第 1 章　绪论

本章主要介绍程序、程序设计以及程序设计语言的基本概念，概述 C 语言的发展、特点以及 C 语言的标准，通过几个 C 语言源程序的实例结构，介绍 C 语言程序基本结构与书写规则，并对本教材组织结构和主要内容做简单说明。

1.1　程序与程序设计语言

1.1.1　计算机与程序

计算机是当今信息化社会中必不可少的工具。它是一种按照事先编写的程序，自动对数据进行输入、处理、输出和存储的系统。计算机要完成不同的工作，就要运行不同的程序。程序就是为完成某项任务而编写的一组计算机指令序列。编写程序的过程称为程序设计。程序设计是软件开发的关键步骤，软件的质量主要是通过程序的质量来体现的。在进行程序设计之前必须根据实际需求确定使用什么程序设计语言来编写程序。

1.1.2　程序设计语言

人与人之间交流需用相互理解的语言沟通，人与计算机交流也要使用相互理解的语言。程序设计语言就是用来实现人与计算机之间交流的，它经历了从机器语言、汇编语言到高级语言的发展历程。

1. 第一代语言——机器语言

机器语言是由 0 和 1 组成的指令序列。例如，指令 1011011000000000 表示要计算机执行一次加法操作；而指令 1011010100000000 则表示要计算机执行一次减法操作。它们的前八位表示操作码，后八位表示地址码。

机器代码可以直接被计算机所识别，因此机器语言最大的特点便是效率高，执行速度快。但是采用机器代码编写程序，要求程序员熟记所用计算机的全部指令代码和代码的含义，手编程序时，程序员必须自己处理每条指令和每一个数据的存储分配、输入/输出，还要记住编程过程中每步所使用的工作单元处在何种状态。可想而知，用机器语言编写程序是一件十分烦琐且容易出错的工作。

2. 第二代语言——汇编语言

由于用机器语言编程存在工作量大、易于出错等问题，因此人们考虑采用一些简洁的英文字母、符号串替代特定指令的二进制串，使表达方式更接近自然语言。例如用“ADD”代表加法，“MOV”代表数据传送等，这样人们就很容易读懂并理解程序在干什么，纠错及维护都很方便。这种采用英文缩写的助记符标识的语言称为汇编语言。

但是，计算机并不认识这些符号，因此需要一个专门的程序，负责将这些符号翻译成二进制数形式的机器语言才能被计算机执行，这种翻译程序称为汇编程序。

汇编语言是一种与机器语言一一对应的程序设计语言，虽然不是用 0、1 代码编写，但实质是相同的，都是直接对硬件进行操作，只不过指令采用助记符标识，更容易识别和记忆。

机器语言和汇编语言均与特定的计算机硬件有关，程序的可移植性差，属于低级语言。由于汇编语言源程序的每一句指令只能对应实际操作过程中一个很细微的动作，如移动、加法等，因此汇编源程序一般比较冗长、复杂，容易出错，而且使用汇编语言编程需要有更多的计算机专业知识，所以人们只有在直接编写面向硬件的驱动程序时才采用它。

3. 第三代语言——高级语言

到了 20 世纪 50 年代中期，人们研制了高级语言。高级语言是用接近自然语言表达各种意义的“词”和常用的“数学公式”形式，按照一定的“语法规则”编写程序的语言。这里的“高级”，是指这种语言与自然语言和数学公式相当接近，而且不依赖于计算机的型号，通用性好。高级语言的使用，改善了程序的可读性、可维护性和可移植性，大大提高了编写程序的效率。

用高级语言编写的程序称为高级语言源程序，不能被计算机直接识别和执行，也要用翻译的方法把高级语言源程序翻译成目标程序才能执行。

高级语言的出现大大简化了程序设计，缩短了软件开发周期，显示出强大的生命力。此后，编制程序已不再是软件专业人员才能做的事，一般工程技术人员花上较短的学习时间，也可以使用计算机解题。随着计算机应用日益广泛地渗透到各学科和技术领域，后来发展了一系列不同风格的、为不同对象服务的程序设计语言，其中较为著名的有 FORTRAN、BASIC、COBOL、ALGOL、LISP、LP/1、PASCAL、C 等十几种语言。

1.1.3 高级语言程序的开发过程

有人认为程序设计是一门艺术，而艺术在很大程度上是基于人的灵感和天赋，它往往没有具体的规则和步骤可循。对于一些小型程序的设计，上述说法可能有一些道理。但是，对于大型复杂程序的开发，灵感和天赋不是很好的解决之道，几十年的程序开发实践已表明这一点。事实上，程序设计是一门科学，程序的开发过程是有规律和步骤可循的。通常，高级语言程序的开发遵循以下步骤。

1. 明确问题

用计算机解决实际问题，首先要明确解决什么问题，即做什么？如果对问题都没有搞清楚或理解错了，就试图解决它，其结果是可想而知的。

2. 算法设计

明确问题之后，就要考虑如何解决它，即如何做？计算机解决问题的方式就是对数据进行处理，因此，首先要对问题进行抽象，抽取出能够反映问题本质特征的数据并对其进行描述，然后设计计算机对这些数据进行处理的操作步骤，即算法设计。

3. 选择某种语言进行编程

算法设计完成后，就必须要用某种实际的程序设计语言来表达，即编程实现。现在的程序设计语言很多，用哪一种语言来编程呢？从理论上讲，虽然各种语言之间存在着或多或少的差别，但它们大多数都是基于冯·诺依曼体系结构的，它们在表达能力方面是等价的，因此对于同一个设计方案，用任何一种语言都能实现。

在实际中，采用哪一种语言来编程可以考虑以下因素决定：

- 设计方案。例如，对于采用功能分解的设计方案，用某种过程式程序设计语言进行编程比较合适；对于面向对象的设计方案，采用面向对象的程序设计语言来实现，

就比较自然和方便。

- 编程语言效率的高低、使用的难易程度、数据处理能力的强弱，等等。
- 一些非编程技术的因素。例如，编程人员的个人喜好。

选定了编程语言之后，下面就是使用该语言编写程序。对于同一个设计方案，不同的人会写出不同风格的程序。程序设计风格的好坏会影响到程序的正确性和易维护性。程序设计风格取决于编程人员对程序设计的基本思想、技术以及语言掌握的程度。

4. 测试与调试

程序写好之后，其中可能含有错误。程序错误通常有 3 种：

- 语法错误。是指程序没有按照语言的语法规则来书写，这类错误可由编译程序来发现。
- 逻辑（或语义）错误。是指程序没有完成预期的功能。
- 运行异常错误。是指对程序运行环境的非正常情况考虑不周而导致的程序异常终止。

这些错误可能是编程阶段导致的，也有可能是设计阶段甚至是问题定义阶段的缺陷。

程序的逻辑错误和运行异常错误一般可以通过测试来发现。测试方法有很多，比如：

- 静态测试。即不运行程序，而是通过对程序的静态分析，找出逻辑错误。
- 动态测试。即利用一些测试数据，通过运行程序，观察程序的运行结果是否与预期的结果相符。

值得注意的是，不管采用何种测试手段，都只能发现程序有错，而不能证明程序正确。例如：想要用动态测试技术来证明程序没有错误，就必须对所有可能的输入数据来运行程序并观察运行结果，这往往是不可能的，并且也没有必要。测试的目的就是要尽可能多地发现程序中的错误。

测试工作不一定要等到程序全部编写完成才开始进行，可以采取编写一部分、测试一部分的方式来进行，最后再对整个程序进行整体测试。即先进行单元测试，再进行集成测试。

如果通过测试发现程序有错误，那么就需要找到程序中出现错误的位置和原因，即错误定位。给错误定位的过程称为调试(debug)。调试一般需要运行程序，通过分段观察程序的阶段性结果来找出错误的位置和原因。

5. 运行与维护

程序通过测试后就可交付使用了。由于所有的测试手段只能发现程序中的错误，而不能证明程序没有错误，因此，在程序的使用过程中可能会不断发现程序中的错误。在使用过程中发现并改正错误的过程称为程序的维护。程序维护可分成 3 类：

- 正确性维护。是指改正程序中的错误。
- 完善性维护。是指根据用户的要求使得程序功能更加完善。
- 适应性维护。是指把程序移植到不同的计算平台或环境中。

1.2 C 语言的发展和特点

1.2.1 C 语言的发展历史

1969 年，美国贝尔实验室的 Ken Thompson 和 Dennis Ritchie 开始研制 UNIX 操作系统。UNIX 的早期版本是用汇编语言编写的。但汇编语言依赖于计算机硬件，程序的可读性和可

移植性都比较差。为了提高程序的可读性和可移植性，希望改用高级语言编写系统软件，但一般的高级语言不能像汇编语言一样直接对硬件进行操作。因此，他们决定开发一种高级语言来编写 UNIX 操作系统。

1970 年，Ken Thompson 以 BCPL 语言为基础，设计出了很简单又能访问硬件的 B 语言(BCPL 的第一个字母)，并用 B 语言对用汇编语言编写的 UNIX 操作系统进行了部分改写，此时的 B 语言过于简单，功能有限。1972~1973 年间， Dennis Ritchie 在 B 语言的基础上设计出了 C 语言（BCPL 的第二个字母）。C 语言既保持了 BCPL 和 B 语言精炼和访问硬件的优点，又克服了它们过于简单和无数据类型等缺点。1973 年，Ken Thompson 和 Dennis Ritchie 将 UNIX 操作系统的 90%用 C 语言改写。

多年来，UNIX V 系统配备的 C 语言一直是公认的 C 语言标准，这在由 Brian Kernighan 和 Dennis Ritchie 合著的“The C Programming Language”(Prentice-hall 于 1987 年出版，国内在 2004 年出版了该书的中译本《C 程序设计语言（第 2 版）》)书中介绍。

在 C 语言的发展过程中还有两个重要的标准：C89 和 C99 标准。1983 年，美国国家标准协会(ANSI)着手制定 C 语言的标准，于 1989 年正式被批准为 ANSX3.159-1989，一年以后，该标准也被 ISO(国际标准化组织)接收，定为 ISO/IEC 9899:1990。通常仍称 C89 标准。

随着 C 语言的继续发展，1999 年 C99 标准应运而生。C99 保持了几乎所有 C89 的特征。总的来说，C99 与 C89 之间有 3 种变化：

- 在 C89 基础上增加的特性。其中最主要的是 C99 增加了 5 个新的关键字（C89 具有 32 个关键字，而 C99 达到 37 个关键字）：_Bool、_Imaginary、restrict、_Complex 和 inline。

  增加的其他特性主要包括：变长数组；单行注释；long long int 数据类型；可以在语句可以出现的任何地方定义变量；对预处理程序的增加；for 语句内的变量声明；柔性数组结构成员；新的库和头文件等。
- 删除了 C89 中的某些特性。例如，“隐含的 int”。在 C89 中，大多数情况下，当没有明确指定类型标识符时，通常认为其为 int 类型，但这一点在 C99 中是不允许的。此外 C99 中还删除了函数的隐含声明。在 C89 中，如果一个函数在被使用前未被声明，将被视为隐含的函数声明，但 C99 不支持这一点。
- 修改了 C89 中的某些特性。例如：放宽的转换限制、扩展的整数类型、增强的整数类型提升规则、对 return 语句的约束等。

1.2.2 C 语言的特点

与其他高级语言相比，C 语言之所以发展迅速，成为最受欢迎的语言之一，主要原因是它具有强大的功能。归纳起来，C 语言具有以下一些特点：

1. C 语言是中级语言

通常，C 语言被称为中级语言。这并不是说它功能差、难以使用，而是 C 语言既具有高级语言的基本结构和语句，又具有低级语言的实用性，因此人们称之为“高级语言中的低级语言”或“中级语言”。例如，C 语言允许直接访问物理地址，可以像汇编语言一样对位、字节和地址进行操作。

2. C 语言是结构化程序设计语言

结构化语言的特点是代码与数据分隔，即程序的各个部分除了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、维护以及调试。作为一种结构化程序设

计语言，其逻辑结构由顺序、选择和循环三种基本结构组成，以函数作为模块，实现程序的模块化设计，符合现代编程风格。

3. 语言简洁、紧凑，使用方便、灵活

C89 标准定义的 C 语言只有 32 个关键字，9 种控制语句。和 IBM-PC 的 BASIC 相比，BASIC 包含的关键字达 159 个之多。C 程序主要由小写字母组成，书写格式自由。C 程序比较简练，源程序短，输入程序时工作量少。

4. 运算符和数据结构丰富，表达式多样

C 语言共有 34 种运算符。在 C 语言中把括号、赋值、强制类型转换等都作为运算符处理，灵活使用各种运算符可以实现在其他高级语言中难以实现的运算。表达式类型多样，既提高了编译效率和目标代码的质量，又提高了程序的可读性。

C 语言提供了各种各样的数据类型，如整型、实型、字符型、数组类型、指针类型、结构体类型等，能够实现各种数据结构，如线性表、链表、栈、队列、树、图等。尤其是指针类型数据，使用起来灵活、多样，程序效率更高。

5. 语法限制不太严格，程序设计自由度大

C 语言编译系统的语法检查不太严格。例如，在 C 语言中对数组下标越界不进行检查，由编程者自己保证程序的正确；变量类型使用灵活，整型和字符型变量可以通用等。其优点是允许编程者有较大的自由度。但明显的缺点是增加了程序的不安全因素。这就要求编程者在编程时自我约束，养成良好的、严谨的编程习惯，程序编好后要仔细检查。

6. 生成的目标代码质量高

C 语言生成的目标代码只比汇编语言生成的代码效率低 20%左右，程序执行的效率高。

7. C 程序的可移植性好

与汇编语言相比，C 程序基本上不做或稍做修改就可在其他型号的计算机上运行。

总之，由于 C 语言的上述特点，使得 C 语言越来越受到程序设计人员的重视，在很多领域得到了广泛应用。

当然 C 程序也存在一些不足，如运算优先级太多，不便于记忆，类型检验太弱，虽然转换比较方便，但同时也增加了程序的不安全因素等。

1.2.3 C 和 C++

C++是以 C 语言为基础的面向对象程序设计语言，对于 C89 而言，因为 C++标准的制定包容了 C89 的全部内容，所以 C++实现了它的全部特性。而对于 C99 而言，部分 C99 特性 C++并未包含。因此，对 C99 而言，C++不是 C 语言的超集，C 语言也不是 C++的子集。

一般情况下，绝大部分 C++编译器可以编译 C 和 C++程序。因此，很多编程者喜欢用 C++编译器来编译 C 程序。这是可行的，但是对于初学 C 语言的读者来说，一定要注意：它们毕竟是两种不同的环境，很多特性并不具有通用性。部分 C++编译器建立在 C89 的基础之上，许多 C99 的新特性未必能够编译通过(当然，也有部分编译器加入了 C99 的几乎全部新特性)。本教材使用 Visual C++ 6.0 提供的编译器，只部分支持 C99 标准，因此本教材以 C89 标准来编写程序。

还有一点需要注意的是，C 语言源程序的扩展名是.c，而 C++源程序的扩展名是.cpp。

因此，C 语言学习者在使用 C++工具编译 C 程序时要注意将源文件的后缀命名为.c，而不是.cpp。如果扩展名为.cpp，那么编译器将按照 C++的要求编译源文件。

1.3 C 程序的结构与书写风格

了解和掌握程序的结构与书写风格是编写程序的基础，一般来说，对于刚开始学习 C 的读者来讲，一个程序可看作是由函数构成的。为了对程序中的有关内容进行说明，在程序的开头常包含有一些声明。下面通过一个简单的例程来介绍 C 程序的结构与书写风格。

【例 1.1】 一个简单的 C 程序。

程序如下：

```
/*This is the first C program.*/
#include <stdio.h>
int main( )
{
    printf("Hello.\n");   //屏幕输出

    return 0;
}
```

在 Visual C++集成开发环境中输入程序，经过编译、连接后运行，运行结果截图如下：

说明：

（1）注释

程序的第 1 行是注释语句。在 C 语言中，注释是程序员为了增加程序的可读性而增加的说明性信息，对程序的运行不起作用，对源程序进行编译时，注释会被忽略。

在 C89 中，注释由“/*……*/”来完成，/*和*/中间所包含的任何多行内容即为注释部分。在 C99 中增加了单行注释功能，即注释也可以用“//”来表达，从“//”开始一直到本行结束的所有内容都属于注释部分。如，本例第一行注释可写为：

```
// This is the first C program.
```

本书中的例程两种注释都采用。一般的，注释符“//”用于行注释，注释符“/*……*/”用于对程序块注释。

在 C 语言程序中，适当的注释可以帮助程序阅读人员理解程序，不过注释过多会使程序看起来不够清晰简洁。另外，标准 C 可以在 C 程序的任意位置进行注释，但是良好的编程风格要求变量的定义部分最好用注释来详细说明每一个变量的意义、作用及 其使用范围；在函数前面可以注释说明整个函数的意义及其作用；语句前面或者语句末尾 也可以用注释来说明该语句的作用等。一般不要将注释放在语句的下一行，这会让读者误以为是对下一行的注释。

在程序的调试中，也常常可以通过增加注释符来临时删除一些语句，帮助我们完成语句的修改，请读者在后面的学习中体会这一点。

（2）main()函数

C 程序由函数构成，函数是程序的基本单位。本例中这个简单的 C 程序只包括一个函数，函数名为 main()，这是主函数。主函数是特殊函数，不可以随便取名，它的名字是唯一

的。一个 C 程序可以包括多个函数，但有且只有一个主函数。主函数可以放在 C 程序的任意位置，但任何一个 C 程序都是从主函数开始执行，而且也在 main()函数中结束。

在 main()函数中通常还需要调用其他函数，这些被调用的函数可以是用户自己定义的函数，也可以是系统提供的库函数。因此，使用 C 语言编程实际就做两件事：编制不同功能的函数；调用这些函数。

（3）函数参数和返回值

函数名 main 后的一对括号中可以包含若干个参数，这里 main( )函数不带任何参数，但是括号仍然保留。

需要注意的是，在 C89 标准中，如果函数没有参数，函数名后的小括号内可以什么也不写。但是 C99 标准规定，在无参函数的函数名后的小括号内加上 void 以明确声明该函数没有参数。本书所有 main()函数采用 C89 标准。

main()函数前面的 int 说明了主函数的返回值类型。在 C99 标准中，main()函数的返回值类型必须是 int 型。而 C89 标准中也可返回 void 型，默认则返回 int(即不书写任何返回类型)。根据 C99 标准，本书所有 main()函数都返回 int 型。

（4）函数体

花括号{ }用来标识函数的开始和结束，必须成对出现。包含在一对花括号之间的部分就是函数体。函数的功能也就是函数体所要完成的工作。

函数体由语句组成，语句必须以分号结尾。本例主函数 main()仅包含两条语句，其中第 1 条语句是调用库函数 printf()，在显示器上输出相应的信息。第 2 条语句“return 0;”的功能是退出 main()函数，返回操作系统。

（5）标准函数与头文件

printf()函数是一个由系统定义的标准输出库函数，其功能是在显示器上输出内容，实现输出操作。在 C 语言里没有专用的输入、输出语句，输入/输出操作都是通过函数实现的。C 语言提供了大量的标准库函数供编程者使用。

从某种意义上说，使用 C 的过程就是不断熟悉各种库函数的过程。

当需要用到某些标准库函数时，需要将对应的头文件用#include 命令包含在程序首部。头文件提供了各类标准库函数的原型说明。本例由于使用了 printf()函数，因此在程序第 1 行有#include <stdio.h>命令。

1.4 本教材组织结构和主要内容

程序设计是一门理论性和实践性都很强的课程。本书采用理论和实践相结合的方式组织 C 语言程序设计内容，由理论教学篇和上机实践篇两个部分组成。理论教学篇以程序设计过程为主线，以任务驱动方式讲授 C 语言的各种语言成分以及程序设计的基本概念、基本理论和基本方法，上机实践篇以 Visual C++6.0 为程序开发环境，介绍 C 语言程序的上机操作过程、程序调试方法以及实验内容。通过本课程的学习，学生应该能够正确理解 C 语言的各种语言成分，阅读 C 程序；掌握结构化程序设计方法，形成良好的程序设计风格；具备一定的程序设计能力，具备较强的上机操作和程序调试技能。

理论教学篇分成 12 章，第 1 章绪论，介绍程序设计语言的发展、C 语言的特点和 C 程序的基本结构；第 2 章介绍数据的基本类型与基本运算；第 3 章介绍数据的输入输出方式，第 4 章介绍程序的基本语句与基本结构，包括顺序、分支、循环 3 种基本结构以及结构化程序设计；第 5 章介绍了函数和模块化程序设计；第 6 章介绍数组；第 7 章介绍指针及其应用；第 8 章介绍自定义数据类型与链表，包括结构体、共用体和枚举类型；第 9 章介绍文件的基

本操作；第 10 章介绍编译预处理与多文件组织；第 11 章介绍位运算操作，为选讲内容；第 12 章给出一个综合应用实例。各章节拓扑关系如图 1.1 所示，其中打“*”号表示为选讲内容。上机实践篇给出了 10 个实验内容和 2 道课程设计题，对每个实验明确实验要求，提示分析问题解决问题方法和程序设计思路。

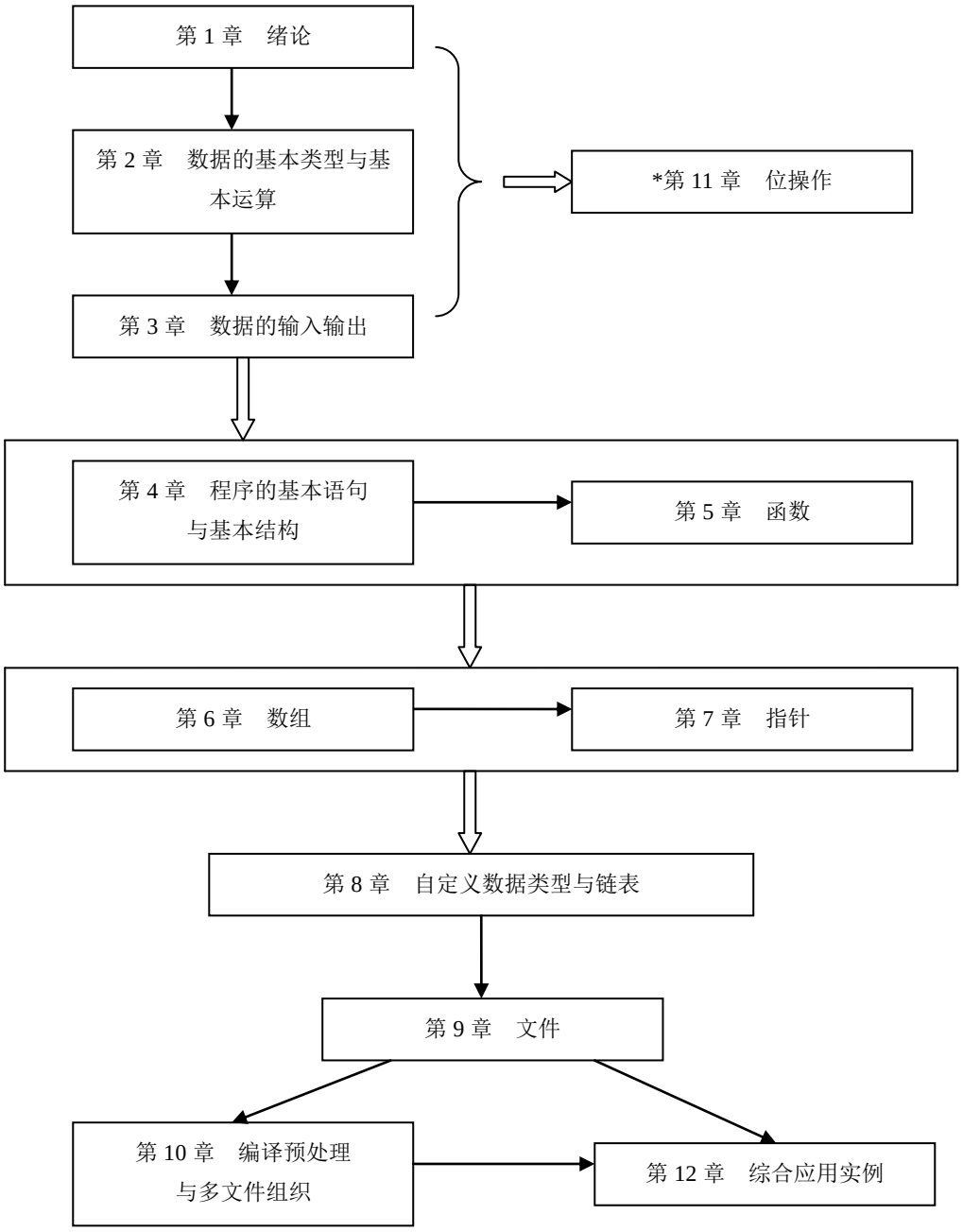

图 1.1　理论教学篇各章节拓扑关系

本章小结

本章介绍了程序、程序设计、程序设计语言的概念以及程序的开发过程；阐述了 C 语言的发展、特点以及 C 语言的标准；介绍了 C 程序的基本结构与书写风格。

关于 C 程序的基本结构与书写风格，概括如下：

- C 程序都是由一个或多个函数构成，其中，有且只有一个主函数 main( )，C 程序的执行都是从主函数开始。
- 函数是 C 程序的基本单位。C 程序中有两种类型的函数：一类是系统提供的标准函数库的函数，另一类是用户自定义函数。
- 函数由语句构成，语句必须以分号结尾。C 程序书写格式自由，一条语句可以占多行，一行也可以含多条语句。多条语句可以用{}括起来以构成复合语句。
- C 语言没有输入/输出语句，输入/输出操作是通过函数完成的。
- C 语言区分大小写字母，C 程序习惯上使用小写字母书写，对于一些特殊含义的变量，偶尔也用大写字母表示。
- C 语言用/* */或//（C99 标准，Visual C++支持）作注释。

习题 1

1. 简要说明程序、程序设计、程序设计语言的概念。
2. 简要说明高级语言程序的开发过程。
3. 简述 C 语言的主要特点。
4. 简述 C 程序的结构。