用C++建造专家系统

程慧霞 等 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

用C++建造专家系统

程慧霞 李龙澍 倪志伟 曹先彬 编著

电子工业出版社

内 容 简 介

本书的内容主要是论述面向对象程序设计的基本概念和基本的程序设计方法，扼要介绍面向对象程序设计语言 C++的基础，重点是如何用 C++建造专家系统，内容包括:专家系统的基本结构，面向对象的知识表示、推理模型和用神经网络的方法建造专家系统，并包括近年来研制的有关科研成果及应用 C++编制的专家系统的实例。

本书可供从事计算机技术教学、科研的专业人员参考，也可作为计算机、自动化、电子工程专业本科生、研究生的教材和参考书。

用 C++建造专家系统

程慧霞 李龙澍 倪志伟 曹先彬 编著

责任编辑：吴 源

*

电子工业出版社出版

北京市海淀区万寿路 173 信箱（100036）

电子工业出版社发行 各地新华书店经销

北京科技大学印刷厂印刷

*

开本：787×1092 毫米 1/16 印张：14.5 字数：352 千字

1996 年 6 月第一版 1996 年 6 月北京第一次印刷

印数：4000 册 定价：20.00 元

ISBN 7-5053-3290-2/TP・1233

前　言

专家系统是人工智能应用的重要领域，从 1965 年 E·A·Feigenbaum 开始研制世界上第一个专家系统 DENDRAL 以来，至今已经有近三十年的历史，它经历过自己辉煌的发展时期，研究的高潮是 1977 年知识工程概念提出之后，步入了高速发展时期，有成千上万个专家系统研制成功，它们广泛地应用于医疗诊断、气象预报、电路设计、农作物产量预报、军事系统、地质勘探、科学分析等多个领域，产生了显著的经济效益和社会效益。八十年代中期，专家系统的进一步研究遇到了困难，知识的贫乏、推理的单调、自学习功能差等问题没有取得突破性的进展，但面向多个应用领域的专家系统走上了实用的商业化阶段。在我国专家系统的应用也十分广泛，其研究和开发工作仍有广阔的前景，正如 E.A.Feigenbaum 在 1993 年 4 月访问中国时所说："你们拥有所需要的一切，有合适的工作站、软件以及智慧，希望你们很快开发出自己的、功能强大的专家系统。"

面向对象的程序设计方法是 90 年代世界计算机科学技术发展的导向技术之一，而 C++ 被认为是支持面向对象的程序设计语言，是国际上九十年代程序设计的主流语言，C++里有许多特性，如封装性、继承性和重载性等，使其适宜于建造专家系统。用它来开发专家系统无疑给专家系统带来了新鲜活力和生机。

神经网络的方法从八十年代中期开始在世界范围内形成了新的高潮，它的并行性、容错性和自适应性等在许多领域得到了应用。我们将其应用于建造专家系统亦取得了一定的经验。本书中将作者多年研究、开发的几个专家系统和开发工具软件融于其中，使得内容面向应用，突出如何建造专家系统。

全书分九章，第一章绪论、第二章面向对象程序设计概述、第三章 C++语言基础，由程慧霞执笔，第四章专家系统概述，由李龙澍执笔，第五章面向对象的知识表示的第一、二、三、四节由倪志伟执笔，第五节由李龙澍执笔，第六章面向对象的推理模型由李龙澍执笔，第七章专家系统设计由倪志伟执笔，第八章神经网络与专家系统第一节、第二节由倪志伟执笔，第三节由曹先彬执笔，第九章专家系统实例研究，第一节、第二节由李龙澍执笔，第三节由曹先彬执笔。该书是作者"七·五"和"八·五"期间从事有关专家系统和面向对象程序设计研究工作的成果及教学工作的心得体会，全书由程慧霞教授主编统稿。另外，安大贾瑞玉、张霞、余先庆、吴芝生也参加了部分工作，在此一并表示感谢。限于作者水平和能力，编著时间仓促，因此书中一定有不少错误和缺点，恳请读者提出宝贵意见，以便修订，谢谢。

程慧霞

1995 年 12 月

于安徽大学

目　录

第一章 绪 论

1.1 专家系统及其发展现状

专家系统是人工智能的重要应用领域，自 60 年代开始，经历了近三十年的发展历史，取得了很大的成功，数以万计的专家系统相继出现，被广泛地应用于国民经济的各个领域，如：医学、农业、军事、气象、地质、法律、科学技术、教育等方面。专家系统的成功应用使社会看到了人工智能研究的价值，特别是知识表示、获取的研究、不确定性推理的研究，大大丰富和扩展了 AI 的研究领域，人们开始从探索思维规律转向智能行为的研究，提出了新的课题——知识工程。

近十几年来，我国对专家系统的研究和开发应用亦取得了很大的成绩，如：医学专家系统、气象专家系统、农业专家系统、故障诊断专家系统等，实际应用均具有较高水平。

什么是专家系统呢?所谓专家系统就是一个在某一特定的领域内，运用人类专家的丰富知识进行推理求解的计算机程序系统。它是基于知识的智能系统，主要包括知识库、数据库、推理机制、解释机制、人机接口和知识获取等功能模块。一个专家系统具有以下主要特点：启发性、透明性和灵活性，即能利用专家的知识进行启发式推理，能够解释其推理过程，对用户的询问作出回答，并且能够不断地、灵活地增加新的知识。

专家系统在近三十年的发展历史中有其成功之处，但实践证明专家系统还存在着三大突出问题：

知识的脆弱性：目前专家系统的性能像一个窄的尖峰，它的适用范围非常狭窄，在这个范围内有足够的知识，可以解决问题，但超出这个范围，它的性能很快下降到零。

推理的单调性：不具备常识推理，推理方法单调、固定。

自动获取知识能力差：基本上不能在运行的过程中发现和创造知识，系统的功能实际上局限于原有的知识，存在知识获取的瓶颈问题，缺乏直觉顿悟能力。由于这些原因,在八十年代中期，专家系统的研制遇到了困难，热潮过去了。由于在专家系统中，专家的经验知识非常重要，是系统成败的关键，因而人们为了克服知识贫乏的致命弱点，根据人类认识论的规律，对常识推理和机器学习进行了重要的研究，取得了显著成绩。常识推理中有非单调推理和定性推理两种不同的研究思路，其中非单调推理否定了知识越多越好的传统认识，据此人们创立了如定理推理方面的常识知识理论。在机器学习方面进展不大，正如 Minsky 所说：“没有机器能通过学习去识别 X，除非它具有表示 Z 的 Schema，…”

人工智能的研究工作者深感常识知识的表达和获取的困难，开始对 AI 进行反思，从理论基础、技术方法上寻找新的途经。

1982 年，Rumelhart 和 Mcclelland 在美国加利福尼亚大学圣地亚哥分校建立了 PDP 研究中心，提出了 PDP 理论及其模型。PDP 模型主要是探索认知过程的微结构，在网络层次

上模拟人的认识活动。PDP 研究小组提出的七个模型：相互激活与竞争模型(IAC)、约束满足模型(CS)、模式联想机模型(PA)、反向传播模型(BP)、自联想机(AA)、竞争学习(CL)和相互激活(IA)模型，特别是 82 年 Hopfield 模型的出现，使一度处于低潮的人工神经网络的研究再度兴起新的高潮。人工神经网络模仿人类大脑的结构和功能，它是由多个非常基本的、简单的处理单元——神经元相互按某种方式联接而成的一种信息处理系统，是一个以有向图为拓扑结构的非线性动力系统。人工神经网络的主要特点是高度平行性、容错性和自适应性．近几年来使用这种新的方法和途径在模式识别、信号处理、组合优化和智能控制等领域进行了大量的研究，出现了很多成果，但没有重大的进展。人们在进一步深入的研究过程中，渐渐感到困难很多，就其原因是人工神经网有一个根本的弱点：虽然它能进行自学习，但麻省理工学院计算机科学系 Avrim Blum 和 Ronald L Rivest 却证明了："训练 3 个结点的神经网络是 NP 完全问题"。

其次，神经网络方法目前只适合较小规模的情况，当规模很大时，网络的泛化问题不易解决，安徽大学张玲教授和清华大学张钹教授在论文："神经网络中 B P 算法的分析"中深刻地分析了 BP 算法学习过程收敛速度慢，网络性能差和局部极值的出现等问题，指出 BP 算法具有不完备性。

人工智能发展的历史表明，不论是传统的符号主义还是神经网络的联结主义或近几年提出的行为主义，各学派都从不同角度在不同层次上，对人类智能的不同方面：思维、感知、行为等进行了研究，各有其局部相对真理性，国内外著名的人工智能专家较一致地认为：需要把两者结合起来，使之成为能发挥各自优点的综合集成系统。著名的中国科学院院士戴汝为提出了"人机结合的大成智慧新时代的观点，"这一时代的观点是群体专家知识的利用，开辟了集智慧之大成的道路。E.A.Feigenbaum 认为：21 世纪新一代的专家系统将是遍布全球的多种专家知识库及知识应用网络。

坚定信念，加强应用，增强学科发展的活力，发展 AI 的理论和技术，混合型专家系统就是在这样的背景倡导下，从 80 年代初期开始出现的，我们研制的小麦产量预测专家系统和农业综合应用软件包，就是一个混合型专家系统的成功实例。

1.2 C++语言的特点及其建造专家系统的适宜性

C++是 1983 年由 Bell 实验室开发成功的，至 90 年代已发展成主流的程序设计语言，被广泛地应用于系统程序设计、数据库、人工智能专家系统及计算机辅助设计等。它的成功应用主要有二方面的原因。第一个原因是，C++是 C 的一个扩展集，C++最初的版本称为："带类的 C"。这两种语言的基本语法和语义是相同的，C++模块与 C 模块基本兼容，能自由链接，所以现有的 C 库函数能直接被 C++程序所调用，特别是熟悉 C 的程序员仅需学习 C++语言的新特点就可以很快掌握 C++进行编程。另一更重要的原因是：C++支持面向对象的程序设计方法，它是一种新的软件设计的方法，更接近于人类的思维活动。它模仿对现实世界自然结构的认知过程进行系统的分析与设计，使人们的认识系统与计算系统一致，问题空间与求解空间在结构上一致，提高了软件的可重用性、可扩展性，可设计出更加结构化、可扩充、易移植和易维护的程序，成为开发大型复杂软件系统行之有效的开发

方法。所以 OOP 发展迅速，已成为 90 年代的程序设计方法，许多程序设计语言亦都加入了面向对象的编程技术，如：Borland C++，Turbo C++，Turbo Pascal V 5.5 以上版本，Microsoft C/C++V7.0，Fortran 90 等都先后引进了这种全新的程序设计思想。

C++的关键技术是类。类是用户自定义的类型，它提供了面向对象编程的主要特性：模块性、封装性、继承性、多态性和软件重用等。

在 C++中，程序表示为不同类型的对象的集合，一个类型是由一组值和作用于该组值的一组操作集合在一起构成的。类实质上定义的是一种对象类型，它描述属于该类型的所有对象的特性，由该类对象共享。例如，文本 text 可以定义为一个类。

在专家系统的开发中使用 C/C++具有很多优点。专家系统包含三个主要组成部分：知识库、推理机和用户接口。在专家系统的开发阶段可以使用 C 编制推理机，这样很易与 C 软件包中的窗口图形软件一起构成友好的、图文并茂的用户界面。在知识库的表示中，主要是专家知识的表示，常用的有规则、语义网络、框架、逻辑等。不确定知识表示包括领域知识的不确定性和领域问题求解知识的不精确性；逻辑和简单规则的表示可用 C 语言实现；对于框架和结构化的规则用 C++描述较为合适；框架知识表示是在 M.Minsky 在 1925 年发表的框架理论基础上产生的，该理论提出了以层次结构来表达知识的观点。在框架型知识表示中，知识库由一系列的框架组成，每个框架可以表示一个事物和概念。框架之间最通常的联系方法是 A kind of 的关系，共同的属性在上层框架中定义，特殊的属性在下层框架中定义，上下框架之间引入属性继承的概念，而每个框架又由若干个命名的槽构成，槽是一种数据结构，可以有值，亦可以是指向其它框架的指针，每个槽可以有若干侧面，每个侧面又可以有若干值。

因此设计框架提供分类和继承机制需要有四个基本元素：类(class)，单元(unit)，槽(slot)，和继承 (inheritance)，其中单元可以是类的单元，亦可以是实体的单元，它们的关系如图 1.1 所示：

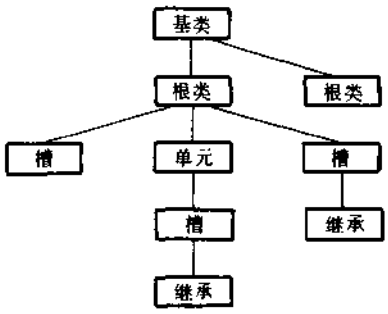


图 1.1 类、单元、槽和继承之间的关系

由上述可以看出，知识表示中的框架结构与 OOP 技术中类的层次继承等特性十分相似。我们可以用父类表示上层框架，子类表示下层框架，槽亦可以由类或另行定义其他形式来表示。类的属性可以表示槽的各个侧面及槽值，而类的每一个对象实例代表框架所示

实体类的一个特定实体。

例:

frame name : employee

slot 1 : name

slot 2 : feeling

slot 3 : age (x | 18 ≤ x ≤ 60)

所以框架可以很方便地用C++语言中的一个雇员类表示如下:

```
enum TYPE{ KIND ,  OPEN ,  COLD}
Class  employee{
        char Name ;
        TYPE Feeling;
        int Age;
        public :
        emplyee( char iname; TYPE ifeeling;int iage)
        {
          Name=iname;
          Feeling= ifeeling;
          if( 18 <=iage && iage <=60) Age=iage;
          else    cout<< " The Age is ERROR"
        }
        void get_value () {略};  // 取多属性值
    }
```

在具体实现时可按下列步骤进行: 首先建立领域知识类的层次结构框图, 然后按照每一个类定义槽结构, 对于每一个槽, 定义槽值的成员函数来管理槽值, 对父类和子类, 则用C++继承机制自然地实现。

第二章 面向对象的程序设计概述

2.1 面向对象程序设计的基本概念

面向对象程序设计被认为是90年代计算机科学技术领域的导向技术之一，已成为计算机界研究的热门课题，目前人们正从更多、更广的角度来研究面向对象的设计方法、技术和理论。从认识论的观点出发，基于对象比较符合人类的思维方式，因此它不仅限于程序设计领域，而且已经渗透到了计算机的有关分支，出现了面向对象的计算机体系结构，面向对象的数据库技术，面向对象的多媒体技术，面向对象的知识表示和方法等，许多专家学者从哲学的认识方法论的角度研究这一方法。以该思想为基础的程序设计语言也得到迅速发展并显示出强大的生命力。

2.1.1 OOP的出现背景

60年代以来，随着以科学计算为主的计算机应用的发展，出现了各种高级程序设计语言及其大型编译系统和操作系统等复杂的软件系统，软件的开销大大增加，系统可靠性和可维护性都明显的降低了。例如: IBM公司开发的OS/360系统，花费几千人年，耗资几千万美元，却拖延了若干年才交付使用。计算机界出现了软件危机并日益严重。面对这样的危机，人们试图参照土木工程、机械工程的技术来进行软件的研制，即用“工程化”的思想来解决软件研制中的困难，从而产生了“软件工程”方法学。十几年来，软件工程形成了一套科学的工程方法并开发了一套方便的工具系统，但未能从理论、方法和实践上根本解决“软件危机”。其原因是用冯·诺依曼计算机求解问题的问题空间结构与求解问题方法的方法空间结构不相一致，导致了程序设计方法和计算机体系结构方面的诸多问题。面向对象的程序设计正是在这样的背景下，根据人类认识世界的思维活动、从认识方法论的角度提出的，它改变了过去传统的以过程和操作为中心来设计系统的结构化程序设计方法，而是以“对象”或“数据结构”为中心设计软件，使得软件结构比较稳定，软件的可重用性也比较好。

面向对象的程序设计一般包括：面向对象的分析、面向对象的设计和面向对象的实现三个阶段。它能较好地反映人们求解问题的方式和方法，为克服软件危机提供了一种新的方法。估计在90年代将会在理论和方法上有更大的突破，如已经出现的基于代数的面向对象的理论模型，我国提出的类型程序设计方法学等，它们都有着重要的理论意义和应用价值。

2.1.2 基本概念

一、对象(Object)

“对象”一词的含义甚广，客观世界中的任何事物在一定的前提条件下，都可以成为认识研究的对象。因而“对象”目前还未有统一的形式定义。按照哲学的观点，对象有两

重性，即对象的静态描述和动态描述。前者表示对象的类别属性，后者则表示对象的行为特性，它们相互影响又相互依存，在面向对象的程序设计中，“对象”是系统中的基本实体，它的静态描述可表示为一个四元组:

对象::=(ID， DT， OP， FC)

其中: ID 对象的标识或对象的名字

DT 数据

OP 操作

FC 对外接口，或对象受理操作的名称集。

对象的动态描述表示为一个自动机:

对象::=(ID、 DT、 OP、 ST、 PT)

其中: ID 对象的标识或名字

OP 操作集，在动态情况下，每一个操作 OP 隐含对应一个状态转移函数 Fi

ST 状态集

PT 输入符集

如图 2.1 所示

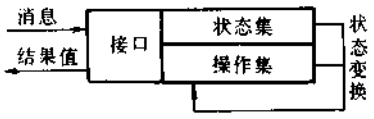


图 2.1 对象的动态自动机

对象内部的操作分为两类: 一类为修改自身属性状态的操作，另一类是产生输出结果的操作。简言之，对象是把数据和操作该数据的代码封装在一起的实体。这是说，对象有着特殊的属性和行为方式。在面向对象的程序设计中，对象是用户定义了类型的变量，对象之间只能通过函数调用相互通信，一个对象可以调用另一个对象中的公有函数。当对象的一个函数被调用时，对象执行其内部的代码来响应这个调用。

二、类(class)

类是一种对象类型，它描述属于该类型对象的共同特性。这种特性包括操作特性和存贮特性。类具有继承性(inheritance)，一个类可以是某一类的子类，从而从父类那里继承所有的特性。类中的每一个对象作为该类的一个实例，成为一个成员对象。例如: 字符 char 是一个类，它描述字符的性质，具有字符运算和存贮的特性。在程序的结构上，类形成一个具有特殊功能的模块，具有五个方面的属性:

名称: 类名

超类(super class): 类的父类名

类变量(class veriables).该类所有对象共享的变量集。

实例对象变量(Instance variables): 类的每一个对象所具有的私有变量

类属性(class attributes): 类的对象所共享的方法(methods)、外部方法(External methods)

和规则(Rules)集。我们用五元组形式化地描述类:

类(class)::=(ID， INH， DT， OI， IF)

其中 ID 是类的标识符或名字； INH 是类的继承性描述； DT 是数据结构描述； OI 是操作集名的具体实现； IF 是对外接口。

三、消息与方法

消息传递是对象间进行通讯的唯一手段，一对象可以通过传递消息与别的对象联系。消息的功能是请求对象执行某种操作，某一对象在执行相应的操作时，又可以请求其他对象完成某种操作。在系统内一般是当消息发送至一个对象时，系统根据选择器名查找方法，并先从接收消息的对象所在的类，找到父类，直到找到方法执行。对象在执行方法后不一定要返回信息，这与子程序的调用/返回完全是不相同的概念。消息中只包含发送对象的操作要求，而不包含如何完成操作的具体方法。

例如: first cos

解释: 一个消息请求名为 first 的对象计算它的 cos 值。

例如: List Addlast; newvalue

解释: 一个消息请求名为 newvalue 的对象，在对象 List 中加入新值，成为它的最后一个元素。可以看出，消息是发送给某个对象的动作标识符，说明对象要执行的操作，因此消息具有固定的模式。使用对象只需了解它的消息模式，而不必知道具体执行的细节。程序的执行就体现为对象间的消息传递。

方法则是对对象实施各种操作的描述，亦即消息的具体实现。

例如　　2　factorial → 2

在上例中，2 是一个对象，factorial 是一个消息，它请求对象执行阶乘运算，而 factorial 的内部算法就是一个方法。一般情况下，一个消息由三部分组成: 接受消息的对象，消息选择符，一个或多个参数。上例中 2 是接受者对象，factorial 是消息选择符，这里表达式中不包含参数。消息选择符相当于函数名；接受者对象相当于函数参数；函数的定义就是方法。一个消息常常返回一个单独对象作为结果。

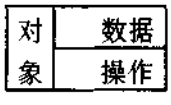
2.2　面向对象程序设计的基本特征

一、模块性(module)

一个对象是系统中基本的运行实体，其内部状态不受或很少受外界的影响，它具有模块化最重要的特性: 抽象和信息隐蔽。模块反映了数据和对象的抽象，是设计良好软件系统的基本属性，每一个模块是程序可单独编址的元素。

二、封装性(Encapsulation)

封装是一种信息隐蔽技术。就是把数据和加工数据的操作封装在一起，构成一个具有类类型的实例即对象。对象是封装的数据和操作。如图示:



对象将自己的功能封装起来，以便将对象的使用者和设计者分开，从而加快软件研制的速度。

三、继承性(Inheritance)

继承是类的特性。即子类可以继承父类的特性，系统的处理能力可以通过对象的继承性实现共享。继承可以用一个偏序关系来定义:

inh::=(C，>=);

其中 C 是具有继承关系的所有类，>=表示继承关系。显然继承关系具有传递性: 如果(C_2>=C_1)，(C_3>=C_2)则 C_3>=C_1，继承关系常称: "(is a)" 关系。因为若 A 类继承 B 类，B 类是父类，A 类是子类，A 类继承 B 类的全部性质，所以 A 即是 B "(A is a B)"，一般情况下 A 也可以具有 B 所没有的特性和内容，即增加的部分。见图 2.2

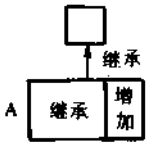


图 2.2 继承性

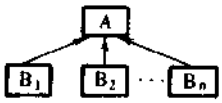


图 2.3 多重继承

一个类可以有多个子类，也可以有多个父类，所以一个类可以继承多个类，这种继承方式就称为多重继承，如图 2.3 。

四、动态连接(Dynamic binding)

面向对象的程序设计中，对象功能的执行是在运行时、消息传送时确定的，因此可以实现对象间的动态连接，这样比较灵活，有利于建立类库，便于重用和扩充。

五、多态性(Polymorphism)

面向对象的程序设计语言支持多态性。从本质上看，多态就是一个名字可以有多种语义。允许把同一消息送到一个父类的对象和它的子类的对象上，多态性主要强调在一个类等级中，可以使用相同函数的多个版本，取决于运行时人口参数的类型、存取方式和返回值，可在不同的运行时间执行。

六、易维护性和开放型设计

由于对象实现了抽象和封装，使可能的错误局限于自身，不易传播，易于查错和修改，同时利用对象的继承性，可使系统的功能不断地根据需要进行扩充，而很小影响到其它软件的运行。

2.3 面向对象的程序设计方法

早期的程序设计是艺术和技巧的活动。从 60 年代开始随着多种大型的、复杂的操作系统、编译系统等软件系统的陆续出现，软件的开销大大增加，但可靠性却难以保证，出现了"软件危机"。"软件危机"的出现促进了程序设计方法学的研究，最为代表性的研究工作是: O.J.Dahl，E.W.Dijkstra 及 C.A.Hoare 合著的《结构程序设计》一书，在该书中提

出了自顶向下、逐步求精、模块化和层次化的结构化程序设计方法，以后人们开始仿照建筑、机械等工程为程序设计制定规范及其标准，试图用“工程化”的思想去解决软件研制中的困难，解决“软件危机”问题。计算机科学家在 70 年代末及 80 年代初，掀起了一股程序设计是一门科学的浪潮。 E.W.Dijkstra 于 1976 年出版了专著《程序设计规范》(A Discipline of programming)提出了将程序设计与证明技术结合在一起，认为程序设计的过程就是证明的过程，以后就出现了软件自动生成的概念，虽然 20 多年来，在实践中解决了一些问题，但没有根本性的突破。而在实际上大量研制软件过程中，20 多年来程序员的编制程序的能力却大大提高了，从过去的 2000 行/人年，提高到现在的 20000 行/人年，甚至更多。总结其关键的技术是：软件重用技术。提倡软件的重用，增强了程序的规范化和标准化。

传统的结构化程序设计是自顶向下，按功能划分模块，按照面向任务的观点，逐步求精将给定问题域中的具体任务细化为若干个子任务，构成基本的“模块”，其大致流程是：