第 3 章　顺序结构程序设计

【教学目的】

1. 了解 C 程序的三种基本结构。
2. 掌握顺序结构程序的设计思想和方法。
3. 掌握常用的几种顺序结构语句。
4. 掌握根据题意编写出简单的顺序结构程序。

【教学重点】

C 语言实现顺序结构的语句及使用方法。

【教学难点】

C 语言对标准输入函数（scanf）、标准输出函数（printf）的格式符规定。

从程序流程的角度看，程序可以分为三种基本结构，即顺序结构、选择结构、循环结构，这三种基本结构可以组成所有的复杂程序。C 语言提供了多种语句来实现这些程序结构。本章介绍 C 语言程序设计的基本方法和基本的程序语句，使读者对 C 程序有一个初步的认识，为后面各章的学习奠定基础。

3.1　C 语句概述

C 语言的语句是用来向计算机系统发出操作指令进而完成一定任务的。从第 1 章已知，一个 C 程序的函数包含声明部分和执行部分，执行部分由语句组成，由执行语句实现程序的功能。C 语句可分为 6 类，即表达式语句、赋值语句、函数调用语句、控制语句、复合语句和空语句。

1. 表达式语句

表达式语句是最简单的可执行语句，由表达式加上分号“;”组成。其一般形式为

```
表达式;
```

例如，

```
i++;
z=x+y;
```

执行表达式语句就是计算表达式的值。有效的表达式语句一般都要有赋值运算，否则，不做任何赋值运算的表达式是无意义的。例如，x-y;是减法运算语句，计算结果并不能保留，因此无实际意义。

2. 赋值语句

赋值语句是由赋值表达式再加上分号构成的表达式语句。其一般形式为

```
变量=表达式;
```

赋值语句的功能和特点都与赋值表达式相同，它是程序中使用得最多的语句之一。使用赋值语句时需要注意以下几点：

（1）由于赋值符“=”右边的表达式也可以是一个赋值表达式，因此形式

```
变量=(变量=表达式);
```

是成立的，它形成了嵌套的情形。展开之后的一般形式为

```
变量=变量=…=表达式;
```

例如，

```
x=y=z=i=j=6;
```

按照赋值运算符的右结合性，因此它实际上等效于

```
j=6;
i=j;
z=i;
y=z;
x=y;
```

（2）注意在变量说明中给变量赋初值和赋值语句的区别。给变量赋初值是变量说明的一部分，赋初值后的变量与其后的其他同类变量之间仍必须用逗号分隔，而赋值语句则必须用分号结尾。例如，

```
int x=10,y,z;
```

（3）在变量说明中，不允许连续给多个变量赋初值。例如，如下变量说明是错误的：

```
int x=y=z=6;
```

必须写为

```
int x=6,y=6,z=6;
```

而赋值语句允许连续赋值。

（4）注意赋值表达式和赋值语句的区别。赋值表达式是一种表达式，它可以出现在任何允许表达式出现的地方，而赋值语句则不能。

例如，下述语句是合法的：

```
if((y=x+1)<0) y++;
```

语句的功能是，若表达式 y=x+1 小于 0，则 y++。

又如，下述语句是非法的：

```
if((y=x+1;)<0) y++;
```

因为 y=x+1;是语句，不能出现在表达式中。

3. 函数调用语句

函数调用语句由函数名、实际参数加上分号“;”组成。其一般形式为

```
函数名(实际参数表);
```

执行函数调用语句就是调用函数体并把实际参数赋予函数定义中的形式参数，然后执行被调函数体中的语句，求取函数值。例如，

```
printf("This is C Program.");
```

调用库函数，输出字符串。printf 是函数名，"This is C Program."是实际参数。函数名既可以是 C 语言提供的库函数名，又可以是我们自己定义的函数名。自定义函数将在第 7 章中详细介绍。

4. 控制语句

控制语句用于控制程序的流程，一般指能改变顺序结构的语句，以实现程序的各种结构。

控制语句由特定的语句定义符组成。C 语言有 9 种控制语句，可分成如下 3 类。

（1）条件判断语句：if、switch 语句。

（2）循环执行语句：while、do while、for 语句。

（3）转向语句：break、continue、goto、return 语句。

熟练掌握控制语句是学会程序设计的基础。

5. 复合语句

复合语句是用一对花括号{}将数据说明语句和若干条有序的执行语句（或仅有一条或若干条执行语句）组合在一起构成的。其一般形式为

```
{ 数据说明语句;
  执行语句;
}
```

在程序中应把复合语句视为单条语句，而不能视为多条语句。例如，

```
{
  int a,b,c,x,y,z;
  c=a+b;
  z=x+y;
  printf("%d %d",c,z);
}
```

它是由多条语句用{}括起来组成的一条复合语句。程序在执行复合语句时，按在其中的语句顺序执行，每条执行语句又可以是控制语句或可以用复合语句来代替，这样，C 语言的语句就形成了一种层次结构。复合语句内的各条语句都必须以分号“;”结尾，且在右花括号“}”外不能加分号。

6. 空语句

C 语言中有一种很特殊的语句，即空语句。其一般形式为

```
;
```

只有分号“;”组成的语句称为*空语句*。空语句是什么也不执行的语句。尽管空语句不会有任何命令执行，但仍然是一条有用的语句，在程序中空语句可用做空循环体。

例如，

```
for(s=0,i=0;i<100;i++,s=s+i);
```

该语句的功能是，计算 1 到 100 的累加和。这里的循环体为空语句，没有空语句会出现语法错误。

3.2　程序的三种基本结构

结构化程序设计方法是人们总结出的一套良好的程序设计原则和方法，按照这种方法设计的程序结构清晰、层次分明，且易于阅读和维护，能提高程序设计的质量和效率。

结构化程序设计的基本思想是：任何程序都可以用三种基本结构的组合来实现，且每个基本结构都可以包含一条或若干条语句。这三种基本结构是顺序结构、选择结构和循环结构，它们都具有一个入口和一个出口。

1．顺序结构

顺序结构见图 3.1。顺序结构的程序流程是按照书写顺序依次执行的。

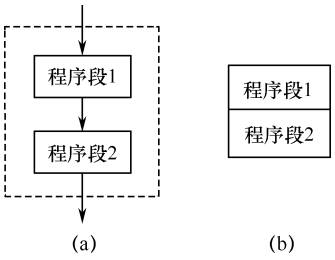

图 3.1 顺序结构

先执行程序段 1 操作，再执行程序段 2 操作，两者是顺序执行的关系。图中（b）是 N-S 结构化流程图（下同）。

2．选择结构

选择结构的程序流程先对给定的条件进行判断，再根据判断的结果决定执行哪个分支，如图 3.2 所示。

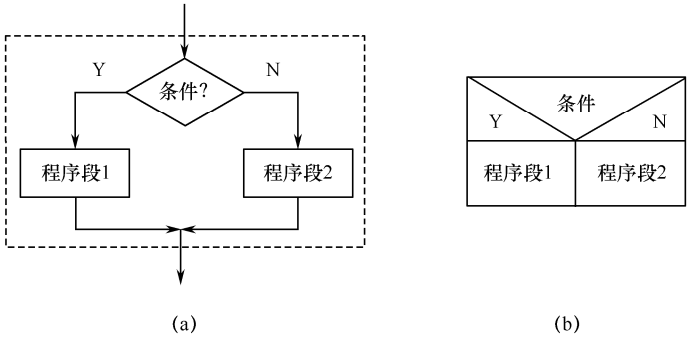

图 3.2 选择结构

当条件成立（为“真”）时，执行程序段 1，否则执行程序段 2。注意：根据条件判断的结果只能执行程序段 1 或程序段 2 之一。

3．循环结构

循环结构在给定条件成立时反复执行某段程序。循环结构有如下两种。

（1）当型循环结构，如图 3.3 所示。

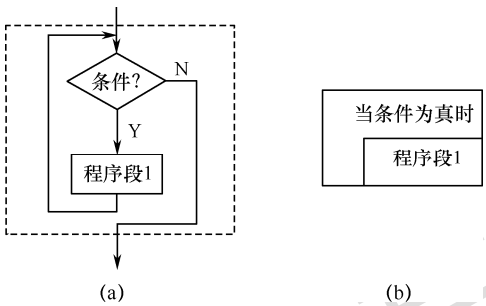

图 3.3 当型循环结构

当条件成立（为“真”）时，反复执行程序段 1 的操作，直到条件为“假”时才停止循环。

（2）直到型循环结构，如图 3.4 所示。

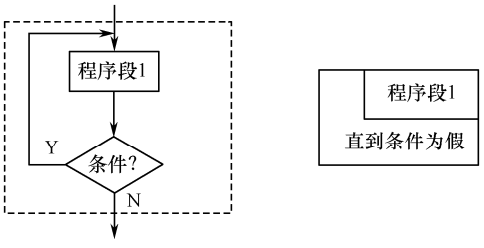

图 3.4　直到型循环结构

先执行程序段 1 的操作，再判断条件是否为“真”，若条件为“真”，则继续执行程序段 1，如此反复，直到条件为“假”时停止循环。

关于三种基本结构的特征及结构化程序设计方法，本章只简单介绍到这里。本章中重点介绍几种最基本的语句及由它们构成的顺序结构程序。第 4 章、第 5 章将分别介绍选择结构和循环结构的程序设计。

3.3　数据的输入/输出

数据输出是指从计算机向外部输出设备（如显示器）输出数据，简称“输出”；数据输入是指从外部输入设备（如键盘）向计算机输入数据，简称“输入”。在 C 语言中，所有数据的输入和输出都是由库函数完成的，因此都是函数语句。C 语言函数库中有一批“标准输入/输出函数”，它们是以标准的输入/输出设备（一般为终端设备）为输入/输出对象的，包括 printf（格式输出）、scanf（格式输入）、putchar（输出字符）、getchar（输入字符）、puts（输出字符串）、gets（输入字符串）等函数。在本章中介绍前面 4 个最基本的输入/输出函数。

使用 C 语言库函数时，要用预编译命令“#include”将有关的“头文件”包含到用户源文件中，因为头文件中包含了与用到的函数有关的信息。例如，在使用标准输入/输出库函数时，要用到“stdio.h”文件，因此要用“#include”命令将“stdio.h”文件包含到源文件中。“#include”命令一般放在程序的开头，因此这类以“.h”为后缀的文件称为头文件。在调用标准输入/输出库函数时，文件开头应包含预编译命令“#include <stdio.h>”。有关预编译命令的介绍，详见第 8 章。

3.3.1　格式输出函数 printf

printf 函数称为格式输出函数。前面的例题中已用过 printf 函数，其作用是向终端输出设备（如显示器）输出若干任意类型的数据。

1. printf 函数调用的一般形式

printf 函数是一个标准库函数，其函数原型在头文件“stdio.h”中，调用的一般形式为

```
printf("格式控制字符串",输出表列)
```

例如，

```
printf("%d,%f\n",x,y)
```

其中“格式控制字符串”用于指定输出格式。格式控制字符串可由格式字符和非格式字符串组成。

① 格式字符串：是以“%”开头的字符串，由“%”和格式字符组成，以说明输出数据的类

型、形式、长度、小数位数等。例如，“%d”表示按带符号十进制整型数输出，“%f”表示以小数形式输出，“%c”表示按字符输出等。

② 非格式字符串：包括普通字符和转义字符。普通字符在输出时按原样输出，在显示中主要起提示作用。例如，前面的 printf 函数中双引号内的逗号。转义字符是以“\”开头的特殊字符，通常起控制操作的作用，例如“\n”就是回车换行操作。关于转义字符，前面已详细介绍过，这里不再赘述。

“输出表列”就是各个输出项，即需要输出的一些数据，也可以是表达式，要求格式控制字符串和各个输出项在数量和类型上一一对应。例如，

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

双引号中的字符除格式说明的“%d”外，还有非格式说明的普通字符和转义字符，普通字符（空格、逗号、a、=、b）在输出时按原样输出，转义字符“\n”在输出时产生回车换行操作。如果 a 和 b 的值分别为 5、6，那么输出为

```
5 6
a=5,b=6
```

2. 格式控制字符串

格式控制字符串的一般形式为

```
[标志][输出最小宽度][.精度][长度]类型
```

其中方括号[]内的项为可选项。各项的含义介绍如下。

（1）格式字符

类型字符用以表示输出数据的类型，对不同类型的数据用不同的格式字符表示，常用的有以下几种格式字符。

① d 格式符。以带符号的十进制数形式输出整数。例如，

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

若 a=-1，b=12，则输出结果为

```
-1,12
```

② o 格式符。以八进制数无符号形式输出整数。例如，

```
printf("%o",a);
```

若 a=-1，则输出结果为

```
177777
```

这是由于-1 在内存单元中是以补码形式存放的，其存放形式如下所示：

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

输出的数值不带符号，最高位作为八进制数的一部分输出。

③ x，X 格式符。以十六进制数无符号形式输出整数。例如，

```
printf("%x",a);
```

若 a=-1，则输出结果为

```
ffff
```

同样，输出的数值不带符号。

④ u 格式符。以无符号十进制数形式输出整数。例如，

```
printf("%u",a);
```

若 a=-1，则输出结果为

```
65535
```

⑤ c 格式符。用来输出一个字符。例如，

```
printf("%c",c);
```

若 c='a'，则输出结果为

```
a
```

注意：“%c”中的 c 是格式符，逗号右边的 c 是变量名，不要搞混。

在 C 语言中，一个整数可以用字符形式输出，在输出前，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用整数形式输出。例如，

```
printf("%c,%d",i,i);
```

若 i=97，则输出结果为

```
a,97
```

又如，

```
printf("%c,%d",c,c);
```

若 c='a'，则输出结果为

```
a,97
```

⑥ s 格式符。用来输出一个字符串。例如，

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

输出结果为

```
program
```

注意：输出"program"字符串时，双引号不输出。

⑦ f 格式符。以小数形式输出单、双精度实数。例如，

```
printf("%f",x);
```

若 x=123.123123，则输出结果为

```
123.123112
```

注意：输出单精度实数时，有效位数一般为 7 位，小数位数为 6 位（由系统自动指定）。显然，以上输出结果中前 7 位数字 1231231 是有效数字，而后两位 12 是无效数字，所以输出的数据并非都是准确的。

因此，输出双精度实数时，有效位数一般为 16 位，小数位数默认也是 6 位。例如，

```
printf("%f",x);
```

若 x=123123123123.123123，则输出结果为

```
123123123123.123109
```

可以看出，后两位数字 09 是无意义的，因为已超过有效位数（16）。

⑧ e 格式符。以指数形式输出实数。例如，

```
printf("%e",65.4321);
```

则输出结果为

```
6.543210e+001
```

输出的实数按规范化的指数形式输出，即小数点前必须有且只能有 1 位非零数字。C 编译系统自动指定 6 位小数。指数部分占 5 位（如 e+001），其中“e”占 1 位，指数符号“+”占 1 位，指数“001”占 3 位。因此，输出的数据共占 13 列宽度（注意，不同系统的规定略有不同）。

⑨ g 格式符。根据数值的大小自动选择 f 格式或 e 格式（选择输出时占宽度较小的一种格式）输出实数，且不输出无意义的零。例如，

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

若 y=543.321，则输出结果为

```
543.321000,5.433210e+002,543.321
```

可以看出，用%f 格式输出的实数共占 10 列，用%e 格式输出共占 13 列，用%g 格式输出共占 7 列（自动从以上两种格式中选择较短的一种%f 格式），且按%f 格式用小数形式输出，最后 3 个小数位的“0”为无意义的 0，不输出，因此输出 543.321。

注意，%g 格式使用频率较低。

以上介绍了 9 种格式字符，表 3.1 对它们进行了归纳。

表 3.1 printf 格式字符

格式字符	格式字符含义
d	以十进制数形式输出带符号整数（正数不输出符号）
o	以八进制数形式输出无符号整数（不输出前缀 0）
x，X	以十六进制数形式输出无符号整数（不输出前缀 0x），用小写 x 时，十六进制数的 a～f 以小写字母形式输出，用大写 X 时，则以大写字母形式输出
u	以十进制数形式输出无符号整数
c	以字符形式输出，只输出单个字符
s	输出字符串
f	以小数形式输出单、双精度实数，隐含输出 6 位小数
e，E	以指数形式输出单、双精度实数，用小写的 e 时指数以“e”表示（如 1.123000e+003），用大写的 E 时指数以“E”表示（如 1.123000E+0003）
g，G	选用%f 或%e 中输出宽度较短的一种格式输出单、双精度实数，不输出无意义的 0。用大写 G 时，若以指数形式输出，则指数以大写字母表示

（2）标志

在 C 语言中，标志字符有-、+、#、空格（用□表示）4 种。

① “-”标志字符。输出的数据向左端靠齐，右端补空格。例如，

```
printf("%5d,%-5d",a,a);
```

若 a=123，则输出结果为

```
□□123,123□□
```

② “+”标志字符。输出数值的符号（正号或负号）。例如，

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

若 a=123，b=-123，则输出结果为

```
+123,-123
```

③ “#”标志字符。对 c、s、d、u 格式符无影响；对 o 格式符，在输出数值时要加前缀 0；对 x 格式符，在输出数值时要加前缀 0x；对 e、g、f 格式符，当输出结果有小数时才给出小数点。例如，

```
printf("%#d,%#o,%#x",a,b,c);
```

若 a=1，b=2，c=3，则输出结果为

```
1,02,0x3
```

④ “□”标志字符。输出的数值为正时先输出空格，为负时先输出负号。例如，

```
printf("%□d,%□d",a,b);
```

若 a=123，b=-123，则输出结果为

```
□123,-123
```

（3）输出最小宽度

输出最小宽度是一个十进制整数，用来表示输出数据占用的最小域宽。若实际位数多于数据定义的宽度，则按实际位数输出；若实际位数少于数据定义的宽度，则根据标志字符在左端或右端补空格。例如，

```
printf("%10f,%10.2f,%.2f,%-10.2f",a,a,a,a);
```

若 a=12.345，则输出结果为

```
□12.345000,□□□□□12.35,12.35,12.35□□□□□
```

① 注意：上例中的“10”为指定输出数值的最小宽度，“2”为输出数值要保留的小数位数（最后一位小数要遵循四舍五入规则）。当格式符中未给出数值的宽度而只定义了小数位数时，如格式“%.2f”，则输出数值时在保留小数位后，按数值的实际位数输出。例如，

```
printf("%2s,%7.4s,%.3s,%-7.2s ","WORLD","WORLD","WORLD","WORLD");
```

输出结果为

```
WORLD,□□□WORL,WOR,WO□□□□□
```

② 注意：上例中的“2”“7”为指定输出字符串的最小宽度，“4”“3”为输出字符串时要截取的字符个数。当格式符中未给出字符串的宽度而只定义了要截取的字符个数时，如格式“%.3s”，则输出字符串时只输出截取的字符个数。

（4）精度

精度格式符以“.”开头，后面跟十进制正整数。本项的意义是：如果输出的是数字，那么表示小数的位数，如格式“%.2f”，其中 2 表示小数位数；如果输出的是字符，那么表示输出字符的个数，如格式“%.3s”，其中 3 表示输出字符的个数；如果数据实际位数大于所定义的精度数，那么要截去超过的部分。记住，如果输出的是数字，那么最后一位小数应遵循四舍五入规则。

（5）长度

长度格式符为 h 和 l 两种，h 表示按短整型输出，l 表示按长整型输出。

注意：对于 long 型数据应当用%ld 格式输出，long 型数据也可以指定字段宽度。如上例所示，“l”字符还可添加到 o、x、u 格式符之前。

综上所述，在 printf 函数的格式说明中，%和格式字符之间可以插入几种常用的附加符号（又称修饰符），如表 3.2 所示。

表 3.2　printf 的附加格式说明字符

字　符	字符含义
l	表示整数是长整型，可添加到 d、o、x、u 格式符之前
m	代表一个正整数，表示数据的最小宽度，如“%md”
n	代表一个正整数。对于实数，表示输出的是 n 位小数，如“%m.nf”；对于字符串，表示截取的字符个数，如“%m.ns”
-	表示输出的数据在域内向左靠，右端补空格

使用 printf 函数时的注意事项如下：

① 数据类型与格式说明一定要匹配，否则会出现错误。

② 除 X、E、G 可以大写外，其他格式字符必须用小写字母，如%f 不能写为%F。

③ 可以在 printf 函数的“格式控制字符串”内包含“转义字符”，如“\n”“\b”等。

④ 若要输出“%”字符，则可用两个连续的%表示，如 printf("%d%%",6);，其输出为 6%。

⑤ 上面介绍的 9 种格式字符（d、o、x、u、c、s、e、f、g）必须以“%”开头，以上述 9 种格式字符之一结束，才能作为格式字符，否则将作为普通字符原样输出。例如，

```
printf("d=%d,c=%c",d,c);
```

若 d=123，c='b'，则输出结果为

```
d=123,c=b
```

第一次出现的 d、c 是普通字符，原样输出；第二次出现的 d、c 是格式字符；第三次出现的 d、c 则是需输出的数据。

3.3.2 格式输入函数 scanf

scanf 函数称为格式输入函数，我们在前面的例题中已初步接触了 scanf 函数，它的作用是按用户指定的格式从标准输入设备（如键盘）把数据输入指定的变量。

1. scanf 函数调用的一般形式

scanf 函数是一个标准库函数，其函数原型在头文件“stdio.h”中，与 printf 函数相同。调用的一般形式为

```
scanf (“格式控制字符串”，地址表列)
```

例如，

```
scanf("%d%d",&a,&x)
```

其中，用双引号括起来的格式控制字符串用于指定输入格式，作用与 printf 函数中的相同，但 scanf 函数中不能显示非格式字符串，即不能显示提示字符串。

地址表列是由若干地址组成的表列，可以是变量的地址，或是字符串的首地址。例如，&a 和&x 分别表示变量 a 和变量 x 的地址，这些地址是由编译系统在内存中给变量 a 和 x 分配的地址，在编译连接阶段分配。&a 和&x 中的“&”是“地址运算符”。

上面的 scanf 函数的作用是：按照 a、x 在内存中的地址将 a、x 的值存入。其中“%d%d”表示按十进制整数形式输入数据。输入数据时，由于没有非格式字符在“%d%d”之间作为输入时的间隔，因此在输入时要用一个或多个空格间隔，也可以用回车键或跳格键 Tab 作为每两个输入数之间的间隔，而不能用其他字符（如“,”）作为输入数据间的分隔符。例如，

① 1□□2↙

② 1↙2↙

③ 1（按 Tab 键）2↙

④ 1,2↙ （不合法）

注意：使用 scanf 函数时，应弄清楚变量的地址和变量的值这两个概念。变量的地址是 C 编译系统分配的，变量的值是用户从键盘输入的。例如，&a 表示变量 a 的地址，若从键盘输入 1 赋值给变量 a，那么，1 就是变量 a 的值。

2. 格式控制字符串

格式控制字符串的一般形式为

```
%[*][输入数据宽度][长度]类型
```

其中带方括号的项为任选项。各项的意义如下。

（1）类型

表示输入数据的类型，其格式字符及含义如表 3.3 所示。

表 3.3　scanf 格式字符

格式字符	格式字符含义
d	输入带符号的十进制整数
o	输入无符号的八进制整数
x，X	输入无符号的十六进制整数（大小写作用相同）
u	输入无符号的十进制整数
c	输入单个字符
s	输入字符串
f	用小数形式输入实数
e，E；g，G	与 f 作用相同，三者可以相互替换（大小写作用相同）

（2）“*”符

“*”符用来表示该输入项在读入后不赋给相应的变量，即跳过该输入值。例如，

```
scanf("%2d%*2d%3d",&a,&b);
```

当输入信息为 1234567 时，将把 12 赋给 a，%*2d 表示读入 2 位整数但并不赋给任何变量，即 34 被跳过，再读入 3 位整数 567 赋给 b。当一批数据中不需要某些数据时，可以利用此方法“跳过”它们。

（3）输入数据宽度

用十进制整数来指定输入数据的宽度，即指定输入数据所占的列数，系统会自动按宽度截取所需的数据。例如，

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

输入

```
1234512345↙
```

系统自动将 123 赋给变量 a，将 4512 赋给变量 b，其余部分（345）被截去。

此方法也可用于字符型。例如，

```
scanf("%5c",&ch);
```

输入

```
abcdefg↙
```

由于 ch 只能容纳一个字符，所以系统把读入的 5 个字符（abcde）中的第一个字符'a'赋给变量 ch，其余部分（fg）被截去。

（4）长度

长度格式符有 l 和 h 两种。l 用于输入长整型数据（如%ld、%lo、%lx）和双精度浮点数（如%lf、%le）；h 用于输入短整型数据（如%hd、%ho、%hx）。

综上所述，在 scanf 函数的格式说明中，%和格式字符之间可以插入几种常用的附加符号（又称修饰符），如表 3.4 所示。

3. 使用 scanf 函数时应注意的几个问题

1）在 scanf 函数中没有精度控制。例如，

```
scanf("%6.2f",&x);
```

是非法的，不能企图用此语句输入小数为 2 位的实数。

表 3.4 scanf 的附加符号

字　符	字符含义
L	表示输入长整型数据
H	表示输入短整型数据
域宽	指定输入数据所占列宽，为正整数
*	表示该输入项在读入后不赋给相应变量

2）scanf 函数中“格式控制字符串”之后应该是变量地址，而不应该是变量名。例如，

```
scanf("%d%d",a,c);
```

是非法的，应改为

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

才是合法的。这是 C 语言与其他高级语言的不同之处，初学者一定要注意不要在此出错。

3）在输入多个数值数据时，若“格式控制字符串”中没有非格式字符作为输入数据之间的间隔，则可用空格、Tab 或回车键作为间隔。在编译时，若遇到空格、Tab、回车键或非法数据及指定的宽度，则认为该数据结束。例如，

```
scanf("%d%c%4d",&a,&ch,&b);
```

若输入

```
123A12345↙
```

则在输入 123 之后遇到字符'A'（非数字），因此认为第一个数据到此结束，即把 123 赋给变量 a。接下来将字符'A'赋给变量 ch，注意在该字符之后不需要加空格，因为%c 只要求输入一个字符。最后将数值 1234 赋给变量 b（只取 4 列，5 被截去）。

4）在输入字符数据时，若“格式控制字符串”中没有非格式字符，则认为所有输入的字符（如空格和转义字符）均为有效字符。例如，

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

若输入

```
a□b□c↙
```

则把字符'a'赋给 c1，字符'□'赋给 c2，字符'b'赋给 c3。初学者一定要注意空格（'□'）字符也是一个字符，这里不能用空格作为两个字符的间隔。只有当输入为

```
abc↙
```

时，才能把'a'赋给 c1，'b'赋给 c2，'c'赋给 c3。

如果在“格式控制字符串”中加入空格作为间隔，例如把上例改为

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

则输入时各数据之间可加空格（如 d□e□f↙）。

5）如果在“格式控制字符串”中除格式说明外，还有其他非格式字符，那么输入时也要输入与这些非格式字符相同的字符。例如，

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

其中用非格式字符“,”作为间隔符。

若输入：

① 10,11,12↙　　（合法）

② 10□11□□12↙（不合法）

③ 10;11.12↙　　（不合法）

注意：间隔符是“,”时，输入数据之间只能用“,”作为分隔符，而不能用空格或其他字符。又如

```
scanf("x=%d,y=%d",&x,&y);
```

若输入：

① x=13, y=14↙　　（合法）

② x:13, y:14↙　　（不合法）

3.3.3　字符输出函数 putchar

putchar 函数是字符输出函数，它的作用是向终端输出设备（如显示器）输出单个字符。其一般形式为

```
putchar(字符变量)
```

例如，

```
putchar('a');          //输出小写字母 a
putchar(c);            //输出字符变量 c 的值
putchar('\n');         //换行，使输出的当前位置移到下一行的开头
```

对控制字符则执行控制功能，也可以输出其他转义字符。例如，

```
putchar('\\');         //输出反斜杠字符
putchar('\'');         //输出单引号字符
putchar('\r');         //回车，使输出的当前位置移到本行的开头
```

在程序中，若使用到 putchar 函数，则必须要用文件包含命令#include <stdio.h>（该命令一般放在文件开头）。例如，

```
#include <stdio.h>
int main()
{
 char a='O', b='K';
 putchar(a); putchar(b); putchar('\n');
 putchar(a); putchar('\n');putchar(b); putchar('\n');
 return 0;
}
```

输出结果为

```
OK
O
K
```

3.3.4　字符输入函数 getchar

getchar 函数是字符输入函数，它的作用是从终端输入设备（如键盘）输入一个字符。其一般形式为

```
getchar()
```

该函数的值就是从输入设备得到的字符。使用 getchar 函数时，通常把输入的字符赋给一个字符变量，构成赋值语句。例如，

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

```
int main()
{
 char c;
 printf("input a character:");
 c=getchar();
 putchar(c);
 return 0;
}
```

在运行时，如果从键盘输入字符'C'并按回车键，那么会在屏幕上看到输出的字符'C'。

```
C↙          （输入'C'之后按回车键，字符才能送到内存，赋给变量 c）
C           （输出变量 c 的值'C'）
```

注意：

① 在程序中，若用到 getchar 函数，则同样要用文件包含命令#include <stdio.h>。

② getchar 函数只能接收单个字符，即使输入数字也要按字符处理。输入多于一个字符时，只接收第一个字符。

③ getchar 函数得到的字符可以不赋给任何变量，而只作为表达式的一部分。例如，

```
putchar(getchar());
```

若 getchar()的值为'C'，那么 putchar 函数输出'C'。

当然，也可以用 printf 函数输出：

```
printf("%c",getchar());
```

在 C 语言中，输入/输出是最基本的操作，几乎每个程序都会包含输入/输出语句。由于对格式输入/输出的规定比较烦琐，用得不合适就得不到预期的结果，因此对编程初学者而言，要想学好这部分知识，就要花费一定的时间和精力。建议初学者多上机练习，重点掌握最常用的一些规则，其他部分需要时再查阅。

3.4 顺序结构程序设计举例

顺序结构程序设计，就是将 C 语句按照正确的逻辑规则自上而下地书写。顺序程序设计的步骤归纳如下。

（1）若程序中使用了标准库函数，则应使用编译预处理命令包含相应的头文件。

（2）在函数体中，首先定义变量，为变量分配内存空间。

（3）通过赋值语句或输入函数为定义的变量赋初值。

（4）计算部分。

（5）用输出函数输出结果。

下面介绍几个顺序程序设计的例子。

【例 3.1】编写程序，输入两个整数 a 和 b，交换它们的值，并输出交换前后的数。

分析：要交换两个整数的值，需要引入第三个变量 t 来完成交换工作。先将 a 赋给 t，用它来临时保存 a 的值；再将 b 赋给 a，这时 a 就得到了 b 的值；由于 a 的值已存放于 t 中，最后将保存 a 的变量 t 赋给 b，这样就完成了两个整数的交换。

程序流程图如图 3.5 所示。程序代码如下：

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

```
int main()
{
 int a,b,t;
 printf("Please input two integer numbers:");
 scanf("%d,%d",&a,&b); /*输入 2 个用逗号分隔的整数*/
 printf("before changed:a=%d b=%d\n",a,b);
 t=a;                  /*交换两个整数的值*/
 a=b;
 b=t;
 printf("after changed:a=%d b=%d\n",a,b);
 return 0;
}
```

运行结果如下：

```
Please input two integer numbers:7,9↙
before changed:a=7 b=9
after changed:a=9 b=7
```

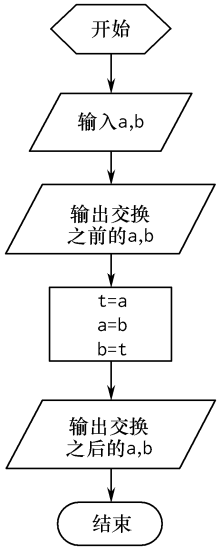

图 3.5　例 3.1 程序流程图

在 Microsoft Visual C++ 2010 Express 环境下，程序的运行结果如图 3.6 所示。

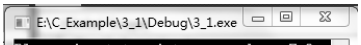

图 3.6　例 3.1 程序的运行结果

注意：本程序是一个表达顺序结构思想的经典例子，程序中第 7、8、9 行三条语句的顺序不能随便交换，执行顺序一旦改变，就不能达到预期目的。

【例 3.2】从键盘输入任意 3 个整数，求出三者之和及平均值。

分析：首先定义 3 个整型变量，从键盘输入 3 个整数存入变量 iNum1、iNum2 和 iNum3；接下来对 3 个整数求和，赋给变量 iSum；再对 3 个整数求平均值，赋给变量 rAverage；最后输出

iSum 和 rAverage。

程序流程图如图 3.7 所示。程序代码如下：

```
#include <stdio.h>
int main()
{
  int iNum1,iNum2,iNum3,iSum;
  float rAverage;
  printf("Please input three integer numbers:");
  /*输入 3 个整数*/
  scanf("%d%d%d",&iNum1,&iNum2,&iNum3);
  iSum=iNum1+iNum2+iNum3;              /*求和*/
  rAverage=iSum/3.0;                   /*求平均值*/
  printf("iSum=%8d\n",iSum);           /*输出和*/
  printf("rAverage=%7.2f\n",rAverage);/*输出平均值*/
  return 0;
}
```

运行结果如下：

```
Please input three integer numbers:13□14□15↙
iSum=□□□□□□42
rAverage=□□14.00
```

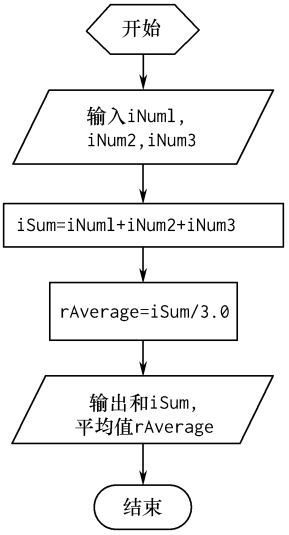

图 3.7　例 3.2 程序流程图

在 Microsoft Visual C++ 2010 Express 环境下，程序的运行结果如图 3.8 所示。

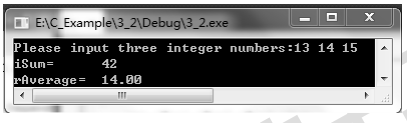

图 3.8　例 3.2 程序的运行结果

注意：程序内 rAverage=iSum/3.0 中的 3.0 不能改为 3，请读者自己分析。

【例 3.3】用格式输入函数输入 4 个小写字母，并用格式输出函数输出这 4 个字母对应的大写字母及它们的 ASCII 码值。

分析：首先定义 4 个字符型变量，调用格式输入函数从键盘输入 4 个字母存入变量 c1、c2、c3 和 c4；然后将小写字母转换为大写字母；最后顺序输出大写字母和 ASCII 码值。

程序流程图如图 3.9 所示。程序代码如下：

```
#include <stdio.h>
int main()
{
  char c1,c2,c3,c4;
  printf("Please input four characters:");
  scanf("%c,%c,%c,%c",&c1,&c2,&c3,&c4);/*输入 4 个字母*/
  c1=c1-32;                        /*将小写字母转换成大写字母*/
  c2=c2-32;
  c3=c3-32;
  c4=c4-32;
  printf("%c%5d\n",c1,c1);         /*输出大写字母和 ASCII 码值*/
  printf("%c%5d\n",c2,c2);
  printf("%c%5d\n",c3,c3);
  printf("%c%5d\n",c4,c4);
  return 0;
 }
```

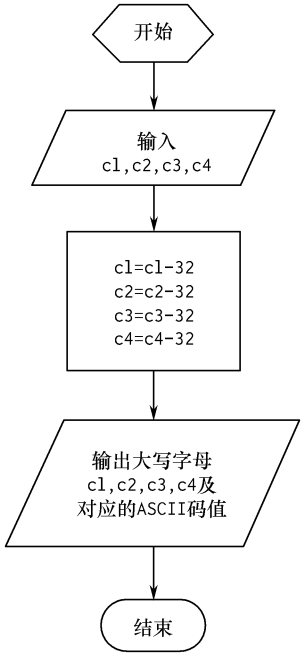

图 3.9　例 3.3 程序流程图

运行结果如下：

```
Please input four characters:a,b,c,d↙
A□□□65
B□□□66
```

```
C□□□67
D□□□68
```

在 Microsoft Visual C++ 2010 Express 环境下，程序的运行结果如图 3.10 所示。

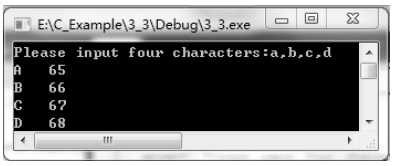

图 3.10　例 3.3 程序的运行结果

【例 3.4】输入三角形的三边长，求三角形的面积。已知三角形的三边长 a, b, c 能构成三角形，这个三角形的面积公式为 $\text{area}=\sqrt{s(s-a)(s-b)(s-c)}$，其中 $s=(a+b+c)/2$。

分析：首先定义三角形的三边长 a、b、c，以及 s、面积 rArea 共 5 个实型变量，调用格式输入函数从键盘输入 3 个数存入变量 a、b、c；然后根据公式求出面积赋给 rArea；最后将结果输出。

程序流程图如图 3.11 所示。程序代码如下：

```
#include <stdio.h>
#include <math.h>
int main()
{
   float a,b,c,s,rArea;
   printf("Please input three numbers:");
   scanf("%f,%f,%f",&a,&b,&c);        /*输入三角形的三边长*/
   s=1.0/2*(a+b+c);
   rArea=sqrt(s*(s-a)*(s-b)*(s-c));   /*求出三角形的面积*/
   printf("a=%7.2f,b=%7.2f,c=%7.2f,s=%7.2f\n",a,b,c,s);
   printf("area=%7.2f\n",rArea);      /*输出三角形面积*/
   return 0;
}
```

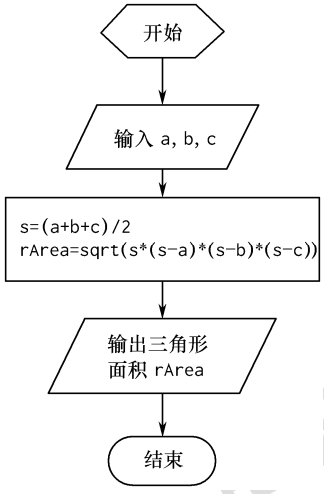

图 3.11　例 3.4 程序流程图

运行结果如下：

```
Please input three numbers:3,4,5↙
a=□□□3.00,b=□□□4.00,c=□□□5.00,s=□□□6.00
area=□□□6.00
```

在 Microsoft Visual C++ 2010 Express 环境下，程序的运行结果如图 3.12 所示。

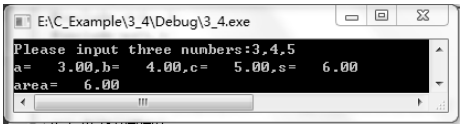

图 3.12　例 3.4 程序的运行结果 1

若从键盘输入 9,12,15，则程序的运行结果如图 3.13 所示。

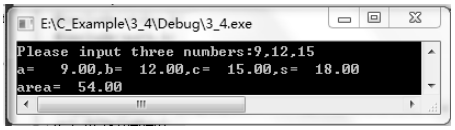

图 3.13　例 3.4 程序的运行结果 2

注意：以后只要在程序中用到标准库函数中的数学函数，都应用#include 命令包含“math.h”头文件。本程序中使用了求平方根函数 sqrt。另外，为简单起见，本程序假设输入的三个数能够构成一个三角形。

【例 3.5】求方程 $ax^2+bx+c=0$ 的根，a, b, c 由键盘输入，设 $b^2-4ac>0$。

分析：由题得知 $b^2-4ac>0$，所以该方程有两个实数解。根据一元二次方程的求根公式得

$$x_1=\frac{-b+\sqrt{b^2-4ac}}{2a},\quad x_2=\frac{-b-\sqrt{b^2-4ac}}{2a}$$

令 $p=\frac{-b}{2a}$，$q=\frac{\sqrt{b^2-4ac}}{2a}$，rDisc=$b^2-4ac$，则有 $x_1=p+q$，$x_2=p-q$。因此，在定义变量时，需定义系数 a、b、c，以及 rDisc、p、q、x1、x2 共 8 个实型变量，然后调用格式输入函数从键盘输入 3 个数存入变量 a、b、c；接下来根据公式求出 x1 和 x2；最后将结果输出。

程序流程图如图 3.14 所示。程序代码如下：

```
#include <stdio.h>
#include <math.h>
int main()
{
  float a,b,c,rDisc,p,q,x1,x2;
  printf("Please input three numbers:");
  scanf("%f,%f,%f",&a,&b,&c);              /*输入方程的 3 个系数*/
  rDisc=b*b-4*a*c;
  p=-b/(2*a);
  q=sqrt(rDisc)/(2*a);
  x1=p+q;                                  /*求出方程的根*/
  x2=p-q;
  printf("x1=%7.2f\nx2=%7.2f\n",x1,x2);    /*输出方程的根*/
  return 0;
 }
```

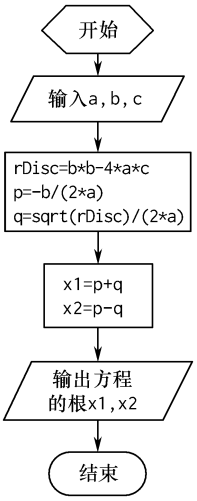

图 3.14　例 3.5 程序流程图

运行结果如下：

```
Please input three numbers:1,3,2↙
x1=□□-1.00
x2=□□-2.00
```

注意：程序中同样使用了预处理命令`#include <math.h>`。

在 Microsoft Visual C++ 2010 Express 环境下，程序的运行结果如图 3.15 所示。

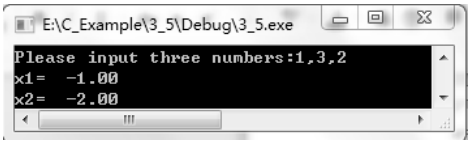

图 3.15　例 3.5 程序的运行结果 1

若从键盘输入 2,8,5，则程序的运行结果如图 3.16 所示。

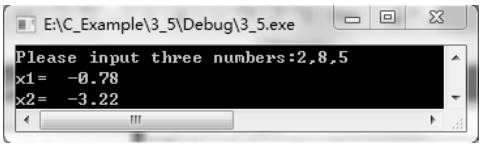

图 3.16　例 3.5 程序的运行结果 2