希望
Beijing Hope Electronic Press
www.bhp.com.cn

21世纪高职高专信息技术教材

C++程序设计教程

北京希望电子出版社　总策划
刘振安　编　著

科学出版社
www.sciencep.com

Beijing Hope Electronic Press
希望
www.bhp.com.cn

21世纪高职高专信息技术教材

C++程序设计教程

北京希望电子出版社 总策划
刘振安 编 著

科学出版社
www.sciencep.com

内 容 简 介

初学C++的人会感觉有些难度，比如，没有使用类的概念，也很难接受新的思维方法。所以本书将必要的基础知识通过使用类来讲解，在学生对类的性质有了感性认识之后，再深入讨论，这样比较接近人的思维规律。

本书根据高职院校的特点，进行合理取舍，展现它们的最新特征。全书把重点放在程序设计方法上，将内容划分为两大部分：面向过程和面向对象。在讲授面向过程时，直接引入使用对象的概念，通过使用对象设计面向过程的程序，熟悉使用对象的方法，通过使用C++提供的类，建立对象行为及实例的概念，为面向对象程序设计打下基础。

本书不要求读者学过C语言，面向过程设计部分的思想也适合C语言，只是实现有些差异而已，所以也可以用来学习C语言编程。这部分还介绍了面向对象和面向过程所共有的许多设计方法，所以对于已经学过C语言的读者，还必须重新学习这部分的内容以建立面向对象的概念。

本书取材新颖、结构合理、概念清楚、语言简洁、通俗易懂、实用性强、易于教学。本书特别适合作为高职高专的教材，也可以作为培训班教材，自学教材及工程技术人员的参考书。

需要本书或技术支持的读者，请与北京中关村083信箱（邮编100080）发行部联系，电话：010-82702660，010-82702658，010-62978181转103或者238，传真：010-82702698，E-mail：yanmc@bhp.com.cn。

图书在版编目（CIP）数据

C++程序设计教程/刘振安编著. —北京：科学出版社，2005.2

21世纪高职高专信息技术教材

ISBN 7-03-013460-5

Ⅰ. C… Ⅱ. 刘… Ⅲ. C语言—程序设计—高等学校：技术学校—教材 Ⅳ. TP312

中国版本图书馆CIP数据核字（2004）第044678号

责任编辑：肖寒 /校 对：计 芳

责任印刷：媛明 /封面设计：梁运丽

科学出版社 出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

北京市媛明印刷厂印刷

科学出版社发行 各地新华书店经销

*

2005年2月第 一 版 开本：787×1092 1/16

2005年2月第一次印刷 印张：18

印数：1—5 000 字数：413478

定价：25.00元

21世纪高职高专信息技术教材编委会名单

（排名不分先后）

主　任　高　林

副主任　谢玉声　　袁启昌

胡伏湘　　陆卫民

委　员

阮东波	侯晓华	曹冬梅
周文革	蒋建强	杨章静
程　刚	王趾成	尹　静
连晋平	龙　超	田　更
唐伟奇	罗映峰	吴　军
慕东周	冯矢勇	杨金龙
朱作付	徐　萍	崔俊杰
杨旭东	李　淼	陈翠娥
米　昶	李超燕	陈　春
孙　杰	景鹏森	徐建华
唐燕青	陈孟建	郑明红
刘　毅	宗小驷	韩素华
邱建国	韦　伟	

序

高等职业教育目前已成为我国高等教育的重要组成部分，对于推动我国社会主义现代化建设起着不可忽视的作用。计算机教育在整个高职教育中有着举足轻重的地位，因为计算机的普及已经涉及到各个行业。对于传统的学习计算机知识的方法即理论为主、应用为辅的教学模式，相对高职教育来说有些不太适合，针对这种情况，就需要一些符合高职教育特点的教材来满足这种需求。

为解决教材供需不平衡的矛盾，北京希望电子出版社与全国高等学校计算机基础教育研究会高职高专专业委员会联合组织国内十几所高职院校，聘请“双师”型教师共同编写针对高职特点的教材30多种，以及实训类教材10多种，并请专家论证了本套教材的体系、风格、结构、内容等方面的可行性与可操作性。该系列教材体现“重在能力素质培养”的目标，结合教育部的教学大纲要求，在实用性、新颖性、可读性几个方面都有所突破。

高职教材建设是教学改革重要的环节，高等职业技术教育专业设置要与劳动力市场需求相结合，教学内容与国家职业标准相衔接。采取“订单教学”的校企合作培养模式，实行学业文凭和职业资格两种证书制度，使一线技术人才培养实现教学与市场“零距离”、毕业生上岗“零适应期”。这种以市场为导向实行的订单教学，能够直接为用人单位培养实用型人才，是一条富有特色的职教之路，可以保证同学们将来在就业和升学两条渠道上有最大的发展空间。所以，高校就要突出应用技能培养的办学特色，按照人才市场供求信号进行学科、专业和教学内容的调整，以适应社会需要。在培养学生的知识、能力、技能方面都要与其他综合性本科院校有所区别。

本系列教材就是遵循这种订单式教学的需要，一方面是设定系统理论知识的教材，这种教材的内容按照“必需、够用”的原则，构筑坚实的具有高职特色的理论体系基础；另一方面是训练职业动手能力的实训教材，按照“切实、实用”的原则，培养动手能力强的人才。以上两种教材相互配合，既可以单独使用，也可以配套使用。

高职教材建设还在探索中，如何能满足企业对人才的需求，跟上时代发展的步伐，这些都是亟需解决的问题。本丛书旨在抛砖引玉，希望更多的优秀教师参与到教材建设中来，真诚希望广大教师、学生与读者朋友在使用本丛书过程中提出宝贵意见和建议，为下一次的修订与改版做准备，使本丛书日臻完美。

若有投稿或建议，请发至本丛书出版者电子邮件：textbook@bhp.com.cn

21世纪高职高专信息技术教材编委会

前　言

计算机科学发展的每一步几乎都在软件设计和程序设计语言中得到充分体现。程序设计方法和技术在各个时期的发展不仅直接导致了一大批风格各异的程序设计语言的诞生，而且对计算机理论、硬件、软件以及计算机应用技术等多方面都产生了深远的影响。

本书把重点放在程序设计方法上，将内容划分为两部分：面向过程和面向对象。在讲授面向过程时，直接引入使用对象的概念，通过使用对象设计面向过程的程序，熟悉使用对象的方法；通过使用 C++提供的类，建立对象行为及实例的概念，为面向对象程序设计打下基础。

本书将对象贯穿于每一章，强化对象的概念，以利于概念的建立和学习。全书共分 12 章。第 1 章和第 2 章是程序设计基础，分别介绍基于过程和面向对象程序设计的基本概念，引入面向对象的概念并介绍 C++程序和面向对象编程的基础知识。第 3 章至第 5 章侧重于过程编程和对象的概念。第 3 章是结构化编程基础，结合实例，简要介绍 C++语言的对象在基于过程设计中的使用方法，以及结构化程序的基本设计原理，既为基于过程的编程打下基础，也加深使用对象的概念。第 4 章是构造类型初探，介绍以基本类型为基础构造出来的几个典型的构造类型，并简要说明它们的使用方法，从而为程序设计提供新的舞台。第 5 章是函数，将涉及多文件编程的知识，多文件编程更是面向对象编程必须遵循的原则。第 6 章至第 12 章是 C++面向对象编程。其中，第 6 章和第 7 章是对象和类，这两章首先抽象对象的概念，以便更好地描述对象，然后讨论对象的 3 大要素并介绍面向对象语言的 4 大特征，重点介绍在 C++中定义类、建立和使用对象的方法。第 8 章是继承和派生类，将讨论 C++语言继承方面的语法特征和一般的使用方法。第 9 章是多态性和虚函数，由于多态性是一个与实现有关的概念，因而难于理解和掌握，介绍了运行时的多态性，并通过图解和大量程序实例帮助读者更好地理解多态性。第 10 章是类的成员和对象，讲述类的基本结构和一些特殊的成员（数据成员及成员函数）。第 11 章是运算符重载及流类库，介绍运算符重载的基础知识、流类库的概念及使用流类库进行文件存取的基本方法。第 12 章是面向对象课程设计，将重点讨论一些深入课题及设计实例。另外，每章至少给出一个实验题，实验题的难易程度也不相同。习题中的某些题目也可以作为实验题，学生可以选做一些实验，同时也可避免每届学生雷同。

教育部计算机课程指导委员会副主任委员，中国科学技术大学计算机系高性能计算中心主任陈国良教授、原安徽大学副校长，计算机系程慧霞教授及南京大学计算机系陈本林教授在百忙之中审阅了书稿并提出许多宝贵意见，特此表示感谢。写作中还参考了大量资料，有的收入参考文献之中，还有些没有收入其中。特此对这些作者表示感谢。

孙忱、刘燕君、王晋军、孙捷、周航、葛愿、章守信、潘剑锋、蒋琳等参加了本书的编写工作。尽管我们已经竭尽全力，但因才疏学浅，遗漏之处在所难免，敬请广大同行和读者批评指正。联系地址：zaliu@ustc.edu.cn

编　者

目　录

第 1 章　面向对象程序设计基础知识 1
1.1　面向过程的程序设计方法 1
1.1.1　自然语言与计算机语言之间的鸿沟 1
1.1.2　面向过程与结构化程序设计 2
1.2　面向对象的程序设计方法 5
1.3　面向对象语言的发展 7
1.4　C++的面向过程和面向对象程序设计 8
1.5　C++面向对象程序设计特点 9
1.5.1　对象 9
1.5.2　抽象和类 10
1.5.3　封装 11
1.5.4　继承 12
1.5.5　多态性 12
1.6　数据对象和数据类型 12
1.6.1　数据对象、变量和常量 12
1.6.2　数据类型 14
1.6.3　基本数据类型的实现 14
1.7　本书的结构 15
1.8　习题 16
第 2 章　C++程序设计基础 17
2.1　C++的基本数据类型和表达式 17
2.1.1　初识 C++ 的函数和对象 17
2.1.2　标识符 20
2.1.3　变量对象 21
2.1.4　基本数据类型 21
2.1.5　变量对象的存储类型 22
2.1.6　常量对象 26
2.1.7　匈牙利命名法 28
2.1.8　算术运算符和运算表达式 29
2.1.9　赋值运算符与赋值表达式 30
2.1.10　逗号运算符与逗号表达式 32
2.2　面向对象的标记图 32
2.2.1　类和对象的 UML 标记图 32
2.2.2　表示对象的结构与连接 33
2.2.3　使用实例 35
2.2.4　对象、类和消息 36
2.3　使用类和对象实例 37
2.3.1　使用 C++的 string 对象 37
2.3.2　使用 string 类的典型成员函数实例 38
2.3.3　使用对象小结 40
2.4　典型例题及错误分析 40
2.4.1　典型例题 40
2.4.2　初学者最容易出现的语法错误 41
2.4.3　容易出现的其他错误 42
2.5　程序的编辑、编译和运行的基本概念 44
2.6　实验　如何编辑、编译、调试和运行一个实际程序 45
2.7　习题 45
第 3 章　结构化编程基础 47
3.1　典型 C++程序结构 47
3.1.1　函数和函数原型 48
3.1.2　const 修饰符和预处理程序 49
3.1.3　程序注释 50
3.1.4　程序语句 51
3.1.5　大小写字母的使用 53
3.1.6　程序的书写格式 53
3.1.7　数据的简单输入输出格式 53
3.2　关系运算与逻辑运算 57
3.3　结构化程序设计概述 59
3.4　控制选择结构 59
3.4.1　用 if 语句实现选择结构设计 59
3.4.2　用 switch 语句实现选择结构设计 63
3.5　循环控制结构设计 64
3.5.1　while 语句 64
3.5.2　do while 语句 66
3.5.3　for 语句 67
3.5.4　break 语句、countinue 语句及 goto 语句 68

3.5.5 控制语句的嵌套 70
3.6 典型例题及错误分析 71
3.6.1 典型例题 71
3.6.2 错误分析 74
3.7 实验 编程调试实验 77
3.8 习题 77
第 4 章 构造类型初探 79
4.1 指针 79
4.1.1 构造指针类型 79
4.1.2 指针类型及指针运算 81
4.1.3 对指针使用 const 限定符 83
4.1.4 进一步讨论指针 85
4.2 引用 87
4.3 数组 89
4.3.1 一维数组 89
4.3.2 数组与指针的关系 92
4.3.3 多维数组 93
4.3.4 字符串数组和 string 对象 95
4.3.5 指针数组 96
4.3.6 命令行参数 97
4.4 类型定义关键字 typedef 97
4.5 枚举 98
4.6 结构 99
4.6.1 结构定义及其对象的初始化 99
4.6.2 结构数组 100
4.6.3 结构指针 101
4.6.4 动态分配内存 103
4.7 联合 103
4.8 使用数组与指针易犯的错误 104
4.8.1 数组使用错误 104
4.8.2 指针使用不当 104
4.9 实验 综合实验 107
4.10 习题 108
第 5 章 函数和函数模板 110
5.1 函数基础知识 110
5.1.1 函数基本要素 110
5.1.2 函数调用形式 112
5.1.3 递归调用 114
5.2 函数参数的传递方式 114
5.2.1 传值和传地址 115
5.2.2 传引用方式 116
5.2.3 默认参数 116
5.2.4 正确选择函数原型及传递参数 117
5.3 深入讨论函数返回值 121
5.3.1 返回引用的函数 121
5.3.2 返回指针的函数 122
5.3.3 返回对象的函数 123
5.3.4 函数返回值作为参数 124
5.4 内联函数 124
5.5 函数重载 125
5.6 函数模板 126
5.7 解题和算法描述 128
5.7.1 计算机解题 128
5.7.2 常用过程设计的算法描述方法 131
5.8 综合实例 132
5.9 错误分析 135
5.10 文件中的函数调用 136
5.10.1 使用多个文件进行模块化设计 136
5.10.2 头文件和函数原型的作用 138
5.10.3 组合为一个工程项目 138
5.10.4 使用文件包含的方法 140
5.10.5 #define 和 const 的异同 140
5.11 实验 编辑多文件程序实验 141
5.12 习题 142
第 6 章 对象和类的基础知识 144
6.1 类及其实例化 144
6.1.1 定义类 144
6.1.2 使用类的对象 147
6.1.3 数据封装 150
6.1.4 成员函数重载及默认参数 151
6.1.5 this 指针 152
6.1.6 一个类的对象作为另一个类的成员 152
6.2 类和对象的性质 154
6.2.1 对象的性质 154
6.2.2 类的性质 155
6.3 结构和联合 157
6.4 面向对象编程的文件规范 158

6.4.1 编译指令 158
6.4.2 编写类的头文件 160
6.5 实验 使用类和对象的实验 160
6.6 习题 161
第 7 章 类和类模板 163
7.1 构造函数 163
7.1.1 定义构造函数 163
7.1.2 构造函数和运算符 new 165
7.1.3 默认构造函数和默认参数 165
7.1.4 复制构造函数 167
7.2 析构函数 167
7.2.1 定义析构函数 168
7.2.2 析构函数和运算符 delete 168
7.2.3 默认析构函数 169
7.3 调用复制构造函数 169
7.4 重载对象的赋值运算符 172
7.5 对象成员的初始化 175
7.6 类模板 177
7.7 容易混淆的问题 180
7.8 实验 使用包含构造新类 180
7.9 习题 181
第 8 章 继承和派生 182
8.1 继承和派生的基本概念 182
8.2 单一继承 184
8.2.1 单一继承的一般形式 184
8.2.2 派生类的构造函数和析构函数 184
8.2.3 类的保护成员 186
8.2.4 访问权限和赋值兼容规则 187
8.3 继承类模板 192
8.4 多重继承 196
8.5 二义性及其支配规则 197
8.5.1 二义性和作用域分辨符 197
8.5.2 派生类支配基类的同名函数 199
8.6 典型问题分析 200
8.7 实验 公有派生的赋值兼容性规则 201
8.8 习题 201
第 9 章 多态性和虚函数 202
9.1 多态性 202
9.1.1 静态联编中的赋值兼容性及名字支配规律 202
9.1.2 动态联编的多态性 204
9.2 虚函数 206
9.2.1 虚函数的定义 206
9.2.2 虚函数实现多态性的条件 206
9.2.3 进一步探讨虚函数与实函数的区别 208
9.2.4 构造函数和析构函数调用虚函数 211
9.2.5 纯虚函数与抽象类 213
9.3 多重继承与虚函数 215
9.4 容易混淆的问题 216
9.5 实验 虚函数与多态性 216
9.6 习题 217
第 10 章 类的成员和对象 219
10.1 静态成员 219
10.2 友元函数 222
10.3 const 对象和 volatile 对象 226
10.4 数组和类 230
10.5 实验 友元函数和常对象性质实验 232
10.6 习题 233
第 11 章 运算符重载及流类库 234
11.1 运算符重载 234
11.1.1 运算符重载的实质 234
11.1.2 类运算符和友元运算符的异同 234
11.1.3 ++和--运算符的重载 237
11.2 流类库 239
11.2.1 流类库的基本类等级 239
11.2.2 运算符“<<”和“>>”的重载 240
11.2.3 格式控制 242
11.3 文件操作 244
11.3.1 文件操作方式 244
11.3.2 常用输出文件流成员函数 245
11.3.3 二进制输出文件 248
11.3.4 常用输入流及其成员函数 248
11.3.5 文件读写综合实例 252
11.4 实验 文件综合实验 255
11.5 习题 256
第 12 章 面向对象课程设计 257

12.1 过程抽象和数据抽象 257
12.2 发现对象并建立对象层 258
12.3 定义数据成员和成员函数 260
12.4 如何发现基类和派生类结构 262
12.5 接口继承与实现继承 263
12.6 链表 266
12.6.1 简单的链表实例 266
12.6.2 改进封装性的实例 268
12.7 实验 改进链表实验 271
12.8 习题 271
附录 A 按字母表顺序排序的 C 和 C++保留字 273
附录 B C 语言关键字 274
附录 C C 语言的 printf 格式输出函数 275
附录 D C 语言的 scanf 格式输入函数 277
附录 E 主要参考文献 279

本章主要内容：

- 面向过程的程序设计方法
- 面向对象的程序设计方法
- 面向对象语言的发展
- C++的面向过程和面向对象程序设计
- C++面向对象程序设计特点
- 数据对象和数据类型
- 本书的结构

第1章 面向对象程序设计基础知识

计算机科学发展的每一步几乎都在软件设计和程序设计语言中得到充分体现。软件是一个发展的概念，随着软件开发规模的扩大和开发方式的变化，人们开始将程序设计语言作为一门科学来对待。程序设计方法和技术在各个时期的发展不仅直接导致了各种程序设计语言的产生，还对计算机理论、硬件、软件以及计算机应用技术等多方面都产生了深远的影响。

本章将使用伪码，以设计一个输入三角形的3个顶点坐标、计算3条边的长度的算法为例，分别介绍基于过程和基于对象程序设计的基本概念，引入面向对象的基础知识。

1.1 面向过程的程序设计方法

本节将简要说明自然语言与程序设计语言之间的关系，程序设计语言发展过程和面向过程的编程思想及其不足之处。

1.1.1 自然语言与计算机语言之间的鸿沟

软件开发是一个对给定问题求解的过程。从认识论的角度看，可以归为两项主要活动：认识与描述。软件开发者将被开发的整个业务范围称作“问题域”，“认识”就是在所要处理的问题域范围内，通过人的思维，对该问题域客观存在的事物以及对所要解决的问题产生正确的认识和理解，包括弄清事物的属性、行为及彼此之间的关系并找出解决问题的方法。因为人类的任何思维活动都是借助于他们所熟悉的某种自然语言进行的，所以开发人员对问题域的认识是人类的一种思维活动。例如，有些人讲话很流利，但考虑问题时却很糊涂，这说明人们还需要具有正确的思维方法。尤其是在软件开发过程中，要求人们对问题域的理解，要比日常生活中对它的理解更深刻、更准确。这需要许多以软件专业知识为背景的思维方法。

“描述”是指用一种语言把人们对问题域中事物的认识、对问题及其解决方法的认识描述出来。最终的描述必须使用一种能够被机器读得懂的语言，即编程语言。

直接使用机器语言来编写程序是一种相当复杂的手工劳动，它要求使用者熟悉计算机的有关细节，一般的工程技术人员难以掌握。

汇编语言出现于20世纪50年代初期，其主要特征是可以用助记符来表示每一条机器指令。由于汇编语言比机器语言容易记忆，编程效率就比机器语言前进了一大步。但汇编

语言程序的大部分语句还是和机器指令一一对应的，与机器的相关性仍然很强。

用汇编语言编好的程序需要由相应的翻译程序翻译成机器语言程序后方可执行。用程序设计语言写成的程序称为源程序，可以在具有该种语言编译系统的不同计算机上使用。源程序必须翻译成机器语言才能执行。逐条翻译并执行的翻译程序称为解释程序，例如，BASIC 语言解释程序。而将源程序一次翻译成目标程序然后再执行的翻译程序称为编译程序，例如，FORTRAN、C 和 C++编译程序。

高级语言起始于 20 世纪 50 年代末期，因为它们更接近自然语言和数学语言，所以用它们编写的程序可读性强，交流较方便。60 年代中期，FORTRAN、COBOL、LISP 和 ALGOL 语言已相继出现，70 年代以来，随着结构化程序设计思想的日益深入，使得这段时期问世的几种程序设计语言的控制结构大为简化，比较有代表性的有 PASCAL 和 C 语言等，它们均属于面向过程的程序设计语言。从 60 年代起，ALGOL，PL/1 与 COBOL 都逐渐消失。尽管 PASCAL 语言的许多结构用在 Ada 语言中，但它在 70 年代早期已度过了它的鼎盛时期，逐渐成为教学用语言。目前仍在使用的、较老的语言都经历了阶段性的修改，从而反映来自其他计算机领域的明显影响。例如，FORTRAN 90 和 Ada 95，LISP 更新为 Scherne LISP，以后又改为 Common LISP。这些语言又称为面向过程的语言。较新的语言如 C++，则是从 C 语言演化而来的混合型面向对象语言。

由此可见，人们借助自然语言所产生的对问题域的认识远远不能被机器理解和执行，而机器能够理解的编程语言又很不符合人的思维习惯。人们习惯使用的语言和计算机能够理解并执行的编程语言之间存在着很大的差距，这种差距称为“语言的鸿沟”。程序设计语言发展的趋势就是为了使这种鸿沟变窄。图 1-1 给出随着语言发展鸿沟变窄的示意图。

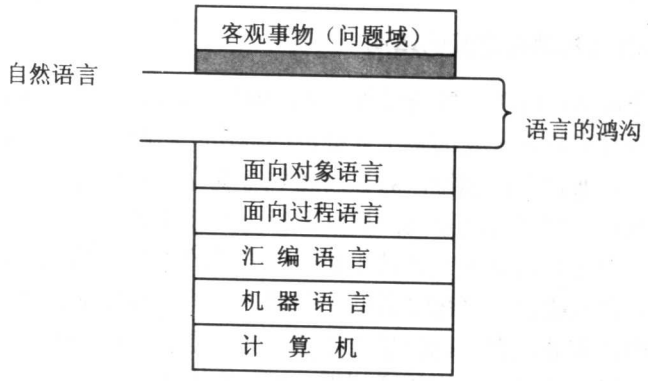

图 1-1 语言的发展使鸿沟变窄

由于人的认识差距，所以问题域与自然语言之间也有缝隙。机器语言与自然语言的鸿沟最宽，随着编程语言由低级向高级的发展，它们与自然语言之间的鸿沟在逐渐变窄。

1.1.2 面向过程与结构化程序设计

C 语言是美国 Bell 实验室开发成功的。当时的高级语言基本上都不适合开发系统软件，而 C 语言却成功地开发了 UNIX 操作系统。它的表达式简洁，具有丰富的运算符和良好的控制结构与数据结构。目前，其应用领域已不限于系统软件的开发，而成为最流行的程序

设计语言之一。C 语言是典型的面向过程的语言。所谓“面向过程”，就是不必了解计算机的内部逻辑，而把精力主要集中在对如何求解问题的算法逻辑和过程的描述上，通过编写程序把解决问题的步骤告诉计算机。

【例 1.1】给出输入三角形的 3 个顶点坐标，计算三条边的长度的过程算法描述。

从面向过程的角度看，问题的实质是取得 3 个顶点的坐标，然后计算每两点之间的距离。可以将求解过程简单地描述如下：

输入：3 个坐标，即 6 个数据。

输出：三条边的长度。

算法设计

接受 3 组数据，每组 2 个数据。

（x1,y1）　← 第 1 组数据

（x2,y2）　← 第 2 组数据

（x3,y3）　← 第 3 组数据

计算每两点之间的距离

AB ← func（x1,y1,x2,y2）

AC ← func（x1,y1,x3,y3）

BC ← func（x2,y2,x3,y3）

输出（AB，AC，BC）

// 函数 func 的说明

func（a,b,c,d）

参数：a,b,c,d

功能：求（a-c）^{2}+（b-d）^{2}的平方根

算法结束

函数 func 相当于 C 函数库中的 sqrt 函数，可以直接使用 sqrt 函数求平方根。例如，：

AB ← sqrt（（x1-x2）*（x1-x2）+ （y1-y2）*（y1-y2））

这种算法的特点是按部就班地求解，而且条理清晰易懂。

由此可以看出，面向过程的程序设计的关键是考虑使用结构化设计方法，使程序模块化。因为它是以函数过程和数据结构为中心，所以不能直接反映出人类认识问题的过程。

随着应用需求的扩大和变化，软件生产的方法和效率仍然远远跟不上社会发展的需要。软件工作者开始考虑如何提高软件质量和生产效率，即使用系统方法代替个人经验、智慧、技巧等，使建立软件系统的过程遵从一系列规范化阶段，包括需求分析、概要设计、详细设计、实现、组装测试、运行和维护等。这就将软件设计工作推进到软件工程时代。

结构化程序设计被称为软件发展中的第 3 个里程碑，其影响将比前两个里程碑（子程序、高级语言）更为深远。结构化程序设计的概念，最早是由 Dijkstra 提出的。他在 1965 年召开的 IEIP 会议上提出“GOTO 语句可以从高级语言中取消”，“一个程序的质量与程序中所含的 GOTO 语句的数量成反比”。

Tom.De Marcod 在《结构化分析与系统规格说明》教材中提出基于模型的软件工程概念，他认为对于复杂软件系统的创建，必须首先为它们建立系统的书面模型。另一个有影响的软件理论是 Niklans Wirth 提出的“算法+数据结构=程序”。将软件划分成若干个可以

单独命名并分别编写的部分，称其为模块。模块化使软件能够有效地管理、维护、分析和处理复杂问题。在80年代，软件工作者普遍接受了模块化程序设计方法，接下来就是争论如何建立模块。有人认为最佳途径是使用函数，有人认为每个模块只应容纳一个数据结构，有人认为每个模块只应做一件事，还有人提出了事件驱动概念。这些代表性解决方案也促进了程序设计语言的发展。

C 语言是结构化程序设计语言，它的程序设计特点就是函数设计。所谓函数，就是模块的基本单位，是对处理问题的一种抽象。例如，将求绝对值的功能抽象为 abs（参数），就有 abs（68）=68 和 abs（-68）=68。称 abs 为求一个数的绝对值函数，而称 68 和-68 为函数 abs 的参数。把一切逻辑功能完全独立的或相对独立的程序部分都设计成函数，并让每一个函数只完成一个功能。这样，一个函数就是一个程序模块，程序的各个部分除了必要的信息交流之外，互不影响。相互隔离的程序设计方法就是模块化程序设计方法。C 语言的这种程序结构化和模块化设计方法，特别适合于大型程序的开发。它解决了过去组成大系统时所产生的多文件的组织与管理问题。

在程序规模比较大时，一般是根据结构化程序设计方法将程序划分成多个源文件。在编译该程序时，可以按一个个源文件为单位，分别进行编译并产生与之对应的目标文件，然后再用连接程序把所生成的多个目标文件连接成一个可执行文件。称 C 语言的这种编译过程为分块编译。

C 语言的这种分块编译处理方式可以使一个程序同时由多个人进行开发，为大型软件的集体开发提供了有力的支持。分块编译的优点还在于修改一个源文件中的程序后，并不需要把整个程序的所有文件重新编译，这就大大节省了时间。

下面是用 C++语言编写一个求三角形两点之间距离、面向过程的算法思想。

输入：3 个坐标，即 6 个数据。

输出：3 条边的长度。

算法设计

```
接受 3 组数据，每组两个数据。
    （x1,y1） ← 第 1 组数据
    （x2,y2） ← 第 2 组数据
    （x3,y3） ← 第 3 组数据
计算每两点之间的距离
    AB ← func（x1,y1,x2,y2）
    AC ← func（x1,y1,x3,y3）
    BC ← func（x2,y2,x3,y3）
输出（AB，AC，BC）
// 函数 func 的说明
 func（a,b,c,d）
 参数：a,b,c,d
 功能: 求（a-c）2+（b-d）2 的平方根
```

算法结束

函数 func 相当于函数库中求平方根的 sqrt 函数。

由此可见，算法是通过程序告诉计算机如何执行的，构成程序的符号系统是语言，语言是描述算法的工具。计算机语言（程序设计语言）的发展过程就是其功能不断完善、描述问题的方法逐步接近人类思维方式的过程。在计算机语言的发展过程中，新技术和新思想也不断出现。

因为 C++是混合型语言，所以可以使用 C++编译器所提供的对象，设计出更好的、面向过程的软件系统。

1.2　面向对象的程序设计方法

程序设计语言是计算机科学中一个不断变化的领域。有的语言开始退出舞台，让位给新的语言（如 Smalltalk，ML，Prolog）。有的语言，如 FORTRAN 则不断地演化发展着，以巩固自己的地位。同时，人们又力求开发出更好的程序设计语言。

结构化程序设计技术是 20 世纪 70 年代研究的中心问题，而今天的热点则是面向对象程序设计语言，如 Smalltalk，Traits，Eiffel 和 Java 等语言都是面向对象的程序设计语言。C++则是在标准 C 语言的基础上，引入“面向对象”的概念而扩充形成的混合型面向对象语言。

面向对象的程序设计方法不是以函数过程和数据结构为中心，而是以对象代表求解问题的中心环节。它追求的是现实问题空间与软件系统解决空间的近似和直接模拟。这就改变了原来计算机程序的分析、设计和实现的过程与方法之间的脱节和跳跃状态，从而使人们对复杂系统的认识过程与系统的程序设计实现过程尽可能地一致。

面向对象方法的产生是计算机科学发展的要求。20 世纪 80 年代，特别是 90 年代以来，软件的规模进一步扩大，对软件可靠性和代码可重用性的要求也进一步提高。就是在这样的背景下，面向对象的程序设计方法应运而生。与传统的程序设计方法相比，面向对象的程序设计具有抽象、封装、继承和多态性等特征。

“面向对象”不仅仅作为一种技术，更作为一种方法论贯穿于软件设计的各个阶段。面向对象的技术在系统程序设计、数据库及多媒体开发等领域都得到广泛应用。专家们预测，面向对象的程序设计思想将会主导今后程序设计语言的发展。

【例 1.2】给出输入三角形的 3 个顶点坐标，计算 3 条边的长度的面向对象的算法描述。

根据“问题域”设计程序模块。这里首先从顶点坐标考虑，顶点坐标就是一个点，通过这个点，可以得到它与另一个点的距离，从而将平面抽象为点对象的集合。三角形的 3 个顶点就是点类的 3 个实际对象。求三角形的 3 条边长，就是求每两个点对象之间的距离。因此，重点应放在如何描述这个点类上。

设计 point 类，这个类具有两个坐标值（x,y），称为类的属性。为类设计一个与类同名的特殊函数 point，point 表现了类的特定行为，即用来初始化这个点的属性值，能使不同点的对象具有不同的 x 和 y 值（也就是不同的属性值）。这个点类能向外界提供自己的属性值，并能计算它与另一个对象之间的距离，可像图 1-2 那样描述这个类。

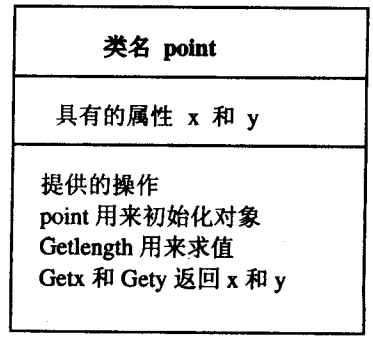

图 1-2　point 类示意图

第 1 个方框中是类名，第 2 个方框中是坐标点的数据，称为属性（或称数据成员）。第 3 个方框中表示类所提供的具体操作方法，实际上是如何使用数据 x 和 y，以实现预定功能的函数，这里称为成员函数。point 是一种特殊成员函数，称为类的构造函数。真正的含意是说可以用它们初始化类的数据成员的值，从而构造出类的一个具体的对象。

为了简单，假设只有一个带有两个参数的 point 函数（以后将会看到，类 point 可以有多个功能各异的同名成员函数），如果需要定义对象 A，使用的方法如下：

point A（x1,y1）;

这里 x1 和 y1 是对象 A 的坐标。对于 A 和 B 之间的距离，既可以表示为 A.Getlength（B），也可表示为 B.Getlength（A）。这个道理很明显，只要表示为两个点的对象即可。至于是 A 发出求 A 与 B 之间的距离的消息，还是 B 发出求 B 与 A 之间的距离的消息，则是无关紧要的。它们是调用同一个消息处理函数 GetLength。算法描述为：

输入：3 个类的对象。

输出：任意两个对象之间的距离。

算法设计：

设计类 point

类的结构图如图 1-2 所示

产生 3 个对象。

point A（x1,y1） ←对象 A

point B（x2,y2） ←对象 B

point C（x3,y3） ←对象 C

计算对象之间的距离

AB ← A.Getlength（B）

AC ← A.Getlength（C）

BC ← B.Getlength（C）

输出（AB，AC，BC）

算法结束

求解的中心环节是以点这个客观世界的对象为依据，这正符合人们对客观世界的认识。

所描述的这个点的类很稳定，可以用它来描述其他对象。例如，从点出发，增加一个点组成一条线段，或增加一条半径来描述一个圆。经验证明，对任何软件系统而言，其中最稳定的成分是对应的问题域。与功能相比，一个问题域中的对象一般总能保持相对稳定性，因而以面向对象方式构造的软件系统的主体结构也具有较好的稳定性和可重用性。因此，采用“消息+对象”的程序设计模式，具有满足软件工程发展需要的更多优势。

结构化程序设计的提出和发展主要是为了解决日益复杂和规模庞大的软件开发的要求。随着软件的进一步庞大和复杂，随着编程实践的深入，人们发现，当软件复杂到一定程度时，结构化程序设计也不足以满足需要。一般说来，当软件的规模在三四万行以内时，结构化的方法还是可以满足需要的，当程序的规模超过这个尺度时，开发和维护就会变得越来越困难。发生这种情况的根本原因是结构化程序设计方法与客观世界以及人们的分析思考方式都非常不一致。这种不一致实际上是不合理的一种表现。这种不合理性的一个具体结果是，结构化程序设计中的分而治之的想法尽管非常好，但在结构化程序设计语言和

结构化程序设计方法下却难以贯彻到底。比如，结构化程序设计要求尽量不用全局变量，但当程序规模大到一定程度时，以功能抽象为基础的结构化程序几乎不可避免地引入大量的全局变量。也就是说，实际上已经没有办法满足结构化程序设计的基本要求。而在面向对象程序设计中，可将一组密切相关的函数统一封装在一个对象中，以便能合理而有效地避免全局变量的使用。可以认为，面向对象方法更彻底地实现了结构化程序设计的思想。

结构化程序设计使用的是功能抽象，面向对象程序设计不仅能进行功能抽象，还能进行数据抽象。“对象”实际上是功能抽象和数据抽象的统一。

面向对象的早期，软件界将精力集中于用面向对象解决代码编写（即 OOP 阶段）问题。因此，这方面是成熟得最早的，也是一开始就很有说服力的。面向对象程序大大提高了软件的重用性，其代码更容易理解，更容易维护，也更优美。当 OOP 取得了很大成功以后，软件界信心大增，人们把眼光投向了更广阔的领域，面向对象几乎被当成了软件界的万能良药。在这样的背景下，面向对象分析（OOA）和面向对象设计（OOD）成了人们研究的重点，一直到 20 世纪 90 年代中期，这方面的方法和技术才真正统一。1992 年 OMG（面向对象管理组）制定的面向对象分析和设计的国际标准 UML（unified modeling language）问世。面向对象技术应用又达到了一个新的水平。

时至今日，各种面向对象的技术和方法仍在发展中。需强调的是，面向对象语言和面向对象编程不能简单地等同起来。正像不能说采用结构化程序设计语言编写的程序一定符合结构化编程的特点一样，并不能说采用面向对象语言的编程就一定具有充分的面向对象特性。

从本节的例子可见，在面向对象程序设计技术中，对象是具有属性（又称状态）和操作（又称方法、行为方式和消息等）的实体。

1.3　面向对象语言的发展

面向对象程序语言发展的主要里程碑是 Smalltalk 语言，它完整地体现并进一步丰富了面向对象的概念。1980 年美国 Xerox Palo Alto 研究中心推出了 Smalltalk 后，相继开发了配套工具环境，使之走向实用，其缺点是人们需要从头学习一门全新的语言。在 20 世纪 80 年代中期，面向对象语言已形成几大类别：一类是纯面向对象的语言，如 Smalltalk 和 Eiffel；另一类是混合型的面向对象语言，如 C++和 Objective C；还有一类是与人工智能语言结合形成的，如 LOOPs、Flavors 和 CLOS；适合网络应用的有 Java 等。

面向对象是一个认识论和方法学的基本原则。人对客观世界的认识和判断，常常采用由一般到特殊（演绎法）和由特殊到一般（归纳法）的两种方法，这实际上是对认识判断的问题域对象进行分解和归类的过程。

面向对象分析（object-oriented analysis，OOA）是面向对象软件工程方法的第一个环节，它包括一套概念原则、过程步骤和归类的过程。OOA 的任务是采用面向对象方法，把对问题域和系统的认识理解，正确地抽象为规范的对象（包括类，继承层次）和消息连接关系，形成面向对象模型，为后续的面向对象设计（object-oriented design，OOD）和面向对象编程（object-oriented program，OOP）提供指导。而且 OOA 与 OOD 能够自然地过渡和结合，也是面向对象方法值得称道的一个优点。OOA 和 OOD 的区别主要是前者与系统