高职高专

高职高专工作过程·立体化创新规划教材

——计算机系列

C++程序设计与应用教程

梁　明　黄昊宇　主　编
王继民　严云洋　副主编
冷育荣　主　审

赠送电子课件

- 以培养技能型创新人才为目标，设置丰富的版块合理安排全文，突出实用性和可操作性。
- 以工作过程为导向，全面展示案例实施的全过程，提炼技术要点，即学即用面向就业。
- 以强化实际操作技能为主线，答疑解惑，解决工作实践中的常见问题。

清华大学出版社

高职高专工作过程·立体化创新规划教材——计算机系列

C++程序设计与应用教程

梁　明　黄昊宇　主　编
王继民　严云洋　副主编
冷育荣　主　审

清华大学出版社
北　京

内 容 简 介

本书系统、全面地介绍了标准 C++的基础知识。全书分为 11 章，包括 C++语言概述，数据类型、运算符和表达式，基本控制结构，函数，自定义数据类型和预处理，类和对象，继承和多态，运算符重载，模板，C++流和异常处理等内容。

本书以“工作场景导入”→“知识讲解”→“回到工作场景”→“工作实训营”为主线编写，讲解了解决任务所需要的各个知识点，再通过所学的知识点来完成任务，并且在每一章都有配套的实训和知识拓展，以便于读者掌握各章的重点及提高操作能力。书中包含了大量的应用实例，且示例代码具有统一、良好的编程风格，能够对读者起到很好的引导性作用。

本书针对 C++初学者，从 C 语言基础知识开始介绍，然后在此基础上详细阐述了 C++的新特性，因此并不要求读者有较多的 C 语言方面的背景知识。本书既可作为高职高专院校计算机及相关专业的教材，也可作为各类培训班的培训教程，以及供初学者自学 C++语言时使用。

图书在版编目(CIP)数据

C++程序设计与应用教程/梁明，黄昊宇主编；王继民，严云洋副主编；冷育荣主审. --北京：清华大学出版社，2012.1
(高职高专工作过程·立体化创新规划教材——计算机系列)
ISBN 978-7-302-27007-2

Ⅰ. ①C…　Ⅱ. ①梁…　②黄…　③王…　④严…　⑤冷…　Ⅲ. ①C 语言—程序设计—高等职业教育—教材　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2011)第 200692 号

责任编辑：章忆文　杨作梅
装帧设计：山鹰工作室
责任校对：周剑云
责任印制：何　芊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京市人民文学印刷厂
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：26.5　**字　数**：641 千字
版　　次：2012 年 1 月第 1 版　　**印　次**：2012 年 1 月第1 次印刷
印　　数：1～4000
定　　价：43.00 元

产品编号：035618-01

丛 书 序

高等职业教育强调“以服务为宗旨，以就业为导向，走产学结合发展道路”。这也体现了高等职业教育的本质，是其适应社会发展的必然选择。服务社会、促进就业和提高社会对毕业生的满意度，是衡量高等职业教育是否成功的重要指标。为了提高高职院校的教学质量，培养符合社会需求的高素质人才，我们计划打破传统的高职教材以学科体系为中心，讲述大量理论知识，再配以实例的编写模式，设计一套突出应用性、实践性的丛书。一方面，强调课程内容的应用性。以解决实际问题为中心，而不是以学科体系为中心；基础理论知识以应用为目的，以“必需、够用”为度。另一方面，强调课程的实践性。在教学过程中增加实践性环节的比重。

2009 年 5 月，我们组织全国高等职业院校的专家、教授组成了“高职高专工作过程·立体化创新规划教材”编审委员会，全面研讨人才培养方案，并结合当前高职教育的实际情况，历时近两年精心打造了这套“高职高专工作过程·立体化创新规划教材”丛书。我们希望通过这一套全新的、突出职业素质需求的高质量教材的出版和使用，能够促进技能型人才培养的发展。

本套丛书以“工作过程为导向”，强调以培养学生的职业行为能力为宗旨，以现实的职业要求为主线，选择与职业相关的教学内容组织开展教学活动和过程，使学生在学习和实践中掌握职业技能、专业知识及工作方法，从而构建属于自己的经验和知识体系，以解决工作中的实际问题。

本丛书首推书目

- 计算机应用基础
- 办公自动化技术应用教程
- 计算机组装与维修技术
- C++程序设计与应用教程
- C 语言程序设计
- Java 程序设计与应用教程
- Visual Basic 程序设计与应用开发
- Visual C# 2008 程序设计与应用教程
- 网页设计与制作
- 计算机网络安全技术
- 计算机网络规划与设计
- 局域网组建、管理与维护实用教程
- 基于.NET 3.5 的网站项目开发实践
- Windows Server 2008 网络操作系统
- 基于项目教学的 ASP.NET(C#)程序开发设计
- SQL Server 2008 数据库技术实用教程
- 数据库应用技术实训指导教程(SQL Server 版)

- 单片机原理及应用技术
- 基于 ARM 的嵌入式系统接口技术
- 数据结构实用教程
- AutoCAD 2010 实用教程
- C# WEB 数据库编程

丛书特点

(1) 以项目为依托，注重能力训练。以“工作场景导入”→“知识讲解”→“回到工作场景”→“工作实训营”为主线编写，体现了以能力为本位的教育模式。

(2) 内容具有较强的针对性和实用性。丛书以贴近职业岗位要求、注重职业素质培养为基础，以“解决工作场景”中的任务为中心展开内容，书中每一章节都涵盖了完成工作所需的知识和具体操作过程。基础理论知识以应用为目的，以“必需、够用”为度，因而具有很强的针对性与实用性，可提高学生的实际操作能力。

(3) 易于学习、提高能力。通过具体案例引出问题，在掌握知识后立刻回到工作场景解决实际问题，使学生很快上手，提高实际操作能力；每章末的“工作实训营”板块都安排了有代表意义的实训练习使学生进一步提高操作能力。

(4) 示例丰富、由浅入深。书中配备了大量经过精心挑选的例题，既能帮助读者理解知识，又具有启发性。针对较难理解的问题，例子都是从简单到复杂，内容逐步深入。

读者定位

本系列教材主要面向高等职业技术院校和应用型本科院校，同时也非常适合计算机培训班和编程开发人员培训、自学使用。

关于作者

丛书编委会特聘执教多年且有较高学术造诣和实践经验的名师参与各册之编写。他们长期从事有关的教学和开发研究工作，积累了丰富的经验，对相应课程有较深的体会与独特的见解，本丛书凝聚了他们多年的教学经验和心血。

互动交流

本丛书保持了清华大学出版社一贯严谨、科学的图书风格，但由于我国计算机应用技术教育正在蓬勃发展，要编写出满足新形势下教学需求的教材，还需要我们不断的努力实践。因此，我们非常欢迎全国更多的高校老师积极加入到“高职高专工作过程·立体化创新规划教材——计算机系列”编审委员会中来，推荐并参与编写有特色、有创新的教材。同时，我们真诚希望使用本丛书的教师、学生和读者朋友提出宝贵意见和建议，使之更臻成熟。联系信箱：Book21Press@126.com。

丛书编委会

前　言

C++语言是以 C 语言为基础开发的一种集面向对象编程和传统的过程化编程于一体的编程语言，是对 C 语言的继承。在过去，随着新特性的不断添加，C++一度成为一种动态的语言，不过自从 2003 年 ISO/ANSI C++标准第二版发布以后，C++已经稳定下来了。目前常用的编译器支持该标准要求的多数甚至全部特性，学习 C++语言，需要花时间掌握这些新特性的应用。

全书分为 11 章，各章均通过导入工作场景引出问题，然后详细讲解用来解决问题的知识点，最后回到工作场景中解决问题。本书主要内容如下：

第 1 章主要介绍 C++语言的发展、C++程序的基本框架和开发过程。

第 2 章主要介绍 C++变量的定义和使用、运算符和表达式的概念以及运算符的优先级和结合性等。

第 3 章主要介绍 C++语言的基本语句类型以及选择和循环结构程序的设计。

第 4 章主要介绍函数的编写和调用、函数之间传递信息的机制、变量的作用域和生存期、C++多模块程序结构以及预处理指令。

第 5 章主要介绍数组、指针、引用、结构体、共用体的定义和使用，以及编译预处理的相关内容。

第 6 章主要介绍类和对象的定义以及对象的使用方法。

第 7 章主要介绍继承和多态的概念，其中继承又分为单继承和多继承。

第 8 章首先介绍运算符重载的必要性，然后介绍两种运算符重载的方式，以及特殊运算符的重载。

第 9 章主要介绍函数模板的实现、类模板的实现以及 C++标准模板库。

第 10 章主要介绍流的概念、I/O 处理类的层次结构以及基于控制台、文件和字符串的 I/O 处理。

第 11 章主要介绍异常的概念，以及 C++处理异常的方式。

本书具有以下特点。

(1) 结构清晰、模式合理。以“工作场景导入”→“知识讲解”→“回到工作场景”→“工作实训营”为主线编写，以这种新颖的模式合理安排全文。

(2) 针对性强、实用性强。本书以“工作场景”为中心展开内容，各章都涵盖了完成工作所需的知识和具体操作过程，因而具有很强的针对性与实用性，有利于学生提高实际操作能力。

(3) 上手快、易教学。通过具体案例引出问题，在掌握知识后立刻回到工作场景解决问题，使学生容易上手；以教与学的实际需要取材谋篇，方便老师教学。

(4) 安排实训，提高能力。每章都安排了“工作实训营”板块，针对问题给出明确的解决步骤，并对工作实践中的常见问题进行分析，使学生进一步提高应用能力。

(5) 本书包含大量的范例代码，其中绝大部分都是完整的程序，可在编译器中调试和运行。

本书注重实际应用，既可作为高职高专院校计算机及相关专业的教材，也可作为各类培训班的培训教程。此外，本书也适合于有关工程技术人员、技师参考阅读。

本书由梁明、黄昊宇任主编，王继民、严云洋任副主编，冷育荣主审。另外，何光明、王珊珊、杨鹏飞、张居晓、姚昌顺、许勇、杨明、杨萍、赵传申、李海、赵明、张伍荣、范荣钢、钱阳勇、陈芳等人员也参与了编写工作。限于作者水平，书中难免存在不当之处，恳请广大读者批评指正。

编　者

目　录

第 1 章　C++语言概述 1

1.1　工作场景导入 2
1.2　C++语言的发展 2
1.3　C++语言的特点 3
1.4　面向对象程序设计 4
1.5　C++程序的基本框架 5
1.5.1　最简单的 C++程序 5
1.5.2　结构化程序设计框架 7
1.5.3　面向对象程序设计框架 7
1.6　C++程序的开发过程 9
1.6.1　创建项目 9
1.6.2　编辑 11
1.6.3　编译 12
1.6.4　链接 13
1.6.5　运行和调试 13
1.7　回到工作场景 13
1.8　工作实训营 14
1.9　本章小结 14
1.10　习题 15

第 2 章　数据类型、运算符和表达式 17

2.1　工作场景导入 18
2.2　标识符和关键字 18
2.2.1　标识符 18
2.2.2　关键字 19
2.3　C++语言的数据类型 20
2.3.1　整型 21
2.3.2　浮点型 21
2.3.3　字符型 22
2.3.4　布尔型 22
2.3.5　空类型 22
2.4　常量 23
2.4.1　逻辑型常量 23
2.4.2　整型常量 23
2.4.3　实型常量 24
2.4.4　字符型常量 25
2.4.5　字符串常量 26
2.5　变量 27
2.5.1　变量的定义 27
2.5.2　变量的赋值 28
2.5.3　符号常量声明语句 28
2.6　运算符和表达式 29
2.6.1　运算符和表达式的概念 29
2.6.2　运算符与运算类型 29
2.6.3　赋值运算 30
2.6.4　算术运算符和算术表达式 31
2.6.5　关系运算符和关系表达式 33
2.6.6　逻辑运算符和逻辑表达式 34
2.6.7　位运算 36
2.6.8　逗号表达式 39
2.6.9　优先级和结合性 39
2.6.10　数据类型转换 40
2.7　回到工作场景 43
2.8　工作实训营 44
2.9　本章小结 45
2.10　习题 45

第 3 章　基本控制结构 47

3.1　工作场景导入 48
3.2　C++语句 48
3.3　顺序结构 51
3.4　选择结构 52
3.4.1　if 语句 52
3.4.2　switch 语句 57
3.5　循环结构 61
3.5.1　for 语句 61
3.5.2　while 语句 63
3.5.3　do…while 语句 64
3.5.4　循环的嵌套 66
3.5.5　三种循环的比较 67

3.6 跳转语句 67
3.6.1 break 语句 67
3.6.2 continue 语句 68
3.6.3 goto 语句 70
3.7 回到工作场景 71
3.8 工作实训营 73
3.9 本章小结 73
3.10 习题 74
第 4 章 函数 77
4.1 工作场景导入 78
4.2 函数的基本概念 78
4.3 函数定义 79
4.3.1 无参函数的定义 80
4.3.2 有参函数的定义 80
4.4 函数调用 82
4.4.1 函数调用的一般形式 82
4.4.2 函数调用的方式 82
4.4.3 函数调用的注意点 84
4.5 函数声明 84
4.5.1 进行函数声明 85
4.5.2 函数声明的作用域 87
4.6 函数返回类型 88
4.7 函数参数 89
4.7.1 参数的传递方式 89
4.7.2 默认参数 90
4.8 函数重载 92
4.8.1 重载函数的定义 93
4.8.2 重载函数的绑定 95
4.9 内联函数 96
4.10 函数嵌套与递归 97
4.10.1 函数嵌套 97
4.10.2 函数递归 98
4.11 变量作用域与存储类型 100
4.11.1 作用域 100
4.11.2 局部变量和全局变量 103
4.11.3 动态变量和静态变量 107
4.11.4 变量的存储类型 108
4.12 回到工作场景 111
4.13 工作实训营 112
4.14 本章小结 113
4.15 习题 113
第 5 章 自定义数据类型和预处理 117
5.1 工作场景导入 118
5.2 数组 118
5.2.1 一维数组 119
5.2.2 二维数组 122
5.2.3 多维数组 127
5.2.4 字符数组 128
5.3 指针 134
5.3.1 指针的基本概念 134
5.3.2 指针变量的定义 135
5.3.3 指针的操作 136
5.3.4 常量指针和指针常量 139
5.3.5 指针与数组 140
5.3.6 指针与函数 144
5.3.7 指针数组与数组指针 150
5.3.8 多级指针 152
5.3.9 动态存储分配 153
5.4 引用 156
5.5 结构体 158
5.5.1 结构体类型的声明 158
5.5.2 结构体变量 159
5.5.3 结构体变量的引用与初始化 161
5.5.4 结构体数组 163
5.5.5 结构体指针 165
5.5.6 用结构体指针操作链表 168
5.6 共用体 172
5.6.1 共用体类型定义 172
5.6.2 共用体变量的引用 174
5.7 枚举类型 175
5.7.1 枚举类型的定义 175
5.7.2 给枚举变量赋初值 175
5.8 类型定义 typedef 176
5.9 编译预处理 178
5.9.1 文件包含 178

5.9.2 宏定义 179
5.9.3 条件编译 183
5.10 回到工作场景 185
5.11 工作实训营 191
5.12 本章小结 191
5.13 习题 192

第 6 章 类和对象 197

6.1 工作场景导入 198
6.2 类 198
6.2.1 类的定义 198
6.2.2 类的数据成员 199
6.2.3 类的成员函数 199
6.2.4 类成员的访问控制 202
6.3 对象 203
6.3.1 对象的定义 204
6.3.2 对象成员的访问 204
6.3.3 对象的存储 208
6.3.4 对象的赋值 210
6.3.5 成员对象 210
6.4 构造函数和析构函数 211
6.4.1 构造函数 211
6.4.2 析构函数 220
6.5 对象的生存期 223
6.5.1 全局对象、静态对象与局部对象 223
6.5.2 自由存储对象 225
6.6 this 指针 227
6.7 静态成员 229
6.7.1 静态数据成员 229
6.7.2 静态成员函数 231
6.8 const 成员 232
6.8.1 const 对象 232
6.8.2 const 成员函数 233
6.8.3 常量数据成员 235
6.9 友元 235
6.9.1 友元函数 235
6.9.2 友元类 239
6.10 对象数组 240
6.11 回到工作场景 241
6.12 工作实训营 243
6.13 本章小结 244
6.14 习题 244

第 7 章 继承和多态 247

7.1 工作场景导入 248
7.2 继承与派生 248
7.2.1 基本概念 248
7.2.2 单继承 249
7.3 派生类对基类成员的访问 253
7.3.1 公有继承 253
7.3.2 私有继承 255
7.4 多继承与虚基类 256
7.4.1 多重继承的定义 256
7.4.2 多继承中的二义性问题 257
7.4.3 虚基类的定义 260
7.4.4 虚基类的构造函数 261
7.5 派生类的构造函数和析构函数 263
7.5.1 派生类的构造函数 263
7.5.2 派生类的析构函数 266
7.6 子类型关系 269
7.7 虚函数与多态性 271
7.7.1 多态性的概念 271
7.7.2 虚函数 271
7.7.3 虚析构函数 274
7.7.4 纯虚函数与抽象类 275
7.8 回到工作场景 279
7.9 工作实训营 283
7.10 本章小结 284
7.11 习题 284

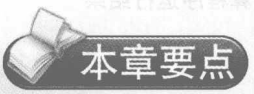
第 8 章 运算符重载 289

8.1 工作场景导入 290
8.2 运算符函数与运算符重载 290
8.3 典型运算符的重载 292
8.3.1 实现一个分数类 292
8.3.2 重载单目运算符 293
8.3.3 重载双目运算符 297

8.3.4 其他常用运算符的重载 300
8.4 重载运算符的基本原则 312
8.5 回到工作场景 316
8.6 工作实训营 319
8.7 本章小结 320
8.8 习题 320
第 9 章 模板 323
9.1 工作场景导入 324
9.2 模板的概念 324
9.3 函数模板 326
9.3.1 函数模板的定义与调用 326
9.3.2 函数模板的使用 328
9.4 类模板 330
9.4.1 类模板的定义与使用 330
9.4.2 类模板的派生与继承 335
9.5 C++标准模板库 336
9.6 回到工作场景 338
9.7 工作实训营 342
9.8 本章小结 342
9.9 习题 342
第 10 章 C++流 345
10.1 工作场景导入 346
10.2 C++流的概念 346
10.2.1 C++流的逻辑结构 347
10.2.2 基本 I/O 流类体系 349
10.2.3 预定义流对象 355
10.2.4 提取运算符(>>)和插入运算符(<<) 356
10.2.5 其他输入/输出方式 357
10.2.6 用户自定义的 I/O 361
10.3 输入/输出的格式控制 361
10.3.1 默认的输入/输出格式 361
10.3.2 格式标志与格式控制 361
10.3.3 输入/输出宽度的控制 362
10.3.4 浮点数输出方式的控制 362
10.3.5 输出精度的控制 363
10.3.6 对齐方式的控制 364
10.3.7 填充字符的控制 365
10.3.8 插入换行符 365
10.3.9 小数点处理方式的控制 366
10.3.10 其他格式控制方法 367
10.4 文件流 368
10.4.1 文件的概念 368
10.4.2 文件输出 370
10.4.3 文件输入 375
10.4.4 文件随机存取 377
10.5 基于 I/O 类库的字符串 I/O 381
10.6 回到工作场景 383
10.7 工作实训营 388
10.8 本章小结 388
10.9 习题 389
第 11 章 异常处理 391
11.1 工作场景导入 392
11.2 异常的概念 392
11.3 C++的异常处理机制 395
11.3.1 抛出异常 throw 395
11.3.2 捕获和处理异常 try…catch 396
11.3.3 自定义异常类 398
11.3.4 异常处理的嵌套 400
11.3.5 异常规范 402
11.3.6 函数堆栈的回退 403
11.4 回到工作场景 407
11.5 工作实训营 410
11.6 本章小结 410
11.7 习题 410

第1章

C++语言概述



- 程序设计方法和程序设计语言的发展。
- C++语言的发展。
- 简单C++程序的构成及开发。

- 理解C++的基本框架。
- 熟悉C++程序的开发过程，能够自行开发一个完整的C++程序。

1.1 工作场景导入

【工作场景】

一个薪资制公司按照下面的方式计算其员工的收入。员工每周的工资取决于其每周工作时间(小时)和时薪。假定一个标准工作周的工作时间是 30 小时。工作时间小于或者等于 30 小时的员工工资就是其工作时间乘以时薪。超过 30 小时的时间被看做是“加班时间”，“加班时间”的工资是一般工作时间工资的 1.5 倍乘以加班时间(小时)，也就是说加班员工每周的工资是：30×时薪+加班时间×时薪×1.5。用这些信息创建一个计算该公司员工每周总收入的程序。程序运行结果如图 1-1 所示。

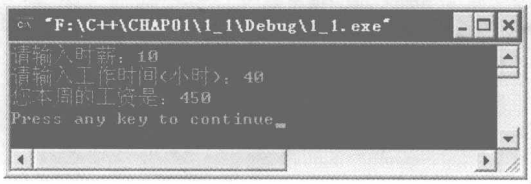


图 1-1 工资计算程序运行结果

【引导问题】

(1) 什么是 C++语言？
(2) 什么是面向对象程序设计？
(3) C++程序的基本框架都包括哪些部分？
(4) 如何使用 Visual C++控制台开发 C++应用程序？

1.2 C++语言的发展

C++是当今非常流行的一种支持结构化程序设计、面向对象程序设计及泛型程序设计的高级程序设计语言。它适合作为系统描述语言，既可用来写系统软件，也可用来写应用软件。它是 20 世纪 80 年代初由贝尔实验室在 C 语言的基础上，借鉴其他面向对象程序设计语言的特性而开发的。

C 语言最早的原型是 Algol 60。1963 年，剑桥大学将其发展成为 CPL(Combined Programming Language)。1967 年，剑桥大学的 Matin Richards 对 CPL 语言进行了简化，产生了 BCPL 语言。1970 年，美国贝尔实验室(Bell Labs)的 Ken Thompson 对 BCPL 进行了修改，并取名为 B 语言，意思是提取 CPL 的精华，并用 B 语言编写了第一个 UNIX 系统。1973 年，AT&T 贝尔实验室的 Dennis Ritchie(D.M.Ritchie)在 BCPL 和 B 语言的基础上设计出了一种新的语言，取 BCPL 中的第二个字母为名，这就是大名鼎鼎的 C 语言。随后不久，UNIX 的内核(Kernel)和应用程序全部用 C 语言改写，从此，C 语言成为 UNIX 环境下使用最广

泛的主流编程语言。

C 语言是一种常规用途的语言，可用来编写任何形式的程序，而 UNIX 操作系统直接促成了它的成功与普及。如果想维护 UNIX 操作系统，就需要使用 C 语言。C 语言和 UNIX 操作系统的配合是如此的天衣无缝，以至于不久以后，不仅系统程序，就连 UNIX 下运行的几乎所有商业程序都开始用 C 语言来编写。随着 C 语言越来越流行，逐渐出现了为其他流行操作系统编写的 C 语言版本，C 语言的应用开始不受 UNIX 机器的局限。不过，虽然 C 语言非常流行，但它并不是完美无缺的。

C 语言的特殊性在于，它虽然是一种高级语言，但也包含了低级语言的许多特点。C 语言其实处在一种非常高级的语言和一种低级语言之间，其优点和缺点都很突出。类似于(低级)汇编语言，C 语言程序可直接操纵计算机的内存；另一方面，C 语言又具有高级语言的许多特点，所以比汇编语言更容易理解和编写。这使得 C 语言成为编写系统程序的理想选择，但对于其他程序(有时甚至包括一些系统程序)，C 语言并不像其他语言那样容易理解。另外，它也不像另外一些高级语言那样提供了对自动检查的完美支持。

为了解决上述问题以及 C 语言的另外一些缺陷，AT&T 贝尔实验室的 Bjarne Stroustrup 在 20 世纪 80 年代初开发了 C++语言。C++并不是对 C 语言的功能做简单的改进和扩充，而是一种本质性革新。C 语言的大多数特性都成为 C++的一个子集，所以大多数 C 程序其实也是 C++程序(反之则不成立，许多 C++程序都绝对不是 C 程序)，这对于继承和开发当前已广泛使用的软件是非常重要的，可节省大量的人力和物力。和 C 语言不同，C++具备了“面向对象程序设计”(Object-Oriented Programming，OOP)的能力。OPP 是一种功能非常强大的编程技术，可以使得程序的各个模块的独立性更强，程序的可读性和可理解性更好，程序代码的结构性更加合理。这对于设计和调试一些规模大的软件是非常重要的。再者，用 C++设计的程序具有扩充性强的特点，这对于编写一些大的程序而言是非常重要的。

到目前为止，运用得较为广泛的 C++集成开发环境有 VC++(Visual C Plus Plus)、BC++(Borland C Plus Plus)、AT&T C++等。它们都是在标准的 C++语言的基础上，结合自己的开发平台，对 C++的标准进行约束和扩充，以更好地适应自身平台下的程序设计和软件开发。它们并不是新的语言种类。

C++语言继承了 C 语言所有的特征，这使得以前大量的 C 语言代码可以在 C++环境中重新编译利用；同时，C++语言也从其他面向对象的语言(如 Ada、Simula 67)借鉴了某些面向对象的特征，从而提供了对面向对象程序设计的支持。

1.3　C++语言的特点

C++一直是程序设计语言的主流之一，因为 C++既具有面向对象语言所共有的功能，又很好地集成了结构化语言 C 的优点，因此独具一格，极受程序员的青睐。

C++几乎在所有的计算机环境中都非常普及，包括个人电脑、UNIX 工作站和大型计算机。如果将 C++与之前的编程语言相比，就可以看出 C++的这种普及率是非常高的。除此以外，大多数专业程序员总是愿意使用他们已熟知的、使用起来得心应手的语言，而不是

使用新的、不熟悉的语言，花大量的时间来研究其特性。当然，C++是建立在 C 的基础之上(在 C++出现之前，许多环境都使用 C 语言)，这对于 C++的普及有很大的帮助。但是 C++的流行远不止这一个原因，更重要的原因是 C++有许多优点。

- C++适用的应用程序范围极广。C++几乎可用于所有的应用程序，从字处理应用程序到科学应用程序，从操作系统组件到计算机游戏等。
- C++可用于硬件级别的编程，例如实现设备驱动程序。
- C++从 C 语言中继承了过程化编程的高效性，并集成了面向对象程序设计的功能。
- C++在其标准库中提供了大量的功能。
- 有许多商业 C++库支持数量众多的操作系统环境和专门的应用程序。

因为几乎所有的计算机都可以使用 C++编程，所以 C++语言几乎普及到所有的计算机平台上。因此，把用 C++编写的程序从一台机器迁移到另一台机器上不需要花费什么力气。

1.4 面向对象程序设计

面向对象程序设计(Object-Oriented Programming，简记为 OOP)立意于创建软件重用代码，具备更好的模拟现实世界环境的能力，这使得它被公认为是自上而下编程的优胜者。它通过给程序中加入扩展语句，把函数“封装”进编程所必需的“对象”中。面向对象的编程语言使得复杂的工作条理清晰、编写容易。说它是一场革命，不是对对象本身而言，而是对它处理工作的能力而言。

面向对象程序设计并不与传统的程序设计和编程方法相兼容，如果程序中只是部分面向对象反而会使情况变得更糟。除非整个开发环境都是面向对象的，否则对象产生的好处还没有带来的麻烦多。有人可能会说 C++不是一个纯面向对象的语言， C++是一个混合型语言，它既可以使用面向对象的思想进行程序设计，也可以进行传统的过程化编程。然而，对于大型项目的开发，往往需要在 C++中使用纯面向对象程序设计去声明类，而且在项目开发中只用对象和类。面向对象程序设计代码很容易维护、理解和重复使用，这些都是软件工程的基础。

对象(Object)是问题域或实现域中某些事物的一个抽象，用于反映此事物在系统中需要保存的信息和发挥的作用；它是一组属性和有权对这些属性进行操作的一组服务的封装体。简单地说，一个人可以是一个对象，一台 DVD 播放器也可以说是一个对象。一个人作为一个对象时，要有两个要素：一是个人的静态特征，如：身高、体重、学历等，这些特征就称为“属性”；二是个人的动态特征，如正在骑车、跑步等，这些特征就称为“行为”。任何一个对象都必须包括属性和行为这两个要素。DVD 作为一个对象时，其属性包括生产厂家、价格等，其行为包括播放、快进、停止等。

在面向对象语言中可以定义类，类(Class)是指变量与一些使用这些变量的函数(传递不同数量的参数)的集合。在现实中，类是对一组具有共同特征的客观对象的抽象，它将改组对象所具有共同特征封装起来，以说明该组对象的能力和性质。

1.5　C++程序的基本框架

C++同时支持结构化程序设计、面向对象程序设计和泛型程序设计。对于简单的应用可以采用结构化程序设计方法，要管理和实现大型项目就要用面向对象的程序设计方法。泛型程序设计一般用来实现通用的任务。

1.5.1　最简单的 C++程序

下面先介绍几个简单的 C++程序，然后从中分析 C++程序的特点和构成。

【例 1.1】利用 C++输出一个“Hello World!”字符串。

程序代码如下：

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world!\n";/*输出 Hello world! */
    return 0;
}
```

程序运行结果如图 1-2 所示。

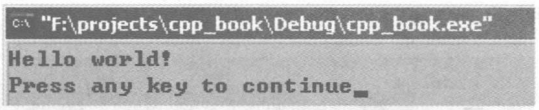


图 1-2　例 1.1 程序的运行结果

分析：

① /*和*/之间的所有行都被认为是注释，它对程序的行为没有任何影响，编程人员可以用它们在代码中包含简短的解释或说明。本例中的注释是关于程序输出的一个简要描述。

② #include <iostream>是以井号(#)开头的预编译指令行，#include <文件名>通常用于将其他文件内容包含到当前程序中。本例中，指令#include <iostream>告诉预处理程序把 iostream 标准文件包含进来。这个特殊的文件(iostream)中包含有 C++标准输入/输出的功能，本例中使用它的 cout 来实现输出功能。

③ using namespace std;告诉编译器包含 std 名称空间中的功能。C++标准库中的所有功能都被包含在 std 名称空间中。因此为了使用这些功能(如#include<iostream>引入的输出功能)，在使用标准库的 C++程序里这一行是经常出现的。

④ main 是主函数的名字，每个 C++程序都必须包含一个 main 函数，它是所有 C++程序的起始运行点。一个函数由两个部分组成：函数头(如 int main())和函数体(由一对大括号括起来)组成。

⑤ cout << “Hello world!\n”;的功能是向标准输出设备(通常为显示器)输出一行字符。cout 表示 C++中的标准输出流，双引号内的字符串被原样输出，\n 是换行符，即在输

出“Hello world!”后让光标换到下行的行首。cout 是在 std 名称空间中定义的，定义的代码在 iostream 标准文件中，这就是为什么要在代码中包含 iostream 文件并且声明将要使用这个特殊名称空间的原因。

⑥ return 0;表示 main 函数将要结束，同时返回一个值给调用本程序的操作系统。main 函数的返回代码 0 通常被理解为程序在运行期间没有任何错误并按照预先设计的那样工作，这是结束一个 C++程序时最常用的方法。

【例 1.2】 从键盘输入两个整数，利用独立的函数求这两个数的和，并输出。

程序代码如下：

```
#include <iostream>
using namespace std;

int add(int a, int b)
{
    return (a+b);
}

int main()
{
    int x; int y; int sum;  //定义 3 个整型变量 x、y、sum
    cout << "请输入两个整数，用空格分隔，按回车键(Enter):";
    cin >> x >> y;  //从键盘获取两个整数
    sum = add(x, y);  //利用函数 add 求 x、y 的和
    cout << "add="<<sum<<endl;
    return 0;
}
```

输出结果如图 1-3 所示。

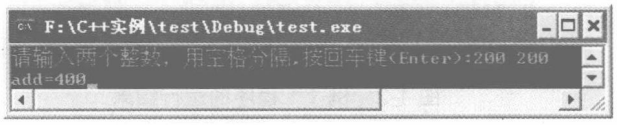


图 1-3 例 1.2 的输出结果

分析：

① int x;表示定义一个变量 x，即系统分配一块内存，并把该块内存取名为 x。

② 双反斜杠(//)用来表示注释，从“//”开始到本行结束的所有字符都是注释。

③ cin 表示 C++的输入流，cin>>x>>y;表示从标准输入设备(通常为键盘)中接收两个整数到变量 x、y 中。

④ cout << “add=” <<sum<<endl;表示连续输出字符串和变量 sum 的值及换行符到标准输出设备。

⑤ 本例中定义了一个函数 add，在 C++中，一个函数可以看做是完成某个独立功能的代码的集合。add 函数由函数头 int add(int a, int b)和其后由大括号括起来的函数体组成，函数头指定了函数的返回值类型(本例为整数 int)、函数名(本例为 add)、函数需要的参数以及类型(本例为两个整数 a、b)。

⑥ main 中的 sum=add(x, y);引用了 add 函数的功能来实现 x、y 两个数求和。其执行的顺序是：先将 x、y 的值交给 a、b，然后执行 add 中的代码完成两个整数相加，返回的和存放到 sum 对应的内存空间中。