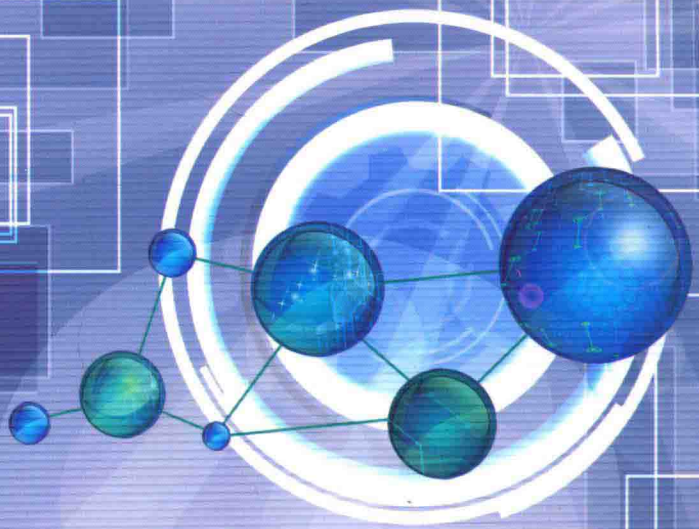
普通高等教育仪器类“十三五”规划教材



C++程序设计

徐耀松　郭　磊　尹玉萍　主　编
王丹丹　屠乃威　马玉芳　副主编

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

普通高等教育仪器类“十三五”规划教材

C++程序设计

主　编　徐耀松　郭　磊　尹玉萍

副主编　王丹丹　屠乃威　马玉芳

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

C++具有高效实用的特点，可以进行面向过程和面向对象程序设计，目前是应用最广泛的编程语言之一。本书介绍了C++程序设计的基本内容及应用方法，包括数据类型与表达式、程序设计方法、函数、数组、指针、类和对象、重载、继承与派生等。内容上循序渐进，突出专业知识的综合应用。本书结构合理、内容翔实、实例丰富，具有较高的应用性。

本书适合C++程序设计的初学者学习使用，也可作为高等院校测控技术与仪器、自动化、电子信息工程、机电一体化和计算机应用等专业的教材。

图书在版编目(CIP)数据

C++程序设计 / 徐耀松，郭磊，尹玉萍主编. —北京：电子工业出版社，2017.6
普通高等教育仪器类“十三五”规划教材
ISBN 978-7-121-31615-9

Ⅰ. ①C… Ⅱ. ①徐… ②郭… ③尹… Ⅲ. ①C语言－程序设计－高等学校－教材 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字（2017）第107702号

策划编辑：赵玉山
责任编辑：刘真平
印　　刷：三河市良远印务有限公司
装　　订：三河市良远印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编　100036
开　　本：787×1 092　1/16　印张：20.5　字数：524.8千字
版　　次：2017年6月第1版
印　　次：2017年6月第1次印刷
定　　价：49.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

本书咨询联系方式：zhaoys@phei.com.cn。

普通高等教育仪器类“十三五”规划教材

编委会

前　言

本书循序渐进地介绍了C++程序设计的相关概念、方法。内容上突出工程特色，以工程教育为理念，围绕培养应用创新型工程人才这一目标，着重学生的独立研究能力、动手能力和解决实际问题能力的培养，将测控技术与仪器专业工程人才培养模式和教学内容的改革成果体现在教材中，通过科学规范的工程人才教材建设促进专业建设和工程人才培养质量的提高。

全书共12章。第1章介绍了进行C++语言程序设计的预备知识，包括数制、数据的表示方法，介绍了C++语言开发工具；第2章介绍了C++语言程序设计中的数据类型与表达式；第3章介绍了基于过程的程序设计方法，主要包括输入/输出流、关系运算与逻辑运算、选择结构、循环结构等内容；第4章介绍了函数与预处理方法，介绍了常见的函数形式与用法；第5章介绍了数组的概念、创建方法及使用方法；第6章介绍了指针的概念，并对指针的应用进行详细解析；第7章介绍了自定义数据类型，包括结构体类型、链表、共用体等内容；第8章介绍了类，这是面向对象编程中最基本的概念，详细介绍了类的定义及使用方法、构造函数与析构函数及对象的使用方法；第9章介绍了运算符重载的定义、方法、规则等内容；第10章介绍了继承和派生的概念及工作方式；第11章介绍了多态性的概念、虚函数的定义与使用方法；第12章介绍了输入/输出流的常用函数及使用方法。

本书第1、8章由徐耀松执笔；第2、3、5、9、10章由郭磊执笔；第4、6、7章由尹玉萍执笔；第11章由王丹丹、屠乃威、马玉芳、谢国民共同执笔；第12章由郭磊和徐耀松共同执笔。全书的写作思路由付华教授提出，由付华和徐耀松统稿。此外，李猛、任仁、陶艳风、代巍、汤月、司南楠、陈东、谢鸿、郭玉雯、于田、孟繁东、曹坦坦等也参加了本书的编写。在此，向对本书的完成给予了热情帮助的同行们表示感谢。

由于作者的水平有限，加上时间仓促，书中的错误和不妥之处，敬请读者批评指正。

编　者

2017年2月

目　录

第 1 章　预备知识 …… (1)
1.1　C++简介 …… (1)
1.2　计算机数据表示方法 …… (2)
1.2.1　二进制、八进制、十六进制 …… (3)
1.2.2　表示数据的字节和位 …… (5)
1.2.3　内存 …… (5)
1.3　C++开发工具 …… (7)
第 2 章　C++的数据类型 …… (13)
2.1　C++的数据类型 …… (13)
2.2　常量 …… (14)
2.2.1　数值常量 …… (15)
2.2.2　字符常量和字符串常量 …… (15)
2.2.3　布尔常量 …… (17)
2.2.4　符号常量 …… (17)
2.3　变量 …… (18)
2.3.1　变量名规则 …… (18)
2.3.2　定义变量 …… (18)
2.3.3　对变量赋初值 …… (18)
2.3.4　常变量 …… (19)
2.4　C++的运算符 …… (19)
2.5　算术运算符与算术表达式 …… (20)
2.5.1　基本的算术运算符 …… (20)
2.5.2　算术表达式和运算符的优先级与结合性 …… (20)
2.5.3　表达式中各类数值型数据间的混合运算 …… (21)
2.5.4　自增（++）和自减（--）运算符 …… (21)
2.6　赋值运算符和赋值表达式 …… (22)
2.6.1　赋值运算符和赋值表达式概述 …… (22)
2.6.2　赋值过程中的类型转换 …… (22)
2.6.3　复合赋值运算符 …… (22)
2.7　逗号运算符和逗号表达式 …… (23)
2.8　强制类型转换运算符 …… (23)
思考与练习 …… (24)
第 3 章　基于过程的程序设计 …… (26)
3.1　基于过程的程序设计和算法 …… (26)
3.1.1　算法的概念 …… (26)

3.1.2 算法的表示 …… (27)
3.2 C++的程序结构和 C++语句 …… (28)
3.3 C++的输入与输出 …… (29)
3.3.1 输入流与输出流的基本操作 …… (29)
3.3.2 在标准输入流与输出流中使用控制符 …… (31)
3.3.3 用 getchar 和 putchar 函数进行字符的输入和输出 …… (33)
3.3.4 用 scanf 和 printf 函数进行输入和输出 …… (34)
3.4 编写顺序结构的程序 …… (35)
3.5 关系运算和逻辑运算 …… (36)
3.5.1 关系运算和关系表达式 …… (36)
3.5.2 逻辑常量与逻辑变量 …… (37)
3.5.3 逻辑运算和逻辑表达式 …… (38)
3.6 选择结构和 if 语句 …… (39)
3.6.1 if 语句的形式 …… (39)
3.6.2 if 语句的嵌套 …… (42)
3.6.3 条件运算符和条件表达式 …… (43)
3.6.4 多分支选择结构与 switch 语句 …… (44)
3.7 循环结构和循环语句 …… (46)
3.7.1 用 while 语句构成循环 …… (46)
3.7.2 用 do…while 语句构成循环 …… (47)
3.7.3 用 for 语句构成循环 …… (49)
3.7.4 循环嵌套 …… (50)
3.7.5 break 语句和 continue 语句 …… (51)
3.7.6 循环结构程序设计举例 …… (52)
思考与练习 …… (55)
第 4 章 函数与预处理 …… (57)
4.1 函数概述 …… (57)
4.2 函数定义与函数声明 …… (59)
4.2.1 定义无参函数的一般形式 …… (59)
4.2.2 定义有参函数的一般形式 …… (59)
4.2.3 函数声明 …… (60)
4.3 函数的调用 …… (62)
4.3.1 函数调用的一般形式 …… (62)
4.3.2 函数调用的方式 …… (64)
4.3.3 函数的返回值 …… (64)
4.3.4 函数的值传递方式 …… (65)
4.4 内置函数 …… (66)
4.5 函数的重载 …… (67)
4.6 有默认参数的函数 …… (68)
4.7 函数模板 …… (69)
4.8 函数的嵌套调用 …… (70)

4.9 函数的递归调用 …… (71)
4.10 局部变量和全局变量 …… (76)
4.10.1 局部变量 …… (76)
4.10.2 全局变量 …… (77)
4.11 变量的存储类别 …… (79)
4.11.1 动态存储方式与静态存储方式 …… (79)
4.11.2 自动变量 …… (79)
4.11.3 用 static 声明静态局部变量 …… (80)
4.11.4 用 register 声明寄存器变量 …… (81)
4.11.5 用 extern 声明外部变量 …… (81)
4.12 内部函数和外部函数 …… (83)
4.12.1 内部函数 …… (83)
4.12.2 外部函数 …… (84)
4.13 预处理命令 …… (85)
4.13.1 文件包含 …… (85)
4.13.2 条件编译 …… (86)
4.13.3 宏定义 …… (86)
4.13.4 关于 C++标准库 …… (87)
思考与练习 …… (88)
第 5 章 数组 …… (90)
5.1 数组的概念 …… (90)
5.2 一维数组的定义和引用 …… (91)
5.2.1 一维数组的定义 …… (91)
5.2.2 一维数组元素的引用 …… (92)
5.2.3 一维数组的初始化 …… (93)
5.3 二维数组的定义和引用 …… (97)
5.3.1 二维数组定义的一般格式 …… (97)
5.3.2 二维数组元素的引用 …… (97)
5.3.3 二维数组的初始化 …… (98)
5.3.4 二维数组应用举例 …… (98)
5.4 用数组名做函数参数 …… (100)
5.5 字符数组 …… (102)
5.5.1 字符数组的定义和初始化 …… (103)
5.5.2 字符串处理函数 …… (105)
5.5.3 字符数组应用举例 …… (107)
5.6 C++处理字符串的方法——字符串类与字符串变量 …… (108)
思考与练习 …… (111)
第 6 章 指针 …… (112)
6.1 什么是指针 …… (112)
6.2 变量与指针 …… (113)
6.2.1 定义指针变量 …… (114)

6.2.2 指针变量赋值 …… (115)
6.2.3 引用指针变量 …… (116)
6.2.4 指针作为函数参数 …… (116)
6.3 数组与指针 …… (119)
6.3.1 指向数组元素的指针变量 …… (119)
6.3.2 指针的运算 …… (119)
6.3.3 通过指针引用数组元素 …… (120)
6.3.4 用数组名做函数参数 …… (122)
6.4 字符串与指针 …… (126)
6.4.1 字符串的表示方法 …… (126)
6.4.2 字符指针做函数参数 …… (128)
6.4.3 字符指针与字符数组的区别 …… (128)
6.5 函数与指针 …… (129)
6.5.1 函数的指针和指向函数的指针变量 …… (129)
6.5.2 返回指针值的函数 …… (130)
6.6 指针数组和指向指针的指针 …… (130)
6.6.1 指针数组的概念 …… (130)
6.6.2 指向指针的指针 …… (131)
6.7 指针运算小结 …… (132)
6.8 引用 …… (133)
6.8.1 什么是变量的引用 …… (133)
6.8.2 引用作为函数参数 …… (134)
思考与练习 …… (135)
第 7 章 自定义数据类型 …… (137)
7.1 结构体类型 …… (137)
7.1.1 结构体类型的定义 …… (137)
7.1.2 结构体变量的定义 …… (139)
7.1.3 结构体变量的初始化 …… (141)
7.1.4 结构体变量的引用 …… (142)
7.1.5 结构体数组 …… (143)
7.1.6 指向结构体变量的指针 …… (145)
7.1.7 结构体数据做函数参数 …… (148)
7.1.8 动态内存分配 …… (150)
7.2 共用体 …… (152)
7.2.1 共用体的定义 …… (152)
7.2.2 共用体变量的引用 …… (153)
7.2.3 共用体的特点 …… (154)
7.2.4 共用体变量的应用 …… (154)
7.3 枚举类型 …… (156)
7.4 用 typedef 定义类型 …… (156)
思考与练习 …… (157)

第 8 章　类和对象 …… (159)
8.1　类 …… (160)
8.1.1　类的定义 …… (160)
8.1.2　类成员的可见性 …… (162)
8.1.3　类的数据成员 …… (164)
8.1.4　类的成员函数 …… (165)
8.1.5　类与结构的区别 …… (170)
8.2　对象 …… (170)
8.2.1　对象的创建 …… (170)
8.2.2　访问对象的成员 …… (171)
8.2.3　类与对象的关系 …… (173)
8.3　this 指针 …… (174)
8.4　构造函数 …… (175)
8.5　析构函数 …… (176)
8.6　调用构造函数和析构函数的顺序 …… (179)
8.6.1　实例 1 …… (179)
8.6.2　实例 2 …… (181)
8.7　对象数组 …… (182)
8.8　对象指针 …… (186)
8.8.1　对象指针和对象引用 …… (186)
8.8.2　对象指针和对象引用做函数参数 …… (188)
8.9　共用数据的保护 …… (190)
8.9.1　常对象 …… (190)
8.9.2　常成员函数 …… (191)
8.10　对象的动态建立和释放 …… (192)
8.11　对象的赋值和复制 …… (195)
8.11.1　对象的赋值 …… (195)
8.11.2　对象的复制 …… (197)
8.12　静态成员 …… (201)
8.12.1　静态数据成员 …… (201)
8.12.2　静态函数成员 …… (202)
8.13　友元 …… (203)
8.13.1　问题的提出 …… (203)
8.13.2　友元函数 …… (204)
8.13.3　友元类 …… (205)
8.14　类模板 …… (206)
思考与练习 …… (212)
第 9 章　运算符重载 …… (214)
9.1　什么是运算符重载 …… (214)
9.2　运算符重载的方法 …… (215)
9.3　C++运算符重载的规则 …… (218)

9.4 运算符重载函数作为类成员函数和友元函数 …… (218)
9.5 重载双目运算符 …… (220)
9.6 重载单目运算符 …… (222)
9.7 重载流插入运算符和流提取运算符 …… (225)
9.8 不同类型数据间的转换 …… (227)
思考与练习 …… (229)
第 10 章 继承与派生 …… (230)
10.1 继承与派生的概念 …… (231)
10.2 派生类的声明方式 …… (232)
10.3 派生类的构成 …… (233)
10.4 派生类成员的访问属性 …… (235)
10.4.1 公有继承 …… (235)
10.4.2 私有继承 …… (237)
10.4.3 保护成员和保护继承 …… (238)
10.4.4 多级派生时的访问属性 …… (240)
10.5 派生类的构造函数和析构函数 …… (243)
10.5.1 基类构造函数不包括参数 …… (243)
10.5.2 基类构造函数包括参数 …… (244)
10.5.3 有子对象的派生类构造函数 …… (246)
10.5.4 多层派生时的构造函数 …… (248)
10.6 多重继承 …… (251)
10.6.1 声明多重继承 …… (251)
10.6.2 多重继承派生类的构造函数 …… (253)
10.6.3 多重继承引起的二义性问题 …… (255)
10.6.4 虚基类 …… (258)
10.7 基类与派生类的转换 …… (261)
10.8 继承与组合 …… (264)
思考与练习 …… (265)
第 11 章 多态性与虚函数 …… (267)
11.1 多态性的概念 …… (267)
11.2 基类对象的指针指向派生类对象 …… (271)
11.3 虚函数 …… (272)
11.3.1 虚函数的定义 …… (272)
11.3.2 虚函数的使用方法 …… (274)
11.3.3 虚函数与实函数的区别 …… (275)
11.3.4 在构造函数和析构函数中调用虚函数 …… (278)
11.3.5 虚析构函数 …… (279)
11.4 纯虚函数与抽象类 …… (282)
11.4.1 纯虚函数 …… (282)
11.4.2 抽象类 …… (284)
思考与练习 …… (285)

第 12 章　输入/输出流 …… (288)
12.1　C++的输入/输出 …… (288)
12.1.1　输入/输出的含义 …… (288)
12.1.2　流与标准库 …… (289)
12.2　标准输出流 …… (290)
12.2.1　cout、cerr 和 clog 流 …… (290)
12.2.2　格式输出 …… (292)
12.2.3　用流成员函数 put 输出字符串 …… (295)
12.3　标准输入流 …… (296)
12.3.1　cin 流 …… (296)
12.3.2　用于字符输入的流成员函数 …… (297)
12.4　文件操作与文件流 …… (299)
12.4.1　文件流 …… (299)
12.4.2　打开和关闭文件 …… (299)
12.4.3　对 ASCII 文件操作 …… (301)
12.4.4　二进制文件的读写操作 …… (303)
12.4.5　随机访问二进制文件 …… (306)
12.5　字符串流 …… (307)
思考与练习 …… (309)
附录 A　ASCII 码表 …… (310)
参考文献 …… (314)

第1章

预备知识

本章知识点：

- C++编程语言
- 计算机数据表示方法
- C++开发工具

基本要求：

- 了解C++编程语言的发展历程
- 掌握计算机表示数据的方法
- 理解C++开发工具的使用方法

能力培养目标：

通过本章的学习，掌握计算机中数据表示方法以及进行C++编程语言开发的工具。

1.1 C++简介

C++编程语言是在C语言的基础上发展而来的。C语言是一种通用的、过程式的编程语言，广泛用于系统与应用软件的开发，具有高效、灵活、功能丰富、表达力强和较高的移植性等特点，在程序员中备受青睐，是最近25年使用最为广泛的编程语言。C语言是在由UNIX的研制者丹尼斯·里奇（Dennis Ritchie）和肯·汤普逊（Ken Thompson）于1970年所研制出的B语言的基础上发展和完善起来的。目前，C语言编译器普遍存在于各种不同的操作系统中，如UNIX、MS-DOS、Microsoft Windows及Linux等。但是随着软件规模的增大，用C语言编写程序就显得吃力了。C语言是结构化和模块化的语言，它是基于过程的，在处理较小规模的程序时，程序员用C语言还比较得心应手。但是问题比较复杂、程序的规模比较大时，结构化程序设计方法就显出它的不足。C程序的设计者必须细致地设计程序中的每一个细节，准确地考虑到程序运行时每一时刻发生的事情。这对程序员的要求是比较高的，如果面对的是一个复杂问题，程序员往往感到力不从心。当初提出结构化程序设计方法的目的是解决软件设计危机，但这个目标并未完全实现。

为了解决软件危机，在20世纪80年代提出了面向对象的程序设计（Object Oriented Programming，OOP），需要设计出能支持面向对象的程序设计方法的新的语言。在实践中，由于C语言是深入人心、使用广泛、面对程序设计方法的革命，最好的办法不是另外发明一种新的语言去代替它，而是在它的原有基础上发展。在这种形势下，C++应运而生。

C++这个词通常被读作“C加加”，而西方的程序员通常读作“C plus plus”、“CPP”。它是

一种使用非常广泛的计算机编程语言。C++是一种静态数据类型检查的、支持多重编程范式的通用程序设计语言。它支持过程化程序设计、数据抽象、面向对象程序设计、泛型程序设计等多种程序设计风格。C++扩充和完善了C语言，成为一种面向对象的程序设计语言。相对于C语言，C++提出了一些更为深入的概念，它所支持的这些面向对象的概念容易将问题空间直接地映射到程序空间，为程序员提供了一种与传统结构程序设计不同的思维方式和编程方法。因而也增加了整个语言的复杂性，掌握起来有一定难度。

C++由美国AT&T贝尔实验室的本贾尼·斯特劳斯特卢普博士在20世纪80年代初期发明并实现（最初这种语言被称作“C with Classes”，即带类的C）。开始，C++是作为C语言的增强版出现的，从给C语言增加类开始，不断地增加新特性。虚函数（virtual function）、运算符重载（operator overloading）、多重继承（multiple inheritance）、模板（template）、异常（exception）、RTTI、命名空间（name space）逐渐被加入标准。

C++编程语言的优点如下：

（1）C++设计成静态类型，是和C语言同样高效且可移植的多用途程序设计语言。

（2）C++直接和广泛地支持多种程序设计风格（程序化程序设计、资料抽象化、面向对象程序设计、泛型程序设计）。

（3）C++给程序设计者更多的选择。

（4）C++尽可能与C兼容，借此提供一个从C到C++的过渡。

（5）C++避免平台限定或没有普遍用途的特性。

（6）C++不使用会带来额外开销的特性。

（7）C++设计无须复杂的程序设计环境。

（8）出于保证语言的简洁和运行高效等方面的考虑，C++的很多特性都是以库（如STL）或其他的形式提供的，而没有直接添加到语言本身里。

（9）C++在一定程度上可以和C语言很好地结合，甚至目前大多数C语言程序是在C++的集成开发环境中完成的。C++相对众多的面向对象的语言，具有相当高的性能。

C++引入了面向对象的概念，使得开发人机交互类型的应用程序更为简单、快捷。很多优秀的程序框架包括MFC、QT、wxWidgets使用的就是C++。

然而C++也有其自身的约束性：C++由于语言本身过度复杂，其语义甚至难于理解。由于本身的复杂性，复杂的C++程序的正确性相当难以保证。

1.2 计算机数据表示方法

计算机数据表示是指计算机硬件能够辨认并进行存储、传送和处理的数据表示方法。一台计算机的数据表示方法是计算机设计人员规定的，尽管数据的来源和形式有所不同，但输入这台计算机并经它处理的全部数据都必须符合规定。软件设计人员还可以依此来规定各数据类型（如虚数、向量等）和组织复杂的数据结构（如记录、文件等）。

早期的机械式和继电式计算机都用具有10个稳定状态的基本元件来表示十进制数据位0,1,2,…,9，一个数据的各个数据位是按10的指数顺序排列的，如$386.45=3\times10^2+8\times10^1+6\times10^0+4\times10^{-1}+5\times10^{-2}$。但是，要求处理机的基本电子元件具有10个稳定状态比较困难，十进制运算器逻辑线路也比较复杂。二进制是计算技术中广泛采用的一种数制，由18世纪德国数理哲学大师莱布尼兹发现。二进制运算比较简单，能节省设备，而且二进制与处理机逻辑运算能协调一致，

便于用逻辑代数简化处理机逻辑设计。因此，二进制得到广泛应用。

1. 定点表示法

在二进制中，0 和 1 分别由处理机电子元件的两个稳定状态表示，2 为数的基底。

2. 字符数据表示法

用二进制位序列组成供输入、处理和输出用的编码称为字符数据。字符数据包括各种运算符号、关系符号、货币符号、字母和数字等。中国通用的是 1980 年颁布的国家标准 GB 1988—80《信息处理交换用的七位编码字符集》，它以 7 个二进制位表示 128 个字符。它包括 32 个控制字符集、94 个图形字符集、一个间隔字符和一个抹掉字符。

1.2.1 二进制、八进制、十六进制

二进制数据是用 0 和 1 两个数码来表示的数，它的基数为 2，进位规则是“逢二进一”，借位规则是“借一当二”。当前的计算机系统使用的基本上是二进制系统，所有输入计算机的信息最终都要转化为二进制。最基本的单位为bit。

编程中，常用的还是十进制。

比如：

```
int a = 100，b = 99;
```

不过，由于数据在计算机中的表示最终以二进制的形式存在，所以有时候使用二进制可以更直观地解决问题。

首先来看一个二进制数：1111，它是多少呢？由于 1111 才 4 位，所以可以直接记住它每一位的权值，并且是从高位往低位记：8、4、2、1。即最高位的权值为 2^3=8，然后依次是 2^2=4，2^1=2，2^0 = 1。则二进制数 1111 对应的十进制数就是 1×8+1×4+1×2+1×1=15。

记住 8421，对于任意一个 4 位的二进制数，都可以很快算出它对应的十进制值。

采用二进制计数制，对于计算机等数字系统来说，运算、存储和传输极为方便。然而，二进制数用来表示一个数据的时候过于烦琐，比如 int 类型占用 4 个字节，32 位。如十进制数 100，用 int 类型的二进制数表达将是：

0000 0000 0000 0000 0110 0100

面对这么长的数进行思考或操作，没有人会喜欢。因此，C 和 C++编程语言没有提供在代码中直接写二进制数的方法。

用十六进制或八进制可以解决这个问题。因为，进制越大，数的表达长度也就越短。不过，为什么偏偏是十六或八进制，而不是其他的诸如九或二十进制呢？这是因为 2、8、16，分别是 2 的 1 次方、3 次方、4 次方，这一点使得三种进制之间可以非常直接地互相转换。八进制或十六进制缩短了二进制数，但保持了二进制数的表达特点。

1. 二进制数转换为十进制数

二进制数第 0 位的权值是 2^0，第 1 位的权值是 2^1，依次类推。

设有一个二进制数：0110 0100，转换为十进制数的计算如下。

用竖式计算：

0110 0100　换算成　十进制数

第 0 位　$0 * 2^0 = 0$

第 1 位　$0 * 2^1 = 0$

第 2 位　$1 * 2^2 = 4$

第 3 位　$0 * 2^3 = 0$

第 4 位　$0 * 2^4 = 0$

第 5 位　$1 * 2^5 = 32$

第 6 位　$1 * 2^6 = 64$

第 7 位　$0 * 2^7 = 0$　+

100

用横式计算：

$$0 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3 + 0 * 2^4 + 1 * 2^5 + 1 * 2^6 + 0 * 2^7 = 100$$

0 乘以多少都是 0，所以可以直接跳过值为 0 的位：

$$1 * 2^2 + 1 * 2^3 + 1 * 2^5 + 1 * 2^6 = 100$$

2. 八进制数转换为十进制数

八进制就是逢 8 进 1。

八进制数采用 0～7 这八个数来表达一个数。

八进制数第 0 位的权值为 8^0，第 1 位权值为 8^1，第 2 位权值为 8^2，依次类推。

设有一个八进制数：1507，转换为十进制数的计算如下。

用竖式计算：

第 0 位　$7 * 8^0 = 7$

第 1 位　$0 * 8^1 = 0$

第 2 位　$5 * 8^2 = 320$

第 3 位　$1 * 8^3 = 512$　+

839

同样，也可以用横式直接计算：

$$7 * 8^0 + 0 * 8^1 + 5 * 8^2 + 1 * 8^3 = 839$$

结果是八进制数 1507 转换成十进制数为 839。

在 C 和 C++语言中，如何表达一个八进制数呢？如果这个数是 876，可以断定它不是八进制数，因为八进制数中不可能出 7 以上的阿拉伯数字。但如果这个数是 123、567 或 12345670，那么它是八进制数还是十进制数都有可能。所以 C/C++规定，一个数如果要指明它采用八进制，必须在它前面加上一个 0，例如，123 是十进制数，但 0123 则表示采用八进制。这就是八进制数在 C/C++中的表达方法。

3. 十六进制数转换成十进制数

十六进制就是逢 16 进 1，但只有 0～9 这十个数字，所以用 A、B、C、D、E、F 这六个字母来分别表示 10、11、12、13、14、15。字母不区分大小写。

十六进制数的第 0 位的权值为 16^0，第 1 位的权值为 16^1，第 2 位的权值为 16^2，依次

类推。

所以，在第 N（N 从 0 开始）位上，如果是数 X（X 大于等于 0，并且 X 小于等于 15，即 F），表示的大小为 $X*16^N$。

假设有一个十六进制数 2AF5，那么如何换算成十进制数呢？

用竖式计算：

第 0 位：$5 * 16^0 = 5$

第 1 位：$F * 16^1 = 240$

第 2 位：$A * 16^2 = 2560$

第 3 位：$2 * 16^3 = 8192$ +

10997

直接计算就是：

$$5 * 16^0 + F * 16^1 + A * 16^2 + 2 * 16^3 = 10997$$

在上面的计算中，A 表示十进制数 10，而 F 表示十进制数 15。

现在可以看出，所有进制换算成十进制，关键在于各自的权值不同。

如果不使用特殊的书写形式，十六进制数也会和十进制数相混。比如随便一个数 9876，就看不出它是十六进制数还是十进制数。C/C++规定，十六进制数必须以 0x 开头。比如 0x1 表示一个十六进制数，而 1 则表示一个十进制数。另外，如 0xff、0xFF、0X102A 等，其中的 x 不区分大小写（注意：0x 中的 0 是数字 0，而不是字母 O）。

1.2.2 表示数据的字节和位

位（bit）又称“比特”，是存储信息的最小单位，它的值是二进制 1 或 0。计算机中所有的信息都是以“位”（bit）为单位表示、存储及传递的。早期的 DOS 操作系统就是 8 位的，后期的 DOS 操作系统是 16 位的，Win9X 操作系统是基于 DOS 的，所以也是 16 位的，NT 核心的 Windows 是 32 位的，现在也有了 64 位的 XP/Win7 操作系统等，CPU 也有 64 位的，所说的位就是 bit 的意思，即二进制数的长度。

每 8 个位（bit）组成一个字节（byte）。字节是什么概念呢？一个英文字母就占用一个字节，也就是 8 位，一个汉字占用两个字节。一般位简写为小写字母“b”，字节简写为大写字母“B”。

每一千个字节称为 1KB，注意，这里的“千”不是通常意义上的 1000，而是指 1024。即 $1KB=1024B=2^{10}B$。

每一千个 KB 就是 1MB（同样这里的 K 是指 1024），即

$$1MB=1024KB=1024\times1024B=1048576B$$

每一千个 MB 就是 1GB，即 1GB=1024MB。例如，一篇 10 万汉字的小说，如果存到磁盘上，需要占用多少空间呢？

100000 汉字=200000B=(200000B÷1024)KB≈195.3KB≈(195.3KB÷1024)MB≈0.19MB

另外一个例子就是网速，提到网速的时候经常会省略单位，往往只是说 G、M、K，其实 G、M、K 是数量的简略表示法，换算公式为：1G=1024M，1M=1024K，1K=1024，就相当于人们常说的亿、万、千，只是数量的简略表示而已，并不是单位，单位是字节/秒（Bps）。

1.2.3 内存

内存（Memory）也称为内存储器，其作用是暂时存放 CPU 中的运算数据，以及与硬盘等