C YUYAN CHENGXU SHEJI

# C语言程序设计

主　编／甘　岚
副主编／雷莉霞　范　萍　刘媛媛

SWJTUP 西南交通大学出版社

# C 语言程序设计

**主　编**　甘　岚
**副主编**　雷莉霞　范　萍　刘媛媛

西南交通大学出版社
·成　都·

图书在版编目（CIP）数据

C语言程序设计／甘岚主编. —成都：西南交通大学出版社，2015.9
ISBN 978-7-5643-4257-9

Ⅰ. ①C… Ⅱ. ①甘… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2015）第204059号

C语言程序设计

主编 甘 岚

| | |
|---|---|
| 责任编辑 | 宋彦博 |
| 特邀编辑 | 穆 丰 |
| 封面设计 | 墨创文化 |
| 出版发行 | 西南交通大学出版社<br>（四川省成都市金牛区交大路146号） |
| 发行部电话 | 028-87600564 028-87600533 |
| 邮政编码 | 610031 |
| 网址 | http://www.xnjdcbs.com |
| 印刷 | 四川森林印务有限责任公司 |
| 成品尺寸 | 185 mm × 260 mm |
| 印张 | 22.5 |
| 字数 | 562千 |
| 版次 | 2015年9月第1版 |
| 印次 | 2015年9月第1次 |
| 书号 | ISBN 978-7-5643-4257-9 |
| 定价 | 48.80元 |

课件咨询电话：028-87600533
图书如有印装质量问题 本社负责退换

# 前　言

C 语言是一门通用计算机编程语言，应用广泛。其设计目标是提供一种能以简易的方式编译、处理低级存储器、产生少量的机器码以及不需要任何运行环境支持便能运行的编程语言。C 语言也很适合搭配汇编语言来使用（往往可以通过内联汇编语言或与汇编语言目标文件一起连接）。对于任何一种操作系统环境，C 函数的 ABI（Application Binary Interface）与汇编语言的子过程（routine/procedure）的 ABI 一定是完全兼容的。尽管 C 语言提供了许多低级处理的功能，但仍然保持着良好跨平台的特性，以一个标准规格写出的 C 语言程序可在许多计算机平台上进行编译，甚至包含一些嵌入式处理器（单片机或称 MCU）以及超级计算机等作业平台。另外，C 语言还扩充了图形、彩色、窗口等功能以及高效的集成开发环境，赢得了广大用户的喜爱。

本书对 C 语言作了全面、详细、系统的介绍，并选择 Visual C++6.0 可视化工具作为本书例题的编译平台，使传统的面向过程的编程语言与现代的面向对象的可视化编译环境有机结合。全书共 11 章，第 1 章介绍了 C 语言的发展、特点、应用领域以及开发工具等内容；第 2 章介绍了常见的数据结构和一些常用算法等内容；第 3 章介绍了 C 语言基本的数据类型、标识符和关键字的概念以及运算符和表达式等基本概念；第 4、5、6 章介绍了 C 语言中的三种常用程序结构，即顺序结构、选择结构和循环结构；第 7 章介绍了数组的基本概念以及一维、二维、多维和字符数组的定义及使用；第 8 章介绍了函数的定义和使用方法以及编译预处理的相关知识；第 9 章介绍了指针的概念及使用方法；第 10 章介绍了构造结构体的方法；第 11 章介绍了文件的使用等内容。

本书注重教材的可读性和实用性，每章的内容均是作者根据多年 C 语言及计算机相关专业课程的教学实践组织而成，学习目标和意义明确，难点和关键知识点阐述详细，并附有大量的图表，以方便读者正确、直观地对问题进行理解。全书精选了大量例题，例题程序由浅入深，强化了知识点、算法、编程方法与技巧，并给出了详细的解释。全部例题已在 Visual C++6.0 平台调试通过，可直接引用。此外，本书还简要介绍了数据结构与算法，使学生能够对程序设计有全面的认识，从大的方向了解程序设计语言的基本概念，从而更易于接受课程的内容，这正好适应了目前我们国家提倡的对大学生进行计算机思维教学的需要。

为了帮助读者更好地学习使用本书，作者还另编写了一本《C 语言程序设计实验指导与习题解答》，作为本书的配套参考书，供读者进行理论练习和上机实践。

本书由华东交通大学甘岚教授任主编，雷莉霞、范萍、刘媛媛任副主编。其中，范萍编写了第 3、4 章，雷莉霞编写了第 5、6、9、10 章，刘媛媛编写了第 8、11 章。甘岚教授编写了第 1、2、7 章，并负责全书统稿工作。

在本书的编写工程中，得到了华东交通大学信息工程学院基础部全体老师的热情支持和指导，在此表示衷心感谢。

由于作者水平有限，加之时间仓促，书中不当之处在所难免，敬请读者批评指正。

编　者

2015 年 5 月

# 目 录

# 第 1 章　C 语言程序设计概述

C 语言从诞生起，就成了主流的程序设计语言，近几年也一直稳居在编程语言排行榜的第一、二名，和 JAVA 语言轮流把持第一的位置。C 语言是一种计算机程序设计语言，既具有高级语言的特点，又具有汇编语言的特点。因此它可以作为工作系统设计语言，编写系统应用程序，也可以作为应用程序设计语言，编写不依赖计算机硬件的应用程序。它的应用范围广泛，具备很强的数据处理能力，不仅仅是在软件开发方面，在各类科研中都有广泛应用。

本章从介绍程序设计语言的基本概念入手，着重介绍了 C 语言的发展与特点、应用领域、C 程序的构成及其运行步骤与运行环境等内容。

## 1.1　程序设计语言概述

计算机主要是由两大部分构成的——硬件和软件。主机、显示器等都属于硬件。仅有硬件的计算机是没有办法使用的，还必须有软件支持。软件又分为：① 系统软件，也就是用户经常用的操作系统，如 Windows XP，Windows 7 等；② 通用软件和应用软件，如 Office 办公软件与 QQ 等。而软件的主体就是程序，因此程序设计语言是计算机科学技术中非常重要的一个部分。

### 1.1.1　程序设计语言的发展与分类

程序设计语言（Program Design Language，PDL），又称编程语言，是一组用来定义计算机程序的语法规则。它是一种被标准化的交流技巧，用来向计算机发出指令。一门计算机语言让程序员能够准确地定义计算机所需要使用的数据，并精确地定义在不同情况下所应当采取的行动。

正如人们交流思想需要使用各种自然语言（如汉语、英语、法语等）一样，人与计算机之间交流信息必须使用人和计算机都能理解的程序设计语言。程序设计语言也叫计算机语言，是一套关键字和语法规则的集合，可用来产生由计算机进行处理和执行的指令。计算机语言也有一个发展过程，最开始是机器语言，也就是用二进制代码表示的语言。那时候编程恐怕是非常痛苦的事，因为程序员要会用 0 和 1 表示一切。后来，人们把一些常用的指令用英语单词表示出来，形成了汇编语言。这个时候编程也是比较痛苦的，因为程序员不仅要记住那些单词的含义，还必须告诉计算机每一步要怎么做，而计算机又是一个“非常笨的东西”，漏掉一个步骤它就罢工。此外，汇编语言的可移植性差，也就是说程序员在这台电脑上写的程序到另一台电脑上可能就不能用了。之后，为了方便软件移植，高级语言诞生了，它不要求程序员掌握计算机的硬件运行，只要写好上层代码，编译软件会将高级语言翻译成汇编语言，然后再将汇编语言转化成计算机语言，从而在计算机中执行。因此，程序员使用高级语言写的代码可以移植到其他计算机执行，而不用考虑计算机硬件的特性。

程序设计语言有很多种，常用的不过十多种，按照程序设计语言与计算机硬件的联系程度将其分为三类，即机器语言、汇编语言和高级语言。前两类依赖于计算机硬件，有时统称为低级语言，而高级语言与计算机硬件联系不密切。

### 1. 机器语言

机器语言是用二进制代码表示的计算机能直接识别和执行的一种机器指令的集合。它是计算机的设计者通过计算机的硬件结构赋予对计算机的操作功能。机器语言具有灵活、直接执行和速度快等特点。不同型号的计算机其机器语言是不相通的，按照一种计算机的机器指令编制的程序，不能在另一种计算机上执行。例如运行在 IBM PC 机上的机器语言程序不能在 51 单片机上运行。

机器指令由操作码和操作数组成，操作码指出要进行什么样的操作，操作数指出完成该操作的数或该数在内存中的地址。

例如，计算 1+2 的机器语言程序如下：

```
10110000 00000001     ; 将 1 存入寄存器 AL 中
00000100 00000010     ; 将 2 与寄存器 AL 中的值相加，结果放在寄存器 AL 中
11110100              ; 停机
```

由此可见，用机器语言编写程序，编程人员要首先熟记所用计算机的全部指令代码和代码的涵义。编写程序时，程序员必须自己处理每条指令和每一数据的存储分配和输入输出，还得记住编程过程中每步所使用的工作单元处在何种状态。这是一件十分烦琐的工作，编写程序花费的时间往往是实际运行时间的几十倍或几百倍。而且，编出的程序全是些 0 和 1 的指令代码，直观性差，难以记忆，还容易出错。

### 2. 汇编语言

为了克服机器语言的缺点，人们采用了有助于记忆的符号（称为指令助记符）与符号地址来代替器指令中的操作码和操作数。指令助记符是一些有意义的英文单词的缩写和符号，如用 ADD（Addition）表示加法，用 SUB（Subtract）表示减法，用 MOV（Move）表示数据的传送，等等。而操作数可以直接用十进制数书写，地址码可以用寄存器名、存储单元的符号地址等表示。这种表示计算机指令的语言称为汇编语言。

例如，上述计算 1+2 的汇编语言程序如下：

```
MOV AL，1             ; 将 1 存入寄存器 AL 中
ADD AL，2             ; 将 2 与寄存器 AL 中的值相加，结果放在寄存器 AL 中
HLT                   ; 停机
```

由此可见，汇编语言克服了机器语言难读难改的缺点，同时保持了占存储空间小、执行速度快的优点，因此许多系统软件的核心部分仍采用汇编语言编制。但是，汇编语言仍是一种面向机器的语言，每条汇编命令都一一对应于机器指令，而不同的计算机在指令长度、寻址方式、寄存器数目等方面都不一样，这使得汇编语言具有通用性差，可读性也差的特点。

### 3. 高级语言

所谓高级语言就是更接近自然语言、数学语言的程序设计语言。它是面向应用的计算机

语言，与具体的机器无关。其优点是符合人类的叙述问题的习惯，而且简单易学。高级语言与计算机的硬件结构及指令系统无关，它有更强的表达能力，可方便地表示数据的运算和程序的控制结构，能更好地描述各种算法，而且容易学习和掌握。但高级语言编译生成的程序代码一般比用汇编程序语言设计的程序代码要长，执行的速度也慢。

高级语言并不特指某一种具体的语言，而是包括很多编程语言，如目前流行的 JAVA，C，C++，C#，PASCAL，PYTHON，LISP，PROLOG，FOXPRO 等，这些语言的语法、命令格式都不相同。

例如，上述计算 1+2 的 BASIC 语言程序如下

```
A=1+2          ，将 1 加 2 的结果存入变量 A 中
PRINT A        ，输出 A 的值
END            ，程序结束
```

这个程序和我们平时的数学思维是相似的，非常直观易懂，容易记忆。

### 1.1.2　程序设计的过程

计算机程序设计的过程包括问题定义、算法设计、程序设计以及调试运行。整个开发过程都要编制相应的文档，以便管理。

#### 1. 问题定义

在计算机能够理解一些抽象的名词并做出一些智能的反应之前，必须要对交给计算机的任务做出定义，并最终翻译成计算机能识别的语言。问题定义的方法很多（对此在软件工程的需求分析中会有更多解释，包括描述方法和工具），但一般包括三个部分：输入、输出和处理。

#### 2. 算法设计

问题定义确定了未来程序的输入、输出以及处理，但并没有具体说明处理的步骤，而算法则是对解决问题步骤的描述。

#### 3. 程序设计

问题定义和算法设计已经为程序设计规划好了蓝本，下一步就是用真正的计算机语言表达了。不同的语言写出的程序有时会有较大的差别。

#### 4. 调试运行

程序编写可以在计算机上进行，也可以在纸张上进行，但最终要让计算机来运行则必须输入到计算机中，并经过调试，以便找出错误，然后才能正确地运行。

#### 5. 文　档

对于微小的程序来说，有没有文档显得并不怎么重要，但对于一个需要有多人合作，并且开发、维护时间较长的软件来说，文档就是至关重要的。文档用于记录程序设计的算法、实现以及修改的过程，保证程序的可读性和可维护性。程序中的注释就是一种很好的文档。

## 1.2 程序设计方法

### 1.2.1 结构化程序设计

#### 1. 结构化程序设计的出现

在早期由于计算机存储器容量非常小，人们设计程序时首先考虑的问题是如何减少存储器开销，硬件的限制不容许人们考虑如何组织数据与逻辑，为此程序员使用各种技巧和手段编写高效的程序。其中显著的特点是程序中大量使用 GOTO 语句，使得程序结构混乱、可读性差、可维护性差以及通用性差。但是，随着大容量存储器的出现及计算机技术的广泛应用，这种方式的程序编写越来越困难，因为程序的大小以算术级数递增，而程序的逻辑控制难度则以几何级数递增，人们不得不重新考虑程序设计的方法。

结构化程序设计是进行以模块功能和处理过程设计为主的详细设计的基本原则，其概念最早由荷兰科学家 E. W. Dijkstra 提出，它的主要观点是采用自顶向下、逐步求精的程序设计方法；使用三种基本控制结构构造程序，任何程序都可由顺序、选择、循环三种基本控制结构构造而成。

结构化程序设计曾被称为软件发展中的第三个里程碑。该方法的要点是：

（1）主张使用顺序、选择、循环三种基本结构来嵌套连接成具有复杂层次的“结构化程序”，严格控制 GOTO 语句的使用。用这样的方法编出的程序在结构上具有以下效果：

① 以控制结构为单位，只有一个入口，一个出口，所以程序员能独立地理解这一部分。

② 能够以控制结构为单位，从上到下顺序地阅读程序文本。

③ 由于程序的静态描述与执行时的控制流程容易对应，所以能够方便正确地理解程序的动作。

（2）“自顶而下，逐步求精”的设计思想，其出发点是从问题的总体目标开始，抽象低层的细节，先专心构造高层的结构，然后再一层一层地分解和细化。这使设计者能把握主题，高屋建瓴，避免一开始就陷入复杂的细节中，使复杂的设计过程变得简单明了，过程的结果也容易做到正确可靠。

（3）“独立功能，单入口单出口”的模块结构，减少模块的相互联系使模块可作为插件或积木使用，降低了程序的复杂性，提高可靠性。编写程序时，所有模块的功能通过相应的子程序（函数或过程）的代码来实现。程序的主体是子程序层次库，它与功能模块的抽象层次相对应，该编码原则使得程序流程简洁、清晰以及可读性强。

（4）主程序员组。主程序员组的组织形式指开发程序的人员组织方式应采用由一个主程序员（负责全部技术活动）、一个后备程序员（协调、支持主程序员）和一个程序管理员（负责事务性工作，如收集、记录数据，文档资料管理等）为核心，再加上一些专家（如通信专家、数据库专家）、其他技术人员组成小组。

其中，（1）、（2）是解决程序结构规范化问题，（3）是解决将大划小，将难化简的求解方法问题，（4）是解决软件开发的人员组织结构问题。

#### 2. 程序控制结构

按照结构化程序设计的观点，任何算法功能都可以由以下三种基本结构实现：顺序结构、选择结构和循环结构。

1）顺序结构

顺序结构是最自然的程序结构方法，由前到后执行，如图 1-1 所示。由图中可以看出这两个程序模块是顺序执行的，即首先执行程序模块 1，然后执行程序模块 2。从逻辑上看，顺序结构中的两处程序模块可以合并成一个新的程序模块。通过这种方法，可以将许多顺序执行的语句合并成一个比较大的程序模块。

2）选择结构

选择结构如图 1-2 所示。从图中可以看出，根据逻辑条件成立与否，分别选择执行程序模块 1 或者程序模块 2。虽然选择结构比顺序结构稍微复杂一些，但是仍然可以将其整体作为一个新的程序模块：一个入口（从顶部进入模块开始判断），一个出口（无论执行了程序模块 1 还是程序模块 2，都应从选择结构框的底部出来）。

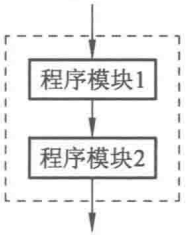


图 1-1　顺序结构

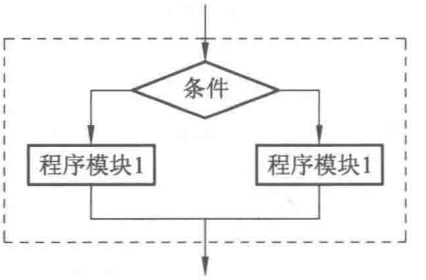


图 1-2　选择结构

3）循环结构

循环结构有两种格式，如图 1-3 所示。在图 1-3（a）中，进入循环结构后首先判断条件是否成立，如果成立则执行<程序模块>，反之则退出循环结构。在图 1-3（b）中，执行完程序模块后再去判断条件，如果条件仍然成立则再次执行内嵌的程序模块，循环反复，直至条件不成立时退出循环结构。与顺序结构和选择结构相同，循环结构也可以抽象为一个新的模块。

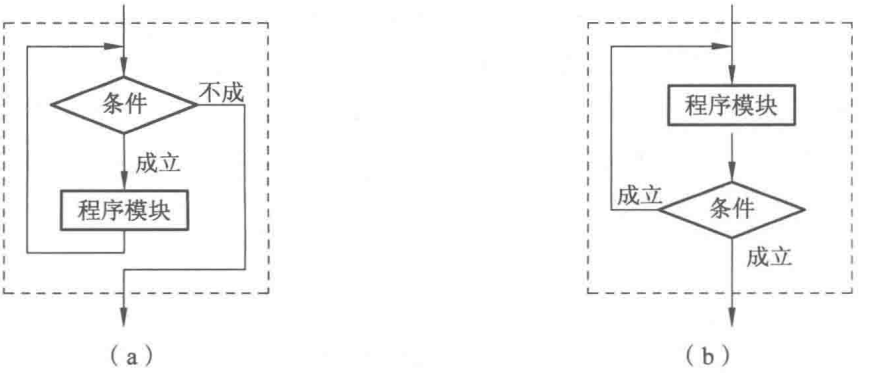


图 1-3　两种循环结构控制结构

### 3. 设计方法

1）自顶向下

程序设计时，应先考虑总体，后考虑细节；先考虑全局目标，后考虑局部目标。不要一

开始就追求众多的细节，应先从最上层总目标开始设计，逐步使问题具体化。

2）逐步细化

对复杂问题，应设计一些子目标作为过渡，逐步细化。

3）模块化

一个复杂问题，肯定是由若干稍简单的问题构成。模块化是把程序要解决的总目标分解为子目标，再进一步分解为具体的小目标，把每一个小目标称为一个模块。

4. 限制使用 GOTO 语句

结构化程序设计方法起源于对 GOTO 语句的认识和争论。肯定的结论是，在块和进程的非正常出口处往往需要用 GOTO 语句，使用 GOTO 语句会使程序执行效率较高；在合成程序目标时，GOTO 语句往往是有用的，如返回语句用 GOTO。否定的结论是，GOTO 语句是有害的，是造成程序混乱的祸根，程序的质量与 GOTO 语句的数量呈反比，应该在所有高级程序设计语言中取消 GOTO 语句。取消 GOTO 语句后，程序易于理解、排错、容易维护，容易进行正确性证明。作为争论的结论，1974 年 Knuth 发表了令人信服的总结，并证实了：

（1）GOTO 语句确实有害，应当尽量避免。

（2）完全避免使用 GOTO 语句也并非是个明智的方法，有些地方使用 GOTO 语句，会使程序流程更清楚、效率更高。

（3）争论的焦点不应该放在是否取消 GOTO 语句上，而应该放在用什么样的程序结构上。其中最关键的是，应在以提高程序清晰性为目标的结构化方法中限制使用 GOTO 语句。

### 1.2.2 面向对象程序设计

虽然结构化程序设计方法具有很多的优点，但它仍是一种面向过程的程序设计方法，即把数据和处理数据的过程分离为相互独立的实体。当数据结构改变时，所有相关的处理过程都要进行相应的修改，每一种相对于老问题的新方法都要带来额外的开销，程序的可重用性差。

同时由于图形用户界面的应用，程序运行由顺序运行演变为事件驱动，使得软件使用起来越来越方便，但开发起来却越来越困难，对这种软件的功能很难用过程来描述和实现，使用面向过程的方法来开发和维护都将非常困难。

由于上述缺陷已不能满足现代化软件开发的要求，一种全新的软件开发技术应运而生，这就是面向对象程序设计（Object Oriented Programming，OOP）。面向对象程序设计方法是于 20 世纪 60 年代后期首次提出的，20 世纪 80 年代开始走向商用。

#### 1. 面向对象的基本概念

1）对象

对象是人们要进行研究的任何事物，从最简单的整数到复杂的飞机等均可看作对象，它不仅能表示具体的事物，还能表示抽象的规则、计划或事件。

2）对象的状态和行为

对象具有状态，一个对象用数据值来描述它的状态。对象还有操作，用于改变对象的状态，对象及其操作就是对象的行为。对象实现了数据和操作的结合，使数据和操作封装于对象的统一体中。

3）类

具有相同或相似性质的对象的抽象就是类。因此，对象的抽象是类，类的具体化就是对象，也可以说类的实例是对象。

类具有属性，它是对象的状态的抽象，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来描述。

4）类的结构

在客观世界中有若干类，这些类之间有一定的结构关系。通常有两种主要的结构关系，即一般——具体、整体——部分结构关系。

（1）一般—具体结构称为分类结构，也可以说是“或”关系，或者是“is-a”关系。

（2）整体—部分结构称为组装结构，它们之间的关系是一种“与”关系，或者是“has-a”关系。

5）消息和方法

对象之间进行通信的结构叫作消息。在对象的操作中，当一个消息发送给某个对象时，消息包含接收对象去执行某种操作的信息。发送一条消息至少要包括说明接受消息的对象名、发送给该对象的消息名（即对象名、方法名）。一般还要对参数加以说明，参数可以是认识该消息的对象所知道的变量名，或者是所有对象都知道的全局变量名。

类中操作的实现过程叫作方法，一个方法由方法名、参数、方法体构成。抽象与实例的对应关系如图 1-4 所示。

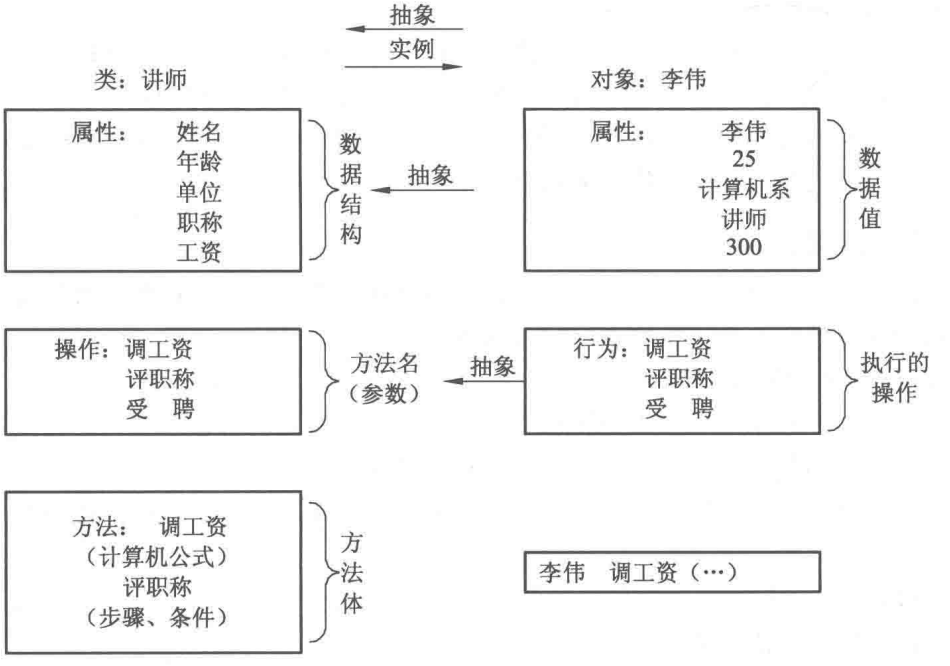


图 1-4　对象、类和消息传递

2. 面向对象的特点

（1）抽象。类的定义中明确指出类是一组具有内部状态和运动规律对象的抽象。抽象是一种从一般的观点看待事物的方法，它要求我们集中于事物的本质特征（内部状态和运动规