HZ BOOKS
华章教育

普通高等院校计算机课程规划教材

C++程序设计教程

皮德常　编著

本书为教师配有电子教案

机械工业出版社
China Machine Press

普通高等院校计算机课程规划教材

C++程序设计教程

皮德常　编著

机械工业出版社
China Machine Press

C++是一种实用的程序设计语言，是高校学生学习程序设计的一门必修专业课程，同时也是编程人员最广泛使用的工具。学好C++，可以很容易地触类旁通其他语言，如Java和C#等。

本书针对初学者和自学者的特点，在总结过去的教学和实践经验的基础上，编写而成。写作风格别具一格，语言流畅，风趣，恰如其分的举例易于读者理解和掌握C++程序设计，同时，在写作中还特别注重培养学生的独立思考能力。教材结合实例讲解了C++的基本概念和方法，力求将复杂的概念用简洁、通俗、有趣的语言描述，做到了深入浅出、循序渐进，从而使学生能体会学习的快乐，及在快乐中学习。

全书共11章，主要包括C++基本数据类型、流程控制、函数、数组、指针、结构体、文件操作、类的基础部分、类的高级部分、继承、多态、虚函数、异常处理、课程设计要求等。书中列举了数百个可供直接使用的程序示例代码，并给出了运行结果，使学生在学习时更为直观。

本书配有适当的习题，并提供了该书的电子教案，特别适合用作大学计算机专业和非计算机专业的程序设计课程教材，也非常适合那些具有C编程经验，又想转向C++编程的读者阅读。

图书在版编目（CIP）数据

C++程序设计教程 / 皮德常编著. —北京：机械工业出版社，2009.3
（普通高等院校计算机课程规划教材）

ISBN 978-7-111-26247-3

Ⅰ.C…　Ⅱ.皮…　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（2009）第015135号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：姚　蕾
北京市慧美印刷有限公司印刷
2009年3月第1版第1次印刷
184mm×260mm　·　20.25印张
定价：36.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

前　言

C++是一种实用的程序设计语言，是高校学生学习程序设计的一门必修专业课程，同时也是编程人员最广泛使用的语言。学好C++，可以很容易地触类旁通其他语言，如Java和C#等。

本书主要是针对C++初学者和自学者而编写的，结合实例讲解了C++的基本概念和方法，力求将复杂的概念用简洁、通俗的语言描述，做到深入浅出、循序渐进。本书适合用于大学计算机专业和非计算机专业，也可供具有C语言基础的自学者使用。本书的特点如下：

1. 本书主要讲解了C++程序设计的编程方法，它是计算机科学与技术专业学生的编程基础。

2. 本书是作者教学经验的结晶。作者10多年来一直从事程序设计方面的教学和科研工作，主讲过程序设计方面的多门课程，积累了丰富的教学经验。“从实践到理论，再从理论到实践，循序而渐进”是作者教学的心得体会，编写教材也不例外。了解学生的薄弱环节和学习特点，将自己的知识、授课方法和教学经验整理成书，使更多的学生受益，是作者的梦想和追求。

3. 在内容安排上，本书尽量提前讲解文件操作（许多书都是在最后讲解）这部分内容。因为文件是很实用、也是比较难学的一章，所以这种安排也为学生进行课程设计和实验做了铺垫。

4. 在作业安排上，从易到难，环环相扣。作者在教学中发现，许多学生学过C++，却不会编程。因此，本书设计了许多与实际有关的习题，并且它们彼此相关。

5. 强调课程设计。C++课程应该有课程设计，在本书的最后给出了一个课程设计要求，希望学生能独立、认真地完成。这对提高学生的编程能力，巩固学过的知识大有裨益。

6. 力求通俗易懂。编写本书的目的是让学生通过自学或在教师的讲授下，能够运用C++语言的核心要素，进行程序设计。因此，本书围绕着如何进行C++编程展开。为了便于学生的学习，作者力求语言通俗易懂，将复杂的概念采用浅显的语言描述，做到易学、易用、有趣，从而便于读者理解和掌握C++编程思想和方法。

7. 强调程序的可读性。本书中的程序全部采用统一的程序设计风格。例如，类名、函数名和变量名的定义做到“见名知义”，采用缩排格式组织程序代码并配以尽可能多的注释。希望学生能够模仿这种程序设计风格。

8. 包含大量的程序示例，并给出运行结果。凡是程序开头带有程序编号的程序，都是完整的程序，可以直接在计算机上编译运行。

9. 采用醒目的标记来显示知识点。这些标记是注意、警告和思考等，它们穿插在内容中，帮助学生尽快找到重要的信息。

注意：值得关注的地方，也是作者在教学中发现学生容易搞错的知识点。

警告：这是容易混淆的知识点。

思考：提出问题，引导进行思考，培养思考能力。

本书的电子教案采用PowerPoint 2003制作，可以在讲课时用多媒体投影演示，这可部分取代板书。教师不仅可以使用本教案，还可以方便地修改和重新组织其中的内容以适应自己的教学需要。使用本教案可以减少教师备课时编写教案的工作量，以及因板书所耗费的时间和精力，从而提高单位课时内的知识含量。

我们向使用本教材的教师免费提供电子教案。需要本教案的教师可以直接与机械工业出版社联系。

在本书编写的过程中，作者得到了许多同事的帮助，他们是王珊珊、臧冽、张志航、郑洪源、刘学军、陈丹等，他们为作者提出了许多宝贵的意见和建议。作者的研究生马程、张玉、方卓然、张伟、王强和程冉等人，为本书做了大量的程序验证工作。在作者教学的过程中，也得到了许多学生的启发，促使作者在写书的过程中，要有的放矢，避免学生走弯路。

在本书出版之际，我们感谢南京航空航天大学的林钧海教授，他在百忙之中认真审阅了本书，并提出了宝贵的意见。我们按他的意见进行了相应修改。

感谢您选择本书，欢迎您对本书的内容提出批评和修改建议，作者将不胜感激。作者的电子邮件地址：dc.pi@163.com。

作　者

2009年初

目　录

前言

第1章　C++程序设计基础 ……1
1.1　为什么要学习C++程序设计 ……1
1.2　简单的C++程序举例 ……2
1.3　注释方法 ……2
1.4　编程风格 ……3
1.5　C++程序的词法单位 ……3
1.5.1　C++程序中的字符 ……3
1.5.2　标识符 ……3
1.5.3　关键字 ……4
1.6　C++的基本数据类型 ……4
1.7　变量与常量 ……5
1.7.1　变量 ……5
1.7.2　文字常量 ……6
1.7.3　符号常量 ……8
1.7.4　常变量 ……8
1.8　运算符和表达式 ……8
1.8.1　算术运算符和算术表达式 ……8
1.8.2　运算符的优先级和结合性 ……9
1.8.3　赋值运算符和赋值表达式 ……9
1.8.4　自增、自减运算 ……10
1.8.5　关系运算符和关系表达式 ……10
1.8.6　逻辑运算符和逻辑表达式 ……11
1.8.7　位运算和算术表达式 ……12
1.8.8　逗号运算符与逗号表达式 ……14
1.8.9　sizeof 运算符 ……14
1.8.10　C++的运算符、优先级和结合性 …14
1.9　语句 ……15
1.10　类型转换 ……15
1.10.1　赋值时的类型转换 ……15
1.10.2　混合运算时的类型转换 ……17
1.10.3　强制类型转换 ……17
1.11　简单的输出和输入方法 ……18
1.11.1　cout对象和cin对象 ……18
1.11.2　格式化输出 ……20
1.11.3　采用函数成员实现格式化输出 ……23
1.11.4　对函数成员的初步讨论 ……24
1.11.5　指定输入域宽 ……24
1.11.6　读取一行 ……25
1.11.7　读取一个字符 ……25
1.11.8　读取字符时易出错的地方 ……26
1.12　枚举类型 ……26
1.12.1　枚举类型的定义 ……26
1.12.2　枚举类型的变量 ……27
1.12.3　枚举类型的应用 ……27
思考与练习 ……28
第2章　C++的流程控制 ……30
2.1　算法的基本概念和表示方法 ……30
2.1.1　算法的基本概念 ……30
2.1.2　算法的表示 ……30
2.1.3　算法的三种基本结构 ……31
2.2　选择结构程序设计 ……32
2.2.1　基本的if语句 ……32
2.2.2　嵌套的if语句 ……34
2.2.3　条件运算符 ……36
2.3　switch语句 ……36
2.4　循环结构程序设计 ……38
2.4.1　while循环 ……38
2.4.2　do-while循环 ……39
2.4.3　for循环 ……40
2.4.4　循环嵌套 ……42
2.4.5　break语句 ……43
2.4.6　continue语句 ……44
2.4.7　应该少用的goto语句 ……44
2.5　程序设计应用举例 ……44
思考与练习 ……48
第3章　函数 ……51
3.1　函数的定义和调用 ……51
3.1.1　概述 ……51
3.1.2　定义函数 ……51
3.1.3　调用函数 ……52

3.2 函数的声明 ……53
3.3 函数的参数传递和返回值……55
3.3.1 函数参数的传递方式 ……55
3.3.2 函数的返回值 ……56
3.4 局部变量和全局变量 ……57
3.4.1 内存存储区的布局 ……57
3.4.2 局部变量 ……58
3.4.3 全局变量 ……58
3.4.4 局部变量与栈 ……59
3.5 变量的存储类别 ……60
3.5.1 auto修饰的变量 ……60
3.5.2 register修饰的变量 ……60
3.5.3 static修饰的变量 ……61
3.5.4 extern修饰的变量 ……61
3.6 默认参数 ……62
3.7 引用作参数 ……64
3.8 函数重载 ……66
3.9 函数模板 ……68
3.9.1 从函数重载到函数模板 ……68
3.9.2 定义函数模板的方法 ……71
3.9.3 在函数模板中使用多种类型 ……71
3.9.4 函数模板重载 ……71
3.10 内联函数 ……72
3.11 函数的递归调用 ……73
3.12 函数的调试方法 ……78
3.13 编译预处理 ……79
3.13.1 宏定义 ……79
3.13.2 文件包含 ……80
3.13.3 条件编译 ……81
思考与练习 ……82
第4章 数组 ……85
4.1 一维数组 ……85
4.1.1 一维数组的定义和应用 ……85
4.1.2 引用一维数组元素 ……86
4.1.3 数组无越界检查 ……86
4.1.4 数组初始化 ……86
4.2 多维数组 ……87
4.2.1 二维数组的定义 ……88
4.2.2 二维数组的初始化 ……88
4.2.3 引用二维数组元素 ……89
4.3 数组作函数参数 ……90
4.3.1 数组元素作函数参数 ……90
4.3.2 数组名作函数参数 ……90
4.4 常用算法举例 ……91
4.5 字符数组与字符串 ……102
4.5.1 字符数组的定义 ……102
4.5.2 字符数组的初始化 ……102
4.5.3 字符串 ……103
4.5.4 字符数组的输入和输出 ……103
4.6 处理字符和字符串 ……104
4.6.1 处理字符的宏 ……104
4.6.2 处理C风格字符串的函数 ……106
4.6.3 自定义字符串处理的函数……108
4.7 标准C++的string ……110
4.7.1 如何使用string类型 ……110
4.7.2 string对象的比较 ……110
4.7.3 string对象的初始化 ……111
4.7.4 string的函数成员 ……111
4.7.5 string对象应用举例 ……113
思考与练习 ……114
第5章 指针 ……115
5.1 指针的概念 ……115
5.2 指针变量 ……115
5.2.1 定义指针变量 ……115
5.2.2 运算符 & 和 * ……116
5.2.3 引用指针变量 ……117
5.3 指针与数组 ……118
5.3.1 指向数组元素的指针 ……119
5.3.2 指针的运算 ……119
5.3.3 二维数组与指针 ……121
5.4 指针与函数 ……125
5.4.1 基本类型的变量作函数形参 ……125
5.4.2 引用作函数形参 ……126
5.4.3 指针变量作函数形参 ……127
5.4.4 返回指针的函数 ……129
5.4.5 指向函数的指针 ……130
5.5 指针数组与指向指针的指针 ……131
5.5.1 指针数组 ……131
5.5.2 main函数的参数 ……133
5.5.3 指向指针的指针 ……134
5.5.4 再次讨论main 函数的参数 ……134
5.6 内存的动态分配和释放 ……135
5.7 void和const修饰指针变量 ……137
5.7.1 void修饰指针 ……137

5.7.2 const修饰指针 ……138
5.8 对容易混淆概念的总结 ……139
思考与练习 ……141
第6章 结构体与链表 ……145
6.1 抽象数据类型 ……145
6.2 结构体的定义及应用 ……145
6.2.1 定义结构体类型 ……145
6.2.2 定义结构体类型的变量 ……146
6.2.3 初始化结构体类型的变量 ……147
6.2.4 结构体类型变量及其成员的引用 …148
6.2.5 结构体数组与指针 ……150
6.3 用typedef定义类型 ……153
6.4 单向链表 ……154
6.4.1 链表的概念 ……154
6.4.2 带头结点的单向链表常用算法 ……155
思考与练习 ……159
第7章 文件操作 ……161
7.1 文件的基本概念 ……161
7.1.1 文件命名的原则 ……161
7.1.2 使用文件的基本过程 ……161
7.1.3 文件流类型 ……161
7.2 打开文件和关闭文件 ……162
7.2.1 打开文件 ……162
7.2.2 文件的打开模式 ……163
7.2.3 定义流对象时打开文件 ……164
7.2.4 测试文件打开是否成功 ……164
7.2.5 关闭文件 ……165
7.3 采用流操作符读写文件 ……165
7.3.1 采用“<<”写文件 ……165
7.3.2 格式化输出在写文件中的应用 ……167
7.3.3 采用“>>”从文件读数据 ……168
7.3.4 检测文件结束 ……169
7.4 流对象作参数 ……170
7.5 出错检测 ……171
7.6 采用函数成员读/写文件 ……173
7.6.1 采用“>>”读文件的缺陷 ……173
7.6.2 采用函数getline读文件 ……174
7.6.3 采用函数get读文件 ……175
7.6.4 采用函数put写文件 ……176
7.7 多文件操作 ……176
7.8 二进制文件 ……177
7.8.1 二进制文件的操作 ……177
7.8.2 读/写结构体记录 ……179
7.9 随机访问文件 ……181
7.9.1 顺序访问文件的缺陷 ……181
7.9.2 定位函数seekp和seekg ……182
7.9.3 返回位置函数tellp和tellg ……184
7.10 输入/输出二进制文件综合举例 ……185
思考与练习 ……189
第8章 类的基础部分 ……190
8.1 面向过程的程序设计与面向对象程序设计的区别 ……190
8.1.1 面向过程的程序设计的缺陷 ……190
8.1.2 面向对象程序设计的基本思想 ……190
8.2 类的基本概念 ……191
8.3 定义函数成员 ……194
8.4 定义对象 ……194
8.4.1 访问对象的成员 ……194
8.4.2 指向对象的指针 ……195
8.4.3 引入私有成员的原因 ……196
8.5 类的多文件组织 ……197
8.6 私有函数成员的作用 ……199
8.7 内联函数 ……200
8.8 构造函数和析构函数 ……201
8.8.1 构造函数 ……201
8.8.2 析构函数 ……203
8.8.3 带参构造函数 ……204
8.8.4 构造函数应用举例——输入有效的对象 ……206
8.8.5 重载构造函数 ……208
8.8.6 缺省构造函数的表现形式 ……209
8.9 对象数组 ……210
8.10 类的应用举例 ……212
8.11 抽象数组类型 ……216
8.11.1 创建抽象数组类型 ……216
8.11.2 扩充抽象数组类型 ……218
思考与练习 ……222
第9章 类的高级部分 ……224
9.1 静态成员 ……224
9.1.1 静态数据成员 ……224
9.1.2 静态函数成员 ……226
9.2 友元函数 ……228
9.2.1 外部函数作为类的友元 ……229

9.2.2 类的成员函数作为另外一个类的友元 ……229
9.2.3 一个类作为另外一个类的友元 ……233
9.3 对象赋值问题 ……233
9.4 拷贝构造函数 ……235
9.4.1 缺省的拷贝构造函数 ……236
9.4.2 调用拷贝构造函数的情况 ……237
9.4.3 拷贝构造函数中的常参数 ……238
9.5 运算符重载 ……239
9.5.1 重载赋值运算符 ……239
9.5.2 this指针 ……241
9.5.3 重载双目算术运算符 ……243
9.5.4 重载单目算术运算符 ……245
9.5.5 重载关系运算符 ……246
9.5.6 重载流操作符“<<”和“>>” ……247
9.5.7 重载类型转换运算符 ……248
9.5.8 重载“[ ]”操作符 ……253
9.5.9 重载运算符时要注意的问题 ……257
9.5.10 运算符重载综合举例——自定义string类 ……257
9.6 对象组合 ……264
思考与练习 ……265
第10章 继承、多态和虚函数 ……267
10.1 继承 ……267
10.2 保护成员和类的访问 ……271
10.3 构造函数和析构函数 ……274
10.3.1 缺省构造函数和析构函数的调用 ……274
10.3.2 向基类的构造函数传参数 ……275
10.3.3 初始化列表的作用 ……277
10.4 覆盖基类的函数成员 ……278
10.5 虚函数 ……280
10.6 纯虚函数和抽象类 ……283
10.6.1 纯虚函数 ……283
10.6.2 抽象类 ……284
10.6.3 指向基类的指针 ……286
10.7 多重继承 ……287
10.8 多继承 ……288
10.9 类模板 ……291
10.9.1 定义类模板的方法 ……291
10.9.2 定义类模板的对象 ……293
10.9.3 类模板与继承 ……294
思考与练习 ……296
第11章 异常处理 ……299
11.1 异常 ……299
11.1.1 抛出异常 ……299
11.1.2 处理异常 ……300
11.2 基于对象的异常处理 ……301
11.3 捕捉多种类型的异常 ……303
11.4 通过异常对象获取异常信息 ……304
11.5 再次抛出异常 ……306
思考与练习 ……307
课程设计 ……308
参考文献 ……313

第1章 C++程序设计基础

C++是在C的基础上扩充而成的，以其独特的机制广泛地应用在计算机领域。本章主要讲述C++的基本知识，先主要介绍词法单位、数据类型、运算符、变量和常量、表达式和语句等基础知识，然后介绍简单的输入与输出方法。

1.1 为什么要学习C++程序设计

随着计算机软硬件技术的发展，计算机应用规模不断提高，在软件开发语言和工具方面不断地推陈出新，新语言、新工具层出不穷。目前，国内许多高校，无论是计算机专业还是非计算机专业，都开设了C++语言课程，并且将它作为一门专业必修课程。

C++是C的扩充版本。C++对C的扩充是由Bjarne Stroustrup于1980年在美国新泽西州玛瑞惠尔的贝尔实验室提出来的，起初，他把这种语言称为“带类的C”，到1983年才改名为C++。

在计算机刚发明时，人们采用打孔机直接进行机器指令程序设计，当程序长度有几百条指令时，采用这种方法就困难了。后来人们设计了用符号表示机器指令的汇编语言，从而能够处理更大更复杂的程序。到了20世纪60年代出现了结构化程序设计方法（目前的C就采用这种方法），这使得人们能够容易编写较为复杂的程序。但是，一旦程序设计达到一定的程度，即使结构化程序设计方法也变得无法控制，其复杂性超出了人的管理限度。例如，一旦C程序代码达到了25 000～100 000行，系统就变得十分复杂，程序员很难控制，而C++的目的就是为了解决这个问题，其本质就是让程序员理解和管理更大、更复杂的程序。因此，采用支持面向对象的C++是时代发展的需要。

C++吸收了C和Simula67（一个古老的计算机语言）的精髓，它具有C所无法比拟的优越性。C++在维持C原来特长（如效率高和程序灵活）的基础上，借鉴了Simula67的面向对象的思想，将这两种程序设计语言的优点相结合。C++的程序结构清晰、易于扩展、易于维护同时又不失效率。目前，C++已超出了当初设计其的目的，成功地应用到数据库、数据通信等系统，并成功地构造了许多高性能的系统软件。C++与C相比，具有三个重要的特征，从而使其优越于C。

第一个特征是支持抽象数据类型（Abstract Data Type，ADT）。在C++中ADT表现为类，是对对象的抽象，而对象是数据和操作该数据代码的封装体，它提供了对代码和数据的有效保护，可防止程序其他不相关的部分偶然或错误地使用对象的私有部分，这是C所无法实现的。

第二个特征是多态性，即一个接口，多重算法。C++既支持早期联编又支持滞后联编，而C仅支持前者。

最后一个特征是继承性。继承性一方面保证了代码复用，确保了软件的质量；另一方面也支持分类的概念，从而使对象成为一般情况下的具体实例。

这三个特性，我们将在后面的章节给予详细的讲解。

C++对C基本上完全兼容，很多用C写的应用程序都可以在C++环境中使用，因此C++不是一个纯粹的面向对象的程序设计语言，它既支持面向对象的程序设计方法，又支持面向过程的程序设计。

目前许多系统软件，如操作系统、数据库管理系统（DBMS）等都是采用C++所写的，所以从事有关软件开发、自动控制和计算机应用的人员，不掌握C++简直寸步难行。可以说，掌握C++编程已成为许多专业学生的必然选择。

C++有很多版本，国内比较流行的是微软推出的Visual C++。本教程采用的是Visual C++ 6.0。

1.2 简单的C++程序举例

【例1-1】下面通过一个简单程序来分析C++程序的基本构成和特点。为了便于解释程序，我们给程序加了行号，这在写程序时是不需要的。

```
1        #include  <iostream>
2        using namespace std;
3        int  main ( )
4        {
5             int a, b;                                    //定义2个变量
6             cout << "输入变量a和b：" ;
7             cin >> a >> b;                               /*从键盘输入a和b的值*/
8             cout << "a + b = " << a + b << endl;
9             return 0;
10       }
```

将该程序以扩展名为.cpp文件的形式保存，经过编译、链接生成可执行文件，运行后显示如下信息：

```
输入变量a和b：22   88 [Enter]
a + b = 110
```

用户输入22和88并按回车键后，输出它们的和110。

C++程序由编译预处理指令、程序主体和注释组成。

第1行是编译预处理指令，include称为文件包含，它使后面的iostream成为本程序的一部分，这样本程序可以直接使用包含文件中的函数。编译预处理是C++提供的组织程序的一种工具，将在3.13节介绍。

第2行是用using告诉编译器名字空间的名字，上例中的std代表系统提供的标准名字空间，当采用“using namespace std ;”说明以后，程序就可以访问std名字空间中的实体。

3～10行是程序的主体，其中第3行是函数头，4～10行是函数体。一个C++程序由一个或多个函数构成，并且在这些函数中有且仅有一个主函数main，它是程序执行的入口。任何一个函数都由“{ }”中的语句序列描述，如本例中的第4行和第10行，而它们括起来的5～9行是程序执行的语句，并且任何一个语句都以“;”结束。C++程序严格区别字母的大小写。

1.3 注释方法

C++的注释形式有两种，一种是“/* */”格式，这是C语言中的注释风格。因为C++是C的扩充版，所以它支持C的注释方法。在C语言中，注释以“/*”开头，以“*/”结束。编译器将忽略这两个符号之间的所有语句，如例1-1程序中的第7行采用的就是这种注释风格。另一种注释格式是双斜线//，在双斜线之后的部分都会被视为注释，如例1-1程序中的第5行。注释是程序员用来说明程序或解释代码的语句。注释是程序的组成部分，但编译器在编译时忽略它，不构成可执行代码。它也属于编程风格中关键的一环。

许多程序员都会在源程序中，尽可能少地加注释语句。因为编写源代码本身已经够痛苦了。但是，养成加注释的习惯是很有帮助的。编程时可能要花费额外的时间来写注释，但在以后将会节省很多的时间。假设你辛苦数月，编写了一个大约8000～60 000行C++程序，你完成了编码，并且调试成功，交给了客户使用，于是你继续做下一个项目。一年以后需要对程序进行修正。当你打开数万行没有注解的源代码时，你发现有许多函数，已经不知道它们的功能，当初设计的目的是什么都不知道。这是你就会想，要是当初加一些必要的注释就好了，但为时已晚，你目前要做的只能是用较多的时间来理解源程序，或完全重写源程序。

注意：不必为程序的每一行都加注释，也不必为一目了然的代码加注释，只要注解适当的代码，有助于他人理解即可。

C注释方法能跨越多行，便于对多行注释，但对单行注释并不方便。程序员往往把两者结合起来使用：使用C注释方法完成多行注释，使用C++注释方法完成单行注释。

1.4 编程风格

程序员使用标识符、空格、Tab键、空行、标点符号、代码缩进排列和注释等，来安排源代码的方式，就构成了编程风格中重要的组成部分。

当编译器对源程序进行编译时，它会将程序处理为一个长字符串。一条语句不在同一行或者操作符和操作数之间有空格，都不会影响编译。但阅读程序的人很难读懂这种书写不规范的程序。例1-2虽然没有什么语法错误，但却很难读。

【例1-2】一个令人难以理解的程序

```
#include  <iostream>
using namespace std;
int  main ( ){int a, b; cout << "输入变量a和b: " ; cin
>> a >> b; cout << "a + b = " << a + b << endl; return 0; }
```

上面的程序虽然并没有违反C++的语法规则，但却难以阅读。较为理想编程风格应是在编程时使用空格、空行和代码缩进排列，从而使别人能很快读懂你的程序。

警告：尽管编程风格的自由度很大，但还是应该遵循程序设计的国际常规。这样，其他程序员才会很容易读懂你的程序。建议你模仿我们的编程风格。当然，你也可以参考国际知名公司的风格，如微软、IBM和SUN等，其风格都很好。不同的公司其风格不尽相同，可以参考其提供的样式文件，这样对培养编程风格大有裨益。

1.5 C++程序的词法单位

本节介绍C++程序使用的字符、关键字、标识符和标点符号。

1.5.1 C++程序中的字符

标准的C++程序采用0x00～0x7F范围内定义的ASCII码所表示西文字符作为程序的基本字符单位，主要包括26个小写英文字母、26个大写英文字母、10个阿拉伯数字和其他一些符号，如+、－、*、/等，其中每个ASCII码字符占用1个字节。

1.5.2 标识符

标识符是程序员自己定义的“单词”，标准C++的标识符由字母、下划线和数字组成，且第一个字符不能为数字，长度一般不超过32个，文件名只识别前8个字符。标识符大小写敏感。同一个单词的不同大小写被编译器看做是不同的标识符。在实际使用时应尽量采用有意义的单词作标识符，做到见名知意。

虽然C++编译器允许用户定义的标识符以下划线开始，但系统定义的内部符号以下划线或双下划线开始，所以自定义的标识符不提倡以下划线开始。

下面均是合法的并且做到见名知意的标识符：

```
studentName、StudentName、name_of_student、salary
```

通过变量名studentName，可以很容易地看出该变量的意义。这种编程风格使得程序易于理解和维护，因为一个大的系统通常由数万行源代码构成，程序的可读性十分重要。

定义标识符应尽量选择有含义的英文单词，如下面的变量命名就毫无意义：

```
abc、xyzw、a123、x888888
```

标识符是区分大小写的，变量studentName和StudentName就是不同的变量。之所以将变量studentName中的“N”大写，是为了增强程序的可读性。例如，studentname是由清一色的小写字母构成的变量名，使人不易读懂，因此这不是好的变量名。

总之，选择标识符时，要尽可能做到“见名知义”，选择有含义的单词符号作标识符，使别人（包括你本人）容易读懂你的程序。

下面是非法的标识符：

```
8abc                                    // 不能以数字开头定义标识符
Student  Name                           // 标识符中间不能有空格
$bill                                   // 不能以$开头定义标识符
```

1.5.3 关键字

关键字又称保留字，是系统定义的一些特殊标识符，它们具有特定含义，不允许程序员将它们挪作他用，如作为一般标识符使用。C++中常用的部分关键字见表1-1。

表1-1 C++常用关键字

类 型	关键字
数据类型说明符和修饰符	bool、char、class、const、double、enum、float、int、long、short、signed、struct、union、unsigned、void、volatile
存储类型说明符	auto、extern、inline、register、static
访问说明符	friend、private、protected、public
语句	break、case、catch、continue、default、do、else、finally、for、goto、if、return、switch、throw、try、while
运算符和常量	delete、false、new、sizeof、true
其他	asm、explicit、namespace、operator、template、this、typedef、typename、using、virtual

一些不常用的关键字还有bad_cast、bad_typeid、const_cast、dynamic_cast、except、mutable、reinterpret_cast、static_cast、type_info、typeid和wchar_t。我们会在后面的章节中逐步引入并介绍常用关键字的含义，其他内容读者可参考有关手册，如“C++大全”。

1.6 C++的基本数据类型

从程序设计语言原理的角度讲，C++是一种强类型的语言，必须严格遵循“先定义后使用”的原则。读者可以在后续的学习中慢慢品味该原则。

C++的数据类型分为两大类：基本数据类型和导出数据类型。

基本数据类型也称为是C++预定义的类型或内置数据类型，包括字符型（char）、整型（int）、单精度实型（float）、双精度实型（double）、布尔型（bool）和空类型（void）。这些预定义的类型，不仅定义了数据类型，还定义了常用的操作。

导出数据类型是由基本数据类型构造出来的数据类型，包括数组、指针、引用、结构体、共用体、枚举和类等。

字符型用来存放一个字符的ASCII码值，可以将其看成一个8位二进制码的整数。如大写字母A的ASCII码是65，那么在一个字符变量的空间中存放的就是65。宽字符类型(wchar_t)也称为双字节字符，往往采用Unicode编码格式，该标准中的所有字符都是双字节的，这样可以统一处理西文、

中文、阿拉伯文和其他语言的符号。但宽字符类型不属于基本类型，限于篇幅，本书不做介绍。

整型用来存放一个整数，一般占用4个字节，无符号数采用原码的形式表示，而有符号数采用补码表示。

在类型标识符char和int之前加上修饰词后，可以得到其他类型的整型数。这些修饰词有signed（有符号的）、unsigned（无符号的）、long（长的）和short（短的）。

实型用来存放实型数据，C++提供了float和double两种实型类型，因占用的字节数不同，其表示的数据范围也不同。

逻辑型也称布尔型。为了纪念英国的数学家乔治·布尔（George Boolean），就在程序设计语言中引入了“布尔”类型。布尔型用来处理逻辑量，取值只有true（真）和false（假）两个，占1个字节，将非0值解释为true，将0值解释为false。

注意：从程序程序设计语言原理的角度讲，布尔变量的取值只能是真或假，但C++中对它的定义很不严格，将0看做false，将非0看做true。这是为了向下兼容，即兼容它的子集C语言，因为C就是这样定义的。

空类型（void）用来定义指针，或用来说明函数的返回值类型，将在5.7.1节讲解。

C++允许在整数、字符或浮点类型前面加上修饰符，如short（短类型）、long（长类型）、signed（有符号类型）和unsigned（无符号类型）。表1-2列出了C++中基本数据类型及其变量的取值范围。

表1-2 C++中所有的基本数据类型

类 型	名 称	占用字节数	取值范围
bool	布尔型	1	true, false
[signed] char	有符号字符型	1	−128 ~ 127
unsigned char	无符号字符型	1	0 ~ 255
[signed] short [int]	有符号短整型	2	−32768 ~ 32767
unsigned short [int]	无符号短整型	2	0 ~ 65535
[signed] int 或 signed	有符号整型	4	$-2^{31} \sim (2^{31}-1)$
unsigned [int]	无符号整型	4	$0 \sim (2^{32}-1)$
[signed] long [int]	有符号长整型	4	$-2^{31} \sim (2^{31}-1)$
unsigned long [int]	无符号长整型	4	$0 \sim (2^{32}-1)$
float	实型	4	$-10^{38} \sim 10^{38}$
double	双精度实型	8	$-10^{308} \sim 10^{308}$
long double	长双精度实型	8	$-10^{308} \sim 10^{308}$

注意1：可选项问题。例如，[signed] char 中的“[signed]”代表可选项，即它等价于signed char或char。

注意2：有些教材将void类型作为基本数据类型，我们不赞成这种观点。因为void在C++中仅能用来修饰指针和函数。后面的章节会讲解它的用途。

1.7 变量与常量

1.7.1 变量

C++作为一种强类型的程序设计语言，在使用变量前应当先说明类型，然后再定义变量，以便于编译器分配内存空间，以及对程序中的数据类型和运算等进行常见错误检查，以提高程序的编译和运行效率。

在程序运行中，值可变的量称为变量。变量名必须用标识符，即名字来标识。变量根据其取

值范围的不同可分为字符型、整型、实型变量等。在程序运行时刻，系统将会给每个变量分配一段连续的内存单元，用来存放变量的值。

变量有三个要素：变量名、变量的内存空间和变量的值。

1. 定义变量

定义变量的一般格式为：

```
[<存储类别>] <数据类型> <变量名1>[, <变量名2>,  ......<变量名n >];
```

其中，“< >”括起来的是必选项。存储类别将在3.5节介绍。下面定义几个变量：

```
bool  b;                                  // 定义1个布尔型变量 b
char gender, ch;                          // 定义2个字符型变量gender和ch
int a, b ;                                // 定义2个整型变量a和b
double dx ;                               // 定义1个双精度实型变量dx
float f ;                                 // 定义1个单精度实型变量f
unsigned u ;                              // 定义1个无符号整型变量u
```

变量必须先定义后使用，原因如下：

1）变量定义后就具有变量的前两个要素，即具有了变量名和变量的类型。编译系统根据类型给变量分配存储空间，并建立变量名和其存储空间的对应关系，于是可以通过变量名给变量的存储空间赋值或读取该存储空间中变量的值。

2）变量确定类型后，编译器可以对变量参与的运算做合法性检查。

2. 变量赋值

当使用变量时，变量必须有一个确定的值，给变量赋值的方法有两种：

1）变量定义后，用赋值语句赋初值。例如：

```
int   a;
char  gender;
a = -12+100;
gender = 'M';
```

此处“=”是赋值运算符，表示将赋值号右边的运算结果放入赋值号左边变量对应的存储空间中。

2）在定义变量的同时，直接对变量赋初值，称为变量的初始化。例如：

```
int   a = 12;                             // 采用12初始化变量a
char gender='M';                          // 采用'M'初始化变量gender
```

注意：初学者往往有一个疑问，如果变量不赋值，其值是什么，例如：

```
int   a;
```

在不赋值的情况下，a的值可能是0，也可能是一个不确定的值。既然存在不确定性，就可能导致程序出错，所以我们要对变量赋一个确定的值。

1.7.2 文字常量

在程序运行过程中，值不能被改变的量称为常量。其中文字常量是指程序中直接使用的常量。文字常量存储在代码区，而不是数据区，对它的访问不是通过地址进行的。根据取值方式和表达方法的不同，文字常量分为整型、实型、字符型和字符串型常量。

1. 整型常量

1）十进制整数，如789、−456。

2）八进制整数，如0789、−026。

八进制整数以0（零）开头，在数值中可以出现数字符号0~7。

3）十六进制整数，如0x789、−0xAB。

十六进制整数以0x（零x）或0X开头，在数值中可以出现数字符号0~9、A~F（或小写的a~f）。

4）长整型整数与无符号型整数。

长整型整数，如12L、0234L、−0xABL、12l、0234l、−0xABl。

无符号型整数，如12U、0234U、0xABU、12u、0234u、0xABu。

在一个常数后加L（英文大写）或l（英文小写）表示该常数是长整型的整数；在一个常数后加U（英文大写）或u（英文小写）表示该常数是无符号型的整数。

2. 实型常量

实型常量在内存中以浮点形式存放，在程序中书写时，均为十进制数，无数制区分。两种书写形式分别为：

1）小数形式：必须写出小数点，如1.65、1.、.123均是合法的实型常量。

2）指数形式：也称为科学表示法形式，如1.23×10^5和1.23×10^{-5}在程序中表示为1.23e5或1.23e−5。小写的e也可以写成大写的E，e或E前必须有数字，e或E后必须是整型数；1000应写成1e3，而不能写成e3。

3. 字符型常量

用单引号括起来的一个字符称为字符型常量（简称字符常量），如'a'、'A'、'?'、'#'。在内存中对应存放字符的ASCII码值，其数据类型为char。用这种方式只能表示键盘上的可输入字符。有些控制字符，如回车符、换行符、制表符、响铃、退格等，就不能用这种方式表示。ASCII码值为10的字符表示换行，就无法用前述方式表示。

针对这些无法直接表示的或具有特殊含义的字符，C++提供了另外一种称为转义序列的表示方法，即“转义”字符。转义字符是以反斜杠“\”开始的特殊字符常量。如'n' 表示字母n，而 '\n' 仅代表一个字符，即换行符，跟在“\”后的字母n的含义发生了改变，所以称为转义字符。在表1-3中列出C++中定义的转义字符及其含义。

表1-3 C++中定义的转义字符及其含义

转义字符	名 称	功能或用途
\a	响铃	用于输出响铃
\b	退格（Backspace键）	输出时回退一个字符位置
\f	换页	用于输出
\n	换行符	用于输出，移至下一行行首
\r	回车符	用于输出，回退至本行行首
\t	水平制表符（Tab键）	用于输出，跳至下一制表起始位置
\v	纵向制表符	用于制表
\\	反斜杠字符	用于表示一个反斜杠字符
\'	单引号	用于表示一个单引号字符
\"	双引号	用于表示一个双引号字符
\nnn	nnn是ASCII码的八进制值，最多三位	用八进制ASCII码表示字符
\xhh	hh是ASCII码的十六进制值，最多两位	用十六进制ASCII码表示字符

表1-3中最后2行是所有字符的通用表示形式，即用反斜杠加ASCII码表示，它可以表示任一字符。如 '\n' 表示控制字符“换行”，它的ASCII码是十进制数10，10的八进制和十六进制表示分别是12和a，因此 '\n' 也可以表示成 '\12' 和 '\xa'。字母A的ASCII码是十进制数65，它的八进制数和十六进制表示分别是101和41，所以字母A也可以写成 'A'、'\101'或'\x41'的形式。

4. 字符串常量

字符串常量是用双引号括起来的字符序列，如"123"、"I am a Chinese."、"a"。字符串常量在内存中是按顺序逐个存储串中字符的ASCII码，并在末尾加一个结尾标志'\0'字符，称为串结束符。'\0' 表示ASCII码值为0的字符，即ASCII码表中第一个字符，也称为“空字符”，其值为0。字符串的长度是串中'\0'字符之前的所有字符的个数，因此，字符串常量实际占用的字节数是串长加1。

注意：字符串常量"a"和字符常量'a'不同。字符常量'a'在内存中占1个字节，而字符串常量"a"在内存中占有两个字节。

思考：如果在程序中出现了下面3个串，它们分别代表什么？

```
"I\'ve done"    "dog \'s toy"    "\"Love\""
```

1.7.3 符号常量

我们可以在程序中直接书写常量，但有时会遇到一些麻烦。如在程序中需要多次使用3.1415926，这样编写程序时可能出现数字书写错误，同时如果精度上需要变化，如将3.1415926改为3.14，就需要在程序中进行多处修改。

C++提供了一种称为符号常量的机制，以避免上述麻烦。即用一个标识符代表一个常量，称为符号常量。只要在程序的开头定义一个符号常量，令其代表一个数值，在程序的后面使用该符号常量即可。符号常量的定义形式为：

```
#define   PI        3.1415926
```

定义符号常量的好处是，如果在程序中多处使用了同一个常量，当需要对该常量修改时，只需在定义处修改即可，而不需要修改程序中的多处。给符号常量取有意义的名字有利于提高程序的可读性，另外，一般用大写字母给符号常量命名。

1.7.4 常变量

const是constant的缩写，是“恒定不变”的意思。被const修饰的变量或对象都将受到强制保护，可以预防意外的改变。在C++中采用const定义一种称为常变量的量，其定义形式为：

```
const double pi=3.14159;
```

pi具有变量的三个要素，即变量名、存储空间和值。但必须在定义时赋初值，且它的值在程序的运行过程中不能被改变。常变量存储在数据区，并且可以按地址访问，但在初始化后不允许再次被赋值。编译器在编译时会对常变量进行类型检查，这比符号常量要好。因此在C++编程中，完全可以用const常量取代1.7.3节介绍的符号常量。

1.8 运算符和表达式

对变量或常量进行运算或处理的符号称为运算符，参与运算的对象称为操作数。

数据处理是通过运算实现的。为表示一个计算过程，需要使用表达式。表达式是由运算符、运算量构成的一个计算序列。在C++有很多运算符，如算术运算符（+、−、*、/、%）、关系运算符（>、>=、<、<=、==、!=）等。

1.8.1 算术运算符和算术表达式

C++的五个算术运算符是+（加）、−（减）、*（乘）、/（除）、%（求余），它们是二元运算符，其中 +（正号）和−（负号）又可用作一元运算符。

运算符总是和操作数封装在一起，相同的运算符对不同类型的操作数执行运算，其结果是有差异的，称为运算符重载。对于除运算符“/”，当两个运算量均为整数时为整除，如5/2结果为2。