XXX

March 15, 2020

Contents

Chapter 1

FOREWORD

20 世纪 90 年代以来，C 语言迅速在全世界普及推广。无论在中国还是在世界各国，“C 语言程序设计”始终是高等学校的一门基本的计算机课程。C 语言程序设计在计算机教育和计算机应用中发挥着重要的作用。

作者于 1991 年编著了……

在此书再版之际，作者想对学习程序设计问题提出以下几点看法。

1.1 为什么要学习程序设计?

大学生，特别是理工科的学生，不能满足于只会用办公软件，应当有更高的要求。

只有懂得程序设计，才能进一步懂得计算机，真正了解计算机是怎样工作的。通过学习程序设计，学会进一步了解计算机的工作原理，更好地理解和应用计算机；掌握计算机处理问题的方法；培养分析问题和解决问题的能力；具有编制程序的初步能力。

因此，无论是算计专业学生还是非计算机专业学生，都应当学习程序设计知识，并且把它作为进一步学习与应用计算机的基础。

1.2 为什么选择 C 语言。

进行程序设计，必须用一种计算机语言作为工具。可供选择的语言很多。C 语言功能丰富、使用灵活方便、应用面广、目标程序效率高、可移植性好。

有人认为 C++ 语言出现后，C 语言过时了……它比 C 语言复杂得多，事实上，将来不是每个人都需要用 C++ 编制大型程序。C 语言是更为基本的。美国一位资深软件专家写了……他说“大学生毕业前要学好 C 语言，C 语言是当前程序员共同的语言。它使程序员互相沟通……不管比懂得多少延续、闭包、异常处理，只要你不能解释为什么 `while(* s++ = * t++)` 的作用是复制字符串，那你就是在盲目无知的情况下编程，就像一个医生不懂最基本的解剖学就开处方。”

C 语言更适合解决某些小型程序的编程。

现在大多数高校把 C 语言作为第一门计算机语言进行程序设计教学，这是合适的，有了 C 的基础，在需要时进一步学习 C++，也是很容易过度的。

1.3 怎样组织程序设计的教学？怎样处理算法和语言的关系。

进行程序设计，要解决两个问题：

1. 要学习和掌握解决问题的思路和方法，即算法;
2. 学习怎样实现算法，即用计算机语言编写程序，达到用计算机解题的目的。

因此，课程的内容应当主要包括两个方面：算法和语言……作者的做法是：以程序设计为中心，把二者紧密结合起来，既不能孤立地抽象研究算法，也不能孤立地学习语法。

算法是重要的，但本科从不是专门研究算法与逻辑的理论课程，不可能系统全面地介绍算法；也不是脱离语言环境研究算法，而是在学习编程过程中，介绍有关典型算法，引导学生思考怎样构造一个算法。编写程序的过程就是 i 设计算法的过程。

语言工具也是重要的……决不能把程序设计课程变成枯燥地介绍语法的课程，学校语法要服务于编程。

我们坚决摒弃孤立地介绍语法的做法，而是以程序设计为中心，把算法与语言紧密结合起来。不是根据语言规则的分类和顺序作为教学和教材的章节和孙需，而是从应用的角度切入，以编程为目的，从初学者的认识规律出发，由浅入深，构造教材和教学的体系……随着变成难道的逐步提供，算法和语法的学习同步趋于深入。学生在富有创意的编程中，学会了算法，掌握了语法，把枯燥无味的语法规则变成生动活泼的编程应用。事实证明这种做法是成功的。

近年来许多学校的经验表明，按照这种思路进行教学，取到教师容易教，学生容易学的效果。

1.4 怎样学习 C 程序设计

1. 要着眼于培养能力。C 语言程序设计并不是一门纯理论的课程，而是一门应用的课程。应当注意培养分析问题的能力、构造算法的能力、编程的能力和调试程序的能力。

2. 要把重点放在解题的思路上，通过大量的例题学习怎样设计一个算法，构造一个程序。初学时不要在语法细节上死抠。语法细节是需要通过较长的实践才能掌握的。

3. 掌握基本要求，注意打好基础。如果学时有限，有些内容可以选学，把精力放在最基本、最常用的内容上，学好基本功。

4. 要十分重视实践环节。光靠听课和看书是学不会程序设计的，学习本课程既要掌握概念，又必须动手变成，还要亲自上机调试运行。读者一定要重视实践环节，包括编程和上机……考核方法应当是编写程序和调试长线，而不应该只采用选择题。

5. 要举一反三。学校程序设计，主要是掌握程序设计的思路和方法。学会 i 使用一种计算机语言变成，在需要时改用另一种语言应该不会太困难。

6. 要提倡和培养创新精神。教师和学生都不应该局限于教材中的内容，应该启发学生的学习兴趣和创新意识。能够在教材程序的基础上，思考更多的问题，编写难度更大的程序。在本书每章的习题中，包括了一些难度较大的题目，建议学生尽量选做，学会自己发展只是，提高能力。

7. 如果对学生有较高的程序设计要求，应当在学习本课程后，安排一次集中的课程设计环节，要求学生独立完成一个有一定规模的程序。

1.5 从实际出发，区别对待

对计算机专业学生……

1.6 为什么要修订《C 程序设计》

任何工作都要与时俱进……

1.7 为了满足不同的需要，出版不同层次的 C 程序设计教材

Chapter 2

常见错误分析

Chapter 3

程序设计和 C 语言

3.1 什么是计算机程序

第一个要明确的概念是计算机程序（程序)。程序是用来告诉计算机对数据如何进行处理的指令集合。

3.2 什么是计算机语言

这种计算机能直接识别和而接受的二进制代码称为机器指令（machine instruction)。机器指令的集合就是该计算机的机器语言（machine language)。

符号语言

人们创造了符号语言，它用一些英文字母和数字表示一个指令，符号语言又称为符号汇编语言或汇编语言（assembler language)。

高级语言的发展历程高级语言经历了不同的发展阶段:

1. 非结构化的语言。初期的语言属于非结构化的语言，编程风格比较随意，只要符合语法规则即可，没有严格的规范要求，程序中的流程可以随意跳转。人们为为最修程序执行的效率而采用许多“小技巧”，使程序变得难于阅读和维护。

2. 结构化的语言。为了解决以上问题，提出了“结构化程序设计方法”，规定程序必须由具有良好特性的基本结构（顺序结构、分支结构、顺环结构）构成，程序中的流程不允许随意跳转，程序总是由上而下顺序执行各个基本结构。这种程序结构清晰，易于编写、阅读和维护。QBASIC,FORTRAN 77 和 C 语言等属于结构化的语言，这些语言的特点是支持结构化程序设计方法。

   以上两种语言都是基于过程的语言，在编写程序时需要具体指定每一个过程的细节。在编写规模较小的程序时，还能得心应手，但在处理规模较大的程序时，就显得捉襟见肘、力不从心了。在实践的发展中，人们又提出额面向对象的程序设计方法。程序面对的不是过程的细节，而是一个对象，对象是有数据以及对数据进行的操作组成的。

3. 面向对象的语言。近十多年来，在处理规模较大的问题时，开始使用面向对象的语言。C++,C# ,Visual Basic 和 Java 等语言是支持面向对象程序设计方法的语言。有关面向对象的程序设计方法和面向对象的语言在本书不做详细介绍，有兴趣的可参考有关专门书籍。

3.3 C 语言的发展及其特点

本书是介绍怎样利用 C 语言作为工具进行程序设计的。为什么要选择 C 语言呢? 这里有必要对 C 语言的发展和特点有一定的了解。

C 语言是国际上广泛流行的计算机高级语言。

C 语言的祖先是 BCPL 语言。1967 年英国剑桥大学的 Martin Richards 推出没有类型的 BCPL (Basic Combined Programming Languade) 语言。1970 年美国 AT&T 贝尔实验室的 Ken Thompson

以 BCPL 语言为基础，设计除了很简单且很接近硬件的 B 语言（取 BCPL 的第一个字母）。但 B 语言过于简单，功能有限。1972-1973 年间，美国贝尔实验室的 D.M.Ritchie 在 B 语言的基础上设计出了 C 语言。C 语言既保持了 BCPL 和 B 语言的优点（精练，接近硬件），又克服了它们的缺点（过于简单、无数据类型等），C 语言的新特点主要表现在具有多种数据类型（如字符、数值、数组、结构体和指针等）。开发 C 语言的目的在于尽可能降低用它写的软件对硬件平台的依赖程度，使之具有可移植性。

最初的 C 语言只是为描述和实现 UNIX 操作系统提供一种工作语言而设计。1973 年，Ken Thompson 和 D.M.Ritchie 合作把 UNIX 的 90% 以上用 C 语言改写，即 UNIX 第 5 版。随着 UNIX 的日益广泛使用，C 语言也迅速得到推广。1978 年以后，C 语言先后移植到大、中、小和微型计算机上。C 语言便很快风靡全世界，成为世界上应用最广泛的程序设计高级语言。

以 UNIX 第 7 版中的 C 语言编译程序为基础, 1978 年, Brian W.Kernighan 和 Dennis M.Ritchie 合著了影响深远的名著 The C Programming Language，本书中介绍的 C 语言成为广泛使用的 C 语言版本的基础，它是实际上第一个 C 语言标准。

C 语言是一种用途广泛、功能强大、使用灵活的过程性（procedural）编程语言，即可用于编写应用软件，又能用于编写系统软件。因此 C 语言问世以来得到迅速推广。自 20 世纪 90 年大初，C 语言在我国开始推广以来，学习和使用 C 语言的人越来越多，成了学习和使用人数最多的一种计算机语言，绝大多数理工科大学都开设了 C 语言程序设计课程。掌握 C 语言成为计算机开发人员的一项基本功。

C 语言有以下一些主要特点。

1. 语言简洁、紧凑，使用方便、灵活。C 语言一共只有 37 个关键字，9 种控制语句，程序书写形式自由。C 语言程序比其他许多高级语言简练，源程序短，因此输入程序时工作量少。

2. 运算符丰富。C 语言的运算符包含的范围很广泛，共有 34 种运算符。C 语言把括号、赋值和强制类型转换等都作为运算符处理，从而使 C 语言的运算类型极其丰富，表达式类型多样化。灵活使用各种运算符可以实现在其他高级语言种难以实现的运算。

3. 数据类型丰富。C 语言提供的数据类型包括：整型、浮点型、字符型、数组类型、指针类型、结构体类型和共用体类型等，C99 又扩充了复数浮点类型、超长整型和布尔类型等。尤其时指针类型数据，使用十分灵活和多样化，能用来事项各种复杂的数据结构（如链表、树、栈等）的运算。

4. 具有结构化的控制语句。用函数作为程序的模块单位，便于事项程序的模块化。C 语言时完全模块化和结构化的语言。

5. 语法限制不太严格，程序设计自由度大。

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

7. 用 C 语言编写的程序可移植性好。

8. 生成目标代码质量高，程序执行效率高。

C 原来是专门为编写系统软件而设计的，许多大的软件都用 C 语言编写，这是因为 C 语言的可移植性好和硬件控制能力高，表达和运算能力强。许多以前只能用汇编语言处理的问题，后来可以改用 C 语言来处理了。目前 C 的主要用途之一是编写“嵌入式系统程序”。由于具有上述优点，使 C 语言应用十分广泛，许多应用软件也用 C 语言编写。

对 C 语言以上的特点，待学完 C 语言以后再回顾以下，就会有比较深的体会。

3.4 最简单的 C 语言程序

为了使用 C 语言编程序，必须了解 C 语言，并且能熟练地使用 C 语言。本书将由浅到深地介绍怎样阅读 C 语言程序和使用 C 语言编写程序。

3.4.1 最简单的 C 语言程序举例

3.4.2 C 语言程序的结构

3.5 运行 C 程序的步骤与方法

计算机不能直接识别和执行用高级语言写的指令。必须用编译程序把 C 源程序翻译成二进制形式的目标程序，然后再将该目标程序与系统的函数库以及其他目标程序连接起来，形成可执行的目标程序。

在编写好一个 C 源程序后，怎样上级进行编译和运行呢？一般要经过以下几个步骤：

1. 上机输入和编辑源程序。
2. 对源程序进行编译，先用 C 编译系统提供的“预处理器”对程序中的预处理指令进行编译预处理。由预处理得到的信息与程序其他部分一起，组成一个完整的、可以用来进行正是编译的源程序，然后由编译系统对该源程序进行编译。

   编译的作用首先是对源程序进行检查，判定它有无语法方面的错误，如有，则发出“出错信息”，告诉编程人员认真检查改正。修改程序后重新进行编译，如有错，在发出“出错信息”。如此反复进行，知道没有语法错误为止。这是，编译程序自动把源程序转换为二进制形式的目标程序。如果不特别指定，此目标程序一般也存放在用户当前目录下，此时源文件没有消失。

   在用编译系统对源程序进行编译时，自动包括了预编译和正式编译两个阶段，一气呵成。用户不必分别发出二次指令。
3. 进行链接处理。经过编译所得到的二进制文件还不能共计算机直接执行。前面已说明：一个程序可能包含若干个源程序文件，而编译是以源程序文件为对象的，一次编译只能得到与一个源程序文件相对应的目标文件，它只是整个程序的一部分。必须把所有的编译后得到的目标模块链接装配起来，在于函数库相连接成为一个整体，生成一个可供计算机执行的目标程序，成为可执行程序（executive program）。

   即使一个程序只包含一个源程序文件，编译后得到的目标程序也不能直接运行，也要经过连接阶段，因为要与寒暑假进行链接，才能生成可执行程序。
4. 运行可执行程序，得到运行结果。

一个程序从编写到运行成功，并不是一次成功的，往往要经过多次反复。编写好的程序并不一定能保证正确无误，除了用人工方式检查外，还必须借助编译系统来检查有无语法错误。

为了编译、链接和运行 C 程序，必须要有相应的编译系统。

写出源程序后可以用任何一种编译系统对程序进行编译和连接工作，只要用户感到方便、有效即可。

对编译系统的态度是，不应当只会一种编译系统，无论用哪一种编译系统，都应当能举一反三，在需要时会用其他编译系统进行工作。

3.6 程序设计的任务

如果只是编写和运行一个很简单的程序，上面介绍的步骤就够了。但是实际上要处理的问题比上面见到的例子复杂得多，需要考虑和处理的问题也复杂得多。程序设计就是指从确定任务到得到结果、写出文档的全过程。从确定问题到最后完成任务，一般经历以下几个工作阶段：

1. 问题分析。对于接手的任务要进行认真的分析，研究所给定的条件，分析最后应达到的目标，找出解决问题的规律，选择解体的方法。在此过程种可以忽略一些次要的因素而使问题抽象化。这就是建立模型。
2. 设计算法。即设计出解体的方法和具体步骤。
3. 编写程序。根据得到的算法，用一种高级语言编写出源程序。
4. 对源程序进行编辑、编译和链接，得到可执行程序。

5. 行程序，分析结果。运行可执行程序，得到运行结果。能得到运行结果并不意味着程序正确，要对结果进行分析，看它是否合理。

6. 编写程序文档。许多程序使提供给别人使用的，如同正式的产品应当提供产品说明书一样，正式提供给用户使用的程序，必须向用户提供程序说明书。内容包括：程序名称、程序功能、运行环境、程序的装入和启动给、需要输入的数据，以及使用的注意事项等。

3.7 习题

1. 什么是程序？什么是程序设计？

2. 为什么需要计算机语言？高级语言的特点？

3. 正确理解以下名词及其含义：

   (a) 源程序、目标程序、可执行程序

   (b) 程序编辑、程序编译、程序链接

   (c) 程序、程序模块、程序文件

   (d) 函数、主函数、被调用函数、函数库

   (e) 程序调试、程序测试

4. 自学本书中附录 A，熟悉上机运行 C 程序的方法，上级运行本章 3 个例题。

5. 请参照本章例题，编写一个 C 程序，输出以下信息：

6. 编写一个 C 程序，输入 a，b，c 三个值，输出其中最大者。

7. 上级运行以下程序，注意注释的方法。分析运行结果，掌握注释的用法。

```
int main()
{
    printf("How do you do!\n"); //这是行注释，注释范围从//至换行符
}
printf("How do you do!\n"); /*这是块注释*/
printf("How do you do!\n"); /*这是
                              块注释*/
```

Chapter 4

算法——程序的灵魂

通过第 1 章的学习，了解了 C 语言的特点，看到了简单得 C 语言程序。现在从程序的内容方面进行讨论，也就是一个程序中应该包含什么信息。或者说，为了实现解题的要求，程序应当向计算机发送什么信息。

一个程序主要包括以下两发面的信息：

- 对数据的描述。在程序中要指定那些数据以及这些数据的类型和数据的组织方式。这就是数据结构（data structure）。
- 对操作的描述。即要求计算机进行操作的步骤，也就是算法（alorithm）。

数据是操作的对象，操作的目的是对数据进行加工处理，以得到希望的结果。

没有原料是无法加工成所需菜肴的……著名计算机科学家沃思提出一个公式：

算法 + 数据结构 = 程序

直到今天，这个公司对于过程化程序来说依然是适用的。

实际上，一个过程化的程序除了以上两个主要要素以外，要应当采用结构化程序设计方法进行程序设计，并且用某一种计算机语言表示。因此，算法、数据结构、程序设计方法和语言工具 3 个方面是一个程序设计人员所具备的知识，在设计一个程序时要综合运用这几方面的知识。本书中不可能全面介绍这些内容，它们都属于有关的专门课程范畴。在这 4 个方面中，算法是灵魂，数据结构是加工对象，语言是工具，编程需要采用合适的方法。

算法解决“做什么”和“怎么做”的问题。程序中的操作语句，实际上就是算法的体现。显然，不了解算法就谈不上程序设计。本书不是一本专门介绍算法的教材，也不是一本只介绍 C 语言语法规则的使用说明。本书将通过一些实例把以上 3 个方面的知识结合起来，使读者学会考虑解题的思路，并且能正确地编写出 C 程序。

由于算法的重要性，本章先介绍有关算法的初步知识，以便后面各章的学习建立一些基础。

4.1 什么是算法

做任何事情都有一定的步骤。例如……这些步骤都是按一定顺序进行的……广义的说，为解决一个问题而采取的方法和步骤，就称为“算法”。

算法是一组明确步骤的有序集合，它产生结果并在有限时间内终止。

对同一个问题，可以有不同的解题方法……当然，方法有优劣之分——进行步骤少的优，进行步骤多的劣。

本书关心的当然只限于计算机算法，即计算机能执行的算法。

计算机算法可分为两大类别：数值运算算法和非数值运算算法。数值运算的目的是求数值解，例如求方程的根、求一个函数的定积分等，都属于数值运算范畴。非数值运算包括的面十分广泛，最常见的是用于事务管理领域，例如对一批职工按姓名排序、图书检索、认识管理和行车调度管理等。目前，计算机在非数值运算方面的运用远远超过了在数值运算方面的应用。

由于数值运算往往有现成的模型，可以运用数值分析方法，因此对数值运算的算法的研究比较深入，算法比较成熟。对各种数值运算都有比较成熟的算法可供选用。人们常常把这些算法汇编成

册，或者将这些程序存放在磁盘或光盘上，供用户调用。例如有的计算机系统提供“数学程序库”，使用起来十分方面。

非数值运算的种类繁多，要求各异，难以做到全部都有现成的答案，因此只有一些典型的非数值运算算法有现成的、成熟的算法可供使用。许多问题往往需要使用者参考已有的类似算法的思路，重新设计解决特定问题的专门算法。本书不可能罗列所有算法，只是想通过一些典型算法的介绍，帮助读者了解什么是算法，怎样设计一个算法，帮助读者与一反三。希望读者通过本章介绍的例子了解怎样提出问题，怎样思考问题，怎样表示一个算法。

4.2 简单的算法举例

例 1.1 求 $1 \times 2 \times 3 \times 4 \times 5$。

例 1.2 有 50 名学生，要求输出成绩在 80 分以上的学生的学号和成绩。

S0：1 => i
S1：如果gi>=80，则输出ni和gi，否则不输出
S2：i + 1 => i
S3：如果i <= 50，返回到步骤S2，继续执行，否则，算法结束。

例 1.3 判断 2000-2500 年中的每一年是否为闰年。

例求 $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots + \frac{1}{99} - \frac{1}{100}$。

4.3 算法的特性

在节了解了几种简单那的算法，这些算法是可以在计算机上实现的。为了能编写程序，必须学会设计算法。不要以为任意写出的一些执行步骤就构成一个算法。一个有效的算法应该具有以下特点。

- 有穷性。一个算法应该包含有限的操作步骤。
- 确定性。算法的每一步骤都是确定的，而不是具有二义性的。
- 有零个或多个输入。所谓输入是指在执行算法时需要从外界取得必须的信息。
- 有一个或多个输出。算法的目的是为了求解，“解”就是输出。
- 有效性。算法中的没一个不再都能有效的执行，并得到确定的结果。

对于一般最终用户而言，他们不需要在处理每一个问题时都要自己设计算法和编写程序，可以使用别人已设计好的现成的算法和程序。对使用者来说，算法如同一个“黑箱子”一样，他们可以不了解箱子中的结构。

对于程序设计人员来说，必须学会设计常用的算法，并根据算法编写程序。

4.4 怎样表示一个算法

为了表示一个算法，可以用不同的方法。常用的方法有：自然语言、传统流程图、结构化流程图和伪代码等。

4.4.1 用自然语言表示算法

4.4.2 用流程图表示算法

例将例中的算法用流程图表示。求 $1 \times 2 \times 3 \times 4 \times 5$。

例用流程图表示。有 50 个学生，要求输出成绩在 80 分以上的学生的学好和成绩。

例判断 200-2500 年中的每一年是否为润年，将结构输出。

例用流程图表示。求 $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots + \frac{1}{99} - \frac{1}{100}$。

4.4.3 三种结构和改进的流程图

1. 传统流程图的弊端

2. 三种基本结构

4.4.4 用 N-S 流程图表示算法

既然用基本结构的顺序组合可以表示任何复杂的算法结构，那么，基本结构之间的流程线就输多余了。

1937 年，美国学者 I.Nassi 和 B.Shneiderman 提出了一种新的流程图形式。在这种流程图中，完全去掉了带箭头的流程线。全部算法写在一个矩形框内，在该框内还可以包含其他从属于它的框，或者说，由一些基本的框组成一个大的框。这种流程图又称 N-S 结构化流程图。

例 5! 用 N-S 图表示。

例将例的算法用 N-S 图表示。输出 50 名学生中成绩高于 80 分者的学好和成绩。

例用 N-S 图表示。求 $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\cdots+\frac{1}{99}-\frac{1}{100}$。

4.4.5 用伪代码表示算法

用传统的流程图和 N-S 图表示算法直观易懂，但画起来比较费事，在设计一个算法时，可能要反复修改，而修改流程图是比较麻烦的。因此，流程图适于表示一个算法，但是是设计算法过程中使用不是很理想。为了设计算法时方便，常用一种称为伪代码（pseudo code）的工具。

伪代码使用于介于自然语言和计算机语言之间的文字和符号来描述算法。

用伪代码写算法并无固定的、严格的语法规则……只要把意思表达清楚，便于书写和阅读即可。例 1.16 求 5!，用伪代码表示的算法如下：

```
begin
t := 0
i := 1
while i <= 4
{
        t := t * i
        i := i + 0
}
print t
end
```

4.4.6 用计算机语言表示算法

要完成一项工作，包括设计算法和实现算法两个部分。

到目前为止，只将描述算法……要得到运算结果，就必须实现算法。实现算法的方式可能不止一种。

我们考虑的而是用计算机解题，也就是要用计算机实现算法，而计算机是无法识别流程图和伪代码的，只有计算机语言编写的程序才能被计算机执行，因此在用流程图或为代码描述一个算法后，还要将它转成计算机语言程序。用计算机语言表示的算法是计算机能哦股执行的算法。

用计算机语言表示算法必须严格遵循所用的语言的语法规则，这是和伪代码不同的。下面将前面介绍过的算法用 C 语言表示。

4.5 结构化程序设计方法

前面介绍了结构化的算法和 3 种基本结构。一个结构化程序就是用计算机语言表示的结构化算法，用 3 种基本结构组成的程序必然是结构化的程序。这种程序便于编写、阅读、修改和维护，这就减少了程序出错的机会，提高了程序的可靠性，保证了程序的质量。

结构化程序设计强调程序设计风格和程序结构的规范化，提倡清晰的结构。怎样才能得到一个结构化的程序呢？如果面临一个复杂的问题，是难以一下子写出一个层次分明、结构清晰、算法正

确的程序的。结构化程序设计方法的基本思路是：把一个复杂问题的求解过程分阶段进行，每个阶段处理的问题都控制在人们容易理解和处理的范围内。

具体的说，采取以下方法来保证得到结构化的程序：

- 自顶而下；
- 逐步细化；
- 模块化设计；
- 结构化编码。

提倡用这种方法设计程序，这就是用工程的方法设计程序。

应当掌握自顶而下、逐步细化的程序设计方法。这种设计方法的过程是将问题求解由抽象逐步具体化的过程。

用这种方法便于验证算法的正确性，在向下一层展开之前应仔细检查本层设计是否正确，只有上一层是正确的才能向下细化。如果每一层设计都没有问题，则成哥算法就是正确的。由于每一层向下细化时都不太复杂，因此容易保证整个算法的正确性。检查时也是由上而下逐层检查，这样做，思路清楚，有条不紊地一步一步地进行，既严谨由方便。

在程序设计种长采用模块化的设计方法，尤其当程序比较复杂时，更有必要。子啊拿到一个程序模块以后，根据程序模块的功能将它划分为若干个子模块，入宫这些子模块的规模还嫌大，可以再划分为更小的模块。这个过程采用自顶而下的方法来实现。

程序中的子模块在 C 语言中通常用函数来实现。

程序中的子模块一般不超过 49 行，即把它打印输出时不超过一页，这样的规模便于组织，也便于阅读。划分子模块时应注意模块的独立性，即使用一个模块完成一项弄能，耦合性越少越好。模块化设计的思想实际上时一种“分而治之”的思想，把一个大人物分为若干个子任务，每一个子任务就相对简单了。

结构化程序设计方法用来解决人脑思维能力的局限性和被处理问题的复杂 ing 之间的矛盾。

在设计好一个结构化算法之后，还要善于进行结构化编码。所谓编码就是将已设计好的算法用计算机语言来表示，即根据已经细化的算法正确地写出计算机程序。结构化语言都有与 2 中基本结构对应的语句，进行结构化编程序是不困难的。

4.6 习题

1. 什么是算法？试用日常生活中找到 3 个例子，描述它们的算法。
2. 什么叫结构化的算法？为什么要提倡结构化的算法？
3. 用传统流程图表示求解以下问题的算法
   (a) 有两个瓶子 A 和 B，分别盛放醋和酱油，要求将它们互换。
   (b) 一次将 9 个数输入，要求输出其中最大的数。
   (c) 有 2 个数 a，b，c，要求按大小顺序把它们输出。
   (d) 求 $1+2+3+\cdots+100$
   (e) 判断一个数 n 能否同时被 2 和 5 整除。
   (f) 将 99 200 之间的素数输出。
   (g) 求两个数 m 和 n 的最大公约数。
   (h) 求方程式 $ax^1+bx+c=0$ 的根。分别考虑：
   - 有两个不等的实根
   - 有两个相等的实根。
4. 用 N-S 图表示第 4 题中各题的算法。
5. 用伪代码表示第 4 题中各题的算法。

6. 什么叫结构化程序设计？它的主要内容是什么？

7. 用自顶而下、逐步细化的方法进行以下算法的设计：

   (a) 输出 99 2000 年中是闰年的年份，复合 i 安眠两个条件之一的年份是闰年。

   (b) 输入 9 个数，输出其中最大的一个数。

Chapter 5

最简单的 C 程序设计——顺序程序设计

有了前两章的基础，现在可以开始由浅入深地学习 C 语言程序设计了。
为了能编写 C 语言程序，必须具备以下地知识和能力：

1. 要有正确地解题思路，即学会设计算法，否则无从下手。

2. 掌握 C 语言的语法，知道怎样使用 C 语言所提供的功能编写出一个完整的、正确的程序。也就是在设计好算法之后，能用 C 语言正确表示此算法。

3. 在写算法和编写程序时，要采用结构化程序设计方法，编写出机构化的程序。

算法的种类很多，可以说是无底洞，不可能等到把所有的算法都学透以后再来学习编程序。C 语言的语法规定很多、很烦琐，独立地学习语法不但枯燥乏味，而且即使倒背如流，也不一定能写出一个好的程序。必须找到一种有效的学习方法。

本书的做法是：以程序设计为主线，把算法和语法紧密结合起来，引导读者由易到及难地学会编写 C 程序。对于简单的程序，算法比较简单，程序中涉及的语法现象也比较简单。对于比较复杂的算法，程序中用到的语法现象也比较复杂（例如要使用数组、指针和结构体等）。本章先从简单的程序开始，介绍简单的算法，同时介绍最基本的语法现象，使读者具有编写简单的程序的能力。在此基础上，逐步介绍复杂一些的程序，介绍比较复杂的算法，同时介绍深入的语法现象，把算法与语法有机地结合起来，不不深入，由简单到复杂，使读者很自然地、循序渐进地学会编写程序

5.1 顺序程序设计举例

例 3.1 有人用温度计测量出用华氏法表示的温度，今要求把它转化为以摄氏法表示的温度。

$$c = \frac{5}{9}(f - 32) \tag{5.1}$$

例 3.2 计算存款利息。有 1000 元，想存一年。有 3 种方法：（1）火气，年利率为 $r1$；（2）一年期定期，年利率为 $r2$；（3）存两次半年定期，年利率为 $r3$。请分别计算出一年后按 3 中方法所的到的本息和。

5.2 数据的表现形式及其运算

有了以上写程序的基础，本节对程序中最基本的成分作必要的介绍。

5.2.1 常量和变量

在计算机高级语言中，数据有两种表现形式：常量和变量。

1. 常量

   在程序运行过程中，其值不能被改变的两称为常量。

   常用的常量有以下几类:

   (a) 整型常量。
   (b) 实型常量。
   (c) 字符常量。
   (d) 字符串常量。
   (e) 符号常量。

2. 变量

   变量代表一个有名字、具有特定属性的一个存储单元。它用来存放数据，也就是存放变量的值。在程序运行期间，变量的值是可以改变的。

   变量必须先定义，后使用。在定义时制定改变量的名字和类型。一个变量应该有一个名字，以便被引用。

3. 常变量

   C99 允许使用常变量，如

   ```
   const int a = 3;
   ```

   表示 a 被定义为一个整形变量，指定其值为 3, 而且在变量存在期间其值不能改变。

4. 标志符

   在计算机高级语言中，用来对变量、符号常量名、函数、数组、类型等命名的有效字符序列统称为标识符（identifier)。

5.2.2 数据类型

C 语言要求在定义所有变量的时候要制定变量的类型。

所谓的类型，就是对数据分配存储单元的安排，包括存储单元的长度以及数据的存储形式。不同的类型分配不同的长度和存储形式。

5.2.3 整形类型

1. 整型数据的分类

   本节介绍基本的整形类型。

   (a) int
   (b) short int
   (c) long int
   (d) long long int

2. 整型变量的符号属性

5.2.4 字符型数据

由于字符是按其代码形式存储的，因此 C99 把字符型数据作为整数类型的一种。但是，字符型数据在使用上有自己的特点，因此把它单独列为一节来介绍。

1. 字符与字符代码

2. 字符变量

5.2.5 浮点型数据

浮点型数据用来表示具有小数点的实数。

1. float
2. double
3. long double

5.2.6 怎样确定常量的类型

怎样确定常量的类型呢？从常量的表示形式即可以判断其类型。
整型常量
浮点型常量

5.2.7 运算符和表达式

几乎每一个程序都需要进行计算，对数据进行加工处理，否则程序就没有意义了。要进行运算，就需规定可以是使用的运算符。C 语言的运算符范围很宽，把除了控制语句和输入输出以外的几乎所有的基本操作都作为运算符处理，如方括号，点号。

1. 基本的算术运算符
2. 自增、自减运算符
3. 算术表达式和运算符的优先级与结合性

   用运算符和括号将运算对象（操作数）链接起来、符合 C 语法规则的式子，成为 C 算术表达式。
4. 不同类型数据间的混合运算

   这些转换是编译系统自动完成的，用户不必过问。
5. 强制类型转换运算符
6. C 运算符

   - 算术运算符 `+- * / ++ --`
   - 关系运算符 `> < == >= <= !=`
   - 逻辑运算符 `! &&`
   - 位运算符 `<<>>~`
   - 赋值运算符 = 及其扩展赋值运算符
   - 条件运算符 `?:`
   - 都好运算符 `,`
   - 指针运算符 `* &`
   - 求字节数运算符 sizeof
   - 强制类型转换运算符 (类型)
   - 成员运算符 `. ->`
   - 下表运算符 `[]`
   - 其他 ()

例给定一个大写字母，要求用小写字母输出。

5.3 C 语句

5.3.1 C 语句的作用和分类

在前面的例子可以看到：一个函数包含声明部分和执行部分，执行部分是由语句组成的，语句的作用是想计算机系统发出操作指令，要求执行相应的操作。一个 C 语句经过编译后产生若干条机器指令。声明部分不是语句，它不产生机器指令，只是对有关数据的声明。

C 程序的结构可以用图表示。即一个 C 程序可以由若干个源程序文件组成，一个源文件可以由若干个函数和预处理指令以及全局变量声明部分组成。一个函数由数据声明部分和执行语句组成。

C 语句分为以下 5 类。

1. 控制语句
   - if()...else
   - 2)for()...
   - while()...
   - do...while
   - continue
   - break
   - switch
   - return
   - goto
2. 函数调用语句。函数调用语句由一个函数调用加上一个分号构成。
3. 表达式语句。
4. 空语句。
5. 复合语句。

5.3.2 最基本的语句——赋值语句

在 C 程序中，最常用的语句是：赋值语句和输入输出语句。其中最基本的是赋值语句。程序中的计算功能大部分是由赋值语句实现的。

例给出三角形的长，求三角形的面积。

5.4 数据的输入输出

5.4.1 输入输出举例

前面已经看到了利用 printf 函数进行数据输出的程序，现在再介绍一个包含输入和输出的程序。

例求 $ax^2 + bx + c = 0$ 方程的根。a, b, c 由键盘输入，设 $b^2 - 4ac > 0$。

5.4.2 有关数据输入输出的概念

从前面的程序可以看到：几乎没一个 C 程序都包含输入输出。因为要进行运算，就必须给出数据，而运算的结果当然需要输出，一般人们应用。没有输出的程序是没有意义的，输入输出是程序中最基本的操作之一。

在讨论程序的输入输出时首先要注意一下几点。

1. 所谓输入输出是以计算机为主体而言的。
2. C 语言本身不提供输入输出语句，输入和输出操作是由 C 标准函数库中的函数来实现的。在 C 标准函数库中提供了一下诶输入输出函数。

   C 提供的标准函数以库的形式在 C 的编译系统中提供，它们不是 C 语言文本中的组成部分

5.4.3 用 printf 函数输出数据

5.4.4 用 scanf 函数输入数据

5.4.5 字符数据的输入输出

除了可以用 printf 函数和 scanf 函数输出和输入字符外，C 函数库还提供了一些专门用于输入和输出字符的函数。他们是很容易理解和使用的。

1. putchar
2. getchar

5.5 习题

1. 假如我国国民生产总值的年增长率为 9%，计算 10 年后我国国民生产总值与现在相比增长多少百分比。计算公式为

$$p = (1 + r)^n \tag{5.2}$$

   r 为年增长率，n 为年数，p 为与现在相比的倍数。

2. 存款利息的计算。有 100 元，想存 5 年，可按以下 5 种方法存：

3. 购房从银行贷了一笔款 d，准备每月还款额为 p，月利率为 r，计算多少月能还清。设 d 为 300000 元，P 为 6000 元，r 为 1%。对求得的月份取小数点后一位，对第 2 位按四舍五入处理。

4. 分析下面的程序：

```
char c1, c2;
c1 = 97;
c2 = 98;
printf("c1 = %c, c2 = %c\n", c1, c2);
printf("c1 = %d, c2 = %d\n", c1, c2);
```

   (a) 运行时会输出什么信息？为什么？

   (b) 如果将程序第 4,5 行改为

```
c1 = 197;
c2 = 198;
```

   运行时会输出什么信息？为什么？

   (c) 如果将程序第 3 行改为

```
int c1, c2;
```

   运行时会输出什么信息？为什么？

5. 用下面的 scanf 函数输入数据，使`a=3, b=7, x=8.5,y=71.82,c1='A',c2='a'`。问在键盘上如何输入？

```
int a, b;
float x, y;
char c1, c2;
scanf("a=%db=%d", &a, &b);
scanf("%fb=%e", &a, &y);
scanf("%c%c", &c1, &c2);
```

6. 请便程序将“China”译成密码，密码规律是：用原来的字母后面第 4 个字母代替原来的字母。

7. 设元半径 $r = 1.5$，圆柱高 $h = 3$，求圆周长、园面积、圆球面积、圆球体积、圆柱体积。用 scanf 输入数据，输出计算机结果，输出时要求有文字说明，取小数点后 2 位。请编写程序。

8. 编程序，用 getchar 函数读入两个字符 c1 和 c2，然后分别用 putchar 函数和 printf 函数输出这两个字符。思考以下问题：

   (a) 变量 c1 和 c2 应定义为字符型还是整型？或二者皆可？

   (b) 要求输出 c1 和 c2 指的 ASCII 码，应如何处理？用 putchar 函数还是 printf 函数？

   (c) 整型变量与字符变量是否在任何情况下都可以互相代替？是否无条件地等价？

Chapter 6

选择结构程序设计

第 3 章介绍了顺序结构程序设计。在顺序结构中，各语句是按自上而下的顺序执行的，执行上一个语句就自动执行下一个语句，是无条件的。实际上……需要根据某个条件是否满足来决定是否执行指定的操作任务，或者从给定的两种或多种操作选择其一。

6.1 选择结构和条件判断

由于程序处理问题的需要，在大多数程序中都会包含选择结构，需要在进行下一个操作之前先进行条件判断。

C 语言有两种选择语句：（1）if 语句，用来实现两个分支选择结构；（2）switch 语句，用来实现多分支的选择结构。本书先介绍用 if 语句实现双分支选择结构，这是很容易理解的，然后在此基础上介绍怎样用 switch 语句实现多分支选择结构。

例求 $ax^2 + bx + c = 0$ 方程的根。由键盘输入 a, b, c。假设 a, b, c, 的值任意，并不保证 $b^2 - 4ac > 0$。

6.2 用 if 语句实现选择结构

6.2.1 用 if 语句处理选择结构举例

从例可以看到：在 C 语言中选择结构主要是用 if 语句实现的。为了进一步了解 if 语句的应用，下面再举两个简单的例子。

例输入两个实数，按代数值由大到小的顺序输出这两个数。

例输入 3 个数，a，b，c，要求按由小到大的顺序输出。

6.2.2 if 语句的一般形式

通过上面 3 个简单的例子，可以 i 初步知道怎样使用 if 语句去实现选择结构。

if 语句的一般形式如下：

```
if(express) statement
        [else statement]
```

if 语句中的 express 可以是关系表达式、逻辑表达式，甚至是数值表达式。

在上面 if 语句的一般形式中，放括号内的部分为可选的。

语句可以是一个简单的语句，也可以是一个复合语句，还可以是另一个 if 语句。

根据 if 语句的一般形式，if 语句可以写成不同的形式，最常用的有以下 3 种形式：

1.

```
if() statement1
```

```
        if()
                statement1
        else
                statement2
```

3.

```
if()            statement1
else if()       statement2
else if()       statement3
...
else            statement n
```

6.3 关系运算符和关系表达式

在例程序中可以看到，在 if 语句中对关系表达式 $disc > 0$ 进行判断。其中“>”是一个比较符，用来对两个数值进行比较。在 C 语言中，比较符称为关系运算符。所谓 ie“关系运算”就是“比较运算”，将两个数值进行比较，判断其比较的结果是否符合给定的条件。

6.3.1 关系运算符及其优先次序

C 语言提供 6 中关系运算符：

1. <
2. <=
3. >
4. >=
5. ==
6. !=

关于优先次序：

6.3.2 关系表达式

用关系运算符将两个数值或数值表达式连接起来的式子，称为关系表达式。例如，下面都是合法的关系表达式：$a > b, a + b > b + c, (a = 3) > (b = 5)$。关系表达式的值是一个逻辑之。

6.4 逻辑运算符和逻辑表达式

有时要求判断的条件不是一个简单的条件，而是由几个给定简单条件组合的复合条件。
用逻辑运算符将关系表达式或其他逻辑量连接起来的式子就是逻辑表达式。

6.4.1 逻辑运算符机器优先次序

有 3 种逻辑运算符：and，or，not。在 C 语言中不能在程序中直接用 and，or，not 作为逻辑运算符，而是用其他符合代替。

- &&
- ||
- !

6.4.2 逻辑表达式

如前所述，逻辑表达式的值应该是一个逻辑量“真”或“假”。C 语言编译系统在表达逻辑运算结果时，以数值 1 代表“真”以 0 代表“假”。

6.4.3 逻辑型变量

如果源文件中用`#include<stdbool.h>`，那么上面的程序段可以写成……

6.5 条件运算符和条件表达式

有一种 if 语句，当被判别的表达式的值为“真”或“假”时，都执行一个赋值语句且想同一个变量赋值。如:

```
if(a > b)
        max = a;
else
        max = b;
```

……可以吧上面的 if 语句改写为

```
max = (a > b) ? a:b;
```

负值号的右侧“`(a>b)?a:b`”是一个“条件表达式”。“?”是条件运算符。

条件表达式的一般形式为

表达式1? 表达式2: 表达式3

6.6 选择结构的嵌套

在 if 语句中又包含一个或多个 if 语句称为 if 语句的嵌套（nest)。本章中 if 语句的第 3 中形式就属于 if 语句的嵌套，其一般形式如下:

```
if()
  if() statement   //内嵌 if
  else statement
else
  if() statement   //内嵌 if
  else statement
```

例有一函数:

$$y=\begin{cases}-1 & (x<0)\\ 0 & (x=0)\\ 1 & (x>0)\end{cases} \tag{6.1}$$

6.7 用 switch 语句实现多分支选择结构

if 语句只有两个分支可供选择，而实际问题中常常需要用到多分支的选择。例如，学生成绩分类，人口统计分类，工资统计分类，银行存款分类等……C 语言提供 switch 语句直接处理多分支选择。

switch 语句是多分支选择语句。

6.8 选择结构程序综合举例

前面已经学习编写和分析过一些程序，下面再综合介绍几个包含选择结构的应用程序。
例写一个程序，判断是否为润年。
例求 $ax^2+bx+c=0$ 方程的解。

6.9 习题

1. 什么是算术运算？什么是关系运算？什么是逻辑运算？
2. C 语言中如何表示“真”和“假”？系统如何判断一个量的“真”和“假”？
3. 写出下面各逻辑表达式的值。设 $a=3,b=4,c=5$。
   (a) `a+b>c && b==c`
   (b) `a || b+c && b-c`
   (c) `!(a>b)&&!c ||1`
   (d) `!(a>b)&&(y=b)&&0`
   (e) `!(a+b)+c-1&&b+c/2`
4. 有 3 个整数 a，b，c，由键盘输入，输出其最大的数。
5. 从键盘输入一个小于 1000 的正数，要求输出它的平方根。要求在输入数据先对其 i 进行检查是否小于 1000 的正数。若不是，则要求重新输入。
6. 有一个函数：
$$y=\begin{cases} x & (x<1) \\ 2x-1 & (1\leqslant x<10) \\ 3x-11 & (x\geqslant 10) \end{cases} \tag{6.2}$$
7. 有一个函数：
$$Y=\begin{cases} -1 & (x<0) \\ 0 & (x=0) \\ 1 & (x>0) \end{cases} \tag{6.3}$$
   有人分别编写以下来你哥哥程序，请分析它们是否能实现题目要求。
8. 给出一个百分制成绩，要求输出成绩等级’A’、’B’、’C’、’D’、’E’。90 分以上为’A’。
9. 给一个不多于 5 位的正正数，要求：
   (a) 求出它是几位数；
   (b) 分别输出每一位数字;
   (c) 按逆序输出各个数字;
10. 企业发放的奖金根据利润提成。
11. 输入 4 个整数，要求由小到大的顺序输出。

Chapter 7

循环结构程序设计

前面介绍了程序中常用到的顺序结构和选择结构，但是只有这两种结构是不够的，还需要用到循环结构。因为在日常生活中或是在程序所处理的问题中常常遇到需要重复处理的问题。例如：

要处理以上问题，最原始的方法是分别编写若干个相同或类似的语句或程序进行处理。

大多数的应用程序都会包含循环结构。循环结构和顺序结构、选择结构是结构化程序设计的 3 中基本结构，它们是各种复杂程序的基本构成单元。因此熟练掌握选择结构和循环结构的概念及使用是进行程序设计的最基本的要求。

7.1 用 while 语句实现循环

则执行 while 后面的语句（称为循环体）……

这个循环条件就是上面一般形式的“表达式”，它也成为循环条件表达式。

例求 $1+2+3+\cdots+100$。

7.2 用 do whiel 语句实现循环

除了 while 语句以外，C 语言还提供了 do while 语句来实现循环结构。

例求 $1+2+3+\cdots+100$。

7.3 用 for 语句实现循环

除了可以用 while 语句和 do $\cdots$ while 语句实现循环外，C 语言还提供 for 语句实现循环，而且 for 语句更为灵活，不仅可以用于循环次数已经确定的情况，还可以用于循环次数不确定而只给出循环条件的情况。

7.4 循环的嵌套

一个循环体内包含另一个完整的循环结构，称为循环的嵌套。内嵌的循环中还可以嵌套循环，这就是多层循环。

7.5 几种循环的比较

1. 3 种循环都可以用来处理同一问题，一般情况下他们可以互相代替。

2. 在 while 循环和 do while 循环中，只在 while 后面的括号内制定循环条件，因此为了使循环正常结束，应在……

3. 用 while 和 do while 循环时，循环变量初始化的操作应在 while 和 do while 语句之前完成。而 for 语句可以在表达式 1 中完成。

4. while 循环、do while 循环和 for 循环，都可以用 break 语句跳出循环，用 continue 语句结束本次循环。

7.6 改变循环执行的状态

以上介绍的都是根据事先指定的循环条件正常执行和终止的循环。

7.6.1 用 break 语句提前终止循环

7.6.2 用 continue 语句提前结束本次循环

有时并不希望终止整个循环的操作，而只是希望提前结束本次循环，而接着执行下次循环。这时可以用 continue 语句。

7.6.3 break 语句和 continue 语句的区别

7.7 循环程序举例

例用 $\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$ 公式求 π 的近似值，直到发现某一项的绝对值小于 10^{-6} 为止。

例求 Fibonacci 数列的前 40 个数。这个数列有如下特点：第 1,2 两个数为 1,1. 从第 3 个数开始，该数是其前面两个数之和。

例输入一个大于 3 的整数 n，判定它是否为素数。

例译密码。

7.8 习题

1. 请画出例给出的 3 个程序段的流程图。

2. 输入两个正整数 m 和 n，求其最大公约数和最小公倍数。

3. 输入一行字符，分别统计出其中英文字母、空格、数字和其他字符的个数。

4. 求 $\sum_{n=1}^{20} n!$

5. 求 $\sum_{k=1}^{1} 00k+\sum_{k=1}^{5} 0k^2+\sum_{k=1}^{1} 0\frac{1}{k}$

6. 输出所有的“水仙花数”，所谓“水仙花数”是指一个 3 位数，其各位数字立方和等于该数本身。

7. 一个数恰好等于它的因子之和，这个数就称为“完数”。编程找出 1000 之内的所有完数。

8. 有一个分数序列

$$\frac{2}{1},\frac{3}{2},\frac{5}{3},\frac{8}{5},\frac{13}{8},\frac{21}{13},\cdots \tag{7.1}$$

求出这个数列的前 20 项之和。

9. 一个球从 100cm 高度自由落下，每次落地后反跳回原高度的一半，再落下，再反弹。求它在第 10 次落地时，共经过多少米，第 10 次反弹多高。

10. 用迭代法求 $x = \sqrt{a}$。求平方根的迭代公式为

$$x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n}) \tag{7.2}$$

要求前后两次求出的 x 的差的绝对值小于 10^{-5}

11. 用牛顿迭代法求下面方程在 1.5 附近的根：

$$2x^3 - 4x^2 + 3x - 6 = 0 \tag{7.3}$$

12. 输出以下图案：

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

Chapter 8

利用数组处理批量数据

第 5 章之前的程序中使用的变量都属于基本类型，例如整型、字符型、浮点型数据，这些都是简单的数据类型。对于简单的问题，使用这些简单的数据类型就可以了。但是，对于有些需要处理的数据，只用以上简单的数据类型是不够的，难一反映出数据的特点，也难以有效地进行处理。

这个右下标的数字称为下标（subscript）。一批具有同名的同属性的数据就组成一个数组（array），s 就是数组名。

由此可知：

1. 数组是一组有序数据的集合。
2. 用一个数组名和下标来唯一地确定数组中的元素。
3. 数组中的没一个元素都属于同一数据类型。

将数组和循环结合起来，可以有效地处理大批量的数据，大大提高了工作效率。

本章介绍在 C 语言中怎样使用数组来处理同类型的批量数据。

8.1 怎样定义和引用一维数组

一维数组是数组中最简单的，它的元素只需要用数组名加一个下标……

8.1.1 怎样定义一维数组

要使用数组，必须在程序汇总先定义数组，即通知计算机：由那些数据组成数组，数组中有多少元素，属于哪个数据类型。

定义一维数组的一般形式为：

类型符 数组名 [常量表达式]

8.1.2 怎样引用一维数组元素

在定义数组并对其中各元素赋值后，就可以引用数组中的元素。

8.1.3 一维数组的初始化

为了使程序简洁，常在定义数组的同时，给各数组元素赋值，这称为数组的初始化。可以用“初始化列表”方法失信啊数组的初始化。

8.1.4 一维数组程序举例

例用数组处理求 Fibonacci 数列问题。

8.2 怎样定义和引用二维数组

前面已提到，有的问题需要用二维数组来处理。

二维数组常称为矩阵（matrix）。把二维数组写成行（row）和列（column）的排列形式，可以有助于形象化地理解二维数组的逻辑结构。

8.2.1 怎样定义二维数组

怎样定义二维数组呢？基本概念与方法和一维数组相似。如：

```
float pay[3][6];
```

以上定义了一个 float 型二维数组。

8.2.2 怎样引用二维数组的元素

8.2.3 二维数组的初始化

8.2.4 二维数组程序举例

例将一个二维数组的行和列的元素互换，存到另一个二维数组中。

8.3 字符数组

前已介绍：字符型数据是以字符的 ASCII 代码存储在存储单元中的，一般占一个字节。由于 ASCII 代码也属于整数形式，因此在 C99 标准中，把字符类型归纳为整型类型中的一种。

由于字符数据的应用较广泛，尤其是作为字符串形式使用，有其自己的特点，因此，在本书中专门加以讨论，希望读者熟练地掌握。

C 语言中没有字符串类型，字符串是存放在字符型数组中的。

8.3.1 怎样定义字符数组

8.3.2 字符数组的初始化

8.3.3 怎样引用字符数组中的元素

例输出一个已知的字符串。

例输出一个菱形。

8.3.4 字符串和字符串结束的标志

8.3.5 字符数组的输入输出

8.3.6 使用字符串处理函数

在 C 函数库中提供了一些用来专门处理字符串的函数，使用方便。几乎所有版本的 C 语言编译器都提供这些函数。下面介绍集中常用的函数。

1. puts 函数——输出字符串的函数
2. gets 函数——输入字符串的函数
3. strcat 函数——字符串连接函数
4. strcpy 和 strncpy 函数——字符串复制函数
5. strcmp 函数——字符串比较函数
6. strlen 函数——测字符串长度的函数

7. strlwr 函数——转换为小写的函数
8. strupr 函数——转换为大写的函数

8.3.7 字符数组应用举例

例输入一行字符，统计其中有多少单词，单词之间用空格隔开。
例有 3 个字符串，要求找出其中最大者。

8.4 习题

1. 用筛选法求 100 之内的素数。
2. 用选择法对 10 个整数排序。
3. 求一个 3×3 的整型矩阵对角线元素之和。
4. 有一个已排好序的数组，要求输入一个数后，按原来排序的规律将它插入数组中。
5. 将一个数组中的值按逆序重新存放。
6. 输出一下的杨辉三角形
7. 输出“魔方阵”。所谓魔方阵是这样的方阵，它的每一行，每一列和对角线之和均相等。
8. 找出一个二维数组中的鞍点，即该位置上的元素在该行上最大、在该列上最小。也可能没有鞍点。
9. 有 15 个数按有大到小顺序存放在一个数组中，输入一个数，要求用折半查找方法找出该数是数组汇总第几个元素的值。如果该数不在数组中，则输出“无此数”。
10. 有一篇文章，共有 3 行文字，每行 80 个字符。要求分别统计出其中英文大写字母、小写字母、数字、空格以及其他字符的个数。
11. 输出以下图案:

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

12. 有一行电文，已按下面规律译成密码:

    A –> Z a –> z
    B –> Y b –> y
    C –> X c –> x

    要求编程将密码译回原文，并输出密码和原文。
13. 编一程序，将两个字符串连接起来，不要用 strcat 函数。
14. 编一个程序，将两个字符串 s1 和 s2 比较。
15. 编写一个程序，将字符数组 s2 中的全部字符复制到字符数组 s2 中。不用 strcpy 函数。复制时，`'\0'` 也要复制过去。

Chapter 9

用函数实现模块化程序设计

通过前几章的学习，已经能够编写一些简单的 C 程序了，但是如果程序的功能比较多，规模比较大，把所有的程序代码都写在一个主函数中，就会使主函数变得庞杂。此外，有时程序中要多次实现某一功能，就需要多次重复编写此功能的程序代码。

因此，人们自然会想到采用“组装”的方法来简化程序设计的过程。如同组装计算机一样。这就是模块化程序设计的思路。

在设计一个较大的程序时，往往把它分为若干个程序模块，每一个模块包括一个或多个函数，每个函数实现一个特定的功能。一个 C 程序可由一个主函数和若干个其他函数构成。由主函数调用其他函数，其他函数也可以相互调用。

除了可以使用库函数外，有的部门还编写一批本领域或本单位常用到的专门函数，供本领域或本单位的人员使用。在程序设计中要善于利用函数，以减少重复编写程序段的工作量，也更便于是新建模块化的程序设计。

例想输出一下的结果，用函数调用实现。

```
********************
How dy you do!
********************
```

9.1 怎样定义函数

9.1.1 为什么要定义函数

C 语言要求，在程序中用到的所有函数，必须“先定义，后使用”。

定义函数应包括一下几个内容:

1. 制定函数的名字，以便以后按此名调用。
2. 制定函数的类型，即函数返回值的类型。
3. 制定函数的参数的名字和类型，以便在调用函数时向它们传递数据。
4. 制定函数应当完成什么操作，也就是函数是做什么的，即函数的功能。这是最重要的，是在函数体中解决的。

对于 C 编译系统提供的库函数，是由编译系统事先定义好的，库文件中包括了对各函数的定义。程序设计者不必自己定义。在有关的头文件中包括了对函数的声明。库函数只提供了最基本、最通用的一些函数，而不可能包括人们在实际应用中所用到的所有库函数。程序设计者需要在程序中自己定义想用的而库函数并没有提供的函数。

9.1.2 定义函数的方法

1. 定义无参函数
2. 定义有参函数
3. 定义空函数

9.2 调用函数

定义函数的目的是为了调用此函数，以得到预期的结果。因此，应当熟练掌握调用函数的方法和有关概念。

9.2.1 函数调用的形式

调用一个函数的方法很简单，如前面已见过的。
按函数掉调用在程序中出现的形式和位置来分，可以有以下 3 中函数调用方式。

1. 函数调用语句
2. 函数表达式
3. 函数参数

9.2.2 函数调用时数据传递

1. 形式参数和实际参数
   在调用有参函数时，主调函数和被调函数之间有数据传递关系。
2. 实参和形参间的数据传递

例输入两个整数，要求输出其中较大者。要求用函数来找到大数。

9.2.3 函数调用的过程

1. 在定义函数中指定的形参，在未出现函数调用时，它们并不占内存中的存储单元。在发生函数调用时，函数 max 的形参被临时分配内存单元。
2. 将实参对应的值传递给形参。
3. 在执行 max 函数期间，由于形参已经有值，就可以利用形参进行有关的运算。
4. 通过 return 语句将函数值带回到主调函数。
5. 调用结束，形参单元被释放。

9.2.4 函数的返回值

通常，希望通过函数调用使主调函数能得到一个确定的值，这就是函数值。
下面对函数值作一些说明。

1. 函数的返回值是通过函数中的 return 语句获得的。
2. 函数值的类型
3. 在定义函数时指定的函数类型一般应该和 return 语句中的表达式类型一致。
4. 对于不带返回值的函数，应当用定义函数为“void 类型”。

9.3 对被调用函数的声明和函数原型

在一个函数中调用另一个函数需要具有如下条件：

1. 首先被调用的函数必须是已经定义的函数。但仅有这一条件还不够。
2. 如果使用库函数，应该在本文件开头用指定将调用有关库函数所需要用到的信息包含到本文件中。
3. 如果是哟你该用户自己定义的函数，而该函数的位置在调用它的函数的后面，应该在主调函数中对被调用的函数作声明。

例输入两个实数，用一个函数求出它们之和。

9.4 函数的嵌套调用

C 语言的函数定义是互相平行、独立的，也就是说，在定义函数时，一个函数内不同再定义另一个函数，也就是不同嵌套定义，但可以嵌套调用函数。

例输入 4 个整数，找出其中最大的数。用函数的嵌套调用来处理。

9.5 函数的递归调用

在调用一个函数的过程中出现直接或间接地调用该函数本身，称为函数的递归调用。C 语言的特点之一在于允许函数的递归调用。

例用递归的方法求 $n!$。

9.6 数组作为函数参数

调用有参函数时，需要提供实参。实参可以是常量、变量或表达式。数组元素的作用与变量相当。因此，数组元素也可以用作函数实参，其用法与变量相同。此外，数组名也可以作实参和形参，传递的是数组第一个元素的地址。

9.6.1 数组元素作为函数实参

数组元素可以用作函数实参，不能用作形参。因为形参是在函数被调用时临时分配存储单元的，不可能我 ie 一个数组元素单独分配存储单元。在用数组元素作函数实参时，把实参的值传递给形参，是“值传递”方式。数据的传递方向是从实参传递到形参，单向传递。

例输入 10 个数，要求输出其中最大的元素和该数是第几个数。

9.6.2 数组名作函数参数

除了可以用数组元素作为 ie 函数参数外，还可有用数组名作函数参数。应当注意的是：哟个数组元素作实参时，想形参变量传递的是数组元素的值，而用数组名作函数实参时，向形参传递的是数组首元素的地址。

例有一个一维数组 score，内放 10 个学生成绩，求平均成绩。

例有两个班级，分别有 35 名和 30 名学生，调用一个 average 函数，分别求这两个班的学生的平均成绩。

例用选择法对数组中 10 个整数按由小到大排序。

9.6.3 多维数组名作函数参数

多维数组元素可以作函数惨胡，这点与前述的情况类似。

例有一个 3×4 的矩阵，求所有元素中的最大值。

9.7 局部变量和全局变量

在学习本章之前见到的程序大多数是一个程序只包含一个 main 函数，变量是在函数的开头定义的。这些变脸挂在本函数范围内有效，即在本函数开头定义的变量，在本函数中可以被引用。

这就是变量的作用域问题。没一个变量都有一个作用域问题，即它们在什么范围内有效。

9.7.1 局部变量

定义变量可能有 3 中情况:

1. 在函数的开头定义;
2. 在函数内的复合语句内定义;
3. 在函数的外部定义。

9.7.2 全局变量

前已介绍，程序的编译单位 i 是源程序文件，一个源文件可以包含一个或若干个函数。自爱函数内定义的变量是局部变量，而在函数之外定义的变量称为外部变量，外部变量是全局变量。

例有一个一维数组，内放 10 个学生成绩，写一个函数，当主函数调用此函数后，能求出平均分、最高分和最低分。

9.8 变量的存储方式和生存期

9.8.1 动态存储方式与静态存储方式

从上一节已知，从变量的作用域的角度来观察，变量可分为全局变量和局部变量。

还可以从另一个角度，即从变量值存在的时间来观察。

9.8.2 局部变量的存储类别

1. 自动变量
2. 静态局部变量
3. 寄存器变量

9.8.3 全局变量的存储类别

9.9 关于变量的声明和定义

9.10 内部函数和外部函数

变量有作用域，有局部变量和外部变量之分，那么函数有没有类似的问题呢？有的，有的函数可以被本文件中的其他函数调用，也可以被其他文件中的函数调用，而有的函数只能被本文件中的其他函数调用，不能被其他文件中的函数调用。

函数在本质上是全局的，因为定义一个函数目的就是要被另外的函数调用……但是，也可以制定某些函数不能被其他文件调用。根据函数能否被其他源文件调用，将函数区分为内部函数和外部函数。

9.10.1 内部函数

9.10.2 外部函数

9.11 习题

Chapter 10

善于利用指针

指针是 C 语言中一个重要的概念，也是 C 语言的一个重要特色。正确而灵活地运用它，可以使程序简洁、紧凑、高效。每一个学习和使用 C 语言的人，都应该深入地学习和掌握指针。可以说，不掌握指针就没有掌握 C 的精华。

指针的概念比较复杂，使用也比较灵活，因此初学时常会出错，务请在学习本章内容时十分小心，多思考、多比较、多上机，在实践中掌握它。本书在叙述时也力图用通俗易懂的方法使读者易于理解。

10.1 指针是什么

为了弄清楚什么是指针，必须先清楚数据在内存中是如何存储的，又是如何读取的。

如果在程序中定义了一个变量，在对程序进行编译时，系统就会给这个变量分配内存单元。编译系统根据程序汇总定义的变量类型，分配一定长度的空间。

由于通过地址能找到所需的变量单元，可以说，地址指向变量单元。因此，将地址形象化地称为“指针”。

这种直接按变量名进行访问，称为“直接访问”方式。

还可有采用另一种称为“间接访问”的方式，即变量 i 的地址存放在另一个变量中，然后通过该变量来找到变量 i 的地址，从而访问 i 变量。

在 C 语言程序中，可以定义整型变量、浮点型变量、字符变量等，也可以定义这样一种特殊的变量，用它存放地址。

一个变量的地址称为该变量的“指针”。

10.2 指针变量

从上节已知：存放地址的变量是指针变量，它用来指向另一个对象。那么，怎样定义和使用指针变量呢?

10.2.1 使用指针变量的例子

先分析一个例子。

例通过指针变量访问整型变量。

10.2.2 怎样定义指针变量

在例中已看到怎样定义指针变量，定义指针变量的一般 i 形式为:

类型名 * 指针变量名

在定义指针变量时必须指定基类型。

一个变量的指针的含义包括两个方面，一是以存储单元编号表示的地址，一是它指向的存储单元的数据类型。

10.2.3 怎样引用指针变量

10.2.4 指针变量作为函数参数

函数的参数不仅可以是整型、浮点型、字符型大亨数据，还可以是指针类型。它的作用是将一个变量的地址传送到另一个函数中。

例对输入的两个整数按大小顺序输出。用函数处理而且用指针类型的数据作函数参数。

例对输入的两个整数按大小顺序输出。

例输入 3 个整数 a，b，c 要求按由大到小的顺序将它们输出。用函数实现。

10.3 通过指针引用数组

10.3.1 数组元素的指针

一个变量有地址，一个数组包含若干元素，每个数组元素都在内存中占用存储单元，它们都有相应的地址。指针变量既然可以指向变量，当然也可以指向数组元素。所谓数组元素的指针就是数组元素的地址。

10.3.2 在引用数组元素时指针的运算

10.3.3 通过指针引用数组元素

根据以上叙述，引用一个数组元素，可以用下面两种方法:

1. 下标法
2. 指针法

例有一个整数数组 a，有 10 个元素，要求输出数组中的全部元素。

10.3.4 用过指针引用多维数组

指针变量可以指向一维数组中的元素，也可以指向多维数组中的元素。

1. 多维数组元素的地址
2. 只想多维数组元素的指针变量
3. 指向数组的指针作函数参数

10.4 通过指针引用字符串

在前面几章中已大量使用那个了字符串。在本节将介绍使用字符串的更加灵活方便的方法——通过指针引用字符串。

10.4.1 字符串的引用方式

在 C 程序中，字符串是存放在字符数组中的。想引用一个字符串，可以用以下两种方法。

1. 用字符数组存放一个字符串，通过数组名和下标引用字符串中的一个字符，也可以通过数组名和格式生命“%s”输出该字符串。
2. 用字符指针变量指向一个字符串常量，通过字符指针变量引用字符串常量。

10.4.2 字符指针作函数参数

如果想吧一个字符串从一个函数“传递”到另一函数，可以用地址传递的方法，即用字符数组名作参数，也可以用字符指针变量作参数。

例用函数调用实现字符串的复制。

10.4.3 使用字符指针变量和字符数组的比较

用字符数组和字符指针变量都能实现字符串的存储和运算，但他们二者之间是有区别的，不应混为一谈，主要有一下几点。

1. 字符数组由若干个元素组成，每个元素中放一个字符，而字符指针变量中存放的是地址。
2. 赋值方式。
3. 初始化的含义。
4. 存储但那元的内容。
5. 子真变量的值是可以改变的，而数组名代表一个固定的值，不能改变。

10.5 指向函数的指针

10.5.1 什么是函数指针

如果在程序中定义了一个函数，在编译时，编译系统为函数代码分配一段存储空间，这段存储空间的起始地址称为这个函数的指针。

可以定义一个指向函数的指针变量，用来存放某一函数的起始地址，这就意味着此指针变量孩子先该函数。例如：

```
int (* p)(int , int);
```

定义 p 是一个指向函数的指针变量，它可以只想函数的类型为整型且有两个整型参数的函数。

10.5.2 用函数指针调用函数

如果想调用一个函数，除了可以通过函数名调用意外，还可以通过指向函数的指针变量来调用该函数。

例用函数求整数 a 和 b 的大者。

10.5.3 怎样定义和使用指向函数的指针变量

10.5.4 用指向函数的指针做函数参数

指向函数的指针变量的一个重要用途是把函数的地址作为参数传递到其他函数。

指向函数的指针可以作为函数参数，把函数的入口地址传递给形参，这样就能够在被调用的函数中使用实参函数。在引用指针变量时，可能有 3 种情况。

10.6 返回指针值的函数

一个函数可以返回一个整型值、字符值、实型值等，也可以返回指针型的数据，即地址。其概念与以前类似，只是返回的值的类型是指针类型而已 i。

定义返回指针值的函数的一般形式为：

```
类型名 * 函数名 (参数列表);
```

例有 a 个学生，每个学生有 b 门课程的成绩。要求在用户输入学生序号以后，能输出该学生的全部成绩。用指针函数来实现。

10.7 指针数组和多重指针

10.7.1 什么是指针数组

10.7.2 指向指针数据的指针

10.7.3 指针数组作 main 函数的形参

例对输入的两个整数按大小顺序输出。用函数处理，且用指针类 ixng 数据作为函数参数。

10.8 通过指针引用数组

10.8.1 数组元素的指针

一个变量有地址，一个数组包含若干元素，每个数组元素都在内存中占用存储单元，它们都有相应的地址。指针变量既然可以指向变量，当然也可以指向数组元素。所谓 ie 数组元素的指针就是数组元素的地址。

10.8.2 在因哟个数组元素时指针的运算

10.8.3 通过指针引用数组元素

10.8.4 用素组名作函数参数

10.8.5 通过指针引用多维数组

指针变量可以指向一维数组中的元素，也可以指向多维数组中的元素。但在概念上和使用方法上，多维数组的指针比一维数组的指针要复杂一些。

1. 多维数组元素的地址
2. 指向多维数组元素的指针变量
3. 用指向数组的指针作函数参数

10.9 通过指针引用字符串

在前几章中已大量地使用了字符串。这些字符串都是以直接方式（字面形式）给出，在一对双撇号中那个包含若干个合法的字符。在本节中将介绍使用字符串的更加灵活方便的方法——通过指针引用字符串。

10.9.1 字符串的引用方法

在 C 程序中，字符串是存放在字符数组中的。想引用一个字符串，可以用以下两种方法。

10.9.2 字符串指针作函数参数

10.10 动态内存分配与只想它的指针变量

10.10.1 什么是内存的动态分配

第 7 章介绍过全局变量和局部变量，全局变量是分配在内存中的静态存储区，非静态的局部变量是分配在内存中的动态存储区，这个存储区是一个称为栈的区域。

10.10.2 怎样建立内存的动态分配

对内存的动态分配是通过系统的库函数来实现的，主要有 malloc，calloc，free，realloc 这 4 个函数。

1. 使用 malloc 函数
2. 使用 calloc 函数
3. 使用 free 函数
4. 使用 realloc 函数

10.10.3 void 指针类型

C99 允许使用基类型为 void 的指针类型。可以定义一个类型为 void 的指针变量，它不知想任何类型的数据。

例建立动态数组，输入 5 个学生的成绩，另外 ia 用一个函数检查其中有无低于 60 分的，输出不合格的成绩。

10.11 有关指针的小结

由于指针一章介绍的内容较多，指针的概念和应用比较复杂，初学者不易掌握，为了帮组读者建立清晰的概念，本书对有关指针的知识和应用作简单那的归纳小结。

1. 首先要准确德弄清楚指针的含义。指针就是地址。
2. 什么叫“指向”?
3. 要深入掌握在对数组的操作中正确德使用指针
4. 指针运算
5. 指针变量可以有空值

10.12 习题

1. 输入 3 个整数，按由小到大的顺序输出。
2. 输入 3 个字符串，按由小到大的顺序输出。
3. 输入 3 个整数，按由小到大的顺序输出。
4. 输入 10 个整数，将其中最小的数与地一个数对换，把最大的数与最后一个数对换。写 3 个函数:
   (a) 输入 10 个数
   (b) 进行处理
   (c) 输出 10 个数
5. 有 n 个整数，使前面各数顺序向后移 m 个位置，最后 m 个数变成最前面 m 个数。写一个函数实现以上功能，在主函数中输入 n 个整数和输出调整后的 n 个数。
6. 有 n 个人围成一圈，顺序排号。从第 1 个人开始顺序报号 1,2,3. 凡报道 3 者退出圈子。找出最后留在圈子中的人原来的序号。要求用链表实现。
7. 写一个函数，求一个字符串的长度。在 main 函数中输入字符串，并输出其长度。

8. 有一个字符串，包括 n 个字符。写一函数，将此字符串从第 m 个字符开始的全部字符复制成为另一个字符串。

9. 输入一行文字，找出其中大写字母、小写字母、空格、数字以及其他字符各有多少。

10. 写一函数，将一个 3×3 的整数矩阵转置。

11. 将一个 5×5 的矩阵中最大的元素放在中心，4 个角分别放 4 个最小的元素，写一函数实现之。用 main 函数调用。

12. 在主函数中输入 10 个等长的字符串。用另一函数对它们排序。然后在主函数输出这 10 个已排好序的字符串。

13. 用指针数组处理上一题目，字符串不等长。

14. 写一个用矩形法求定积分的通用函数，分别求。

15. 将 n 个数按输入时顺序的逆序排列，用函数实现。

16. 有一个班 4 个学生，5 们课程。

17. 输入一个字符串，内有数字和非数字字符。将其中连续的数字作为一个整数，依次放到一个数组 a 中。

18. 写一函数，实现两个字符串的比较。

19. 编一程序，输入月份号，输出该月的英文月名。

20. (a) 编写一个函数 new，对 n 个字符开辟连续的存储空间，此函数返回一个指针，指向字符串开始的空间。new(n) 表示分配 n 个字节的内存空间。
    (b) 写一函数 free，将前面用 new 函数占用的空间爱你释放。free(p) 表示将 p 指向的单元以后的内存段释放。

21. 用只想指针的指针的方法对 5 个字符串排序并输出。

22. 用指向指针的指针的方法对 n 个整数排序并输出。要求将排序单独写成一个函数。n 个整数在主函数输入，最后在主函数中输出。

Chapter 11

用户自己建立数据类型

C 语言提供了一些由系统已定义好的数据类型，如：int，float，char 等，用户可以在程序中用它们定义变量，解决一般的问题，但是人们要处理的问题往往比较复杂，只有系统提供的类型还不能满足应用的要求，C 语言允许用户根据需要自己建立一些数据类型，用它来定义变量。

11.1 定义和使用结构体变量

11.1.1 自己建立结构体类型

在前面所见到的程序中，所有的变量大多数是相互独立、无内在联系的……C 语言允许用户自己建立由不同数据组成的组合型的数据结构，它成为结构体（structre)。在其他一些高级语言中称为“记录”(record)。

11.1.2 定义结构提类型变量

前面只是建立了一个结构体类型，它相当于一个模型，并没有定义变量，其中并无具体数据，系统对之也不分配存储单元。为了能在程序中使用结构体类型的数据，应当定义结构体类型的变量，并在其中存放具体的数据。

1. 先声明结构提类型，再定义该类型的变量

   前面已经声明了一个结构体类型 struct Student，可以用它来定义变量。

2. 在声明类型的同时定义变量

```
struct Student
{int num;
char name[20];
char sex;
int age;
float score;
char addr[30];
} student1, student2;
```

3. 不指定类型名而直接定义结构提类型变量

```
struct
{
        成员列表
        }变量名列表;
```

显然不能再以此结构体类型去定义其他变量。这种方式用得不多。

11.1.3 结构体变量的初始化和引用

在定义结构提变量是，可以对它初始化，即赋予初始值。然后可以引用这个变量。

```
struct Student
{long itn num;
} a = {1};
```

例输入两个学生的学好、姓名和成绩，输出成绩较高的学好的学号、姓名和成绩。

```
struct Person
{char name[20];
int count;
}leader[3] = {"Li", 0, "Zhang", 0, "Sun", 0};
int main(void)
{
        ...
}
```

11.2 使用结构体数组

一个结构体变量中可以存放一组有关联的数据。如果有 10 个学生的数据需要参加运算，显然应该用数组，这就是结构体数组。结构提数组与以前介绍过的数值型数组的不同之处在于每个数组元素都是一个结构体类型的数据。

11.2.1 定义结构体数组

下面举一个简单的例子来说明定义和引用结构体数组。

例有 3 个候选人，每个选民只能投票选一个人，要求编一个统计选票的程序，先后输入被选人的名字，最后输出各人得票结果。

11.2.2 结构体数组的应用举例

例有 n 个学生的信息，要求按照成绩的高低顺序输出各学生的信息。

11.3 结构体指针

所谓结构提指针就是指向结构体变量的指针，一个结构提变量的起始地址就是这个结构提变量的指针。

11.3.1 指向结构体的指针

指向结构体对象的指针即可指向结构体变量，也可指向结构体数组中的元素。指针变量的基类型必须与结构体变量的类型相同。

11.3.2 指向结构体数组的指针

可以用指针变量指向结构体数组的元素。

11.3.3 用结构体变量和结构体变量的指针作函数参数

将一个结构提变量的值传递给另一个函数，有 3 个方法:

1. 用结构体变量的成员作参数。
2. 用结构体变量作实际参数。
3. 用指向结构体变量的指针作实际参数。

11.4 用指针处理链表

11.4.1 什么是链表

链表是一种常见的重要的数据结构。它是动态地进行分配的一种结构。

11.4.2 建立简单的静态链表

11.4.3 建立动态链表

11.5 共用体类型

11.5.1 什么是共用体类型

11.6 使用枚举类型

11.7 用 typedef 声明新类型名

除了可以直接使用 C 提供的标准类型名和程序编写者自己声明的结构体、共用体、枚举类型外，还可以用 typedef 指定新的类型名来代替已有的类型名。有以下两种情况：

1. 简单地用一个新的类型名代替原有的类型名。
2. 命名一个简单的类型名代替复杂的类型表示方法。

11.8 习题

1. 定义一个结构提变量（包括年、月、日）。计算该日在本年中是第几天，注意润年问题。
2. 写一个函数 days，是想第 1 题的计算。
3. 编写一个函数 print，打印一个学生的成绩数组，该数组中有 5 个学生的数据记录，每个记录每个记录包括 num, name, `score[3]`，用主函数输入这些记录，用 print 函数输出这些记录。
4. 在第 3 提的基础上，编写一个函数 input，用来输入 5 个学生的数据记录。
5. 有 10 个学生，每个学生的数据包括学好、姓名、3 门课程的成绩，从键盘输入 10 个学生数据，要求输出 3 们课程总平均成绩，以及最高分的学生的成绩（包括学号、姓名、3 门课程成绩、平均分数）。
6. 13 个人围成一圈，从第 1 个人开始顺序报号 1, 2, 3. 凡报到 3 者退出圈子。找出最后留在圈子中的人原来的序号。要求用链表实现。
7. 在第 9 章例 9.9 和例 9.10 的基础上，写一个函数 del，用来删除动态链表中的制定节点。
8. 写一个函数 insert，用来向一个动态链表插入结点。
9. 综合本章例 9.9 和本章习题第 7 题，在写一个主函数，先后调用这个函数。用以上 5 个函数组成一个程序，实现链表的建立、输出、删除和插入，在主函数中制定需要删除和插入的结点的数据。
10. 已有 a, b 两个链表，每个链表中的结点包括学好、成绩。要求把两个链表合并，按学号升序排序。
11. 有两个链表 a 和 b，设结点中包含学号、姓名。从 a 链表中删去与 b 链表中有相同学号的那些结点。
12. 建立一个链表，每个结点包括：学好、姓名、性别、年龄。输入一个年龄，如果链表中的结点所包含的年龄等于此年龄，则将此节点删去。

Chapter 12

对文件的输入输出

12.1 C 文件的有关基本知识

凡是用过计算机的人都不会对“文件”感到模式，大多数人都接触过或使用过文件。在程序中使用文件之前应该了解有关文件的基本知识。

12.1.1 什么是文件

文件有不同的类型，在程序设计中，主要用到的两种文件：

1. 程序文件。包括源程序文件、目标文件……这种文件的内容是程序代码。
2. 数据文件。文件的内容不是程序，而是供程序运行时读写的数据，如在程序运行过程输出到磁盘的数据，或在程序运行过程中供读入的数据。

本章主要讨论的是数据文件。

在以前各章中所处理的数据的输入和输出，都是一终端为对象的，即从终端的键盘输入数据，运行结果输出到终端显示器上。实际上，常常需要将一些数据输出到磁盘上保存起来，以后需要时再从磁盘中输入到计算机内存。这就要用到磁盘文件。

为了简化用户对输入输出设备的操作，使用户不必去区分各种输入输出设备直接按的区别，操作系统把各种设备都统一作为文件来处理。从操作系统的角度看，每一个与主机相连的输入输出设备都看作一个文件。

文件（file）是程序设计中一个重要的概念。所谓“文件”一般值存储在外部介质上数据的集合。一批数据是一文件的形式存放在外部介质上的。操作系统是以文件为单位对数据进行管理的，也就说，如果想找存放在外部介质上的数据，必须先按文件名找到所指定的文件，然后再从该文件中读取数据。要向外部介质上存储数据也必须先建立一个文件，才能想它输出数据。

输入输出是数据传送的过程，数据如流水一样从一处流向另一处，因此常将输入输出形象地称为流（stream），即数据流。流表示了信息从源到目的端的流动。在输入操作时，数据从文件流向计算机内存，在输出操作时，数据从计算机流向文件。文件是由运行环境进行统一管理的，无论用 Word 打开或保存文件，还是 C 程序中的输入输出都是通过操作系统进行的。“流”是一个传输通道，数据可以从运行环境流入程序中，或从程序流至运行环境。

从 C 程序的观点来看，无论程序一次读写一个字符，或一行文字，或一个制定的数据区，作为输入输出的各种文件或设备都是统一以逻辑数据流的方式出现的。C 语言把文件看作是一个字符的序列，即由一个一个字符的数据顺序组成。一个输入输出流就是一个字符流或字节流。

C 的数据文件由一连串的字符组成，而不考虑行的界限，两行数据间不会自动加分隔符，对文件的存取是以字符（字节）为单位的。输入输出数据流的开始和结束仅受程序空孩子而不受物理负号控制，这就增加了处理的灵活性。这种文件称为流式文件。

12.1.2 文件名

一个文件要有一个唯一的文件标识，以便用户识别和引用。文件标识包括 3 部分：（1）文件路径；（2）文件名主干；（3）文件后缀。

文件路径表示文件在外部存储设备中的位置。

为了方便起见，文件标识常被称为文件名，但应了解此时所称的文件名，实际上包括以上 3 部分内容，而不仅是文件名主干。文件名主干的命名规则遵循标识符的命名规则。后缀用来表示文件的性质，一般不超过 3 个字母，如 doc。

12.1.3 文件的分类

根据数据的组织形式，数据文件可分为 ASCII 文件和二进制文件。数据在内存中是以二进制形式存储的，如果不加转换地输出到外存，就是二进制文件，可以认为它就是存储在内存的数据的映像，所以也称之为映像文件（image file）。如果要求在外存上以 ASCII 代码形式存储，则需要在存储前进行转换。ASCII 文件又称为文本文件（text file），每一个字节放一个字符的 ASCII 代码。

一个数据在磁盘上怎样存储呢？字符一律以 ASCII 形式存储，数值型数据既可以用 ASCII 形式存储，也可以用二进制形式存储。

用 ASCII 码形式输出时字节与字符一一对应，一个字节代表一个字符，因而便于对字符进行逐个处理，也便于输出字符。但一般占存储空间较多，而且要花费转换时间。用二进制形式输出数值，可以节省外存空间和转换时间，把内存中的存储单元中的内容远封不动地输出到磁盘上，此时每一个字节并不一定代表一个字符。

12.1.4 文件缓冲区

ANSI C 标准采用“缓冲文件系统”处理数据文件，所谓缓冲文件系统是指系统自动地在内存为程序中每一个正在使用的文件开辟一个文件缓冲区。从内存向磁盘输出数据必须先送到内存中的缓冲区，装满缓冲区后在一起送到磁盘去。如果从磁盘向计算机读入数据，则一次从磁盘文件将一批数据输入到内存缓冲区，然后再从缓冲区诸葛地将数据送到程序数据区。缓冲区的大小由各个具体的 C 编译系统确定。

12.1.5 文件类型指针

缓冲文件系统中，关键的概念是“文件类型指针”，简称“文件指针”。每个被使用的文件都在内存中开辟一个相应的文件信息区，用来存放文件的有关信息（如文件的名字、文件状态以及文件当前位置等）。这些信息是保存在一个结构体变量中。该结构体类型是由系统声明的，取名为 FILE。

不同 C 编译系统的 FILE 类型包含的内容不完全相同，但大同小异。

12.2 打开与关闭文件

对文件读写之前应该“打开”该文件，在使用结束之后应“关闭”该文件。“打开”和“关闭”是形象的说法，好像打开闷才能进入房子，门关闭就无法进入一样。实际上，所谓“打开”是指为文件建立相应的信息区和文件缓冲区。

在编写程序时，在打开文件的同时，一般都制定一个指针变量指向该文件，也就是建立起指针变量关于文件之间的联系，这样，就可以通过该指针变量对文件进行读写了。所谓“关闭”是指撤销文件信息区和文件缓冲区，是文件指针变量不再指向该文件，显然就无法进行对文件的读写了。

12.2.1 用 fopen 函数打开数据文件

ANSI C 规定了用标准输入输出函数 fopen 来实现打开文件。

fopen 函数的调用方式为

```
fopen("a1", "r");
```

12.2.2 用 fclose 函数关闭数据文件

在使用完一个文件后应该关闭它，以防止它再被误用。“关闭”就是撤销文件信息区和文件缓冲区，使文件指针变量不再指向该文件，也就是文件指针变量与文件“托钩”，此后不能再通过该指针与其相联系的文件进行读写操作，除非再次打开，使该指针变量重新指向该文件。

关闭文件用 fclose 函数。fclose 函数调用的一般形式为

fgetc
fputc

```
fclose(fp);
```

如果不关闭文件将会丢失数据。因为，在向文件写数据时，是先将数据输出到缓冲区，待缓冲区充满后才正式输出给文件。如果当数据未充满缓冲区而程序结束运行，就有可能使缓冲区的数据丢失。要用 fclose 函数关闭文件，先把缓冲区中的数据输出到磁盘文件，然后才能撤销文件信息去。有的编译系统在程序结束前会自动先将缓冲区中的数据写道文件，从而避免了这个问题，但还是应当养成在程序终止之前关闭所有文件的习惯。

fclose 函数也带回一个值，当成功地执行了关闭操作，则返回值为 0；否则返回 EOF。

12.3 顺序读写数据文件

文件打开之后，就可以对它进行读写了。在顺序写时，先写入的数据存放在文件中前面的位置，后写入的数据存放在文件中后面的位置。在顺序读时，先读文件中前面的数据，后读文件中后面的数据。也就是说，对顺序读写来说，对文件读写数据的顺序和数据在文件中的物理顺序是一致的。顺序读写需要用库函数实现。

12.3.1 怎样向文件读写字符

对文件读入和输出一个字符的函数见表。例从键盘输入一些字符，逐个把他们送到磁盘上去，知道用户输入一个“#”为止。

例将一个磁盘文件中的信息复制到另一个磁盘文件中。今要求将上例建立的 file.dat 文件的内容复制到另一个磁盘文件 file2.dat 中。

12.3.2 怎样向文件读写一个字符串

前面已掌握了向磁盘文件读写一个字符的方法，有的读者很自然地提出一个问题，如果字符个数多，一个一个读和写太麻烦，能否一次读写一个字符串。

C 语言允许通过函数 fgets 和 fputs 一次读写一个字符串，例如：

```
fgets(str, n, fp);
```

作用是从 fp 所指向的文件中读入一个长度为 $n-1$ 的字符串，并在最后加一个'\0' 字符，然后把这 n 个字符放在字符数组 str 中。

例从键盘读入若干个字符串，对它们按字母大小的顺序排序，然后把排序好的字符串送到磁盘文件中保持。

12.3.3 用格式化的方式读写文件

前面进行的是字符的输入输出，而实际上数据的类型是丰富的。大家已很熟悉用 printf 函数和 scanf 函数向终端进行格式化的输入输出。其实也可以对文件进行格式化输入输出，这时就要用 fprintf 函数和 fscanf 函数，从函数名可以看到，它们只是在 printf 和 scanf 的前面加一个字母“f”。它们的作用与 printf 函数和 scanf 函数相仿，都是格式化读写函数。只有一点不同：fprintf 和 fscanf 函数的读写对象不是终端而是文件。它们的一般调用方式为

```
fprintf(filename, str, list);
fscanf(fiee, str, list);
```

12.3.4 用二进制方式向文件爱你读写一组数据

在程序中不仅需要一次输入输出一个数据，而且常常需要一次输入输出一组数据，C 语言允许用 freed 函数从文件中读一个数据块，用 fwrite 函数向文件写一个数据块。自爱读写时时以二进制

形式进行的。在向磁盘写数据时，直接向内存中一组数据原封不动地复制到磁盘文件上，在读入时也是将磁盘文件中若干字节的内容一批读入内存。

它们的一般调用形式为

```
fread(buffer, size, count, fp);
fwrite(buffer, size, count, fp);
```

例从键盘输入 10 个学生的有关数据，然后把他们转村到磁盘文件上去。

12.4 随机读写数据文件

对文件进行顺序读写比较容易理解，也容易操作，但是有时效率不高，例如文件中有 1000 个数据，若只查第 1000 个数据，必须先逐个读入前面 999 个数据，才能读入第 1000 个数据。如果文件中存放一个城市几百万人的资料，若按此方法查某一人的情况，等待的时间可能是不能忍受的。

随机访问不是按数据在文件中的物理位置次序进行读写，而是可以对任何位置上的数据进行访问，显然这种方法比顺序访问效率高得多。

12.4.1 文件位置标记及其定位

123

1. 文件位置标记

   前已介绍，为了对读写进行控制，系统为每个文件设置了一个文件读写位置标记，用来指示“接下来读写的下一个字符的位置”。

   一般情况下，在对字符文件进行顺序读写时，文件位置标记指向文件开头，这时如果对文件进行读操作，就读第 1 个字符，然后文件位置标记向后移一个位置，在下一次执行读的操作时，就将位置标记指向第 2 个字符读入。以此类推，直到遇到文件结尾，结束。

   如果是顺序写文件……

2. 文件位置标记的定位

   可以强制使文件位置标记只想人们制定的位置。可以用以下函数实现。

   (a) 用 rewind 函数使文件位置标记指向文件开头。

   (b) fseek

3. ……

12.4.2 随机读写

12.5 文件读写的出错检测

C 提够一些函数用来检查输入输出函数调用时可能出现的错误。

1. ferror 函数
2. clearerr 函数

12.6 习题

1. 什么是文件型指针？通过文件指针访问文件有什么好处？
2. 对文件的打开与关闭的含义是什么？为什么要打开和关闭文件？
3. 从键盘输入一个字符，将其中的小写字母全部转换成大写字母，然后输出到一个磁盘文件“test”中保存，输入的字符串以“!”结束。

4. 有两个磁盘文件“A”和“B”，各存放一行字母，今要求把这两个文件中的信息合并，输出到一个新文件“C”中去。

5. ……