第3章 按部就班——顺序结构程序设计

3.1 程序设计的基本步骤及程序执行的流程

我们已经了解了一个完整的C语言程序由哪些基本要素组成。当我们需要编写程序去解决某个问题时，通常要按如下步骤设计程序。

(1) 首先针对要求解的问题思考解决思路或建立相应的数学模型。

(2) 根据解决思路或数学模型设计出程序的算法，并将其用易于理解的方式表示出来。

(3) 用正确的语法将算法变换成对应的程序语句，从而编写出完整的可以正确执行的程序。

(4) 根据程序运行结果，验证算法的正确与否，对于不正确的算法要返回第一步进行修改或重新设计。

要想设计正确的算法，通常会用到“顺序、选择、循环”3种基本的程序控制结构，无论多复杂的程序，都可由这3种基本流程完成。

(1) 顺序结构：各操作按照从上到下的顺序按部就班顺序执行，是一种最简单的基本结构。如同现实世界中，我们解决许多问题要遵从一定的先后次序是一样的，是一种非常常见的情形。通过本章的学习，读者将掌握如何设计和编写这种简单结构的程序。

(2) 选择结构：又称分支结构，根据是否满足给定的条件从多种操作中选择一种操作。将在第5章进行详细说明。

(3) 循环结构：在一定条件下重复执行某种操作。将在第6章进行详细说明。

3.2 算法及其表示形式

计算机科学把程序设计过程分为两个基本任务：算法设计和算法实现。算法设计是指用文字或符号描述一组解决某个问题的步骤，而算法实现是指把算法设计的描述翻译成让计算机能够执行的代码。算法的表示方式不仅可以指导程序设计人员理解算法的基本实现思路，而且可以促进思考，提高程序的可读性、可靠性，改善程序执行效率，因此算法设计是计算机编程的关键。

为了便于我们对程序设计算法的理解，我们需要学习算法的表示方法，通常表示算法有以下几种方式：自然语言表示、流程图表示、伪代码表示。

3.2.1 用自然语言表示算法

自然语言就是人们日常使用的语言，可以是汉语、英语或其他语言。虽然用自然语言表

示算法通俗易懂，但自然语言表示的含义往往不太严格，文字也比较冗长，容易出现歧义。例如，某天你要出门，你妈妈对你说："能穿多少穿多少。"这句话的意思是要多穿还是少穿，要根据当时的天气情况才能判断。因此，除了一些比较简单的问题外，一般不用自然语言表示算法。

3.2.2 用流程图表示算法

流程图是用一些特定的符号来表示程序的各种操作及执行流程。用流程图表示算法直观形象，易于理解，是我们常用的算法表示方法，也是本书主要的算法表示方法，需要读者熟练掌握。

1. 传统流程图

美国国家标准学会(American National Standards Institute，ANSI)规定了一些常用的符号标准，如图 3-1 所示，且已为世界各国程序设计人员普遍采用。

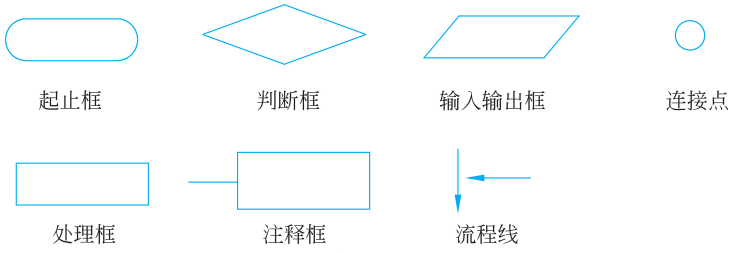


图 3-1 传统流程图常用符号

1966 年，Bohra 和 Jacopini 运用这些基本符号，针对前述 3 种基本的程序控制结构，提出了 3 种良好的算法表示单元。

(1) **顺序结构**。如图 3-2 所示，虚线框内是一个顺序结构，其中 A、B 两个操作指令按照箭头方向顺序执行。

(2) **选择结构**。如图 3-3 所示，虚线框内是一个选择结构。此结构必须包含一个判断框(菱形)，根据给定的 P 条件是否成立而选择是执行 A 指令还是执行 B 指令。

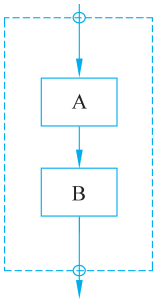


图 3-2 顺序结构

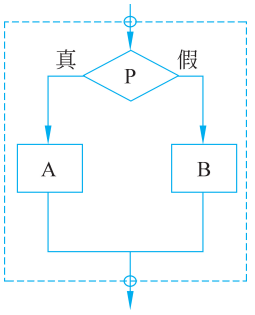


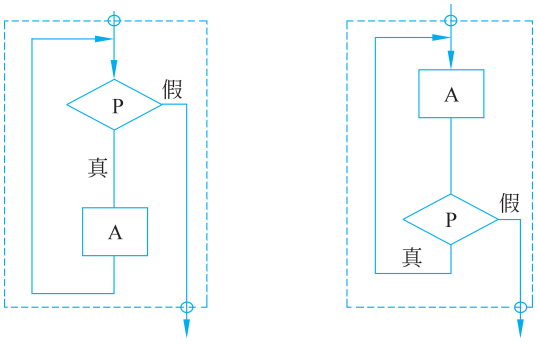
图 3-3 选择结构

(3) **循环结构**。反复执行某一部分操作，主要分为以下两类。

- while 循环结构。如图 3-4(a)所示，当给定的条件 P 成立(值为真)，执行 A 指令，执行完 A 后，返回并再次判断条件 P 是否成立，如果仍然成立，则再次执行 A 指令，如此反复判断 P 条件并执行 A 指令，直到某次 P 条件不成立(值为假)，此时不再执行

A 指令，退出循环结构，继续执行后续语句。

- do-while 循环结构。如图 3-4(b)所示，先执行 A 指令，再判断给定的条件 P 是否成立，如果条件 P 成立(值为真)，返回再次执行 A 指令，如此反复执行 A 指令并判断 P 条件，直到某次 P 条件不成立(值为假)，此时不再返回执行 A 指令，退出循环结构，继续执行后续语句。



(a) while循环　　(b) do-while循环

图 3-4　循环结构

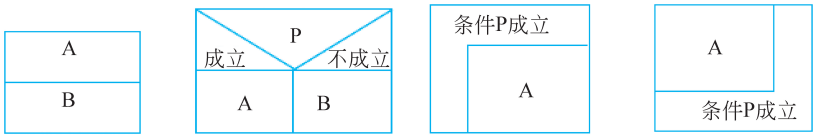
2. N-S 流程图

传统流程图由于流程线的存在，比较占空间，不太适合较为复杂的算法表示，1973 年，美国学者 I. Nassi 和 B. Shneiderman 提出了一种新的流程图形式，去掉了带箭头的流程线，大大节约了绘图空间，很好满足了复杂算法的表示需求。该流程图由提出者的名字首字母命名，称为 N-S 流程图。

(1) **顺序结构**。如图 3-5(a)所示，A、B 两个操作指令按照从上到下顺序执行。

(2) **选择结构**。如图 3-5(b)所示，按照从上到下顺序，先判断条件 P 的值，根据给定的 P 条件是否成立而选择执行 A 指令或 B 指令。

(3) **循环结构**。while 循环结构如图 3-5(c)所示，先判断条件 P 是否成立，如果成立执行循环体中的 A 指令，而后反复判断条件 P，成立执行 A，直到条件 P 不成立，退出循环。do-while 循环结构如图 3-5(d)所示，先执行循环体中的 A 指令，再判断条件 P 是否成立，如果成立返回去再次执行 A 指令，如此循环往复，直到条件 P 不成立退出循环。



(a) 顺序结构　(b) 选择结构　(c) while循环　(d) do-while循环

图 3-5　N-S 流程图

3.2.3　用伪代码表示算法

伪代码是用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。它如同一篇文章一样，自上而下地写下来。每一行(或几行)表示一个基本操作。它不用图形符号，不需遵循固定的、严格的语法规则，可用英文也可中英文混用，但一般要写成清晰易懂的格式，因此其书写方便、格式紧凑、修改方便、容易看懂，便于向符合语法规则的计算机语言算法(程序)

过渡。由于具有这些特点，其一般用来设计复杂程度不高的程序算法。

【例 3.1】 求自然数 1～10 之和，用伪代码表示其算法。

```
begin  (算法开始)
0=>sum
1=>i
当 i<=10
{
    sum+i=>sum
     i+1=>i
}
输出 sum
end  (算法结束)
```

3.3 实际问题引例

问题 1： 已知，某银行一年期的利率为 1.5%，二年期的利率为 2.1%。某人去银行存钱，一年期存款 5 000 元，二年期存款 10 000 元。到期后他共获得多少存款和利息？

解题思路：求解该问题的关键是，先求出一年期存款所获得的利息，再求出二年期存款所获得的利息，最后将这两项利息相加，再加上存款数目就得到总的存款收益。

问题 2： 某人去银行兑换外币，人民币对美元的汇率为 1 人民币兑换 0.151 2 美元，若要兑换的人民币的金额为 10 000 元，则可兑换多少美元？

解题思路：该问题的求解实际上是一个一元一次方程的求解，只需将人民币的总金额乘以 0.151 2，即可求出 10 000 人民币所对应的美元总数。

问题 3： 已知直角三角形的两条直角边分别为 3 和 4，求三角形的周长和面积。

解题思路：欲求三角形的周长，必须首先求出三角形第三条边的长度，可利用勾股定理求出，再分别计算三角形的周长和面积，这是一个按照顺序逐步求解的解题过程。

问题 4： 输入三角形的三边长，使用海伦公式计算三角形的面积（假设三边长符合组成三角形的边长条件）。

解题思路：假设在平面内，有一个三角形，知边长分别为 a、b、c，则三角形的面积 S 可由以下公式求得：

$$S=\sqrt{p(p-a)(p-b)(p-c)}$$

而公式中的 p 为半周长：

$$p=(a+b+c)/2$$

据此，我们可以用 N-S 流程图表示算法，如图 3-6 所示。

可以看到，该问题算法主要由 4 个步骤组成，按照从上至下的顺序，就可得到三角形的面积，这是一个简单的顺序结构算法。

上述几个问题都是现实生活中经常遇到的，每个问题的求解我们都将按照一定的顺序进行，先完成一个步骤，再完成下一步骤。这就是本章要介绍的顺序结构程序设计。

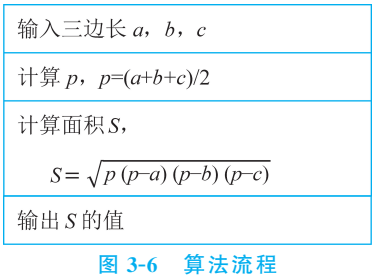


图 3-6 算法流程

编写程序：有了解题思路，我们就可以将算法用 N-S 流程图等方法表示出来，编写求解问题的程序。前面章节我们已经了解到 C 程序的基本框架为

```
#include<stdio.h>
int main()
{
    ...
    return 0;
}
```

我们要做的就是将程序中省略号的部分，用合乎 C 语言语法的语句进行替换，实现算法，要做到这件事，我们需要先掌握 C 语言的基本语法知识。

3.4 C 语句

3.4.1 分类

通过前面章节的学习，我们已经了解到 C 程序是由函数中的一条一条语句组成的，从表现形式和功能来看，通常 C 语句分为 5 类。

1. 控制语句

控制语句完成程序流程的控制功能，有以下 9 种。

(1) if()-else：条件语句。

(2) for()：循环语句。

(3) while()：循环语句。

(4) do-while()：循环语句。

(5) continue：结束本次循环语句。

(6) break：中止语句。

(7) switch：多分支。

(8) return：返回语句。

(9) goto：转向语句。

上面语句表示形式中的“()”表示里面有判别条件，“-”表示内嵌有语句，例如：

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

这里“a>b”就是判别条件，而“max=a;”和“max=b;”为内嵌语句。即比较变量 a 与 b 值的大小，将较大的值存放到变量 max 中。

2. 函数调用语句

函数调用语句由一个函数调用加一个分号构成。例如：

```
printf("Welcome to the world of C.");
```

3. 表达式语句

表达式语句由一个表达式加一个分号构成。例如：

```
a=b+5;
```

其中“a=b+5”为表达式，加一个分号为语句。

4. 空语句

空语句是只有一个分号的语句。例如：

```
;
```

空语句无实际操作，一般用来作为循环语句中的循环体(表示循环什么也不做)。

5. 复合语句

复合语句是用一对{}括起来的语句。例如：

```
{
    z=x+y;
    t=z/100;
    printf("%f",t);
}
```

复合语句常用在 if 语句或循环中，将一条或多条内嵌语句用大括号括起来组成复合语句，除了能保证程序的执行流程正确，也能使程序更清晰，可读性更强。

注意： 复合语句中最后一条语句后的分号不能忽略不写，且大括号后不能加分号。

由以上 5 类语句，我们可以看到 C 语句的标志是每条语句结尾有一个分号，没有分号就不能称其为语句。

3.4.2 赋值语句

在 C 程序中，最常使用赋值语句和输入/输出语句。我们通常会把程序中的值赋给某些变量，用于进行下一步运算或者作为结果输出。本节我们先介绍赋值语句，3.5 节再来学习程序的输入/输出。

1. 赋值运算符

一个表达式中的“=”就是赋值运算符，和数学里表示式子两边相等不同，它的作用是将“=”右边的数据赋给“=”左边的某个变量，例如：

```
a = 3+6
```

这条赋值表达式的作用是将“=”右边的计算结果 9，赋值给左边的变量 a。

注意： 赋值运算符左侧的标识符称为“左值”，出现在赋值运算符右侧的表达式称为“右值”。右值可以是一个数值，也可以是能计算出结果的表达式，而左值只能是一个变量而不能是表达式或常量。

2. 复合赋值运算符

在赋值符“=”之前加上其他双目运算符，可以构成复合运算符。+=、-+、*=、/=、%=都是较为常见的复合赋值运算符，例如下面的复合赋值表达式：

```
a + = 8;
```

等价于

```
a = a+8;
```

```
a %= b;
```

等价于

```
a = a%b;
```

```
x *= y+8;
```

等价于

```
x = x * (y+8);
```

注意：右值如果是表达式，要看作一个整体加括号，保证运算的优先级。

C 语言采用这种复合运算符，一是为了简化程序，使程序精练，二是为了提高编译效率，能产生质量较高的目标代码。

3. 赋值运算的结合性

赋值运算符与复合赋值运算都是按照“自右而左”的结合顺序，例如：

```
a=b=8;
```

等价于

```
a=(b=8);
```

即先执行“b＝8”，再执行“a＝b”的运算。

```
a+=a-=a*a;
```

等价于

```
a+=(a-=a*a);
```

即先进行“a－＝a＊a”的运算，再将运算结果作为“a＋＝”的右值，计算出结果。

4. 赋值表达式的特殊用法

赋值表达式作为表达式的一种，不仅可以出现在赋值语句中，而且可以以表达式形式出现在其他语句(如输出语句、循环语句等)中。例如：

```
printf("%d",a=b+8);
```

将“b＋8”的结果存放到变量 a 中，并输出 a 的值。

5. 赋值过程中的类型转换

如果赋值运算符两侧的类型一致，则直接进行赋值。

如果赋值运算符两侧的类型不一致，但都是数值型或字符型时，在赋值时要进行类型转换。类型转换是系统自动进行的。转换规则如下。

(1) 将浮点型数据(包括单、双精度)赋给整型变量时，先对浮点数取整，然后赋予整型变量。例如：

```
int i;
i = 7.8;
```

这时 i 的值为 7，由于 i 是整型变量，对右值做了截断处理。

(2) 将整型数据赋给单、双精度变量时，数值不变，但以浮点数形式存储到变量中。

(3) 将一个双精度型数据赋给单精度变量时，截取其前面 7 位有效数字，存放到单精度变量的存储单元(4 字节)中。但应注意数值范围不能溢出；将一个单精度型数据赋给双精度变量时，数值不变，有效位数扩展到 16 位，在内存中以 8 字节存储。

(4) 字符型数据赋给整型变量时，将字符的 ASCII 码赋给整型变量。例如：

```
int i;
i = 'a';
```

这时 i 的值为 97，即'a'字符的 ASCII 码。

(5) 将一个占多字节的整型数据赋给一个占字节少的整型变量或字符变量时，只将其

低字节原封不动地送到该变量，如图 3-7 所示。

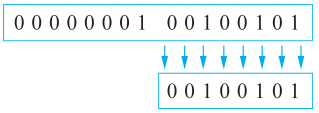


图 3-7　低位字节赋值

例如：

```
int i=293;                          //i=293
char c='A';
c=i;                                //c=37
```

注意：要避免进行这种赋值，因为赋值后数值可能失真。如果一定要进行这种赋值，应当保证赋值后数值不会发生变化。

（6）将有符号整数赋值给长度相同的无符号整型变量时，按字节原样赋值。

（7）将无符号整数赋值给长度相同的有符号整型变量时，应注意不要超出有符号整型变量的数值范围，否则会出错。

6. 赋值语句

赋值语句是由赋值表达式加上一个分号构成。赋值语句具有计算和赋值双重功能。程序中的计算功能主要是由赋值语句来完成。

当我们掌握了赋值语句的用法，我们就可以对变量进行初始化，并进行相应的运算了。现在让我们补充 3.3 节的问题 4 中缺少的语句。

```
#include<stdio.h>
#include<math.h>                            //包含数学函数库，因为要调用 sqrt 函数
int main()
{
    float a=3.4,b=4.5,c=5.6;                //a、b、c 为三边长
    float p,S;                              //p 为半周长，S 为面积
    p=1.0/2*(a+b+c);
    S=sqrt(p*(p-a)*(p-b)*(p-c));            //调用开算术平方根函数
    …
    return 0;
}
```

可以看到，算法的前三步我们已经完成，接下来读者需要学习 C 语言的输入/输出语句，从而完成最后一步，输出三角形的面积。

3.5　数据的输入/输出

3.5.1　数据输入/输出的概念

输入/输出是以计算机主机为主体而言的，所谓输出是指从计算机向外部（标准）输出设备（显示器、打印机）输出数据，反之，输入是指从（标准）输入设备（键盘、鼠标、扫描仪）向计算机输入数据。C 语言本身不提供输入/输出语句，输入/输出操作是由 C 函数库中的函数来实现的。

在使用系统库函数时，要用预编译命令 #include 将有关的“头文件”包括到用户源文件中（通常写在程序开头）。例如：

```
#include<stdio.h>
```

或

```
#include "stdio.h"
```

其中#include<stdio.h>先从系统目录开始查找要包含的头文件，若找不到，再到用户项目目录查找，适用于要包含系统提供的头文件情况，以提升查找效率。而#include"stdio.h"先从项目目录开始查找，若找不到，用户可制定路径查找，常用于包含用户自定义头文件的情况。

> **知识点小贴士：** stdio.h 头文件
>
> stdio 是 standard input & output 的意思，即标准输入/输出头文件，包含该头文件通常是为在程序中做输入/输出操作时调用相应的库函数做引用声明。

3.5.2 字符数据的输入/输出

常用的字符输入/输出函数如下。

- 字符输入函数：getchar。
- 字符输出函数：putchar。
- 格式输入函数：scanf。
- 格式输出函数：printf。
- 字符串输入函数：gets。
- 字符串输出函数：puts。

3.5.2.1 用 putchar 函数输出一个字符

格式：

```
putchar(c)
```

参数：c 为字符常量、变量或表达式。

功能：把字符 c 输出到显示器。

返值：正常，为显示的字符 ASCII 码；出错，返回－1(EOF)。

【例 3.2】 用 putchar 函数输出字符。

```
1   #include<stdio.h>
2   void main()
3   {
4       char a='C', b='A', c='R';
5       putchar(a);                                  //输出变量 a 的值
6       putchar(b);
7       putchar(c);
8       putchar('\n');
9   }
```

程序运行结果：

```
CAR
```

还可用 putchar 函数输出转义字符，例如：

```
putchar('\102');
```

程序运行结果：

```
B
```

此处的 102 为八进制数，等于十进制的 66，因而输出字符 B。

3.5.2.2 用 getchar 函数输入一个字符

格式：

```
getchar()
```

功能：从键盘读一字符。

返值：正常，返回读取的代码值；出错，返回－1(EOF)。

注意：无参数。

【例 3.3】 用 getchar 函数输入字符。

```
1   #include<stdio.h>
2   void main()
3   {
4       char c;
5       c=getchar();                          //输入的字符存入变量 c 中
6       putchar(c);
7       putchar('\n');
8   }
```

程序运行情况：

```
B↙
B
```

从键盘输入字符 B 按 Enter 键，屏幕上将显示输出的字符 B。

【例 3.4】 getchar、putchar 函数的使用。

```
1   #include<stdio.h>
2   void main()
3   {
4       char a, b, c;
5       a=getchar();                          //输入的字符存入变量 a 中
6       b=getchar();
7       c=getchar();
8       putchar(a);                           //输出变量 a 的值
9       putchar(b);
10      putchar(c);
11      putchar('\n');
12  }
```

程序运行情况：

```
CAR↙
CAR
```

可以看到连续输入 CAR 并按 Enter 键后，CAR 三个字符分别通过 getchar 函数存入了三个变量中，又通过 putchar 函数将变量的值输出。

如果不连续输入字符，而在输入一个字符后马上按 Enter 键，会得到如下运行情况：

```
C↙
A↙
C
A
```

请读者自行思考是什么原因？（提示：输入的回车符为有效字符）。

3.5.2.3 用 printf 函数输出数据

printf 函数(格式输出函数)的作用是向终端(或系统隐含指定的输出设备，通常是显示

器)输出若干个任意类型的数据,并可灵活指定数据输出格式。

> **知识点小贴士:** printf 函数
>
> printf 函数又称为标准输出函数,它在 stdio.h 库函数中定义,只能在控制台程序中使用。

1. printf 函数的一般格式

printf 函数的一般格式为

```
printf(格式控制,输出表列)
```

例如:

```
printf("%d, %c, %f\n", i, c, f);
```

printf 函数的参数包括两部分:

(1)"格式控制"是用双引号括起来的字符串,也称转换控制字符串,包括以下两种信息。

- 格式说明。格式说明由"%"和格式字符组成,如%d、%f 等。它的作用是将输出的数据转换为指定的格式输出。格式说明总是由"%"字符开始的。
- 普通字符。普通字符即需要原样输出的字符。如上面 printf 函数中双引号内的空格、逗号和换行符。

(2)"输出表列"是需要输出的一些数据,可以是常量、变量或表达式,通常与格式控制中的控制符按顺序一一对应。

【例 3.5】 printf 函数的格式说明与输出表列。

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

格式说明 输出表列

程序运行结果:

```
a=3  b=4
```

注意: 此处的"a="、空格、"b="就是原样输出的普通字符。而前后两个"%d"为格式说明符,控制输出表列中整型变量 a 和 b 的数据输出格式。

2. 基本的格式说明符

基本的格式说明符有以下几种。

(1) %d 或 %i 格式符。按十进制整型数据的实际长度输出。

(2) %c 格式符。用来输出一个字符。

【例 3.6】 字符数据的输出。

```
1  #include<stdio.h>
2  void main()
3  {
4      char c='a';
5      int i=97;
6      printf("%c,%d\n",c,c);
7      printf("%c,%d\n",i,i);
8  }
```

运行结果：

```
a, 97
a, 97
```

无论整型变量还是字符型变量，都可以用%d和%c格式符得到相应的运行结果，但要注意，超出ASCII码范围，整型变量就无法输出对应的正确字符了。

（3）%s格式符。用来输出一个字符串。

例如：

```
printf("%s", "CHINA");
```

输出字符串"CHINA"（不包括双引号）。

（4）%f格式符。用来输出实数，以小数形式输出，如不指定整个字段的长度，由系统自动指定。一般的处理方法：整数部分全部输出，并输出6位小数。

【例3.7】 输出实数时的有效位数。

```
1   #include<stdio.h>
2   void main()
3   {
4       float x,y;
5       x=111111.111;y=222222.222;
6       printf("%f\n",x+y);
7   }
```

程序运行结果：

```
333333.328125
```

结果中只有前7位是有效数字。由于x和y是单精度变量，所以"x+y"也只能保证7位的精度，后面几位没有意义。

（5）%e格式符。用格式说明%e指定以指数形式输出实数。

例如：

```
printf("%e",123.456);
```

输出如下：

```
1.234560 e+002
```

C编译系统自动指定给出数字部分的小数位数为6位，指数部分为5位。

printf函数支持的基本格式控制符如表3-1所示。

表3-1 printf函数支持的基本格式控制符

格式字符	说　明	举　例	结　果
d,i	十进制整数	int a=567; printf("%d",a);	567
u	无符号的十进制整数	int a=567; printf("%u",a);	567
o	八进制无符号整数	int a=65; printf("%o",a);	101
x,X	十六进制无符号整数(大小写作用相同)	int a=255; printf("%x",a);	ff
c	单个字符	char a=65; printf("%c",a);	A
s	字符串	printf("%s","ABC");	ABC
f	小数形式输出浮点数	float a=567.89; printf("%f",a);	567.890000

续表

格式字符	说　明	举　例	结　果
e,E	指数形式输出浮点数	float a=567.89; printf("%e",a);	5.678900e+02
g,G	以 e 和 f 中较短的一种进行输出,且不输出无意义的零	float a=567.89; printf("%g",a);	567.89
%%	百分号本身	printf("%%");	%

说明:

(1) 除了 x、e、g 格式字符可以大写外,其余格式控制符要用小写。

(2) 格式字符与输出项个数应相同,按先后顺序一一对应。

(3) 可以在 printf 函数中的"格式控制"字符串中包含转义字符。

(4) 一个格式说明必须以"%"开头,以 9 个格式字符之一为结束。

(5) 格式字符与输出项类型不一致,自动按指定格式输出。

另外,在"%"与格式控制符之间可以插入附加格式字符,即表示为

```
% 附加格式字符　格式字符
```

以实现对输出格式更为灵活的控制。

printf 函数支持的附加格式控制符如表 3-2 所示。

表 3-2　printf 函数支持的附加格式控制符

修　饰　符	功　能
m	输出数据域宽,数据长度<m,左补空格,否则按实际输出
.n	对实数,指定小数点后位数(四舍五入)
	对字符串,指定实际输出位数
—	输出数据在域内左对齐(默认右对齐)
+	在有符号数的正数前显示正号(+)
0	输出数值时指定左面不使用的空位自动用 0 填充
#	在八进制和十六进制数前显示 0,0x 前导符
l	在 d、o、x、u 前,指定输出精度为 long 型
	在 e、f、g 前,指定输出精度为 double 型

【例 3.8】 附加格式控制符的使用。

```
int a=1234;
float f=123.456;
printf("%08d\n",a);                    //输出 00001234
printf("%010.2f\n",f);                 //输出 0000123.46
printf("%0+8d\n",a);                   //输出 000+1234
printf("%0+10.2f\n",f);                //输出 000+123.46
```

3.5.2.4　用 scanf 函数输出数据

scanf 函数(格式输入函数)的作用是按照变量在内存的地址将变量值存进去。

知识点小贴士： scanf 函数

scanf 函数又称为标准输入函数，其只能在控制台程序中使用，在 stdio.h 库函数中定义。

scanf 函数一般格式为

```
scanf(格式控制,地址表列)
```

例如：

```
scanf("%d%c", &i,&c);
```

其中“%d%c”为格式控制部分，支持的格式控制符与 printf 函数一样。输入数据时，在两个数据之间以一个或多个空格间隔，也可以 Enter 键、Tab 键作为间隔。

地址表列是由若干地址组成的表列，可以是变量的地址，或字符串的首地址，如语句中的“&i,&c”。“&”是地址运算符，用于自动获取变量在内存中的地址。

scanf 函数支持的基本格式控制符如表 3-3 所示。

表 3-3　scanf 函数支持的格式控制符

格式字符	说　明
d,i	用来输入有符号的十进制整数
u	用来输入无符号的十进制整数
o	用来输入无符号的八进制整数
x,X	用来输入无符号的十六进制整数(大小写作用相同)
c	用来输入单个字符
s	用来输入字符串,将字符串送到一个字符数组中,在输入时以非空白字符开始,以第一个空白字符结束。字符串以串结束标志'\0'作为其最后一个字符
f	用来输入实数,可以用小数形式或指数形式输入
e,E,g,G	与 f 作用相同,e 与 f、g 可以互相替换(大小写作用相同)

scanf 函数支持的附加格式控制符如表 3-4 所示。

表 3-4　scanf 函数支持的附加格式控制符

修饰符	功　能
l	用于输入长整型数据(可用%ld、%lo、%lx、%lu)以及 double 型数据(用%lf 或%le)
h	用于输入短整型数据(可用%hd、%ho、%hx)
域宽	指定输入数据所占宽度(列数),域宽应为正整数
*	表示本输入项在读入后不赋给相应的变量

【例 3.9】 scanf 函数附加格式控制符的使用。

```
scanf("%4d%2d%2d",&yy,&mm,&dd);
```

输出如下：

```
输入　19991015↙
```

由于指定了域宽，则连续输入的数据会自动根据域宽分割，1999 存入变量 yy 中，10 存

入变量 mm 中,15 存入变量 dd 中。

前面已经介绍,输入多个数据时,默认用空格、Enter 键、Tab 键做输入数据的分隔符,除此之外,还可自定义分隔符。

【例 3.10】 scanf 函数指定输入分隔符。

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

输出如下:

```
输入  3,4↙
```

此处输入语句用了逗号作为指定分隔符,输入数据时必须用逗号做分隔符,才能将 3 存入变量 a,4 存入变量 b。如果再用空格等默认分隔符输入数据,将会得不到正确的数据。以此类推,读者可以自行指定其他的数据输入格式。

注意:

(1) 用“%c”格式符时,空格和转义字符会作为有效字符输入。例如:

```
scanf("%c%c%c",&c1,&c2,&c3);
```

输出如下:

```
输入  a  b  c↙
```

则字符'a'存入变量 c1,两个空格字符存入变量 c2,字符'b'存入变量 c3。

(2) 输入数据时,遇到以下情况认为该数据结束。

- 遇到空格、Tab 键或 Enter 键。
- 遇到宽度结束。
- 遇到非法输入。

例如:

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

输出如下:

```
输入  1234a123o.26↙
```

则 1234 存入变量 a,字符'a'存入变量 b,123 存入变量 c。

至此,读者已学习了基本的输入/输出方法,可以将 3.3 节的问题 4 补充完整:

```
#include<stdio.h>
#include<math.h>                          //包含数学函数库,因为要调用 sqrt 函数
int main()
{
    float a=3.4,b=4.5,c=5.6;              //a、b、c 为三边长
    float p,S;                            //p 为半周长,S 为面积
    p=1.0/2*(a+b+c);
    s=sqrt(p*(p-a)*(p-b)*(p-c));          //调用开算术平方根函数
    printf("a=%7.2f,\nb=%7.2f,\nc=%7.2f\n",a,b,c);
    printf("s=%7.2f\n",s);
    return 0;
}
```

输出如下:

```
输入 3.4,4.5,5.6↙
a=   3.40,
```

```
b=   4.50,
c=   5.60
s=   7.65
```

3.6 顺序结构程序设计举例

1. 求存款利息

已知，某银行一年期的利率为1.5%，二年期的利率为2.1%。某人去银行存钱，一年期存款5000元，二年期存款10 000元。到期后他共获得多少存款利息？（3.3节的问题1）

【问题分析】

欲求该人能够获得的存款利息，需要先求出一年期存款所获得的利息，再求二年期存款所获得的利息，最后将一年期所获利息与二年期所获利息相加即可得到总的存款利息。

【问题求解】

综合案例1：求存款利息

```
#include<stdio.h>
int main()
{
    float a, b, sum;
    a = 5000 * 0.015;
    b = 10000 * 0.021;
    sum = a + b;
    printf("%f", sum);
    return 0;
}
```

程序运行结果：

```
285.000000
```

【应用扩展】

由此，我们可以扩展计算企业利润、个人所得税、股票收益率等。

2. 兑换外币

某人去银行兑换外币，人民币对美元的汇率为1人民币兑换0.151 2美元。若要兑换的人民币金额为10 000元，则可兑换多少美元？（3.3节的问题2）

【问题分析】

求解该问题只需将已有人民币的总金额乘以0.151 2即可。

【问题求解】

综合案例2：外币兑换

```
#include<stdio.h>
int main()
{
    float d;
    d = 10000 * 0.1512;
    printf("%f", D);
    return 0;
}
```

程序运行结果：

```
1512.000000
```

【应用扩展】

本实例中，如果汇率不变，已知有 5 000 美元，请问可以兑换多少人民币呢？

3. 出国货币兑换

随着我国经济的快速增长，人们的生活水平日益提高，越来越多的人选择出国旅游以增长阅历，但要在异国消费，首先便是兑换目的旅游国的货币。假如，你要携带父母到美国旅游，已知今日的人民币兑换美元的汇率为 rate，请编程实现给定任意的人民币 x，计算可以兑换多少美元 y。

【问题分析】

本问题的求解关键是要找到人民币与美元之间的换算公式。由于货币汇率是指两种不同货币之间的兑换价格，若已知人民币兑换美元的汇率为 rate，则人民币 x 与美元 y 的兑换公式为 $y=x\times$rate。

【问题求解】

根据问题分析所得人民币与美元的兑换公式，可以在给定汇率下，对任意的人民币完成与美元之间的兑换，其算法流程如图 3-8 所示。

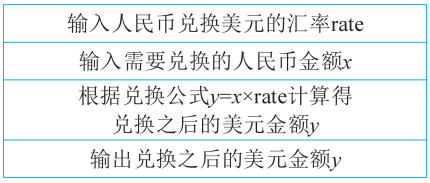


图 3-8　外部兑换算法流程图

根据流程图 3-8，可以编写相应的 C 语言程序代码如下：

综合案例 3：货币兑换

```
#include<stdio.h>
int main()
{
    double rate, x, y;
    printf("请输入当前的人民币兑换美元的汇率(大于 0 且小于 1 的小数):");
    scanf("%lf", &rate);
    printf("请输入所要兑换的人民币金额(大于 0 的数):");
    scanf("%lf", &x);
    //根据汇率将人民币兑换成美元
    y = x * rate;
    printf("%.2f 人民币在汇率为 %.4f 时可以兑换 %.2f 美元\n", x, rate, y);
    return 0;
}
```

程序运行结果：

```
请输入当前的人民币兑换美元的汇率(大于 0 且小于 1 的小数):0.1439↙
请输入所要兑换的人民币金额(大于 0 的数):10000↙
10000.00 人民币在汇率为 0.1439 时可以兑换 1439.00 美元
```

【应用扩展】

本问题的求解思路可以运用于很多不同单位的数据之间的换算，如重量换算、温度换算、体积换算等。

4. 求三角形的周长和面积

已知直角三角形的两条直角边分别为 3 和 4，求三角形的周长和面积。(3.3 节的问题 3)

【问题分析】

本题已知直角三角形的两条直角边，欲求直角三角形的周长，必须首先求出三角形的第

三条边的长度，可利用勾股定理求出。再分别计算三角形的周长和面积。

【问题求解】

综合案例 4：求直角三角形的周长和面积

```
#include<stdio.h>
#include<math.h>
int main()
{
    int a, b;
    float c, L, S;                          //变量 c 为直角三角形第三条边，L 为周长，S 为面积
    scanf("%d,%d", &a, &b);
    c = sqrt(a * a + b * b);
    L = a + b + c;
    S = 1.0 / 2 * a * b;
    printf("%f, %f", L, S);
    return 0;
}
```

程序运行结果：

```
3,4↙
12.000000, 6.000000
```

【应用扩展】

由此，我们也可以已知正方形的边长，计算正方形的周长和面积；已知长方形的长和宽，求长方形的周长和面积；已知圆的半径，求圆的周长和面积等。

5. 输出简单的字符图形

请用输出函数输出如图 3-9 所示的字符图案。

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

(a) 菱形

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

(b) 平行四边形

图 3-9　菱形和平行四边形图案

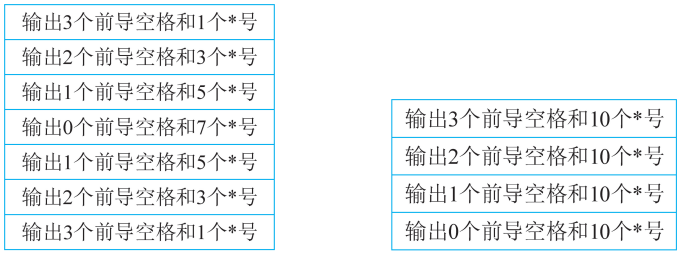
【问题分析】

通过观察可知，图 3-9(a)中的菱形总共有 7 行，每行星号(*)的个数依次为 1、3、5、7、5、3、1。从形态上看，每行星号都是居中显示的。图 3-9(b)中的平行四边形总共有 4 行，各行星号的个数均为 10。从形态上看，上一行的第一个星号与下一行的第一个星号相比在前面多了一个空格。为了能够使输出的星号呈现出题目要求的形态，可以在正式输出每行星号之前，先输出相应个数的空格符来实现。

【问题求解】

根据以上分析，借助流程图绘制软件可以作出相应的算法流程图，如图 3-10 所示。

根据算法流程图，可以写出相应的 C 语言实现代码。显示图 3-9(a)中的菱形图案的 C 语言代码如下：



(a) 菱形的算法流程图　　(b) 平行四边形的算法流程图

图 3-10　算法流程图

综合案例 5(a)：显示图 3-9(a)中的菱形

```
#include<stdio.h>
int main()
{
    printf("    * \n");                    //输出第 1 行的星号,前面有 3 个空格
    printf("   ***\n");                    //输出第 2 行的星号,前面有 2 个空格
    printf(" *****\n");                    //输出第 3 行的星号,前面有 1 个空格
    printf("*******\n");                   //输出第 4 行的星号,前面有 0 个空格
    printf(" *****\n");                    //输出第 5 行的星号,前面有 1 个空格
    printf("   ***\n");                    //输出第 6 行的星号,前面有 2 个空格
    printf("    * \n");                    //输出第 7 行的星号,前面有 3 个空格
    return 0;
}
```

显示图 3-9(b)中的平行四边形图案的 C 语言代码如下：

综合案例 5(b)：显示图 3-9(b)中的平行四边形

```
#include<stdio.h>
int main()
{
    printf("   **********\n");             //输出第 1 行的星号,前面有 3 个空格
    printf("   **********\n");             //输出第 2 行的星号,前面有 2 个空格
    printf(" **********\n");               //输出第 3 行的星号,前面有 1 个空格
    printf("**********\n");                //输出第 4 行的星号,前面有 0 个空格
    return 0;
}
```

【应用扩展】

本问题的算法核心是充分运用 printf 函数输出数据时的格式控制能力，结合输出前导空格和回车换行，可以打印各式各样的图案。但现在的这种实现方式在打印很大的图案时会显得很烦琐，在学习完循环结构程序设计之后，我们可以用更有效的方法来实现。

6. 简易计算器菜单

请用输出函数实现在命令行提示符窗口中输出简易的计算机菜单，如图 3-11 所示。

```
==========简易计算器==========
1. 加法运算
2. 减法运算
3. 乘法运算
4. 除法运算
您要执行的运算是（请输入上述菜单项前的编号）：
```

图 3-11　简易的计算机菜单

【问题分析】

题目中要求输出的简易计算机菜单信息都是由普通字符串构成，使用 printf 函数来输出即可。每个菜单项和提示信息各自占据一行，则可以通过回车换行来实现。

【问题求解】

根据问题分析可以写出相应的 C 语言代码如下：

综合案例 6：输出菜单

```
#include<stdio.h>
int main()
{
    printf("==========简易计算器==========\n");
    printf("1. 加法运算\n");
    printf("2. 减法运算\n");
    printf("3. 乘法运算\n");
    printf("4. 除法运算\n");
    printf("您要执行的运算是(请输入上述菜单项前的编号):");
    return 0;
}
```

【应用扩展】

在一个完整的 C 语言应用程序中，通常包含有多个功能模块，为了提供更加人性化的功能，一般都会给出一个功能菜单供用户选择所要执行的操作。

7. 交换两个瓶中的液体

有两个瓶子 A 和 B，分别盛放醋和酱油，要求将它们互换（A 瓶原来盛放醋，现改盛酱油，B 瓶则相反）。

【问题分析】

要使 A、B 容器所盛的液体互换，关键是需要借助一个空的容器 C（假定 3 个容器的大小相同，避免液体溢出）来暂存交换中的液体。具体交换步骤：先将 A 中的液体倒入 C 中，再将 B 中液体倒入 A 中，最后将 C 中的液体倒入 B 中。

【问题求解】

根据以上分析，借助流程图绘制软件可以画出相应的算法流程图，如图 3-12 所示。

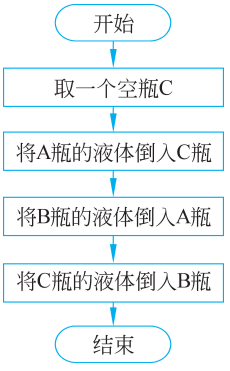


图 3-12　算法流程图

根据问题分析可以写出相应的 C 语言代码如下：

综合案例 7：互换两个瓶子 A 和 B 中的液体

```
#include<stdio.h>
int main()
{
    int a, b, c;                                    //代表三个同样的瓶子
    printf("请输入两个整数:");
    scanf("%d%d",&a,&b);
    c = a;
    a = b;
    b = c;
    printf("交换后的结果是:%d %d\n",a,b);
    return 0;
}
```