C++ 程序设计与实例

刘天海等 编著

湖北出版社

C++程序设计与实例

刘天海　高文海　胡新亮
周克昌　朱文林　张　昱
刘耀炜　吴培稚　曹永建
沈建华　修济刚　牟其铎　编著

测　绘　出　版　社

(京)新登字 065 号

内 容 提 要

本书是由使用C++语言进行大型软件开发的人员编写。作者通过切身实践由浅入深地阐明了C++的有关概念和在程序设计中的实现方法以及在编程中常见问题的处理办法，同时论述了大型软件系统研制中的程序控制与调度、内存管理与覆盖等实用技术和程序调试方法，并以实例形式介绍了软件系统研制中的文件管理、各种窗口与菜单的制作、屏幕绘图、电子幻灯制作、C++语言与汇编语言的接口等。另外还给出经调试通过的部分常用算法程序，可供读者参考选用。本书面向具有C或C++语言基础的广大读者，既可作为学习C++语言的教材，也可作为进行C++程序设计和大型软件系统开发的参考书。

C++程序设计与实例
刘天海等 编著
*
测绘出版社出版发行
责任编辑：易杰军
北京通县财联印刷厂印刷
新华书店总店科技发行所经销
*
开本 787×1092 1/16 · 印张 14.5 · 字数 341 千字
1993 年 12 月第一版 · 1993 年 12 月第一次印刷
印数 0001—3000 册 · 定价 18.50 元
ISBN 7—5030—0680—3/TP · 11

前　言

面向对象程序设计方法是一种新的软件设计思维方法，它突破了传统的结构化程序设计方法的局限性，使人们可以模仿建立现实世界模型的方式进行程序设计，从而使所开发出的应用软件或系统软件更易于维护、扩充和移植。

C++语言全面支持面向对象的程序设计方法，也支持传统的结构化程序设计方法。进入90年代以来，C++语言已逐渐成为研制大型应用软件采用的主流语言，随之也出版了大量的C++语言参考书。本书并非现有C++资料的集合或编译，而是在通过应用Borland C++语言成功地研制出"中国地震分析预报软件系统(SSEPC)"之后，结合作者的体会并参考有关文献而编写的。SSEPC涉及到11门专业学科，集中了我国地震分析预报和地震科学研究专家20多年之经验，涉及内容广且处理数据量大，在各方面都具有相对先进性，并在操作维护上具有灵活性。

本书第一章介绍面向对象程序设计方法的基本思想和特点，并结合SSEPC实例介绍了其应用。第二章介绍C++的封装、继承和多态性等重要概念及作者的编程经验。第三章重点介绍了操作系统对内存的管理、扩充内存和扩展内存的使用、C++语言的内存管理细节以及C++的内存覆盖技术；还结合实例讨论了在程序中如何实现栈空间和堆空间的安全使用问题，以避免应用软件系统的崩溃。第四章介绍了在C++中调用汇编语言子程序和在汇编语言中调用C++函数的方法以及注意问题。第五章介绍了在集成环境下程序的调试经验。第六章讨论了应用软件的总控技术和软件发行包的制作，重点介绍了各种窗口和菜单的制作技术，最后介绍了用C++语言实现文件名和目录名的搜索、排序、显示和多文件名的光棒选择等应用程序。第七章介绍了用C++语言实现的屏幕绘图技术和图形压缩存储及电子幻灯制作技术。第八章给出了用C++语言编写的常用算法程序。

编写本书的目的旨在帮助读者在使用C++语言进行应用程序研制和大型软件系统开发方面少走弯路。为方便读者理解C++语言的有关概念和掌握其程序设计的技巧，书中常以实例加以介绍说明。对于较长的程序段，将以磁盘的形式由测绘出版社另行出版，并与本书相配套，可供软件研制开发人员直接选用。

由于我们水平有限，错误之处恳请读者批评指正。

作　者

1993年8月

目 录

第一章　面向对象程序设计方法概述

本章主要阐述面向对象程序设计的概念和特点，并以中国地震分析预报软件系统(SSEPC)为实例，介绍使用面向对象程序设计方法的一些体会。

§1.1　OOP 方法的特点

现行结构化程序设计方法强调模块功能化，提出自顶向下、逐步分解、搭积木式的程序设计原则。但是模块中只突出了实现功能的操作方法(函数)，而被操作的数据(变量)处于实现功能的从属地位，即数据与操作相分离。因此，在工程化的大型软件设计中，要对每个模块中的结构名、变量名、函数名等做严格规定，程序员之间要经常沟通，以防止自己定义的结构或函数被别人非法使用或修改；另外，许多重要的函数或模块功能的实现与一些关键的数据结构有关。当一个或多个这样的数据结构发生了变化，使得许多函数和模块必须重写，有时可能导致整个软件系统的结构做重大改变。随着软件规模和复杂性的增长，这种缺陷就日益明显。事实上软件开发过程中的系统调试和软件推广使用过程中的维护，将随着软件规模的不断增长而变得更为复杂和困难。目前软件的应用领域越来越广泛，处理的数据量和各种数据类型的复杂性已接近或达到结构化程序设计方法无法管理的程度。

为了克服现行结构化程序设计在实际应用中的局限性，80 年代初出现了一种新的高效的计算机程序设计方法——面向对象的程序设计(ObjectOrientedProgramming，简称 OOP)。这种方法引入了新的概念和思维方式，使软件人员在程序设计中能够模仿建立世界模型的方法，对系统的复杂性进行概括、抽象和分类，使软件设计与实现形成一个由抽象到具体、由简单到复杂这样一个循序渐进的过程，从而可以解决大型软件研制中存在的效率低、质量难以保证、调试复杂、维护困难等一系列问题。

当然，OOP 并不是完全抛弃现行结构化程序设计方法，而是吸取了结构化程序设计的优点，加入了更能直接地、概括地描述事物的方法，使得模块化程度更高、模块之间的接口更简洁，更具有通用性更有利于软件系统的开发、维护和扩充。

1.1.1　OOP 方法的对象和类的概念

1. 对象(Object)

OOP 方法中的对象有以下几个特点：

①对象是程序设计所涉及的事物。从最简单的数据求和到复杂的数据处理过程以及更大的某一项任务，都可以看作对象。对象概念有很强的描述与表达能力，现实世界中的事物都可以抽象为逻辑世界中的一个对象。

②对象是程序设计中包含数据与操作的相对独立的模块化单元。首先，对象具有状态，

这种状态是通过数据来描述的；同时，对象的状态又是可以改变的，改变对象的状态是通过一系列对数据的操作来完成。从动态的观点来看，数据及其操作本身就是千变万化的。现行的结构化程序设计方法将数据与操作（函数或过程）相分离，由于其操作为大家共享而又无特殊性，数据只是从属于操作的符号。因此，当需要解决的问题极其复杂时，就很难对数据和操作及它们之间的关系进行准确有效的描述。而 OOP 将数据与操作同等看待，将它们封装在“对象”这个统一体内，并提供一种机制，使对象内的私有数据有其私有操作，从而可灵活而有针对性地描述具体对象的独特行为，实现对现实世界的复杂事物直接、准确的模拟。

③对象具有唯一的标识功能。在众多定义的对象中，这种识别功能可以识别出特指的对象，并且在对象被建立直至消失的整个“生命期”内，其识别符保持不变。随着对象的删除，对象的一切识别特征将随之消失。

2. 类（Class）

类是 OOP 中的另一个重要概念，其定义如下：类是对具有相同语义性质（具体指相同的数据结构、操作特性、约束规则）的一组对象的抽象性描述。类实现对共同语义特性的一组对象的抽象，抽取这组对象的共性，并作统一说明。而对象则是类中具体描述的一个实例。这样，就大大简化了对象的定义。这种高度抽象的特性，正是面向对象程序设计的基本特征。

类不仅可表达对结构化数据的抽象，而且可表达对非结构化数据（分数维等）的抽象。面向对象程序设计方法的这种将数据结构上的抽象与功能上的抽象有机结合的能力，使得用户可以根据应用的需要灵活地定义各种复杂的数据类型，以便充分表达现实世界的事物，而不必象结构化编程方法那样，要经过多种转换等处理。

1.1.2 OOP 的重要特性

1. 封装性

为了提高软件的可靠性，要求软件设计模块化。正如集成电路的模块化提高了计算机硬件的可靠性一样，封装性正是确保软件具有优良的模块结构、保证软件可靠性的有效途径。

封装性的具体含义是程序模块内部的高内聚和其外部的低偶和。高内聚表现在软件模块的内部应有明确的范围，内部的实现受到完全保护；低偶和则表现为软件模块应有清楚明了和简单的外部边界，并应具有友好界面（接口）以方便地实现软件模块之间的相互连接。

首先，对象是封装的最基本的单位。在一个对象中，数据和代码可以是这个对象私有的，不能被对象外的部分直接访问，这样一来，对象就提供了一种高级保护以防止程序的无关部分偶然地修改或错误地使用对象中的私有部分。其次，类是具有封装性的程序模块。类除可以使数据或操作成为公有的（public）外，也可以使数据或操作成为私有的（private）或保护的（protected），体现了类的高内聚特点；类所定义的数据，通过操作访问，类与类之间通过对象发送消息（Message）来相互通讯和交互，体现了类之间的低偶和特性。

OOP 的封装性，使程序员可以编制出优良的模块化程序，在程序的调试中，许多错误可以很快追溯到一个特定的类，调试和维护变得容易和简单。

2. 继承性

继承是衡量一种语言是否真正体现面向对象程序设计思想的重要标志。那些基于对象或基于类而非面向对象的编程语言都不具有这一特性。

继承性体现并扩充了类的共享机制，具体表现在：

①对象自动继承类的全部语义。也就是说，在某个类被定义之后，只要声明一个对象是该类的实例，这个对象就能共享类定义中的数据和操作，即自动地继承该类的全部语义，而无需再作重复说明。

②类层次的继承机制是 OOP 继承性的精华。所谓类层次（或类等级）的继承是指类层次结构中的派生类能自动地继承基类的全部语义。如果类层次具有两层以上，这种继承还具有传递性，即最低层的派生类可自动地继承其上层、直至顶层基类的全部语义。这种多层继承性，就是 OOP 的多层继承机制。

③除了上述的多层继承外，某些 OOP 语言（如 C^{++} 语言）还支持派生类继承多个基类语义的多重继承。多重继承可以看作是横向继承。

OOP 的继承性，具有以下几个显著作用：

①进一步扩充并增强 OOP 中类的共享机制。在类层次结构中，基类的语义不仅可由本类的对象共享，而且可以由其派生类、派生类的派生类所具有的对象共享。类的共享范围的扩大，可显著地减小创建类与新对象的工作量，并有效地节约了存储空间。

②提高代码的可重用性和软件的可靠性。这种重用性主要是通过继承基类中的方法来实现的。

③高质量、高效能地实施软件工程方案。将工具类库以目标代码的形式发给其他程序员，其他程序员不访问源代码就可以从类库中派生出新的类。这样，就可以按照分散实施、集中管理的办法开发大型应用软件。

3. *多态性*

多态性一词来源于希腊语“具有多种形态”。多态性有时又称为“同一接口，多种方法”，即用同一个名字定义功能相近的不同操作。这意味着既可使操作的特定动作互有区别，又可以用同样的接口访问。

多态性的另一个特征是体现在程序中方法调用的编译连接与运行连接的不同作用上。编译连接就是程序编译时就将对象与方法的代码连接起来，其优点是执行速度快，所需内存小，缺点是灵活性差。运行连接意味着在程序运行时才决定对象与哪一个方法连接起来，其优点是灵活性强，有利于建立类库，便于系统扩充，而对运行速度的影响很小。

§1.2 OOP 方法设计技术

1.2.1 面向数据流分析与面向对象分析

面向数据流分析法是从上而下分析法。它象一棵树，树上一个节点的变动都将引起下面所有节点的变动。这种伤筋动骨的变动，无疑是程序员最头痛的事。

面向对象的分析设计技术以对象概念为中心，在数据上支持抽象和层次关系，在动作上与属性一起被归类封装，类内的增删只影响内部，在分析中既可以自上而下，又可以自下而上的构造，或者两者结合进行。面向对象分析法更接近外部世界。

面向数据流分析方法与面向对象分析方法不同之处很多，这里只介绍最基本的不同，即思维方式的不同。

数据流分析方法把事物抽象成一系列逻辑加工单元，加工单元接受数据流，使之变成输出流，因此数据流是联系加工单元的纽带。不难想象，对于一个复杂系统，数据流图将异常复杂，软件设计将是厚厚的一本书。

面向对象分析方法试图用类来模拟客观世界，它的思维方法是个别到一般，一般到个别。例如分析一系列具体的狗(一群对象)，抽象出狗的属性：眼睛，四条腿，尾巴……，做的动作有吃东西，走路，摇尾巴，叫喊……。这就形成了狗类，这就是个别到一般。现在需要狗叫喊，没有具体的狗是永远叫不出来的，因此要确定一只具体的狗——对象，再让它叫——调用行为，这就是一般到个别。如果狗的尾巴与我们研究的问题无关，那么可以在狗属性内排除尾巴，在它的动作中排除与尾巴有关的动作，即我们模拟的客观世界是根据问题需要抽取其行为和属性的。

因此类存在于客观世界中，又有别于客观世界。这种个别到一般，一般到个别的分析方法是一般科学研究和人类在抽象思维中采用的基本方法。类的属性和行为统称为成员：属性亦称为数据，数据结构，数据成员；动作亦称为行为，操作，代码，成员函数。数据和代码在类中平起平坐，相互依赖谁也离不了谁。类有别于其它类，类是封装的。

比较两种方法，面向对象设计和分析，灵活，方便，更适合于大型软件的研制开发。当然，面向对象程序设计也是一种结构化设计，它和传统的结构化设计有较大差别。

1.2.2 类的模块化

类本来就具有模块性，为了更深一层的程序开发，可以把每个类或一组相关的类说明包装在一起，放在一个单独的头文件中，而其它非内部成员函数的定义，放入一个单独的源文件中，可以实现由多个文件组成的程序。同时把类进一步地模块化，程序员可以以目标码的形式把类分发给其他程序员，其他程序员可以不用访问你的源代码就可以从你的类中派生出新的特定的类。这种多文件组成程序和继承(不透明的)，无疑是大型软件工程求之不得的优良特性。

1.2.3 用类模拟客观世界

用计算机来解决客观世界中提出的各种各样的问题时，总要给问题提供一个模型，建立模型的方法多种多样，直接模拟客观世界本身无疑是最自然的又是最合理的一种方法。客观世界是由事物组成的，事物具有属性和行为，C++将结构扩充为既包含属性(数据成员)又包含行为(成员函数)的类，由类和对象表示一般和具体关系，由继承表示客观世界中层次、分支、等级关系。面向对象的程序设计(OOP)最根本的任务是分类、建立等级。例如模拟汽车时，汽车属性有轮子、功率、重量等，行为有制动、启动、油耗等，又可以派生出轿车、重载车等派生类，它们又有自己特殊的属性和行为，随着门类增加，反过来也可以修改汽车公有的属性和行为。

1.2.4 OOP 方法系统控制

系统控制问题是任何软件系统必须解决的一件事，也是一件难事。因为系统控制需要考虑的因素较多，首先要考虑符合用户要求，性能和习惯；其次在相应条件下争取最好的解决办法。

在面向数据流分析中，系统控制可以顺着数据流方面控制，在必要的地方，程序需要用户进行选择，输入文件名和参数，人为干预，最后运行到终点，得到结果。这种控制层次一般偏多，用户选择余地偏少，程序作成，用户想改变其中的处理流程是不允许的。

面向对象分析法中，程序根据问题需要建立类。建立在类上的功能模块是成员函数的排列组合(当然首先是有意义的)，或者说用户选择和程序员的选择余地都很大，可以从中选择一部分作为系统控制元素。仍以狗类为例，用户带来一只狗，要鉴别它属于那一种，比较过程可以从毛长、毛短开始向下鉴别。如果手头有一张狗跳跃照片，程序员可以把类中所有跳跃成员函数运行一遍，比较那种最象。之所以可以这样做，是因为对象可以定义在类的各个等级上，要实现其中某个动作，只不过运行二行程序而已，而不用考虑是怎样实现的。

面向对象的设计，其系统控制和系统结构是分离的。它把上面所属用户、结构、资源三个因素减少到两个。系统资源是任何一个软件工程都必须考虑的因素，因此软件人员主要考虑用户的需要，而不要太多顾及系统结构影响，这样的控制则更接近客观世界。

面向对象设计把系统控制看成一个类，它的调用可以遵循指针——功能模块方式进行。实际上在系统控制类中，只要有一项属性是调用功能指针就可以了，其它则考虑如何满足用户的需要：每层菜单怎么取名、如何放置、颜色如何、选择操作方法等等。从整体来看控制菜单也是一棵树(这里不是说系统控制类的继承关系是一棵树)，每个菜单名又处在这棵树的节点上。面向对象设计的系统控制类的属性大致有：节点编码，节点名称，节点颜色，节点显示 X 坐标，节点显示 Y 坐标，节点提示串，节点求助文件名，节点性质，节点代表程序指针……。控制类的操作大致有：节点显示和消失，同层节点窗口显示，节点选择移动，层次进退控制，显示提示，求助文件显示，程序调用……。

我们采用这种系统控制设计方法成功地研制了《中国地震分析预报软件系统》，它把类属性做成属性文件，使系统控制变成了一个工具——通用控制工具，它适用于以树方式组织的所有软件系统。

1.2.5 类的层次划分

层次关系是一种普遍的自然现象，软件系统一般有层次关系。

数据流模型是外部实在系统抽象，层间关系为：上层是下层的抽象，下层是上层的分解。因此它采用从上层至下层的方法建立模型，直至所有加工清楚为止。这种方法一方面要求分析人员对全系统有清楚了解，另一方面因为结束标志是加工清楚，由此形成数据流图中层次与实在世界的层次关系的不一致，只能作为一种施工图来对待。

面向对象的层次关系体现在类的继承关系上。这种关系很象一种分类学，以狗类为例，狗类中分出二个子类：长毛类和短毛类，其中短毛类再分狼狗类，牧羊犬类，猎狗类……。猎狗不仅继承了祖父类特征：四条腿，吃东西等，又继承了父亲的特征：短毛。显然这是一种从

高层向下分解的方法，高层最普遍，问题也最简单，沿着分支向下，每一层都比以前更具体，下一层包含上一层所有特征和行为。在这里类获得了等级，上一层称父类或基类，下一层称子类或派生类。无疑在狗类中增加一个类(如警犬类)，把它放在那个层次上必须认真考虑。不仅如此，狼狗不仅继承了狗的特性，而且继承了狼类特性，因此狼狗有多个父类，面向对象分析中称多重继承。

面向对象分析还允许自底向上或上下结合的方式构造模型。这两种分析法比较适用于科学研究，一开始，不需要对整个系统做到全部清楚，随着研究工作深入可以不断调整和完善，并保证原来的研究成果仍然有效。

例如，气象学家研究温度与环境关系，并不知道最后结果是什么。一开始可以建立温度类，它的属性是台站号、经度、纬度、海拔、日期、温度值。动作是增删改温度值和绘地图，绘等温线。研究中发现温度与海拔有关，就在类中增加动作——海拔修正值成员函数。进一步研究发现温度与观测点周围植被有关，这种关系也许是海拔决定植被，植被又影响温度？这种关系将由生物学家来研究，在没有研究清楚之前，可以把类作调整，设立一个祖父类，它含台站号、经度、纬度、海拔，成员函数是绘地图底图；父类是植被，等待生物学家补上；子类是温度类，属性是日期、温度值，成员函数是增删改温度值、绘等温曲线、海拔改正等。不难发现，第一阶段研究中得到的结果，在第二阶段中全部保留。假如生物学家研究结果是植被与温度无关，可以把植被类移去，程序一样运行；如果有关，把植被类放在父类位置上，子类中成员函数加上——植被改正，程序马上又可以运行。之所以把经度、纬度、海拔等抽象到基类，是因为它们是地图数据，地球上坐标，最普通，又最简单。其它如温度、气压、中立值等均与之相联系，是它们的共性。

1.2.6 系统的单根与多根

根是指一棵树的根，它是相对于枝和叶而言的。在数据流分析中，由于把整个系统分成输入、处理、输出三大块，然后再往下细分，因此可以说它是一种单根的。在面向对象的设计分析中，树体现在类及继承关系中，它取决于对数据分类，因此它应该是一种多根的系统。C++语言的某些语法，也可以把这些不同类归到一起，使系统变成一棵树，而成为单根的。到底单根的好还是多根的好，笔者认为不能一概而论。对于一个大型软件系统，参加开发的人员较多，还是多根的好，这样互相制约较少，数据归类也比较自然，对系统的开发是有利的，尤其当内存较小时，多根的更显得灵活方便(覆盖需要)。

§1.3 应用OOP方法进行系统分析和设计

下面以我们利用面向对象的程序设计方法开发研制的SSEPC软件系统为例，介绍如何应用OOP方法进行系统分析和设计。

中国地震分析预报软件系统(TheSoftwareSystemforEarthquake PredictioninChina，简称SSEPC)是将我国地震分析预报中所采用的地震数据处理方法、地震预报数学、物理模型用计算机实现的大型应用软件。该软件需要将地震分析预报的一系列计算、识别、推理、控制、判断、决策等工作用计算机实现。

该系统庞大，控制结构复杂，处理方法繁多，采用现行的结构化程序设计很难实现系统所应达到的目标。因此，采用了面向对象的程序设计方法，从而保证了系统按期完成、实用可靠，同时也有利于今后系统的扩充和维护。要实现这样一个比较大的应用软件系统，必须对该项工程进行系统分析和系统设计。在系统分析和设计时，充分利用了面向对象程序设计的思维方法，对系统的功能进行抽象分类，建立系统逻辑模型，按照逻辑模型的设计，进行类层次划分和建立相应的类库。最后，通过调用类库提供的方法(成员函数)或扩充类库，实现系统设计的各种功能。下面将概要介绍我们在进行这方面工作的一些体会。

1.3.1 软件系统分析

对 SSEPC 系统的分析主要包括系统功能分析、用户需求分析、系统数据分析和建立系统逻辑模型等工作。

1. 系统功能分析

任何一个软件系统都有特定的系统目标。为了实现这个目标，系统必须具备完成目标内容的各种功能。SSEPC 系统实现的功能，应能完成下述几个方面的任务：

①该系统首先应能够完成单学科专业的常规处理工作。即按专业进行资料的预处理、常规处理、排除资料中可能存在的干扰因素，并按不同数学处理方法进行可靠性检验，从中提取与地震可能有关的信息。

②考虑到用户对不同预报时段——长、中、短、临的要求，打破各专业学科的界限，以地震预报方法为单元，进行相应的资料分析处理，提取与地震有关的异常信息。

③为了满足用户对数据处理流程中的特殊要求，应能完成数据处理中一个或若干个处理过程，从而给用户提交相应的结果。

④对于已提取信息需要进一步加工处理，如制表、作图等工作，该系统还应提供一套工具软件，以满足用户的要求。

据上述系统要实现的目标和需完成的任务，画出系统功能图。图 1.1 是 SSEPC 系统功能图。在实际应用中，图中所列的各种方法还应继续往下划分，直至划分到一个独立的处理方法为止。下面需要进行的是用户对每一个处理方法的需求分析。

2. 用户需求分析

用户需求分析是为了了解用户对各功能模块的具体需要，如：输入输出、数据存储及不同的数据处理方法等。数据输入主要有格式化输入(直接读数据库)和非格式化输入(键盘等)。输出要求各种方法操作简单、功能灵活、界面格式统一，输出方式多样，如各种报表、图形和电子幻灯等。

3. 数据分析

在用户需求分析的基础上要对系统所要处理的数据做进一步分析，其目的是弄清各种数据的类型、数据标准和数据量，以便对数据进行正确描述。由于 OOP 的特点就是强调数据在程序设计中的重要地位，以便达到数据与操作封装的目的。因此，数据分析对下面要进行的系统逻辑模型的建立及今后的类层次划分都是非常重要的。

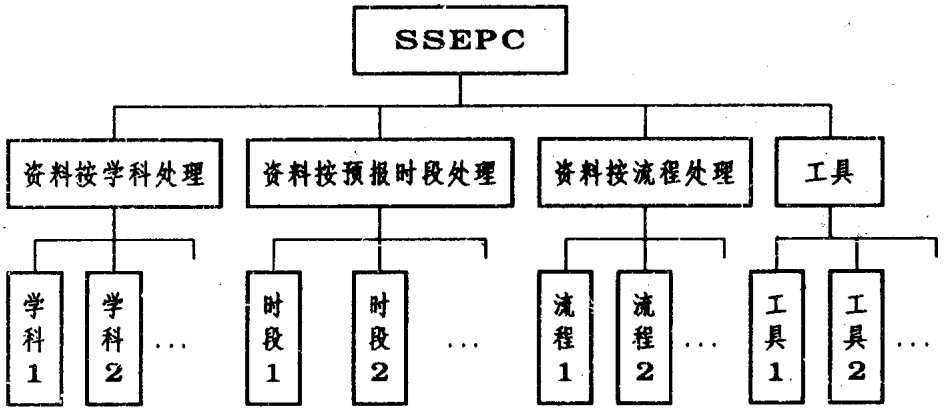

图 1.1　SSEPC 系统功能示意图

4. 建立系统逻辑模型

在系统功能分析、用户需求分析和数据分析的基础上，要建立系统的逻辑模型。建立逻辑模型的过程，在 OOP 中应分为两个阶段：

第一个阶段我们称之为现实世界事物向逻辑世界对象的转换。例如，软件系统中的一个具体处理方法，转换为对象就是一个功能模块。如图 1.2 所示。图中的现实模型(a)与逻辑模型(b)中的层次关系是一一对应的。这一阶段也就是结构化程序设计中的模块划分阶段。

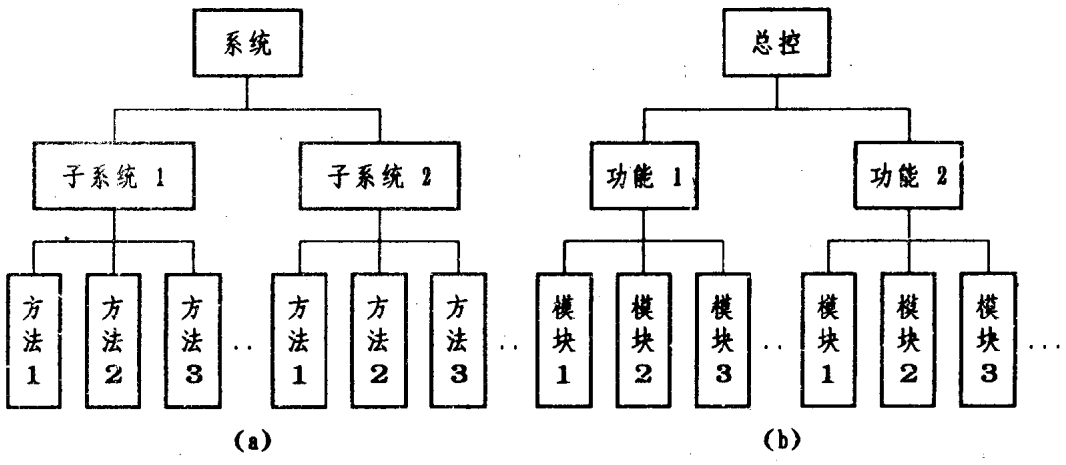

图 1.2　现实模型与逻辑模型的对应关系

第二个阶段我们称之为对象的抽象与分类阶段。它主要是摹仿图形分类学的方法实现的，也正是 OOP 思维方法的具体体现。在这一阶段中，需要将逻辑模型最底层的对象根据功能、用户需求、处理数据类型等方面的特点，做进一步分析，提取其中具有共同属性的对

象，这些属性包括功能、数据描述、操作等，并将对象分类、分层划分，直至形成一个最基本的模块结构图。图 1.3 表示了一个从用户界面角度进行对象分类、分层的过程(这里做了适当的简化)。它是按 OOP 方法给出的界面类层次结构逻辑图。在图 1.3 中，界面类是一个高层次的基类，它所描述的是表达所有界面对象共同属性的参数和对参数的操作。比如一些有关视配器的参数和操作等。界面类的派生类——文本窗口，除了具有界面类的属性外，还具有自己的属性。如一些有关窗口大小、色彩的参数及开窗、关窗的操作等。对于文本窗口的派生类，如运行窗类，除了具有界面类、文本窗口类的属性外，还有代表自己特征的属性，如运行时的提示信息等。从以上的例子可以看出，使用 OOP 方法进行系统分析，主要是进行类层次的划分，其目的是为下面进行的系统设计和实现提供依据。

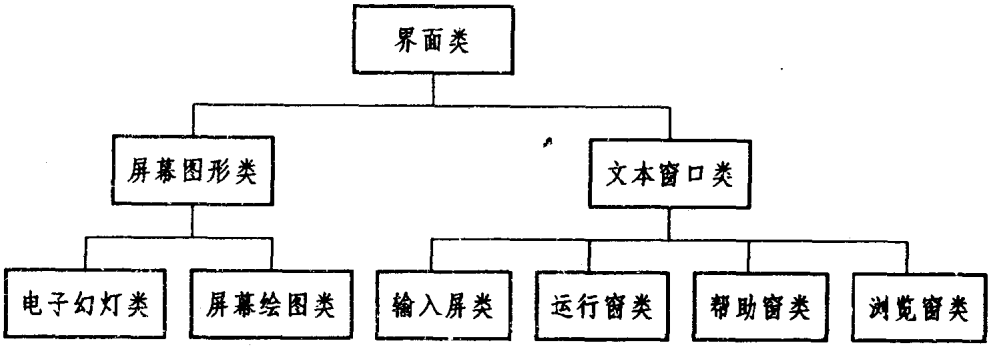

图 1.3　界面类层次结构逻辑图

1.3.2　系统设计与实现

系统设计是在系统分析的基础上进行的。系统设计的内容是根据系统提出的目标进行系统的总体设计，根据系统的逻辑模型图(图 1.2(b))进行功能模块的详细设计，同时根据类层次结构逻辑图(图 1.3)进行基本模块(通用性更强的模块)的详细设计。在系统实现的程序编码中，系统的总体功能由系统总控函数(主菜单)完成。该函数被定义在总控类里。各种功能模块(例如一个有输入输出功能的滤波方法)是通过方法控制函数完成的，它通常是调用或扩充基本模块的功能来实现。该函数也被定义在一个特定的方法类中。各层次的基本模块，是通过类的继承机理或类对象的调用方式，并根据其自身特点增加相应的数据成员和函数成员实现其相应功能。这些基本模块通常是制作成类库供自己或其他程序员使用。

第二章　C++基础与编程技术

C++是对C语言的一个扩充，它体现了面向对象程序设计语言的特征，如封装、多态性和继承等。C++与C语言的主要区别在于C++具有类及有关特性。本章将扼要介绍C++的主要特性、概念及如何在编程过程中实现的技术，以作为进一步学习C++高级程序设计技巧的基础。

§2.1　BorlandC++系统的安装

2.1.1　硬件和软件环境

BorlandC++可在IBMPC系列微机及与其兼容的微机上运行，包括XT、AT、PS/286、386、486微机等。BorlandC++2.0版要求DOS3.0以上版本和至少640KB内存，监视器为80列即可，至少应有一个硬盘启动器和一个软盘启动器。由于BorlandC++2.0版系统本身约占15兆字节，所以硬盘容量以在20兆字节以上为宜。BorlandC++3.1版要求DOS3.31或更高版本，内存至少是640KB字节加1兆字节扩展内存(因BorlandC++3.1版取消了实模式，在保护模式下运行要求至少有这么大的内存)。BorlandC++3.1版系统本身约占30多兆字节，另外要求有约5兆字节的自由空间，所以要求硬盘分区能够给一个逻辑盘分配40兆字节以上的空间，这只有DOS的3.31版或更高版本才能做到。当然硬盘容量至少应有40兆字节。另外，用于Windows环境的BorlandC++版本要求Windows3.0或更高版本，至少2兆扩展内存以及能兼容Windows的显示器。协处理器(如8087、80287、80387等)是可选的，尽管没有协处理器也可进行软件仿真运算，但有协处理器可使程序运行效率大大提高。BorlandC++2.0版和3.1版都支持鼠标器，如果你的机器配了鼠标器，则必须与以下版本之一全兼容：

MicrosoftMouse6.1版或更高版本，或任何与此兼容的Mouse版本；

LogitechMouse3.4版或更高版本；

MouseSystems'PCMouse6.22版或更高版本；

IMSIMouse6.11版或更高版本；

GenusMouse9.0版或更高版本。

鼠标器也是一个选件，它不是运行BorlandC++所必需的，但如果运行Windows环境下的BorlandC++版本，最好配上鼠标器。

2.1.2　安装过程

BorlandC++2.0版包括7片1.2兆字节的高密软盘，3.1版包括11片1.44兆字节的高

密软盘或 15 片 1.2 兆字节的高密软盘。由于有些文件是压缩存储的，所以不能用简单的 COPY 命令把文件从软盘拷贝到硬盘上，而必须运行安装程序 INSTALL.EXE，安装过程如下：

1. 把含有安装程序的软盘插入 A 驱动器，键入：INSTALL，然后按 Enter 键。
2. 显示出安装屏幕后，按 Enter 键。
3. 顺序回答所显示的各项提示，可根据你的需要，改变其中的缺省值。
4. 在安装过程完成后，应使 CONFIG.SYS 文件中含有：

FILE=20

其中数字 20 也可是大于 20 的数。还应把下行加入到 AUTOEXEC.BAT 文件中：

PATH=C:\BORLANDC\BIN

其中的 C:\BORLANDC 字样应与你在安装过程中指定的盘符和路径名一致。

§2.2 C++特征及其实现方法

2.2.1 封装(Encapsulation)

封装就是把一个数据结构和对数据进行操作的函数组合在一起，它是借助于一种新的结构和数据类型机制——类来实现的。C++可用三个关键字 class、struct、union 来定义类，把函数(称为类的成员函数)和数据(称为类的数据成员)组合在一起，成为一个新的类型标识符，可用它来说明该类的实例，即对象。例如：

```
classShow_x{                    voidinput(void){cin≫x;};
private:                        voidoutput(void){cout≪x;};
intx;                           };
public:
```

这里通过定义一个名叫 Show_x 的类，把数据 x 和两个对数据进行操作(这里是输入、输出)的函数 input()、output()封装到了一起，由于数据 x 是本类的私有成员，所以只能由本类中的两个成员函数 input()、output()对其进行访问，非本类成员的其他任何函数都不能访问它。同时，即使数据成员发生了变化，如从数组变为链表，也不影响成员函数对其进行访问。封装提供了防止程序中不相关部分访问、修改或不正确使用私有成员的有效方法。

被封装在一起的包含数据和对数据进行操作的代码的实体叫做对象(Object)。通俗地说，对象就是具体的、实际的类，而类是一组具有公共特征对象的抽象。例如我们在前面定义了一个名叫 Show_x 的类，如前所述，我们可以用这个名字来建立一个对象：

```
Show_xAn_object;
```

这里，An_object 就是类 Show_x 的一个对象，它是一个具体的 Show_x 类。

2.2.2 继承(Inheritance)

继承是由一个对象获得另一个对象特征的过程。在 C++中，继承通过在一个类中声明另外的类来实现。继承的例子在日常生活中很多，如杨树是乔木类的一部分，乔木是树木类