CHAPTER 3

第3章

顺序结构程序设计

本章学习目标：

- 熟悉 C 语言程序结构；
- 掌握 C 语言语句的分类；
- 掌握 C 语言输入输出函数语句；
- 熟练使用 C 语言语句编写顺序结构程序。

本章介绍 C 语言的基本语句和顺序结构程序设计，为后续复杂程序设计打基础。结构化程序设计的顺序、分支和循环三种结构中，顺序结构是程序的基本结构，也是程序的默认结构。在掌握 C 语言基本语句的基础上，要能够熟练编写顺序结构的 C 语言程序。

3.1 C 语言语句概述

3.1.1 C 语言程序结构

在第 2 章中，我们学习了 C 程序中的一些基本要素，如变量、常量、运算符和表达式等。本章主要介绍如何用这些基本元素构成 C 语言语句，如何使用 C 语言语句编写简单 C 语言程序。

通过第 1 章的学习我们知道，程序是计算机指令的集合，可执行程序是二进制机器指令的集合。C 语言作为高级语言，使用人类容易理解的语言来编写程序，C 语言程序是由若干条高级语言语句构成的，一条 C 语言语句编译之后形成若干条机器指令。若干条 C 语言语句按照某种结构形式组织在一起完成一个相对独立的功能，就形成了 C 语言函数，C 语言程序从整体结构上看就是由若干个函数构成的。C 程序源文件结构可以用图 3-1 表示。一个 C 程序的源程序文件可以由若干个函数和预编译命令组成，一个函数由数据定义部分和执行语句部分组成。

C 程序函数中一般都包括数据描述（数据定义部分）和数据操作（语句部分）两部分。数据定义部分主要定义数据结构（用数据类型表示）和数据初值。数据操作的任务是对已提供的数据进行加工，这些数据加工是由程序语句来完成。

数据加工逻辑可以是顺序执行一系列简单加工，也可以是根据条件选择执行某些加工工件或者循环执行某些加工操作，所以程序流程控制的三种基本结构：顺序、分支和循环。

本章中主要介绍 C 语言流程控制中默认的顺序结构。顺序结构程序就是顺序执行若干语句操作，例如两条语句：A 和 B，按顺序先执行 A 再执行 B，如图 3-2 所示。

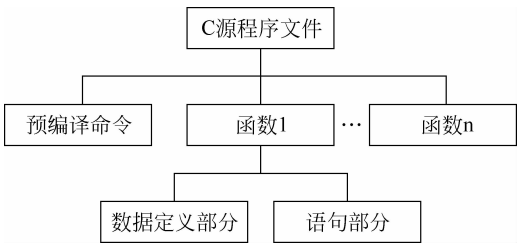

图 3-1　C 程序源文件结构图

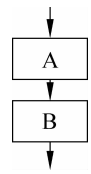

图 3-2　顺序结构程序流程图

3.1.2　C 语言语句分类

C 语言语句大体上可以分 5 大类。

1. 流程控制语句

控制语句是完成一定控制功能的语句，在 C 程序中有 9 种控制语句，分别是：

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　　终止执行 switch 或循环语句
⑦ switch　　多路分支选择语句
⑧ goto　　转向语句
⑨ return　　从函数返回语句

上面的 9 种语句中的括号( )表示其中是一个条件，～表示内嵌的执行语句。例如："for( ) ～"具体的语句可以写成：

```
for(i = 1;i <= 100;i++)
            sum += i;
```

2. 函数调用语句

由函数调用加一个分号构成一个语句，例如：

```
printf("This is a C program.\n");
```

3. 表达式语句

由一个表达式加一个分号构成一个语句，最典型的是赋值语句，例如：

```
i = 1;
```

任何表达式都可以加上一个分号成为语句，例如：

```
i++;
```

4. 空语句

空语句就是一个分号，例如：

```
;
```

空语句什么也不执行，有时用来做转向点，或循环语句中的循环体(循环体是空语句，表示循环体什么也不做)。

5. 复合语句

用“{}”把一些语句括起来，使之成为复合语句，例如：

```
{
    sum += i;
    i++;
}
```

注意：复合语句形式上当一条语句使用，可以用于循环语句中的循环体语句或分支语句中的执行语句。另外，复合语句中可以进行数据定义，但定义的变量仅在该复合语句中有效。

单纯的顺序结构程序不需要流程控制语句，所以顺序程序语句一般由表达式语句、函数调用语句、复合语句以及空语句构成。本章重点介绍后续程序设计中频繁用到的输入、输出函数语句。

3.2 输入输出函数语句

3.2.1 流

C 语言没有用于完成 I/O(输入输出)操作的关键词，输入和输出操作时用库函数来完成，在讨论 C 语言的 I/O 系统之前有必要先搞清楚“流”的概念。C 语言 I/O 系统为编程者提供了一个统一的接口，与具体的被访问设备无关，也就是说，C 语言 I/O 系统在编程者和被使用设备之间提供了一层抽象的东西，这就是“流”。而具体的实际设备是“文件”，文件将在以后的章节中介绍。

ANSI 标准中定义了一组完整的 I/O 操作函数，叫做“缓冲文件系统”，缓冲文件系统指系统自动地在内存区为每一个正在使用的文件名开辟一个缓冲区，磁盘向内存读入数据或是内存向磁盘输出数据必须先通过内存缓冲区操作。

缓冲文件系统在设计上可以支持多种设备，包括终端(系统隐含指定的输出设备)、磁盘、键盘等。虽然各种设备差别很大，但缓冲文件系统把每个设备都转换成一个逻辑设备，叫做流。所有流都具有相同的行为，因为流很大程度上与设备无关，一个用来进行磁盘文件写入操作的函数也可以用来进行控制台设备(屏幕、键盘)写入。流分为两种类型：文本流和二进制流。文本流是一行行字符，换行符表示一行的结束，所读写的字符与外部设备中的字符没有一一对应的关系。二进制流是由与外部设备中内容一一对应的系列字节组成的，使用时没有字符翻译过程，所读写的字节数目与外设中的数目也相同。

对编程人员而言，所有的 I/O 操作都通过流进行，所有的流都是一样的，都是一系列的字符，编程人员无须考虑具体的物理设备，而只针对这个逻辑设备——“流”去考虑编程问题就行了。

3.2.2 字符输入输出函数

1. putchar 函数

1) 函数形式

```
putchar(c);
```

注意：c 是字符变量或是整型变量。

2) 函数功能

putchar 函数的作用是向终端输出一个字符，函数调用 putchar(c)时向终端上输出字符变

量 c 的值。

在使用标准 I/O 库函数时，需要用预编译命令“#include”将“stdio.h”文件包含到用户源文件中。即：

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

stdio.h 是标准输入输出的头文件，它包含了与标准 I/O 库有关的变量定义和宏定义。在需要使用标准 I/O 库中的函数时，应该在程序前使用预编译命令将“stdio.h”文件包含进来（在使用 printf 和 scanf 函数时可以例外）。

2. getche 函数

1）函数形式

```
getche( );
```

2）函数功能

getche 函数的作用是从键盘上读入一个字符，函数调用 getche( )时，等待从键盘上键入一个字符，返回它的值并在屏幕上自动回显该字符。

【例 3-1】 从键盘输入一个小写字母，转换成大写字母输出。

分析：查阅 ASCII 码表，大小写字母的 ASCII 码值差为 32，只需判断出输入的字母是大写字母或小写字母就可以加 32 或是减去 32 进行转换。

```
/* 程序 3-1,源程序文件 pro3-1.c */
#include<stdio.h>
main( )
{
    char ch;
    printf("请输入一个字符:\n");
      ch=getche();
      printf("\n 您输入的字符对应的大写字符: \n");
    putchar(ch-32);
    putchar('\n');
}
```

程序的运行结果如图 3-3 所示。

图 3-3　例 3-1 运行结果

说明：putchar('\n')；输出转义字符回车换行符，ch 得到从键盘上输入的字符，“ch-32”为对应的大写字符，getche 函数在这里可以换成 getchar 函数，调用 getchar 函数时，用户输入的字符显示在屏幕上，而且等用户按 Enter 键后才运行。好处是用户可以先判断输入内容是否正确，当输入错误后可以删掉输入的字符重新输入，即执行过程完全由用户控制。调用 getche 函数时，用户输入的字符也显示在屏幕上，但不等待用户按 Enter 键就运行。

思考：如果想输入一个大写字母转换成小写字母怎么办？

3.2.3　格式输入输出函数

1. printf 函数

1）函数形式

```
printf("控制字符串",输出参量表);
```

控制字符串由两种不同类型的字符组成，第一类是原样字符，函数把它们显示在屏幕上；

另一类是格式说明字符，它们定义输出参量的显示格式。每个格式说明的开头是一个百分号“%”，后面是一个类型字符。格式字符说明如表 3-1 所示。输出参量的个数与控制字符串中格式说明的个数应该一致，顺序上也一一对应。例如：

```
printf("max = %d min = %d\n",max,min);
```

原样字符分为普通字符和转义字符。例如：

```
printf("Hello,world!");
```

的作用是向屏幕上显示输出“Hello，world!”，字符均为普通字符。而

```
printf("\\A\102\C\\");
```

中“\\”、“\102”就是转义字符，分别向屏幕上显示输出反斜线“\”和大写字母“B”。

表 3-1　printf 函数格式字符说明

格式说明字符	作　　用
d	按照有符号的十进制形式输出整数(正整数不带＋号)
o	按照无符号八进制形式输出整数(不带前导符 0)
x	按照无符号十六进制形式输出整数(不带前导符 0x)
u	按照无符号十进制形式输出整数
c	按照字符形式输出一个字符
s	按照字符形式输出一个字符串
e	按照指数形式输出浮点型单、双精度数，数字部分小数位数隐含输出 6 位
f	按照小数形式输出浮点型单、双精度数，隐含输出 6 位小数
g	按照%e 或%f 格式中输出宽度较短的一种格式输出

2）函数功能

printf 函数的功能是向终端(没有特别说明时多指屏幕)输出若干个任意类型的数据。前面所提及的 putchar 函数只能输出一个字符，printf 函数可以输出多个数据，且类型也是任意的。

在格式字符说明中，还可以在“%”和格式字符说明间插入附加格式字符，如表 3-2 所示。

表 3-2　附加格式字符说明

格式说明字符	作　　用
l	用于输出长整型数据，可以加在格式字符 d、o、x、u 前面
m	输出数据的宽度
.n	对浮点型数据，表示输出 n 位小数；对字符串，表示截取字符个数
0	输出数值时指定左面不使用的空位置自动填 0
#	在八进制和十六进制数前显示前导 0，0x
—	输出的数字或字符在输出区域内左对齐
＋	指定在有符号数的正数前显示正号(＋)

3）格式字符使用说明

① %d 格式字符，用于输出有符号十进制整数，具体用法如下：

- %d，按整型数据的实际长度输出；
- %md，m 为指定的输出宽度，如果输出整型数据的位数小于 m，则左端补空格，一个字

符位补一个空格；若数据位数大于 m，则按实际位数输出；

- %ld，输出长整型 long 数据。

【例 3-2】 输出整型数据。

```
/* 程序 3-2,源程序文件 pro3-2.c */
#include <stdio.h>
main( )
{
    int a = 1,b = 2;
    long c = 3;
    printf(" %d\n %10d\n %ld\n",a,b,c);
}
```

程序的运行结果如图 3-4 所示。

② %o 格式字符，用于输出八进制整数。例如：

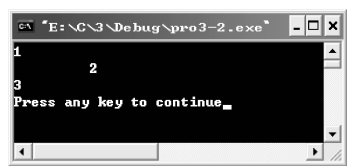

图 3-4　例 3-2 运行结果

```
int a = 8;
printf(" %d, %o",a,a);
```

输出时显示 8，10。

③ %x 格式字符，用于输出十六进制整数。例如：

```
int a = 255;
printf(" %d, %x",a,a);
```

输出时显示 15，ff。

④ %u 格式字符，用于输出 unsigned 型整数，即无符号数，以十进制形式输出。

有符号整数和无符号整数都可以用%u 格式输出。例如：

```
short int a = 65535;
int b = 65535;
printf(" %d, %u",a,b);
```

输出时显示－1，65535。

⑤ %c 格式字符，用于输出一个字符。%c 不仅能输出字符型变量的值，而且当一个整型变量的数值在 0～255 范围内时，也可以按字符形式输出。例如：

```
char a = 'c'; int b = 97;
printf(" %c, %c",a,b);
```

输出时显示 c，a。

⑥ %s 格式字符，用于输出一个字符串。例如：

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

输出时显示 C program。

⑦ %f 格式字符，用于输出浮点型数据（包含单、双精度），输出时以小数形式表示。用法如下：

- %f，不指定宽度，由系统自动指定，整数部分全部输出，小数部分输出 6 位。
- %m.nf，指定输出数据占 m 位宽度，其中含 n 位小数。如果数据长度小于 m，则左端补空格；如果数据长度大于 m，则按实际长度输出。

【例 3-3】 按小数形式输出浮点型数据。

```
/* 程序 3-3,源程序文件 pro3-3.c */
#include <stdio.h>
```

```
main()
{
        float f = 1.2345;
        printf(" % f, % 10f, % 10.2f\n",f,f,f);
}
```

程序的运行结果如图 3-5 所示。

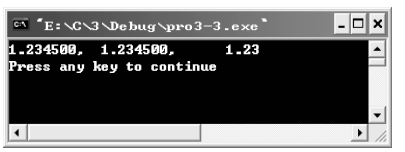

图 3-5 例 3-3 运行结果

⑧ %e 格式字符,用于输出浮点型数据,按指数形式输出。

- %e,不指定输出数据所占的宽度和数字部分的小数位数,由系统自动给定 6 位小数,指数部分占 5 位,其中 e 占一位,指数符号占一位,指数占 3 位。数值按标准化指数形式输出。
- %m.ne,此处 m 指输出宽度,与上面所述一致。n 指数据的小数部分位数。

【例 3-4】 按指数形式输出浮点型数据。

```
/ * 程序 3-4,源程序文件 pro3-4.c * /
# include < stdio.h >
main( )
{
        float f = 123.45678;
        printf(" % e, % 10e, % 10.2e\n",f,f,f);
}
```

程序的运行结果如图 3-6 所示。

⑨ %g 格式字符,用于输出浮点型数据,根据输出数值长度的大小,自动选择 f 格式或 e 格式宽度较小的一种,且不输出无意义的零。

【例 3-5】 按指数或小数形式中较短的一种方式输出浮点型数据。

```
/ * 程序 3-5,源程序文件 pro3-5.c * /
# include < stdio.h >
main( )
{
        float f = 123.45678;
        printf(" % e, % f, % g\n",f,f,f);
}
```

程序的运行结果如图 3-7 所示。

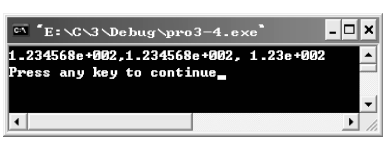

图 3-6 例 3-4 运行结果

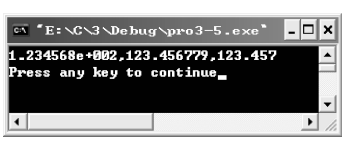

图 3-7 例 3-5 运行结果

注意:

(1) 格式字符为小写,如%d 不能写成%D;

(2) 可以在格式控制字符串中加入转义字符,如“\n”、“\t”等;

(3) 如果想输出“%”,应该用连续两个百分号表示,如“%%”。

2. scanf 函数

1) 函数形式

```
scanf("控制字符串",输入参量表);
```

控制字符串包含三种不同的字符内容：格式说明、空白字符和非空白字符。输入格式说明也是一个百分号加上格式字符，这个格式说明告诉 scanf 函数将读入什么类型的数据。格式说明字符如表 3-3 所示。

表 3-3　scanf 函数格式字符说明

格 式 字 符	说　　明
d	按照有符号的十进制形式输入整数
o	按照无符号八进制形式输入整数
x	按照无符号十六进制形式输入整数
c	输入一个字符
s	输出一个字符串
e	按照指数形式或小数形式输入浮点型单、双精度数
f	按照指数形式或小数形式输入浮点型单、双精度数。e 和 f 可以互换

对格式控制字符串中的空白字符处理，scanf 函数会在读操作中略去输入流中的一个或多个空白字符。空白字符可以是空格、制表符或换行符。实际上，在控制字符串中有空白字符时 scanf 函数在读操作时全部读入，但是并不存储它们，遇到非空白字符（包括 0 在内）才存储起来。例如“%d%d”使 scanf 函数先读入一个整型数据，读入遇到空格、制表符或换行符时，再读入另一个整型数据，最后通过换行符结束输入（Enter 键）。

一个非空白字符会使 scanf 函数在读入时排除与这个非空白字符相同的字符。例如“%d,%d”使 scanf 函数先读入一个整型数据，然后排除读入的逗号（不存储），再读入另一个整型数据，最后通过换行符结束输入（Enter 键）。如果读入数据时没有遇到这个非空白字符，scanf 函数就终止了。

输入参量表中用于接收数值的变量必须用它们的地址来表示，即参量表中列出的必须是指向这些变量的指针。对于单个变量（指针变量除外）而言，其指针表示是在变量名前面加地址运算符“&”。

2）函数功能

scanf 函数的功能是输入若干个任意类型的数据，并存储在内存相应位置中，程序中以变量形式表示。前面所提及的 getche 函数只能输入一个字符，scanf 函数可以输入多个数据，且类型也是任意的。

与 printf 函数一样，scanf 函数在格式字符说明中也可以在%和格式字符间插入附加格式字符，如表 3-4 所示。

表 3-4　附加格式字符说明

格 式 字 符	说　　明
l	输入长类型数据，可用在整型 d、o、x 前面，以及浮点型的 f 和 e 前面
h	输入短类型数据，可用在整型 d、o、x 前面
m	输入数据所占的宽度
*	表示本输入项在读入后不赋给相应的变量

【例 3-6】　输入整型数据。

分析：输入多个数据时，既可以采用空白字符也可以采用非空白字符分隔数据。本例中，用逗号作为分隔符分隔多个输入数据，属于非空白字符。

```
/* 程序 3-6,源程序文件 pro3-6.c */
#include <stdio.h>
main( )
{
        int a,b,c;
        scanf("%d,%o,%x",&a,&b,&c);
        printf("a=%d\nb=%d\nc=%d\n",a,b,c);
}
```

运行输入“15,017,0xf”后,程序的运行结果如图 3-8 所示。

【例 3-7】 %c 格式输入字符型数据。

```
/* 程序 3-7,源程序文件 pro3-7.c */
#include <stdio.h>
main( )
{
        char ch1,ch2,ch3;
        scanf("%c,%c,%c",&ch1,&ch2,&ch3);
        printf("%c\n%c\n%c\n",ch1,ch2,ch3);
}
```

运行输入“a,b,c”后,程序的运行结果如图 3-9 所示。

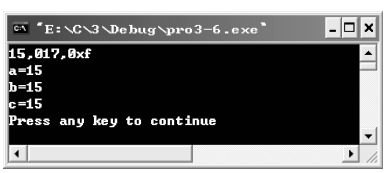

图 3-8 例 3-6 运行结果

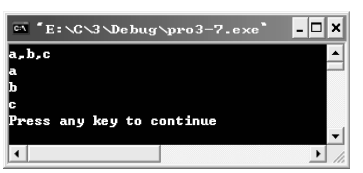

图 3-9 例 3-7 运行结果

思考:如果将例 3-7 中“scanf("%c,%c,%c",&ch1,&ch2,&ch3);”这句中格式控制字符串里面“%c”之间的逗号去掉,运行输入“a,b,c”或“a b c”将会得到什么显示结果?

【例 3-8】 %s 格式输入字符型数据。

```
/* 程序 3-8,源程序文件 pro3-8.c */
#include <stdio.h>
main( )
{
        char str[20];
        scanf("%s",str);
        printf("%s\n",str);
}
```

运行输入 abbcdef 后,程序的运行结果如图 3-10 所示。

【例 3-9】 输入浮点型数据。

```
/* 程序 3-9,源程序文件 pro3-9.c */
#include <stdio.h>
main( )
{
        float f1,f2;
        scanf("%f%e",&f1,&f2);
        printf("%.2f\n%.2f\n%.2e\n%.2e\n",f1,f2,f1,f2);
}
```

运行输入“1.23 1.23e+002”后,程序的运行结果如图 3-11 所示。

图 3-10　例 3-8 运行结果

图 3-11　例 3-9 运行结果

3.3　顺序结构程序举例

【例 3-10】　鸡兔同笼问题，已知鸡和兔的总数为 40，总脚数为 120，求鸡兔各有多少只。

分析：可以根据已知条件列二元一次方程组，设鸡的数量为 x，兔的数量为 y，则方程组为：$\begin{cases} x+y=40 \\ 2x+4y=120 \end{cases}$。求解方程组，x=(4×40−120)/2，y=(120−2×40)/2。

程序如下：

```
/* 程序 3-10,源程序文件 pro3-10.c */
#include <stdio.h>
main( )
{
        int x,y,m,n;      //x 为鸡的数量,y 为兔的数量,m 是头的数量,n 是脚的数量
        m = 40;
        n = 120;
        x = (4 * m - n)/2;
        y = (n - 2 * m)/2;
        printf("鸡的数量: %d; 兔的数量: %d\n",x,y);
}
```

图 3-12　例 3-10 运行结果

程序的运行结果如图 3-12 所示。

【例 3-11】　输入三角形的三边长，求三角形的面积。

分析：利用三角形三边长求面积需要用海伦公式，$s=\frac{1}{2}\times(a+b+c)$，$area=\sqrt{s\times(s-a)\times(s-b)\times(s-c)}$。

注意：因为需要用到开方函数 sqrt，所以在程序前面要包含 math.h 头文件。

程序如下：

```
/* 程序 3-11,源程序文件 pro3-11.c */
#include <math.h>
#include <stdio.h>
main()
{
        float a,b,c,s,area;
        scanf("%f,%f,%f",&a,&b,&c);
        s = 1.0/2 * (a + b + c);
        area = sqrt(s * (s - a) * (s - b) * (s - c));
        printf("a = %5.2f,b = %5.2f,c = %5.2f,area = %5.2f\n",a,b,c,area);
}
```

运行时输入：3,4,5 后，程序的运行结果如图 3-13 所示。