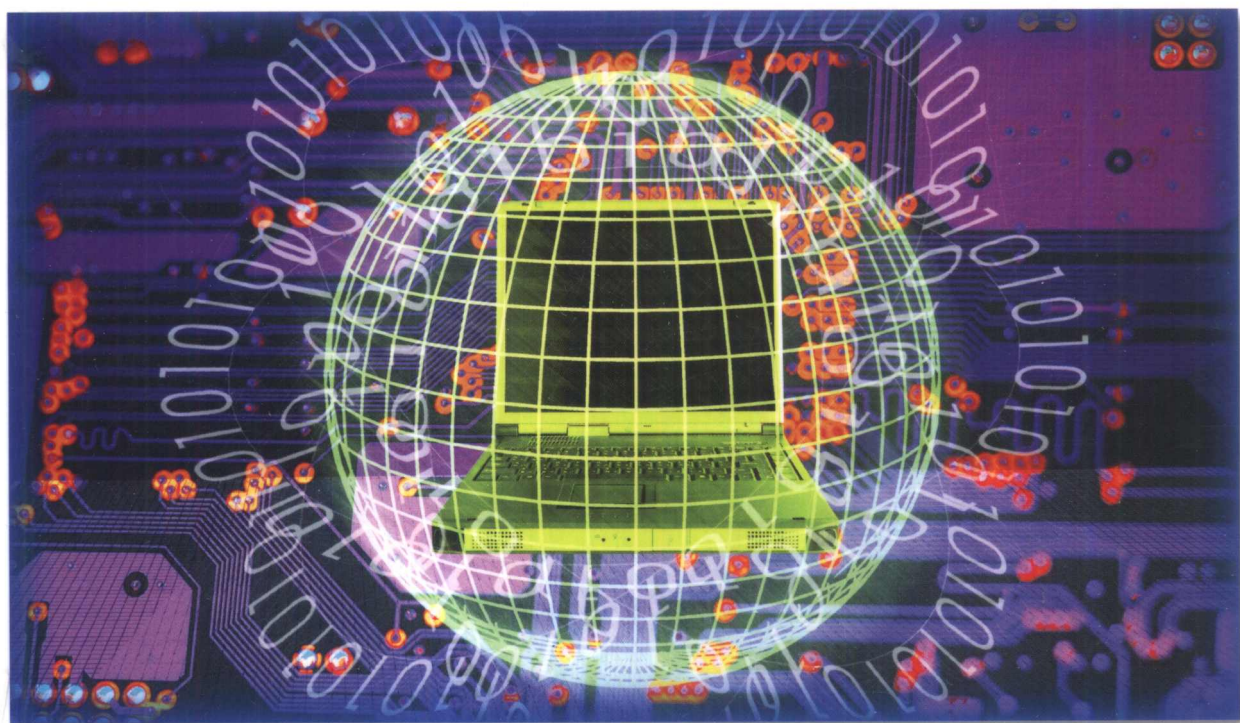


21 世纪高等院校计算机系列教材

Visual C++

程序设计与软件技术基础

马建红 沈西挺 主编 范贻明 主审



36

21 世纪高等院校计算机系列教材

Visual C++ 程序设计与软件技术基础

马建红 沈西挺 主编

范贻明 主审



A1029847

中国水利水电出版社

内 容 提 要

Visual C++功能强大,使用灵活,但对于学习程序设计的人来说是较难深入的。故为进一步提高学习 VC 编程人员的计算机软件专业知识水平和编程能力,本书采用循序渐进逐步深入的方法,对 VC 的基础、Windows 编程及面向对象程序设计方法进行了深入的讲解,同时介绍了软件工程及数据结构等计算机软件专业基础知识,使读者能够融会贯通地了解计算机软件基础知识,而且易学易懂。

该书内容包括四大部分:第一部分为 C++语言基础及面向对象程序设计;第二部分为 Visual C++的 Windows 编程基础;第三部分为数据结构;第四部分为软件工程。

本书是学习计算机软件设计的基础教程。既可作为大专院校各专业高级程序设计的正式教材,又可作为研究生计算机软件基础教育的教材,另外还可供计算机培训班或读者自学使用。

图书在版编目(CIP)数据

Visual C++程序设计与软件技术基础/马建红等主编. —北京:中国水利水电出版社, 2002

(21 世纪高等院校计算机系列教材)

ISBN 7-5084-1100-5

I .V... II.马... III.C 语言—程序设计—高等学校—教材 IV.TP312

中国版本图书馆 CIP 数据核字 (2002) 第 057059 号

书 名	Visual C++程序设计与软件技术基础
主 编	马建红 沈西挺
主 审	范贻明
出版、发行	中国水利水电出版社(北京市三里河路 6 号 100044) 网址: www.waterpub.com.cn E-mail: mchannel@public3.bta.net.cn (万水) sale@waterpub.com.cn
经 售	电话: (010) 68359286 (万水)、63202266 (总机)、68331835 (发行部) 全国各地新华书店
排 版	北京万水电子信息有限公司
印 刷	北京市天竺颖华印刷厂
规 格	787×1092 毫米 16 开本 19.75 印张 434 千字
版 次	2002 年 8 月第一版 2002 年 8 月北京第一次印刷
印 数	0001—5000 册
定 价	26.00 元

凡购买我社图书,如有缺页、倒页、脱页的,本社发行部负责调换

版权所有·侵权必究

前 言

信息技术与计算机科学的飞速发展,促使计算机基础教育要不断发展和创新。教育部相应提出了“计算机文化基础”、“计算机技术基础”和“计算机应用基础”三个层次的教学课程体系,及面向 21 世纪的计算机基础教育问题。在“计算机技术基础”第二层次的教育中,要求学生掌握一门程序设计语言。自 2000 年以来全国高校“计算机文化基础”的内容都已进入到了 Windows 的图形界面平台,作为后续课的“程序设计”再返回到 DOS 界面下讲解调试 Fortran、Pascal、C 等语言已不受青年教师和高年级学生的欢迎。因此我们认为“程序设计”课程应尽早进入到 Windows 平台讲授。

为满足飞速增长的非计算机专业的全体研究生的要求,及理、工类高年级本科生提高程序设计水平的要求,我们编写了《Visual C++程序设计与软件技术基础》一书。虽然市场上关于 Visual C++的书籍很多,但是适合以上读者群体的高校教材尚觉缺少,因此我们根据近几年的实际教学经验选定了本书的结构和内容。

学生在掌握了“计算机文化基础”、Fortran、Basic、C 等内容的基础上提高到面向对象的程序设计水平,同时补足关于软件工程和数据结构的基础知识。

本书的特点是要读者掌握 Visual C++最基本、最主要的内容,从而能结合本人专业独立设计和调试出高水平的面向对象的程序。本书共分四部分:

第一部分: C++语言及面向对象程序设计

第二部分: Visual C++的 Windows 编程

第三部分: 数据结构

第四部分: 软件工程

参加本书编写的有马建红、沈西挺、贺士娟、孙卫忠、魏明军、史巧硕、安杰、杨素梅、李艳、李娟、薛美云、刘丽梅、张平、李琳、马岱等。全书由范贻明教授审阅,由于时间紧迫和水平有限,特别是作为研究生的基础课教材,难免有不足不妥之处,请广大读者和专家批评指正。

编者

2002年6月于天津

目 录

前言

第一部分 C++语言及面向对象程序设计

第 1 章 概述	1
1.1 面向对象程序设计	1
1.1.1 面向对象程序设计的发展	1
1.1.2 面向对象程序设计语言	1
1.1.3 C++对面向对象技术的支持	2
1.1.4 Visual C++的发展	3
1.2 简单的 C++程序	3
1.2.1 C++程序	3
1.2.2 C++的头文件	4
1.2.3 源程序	4
1.3 程序的调试与运行	6
1.3.1 启动 Visual C++	6
1.3.2 编辑源文件	6
1.3.3 编译源程序	10
1.3.4 链接目标程序, 形成可执行文件	10
1.3.5 运行程序	11
第 2 章 C++语言的基本元素	12
2.1 字符集与标识符	12
2.1.1 字符集	12
2.1.2 标识符	12
2.1.3 关键字	13
2.2 数据类型	13
2.3 常量与变量	14
2.3.1 常量	14
2.3.2 变量	17
2.4 运算符与表达式	18
2.4.1 算术运算	19

2.4.2	关系运算	20
2.4.3	逻辑运算	21
2.4.4	赋值运算	22
2.4.5	逗号运算	22
2.4.6	条件表达式.....	23
2.4.7	类型转换	23
2.5	C++的输入输出流	24
2.5.1	C++的无格式输入输出	24
2.5.2	输入输出格式控制.....	25
2.6	语句	28
2.6.1	条件语句	28
2.6.2	循环语句	31
2.6.3	转移语句	33
2.7	函数	34
2.7.1	函数的定义与说明.....	35
2.7.2	函数调用	36
2.7.3	参数传递机制.....	37
2.7.4	return 语句.....	38
2.7.5	嵌套调用与递归函数.....	39
2.7.6	内联函数	40
2.7.7	带缺省参数的函数.....	41
2.7.8	函数重载	41
2.8	程序结构.....	42
2.8.1	作用域	42
2.8.2	全局变量和局部变量.....	44
2.8.3	生命期	45
2.8.4	编译预处理.....	46
	习题 2	48
第 3 章	构造类型.....	50
3.1	数组	50
3.1.1	一维数组	50
3.1.2	二维数组	53
3.1.3	字符数组与字符串.....	54
3.2	指针	57
3.2.1	指针	57
3.2.2	指针与数组.....	61
3.2.3	多级指针与指针数组.....	63

3.2.4	数组指针	65
3.2.5	指针与函数	66
3.3	引用	71
3.3.1	引用的定义与使用	71
3.3.2	引用与函数	72
3.4	结构体	73
3.4.1	结构体与结构体变量	73
3.4.2	结构体数组	76
3.4.3	结构体指针	77
3.5	内存管理	79
习题 3		81
第 4 章	类的构造	82
4.1	面向对象程序设计术语	82
4.2	类	84
4.2.1	类的定义	85
4.2.2	内联的成员函数	86
4.3	对象	87
4.3.1	对象的创建	87
4.3.2	成员的访问	87
4.4	构造函数与析构函数	89
4.4.1	构造函数	89
4.4.2	析构函数	92
4.5	对象数组与对象指针	93
4.5.1	定义对象数组	93
4.5.2	访问对象数组元素的成员	95
4.5.3	对象指针	96
4.6	类的拷贝构造函数	98
4.7	对象与函数	100
4.7.1	对象做函数参数	100
4.7.2	对象指针做函数参数	102
4.7.3	对象的引用做函数参数	103
4.8	类的静态成员	103
4.9	对象作为类的数据成员	104
4.9.1	对象作为类的数据成员举例	104
4.9.2	对象成员的初始化	106
4.10	this 指针	108
4.11	关于作用域	108

4.11.1 类的作用域.....	108
4.11.2 对象的作用域.....	110
4.12 友元.....	110
4.12.1 友元函数的定义.....	110
4.12.2 友元函数的作用.....	110
4.13 类模板.....	115
4.13.1 类模板的定义.....	115
4.13.2 定义类模板对象.....	116
4.14 程序举例.....	117
习题 4	118
第 5 章 派生类	121
5.1 派生类的定义.....	121
5.1.1 定义派生类.....	121
5.1.2 继承基类成员.....	122
5.1.3 派生类中成员的重定义.....	124
5.1.4 将派生类声明为基类的友元类.....	125
5.2 派生类的构造函数.....	125
5.2.1 定义派生类的构造函数.....	125
5.2.2 派生类的构造函数与基类的构造函数的调用顺序.....	126
5.3 基类与派生类之间的转换.....	128
5.4 虚函数.....	130
5.4.1 多态性	130
5.4.2 虚函数	132
5.4.3 实现运行时多态性的原理.....	133
5.4.4 虚函数与派生类.....	135
5.4.5 虚析构函数.....	138
5.5 抽象基类.....	139
5.6 程序举例.....	141
习题 5	142
第 6 章 运算符重载	145
6.1 重载运算符.....	145
6.1.1 重载运算符的意义.....	145
6.1.2 运算符重载形式.....	146
6.2 可重载的运算符.....	148
6.3 重载运算符举例.....	149
习题 6	151
第 7 章 I/O 流.....	152

7.1 标准 I/O 流	152
7.1.1 输入成员函数	152
7.1.2 输出成员函数	153
7.1.3 测试 I/O 成员函数	154
7.2 文件 I/O 流	154
7.2.1 文件的打开与关闭	155
7.2.2 文件的读写	156
习题 7	157

第二部分 Visual C++ 的 Windows 编程

第 8 章 Visual C++ 创建 Windows 程序	158
8.1 Windows 应用程序	158
8.1.1 Windows 编程特点	158
8.1.2 Windows 程序的消息驱动	159
8.1.3 Windows 应用程序的组成	159
8.1.4 源程序组成结构	160
8.1.5 编写一般的 Windows 应用程序	161
8.2 利用 MFC 生成 Windows 程序	162
8.2.1 MFC 类库简介	162
8.2.2 利用 MFC 创建 Windows 应用程序	163
8.2.3 分析 MFC 编写的 Windows 程序的执行过程	167
8.2.4 Visual C++ 应用程序向导 AppWizard	169
习题 8	173
第 9 章 对话框和控件	175
9.1 对话框	175
9.1.1 模式对话框和非模式对话框	175
9.1.2 对话框的组成	175
9.1.3 对话框类 CDialog	175
9.1.4 CWnd 类提供的管理对话框的成员函数	177
9.2 对话框的设计	177
9.2.1 使用 ClassWizard 建立对话框类	177
9.2.2 对话框类成员变量的加入	178
9.3 创建基于对话框的应用程序	178
9.3.1 使用文本控件	178
9.3.2 使用按钮控件	186
9.3.3 使用列表控件	191
习题 9	197

第 10 章 菜单	198
10.1 菜单概述	198
10.1.1 菜单资源概述	198
10.1.2 建立菜单及菜单消息映射	200
10.2 菜单程序设计	201
10.2.1 对话框应用程序中的菜单设计实例	201
10.2.2 文档/视图结构中应用程序的菜单设计实例	205
10.2.3 应用程序中快捷菜单设计实例	206
习题 10	207
第 11 章 文档和视图结构	208
11.1 MFC 应用程序的四大对象	208
11.2 文档类和视图类	210
11.2.1 文档类及其派生类	210
11.2.2 视图类及其派生类	212
11.3 文档/视图结构编程举例	213
11.3.1 单文档/视图结构举例	213
11.3.2 多文档/视图结构的应用	215
习题 11	220
第 12 章 绘图	221
12.1 图形设备接口	221
12.1.1 设备描述表 DC (Device Content)	221
12.1.2 设备描述表类	222
12.2 画笔和画刷	222
12.2.1 选定 GDI 对象	222
12.2.2 画笔	223
12.2.3 画刷	224
12.3 绘图函数	224
12.4 绘图示例	225
习题 12	226

第三部分 数据结构

第 13 章 数据结构概念与算法	227
13.1 数据结构的有关概念	227
13.1.1 几个基本概念	227
13.1.2 数据结构的有关概念	228
13.2 算法及其算法分析	230
13.2.1 算法的概念及特性	230

13.2.2	算法设计的要求.....	231
13.2.3	算法分析.....	232
习题 13		233
第 14 章	线性逻辑结构	234
14.1	线性表.....	234
14.1.1	线性表的概念和基本特性.....	234
14.1.2	线性表的顺序存储结构及其某些运算.....	235
14.1.3	线性表的链式存储结构及其某些运算.....	237
14.2	栈和队列.....	240
14.2.1	栈.....	240
14.2.2	队列.....	241
14.3	串和数组.....	242
14.3.1	串.....	242
14.3.2	数组.....	243
14.3.3	特殊矩阵.....	244
习题 14		245
第 15 章	数据的非线性逻辑结构——树	246
15.1	树、根树的相关概念.....	246
15.1.1	树的概念、特性等.....	246
15.1.2	根树的概念及其特性.....	246
15.2	根树的存储结构和遍历.....	248
15.2.1	根树的几种常用的存储结构.....	248
15.2.2	根树和森林的遍历.....	249
15.3	二叉树.....	250
15.3.1	二叉树的概念及性质.....	250
15.3.2	二叉树的存储结构.....	251
15.3.3	二叉树的遍历.....	252
15.3.4	二叉树与一般根树、森林的相互转换.....	253
15.4	Huffman 树.....	254
15.4.1	Huffman 树的概念.....	254
15.4.2	Huffman 树的构造.....	255
15.4.3	Huffman 树的应用.....	255
习题 15		256
第 16 章	数据结构的工程应用——查找和排序	257
16.1	查找.....	257
16.1.1	关于查找的一些基本概念.....	257
16.1.2	静态查找表查找的一些基本方法.....	258

16.1.3	动态查找表查找.....	261
16.1.4	关于 Hash 表查找的简述	262
16.2	排序.....	263
16.2.1	关于排序的基本概念.....	263
16.2.2	基于比较的几种内部排序方法.....	264
16.2.3	不需要比较的内部排序法——基数排序.....	266
习题 16		266

第四部分 软件工程

第 17 章	软件工程概述	268
17.1	软件与软件危机.....	268
17.1.1	软件的概念及其特点.....	268
17.1.2	软件的发展与软件危机.....	269
17.2	软件工程与软件工程目标.....	270
17.2.1	软件工程的基本概念.....	270
17.2.2	软件工程的基本目标.....	271
17.3	软件生存期和生存期模型.....	271
17.3.1	软件生存期.....	271
17.3.2	软件生存期模型.....	272
习题 17		273
第 18 章	可行性研究与软件计划	274
18.1	可行性研究的目的和内容.....	274
18.2	可行性研究的主要步骤.....	275
18.3	可行性研究的描述工具与软件计划.....	276
18.3.1	可行性研究的描述工具.....	276
18.3.2	软件计划.....	277
习题 18		277
第 19 章	软件的需求分析.....	278
19.1	需求分析阶段的基本任务.....	278
19.2	需求分析的结构化方法.....	279
19.2.1	结构化分析方法的基本思想和原则.....	279
19.2.2	结构化分析方法的基本步骤.....	280
19.3	需求分析的面向对象方法.....	281
19.4	结构化分析的描述工具——数据流图.....	282
19.5	数据流图的配套工具——数据词典.....	284
19.5.1	数据流图与数据字典在系统需求描述上的关系.....	285
19.5.2	数据字典条目的分类与编写说明.....	285

习题 19	286
第 20 章 软件设计	287
20.1 软件系统的总体设计	287
20.1.1 模块及其模块的划分原则	287
20.1.2 系统总体设计的结构化设计方法	289
20.1.3 系统总体设计阶段的文件提交及复审	291
20.2 软件系统的详细设计	291
习题 20	292
第 21 章 软件的物理实现——编码	293
21.1 程序设计语言与编程风格	293
21.1.1 程序设计语言的特性	293
21.1.2 程序设计语言的选择考虑	294
21.1.3 程序设计风格问题	294
21.2 结构化程序设计方法	294
21.3 面向对象的程序设计方法	294
习题 21	296
第 22 章 软件测试与维护	297
22.1 软件测试	297
22.1.1 软件测试的任务和测试的出发点	297
22.1.2 软件测试的基本原则	297
22.1.3 软件测试的基本手段	297
22.1.4 软件测试的几个基本步骤	298
22.2 软件维护	300
22.2.1 软件维护的概念及维护的动因	300
22.2.2 软件维护的种类	301
22.2.3 软件维护的步骤	301
22.2.4 软件的可维护性	302
22.2.5 提高软件可维护性的几个方面	302
22.2.6 维护的管理	302
习题 22	302
参考文献	303

第一部分 C++语言及面向对象程序设计

第 1 章 概述

1.1 面向对象程序设计

1.1.1 面向对象程序设计的发展

20 世纪 50 年代的程序设计都是用机器语言或汇编语言编写的，使程序的设计相当麻烦，严重影响了计算机的普及应用。随着计算机的应用日益广泛，发展了一系列不同风格的、为不同对象服务的程序设计语言，如 Fortran 语言、ALGOL、COBOL、C、Pascal、Ada、Lisp、Prolog、FFP 等。高级语言的蓬勃发展，使编译和形式语言理论日趋完善，但就整个程序设计方法而言，并无实质性的改善。自 20 世纪 60 年代末到 70 年代初，出现了大型的软件系统，如操作系统、数据库等，这给程序设计带来了新的问题。大型软件系统的研制需要花费大量的人力和物力，但编写出来的软件可靠性差，错误多，难以维护，已经到了程序员无法控制的地步，这就是“软件危机”。

在 1969 年，E.W.Dijkstra 首先提出了结构化程序设计的概念，他强调了从程序结构和风格上研究程序设计，为“软件危机”起了很大的缓解作用。到 20 世纪 70 年代末，结构化设计方法得到了很大的发展，Niklaus Wirth 又提出了“算法+数据结构 = 程序设计”，程序设计方法，将软件划分成若干个可单独命名和编址的部分，它们被称为模块，模块化使软件能够有效地被管理和维护，能够有效地分解和处理复杂问题。在 20 世纪 40 年代，模块化程序设计方法普遍被人们接受。

软件开发是对问题的求解过程，从认识论角度看，软件开发过程包括人们对要解决问题及相关事物的认识和基于认识所进行的描述。而结构化设计方法不能直接反映出人类认识问题的过程。随着软件系统越来越复杂庞大，结构化程序设计方法已显力不从心。

20 世纪 40 年代，在软件开发中各种概念和方法积累的基础上，就如何超越程序的复杂性障碍，如何在计算机系统中自然的表示客观世界等问题，人们提出了面向对象的设计方法，它不是以过程为中心，而是以对象代表的问题解为中心环节，提出了“对象+对象+=程序设计”理论，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。

1.1.2 面向对象程序设计语言

早在 20 世纪 60 年代，就出现了最早的面向对象语言 Simula67 语言，具有了类和对象

的概念，随后又推出了纯面向对象设计语言。如 40 年代美国 Xerox Palo Alto 研究中心推出了 Smalltalk，它完整地体现并进一步丰富了面向对象的概念。这时面向对象语言已形成几大类别：

(1) 纯面向对象语言，如 Smalltalk 和 Eiffel。

(2) 混合型面向对象语言。这类语言一般是在其他语言的基础上开发出来的面向对象的语言，如 C++ 和 ObjectiveC。

(3) 与人工智能语言结合形成的语言，如 LOOPS、Flavors 和 CLOS。

(4) 适合网络应用面向对象语言，如 Java 语言等。

目前面向对象方法被广泛应用，C++ 语言成为当前最流行的面向对象语言之一。它是由 AT&T 公司的贝尔实验室的 Bjarne Stroustrup 博士开发的，是一门高效实用的混合型程序设计语言，它最初的设计目标是：支持面向对象编程技术；支持抽象形态的类；更好的 C 语言。C++ 语言包括两部分：一部分是 C++ 基础部分，它是以 C 语言为核心的；另一部分是 C++ 面向对象特性部分，是 C++ 对 C 语言的扩充部分。这样它既支持面向对象程序设计方法，又支持结构化程序设计方法，同时广泛的应用基础和丰富的开发环境的支持，也使面向对象设计得到很快普及。

1.1.3 C++ 对面向对象技术的支持

C++ 作为一种面向对象程序设计语言，具有对象、类、方法、消息等概念，同时支持面向对象方法的封装机制和继承、多态性机制。

1. C++ 对封装机制的支持

封装有两个含义：一是把属性（数据结构）同服务（操作数据的函数）结合在一起；如有一组数据是准备画一图形所需的信息，在传统的 C 语言中要将数据与画图程序分开处理，而在 C++ 语言中通过 class 重新定义类，将数据与操作数据的函数结合起来。封装的另外一个含义是尽可能的将对象的内部数据进行隐藏。这样在对象之外，只可以看到对象的行为，而不能看到对象的细节，也就不可能修改对象的内部信息，以保证对象的独立性。C++ 语言定义类时，可用访问权限关键字 private（私有的）、protected（保护的）、public（公有的）限制对象内部数据的访问权限。

2. C++ 对继承机制的支持

在 C++ 中增加了一种重要的机制——继承和派生。可以从一个或多个已经定义的基类中继承属性和服务，同时也可以重新定义或加进新的数据和行为，这样建立起类的层次结构。

3. C++ 对多态性机制的支持

多态性有静态和动态之分。静态多态性是在编译阶段由编译程序根据操作数据从多个同名函数中选择响应的调用；动态多态性是在执行阶段才根据产生的信息确定需要调用的函数，调用不同的函数就意味着执行不同的处理方式。在 C++ 中，对静态多态性的支持是通过提供对函数的重载功能，命名相同的函数，其实现的功能不同，C++ 会依据所给定的函数参数来选择相应的功能函数。C++ 通过派生类、虚函数、对象指针等概念完成动态多态性功能。

1.1.4 Visual C++的发展

随着 C++ 逐渐成为 ANSI 标准, 这种新的面向对象程序设计语言迅速成为程序员最广泛使用的工具。众多 C++ 语言的开发环境也随之不断地推出, 竞争十分激烈。1986 年 Borland 公司开发了 Turbo C++ 程序语言, 而后又推出了 Borland C++ 版本。Microsoft 公司于 20 世纪 80 年代中期在 Microsoft C 6.0 的基础上开发了 Microsoft C/C++ 7.0, 同时引进了 Microsoft Foundation Class (MFC) 库 1.0 版本, 完善了源代码。以前这些版本都是依赖于 DOS 环境, 或在 Windows 下的 DOS 模式下运行。不久 Microsoft 公司推出的 Microsoft C/C++ 8.0, 即 Visual C++ 1.0 版本, 它是 Microsoft 公司推出的第一个真正的基于 Windows 环境下的可视化的集成开发环境, 将编辑、编译、链接和执行集成为一体。从 Visual C++ 1.5 版本以后, Microsoft 公司决定不再将更多的努力花在支持 16 位编程上, 虽然 Visual C++ 2.0 仍提供对 16 位的支持, 但从 2.0 版本以后, Visual C++ 更多的用来创建 32 位程序。在版本上, Microsoft 公司没有推出 3.0 版本, 版本号直接从 2.0 跳到 4.0, 这样, Visual C++ 与 MFC 的版本号取得一致。由于 Internet 的流行明显的影响了产品的设计, 在 4.0 版本中, Visual C++ 引进了为 Internet 编程而设计的新类库。5.0 版本也增加了一些新类, 但注意力更多的集中在改善产品的界面上, 以提供一个更好的在线帮助系统、更高级的宏能力和对在开发者组内进行类和其他代码共享的支持。6.0 版本在功能上做了进一步的改进。Visual C++ 经历了从 1.0 到 6.0 等版本的发展, 软件系统逐渐庞大, 功能日益完善。

1.2 简单的 C++ 程序

1.2.1 C++ 程序

下面通过一个简单的例子认识 C++ 程序。程序名为 hello.cpp。

例 一个简单的 C++ 程序。

```
#include <iostream.h>
void student(char *s)
{
    cout << "I am a"<<s;
}
void main()
{
    cout << "hello world!"<<endl;    // 输出字符串 "hello world!"
    student("student");              // 调用 student 函数
}
```

运行该程序, 在屏幕上出现: hello world!

I am a student

C++ 程序的书写格式比较自由, 一行内可以写几句, 也可以将一条语句分写在多行上 (字符串不能分行写)。为了程序书写的比较清晰, 一般采用缩进格式。

C++ 的程序是由一个或多个源代码文件构成。C++ 的源文件分两类: 头文件和源程序。

1.2.2 C++的头文件

为了使程序的结构清晰，一般将类型及类的定义、函数的说明等放在一个源代码文件中，成为头文件（扩展名为.h 或.H）。头文件可以由系统提供，用户可直接使用。如程序中包含的 `iostream.h` 文件是 C++ 系统提供的定义了标准输入、输出流操作的头文件；也可以由用户根据程序需要自己编写头文件。

1.2.3 源程序

将变量、类型的定义及函数的声明放在头文件中，而使用这些变量或函数的程序放在另一个文件中，成为 C++ 源程序文件，扩展名为.CPP。

C++ 源程序中包括编译预处理指令、注释和函数体等内容。

1. 注释

注释是程序员对程序语句所做的说明，是提高程序的可读性的一种手段。C++ 的注释语句可用“//”两个符号注释从该符号开始到该行的结束。也可用一对“/*”和“*/”符号成对使用，用于注释从“/*”开始到“*/”结束的任意多行语句。在生成执行程序时，C++ 忽略注释，并把每一个注释都看做是空格。

2. 编译预处理命令

每个以“#”符号开头的行，都是编译预处理行，并且在一行中，只能写一行编译预处理命令。

在例 1.1 程序中 `#include <iostream.h>` 为文件包含命令。文件包含是指一个源文件可以包含另外一个源文件的全部内容。在 C++ 中用 `#include` 命令实现。文件包含形式：

```
#include <文件名>
```

或

```
#include "文件名"
```

在文件包含命令中使用尖括号或引号，可使预处理程序在查找包含文件时采用的策略不同。使用尖括号时，预处理程序在系统规定的文件夹（通常是 `include` 文件夹）下寻找包含文件，若使用引号，预处理程序首先在当前文件夹下查找包含文件，若找不到，再查找系统规定的文件夹。

例如，在头文件 `head.h` 中定义了变量 `MAX` 等，源程序 `file.cpp` 中包含 `head.h` 文件后，就可以使用 `head.h` 文件中的变量。

