第3章 顺序结构程序设计

程序设计是一个复杂而精细的过程。早期的程序设计是自由的、技巧性很强的个性化活动，编写的程序往往晦涩难懂，不易维护，可靠性差，由程序错误而引起的信息丢失、系统报废事件屡有发生，这种现象导致了20世纪60年代末爆发的软件危机。从那时开始，人们开始反思程序设计的规律和方法，并着手对软件开发方法和软件生产管理进行研究。模块化和结构化程序设计方法就是在这种背景下产生的。结构化程序设计方法是最早的程序设计方法之一，对于某个求解问题，在进行结构化程序设计时，人们首先从整体的角度考虑问题，将问题分割成若干个逻辑上相互独立的模块，然后分别实现，最后再把这些独立模块组装起来。用结构化程序设计方法得到的程序不仅结构良好、清晰易读，而且易维护、易排错、易于正确性验证，在一定程度上缓解了软件危机，改善了软件开发的状况。更重要的是，它向人们揭示了研究程序设计方法的重要性，并为后来的程序设计方法奠定了基础。

本章主要介绍结构化程序设计方法、程序的三种基本结构、C语句的概念和输入输出函数的使用方法等，最后介绍顺序程序设计的基本方法。

3.1 结构化程序设计方法

用计算机处理问题，编写程序只是其中一个步骤，而算法设计是整个程序设计的核心。如果算法不当，程序编写技艺再高也得不到正确的结果，而不同的算法对程序运行的效率和结果的精度会产生极大的影响，程序的质量主要由算法决定。

面对一个较复杂的求解问题，不要急于编写程序，应遵循问题分析、算法设计、流程描述的顺序做好准备工作，然后再着手编写程序。由于本书所举的示例程序较短，算法也相对简单，因此没有严格按照上述步骤进行。

什么样的算法是好的？判断程序好坏的标准又是什么？

在计算机发展初期，计算机内存小，运算速度慢。程序设计追求代码短小、精练、运算速度快，即效率第一。程序设计采用的手工方式，全凭编程者的个人技巧和爱好，这往往导致程序晦涩难懂，难以检查。随着计算机的飞速发展，计算机速度越来越快，内存容量越来越大，对效率的苛求有所缓解。更何况程序的效率主要是由算法确定，编程的技巧并不能对程序的效率产生决定性的影响。另一方面，程序的规模越来越大，软件开发采用的是团队化、规模化的生产方式。随之而来的是出错的可能性大了，出错所带来的后果也越来越严重，甚至是灾难性的。这时，判断程序好坏的标准就从效率第一变成要求程序有良好的可读性，以

提高程序设计的质量,便于查错和维护,减少软件设计的成本。即把程序的可靠性与可维护性摆在了首要位置。

为了从根本上保证程序的正确与可靠,1968 年,著名计算机科学家 E. W. Dijkstra 指出了程序设计中过去常用的 goto 语句的三大危害,反对滥用 goto 语句,代之以软件生产方式的科学化、规范化、工程化,并由此产生了结构化程序设计方法和“软件工程”概念。

结构化程序设计方法是一套指导软件开发的方法,涵盖了系统分析、系统设计和程序设计三方面的内容。结构化的程序设计采用自顶向下、逐步细化、模块化的方法进行程序设计。它强调程序设计风格和程序结构的规范化,提倡清晰的程序结构。结构化程序设计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。具体实现步骤为

(1) 按自顶向下逐步求精的方法对问题进行分析、设计;

(2) 系统的模块设计;

(3) 结构化编码。

1. 自顶向下分析设计问题

由于人的思维能力有限,人们对问题规模的驾驭能力就受到限制,结构化程序设计方法是解决人脑思维能力的局限性与所处理问题的复杂性之间矛盾的一个有效办法。自顶向下、逐步求精的结构化程序设计方法,可以把大的复杂问题分解成若干个小问题,然后各个击破。

自顶向下、逐步求精就是对一个复杂问题,首先进行上层(整体)的分析与设计,按其组织或功能将问题分解成若干个子问题,如果所有的子问题都得到了解决,整个问题就解决了。而解决子问题无论在规模上还是在复杂性上都大大低于原问题。如果子问题仍然十分复杂,再对它进一步分解,如此一层一层地分解下去,直到处理对象相对简单,容易处理为止。每一次分解都是对上一层进行细化,逐步求精,最终形成一种层次结构(树形结构),能精确地描述问题及问题的处理方式。

例如,想为图书馆开发一个图书馆管理系统,首先按其功能把整个管理分为 4 个模块,即图书登录、借书、还书和预约。一旦这 4 个功能都能实现,连接起来就形成了图书馆管理系统。这 4 个模块还比较复杂,难于直接实现,可以进一步分解每一个模块。图 3.1.1 给出了图书馆管理系统设计的层次结构图。图中的每个方框都是程序设计中的模块,在 C 语言中用函数实现,相互之间的连接线就是函数之间的调用关系。

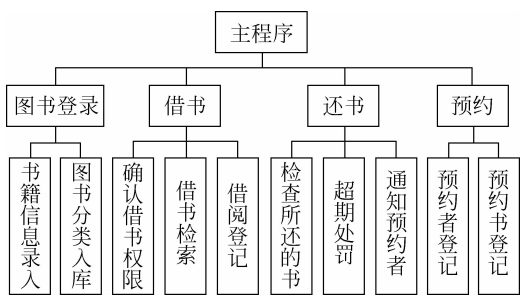


图 3.1.1 图书馆管理系统设计的层次结构

按照自顶向下方法设计系统，有助于各模块的设计、调试测试以及系统最终按层次集成。

2. 模块化程序设计

通过按自顶向下、逐步求精方法，可把复杂问题分解成许多容易解决的小问题，每个小问题就是系统中的一个模块。每个模块用一个程序模块实现，再把这些小程序模块像搭积木那样合成起来，形成解决整个复杂问题的大程序。因此，程序的模块化就是把程序划分成若干个模块，每个模块完成一个特定的功能。把这些模块综合起来组成一个整体，就可以完成指定问题的全部功能要求。

在设计每一个具体程序模块时，程序模块中包含的语句一般不要超过 50 行，这样既便于程序员的思考与设计，也利于程序的阅读。一个模块应具有良好的独立性，使得程序模块的编写、调试都可以独自完成，尽量减少模块之间的相互影响，以免带来相互间的干扰。在 C 语言中，模块用函数实现，一个模块对应一个函数。如果模块功能较复杂，可以进一步调用低一层的模块函数，以实现结构化的程序设计思想。程序的模块化的另一个好处是有助于软件开发工程的组织管理，一个复杂的大型程序可以由许多程序员分工编写不同的模块，从而加快软件开发的速度，缩短开发周期。

当一个软件经模块化设计后，每一个模块可以独立编程。业已证明，任何只含有一个入口和一个出口的程序均可由顺序结构、选择结构或循环结构组成。不仅程序本身是单入口单出口的，而且程序的每一局部结构也应是单入口单出口。从结构上讲，进入顺序结构、选择结构和循环结构是单入口的，当执行完该结构离开时，也必定是单出口的。因此，在结构化程序设计时，要尽量采用 3 种基本程序控制结构，尽量少用或不用类似 goto 这样的语句，因为这类语句会破坏程序整体的结构性，使程序结构变得复杂，降低了程序的可读性。

在 C 语言中，函数的重要性是不言而喻的。作为构成模块化结构的最小单位，函数是独立的程序单元。函数一次定义，可以多次调用，实现代码重用。多个函数还可以组织在一起构成所谓的函数库。

3.2　语句的概念

语句是 C 程序的基本功能单元，一个 C 程序应当包含若干语句。和其他高级语言一样，C 语言的语句用于向计算机系统发出操作指令。程序的每一个语句都意味着为完成某一任务而进行处理的动作。通常简单的语句表示一种单一功能的操作，而复合语句、选择语句和循环语句等复杂语句可能代表多个操作组成的一种复杂操作功能。语句的意义称为该语句的语义。

C 程序的结构可以用图 3.2.1 表示，即一个 C 程序可以由若干源程序文件组成，一个源程序文件可以由若干个函数和预处理命令组成，一个函数由数据定义和执行语句序列组成。

C 程序一般都包括数据描述和数据操作，数据描述定义数据类型和初值，数据操作完成对数据的加工处理。

组成 C 程序的语句包括表达式语句、控制语句、复合语句和空语句等。

C 语句以分号“；”作为结束标志。一般每个编辑行写一个语句，但在相邻的语句都较

短时,也可以在一行中写多个语句;当语句比较长时,可以把一个语句写成多行。

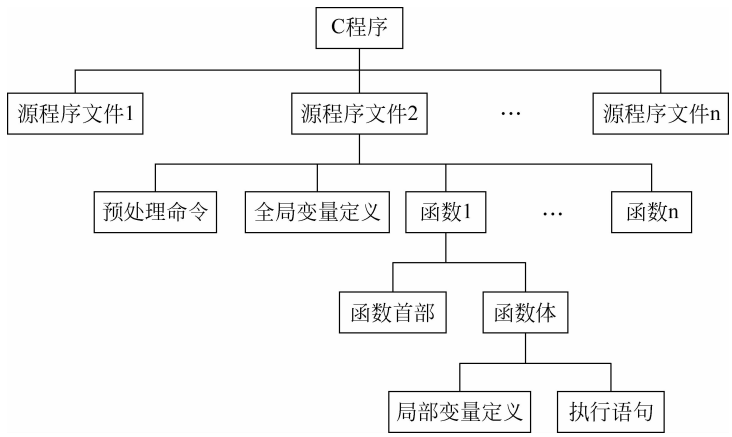


图 3.2.1 C 程序组成结构示意图

1. 表达式语句

在表达式的末尾加上分号";"就构成了表达式语句,表达式语句实现对数据的处理。C 程序中大多数语句是表达式语句,所以有人把 C 语言称作"表达式语言"。最典型、最常用的表达式语句是赋值语句和函数调用语句等,例如:

```
i = 1, j = 2, k = 3;                          //赋值语句
y = 5 * sqrt(27.0) + 3;
i++;
fun(j, j + k, 6);                             //函数调用语句
printf("This is a C program.\n ");            //函数调用语句
```

2. 控制语句

C 程序是按照函数中语句的书写顺序执行的,对于大多数应用程序来说,只有自上而下的顺序执行显然是不够的,往往需要在多个动作中选择其中之一,或能重复执行一系列步骤。而控制语句可以改变程序的执行顺序,使程序中语句的执行顺序与书写顺序不一致。C 语言提供了 9 种控制语句,包括:

(1) 选择语句:if_else 语句(双分支选择语句)、switch 语句(多分支选择语句);

(2) 循环语句:while 语句、do_while 语句、for 语句;

(3) 转向语句:break 语句(中止执行 switch 或循环语句)、continue 语句(结束本次循环语句)、goto 语句(无条件转向语句)和 return 语句(从函数返回语句)等。

这些语句的语法、含义和运用是学习编程的重点,分别在第 4 章和第 5 章进行详细介绍。

3. 复合语句

复合语句(也称为块语句),由大括号{}把若干个语句括起来组成。例如,{temp=x; x=y; y=temp;}就是一个复合语句。

复合语句在语法上相当于一个语句。因此，在C程序中，当需要把若干个语句作为一个语句使用时，可以使用复合语句。复合语句作为一个语句又可以出现在其他复合语句中，即复合语句是可以嵌套的。

复合语句内的各个语句都必须以分号“;”结束，但在复合语句结束标志右大括号}后面则不必加分号。

复合语句主要用于以下两种情形：

(1) 语法上要求用一个语句，但实际需要由多个语句才能完成操作。例如if语句的内嵌语句或for语句、while语句的循环体。

(2) 形成局部化的封装体。在C程序中，函数和块语句都是局部化的封装体，例如在块语句中定义的变量只能在本块范围内使用。

4. 空语句

只有一个分号“;”的语句称为空语句。事实上，它也可看成是一个特殊的表达式语句，但不做任何操作。其作用是用于语法上需要一条语句的地方，而该地方又不需做任何操作，例如空语句可以用作循环语句中的循环体：

```
for (i = 1; i <= 100000; i++);
```

这样的循环语句表示循环体什么也不执行，用于程序中需要“耗时”的地方。

3.3　赋值语句

在赋值表达式的末尾加上一个分号可构成赋值语句。由于赋值语句广泛运用，本节专门加以讨论。赋值语句的一般形式为

```
变量 = 表达式;
```

其中，赋值号“=”左边是变量名，“=”右边是表达式。例如，“x=x+5”是赋值表达式，而“x=x+5;”则是赋值语句。

赋值语句是C语言程序设计中最基本的、最常用的语句，兼有表达式计算和赋值的双重功能，即计算赋值运算符“=”右边表达式的值，并将该值赋给“=”左边的变量。

注意给变量赋初值和赋值语句二者的区别。例如，

```
int a = 5;                                  //变量定义并赋初值
a = 5;                                      //赋值语句
```

下面的变量定义方式是错误的：

```
int a = b = c = 5;
```

必须写为

```
int a = 5,b = 5,c = 5;
```

而语句：

```
a = b = c = 5;
```

则是合法的。

[**例 3.3.1**] 交换变量 x 和 y 的值。

解 两个人交换座位,只要两人同时起立,各自去坐对方的位置即可,这种交换是直接交换。而要想将一杯水和一杯可乐互换,不能直接从一个杯子倒入另一个杯子,必须借助一个空杯子,先把水倒入空杯,再将可乐倒入已空的水杯,最后把水倒入已空的可乐杯,才能实现水和可乐的交换,这是间接交换。计算机的内存类似于杯子这样的“容器”,且有“取之不尽、一冲就走”的特点,故程序设计中交换两个变量的值必须借助于第三个变量,进行间接交换,如图 3.3.1 所示。程序代码如下:

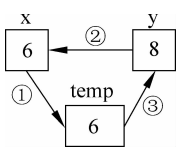


图 3.3.1 相互交换两个变量值

```
#include <stdio.h>
void main()
{
   int  x = 6, y = 8, temp;
   printf("Before: x = %d, y = %d\n",x,y);
   temp = x;                                    //将 x 的初值赋予变量 temp
   x = y;                                       //仅改变变量 x 的值,y 的值不变
   y = temp;                                    //变量 y 被赋予新的值,原值被覆盖
   printf("After: x = %d, y = %d\n", x,y);
}
```

运行结果:

```
Before: x = 6,y = 8
After: x = 8,y = 6
```

此例告诉我们:

(1) 每个变量在某一时刻只能存放一个值。一旦对变量进行赋值,变量将保存该值,直至被重新赋值;

(2) 若将一个变量 y 的值赋予变量 x,则 x 和 y 二者有相同的值,且 y 变量的值不会消失。

[**例 3.3.2**] 输入一个 4 位整数,然后打印出它的 4 位数字之和。

解 可以通过恰当的算术表达式计算得到一个整数的各位数字。程序代码如下:

```
#include <stdio.h>
void main()
{  int  n,a,b,c,d,sum;
   printf("Input n:\n");
   scanf ("%d", &n );
   a = n%10;                                    //求个位数
   b = n/10%10;                                 //求十位数
   c = n/100%10;                                //求百位数
   d = n/1000;                                  //求千位数
   sum = a + b + c + d;
```

```
    printf("n = %d, sum = %d\n", n, sum);
}
```

运行结果：

```
Input n:2568 <回车>
n = 2568, sum = 21
```

3.4　输入输出函数

程序一般可以由三部分组成：输入初始数据、计算处理和输出结果。数据的输入与输出是程序与用户之间的交互界面。C 语言本身不提供输入输出语句，所有的数据输入输出都是由库函数完成。在使用 C 语言库函数时，要用编译预处理命令 #include<stdio.h>将 stdio.h 包含到源文件中，stdio.h(standard input/output)文件包含了对输入/输出函数的声明。

函数 printf 和 scanf 是使用最频繁的输入/输出函数，下面介绍它们的使用方法。

3.4.1　格式输出函数

printf 函数称为格式输出函数，函数名最末字符 f 即为“格式”(format)之意。其功能是按用户指定的格式，将指定的数据显示到屏幕上。printf 函数调用的一般形式为

```
printf("格式控制字符串",表达式 1,表达式 2,…,表达式 n);
```

功能：按格式控制字符串中的格式依次输出表达式 1，表达式 2，…，表达式 n 的值。

例如，

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

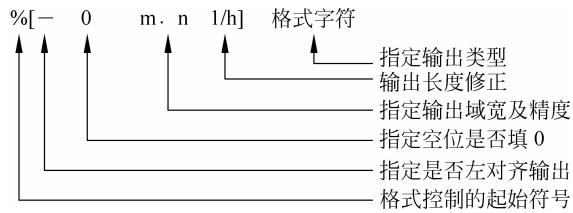
说明：格式控制字符串是用双引号括起来的字符串，用于指定输出格式。格式控制字符串一般由两种成分组成，即普通字符和格式控制字符。在输出时，普通字符(包括转义字符)按原样显示在屏幕上，起提示作用。例如，printf("%d, %c\n", a, b)中双引号内的逗号、空格和换行符等均为普通字符，将按原样显示。格式控制符，由%和格式字符等组成，如%d，%8.2f，%c 等，作用是将输出的数据按指定的格式输出，格式控制字符串中的第一个%与输出表列中的表达式 1 搭配，第二个%与表达式 2 搭配，依此类推，直到所有%与输出表达式搭配完为止。因此，必须保证格式控制字符串中以%开头的格式控制符个数与输出表列中表达式的个数及类型精确地匹配，否则将会产生不可预测的输出结果。

特别地，格式控制字符串中可以不出现%，用于直接输出字符串，例如：

```
printf("Hello!\n");
```

表示将字符串“Hello!\n”输出。

格式控制符总是由%字符开始，并以一个类型描述符结束，中间是可选的附加说明项。其完整格式为



C语言中不同类型的数据采用不同的格式控制字符。常用的格式控制字符有以下几种。

1．整型格式控制符

用于控制整型数据的输出格式,包括十进制、八进制、十六进制三种格式。

(1) 十进制格式：以十进制形式输出整型数据,其格式控制符为

%d或%md	用于基本整型
%ld或%mld	用于长整型
%u或%mu	用于无符号基本整型
%lu或%mlu	用于无符号长整型

(2) 八进制形式：以八进制形式输出整型数据,其格式控制符为

%o或%mo	用于基本整型
%lo或%mlo	用于长整型

(3) 十六制形式：以十六进制形式输出整型数据,其格式控制符为

%x或%mx	用于基本整型
%lx或%mlx	用于长整型

在以上各种整型格式控制符中,m为整数值,表示输出的整型数据所占总宽度。如果数据的位数小于m,则左端补以空格,若大于m,则按实际位数输出。如果在格式控制符中没有给出m,则输出数据的所有数位。例如,

```
printf("%3d, %4d", a, b);
```

若int a＝12,b＝12345,则输出结果为(其中□表示空格,下同)

```
□12, 12345
```

[**例3.4.1**] 不同格式的整数输出。

```
#include<stdio.h>
void main()
{
   int a = 120;
   long int b = 135790;
   printf("a=%d, b=%d\n", a, b);          //a、b按十进制数格式输出
   printf("a=%o, b=%o\n", a, b);          //a、b按八进制数格式输出
   printf("a=%x, b=%x\n", a, b);          //a、b按十六进制数格式输出
   printf("a=%u, b=%u\n", a, b);          //a、b按无符号数格式输出
}
```

运行结果：

```
a = 120, b = 135790
a = 170, b = 411156
a = 78, b = 2126e
a = 120, b = 135790
```

本例中多次输出了 a、b 的值，但由于不同格式控制串指定了不同的输出格式，输出的结果各不相同。

以八进制形式输出整数时，其前导数字 0 不输出；以十六进制形式输出整数时，其前导数字 0x 不输出。

2. 浮点型格式控制符

用于控制浮点型数据的输出格式，包括小数、指数和普通三种形式。

(1) 以小数形式输出浮点型数据，其格式控制符为

```
%f 或 %m.nf 或 %-m.nf
```

m 表示输出数据所占的总宽度，包括小数点所占的 1 列，n 表示小数部分所占的位数。如果数值的宽度小于 m，则左端补齐空格。如果数据的整数位＋n＋1 大于 m，则数据的整数部分按实际的位数输出。

%f 用于输出单精度和双精度浮点数，没有给出输出数据的位数，则按系统默认位数输出，即输出数据的全部整数和 6 位小数。

%－m.nf 与%m.nf 功能基本相同，差别是当输出数据的实际位数小于 m 时，输出的数值向左端靠，右边补足空格。

［**例 3.4.2**］　输出实数时观察其有效位数。

```
void main()
{
    float x,y;
    x = 333333.333; y = 444444.444;
    printf("%f",x + y);
}
```

运行结果：

```
777777.781250
```

显然，只有前 7 位数字是有效数字。

注意：double 型双精度数据输出使用%lf 与%f 无区别，它的有效位数一般为 16 位。

［**例 3.4.3**］　输出双精度数时观察其有效位数。

```
void main()
{
   double x,y;
   x = 3333333333333.333333333;
   y = 4444444444444.444444444;
   printf("%f",x + y);
}
```

运行结果：

```
7777777777777.777300
```

可以看到最后 3 位小数(超过 16 位)是无意义的。

(2) 以指数形式输出浮点型数据,其格式控制符为

```
%e或%m.ne
```

例如 printf("%e", 123.456789);

则输出为：1.234568e+002。

(3) 以普通形式输出浮点型数据,其格式控制符为

```
%g
```

%g 格式由系统根据数值的大小自动选用%f 或%e。在事先不能确定输出浮点数的宽度有多大的情形下,用%g 格式来输出是一种好的选择。

例如,若 float f = 123.468; 则

```
printf("%f□□%e□□%g", f, f, f);
```

输出结果如下：

```
123.468000  □□1.234680e+002□□123.468
```

3. 字符型格式控制符

用于说明字符型数据的输出格式。其格式控制符为

```
%mc或%c
```

其中,m 表示输出宽度,即在输出字符的左边将要补 m−1 个空格。例如,

```
char c='A';
printf("%5c",c);
```

则输出□□□□A,即 c 变量输出占 5 列,前 4 列补空格。

上述输出语句中的前一个 c 是格式符,后一个 c 是变量名。

值在 0～127 范围内的整数,也可以按字符形式输出,系统会将该整数作为 ASCII 码转换成相应的字符；反之,一个字符数据也可以按整数形式输出。

[**例 3.4.4**] 字符数据的输出。

```
#include<stdio.h>
void main()
{
   char c='B';
   int i=66;
   printf("%c,%d\n",c,c);
   printf("%c,%d\n",i,i);
}
```

运行结果：

```
B,66
B,66
```

4. 字符串格式控制符

用于输出一个字符串，其格式控制符为

```
%s, %ms, %-ms, %-m.ns
```

［**例 3.4.5**］ 字符串数据的输出。

```
void main()
{
   printf("%s%s","China","□Beijing");
   printf("%3s, %7.2s, %.4s, %-5.3s", "Hello", "Hello", "Hello", "Hello");
}
```

运行结果：

```
China□Beijing
Hello,□□□□□He,Hell,Hel□□
```

说明：第一个%3s，实际字符串长度超过 3，按实际长度输出；第二个%7.2s，输出字符串左端 2 个字符，补足 5 个空格，宽度为 7；第三个%.4s，输出字符串左端 4 个字符，宽度为 4；第四个%－5.3s，输出字符串左端 3 个字符，右端补足 2 个空格。

使用 printf 函数时，应注意格式控制符和输出数据之间的相互匹配。如果使用了与输出数据类型不匹配的格式控制符，C 编译系统并不显示错误信息，而按用户选择的错误格式控制符进行输出，将导致输出结果没有意义。

［**例 3.4.6**］ 格式不匹配的输出示例。

```
void main()
{
        float f = 12345;
        int i = 100;
        printf("f = %d\n",f);
        printf("i = %f\n",i);
}
```

输出结果：

```
f = 0
i = 0.000000
```

3.4.2　格式输入函数

scanf 函数称为格式输入函数，即按指定格式从键盘读取数据并赋给指定变量。其调用形式为

```
scanf("格式控制字符串",输入项地址列表)
```

说明：

(1) 格式控制字符串和 printf 函数中的格式控制字符串一样，由两类字符组成，即格式控制符和普通字符。

(2) 输入项地址列表由若干个地址组成，以逗号“,”分隔。每个地址对应输入数据所要存储的内存地址，可以是变量的地址(地址运算符“&”后跟变量名)，或字符数组名。

(3) 格式控制字符串中以%开头格式控制符的个数与输入项地址表列的项数必须相同且类型匹配，否则将产生不可预测的结果。

(4) 输入 long int 型数据必须用%ld；输入 double 型数据必须用%lf 或%le。例如，

```
double a;
scanf("%f",&a);
```

变量 a 将不能正确获得从键盘输入的数据。

(5) scanf 函数中输入项地址列表应当是变量地址，而不是变量名，即变量名前的 & 运算符不能缺省。

(6) 输入数据时，在数据之间以一个或多个空格间隔，也可以用回车键、跳格键 tab 间隔。

(7) 在格式控制字符串中应谨慎使用普通字符。如果在格式控制字符串中使用普通字符，输入数据时应按原字符输入。例如，

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

应当从键盘输入：a=3(回车)而不是 3(回车)。如果输入时遗漏了普通字符“a=”，变量 a 将得不到预想的输入。

[**例 3.4.7**] 用 scanf 函数输入数据。

```
#include<stdio.h>
void main()
{
    int a; float b; double c;
    printf("Input a, b, c:\n");
    scanf("%d%f%lf",&a,&b,&c);
    printf("a=%d,b=%f,c=%lf\n",a,b,c);
}
```

运行结果：

```
Input a, b, c:
6□3.14□8.967(回车)
a=6, b=3.140000, c=8.967000
```

说明：

(1) 由于 scanf 函数本身不能显示提示串，故通过 printf 语句在屏幕上输出提示信息“Input a，b，c:\n”。

(2) 执行 scanf 语句，则等待用户输入。用户输入 6□3.14□8.967 后按下回车键，也可

输入 6(回车)3.14(回车)8.967(回车)。

(3) 如果把示例中 scanf 语句的格式串"%d%f%lf"写成"%d,%f,%lf",则用户正确的输入应该是 6,3.14,8.967(回车),即三个数据之间必须有逗号。

3.4.3　字符输出函数

C 语言提供了专门用于字符输出的 putchar 函数。其调用形式为

```
putchar(c);
```

其中,c 可以是字符型常量、字符型变量或整型变量。

功能:在屏幕的当前光标位置处显示 c 所表示的一个字符。

[**例 3.4.8**]　用 putchar 函数输出单个字符。

```
#include<stdio.h>
void main()
{
  char a, b,c;                                  //定义字符型变量 a,b,c
  a='O'; b='K'; c='!';                          //对变量 a,b,c 进行赋值
  putchar(a); putchar(b); putchar(c);           //输出 OK!
  putchar('\n');                                //输出换行
  putchar('\x41'); putchar('\102');             //输出 AB
}
```

putchar 函数一次调用只能输出一个字符,不能写成 putchar(a,b,c);。

3.4.4　字符输入函数

字符输入函数 getchar 没有参数,其一般形式为

```
getchar()
```

此函数的功能是接收从键盘输入的一个字符,函数的返回值就是该字符。

[**例 3.4.9**]　输入单个字符。

```
#include<stdio.h>
void main()
{
    char c; int i;
    c=getchar(); printf("c=%-4c", c);
    i=getchar(); printf("i=%-3d", i);
    printf("c1=%-4c", getchar());
}
```

在运行时,如果从键盘输入 3 个字符:abc(回车)。

运行结果:

```
c=a□□□i=98□c1=c
```

说明:如果需要连续输入多个字符,输入的字符不能有定界符,各字符之间也不能有空

格。例如本示例中以下输入方式都是错误的：

```
'a''b''c'(回车)
a b c (回车)
```

3.5 顺序结构程序设计举例

下面介绍几个顺序结构程序设计的例子。

[**例 3.5.1**] 输入长方体的长、宽、高，求长方体的体积和表面积。

解 设长方体的长、宽、高分别为 x、y、z。从数学公式可知，其体积 V 和表面积 s 分别为：V＝xyz 和 s＝2xy＋2xz＋2yz。据此，先定义变量，分别存放长、宽、高、体积和表面积，并考虑其类型为单精度浮点型。然后，分别用语句实现数据输入、数据处理和数据输出，其中，数据处理是将计算体积 V 和表面积 s 的数学公式用 C 表达式给出。程序代码如下：

```
#include<stdio.h>
void main()
{
    float x,y,z,v,s;
    printf("Input x,y,z:\n");
    scanf("%f,%f,%f",&x,&y,&z);
    v=x*y*z;
    s=2*(x*y+x*z+y*z);
    printf("x=%8.3f, y=%8.3f, z=%8.3f, v=%8.3f\n",x,y,z,v);
    printf("s=%8.2f\n",s);
}
```

运行结果：

```
屏幕提示: Input x, y, z:
键盘输入: 2.2, 3.3, 4.4(回车)
屏幕显示: x=   2.200, y=   3.300, z=   4.400, v=   31.944
          s=   62.92
```

说明：

(1) 本程序由 4 个函数调用语句和 2 个赋值语句组成，按照语句的书写顺序依次执行。这种结构属于顺序结构，程序中的每一个语句都执行一次，而且只能执行一次。

(2) 整个程序由三部分组成：数据输入部分——输入长宽高 x、y、z 的值；计算处理部分——求体积 v 和表面积 s；数据输出部分——按要求输出结果 v 和 s。

[**例 3.5.2**] 编写程序，输入两个整数，输出它们的和、差、积、商及余数。

解 程序代码如下：

```
#include<stdio.h>
main()
{
    int m,n,a,b,c,d,e;
    printf("Input m,n:\n");
    scanf("%d%d",&m,&n);
```

```
    a = m + n;   b = m - n;   c = m * n;   d = m/n;   e = m % n;
    printf("m + n = %d\tm - n = %d\tm * n = %d\tm/n = %d\tm % n = %d\n", a,b,c,d,e);
}
```

运行结果：

```
屏幕提示：Input m,n:
键盘输入：5 3(回车)
屏幕显示：m + n = 8     m - n = 2      m * n = 15      m/n = 1      m % n = 2
```

说明：

(1) 程序可以将多个语句写在同一行，同一行的语句从左到右依次执行。

(2) 为了显示字符%，可在 printf 函数的格式控制字符串中使用两个连续的%。

[**例 3.5.3**] 求 $ax^2+bx+c=0$ 方程的根。系数 a、b、c 由键盘输入，假设 $b^2-4ac>0$。

解 根据一元二次方程的求解公式可以知道，当 $b^2-4ac>0$ 时，方程有两个相异的实根：

$$x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

$$x_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}$$

为计算方便，可以将上面的分式分为两项：

$$p = \frac{-b}{2a}, \quad q = \frac{\sqrt{b^2-4ac}}{2a}$$

$$x_1 = p+q, \quad x_2 = p-q$$

源程序代码如下：

```
#include <stdio.h>
#include <math.h>
void main()
{
    float a, b, c, disc, x1, x2, p, q;
    printf("Input a,b,c:\n");
    scanf("%f%f%f", &a, &b, &c);
    disc = b * b - 4 * a * c;
    p = -b/(2 * a);
    q = sqrt(disc)/(2 * a);
    x1 = p + q; x2 = p - q;
    printf("x1 = %5.2f\nx2 = %5.2f\n", x1, x2);
}
```

运行结果：

```
屏幕提示：Input a,b,c:
键盘输入：1   1   -2(回车)
屏幕显示：
x1 =  1.00
x2 = -2.00
```

说明：

(1) 程序中 sqrt()为求平方根函数。为能调用数学函数库中的函数，必须在程序的开头写#include<math.h>命令，表示包含文件 math.h。文件 math.h 包含了常用数学函数的声明，如绝对值函数 abs()、fabs()、三角函数等。

(2) 当用户输入的系数 a，b，c 使得 b^2-4ac 小于零或输入 a 为零时，该程序将产生运行错误。

本章所举例子都属顺序结构，不难发现，所有程序都可以看成是由数据输入、数据处理和数据输出 3 个部分组成，程序无非是对于特定的输入数据进行处理并输出处理结果的指令序列。因此，学习 C 语言中丰富的控制语句能进行更复杂的流程控制，是后续章节学习的重点。

习题

一、思考题

1. C 语言中的语句有哪几类？
2. 顺序结构的语句有哪些？
3. 怎样区分表达式和表达式语句？什么时候用表达式，什么时候用表达式语句？
4. C 语言的输入输出功能是表达式语句吗，为什么？
5. 编写 4 个不同的 C 语句，他们都可以把整型变量 x 的值加 1。
6. 查找下列程序段中的错误，并将其改正。

(1)
```
float x,y;
scanf("%f,%f",x,y);
```

(2)
```
double f=3.1415926;
printf("%d",f);
```

(3)
```
float x,y;
scanf("%f%f\n",x,y);
```

(4)
```
double x; long y;
scanf("%f%d",&x,&y);
```

二、选择题

1. 运行下面程序，输出结果是_____。

```
void  main()
{   int  a=5;
    printf("a=%d\n",++a+2);
}
```

A) a=6　　B) 8　　C) a=5　　D) a=8

2. putchar 函数可以向屏幕输出一个_____。

A) 整型变量值　　B) 实型变量值

C) 字符串　　D) 字符或字符变量值

3. 运行以下程序，从键盘输入 25,13，10<回车>，则输出结果是_____。

```
void main()
{
    int  a1,a2,a3;
    scanf("%d, %d, %d",&a1,&a2,&a3);
    printf("a1+a2+a3=%d\n",a1+a2+a3);
}
```

A）a1＋a2＋a3＝48　　B）a1＋a2＋a3＝25

C）a1＋a2＋a3＝10　　D）不定

4. 设有以下程序段，则输出结果是_____。

```
char  c1='b', c2='e';
printf("%d,%c\n",c2-c1,c2-'a'+'A');
```

A）2，M　　B）3,E

C）2，E　　D）输出结果不确定

5. 下面程序的执行结果是_____。

```
void main()
{  int a,b;
     a=20; b=10;
     a+=a+b;
     a-=a-b;
     printf("%d\n", a);
}
```

A）10　　B）－10　　C）30　　D）0

6. 下面的语句_____正确地描述了计算公式 $y=\frac{ax^3}{x-b}$。

A）y = ax*x*x / x-b　　B）y = ax*x*x / (x-b)

C）y = (a*x*x*x) / (x-b)　　D）y = a*x*x*x / x-b

7. 有以下程序，叙述中正确的是_____。

```
void main()
{ char a1='M',a2='m';
  printf("%c\n",(a1,a2));
}
```

A）程序输出大写字母 M　　B）程序输出小写字母 m

C）程序运行时产生出错信息　　D）格式说明符不足，编译出错

三、编程题

1. 若 a＝3,b＝4,c＝5,x＝1.2,y＝2.4,z＝－3.6,u＝51274,n＝128765,c1＝'a',c2＝'b'。想得到以下的输出结果，请写出程序(包括定义变量和输出设计)。

要求输出的结果如下：

```
a= 3  b= 4  c= 5
x=1.200000,y=2.400000,z=-3.600000
```

```
x+y= 3.60  y+z= -1.20  z+x= -2.40
u= 51274  n=  128765
c1 = 'a' or 97(ascii)
c2 = 'b' or 98(ascii)
```

2. 编写程序,输入圆的半径,计算并输出其周长和面积。常量 pi 的值取 3.14159,周长和面积取小数点后 2 位数字。

3. 编写程序,把整数华氏温度 f 转换为浮点型的摄氏温度 c。转换公式为 c=5/9(f-32),输出要有文字说明,取 2 位小数。

4. 编写程序,输入三角形的三边的边长,求三角形面积。三角形面积的计算公式为:p=(a+b+c)/2 ,S=sqrt(p(p-a)(p-b)(p-c))。

5. 编写程序,使用类似 printf("AA\aBB");语句,分别测试以下 10 个转义字符的显示效果:\a, \b, \n, \r, \t, \v, \',\" ,\\, \? 。

6. 编写程序,输入一个小写字母,输出其对应的大写字母。

7. 编写程序,从键盘输入两个字符分别存放在变量 c1 和 c2 中,要求交换 c1 和 c2 的值并输出。

8. 编写程序,设银行定期存款的年利率 rate 为 3.25%,存款期为 n 年,存款本金为 capital 元,计算并输出 n 年后的本利之和 deposit。

interest(利息) = principal(本金) * rate(年利率) * n(年)

9. 编写程序,输入销售员的销售额,计算并输出其月工资。公司规定销售人员的工作由底薪加提成构成,底薪为 1000 元,提成为当月总销售额的 9%。