第 1 章　认识 C 语言

计算机是现代信息处理的重要工具。计算机通过软件操控硬件的方式来实现各种信息处理任务。各式各样的软件极大地丰富了计算机的功能，不断拓展其应用领域，帮助人们解决了众多生产、生活和学习中的实际问题。

C 语言作为当今最流行的程序设计语言之一，被广泛应用于各行各业的程序设计中。本章首先简要介绍 C 语言的历史及特点，然后详细介绍 C 语言程序的基本结构、操作过程、基本语法元素及编程规范。

本章要点：

- 了解 C 语言的历史及特点
- 掌握算法的概念及其流程图表示方法
- 掌握编写 C 语言程序的一般操作过程
- 理解 C 语言程序的基本结构
- 掌握 C 语言的基本语法元素(标识符、常量和变量等)和基本数据类型
- 了解 C 语言的语法规范和编程规范

程序设计技术与主流编程语言

1.1　C 语言的历史及特点

软件的主体是程序，程序是能实现特定功能的一组有序的指令序列。计算机程序一般采用某种计算机语言并根据特定的方法和步骤编制而成。C 语言是一种功能强大的专业化编程语言，具有简洁、高效、灵活等特点，一直深受专业程序员和业余编程爱好者的喜爱。

1.1.1　计算机语言的发展

计算机语言是人类和计算机进行相互“交流”的语言。自计算机诞生以来，人们已先后发展了多种计算机语言，根据这些语言的编程特点，可将其分为机器语言、汇编语言和高级语言三种。

1. 机器语言

计算机只能识别由 0 和 1 组成的机器代码。不同的计算机具有不同的指令集合，亦即具有自身可以理解的机器语言。最初，人们直接使用符号 0 和 1 表示的机器语言进行编程。显然，这样编写的程序虽然具有计算机能够直接识别并执行的优点，但本身所固有的程序难以理解、和具体硬件设备相关、且极易出错等缺点使得这种编程方式较难掌握，而且编写的程序通常不具备任何移植性。例如，为一台 IBM PC 机编写的机器语言程序，无法移植到一台

APPLE MAC 计算机上，甚至不能在另一台不同型号的 IBM PC 机上运行。

2. 汇编语言

汇编语言采用类英语单词助记符来简化程序的编写，这大大降低了编程的难度，程序的可读性也得到明显提高。但由于计算机并不能直接识别程序中所用的助记符，因此用汇编语言编写的程序必须通过一个被称为“汇编程序”的中间转换工具进行处理，才能得到计算机能够执行的机器语言程序，这个过程通常称为“汇编”。实际上，汇编语言所采用的助记符本质上是对特定 CPU 内部指令的一种简略记法，因此和机器语言一样，汇编语言也是和硬件平台相关的。通常，人们将与设备相关的机器语言和汇编语言统称为低级语言。

3. 高级语言

人们一直渴望能以接近人类自然语言的方式来编写程序，高级语言正是在这样的背景下产生的。所谓高级语言，是相对机器语言和汇编语言而言的，这种编程方式采用类似人的自然语言(主要指英语)的方式进行程序设计，不用考虑 CPU 内部执行程序指令的具体步骤，因而显著降低了编程难度。用高级语言编制的程序具有简洁直观、可读性好、编程效率高、移植性强等优点，但和汇编语言一样，计算机并不能直接识别高级语言编写的程序，必须通过中间转换工具处理后才能转换成计算机可识别的二进制代码，即机器指令。

对于高级语言编制的程序(通常称为源程序)，常采用“解释”和“编译”两种方式将其转换为机器语言程序。其中，解释方式是一种“边翻译、边执行”的过程，类似日常生活中的“口译”，是一种对源程序进行逐条解释为机器指令并立即执行的方式，最终不产生任何目标程序。而编译方式是一种“整体翻译”的过程，类似日常生活中的“笔译”，该方式通过特定的工具——“编译器”将高级语言源程序转换为用机器语言表示的目标程序，然后再执行。一般而言，解释方式占用内存少，易于查错和移植程序，但执行速度稍慢；而编译方式执行速度较快，但每次修改源程序后必须重新进行编译处理。C 语言是编译型高级程序设计语言的典型代表。

想一想：

为了实现“将变量 a 的值加上 5 之后再存储回去”，分别用机器语言、汇编语言和高级语言编制的程序如下：

机器语言	汇编语言	高级语言
00100100　01000000	LOAD a	a=a+5;
00000010　00000101	ADD 5	
00110000　01000000	STORE a	

从上面的程序片段来看，三种编程语言编制的程序各有什么优缺点？高级语言究竟“高级”在哪里？

1.1.2 C 语言的历史

1. C 语言的起源

20 世纪 60 年代，美国贝尔实验室的研究人员肯·汤普逊(Ken Thompson)在 BCPL(Basic

Combined Programming Language）语言的基础上设计出一种小型的 B 语言，并用于开发 UNIX 操作系统。1972 年，同在贝尔实验室的丹尼斯·里奇（Dennis M. Ritchie）在 B 语言的基础上进一步精炼，设计出一种高效的语言，并取 BCPL 的第二个字母，将其命名为 C 语言。

C 语言一经推出，其强大的功能和高效灵活的特点立即引起人们的广泛关注。C 语言的编译器被广泛移植到当时流行的 IBM 360/370、Honeywell 11 以及 VAX-11/780 等多种计算机平台上，C 语言迅速成为首选的编程语言之一。

2. C 语言的标准化

在 C 语言发展的最初几年，并没有相应的语言标准，直到 1978 年，布莱恩·柯林汉（Brain Kernighan）和丹尼斯·里奇共同出版经典著作《The C Programming Language》，才标志着 C 语言开始有了规范版本。这个版本是 C 语言第一个事实上的标准，通常称之为 K&R C 或经典 C。鉴于这部经典著作在 C 语言发展过程中的重要影响，人们通常称之为 C 语言的“圣经”。

随着 C 语言的不断发展和迅速普及，C 语言需要一个更加全面、准确和严格的标准成为了人们的普遍共识。鉴于此，美国国家标准协会（ANSI）于 1983 年组建了一个专门委员会负责制定本国的 C 语言标准。1989 年，ANSI 发布了第一个完整的 C 语言标准，简称 ANSI C 或 C89。该标准对 C 语言进行了完整定义和准确描述，并给出了 C 语言的标准库。C89 标准的制定有力地推动了 C 语言的进一步发展，C 语言和 C 语言的编译器被越来越广泛地应用到各种系统中。1990 年，国际标准化组织（ISO）和国际电工委员会（IEC）审核通过了 C89 标准，将其作为 C 语言的国际标准，并命名为 ISO/IEC 9899：1990，简称 C90 标准。事实上，C90 并未对 C89 做任何更改和修订。也就是说，C90 和 C89 是完全相同的标准。

1999 年，为使 C 语言更好地支持国际化编程，解决和弥补明显错误，并进一步提高其计算实用性，ISO/IEC 对 C 语言标准进行了修订，发布了 C99 标准。C99 的修订保留了 C 语言精练、实用的精髓，仅对一些必要的地方进行了修正和完善，并补充了一些新的特性。之后，ISO/IEC 于 2011 年进一步发布了 C11 标准，着重在编程安全和语言可选项支持等方面进行了修订和完善，这也是目前最新的 C 语言国际标准。不过，需要说明的是，由于需要维护难以计数的旧版本 C 代码，加之各种应用平台的多样性和复杂性，C99 和 C11 标准的一些新特性尚未得到广泛应用。

3. 基于 C 的语言

C 语言作为一种流行的高级语言，对现代编程语言的发展产生了重要影响。许多主流的现代编程语言都是以 C 语言为基础发展起来的，其中借鉴了大量 C 语言的优秀特性。下面选择几种有代表性的语言简要介绍。

✧ C++：保留了所有的 C 特性，增加了类和对象等特性以支持面向对象编程。

✧ Java：以 C++为基础，摒弃内存管理、多重继承等复杂机制的面向对象编程语言。

✧ C#：　基于 C++和 Java 发展起来的一种现代编程语言。

✧ Perl：最初是一种非常简单的脚本语言，在发展过程中采用了许多 C 语言的特性。

或许有读者会有这样的疑问：“那 C 语言还值得学习吗？”从上面的介绍不难发现，当今的许多主流程序设计语言大多承袭了 C 语言许多的优良特性，学习 C 语言有助于更好地理解和掌握这些基于 C 的语言的特性，有助于掌握具体一个功能模块的设计与编写方法。此外，凭借自身的简洁高效和广泛的硬件平台支持等优势，C 语言在一些内存或处理能力受限的小型控制系统和嵌入式开发中仍然具有无可比拟的优越性。

1.1.3 C 语言的特点

C 语言之所以能够从众多的计算机语言中脱颖而出，主要归功于其优良的设计特性、高效和快速性、功能强大且兼顾灵活性、良好的可移植性等特点。

1. 结构化和模块化编程

C 语言是一种结构化程序设计语言，采用面向过程的程序设计(Process Oriented Programming，POP)方法。这种方法以程序的可读性和清晰性为目标，采用结构化和模块化设计思想，易于维护和重用代码，易于通过自顶向下和逐步求精的方案解决各种复杂问题。采用 C 语言编制的程序，不仅层次清晰，逻辑性强，而且便于阅读和调试。同时，C 语言采用“函数”方式进行模块化设计，这不仅使编写的程序更可靠、易懂、易于维护，而且增强了代码重用的可行性，为编写复杂程序奠定了坚实的基础。

2. 高效和快速性

C 语言是在实际编程过程中开发出来的一种实用语言，具有高效、快速的特点。用 C 语言编写的程序代码十分紧凑，执行速度较快。C 语言程序甚至可以达到汇编语言的执行效率和最大内存利用率。

3. 功能强大且兼顾灵活

C 语言把高级语言的基本结构和低级语言的实用性紧密结合起来。这使得 C 语言不仅适于编写各种应用软件，还适于编写类似 UNIX 这样的系统软件，以及诸如 Fortran、Perl、Python、Pascal、LISP 和 Basic 等多种其他高级语言的编译器或解释器，实现对硬件的直接访问和控制。同时，C 语言具有丰富的运算符，使得编程者在处理许多任务时表达方式极其简洁、灵活和方便。

4. 可移植性

C 语言是一种与硬件较少关联的高级语言，且 C 编译器在不同计算机系统上广泛使用，这使得用 C 语言编写的程序无须修改或经少量修改便可在各种不同的计算机上运行。而且，C 语言发展多年，应用平台和辅助工具不计其数，这也为 C 程序的移植性提供了便利。当然，针对特定硬件设备或操作系统特殊功能编写的代码，通常是不可移植的。

C 语言的特点并不仅仅局限于上述几个方面，诸如丰富的 C 标准库、与 UNIX 系统无缝集成等众多优势使之仍然活跃在当今的程序设计领域。

1.2 算法及其表示

在日常生活中，要做出一道美味的菜肴，需要特定的方法和步骤。做事、解题也要有清晰的思路和明确的过程，程序设计也要遵循一定的步骤。

1.2.1 算法的概念

算法(Algorithm)是指为解决某个具体问题而采取的方法和步骤的集合。当我们编制程序解决一个实际问题时，一个好的算法有助于问题的快速解决，而不好的算法往往耗费较多的

时间和计算资源，错误的算法则根本无法得到正确的结果。

C 语言是一种结构化的编程语言，采用面向过程的程序设计方法。正如 Pascal 之父、结构化程序设计的先驱尼古拉斯·沃斯(Niklaus Wirth)指出：程序=算法+数据结构，算法是程序设计的灵魂，程序设计的关键在于算法。也就是说，编制程序首先要采用合适的数据类型存储数据，然后采用好的方法和步骤对数据进行加工处理，才能得到预期的结果。

1.2.2　算法的特征

使用计算机程序求解问题，要有一个明确的起点，确定的步骤序列，程序终止执行时要能给出最后的结果。因此，一个算法应具备如下 5 个重要特征。

(1) 输入：0 或多个输入，其中 0 输入表示算法本身已给出初始条件。

(2) 输出：1 或多个输出，没有输出的算法是毫无意义的。

(3) 有穷性：算法的执行步骤是有限的，且每一步骤的执行时间是可容忍的。

(4) 确定性：算法的每一步骤均具有确切的含义，不允许出现歧义。

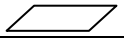
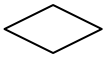
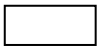
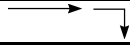
(5) 可行性：算法的每一步骤都可以通过已经实现的基本运算执行有限次数来实现。

1.2.3　算法的表示

在程序设计过程中，有必要将算法的主要步骤清晰地记录和表示出来，这不仅有利于编程者之间交流算法的主要思路，而且有利于算法的后期改进和优化。常见的算法表示方法有伪代码、流程图、N-S 图和 PAD 图等，本书以流程图为例介绍算法的具体表示方法。

流程图是一种采用程序框、流程线及简要文字说明来表示算法的有效方法。其中，程序框图用于表示算法中的具体步骤，流程线表示算法的执行顺序。流程图中常用的符号如表 1-1 所示。

表 1-1　流程图中常用的符号

符号	名称	功能
	起止框	表示算法的起始和结束，有时为了简化流程图也可省略
	输入/输出框	表示算法的输入和输出的信息
	判断框	判断条件是否成立，成立时在出口处标明“是”或“Y”，不成立时标明“否”或“N”
	处理框	赋值、计算。算法中处理数据需要的算式、公式等分别写在不同的用以处理数据的处理框内
	流程线	连接程序框，带有控制方向

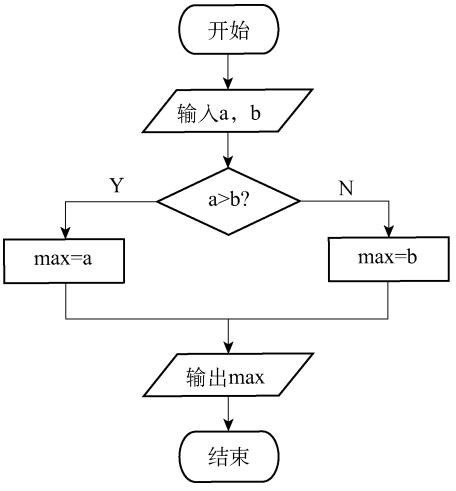


图 1-1　输出两个数中较大者的算法流程图

例如，对于键盘输入的两个数，比较后输出其中较大的数，其算法流程图如图 1-1 所示。在实际程序设计过程中，可以事先对问题进行具体分析，然后采用诸如 Raptor 可视化快速算法设计工具进行算法和流程图设计，这样有助于编制逻辑清晰、结构合理的程序。

1.3 C 程序的操作过程

C 语言是一种编译型语言，从创建 C 程序到得到最终结果，一般要经历编辑、编译、链接和运行 4 个主要阶段，如图 1-2 所示。无论哪个环节出现问题，都必须返回到前面的过程，修改错误后，再重新执行这几个过程。下面以 Windows 平台下的 Microsoft Visual C++ 2010 为例，具体介绍每个步骤的含义和操作方法。

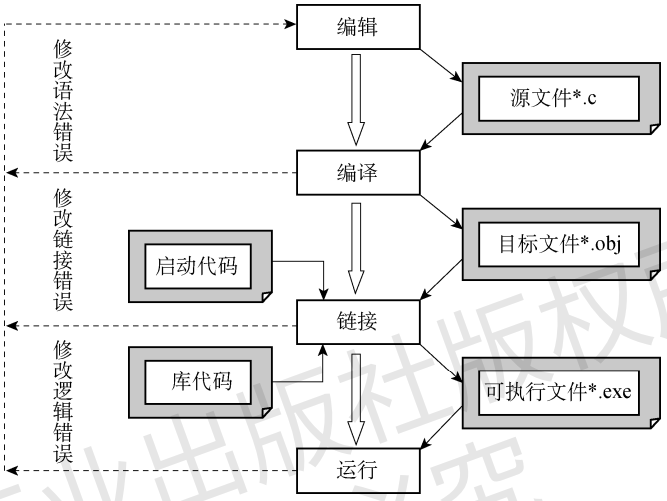


图 1-2 C 程序的基本操作过程

1.3.1 编辑

实际上，在正式编写程序代码之前，一般还需要对程序所要实现的功能进行分析，绘出算法流程图，这对于复杂程序的设计尤为重要。编辑 C 程序的过程，是指将程序算法用 C 语言进行实现，在编辑器中输入程序代码，并将其保存为扩展名为.c 的 C 源程序。这个过程可采用文本编辑器或专用的 C 编辑器完成。当然，以 VC++ 2010 为代表的集成开发环境（Integrated Development Environment，IDE）提供了更大的方便，可实现图 1-2 所示的所有 C 操作过程，且可方便地对程序进行调试和测试。

说明：

VC++ 2010 采用“项目”对程序进行管理。一个项目可以只包含一个.c 文件，也可包含多个.c 文件。因此，一般情况下，需先新建一个项目，然后再在项目中添加 C 源程序。

（1）新建项目：文件→新建→项目→Win32 控制台应用程序→输入项目名称和保存位置→在“应用程序设置”窗口中选择“空项目”并取消“预编译头”选项。

（2）向项目中添加 C 文件：项目→添加新项→C++文件→输入文件名（含.c 扩展名）并选择“添加”按钮，在打开的编辑窗口输入程序代码即可。

1.3.2 编译

编译就是指调用 C 编译器将编写好的 C 源程序（*.c）翻译为本机目标代码文件（*.obj）的过

程。在此过程中，C 编译系统会首先自动调用预处理器对 C 源程序进行一定的处理，这些处理通常包括把其他文件“包含”到要编译的文件中，或者使用文本替换专门的符号。在编译阶段，如果所编写的 C 源程序没有任何错误，则将生成扩展名为.obj 的目标文件，该文件是用本机机器语言表示的代码，但还只是一个中间代码，还不能直接运行。如果编译时发生错误，则不会生成目标文件，此时编译器会给出错误提示信息，必须根据这些信息对源程序的语法错误进行修改后，再重新进行编译，直到程序成功通过编译为止。

说明：

在 VC++ 2010 环境中，使用“生成”菜单下的“生成解决方案”、“生成××”项目、“编译”等命令均可完成编译处理，也可采用“生成”工具栏完成相应的操作，或在“解决方案资源管理器”中的相应 C 文件上右击，通过快捷菜单中的“编译”命令实现单个文件的编译处理。

1.3.3　链接

链接是将目标代码文件(*.obj)和所需的其他附加代码(如启动代码和标准库函数)链接在一起，以形成可执行文件(*.exe)的过程。该过程由链接器负责完成。实际上，编译生成的目标代码文件已用机器语言表示，但它还不能直接运行，尚需和系统启动代码以及库代码进行链接，才能最终生成可直接执行的完整程序。其中，启动代码是程序和操作系统之间的接口，而库代码是程序中调用到的标准库函数的目标代码。一般来说，C 程序总会引用到标准库函数或其他地方定义的函数。标准库函数是 C 语言提供的一个大型函数库，包含许多有用的函数模块，这些函数能够满足编程人员大多数的需要，且其编写是严格和高效的，而其他地方的函数主要指编程者自己或他人创建的函数，以满足一些特殊应用要求。在链接阶段，如果程序被正常链接，则系统将会生成最终的可执行文件；如果链接失败，可能是因为找不到所需的库函数，此时需重新修改程序，然后再重新进行编译和链接。

可见，目标文件和可执行文件虽然都是由机器语言指令组成的，但目标文件只包含编译器将源文件翻译后的机器语言代码，而可执行文件还包含程序中使用到的库函数和启动代码的机器代码。

说明：

在 VC++ 2010 环境中，没有单独的“链接”命令，编译器会自动调用链接器，完成目标文件、启动代码以及库代码的合并。

1.3.4　运行

运行是将链接得到的可执行文件(*.exe)加载入内存，由 CPU 进行执行处理，以测试程序运行结果是否正确的过程。在此阶段，如果程序编写正确无误，算法设计恰当、合理，则可得到预期的结果。相反，程序虽然能够运行，但执行结果却不正确，这多数是设计逻辑有误造成的。或者说，是因为算法设计有问题所导致的。这时，必须重新审视所设计的算法，分

析程序执行逻辑，以便发现导致错误的原因，进而修改源程序中的运行逻辑错误，然后再重新进行编译、链接和运行的整个过程。

说明：

在 VC++ 2010 环境中，可采用“调试”菜单下的“开始执行(不调试)”命令或“调试”工具栏上的相应按钮运行程序。同时，可以利用调试工具栏上的按钮对程序进行调试运行，以检查程序是否含有逻辑错误。

1.4 C 程序的基本结构

通过前面的学习，读者已了解 C 程序的一般操作过程。本节通过三个基本的实例带你轻松进入 C 语言的编程世界，并通过它们了解 C 程序的基本结构，熟悉 C 程序的编程方法和过程。

1.4.1 第一个基本的 C 程序

【例 1.1】创建第 1 个基本的 C 程序，在屏幕上输出两行文本。

问题解析：首先，在 VC++ 2010 中新建一个名为 test1 的项目，然后在项目中添加新项 first.c，之后便可在打开的编辑器窗口中输入程序。程序输入完毕并保存后，进行编译(含链接)处理，最后执行可得到运行结果。

程序代码如下：

```
/* 我的第一个 C 语言程序
* Date: 2017-09-01
*/
#include <stdio.h>
void main()
{
     printf("Welcome to C!\n");  //调用标准库函数 printf( )进行屏幕输出并换行
     printf("This is my first C program.");  //输出但不换行
}
```

程序运行结果如下：

```
Welcome to C!
This is my first C program.
```

例 1.1 解析

说明：

(1) C 程序由“函数”组成，每个函数就是一个功能模块，函数是 C 程序的基本组成部分。在本例中，只定义了一个函数 main()，并在其中调用标准库函数 printf()在屏幕上输出信息。

(2) main 函数又称为主函数，它是 C 程序执行的起点，任何 C 程序有且仅有一个 main 函数。main 之后的“()”表示这是一个函数且不带参数，也即运行程序时不需要给 main 函数提供任何值。main 之前的 void 其含义为“空，没有”，表示调用该函数不会向调用者返回

任何值(此时的调用者为计算机系统自身)。这一行“void main()”统称为函数头。

(3)任何函数都以左花括号“{”开始，以右花括号“}”结束，两者之间的部分称为函数体，它一般由一组 C 语句序列组成，且每条 C 语句都以分号“;”结束。

(4)printf()函数的作用是在显示器上输出双引号里的内容。第一次调用 printf 函数的输出结果为 Welcome to C!，而不是 Welcome to C!\n，这是因为\n 为转义字符，具有特殊的含义，表示换行。也就是说，当输出时遇到\n，之后的内容将被输出到下一行上。另外，由于 printf 函数被定义在 stdio.h 文件中，为了让系统知道 printf 到底是什么，必须在文件开头部分使用文件包含命令#include<stdio.h>。其中，“#”符号表明，C 预处理器在编译器接手之前先处理这条命令。这样在编译时，编译器就会预先将 stdio.h 文件里的内容包含到 firtst.c 中，然后再进行编译。stdio.h 称为标准输入/输出头文件，它提供了键盘输入和屏幕输出的支持。在 C 语言中，类似的扩展名为.h 的文件被称为头文件，常用于组织 C 标准函数库中的函数。

(5)程序最前面的/*……*/之间的内容称为程序注释，主要用于增强程序的可读性。在编译时，编译器会忽略这些注释语句，但养成良好的注释习惯有助于程序使用者阅读和理解程序。C 语言的标准注释方法以“/*”开始，直到遇到“*/”结束，适用于单行和多行注释。在 C99 标准中，也可以采用“//”进行单行注释，如 printf 函数调用语句的注释。

由本例可见，C 程序一般由一个或多个函数组成，函数是 C 程序的基本模块，每个函数由函数头和函数体组成。函数头包括函数名、传入该函数的信息类型和函数的返回类型。函数体被花括号括起来，由一系列语句组成。简而言之，一个可运行的基本 C 程序结构如下：

```
#include <stdio.h>
void main( )
{
   语句(以分号结束)
 }
```

想一想：

要输出以下的结果，该示例程序应该如何修改？

```
******************
*  Welcome to C! *
******************
```

1.4.2　含有变量的 C 程序

很少有程序会像例 1.1 那样简单，大多数程序在产生输出之前往往需要执行一系列的计算，此时就需要在程序执行过程中有一种临时存储数据的方法。C 语言采用“变量”来存储数据。

【例 1.2】给定长方体的长、宽和高，计算长方体的体积。

问题解析：长方体的体积等于长、宽和高的乘积，因此可设计 4 个实型变量，其中 3 个用于保存长、宽和高的数值，另一个用于保存这三者的乘积，最后通过调用 printf()函数输出

计算结果。

程序代码如下：

```
/* 计算长方体的体积  */
#include <stdio.h>
void main()
{
    double length, width, height, volume;   //声明 4 个双精度实型变量
    length=5.5; width=3.0; height=1.2;      //三条赋值语句为变量赋值
    volume=length*width*height;             //C 语言中的乘号用*表示
    printf("Dimensions:%f*%f*%f\n",length,width,height);    //输出长宽高
    printf("The volume of the cuboid is: %.2f\n",volume);   //输出长方体的体积
}
```

程序运行结果如下：

```
Dimensions:5.500000*3.000000*1.200000
The volume of the cuboid is: 19.80
```

例 1.2　解析

说明：

（1）main 函数的第 1 条语句为变量声明语句，定义了 length, width, height, volume 四个双精度实型(double)变量。变量是用来存储数据的内存空间，声明变量的含义是指通知 C 编译系统为所定义的变量分配合适大小的存储空间。实际上，声明变量意味着把特定的标识符(即变量的名称，如这里的 length, width, height, volume)与计算机内存中特定的位置联系起来，同时也确定了存储在该位置的信息类型或数据类型。每个变量的存储空间大小主要由变量的类型和机器平台决定。变量的类型可以是字符型(char)、整型(int)、单精度实型(float)和双精度实型(double)等，它们在 32 位机器的存储空间大小通常为 1、4、4 和 8 个字节。

（2）系统为变量分配相应的存储空间时，变量的值一般为随机数(和此时分配到的具体存储单元有关)。之后，便可通过赋值语句对变量的值进行修改。例如，在本例中，通过赋值操作给 length 变量赋值 5.5，给 width 变量赋值 3.0，给 height 变量赋值 1.2，并使用它们的乘积为 volume 变量赋值。

（3）C 语言是一种格式自由的语言。同一行上可以写一条或多条语句，一条语句也可以写在多行上。但为了程序的清晰性和可读性，建议每行就写一条 C 语句。

（4）本例中的 printf()函数比例 1.1 复杂。实际上，printf 函数称为格式化输出函数，其中的%f 和%.2f 均为格式占位符，其作用是指明输出项的位置和形式。第 1 条 printf 函数调用语句中的 3 个%f 表示要输出 3 个实型值，实际输出时会分别用 length, width, height 的值进行替代。第 2 条 printf 调用语句中的%.2f，同样表示要输出 1 个实型量，但限定小数点后的位数为 2 位。对于双引号内的其他内容，除转义字符外，其余字符将原样输出。

想一想：

对于不同的数据类型，格式化输出函数 printf()采用不同的格式占位符。例如，对于整型

数据(数据类型为 int)，输出时采用%d。若本例长方体的长、宽、高均为整型，程序应如何修改？

1.4.3　包含多个函数的 C 程序

函数是 C 程序的基本模块，例 1.1 和例 1.2 都只定义了 main()函数，并在其中调用了标准库函数 printf()实现程序的输出。下面的程序示例演示了如何将自己编写的函数加入程序中的方法，这是更常见的 C 程序结构。

【例 1.3】编写一个函数实现输出两个整型数中较大者的功能，并通过 main 函数进行测试。

问题解析：例 1.2 中变量的值是在编程时通过赋值语句给定的，这导致程序的灵活性较差。通过系统提供的 scanf()函数，可实现程序运行时动态地从键盘输入数据。另外，1.2.3 节已给出判断两个数中较大者的算法流程，为了程序的清晰性和代码复用，可根据该算法流程编制一个单独的功能模块函数 findMax()，然后在 main()函数中进行调用。

程序代码如下：

```
#include <stdio.h>
int findMax(int a, int b);      //函数原型声明
void main( )
{
    int x,y,t;
    printf("Please input two integers (x, y): ");   //输入操作提示
    scanf("%d,%d", &x,&y);      //从键盘输入数据
    t= findMax(x,y);            //调用 findMax 函数寻找最大的数
    printf("The maximum is: %d\n",t);
}
int findMax(int a, int b)       //定义寻找两个数中较大者的函数
{
    int max;
    if(a>b)                     //该 if 语句得到较大值，存放在 max 变量中
         max=a;
    else
         max=b;
    return max;                 //通过 return 语句返回 max
}
```

程序运行结果如下：

```
Please input two integers (x, y): 18,30(运行时由用户输入)
The maximum is: 30
```

说明：

例 1.3　解析

(1) 本例充分体现了 C 语言的模块化编程理念，将寻找两个数中较大者的功能定义为 findMax 函数，供 main 函数调用。main 函数是 C 程序的运行起点，无论 findMax 函数的定义放在 main 函数之前或之后，程序总是从 main 函数的第一条语句开始执行。

(2) scanf()函数可实现程序运行时从键盘输入数据，大大提高了程序的灵活性。输入数据时，一定要按照 scanf 双引号中的格式进行输入。对于本例来说，双引号中指定的格式为"%d，%d"，表明需要输入两个整数(每个%d 代表一个整数)，且中间以逗号分隔。scanf 函数

中&x，&y 称为输入地址列表，表示要将输入的两个整数分别存放在变量 x 和 y 的相应地址空间中。其中，&符号为取地址运算符。

程序在执行到该 scanf 语句时，将等待用户输入，用户在输入 18，30 后，按下回车键便可结束输入。

在 scanf 语句之前，经常配套使用一条 printf()函数语句进行输入操作提示说明，如本例中的 printf("Please input two integers(x，y)：")。这是一种习惯用法，其目的是让用户知道此时程序正等待用户输入两个整型数据。

(3) findMax()函数在程序中共出现了 3 次。第 1 次是程序的第 2 行，称为函数原型声明，其作用是告知编译器在程序中要使用 findMax 函数，且该函数将返回一个整数(int)，调用时需要提供的参数为两个整型量；第 2 次出现在 main 函数的“t= findMax(x，y)”语句中，这是函数调用语句，需给出函数名和具体的参数。函数调用结束，将函数返回值赋值给变量 t，程序转向 main 函数继续执行；第 3 次是 findMax 函数的定义，其形式和 main 函数相同，都包含函数头和函数体两部分，函数头重述了函数原型信息，函数体实现具体的功能。在本例的 findMax 函数定义中，通过 if 语句比较出 a 和 b 中的较大者，并将其暂时存放在变量 max 中，最后通过 return 语句把 max 的值传回并赋值给 main 函数中的变量 t。

(4) 要提醒读者的是，何时执行 findMax 函数取决于它在 main 函数中被调用的位置，而不是该函数定义在程序中的位置。无论 main 函数在程序中处于什么位置，所有的 C 程序都是从 main 函数开始执行。一般地，人们习惯把 main 函数放在开头，因为它提供了程序的基本框架。需要说明的是，当 findMax 函数在 main 函数之前定义时，许多编译器可以省略函数原型声明语句，但保持本示例程序的结构更有利于程序清晰性。

想一想：

本例中将 findMax 函数单独进行定义有什么好处？如何理解“函数是 C 程序的基本模块”这句话的含义？

1.5 标识符、关键字和保留字

在例 1.2 和例 1.3 中，需要给变量和函数等进行命名。实际上，C 语言中的很多对象都需要命名，以便在程序中使用。所谓标识符，是指变量名、函数名、数组名、符号常量名等用户自定义的名称。

C 语言规定，标识符的命名必须遵循如下规则：

- ✧ 只能由英文字母、数字和下划线(_)组成，且必须以字母或下划线开头；
- ✧ 严格区分大小写，sum 和 Sum 是不同的标识符；
- ✧ 不允许使用 C 关键字和保留字为标识符命名。

什么是关键字呢？关键字是 C 语言的词汇，是系统中预先规定的、具有特殊含义的名字，

不能再作为标识符使用。C 语言的关键字如表 1-2 所示。其中，粗体表示 C90 标准新增的关键字(5 个)，斜体表示 C99 标准新增的关键字(5 个)。

表 1-2　C 语言中的关键字

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

此外，还有一些保留标识符，C 语言已经指定了它们的用途或保留它们的使用权，用户使用它们虽不至于引起语法错误，但可能引起名称冲突或导致其他问题。例如，以下划线开头的标识符通常被操作系统用作系统变量和系统函数，用户应尽量避免采用此种形式。再如，一些标准库函数名称，如 scanf 和 printf 等，用户也不该再用做其他对象的名称，以免造成冲突。

根据上述规则，以下标识符均为合法的 C 标识符：

```
i, j, a3, abc_cba, findMax, _flag, sum;
```

但以下的标识符均为不合法的标识符：

```
33abc       //以数字开头
your name   //含有空格
aa$bb       //含有特殊字符
inline      //和关键字同名
main        //和 C 主函数同名
```

同时，要提醒读者注意的是，为了提高程序的可读性，标识符应尽量做到“见名知意”，不要一味地只图书写简单。例如，要存储姓名、年龄和成绩等数值，变量名 name、age 和 score 就比 aa、bb 和 cc 好，尽管它们都属于合法的标识符。

1.6　常量、变量和数据类型

程序离不开数据。程序的本质是对初始信息(数据)进行加工处理，然后得到所需结果的过程。数据在 C 程序中具有不同的表现形式和存储方法，这就涉及常量、变量和数据类型的概念。

1.6.1　常量

常量是指在程序运行过程中值不会改变的量。根据数据类型不同，常量可以分为多种不同的类型。如，整型常量 10，0，−25 等；实型常量 5.15，−0.23 等；字符型常量'A'，'3'，'\n'等；字符串常量"x"，"Welcome to C！"等。每种类型常量的表现形式都和具体的数据类型有

关，将在 1.6.3 节中结合数据类型进行详细介绍。

C 编译系统不会为常量分配存储空间，常量通常用于给变量进行赋值。

此外，C 语言还有一种用标识符代表的常量，称为符号常量。顾名思义，符号常量就是用一个符号(标识符)代替一个具体的常量值，以说明该常量值的实际意义，进而增强程序的可读性，并方便程序的维护。下面的程序示例演示了符号常量的具体用法。

【例 1.4】输入圆的半径，计算圆的周长和面积。

问题解析：圆的周长和面积都涉及圆周率 π，常取为 3.14，因此可利用符号 PI 代替该常量值，以增强程序的可读性，并在需提高计算精度时，方便程序的维护和修改。

程序代码如下：

```
/* 计算圆的周长和面积 */
#include <stdio.h>
#define PI 3.14
void main( )
{
    double r,circum,area;          //定义 3 个双精度实型变量用于保存半径、周长和面积
    printf("请输入半径：");
    scanf("%lf",&r);               //通过键盘输入半径的值
    circum=2*PI*r;                 //计算圆的周长
    area=PI*r*r;                   //计算圆的面积
    printf("r=%f,circum=%f, area=%f\n",r,circum,area);      //输出计算结果
}
```

程序运行结果如下：

```
请输入半径：1
r=1.000000,circum=6.280000, area=3.140000
```

例 1.4　解析

说明：

(1)“#define PI 3.14”定义了符号常量 PI，和文件包含命令一样，它也属于编译预处理命令，放在文件开始部分。当编译程序时，C 编译系统首先扫描源文件，查找符号“PI”，然后用数值“3.14”进行替换，之后才正式对程序进行编译处理。为了和变量进行区别，符号常量一般采用大写标识符表示。

(2) 若需提高计算精度，可直接修改符号常量的定义。如，将第 2 行直接修改为“#define PI 3.1415”，而不用修改 main 函数中的任何语句，这大大提高了程序后期维护的方便性。

(3) 在本例中，半径 r 的值通过 scanf 函数获取。要注意的是，输入双精度实型(double)数据时，采用%lf 格式符。

1.6.2　变量

变量是指在程序执行过程中值可以发生变化的量。变量本质上是用来存储数据的内存空间。C 语言规定，变量在使用之前必须先进行声明，并为其提供值，之后才能使用。

1. 声明变量

在 C 语言中，声明变量的一般形式为：

```
数据类型 变量名;
```

其中，变量名即指变量的名称，遵守 C 语言标识符的命名规则。数据类型决定了 C 编译器为变量分配的内存空间所能存储的数据种类。可见，声明变量把一个具体的标识符名称和计算机内存中的一个特殊的位置联系起来，同时确定了该位置存储的信息类型。

例如，可以采用如下的形式声明变量 height 和 radium：

```
int height;
float radium;
```

第一条声明语句说明 height 是一个整型(int)变量，第二条声明语句则表示 radium 是一个单精度实型(float)变量。注意，每一条完整的声明语句都以分号结束。

如果几个变量具有相同的类型，可以单独进行声明，也可以一起进行声明，此时在变量名之间用逗号进行分隔，例如：

```
int length, width, height;
double r, area;
```

另外，许多 C 编译器规定所有变量必须在第一条可执行语句之前声明，否则将导致编译错误，参看下面的示例程序。

【例 1.5】演示变量在第一条可执行语句之后声明时将导致编译错误。

程序代码如下：

```
#include <stdio.h>
void main()
{
   int x;
   x=1;
   float y;
   y=2.8f;  //float 类型常量最后面有字母 f
   printf("x=%d,y=%f\n",x,y);
}
```

程序运行结果如下：

```
多数编译器下无法通过编译。
```

例 1.5　解析

说明：

(1) C99 之前的标准规定变量声明部分必须放在所有可执行语句的前面，这样的规定似乎较为呆板，但其实有利于理解程序的用途，并提前对程序做出合理的计划，使程序更加清晰、可读。

(2) C99 标准采用 C++的惯例，允许把变量申明放在代码块的任何位置。但为了更好地和旧式 C 系统兼容，建议最好遵守初始的约定。

想一想：

如何修改本示例程序，才能使之正常通过编译，得到正确的输出结果?

2. 变量初始化

在前面的程序示例中，已介绍过两种为变量提供值的方法：一是声明变量后，使用赋值

运算符“=”为变量赋值；二是通过标准库函数 scanf()获得值。其实，在声明变量时，也可以同时为变量赋一个初始值，这种方法称为变量初始化。例如：

```
int a=3;
float area=0;
double length=5.5, width=3.0, height=1.2, volume;
```

上述示例中的最后一行，只初始化了前 3 个变量，而未对 volume 变量进行初始化。若在同一条语句中声明多个同类型变量且它们的初始值相同时，可采用以下的形式：

```
int x=1, y=1, z=1;
```

但如下写法将导致编译时语法错误：

```
int x=y=z=1;  //不能写成连等形式，因为此时编译器不知道有 y 和 z 变量
```

需要提醒大家的是，若变量定义时没有被初始化，那么该变量的值一般是一个随机数(后续章节介绍的静态变量和全局变量除外)，直到有赋值语句或键盘输入语句对该变量的值进行修改为止。因此，为了避免变量的意外取值导致的计算错误，能够初始化的变量最好在声明时就进行初始化。

1.6.3 数据类型

变量的存储方式取决于采用何种数据类型。C 编译器通过用户的书写形式将常量识别为不同的类型。不同的数据类型在内存中所占的存储单元大小不同，内部存储方式不同，取值范围不同，甚至能够参与运算的种类也不相同。

C 语言提供了丰富的数据类型，如图 1-3 所示。本小节将介绍其中的基本类型，有关其他数据类型详见后续章节。

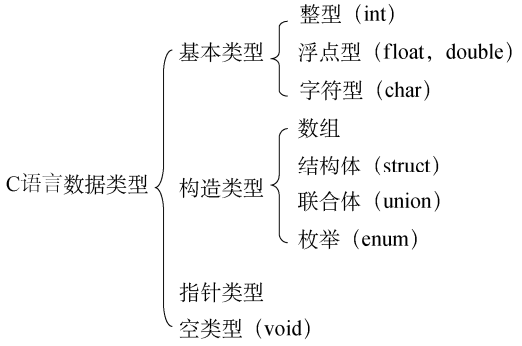


图 1-3 C 语言的数据类型

1. 整型

整型数据即指没有小数点的量，也就是人们通常所说的整数。在 C 语言中，基本整数类型采用 int 关键字表示，默认为有符号的整数。而关键字 short、long、unsigned 和 signed(通常不必使用)则用于提供基本整型的变式。在 32 位平台下，各种整数类型的典型基本特性如表 1-3 所示。

表 1-3　C 语言整数类型的典型基本特性

类型	变量声明实例	字节数	取值范围	常量形式
int	int x；	4	-2^{31}～$2^{31}-1$	十进制：30，−125，+5 八进制：017，−015 十六进制：0x1A，−0X28
short [int]	short x；或 short int x；	2	-2^{15}～$2^{15}-1$	同上，注意取值范围
long [int]	long x；或 long int x；	4	-2^{31}～$2^{31}-1$	285l，−1024L
unsigned int	unsigned int x；	4	0～$2^{32}-1$	25u，255U
unsigned short [int]	unsigned short x；	2	0～$2^{16}-1$	同上，注意取值范围
unsigned long [int]	unsigned long x；	4	0～$2^{32}-1$	25lu，255UL

说明：

(1) 声明 int、short、long 整型变量时，若没有明确指出其为 unsigned 类型，则隐含为 signed 类型，也即 signed 关键字通常可省略不写，变量默认是有符号的。其实，有符号和无符号整数的区别仅在于对数据最高位的解释不同。若最高位解释为数据位，则为 unsigned 类型；若最高位解释为符号位，则为 signed 类型。

(2) 各种类型变量所占字节数和取值范围通常和所使用的平台有关。ISO C 只规定 int 的取值范围最小为−32768～32767(即：-2^{15}～$2^{15}-1$)，且 int 类型不能比 short 类型短，long 类型不能比 int 类型短。

(3) 整型常量可以采用多种进制书写，一般采用十进制即可。八进制(以 0 开头)和十六进制(以 0x 开头)更适合用于底层程序的编写。十进制整型常量通常为 int 型，要明确表示长整数，需在后面加上字母 L(或 l)；要明确表示无符号常量，在末尾加字母 U(或 u)。

(4) 若程序调用了 scanf 和 printf 函数，要特别注意相应的格式占位符，%d 用于 int 类型，%hd 用于 short 类型，%ld 用于 long 类型，%u 用于 unsigned int 类型，详细介绍可参见第 2 章。

(5) 对整数执行算术运算时，需特别注意变量的存储空间是否能够容纳下要存入的数，以避免发生“数据溢出”现象。例如，下面的代码将无法获得正确的结果：

```
short x=200, y=300, result;
result=x*y;
printf("%d\n",result);
```

2. 浮点型

浮点型又称为实型，是指带有小数点的数，即人们常说的实数。由于实数在计算机内部采用浮点形式(小数点位置不固定)进行存储，因而称为浮点型。C 语言中的浮点类型包括单精度浮点型(float)、双精度浮点型(double)和长双精度浮点型(long double)三种。在 32 位平台下，各种浮点型的典型基本特性如表 1-4 所示。

表 1-4　C 语言浮点类型的典型基本特性

类型	变量声明实例	字节数	取值范围	有效位数	常量形式
float	float a；	4	-3.4×10^{-38} ～3.4×10^{38}	6 或 7	小数形式：3.14f， −12.0F，0.13F，15f 指数形式：1.45e+3f， −2.2e−2F，25e5f

(续表)

类型	变量声明实例	字节数	取值范围	有效位数	常量形式
double	double a;	8	-1.7×10^{-308} ～1.7×10^{308}	15 或 16	小数形式：3.14，−12.0，0.13，15 指数形式：1.45e+3，−2.2e-2，25e5
long double	long double a;	8	-1.7×10^{-308} ～1.7×10^{308}	15 或 16	小数形式：3.85l 指数形式：−2.34e−3L

说明：

(1) 和整型一样，浮点型数据所占用的字节数、取值范围和有效位数取决于机器平台。C 标准仅规定，float 类型必须至少能表示 6 位有效数字，而没有说明三种浮点类型能提供的精度到底是多少，因为不同的计算机可以采用不同的方法存储浮点数。

(2) 实型常量可采用小数和指数两种形式进行表示。在指数表示形式中，要求 e(或 E)之前必须有数，之后的指数必须为整数。也就是说，e3，2.2E-2.5，e2 都是错误的表示方法。实型常量必须包含小数点(或为指数)。因此，1.0 是 double 型常量，而 1 是 int 型常量。默认形式的实型常量，如 3.14 为 double 型，若要表示 float 型和 long double 型常量，可分别加上后缀 F(或 f)和 L(或 l)，例如：3.14f，100.5L 等。

(3) 在 scanf 和 printf 函数的格式说明中，%f、%e 和%g 用于 float 类型；对于 double 类型，scanf 函数中使用%lf 进行读数据，printf 中仍然采用%f、%e 和%g；对于 long double 类型，采用%Lf 进行读写，详见第 2 章。

(4) 由于存储方式的原因，浮点数通常只是实际值的近似值，存储时存在舍入误差。因此，在实际的编程中，要避免直接对两个实数进行相等比较。同时，还要避免将一个很小的浮点数和一个非常大的浮点数进行各种算术运算，以免导致计算结果的不正确，或发生浮点值的“上溢”或“下溢”现象。例如，下面的代码将导致浮点数上溢。

```
float toobig=3.4E38*100.0f;
printf("%e\n",toobig);
```

【例 1.6】测试浮点型变量所占的字节数及其能表示的有效位数。

程序代码如下：

```
#include <stdio.h>
void main()
{
    float x;
    double y;
    long double z;
    x=1234567890123.456789f;
    y=1234567890123.456789;
    z=1234567890123.456789L;
    //下面采用 sizeof 运算符输出各数据类型所占字节数
    printf("%d,%d,%d\n",sizeof(x),sizeof(double),sizeof(long double));
    printf("%f\n%f\n%lf\n",x,y,z);   //若要按指数形式输出，可采用格式符%e
}
```

程序运行结果如下：

```
4,8,8
1234567954432.000000
1234567890123.456800
1234567890123.456800
```

例 1.6　解析

说明：

(1) 要测试数据或数据类型在系统中所占的字节数，可采用容量测试运算符 sizeof，其用法为：sizeof(变量名)或 sizeof(数据类型)。对于采用变量名的情况，可省略“()”。本例中，float、double 和 long double 三种类型的字节数分别为 4，8，8。

(2) 从输出结果看，三种实型变量输出的小数点后的位数都为 6，但它们能精确表示数据的位数并不相同。其中，float 型变量的有效位数为 7，而 double 和 long double 型变量的有效位数为 16。因此，若对数据精度要求不高，可采用 float 类型，否则采用 double 或 long double 类型。

想一想：

从基本特性上看，double 型变量和 long double 型变量十分相像，那为什么还需要使用 long double 型呢？查阅网络或其他相关资料，给出相应的理由。

3. 字符型

字符是指程序中使用的英文字母、数字符号、标点符号以及各种特殊控制码等。计算机通常使用数字编码来处理字符，即用特定的整数表示字符。因此，从技术层面看，字符类型本质上是一种整数类型。例如，在 C 语言使用的 ASCII 码方案中，整数 65 代表大写字母 A，储存字母 A 实际上储存的是整数 65。

在 C 语言中，使用 char 关键字处理字符类型。char 类型被定义为 8 位存储单元，即 1 个字节。标准 ASCII 码的范围是 0～127，只需 7 位二进制数即可表示。因此，char 类型容纳标准 ASCII 码绰绰有余。

对于程序中用到的字符型常量，C 语言采用单引号将它们括起来，如'a'，'B'，'+'，'3'，''等。而对于一些代表行为或非打印字符，可采用转义序列进行处理。所谓转义序列，是指使用特殊的符号序列表示的一些特殊符号，它们以“\”开头进行表示。例如，'\n'表示换行，'\"'表示双引号字符，'\\'表示反斜杠字符，'\007'表示计算机扬声器发出蜂鸣声(蜂鸣字符)等。C 语言中的常见转义序列及其意义如表 1-5 所示。

表 1-5　C 语言常见转义序列及其意义

转义字符	ASCII 码值	意义
\a	0X07	警告响铃
\b	0X08	光标位置从当前位置回退一格
\t	0X09	水平制表符，光标移到 1、9、17、25 等字符位置
\n	0X0A	换行，光标移到下一行开始处

（续表）

转义字符	ASCII 码值	意义
\v	0X0B	垂直制表符
\f	0X0C	换页，即光标移到下一页的开始处
\r	0X0D	回车但不换行，即光标移到当前行的开始处
\\	0X5C	反斜杠(\)
\'	0X27	单引号(')
\"	0X22	双引号(")
\?	0X3F	问号(？)
\ddd	整数	1～3 位八进制数代表的字符，如'\101'代表 'A'
\xhh	整数	1～2 位十六进制数代表的字符，如 '\x41'代表 'A'

由表 1-5 的最后两行可知，可以用八进制数或十六进制数表示的 ASCII 码值来表示相应的字符。因此，用转义序列可输出 ASCII 码表中的任一字符。如，对于字符'A'，可以用数字 65 表示，也可将其表示为'\101'，还可以表示为'\x41'。

至此，可以声明字符型变量来处理各种字符，例如：

```
char grade='A';     //较常见的处理字符方式
char grade=65;      //正确，保存的是字符'A'
char grade='\101';  //正确，仍然为字符'A'
```

但以下的变量初始化方式将导致语法错误：

```
char grade=A;       //错误，系统会将 A 看成一个变量
char grade="A";     //错误，  "A"是字符串常量，而不是字符常量
```

可见，'A'和"A"是完全不同的，前者是字符常量，后者是字符串常量。在 C 语言中，字符串常量通常由一个或多个字符常量外加一个字符串结束标记符'\0'组成。也就是说，对于字符串"A"，实际包含的字符是'A'和 '\0'，在内存中需要两个字节进行存储。另外，C 语言没有相应的字符串变量，对字符串的处理一般采用字符数组方法，详见第 5 章。

例 1.7　解析

【例 1.7】从键盘输入一个字符，分别输出它的 ASCII 码值和字符本身。

程序代码如下：

```
#include <stdio.h>
void main()
{
    char ch;
    printf("输入一个字符：");  //提示性输入语句
    scanf("%c",&ch);          //以%c 格式读入字符，存到 ch 变量中
    printf("字符%c 的 ASCII 码值是%d。\n",ch,ch);  //按%c 和%d 格式输出 ch
}
```

程序运行结果如下：

```
输入一个字符： a
字符 a 的 ASCII 码值是 97。
```

说明：

由于 char 类型数据内部采用整数存储方式，因此它和 int 类型数据在 ASCII 码范围内是互为通用的。可以给一个 char 变量赋值一个普通字符，也可给它赋值一个在相应范围内的整数。同样，要输出一个字符型数据，具有两种输出形式，可以采用%c 格式符输出字符本身,也可以采用%d 格式符输出字符对应的 ASCII 码值，如图 1-4 所示。

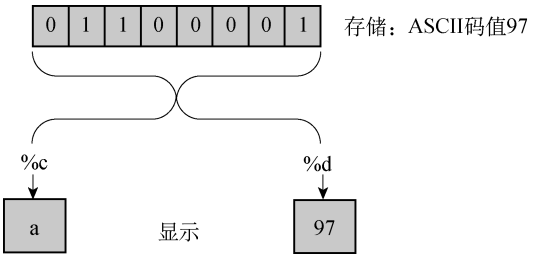


图 1-4　字符数据的存储与输出

1.7　C 语言的语法规范和编程规范

通过对本章的学习，读者已经对 C 程序的基本结构、操作过程及其基本语法规则有了一定的了解。同时，是否会有这样的感受：为什么编写程序时总是有那么多的错误和警告呢？其实，这只是因为我们才刚刚开始步入 C 程序设计之门，对其中的很多语法规范和编程习惯还没有深刻地领会和掌握。

为了便于理解和学习，本小节将一些容易导致程序出错的编程规则归并到“基本语法规范”，而将另一些尽管不会导致程序错误但却是高质量代码所必须遵守的编程约定归并到“基本编程规范”之中。

1.7.1　基本语法规范

(1) 函数是 C 程序的基本单元。一个完整的 C 程序要么仅有一个 main 函数，要么由一个 main 函数和若干个其他函数共同组成。main 函数是程序执行的起点，也是程序执行的终点，其他函数由 main 函数直接或间接地进行调用。当程序较为复杂时，可以将不同的函数组织在同一项目下的不同的.c 文件中，但只有一个文件中能含有 main 函数。

(2) 每个函数都是一个独立的功能模块。C 语言提供了丰富的库函数供直接调用，用户也可设计自己的函数。调用函数时，函数名后必须有小括号，即使该函数不带任何参数。

(3) C 语言严格区分大小写。C 编译系统把大小写字母看做两个完全不同的字符，sum，Sum，SUM 都是不同的标识符。

(4) C 标识符的命名规则。由英文字母、数字和下划线组成，且首字符不能为数字。标识符不允许使用 C 关键字和保留字，试图定义一个变量的名称为 float、printf 等将导致语法错误或引起其他问题。

(5) C 程序的具体功能由每一条 C 语句完成，每条 C 语句都以分号结束。分号是 C 语句不可缺少的组成部分。置于文件开始位置处的#include 和#define 等指令被称为预处理命令。它们是命令，不是语句，因此最后都不带分号。

(6) C 语言允许一行写多条语句，也允许将一条语句写在多行。要注意的是，尽量避免一行写多条语句。而且，要将一条语句写在多行时，只能在可以插入空格的地方进行换行。

1.7.2 基本编程规范

（1）保持程序结构清晰，简单易懂。避免使用复杂、过长和难以理解的 C 语句，避免出现冗余的代码，始终让程序目标明确、简捷有效。同时，为提高程序的清晰性并兼顾执行效率，应尽量使用标准函数库中的函数来编写程序。例如，要计算一个数的平方根，可直接使用 math.h 函数库中的 sqrt()函数；要求其绝对值，直接使用 fabs()函数。

（2）不要试图在一个函数内完成程序的所有功能。必要时，对程序功能进行合理划分，并使用结构化设计方法编制每个函数。函数体应尽量保持变量申明→数据处理→显示输出的程序结构。在利用选择语句和循环语句进行数据处理时，应尽量避免设计过多的嵌套层次，避免复杂的分支和程序转向，力求保持程序逻辑的清晰性。

（3）注重程序的可读性。包括：①变量名、函数名等标识符命名最好“见名知意”；②养成良好的注释习惯，对关键变量、关键语句、函数功能、文件功能等进行必要的说明和解释；③养成良好的缩进风格，能够利用缩进方式显示程序的逻辑结构，尽量使同一层次上的代码保持相同的缩进量(以 Tab 键为单位)；④尽可能每条语句各占一行；⑤合理使用空行和空白字符，在函数中用空行分隔概念上的多个部分。

1.8 自测练习

1. C 语言是一种(　　)。

A. 低级语言　　B. 高级语言　　C. 机器语言　　D. 汇编语言

2. 以下描述中错误的是(　　)。

A. 不同的计算机可以理解的机器语言也不同

B. 机器语言和汇编语言都和具体的硬件平台相关，统称为低级语言

C. 高级语言编写的程序较为简洁，可读性较好

D. 编译方式执行速度较快，且每次修改源程序后不必重新进行编译处理

3. 以下描述中错误的是(　　)。

A. 编写 C 程序的过程通常称为编辑，结果将得到.c 源文件

B. C 源程序经过编译，将生成扩展名为.obj 的本机目标代码文件

C. 链接过程通常是将目标代码文件和程序中使用到的其他代码连接在一起

D. 运行结果不对，通常是由于语法错误引起的

4. 以下说法中正确的是(　　)。

A. C 程序是从第一个定义的函数开始执行的

B. 在 C 程序中，要调用的函数必须在 main 函数中定义

C. C 程序是从 main 函数开始执行的

D. C 程序中的 main 函数必须放在程序的开始部分

5. 语句“printf("Welcome\nto\nC!");”将输出(　　)。

A. 1 行　　B. 2 行　　C. 3 行　　D. 4 行

6. 关于下面的程序，正确的说法是(　　)。

```
#include <stdio.h>
void main( )
{  float a=b=2, result;
   result=a/b;
   printf("result=%f\n", result);
}
```

A. 因为变量声明问题导致编译出错

B. 因为 result=a/b；语句有误导致编译出错

C. 输出结果为 result=2.000000

D. 输出结果为 result=1.000000

7. 下列程序运行时输入 *3 4*，则程序的输出结果是(　　)。

```
#include <stdio.h>
int multi(int a,int b)
{ return a*b;
}
void main()
{ int x,y,t;
  printf("Please input two integers (x and y): ");
  scanf("%d%d", &x,&y);
  t= multi(x,y);
  printf("%d\n",t);
}
```

A. 3　　B. 4　　C. 7　　D. 12

8. 以下程序的运行结果是(　　)。

```
#include <stdio.h>
#define RATE 0.05
void main( )
{ double salary,tax;
  salary=5000.00;
  tax=(salary-3000)*RATE;
  printf("The result is: %.2f\n", salary-tax);
}
```

A. 5000.00　　B. 100.00　　C. 4900.00　　D. 3000.00

9. 以下程序的运行结果为(　　)。

```
#include <stdio.h>
void main()
{  printf("%d\n",sizeof(char)+sizeof(int)+sizeof(double));
}
```

A. 148　　B. 1，4，8　　C. 144　　D. 13

10. 下列程序运行时输入 *B* ，则程序的输出结果是(　　)。

```
#include <stdio.h>
void main()
{  char ch;
   int digit;
   printf("输入一个大写字母：");
   scanf("%c",&ch);
```

```
    digit=ch-'A';
    printf("%c, %d\n",digit+'a',digit);
}
```

A. 编译出错　　B. b，B　　C. b，1　　D. B，1

扫描以下二维码就可以获取本章自测练习的解析哦！

自测练习解析

第 1 章教学

第 1 章资源