C#2.0 程序设计初步

C#2.0chengxushejichubu

贺 敏 主编

辽宁科学技术出版社
LIAONING SCIENCE AND TECHNOLOGY PUBLISHING HOUSE

C# 2.0 程序设计初步

贺　敏　主编

辽宁科学技术出版社
沈　阳

图书在版编目（CIP）数据

C# 2.0 程序设计初步/贺敏主编. — 沈阳：辽宁科学技术出版社，2007.9

ISBN 978-7-5381-5188-6

Ⅰ.C… Ⅱ.贺… Ⅲ.C语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2007）第 120672 号

出版发行：辽宁科学技术出版社
（地址：沈阳市和平区十一纬路 29 号 邮编：110003）
印 刷 者：沈阳市第二印刷厂
经 销 者：各地新华书店
幅面尺寸：184mm×260mm
印 张：14.5
字 数：300 千字
出版时间：2007 年 9 月第 1 版
印刷时间：2007 年 9 月第 1 次印刷
责任编辑：韩延本 高 鹏
封面设计：留藏设计工作室
版式设计：于 浪
责任校对：李淑敏

书 号：ISBN 978-7-5381-5188-6
定 价：33.00 元
联系电话：024-23284372
邮购热线：024-23284502
E-mail：lkzzb@mail.lnpgc.com.cn
http：//www.lnkj.com.cn

前　言

软件工程的目的就是提高软件系统的质量和开发、维护效率。作为软件工程技术中的重要一环，程序设计语言的每一次进步都体现了软件工程的先进思想，都是大师们才华和创意的结晶。可以说，软件发展的终极目标就是有一天不再有程序员，工程师、科学家以及商业管理者都可以轻松地构建自己所需的软件系统。

如果这个目标真的实现的话，回顾程序设计语言的发展史，C#语言的出现必将是其中浓墨重彩的一笔。完全面向对象、面向组件、样板式的设计、简洁的语法、标准化的接口，这一切使得C#语言更像是程序设计语言中的艺术品。无数非专业人员迅速地学习并掌握了这门语言，编程的门槛进一步降低了。

然而，作为一个新生事物，C#语言不可避免地受到了诸多怀疑和批评。寄生于.NET框架之上，而且比起C/C++来底层控制能力大幅降低，这使得许多有经验的开发人员不愿接受它。不少专家早早地下了定论：C#不可能像汇编、Fortran、Pascal或C/C++那样成为程序设计语言发展历程上的一个里程碑。

C#从正式推出到现在不过5年多的时间，而C++从面世到成为面向对象技术的集大成者走过了10多年。重要的是，微软对C#极为重视，有了这个支持使得C#自我完善的脚步之快出乎所有人的意料，从1.0升级到2.0，C#语言有了很大的改进，现在，C# 3.0也即将正式面世。更让人意外的是，面对质疑，C# 2.0并没有增加底层控制能力，也没有脱离对.NET框架的依赖。所改进的重点几乎都集中在C#语言最初的强项上，带来的是更灵活、更简便、更高效的开发方式。

这种看似一意孤行的做法实际上却是相当符合情理的。C#的推出并不是要取代C/C++，而是定位为商业应用开发语言，因此，对底层的控制能力并不是它所要考虑的。C#语言中的对象框架原本就是一个简洁而完整的设计，而C# 2.0又把对象和泛型这两个编程概念进行了完美的结合。并且C++中的多继承、指针等语法本身就是双刃剑，因此，C#干脆抛弃了它们，使得C#语法更加简洁易学，也许不久的将来，C#将在许多大学计算机专业的高级语言课程上取代C++。

.NET框架是微软.NET战略的重要组成，是一个全新的开发平台，可以大大提高软件的开发效率。C#语言与其说是寄生于.NET，还不如说C#是为.NET框架量身定做的语言。考虑到微软及Windows平台的统治力，也许有朝一日.NET将成为组件技术的集大成者，那么，作为.NET的官方语言的C#，必然会成为网络服务技术的代名词。

本书可以作为计算机专业师生学习C#语言的教材和辅导书，也可以供软件从业人员以及程序设计爱好者自学参考，尤其适合没有编程基础而又想成为一名程序员的初学者。

全书采用循序渐进的方式，逐步引导读者全面系统地掌握C#语言的语法规则和基本特性，主要内容安排如下。

第1章 绪论：主要介绍了一些程序设计导论方面的内容，如数的表示、简单的数

理逻辑、程序与算法描述、C#简介等。

第2章 初识C#：以第一个C#程序介绍了C#程序的基本结构，另外介绍了.NET Framework的相关知识。

第3章 数据类型：主要介绍了C#中的系统预定义数据类型及用户自定义数据类型。

第4章 变量和常量：主要介绍C#中的数据存储。

第5章 表达式和语句：介绍了C#中各种常用的表达式运算，如算术运算、关系运算、逻辑运算、位运算等。

第6章 数组：重点介绍了C#中处理多个相同类型数据的一维数组。

第7章 控制结构：重点介绍了面向结构编程的重要元素，包括分支、循环、跳转等多种语法结构，并安排了大量的示例以帮助读者熟练掌握本章内容。

第8章 面向对象初步：重点介绍了类的各个方面的知识，包括类和对象、类的组成、构造器、方法重载、参数传递方式、运算符重载等。

第9章 再论字符串类型：详细介绍了字符串类型的常用属性和方法，并介绍了StringBuilder的用法。

第10章 System常用类探访：重点介绍了FCL中的一些在程序设计中常用的类型，如文件读写、集合类、时间类、随机类等。

书中的每一章开始都提出了学习目标，点出该章的重点知识；每一章结束都进行了小结，对该章的关键知识点进行回顾。此外，各重点章节都配备了一定数量的习题，通过这些练习能帮助读者有的放矢地复习所学的内容。另外，本书后面的附录文档注释、中英文对照、代码规范，可以作为参考，供读者随时查阅。

本书通过大量精心设计、极具代表性的源程序来讲解C#的各种特性，其中可实际运行的完整程序近100个。本书还以一个学生管理信息系统（CUI应用程序）为例，完成了其中的核心功能。读者学完本书，应该可以完整实现本系统。编译这些源代码需要Microsoft .NET Framework 2.0，它提供了C#语言的编译器CSC。可以从Microsoft的网站上免费获取，其下载地址是：

http://www.microsoft.com/downloads/details.aspx?familyid=b7adc595-717c-4ef7-817b-bdefd6947019&displaylang=en，文件大小约24MB。

没有许多人的帮助，编者是不可能完成本书的。首先，要感谢我所任教的成都东软信息技术职业学院为我提供了良好的编书环境，让我在教学实践中不断改进和完善本书。其次，张应辉院长和胡景德副院长一直关注和支持可视化专业的教学改革，特别是胡院长亲自指导了教学改革，并多次询问该书的进度并给予了有益的指示。另外，我所在的软件技术教研室副主任段恩泽老师在本书的撰写过程中也给予了很大的支持。如果没有他们，该书是不可能产生和完成的。

辽宁科学技术出版社为本书的修订和出版做了大量的工作，与他们的合作非常愉快。

最后，还要特别感谢我的家人。为了完成本书，编者投入了大量的时间和精力，牺牲了许多周末和节假日。特别是编书期间，编者父亲做手术都没能抽出更多的时间陪

伴，深感惭愧。

尽管编者在写作过程中非常认真和严谨，但书中难免存在错误和不足之处，恳请广大读者批评指正。如果您对本书有什么意见、问题或想法，欢迎您通过下面的邮件地址与编者联系，编者将不胜感激。

E－mail：he. min@ neusoft. com

请在邮件的主题栏中注明：C# 2.0 程序设计初步。

编　者

2007 年 6 月

目 录

第1章 绪论 …… 1
1.1 计算机内的数的表示 …… 1
1.1.1 常用的进制数 …… 2
1.1.2 进制转换 …… 3
1.1.3 原码、反码和补码 …… 5
1.1.4 常用的编码方式 …… 7
1.2 简单的逻辑运算 …… 8
1.2.1 或、与、非运算 …… 8
1.2.2 真值表 …… 9
1.3 程序与算法 …… 11
1.3.1 程序 …… 11
1.3.2 算法 …… 13
1.3.3 算法描述 …… 14
1.4 C#语言简介 …… 15
1.4.1 C#的特点 …… 15
1.4.2 C#的发展 …… 16
1.5 小结 …… 16
1.6 习题 …… 16
第2章 初识 C# …… 18
2.1 Microsoft . NET 策略 …… 18
2.2 公共语言架构（CLI） …… 19
2.3 . NET Framework …… 19
2.3.1 公共语言运行时（CLR） …… 20
2.3.2 基本框架库 …… 21
2.4 C#程序基本结构 …… 22
2.4.1 注释 …… 23
2.4.2 命名空间和类 …… 24
2.4.3 程序入口点 …… 25
2.4.4 命令行编译方式 …… 26
2.4.5 编译时错误 …… 28
2.4.6 C#代码规范 …… 28
2.5 C#程序的编译过程 …… 29
2.6 C#学习拓扑 …… 29
2.7 小结 …… 30

2.8 习题 …… 30
第3章 数据类型 …… 32
3.1 值类型与引用类型 …… 32
3.2 基本数据类型 …… 32
3.2.1 整数类型 …… 33
3.2.2 实数类型 …… 35
3.2.3 字符类型 …… 36
3.2.4 布尔类型 …… 36
3.2.5 字符串类型 …… 36
3.2.6 统一型别 object …… 38
3.3 用户自定义数据类型 …… 38
3.3.1 枚举类型 …… 38
3.3.2 结构类型 …… 39
3.3.3 引用类型 …… 41
3.4 类型转换 …… 42
3.4.1 数值转换 …… 42
3.4.2 枚举转换 …… 43
3.4.3 数值与字符串的转换 …… 44
3.4.4 装箱和拆箱 …… 46
3.5 小结 …… 47
3.6 习题 …… 47
第4章 变量和常量 …… 49
4.1 变量 …… 49
4.1.1 变量的命名 …… 49
4.1.2 变量的分类 …… 50
4.1.3 声明与初始化 …… 50
4.1.4 生存期与可见性 …… 51
4.2 常量 …… 52
4.3 小结 …… 53
4.4 习题 …… 53
第5章 表达式和语句 …… 54
5.1 语句 …… 54
5.2 操作符的优先级和结合性 …… 54
5.3 算术表达式 …… 56
5.3.1 基本算术运算 …… 56
5.3.2 模运算 …… 58
5.3.3 字符串加法 …… 59
5.3.4 自增、自减表达式 …… 60

5.4 关系表达式 …… 61
5.4.1 判等运算 …… 61
5.4.2 比较大小 …… 62
5.4.3 类型判断 …… 63
5.5 逻辑表达式 …… 63
5.5.1 基本逻辑运算 …… 64
5.5.2 短路表达式 …… 65
5.6 赋值运算 …… 65
5.6.1 简单赋值 …… 66
5.6.2 复合赋值 …… 66
5.7 位运算表达式 …… 67
5.7.1 取补运算 …… 67
5.7.2 与、或和异或运算 …… 68
5.7.3 移位运算 …… 70
5.8 其他运算 …… 70
5.8.1 问号表达式 …… 70
5.8.2 溢出检查表达式 …… 71
5.8.3 类型表达式 …… 72
5.9 小结 …… 74
5.10 习题 …… 74
第6章 数组 …… 76
6.1 数组的定义 …… 76
6.2 数组元素的访问 …… 77
6.3 数组的常用操作 …… 79
6.4 其他数组介绍 …… 80
6.5 小结 …… 80
第7章 控制结构 …… 81
7.0 引例 …… 81
7.1 分支结构 …… 83
7.1.1 if 语句 …… 83
7.1.2 switch 语句 …… 86
7.2 循环结构 …… 88
7.2.1 while 循环 …… 88
7.2.2 do – while 循环 …… 90
7.2.3 for 循环 …… 92
7.2.4 foreach 循环 …… 94
7.3 跳转语句 …… 95
7.3.1 break 语句 …… 95

7.3.2 continue 语句 …… 95
7.3.3 return 语句 …… 96
7.3.4 goto 语句 …… 96
7.4 控制语句的应用 …… 98
7.4.1 求质数 …… 99
7.4.2 九宫算术 …… 101
7.4.3 主界面设计 …… 103
7.5 小结 …… 106
7.6 习题 …… 106
第 8 章 面向对象初步 …… 108
8.1 面向对象程序设计基础 …… 108
8.2 命名空间和类 …… 109
8.3 类的组成 …… 110
8.4 类和结构的区别 …… 110
8.5 字段 …… 111
8.6 成员访问修饰符 …… 112
8.7 类的初始化 …… 112
8.7.1 构造器 …… 112
8.7.2 new 操作符 …… 113
8.8 类和对象 …… 114
8.8.1 static 成员 …… 114
8.8.2 const 字段 …… 115
8.8.3 this 指针 …… 115
8.9 方法 …… 116
8.9.1 方法的组成 …… 116
8.9.2 参数传递方式 …… 121
8.9.3 方法重载 …… 128
8.9.4 运算符重载 …… 131
8.10 属性和索引 …… 136
8.10.1 属性 …… 136
8.10.2 索引 …… 138
8.11 类的应用 …… 140
8.11.1 求质数 …… 140
8.11.2 九宫算术 …… 144
8.11.3 案例——学生集合类的设计 …… 147
8.12 小结 …… 153
8.13 习题 …… 153
第 9 章 再论字符串类型 …… 155

9.1 构造器 …… 155
9.2 字段和属性 …… 156
9.3 常用方法 …… 156
9.3.1 字符操作和子串操作 …… 156
9.3.2 比较字符串和连接字符串 …… 160
9.3.3 字符替换 …… 162
9.3.4 字符填充 …… 164
9.3.5 字符修剪 …… 165
9.3.6 字符串分割 …… 165
9.4 StringBuilder 类 …… 167
9.4.1 字符串的不变性 …… 167
9.4.2 StringBuilder 的构造器 …… 168
9.4.3 StringBuilder 的用法 …… 168
9.5 小结 …… 172
9.6 习题 …… 172
第 10 章 System 常用类探访 …… 173
10.1 统一型别系统——object …… 173
10.2 数据容器类 …… 174
10.2.1 ArrayList 链表 …… 174
10.2.2 Hashtable 哈希表 …… 175
10.3 System. IO …… 176
10.3.1 处理文件扩展名：Path Class …… 177
10.3.2 目录操作（Directory and DirectoryInfo） …… 178
10.3.3 文件操作（File and FileInfo） …… 182
10.3.4 文件读写 …… 184
10.3.5 文件读写的应用 …… 188
10.4 其他 …… 195
10.4.1 Console Class …… 195
10.4.2 Environment Class …… 196
10.4.3 Math Class …… 198
10.4.4 DateTime Class …… 199
10.4.5 Random Class …… 200
10.5 小结 …… 203
附录 A 文档注释 …… 204
附录 B 中英文对照 …… 207
附录 C C#代码规范 …… 214

第1章 绪 论

计算机是20世纪最伟大的科技成果之一，它引发了社会生产力的迅猛发展，使人们观念有了飞跃性的变革。在短短几十年内人们迅速地从铅与火的时代走进了光与电的时代，过去靠人工进行的大量繁琐的工作现在都可以由计算机来完成，计算机还能做许多人力不便做或不能做的事情。神奇的计算机改变了世界。

人们要使用计算机为自己服务，就必须能够和计算机通信，把人的意图告诉计算机，计算机在理解之后，就可发挥其高速的运算能力和处理能力为人们工作了。计算机语言是人和计算机通信的工具，但由于人们的意图需要通过程序形式提交给计算机，所以与计算机通信必须用计算机语言写出程序，这就称为程序设计。程序语言和程序设计涉及到很多知识，本章首先对一些基本知识作简单的介绍。

【学习目标】

* 掌握计算机内的数的表示，熟练进行进制间的转换。
* 掌握简单的逻辑运算，熟悉与、或、非的运算法则。
* 基本掌握常用的编码方式。
* 基本了解程序和算法，熟练掌握流程图的绘制。
* 了解C#的发展和语法特点。

1.1 计算机内的数的表示

数据形态多种多样，凡是计算机能够处理的对象都称为数据，如文本数据、图形数据、声音数据等。但不管其表象多么复杂，多么千差万别，只要交给计算机处理，全部都统一为二进制数据，也就是说，数据在计算机内是以二进制（即用数字0和1）来表示的。

在计算机中为什么要用二进制？采用二进制有哪些优点？

（1）二进制表示数字在物理上容易实现。

在计算机内部用电子器件的状态表示数字信息。一种数制有多少种不同的数字，电子器件就需要有多少种不同的状态。二进制只有数字0和1，因此，它只需要两个稳定状态的电子元件就可以表示了，而十进制则需要10个不同稳定状态的电子元件来表示。显然对电子元件的制造来讲用二进制要简单，如开关的接通与断开，晶体管的导通与截止，都可以由0和1两个数字来表示。因此，用二进制表示数字，数字的状态少，工作可靠。

（2）二进制运算规则简单。

二进制求和的规则有：

$0+0=0 \quad 0+1=1+0=1 \quad 1+1=10$

二进制求积的规则有：

$0\times0=0$　$0\times1=1\times0=0$　$1\times1=1$

一般地，若一种数制中有 n 个不同的数字，根据排列组合的法则可知其求和与求积各需要 $\frac{n(n+1)}{2}$ 条不同的规则。按这一公式对十进制求和与求积就需要 55 条规则。

（3）采用二进制可以用逻辑代数作为设计分析的工具。

二进制中用 0 和 1 可以表示是与非、高与低等，这恰恰是逻辑代数中的内容。

（4）用二进制可以节约存储设备。

比如要表示 0～999 这 1000 个数时，十进制要用三位数，需 $3\times10=30$ 个状态设备量；用十位二进制可表示 1024 个数，只需 $10\times2=20$ 个状态设备量。

1.1.1 常用的进制数

常用的数制有二进制、十进制、八进制、十六进制，它们有共性也有差别。

（1）数码及进位法则。

数码是构造一种数制所用的不同符号，各种进制的数码为：

二进制：0，1

十进制：0，1，2，3，4，5，6，7，8，9

八进制：0，1，2，3，4，5，6，7

十六进制：0，1，2，3，4，5，6，7，8，9，A（a），B（b），C（c），D（d），E（e），F（f）

在十六进制中有十六个数码，但自 10 起就没有相应的阿拉伯数字可以使用，所以用字母 A、B、C、D、E、F（大小写均可）来表示 10、11、12、13、14、15。

各种数制的共性是：N 进制中的最大数码比 N 少 1；在 N 进制中，逢 N 进 1，借 1 当 N。

（2）位置计数法。

数据中各个数字所处的位置决定它的大小即权值，同样的数字在不同位置上代表的权值是不同的，比如：

$12.3=1\times10^{1}+2\times10^{0}+3\times10^{-1}$

其中 10^{1}、10^{0}、10^{-1}就分别是 1、2、3 所在位置的权值。

采用位置计数法，任何数制下的数都可以表示成多项式的形式。如有数据

$N=K_nK_{n-1}\cdots K_1K_0K_{-1}K_{-2}\cdots K_{-m}$

都可以表示成：

$$N=K_n\times x^n+K_{n-1}\times x^{n-1}+\cdots+K_1\times x^1+K_0\cdot x^0+K_{-1}\times x^{-1}+K_{-2}\times x^{-2}+\cdots+K_{-m}\times x^{-m}=\sum_{i=n}^{-m}K_i\times x^i=\sum_{i=n}^{0}K_i\times x^i+\sum_{i=-1}^{-m}K_i\times x^i=N_i+N_f$$

这里 N_i 为整数部分，N_f 为小数部分，x 为进制数，可以是 2、8、10、16 等。二进制、十进制、八进制、十六进制都采用位置计数法。

1.1.2 进制转换

我们日常习惯使用的是十进制，但在计算机中用的是二进制，所以需要把十进制转换成二进制。但二进制书写麻烦，因此，通常用八进制和十六进制表示，这样就存在各种数制之间的转换问题。

1. 将十进制转换成二进制

十进制的整数和小数转换成二进制时所用的方法不同，因此，应该分别进行转换。

（1）用余数法将十进制整数转换成二进制整数。把十进制整数不断地用 2 去除，将所得到的余数 0 或 1 依次记为 K_0，K_1，K_2，…，直到商是 0 为止，将最后一次所得的余数记为 K_n，则 $K_nK_{n-1}\cdots K_1K_0$ 为该整数的二进制表示。

在演算过程中可用竖式形式，也可用线图形式。

【示例 1-1】$(59)_{10}=(K_n\cdots K_1K_0)_2$

* 竖式演算如下：

2	59	余数 1 = K_0
2	29	余数 1 = K_1
2	14	余数 0 = K_2
2	7	余数 1 = K_3
2	3	余数 1 = K_4
2	1	余数 1 = K_5
	0	

最后得到的余数在最高位，所以有

$(59)_{10}=(K_5K_4K_3K_2K_1K_0)_2=(111011)_2$

* 线图演算如下：

59→29→14→7→3→1→0

÷2 ↓ ↓ ↓ ↓ ↓ ↓

余数 1 1 0 1 1 1

K_0 K_1 K_2 K_3 K_4 K_5

←

同样有

$(59)_{10}=(K_5K_4K_3K_2K_1K_0)_2=(111011)_2$

（2）用进位法将十进制小数转换成二进制小数。把十进制小数不断地用 2 去乘，将所得乘积的整数部分 0 或 1 依次记为 K_{-1}，K_{-2}，K_{-3}，……一般情况下，十进制小数并不一定都能用有限位的二进制小数表示，可根据精度要求转换成一定位数即可。

【示例 1-2】把 0.47 转换成二进制。

用线图形式可演算如下：

$$0.47 \rightarrow 0.94 \rightarrow 0.88 \rightarrow 0.76 \rightarrow 0.52 \rightarrow 0.04$$

$\times 2$ ↓ ↓ ↓ ↓ ↓

整数 0 1 1 1 1

K_{-1} K_{-2} K_{-3} K_{-4} K_{-5}

在取 5 位小数时有

$(0.47)_{10} = (0.K_{-1}K_{-2}K_{-3}K_{-4}K_{-5}) = (0.01111)_2$

（3）十进制转换成二进制时，熟记一些 2 的幂次的十进制及二进制值能加快转换速度。

在转换时，可找出小于但最接近该十进制整数的某个 2 的幂次，用待转换数减去这个幂次，再在得出的数中找小于且最接近该数的 2 的幂次……2 的 n 次方的二进制数的特点是 1 后跟有 n 个 0，利用这一点就能迅速地把二进制数的长度确定，然后在相应的位置上填上 1 或 0 即可。比如：

$(1287.25)_{10} = 1024 + 256 + 4 + 2 + 1 + 0.25$

$= 2^{10} + 2^8 + 2^2 + 2^1 + 2^0 + 2^{-2}$

$= (10000000000 + 100000000 + 100 + 10 + 1 + 0.01)_2$

$= (10100000111.01)_2$

如果一个数比 1024 大很多，可求出该数对 1024 的倍数，把倍数用 2 的幂次之和的形式表示，再与 2^{10} 相乘。

【示例 1－3】$(30000)_{10} = (\quad)_2$

30000 除以 1024 得商 29，余数为 304，将商和余数分别用 2 的幂次之和的形式表示：

商：$29 = 16 + 8 + 4 + 1 = 2^4 + 2^3 + 2^2 + 2^0$

余数：$304 = 256 + 32 + 16 = 2^8 + 2^5 + 2^4$

$(30000)_{10} = 1024 \times 29 + 304 = 2^{10}\ (2^4 + 2^3 + 2^2 + 2^0)\ + 2^8 + 2^5 + 2^4$

$= 2^{14} + 2^{13} + 2^{12} + 2^{10} + 2^8 + 2^5 + 2^4$

$= (111010100110000)_2$

上面介绍了将十进制数转换为对应的二进制数的方法，转换其他进制数（八进制、十六进制）的方法与此类似，只需将前面对进制 2 进行的操作换成对进制 8 或 16 操作即可得到相应的进制数。如 1000 的十六进制数的转换：用 1000 对 16 取余，用得到的商继续取余直至商为 0，将每次得到的余数逆序排列，则得到其十六进制的表示 $(3E8)_{16}$。

2. 将二进制转换成十进制

把二进制数按多项式展开求和即可。

$(101.101)_2 = (1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3})_{10}$

$= (1 \times 4 + 1 \times 1 + 1 \times 0.5 + 1 \times 0.125)_{10}$

$= (5.625)_{10}$

其他进制（八进制、十六进制）转换为十进制，同理操作。

1.1.3 原码、反码和补码

在计算机中，位（bit）是最小的单位，它通常用来表示由二进制位组成的信息长度。把8位二进制位定义为一个字节（Byte），计算机中的存储量就是按字节来计算的。计算机内部数据处理的基本单位称为字（word），它通常也是输入/输出设备和存储器之间传递数据的基本单位，每个字所包含的位数称为字长。

在计算机中，不仅数值用0和1表示，正负号也用0和1表示。一般规定，一个存储单位的最高位（最左边的一位）为符号位，如该位是0表示正，该位是1表示负。

在计算机中，带符号的数通常有三种表示方法：原码、反码和补码。为简单起见，下面我们只考虑用一个字节表示整数时的原码、反码和补码。

1. 原　码

原码是一种机器数，原码表示法就是在机器中最高位用0表示正数，用1表示负数，其余位表示数本身。例如：

+15的原码为：00001111

↑

代表正

-15的原码为：10001111

↑

代表负

+0的原码为：00000000

-0的原码为：10000000

同一个0在计算机中有不同的表示，这不适合计算机的运算。

2. 反　码

在反码表示法中，正负数的表示是不同的。正数的反码和原码是一样的，如：

+15的反码仍然是00001111

负数的反码为：符号位为1，其他各位是对原码求反。如：

-15的反码为11110000

数值0的反码有两种：

+0的反码为00000000

-0的反码为11111111

在反码表示法中，0的表示仍然不唯一。

3. 补　码

鉴于以上情况，计算机内部采用补码方法来表示数值。采用补码方法的好处是简化设计与计算，能把减运算转变为加运算来进行。

我们以时钟为例来说明补码的原理。比如现在是标准时间6点，而你的表停在11点的位置。要调准你的表可用两种方法：①向前拨7格；②向后拨5格。

不管采用哪种方法，都能达到目的，可谓殊途同归。但从算式上来看：

11+7=18（向前拨）

11 - 5 = 6（向后拨）

计算结果并不一样，那么操作结果为什么会一样呢？这是因为表盘的刻度是以 12 为周期的，超过 12 就又重头计数，因此，上面的计算结果 18 因为超过了 12，再从头开始计数得到了 6。这个 12 就成为系统的模数，7 和 5 相对于模数 12 来说是互补的，即一个数是另一个数的补码。从上面可以看出，减一个数就等于加上它的补码，效果是相同的，这就是把减法变成加法的原理。

在计算机中是以一个有限长度的二进制位作为模，比如用一个字节表示一个数，则其模数为 2^8。如运算结果超过 2^8，就从中减去 2^8。反映在内存中，其情况为：

1 00000000

即把 8 位以外的数舍掉。

计算机中的补码是这样定义的：

正数：其补码与原码、反码相同。例如：

$[+15]_{补} = [+15]_{原} = [+15]_{反} = 00001111$

$[+127]_{补} = [+127]_{原} = [+127]_{反} = 01111111$

负数：最高位为 1，其余各位在原码的基础上取反，然后在最低位加 1，简称“求反加 1”。可用如下关系式表示：

$[x]_{补} = [x]_{反} + 1$

注意：各位取反时不包括符号位，即符号位不求反。例如：

$[-15]_{原} = 10001111$　　$[-127]_{原} = 11111111$　　$[-0]_{原} = 10000000$

$[-15]_{反} = 11110000$　　$[-127]_{反} = 10000000$　　$[-0]_{反} = 11111111$

$[-15]_{补} = 11110001$　　$[-127]_{补} = 10000001$　　$[-0]_{补} = 00000000$

因为 $[-0]_{反} + 1 = 100000000$，把超过 8 位的部分舍掉，则 $[-0]_{补}$ 的结果就是 00000000，而 +0 的补码也是 00000000，即在补码表示法中，0 的表示是唯一的。

在 8 位字长情况下，-128 是一个特殊的数，计算机中规定它的表示为 10000000，即它的补码为：

$$[-128]_{补} = 10000000$$

我们也可以这样来看一个补码的生成，如表 1 - 1 所示。

表 1 - 1　补码的生成

数　值	补　码
0	00000000
-1	11111111
-2	11111110
…	…（往下不断减 1）
-127	10000001
-128	10000000

-127 以前各数的补码既可以按 $[x]_{补} = [x]_{反} + 1$ 的公式得到，也可以按在前一个负数补码的基础上减 1 的规则来得到，按这个规则求出 -128 的补码为 10000000 也就很自然了。