高等院校计算机应用系列教材

C 语言程序设计

(第二版)(微课版)

赵　彩　杨宏霞　主　编
丁　凰　许大炜　田文文　副主编

清華大學出版社

北　京

内 容 简 介

本教材围绕应用型人才培养目标，结合学生的思维方式、理解能力以及后续课程中的应用需求进行编写。本书分为 9 章，主要内容包括：C 语言概述，数据类型、运算符及表达式，常用的输入输出函数，程序控制结构，数组，函数，指针，结构体与共用体，以及文件操作等。书中设有“编程经验”模块，融入了各种编程小技巧，可帮助读者在学习过程中少走弯路，快速掌握 C 语言技术精髓，提升程序开发技能。

作为一本微课教材，本书配备了 121 集与书中实例同步的微课视频，学生可以跟着视频学 C 语言，高效、快捷。此外，本书还配套了丰富的教学资源，如实例源代码、PPT 教学课件、课后习题答案，方便教师教学和读者自学。同时本书配有同步的实验教材《C 语言程序设计实践教程(第二版)(微课版)》，方便读者深入学习和上机操作。

本书既可以作为高等学校本科及专科学生 C 语言程序设计的教材，也可以作为自学者的参考用书，同时也可供各类计算机等级应试人员复习参考。

本书配套的电子课件、教案、习题答案和实例源代码扫描封底或前言中的二维码即可获取，扫描书中的视频二维码可以直接观看教学视频。

图书在版编目(CIP)数据

C 语言程序设计 ：微课版 / 赵彩, 杨宏霞主编. 2 版. -- 北京 ：清华大学出版社, 2026. 7. -- (高等院校计算机应用系列教材). -- ISBN 978-7-302-71921-2

Ⅰ. TP312.8

中国国家版本馆 CIP 数据核字第 2026J11V51 号

责任编辑：胡辰浩
封面设计：高娟妮
版式设计：妙思品位
责任校对：马遥遥
责任印制：刘　菲

出版发行：清华大学出版社
网　　址：https：//www.tup.com.cn，https：//www.wqxuetang.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 装 者：北京同文印刷有限责任公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**19.25　　**字　　数：**493 千字
版　　次：2021 年 7 月第 1 版　　2026 年 7 月第 2 版　　**印　　次：**2026 年 7 月第 1 次印刷
定　　价：79.80 元

产品编号：115006-01

前　言

C 语言是世界上最为重要、影响最为深远的程序设计语言之一。在 C 语言的基础上衍生出了 C++、Java 等各种极具生产力的语言。在世界编程语言排行榜中，C 语言近二三十年来长期位居流行编程语言的前两位。掌握 C 语言，是深入理解当今计算机系统工作原理的重要途径。

程序设计既是一种技术，也是一项工程。作为一本合格的程序设计教材，不仅要介绍 C 语言的基本语法知识，还要强调程序设计思想方法的培养，并且引入现代软件工程思想进行开发能力的训练。如何解决好这三个方面的衔接问题，并将它们有机地结合起来，是当前程序设计教材需要解决的核心问题与难点。本书在以下几个方面做了补充。

(1) 讲解 C 语言最基本、最常用的内容，重点放在语言本身的难点和程序设计思想、技巧方面；各章节之间密切结合并且以阶梯式前进，使读者在学习的过程中能够循序渐进。

(2) 书中每一章节的实例都配备了教学讲解微视频，使用手机扫描二维码即可观看、学习，能快速引导初学者入门，感受编程的快乐和成就感，从而增强学习的信心。

(3) 第 9 章的综合案例提供完整的软件系统开发过程，将之前每一章的知识点融入其中，使读者学以致用，最终能够开发出一个学生信息管理系统，并且得到开发软件的实践经验。

(4) 每一章的小结和编程经验总结了一些易错的概念和知识点，并介绍软件开发的实际技巧，使读者在学习过程中就能领悟到高质量的 C 语言编程知识。

(5) 本书新增了部分计算机等级考试的经典试题，旨在帮助学生在掌握 C 语言的理论知识与实践能力的同时，也为参加计算机等级考试打下基础。

本书的实例微课为课程教材数字化建设奠定了基础，能够培养学生程序设计的思维以及分析程序的能力，从而提高学生程序设计的综合能力。

欢迎使用本书进入美妙的 C 语言世界。C 语言博大精深，本书力求从实用易懂的角度进行阐述，希望能够抛砖引玉，帮助读者迈向专业的编程实践。

由于作者水平有限，书中难免有不足之处，恳请专家和广大读者批评指正。在编写本书的过程中参考了相关文献，在此向这些文献的作者深表感谢。我们的电话是 010-62796045，邮箱是 992116@qq.com。

本书配套的电子课件、教案、习题答案和实例源代码扫描下方的二维码即可获取，扫描正文中的视频二维码可以直接观看教学视频。

作　者

2026 年 3 月

目　录

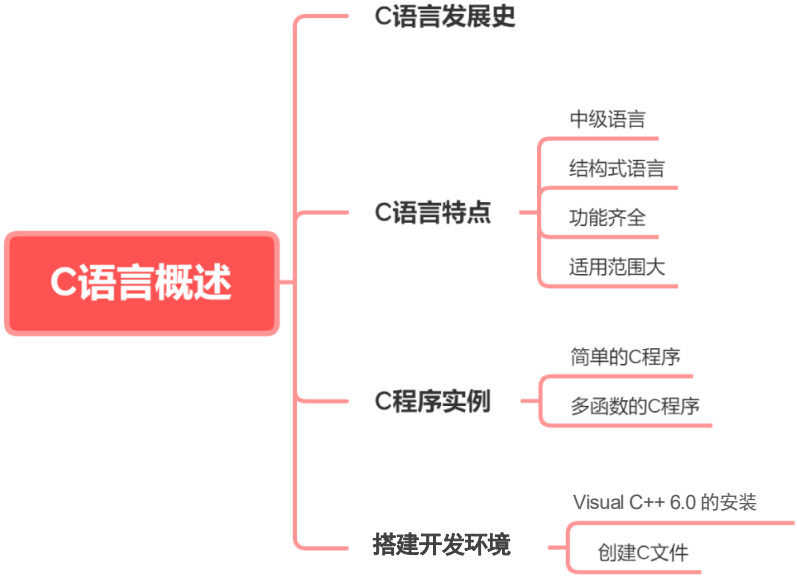
第 1 章

C语言概述

本章概览

本章首先简单介绍 C 语言的发展史、特点和强大应用，然后通过三个简单程序带领大家体验 C 语言的编程过程并了解程序的基本组成。最后详细介绍开发工具 Visual C++ 6.0 的安装与配置。“千里之行，始于足下！”下面就开始我们的 C 语言编程之旅吧！

知识框架



1.1 C 语言发展史

1.1.1 程序语言简述

1. 机器语言

计算机诞生之初，人们只能直接用二进制形式的机器语言写程序。对于人类而言，二进制的机器语言很不方便，用它写程序非常困难，不但工作效率极低，程序的正确性难以保证，而且即便有错误也很难辨认和改正。下面是一台假想计算机上的指令序列：

```
0000000100000000001000 --将存储单元 1000 中的数据装入寄存器 0
0000000100010000 1010 --将存储单元 1010 中的数据装入寄存器 1
0000010100000000000001 --将寄存器 1 中的数据乘到寄存器 0 中的原有数据上
0000000100010000 1100 --将存储单元 1100 中的数据装入寄存器 1
0000010000000000000001 --将寄存器 1 中的数据加到寄存器 0 中的原有数据上
0000001000000000001110 --将寄存器 0 中的数据存入存储单元 1110
```

以上指令序列描述的是计算算术表达式 a × b + c(这里的符号 a、b、c 分别代表地址为 1000、1010 和 1100 的存储单元)，并将结果保存到存储单元 1110 的计算过程。复杂的程序如果用二进制的机器语言来编写的话，将十分困难且难以理解。

2. 汇编语言

为缓解使用机器语言的问题，人们研究出了以符号形式表示且使用起来相对容易的汇编语言。使用汇编语言编写的程序需要经专用软件(汇编系统)加工并翻译成二进制的机器指令后，才能在计算机上运行。

下面是使用某种假想的汇编语言写的程序，它能完成与上述指令系列相同的工作：

```
load 0 a --将存储单元 a 中的数据装入寄存器 0
load 1 b --将存储单元 b 中的数据装入寄存器 1
mult 0 1 --将寄存器 1 中的数据乘到寄存器 0 中的原有数据上
load 1 c --将存储单元 c 中的数据装入寄存器 1
add 0 1 --将寄存器 1 中的数据加到寄存器 0 中的原有数据上
save 0 d --将寄存器 0 中的数据存入存储单元 d
```

汇编语言的每条指令对应一条机器语言指令，但采用了助记的符号名，存储单元也用符号形式的名称来表示，这样每条指令的意义就更容易理解。但是，使用汇编语言编写的程序仍然完全没有结构，而仅仅是使用许多这样的指令堆积形成的长长的指令序列，结构松散。因此，开发复杂程序时，其整体逻辑仍然难以理解。

3. 高级程序语言

为克服低级语言(机器语言与汇编语言)的缺点，1954 年诞生了第一个高级程序语言 FORTRAN，它采用接近于人们习惯使用的自然语言，用类似数学表达式的形式描述数据计算。

FORTRAN 不仅提供了有类型的变量，而且提供了一些流程控制机制，如循环和子程序等。这些高级机制使编程者可以摆脱许多具体细节，方便了复杂程序的书写，写出的程序更容易阅读，错误也更容易辨认和改正。FORTRAN 语言在诞生后受到广泛欢迎。

高级程序语言更接近人们习惯的描述形式，更容易被接受，从而促使更多的人加入程序设计中。使用高级语言编写程序的效率更高，使人们开发出更多的应用系统，这反过来又大大推动了计算机应用的发展。计算机应用的发展又推动了计算机产业的大发展。可以说，高级程序语言的诞生和发展，对于计算机发展到今天起到极其重要的作用。从 FORTRAN 语言诞生至今，人们提出的编程语言超过千种，其中大部分只是实验性的，只有少数编程语言得到广泛使用，如 C、C++、Pascal、Ada、Java、LISP、Smalltalk、PROLOG 等，这些编程语言都曾在计算机程序语言或计算机的发展历史上起过(有些仍在起着)极其重要的作用。随着时代的发展，如今绝大部分程序都是使用高级程序语言编写的。人们也已习惯于用“程序设计语言”特指各种高级程序语言。使用高级程序语言(例如 C 语言)描述前面同样的程序片段时，只需要一行代码：

```
d = a * b + c;
```

这表示要求计算机计算等号右边的表达式的值，然后将计算结果存入由 d 代表的存储单元中。这种表示方式与人们熟悉的数学形式直接对应，因此更容易阅读和理解。高级程序语言完全采用符号形式，使人们可以摆脱难用的二进制形式和具体计算机的细节。此外，高级程序语言还提供了许多高级的程序结构，从而方便人们组织复杂的程序。

计算机能否理解使用这些高级程序语言编写的指令呢？答案是不能，那么如何使计算机执行用高级程序语言编写的程序指令呢？我们需要将这些程序指令翻译成机器指令。编译器就是这样一种用来完成上述翻译任务的特殊程序，每一种编程语言都有不同的编译器。编译器有如下两种工作方式：

- 编译方式。人们首先针对具体的编程语言(例如 C 语言)开发出翻译软件(程序)，其功能是将采用这种高级程序语言编写的程序翻译为所使用计算机的机器语言的等价程序。这样，在使用这种高级编程语言写出程序后，只要将它们发送给翻译程序，就能得到与之对应的机器语言程序。此后，只要命令计算机执行机器语言程序，计算机就能执行我们所要完成的工作了。
- 解释方式。人们首先针对具体的高级程序语言开发解释软件，其功能是逐条读入使用高级程序语言编写的程序，并一步步地按照程序的要求开展工作，完成程序描述的计算任务。有了解释软件，直接将写好的程序发送给运行着解释软件的计算机，就可以完成程序描述的任务了。

1.1.2　C 语言的发展过程

C 语言是由 UNIX 的创建者 Dennis Ritchie(丹尼斯•里奇)和 Ken Thompson(肯•汤普逊)于 1970 年在 BCPL 语言(简称 B 语言)的基础上发展并完善起来的一种编程语言。20 世纪 70 年代初期，AT&T 贝尔实验室的程序员丹尼斯•里奇第一次把 B 语言改为 C 语言。

最初，C 语言运行于多用户、多任务的 UNIX 操作系统上。后来，丹尼斯•里奇用 C 语言改写了 UNIX C 的编译程序，肯•汤普逊又用 C 语言成功重写了 UNIX 内核及其核心工具，从此开创了编程史上的新篇章。UNIX 成为第一个不是使用汇编语言编写的主流操作系统。

随着计算机应用的发展，人们强烈地希望 C 语言能成为一种更安全可靠、不依赖于具体计算机和操作系统(如 UNIX)的标准程序设计语言。美国国家标准局(ANSI)于 20 世纪 80 年代专门成立了小组来研究 C 语言的标准化问题，并于 1989 年颁布了 ANSI C 标准。这个标准已被国际标准化组织和各国标准化机构接受，也被采纳为中国国家标准。此后，经过人们持续的努力和推动，于 1999 年通过了 ISO/IEC 9899:1999 标准(一般称为 C99)，这一新标准对 ANSI C 做了诸多重要的修订与功能扩充。

1.2 C 语言特点

C 语言之所以能被计算机界广泛接受，缘于 C 语言自身的如下特点。

(1) C 语言是中级语言。C 语言把高级语言的基本结构和语句与低级语言的实用性结合了起来。C 语言可以像汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元。

(2) C 语言是结构式语言。结构式语言的显著特点是代码与数据实现了分隔，换言之，程序的各个部分除必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护及调试。C 语言是以函数形式提供给用户的，这些函数除方便调用外，还具有多种循环和条件语句，用于控制程序流向，从而使程序完全结构化。

(3) C 语言功能齐全。C 语言不仅具有各种各样的数据类型，还引入了指针的概念，可使程序效率更高。另外，C 语言的计算功能、逻辑判断功能也比较强大。

(4) C 语言适用范围广。C 语言适用于多种操作系统，如 Windows、Linux、macOS 等，并且适用于多种机型。

在编写需要结合硬件进行操作的场合，C 语言明显优于其他编程语言，一些大型应用软件就是使用 C 语言编写的。

C 语言得到业界的广泛认可。一方面，C 语言在程序设计语言研究领域具有一定价值，由它引出了不少后继语言，另有许多语言吸收了 C 语言的不少优点。另一方面，C 语言对计算机产业和应用领域的发展也起到十分重要的推动作用。正是由于这些原 s 因，C 语言的设计者获得了计算机科学领域的最高荣誉——图灵奖。

1.3 简单的 C 程序实例

1.3.1 C 语言程序的构成与格式

我们先通过几个简单的程序来了解 C 语言程序的编写特点。

【例 1-1】输出“This is a C program.”

```
#include <stdio.h>                    // 编译预处理命令
void main()                           // 定义 main 函数
{                                     // 函数开始的标志
```

教学视频

```
    printf("This is a C program.\n");          // 输出一行信息
}                                              // 函数结束标志
```

程序运行结果如图 1-1 所示。

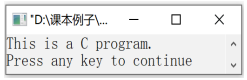


图 1-1　程序运行结果

(1) 上面这个简单的 C 语言程序可分为两个基本部分：第一行是特殊行，include <stdio.h>说明程序需要用到 C 语言系统提供的标准功能(参考标准库文件 stdio.h)。其他几行是程序的基本部分。

(2) main 是主函数名，void 是函数类型。每个 C 语言程序都必须有一个 main 函数，main 函数是每个 C 语言程序的执行起点(入口点)。

main 函数的函数体是用一对花括号{ }括起来的。{和}是函数的开始和结束标志，不可省略。main 函数中的所有操作(或语句)都在这对花括号之间。也就是说，main 函数的所有操作都在 main 函数体中。

(3) printf 是 C 语言提供的库函数，功能是进行输出(显示在计算机屏幕上)，此处用于输出字符串"This is a C program.\n"，也就是在计算机屏幕上显示“This is a C program.”。其中的\n 表示输出后换行。

(4) 注意，函数体中的每条语句必须以分号结束。

(5) 在使用库函数中的输入输出函数时，必须提供有关这些函数的信息：#include <stdio.h>。

【例 1-2】用 C 语言解决求和问题。

教学视频

```
#include <stdio.h>
void main()
{
    int a,b,sum;                    // 定义变量 a、b、sum
    a=123;b=456;                    /*以下 3 行为 C 语句*/
    sum=a+b;
    printf("sum=%d\n",sum);         // %d 为格式控制符
}
```

程序运行结果如图 1-2 所示。

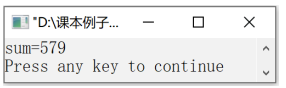


图 1-2　程序运行结果

(1) 同样，这个程序也必须包含 main 函数作为程序的执行起点。{和}之间为 main 函数的函数体，main 函数的所有操作都在 main 函数体中。

(2) 注释可以使用//以及/*和*/来标识。从//开始到换行符结束的内容为单行注释，使用/*和*/括起来的内容为段落注释。注释只是为了改善程序的可读性，在编译、运行时不起作用。因此，注释可以使用汉字或英文字符，既可以出现在一行中的最右侧，也可以单独成为一行。注释允许占用多行，只是需要注意/*与*/必须配对使用，一般不要嵌套使用。

(3) 语句 int a,b,sum;定义了三个整数类型的变量 a、b、sum。C 语言中的变量必须先声明再使用。

(4) a=123;b=456;是两条赋值语句，作用是将整数 123 赋给整型变量 a，并将整数 456 赋给整型变量 b。注意这是两条语句，每条语句均以;结束。也可以将这两条语句写成两行：

```
a=123;
b=456;
```

由此可见，C 语言程序的书写可以相当随意，但是为了保证代码容易阅读，建议遵循一定的规范。

(5) sum=a+b;的作用是将 a、b 两个变量的值相加，然后将结果赋给整型变量 sum。此时，sum 变量的值为 579。

(6) printf("sum=%d\n",sum);的作用是调用库函数 printf 以输出 sum 变量的值。%d 为格式控制符，表示 sum 变量的值将以十进制整数形式输出。

【例 1-3】输入两个数，输出其中较大的那个数。

教学视频

```
#include <stdio.h>
void main()
{                                   // main 函数体开始
    int a,b,c;                      // 定义变量 a、b、c
    scanf("%d,%d",&a,&b);           // 输入变量 a 和 b 的值
    c=max(a,b);                     // 调用 max 函数，将调用结果赋给变量 c
    printf("max=%d\n",c);           // 输出变量 c 的值
}                                   // main 函数体结束
int max(int x,int y)                // max 函数用于判断并返回两个数中较大的那个数
{                                   // max 函数体开始
    int z;                          // 定义函数体中的变量 z
    if(x>y) z=x;                    // 若 x>y，将 x 的值赋给变量 z
    else z=y;                       // 否则，将 y 的值赋给变量 z
    return z;                       // 将变量 z 的值返回到调用 max 函数的地方
}                                   // max 函数体结束
```

程序运行结果如图 1-3 所示。

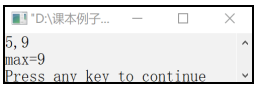


图 1-3　程序运行结果

(1) 这个程序包括两个函数。其中，main 函数仍然是整个程序的执行起点，max 函数的功能是计算两个数中较大的那个数。

(2) main 函数首先调用 scanf 函数以获得两个整数，将它们存入变量 a 和 b，然后调用 max 函数以获得两个数中较大的那个数并赋给变量 c，最后输出变量 c 的值(结果)。

(3) int max(int x,int y)是 max 函数的函数头，这表明 max 函数将获得两个整型参数并返回一个整数。

(4) max 函数同样使用花括号{}将函数体括起来。max 函数的函数体是函数功能的具体实现：从参数中获得数据，处理后得到结果，然后将结果返回给调用函数 main。

例 1-3 还表明除了调用库函数之外，还可以调用用户自己定义并编写的函数。

1.3.2　C 语言程序的结构

结合上面的三个例子，大家对 C 语言程序的基本组成和形式(程序结构)应该有了初步的了解：

(1) C 语言程序由多个函数构成。

① C 语言程序至少需要包含一个 main 函数，也可以包含一个 main 函数和若干其他函数。函数是 C 语言程序的基本单位。

② 被调用的函数可以是系统提供的库函数，也可以是用户根据需要自行设计编写的函数。C 语言是函数式语言，程序的全部工作都是由各个函数完成的。编写 C 语言程序就是编写一个个函数。

③ 函数库资源非常丰富，ANSI C 提供了非常丰富的库函数。

(2) main 函数(主函数)是每个程序的执行起点。

main 函数既可以放在整个程序的最前面，也可以放在整个程序的最后，还可以放在其他函数之间。但 C 程序总是从 main 函数开始执行，而不论 main 函数处在程序中的什么位置。

(3) 函数由函数头和函数体两部分组成。

① 函数头。函数的第一行，格式如下：

返回值类型 函数名([函数参数类型 1 函数参数名 1],…,[函数参数类型 *n*，函数参数名 *n*])

例如 int max(int x, int y);。

注意：函数可以没有参数，但是后面的一对圆括号不能省略，这是规定。

② 函数体。在函数头的下方，使用一对花括号括起来的部分为函数体。如果函数体内有多对花括号，那么最外层才是函数体的范围。函数体一般都包括声明部分和执行部分。

```
{
  [声明部分]：定义函数需要使用的变量。
  [执行部分]：由若干语句组成的命令序列(可在其中调用其他函数)。
}
```

(4) C 语言程序书写自由。

① 在书写 C 语言程序时，一行可以写一条或多条语句，一条语句也可以分多行书写。

② C 语言程序没有行号，并且没有像 FORTRAN、COBOL 那样严格规定书写格式(语句必须从某一列开始)。

③ 每条语句在最后必须以分号结束。

(5) 可以使用//以及/*和*/对 C 语言程序中的任何部分进行注释。

注释可以提高程序的可读性，使用注释是编程人员必须养成的良好习惯。使用注释的原因如下：

① 编写好的程序往往需要修改、完善，事实上，并不存在任何不需要修改和完善的应用系统。很多人会发现，自己编写的程序在经历了一些时间以后，由于缺乏必要的文档和注释，最后连自己都很难再读懂，因而需要花费大量时间重新思考和理解原来的程序。如果一开始就对程序进行注释，刚开始虽然麻烦一些，但日后可以节省大量的时间。

② 实际的应用系统往往由多人合作开发，程序文档、注释是其中重要的交流工具。

(6) C 语言本身不提供输入输出语句，输入输出操作是通过调用库函数(如 scanf、printf 函数)完成的。

输入输出操作涉及具体的计算机硬件，把输入输出操作放在函数中进行处理，可以简化 C 语言的编译系统，便于 C 语言在各种计算机上实现。不同的计算机系统需要对函数库中的函数

进行不同的处理，以便实现相同或类似的功能。

不同的计算机系统除了提供函数库中的标准函数之外，还会按照硬件情况提供一些专用函数。因此，不同计算机系统提供的函数在数量和功能上也存在一定差异。

1.3.3 良好的编程风格

C 语言是一种“格式自由”的编程语言，除若干简单限制外，编写程序的人完全可以根据自己的想法和需要选择程序格式，确定在哪里换行，在哪里增加空格等。这些格式变化并不影响程序的意义，但是，不规定程序格式并不说明格式不重要。对于阅读而言，程序格式非常重要。在多年的程序设计实践中，人们在这方面的认识得到了统一。由于程序可能很长，结构可能很复杂，因此程序必须采用良好的格式来编写，所用格式应很好地体现程序的层次结构，反映各个部分之间的关系。

关于程序格式，人们普遍采用的方式如下：

- 在程序里适当加入空行，分隔程序中处于同一层次的不同部分。
- 将同一层次的不同部分对齐排列，下一层次的内容通过适当地添加空格(在一行的开头添加空格)，使程序结构更清晰。
- 在程序里添加一些说明性信息。

大家在刚开始学习程序设计时就应养成注意程序格式的习惯。对于小的程序，虽然采用良好格式的优势并不明显，但对于稍大一点的程序，情况就不一样了。有人为了方便，根本不关心程序格式，想的只是少输入几个空格或换行，这样做的结果就是导致自己在随后的程序调试检查中遇到更多麻烦。正因为如此，这里特别提醒读者：从一开始编写最简单的程序时就注意养成好习惯。目前多数程序设计语言(包括 C 语言)都是格式自由语言，这使人们能够十分方便地根据自己的需要和习惯写出具有良好格式的程序。总之，优秀的程序员应具备以下良好的编程风格：

- 使用 Tab 键缩进代码；
- 对齐各个层次的花括号；
- 添加足够的注释并使用适当的空行。

1.4 搭建 Visual C++ 6.0 开发环境

“工欲善其事，必先利其器”，本节将详细介绍学习 C 语言程序开发的常用工具：Visual C++ 6.0。Visual C++ 6.0 是功能强大的可视化软件开发工具，已将程序的代码编辑、编译、连接和调试等功能集于一身。

1.4.1 Visual C++ 6.0 的安装

Visual C++ 6.0(中文版)的具体安装步骤如下：

(1) 双击 Visual C++ 6.0 安装文件夹中的 SETUP.EXE 安装文件，如图 1-4 所示。弹出如图 1-5 所示的“程序兼容性助手”界面，单击“运行程序”按钮，进入安装向导界面。

(2) 在如图 1-6 所示的界面中单击“下一步”按钮，进入如图 1-7 所示的“最终用户许可协

议”界面，选中“接受协议”单选按钮，然后单击“下一步”按钮。

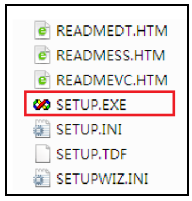


图 1-4　双击安装文件

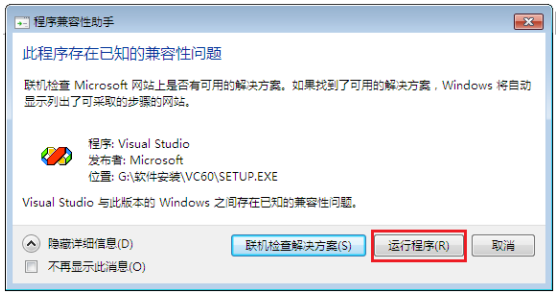


图 1-5　单击“运行程序”按钮

图 1-6　进入安装向导

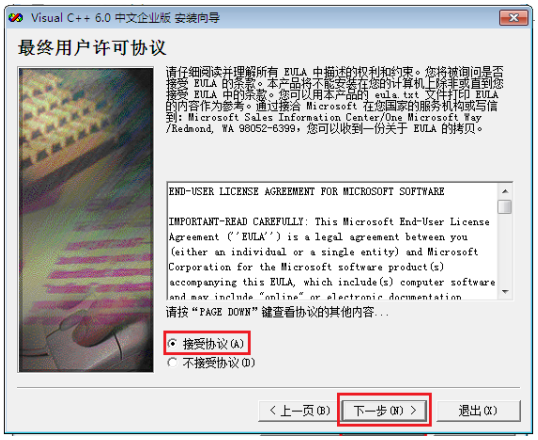


图 1-7　“最终用户许可协议”界面

(3) 进入如图 1-8 所示的“产品号和用户 ID”界面，根据自身情况填写姓名和公司名称，也可采用默认设置，单击“下一步”按钮。

(4) 进入“Visual C++ 6.0 中文企业版”界面，如图 1-9 所示。选中“安装 Visual C++ 6.0 中文企业版”单选按钮，然后单击“下一步”按钮。

图 1-8　“产品号和用户 ID”界面

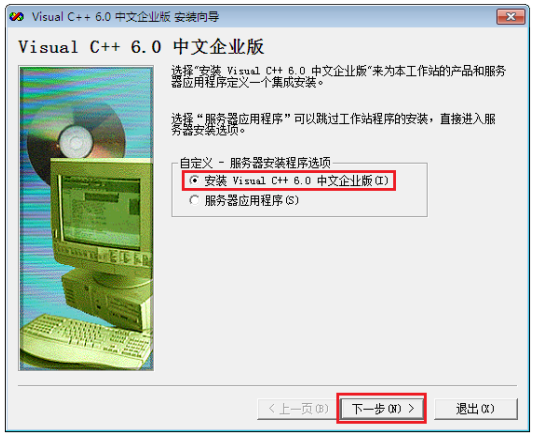


图 1-9　“Visual C++ 6.0 中文企业版”界面

(5) 进入“选择公用安装文件夹”界面，如图 1-10 所示。公用文件默认存储在 C 盘中，单

击“浏览”按钮，可选择其他存放位置。单击“下一步”按钮，进入如图 1-11 所示的安装程序的欢迎界面，单击“继续”按钮。

(6) 进入如图 1-12 所示的产品 ID 确认界面。此处显示了我们将要安装的 Visual C++ 6.0 软件的产品 ID，在向微软请求技术支持时，需要提供软件的产品 ID，单击“确定”按钮。

(7) 如果读者的计算机中安装过 Visual C++ 6.0，尽管已经卸载，在重新安装时也仍会提示如图 1-13 所示的信息。安装软件检测到系统之前安装过 Visual C++ 6.0，如果想要覆盖安装的话，单击“是”按钮；如果要将 Visual C++ 6.0 安装到其他位置，单击“否”按钮。这里单击“是”按钮，继续安装。

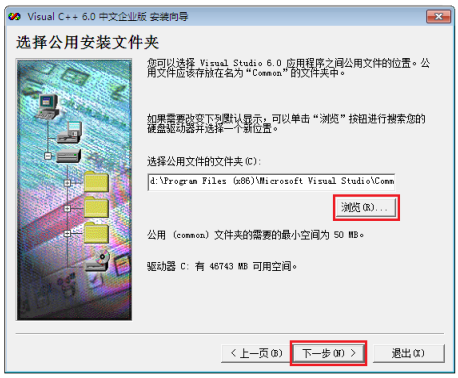


图 1-10 “选择公用安装文件夹”界面

图 1-11 安装程序的欢迎界面

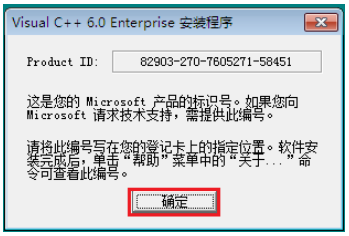


图 1-12 产品 ID 确认界面

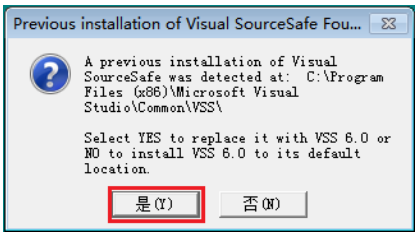


图 1-13 覆盖以前的安装

(8) 进入如图 1-14 所示的安装类型选择界面。Typical 为典型安装，Custom 为自定义安装，这里选择 Typical 安装类型。

(9) 进入如图 1-15 所示的环境变量注册界面。选中 Register Environment Variables 复选框以注册环境变量，单击 OK 按钮。

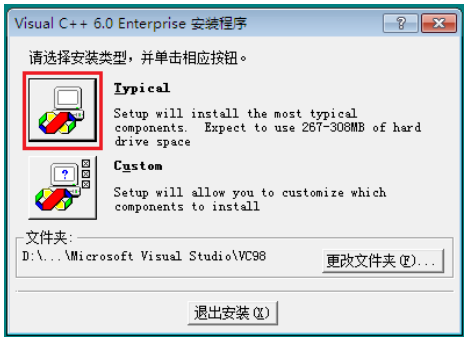


图 1-14 安装类型选择界面

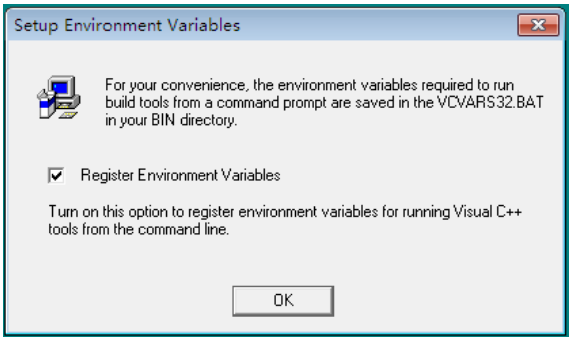


图 1-15 环境变量注册界面

(10) 进入如图 1-16 所示的 Visual C++ 6.0 安装进度界面，当进度条达到 100%时，表示安装成功，如图 1-17 所示。

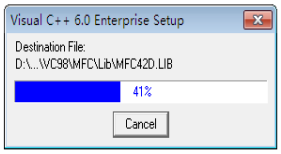


图 1-16　安装进度界面

图 1-17　安装成功

1.4.2　使用 Visual C++ 6.0 创建 C 文件

安装完之后，就可以使用 Visual C++ 6.0 创建 C 文件了，步骤如下：

(1) 选择“开始”菜单中的 Microsoft Visual C++ 6.0 命令，如图 1-18 所示。

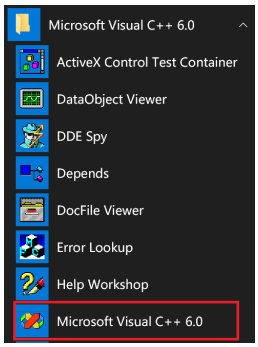


图 1-18　启动 Visual C++ 6.0 开发环境

(2) 进入 Visual C++ 6.0 开发界面，如图 1-19 所示。

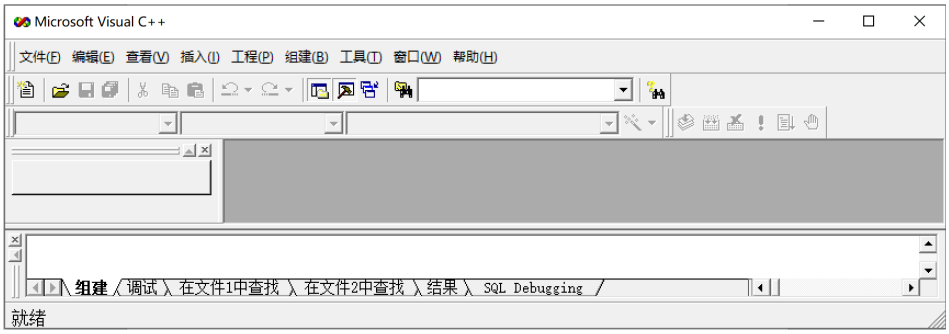


图 1-19　Visual C++ 6.0 开发界面

(3) 在编写程序之前，我们需要新建程序文件。在 Visual C++ 6.0 开发界面中选择“文件”菜单中的“新建”选项，如图 1-20 所示，或者按 Ctrl+N 快捷键，打开“新建”对话框。

图 1-20　选择“新建”选项

(4) 在“新建”对话框中选择想要创建的文件类型。首先选择“文件”选项卡，下方的列表框中显示了可以创建的文件类型。因为要创建 C 源文件，所以选择 C++ Source File 选项。在右侧的“文件名”文本框中输入所要创建的文件的名称，例如 hello.c。在“位置”文本框中设置源文件的保存地址，可以通过单击右边的...按钮来修改源文件的存储位置。具体操作如图 1-21 所示。

图 1-21　创建 C 源文件

(5) 输入文件名称并指定源文件的保存地址后，单击“确定”按钮，即可成功创建一个新的 C 源文件。此时，我们可以在开发环境中看到创建好的 C 源文件，如图 1-22 所示。

(6) 现在开始编写程序代码。程序代码的输入界面如图 1-23 所示。

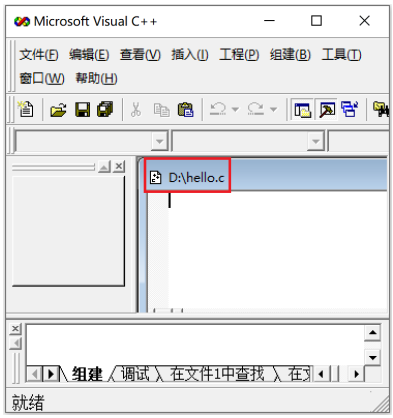


图 1-22　新创建的 C 源文件

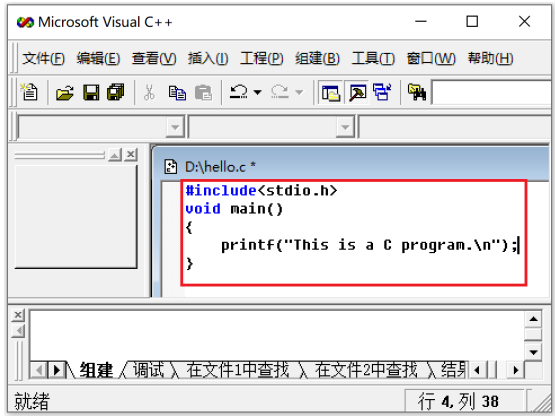


图 1-23　程序代码的输入界面

(7) 编写完程序后，选择“组建”菜单中的“编译”选项，对程序进行编译，如图 1-24 所示。

(8) 系统弹出如图 1-25 所示的提示框，询问是否创建默认的项目工作环境。

(9) 单击“是”按钮，此时会询问是否对改动后的文件进行保存，如图 1-26 所示。

(10) 单击“是”按钮，开始编译程序。程序如果没有错误，即可被成功编译。单击工具栏中的按钮，使用连接程序创建.exe 文件。最后，单击工具栏中的按钮，即可显示程序的执行结果。

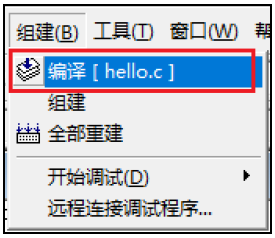


图 1-24　选择“编译”选项

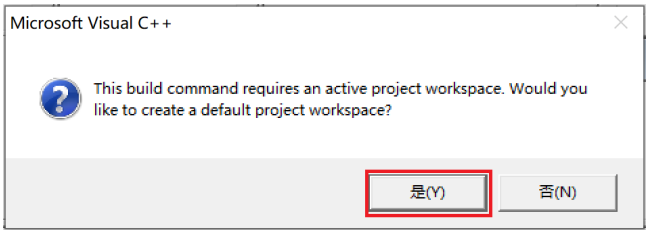


图 1-25　询问是否创建默认的项目工作环境

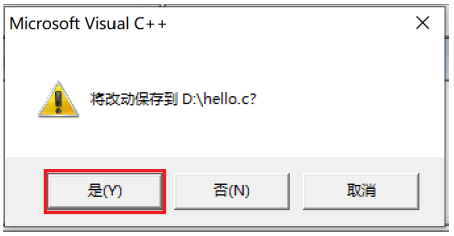


图 1-26　询问是否对改动后的文件进行保存

1.4.3　Visual C++ 6.0 中 C 文件的编辑、编译与运行

C 语言是高级程序设计语言，使用 C 语言编写的程序通常称作 C 语言源程序(扩展名为.c)，这种程序虽然容易使用、书写和阅读，但计算机不能直接执行，因为计算机只能识别和执行二进制形式的机器语言程序。为使计算机完成某个 C 语言源程序描述的工作，就必须把这个 C 语言源程序转换成二进制形式的机器语言程序，这种转换被称为“C 语言源程序的加工”。“C 语言源程序的加工”包括“编译”和“连接”两个步骤，如图 1-27 所示。

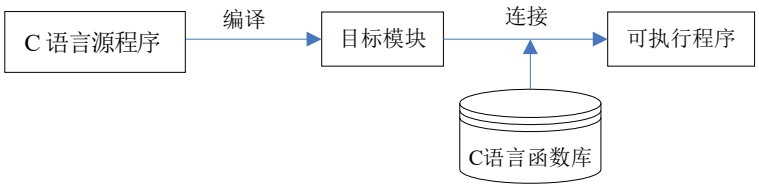


图 1-27　C 语言源程序的加工

第一步，由编译程序对源文件进行分析和处理，生成相应的机器语言目标模块，由目标模块构成的代码文件称为目标文件(扩展名为.obj)。目标文件还不能执行，因为缺少运行 C 语言程序所需的运行系统。此外，C 语言程序一般都要使用函数库提供的某些功能，例如标准函数库中的输出函数 printf。

第二步，由连接程序将编译后得到的目标模块与其他必要部分(运行系统、函数库提供的功能模块等)拼装起来，组成可执行程序(扩展名为.exe)。

1.4.4　编程中的注意事项

程序设计是一种智力劳动，编程时面对的是需要解决的问题，最终完成的是符合解题要求的程序。有了编程语言之后，该如何着手编写程序呢？在程序设计领域里，解决小问题与解决大问题，为完成练习而写程序与为解决实际应用需求而写程序之间并没有本质的区别。

使用编程语言编写程序时需要注意以下几个重要方面：

(1) 培养分析问题的能力，特别是从计算思维和程序实现的角度分析问题的能力。应学会从问题出发，通过逐步分析和分解，把原始问题转换为能用计算机通过程序方式解决的问题。在此过程中，构造出解决方案。这方面的探索没有止境，许多专业性问题都需要用计算机来解决。为此，参与者既需要熟悉计算机，又需要熟悉专业领域；未来的世界将特别需要这种复合型人才。虽然课程和教科书里的问题很简单，但它们是通向复杂问题的桥梁。

(2) 掌握所使用的编程语言，熟悉编程语言中的各种结构，包括它们的形式和意义。编程语言是解决程序问题的工具，要想写好程序，就必须熟悉编程语言。应注意，熟悉编程语言绝不是背诵定义，这种熟悉只有在程序设计实践中才能达成。就像上课再多也不能学会开车一样，仅靠看书、读程序、抄程序不可能真正学会写程序，必须反复亲手实践。

(3) 学会写程序。虽然写过程序的人很多，但是会写程序、是否能写出好程序的人不多。什么是“好程序”？例如，为解决同样的问题，写出的程序是否比较简单就是衡量标准之一。这里可能有算法的选择问题，有语言的使用问题，以及需要确定适用的程序结构等。除了程序本身是否正确之外，人们还特别关注写出的程序是否具有良好的结构，是否清晰，是否易于阅读和理解，以及当问题中的某些条件或要求发生改变时，它们是否容易修改以满足新的要求等。

(4) 检查程序错误的能力。初步编写的程序通常会包含一些错误。虽然编译系统能帮我们查出其中一些错误，并通告发现错误的位置，但确认实际错误和实际位置，以及确定应如何改正等，这些永远是编程人员自己的事。对于系统提出的各种警告，以及系统无法检查出来的错误的认定等，更需要编程人员具备一定的纠错能力，这种能力也需要在学习过程中有意识地加以锻炼。

(5) 熟悉所使用的工具和环境。程序设计需要用到一些编程工具，在具体的开发环境中进行编程时，熟悉工具和环境是很重要的。大部分读者可能要用某种集成开发环境进行编程，熟悉这种集成开发环境能够大大提高我们的工作效率。

1.5 本章小结

本章主要介绍了以下内容：

(1) C 语言的发展过程。

(2) C 语言的特点与标准化。

(3) 通过三个简单的 C 语言实例了解 C 语言程序的结构。

(4) Visual C++ 6.0 开发环境的搭建，C 文件的创建、编辑、编译与运行。

(5) 了解 C 语言程序的执行过程。

(6) 养成良好的编程风格，了解编程中的注意事项。

1.6 编程经验

(1) 在实际的编程过程中，如果遇到 C 语言的某些特征不清楚，可以编写小的程序运行一

下，看看得到的结果和自己了解到的是否一致，从而加深理解。

(2) 所有不以花括号开始和结束的语句都必须以分号结束，以#开始的语句除外。

(3) 一般不要在一行中编写多条语句。

(4) 在编写程序时，花括号应该成对出现；否则，当花括号中的程序较长时，可能会忘记输入匹配的}符号。

(5) C 语言对大小写很敏感，所以在编写 C 语言程序时，大小写一定要分清楚。

(6) 在自定义函数中，一定要添加注释。

1.7 本章习题

1. 简述 C 语言的基本特点。

2. 举例说明 C 语言程序由哪几部分组成。

3. C 语言程序从开发到执行一般需要经过几个阶段？各个阶段的作用是什么？

4. 熟悉自己学习 C 语言程序设计时准备使用的编译系统或集成开发环境，了解编译系统的基本使用方法、基本操作(命令方式或包含窗口和菜单的图形界面方式)，弄清楚如何取得联机帮助信息。设法找到并翻阅编译系统的手册，了解手册的结构和各部分的基本内容。了解在编译系统中编写简单程序的基本步骤。安装并熟悉 Visual C++。

5. 输入本章正文中给出的C程序实例(注意程序格式)，在自己选用的系统中创建C源程序；对 C 源程序进行加工，得到对应的可执行程序；运行可执行程序，看看效果如何(输出了什么信息等)。

6. 下列C语言程序的写法是否正确？若有错误，请改正。

```
1) void main()
   {
       printf("C program1")
   }
```

```
2) void main
   {
      printf("C program1");
      printf("C program2");
   }
```

7. 在 C 语言中，main 函数的用途是什么？

8. 描述 C 语言程序从编写到运行都经过了哪些过程？

9. 说明源代码和可执行程序之间的关系。

10. 编写一个 C 语言程序，生成以下图形。

```
    *
   ***
  *****
 *******
  *****
   ***
    *
```

11. 选择题

(1) 关于程序设计的步骤和顺序，以下说法中正确的是____。

(A) 确定算法后，整理并写出文档，最后进行编码和上机调试。

(B) 首先确定数据结构，然后确定算法，编码并上机调试，最后整理文档。

(C) 首先进行编码和上机调试，然后在编码过程中确定算法和数据结构，最后整理文档。

(D) 首先写好文档，然后根据文档进行编码和上机调试，最后确定算法和数据结构。

(2) 计算机能直接执行的是____。

(A) 源程序　　(B) 目标程序　　(C) 汇编程序　　(D) 可执行程序

(3) 以下叙述中错误的是____。

(A) C 语言的可执行程序是由一系列机器指令构成的。

(B) 用 C 语言编写的源程序不能直接在计算机上运行。

(C) 通过编译得到的二进制目标程序需要连接才可以运行。

(D) 在没有安装 C 语言集成开发环境的计算机上，不能运行通过 C 源文件生成的.exe 文件。

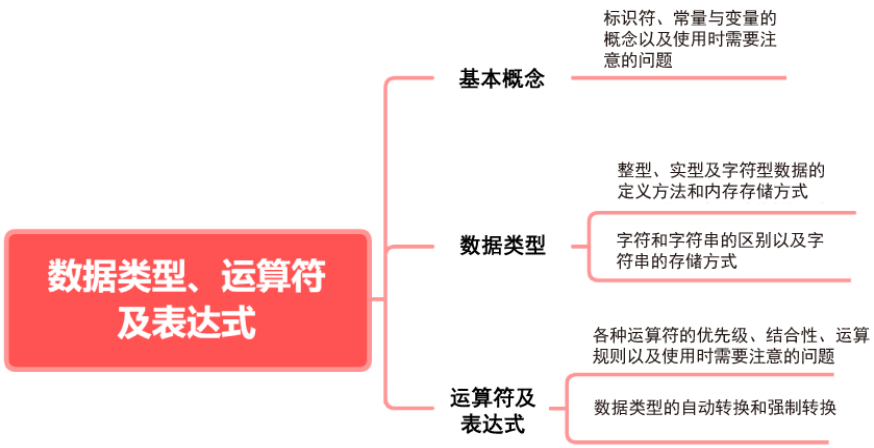
第 2 章

数据类型、运算符及表达式

本章概览

大家现在一定十分渴望开始编写程序，与计算机进行实际的交互。但是，在开始编写真正有用的程序之前，我们必须先学习一些 C 语言的基本概念。本章首先简单介绍常用数制以及常用数制整数之间的转换方法，然后介绍常量、变量、标识符、简单数据类型、运算符、表达式和类型转换等。这些都是进行 C 语言程序设计时需要掌握的基础知识，初学者可能觉得这些知识有点枯燥、琐碎且难以记忆，但是它们都需要重点掌握。

知识框架



2.1 数制

数制又称计数制，是指使用一组有限的符号和统一的规则来表示数值。可使用的数字符号的数目称为基数，假设基数为 n，则称为 n 进制。在日常生活中，常用的计数制是十进制，使用 0~9 共 10 个阿拉伯数字进行计数。

除了使用十进制进行计数以外，还有许多非十进制的计数方法。在计算机领域，常见的计数制还有二进制、八进制和十六进制。任意一个数字都可以使用不同的进制来表示。比如：十进制数$(68)_{10}$可以用二进制表示为$(1000100)_2$，也可以用八进制表示为$(104)_8$，还可以用十六进制表示为$(44)_{16}$。

2.1.1 常用数制

数码是数制中表示基本数值大小的不同数字符号。例如，十进制有 10 个数码——0、1、2、3、4、5、6、7、8、9；二进制有两个数码——0、1；八进制有 8 个数码——0、1、2、3、4、5、6、7；十六进制则有 16 个数码——0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。

基数是数制使用的数码个数。例如，二进制的基数为 2，八进制的基数为 8，十进制的基数为 10，十六进制的基数为 16。

数制中某一位上的 1 所表示的数值大小(所处位置的价值)称为位权。对于 N 进制数，整数部分的第 i 位(从右向左数)的位权为 $N^{(i-1)}$。例如，在十进制数 123 中，3 的位权是 10^0=1，2 的位权是 10^1=10，1 的位权是 10^2=100；在二进制数 1011 中，从右侧开始第一个 1 的位权是 2^0=1，第二个 1 的位权是 2^1=2，0 的位权是 2^2=4，第三个 1 的位权是 2^3=8。

表 2-1 列出了几种常用数制的数值对应关系。注意，八进制数没有 8 和 9，二进制数 1000 对应的八进制数是 10，二进制数 1001 对应的八进制数是 11。

表 2-1　常用数制的数值对应关系

二进制	十进制	八进制	十六进制
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	8	10	8
1001	9	11	9
1010	10	12	A
1011	11	13	B
1100	12	14	C
1101	13	15	D
1110	14	16	E
1111	15	17	F

1. 十进制

十进制是我们日常生活中最常用的计数方式，基数是 10，有 10 个数字符号——0、1、2、3、4、5、6、7、8、9。其中，最大数码是“基数”减 1，也就是 9，最小数码是 0。

2. 二进制

二进制是计算机内部采用的计数方式。在计算机中，所有数据都需要转换为二进制后才能处理，这是由硬件电路的特性决定的。我们存放在计算机中的视频、图片、音乐等，也都是以二进制数码形式存放的。二进制的基数是 2，只有两个数字符号——0 和 1。在给定的数中，如果除了 0 和 1 之外还有其他数字符号，如 1061，那就绝不会是二进制数。

3. 八进制

八进制的基数是 8，有 8 个数字符号——0、1、2、3、4、5、6、7。对比十进制可以看出，八进制相比十进制少了两个数字符号 8 和 9。因此，当给定的数中出现 8 或 9 时，如 23459，那就绝不会是八进制数。八进制数的最大数码是 7，最小数码是 0。

4. 十六进制

在编写程序时，十六进制用得较多，有 16 个数字符号，除了使用十进制中的 10 个数字符号之外，还使用了 6 个英文字母，这 16 个数字符号依次是 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。其中，A～F 分别代表十进制中的 10～15。如果给定的数中出现了字母符号，如 63AB，那就一定不会是八进制数或十进制数。十六进制数的最大数码是“基数”减 1，也就是 15(在十六进制中为 F)，最小数码是 0。

2.1.2　常用数制整数之间的转换

这里只讲述常用数制整数之间的转换，不涉及小数部分的转换，我们提到的数字均指整数。

1. 非十进制向十进制转换

将非十进制数转换为十进制数的方法是，首先将数字按位权展开，然后将各项相加，即可得到相应的十进制数。这种方法被称为“按权相加”法。

例如：

- 二进制数$(1010)_2$可按位权展开为$1\times2^3+0\times2^2+1\times2^1+0\times2^0=8+0+2+0=(10)_{10}$。
- 八进制数$(123)_8$可按位权展开为$1\times8^2+2\times8^1+3\times8^0=64+16+3=(83)_{10}$。
- 十六进制数$(12D)_{16}$可按位权展开为$1\times16^2+2\times16^1+13\times16^0=256+32+13=(301)_{10}$。

2. 十进制向二进制转换

在将十进制数转换为二进制数时，采用的是“除 2 取余，逆序排列”法。具体做法是：用 2 整除十进制数，得到商和余数；再用 2 整除商，又会得到商和余数；如此重复进行，直到商小于 1 为止；然后把先得到的余数作为二进制数的低位有效位，而把后得到的余数作为二进制数的高位有效位，依次排列起来。图 2-1 以$(36)_{10}$为例说明了从十进制数向二进制数转换的具体

过程，将得到的余数逆序写出来，便可以得到$(36)_{10}=(100100)_2$。

		余数	
2	36		
2	18	0	36÷2=18……余 0
2	9	0	18÷2=9……余 0
2	4	1	9÷2=4……余 1
2	2	0	4÷2=2……余 0
2	1	0	2÷2=1……余 0
2	0	1	1÷2=0……余 1

图 2-1　十进制数向二进制数转换

3. 二进制与八进制互换

将八进制数转换成二进制数的方法是，将每 1 位八进制数直接写成相应的 3 位二进制数。例如：$(123)_8=(001\ 010\ 011)_2=(1010011)_2$。

将二进制数转换成八进制数的方法是，从右向左将每 3 位二进制数分成一组，不足 3 位的话，高位用 0 补足 3 位，然后将每一组二进制数直接写成相应的八进制数。例如：$(11110101)_2=(011\ 110\ 101)_2=(365)_8$。

4. 二进制与十六进制互换

将十六进制数转换成二进制数的方法是，将每 1 位十六进制数直接写成相应的 4 位二进制数。例如：$(4DF)_{16}=(0100\ 1101\ 1111)_2=(10011011111)_2$。

将二进制数转换成十六进制数的方法是，从右向左将每 4 位二进制数分成一组，不足 4 位的话，高位用 0 补足 4 位，然后将每一组二进制数直接写成相应的十六进制数。例如：$(100010110011101)_2=(0100\ 0101\ 1001\ 1101)_2=(459D)_{16}$。

2.2 常量与变量

C 语言中存在着两种数据表征形式：常量和变量。常量用来表征数据的值，变量不但可用来表征数据的值，还可用来存放数据。

2.2.1 常量

在程序运行过程中，值不能改变的量称为常量或常数。常量有两种类型：直接常量和符号常量。

1. 直接常量

直接常量又称值常量。根据数据类型的不同，分为：

- 整型常量，如 12、0、28。
- 实型常量，如 4.6、−1.2、3.14、78e2。

- 字符常量，如'A'、'z'、'n'。
- 字符串常量，如"What a wonderful day!"。

2. 符号常量

在 C 语言中，还可以使用标识符来表示常量，称为符号常量。符号常量在使用前需要明确定义。当定义符号常量时，需要在源文件的开头使用宏命令，也就是#define 命令。使用#define 宏命令定义符号常量的形式如下：

```
#define 符号常量名 常量        //注意行尾没有分号，并且不需要指定常量的数据类型
```

【例 2-1】符号常量的使用。

教学视频

```
#include<stdio.h>
#define PRICE 100
#define DISCOUNT 0.8
void main()
{
   int total;
   total=10*PRICE*DISCOUNT;
   printf("total=%d\n",total);
   total=100*PRICE*DISCOUNT;
   printf("total=%d\n",total);
}
```

程序运行结果如图 2-2 所示。

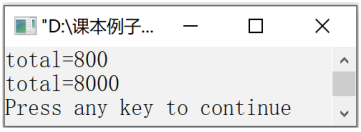


图 2-2　程序运行结果

在上述程序中，PRICE 和 DISCOUNT 都是符号常量，值分别为 100 和 0.8，此后凡是出现 PRICE 和 DISCOUNT 的地方，都会用 100 和 0.8 进行替代。替代工作是在预处理阶段进行的，编译器不会为符号常量分配存储空间。

使用符号常量的好处有以下两点：首先是含义清楚，在上述程序中，凡是出现 PRICE 的地方，我们都知道表示的是 100；其次，当符号常量被多次引用时，可以简化数据的输入，而当需要修改符号常量的值时，只需要对符号常量的定义进行修改即可。

注意：

(1) #define 是宏命令而非 C 语句，所以行尾不加分号。

(2) 推荐使用大写字母定义符号常量的名称，从而与变量标识符区分开来。当然，也可以使用小写，但前后必须一致，因为 C 语言严格区分字母的大小写。

(3) 不要使用赋值语句为符号常量赋值。如果不小心书写了对符号常量进行赋值的语句，编译器会报错。

(4) 不要指定符号常量的数据类型，也不要在常量名和常量值之间添加等号=。

例如：

#define PRICE=100 是错误的。

#define int PRICE 100 也是错误的。

使用符号常量时，需要特别注意符号常量的定义格式，以免出现上述错误。

2.2.2 变量

在程序运行过程中，值可以改变的量称为变量。在定义变量时，需要指定变量的名称。变量在定义后就会拥有一个具有特定属性的内存存储单元，这个内存存储单元用来存放数据。

C 语言规定：变量必须先定义，后使用。

每个变量都有名称，称为变量名。定义变量就是进行变量的声明并指明变量的数据类型，这相当于对变量进行注册。运行程序时，编译器将根据变量的数据类型，在内存中分配大小合适的存储空间，这样就可以将变量名与对应的存储空间联系起来。定义变量的一般格式如下：

```
数据类型  变量 1[,变量 2,…, 变量 n];
```

例如：int a,b=10;

其中，int 是类型标识符，a 和 b 是变量名，10 是变量 b 的初值。上述语句定义了两个 int 型变量，其中变量 a 未赋初值，变量 b 则在定义的同时被赋予初值。

变量的三要素：变量名、变量在内存中占据的存储单元、变量值。这三个要素之间的关系如图 2-3 所示。变量名在程序运行过程中不会改变，但变量的值是可以改变的。

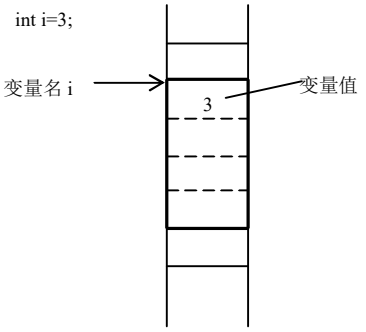


图 2-3　变量的三要素

注意：

(1) 变量在命名时需要遵守标识符定义规则，习惯上变量名用小写字母表示。为增强程序的可读性，所用标识符最好能“见名知意”。例如，表示年级的变量名用 grade 就比用 a 效果要好。

(2) 再强调一次，C 语言规定变量必须“先定义，后使用”。通常在函数开头定义局部变量，在源文件的开头定义全局变量。

(3) 应避免使用 C 语言中的关键字命名变量。

(4) C 语言对大小写敏感。例如，PRICE、Price 和 price 是三个不同的变量，书写时请注意保持大小写一致。

2.2.3　变量的初始化

请按顺序执行如下 4 条语句：

```
int Cats;
printf("%d\n",Cats);          //通常会在计算机屏幕上显示一个大的整数
Cats = 2;                     //将变量 Cats 的值设为 2
printf("%d\n",Cats);
```

请问，在 Cats = 2;这条语句执行之前，变量 Cats 的值是多少?

答案是：变量 Cats 的值将是一个不确定的数。在上述代码中，第一条语句创建了变量 Cats，编译器会给 Cats 变量分配对应的存储空间，但编译器不会预先清空这部分存储空间，也就是说，Cats 指向的是含有残留数据的存储空间。因此，如果仅仅定义变量而不进行初始化，变量中存放的将是没有意义的数值。

在声明变量的同时进行初始化是一种非常好的编程习惯，因为这样既可以避免将不明来源的数据带入程序，也可以避免在创建变量时使用系统垃圾值，从而减小程序出错的概率。示例如下：

```
int Cats = 2;
```

以上语句在将变量 Cats 声明为 int 类型的同时，设定初值为 2。

2.3　标识符和关键字

前面介绍了常量和变量，接下来说明一下在对常量和变量以及后面将要介绍的数组、函数和指针等对象进行命名时需要注意的有关事项。

2.3.1　标识符

在编写程序时，用来对符号常量、变量、数组、函数等对象进行命名的字符序列统称标识符。简单地说，标识符就是对象的名称，定义标识符就是为对象取名。前面用到的变量名 a 和 b、符号常量名 PRICE 和 DISCOUNT，以及函数名 printf 等都是标识符。在 C 语言中，标识符的定义规则如下：

(1) 标识符中只能包含字母、数字和下画线三种字符，第一个字符必须是字母或下画线，不能以数字开头。

下面是一些合法的标识符：

```
sum   average   _total   Class   day   stu_name   p4050
```

以下则是一些不合法的标识符：

```
M.D.John(出现非法字符.)          $123(以$开头)
#33(以#开头)                     3D64(以数字开头)
```

以下画线开头的标识符已保留给系统使用，编写程序时一般不要使用这种标识符，以免与系统内部使用的标识符重名。

(2) C 语言对大小写敏感。在标识符中，如果同一字母的大小写形式不同，那么它们将被看作不同的标识符。例如，a 和 A 是两个不同的标识符，name、Name、NAME、naMe 和 nMAE 也是互不相同的标识符。因此，在编写程序时，请注意保持大小写一致。

(3) 标识符可以包含的字符个数取决于编译器，遵循 C 语言标准的编译器至少支持 31 个字符(c89/c90 标准)，通常只要不超过这个长度就没有问题，但各个编译系统也都有自己的规定和限制。例如，假设某编译系统规定 8 个字符有效，此时 student_name 和 student_number 就是两个相同的标识符。

(4) 标识符既不能与 C 语言中的关键字同名，也不能与系统预先定义的标识符同名。

2.3.2 关键字

在 C 语言的合法标识符中，有个特殊的小集合称为关键字。作为关键字的标识符在程序中具有预先定义好的特殊意义，因此不能用于其他目的，更不能作为普通标识符使用。C 语言中的关键字共 32 个，如下所示。

- 基本类型关键字(5 个)：void、char、int、float、double。
- 类型修饰关键字(4 个)：short、long、signed、unsigned。
- 复杂类型关键字(5 个)：struct、union、enum、typedef、sizeof。
- 存储级别关键字(6 个)：auto、static、register、extern、const、volatile。
- 跳转结构关键字(4 个)：return、continue、break、goto。
- 分支结构关键字(5 个)：if、else、switch、case、default。
- 循环结构关键字(3 个)：do、for、while。

我们现在不准备对它们做更多解释。随着学习的不断深入，读者会逐步接触并记住它们，目前只需要了解关键字这个概念即可。

除关键字外，C 语言中还有一些预定义的标识符，如下所示，这些标识符也不能用作变量名、函数名等。

#define	#endif	#ifdef	#ifndef
#include	#line	#undef	

命名问题并非 C 语言特有，每种程序语言都会规定对象的命名形式。除了不能使用关键字和预定义的标识符之外，编写程序时几乎可以使用任何合法的标识符对自己定义的对象进行命名，名称可以自由选择。命名问题并非一件无关紧要的事情。合理选择程序对象的名称能为写程序、读程序提供有益的提示。因此，我们倡导采用能说明程序对象内在含义的标识符。换言之，程序对象的名称应当具有一定的意义，尽可能做到“见名知义”。与标识符 cp 相比，诸如 current_page 这样的标识符虽然长，却更有助于识别和使用。

2.4 数据类型

C 语言要求在定义变量时就指定变量的数据类型。为什么要这样做呢？在数学中，数值是不分类型的。例如，7 与 8 的乘积是 56，1 除以 3 的值是 0.33333…；而在计算机中，数据是存

放在存储单元中的，这些存储单元在物理上是真实存在的，它们由有限数量的字节构成，所以在计算机中既不可能存放“无穷大”的数，也不可能存放位数无限多的小数。

C 语言允许使用的数据类型如图 2-4 所示。其中，基本类型(包括整型和实型)和枚举类型的值都是数值，统称算术类型。算术类型和指针类型统称纯量类型，因为它们的值是以数字来表示的。枚举类型是程序中由用户定义的整数类型。数组类型和结构体类型统称组合类型，共用体类型不属于组合类型，因为在同一时间内只有一个成员有值。函数类型用来定义函数，包括函数返回值的数据类型和参数类型。

不同类型的数据在内存中占用的存储单元的长度是不同的。例如，Visual C++ 6.0 会为 char 型(字符型)数据分配 1 字节，而为 int 型(基本整型)数据分配 4 字节。另外，用来存储不同类型数据的方法也是不同的。本节只介绍基本数据类型的应用，其他数据类型将在后续章节中逐步介绍。

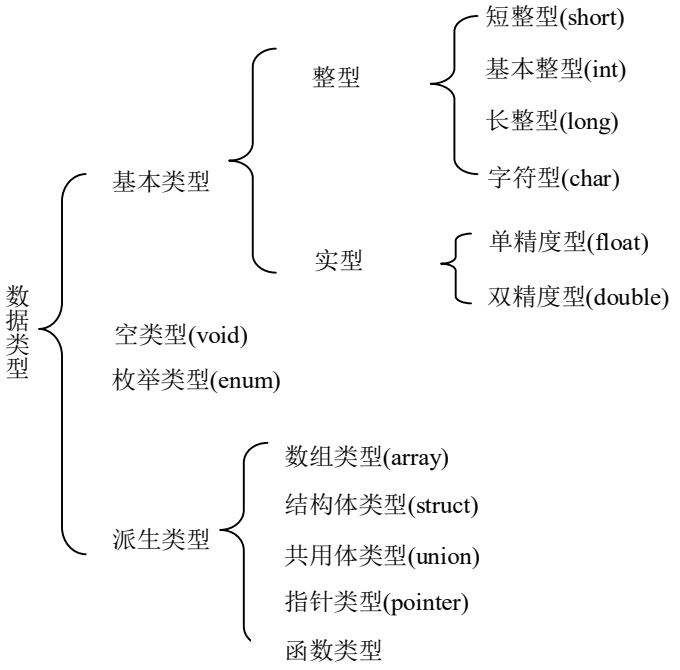


图 2-4　数据类型

2.4.1　整型数据

1. 整型常量的表示方法

在 C 语言中，整型常量可以分别用十进制、八进制和十六进制形式来表示。

(1) 十进制整数不以数字 0 开头。

例如：123 −4560

(2) 八进制整数以数字 0 开头。

例如：0123=$(123)_8$=$(83)_{10}$　　−011=$(-11)_8$=$(-9)_{10}$

(3) 十六进制整数以 0x 或 0X 开头。

例如：0x123=$(123)_{16}$ =$(291)_{10}$　　−0x12=$(-12)_{16}$=$(-18)_{10}$

有了上述 3 种表示方法，就可以对整型变量进行赋值了，如下所示：

```
int   n1= 60;          // n1 的值为(60)10
int   n2= 060;         // n2 的值为(48)10
int   n3= 0x6a;        //n3 的值为(106)10
```

2. 整型数据在内存中的存放形式

在计算机中，内存的最小存储单位是“位”(bit)。计算机会把 8 个二进制位组成 1 个“字节”(Byte)，并给每一个字节分配一个地址。整型数据在内存中是以二进制形式存放的，事实上，正整数以“原码”形式存放，负整数以“补码”形式存放。

补码的计算方法如下：正数的补码就是此数的二进制形式，也就是原码，符号位用 0 表示；对于负数来说，则首先将此数的绝对值写成二进制形式(原码)，然后对所有二进制位按位取反加 1，符号位用 1 表示。

例如，整数 10 和−10 的补码分别为：

$10=(1010)_2=(0000，0000，0000，1010)_{原}=(0000，0000，0000，1010)_{补}$

$-10=(-1010)_2=(1000，0000，0000，1010)_{原}=(1111，1111，1111，0110)_{补}$

通过比较 10 和−10 的补码可以看出：正数的补码与对应负数的补码看上去明显不同。

3. 整型变量的定义

整型变量的基本类型为 int。通过添加修饰符，便可定义更多的整型数据，如表 2-2 所示。

表 2-2　Visual C++ 6.0 中定义的整型数据

类型	占用的字节数	数值范围
[signed] int	4	−2 147 483 648～2 147 483 647，也就是-2^{31}～$(2^{31}-1)$
[signed] short [int]	2	−32 768～32 767，也就是-2^{15}～$(2^{15}-1)$
[signed] long [int]	4	−2 147 483 648～2 147 483 647，也就是-2^{31}～$(2^{31}-1)$
[signed] long long	8	−9 223 372 036 854 775 808～9 223 372 036 854 775 807，也就是-2^{63}～$(2^{63}-1)$
unsigned [int]	4	0～4 294 967 295，也就是 0～$(2^{32}-1)$
unsigned short [int]	2	0～65 535，也就是 0～$(2^{16}-1)$
unsigned long [int]	4	0～4 294 967 295，也就是 0～$(2^{32}-1)$
unsigned long long	8	0～18 446 744 073 709 551 615，也就是 0～$(2^{64}-1)$

根据表达范围，整型可以分为基本整型(int)、短整型(short int)、长整型(long int)、双长整型(long long，这是 C99 新增的数据类型)。使用双长整型可以表示更大范围的整数，但会降低运算速度。

根据是否有符号，整型可以分为有符号(signed，默认)整型和无符号(unsigned)整型。有符号整型数的存储单元的最高位是符号位(0 表示正，1 表示负)，其余位为数值位；无符号整型数的存储单元的全部二进制位都用于存放数值本身而不包含符号。

常用整型变量的定义格式如下。

- 基本整型(int)变量占用 4 字节空间，定义格式如下：

```
int 变量名表;                    //定义有符号的基本整型变量
unsigned int 变量名表;           //定义无符号的基本整型变量
```

- 短整型(short int)变量占用 2 字节空间，定义格式如下：

```
short 变量名表;                  //定义有符号的短整型变量
unsigned short 变量名表;         //定义无符号的短整型变量
```

- 长整型(long int)变量占用 4 字节空间，定义格式如下：

```
long 变量名表;                   //定义有符号的长整型变量
unsigned long 变量名表;          //定义无符号的长整型变量
```

【例 2-2】定义整型变量。

教学视频

```
#include <stdio.h>
void main()
{
    int a,b,c,d;
    a=7;
    b=-7;
    c=a+b;
    d=b-a;
    printf("%d,%d\n",c,d);
}
```

程序运行结果如图 2-5 所示。

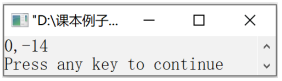


图 2-5　程序运行结果

注意：

(1) 定义变量时，数据类型说明符与变量名之间至少要用一个空格分开，在最后一个变量名之后，必须以分号结尾。

(2) 可以一次性定义多个相同类型的变量，各个变量之间需要使用逗号进行分隔。

(3) 变量在使用之前必须先定义。

(4) 可以在定义变量的同时对变量进行赋值。

(5) 在 Visual C++ 6.0 中，由于为 int 型和 long int 型数据分配的都是 4 字节的存储单元，因此 int 型和 long int 型变量的取值范围一致，一般使用 int 型变量即可。

2.4.2　实型数据

实型数据又称浮点数据，主要用来表示具有小数点的数据。

1. 实型常量的表示方法

实型常量有两种表示方法。

- 小数形式，这种实型常量由数字和小数点组成(必须有小数点)。例如：1.234、1234.、123.4、0.0。
- 指数形式，这种形式与数学中的指数形式类似，格式为aEn，表示 $a\times10^n$。例如：1.23e2表示 1.23×10^2，567e8 表示 567×10^8。

注意：

(1) 使用指数形式表示实数时，字母 e 或 E 之前必须有数字，e 后面的指数必须为整数。例如：e3、2.1e3.5、.e3、e 都不是合法的指数形式。

(2) 所谓规范化的指数形式，是指字母 e 或 E 之前的数用小数表示，并且小数点的左边应当有且只有一位非零数字。例如：2.3478e2、3.0999E5、6.46832e12 都是规范化的指数形式。

(3) 编译系统会将实型常量作为双精度实数来处理，这样可以保证较高的精度，缺点是运算速度会降低。在实型常量的后面加小写字母 f 或大写字母 F，如 1.65f、654.87F，编译系统将按单精度型处理实数；在实型常量的后面加小写字母 l 或大写字母 L，如 1.65l、654.87L，编译系统将按长双精度型处理实数。

2. 实型数据在内存中的存放形式

单精度型数据在内存中一般占 4 字节(32 位)的存储空间。与整数的存储方式不同，实型数据是按照指数形式存储的。系统将实型数据分为小数部分和指数部分分别存放。例如，实型数据 3.14159 在存储空间中的存放示意图如图 2-6 所示。

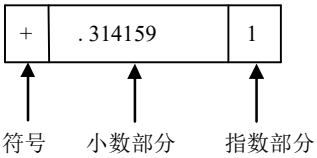


图 2-6　实型数据的存放示意图

C 语言并没有规定用多少位表示小数部分，也没有规定用多少位表示指数部分，这些都由 C 编译系统来定。例如，很多编译系统以 24 位表示小数部分，以 8 位表示指数部分。小数部分占用的位数多，表示实型数据的有效位数也多，因此精度高；指数部分占用的位数多，则表示实型数据的取值范围较大。

3. 实型变量的定义

C 语言中的实型变量分为单精度(float)、双精度(double)和长双精度(long double)三种类型，它们的定义格式如下。

- 单精度实型变量占用 4 字节的存储空间，定义格式为：float 变量名表;。
- 双精度实型变量占用 8 字节的存储空间，定义格式为：double 变量名表;。
- 长双精度实型变量占用 16 字节的存储空间，定义格式为：long double 变量名表;。

表 2-3 展示了实型数据占用的存储空间以及可以表示的数值范围。

表 2-3　实型数据的相关情况

类型	占用的字节数	有效数字位数	数值范围
float	4	7	$-3.4e^{38}$～$3.4e^{38}$
double	8	15	$-1.7e^{308}$～$1.7e^{308}$
long double	8	15	$-1.7e^{308}$～$1.7e^{308}$
	16	19	$-3.4e^{4932}$～$3.4e^{4932}$

注意：

(1) 实型变量也应该先定义，后使用。

(2) 在 Visual C++ 6.0 中，所有的 float 型数据都会被自动转换成 double 型进行处理。

(3) 不同的编译系统针对 long double 型数据的处理方法也不同。Turbo C 会为 long double 型数据分配 16 字节的存储空间；而 Visual C++ 6.0 则对 long double 型和 double 型数据一视同仁，都分配 8 字节的存储空间。请读者在使用不同的编译系统时注意以上差别。

4. 实型数据的精度

在计算机中，数据是使用有限的存储单元进行存储的，因此实型数据提供的有效数字位数是有限的，处于有效位之外的数字将变得不准确，并由此可能引起实型数据的舍入误差。

【例 2-3】将一个有效数字位数超过 7 位的数赋值给 float 型变量时，将引起实型数据的舍入误差，但这并不是程序本身的错。

教学视频

```
#include <stdio.h>
void main()
{
   float x=123456789;
   double y=123456789;
   x=x-50;
   y=y-50;
   printf("\nx=%f,y=%f",x,y);
}
```

程序运行结果如图 2-7 所示。

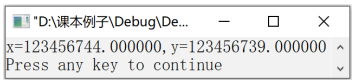


图 2-7　程序运行结果

从运行结果可以看出：float 型变量 x 只准确接收了前 7 位数据，从第 8 位开始数据变得不再准确。因此，当使用实型数据时，一定要注意可能产生的舍入误差。

实型数据由于存在舍入误差，因此在使用时请注意以下事项：

(1) 不要试图用实数精确表示大的整数。记住：实数是不精确的。实数若超出有效位，则超出部分会变得不准确，产生误差。

(2) 一般不判断实数是否“相等”，而是判断是否接近或近似。应避免直接将一个很大的

数与一个很小的数相加或相减，否则可能会“丢失”那个很小的数。

(3) 应根据运算需求选择使用单精度实型还是双精度实型。

2.4.3 字符型数据

1. 字符型变量

在 C 语言中，字符型变量是使用关键字 char 进行定义的，在定义的同时可以赋初值。字符型变量的定义格式一般如下：

```
char 变量名表;
```

在定义字符型变量时，应注意变量的命名应符合标识符的命名规则。下面是字符型变量的定义范例：

```
char c1,c2='s';
```

上述语句定义了两个字符型变量 c1 和 c2，同时变量 c2 被赋值为's'。

2. 普通字符常量

有两种形式的字符常量：一种是普通字符常量；另一种是转义字符常量。

普通字符常量是使用单引号括起来的单个字符，如'a'、'Z'、'3'、'?'、'#'，不能写成'ab'或'12'。请注意，单引号只是定界符，字符常量只能是单个字符，不包括单引号。当把字符常量存储到计算机的存储单元中时，并不是存储字符本身(如 a、Z、#等)，而是存储对应的编码(一般采用 ASCII 码)。例如，字符'a'的 ASCII 码值是 97，因此存储单元中存放的是 97(以二进制形式存放)。

注意：

(1) 单引号中的字符如果大小写形式不同，则代表不同的字符。例如，'a'和'A'是不同的字符常量。

(2) 空格(' ')也是字符常量，不能写成两个连续的单引号('')。

(3) 字符常量只能包含一个字符。因此，c1='a'和 c2='A'是合法的，而 c1='ok'和 c2='we'是非法的。

(4) 字符常量只能用单引号括起来，而不能用双引号括起来。因此，"A"不是字符常量，而是一个字符串。

3. 转义字符常量

除以上形式的普通字符常量外，C 语言还允许使用一种特殊形式的字符常量，就是以反斜杠(\)开头的字符序列。例如，printf 函数中的'\n'表示换行，'\t'表示跳转到下一个水平制表位置。以上这些无法在屏幕上显示的“控制字符”，在程序中也无法用普通字符来表示，而只能采用这样的特殊表示形式。也可以这样理解，当计算机遇到反斜杠(\)开头的字符序列时，就会对它们进行转义处理。表 2-4 列出了常用的转义字符及其含义。

表 2-4　常用的转义字符及其含义

转义字符	字符值	输出结果
\n	换行符	将当前位置移到下一行的开头
\t	水平制表符	将当前位置移到下一个水平制表位置
\v	垂直制表符	将当前位置移到下一个垂直制表位置
\b	退格符(backspace)	将当前位置后退一个字符
\r	回车符(carriage return)	将当前位置移到本行的开头
\f	换页符(form feed)	将当前位置移到下一页的开头
\a	警告(alert)	产生报警音
\\	反斜杠(\)	输出反斜杠字符
\?	问号(?)	输出问号字符
\'	单引号(')	输出单引号字符
\"	双引号(")	输出双引号字符
\0	空字符	输出空字符
\o、\oo、\ooo o 代表八进制数字	八进制数对应的字符	最多与 3 位八进制数对应的字符
\xh、\xhh h 代表十六进制数字	十六进制数对应的字符	最多与两位十六进制数对应的字符

【例 2-4】使用转义字符控制输出。

教学视频

```
#include <stdio.h>
void main()
{
  printf("\n\t\101");          // 将光标移到下一行的行首，再移到下一个水平制表位置，输出 A
                               // 将(101)8=(65)10，对应 A 的 ASCII 码值
  printf("\n\t\bb");           // 将光标移到下一行的行首，再移到下一个水平制表位置
                               // 然后将光标后退一格，输出 b
  printf("\n\\*hello*\\");     // 将光标移到下一行的行首，输出\*hello*\
  printf("\n\t\x41");          // 将光标移到下一行的行首，再移到下一个水平制表位置
                               // (41)16=(65)10，对应 A 的 ASCII 码值
  printf("\a\a\a");            // 如果计算机的声音设备已打开，就可以听到 3 声报警音
  printf("\n");                // 将光标移到下一行的行首
}
```

程序运行结果如图 2-8 所示。

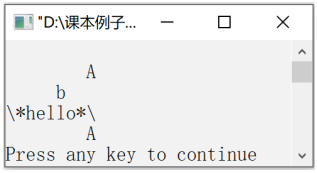


图 2-8　程序运行结果

表 2-5 展示了几种在使用转义字符常量时可能遇到的错误。

表 2-5　几种常见的非法转义字符常量

非法转义字符常量	错误原因
'\197'	9 不能充当八进制数的数位
'\1673'	\后面最多跟 3 位八进制数
'\abc'	\后面漏掉了 x，并且位数超过了 3 位
'ab'	普通字符常量，单引号之间只能有一个字符

4. 字符型数据的运算

当把字符放入字符型变量时，字符型变量中存放的是字符的 ASCII 码值。在所有数据类型中，字符型数据占用的内存空间最少，仅占用 1 字节空间。当把字符常量存储到计算机的存储单元中时，并不存储字符本身，而是存储字符的编码(一般采用 ASCII 码)。因此，字符型变量可以作为整型变量来处理，并参与整型变量允许的各种运算。

例如：

```
a='A';        // a=65
b='A'+5;     // b=65+5
c='0'+'3';      // c=48+51
```

由此可见，字符型数据以 ASCII 码存储的形式与整数的存储形式类似，因此字符型数据还可以当作较短的整型数据使用。字符型变量只占用 1 字节空间，如果作为整型变量使用的话，取值范围为 0~255(无符号整数)或−128~127(有符号整数)。

说明：

(1) 字符型变量具有双重性，既可以解释为字符，也可以解释为整数。

(2) 可以对字符型数据进行算术运算，相当于对相应的 ASCII 码值进行算术运算。

(3) 字符型数据既可以字符形式输出，也可以整数形式输出。

例如：

```
char c='A';
printf("%d\n",c);      //输出数字 65
printf("%c\n",c);      //输出字母 A
```

【例 2-5】练习字符'a'的各种表示方法，理解如何把字符型变量当整型变量使用。

教学视频

```
#include <stdio.h>
void main()
{
   char c1='a';                  //字符'a'的 ASCII 码值为 97
   char c2='\x61';               //字符'a'的十六进制表示
   char c3='\141';               //字符'a'的八进制表示
   char c4=97;                   //(97)10
```

```
    char c5=0x61;                    //(0x61)16=(97)10
    char c6=0141;                    //(0141)8=(97)10
    printf("\nc1=%c,c2=%c,c3=%c,c4=%c,c5=%c,c6=%c\n",c1,c2,c3,c4,c5,c6);
    printf("c1=%d,c2=%d,c3=%d,c4=%d,c5=%d,c6=%d\n",c1,c2,c3,c4,c5,c6);
}
```

程序运行结果如图 2-9 所示。

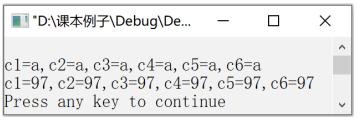


图 2-9　程序运行结果

【例 2-6】对字母进行大小写转换。提示：小写字母的 ASCII 码值比对应的大写字母的 ASCII 码值大 32。

教学视频

```
#include <stdio.h>
void main()
{
    char c1,c2;
    c1='a';
    c2='B';
    c1=c1-32;
    c2=c2+32;
    printf("c1=%c,c2=%c\n",c1,c2);
    c1=c1+3;
    c2=c2+3;
    printf("c1=%c,c2=%c\n",c1,c2);
}
```

程序运行结果如图 2-10 所示。

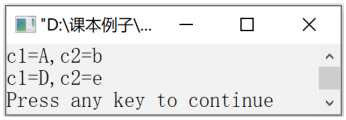


图 2-10　程序运行结果

2.4.4　字符串常量

字符串常量是由一对双引号(" ")括起来的字符序列，例如"How are you?"、"CHINA"、"a"、"C 语言程序设计"。C 语言中有字符串常量但并没有字符串变量，如果需要处理字符串数据，可以使用后续章节中介绍的字符数组和指针。

说明：

(1) 请注意区分字符常量与字符串常量。例如，"a"是字符串常量，而'a'是字符常量。

(2) C 语言规定必须以'\0'(ASCII 码值为 0)字符作为字符串的结束标志。例如，"CHINA"在内存中占用的存储空间大小是 6 字节。

(3) 不能把字符串常量赋给字符型变量。例如，如果 c 为字符型变量，那么语句 c="a";是错误的。

(4) C 语言中没有字符串变量，可以使用字符数组或指针处理字符串数据。

(5) 两个连续的双引号("")也是字符串常量，称为“空串”，但需要占用 1 字节的存储空间以存放'\0'。

2.5 运算符及表达式

运算符是构建表达式的基本工具，大多数编程语言都提供了以下运算符：

(1) 算术运算符，包括加、减、乘、除等。

(2) 关系运算符，用于执行诸如“i 比 0 大”这样的关系比较运算。

(3) 逻辑运算符，用于执行诸如“i 比 0 大但比 10 小”这样的逻辑运算。

此外，还有许多其他运算符。虽然掌握如此众多的运算符是一件非常烦琐的事，但它们对于编写程序特别重要。

2.5.1 运算符的分类

1. 按功能分类

(1) 算术运算符：用于各类数值运算，包括加(+)、减(−)、乘(*)、除(/)、求余(或称模运算%)、自增(++)和自减(--)，共 7 种。

(2) 关系运算符：用于比较运算，包括大于(>)、小于(<)、等于(==)、大于或等于(>=)、小于或等于(<=)和不等于(!=)，共 6 种。

(3) 逻辑运算符：用于逻辑运算，包括与(&&)、或(||)和非(!)，共 3 种。

(4) 位运算符：按二进制位进行运算，包括位与(&)、位或(|)、位非(~)、位异或(^)、左移(<<)和右移(>>)，共 6 种。

(5) 赋值运算符：用于赋值运算，包括简单赋值(=)、复合算术赋值(+=、-=、*=、/=、%=)和复合位运算赋值(&=、|=、^=、>>=、<<=)三类，共 11 种。

(6) 条件运算符：记为“?:”，这是一种三目运算符，用于条件求值。

(7) 逗号运算符：记为“,”，用于把若干表达式组合成单个表达式。

(8) 指针运算符：用于指针运算，包括取值运算符(*)和取址运算符(&)，共两种。

(9) 求字节数运算符：记为 sizeof，用于计算数据类型所占的字节数。

(10) 特殊运算符：比如下标运算符([ ])、指向运算符(->)和成员运算符(.)。

2. 按操作数分类

操作数是指运算符连接的对象，按操作数分类是指按照运算符连接的对象个数进行分类。

(1) 单目运算符(带 1 个操作数)。

! ~ ++ -- - * & sizeof

(2) 双目运算符(带 2 个操作数)。

+ - * / % < <= > >= == !=

<< >> & ^ | && ||