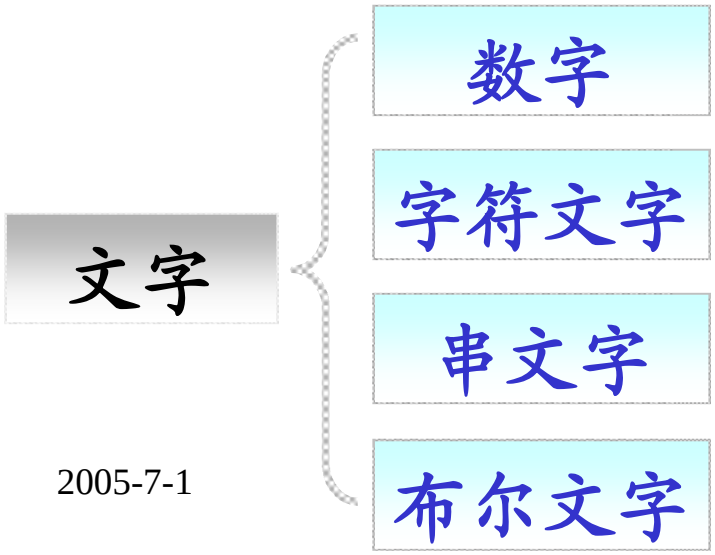
不能使用系统的保留字；

词法记号

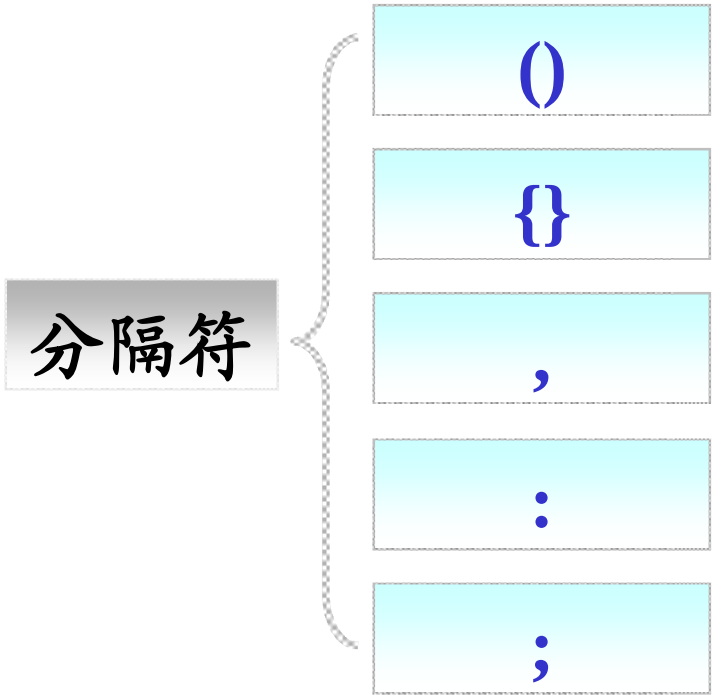
3、运算符



4、各种文字



5、分隔符



空白

一、空白

- 包括：空格、制表符、换行符、注释
- 功能：指示词法记号的开始和结束位置；

二、注释

- /*……*/
- //

C++程序的结构

一、C++示范程序

```
#include <iostream.h>
void main()
{
   cout<<"Hello world! "<<endl;
}
```

I/O流库，提供所有的输入输出操作

cout：流类对象
<<：插入符
提供屏幕输出；

提供键盘输入：
cin：流类对象
>>：提取符
例如：

```
cin>>"Please input two integers:";
cin>>a>>b;
```

endl：换行；

二、C++程序的组成

- 预处理命令
- 输入输出
- 函数
- 语句
- 变量
- 其他