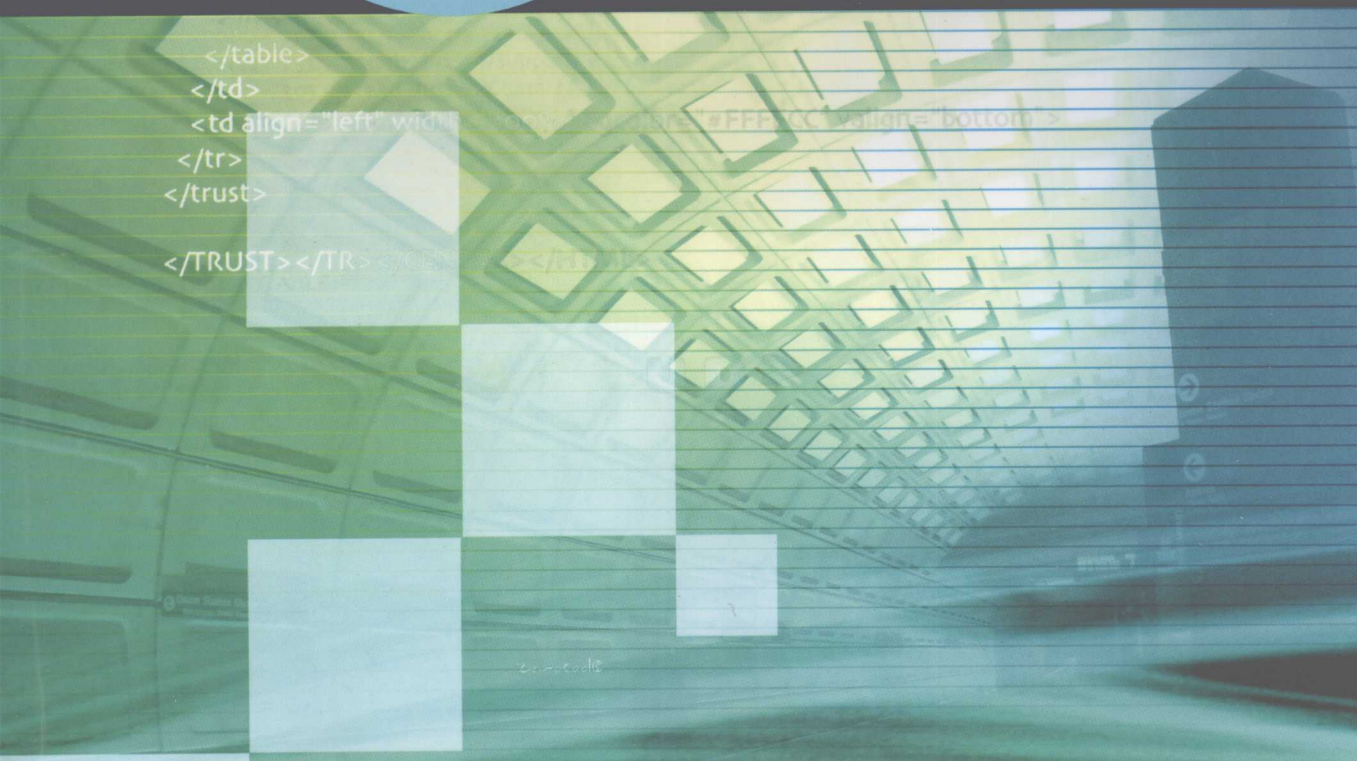
普通高等教育“十一五”规划教材

高等院校计算机科学与技术系列教材

C++程序设计教程

施平安　段　超　郝清赋　编著



冶金工业出版社

普通高等教育“十一五”规划教材
高等院校计算机科学与技术系列教材

C++程序设计教程

施平安　段　超　郝清赋　编著

北　京
冶金工业出版社

内 容 简 介

本书是根据普通高等教育“十一五”规划教材的指导精神而编写的。

C++语言是20世纪70年代在C语言的基础上推出的一种程序设计语言，经过数十年的发展和完善，它凭借着优异的性能和强大的功能，至今依然是软件业选择开发语言的首选和主力军。

本书较为全面地介绍了C++程序设计语言的各种特性和功能。全书共分为14章，内容包括：C++的基本知识、面向对象编程技术和C++高级主题，诸如运算符重载、模板、异常处理、流和字符串处理等。在介绍各个主题时，不仅提供了大量的示例、表格、插图，而且还在各章后面安排了一定量的练习，以供读者在复习时使用。

本书可作为高等院校计算机及其相关专业的教材，也可作为从事C++程序开发人员的参考书。

图书在版编目（CIP）数据

C++程序设计教程 / 施平安，段超，郝清赋编著. —北京：冶金工业出版社，2007.4
普通高等教育“十一五”规划教材
ISBN 978-7-5024-4255-2

I. C… II. ①施…②段…③郝… 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（2007）第044739号

出版人 曹胜利（北京沙滩嵩祝院北巷39号，邮编100009）
责任编辑 肖放
ISBN 978-7-5024-4255-2
广州锦昌印务有限公司印刷；冶金工业出版社发行；各地新华书店经销
2007年4月第1版第1次印刷
787mm×1092mm 1/16； 20印张； 459千字； 310页
32.00元
冶金工业出版社发行部 电话：（010）64044283 **传真：**（010）64027893
冶金书店 地址：北京东四西大街46号（100711） 电话：（010）65289081
（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、关于本书

本书是根据普通高等教育“十一五”规划教材的指导精神而编写的。

C++语言是 Bjarne 博士在 C 语言的基础上开发出来的一种基本编程语言，它是 C 语言的超集。在历经数十年的发展和完善以后，C++语言已经成为软件业首选的程序开发语言。如今，随着科技的飞速发展，各种行业对增加利润和降低成本这两个目标近乎疯狂地追逐，使得软件业在生产生活中的地位越来越重要，进而使得软件从业人员的待遇水涨船高。为了使读者能够更好地学习掌握有关 C++编程方面的知识，我们特地编写了本书，以期那些有志投身于软件业的学生们能够更好、更快地为将来做好必要的知识和技能准备。

二、本书结构

本书共分三个部分，第一部分（第 1 章到第 6 章）讲授 C++程序设计的基础知识，主要包括 C++中的变量、函数、数组和指针等主题；第二部分（第 7 章到第 13 章）主要讲授 C++程序设计的中级知识，主要包括类、继承、运算符重载、流、模板、异常处理；第三部分（第 14 章）将介绍 Windows 编程和 MFC 类库的基本知识。具体结构如下：

第 1 章：计算机编程与 C++概述。主要介绍计算机编程的基本知识，讲述 C++语言的发展历程及其在软件发展中的地位等。

第 2 章：控制结构。主要介绍各种循环结构和选择结构的使用方法；break 语句和 continue 语句；结构化编程的原则。

第 3 章：函数。主要介绍如何使用函数来模块化地构造程序；如何声明函数原型，定义函数体；什么是作用域规则；函数之间传递信息的机制；递归的概念以及如何区分递归和迭代；定义和使用重载函数。

第 4 章：数组。主要介绍 C++数组的概念；声明、初始化数组，并能够引用数组中的元素；如何将数组传递给函数；基本的排序和查找技术；声明和操作多维数组。

第 5 章：指针。主要介绍指针、const 指针和字符指针的概念；指针与数组、函数之间的关系；如何动态地分配数组的内存空间；指针数组的概念；命令行参数的概念及其使用方法。

第 6 章：结构。主要介绍结构的概念；声明、初始化数组，并能够引用结构中的成员；结构与数组、指针和函数之间的关系；链表的概念，如何使用结构来创建、显示和删除链表。

第 7 章：类。主要介绍类的基本概念；类的作用域和成员访问限定符；构造函数的概念及其作用；如何重载构造函数等。

第 8 章：继承。主要介绍继承的概念；基类和派生类的概念和相互关系；protected 成员和访问限定符；派生类中的构造函数和析构函数；多重继承；虚拟继承；构造函数和析构函数在多重继承中的调用顺序。

第 9 章：多态。主要介绍多态的概念；基类对象与派生类对象在继承层次中的相互关系；虚函数的概念及其使用方法和使用时机；抽象类和具体类的概念及其相互区别。

第 10 章：运算符重载。主要介绍运算符重载的概念；运算符重载的限制条件；如何将运算符函数作为类的成员或非类成员；如何重载各种运算符。

第 11 章：流。主要介绍流的基本概念；输出流的概念及其提供的函数；输入流的概念及其提供的函数；如何根据实际需要定制输出格式；文件与流的关系。

第 12 章：模板。主要介绍模板的基本概念；如何根据实际需要重载函数模板；如何重载类模板中的成员函数；模板与继承之间的关系。

第 13 章：异常处理。主要介绍异常和错误处理的基本概念；如何抛出、捕获和处理异常；try 块、catch 块和 throw 语句的作用和用法等。

第 14 章：MFC 库和 Windows 程序开发。主要介绍 Windows 程序和 MFC 库的基本知识；如何定义和注册一个窗口类，如何创建窗口以及如何定义窗口的消息循环等。

三、本书特点

本书是一本 C++程序设计教程，主要介绍使用 C++语言进行编程的相关主题，诸如 C++语言的特性、面向对象编程的概念、数据组织和 C++语言高级特性的其他应用等。为了使读者能尽快地进入状态，本书在结构安排上将遵循由浅入深的原则。从最基本的原理知识入手，首先介绍 C++语言的基本特性，然后逐渐深入到面向对象编程技术和异常处理等较为高级的主题。在介绍各个主题时，将保持语言风格的前后一致，尽量使用浅显的语言来描述各种编程问题。为了配合读者的学习，每个章节不仅提供了大量的示例、表格、插图，而且还在各章后面安排了一定量的练习，以供复习时使用。

四、学习指导

本书的总学时以 60 学时较为合适，根据不同层次的学生和课程设置的要求，书中有些理论性较强的、难度较大的内容可以选学或不学。对于以前没有接触过 C 编程语言的学生，可以依次地学习本书各章的内容；对于以前学习过 C 编程语言的学生，可以跳过第 2、3、4、5 和 6 章的学习，转而直接学习后面的知识。

五、适用对象

本书可作为高等院校计算机及其相关专业的教材，也可作为从事 C++程序开发人员的参考书。

由于编者水平有限，编写时间仓促，书中如有疏漏和不足之处，望广大读者不吝赐教，以便使本书质量得到进一步的提高。联系方式如下：

电子邮箱：**service@cnbook.net**

网址：**www.cnbook.net**

本书电子教案、源代码及习题参考答案可在该网站下载，此外，该网站还有一些其他相关书籍的介绍，可以方便读者选购参考。

编　者

2007 年 3 月

目　录

第 1 章　计算机编程与 C++概述 1

1.1　简介 1
1.2　计算机语言 1
1.3　C++发展简史 2
1.4　程序设计开发技术简介 3
1.4.1　过程式技术 3
1.4.2　结构化技术 4
1.4.3　对象技术 4
1.5　面向对象程序设计技术 5
1.5.1　封装 5
1.5.2　继承 5
1.5.3　多态性 6
1.6　C++标准库 6
1.7　C++编程简单示例：输出文本 6
1.8　标识符 9
1.9　关键字 9
1.10　变量、数据类型和常量 10
1.10.1　变量 10
1.10.2　基本数据类型 11
1.10.3　变量赋值 11
1.10.4　常量 13
1.11　运算符 14
1.11.1　算术运算符 14
1.11.2　关系运算符 15
1.11.3　逻辑运算符 15
1.11.4　按位运算符 16
1.11.5　sizeof 运算符 17
1.11.6　问号运算符 17
1.11.7　复合赋值运算符 17
1.11.8　运算符优先级和结合性 18
1.12　C++编程示例：求平均数 19
小结 20
习题一 21
一、填空题 21
二、判断题 21
三、简答题 21
四、编程练习题 22

第 2 章　控制结构 23

2.1　简介 23
2.2　控制结构 23
2.2.1　goto 语句 23
2.2.2　控制结构 24
2.3　if 选择结构 25
2.4　if/else 选择结构 29
2.5　while 循环结构 31
2.6　do/while 循环结构 33
2.7　for 循环结构 33
2.8　switch 多重选择结构 35
2.9　break 和 continue 语句 37
2.10　C++编程示例：数学运算器 38
小结 40
习题二 41
一、填空题 41
二、判断题 41
三、简答题 41
四、编程练习题 42

第 3 章　函数 43

3.1　简介 43
3.2　函数声明 44
3.3　函数定义 45
3.3.1　定义不包含参数的函数体 45

3.3.2 定义带有参数的函数体 46
3.4 函数调用 46
3.5 作用域规则 48
3.6 递归 49
3.6.1 无穷递归 50
3.6.2 递归编程 50
3.6.3 直接递归和间接递归 51
3.6.4 递归举例 52
3.6.5 递归与迭代 53
3.7 内联函数 54
3.8 函数重载 55
3.9 C++编程示例：字符处理函数 56
小结 57
习题三 58
一、填空题 58
二、判断题 58
三、简答题 58
四、编程练习题 59
第 4 章 数组 60
4.1 简介 60
4.2 声明和初始化数组 60
4.2.1 数组声明 61
4.2.2 初始化 C++数组 61
4.2.3 访问 C++数组 61
4.2.4 数组使用示例 62
4.3 向函数传递数组 63
4.4 数组排序 64
4.4.1 选择排序法 64
4.4.2 插入排序法 65
4.4.3 冒泡排序法 66
4.4.4 快速排序法 67
4.5 数组查找 69
4.5.1 线性查找 69
4.5.2 二分查找 69
4.5.3 比较查找算法 70
4.6 多维数组 71
4.7 字符数组 72
4.8 C++编程示例：统计学生成绩 72
小结 74
习题四 75
一、填空题 75
二、判断题 75
三、简答题 76
四、编程练习题 76
第 5 章 指针 78
5.1 简介 78
5.2 指针 78
5.2.1 什么是指针 78
5.2.2 指针变量 79
5.2.3 指针运算符 79
5.2.4 指针的算术运算 80
5.2.5 void 型指针 81
5.3 const 指针 81
5.3.1 指向常量数据的指针 81
5.3.2 指针常量 82
5.3.3 指向常量数据的指针常量 82
5.4 指针与数组 83
5.4.1 两者间的关系 83
5.4.2 内存分配 84
5.4.3 动态分配数组 85
5.5 字符指针 86
5.5.1 字符指针与字符串 86
5.5.2 <cstring>头文件 87
5.6 指针数组 89
5.7 指针与函数 90
5.7.1 函数指针 90
5.7.2 向函数传递指针参数 90
5.7.3 函数返回指针 91

5.8 命令行参数 91
5.9 C++编程示例：字符串转换函数 92
小结 94
习题五 94
一、填空题 94
二、判断题 95
三、简答题 95
四、编程练习题 95
第 6 章 结构 97
6.1 简介 97
6.2 结构 97
6.2.1 声明结构 98
6.2.2 声明结构类型的变量 98
6.2.3 引用结构成员 99
6.3 结构与数组 100
6.4 结构与指针 102
6.5 结构与函数 103
6.6 链表 105
6.6.1 声明节点 106
6.6.2 创建和显示链表 106
6.6.3 插入链表节点 108
6.6.4 删除链表节点 110
6.6.5 链表示例 111
6.7 C++编程示例：扑克牌游戏 112
小结 114
习题六 115
一、填空题 115
二、判断题 115
三、简答题 115
四、编程练习题 116
第 7 章 类 117
7.1 简介 117
7.2 实现 Angle 类 117
7.3 类作用域和访问类成员 120
7.4 控制对类成员的访问 121
7.5 构造函数 121
7.5.1 一般概念 121
7.5.2 重载构造函数 121
7.5.3 再谈构造函数：拷贝构造函数 124
7.5.4 浅拷贝和深拷贝 126
7.6 析构函数 128
7.7 读写函数与功能函数 129
7.8 构造函数与析构函数小结 132
7.9 const 成员函数和 const 对象 134
7.10 组合：引用作为其他类的成员 136
7.11 友元 137
7.12 使用 this 指针 139
7.13 静态类成员 142
7.14 C++编程示例：string 标准类 145
小结 150
习题七 150
一、填空题 150
二、判断题 151
三、简答题 151
四、编程练习题 151
第 8 章 继承 153
8.1 简介 153
8.2 基类与派生类 154
8.3 继承示例 154
8.4 protected 成员和访问限定符 160
8.5 派生类中的构造函数与析构函数 162
8.6 多重继承 165
8.7 虚拟继承 168
8.8 多重继承中构造和析构的顺序 170
8.9 C++编程示例：string 标准类的迭代器 173

小结 174
习题八 175
一、填空题 175
二、判断题 175
三、简答题 176
四、编程练习题 176

第 9 章 多态 177

9.1 简介 177
9.2 继承层次中对象间的关系 177
9.3 虚函数 182
9.4 C++编程示例：多态示例 183
9.5 抽象类与具体类 186
9.6 虚函数的注意事项 193
9.7 C++编程示例：字符串查找函数 194
小结 195
习题九 196
一、填空题 196
二、判断题 196
三、简答题 196
四、编程练习题 197

第 10 章 运算符重载 198

10.1 简介 198
10.2 运算符重载的基本知识 198
10.3 运算符重载示例：重载>>运算符 199
10.4 运算符重载的限制 202
10.5 用作类成员和非类成员的运算符函数 202
10.6 C++编程示例：重载自增运算符 205
小结 207
习题十 208
一、填空题 208
二、判断题 208
三、简答题 208
四、编程练习题 209

第 11 章 流 210

11.1 简介 210
11.2 流的概念 210
11.3 输出流 212
11.3.1 put 函数 212
11.3.2 write 函数 213
11.4 输入流 213
11.4.1 get 函数 214
11.4.2 getline 函数 216
11.4.3 read 函数 217
11.5 格式化输出 217
11.5.1 设置整数的基数 218
11.5.2 设置浮点数的精度 219
11.5.3 设置宽度 220
11.5.4 对齐 221
11.5.5 填充 222
11.5.6 科学记数法和定点记数法 223
11.5.7 控制大小写 223
11.6 文件与流 224
11.6.1 ofstream 类 224
11.6.2 ifstream 类 227
11.6.3 fstream 类 229
11.7 C++编程示例：字符串流 231
小结 233
习题十一 233
一、填空题 233
二、判断题 234
三、简答题 234
四、编程练习题 234

第 12 章 模板 236

12.1 简介 236
12.2 模板的概念 236
12.2.1 函数模板 237
12.2.2 类模板 237
12.3 函数模板举例 238
12.4 重载函数模板 240
12.5 类模板 241
12.5.1 类模板简单示例 241
12.5.2 为类模板中成员函数的参数提供默认值 242
12.5.3 重载类模板的成员函数 243
12.5.4 为类模板参数提供默认值 245
12.6 C++编程示例：链表类模板 246
12.7 其他模板问题 251
小结 251
习题十二 251
一、填空题 251
二、判断题 252
三、简答题 252
四、编程练习题 252

第 13 章 异常处理 254

13.1 简介 254
13.2 异常处理的基本知识 255
13.3 异常规范 257
13.3.1 try 块 257
13.3.2 catch 块 257
13.3.3 throw 语句 257
13.3.4 注意事项 258
13.4 异常处理举例 260
13.5 处理意外异常 263
13.5.1 异常抛出表 263
13.5.2 处理意外异常 264
13.6 异常与构造函数和析构函数 265
13.6.1 构造函数与异常 265
13.6.2 析构函数与异常 266
13.7 new 操作失败 266
13.8 标准异常的类层次 270
13.9 C++编程示例：字符串内存处理函数 271
小结 273
习题十三 274
一、填空题 274
二、判断题 274
三、简答题 274
四、编程练习题 275

第 14 章 MFC 库和 Windows 程序开发 276

14.1 简介 276
14.2 Windows 编程基础知识 276
14.2.1 Windows 程序的入口点 277
14.2.2 窗口 278
14.2.3 处理事件 280
14.3 Windows 程序开发示例 281
14.4 MFC 库概述 287
14.5 MFC 程序示例：单文档/视图程序 291
14.5.1 创建 MFCSample 应用程序开发项目 291
14.5.2 修改工具栏 292
14.5.3 添加成员变量 292
14.5.4 添加消息处理函数 293
14.5.5 为消息处理函数添加处理代码 295
14.6 MFC 程序示例：对话框 299
14.6.1 创建 DlgSample 应用程序开发项目 299
14.6.2 创建对话框界面 299
14.6.3 添加消息处理函数 300
14.6.4 添加成员变量和成员函数 301

14.6.5 为消息处理函数添加处理代码 303
小结 307
习题十四 308
一、填空题 308
二、判断题 308
三、简答题 308
四、编程练习题 308
参考文献 **310**

第 1 章　计算机编程与 C++概述

学习目标：

- 了解 C++的发展历史
- 了解各种程序设计方法的优缺点
- 能够声明变量，为变量赋值
- 能够使用基本数据类型
- 掌握算术运算符、关系运算符和逻辑运算符的用法
- 掌握按位运算符、sizeof 运算符和其他运算符的用法
- 能够进行类型转换
- 掌握运算符的优先级次序和结合方向

1.1　简介

C++是一种通用的程序设计语言，它以其独特的语言机制和全面支持面向对象编程的优势受到了很多程序员的钟爱，并在计算机科学领域得到了广泛的应用，例如，读者所熟悉的 Windows 操作系统就是用 C++和 C 语言编写的。

从现在开始，将进入 C++语言的奇妙世界。在讨论 C++的语言特性之前，先用一章的篇幅简要地介绍 C++的历史、程序设计方法的发展历程和对象编程技术等一些基础知识，以使读者对 C++语言能够有一个总体的了解，然后介绍关于 C++变量和基本数据类型的主题，最后讨论 C++运算符。

1.2　计算机语言

计算机程序是用某种程序设计语言编写出来的动作序列，这些动作在编写上必须符合程序设计语言所要求的规范，它们代表着程序员的思想，表达了程序员要求计算机执行的操作。目前有数百种程序设计语言，其中有些语言是计算机能够直接理解并予以执行的，而有些语言则需要经过一系列翻译步骤后，才能被计算机所理解并予以执行。大体上，这些程序设计语言可以分为以下三类：

（1）机器语言。

（2）汇编语言。

（3）高级语言。

机器语言是任何一台特定计算机的“自然语言”，这种语言由计算机的硬件设计所定义。只有机器语言才能被计算机所识别和运行。机器语言通常由一系列二进制数组成，它们指示计算机一次执行一个最基本的操作。注意，机器语言是与计算机类型相关的，换句话说，一种特定的机器语言只能在一种计算机类型上使用。但是，机器语言的可读性、易用性实在是太差了，而且特别容易出错。例如，下面这段机器语言程序的作用是将班级中的男生人数（MENNUMBER）与女生人数（WOMENNUMBER）相加，并将结果存入总人数（TOTALNUMBER）中：

```
+1326042447
+1456594391
+1287047227
```

对于人而言，这段机器语言程序代码实在是令人费解。为此，人们后来发明了汇编语言。汇编语言采用类似英语单词的缩写指令来表示基本操作，例如 ADD、SUB、LOAD 和 STORE 等，以改善程序设计语言的可读性。在运行由汇编语言编写的程序时，会首先以计算机速度将汇编语言程序翻译成机器语言程序，然后再运行机器语言程序，以得到所需要的结果。下面这段汇编语言程序也是将班级中的男生人数（MENNUMBER）与女生人数（WOMENNUMBER）相加，并将结果存入总人数（TOTALNUMBER）中，但是比原先的机器语言更加清晰：

```
LOAD    MENNUMBER
ADD     WOMENNUMBER
STORE   TOTALNUMBER
```

注意：尽管在人看来，这段代码非常容易理解，但除非由汇编程序翻译成机器语言，否则计算机仍然不能理解它们。

尽管汇编语言提高了可读性和易用性，但是它的专业性还是太强了，当时编写程序的都是计算机专业人员，这极大地制约了汇编语言的普及，并且对于最简单的任务，也须编写多条汇编指令。为了提高编程的速度，人们又相继开发了高级语言，其中一条语句就能够完成多条汇编语句才能完成的工作，例如：

```
TOTALNUMBER = MENNUMBER + WOMENNUMBER
```

与机器语言和汇编语言相比，高级语言更接近于人们的思维和表达习惯，这就从根本上解决了机器语言和汇编语言所存在的可读性差、难以维护的缺点。从程序员的角度看，他们更需要高级语言，而不是机器语言或者汇编语言。目前，C++和 Java 是最流行的、功能最强大的高级程序设计语言。但是，计算机并不能理解高级语言所表示的指令，因此，在能够执行高级语言所编写的程序之前，必须将其翻译为计算机所能够理解的机器语言。为此，人们又开发了编译程序，以便直接执行高级语言程序，毋需事先将其编译成机器语言。

总之，程序语言越低级，则在描写程序时，指令就越复杂难懂，程序语言越高级，则在描写程序时，指令就越简单易懂。语言越低级，就越能被计算机理解，越高级，就越能被人们理解和接受。计算机程序设计语言的发展总是从低级到高级，直到可以用人的自然语言来加以描述。

1.3 C++发展简史

语言的发展是一个逐步渐进的过程，C++是直接从 C 语言发展过来的，而 C 语言又是从更老的两种程序语言演变而来的。这两种语言分别是 BCPL 和 B，其中 B 语言是 BCPL 的一个解释性后代，BCPL 是 Basic CPL。BCPL 由 Martin Richards 于 1967 年研发成功，用于编写操作系统软件和编译程序。其中最有趣的是 CPL 中 C 的由来，由于当时这个语言是剑桥大学和伦敦大学合作开发的，在伦敦的人员加入之前，C 表示剑桥，伦敦人员加入之后，C 表示 Combined 组合。还有一种非正式的说法，C 表示 Christopher，因为 Christopher 是 CPL 背后的主要动力。

后来，Ken Thompson 以 BCPL 为基础，在他的 B 语言中建模构造了许多相似的特性。不管 BCPL 还是 B 都是“无类型”语言——每个数据项都在内存中占据一个“字”的空间，

要由程序员自行决定将一个数据项作为整数还是作为实数来处理。

C语言由贝尔实验室的Dennis Ritchie在B的基础上研制而成，最初于1972年，在一台DEC PDP-11计算机上实现的。C沿用了BCPL和B的许多重要概念，同时添加了数据类型以及其他一些特性。到20世纪70年代末，C已发展演化为今天所说的“传统C语言”。1978年，Kernighan和Ritchie在他们合著的《The C Programming Language》一书中全面地介绍了传统的C语言，这本书已经成为最经典的计算机学术著作之一。C在各种类型计算机上的迅速推广最终导致了C语言的多种衍生版本。各种版本大体类似，但往往互不兼容。显然，对于希望开发出能够具备良好可移植性代码的开发人员来说，他们迫切地需要一种标准的C语言版本。为此，1983年，由美国国家计算机和信息处理标准协会（X3）正式成立了X3J 11技术委员会，目的是“提供一个无歧义的、与机器无关的语言定义”，并于1989年，正式制订了C语言的标准：ANSI C。Kernighan和Ritchie合著的第二版《The C Programming Language》（1988年出版）非常详尽介绍了ANSI C版本语言的全部内容，这是目前全球使用最广的C语言版本之一。

最初导致C++诞生的原因是在Bjarne博士等人试图去分析UNIX内核的时候，这项工作开始于1979年4月，当时由于没有合适的工具能够有效地分析由于内核分布而造成的网络流量，以及怎样将内核模块化。同年10月，Bjarne博士完成了一个可以运行的预处理程序，称之为Cpre，它为C加上了类似Simula的类机制。在这个过程中，Bjarne博士开始思考是不是要开发一种新的语言，当时贝尔实验室对这个想法很感兴趣，就让Bjarne博士等人组成一个开发小组，专门进行研究。

当时不是叫做C++，而是C with class，这是把它当作一种C语言的有效扩充。由于当时C语言在编程界居于老大的地位，要想发展一种新的语言，最强大的竞争对手就是C语言，所以当时有两个问题最受关注：C++要在运行时间、代码紧凑性和数据紧凑性方面能够与C语言相媲美，但是还要尽量避免在语言应用领域的限制。在这种情况下，一个很自然的想法就是让C++从C语言继承过来，但是Bjarne博士更具有先见之明，他为了避免受到C语言的局限性，参考了很多的语言，例如从Simula继承了类的概念，从Algol68继承了运算符重载、引用以及在任何地方声明变量的能力，从BCPL获得了//注释，从Ada得到了模板、名字空间，从Ada、Clu和ML取来了异常。

1983年12月，Rick Mascitti建议命名为CPlusPlus，即C++。在经历了3次对C++的修订后，于1994年ANSI/ISO委员会制订了ANSI C++标准的草案。以后又经过不断完善，1998年11月，正式地批准了ANSI C++标准。C++包含C的全部特征、属性和优点，同时又增加了对面向对象编程的支持，而这一关键特性引发了软件业关于程序设计开发的一场革命，使人们从过去的过程式开发技术和结构化开发技术过渡为面向对象开发技术。

1.4 程序设计开发技术简介

本节将简要地介绍以下3种程序设计开发技术：过程式技术、结构化技术和面向对象技术。

1.4.1 过程式技术

过程式编程技术关注的是数据的处理过程或处理算法，其基本的编程单位是函数。在

过程式编程技术中，数据和程序是分开存储的，程序员的主要工作是追踪哪些函数调用哪些函数，并使哪些数据发生变化。因此，过程式程序开发人员将精力主要集中在函数的编写上，根据所要执行的任务对函数进行分组，不同的函数组合在一起，就构成了程序。在C这样的过程式编程语言中，数据的主要用途是为函数的动作提供支持。在系统需求文档中，动词帮助程序员决定并设计一系列协同工作、用于实现整个系统的函数。

对于过程式编程来说，它的一个主要问题在于，程序员创建的程序单元不能方便而有效地对应现实世界的实体。因此，它们的重用性较差。最普遍的情况是，程序员的每个新项目都得“从头开始”，从头编写非常相似的软件。大量的重复劳动，浪费了大量宝贵的时间和金钱。

1.4.2 结构化技术

结构化技术的主要思想是功能分解，以达到分而治之的目的。对于一项十分复杂的任务，可以将其分解为一系列较小的功能部件，直至这些功能部件达到可理解的程度。例如，对于计算某家商场中商品总额的任务，可以将其分解为以下子任务：

（1）得到某个商品的数量及其单价。

（2）计算该项商品的价格总数。

（3）计算所有商品的价格总数。

其中，得到某个商品的数量及其单价又可分为以下子任务：

（1）找出该项商品的记录。

（2）读出该项商品的数量。

（3）读出该项商品的单价。

同理，读出该项商品的数量又可以分为以下子任务：

（1）打开商品的记录文件。

（2）找出正确的记录。

（3）从磁盘文件中读取数据。

从上面这个例子中可以看出，结构化技术为解决复杂的问题提供了有力的支持。但是，随着程序规模的不断增大，程序所要处理的数据量也越来越大，数据与处理数据的方法之间的分离使得程序变得愈发难以理解和维护。

1.4.3 对象技术

面向对象编程技术本质上是将数据和处理数据的行为封装在一个整体，即封装在对象中。对象可以在其他对象面前隐藏自己的具体实现细节。尽管有的对象可以通过良好定义的接口与其他对象进行通信，但对象并不知道其他的具体实现细节。例如，驾驶员通过车辆的接口（包括方向装置、油门、刹车、换档装置等）来控制车辆，但是驾驶员无需关心这些机械装置的工作原理。通常，具体实现细节隐藏在对象自身的内部。读者一定也能够体会到，即使不知道发动机、传动装置或排气系统的具体工作机制，人们也能够有效地驾驶车辆。

那么，到底何为对象呢？实际上，对象技术是一种打包方案，帮助我们创建有意义的软件单元。我们生活在一个对象的世界上，环顾四周便可体会到这一点。汽车、飞机、人、

动物、建筑、红绿灯、电梯等等，一切都是对象。在面向对象的语言问世之前，所有的编程语言（如 FORTRAN、Pascal、Basic 和 C）都将重点放在行动（动词）上，而不是放在物件或对象（名词）上。程序员生活在一个对象世界中，却不得不以动词为主进行编程。这种思维模式的强制转变，加大了编写程序的难度。

与过程化编程技术相比，对象编程技术显然更加自然，而且可显著提高工作效率，因为采用对象技术，只要设计得当，那么软件实体（称为对象）一经创建，就可极其方便地将其重复用于未来的项目。此外，在采用面向对象编程技术后，用它写出来的程序更容易理解，更加规范而且更容易维护、修改和调试。

1.5 面向对象程序设计技术

面向对象技术以类的形式来描述现实世界中对象的特征及其相互之间的关系。它将数据和处理数据的操作都封装在对象中，对象中的数据和操作紧密地联系在一起。对象通过良好定义的接口来暴露自己的功能，但是通常不会允许其他对象知道自己的内部实现，实现细节被封装在对象的内部。

在 C 及其他过程式编程语言中，编程通常都是面向行动的；而在 C++中，编程通常都是面向对象的。在 C 中，基本编程单位是函数；而在 C++中，基本编程单位是类，通过实例化类得到类的实例，即对象。

读者以后会体验到，一旦软件被打包成类，这些类便可在未来的软件系统中重用。一组相关的类通常打包成可重用的组件。利用对象技术，今后大多数软件都可通过合并一系列“标准化、可互换的零件”构建，这些零件就是所谓的“类”。因此，你所创建的每个类都有可能成为无价的软件资源，你和其他程序员可以用它使未来的软件开发工作更加容易。

封装、继承和多态性是面向对象程序设计技术的三大基石。

1.5.1 封装

封装是面向对象程序设计技术的一个重要基石，其含义就是将对象的数据和处理数据的方法组合成一个相对独立的系统单位，并尽可能隐藏对象的内部实现细节。在 C++中，类的数据成分称为数据成员，而类的函数成分称为成员函数。

毫无疑问的是，我们完全能够很好地操作一台电视机，而不必知道显像管、声音图像接收处理装置和遥控器的内部工作原理。实际上，如果必须理解这些内部机理，操作电视机会变得更加困难，而且电视机也不会这么普及。

事实上，封装具备以下两层含义：第一层含义是将所有的数据成员和成员函数都包含在一个类中，形成一个密不可分的整体；第二层含义是尽可能地隐藏对象的内部实现细节，对外构成一个访问边界（或者说一堵墙），一个类只会曝露有限的接口与外部进行联系。

1.5.2 继承

继承是面向对象技术能够提高软件复用性的重要原因之一。在继承的帮助下，开发人员就可以在现有类型的基础上定义新的类型，这个新类称为派生类（子类），而现有类型称为基类（父类）。派生类继承了基类的所有数据成员和成员函数，并在此基础上增加了新的功能。

例如，在原有手机上加装一个摄像头以后，使之具备摄像拍照功能，就诞生了现在的照相手机。由于照相手机也是手机，因此它具备手机的所有一般特征，从而手机研发人员只需要将精力放在照相手机特有特征的研发工作上。

1.5.3 多态性

在使用继承的方法构造了新的派生类以后，可以采用多态性为每个派生类指定自己特有的行为。这使得基类中的某一个行为在派生类中具有不同的含义。例如，可以定义一个“雇员”类，它具有“计算雇员工资”行为，但是这个行为在执行时并不确定要计算哪一种雇员类型的工资。在“雇员”类的基础上派生一些特殊类，诸如按月发薪的雇员和按日发薪的雇员它们都继承一般类“雇员”，因此也就自动地具备了“计算雇员工资”行为。可以在特殊类中根据实际情况重新定义“计算雇员工资”行为，使之实现计算月薪和日薪的功能。此外，还可以定义基本工资外加业绩提成的雇员类型，以在“计算雇员工资”行为中实现计算基本工资外加业绩提成的功能。

通过组合使用继承和多态性，开发人员就能够轻易地开发出一系列形似但神离的对象。继承体现了对象的“共性”，而多态性则体现了对象的“个性”。

1.6 C++标准库

C++程序由一系列类和函数构成。可自行编写构成一个 C++程序所需的每一个组件。但大多数 C++程序员都直接利用由 C++标准库提供的现成的、丰富的类和函数。因此，要想认识 C++世界，实际只需要学习两个方面的知识。一是学习 C++语言本身；二是学习如何使用 C++标准库中提供的类和函数。

C++标准库提供以下功能：

（1）基本运行时的支持，诸如内存分配和运行时的类型信息等。

（2）C 标准库。

（3）字符串和 I/O 流。

（4）容器（诸如 vector、list 等）和使用容器的算法（诸如遍历、排序等）。

（5）对数值计算的支持。

对于自行创建函数和类，其优点在于：你可确切地知道它们是如何工作的，以便自己能对 C++代码进行全面检查。其缺点在于：在设计、开发和维护新函数和类时，为保障其正确性和高效率地工作，需要花费更多的时间与精力。采用标准库的函数和类，而不是自行编写相应的版本，不仅可以有效地提高程序的性能，因为这些组件经过精心的编写，可保证高效而正确地运行；而且还可以显著改善程序的可移植性，因为在几乎所有的 C++实现方案中，都已包含了这些标准库函数和类。

1.7 C++编程简单示例：输出文本

本节将通过一个非常简单的示例来展示 C++程序的构成。尽管这个示例只是在屏幕上输出两行内容相同的文本，但是它却包含了许多非常重要的知识要点。下面显示了该示例的完整实现代码：