计算机应用能力培养丛书

Visual C++ 2008程序设计简明教程

严涛 编著

清华大学出版社

计算机应用能力培养丛书

Visual C++ 2008 程序设计简明教程

严 涛 编著

清华大学出版社

北 京

内 容 简 介

本书较为全面地介绍了使用 Visual C++进行程序设计的基础知识和编程技术，全书贯穿了面向对象编程的思想和良好的编程习惯，力争将每个关键知识点讲解得清晰、明了。

全书共 15 章。第 1~8 章介绍了 C++编程基础知识，这是学习 Visual C++编程的前提，重点阐述了类、对象、继承、虚函数等面向对象核心知识点，另外对编程的发展历程、变量、数据类型、数组、字符串、指针、函数等知识进行了讲解。第 9~13 章介绍了 Visual C++编程技术，涵盖了 MFC 编程的基本概念和机制，菜单和工具栏编程、图形编程、对话框和标准控件编程等。本书第 14 章是一些典型的 C++和 Visual C++编程示例，它们分别考察了书中的重要知识点，这些也是企业招聘的常见考点。第 15 章介绍 MFC 如何实现存储和打印。附录 A 则介绍了调试程序的基本方法和技巧。

本书内容丰富，结构清晰，核心概念和关键技术讲解清楚，同时提供了丰富的示例以展示具体应用，是初学 Visual C++编程人员的最佳入门指导，可作为高等学校、高职学校，以及社会各类培训班“Visual C++程序设计”课程的教材。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目(CIP)数据

Visual C++ 2008 程序设计简明教程/严涛　编著. —北京：清华大学出版社，2009.9
(计算机应用能力培养丛书)
ISBN 978-7-302-20883-9

Ⅰ. V…　Ⅱ. 严…　Ⅲ. C 语言－程序设计－教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2009)第 154168 号

责任编辑：王　军　李维杰
装帧设计：孔祥丰
责任校对：胡雁翎
责任印制：
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：
装 订 者：
经　　销：全国新华书店
开　　本：185×260　　**印　张**：21.5　　**字　数**：523 千字
版　　次：2009 年 9 月第 1 版　　**印　次**：2009 年 9 月第 1 次印刷
印　　数：1~4000
定　　价：29.90 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：010-62770177 转 3103　　产品编号：

此为试读，需要完整PDF请访问：www.ertongbook.com

前　言

高职高专教育以就业为导向，以技术应用型人才为培养目标，担负着为国家经济高速发展输送一线高素质技术应用型人才的重任。近年来，随着我国高等职业教育的发展，高职院校数量和在校生人数均有了大幅激增，已经成为我国高等教育的重要组成部分。

根据目前我国高级应用型人才的紧缺情况，教育部联合六部委推出“国家技能型紧缺人才培养培训项目”，并从 2004 年秋季起，在全国两百多所学校的计算机应用与软件技术、数控项目、汽车维修与护理等专业推行两年制和三年制改革。

为了配合高职高专院校的学制改革和教材建设，清华大学出版社在主管部门的指导下，组织了一批工作在高等职业教育第一线的资深教师和相关行业的优秀工程师，编写了适应新教学要求的计算机系列高职高专教材——《计算机应用能力培养丛书》。该丛书主要面向高等职业教育，遵循“以就业为导向”的原则，根据企业的实际需求来进行课程体系设置和教材内容选取。根据教材所对应的专业，以“实用”为基础，以“必需”为尺度，为教材选取理论知识；注重和提高案例教学的比重，突出培养人才的应用能力和实际问题解决能力，满足高等职业教育“学校评估”和“社会评估”的双重教学特征。

每本教材的内容均由“授课”和“实训”两个互为联系和支持的部分组成。“授课”部分介绍在相应课程中，学生必须掌握或了解的基础知识，每章都设有“学习目标”、“实用问题解答”、“小结”、“习题”等特色段落；“实训”部分设置了一组源于实际应用的上机实例，用于强化学生的计算机操作使用能力和解决实际问题的能力。每本教材配套的习题答案、电子教案和一些教学课件，均可在该丛书的信息支持网站（http://www.tupwk.com.cn/GZGZ）上下载或通过 Email（wkservice@tup.tsinghua.edu.cn）索取。读者在使用过程中遇到了疑惑或困难，可以在支持网站的互动论坛上留言，本丛书的作者或技术编辑会提供相应的技术支持。

本书依据教育部《高职高专教育计算机公共基础课程教学基本要求》编写而成，较为全面地介绍了 Visual C++编程的基本概念和编程技术，全书贯穿了面向对象编程的思想和良好的编程习惯，力争将每个关键知识点讲解得清晰、明了。

全书共 15 章，第 1~8 章介绍了 C++编程基础知识，包括编程的发展历程、变量、数据类型、指针、数组、字符串、指针、函数等，重点阐述了类、对象、继承、虚函数等面向对象核心知识点(从一个简单的 CBox 类进行逐步扩展，将这些知识点进行融合，能使学生对这些知识点的概念、用法有更深的理解，并进行有效地贯穿、联系)，这是学习 Visual C++编程的前提。

第 9~13 章介绍了 Visual C++编程技术，涵盖了 MFC 编程的基本概念和机制、菜单和工具栏编程、图形编程、对话框和标准控件编程等。这部分以一个简单绘图程序 Sketcher 为例，从建立基本框架到各种基本功能的实现，贯穿了以上所有知识点，在使读者理解各种编程技术的同时，学会如何在具体的项目中进行应用(读者可在该项目基础上进行扩展，

实现更完善的绘图功能)。

本书第 14 章是一些典型的 C++和 Visual C++编程示例，它们分别考察了书中的重要知识点和难点，这些也是企业招聘的常见考点。第 15 章介绍 MFC 如何实现存储和打印，附录 A 介绍了调试程序的基本方法和技巧。

由于计算机科学技术发展迅速，以及自身水平和编写时间所限，书中如有错误或不足之处，欢迎广大读者对我们提出意见或建议。

编　　者

目　录

第 1 章　Visual C++ 2008 编程概述 ····· 1
1.1　编程的基本概念 ····· 1
1.1.1　编程的发展历程 ····· 1
1.1.2　为何会存在如此多的编程语言 ····· 3
1.1.3　开发环境 ····· 4
1.2　了解 C++语言 ····· 4
1.2.1　C++的诞生 ····· 4
1.2.2　面向对象编程 ····· 5
1.2.3　C++与 Java 和 C#的关系 ····· 6
1.3　使用 Visual C++ 2008 编程 ····· 7
1.3.1　了解 Windows 编程 ····· 7
1.3.2　熟悉开发环境 ····· 8
1.3.3　编写控制台应用程序 ····· 9
1.3.4　编写 MFC 应用程序 ····· 15
本章小结 ····· 16
习题 ····· 17

第 2 章　变量、数据和计算 ····· 18
2.1　C++的程序结构 ····· 18
2.1.1　程序注释 ····· 18
2.1.2　#include 指令——头文件 ····· 19
2.1.3　命令空间和 using 声明 ····· 19
2.1.4　main()函数 ····· 20
2.1.5　程序语句 ····· 21
2.2　C++的基本数据类型 ····· 22
2.2.1　整型数据 ····· 23
2.2.2　字符型数据 ····· 24
2.2.3　浮点型数据 ····· 24
2.2.4　布尔型数据 ····· 25
2.3　字面值 ····· 25
2.4　变量 ····· 26
2.4.1　变量的命名规则 ····· 27
2.4.2　变量的声明和初始化 ····· 27
2.5　C++中的计算 ····· 28
2.5.1　算术运算 ····· 29
2.5.2　关系运算和逻辑运算 ····· 29
2.5.3　赋值运算 ····· 31
2.5.4　运算的优先级 ····· 32
2.5.5　变量间的类型转换 ····· 33
2.5.6　变量的生存时间及作用域 ····· 35
本章小结 ····· 38
习题 ····· 38

第 3 章　流程控制 ····· 40
3.1　分支 ····· 40
3.1.1　在什么情况下使用分支结构 ····· 40
3.1.2　if 语句 ····· 41
3.1.3　嵌套的 if 语句 ····· 43
3.1.4　switch 语句 ····· 44
3.1.5　条件运算符 ····· 46
3.2　循环 ····· 47
3.2.1　do-while 循环 ····· 47
3.2.2　while 循环 ····· 48
3.2.3　for 循环 ····· 49
3.2.4　无限循环 ····· 50
3.2.5　循环的中断 ····· 51
本章小结 ····· 52
习题 ····· 52

第 4 章　数组、字符串和指针 …… 54
4.1　数组 …… 54
4.1.1　数组的声明和初始化 …… 54
4.1.2　二维数组 …… 56
4.2　字符串和字符串数组 …… 58
4.2.1　字符串及其处理 …… 58
4.2.2　字符串数组 …… 61
4.3　指针 …… 62
4.3.1　指针的基本概念 …… 62
4.3.2　指针的声明和初始化 …… 62
4.3.3　指针的运算 …… 63
4.3.4　指针和数组 …… 65
4.4　关于动态内存分配 …… 68
4.4.1　了解自由存储器 …… 68
4.4.2　为数组动态分配内存 …… 69
本章小结 …… 71
习题 …… 71

第 5 章　函数及其详解 …… 73
5.1　函数简介 …… 73
5.1.1　函数的工作过程 …… 73
5.1.2　函数的结构 …… 74
5.1.3　使用函数 …… 75
5.2　给函数传递实参 …… 77
5.2.1　按值传递 …… 78
5.2.2　按引用传递 …… 80
5.2.3　main()函数的实参 …… 82
5.2.4　接收数量不定的函数实参 …… 83
5.3　从函数返回值 …… 84
5.3.1　返回指针 …… 84
5.3.2　返回引用 …… 86
5.3.3　函数中的静态变量 …… 87
5.4　递归函数的调用 …… 88
5.5　函数重载 …… 89
5.5.1　函数重载的概念 …… 90
5.5.2　函数重载和多义性 …… 92
5.6　函数指针 …… 92
5.6.1　声明函数指针 …… 93
5.6.2　函数指针作为参数 …… 94
5.6.3　函数指针数组 …… 95
5.7　函数模板 …… 96
5.8　初始化函数形参 …… 97
5.9　异常 …… 99
5.10　处理内存分配错误 …… 101
本章小结 …… 102
习题 …… 103

第 6 章　类和结构 …… 104
6.1　C++ 中的结构 …… 104
6.1.1　定义并初始化结构 …… 104
6.1.2　访问结构的成员 …… 105
6.1.3　使用指针处理结构 …… 106
6.2　类 …… 108
6.2.1　定义类 …… 109
6.2.2　声明类的对象 …… 110
6.2.3　访问类的数据成员 …… 110
6.2.4　类的成员函数 …… 111
6.2.5　内联函数 …… 112
6.3　类的构造函数 …… 113
6.3.1　构造函数的概念 …… 113
6.3.2　在类定义中指定默认的形参值 …… 115
6.3.3　在构造函数中使用初始化列表 …… 116
6.4　类的私有成员 …… 116
6.4.1　访问私有类成员 …… 118
6.4.2　类的友元函数 …… 119
6.4.3　默认复制构造函数 …… 120
6.5　this 指针 …… 121
6.6　类的 const 对象 …… 123
6.6.1　类的 const 成员函数 …… 123
6.6.2　类外部的成员函数定义 …… 123
6.7　类对象的数组 …… 124
6.8　类对象的指针和引用 …… 125
6.8.1　类对象的指针 …… 126
6.8.2　类对象的引用 …… 127

此为试读，需要完整PDF请访问：www.ertongbook.com

6.9 类的静态成员 …… 129
本章小结 …… 130
习题 …… 131

第 7 章　深入理解类 …… 132
7.1 类的析构函数 …… 132
7.1.1 默认的析构函数 …… 132
7.1.2 析构函数与动态内存分配 …… 134
7.2 实现复制构造函数 …… 136
7.3 运算符重载 …… 138
7.3.1 实现重载的运算符 …… 138
7.3.2 重载赋值运算符 …… 141
7.3.3 重载加法运算符 …… 144
7.3.4 重载递增和递减运算符 …… 146
7.4 类模板 …… 147
7.4.1 定义类模板 …… 147
7.4.2 根据类模板创建对象 …… 149
7.4.3 使用多个形参的类模板 …… 151
7.5 字符串进阶 …… 152
7.5.1 创建字符串对象 …… 152
7.5.2 连接字符串 …… 153
7.5.3 访问与修改字符串 …… 154
7.5.4 比较字符串 …… 157
7.5.5 搜索字符串 …… 157
本章小结 …… 161
习题 …… 162

第 8 章　继承和虚函数 …… 163
8.1 类的继承 …… 163
8.2 继承下的访问控制 …… 167
8.2.1 派生类中构造函数的操作 …… 167
8.2.2 声明类的保护成员 …… 169
8.2.3 继承类成员的访问级别 …… 170
8.3 派生类中的复制构造函数 …… 171
8.4 友元类成员 …… 174
8.5 虚函数 …… 175
8.5.1 使用指向类对象的指针 …… 177
8.5.2 使用引用处理虚函数 …… 177
8.5.3 纯虚函数 …… 177
8.5.4 抽象类 …… 178
8.5.5 间接基类 …… 180
8.5.6 虚析构函数 …… 181
8.6 类类型之间的强制转换 …… 183
8.7 关于嵌套类 …… 184
8.8 关于标准模板库 …… 186
8.8.1 为什么需要学习 STL …… 186
8.8.2 STL 的组成 …… 186
本章小结 …… 187
习题 …… 188

第 9 章　Windows 编程概述 …… 190
9.1 Windows 编程的基本概念 …… 190
9.1.1 理解窗口元素 …… 190
9.1.2 Windows 程序与操作系统 …… 192
9.1.3 事件驱动型程序 …… 192
9.1.4 Windows 消息 …… 192
9.1.5 Windows API …… 192
9.1.6 Windows 数据类型 …… 193
9.1.7 Windows 程序中的变量名前缀 …… 194
9.2 Windows 程序的结构 …… 194
9.2.1 WinMain()函数 …… 195
9.2.2 WindowProc()函数 …… 204
9.3 Windows 程序的组织 …… 207
本章小结 …… 210
习题 …… 210

第 10 章　使用 MFC 编写 Windows 程序 …… 211
10.1 MFC 概述 …… 211
10.1.1 MFC 标记法 …… 211
10.1.2 MFC 程序的组织方式 …… 212
10.2 MFC 的文档/视图概念 …… 215
10.2.1 文档的概念 …… 215
10.2.2 视图的概念 …… 215
10.2.3 连接文档和视图 …… 216

10.2.4 基于 MFC 的 Windows 应用程序 …… 217
10.3 创建 MFC 应用程序 …… 218
10.3.1 创建 SDI 应用程序 …… 219
10.3.2 MFC 应用程序向导的输出结果 …… 222
10.3.3 编译并运行程序 …… 227
10.3.4 程序的工作原理 …… 228
10.3.5 创建 MDI 应用程序 …… 229
本章小结 …… 231
习题 …… 231

第 11 章 处理菜单和工具栏 …… 232
11.1 与 Windows 通信 …… 232
11.1.1 消息映射 …… 232
11.1.2 消息类别 …… 235
11.1.3 处理程序中的消息 …… 235
11.2 扩展 Sketcher 程序 …… 236
11.2.1 创建和编辑菜单 …… 236
11.2.2 为菜单消息添加处理程序 …… 239
11.2.3 添加工具栏按钮 …… 247
本章小结 …… 249
习题 …… 250

第 12 章 在窗口中绘图 …… 251
12.1 窗口绘图的基础知识 …… 251
12.1.1 窗口客户区 …… 251
12.1.2 Windows 图形设备接口 …… 252
12.2 Visual C++的绘图机制 …… 254
12.2.1 应用程序中的视图类 …… 254
12.2.2 CDC 类详解 …… 255
12.3 对鼠标进行编程 …… 260
12.3.1 鼠标发出的消息 …… 262
12.3.2 鼠标消息处理程序 …… 262
12.3.3 使用鼠标绘图 …… 263
12.3.4 运行 Sketcher 程序 …… 279
本章小结 …… 281
习题 …… 281

第 13 章 对话框和标准控件编程 …… 282
13.1 理解对话框和控件 …… 282
13.2 对话框编程 …… 283
13.2.1 创建对话框资源 …… 283
13.2.2 添加对话框类 …… 284
13.2.3 显示和关闭对话框 …… 286
13.2.4 处理对话框中的控件 …… 287
13.2.5 使对话框有效 …… 289
13.3 使用微调按钮控件 …… 291
13.3.1 添加“缩放”菜单项和相应工具栏按钮 …… 292
13.3.2 创建微调按钮 …… 292
13.3.3 生成对话框类 …… 293
13.3.4 显示微调按钮 …… 295
13.4 使用比例系数 …… 296
13.4.1 可缩放的映射模式 …… 296
13.4.2 设置文档大小 …… 297
13.4.3 设置映射模式 …… 297
13.4.4 同时实现滚动与缩放 …… 299
13.5 使用状态栏 …… 300
13.6 使用列表框 …… 303
13.7 使用编辑框控件 …… 305
本章小结 …… 306
习题 …… 306

第 14 章 实训 …… 307
14.1 C++程序设计基础 …… 307
14.1.1 赋值语句 …… 307
14.1.2 类型转换 …… 308
14.1.3 预处理、const 和 sizeof …… 309
14.1.4 动态内存传递 …… 311
14.1.5 递归 …… 313
14.1.6 内螺旋递增序列 …… 314
14.1.7 链表的操作 …… 316
14.2 面向对象编程 …… 319
14.2.1 类和结构 …… 319

14.2.2 成员变量 …… 320
14.2.3 复制构造函数 …… 322
14.2.4 虚函数继承 …… 323
14.3 Visual C++编程 …… 324
14.3.1 带图标的程序菜单 …… 325
14.3.2 带动画效果的状态栏 …… 329
14.3.3 不规则按钮 …… 330
14.3.4 绘制正弦曲线 …… 332

第 15 章 存储和打印文档 …… 334
(http://www.tupwk.com.cn/GZGZ)
15.1 串行化及其应用 …… 334
15.1.1 文档类定义中的串行化 …… 335
15.1.2 文档类实现中的串行化 …… 335
15.1.3 基于 CObject 的类的功能 …… 337
15.1.4 串行化的工作方式 …… 338
15.1.5 应用串行化 …… 339
15.2 打印文档 …… 344
15.2.1 打印的过程 …… 344
15.2.2 获取文档的尺寸 …… 347
15.2.3 存储打印数据 …… 348
15.2.4 准备打印 …… 348
15.2.5 打印后的清除 …… 350
15.2.6 准备设备环境 …… 350
15.2.7 打印文档 …… 351
本章小结 …… 354
习题 …… 354

附录 A 程序调试 …… 355
(http://www.tupwk.com.cn/GZGZ)
A.1 理解调试 …… 355
A.2 基本的调试技术 …… 357
A.2.1 设置断点 …… 358
A.2.2 进入中断模式 …… 359
A.2.3 跟踪变量的值 …… 360
A.2.4 单步执行程序 …… 360
A.2.5 调用堆栈 …… 361
A.2.6 设置跟踪点 …… 361
A.3 添加调试代码 …… 362
A.3.1 调用堆栈 …… 366
A.3.2 单步执行到出错位置 …… 368

附录 B C++关键字 …… 371
(http://www.tupwk.com.cn/GZGZ)

第 1 章

Visual C++ 2008 编程概述

本章介绍编程的基础知识与使用 Visual C++ 2008 进行编程的基本流程。通过本章的学习，应该完成以下**学习目标**：

- ☑ 了解编程的发展历程
- ☑ 理解当前出现多种编程语言的原因
- ☑ 理解什么是开发环境
- ☑ 了解 C++的演变过程
- ☑ 掌握面向对象编程的特点
- ☑ 了解 Windows 编程
- ☑ 熟悉 Visual C++ 2008 编程环境
- ☑ 掌握编写控制台应用程序的流程
- ☑ 对 MFC 编程有一个简单的了解

1.1 编程的基本概念

无论是初学编程的人，还是已经有相当编程经验的开发人员，都应当清晰地了解编程的一些基本概念。这其中包括编程的发展过程、什么是编程、当前为什么会存在那么多的编程语言、什么是开发环境、编译器起什么作用等。

1.1.1 编程的发展历程

编程就是利用某种语言与计算机对话，计算机能够理解这种编程语言的文法和语法，然后按照用户的指令来帮助用户完成工作。编程的实质就是对现实中复杂得难以人工解决的问题进行抽象，转化成计算机可以理解的形式，让计算机进行处理和解决。

许多人认为编程是从 20 世纪末才开始的，其实现代编程和编程语言可以追溯到 20 世纪 40 年代中期。在讲述 20 世纪 40 年代的编程历史之前，仍需再向前追溯到 1822 年。当时英国剑桥大学的学生 Charles Babbage 偶然发现许多有关时间的计算设备(如天文图、潮汐图、航海图等)在测量时都存在临界误差，并且测量频繁。这些误差造成许多船只、人员和货物在海上失踪。

Babbage 认为这些不精确是人为误差造成的，因此他想到利用蒸汽发动机来建立和维护那些图表，由此产生了差分机(Difference Engine)。差分机是一个最终只能执行单操作的单用途机器。为此，Babbage 后来采用了更通用的解析机(Analytical Engine)，该解析机包

此为试读，需要完整PDF请访问：www.ertongbook.com

括了现代计算机的基本组成部分，因此人们将 Babbage 称为“计算机之父”。

19 世纪的研究不断取得发展。1854 年，Charles Boole 描写了符号逻辑系统，并以他的名字来命名，即布尔逻辑，并一直沿用到现在。布尔逻辑提出了一系列逻辑用语，如大于、小于、不等于，并建立了一套符号系统来描述这些术语。

1890 年，美国进行人口普查。人口的不断增加意味着处理这些数据花费的时间越来越长。为了提高和加快数据处理的速度，美国政府举办竞赛来引起人们对计算、传递数据处理设备研究的兴趣。Herman Hollerith 在比赛中获胜，该技术还被成功运用到其他国家的人口普查信息处理中。Hollerith 后来创办了 Hollerith Tabulating 公司，该公司后来与其他两家公司于 1914 年合作创办了 CTR 公司，也就是后来的 IBM(International Business Machines)公司的前身。

到 20 世纪 20 年代中期，数学计算工具很少用于商业、工业或工程领域。事实上，最常用的是类似于计算尺的工具。这种情况在 1925 年开始转变：在美国麻省理工学院(MIT)，Vannevar Bush 建立了一个规模巨大的差分分析器，该机器综合了积分和差分功能，成为世界上最庞大的“计算机”。

编程发展过程中的另一个重要人物是德国科学家 Konrad Zuse，他于 1936 年研制了 Z-1 计算机，这是一台利用继电器和以二进制计算系统为基础的机器，是当代计算机的先驱。Zuse 还发明了现代编程，他于 1946 年开发了世界上第一种编程语言——Plankalkul，他甚至还为 Z-3 计算机编写代码来与自己下国际象棋。

1945 年，计算机领域的一个重要单词——bug 被引入。一名技术人员发现一只小虫陷在电路之间，技术人员随即将这只小虫清除出来并将它贴在记录计算机使用和问题的日志上。日志上写着“发现第一个真正的小虫”。之后，人们便把程序或软件中存在的漏洞或缺陷称为 bug，而发现 bug 并加以纠正的过程便称为“debug”，即调试。

此后，研究进展加速前进。1949 年出现了短代码。这种代码必须手工转换成机器可读代码，即编译。1954 年，IBM 开始开发 FORTRAN 语言，这是第一种商业化的高级编程语言。1959 年，COBOL 语言诞生，这是一种主要使用在大型机器上的商业机构语言，现在仍有许多公司在使用。1968 年，Pascal 语言开始出现，该语言被广泛用于教学。1970 年，出现了 Smalltalk 和 B 语言。之后在 B 语言的基础上产生了 C 语言，C 语言简单、有效、灵活，开创了编程的新时代。C 语言从根本上改变了程序设计和思考问题的方式，其设计原理和语法对之后出现的所有主流编程语言都产生了深刻影响。

C、Pascal，以及 FORTRAN 语言的诞生，使程序员可以离开机器层次，在更抽象的层次上思考和描述问题。而在此之前，程序员需要使用汇编语言、机器语言来直接对计算机硬件进行操作，当程序比较小且简单时，这种方法可以奏效。而当程序进一步增长时，就十分难以控制和管理了。C、Pascal、FORTRAN 等结构化语言的出现，实现了轻松编写中等复杂程度的程序。但当软件项目达到一定的规模时，其复杂性仍然超出程序员的管理能力。事实上，在 20 世纪 70 年代末，这种情况就已经发生了，即“软件危机”。

为了解决这个问题，出现了一种新的思考程序设计方式和程序设计模型——面向对象程序设计(OOP，Object-Oriented Programming)，由此也诞生了一些支持此技术的程序设计语言，如 C++、Java、C#等。这些语言都以新的观点去看待问题，即问题都是由各种不同

属性的对象以及对象之间的消息传递构成。

20 世纪 90 年代以来，Internet 快速发展，由此产生了大量的网络编程语言，如 HTML、JavaScript、CSS、XML、ASP.NET、PHP 等。基于 Internet 的各种应用越来越多，也越来越丰富，网络编程已经成为编程的一个重要领域。

1.1.2　为何会存在如此多的编程语言

自编程语言产生以来，至少出现了 500 种。要理解这一现象，读者应首先理解计算机执行程序代码的过程。

每一类型的计算机其实只能理解一种语言，这种语言不是用来编写代码的语言，而是二进制语言，即由 0、1 组成的代码。编程的时候，确切地说就是编写代码的时候，真正要做的是编写另一程序可以理解的代码，该程序将代码翻译成计算机可以理解的代码，它就是解释程序，也即编译器，如图 1-1 所示。编写代码的时候，不是遵循计算机的规则，而是遵循能把代码翻译成计算机可理解形式的编译器的规则。

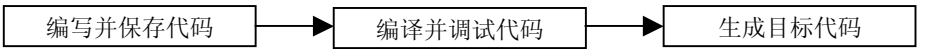

图 1-1　代码的编写过程

在图 1-1 中，程序员可以使用 C++、Basic 或其他编程语言来编写代码并进行保存，然后使用语言对应的编译器对代码进行编译和调试，最终生成可在计算机上运行的 EXE 可执行文件。

编译器是如何编译代码的？

编译器读取输入的代码，检查错误，确保输入的代码符合所有的规则并具有实际意义。如果发现问题，编译器就通知用户并中断运行。如果没有发现问题，就输出一个 EXE 可执行文件，该文件自身可以运行，执行代码行中的指令。

在 C++环境中，编译器输出的是机器代码，可以由 CPU 直接执行。因此，它依赖于特定的 CPU 和操作系统。如果想在不同的系统上运行 C++程序，就需要再次编译程序，使其成为适用于该环境的机器代码。因而，如果要创建能运行在不同环境中的 C++程序，就需要有该程序的多个不同的可执行版本。

而在 Java 和 C#环境下，编译器是将代码编译成一种被称为“中间语言”的伪代码。这种伪代码不依赖于具体的 CPU 和操作系统，而是由运行时系统执行的。对于 Java，这种运行时系统被称为 Java 虚拟机(Java Virtual Machine，JVM)，在 C#中称为公共语言运行时(Common Language Runtime，CLR)。因此，Java 程序可以在任何带有 JVM 的环境中运行，而 C#程序则可以在任何实现了 CLR 的环境中运行，从而实现了代码的跨平台性和可移植性。

下面的代码用不同语言在屏幕上输出“Hello World！”：

```
#include <iostream>          //C++版本的代码
using namespace std;
```

```
int main()
 {
 cout<<"Hello World!";
 }
 class HelloWorld{      //Java 版本的代码
    public static void main(String args[]){
       for(;;){
          System.out.print{"Hello World!"};}}}
<script>               //JavaScript 版本的代码
  alert("Hello World!")
</script>
```

1.1.3 开发环境

开发环境是用于输入代码的程序。有些语言需要指定的开发环境，如 Visual Basic；而有些语言仅需要一个文本编辑器就能编写，如 JavaScript；还有一些语言可以选择开发环境来使用，如 C++。

早期进行编程的各个阶段都要用到不同的程序软件，如先用字处理软件编写源代码，然后用链接程序进行函数、模块链接，再用编译程序进行编译。程序员必须在这几种软件间来回切换操作。现代编程通常是在一个集成的开发环境(IDE，Integrated Development Environment)中进行程序的编辑、编译、调试和发布。

IDE 是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面工具，是集成了代码编写功能、分析功能、编译功能、调试功能等功能的一体化的开发软件服务平台。例如 Microsoft 的 Visual Studio 系列，Borland 的 C++ Builder、Delphi 系列等。不同的技术体系有不同的 IDE，例如 Visual Studio 可以称为 C++、VB、C#等语言的 IDE，Borland 的 JBuilder 则是 Java 的 IDE。

1.2 了解 C++语言

C++建立在 C 的基础之上，它是 C 的一个超集。作为一种优秀的编程语言，C++的设计原理贯穿于整个计算领域，同时也是 Java 和 C#的始祖。要想成为一名专业的编程人员，就必须精通 C++，它是通向现代编程领域的大门。

1.2.1 C ++的诞生

C++的产生源于程序规模和复杂性的不断增长。C 语言可以被认为是现代编程的开端，但使用 C 语言只能进行结构化编程(又称面向过程编程)。结构化编程通过定义清晰的控制语句、带有局部变量的子例程等，解决了编写较为大型程序的难题。但结构化编程的一个很大弊端在于，程序的可重用性低，程序总是针对某一特定问题，如果要解决其他相同或相似的问题，就需要重新编程。当项目规模十分庞大时，采用结构化编程的方法，程序员的工作量会十分庞大，同时系统的性能也会受到严重影响。程序员无法高效地控制和管理软件。

解决方法便是使用面向对象编程，对 C 语言支持面向对象编程的期待最终导致了 C++

的诞生。1979 年，Bjarne Stroustrup 在美国新泽西州 Murray Hill 的贝尔实验室中发明了 C++。最初他将其称为“带类的 C”，1983 年被正式命名为 C++。C++包括 C 的所有功能、属性和优点。Stroustrup 向 C 添加的绝大多数功能都是用于支持面向对象编程的，从本质上讲，C++是 C 的面向对象版本。Stroustrup 将 C++建立在 C 的基础之上，使得 C 向 OOP 的过渡变得十分平滑，同时保持了 C 的高效、灵活及其设计理念。

尽管设计 C++的最初目标是帮助控制和管理超大型的程序，但 C++的作用并不局限于此。事实上，C++的面向对象功能可以高效地运用在所有编程任务上，包括编辑器、数据库、个人文件系统、网络工具、通信程序等。因为 C++共享了 C 的高效性，所以许多大型高性能系统软件都使用 C++来创建，同时 C++也是编写 Windows 程序最常用的语言。

C++自诞生以来，经历了 3 次主要的修订，每一次修订都会对语言进行一些添加和修改。第一次修订是在 1985 年，第二次是在 1990 年，第三次修订则发生在 C++的标准化过程中。对 C++进行标准化是为了统一和规范市场上出现的不同版本的 C++，标准化工作由 ANSI(American National Standard Institute，美国国家标准委员会)以及后来加入的 ISO(International Organization for Standardization，国际标准化组织)负责。

第一个标准草案产生于 1994 年 1 月 25 日，在该草案中，ANSI/ISO 委员会保持了 Stroustrup 最初定义的功能，并添加了一些新的内容。该草案发布不久，Alexander Stepanov 创建了标准模板库(Standard Template Library，STL)，这极大扩展了 C++的功能。STL 的加入延缓了 C++的标准化工作。

C++的标准化过程经历了较长的时间，在这期间，许多新的功能被陆续添加到语言中，还进行了一些小的修改。实际上，ANSI/ISO 委员会定义的 C++版本要比 Stroustrup 的原始版本大得多，也复杂得多。1998 年，C++的 ANSI/ISO 标准最后成形，也就是我们经常说的标准 C++。采用标准 C++编写的代码可以被所有主流的 C++编译器支持。

提示：按照 C++标准委员会的惯例，每 5 年会根据最近 5 年 C++的使用情况对 C++标准库进行一次更新，2009 年恰逢 C++业内的这一大事件。此次新标准的发布，相信会引发学习 C++的新高潮。

1.2.2　面向对象编程

C++的核心是面向对象编程，OOP 是 C++诞生的主要动力。在学习 C++编程之前，用户必须理解 OOP 的基本原理。从普通意义上讲，程序可以采用两种方式来组织：按照代码(将要发生的事情)来组织，或按照数据(将要影响到谁)来组织。

在结构化编程方式下，程序通常是按照代码来组织的，可以理解为“利用代码对数据进行处理”。面向对象编程则是按照数据来组织的，可以理解为“数据控制对代码的访问”。在面向对象的语言中，可以定义数据以及能对这些数据进行处理的方法。因此，一种数据类型可以精确定义施加于其上的各种操作。

为了支持面向对象编程的设计理念，所有的 OOP 语言(C++也不例外)都包含如下 3 个特性：封装性、多态性和继承性。

1. 封装性

封装性是一种编程机制，它将代码和代码要处理的数据绑定在一起，并使其保持安全，

免受外界干扰和误用。在面向对象编程语言中，可以通过创建自包含的“黑箱”将代码和数据绑定在一起。黑箱内部是所有必需的数据和代码。当代码和数据以这种方式连接在一起时，对象就被创建出来了。从这个角度来讲，对象是实现封装性的手段。

对象内部的代码、数据，可能是私有的，也可能是公有的。私有的代码或数据只能由对象内部的程序访问，除非代码或数据是公有的，否则对象之外的程序无法访问。通常情况下，对象的公有部分用来为对象的私有成员提供控制接口。

C++封装的基本单位是类，类定义了对象的形式。它不仅指定了数据，而且指定了要在这些数据上进行的操作。对象是类的实例，即 C++使用类来创建对象，这称为类的实例化。组成类的代码和数据称为类的成员。

2. 多态性

多态性是采用面向对象编程的最直接想法，也是让程序员最激动不已的事情。多态性是指同一个消息要有不同的响应，即两个或多个对象响应同一个消息的能力，每个对象都以自己的方式来响应消息。当然，在相同的对象中，对这个消息的响应应该是相同的。多态性允许程序员在不同的对象之间辨认和使用相似性，例如，不同类型的文件(文本的、图形的，等等)一般都有“打印”功能，用户发送命令“打印”，各类文件则用自己的“打印”方法来响应。

更通俗一些，多态性可以用“多种方法，一个接口”来表达。也就是说，程序员可以为相似的动作设计一个通用的接口。多态性允许为一类通用的动作(如打印文本、打印图像等)指定相同的接口(打印)，降低了程序的复杂性。至于应该选择哪种特定的动作(是打印文本，还是打印图像)，则留给编译器来完成，程序员不必关注这些，只需要知道这个通用接口即可。

3. 继承性

继承性是一个对象获得其他对象属性的过程。在 C++中，类是有层次的，通过向下细化和向上抽象，就可以成长为一个“类树”。构建类树的好处是允许代码重用，这通过继承来实现。这好似达尔文进化论中的进化树，基因的遗传(继承)和变异(方法的重载)。

例如，“人”作为一个类，它可以向下细化为“男人”和“女人”，向上则可以抽象为“哺乳动物”。这就是说，哺乳动物的一些特性也就适用于它的子类——人。而除了这些特性外，人还有自己的一些特性，这些特性将人与其他哺乳动物分开。类似地，男人和女人又继承了人的所有特性，同时拥有各自独特的特性。

如果不使用继承，那么每个对象就必须定义自己的所有特征。而使用了继承性，对象只需要定义使其在类树中保持唯一性的特征即可，它可以从它的上一级继承通用的属性。

1.2.3 C ++与 Java 和 C#的关系

Java 是由 Sun Microsystems 公司开发的，C#则是由 Microsoft 公司开发的。C++是 Java 和 C#的始祖，尽管 Java 和 C#添加、删除和修改了 C++的许多功能，但总的来说，这 3 种语言的语法几乎完全相同。另外，这三种语言的整体风格也十分相似。也就是说，掌握了 C++，就很容易掌握 Java 和 C#；掌握了 Java 和 C#，那么学习 C++也会变得相对容易。

C++、Java 和 C#的主要区别在于三者的计算环境不同。C++用于为特定类型的 CPU 和

操作系统编写高性能的程序；而 Java 和 C#则是为了满足 Internet 联机环境下特殊的编程需求而开发的，因此，跨平台和可移植性就是它们的主要特性。但这样做的代价是程序的执行速度要比 C++慢得多，因而 Java 和 C#的运行时系统介于程序和 CPU 之间，要产生一定的系统开销。因此，如果要创建高性能的软件，使用 C++；如果要创建可移植性高的软件，则使用 Java 或 C#。

1.3　使用 Visual C++ 2008 编程

学习使用 C++进行 Windows 编程并不困难，事实上，Microsoft Visual C++ 2008 使之变得相当容易。为了继续本书的内容，读者需要在本机上安装 Visual Studio 2008 Standard Edition、Visual Studio 2008 Professional Edition 或 Visual Studio 2008 Team System，本书使用的是 Visual Studio 2008 Professional Edition。在安装之前，请读者查看安装的环境说明，以确保能成功安装。

1.3.1　了解 Windows 编程

在编写 Windows 下执行的交互式应用程序时，编程人员总是有两个基本方面需要考虑：需要创建程序的 GUI(Graphical User Interface，图形用户界面)，如窗口、对话框、消息框等，它们用于与用户交互；另外还需创建处理这些交互的代码，以实现应用程序的具体功能。

Visual C++ 2008 提供了用于帮助编程人员开发 Windows 应用程序的许多工具，就可以开发的 Windows 应用程序和程序组件的类型而言，可以有两种选择：可以编写在 CLR 中执行的代码；也可以编写能直接编译为机器代码、从而在本地执行的代码。

就针对 CLR 的 Windows 应用程序而言，可以使用.NET Framework 提供的 Windows Forms 作为 GUI 基础。通过使用 Windows Forms，程序员可以快速开发 GUI，因为可以直观地组装 GUI，而代码是完全自动生成的。之后，程序员只需要定制已经生成的代码，就可以提供所需的功能。

至于本地执行的代码，编程人员可以采用多种方法来编写。其中一种是使用 MFC(Microsoft Foundation Classes，Microsoft 基础类)来编写 Windows 应用程序的图形用户界面。MFC 中的各种类结合起来构成了一个应用程序框架，其目的是让编程人员在此基础上建立 Windows 下的应用程序。相对 SDK 来说，这种方法更简单。因为总体上 MFC 框架定义了应用程序的轮廓，并提供了用户界面的标准实现方法，编程人员所要做的就是通过预定义的接口把具体应用程序特有的东西填入这个轮廓。

提示：Visual C++对使用 MFC 进行 Windows 编程提供了强大支持，AppWizard 可以用来生成初步的框架文件(代码和资源等)，资源编辑器用于帮助直观地设计用户界面，classWizard 用来协助添加代码到框架文件，最后，编译则通过类库实现了应用程序特定的逻辑。

在 CLR 中执行的 C++代码被描述为托管的 C++，因为数据和代码是由 CLR 管理的。非托管的 C++代码被称为本地 C++，因为代码被直接编译成本地机器代码。在 CLR 程序

此为试读，需要完整PDF请访问：www.ertongbook.com