普通高等教育农业农村部“十四五”规划教材

高等院校计算机应用系列教材

C 语言程序设计教程

(第三版)(微课版)

王娟勤　主　编

成宝国　晁晓菲　副主编

清华大学出版社

北　京

内 容 简 介

本书从培养学生利用程序设计解决问题的角度出发，以案例为引导，介绍了 C 语言程序设计基础、基本数据类型、数据运算、程序的基本结构、数组、指针、函数、结构体、共用体、枚举类型、文件、底层程序设计和编译预处理等内容。书中提供了丰富的实用案例、趣味性知识及与人工智能相关的实例，配有微视频讲解，对问题做了深入浅出的分析和总结，有助于引领读者理解编程思维和学习编程技能；每章都配有综合案例，为升华知识提供桥梁；各章的知识结构图，有助于学生理清知识脉络；精选的典型习题，对进一步深化基础知识、提升分析问题和解决问题的能力具有重要作用。

本书采用导学、易学编写策略，每章安排有内容提示、教学基本要求、微视频讲解和总结，正文组织本着由浅入深、循循善导的原则，突出重点和难点。全书逻辑清晰，层次分明，例题丰富。本书既可作为高等院校本科各专业的公共课教材，也可作为C 语言程序设计爱好者及自学人员的参考书。

本书配套的电子课件、习题答案和实例源代码可以到 http://www.tupwk.com.cn/downpage 网站下载，也可以通过扫描前言中的二维码下载。扫描正文中的视频二维码可以直接观看教学视频。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。举报：010-62782989，beiqinquan@tup.tsinghua.edu.cn。

图书在版编目(CIP)数据

C 语言程序设计教程 ：微课版 / 王娟勤主编. 3 版. -- 北京 ：清华大学出版社, 2025. 7. -- (高等院校计算机应用系列教材). -- ISBN 978-7-302-69594-3

Ⅰ. TP312.8

中国国家版本馆 CIP 数据核字第 2025TF5524 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：成凤进
责任印制：沈　露

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：三河市人民印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**20　　**字　　数：**512 千字
版　　次：2017 年 8 月第 1 版　　2025 年 8 月第 3 版　　**印　　次：**2025 年 8 月第 1 次印刷
定　　价：79.80 元

产品编号：107898-01

前　言

如果你想训练严谨的逻辑思维、展现你的设计、智慧，想用计算机编程解决生活中的问题，想探索人工智能领域的奥秘，那就来学习程序设计吧！

本书从初学者的角度出发，以 C 语言为工具，以现实生活中的案例为引导，说明如何分析问题、利用程序设计解决问题的思维方法。书中详述了应用程序的开发，由浅入深，逐步启发、引领学生学习编写规模逐渐加大的程序，将程序设计的基本思想方法和魅力逐步展现出来。

本书依据知识结构将内容共分为 10 章，第 1 章为 C 语言程序设计概述，介绍与程序设计有关的概念，说明 C 语言程序的基本组成、C 语言程序设计开发环境与过程；第 2 章为 C 语言基础，介绍 C 语言的基本数据类型，常量和变量，C 语言的基本运算符、表达式及应用，数据的输入和输出函数的使用；第 3 章为程序设计基本结构，介绍程序的 3 种基本结构，实现选择和循环结构的语句及其应用，介绍常见问题及解决问题的方法；第 4 章为数组，介绍数组的基本概念、使用及处理数组中数据的常用方法；第 5 章为指针，介绍指针的概念，指针的使用，利用指针处理数据的方法；第 6 章为函数，介绍函数的定义、调用及函数间数据传递的方法，说明变量的存储属性及其使用；第 7 章为结构体、共用体与枚举类型，介绍结构体和共用体的定义、使用和区别，介绍枚举类型的定义和使用；第 8 章为文件，介绍文件的基本概念，文件的操作步骤，利用文件实现内存和外存中数据交换的方法；第 9 章为底层程序设计，介绍位运算的运算符号、规则及应用；第 10 章为编译预处理，介绍编译预处理命令的使用、编写大型程序的方法等。每章都提供了适合该章知识点的综合案例，以拓展知识、开阔学生的眼界。

全书在内容组织方面突出以下特色。

(1) 结构新颖。依据所介绍的知识，每章除了安排富有趣味性的实例，还安排了有助于学生升华知识点的综合案例。这些案例均源于生活或学习中的实际应用需求，能够让学生在任务的驱动下，由浅入深，学习和编写规模逐渐加大的程序，在潜移默化中逐步让学生了解、学习计算机如何解决各类问题，从而掌握利用计算机解决问题的方法。

(2) 提供丰富实用的编程“套路”。从典型的程序实例中总结出“套路”，即解决一类问题的方法，从而让初学者迅速掌握基础编程的方法和算法，具有解决实际问题的能力。

(3) 助学。每章安排有内容提示、教学基本要求，例题从问题分析、算法描述、问题总结、注意事项等方面进行完整论述，每章包含知识总结和习题等内容，有利于教师组织教学，也有助于学生进行预习与复习。

(4) 易学。本着“知识量最小而收获最大”的原则，突出主线和重点，分解难点，以循序

渐进的方法，力求让学生对于难点部分学得轻松，知识点掌握牢固。

(5) 想学。例题选材上注重知识性、趣味性和经典性相结合，尽力降低学习过程中的枯燥感，增强学生学习的“幸福指数”。

(6) 爱做。本书习题丰富，每章配有形式多样的习题，尽力吸引学生学后爱做、自觉温故知新。

(7) 配备线上线下立体资源。为配套资源配置了二维码，将内容讲解可视化，学生通过扫描二维码可以观看短视频，领悟知识内涵，学习怎样分析问题和设计、编写代码解决问题。线上线下的立体资源，便于学生预习、复习和自学，方便师生加强课堂互动，进行线上线下混合式教学。

本书在继承第一版和第二版优秀内容的基础上，融合了教学团队在一流课程建设过程中积累的经验和成果。结合教学实践及当前高校计算机基础教育的新要求和融入人工智能的需求，我们对教材内容进行了优化和调整，使其成为一本易于阅读、实用且能够促进思考、辅助培养创新应用能力的教材。

本书由李书琴主审、王娟勤主编。第 1~6 章由王娟勤编写，第 7、8 章由成宝国编写，第 9 章由任国霞编写，第 10 章由晁晓菲编写。

在全书的策划和编写过程中，孙健敏和承担“C 语言程序设计”课程的各位老师，对本书提出了很多宝贵意见并给予了帮助，在此表示最诚挚的感谢。

由于编者水平有限，书中的不足、疏漏之处在所难免，恳请广大读者提出宝贵意见和建议。我们的信箱是 992116@qq.com，电话是 010-62796045。

本书配套的电子课件、习题答案和实例源代码可以到 http://www.tupwk.com.cn/downpage 网站下载，也可以通过扫描下方的二维码下载。扫描正文中的视频二维码可以直接观看教学视频。

书中图标的释义

本书包含一些强调特定知识点的图标，它们能够直观地标识注意、警告、提示、总结和说明等内容。

 需要引起注意的内容。

 防止错误，容易出错的地方。

 提出问题，或大家感觉有疑问的地方。

 重点知识，或有总结内容的地方。

 程序或知识点说明、论述。

强调特定知识点的地方。

 提升知识深度、难度，提高和进阶的内容。

 用于标识使用此方法的优势、优点、好处。

 问题的“设计思路”。

编　者

2025 年 6 月

目　录

第 1 章

C 语言程序设计概述

📖 **本章内容提示：**

介绍 C 语言程序的基本组成、开发 C 语言程序的一般步骤以及 C 语言集成开发环境，让初学者对 C 语言程序的组成结构有大致的了解，并介绍 C 语言的学习方法。

📖 **教学基本要求：**

了解 C 语言程序的基本组成，熟悉 C 语言程序的开发过程，能够在 C 语言集成开发环境中完成对例题的编辑、编译、连接和运行，并得到正确的结果，为后续章节的学习打下良好的基础，并可以开发一个简单的 C 程序。


程序与程序设计语言

计算机语言是人与计算机交流的一种工具，要编写程序、深入地理解计算机的工作原理，就必须学习和掌握计算机语言。在计算机问世的几十年中，出现了多种计算机语言，总体上可以将计算机语言分为机器语言、汇编语言和高级语言三大类。

高级语言接近人类语言和人们习惯使用的数学用语，不依赖于具体的机器，有严格的语法规则。相对于机器语言和汇编语言，高级语言易学易用，所编写的程序易读易改，通用性更强。

C 语言是世界上广泛流行的计算机高级程序设计语言，它于 1973 年由美国贝尔实验室设计发布。由于 C 语言同时具备高级语言的优点和低级语言的功能，而且拥有很好的可移植性，因此它成为程序员最喜欢的编程语言之一。它以功能强大、数据结构丰富、目标代码质量高、程序运行效率高、可移植性好等特点成为众多程序员学习程序设计的首选语言。

1.1 C 语言程序的基本组成

下面通过两个例题来了解 C 语言程序的基本组成。

【例 1-1】在屏幕上输出“Hello,World!”。

```
//example1.1 The first C Program
#include <stdio.h>
int main(void)
{
```

```
    printf("Hello,World!\n");
    return 0;
}
```

程序分析：


C 语言程序基本组成 1


C 语言程序基本组成 2


C 语言程序基本组成 3

(1) main 函数——从哪里开始，到哪里结束。main( )表示“主函数”，每个 C 语言程序都必须有且只能有一个 main 函数，它是每一个 C 语言程序执行的起始点(入口点)和终止点。程序是从 main 函数的第一条语句开始执行，然后顺序执行 main 函数中的其他语句。main 函数执行结束后，整个程序的执行也就结束了，而不论 main 函数书写在程序中的任何位置。

int 表示该主函数返回一个整数值(状态值)。

void 表示 main 函数没有参数，main 函数无须从命令行接收任何信息(main 函数可以带两个参数，通常命名为 argc 和 argv，具体参见本书 6.5 节)。

语句“return 0; ”有两个作用：一是使 main 函数终止(即结束程序)；二是指出 main 函数的返回值是 0(状态值)，这个值表明程序正常终止。为了表示异常终止，main 函数应该返回非 0 值，(实际上，这一返回值也可以用于其他目的) 。即使不打算使用状态值，确保每个 C 程序都返回状态值也是一个很好的习惯，因为将来运行程序的人可能需要测试状态值。

用{和}括起来的语句是主函数 main 的函数体。main 函数中的所有操作(或语句)都包含在这一对花括号之间，即 main 函数的所有操作都在 main 函数体中。

(2) 如何输出数据——函数调用。本程序的主函数 main 中只有一条函数调用语句——printf( )，它是 C 语言的库函数，用于程序中数据的输出(显示在显示器上)。本例是将一个字符串“Hello,World!\n”输出，即在显示器上显示“Hello,World!”。

(3) 程序的最小独立单元——语句。每条语句都以“;”结束。

(4) 编译器如何识别 printf 函数——#include 预处理命令。C 语言规定，函数在调用前必须先声明。printf 函数的声明就包含在头文件“stdio.h”中，头文件是每一个 C 程序必不可少的组成部分，因为一个 C 程序至少要包含输入或输出函数。

(5) 程序里的说明书——注释。“/* example1.1 The first C Program*/”为注释。注释是为了改善程序的可读性(提示、解释作用)，在编译、运行时不起作用(编译时会跳过注释，目标代码中不会包含注释)。注释可以放在程序的任何位置，并允许占用多行，只是需要注意“/*”要与“*/”匹配，不要嵌套注释。

也可以使用“//”作为注释符，一个“//”只能注释一行，而“/*”和“*/”配对可以注释多行。

注释与软件的文档同等重要，要养成书写注释的良好习惯，这对软件的维护相当重要，这是因为程序是要给别人看的(自己也许还会看自己几年前编写的程序)，清晰的注释有助于读者理解算法和程序的思路。

在程序开发过程中，还可以用注释帮助调试程序，即暂时屏蔽一些不需要运行的语句，以后可以方便地恢复。

【例 1-2】求两个整数中的较大者。

```
#include <stdio.h>
int max(int x,int y)                /* 求两个整数中的较大者 */
{ int z;                              /* 声明部分，定义变量 */
  if(x>y)
     z=x;
  else
     z=y;
  return z;                         /* 将 z 值返回，通过 max 带回调用处 */
}
int main(void)
{ int a,b,c;                        /* 声明部分，定义变量 */
  scanf("%d,%d",&a,&b);
  c=max(a,b);                       /* 调用 max，将调用结果赋给 c */
  printf("max=%d",c);
  return 0;
}
```

程序分析：

(1) 本程序包括两个函数。其中，主函数 main 仍然是整个程序执行的起点，函数 max 计算两数中较大的数。

(2) 主函数 main 调用 scanf 函数，获得两个整数，分别存入 a、b 两个变量中，然后调用函数 max，获得两个数中较大的数，并赋给变量 c。最后输出变量 c 的值(结果)。

(3) max 是用户自定义的函数，int max(int x,int y)是函数入口，表示此函数运行时需要获得两个整数值，数据处理结束后会返回一个整数值。

(4) 函数 max 同样也用{和}将函数体括起来。max 的函数体是函数功能的具体实现，它从参数表获得数据，将处理后得到的结果存储于 z 中，然后将 z 返回调用函数 main。

(5) 本例表明函数除调用库函数外，还可以调用用户自定义的函数。

综合上述两个例子，我们对 C 语言程序的基本组成和程序结构有了一个初步了解。

(1) C 语言程序由函数构成(函数是 C 程序的基本单位)，所有的 C 语言程序都由一个或多个函数构成。其中，main 函数必须有且只能有一个。

(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己设计编写的函数。程序的全部工作由各个函数完成。编写 C 语言程序就是编写一个个的函数。

(3) main 函数(主函数)是每个程序执行的起始点。无论 main 函数在程序中的哪个位置，一个 C 语言程序总是从 main 函数开始执行，并且也是从 main 函数结束。

(4) 一个函数由函数首部和函数体两部分组成。

① 函数首部：一个函数的第一行，由函数类型、函数名、参数类型和参数名组成。一般格式为：

```
函数类型 函数名(参数类型及参数列表)
```

② 函数体：函数首部下方用一对花括号括起来的部分。函数体一般包括声明和执行两部分。

例如，函数 max:

函数类型 函数名 参数类型 参数名 参数类型 参数名

```
int        max     (int      x,     int      y)    /* 函数首部 */
{                                                  /* 函数开始 */
  int z;                                           /* 声明部分，定义变量 */
  if(x>y)
    z=x;
  else
    z=y;                                           /* 函数体，执行部分*/
  return z;
}                                                  /* 函数结束 */
```

(5) C 语言程序的书写格式较自由，一行可以写几条语句，一条语句也可以写在多行上。每条语句的最后必须有一个分号“;”，表示语句的结束。

(6) 可以使用“/*”和“*/”的配对，或者“//”对 C 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。

一个较大的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。

(7) 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(scanf、printf 等)来完成的。

输入/输出操作涉及具体计算机硬件，把输入/输出操作放在函数中处理，可以简化 C 语言和 C 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做不同的处理，以便实现同样或类似的功能。

1.2 C 语言程序设计的一般步骤

计算机之所以能够产生如此大的影响，其原因不仅在于人们发明了机器本身，更重要的是，人们为计算机开发出了不计其数的能够指挥计算机完成各种工作的程序。正是这些功能丰富的程序给了计算机无尽的生命力，它们是程序设计工作者智慧的结晶。

所谓程序，是指用计算机语言描述的、为解决某一问题、满足一定语法规则的语句序列，而程序设计就是用某种程序语言编写解决这些问题的步骤的过程。

要设计出一个程序，首先应明确要处理问题中的数据结构(即数据和数据之间的关系)；其次应描述出对问题的处理方法和步骤(即算法)。因此，数据结构与算法是程序设计过程中密切相关的两个方面。著名计算机科学家 Niklaus Wirth 教授提出了关于程序的著名公式(沃思公式)：程序=数据结构+算法。这个公式说明了程序设计的主要任务。

那么，如何用 C 语言进行程序设计呢？一般包含以下步骤。

1. 分析问题

使用计算机解决具体问题时，首先要对问题进行充分的分析，确定问题是什么，针对所要解决的问题，思考程序需要哪些信息，要进行哪些计算和控制，以及程序应该要报告什么信息。在这一步骤中，不涉及具体的计算机语言，可以用一般术语来描述问题。

2. 确定数据结构和算法

在分析求解问题的基础上，确定程序中数据的类型和组织存储形式，即确定存放数据的结构。针对问题的分析和确定的数据结构，选择合适的算法加以实现。注意，这里所说的“算法”泛指解决某一问题的方法和步骤。

3. 编写与编辑源程序


运行C程序的步骤与方法

根据确定的数据结构和算法，用 C 语言把该数据结构和算法严格地用符合 C 语言语法规则的代码描述出来，也就是编写出源程序代码。将源程序输入计算机，并以约定的扩展名“.c”的文本文件形式保存在外存上，例如 file1.c、t.c 等。

用于编辑源程序的软件是编辑程序。编辑程序是提供给用户书写程序的软件环境，可用来输入和修改源程序。

4. 编译、连接与运行

编译是把 C 语言源程序翻译成计算机能识别的二进制指令形式的目标程序。编译过程由编译程序完成。编译程序自动对源程序进行句法和语法检查，当发现错误时，将错误的类型和所在的位置显示出来，提供给用户，以帮助用户修改源程序中的错误。如果未发现句法和语法错误，则对目标代码进行优化后生成目标程序。目标程序的文件扩展名是“.obj”。

计算机能够读懂目标程序，但仍不能执行，这是因为其中还缺少一些内容(如库函数、其他目标程序、各种资源等的二进制程序)。需要把这些内容与目标程序“组合起来”，这个过程称为连接。经过连接之后，就生成了可执行程序，可执行程序的扩展名为“.exe”。连接过程是由连接程序(也称链接程序或装配程序)完成的。

运行程序是指将可执行程序投入运行，得到程序处理的结果。如果程序运行结果与预测结果不一致，必须重新回到第 3 步(也有可能会从第 1 步开始重新分析问题)，对程序进行编辑修改、编译和运行，直到得到正确的结果为止。

与编译、连接不同，运行可以脱离语言处理环境，直接在操作系统环境下进行。

5. 写出程序的文档

与正式产品需要提供产品说明书一样，程序作为提供给用户使用的工具，也必须附带程序说明书。说明书内容应包括以下几个方面：程序名称、程序功能、运行环境、程序的载入和启动方式、需要输入的数据，以及使用注意事项等。这些信息有助于用户更好地使用和维护程序。

许多软件包能够完成从第 3 和第 4 步的全部过程，这种软件包称为集成开发环境(Integrated Development Environment，IDE)，例如 Code::Blocks、Dev-C++等。在集成开发环境中开发 C 语言程序的过程可以用图 1-1 表示。

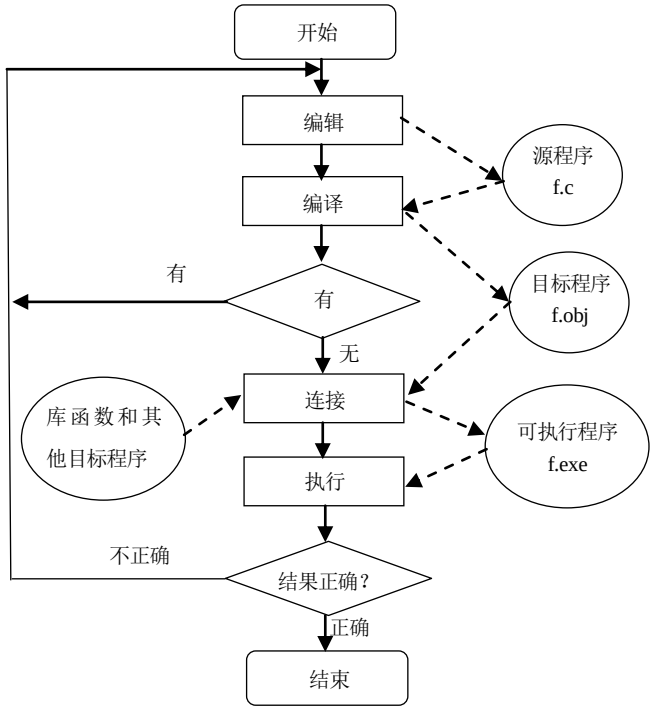


图 1-1　开发 C 语言程序的步骤

1.3　C 语言程序的上机执行过程

如前所述，编写 C 语言程序只是程序设计工作中的一个环节。编写完成的程序需要在计算机上进行调试运行，直到获得正确的运行结果为止。调试和运行程序一般要经过编辑、编译、连接和运行 4 个步骤，才能得到程序的运行结果。编辑、编译、连接和运行通常是在 C 语言集成开发环境中完成的。

Code::Blocks 是一个免费、开源且跨平台的 C/C++集成开发环境(IDE)，支持 Windows、Linux 和 macOS 等系统。该软件体积小巧灵活，具有跨平台支持、代码语法高亮、自动格式化和国际化等功能。

1. 编辑 C 源程序代码

从网上下载 Code::Blocks 的免费安装程序后(例如 Code::Blocks 20.03 版本)，安装并启动 Code::Blocks 软件，主界面如图 1-2 所示。

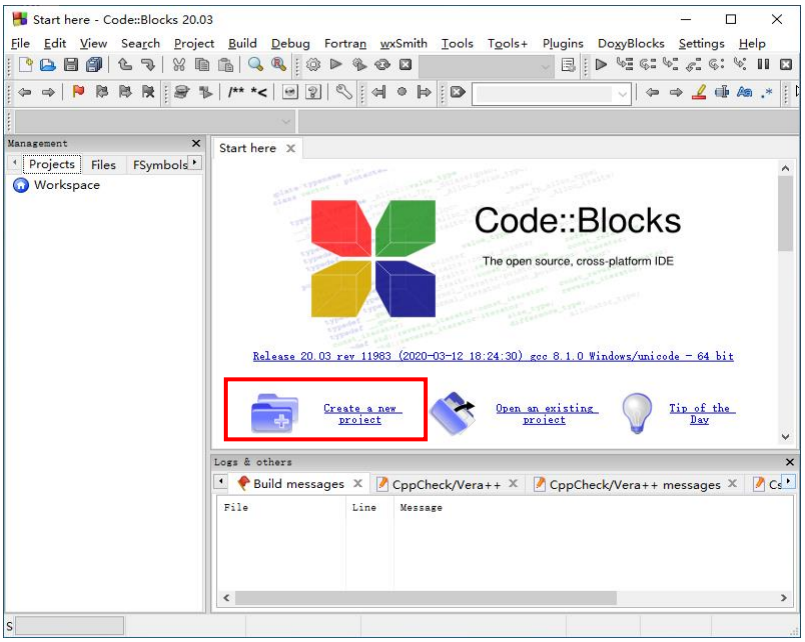


图 1-2　Code::Blocks 主界面

单击窗口右侧中的 Create a new project 图标新建一个项目，或选择 File | New | Project…命令来建新项目。此时会出现 New from template 对话框，如图 1-3 所示。

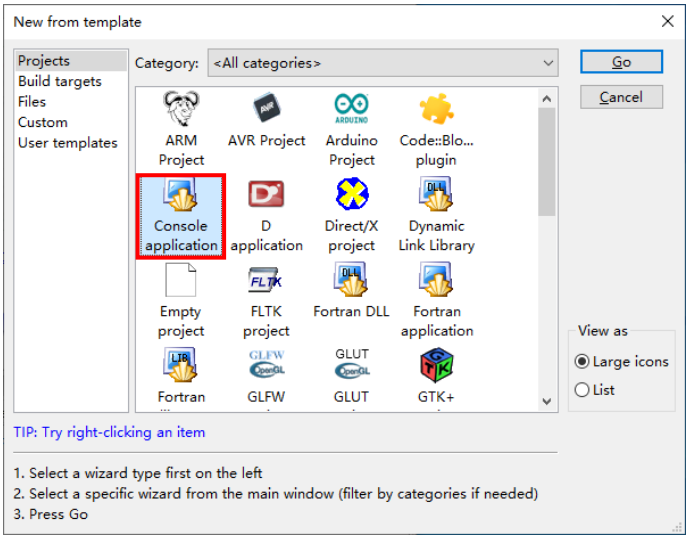


图 1-3　New from template 对话框

选择 Console application(控制台应用程序)，然后单击 Go 按钮，在打开的对话框中单击 Next 按钮，此时将打开语言选择对话框，如图 1-4 所示。在该对话框中选择 C 选项后，单击 Next 按钮。

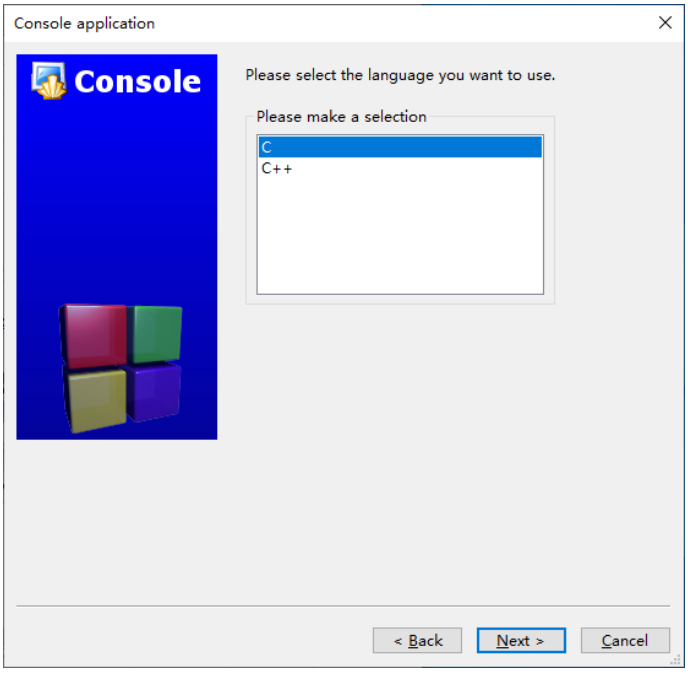


图 1-4　语言选择对话框

在图 1-5 所示的对话框中为新建的项目命名。

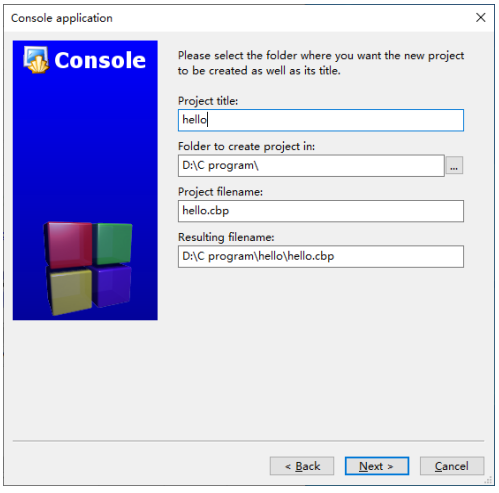


图 1-5　为新建项目命名

图 1-5 所示对话框中主要选项的功能说明如下。

- Project title(项目名称)：通常建议命名按“见名知意”的原则(如 hello)。
- Folder to create project in(项目保存路径)：可以单击该文本框后的省略号按钮来选择保存路径。
- Project filename(项目文件名)：该文件名会默认使用项目名称命名。例如，如果项目的名称为 hello，则项目文件名将为 hello.cbp。
- Resulting filename(最终的文件名)：此处将显示包含文件名的完整路径。

单击 Next 按钮，在随后打开的选择编译器及其参数设置对话框中保留默认值，然后单击 Finish 按钮，在打开的 Codeblocks 窗口中黑体显示的是项目“hello”，单击 Sources 左侧的“+”图标以展开列表，将看到项目下面的“main.c”，双击“main.c”以打开该文件，如图 1-6 所示。

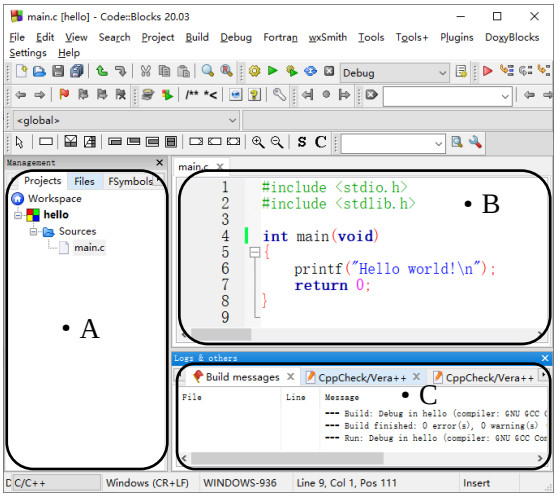


图 1-6　项目 hello 的窗口

图 1-6 窗口中 A、B、C 三个区域的说明如下。

- A(项目窗口)：列出项目中的所有资源及其他项目。
- B(源程序编辑窗口)：可以对源程序文件进行编辑。
- C(调试信息输出窗口)：用于输出程序编译和连接过程中的各种提示信息。

2. 编译、连接和运行程序

在源程序编辑窗口中录入源代码，保存后在工具栏上单击 按钮，完成源程序的编译和连接。如果没有错误，在调试信息输出窗口中会显示生成“.exe”文件。单击工具栏上的 按钮，运行.exe 文件，若程序正确，将会出现图 1-7 所示的显示结果。

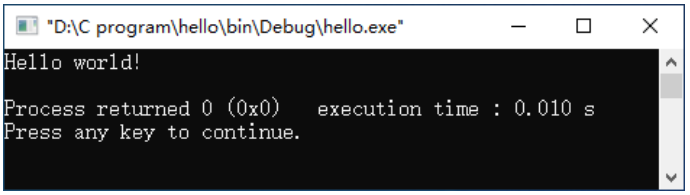


图 1-7　项目 hello 的运行结果

程序调试完成后，在项目“hello”上右击，在弹出的快捷菜单中选择 Close project 命令，关闭该项目，重新新建下一个项目，步骤如前所示。

如果要查看变量或数组等内存空间的数据在程序运行时的变化情况，程序执行的细节、流程，或者程序如果没有编译错误，但运行时有逻辑错误等情况，可以使用 Code::Blocks 中的调试功能去查看和调试程序。

Code::Blocks 调试功能的使用

1.4 C 语言程序设计学习方法

1.4.1 为什么要学习 C 语言程序设计

学习 C 语言程序设计不仅是为了让学生掌握一种编程技能，更是为了培养计算思维。这种思维方式对于激发创造力和提升问题解决能力至关重要。

美国卡内基梅隆大学计算机科学系前系主任周以真教授在 2006 年发表了一篇著名的文章 *Computational Thinking*。文中谈到“计算机科学的教授应当为大学生开一门称为‘怎么像计算机科学家一样思维’的课程，面向非专业的，而不仅仅是计算机科学专业的学生”，这是因为“机器学习已经改变了统计学……计算生物学正在改变着生物学家的思考方式。类似地，计算博弈理论正改变着经济学家的思考方式，纳米计算正改变着化学家的思考方式，量子计算正改变着物理学家的思考方式”，所以“计算思维代表着一种普遍的认识和一类普适的技能，每一个人，不仅仅是计算机科学家，都应热心于它的学习和运用”。

C 语言程序设计在培养学生计算思维方面具有重要作用，主要体现在以下几个方面。

1. 抽象与建模

John Sonmez 曾说过：“编程是一种将复杂问题转化为可管理解决方案的艺术。”这句话深刻揭示了编程的核心价值。

程序设计要求设计者将现实世界的问题抽象为计算机程序，这个过程涉及识别问题的关键要素。抽象是从具体事物中提取共性、忽略非本质细节的过程。这种抽象思维的培养有助于学生在面对复杂问题时，抓住问题的本质，忽略无关细节，从而更有效地解决问题。

将抽象出来的概念转化为具体的模型，这些模型应能够准确反映问题的关键特征和关系，这就是建模的过程。通过这一过程，可以帮助学生更好地理解问题，设计出更加有效的解决方案，从而提升系统设计能力，培养综合素质。这些技能对于学生未来的职业发展至关重要。

在程序设计中，抽象和建模往往是相辅相成的。抽象为学生提供了简化问题和理解问题本质的工具，而建模则帮助学生将这些抽象概念转化为具体可行的解决方案。

2. 逻辑推理

Steve Jobs 曾说：“编程教会你如何思考，就像数学教会你如何推理。”逻辑推理是编程过程中不可或缺的思维活动。在 C 语言编程中，学生需要精确地定义变量、编写条件语句和进行循环控制，这些都要求高度的逻辑严谨性。例如，一个错误的逻辑判断可能导致程序无法正确执行，甚至引发崩溃。因此，C 语言的学习过程促使学生不断思考和验证自己的逻辑，从而培养出严密的逻辑推理能力。

3. 问题分解

Unix 哲学认为：“复杂的问题往往可以通过将它们分解成更小的问题来解决。”这一理念在 C 语言编程中得到了充分体现。C 语言结构化程序设计思维采用了两种方法：①自顶向下，逐步求解。这种方法要求程序员首先从整体上理解问题，然后逐步细化问题的各个部分。最初，创建一个或多个高级模块来表示程序的主要功能，然后逐步将这些模块分解成更小的、更容易管理的子模块。这种设计方法有助于程序员保持对整个程序结构的清晰认识。②模块化。将程

序分解成多个模块或函数，每个模块或函数负责完成特定的任务。模块化不仅有助于代码的复用，还能提高维护和调试的便利性。

4. 算法思维

Donald Knuth 曾说过：“算法是解决问题的步骤。”算法思维是编程的核心。在 C 语言编程中，学生需要学习各种算法，理解它们的原理、优缺点及适用场景。同时，学生还需要学会根据问题的具体需求设计合适的算法。这个过程培养了学生的算法思维能力，使他们能够在面对不同的问题时快速找到有效的解决方案。算法思维不仅对于编程至关重要，还广泛应用于数学、物理、经济等多个领域，是学生综合素质的重要组成部分。

1.4.2　如何学习 C 语言程序设计来培养计算思维

相较于其他的程序设计语言而言，C 语言为程序员提供了更大的发挥空间和更高的运行效率，同时也给予了他们无限的自由。然而，这些优点也正是初学者对它又爱又恨的原因。那么，如何才能真正学好 C 语言呢？

1. 理解原理：“不要只是盲目地写代码，而要去理解其背后的原理。”

—— 林纳斯·托瓦兹(Linux 创始人)

从基本的语法和数据类型开始，逐步深入学习 C 语言，确保理解每一个概念。

2. 动手实践：“学习编程的最好方法是编写代码。”

—— 克里斯·皮恩(软件开发者和作家)

- 编写简单程序：从打印“Hello, World!”开始，逐渐编写计算、条件判断、循环等基本程序。
- 解决实际问题：尝试用 C 语言解决日常生活中的问题，如计算器的实现、排序算法的应用等。
- 参与项目：加入开源项目或自己发起小项目，通过实际的项目开发提升编程技能。

3. 阅读优秀代码：“阅读优质代码是提高编程技能的最佳方法之一。”

—— 史蒂夫·迈克康奈尔(软件开发者)

- 选择高质量的代码库：寻找开源项目中评价高、维护良好的 C 语言代码库进行阅读。
- 分析代码结构：学习优秀代码的组织结构、命名规范和注释风格等。
- 理解算法与数据结构：通过阅读代码学习不同的算法和数据结构，理解它们在解决实际问题中的应用。

4. 持续学习：“学习永远不会停止。”

—— 雷·库兹韦尔(未来学家)

订阅相关的博客、论坛和书籍，持续更新自己的知识库。

5. 反思与调试：“优秀的程序员花费大量的时间来思考，而不是编码。”

—— 约翰·卡马克(游戏开发者)

- 代码审查：定期审查自己的代码，以找出潜在的错误和可以改进的地方。
- 调试技能：掌握调试工具(如GDB)的使用方法，学会通过设置断点和单步执行等方式定位并解决问题。
- 单元测试：编写单元测试来验证代码的正确性，确保在修改代码时不会引入新的错误。

6. 团队合作：“最好的代码是合作的结果。”

—— 布莱恩·克尼汉(计算机科学家，《C 程序设计语言》作者)

- 参与开源项目：加入开源社区，参与 C 语言相关的项目，学习如何与他人协作开发。
- 代码分享与反馈：将自己的代码分享给同事或朋友，接受他们的反馈和建议，不断改进自己的编程风格和技术水平。
- 沟通技巧：学习如何在团队中有效沟通，包括代码审查和会议讨论等场景下的沟通技巧。

通过以上方法，学生可以在学习 C 语言的过程中培养出强大的计算思维能力。这不仅对他们在计算机科学领域的学术和职业生涯有益，还能帮助他们在其他领域更好地分析和解决问题。

1.5 案例：程序的铭牌


进阶提高

每个程序都应该包含识别该程序的信息，如程序名、功能、编写目的、编写日期、作者等文档说明(铭牌)，以帮助阅读者了解该程序的大致信息。一般将这类信息放在程序开头并以程序注释的形式出现。例如，对下面程序加入该程序的铭牌。

```
/*****************************************
*    Name:example1-1.c                   *
*    Purpose:print the message           *
*    Author:WangJuanqin                  *
*    Date:2024.10.15                     *
*****************************************/
#include <stdio.h>
int main(void)
{
   printf("博学之，审问之，慎思之，明辨之，笃行之。\n");
   return 0;
}
```

以上程序的运行结果将输出《礼记·中庸》中的一句话：“博学之，审问之，慎思之，明辨之，笃行之。”这句话强调了学习过程的重要性，而计算思维正是这一过程的体现。学习计算思维能够帮助我们更好地解决问题，适应快速发展的信息时代。

读者可以尝试为自己编写的程序设计一个独特的铭牌模板，通过程序展现自己的智慧和设计才能。

本章小结

本章介绍了 C 语言程序的基本组成、开发流程、C 语言程序集成开发环境和 C 语言程序设计的学习方法。本章教学涉及的有关知识的结构导图如图 1-8 所示。

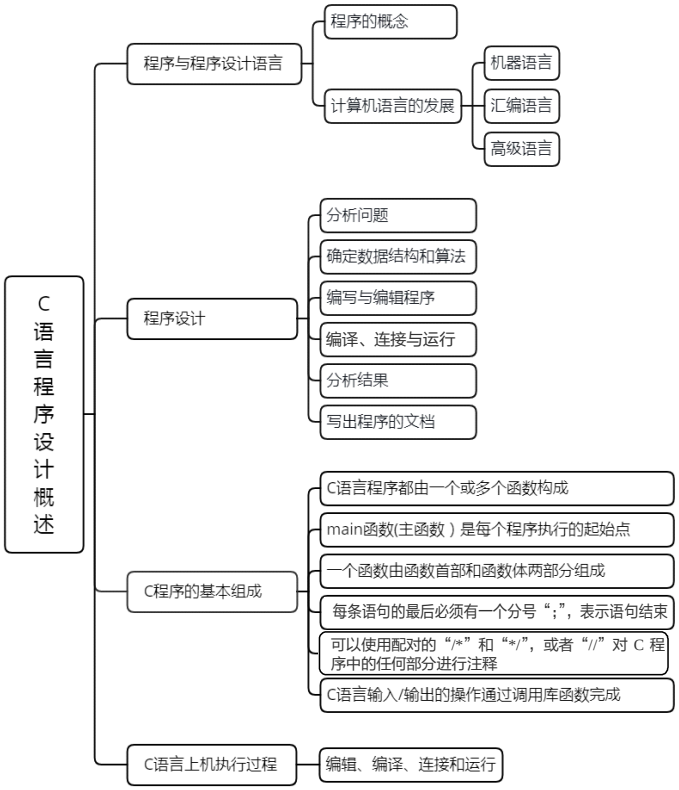


图 1-8　C 语言程序概述知识导图

习　　题

一、单选题

1. C 语言源程序由(　　)组成。

A. 函数　　B. main 函数　　C. 子程序　　D. 过程

2. 以下关于 C 语言源程序执行的叙述中，正确的是(　　)。

A. C 语言源程序的执行总是从第一个函数开始，在最后一个函数中结束

B. C 语言源程序的执行总是从 main 函数开始，在 main 函数中结束

C. C 语言源程序的执行总是从 main 函数开始，在最后一个函数中结束

D. C 语言源程序的执行总是从第一个函数开始，在 main 函数中结束

3. 以下说法中，正确的是(　　)。

A. 一个 C 函数中只允许一对花括号

B. 在 C 语言程序中，要调用的函数必须在 main 函数中定义

C. C 语言不提供输入/输出语句

D. C 语言程序中的 main 函数必须放在程序的开始部分

4. C 语言集成开发系统提供了 C 程序的编辑、编译、连接和运行环境，以下可以不在该环境下进行的是(　　)。

A. 编辑和编译　　B. 编译和连接　　C. 连接和运行　　D. 编辑和运行

5. 下面描述中，不正确的是(　　)。

A. C 程序的函数体由一系列语句和注释组成

B. 注释内容不能单独写在一行上

C. C 程序的函数说明部分包括函数名、函数类型、形式参数等的定义和说明

D. scanf 和 printf 是标准库函数而不是输入和输出语句

6. 以下描述中，正确的是(　　)。

A. 主函数中的花括号必须有，而子函数中的花括号是可有可无的

B. 一个 C 程序行只能写一条语句

C. 主函数是程序启动时唯一的入口

D. 函数体包含了函数说明部分

二、填空题

1. 为解决某一实际问题而设计的语句序列就是_____。

2. 在程序设计中，把解决问题的方法和有限的步骤称为_____。

3. 一个完整的 C 程序至少要有一个而且只能有一个_____函数。

4. 高级语言源程序代码必须经过翻译才能够运行，C 语言采用的翻译方式是_____方式。

5. 上机调试运行一个 C 程序要经过编辑、_____、连接和运行 4 个步骤，才能得到运行结果。

6. 目标程序文件的扩展名是_____。

7. 程序连接过程是将目标程序、_____或其他目标程序、各种资源等的二进制程序连接装配成可执行文件。

8. 因为源程序是_____类型的文件，所以它可以用具有文本编辑功能的任何编辑程序完成编辑。

三、编程题

1. 在屏幕上输出：道虽迩，不行不至；
事虽小，不为不成；
梦虽遥，追则能达；
愿虽艰，持则可圆。

2. 编写程序，从键盘输入两个整数，分别求这两个数字的和、差、积和商，并输出结果。

3. 编写程序，从键盘输入梯形的上底、下底和高，求该梯形的面积并输出结果。

第 2 章

C 语言基础

📖 **本章内容提示：**

计算机程序设计涉及两个基本问题：一个是对数据的描述，另一个是对数据操作的描述。计算机程序的主要任务就是对数据进行处理，编写程序也就是描述对数据的处理过程。本章将主要介绍 C 语言中与数据描述有关的问题，包括基本数据类型、常量和变量等数据对象，以及 C 语言中有关数据运算的基本概念和运算规则。

针对各种类型的数据，本章将详细介绍算术运算(包括自加和自减运算)、关系运算、逻辑运算、赋值运算的运算符、表达式、运算优先级及结合性。同时，我们将讨论运算和赋值过程中的类型转换问题，并介绍相关的各种运算符。此外，还将介绍 C 语言的输入输出函数。

📖 **教学基本要求：**

掌握 C 语言的基本数据类型、各种数据类型的存储特点及取值范围；熟练掌握 C 语言中常量的表示、变量的定义和使用，掌握各种运算符的应用、表达式的书写，掌握利用格式输入函数为基本数据类型变量赋值的方法和利用格式输出函数正确输出数值的方法。

2.1 C 语言的字符集

字符是组成 C 语言的最基本元素。C语言字符集由字母、数字、空格、下画线、标点等 ASCII 码表中的可视字符组成(在字符串常量、源程序的注释中还可以使用汉字及其他非 ASCII 码表中的字符)。

在编写 C 语言程序时，只能使用 C 语言字符集中的字符，对字母区分大小写。如果使用其他非 ASCII 码表中的字符，编译器将把它们视为非法字符而报错(字符串常量、源程序中的注释除外)。

2.2 标识符

C 语言的标识符由字母、数字和下画线组成，其中第一个字符必须是字母或下画线。C 语言中的主要标识符是保留字和用户自定义标识符。

2.2.1 保留字

保留字也称关键字，是 C 语言规定的具有特定意义并有专门用途的标识符，用户只能按其预先规定的意义来使用它，不能改变其意义。C 语言可以使用以下 32 个关键字。

auto	register	static	extern	void	char	short	int
long	signed	unsigned	float	double	struct	union	enum
sizeof	typedef	const	if	else	switch	case	default
do	for	while	goto	break	continue	return	volatile

2.2.2 用户自定义标识符

用户自定义标识符是用来标识变量名、符号常量名、函数名、数组名、类型名等的有效字符序列。例如：

- 合法的用户自定义标识符：sum、f1、average、_total、Class、day、stu_name、lotus_1_2_3 等。
- 不合法的标识符：M.D.John、$123、#33、3days、a>b、if 等。

警告：

C 语言中的大小写字母是两个不同的字符。例如，sum 不同于 Sum，BOOK 不同于 book。

用户自定义标识符不能与保留字同名，也不能与系统预先定义的标识符(如 main、printf 等)同名。

用户自定义标识符的长度要在一定范围内。各编译系统都有自己的规定和限制，一般环境允许取 32 个字符。

用户自定义标识符的命名应当做到见名知义。例如，sum 代表和；aver 代表平均值。

2.3 数据与 C 语言的数据类型


数据与数据类型

数据是对客观事物的符号表示，是所有能被输入到计算机中，且能被计算机处理的符号(数字、字符等)的集合，是计算机操作对象的总称。

数据是程序处理的对象，数据在处理时都需要存放在内存中，不同类型的数据在内存中的存放形式不同，对其施加的操作也有区别。例如，整数和实数在内存中的存放形式不同，整数和实数都可以参与算术运算，但整数可以进行求余运算，而实数不能。

因此，在程序中对各种数据进行处理之前，都要对其类型预先进行说明，一是便于为数据分配相应的存储空间，二是说明了程序处理数据时应采用何种运算方法(操作)。C 语言的数据类型如图 2-1 所示。

数据以变量或常量的形式来描述，每个变量或常量都有数据类型。变量是存储数据的单元，对应某个内存空间，为了便于描述，计算机高级语言中都用变量名来表示其内存空间。所以，程序能在变量中存储值和取出值。

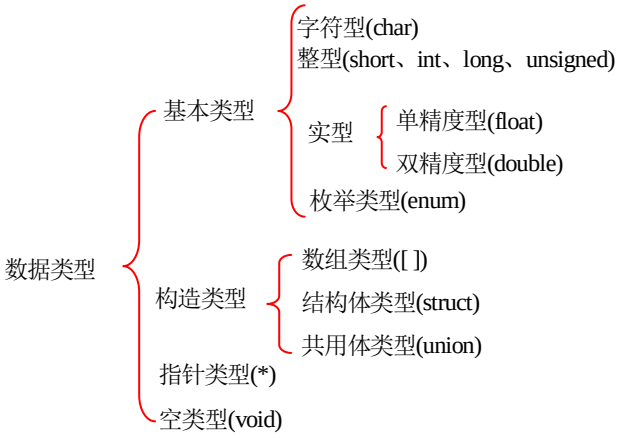


图 2-1　C 语言的数据类型

在定义变量时，说明变量名和变量的类型(如int、float)就是告诉编译器要为变量分配多少字节的空间，以及变量中要存储什么类型的数据。数据类型确定了其内存所占空间的大小，从而确定其存储数据的范围。

C 语言为每个类型定义了一个标识符，通常把它们称为类型名。例如，整数型用 int 标识，字符型用 char 标识。一个类型名由一个或几个关键字组成。

C 语言的数据类型比其他一些程序语言要丰富，它有指针类型，还有构造其他多种数据类型的能力。例如，除了数组类型之外，C 语言还可以构造结构体类型、共用体类型。基本类型结构比较简单，构造类型一般是由其他的数据类型按照一定的规则构造而成，结构比较复杂。指针类型是 C 语言中使用较为灵活、颇具特色的一种数据类型。

本章主要介绍基本数据类型中的整型、实型和字符型三大类型，其他各种数据类型将在后续章节中详细介绍。

2.3.1　整型数据类型

1. 整型数据的编码

整型数据的编码有 3 种形式，即原码、反码和补码，整型数据在计算机中存储的是其补码形式。

1) 原码

原码是指将一个数值的绝对值转换为二进制数，在补齐或截取相应字节位后，将最高位用来表示符号，正数为 0、负数为 1 形成的二进制编码。例如：

10 的双字节原码：　0000 0000 0000 1010

−10 的双字节原码：　1000 0000 0000 1010

2) 反码

正数的反码与原码相同，负数的反码是将原码除符号位外各位逐一取反后形成的二进制编码。例如：

−10 的双字节反码：　1111 1111 1111 0101

3) 补码

正数的补码与原码相同，负数的补码是该数的反码+1 后形成的二进制编码。例如：

−10 的双字节补码： 1111 1111 1111 0110

同样，如果−10 用 4 字节表示，则原码、反码、补码表示形式如下：

−10 的四字节原码： 1000 0000 0000 0000 0000 0000 0000 1010

−10 的四字节反码： 1111 1111 1111 1111 1111 1111 1111 0101

−10 的四字节补码： 1111 1111 1111 1111 1111 1111 1111 0110

警告：

整型数据在计算机中的存储形式是补码。按存储空间的类型，所占字节数的大小不同。

2. 整型数据的表示

在 C 语言中，整型类型以关键字 int 作为基本类型说明符，另外配合 4 个类型修饰符 long、short、signed、unsigned 来改变和扩充基本类型的含义，以适应更灵活的应用。这些修饰符与 int 可以组合成表 2-1 所示的不同整型。其中，方括号中的内容可以省略不写。

表 2-1 整型数据类型

数据类型	类型标识符	所占字节数	所表示数的范围
有符号短整型	[signed] short [int]	2	-2^{15}～$2^{15}-1$(−32768～32767)
有符号整型	[signed] int	4	-2^{31}～$2^{31}-1$(−2147483648～2147483647)
有符号长整型	[signed] long [int]	4	-2^{31}～$2^{31}-1$(−2147483648～2147483647)
无符号短整型	unsigned short [int]	2	0~$2^{16}-1$(0～65535)
无符号整型	unsigned int	4	0~$2^{32}-1$(0~4294967295)
无符号长整型	unsigned long [int]	4	0~$2^{32}-1$(0～4294967295)

C 语言没有规定各种整型类型的表示范围，即在内存中所占的字节数。对于 int 和 long，只规定了 long 类型的表示范围不能小于 int 类型，但也允许它们的表示范围相同。C 语言系统根据各个计算机系统自身的性能，对整型的各类型规定了明确的表示方式和表示范围。

警告：

在 Code::Blocks 中，编译器将整型数据和长整型数据都用 4 字节表示(本书后面所说的整型均占用 4 字节)。

3. 整型数据的溢出

当进行整型数据计算时，若计算结果超出了该类型数据表示的范围，这种情况称为数据溢出。

【例 2-1】编写一个求两数和的 C 程序并在计算机上运行。

```
#include<stdio.h>
int main(void)                  /*  求两数和的主函数  */
{   short a,b;                  /*  定义 a、b 为短整型变量  */
    a=32767;                    /*  为变量 a 赋值 32767 */
```

```
    b=a+2;                  /* 将变量 a 的值加 2 后赋给变量 b */
    printf("b=%d\n",b);     /* 输出变量 b 的值 */
    return 0;
}
```

预测结果：32769

验证结果：−32767

为什么会出现这种情况呢?

变量 a 中值的二进制数为：0111 1111 1111 1111

变量 b 中值的二进制数为：1000 0000 0000 0001

对于变量 b 的值，首位为 1，表示 b 为负数，而计算机中存储的都是数据的补码，要知道负数的原码，就必须对该补码求补。

变量 b 中值的补码为：1000 0000 0000 0001

对补码求反码，得到：1111 1111 1111 1110

对反码+1，得原码为：1111 1111 1111 1111

将这个数转换为十进制，正好是−32767。

对于这种问题，系统往往不给出错误提示，而要靠程序设计者正确使用内存空间来保证其正确性，所以数据类型的使用要仔细，对运算结果的数量级要有基本估计。

2.3.2　实型数据类型

实型数据类型用于表示带小数点的数据，根据表示范围及精度要求的不同，分为单精度类型和双精度类型，实型数据类型表示数据的情况如表 2-2 所示。

表 2-2　实型数据类型

数据类型	类型标识符	存储字节数	取值范围	有效位
单精度类型	float	4	-3.4×10^{38}～3.4×10^{38}	7
双精度类型	double	8	-1.7×10^{308}～1.7×10^{308}	16

实数在计算机中是以指数形式存储的，对于任何形式的实数，均转换成指数形式。例如，将 345.68 转换成 0.34568e3，将 12.45e3 转换成 0.1245e5 的形式存储。

以在计算机中存储 float 类型数据为例，在内存中占据 4 字节，即 32 位，这 32 位分别存放该数据的符号(即数据的正负符号)、规范化的尾数(小数部分)、阶符(指数的正负符号)和阶码(指数)。例如，0.123456e−2 的存放形式可用图 2-2 示意。实际上计算机中存放的是二进制数，这里仅用十进制数说明其存放形式。至于尾数和阶码各占多少二进制位，标准 C 并无具体规定，由各编译系统自行确定。

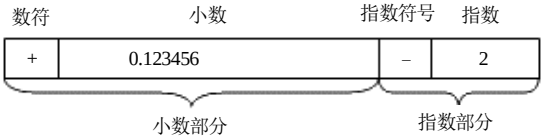


图 2-2　实型数据在计算机中的存放

对于实型数据来说，在计算和存储过程中可能会存在一定的误差，如例 2-2 所示。

【例 2-2】输出实型数据 a、b 的值。

```
#include<stdio.h>
int main(void)
{  float a;                          /* 定义变量 a 为单精度类型 */
   double b;                         /* 定义变量 b 为双精度类型 */
   a=-12345.633;
   b=-0.12345678912345673 99e15;
   printf("a=%f,b=%f\n",a,b);        /* 输出变量 a、b 的值 */
   return 0;
}
```

程序中为单精度变量 a 和双精度变量 b 分别赋值，不经过任何运算就直接输出变量 a、b 的值。理想结果应该是照原样输出，即

a=−12345.633，b=−0.123456789123456 73 99e15

但运行该程序后，实际输出结果是：

a=−12345.632813，b=−123456789123456.730000

因为程序中变量 a 为单精度类型，能存储 7 位有效位，所以输出的前 7 位是准确的；变量 b 为双精度类型，可以存储 16 位有效位，所以输出的前 16 位是准确的。由此可见，由于受计算机存储的限制，使用实型数据会产生一些误差。需要注意实型数据的有效位，合理使用不同的类型，尽可能减少误差。

2.3.3 字符型数据类型

字符型数据即通常的字符，包括计算机所用编码字符集(ASCII 码)中的所有字符。字符型数据在内存中存储的是它们的ASCII 码值，在计算机中占 1 字节的存储空间。例如，字符'A'在计算机中存储的是其 ASCII 码值 65。

除占用的存储空间不同以外，C 语言把字符型数据当成整型数据来处理，所以字符型数据分有符号和无符号两种类型，与整型数据类型通用。表 2-3 为字符型数据类型的相关内容。

表 2-3 字符型数据类型

数据类型	类型标识符	所占字节数	所表示数的范围
有符号字符型	char	1	$-2^7 \sim 2^7-1$(−128～127)
无符号字符型	unsigned char	1	$0 \sim 2^8-1$(0~255)

ASCII 码的取值范围是 0~127，其中，字符存储时可以用 char 类型表示，也可以用 unsigned char 类型表示。扩展的 ASCII 码的取值范围是 0～255，编码在 128～255 范围内的字符只能用 unsigned char 类型存储。

2.4 常量

常量是指在程序运行过程中，其值不能被改变的量。常量是 C 语言中的基本数据对象之一，包括字符常量、整型常量、长整型常量、单精度常量、双精度常量、字符串常量等。


常量

2.4.1　整型常量

C 语言中的整型常量可用以下 3 种形式表示。

1. 十进制整型常量

十进制整型常量与数学上的表示形式相同。

例如：123、−123、8、0、−5、30000 等。

2. 八进制整型常量

八进制整型常量是以数字 0 开头的八进制数字序列，数字序列中只能含有 0～7 这 8 个数字。

例如：056 表示八进制数 56，等于十进制数 46。

3. 十六进制整型常量

十六进制整型常量是以数字 0x 或 0X 开头的十六进制数字序列。数字序列中只能含有 0～9 这 10 个数字和 a、b、c、d、e、f(或 A、B、C、D、E、F)这 6 个字母。

例如：0x123 表示十六进制数 123，等于十进制数 291；0x3A 表示十六进制数 3A，等于十进制数 58。

整型常量的区分：一般数值默认为 int 型，如 123 为 int 型。如果数字后面带有符号，可根据符号区分不同类型，如长整型常量后跟字母 L(或 l)，表示该数为长整型数字，如 123L，表示 123 为长整型。无符号整型常量后跟字母 U(或 u)，表示没有符号的数，如 234U，表示无符号数字 234。

2.4.2　实型常量

在 C 语言中，实型常量一般都作为双精度类型来处理，并且只用十进制数表示。实型常量有两种书写格式：小数形式和指数形式。

(1) 小数形式：由符号、整数部分、小数点及小数部分组成。

例如：12.34、0.123、.123、123.、−12.0、−0.0345、0.0，它们都是合法的小数形式的实型常量。

警告：

其中的小数点是不可缺少的。例如 123.不能写成 123，因为 123 是整型常量，而 123.是实型常量。

(2) 指数形式：由十进制小数、e、指数(或十进制小数、E、指数)组成。

格式中的 e 或 E 前面的数字表示尾数，e 或 E 表示底数 10，而 e 或 E 后面的指数必须是整数，表示 10 的幂次方。例如，12.34e3 表示 12.34×10^3。以下都是合法的指数形式的实型常量：2.5e3、−12.5e−5、0.123E−5、−267.89E−6、0.61256e3。

警告：

指数必须是不超过指数数据表示范围的整数，且在 e 或 E 前必须有数字。例如：e3、3.0e、E−9、10e3.5、.e8、e 都不是合法的指数形式。

注意：

实型常量的区分：对于上述两种书写形式的实型常量，系统均默认为是双精度实型常量。如果要明确表示单精度实型常量，可在上述书写形式的末尾分别加上后缀 f(或 F)即可。

例如：2.3f、-0.123F、2e-3f、-1.5e4F 为合法的单精度实型常量，只能有 7 位有效数字。

对于超过有效数字位的数位，系统在存储时会自动舍去。

2.4.3 字符常量

字符及字符串常量

字符常量是指用一对单引号括起来的一个字符。例如，'x'、'B'、'b'、'$'、'?'、' '(表示空格字符)、'3'都是字符常量，注意其中'B'和'b'是不同的字符常量。

除了以上形式的字符常量外，C 语言还提供了一种特殊的字符常量，即用“\”开头的字符序列，为转义字符。如\n，代表一个“换行”符，这是一种控制字符，在程序中无法用一个一般形式的字符表示，只能采用特殊形式来表示。常用的转义字符序列及其功能如表 2-4 所示。

表 2-4 转义字符序列及其功能

转义字符	功能	转义字符	功能
\n	换行	\t	水平跳格
\r	回车	\b	退格
\\	反斜线字符	\f	走纸换页
\'	单引号字符	\"	双引号字符
\ddd	1~3 位八进制数表示的字符	\xdd	1 或 2 位十六进制数表示的字符

转义字符是一种特殊形式的字符常量，意思是对“\”后面字符原来的含义进行转换，变成某种特殊约定的含义。

用转义字符可以表示任何可显示或不可显示的字符。在实际应用中，转义字符的应用很常见，例如：

```
printf("a=%f\tb=%f\n",a,b);
```

其中，转义字符\t 指在下一个输出区(一个输出区占 8 列)输出其后的内容。\n 指输出一个回车换行，几乎每个程序中都会有一个或若干个这样的转义字符，要注意其使用形式。

2.4.4 字符串常量

字符串常量是指用一对双引号括起来的字符序列。这里的双引号仅起到字符串常量边界符的作用，它并不是字符串常量的一部分。例如：

```
"How are you.", "China", " ", "a"
```

警告：

不要把字符串常量和字符常量混淆，"a"和'a'是不同的数据，前者是字符串常量，后者是字符常量。C 语言规定，在每一个字符串的末尾自动加上一个转义字符\0(ASCII 码值是 0，对应

的字符为空)，作为字符串常量的结束标志。对字符串操作时，这个结束标志是非常重要的。

'a'和"a"在内存中的存放形式如图 2-3 所示，字符常量在内存中占 1 字节，而字符串常量除了每个字符各占 1 字节外，其字符串结束符\0 也要占 1 字节。

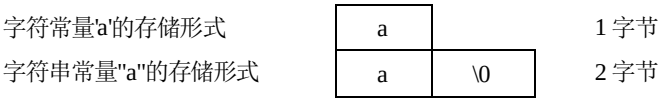


图 2-3　字符常量和字符串常量的存储示意图

不能将字符串常量存储在字符变量中。C 语言没有专门的字符串变量，如果需要存储和处理字符串，一般用字符数组来实现。关于字符数组的内容，在本书第 4 章有详细介绍。

如果字符串常量中出现双引号，则要用反斜线(\)将其转义(例如使用\")，取消原有边界符的功能，使其仅作为双引号字符。例如，要输出字符串：

```
He says:"How do you do."
```

应写成如下形式：

```
printf("He says:\"How do you do.\"");
```

2.4.5　符号常量

在 C 程序中，还可以用一个标识符来表示一个常量。也就是说，用指定的标识符表示某个常量，在程序中需要使用该常量时就可直接引用标识符。

C 语言中用宏定义命令对符号常量进行定义，一般形式如下：

```
#define 标识符 常量
```

其中，#define 是宏定义命令的专用定义符，标识符是对常量的命名，常量可以是前面介绍的几种类型常量中的任何一种。这个被指定的标识符就称为符号常量。关于宏定义命令，本书第 10 章有详细介绍。

【例 2-3】已知圆的半径 *r*，求圆的周长 *c* 和圆的面积 *s*。

```
#define PI 3.1416
int main(void)
{ float r,c,s;
  scanf("%f",&r);
  c=2*PI*r;                    /* 编译时用 3.1416 替换 PI */
  s=PI*r*r;
  printf("c=%f,s=%f\n",c,s);
  return 0;
}
```

在编译预处理时，预处理程序会将程序中宏定义命令之后出现的所有符号常量用宏定义命令中对应的常量一一替代。例如，对于以上宏定义命令，在编译程序前，预处理程序会将程序中的所有 PI 替换为 3.1416。

习惯上人们把符号常量名用大写字母表示，而把变量名用小写字母表示。

使用符号常量的好处如下。

(1) 含义清楚，见名知意。例如上面的程序，看 PI 比看 3.1416 更容易知道它代表π值。因此定义符号常量时应考虑“见名知意”。

(2) 在需要改变一个常量时能做到“一改全改”。例如，要将π值改为 3.1415926 时，如用常量，要修改两处，若多次使用π值，就要修改多次；而用符号常量，只需修改宏定义处常量的值一次即可。编译前，程序中所有的 PI 都会被替换为 3.1415926。

2.5 变量

变量是指在程序运行时其值可以改变的量。这里所说的变量与数学中的变量是完全不同的概念。在 C 语言以及其他各种高级程序设计语言中，变量是指数据在内存中的存储单元。


变量

程序中的一个变量可以被看作一个存储数据的单元，它的功能是存储数据。对变量的基本操作有以下两种。

(1) 向变量中存入数据值，这个操作被称作给变量“赋值”。

(2) 取得变量当前值，以便在程序运行过程中使用，这个操作称为“取值”。

要对变量进行“赋值”和“取值”操作，程序中的每个变量都要有一个变量名，程序是通过变量名来使用变量的。在 C 语言中，变量名作为变量的标识，其命名规则符合标识符的所有规定。

C 语言规定，程序中使用的每个变量都必须“先定义，后使用”。也就是说，首先需要定义一个变量，然后才能使用它。这样使用变量的优点如下。

(1) 只有定义过的变量才可以在程序中使用，这使得变量名的拼写错误容易被发现。

(2) 定义的变量属于确定的类型，编译系统可方便地检查变量所进行运算的合法性。

(3) 在编译时根据变量类型可以为变量分配相应字节的存储空间。

1. 变量的定义

在 C 语言中，变量定义的一般形式如下：

```
类型说明符 变量名表;
```

其中，类型说明符是 C 语言的一种有效的数据类型，如整型类型说明符 int、字符型类型说明符 char 等。变量名表的形式是：变量名 1，变量名 2，…，变量名 n。即用逗号分隔的变量名的集合，最后用分号结束定义。例如：

```
int a, b, c;        /* 定义 a、b、c 为整型变量，用来存储整型数据 */
char ch;            /* 定义 ch 为字符变量，用来存储字符型数据 */
double d, e;        /* 定义 d、e 为双精度实型变量，用来存储双精度型数据 */
```

C 语言语法格式较自由，把同类型多个变量的定义写在同一行是允许的。在 C 程序中，除了不能用关键字作为变量名，可以用任何合法的标识符作为变量名。但是，一般提倡用能说明变量用途的、有意义的名称作为变量名，因为这样的名称对程序的阅读者有一定提示作用，有助于提高程序的可读性。

2. 为变量赋初值

变量初始化是指在定义变量的同时对变量赋初值。例如：

```
int x=10,y=20;     /* 变量初始化 */
```

为变量赋初值有两种形式：一种是对变量初始化，另一种是先定义、后赋值。例如：

```
int x,y;
x=10,y=20;         /* 变量赋值 */
```

基本类型的变量都可以初始化，例如：

```
float x=123.45;    /* 定义 x 为实型变量，且赋初值 123.45  */
int a,b,c=10;      /* 给部分变量赋初值，即仅给 c 赋初值 10  */
double pai=3.14;   /* 定义 pai 为双精度实型变量，且赋初值 3.14  */
char ch='a';       /* 定义字符变量 ch，并赋初值'a'  */
```

但是，变量初始化不是在程序编译时完成的(本书后面将介绍的外部变量和静态变量除外)，而是在程序运行时为变量赋初值。

注意：

如果定义了变量但没有赋值，那么变量中的数据是一个随机数，这一点在程序设计中一定要注意。

2.6 运算符

运算是对数据进行加工的过程，运算符是描述加工类型的符号。C 语言除了提供一般高级语言的算术运算符、关系运算符、逻辑运算符，还提供赋值运算符、位操作运算符、自增/自减运算符等。C 语言中的运算符有以下几类。

- 算术运算符：+、−、*、/、%、++、--。
- 关系运算符：<、<=、>、>=、==、!=。
- 逻辑运算符：&&、||、!。
- 赋值运算符：=、+=、−=、*=、/=、%=、<、<=、>、>=、|=、&=、^=。
- 位运算符：|、^、&、<<、>>、~。
- 条件运算符：?:。
- 逗号运算符：,。
- 其他：*、&、(type)、()、[]、.、->、sizeof。

学习运算符时应注意运算符功能、运算量个数、运算量类型、运算符优先级别、结合方向、结果的类型等。

C 语言的运算符按其在表达式中与运算对象的关系(连接运算对象的个数)，可以分为以下几种。

- 单目运算：一个运算符连接一个运算对象。
- 双目运算：一个运算符连接两个运算对象。
- 三目运算：一个运算符连接三个运算对象。

本节将介绍算术运算符、关系运算符、逻辑运算符和赋值运算符，在后续各章中将介绍其他相关运算符。有关 C 语言的运算符及其结合性详见本书附录 B。

2.6.1 算术运算

1. 基本的算术运算符


算术运算（1）

C 语言允许的算术运算符如下。

+：加法运算符或取正值运算符，如 a+b、+5。

−：减法运算符或取负值运算符，如 a−b、−5。

*：乘法运算符，如 a*b、4*3。

/ ：除法运算符，如 a/b、5/2。

%：模运算或求余运算符，如 5%7。

其中，需要说明以下几点。

(1) “+”和“−”运算符既具有单目运算功能(即取正值运算和取负值运算)，又具有双目运算功能。作为单目运算符使用时，其优先级高于双目运算符。

(2) 在使用除法运算符“/”时要特别注意数据类型。因为两个整数(或字符)相除，结果是整型；如果不能整除，只取结果的整数部分，小数部分全部舍去。例如:

```
5/2=2
```

若相除的两个数中有一个为实数，所得的商也为实数。例如:

```
5.0/2=2.5
```

(3) 模运算“%”也称求余运算符。该运算符要求两个运算对象都为整型，结果是两数相除所得的余数。一般情况下，余数的符号与被除数的符号相同。例如:

```
5%10=5；−8%5=−3；8%−5=3
```

(4) 如果参与+、−、*、/运算的两个数中有一个为实型数据，则结果与该实型数据同类型。

2. 算术表达式

算术表达式是指用算术运算符将运算对象(也称操作数)连接起来的、符合 C 语法规则、对运算对象进行算术运算的式子。运算对象可以是常量、变量、函数等。

例如：

a*b/c−1.5+ 'a'

就是合法的算术表达式。

警告：

使用算术表达式时应注意以下几点。

(1) 算术表达式的乘号(*)不能省略。例如数学式 b^2-4ac，相应的 C 语言表达式应该写成 b*b−4*a*c。

(2) 运算符两侧运算对象的类型应一致，如果不一致，系统将自动按转换规则先对操作对象进行转换，然后再进行相同数据类型的运算。

(3) 算术表达式只能使用圆括号改变运算的优先顺序(不要用{}和[])。可以使用多层圆括号，

此时左右括号必须配对，运算时从内层括号开始，由内向外依次计算表达式的值。

C 语言不提供乘方运算符，对于乘方运算，可以调用标准库函数中的数学函数来实现。标准库函数中常用函数的功能及有关说明参见附录 D。

例如，将下列数学表达式

$$\frac{6\sin(x+y^2)}{a-\sqrt{x+y+1}}$$

写成符合 C 语言规则的表达式。

转换成的 C 语言表达式为 6*sin(x+y*y)/(a−sqrt (x+y+1))。

其中 sin( )和 sqrt( )都是标准库函数中的成员。

【例 2-4】家用台式计算机中硬盘的磁道和扇区如图 2-4 所示。某台计算机上的硬盘共有 9216 个磁道(即 9216 个不同半径的同心圆)，每个磁道分成 8192 个扇区(每个扇区为 1/8192 圆周)，每个扇区可记录 512 字节。电动机使磁盘以 7200 r/min(7200 转/分)的转速匀速转动。磁头在读写数据时是不动的，磁盘每转一圈，磁头沿半径方向跳动一个磁道。试计算：

(1) 一个扇区通过磁头所用的时间是多少秒？

(2) 不计磁头转移磁道的时间，计算机 1 秒内最多可以从一个盘面上读取多少字节？

分析：磁盘的转速为 7200 r/min，所以转一周所用的时间为(1/7200)min，即磁盘的转动周期为 T=(1/120) s。一个扇区的扫描时间为 $t=T/8192$ s，每个扇区的字节数为 512 字节，1 秒内读取的字节数为 1/t*512。

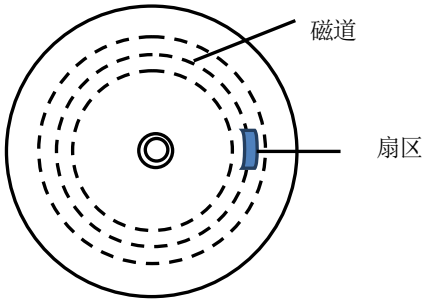


图 2-4　磁盘的磁道和扇区

```
#include<stdio.h>
int main(void)
{
   double T,t,byte;
   T=1.0/120;
   t=T/8192;
   byte=1/t*512;
   printf("每个扇区的扫描时间：%f 秒\n",t);
   printf("一秒内最多可以读取%f 字节\n",byte);
   return 0;
}
```

程序运行结果如下：

```
每个扇区的扫描时间：0.000001 秒
一秒内最多可以读取 503316480.000000 字节
```

这是一个非常简单的程序，它由 3 条赋值语句和两条函数调用语句构成。程序的执行过程是按照书写的先后顺序一步步执行的，程序中的每条语句都被执行一次，而且只能被执行一次。

3. 自加、自减运算符


算术运算（2）

运算符自增“++”、自减“--”使变量的值增 1 或减 1，其操作对象只能是变量。自增、自减运算符的应用形式如下。

- ++i、--i：运算符在变量前面，称为前置运算，表示变量在使用前自动增 1 或减 1。
- i++、i--：运算符在变量后面，称为后置运算，表示变量在使用后自动增 1 或减 1。

警告：

使用自增和自减运算符时应注意以下几点。

(1) ++、--运算符只能用于变量，不能用于表达式或常量。所以下列语句形式都是不允许的:

```
5++; (3*8)++; ++(a+b)
```

(2) ++、--运算符的前置和后置意义不同，使用时应当注意。前置时是在使用变量之前将其值增 1 或减 1；后置时是先使用变量原来的值，使用完之后再使其值增 1 或减 1。

例如:

```
j=3;   k=++j;                /* 则执行结果为 k=4,j=4 */
j=3;   k=j++;                /* 则执行结果为 k=3,j=4 */
j=3;   printf("%d",++j);     /* 则执行结果为 4,j=4 */
j=3;   printf("%d",j++);     /* 则执行结果为 3,j=4 */
a=3;b=5;c=(++a)*b;           /* 则执行结果为 c=20,a=4 */
a=3;b=5;c=(a++)*b;           /* 则执行结果为 c=15,a=4 */
```

(3) 用于++、--运算的变量可以是整型、字符型、浮点型和指针型等类型的变量。

(4) ++、--的结合性是自右向左。

例如，-i++ 等效于-(i++)。

2.6.2 关系运算


关系运算

C 语言提供了关系运算符和逻辑运算符，用来构造 C 程序中的控制条件，实现程序的选择结构和循环结构等控制。

关系运算和逻辑运算的结果都是逻辑值，即“真”和“假”。由于 C 语言中没有逻辑型数据，因此 C 语言规定用整型数据来表示运算结果的逻辑值，用整数“1”表示逻辑“真”，用整数“0”表示逻辑“假”。而在运算量需要逻辑值时，将“非 0”视为“真”，将“0”视为“假”进行运算。

1. 关系运算符

关系运算实际上是比较运算。C 语言规定的 6 种关系运算符及其有关说明如下。

>：大于，如 a>b、3>7。

>=：大于或等于，如 a>=c、5>=2。

<：小于，如 a<b、6<8。

<=：小于或等于，如 a<=b、3<=b。

!=：不等于，如 a!=b、3!=5%7。

==：等于，如 a==b、3==5*a。

关系运算符都是双目运算符，其结合性是从左向右结合。优先级分为两级。

高级(6 级)：<、<=、>、>=。

低级(7 级)：==、!=。

关系运算符的优先级低于算术运算符。

2. 关系表达式

用关系运算符将两个表达式连接起来的式子称为关系表达式。它的一般形式为：

```
表达式 1 关系运算符 表达式 2
```

例如：

```
a+b<c          /* 等价于(a+b)<c */
(a==b)<(b=10%c)
'A'!='a'       /* 对两个字符进行比较 */
```

说明：

(1) 关系表达式只有两种可能的结果：描述的关系要么成立，要么不成立。若关系成立，结果用 1 表示；若关系不成立，结果用 0 表示。因此关系表达式的运算结果一定是逻辑值。

(2) 当关系运算符用于比较两个字符时，比较的是字符的 ASCII 码值。

(3) 关系运算符中判断“等于”的符号是“==”而不是“=”。

进行关系运算时，先计算表达式的值，然后再进行关系比较运算。例如：

```
int a=3,b=2,c=1,d,f;
a>b              /* 表达式结果值为 1(真) */
(a>b)==c         /* 表达式结果值为 1(真) */
b+c<a            /* 表达式结果值为 0(假) */
d=a>b            /* 结果：d=1 */
f=a>b>c          /* 结果：f=0 */
```

2.6.3　逻辑运算

1. 逻辑运算符

逻辑运算

C 语言提供了以下 3 个逻辑运算符。

&&：逻辑与运算符，如 a&&b、a<=x && x<=b。

||：逻辑或运算符，如 a||b、a==b||x==y。

!：逻辑非运算符，如 !a、!a||a>b。

逻辑运算要求运算对象为“真”(非 0)或“假”(0)。以上 3 个逻辑运算符的运算规则如表 2-5 所示。

表 2-5　逻辑运算符的运算规则

a	b	a&&b	a||b	!a	!b
0	0	0	0	1	1
0	非 0	0	1	1	0
非 0	0	0	1	0	1
非 0	非 0	1	1	0	0

逻辑运算符的优先级是：“!”最高(2 级)，属于单目运算符，“&&”次之(11 级)，“||”最低(12 级)；逻辑运算符的优先级低于所有关系运算符，而“!”的优先级高于算术运算符。

2. 逻辑表达式

逻辑表达式是指用逻辑运算符连接表达式形成的式子。逻辑表达式的一般形式为：

```
<单目逻辑运算符>表达式
```

或

```
表达式 1 <双目逻辑运算符> 表达式 2
```

其中，表达式 1、表达式 2 又可以是逻辑表达式，从而形成嵌套形式。

例如：

```
int a=4;b=5;
!a                  /* 值为 0 */
a&&b                /* 值为 1 */
a||b                /* 值为 1 */
!a||b               /* 值为 1 */
4&&0||2             /* 值为 1 */
5>3&&2||8<4-!0      /* 值为 1 */
'c'&&'d'            /* 值为 1 */
```

短路特性：逻辑表达式求解时，并非所有的逻辑运算符都被执行，只有在必须执行下一个逻辑运算符才能求出表达式的解时，才执行该运算符。

```
a&&b&&c  /* 只在 a 为真时，才判别 b 的值；只在 a、b 都为真时，才判别 c 的值 */
a||b||c      /* 只在 a 为假时，才判别 b 的值；只在 a、b 都为假时，才判别 c 的值 */
```

综合应用运算符的以上知识，可以将下面常见的描述用相应的逻辑表达式来表示。

(1) 判断变量 x 能不能被 y 整除：x%y==0。

(2) 判断 x 能不能被 2 整除：x%2==0。

(3) 判断输入的三条边 a、b、c 能不能构成三角形：a+b>c&&b+c>a&&a+c>b。

(4) 判断数 x 是否属于某区间[a,b]：a<=x&&x<=b。

(5) 闰年的判断方法：能被 4 整除但不能被 100 整除，或者能被 400 整除。试判断 y 是不是闰年：y%4==0&&y%100!=0||y%400==0。

2.6.4 赋值运算


赋值运算

1. 赋值运算符

赋值运算符用“=”表示，其功能是将一个数据赋给一个变量。例如：

```
n=12;
```

警告：

赋值运算符“=”与数学中的等号完全不同，数学中的等号表示等号两边的值是相等的，而赋值运算符“=”是将其右边的数据存放到左边指定的内存变量中。

2. 赋值表达式

由赋值运算符将一个变量和一个表达式连接起来的式子称为赋值表达式。它的一般形式如下：

```
<变量名><赋值运算符><表达式>
```

赋值表达式的求解过程：计算赋值运算符右边“表达式”的值，并将计算结果赋值给左边的“变量”。赋值表达式的值就是赋值运算符左边“变量”的值。

例如 a=7;，变量 a 的值为 7，赋值表达式“a=7”的值是 7。

上述一般形式中的表达式也可以是一个赋值表达式，所以可以有下面的式子：

```
a=(b=7);
```

即，将 7 赋予变量 b，表达式 b=7 的值为 7，再赋予变量 a。根据赋值运算符的结合性(从右向左)，上面的表达式等价于 a=b=7。下面是赋值表达式的例子：

```
a=b=c=8;
a=6+(b=3);
a=(b=3)*(c=4);
```

3. 复合赋值运算符

在赋值运算符“=”之前加上其他运算符，可以构成复合赋值运算符，用于完成复合赋值运算操作。C 语言中的复合赋值运算符有：

```
+=、－ =、*=、/=、%=、<<=、>>=、|=、&=、^=
```

由复合赋值运算符构成的表达式的一般形式如下：

```
<变量名><复合赋值运算符><表达式>
```

该式等价于：

```
<变量名>=<变量名><运算符><表达式>
```

即先对变量和表达式进行指定的复合运算，然后将运算结果赋值给变量。

例如：

```
a*=2       /* 等价于 a=a*2 */
a/=b+5     /* 等价于 a=a/(b+5) */
```

警告：

“a*=b+5”与“a=a*b+5”是不等价的，它等价于“a=a*(b+5)”，这里的括号是必需的。

```
a－ =1     /*等价于 a=a－1*/
a%=3       /*等价于 a=a%3*/
```

若 a=12，则 a+=a－=a*a 的结果为 a=－264，该式等价于 a=a+(a=a－(a*a))。

若 a=2，则 a+=a*=a－=a*=3；结果为 a=0，该式等价于 a=a+(a=a*(a=a－(a=a*3)))。

说明：

(1) 赋值运算符的优先级较低，为 14 级(参见附录 B)。

(2) 赋值运算符的结合方向为自右向左。

(3) 赋值运算符的左侧必须是变量，不能是常量或表达式。

复合赋值运算符<<=、>>=、|=、&=、^=与位运算有关将在第 9 章介绍。

2.6.5 逗号运算

逗号运算符是“,”，其作用是将多个表达式连在一起。逗号表达式是用逗号运算符连接表达式组成的式子，它的一般形式如下：

```
表达式 1,表达式 2,…,表达式 n
```

逗号表达式的求解过程：先计算表达式 1 的值，再计算表达式 2 的值，…，计算表达式 n 的值，将表达式 n 的值作为整个逗号表达式的结果。

逗号运算符的优先级为 15 级，是所有运算符中最低的，其结合方向是自左向右。

例如：

```
a=3*5,a*4                    /*  a=15，表达式的值为 60 */
a=3*5,a*4,a+5                /*  a=15，表达式的值为 20 */
x=(a=3,6*3)                  /* 赋值表达式，表达式的值为 18，x=18 */
x=a=3,6*a                    /* 逗号表达式，表达式的值为 18，x=3 */
a=1;b=2;c=3;
printf("%d,%d,%d",a,b,c);    /* 1,2,3 */
printf("%d,%d,%d",(a,b,c),b,c);  /* 3,2,3 */
```

2.6.6 条件运算符和条件表达式

条件运算
与 sizeef 运算

条件运算符是 C 语言中唯一的三目运算符。条件运算符的形式是“? :”，由它构成的表达式称为条件表达式。其一般形式如下：

```
表达式 1？表达式 2:表达式 3
```

功能：先计算表达式 1 的值，若值为非 0，则计算表达式 2 的值，并将表达式 2 的值作为整个条件表达式的结果；若表达式 1 的值为 0，则计算表达式 3 的值，并将表达式 3 的值作为整个条件表达式的结果。

例如：

```
(a==b)? 'Y': 'N'
(x%2==1)? 1:0
(x>=0)? x:－x
(c>= 'a' && c<= 'z')? c－'a'+ 'A':c
```

条件运算符的优先级为 13 级，高于赋值运算符，但低于关系运算符、逻辑运算符和算术运算符。其结合方向是自右向左，当多个条件表达式嵌套使用时，每个后续的“:”总与前面最近的、没有配对的“?”相联系。

例如，条件表达式“a>0 ? a/b:a<0 ? a+b:a−b”等价于“a>0 ? a/b: (a<0 ? a+b:a−b)”。

使用条件表达式可以使程序简洁明了。例如，赋值表达式“z=(a>b)?a:b”中使用了条件表达式，很简洁地表示了判断变量 a 与 b 的值、将较大值赋给变量 z 的功能。所以，使用条件表达式可以简化程序。

【例 2-5】从键盘接收一个字符，要求只把输入的小写字母转换成大写字母，其他字符不变，并显示结果。

```
#include<stdio.h>
int main(void)
{
   char ch,c;
   scanf("%c",&ch);
   c=ch>='a'&&ch<='z'?ch-32:ch;
   printf("%c\n",c);
   return 0;
 }
```

2.6.7　sizeof 运算符

sizeof 是数据长度测试运算符，其一般形式如下：

```
sizeof(exp)
```

其中 exp 表示 sizeof 运算符所要运算的对象，exp 可以是表达式或数据类型名。sizeof 是单目运算符。

功能：求出运算对象在计算机内存中占用的字节数。例如：

```
sizeof(char)              /*  求字符型数据在内存中占用的字节数，结果为 1  */
sizeof(3*1.46/7.28)       /*  求运算结果在内存中占用的字节数，结果为 8  */
```

【例 2-6】用 sizeof 测试所用计算机系统中各种数据类型的数据所占内存的字节数。

```
#include <stdio.h>
int main(void)
{
   char ch='A';
   int x=7,y=8;
   float a=3.27f,b=6000.0f;
   printf("char:%d\n",sizeof(ch));
   printf("short int:%d    int:%d long int:%d\n",sizeof(short int),sizeof(int),
   sizeof(long int));
   printf("float:%d\n",sizeof(a));
   printf("double:%d       long double:%d\n",sizeof(double),sizeof(long double));
   printf("int express:%d\n",sizeof(x+y));
   printf("float express:%d\n",sizeof(a+b));
```

```
    printf("character express:%d\n",sizeof('a'-'0'));
    return 0;
}
```

该程序在 Code::Blocks 中的运行结果如下：

```
char:1
short int:2  int:4  long int:4
float:4
double:8    long double:12
int express:4
float express:4
character express:4
```

2.6.8 类型转换

在 C 语言中，整型、实型和字符型数据之间可以进行混合运算。不同类型的数据在进行赋值、混合运算时，要先转换成同一类型，然后再进行运算。


类型转换

类型转换可归纳成两种转换方式：隐式类型转换和强制类型转换。

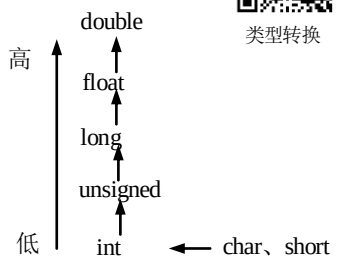


图 2-5　隐式类型转换规则

1. 隐式类型转换

隐式类型转换也称自动类型转换，由系统自动完成，这种类型转换不会体现在 C 语言源程序中。但是，程序设计人员必须了解这种自动转换的规则及结果，否则会引起对程序执行结果的误解。

隐式类型转换通常发生在运算、赋值、输出、函数调用中。其中，函数调用转换在实参与形参的类型不一致时转换(具体转换规则将在本书第 6 章介绍)。

1) 运算转换

C 语言允许进行整型、实型、字符型数据的混合运算，在运算时，会将不同类型的数据转换成同一类型后再进行运算。转换规则如图 2-5 所示。

(1) 横向箭头表示运算时必定发生转换，即运算时将表达式中的所有 char 和 short 型数据转换成 int 型后再进行运算。

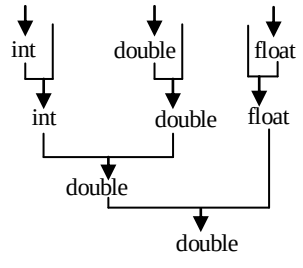
(2) 纵向箭头表示当表达式中含有不同类型数据时的转换方向，如 int 型和 double 型的数据进行运算时，先将 int 型转换成 double 型后再进行运算，运算结果为 double 型。

下面给出类型转换的示例，以加深理解。

例如：

char ch;　int i;　float f;　double d;

ch/i　+　f*d　−　(f+i)



以上步骤中的类型转换是由 C 语言编译系统自动完成的。计算机运算时从左向右扫描表达式，先扫描到 ch/i+，由于“/”比“+”优先级高，先计算 ch/i，将 ch 转换成 int 型进行运算，结果为 int 型。再执行运算符“+”和“*”的比较，“*”的优先级较高，“+”先不运算。再向后扫描，遇到“-”，“-”的优先级低于“*”，所以进行 f*d 运算，将 f 转换成 double 型进行运算，结果为 double 型。从左向右有“+”和“-”，“+”在前，先计算“+”，将“+”左边的 int 型转换成 double 型，与其右边的 double 型相加，结果为 double 型。“-”和“(”相比，括号的优先级高，先计算括号里面的，将 int 型转换成 float 型，运算结果为 float 型。最后进行“-”运算，“-”运算符的左边为 double 型，所以将右边的 float 型转换成 double 型，进行“-”运算，结果为 double 型。

2) 赋值转换

在进行赋值运算时，如果赋值运算符两边的数据类型不一致，将由系统进行自动类型转换。转换原则是以“=”左边的变量类型为准，先将赋值符号右边表达式的类型转换为左边变量的类型，然后赋值。

警告：

在赋值时要注意以下情况。

(1) 将实型数据(单精度和双精度)赋给整型变量，舍弃实数的小数部分。

例如:

```
float a=6.7;int b;b=a;          /* 则 b 的值为 6。*/
```

(2) 将整型数据赋给单精度和双精度实型变量，数值不变，但以浮点数形式存储到变量中。

例如:

```
float a; int b=5; a=b;          /* 则 a 的值为 5.000000。*/
```

(3) 将 double 型数据赋给 float 型变量时，截取其前面 7 位有效数字，存放到 float 变量的存储单元中(32 位)。但应注意数值范围不能溢出。将 float 型数据赋给 double 型变量时，数值不变，有效位数扩展到 16 位。

(4) 将字符型数据赋给整型变量时，由于字符型数据只占 1 字节，而整型变量占 4 字节，因此将字符数据(8 位)放到整型变量的低 8 位中。其他位的数据有以下两种情况。

- 如果所使用的系统将字符处理为无符号字符(unsigned char)，则将字符的 8 位放到整型变量的低 8 位，其余位补 0。
- 如果所使用的系统将字符处理为带符号(char)的字符(如 Code::Blocks)，若字符的最高位为 0，将字符数据(8 位)置于整型变量的低 8 位，整型变量其余位补 0；若字符的最高位为 1，则为整型变量的其余位全补 1。这称为符号扩展，这样做的目的是使数值保持不变。

(5) 将 int、short、long 型数据赋给 char 型变量时，只是将其低 8 位原封不动地送到 char 型变量(即截断)。

(6) 不同类型的整型数据间的赋值是按照存储单元的存储形式直接传送的。将长整型数赋值给短整型变量时，截断直接传送；将短整型数赋值给长整型变量，低位直接传送，高位根据被传送整数的符号进行符号扩展。

【例 2-7】从键盘输入身高(cm)的整数值，显示出标准体重(kg)的实数值。成人标准体重的计算公式为：标准体重=(身高−100)×0.9。

分析：该计算为整型和 double 型的数据混合运算，结果为 double 型。根据存储数据精度的需要，可将结果存储在 float 型变量中。

```
#include<stdio.h>
int main(void)
{
   int height;
   float weight;
   printf("please input your height (cm):");
   scanf("%d",&height);
   weight=(height-100)*0.9;
   printf("your standard weight is %f kg",weight);
   return 0;
}
```

2. 强制类型转换

强制类型转换的一般形式如下：

```
(类型名)表达式
```

其中，表达式是任何一种类型的表达式。强制类型转换运算的含义是将右边表达式的值转换成括号中指定的数据类型。这是一种显式的数据类型转换方式。

例如：

```
(double)n      /* 将 n 强制转换成 double 型。*/
(int)(a*b)     /* 将 a*b 的结果强制转换成整型。注意，不要写成 int (a*b)。*/
(int)a*b       /* 将 a 强制转换成整型后，再与 b 相乘求出结果。*/
```

说明：强制转换得到所需类型的中间值，原变量的类型不变。

【例 2-8】强制类型转换示例。

```
#include<stdio.h>
int main(void)
{
   float x;
   int i;
   x=3.6;
   i=(int)x;
   printf("x=%f,i=%d",x,i);
   return 0;
}
```

程序运行结果如下：

```
x=3.600000,i=3
```

2.7 数据的输入/输出

一个功能相对独立的程序一般包括三部分：第一部分为变量提供数据(数据输入)，第二部分为计算处理部分，第三部分为结果输出。

C 语言没有输入/输出语句。数据的输入/输出通过调用 C 的标准库函数来实现。在使用标准库函数时，需要用预编译命令“#include”将有关的“头文件”包含到用户源文件中，“头文件”中包含了与所用函数有关的说明信息。标准输入/输出函数在头文件“stdio.h”中则说明，要使用它们，可在源程序文件开头部分使用#include<stdio.h>命令。

2.7.1 字符数据的输入/输出

1. putchar( )函数

putchar( )函数为字符输出函数，该函数调用的一般形式如下：


字符输入输出

```
putchar(ch);
```

其中，ch 可以是一个字符常量或字符变量。

功能：向终端(显示器)输出一个字符 ch(可以是可显示的字符，也可以是控制字符或其他转义字符)。

例如：

```
putchar('y');       /* 输出字符 y */
putchar('\n');      /* 输出一个回车换行字符 */
putchar('\101');    /* 输出字符 A */
putchar('\'');      /* 输出字符' */
```

2. getchar( )函数

getchar( )函数为字符输入函数，该函数调用的一般形式如下：

```
ch=getchar();
```

功能：从终端(键盘)接收一个字符，将该字符赋给字符变量 ch。

说明：该函数没有参数，但括号不能省略；ch 为字符变量。

【例 2-9】从键盘输入一个字符，并在显示器上输出该字符。

```
#include<stdio.h>
int main(void)
{ char c;
  c=getchar( );
  putchar(c);
  return 0;
}
```

2.7.2 格式化输出和输入函数

scanf( )和 printf( )函数是 C 语言编程中使用最为频繁的两个函数，它们用来格式化输入和输出数据。scanf( )函数读取用户从键盘输入的数据，并把数据传递给程序；printf( )函数读取程序中的数据，并把数据显示在屏幕上。把这两个函数结合起来，就可以建立人机双向通信，如图 2-6 所示，这让使用计算机更加饶有趣味。

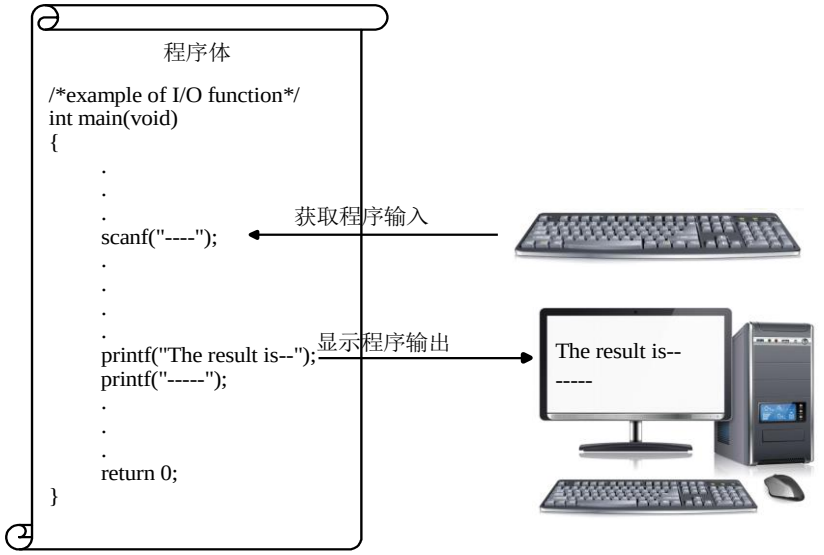


图 2-6　程序中的 scanf( )和 printf( )函数

1. 格式输出函数 printf( )

printf( )函数调用的一般形式如下：

数据格式输出(1)

```
printf(格式字符串，输出项表);
```

功能：按格式字符串中的格式依次输出“输出项表”中的各个数据。

说明：格式字符串用于说明输出项表中各输出项的输出格式，即输出数据的格局安排。输出项表列出要输出的项(常量、变量或表达式)，输出项可以没有，也可以有多个。如果有多个，各输出项之间用逗号分开。

例如：

```
printf("How are you\n");              /* 输出：How are you 并换行 */
printf("r=%d,s=%f\n",2,3.14*2*2);     /* 输出：r=2,s=12.560000 */
```

格式字符串中有以下两类字符。

(1) 非格式字符，非格式字符(或称普通字符)一律按原样输出。

例如：

```
printf("a=%d,b=%f\n",a,b);
```

“a=”、“,”、“b=”按原样输出。

(2) 格式字符，用于指定输出格式。

格式字符的一般形式如下：

%[附加格式说明符]格式符

比如%d、%10.2f 等。其中，%d 格式符表示以十进制整型格式输出数据，而%f 表示以实型格式输出数据，附加格式说明符“10.2”表示输出宽度为 10，输出两位小数。常用的 Printf 格式符见表 2-6，附加格式说明符见表 2-7。

表 2-6　格式符

格式符	功能	示例	结果
d、i	输出带符号十进制整数	int a=567; printf ("%d",a);	567
o	输出无符号八进制整数	int a=65; printf("%o",a);	101
x、X	输出无符号十六进制整数	int a=255; printf("%x",a);	ff
u	输出无符号整数	int a=567; printf("%u",a);	567
c	输出单个字符	char a=65; printf("%c",a);	A
s	输出一串字符	printf("%s", "ABC");	ABC
f	输出实数(6 位小数)	float a=567.789;printf("%f",a);	567.789001
e、E	以指数形式输出实数(尾数含 1 位整数、6 位小数，指数至多 3 位)	float a=567.789; printf("%e",a);	5.677890e+002
g、G	选用 f 与 e 格式中输出宽度较小的格式，且不输出无意义的 0	float a=567.789; printf("%g",a);	567.789

表 2-7　附加格式说明符

附加格式说明符	功能
m(m 为正整数)	数据输出宽度为 m，数据长度<m，左补空格，否则按实际输出
.n(n 为正整数)	对于实数，n 是输出的小数位数；对于字符串，n 表示输出前 n 个字符
-	数据左对齐输出，默认右对齐输出
+	指定在有符号数的正数前显示正号(+)
0	输出数值时指定左侧不使用的空位置自动填 0
#	在八进制和十六进制数前显示前导 0、0x
l	在 d、o、x、u 前，指定输出 long 型数据；在 e、f、g 前，指定输出 double 型数据

说明：

(1) 格式符除 X、E、G 外必须用小写字母，否则无效。

例如：printf("%D", 10);输出%D 而不是 10。

(2) 对于允许大写格式的 X、E、G，输出结果中含有字母的，字母用大写表示。例如：int a=255; printf("%X",a);，结果为 FF。

数据格式输出(2)

(3) 如果输出字符串中包含%，则需要连续使用两个%。

例如：printf("a=%d%%", 10);，输出 a=10%。

(4) 格式符与输出项的个数应相同，按先后顺序一一对应。

(5) 输出转换：格式符与输出项的类型不一致时，自动按指定格式输出。

参照下面的例题来理解附加格式说明符的意义。

【例 2-10】输出数据时使用 m.n 格式。

```
#include<stdio.h>
int main(void)
 { int a=1234;
   float f=123.456;
   char ch='a';
   printf("12345678901234567890123456789012345678901234567890\n");
   printf("%8d,%2d\n",a,a);
   printf("%f,%8f,%8.1f,%.2f,%.2e,%g\n",f,f,f,f,f,f);
   printf("%3c\n",ch);
   return 0;
 }
```

程序执行结果如下：

```
12345678901234567890123456789012345678901234567890
    1234,1234
123.456001,123.456001,    123.5,123.46,1.23e+002,123.456
  a
```

说明：为理解屏幕上的输出位置，可参照程序中先输出的一行“1234567890123...”，以它为标尺行。

【例 2-11】输出字符串时使用 m.n 格式。

```
#include<stdio.h>
int main(void)
 {
   static char a[]="Hello,world!";
   printf("12345678901234567890123456789012345678901234567890\n");
   printf("%s\n%15s\n%10.5s\n%2.5s\n%.3s\n",a,a,a,a,a);
   return 0;
 }
```

程序执行结果如下：

```
12345678901234567890123456789012345678901234567890
Hello,world!
   Hello,world!
     Hello
Hello
Hel
```

【例 2-12】输出数据时使用-。

```
#include<stdio.h>
int main(void)
{
   int a=1234;
   float f=123.456;
   static char c[]="Hello,world!";
```

```
    printf("12345678901234567890123456789012345678901234567890\n");
    printf("%8d,%-8d#\n",a,a);
    printf("%10.2f,%-10.1f#\n",f,f);
    printf("%10.5s,%-10.3s#\n",c,c);
    return 0;
}
```

程序执行结果如下：

```
12345678901234567890123456789012345678901234567890
    1234,1234    #
    123.46,123.5     #
     Hello,Hel       #
```

【例 2-13】输出数据时使用 0、+。

```
#include<stdio.h>
int main(void)
{  int a=1234;
   float f=123.456;
   printf("12345678901234567890123456789012345678901234567890\n");
   printf("%08d\n",a);
   printf("%010.2f\n",f);
   printf("%0+8d\n",a);
   printf("%0+10.2f\n",f);
   return 0;
 }
```

程序执行结果如下：

```
12345678901234567890123456789012345678901234567890
00001234
0000123.46
+0001234
+000123.46
```

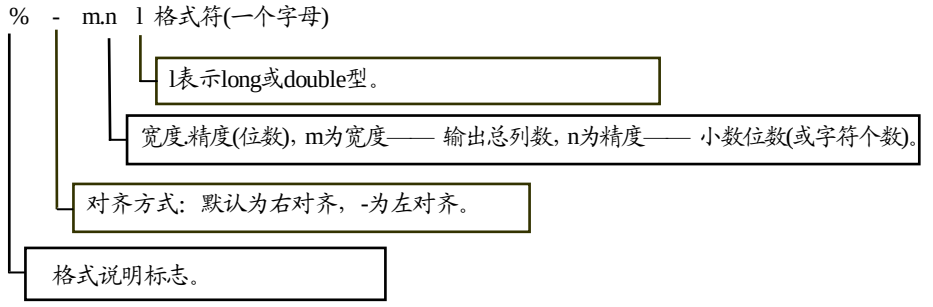
【例 2-14】输出数据时使用#。

```
#include<stdio.h>
int main(void)
 {
   int a=123;
   printf("12345678901234567890123456789012345678901234567890\n");
   printf("%o,%#o,%X,%#x\n",a,a,a,a);
   return 0;
 }
```

程序执行结果如下：

```
12345678901234567890123456789012345678901234567890
173,0173,7B,0x7b
```

格式说明符的一般形式和意义如下。



总结：格式输出函数就是用输出项表中的各项数据，替换格式字符串中的格式字符，与非格式字符一起组成一个完整的字符串并输出。

2. 格式输入函数 scanf( )

与格式输出函数 printf( )对应的是格式输入函数 scanf( )。

scanf( )函数调用的一般形式如下：


数据格式输出(1)

scanf(格式字符串，地址列表);

功能：按格式字符串中规定的格式，从键盘读取输入的数据，并依次赋给指定地址。

说明：“格式字符串”与 printf( )函数中的“格式字符串”大部分相同，但不能显示格式说明符以外的字符，即不能显示提示信息，格式说明符以外的字符要原样输入。

“地址列表”是要接受输入值的各变量地址，变量地址由“&”后跟变量名组成，&是取址运算符，如&a 表示变量 a 的地址。

例如 scanf("%d,%f",&a,&b)，若要将 3 赋给 a，将 2.43 赋给 b，输入数据时要在键盘上输入“3,2.43”，若采用其他的输入方式，a 和 b 均得不到预期的值。

格式字符串的一般形式如下：

%[附加格式说明符]格式符

scanf 格式符有 d、i、o、x、u、c、s、f、e、g 等，功能如表 2-8 所示。

附加格式说明符有 m、h、l、*等，功能如表 2-9 所示。

表 2-8　scanf 格式符

格式符	功能
d、i	用于输入有符号十进制数，为整型变量赋值
o	用于输入无符号八进制数，为无符号整型变量赋值
u	用于输入无符号十进制数，为无符号整型变量赋值
X、x	用于输入无符号十六进制数，为无符号整型变量赋值
c	用来输入单个字符，为字符型变量赋值
s	用来输入字符串，为字符数组赋值
f 或 e(E)	用来输入单精度数，可以是小数形式或指数形式，为单精度类型变量赋值
g(G)	与 f 或 e 的作用相同

表 2-9　scanf 附加格式说明符

附加格式说明符	功能
m(m 为正整数)	指定输入数据所占宽度
l 或 L	用于 d、o、x、u 前，指定输入 long 型数据；用于 e、f、g 前，指定输入 double 型数据
*	读入对应的输入项后，不赋给变量，用于输入时跳过一些数据

说明：

(1) 输入 double 型数据时必须用%lf 或%le，在 printf( )函数中输出 double 型数据可以用%f 或%e。


数据格式输出(2)

(2) 格式字符串中不包含普通字符，输入数据时可以用空格、回车或 Tab 键作为数据的分隔符。

```
scanf("%d%d%d",&a,&b,&c);
```

输入序列为：2␣3␣4↙(␣表示空格，→表示 TAB 键，↙表示回车键)
可以是：2→3→4↙
可以是：2␣␣␣3→␣4↙
也可以是：2↙
　　3↙
　　4↙
等组合，变量可以得到输入的数据。

(3) 格式字符串中包含的普通字符必须按原样输入，例如:

```
scanf("%d:%d:%d",&a,&b,&c);
```

应输入序列：2:3:4↙，数据之间用“:”分隔。

```
scanf("a=%d",&a);
```

应输入序列：a=2↙，a=这样的辅助信息也必须输入。否则，变量得不到预期的数据。

(4) 可以指定输入数据所占的列数，系统自动按它截取所需数据。

```
scanf("%3d%3d%3d",&a,&b,&c);
```

若输入：123456789↙　　则 a 为 123，b 为 456，c 为 789
若输入：12␣3456␣789↙　　则 a 为 12，b 为 345，c 为 6

```
scanf("%3c",&a);
```

若输入 efghijk↙，则 a 的值为 e，因为 a 变量只能存储一个字符。

(5) 用%c 输入字符时，空格、Tab 键和回车键等也作为有效字符被接收，例如:

```
scanf("%c%c%c",&a,&b,&c);
```

若输入：efg↙　　则 a 的值为 e，b 的值为 f，c 的值为 g。
若输入：e␣f␣g↙　　则 a 的值为 e，b 为␣，c 为 f，其中的分隔符也作为有效字符被接收。

(6) 输入数据时，赋值规则是：遇到空格、跳格(Tab 键)、回车键、宽度结束或非法输入时认为数据输入结束。

【例 2-15】格式数据输入示例。

```
#include<stdio.h>
int main(void)
{
   int a,b,d; char c;
   scanf("%d%d%c%3d",&a,&b,&c,&d);       //其中的 3 用于限制获取输入内容的宽度
   printf("a=%d,b=%d,c=%c,d=%d",a,b,c,d);
   return 0;
}
```

如果输入序列为：10␣11A12345↙

输出结果如下：

```
10 11A12345
a=10,b=11,c=A,d=123
```

程序说明：

(1) 对于整型变量 a，获得“10”，遇到空格表示为 a 赋值结束。

(2) 对于整型变量 b，获得“11”。对于整型变量来说，后面的字符“A”是非法输入，所以为 b 赋值结束。

(3) 对于字符型变量 c，刚好对应单字符'A'，为 c 赋值结束。

(4) 对于整型变量 d，本来可以获得赋值 12345，但因为被限制只能获取 3 位，故截取 12345 的前三位 123。

(5) 如果 11 后加一个空格，则这个空格会被赋给变量 c，因为空格也是一个字符。

注意：

printf 格式字符串经常以\n 结尾，但在 scanf 格式字符串的末尾放置换行符通常是一个坏主意。对于 scanf 函数来说，格式字符串中的换行符等价于空格，两者都会引发 scanf( )函数等待下一个数据(非分隔符)输入，例如 scanf("%d\n",&a)这样的格式字符串可能会导致交互式程序一直“挂起”，直到用户输入一个非分隔符为止。

2.8 案例：鸡兔同笼

进阶提高

“鸡兔同笼”是我国隋朝时期数学著作《孙子算经》中的一个有趣而具有深远影响的题目。“今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何”。

分析：设有鸡(雉)x 只，兔 y 只，鸡和兔的总头数为 h，总脚数为 f。根据题意可以写出下面的方程式：

$$\begin{cases} x+y=h \\ 2x+4y=f \end{cases} \xrightarrow{\text{可推出 } x、y \text{ 的表达式}} \begin{cases} x=\dfrac{4h-f}{2} \\ y=\dfrac{f-2h}{2} \end{cases}$$

本例中 *h* 和 *f* 的值已知，可以使用赋值语句将数据直接写到程序中，计算后输出结果。

```
#include <stdio.h>
int main(void)
{
  int x,y,h,f;
  h=35;
  f=94;
  x=(4*h-f)/2;
  y=(f-2*h)/2;
  printf("笼中的鸡是%d 只，兔子是%d 只\n",x,y);
  return 0;
}
```

运行结果如下：

```
笼中的鸡是 23 只，兔子是 12 只
```

本章小结

本章介绍了 C 语言基本数据类型中的整型、实型和字符型数据类型，变量、常量的基本概念，数据在计算机中的存储，算术、关系、逻辑、赋值等运算符与运算规则的基础知识，以及数据的输入/输出函数。这些内容是 C 语言中的基础知识，内容多且繁杂，需要在理解的基础上反复记忆和熟练应用，以达到通过编写简单的程序解决生活中问题的目的。本章涉及的有关知识的结构导图如图 2-7 所示。

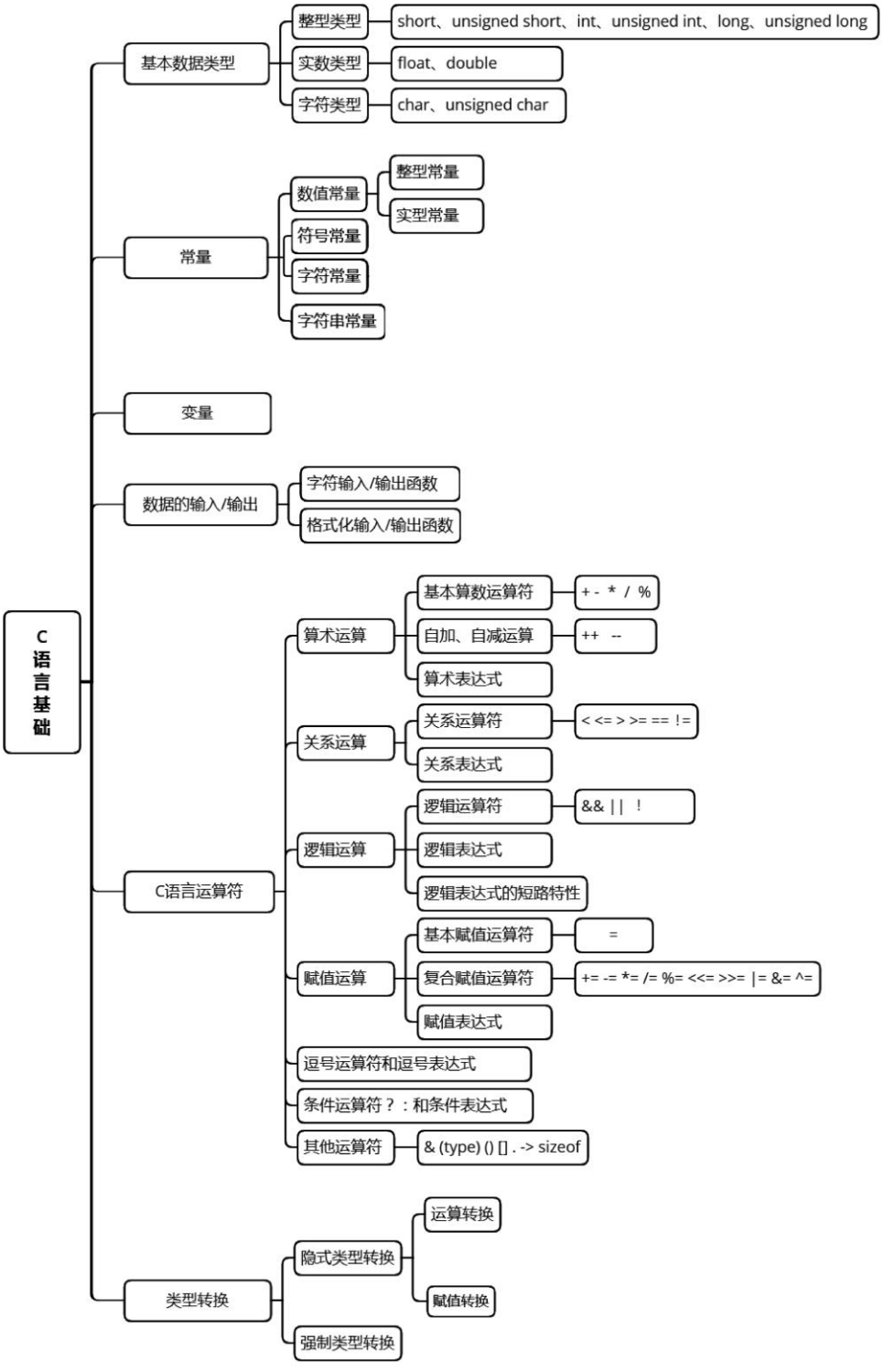


图 2-7　基本数据类型知识导图

习　题

一、单选题

1. 整型(int)数据在内存中的存储形式是(　　)。

A. 原码　　B. 补码　　C. 反码　　D. ASCII 码

2. 在 C 语言中，整数−8 在内存中的存储(占 2 字节)形式是(　　)。

A. 1111 1111 1111 1000　　B. 1000 0000 0000 1000

C. 0000 0000 0000 1000　　D. 1111 1111 1111 0111

3. 下列四组选项中，均不是 C 语言关键字的选项是(　　)。

A. define	B. gect	C. include	D. while
IF	char	scanf	pow
typeprintf	case	sizeof	go

4. 以下选项中不合法的用户标识符是(　　)。

A. abc.c　　B. file　　C. Main　　D. PRINT

5. 下列常数中不能作为 C 语言常量的是(　　)。

A. 0xA6　　B. 4.5e−2　　C. 3e2　　D. 0584

6. 下列可以正确表示字符型常量的是(　　)。

A. "c"　　B. '\t'　　C. "\n"　　D. 297

7. 字符串"\\\22a,0\n"的长度是(　　)。

A. 8　　B. 7　　C. 6　　D. 5

8. 下面的变量定义中，(　　)是正确的。

A. char：a, b, c;　　B.char a; b; c;　　C.char a, b, c;　　D.char a, b, c

9. 若 int a,b,c;，则为它们输入数据的正确输入语句是(　　)。

A. read(a,b,c);　　B. scanf(" %d%d%d" ,a,b,c);

C. scanf(" %D%D%D" ,&a,%b,%c);　　D. scanf(" %d%d%d",&a,&b,&c);

10. 若 float a,b,c;，要通过语句 scanf(" %f %f %f" ,&a,&b,&c);分别为 a、b、c 输入 10、22、33。以下不正确的输入形式是(　　)。

A.10	B. 10.0,22.0,33.0	C. 10.0	D. 10 22
22		22.0	33
33		33.0	

11. 若在键盘上输入 283.1900，想使单精度实型变量 c 的值为 283.19，则正确的输入语句是(　　)。

A. scanf(" %f",&c);　　B. scanf(" %8.4f",&c);

C. scanf(" %6.2f",&c);　　D. scanf(" %8",&c);

12. 执行语句 printf(" |%10.5f|\n",12345.678);，输出的是(　　)。

A. |2345.67800|　　B. |12345.6780|　　C. |12345.67800|　　D. |12345.678|

13. 若 a 为 int 型，且 a=125，执行下列语句后的输出是(　　)。

```
printf("%d,%o,%x\n",a,a+1,a+2)
```

A. 125,175,7D　　B. 125,176,7F　　C. 125,176,7D　　D. 125,175,2F

14. 使用语句 scanf("x=%f,y=%f",&x,&y);，输入变量 x、y 的值(␣代表空格)，正确的输入是(　　)。

A. 1.25,2.4　　B. 1.25␣2.4　　C. x=1.25,y=2.4　　D.x=1.25␣y=2.4

15. 表达式(　　)的值是 0。

A. 3/5　　B. 3/5.0　　C. 3%5　　D. 3<5

16. 已知 int i, a;，执行语句 i=(a=2*3,a*5),a+6;后，变量 i 的值是(　　)。

A. 6　　B. 30　　C.12　　D.36

17. 已知字母 A 的 ASCII 码为十进制数 65，且 ch 为字符型变量，则执行语句 ch='A'+'6'−'3';后，ch 中的值为(　　)。

A. E　　B. 68　　C. 不确定　　D. C

18. 表达式 5>3>1 的值是(　　)。

A. 0　　B. 1　　C. 3　　D. 表达式语法错误

19. 设 x、y、t 均为 int 型变量，则执行语句：x=y=3; t= ++x || ++y; 后，y 的值为(　　)。

A. 3　　B. 不定值　　C. 4　　D.1

20. 已有下列语句组：int x;scanf("%d",&x);，能正确计算并输出 x 绝对值的语句是(　　)。

A. printf("%d 的绝对值是:%d\n",x,x>=0?x:-x);

B. printf("%d 的绝对值是:%d\n",x,x<=0?x:-x);

C. printf("%d 的绝对值是:%d\n",x,x<0?x:-x);

D. printf("%d 的绝对值是:%d\n",x,!x<=0?x:-x);

21. 已有下列语句组：float x,y;scanf("%f%f",&x,&y);，如果 x 为点的横坐标，y 为点的纵坐标，能正确判断点(x,y)在第一象限的逻辑表达式是(　　)。

A. x>0&&!y<0　　B.x>0||y>0　　C. x>0&&y>0　　D. !x<0||!y<0

22. 下列选项中用于判断 ch 是否是字母的表达式是(　　)。

A. ('a' <= ch <= 'z') || ('A' <= ch <= 'Z')

B. ('a' <= ch <= 'z') && ('A' <= ch <= 'Z')

C. (ch >= 'a' && ch <= 'z') && (ch >= 'A' && ch <= 'Z')

D. (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')

23. 设 a 为整型变量，不能正确表达数学关系：10<a<15 的 C 语言表达式是(　　)。

A. 10<a<15

B. a==11|| a==12 || a==13 || a==14

C. a>10 && a<15

D. !(a<=10) && !(a>=15)

24. 下列只有当整数 x 为奇数时，其值为“真”的表达式是(　　)。

A. x%2==0　　B. !(x%2==0)

C. (x-x/2*2)==0　　D. !(x%2)

二、填空题

1. 在 C 语言程序中，用关键字_____定义基本整型量，用关键字_____定义单精度实型变量，用关键字_____定义双精度实型变量。

2. 已知字母 A 的 ASCII 码为 65，以下程序的输出结果是_____。

```
#include<stdio.h>
int main(void)
{  char c1='A',c2='Y';
    printf("%d,%d\n",c1,c2);
    return 0;
}
```

3. 当运行以下程序时，在键盘上从第一列开始输入 9876543210↙(此处↙代表回车)，则程序的输出结果是_____。

```
#include<stdio.h>

int main(void)
{ int a; float b,c;
   scanf(" %2d%3f%4f",&a,&b,&c);
   printf(" \na=%d,b=%f,c=%f\n",a,b,c);
   return 0;
}
```

4. 若定义 double a,b,c;，要求为 a、b、c 分别输入 10、20、30。输入序列为(␣表示空格)：
␣10.0␣␣20.0␣␣30.0↙
则正确的输入语句是_____。

5. 假设变量已正确定义并赋值，写出满足下列条件的 C 表达式。

① 与数学式 $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$ 对应的 C 表达式：_____。

② 取 number 的个位数字：_____。
③ 取 number 的十位数字：_____。
④ 取 number 的百位数字：_____。
⑤ number 是奇数：_____。
⑥ number 是 5 的倍数：_____。

三、编程题

1. 编程定义一个 int 型的变量，初值为 97，依次按字符、十进制、八进制、十六进制格式输出该变量的值。

2. 编写程序，输入时间 10:27 并把它转换成分钟后输出(从零点整开始计算)。

3. 已知地球半径为 6371 km，编写程序，求地球的表面积和体积。

4. 假设一名工人一个月的工资是 4367 元人民币，试计算发工资时人民币各面值(由大到小)的张数。

5. 假设有 3 个电阻并联，阻值分别为 5Ω、15Ω、20Ω，编程求并联后的电阻值。

(提示：设 R 为总电阻，并联的三个电阻分别为 R_1、R_2、R_3，3 个电阻并联后总电阻的计算公式为 $\frac{1}{R}=\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}$。)

6. 从键盘输入一个 4 位正整数，编程将此 4 位数逆置后输出。如输入 1234，输出 4321。

7. 要求输入一个华氏温度 F，编程计算并输出与之对应的摄氏温度 C。公式为：$C=\frac{5}{9}(F-32)$。

8. 由键盘任意输入 3 个数字字符('0'~'9')，将其转换为对应的数字并输出。比如输入的是数字字符'3'，转换为数字 3 并输出。

9. 执行下列程序，按指定方式输入(␣表示空格)，判断能否得到指定的输出结果？若不能，请修改程序，使之能得到指定的输出结果。

输入： 2␣3␣4↙

输出： a=2,b=3,c=4

x=6,y=24

程序：

```
int main(void)
{ int a, b, c, x, y;
   scanf("%d, %d, %d", a, b, c);
   x=a*b; y=x*c;
   printf("%d %d %d", a, b, c);
   printf("x=%f\n",x, "y=%f\n",y);
   return 0;
}
```