第 1 章　计算机程序设计概述

1.1　计算机系统组成

计算机是 20 世纪人类最伟大的创造发明之一，它是电子技术和计算技术空前发展的产物，是科学技术与生产力发展的结晶。计算机极大地推动了科学技术的发展，现已成为当今社会各行各业不可缺少的工具。

计算机是一种能够按照事先存储的程序，自动、高速地进行数值计算和信息处理的现代化智能电子设备。计算机系统由硬件系统和软件系统两部分组成。硬件系统是计算机系统的物理部分，是由电子线路、元器件和机械部件等构成的具体装置；软件系统是计算机系统中运行的各种程序、程序所需数据的集合。计算机系统的基本组成如图 1-1 所示。

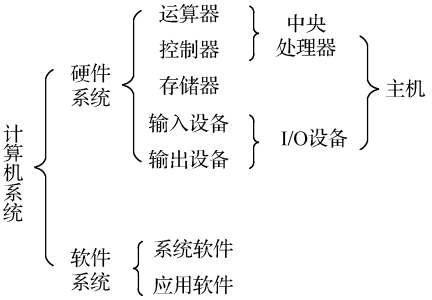


图 1-1　计算机系统的基本组成

通常将运算器和控制器称为中央处理器（Central Processor Unit，CPU），将输入设备和输出设备称为 I/O 设备（输入/输出，Input/Output），将中央处理器和 I/O 设备合称为主机。

1.1.1　硬件系统

当前计算机都是冯・诺依曼结构的，硬件系统由 5 部分构成，如图 1-2 所示，各部分功能如下。

（1）控制器。控制器为计算机的控制中心，从存储器中读取并分析指令，根据指令要求完成相应操作。控制器产生一系列控制命令，使硬件系统各部分协调工作，完成程序和数据的输入和运算并输出结果。

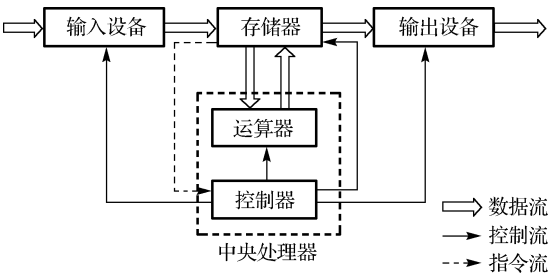


图 1-2　冯・诺依曼结构计算机

（2）运算器。运算器是中央处理器的执行单元，是所有中央处理器的核心组成部分。运算器在控制器的控制下，接收运算数据，完成指令指定的二进制算术运算。

（3）存储器。存储器是可以被中央处理器直接访问而无须通过输入/输出设备的记忆设备。存储器用来保存程序和数据，以及存储运算时的数据和结果。

（4）输入设备。输入设备是用来完成输入功能的部件。通过输入设备可以向计算机送入程序、数据及各种信息。常用的输入设备有键盘、鼠标、扫描仪、磁盘驱动器和触摸屏等。

（5）输出设备。输出设备是用来将计算机的中间运行情况或运行后的结果进行表现的设备。常用的输出设备有显示器、打印机、绘图仪和磁盘驱动器等。

根据冯·诺依曼结构，计算机自动执行程序，即执行指令，可分为如下几个过程。

（1）取指阶段。从存储器某地址处取出要执行的指令，送到中央处理器内部的指令寄存器中暂存。

（2）译码阶段。对保存在指令寄存器中的指令进行分析，翻译出该指令对应的操作。

（3）执行阶段。根据译码结果向各个部件发出相应控制信号，完成指令规定的操作。

（4）取下一条指令。为执行下一条指令做好准备，即产生下一条指令地址。

1.1.2 软件系统

软件系统为指挥计算机工作的程序和程序运行时所需要的数据，以及所需要的数据和文档的集合。软件系统分为系统软件和应用软件两部分。

（1）系统软件。计算机系统必备的基本软件，是管理、监控和维护计算机硬件和软件资源、开发应用的软件。按功能可分为 5 类：操作系统（如 Windows、Linux）、语言处理程序（如汇编程序、编译程序）、程序设计语言程序（如 C、Java 语言）、系统支持和服务程序（如系统诊断程序、查杀病毒程序）、数据库管理系统（如 Oracle、SQL Server）。

（2）应用软件。由系统软件开发的，为解决计算机各类应用问题而编写的程序，具有较强的实用性。按功能可分为两类：用户程序（如天气预报软件）和应用软件包（如 Microsoft Office 套件）。

1.2 程序设计语言

使用计算机，就需要和计算机交换信息。为解决人们和计算机交流的问题，就产生了程序设计语言。程序设计语言是用于编写计算机程序的语言，根据程序设计语言的发展，可分为三个阶段：很难理解的机器语言、较难理解的汇编语言、脱离机器的高级语言。

1．机器语言

机器语言是用二进制代码表示的计算机能直接识别和执行的一种机器指令的集合，是计算机唯一能够直接识别的程序设计语言。机器语言具有灵活、直接执行和速度快等特点。机器语言是直接对计算机硬件产生作用的，因此不同类型的计算机采用的机器语言是不同的，没有通用性，使得机器语言很难被人们掌握和推广，通常只有计算机专家或专业人员才使用机器语言。

【例 1-1】 以下是某 CPU 利用机器语言计算 a=a+b 的代码，其中 a=1，b=2。

```
#01: 11100011 10100000 00000000 00000001
#02: 11100011 10100000 00010000 00000010
#03: 11100000 10000000 00000000 00000001
```

代码解释：

#01：令 a=1；

#02：令 b=2；

#03：将 a 和 b 的值相加，并将结果放在 a 中。

2．汇编语言

为了克服机器语言难理解、难编写、难记忆和易出错的缺点，人们就用与代码指令意义相近的英文缩写词、字母和数字等符号来取代指令代码，于是产生了汇编语言。汇编语言采用了助记符来代表低级机器语言的操作，它和机器语言基本上是一一对应的，更便于记忆。

用汇编语言编写的程序称为汇编语言源程序，需要汇编程序将源程序汇编或翻译成机器语言源程序后，计算机才能执行。

汇编语言和机器语言都是面向机器的程序设计语言，不同的机器具有不同的指令系统，一般将它们称为低级语言。

【例 1-2】 以下是某 CPU 利用汇编语言计算 a=a+b 的代码，其中 a=1，b=2。

```
#01: MOV R0, #1
#02: MOV R1, #2
#03: ADD R0, R0, R1
```

代码解释：

#01：将数值 1 放入寄存器 R0 中；

#02：将数值 2 放入寄存器 R1 中；

#03：将寄存器 R0 的值和寄存器 R1 的值相加，结果放入寄存器 R0 中。

3．高级语言

计算机的发展促使人们去寻求一些与人类自然语言相接近且能为计算机所接受的语意确定、规则明确、自然直观和通用易学的计算机语言。这种与自然语言相近、与具体的计算机指令系统无关、其表达方式更接近人们对求解问题的描述方式的计算机语言称为高级语言。目前广泛使用的高级语言有 PASCAL、C、C++、Java、C#等。

使用高级语言编写的程序称为源程序，计算机并不能直接接受和执行用高级语言编写的源程序，需要通过翻译程序翻译成机器语言形式的目标程序，再与有关的库程序连接成可执行程序，计算机才能识别和执行。这种翻译通常有两种方式：编译方式和解释方式。

【例 1-3】 下面是利用 C 语言计算 a=a+b 的代码，其中 a=1，b=2。

```
#01: int a=1;
#02: int b=2;
#03: a=a+b;
```

代码解释：

#01：使变量 a 等于 1；

#02：使变量 b 等于 2；

#03：将变量 a 和变量 b 相加，结果赋值给变量 a。

1.3　计算机算法简介

做任何事情都有一定的步骤和方法，算法是在有限步骤内解决某一问题所使用的一组定义明确的规则。计算机算法是计算机能够执行的指令和规则，可分为两大类：数值运算算法和非数值运算算法。

1969 年，Edsger W. Dijkstra 首次提出了“结构化程序设计”的概念，1971 年，Niklaus E. Wirth 提出了“通过逐步求精方式开发程序”的思想，提出公式“算法+数据结构=程序”，一个程序应包括对数据的描述，在程序中要指定数据的类型和数据的组织形式，即数据结构（data structure）；和对操作的描述，即操作步骤，也就是算法（algorithm）。

Donald E. Knuth 在他的著作《计算机程序设计艺术》中将算法的特征归纳为如下特点。

（1）输入：一个算法有零个或多个输入。

（2）输出：一个算法应有一个或多个输出，输出是算法计算的结果。

（3）确定性：算法的描述必须无歧义，每个步骤应当是确定的，而不能是含糊的、模棱两可的，以保证算法的实际执行结果是准确符合要求或期望的，通常要求实际运行结果是确定的。

（4）有限性：一个算法应包含有限的操作步骤，而不能是无限的。

（5）有效性：又称可行性，算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。

1.3.1 算法举例

对于计算机程序设计人员，必须会设计算法，并根据算法编写程序。本节给出一些简单问题的算法。

【例 1-4】 求解 5!=1×2×3×4×5。

解答：

（1）对于该问题，最原始方法按照如下步骤求解。

步骤 1：先求 1×2，得到结果 2；

步骤 2：将步骤 1 得到的乘积 2 乘以 3，得到结果 6；

步骤 3：将步骤 2 得到的乘积 6 再乘以 4，得到结果 24；

步骤 4：将步骤 3 得到的乘积 24 再乘以 5，得到结果 120。

这样的算法虽然正确，但太烦琐，不适合编写计算机程序。

（2）改进后的算法如下。

设变量 p 为被乘数，变量 i 为乘数。

S1：令 p=1；

S2：令 i=2；

S3：使 p×i，乘积仍然放在变量 p 中，可表示为 p×i→p；

S4：使 i 的值加 1，即 i+1→i；

S5：如果 i≤5，返回重新执行步骤 S3 以及其后的 S4 和 S5；否则，算法结束。

该算法不仅正确，而且是较好的计算机算法，因为计算机是高速运算的自动机器，实现循环（S3、S4 和 S5）非常容易。

（3）对于改进后的算法，很容易进行扩展。

如果计算 100!，只需将 S5 中的“i≤5”改成“i≤100”即可。

如果计算 1×3×5×7×9，算法只需改动如下。

S2：令 i=3；

S4：i+2→i；

S5：若 i≤11，返回 S3，否则结束。

【例 1-5】 求解 $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\cdots+\frac{1}{99}-\frac{1}{100}$。

解答：

（1）最原始方法按照如下步骤求解。

步骤 1：求 $1-\frac{1}{2}$，得到 $\frac{1}{2}$。

步骤 2：将 $\frac{1}{2}+\frac{1}{3}$，得到 $\frac{5}{6}$。

步骤 3：将 $\frac{5}{6}-\frac{1}{4}$，得到 $\frac{7}{12}$。

……

这样的算法不适合编写计算机程序。

（2）改进后的算法如下。

设变量 sign 为运算符号（+或者−），sum 为运算结果，i 为序列 2, 3, 4,…, 100 中的一项：

S1：sign=1；

S2：sum=1；

S3：i=2；

S4：sign=(−1)×sign；

S5：item=sign×(1/i)；

S6：item =sum+item；

S7：i=i+1；

S8：若 i≤100，返回 S4；否则结束。

1.3.2　算法的表示方法

为了表示一个算法，可以采用不同的方法。常用的方法有自然语言、流程图、N-S 流程图、伪代码、计算机语言等。

1．用自然语言表示算法

自然语言就是人们常用的语言，如汉语或英语。用自然语言表示通俗易懂，但文字冗长，容易出现歧义，而且不方便用自然语言描述包含分支和循环的算法。除了很简单的问题，一般不用自然语言表示算法。

【例 1-6】 用自然语言描述、计算并输出 $y=\sqrt{x}$ 的算法。

解答：自然语言描述如下。

（1）输入 x。

（2）判断 x 是否小于零，如果成立，则输出错误信息；否则，计算 x 的平方根并赋值给 y，输出 y 的值。

2．用流程图表示算法

流程图是用一些约定的几何图形来描述算法，用流程图表示算法直观形象，易于理解。流程图的符号与意义如图 1-3 所示。

一个流程图包括表示相应操作的框、带箭头的流程线、框内外必要的文字说明等部分。

【例 1-7】 将求 5!= 1×2×3×4×5 的算法用流程图表示。

解答：用流程图表示求 5!的算法如图 1-4 所示。

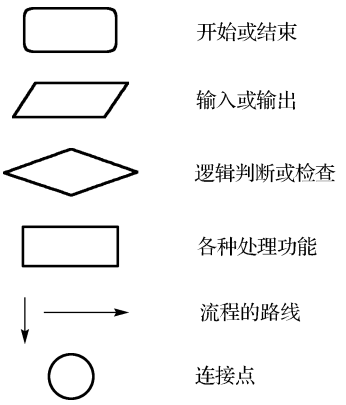


图 1-3　流程图的符号与意义

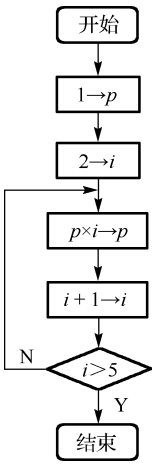


图 1-4　求 5!的算法的流程图表示

3．用 N-S 流程图表示算法

美国学者 I. Nasii 和 B. Shneiderman 于 1973 年提出了一种新的流程图形式，将全部算法写在一个矩形框内，完全去掉了带箭头的流程线，这种流程图称为 N-S 流程图。

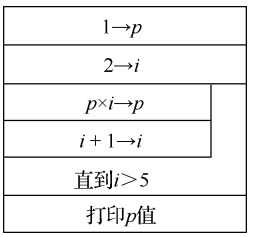


图 1-5　求 5!的算法的 N-S 流程图表示

【例 1-8】 将求 5!= 1×2×3×4×5 的算法用 N-S 流程图表示。

解答：用 N-S 流程图表示求 5!的算法如图 1-5 所示。

4．用伪代码表示算法

伪代码使用介于自然语言和计算机语言之间的文字和符号来描述算法，具有自然语言灵活的特点，同时又接近于程序设计语言的描述。

【例 1-9】 将求 5!= 1×2×3×4×5 的算法用伪代码表示。

解答：用伪代码表示求 5!的算法如下所示。

开始

　　置 *p* 的初值为 1

　　置 *i* 的初值为 2

　　当 *i*<=5，执行下面的操作：

　　　　使 *p*=*p*×*i*

　　　　使 *i*=*i*+1

　　　　(循环体到此结束)

　　打印 *p* 的值

结束

5．用计算机语言表示算法

我们的任务是用计算机解决问题，即用计算机实现算法，用计算机语言表示算法必须严格遵循所用语言的语法规则。

【例 1-10】 将求 5!= 1×2×3×4×5 的算法用计算机语言表示。

解答：用计算机语言表示求 5!的算法如下所示。

```
int product=1;
int i=2;

while (i<=5) {
    product=product*i;
    i=i+1;
}

printf("%d",product);
```

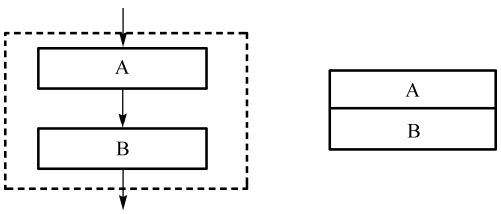
1.3.3　基本程序结构和流程图

已经证明：任何复杂的问题都可以用三种基本结构组成的程序完成。这三种基本结构是：①顺序结构，按指令的顺序依次执行；②选择结构，根据判别条件有选择地改变执行流程；③循环结构，有条件地重复地执行某个程序块。

三种基本结构有如下共同特点：①只有一个入口和出口；②结构内部的每部分都有机会被执行到；③结构内不存在“死循环”。

1．顺序结构

依次顺序地执行程序语句，如图 1-6 所示。

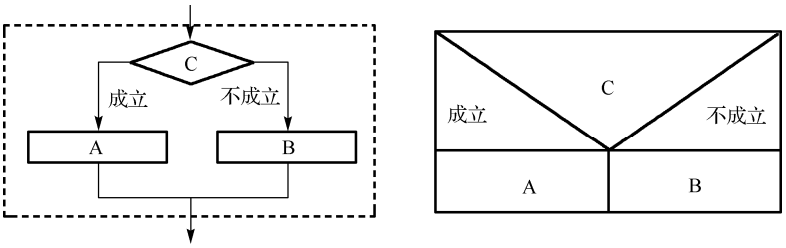


(a) 流程图　　(b) N-S流程图

图 1-6　顺序结构

2．选择结构

首先判断条件 C，若条件 C 成立，执行 A 程序块，否则执行 B 程序块，如图 1-7 所示。



(a) 流程图　　(b) N-S流程图

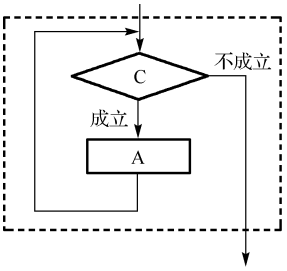
图 1-7　选择结构

3．循环结构

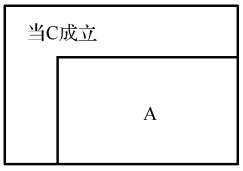
循环结构可以减少源程序书写的工作量，用来描述重复执行的某段程序块。循环结构分为当型循环和直到型循环。

（1）当型循环结构：当条件 C 成立时，就执行 A 程序块，然后再次判断条件 C 是否成立，直到条件 C 的值不成立才向下执行，如图 1-8 所示。

（2）直到型循环结构：先执行 A 程序块，然后判断条件 C 是否成立，若条件 C 不成立 ，再次执行 A 程序块，直到条件 C 成立时才向下执行，如图 1-9 所示。

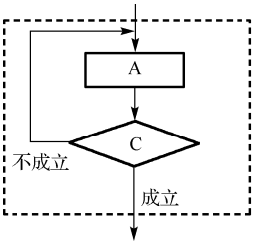


(a) 流程图

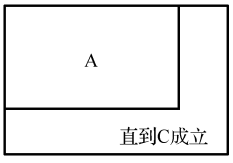


(b) N-S流程图

图 1-8　当型循环结构



(a) 流程图



(b) N-S流程图

图 1-9　直到型循环结构

1.4　数制及进制转换

数可以用两种方法来表示：①按“值”表示。在选定的进位制中表示出该数对应的值，称为进位制数；②按“形”表示。按照一定的编码方法，表示出该数特定的形式，称为编码制数。数的这两种表示方法涉及数制与编码，本节讲述数制及进制转换，下一节讲述数值编码。

1.4.1 基本进位制

1．十进制

十进制是最常用的数制，其特点如下。

（1）数码，表示数的符号。十进制数用 0、1、2、3、4、5、6、7、8、9 等 10 个符号表示，遵循“逢十进一”的原则，即 9+1=10。

（2）基数，为数码的个数。十进制数的基数是 10，即采用 10 个基本数码，任何一个十进制数都可以用 10 个数码按一定规律排列起来表示。

（3）位权，每一位所具有的值。0～9 这 10 个数可以用一位数码表示，10 以上的数则要用两位以上的数码表示。如数 12，右边的“2”为个位数，左边的“1”为十位数，也就是$12=1\times10^1+2\times10^0$。因此，每一数码处于不同位置时，它代表的数值是不同的，即不同的数位有不同的位权。

十进制数表示的数值等于其各位加权系数之和，如数 9876 可写为

$$9876=9\times10^3+8\times10^2+7\times10^1+6\times10^0$$

其中，每位的位权分别为 10^3、10^2、10^1、10^0。

一般地，任意一个 n 位十进制正整数 $(D)_{10}$ 均可表示为：

$$(D)_{10}=k_{n-1}\times10^{n-1}+k_{n-2}\times10^{n-2}+\cdots+k_1\times10^1+k_0\times10^0=\sum_{i=0}^{n-1}k_i\times10^i$$

其中，下脚注 10 表示括号内的数是一个十进制数，也可以用下脚注 D（Decimal）表示。k_i 是第 i 位的系数，为 0～9 当中的某一个数。如果一个数的整数部分有 n 位，小数部分有 m 位，则 i 的取值为 $-m$ 到 $n-1$ 之间（含 $-m$ 和 $n-1$）的所有整数。

【例 1-11】 十进制数 76543 和 2015.21 可表示如下：

$$(76543)_{\mathrm{D}}=(76543)_{10}=7\times10^4+6\times10^3+5\times10^2+4\times10^1+3\times10^0$$

$$(2015.21)_{\mathrm{D}}=(2015.21)_{10}=2\times10^3+0\times10^2+1\times10^1+5\times10^0+2\times10^{-1}+1\times10^{-2}$$

2．二进制

二进制是计算机中广泛采用的一种数制，其特点如下。

（1）数码。二进制数用 0 和 1 两个符号表示，遵循“逢二进一”的原则，即 1+1=10。

（2）基数。二进制数的基数是 2，即采用两个基本数码，任何一个二进制数都可以用两个数码按一定规律排列起来表示。

（3）位权。二进制的各位位权分别为 2^0、2^1、2^2、2^3……任意一个 n 位二进制正整数可表示为：

$$(D)_2=k_{n-1}\times2^{n-1}+k_{n-2}\times2^{n-2}+\cdots+k_1\times2^1+k_0\times2^0=\sum_{i=0}^{n-1}k_i\times2^i$$

其中，下脚注 2 表示括号内的数是一个二进制数，也可以用下脚注 B（Binary）表示。

二进制数表示的数值等于其各位加权系数之和，例如：

$$(1011)_{\mathrm{B}}=(1011)_2=1\times2^3+0\times2^2+1\times2^1+1\times2^0=(11)_{10}$$

二进制数只有两个数码 0 和 1，可以用任何具有两个不同稳定状态的元件来表示，如晶体管的饱和与截止，继电器接点的闭合与断开等。只要规定其中一种状态表示 1，则另一种状态就表示 0，这样就可以表示二进制数了。因此，二进制的实现装置简单可靠，容易用二极管、晶体管等电子器件来实现，具有十进制无法具备的优点，因此广泛应用在计算机系统中。

二进制数的位数通常很多，不便于书写和记忆，例如，要表示十进制数 1111，若用二进制数表示则为 10001010111，若用八进制数表示则为 2127，若用十六进制数表示则为 457，因此在计算机程序设计中，常使用八进制数或十六进制数来表示二进制数。

3．十六进制

中国历史上曾在质量单位上使用过十六进制，如 16 两为一斤（1 斤=500g）。现在十六进制普遍应用在计算机领域。十六进制数特点如下。

（1）数码。十六进制数用 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 等 16 个符号表示（其中数值 10～15 分别用 A～F 来表示），遵循“逢十六进一”的原则，即 F+1=10。

（2）基数。十六进制数的基数是 16，即采用 16 个基本数码，任何一个十六进制数都可以用 16 个数码按一定规律排列起来表示。

（3）位权。十六进制的各位位权分别为 16^0、16^1、16^2、16^3……任意一个 n 位十六进制正整数可表示为：

$$(D)_{16}=k_{n-1}\times16^{n-1}+k_{n-2}\times16^{n-2}+\cdots+k_1\times16^1+k_0\times16^0=\sum_{i=0}^{n-1}k_i\times16^i$$

其中，下脚注 16 表示括号内的数是一个十六进制数，也可以用下脚注 H（Hexadecimal）表示。

十六进制数表示的数值等于其各位加权系数之和，例如：

$$(\mathrm{FE98})_{\mathrm{H}}=(\mathrm{FE98})_{16}=15\times16^3+14\times16^2+9\times16^1+8\times16^0=(65176)_{10}$$

4．八进制

在计算机中，八进制数有时可以取代十六进制数，如 Linux 系统的文件权限设置。十六进制数除了 0～9 之外，要用到 A～F 等符号，八进制数不必用数字以外的符号来表示。

八进制的特点如下。

（1）数码。八进制数用 0、1、2、3、4、5、6、7 等 8 个符号表示，遵循“逢八进一”的原则，即 7+1=10。

（2）基数。八进制数的基数是 8，即采用 8 个基本数码，任何一个八进制数都可以用 8 个数码按一定规律排列起来表示。

（3）位权。八进制的各位位权分别为 8^0、8^1、8^2、8^3……任意一个 n 位八进制正整数可表示为：

$$(D)_8=k_{n-1}\times8^{n-1}+k_{n-2}\times8^{n-2}+\cdots+k_1\times8^1+k_0\times8^0=\sum_{i=0}^{n-1}k_i\times8^i$$

其中，下脚注 8 表示括号内的数是一个八进制数，也可以用下脚注 O（Octal）表示。

八进制数表示的数值等于其各位加权系数之和，例如：

$$(7650)_{\mathrm{O}}=(7650)_8=7\times8^3+6\times8^2+5\times8^1+0\times8^0=(4008)_{10}$$

5．R 进制

综上所述，对一个 R（$R\geqslant2$）进制数，基数为 R，可以用 R 个符号表示一个 R 进制数，遵循“逢 R 进一”的原则。R 进制的各位位权分别为 R^0、R^1、R^2、R^3……任意一个 n 位 R 进制正整数可表示为：

$$(D)_{\mathrm{R}}=k_{n-1}\times R^{n-1}+k_{n-2}\times R^{n-2}+\cdots+k_1\times R^1+k_0\times R^0=\sum_{i=0}^{n-1}k_i\times R^i$$

其中，下脚注 R 表示括号内的数是一个 R 进制数，k_i 表示第 i 位的系数，R^i 为第 i 位的位权。R 进制数表示的数值等于其各位加权系数之和。

1.4.2 进制数间相互转换

1．二进制数、八进制数、十六进制数转换成十进制数

二进制数、八进制数、十六进制数转换成十进制数的方法是按权相加，即求出各位加权系数之和，得到相应的十进制数。

【例 1-12】 二进制数、八进制数、十六进制数转换成十进制数示例。

$$(111011)_2 = 1\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = (59)_{10}$$

$$(F2E.A8)_{16} = 15\times16^2 + 2\times16^1 + 14\times16^0 + 10\times16^{-1} + 8\times16^{-2} = (3886.65625)_{10}$$

$$(1234)_8 = 1\times8^3 + 2\times8^2 + 3\times8^1 + 4\times8^0 = (668)_{10}$$

2．十进制数转换成二进制数、八进制数、十六进制数

将十进制数的整数部分转换为二进制数、八进制数、十六进制数时，可以采用“除 R 倒取余数”法，R 代表所要转换成的数制的基数，步骤如下。

步骤 1：把该十进制数的整数部分除以基数 R，取余数，即为最低位的数码 k_0。

步骤 2：将前一步得到的商再除以基数 R，再取余数，即得次低位的数码 k_1。

步骤 3：重复以上过程，直到商为 0 结束，最后得到的余数即为最高位数的数码 k_{n-1}。

将十进制数小数部分转换为二进制数、八进制数、十六进制数时，可以采用“乘 R 顺取整数”法，R 代表所要转换成的数制的基数，步骤如下。

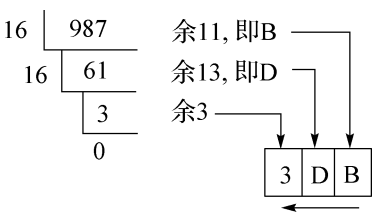
步骤 1：把该十进制数的小数部分乘以基数 R，取整数，即得小数的最高位数码 k_{-1}。

步骤 2：将前一步得到的乘积的小数部分再乘以基数 R，再取整数，即得小数的次低位数码 k_{-2}。

步骤 3：重复以上过程，直到乘积为零，或者达到所需求的精度为止。最后一次取的整数为小数的最低位 k_{-m}。

【例 1-13】 把十进制数 987 转换成十六进制数。

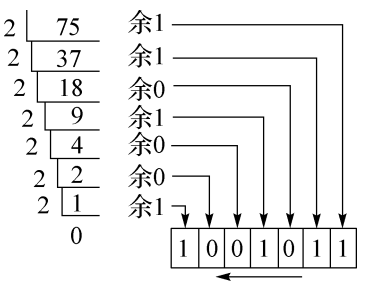
解答：



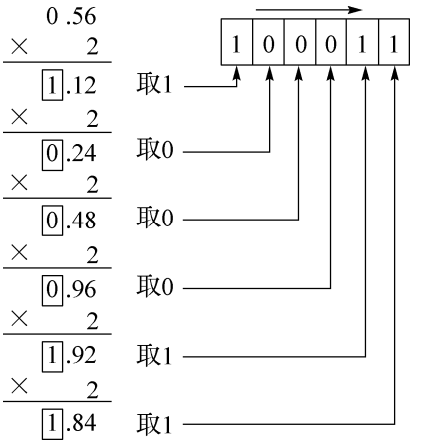
因此，$(987)_{10} = (3DB)_{16}$。

【例 1-14】 将十进制数 75.56 转换为二进制数（误差 $\varepsilon < 1/2^6$）。

解答：（1）整数部分，采用“除 2 倒取余数”法转换。



（2）小数部分，采用“乘 2 顺取整数”法转换。



因此，$(75.56)_{10}=(1001011.100011)_2$ 转换到第 6 位小数时误差 $\varepsilon<1/2^6$ 。

3．二进制数与十六进制数的相互转换

（1）将二进制正整数转换为十六进制数

将二进制数从右向左开始，每 4 位分为一组（不足 4 位左补 0），每组都相应转换为一位十六进制数。

【例 1-15】 将二进制数 11010110111101 转换为十六进制数。

解答：

二进制　0011　0101　1011　1101

十六进制　3　5　B　D

因此，$(110101101111101)_2=(35BD)_{16}$ 。

（2）将十六进制正整数转换为二进制数

将十六进制数的每一位转换为相应的 4 位二进制数即可。

【例 1-16】 将十六进制数 F9E8 转换为二进制数。

解答：

十六进制　F　9　E　8

二进制　1111　1001　1110　1000

因此，$(F9E8)_{16}=(1111100111101000)_2$。

4．二进制数与八进制数的相互转换

（1）将二进制正整数转换为八进制数

将二进制数从右向左开始，每 3 位分为一组（不足 3 位左补 0），每组都相应转换为一位八进制数。

【例 1-17】 将二进制数 11010110111101 转换为八进制数。

解答：

二进制　011　010　110　111　101

八进制　3　2　6　7　5

因此，$(11010110111101)_2=(32675)_8$。

（2）将八进制正整数转换为二进制数

将八进制数的每一位转换为相应的 3 位二进制数即可。

【例 1-18】 将八进制数 7654 转换为二进制数。

解答：

八进制　7　6　5　4

　　↓　↓　↓　↓

二进制　111 110 101 100

因此，$(7564)_8=(111110101100)_2$。

为了方便转换，表 1-1 分别给出了二进制数/十六进制数和二进制数/八进制数的换算表。

表 1-1　二进制数/十六进制数和二进制数/八进制数的换算表

二 进 制 数	十六进制数	二 进 制 数	十六进制数	二 进 制 数	八 进 制 数
0000	0	1000	8	000	0
0001	1	1001	9	001	1
0010	2	1010	A	010	2
0011	3	1011	B	011	3
0100	4	1100	C	100	4
0101	5	1101	D	101	5
0110	6	1110	E	110	6
0111	7	1111	F	111	7

1.5 数 值 编 码

编码是信息从一种形式或格式转换为另一种形式或格式的过程。常用的编码有两类：一类是十进制编码，如美国信息交换标准代码（ASCII）；另一类是二进制编码，如数的机器码表示。

1.5.1 美国信息交换标准代码（ASCII）

ASCII（American Standard Code for Information Interchange，美国信息交换标准代码）是基于拉丁字母的一套计算机编码系统，主要用于显示现代英语。ASCII 是现今最通用的单字节编码系统，被国际标准化组织（ISO）选定为一种国际通用的代码，广泛用于通信和计算机中。

标准 ASCII 码也称为基础 ASCII 码，使用 7 位二进制数来表示所有的大写和小写字母、数字 0～9、标点符号及一些特殊控制字符。例如，空格的 ASCII 码为二进制数 0100000；字符 0 的 ASCII 码为二进制数 0110000；字符 a 的 ASCII 码为二进制数 1100001；字符 A 的 ASCII 码为二进制数 1000001；控制符换行的 ASCII 码为二进制数 0001010；控制符回车的 ASCII 码为二进制数 0001101。

附录 B.1 给出了 ASCII 代码的对照表。

1.5.2 数的机器码表示

在计算机中为了表示正、负数，在数的最高位前设置一个符号位，并规定符号位为“0”时表示该数为正数；符号位为“1”时表示该数为负数。这种带有符号位的数称为机器数，机器数有原码、反码、补码三种表示形式。

1．原码

二进制原码最高位为符号位，其余各位为数值本身的绝对值，又称为“符号+绝对值”表示法。符号位“0”表示正数，符号位“1”表示负数。

例如，$(+99)_{10}$的带符号位 8 位原码表示为$(01100011)_{原}$，其中最高位的 0 代表正数符号的符号位，后 7 位是代表 99 这个数的二进制表示法。$(-99)_{10}$的原码可表示为$(11100011)_{原}$，其中最高位的 1 代表负数符号的符号位，后 7 位代表–99 这个数的绝对值的二进制表示法。

零的原码有两种表示形式：$(+0)_{原}$ = 00000000，$(-0)_{原}$ = 10000000，机器遇到这两种情况都当作 0 处理。

原码表示法简单易懂，但在进行加、减法运算时，符号位不能直接参加运算，而是要分别计算符号位和数值位。当两数相加时，如果是同号，则数值相加；如果是异号，则要进行减法运算，此时还要比较两数的绝对值大小，先用大数减去小数，最后还要判断符号位，这样会导致运算速度将低。

为了解决该问题，引入数的反码和补码表示法。

2．反码

引入反码是便于求负数的补码。二进制反码表示法的规则是：正数的反码与原码相同，负数的反码是符号位为 1，数值是对应原码各位取反。

例如，$(+99)_{10}$ 的带符号位 8 位反码表示为$(01100011)_{反}$，与原码相同。$(-99)_{10}$ 的反码可表示为$(10011100)_{反}$，其中最高位的 1 代表负数，后 7 位是–99 的原码各位取反。

零的反码有两种表示形式：$(+0)_{反}=00000000$，$(-0)_{反}=11111111$。

3．补码

二进制补码表示法的规则是：正数的补码与原码相同，负数的补码是符号位为 1，数值位逐位取反（求其反码），然后在最低位对整个数加 1。

例如，$(+99)_{10}$ 的带符号位 8 位补码表示为$(01100011)_{补}$，与原码相同。$(-99)_{10}$ 的补码可表示为$(10011101)_{补}$，其中最高位的 1 代表负数，后 7 位是–99 的原码的数值部分各位取反后加 1。

零的补码只有一种形式：$(+0)_{补}=(-0)_{补}=00000000$。

一个负数的二进制补码转换成十进制数的规则是：最高位不变，其余位取反后加 1 得到原码。例如，补码 11111001，取反后为 10000110（最高位不变），然后加 1 得 10000111，即十进制数–7。

采用补码表示法进行二进制数的加、减法运算，符号位可以和数值位一起参与运算，减法运算可转换为加法运算，得到的运算结果是补码的形式。两个补码的符号位和数值部分产生的进位相加得到的和，就是运算结果的符号。在用补码计算时，要注意补码的位数必须足够多，能表示运算的绝对值，否则会得到错误的运算结果。

【例 1-19】 若 $a=13$，$b=21$，计算 $a-b$ 的值。

解答：用补码的计算方法为：$a-b=a+(-b)=(a)_{补}+(-b)_{补}=(13)_{补}+(-21)_{补}$

因为$(a)_{补}=00001101$，$(-21)_{补}=11101011$，则 $a-b=11111000$。

结果为补码，转换为原码为 10001000，所以 $a-b$ 的结果为十进制数–8。

表 1-2 给出了几个数的原码、反码、补码的对照表（用 8 位表示）。

表 1-2　原码、反码、补码对照表

十 进 制 数	二 进 制 数		
	原　码	反　码	补　码
+0	00000000	00000000	00000000
–0	10000000	11111111	00000000
+1	00000001	00000001	00000001
–1	10000001	11111110	11111111

（续表）

十 进 制 数	二 进 制 数		
	原　码	反　码	补　码
+7	00000111	00000111	00000111
–7	10000111	11111000	11111001
数值范围	11111111～01111111 (–127～–0, +0～+127)	10000000～01111111 (–127～–0, +0～+127)	10000000～01111111 (–128～0～+127)

1.6 C 语言概述

1.6.1 C 语言简介

C 语言是一种通用的编程语言，广泛用于系统软件与应用软件的开发。C 语言具有高效、灵活、功能丰富、表达力强和可移植性好等特点，在程序员中备受青睐，成为近几十年来使用最广泛的编程语言之一。目前，C 语言编译器普遍存在于各种不同的操作系统中，如 Microsoft Windows、Mac OS X、Linux、UNIX 等。C 语言的设计影响了众多后来的编程语言，如 C++、Objective-C、Java、C#等。

1．发展过程

C 语言的发展过程大致可分为三个阶段：Old Style C、C89 和 C99。Old Style C 指 Ken Thompson 和 Dennis Ritchie 最初发明 C 语言的标准。C 语言源于 BCPL 语言，后者由 Martin Richards 于 1967 年左右所设计。1970 年，Ken Thompson 为运行在 PDP-7 上的首个 UNIX 系统设计了一个精简版的 BCPL 语言，被称为 B 语言。在 PDP-11 计算机出现后，Dennis Ritchie 与 Ken Thompson 着手将 UNIX 系统移植到 PDP-11 上，由于 B 语言自身的特点不适合，为此 Dennis Ritchie 与 Ken Thompson 以 B 语言为基础，在贝尔实验室设计和开发出来。

1973 年，UNIX 操作系统内核正式用 C 语言改写，这是 C 语言第一次应用在操作系统的开发上。1975 年 C 语言开始移植到其他计算机中，Stephen C. Johnson 实现了一套“可移植编译器”，从此，C 语言在大多数计算机上被使用，从最小的微型计算机到 CRAY-2 超级计算机。1978 年，Dennis Ritchie 和 Ken Thompson 合作出版了《C 程序设计语言》，书中介绍的 C 语言标准也被 C 语言程序员称为“K&R C”。

1989 年，C 语言被美国国家标准协会（ANSI）标准化，扩展 K&R C，增加了一些新特性，被称为 C89，是最早的 C 语言规范。

1990 年，国际标准化组织（ISO）成立工作组规定国际标准的 C 语言，通过对 ANSI 标准的少量修改，最终制定了 ISO 9899:1990，又称为 C90。随后，ANSI 亦接受国际标准 C 语言。

C89 是目前最广泛采用的 C 语言标准，大多数编译器都完全支持 C89。经典的 C 语言教材《C 程序设计语言》（第二版）就是基于这个版本的。

在 ANSI 的标准确立后，C 语言的规范在一段时间内没有大的变动。《标准修正案一》在 1994 年为 C 语言创建了一个新标准，支持更多的国际字符集，这个标准被称为 C99（ISO 9899:1999）。C99 被 ANSI 于 2000 年 3 月采用，但目前仍没有得到广泛支持。

2011 年，ISO 正式发布了 C 语言的新标准 C11，之前被称为 C1X，官方名称为 ISO/IEC 9899:2011。

2．C 语言特点

由于 C 语言的强大功能和各方面的优点，在各类大中小和微型计算机上得到了广泛的使用，成为现在最优秀的程序设计语言之一。C 语言有如下特点。

（1）C 语言简洁，紧凑，使用方便，灵活。C 语言语法限制不太严格，程序设计自由度较大。

（2）运算符丰富。C 语言共有 34 种，把括号、赋值、逗号等都作为运算符处理，从而使运算类型极为丰富，可以实现其他高级语言难以实现的运算。

（3）数据结构类型丰富。

（4）具有结构化的控制语句。

（5）C 语言允许直接访问物理地址，能进行位（bit）操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作。

（6）用 C 语言写的程序可移植性较好，生成目标代码质量高，程序执行效率高。

1.6.2　C 语言程序示例

以下通过两个例子来说明 C 语言源程序的基本部分和结构特点。

【例 1-20】 程序 1-1：在屏幕上输出“hello, world!”字符串，通常是初学编程语言时的第一个简单程序。

```
#01: //程序 1-1
#02: #include <stdio.h>
#03: int main(void){
#04:    printf("hello, world!\n");
#05:    return 0;
#06: }
```

程序解释：

#01：程序注释，以“//”开始的行为程序注释，不产生编译代码。

#02：编译预处理指令，包含标准输入/输出头文件，用来提供输入/输出功能。

#03：主函数的函数定义。每个 C 语言源程序都必须有且只能有一个 main 函数。

#04：函数调用语句，printf( )函数是由系统定义的标准函数，可在程序中直接调用，功能是把要输出的内容输出到屏幕显示。

#05：程序运行结束。

程序运行结果如下：

```
hello, world!
```

【例 1-21】 程序 1-2：输入一个实数值，求出正弦值后在屏幕上输出。

```
#01: //程序 1-2
#02: #include <stdio.h>
#03: #include <math.h>
#04: int main(){
#05:    double x = 0.0;
#06:    double s = 0.0;
#07:
#08:    printf("Input a number:");
#09:    scanf("%lf",&x);
#10:    s = sin(x);
#11:    printf("sin(%lf)=%lf\n",x,s);
```

```
#12:
#13:     return 0;
#14: }
```

程序解释：

#03：math.h 为数学运算函数的头文件，用 include 指令包含该头文件，以在程序中使用正弦函数。

#05, #06：定义两个实数变量 x 和 s，程序后面将使用。

#08：显示提示信息，提示用户输入一个数值。

#09：从键盘获得一个实数，保存在 x 中。

#10：求 x 的正弦，结果保存在变量 s 中。

#11：显示程序的运算结果。

程序运行结果如下（下画线表示用户输入内容）：

```
Input a number:10
sin(10.000000)=-0.544021
```

被#include 指令包含的文件通常是由系统提供的，其扩展名为“.h”，因此也称为头文件。C 语言的头文件中包括各个标准库函数的函数原型，在程序中调用一个库函数时，必须包含该函数原型所在的头文件。

1.6.3 C 语言程序编译与执行

对于高级语言程序，需要使用编译器/解释器将其转换成机器代码后才能在计算机中运行。解释器像一位“中间人”，每次执行程序时都要先转换成另一种语言再执行，因此解释器的程序运行速度比较缓慢。常见的解释程序有 BASIC、python、LISP 等。相对地，编译器一次将程序翻译成机器码，可以多次执行而无须再编译，其生成程序无须依赖编译器就可执行，程序运行速度比较快。常见的编译程序有 C、C++、Pascal 等。

如图 1-10 所示，对于 C 语言程序，从编辑源代码到程序运行需要 4 个步骤。

（1）编辑代码。输入编辑程序源代码，生成源程序或头文件。源程序是用户创建的文件，以“.c”为文件扩展名保存。头文件是含有函数的声明和预处理语句，用于帮助访问外部定义的函数，扩展名为“.h”。

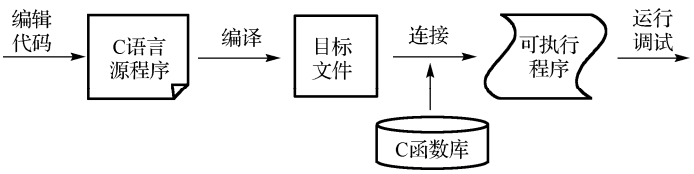


图 1-10 C 语言程序编译与执行

（2）编译。对语法进行分析，翻译生成目标程序。目标文件是编译器的输出结果，常见扩展名为“.o”或“.obj”。

（3）连接。与其他目标程序或库连接装配，生成可执行程序。可执行程序是连接器的输出结果，常见扩展名为“.exe”。

（4）运行，对程序运行调试，输出结果。

通常把程序的编译和连接统称为编译阶段。

上机实验：熟悉 C 语言开发环境

本次实验熟悉 C 语言的开发环境，通过运行简单的 C 语言程序，初步了解 C 语言源程序的特点。了解在 Windows 系统上如何编辑、编译和运行一个 C 语言程序。

Microsoft Visual C++（简称 VC++）是微软公司的集成开发工具，可提供 C 语言、C++及 C++/CLI 等编程语言。VC++提供了一个集源程序编辑、代码编译与调试于一体的开发环境，称为集成开发环境，能够提高软件开发效率。程序员通过 VC++可以访问 C 语言源代码编辑器、资源编辑器，使用内部调试器等。VC++被整合在 Visual Studio 之中，但仍可单独安装使用。

下面使用 VC++编写并运行一个简单的程序。

（1）运行 VC++，选择“文件”菜单中的“新建”命令，打开“新建”对话框，选择“文件”选项卡，在左边的列表框中选择“C++ Source File”项，在右边的“文件名”编辑框中输入文件名“addtest.c”，文件名可以任意取，但是要求文件扩展名为“.c”，以说明是一个 C 语言程序。可以选择合适的位置保存该源文件，如图 1-11 所示。

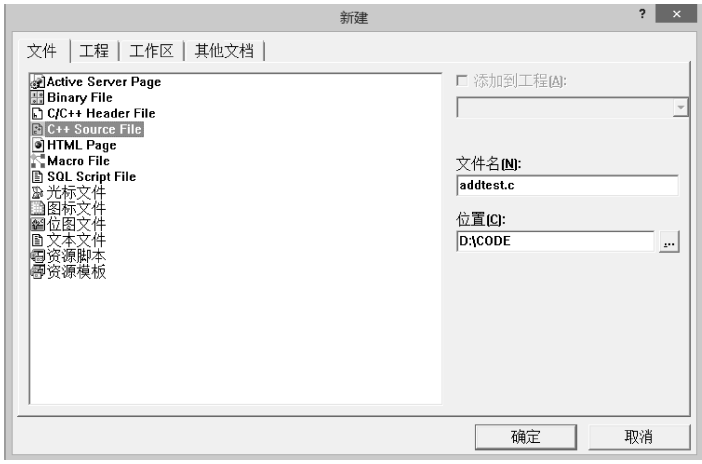


图 1-11　“新建”对话框设置

（2）在窗口中编辑如下的源程序。

```
//calculate the sum of a and b
#include <stdio.h>
int add(int,int);
int main(){
    int a,b,sum;
    a=1;
    b=2;
    sum=add(a,b);
    printf("%d + %d = %d\n",a,b,sum);
    return 0;
    }
```

```
        int add(int x,int y){
        return(x+y);
    }
```

（3）选择“组建”菜单中的“组建”命令，对程序进行编译连接，在编译过程中，VC++会生成一个同名的工作区，不用理会。

如果程序没有错误，则在下方的信息窗口中显示：“-0 error(s)，0 warning(s)”，得到可执行程序addtest.exe。

有时会出现一些警告性信息（warning），不影响程序的执行。如果程序中有 N 个致命性错误（error），则会提示：“-N error(s)，M warning(s)”，此时，拖动信息窗口右侧的滚动条，可以看到所有编译器给出的错误提示，包括错误在第几行，以及是什么错误。双击某行出错信息，程序窗口中会用箭头指示对应出错位置。根据信息窗口中的错误提示修改错误，修改好后，保存并重新编译，直到无错误为止。

（4）选择“组建”菜单中的“执行”命令，运行程序，观察结果。

（5）完成后，选择“文件”菜单中的“关闭工作区”命令，关闭程序工作区。

习　题

1．用流程图表示给出求闰年问题的算法。

判断 1～2100 年中的每一年是否为闰年，将结果输出。

闰年的条件：（1）能被 4 整除，但不能被 100 整除的年份；（2）能被 100 整除，又能被 400 整除的年份。

2．用流程图表示给出求素数问题的算法。

对一个大于或等于 2 的正整数，判断它是不是一个素数。

素数的条件：除了 1 和它本身外，不能被其他自然数整除，也称质数。

3．用流程图表示求解如下问题的算法。

（1）输入三个数 a、b、c，分别输出最大的数和最小的数。

（2）求 $1+2+3+\cdots+99+100$ 的和。

（3）求一元二次方程 $ax^2+bx+c=0$ 的根。注意考虑 $\Delta=b^2-4ac$ 大于、等于、小于零的情况。

4．将下列二进制数、八进制数和十六进制数转换为十进制数。

二进制数：（1）1111，（2）10101010；

八进制数：（1）1111，（2）76543210；

十六进制数：（1）1111，（2）FFFF，（3）FEDCBA。

5．将下列十进制数分别转换为二进制数、八进制数和十六进制数。

（1）1234；（2）9999；（3）255；（4）10.25。

6．将下列二进制数分别转换为八进制数和十六进制数。

（1）1111；（2）10101010；（3）1111111；（4）11111111。

7．将下列八进制数和十六进制数转换为二进制数。

八进制数：（1）1111，（2）76543210；

十六进制数：（1）1111，（2）FFFF，（3）FEDCBA。

8．求出下列十进制数的原码、反码和补码（用 8 位二进制数表示）。

（1）1；（2）–1；（3）63；（4）–63；（5）127；（6）–127。