第 3 章 顺序结构程序设计

CHAPTER 3

程序就是为了解决某个问题所使用的语句的有序集合，这些语句是有执行顺序的。如同写文章时要考虑文章的结构、建房子时要考虑房子的结构一样，在进行 C 语言的程序设计时，不只要描述数据，还要设计操作流程，也就是程序的结构。C 语言是一种结构化的程序设计语言，它包含 3 种基本控制结构：顺序结构、选择结构和循环结构，因此要想写出好的结构化程序，必须熟练掌握程序的 3 种控制结构。

顺序结构是程序设计的最基本结构，结构中的语句执行是按照语句书写顺序进行的，且每条语句都将被执行，其他两种结构可以作为顺序结构的一部分，也可以包含顺序结构。本章主要介绍顺序结构的程序设计。

本章学习重点：

(1) C 语言的基本语句；

(2) C 语言中数据的输入与输出；

(3) 顺序结构程序设计。

本章学习目标：

(1) 掌握 C 语言中基本的语句与用法；

(2) 熟练掌握 C 语言的输入输出函数的使用方法；

(3) 掌握顺序结构程序设计的方法。

3.1 C 语言语句概述

C 语言程序是由函数组成的，函数功能是由执行语句来实现的。C 语言中的执行语句大致可分为 5 类，分别是表达式语句、函数调用语句、空语句、复合语句和流程控制语句。

3.1.1 表达式语句

表达式是由运算符连接操作数所组成的式子。表达式语句就是由表达式加上 C 语言语句的结束标志分号“;”组成，其一般形式为

```
表达式;
```

表达式语句的功能就是计算表达式的值。例如，

```
i++;                          /* 自增表达式语句 */
```

```
x++,y--,x+y;                    /* 逗号表达式语句 */
```

在表达式语句中,应用最广泛的是赋值表达式语句,即由赋值表达式后跟一个分号组成。程序中绝大多数对数据的处理都是用赋值表达式语句来实现的。例如,

```
c=a+b;                          /* 将变量 a 和 b 做加法运算 */
sum=sum+i;                      /* 迭代求和 */
s=b*b-4*a*c;                    /* 一元二次方程中求判别式的值 */
r=m%n;                          /* 求 m,n 的余数 */
```

3.1.2 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(实际参数列表);
```

函数调用语句的作用就是将实际参数的值传给形式参数之后,转去执行被调用函数的语句,完成特定的功能。例如,

```
printf("%d",a);                 /* 输出函数调用语句,输出变量 a 的值 */
c=max(a,b);                     /* 调用自定义函数 max,并将函数的返回值赋给变量 c */
```

C 语言从使用的角度分,有标准函数和用户自定义函数。

1. 标准函数(又称库函数)

C 语言有着丰富的标准函数库,可提供各类标准函数供用户调用。标准函数就是预先设计好的完成某个功能的语句序列,将之放在函数库中,用户只要直接调用即可,不需要另外编写代码。例如,调用标准库函数进行输入输出操作、求数学函数值、对字符串的处理等,如 printf()、scanf()、sqrt()、fabs()(求绝对值函数)、strlen()(求字符串的长度)等都是常用的标准库函数。调用标准库函数时,一定要在程序中包含相应的头文件,例如,要调用标准的输入输出函数,就要用如下语句,将头文件 stdio.h 包含进来:

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

或

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

注意:调用库函数时要查看函数库,了解函数功能和定义,按照规范调用。附录 E 详细列出了 C 语言中使用的标准库函数。

2. 用户自定义函数

用户可以把完成某种功能的语句序列包装成一个函数,即自定义函数。有关自定义函数的定义、调用等相关知识将在第 6 章做详细介绍。

3.1.3 空语句

空语句就是只有分号的语句,其一般形式为

```
;
```

空语句在语法上占有一个简单语句的位置，什么操作也不执行，在程序中空语句可用来作空循环体，条件分支空语句等。例如，

```
while (i++< 5);                /* 连续对 i 进行增 1,直到 i 等于 5 停止 */
```

3.1.4　复合语句

将多条语句用花括号“{ }”括起来，组成一个整体的语句组称为复合语句，在语法上相当于一条语句。例如，

```
{
    t = x;
    x = y;
    y = t;
}
```

注意：

① 复合语句内的各条语句都必须以分号“;”结尾。

② 在复合语句内部定义的变量是局部变量，只在复合语句中有效。

③ 在结束括号“}”外不需要再加分号。

【例 3-1】 复合语句示例。

程序如下：

```
#include < stdio.h >
int main()
{
    int a = 3;
    {
        int a = 4;
        printf("a = %d\n",a);
    }
   printf("a = %d\n",a);
}
```

运行结果为

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

在上面的程序中，主函数语句只有一条复合语句，这条复合语句内部又嵌套了一个复合语句，各复合语句内部定义的变量只在该复合语句内有效，所以有了以上的输出结果。

复合语句常用于流程控制语句中执行多条语句。

3.1.5　流程控制语句

在选择结构、循环结构等控制结构中用于控制程序执行流程的语句、函数调用时的返回语句等都是流程控制语句，例如，

```
while (表达式)
{
    …
}
```

```
return 表达式;
```

这些控制语句将在后续相应章节做详细介绍。

3.2 数据的输入与输出

程序的主要功能就是对数据的处理,其整个流程主要包括数据的输入、数据的处理和数据的输出。数据从计算机外部输入设备读入计算机内存为数据输入;相反,数据从计算机内存取出或写出到外部输出设备为数据输出。C 语言本身不提供输入输出语句,输入输出的功能是由标准输入输出库函数提供的。在使用 C 语言库函数时,要用预处理命令将相对应的头文件包含进来。

C 语言有丰富的输入输出库函数,如用于键盘输入和显示器输出的输入输出库函数、磁盘文件读写的输入输出库函数、硬件端口操作的输入输出库函数等,本节主要介绍用于标准输入设备(键盘)进行输入和标准输出设备(显示器)进行输出的库函数,其对应的头文件为 stdio.h。

3.2.1 字符数据的输入输出

1. 字符输出函数 putchar()

如要从计算机向显示器输出一个字符,可以调用系统函数库中的字符输出函数 putchar()。其函数原型为

```
int putchar(int c);
```

函数功能:向标准输出设备(一般为显示器)输出一个字符,并返回输出字符的 ASCII 码值;若出错,返回 EOF(-1)。

说明:该函数带有一个参数 c,可以是字符常量、字符型变量或整型变量。

例如,

```
putchar(c);                 /* 输出字符变量 c 对应的字符 */
putchar('a');               /* 输出小写字母'a' */
putchar('\101');            /* 输出大写字母'A','\101'是转义字符 */
putchar(65);                /* 整型数据和字符数据是相通的,输出对应字符大写字母'A' */
```

对控制字符则执行控制功能,不在屏幕上显示。例如,

```
putchar('\n');              /* 换行,使输出的当前位置定位到下一行开头 */
```

2. 字符输入函数 getchar()

如要向计算机输入一个字符,可以调用系统函数库中的字符输入函数 getchar(),每调用一次就从标准输入设备上取一个字符。输入的字符可以赋给一个字符变量或者一个整型变量。其函数原型为

```
int getchar(void);
```

函数功能:从标准输入设备(一般为键盘)读入一个字符,并返回读取字符的 ASCII 码值;若出错,返回 EOF(-1)。

说明:

(1) 以回车符为输入结束。当输入多个字符时,返回第一个字符的值,输入字符回显在

屏幕上。

(2) 函数值可以赋给一个字符变量,也可以赋给一个整型变量。例如,

```
int a;
char ch;
a = getchar();
ch = getchar();
```

【例 3-2】 用 getchar()输入大写字母并显示其小写字母。

程序如下:

```
#include <stdio.h>
int main()
{
    char ch;
    ch = getchar();
    putchar(ch + 32);
    return 0;
}
```

运行结果为

```
A↙
a
```

也可以不定义变量,把程序改写成:

```
#include <stdio.h>
int main()
{
    putchar(getchar() + 32);   /* 将接收到的字符输出 */
    return 0;
}
```

其中,"putchar(getchar());"是函数嵌套调用的紧凑形式。

注意:

① 执行 getchar()函数时,输入字符后,只有按下了回车键,程序才把字符真正输入计算机中。

② getchar()函数只能接收单个字符,输入数字也按字符处理。

③ getchar()函数还可以接收在屏幕上无法显示的字符,如控制字符。

【例 3-3】 用 getchar()输入单词 OK 并显示。

```
#include <stdio.h>
int main()
{
    char a,b,c;
    a = getchar();
    b = getchar();
    c = getchar();
    putchar(a);
    putchar(b);
    putchar(c);
    return 0;
}
```

运行程序时,共输入为

```
O↙
K↙
```

则输出结果为

```
O

K
```

出现这样的结果,是因为 getchar()可以接收控制字符。第 1 行输入的不是一个字符'O',而是两个字符:'O'和换行符,其中'O'被赋给了变量 a,换行符赋给了变量 b。第 2 行输入的两个字符:'K'和换行符,其中字符'K'被赋给了变量 c,换行符没有赋给任何变量。

3.2.2 格式输出函数 printf()

1. printf()函数调用的一般形式

printf()函数称为格式输出函数,其关键字最末一个字母 f 即为"格式"(format)之意。它使用的一般形式为

```
printf("格式控制字符串",输出项列表);
```

函数功能:按规定格式向输出设备输出若干任意类型的数据,并返回实际输出的字符数;若出错,则返回负数。

其中,"输出项列表"列出的需要输出的表达式可以是 0 个或多个。每个输出项之间用逗号分隔。输出的数据可以是整数、实数、字符和字符串、变量、表达式和函数等。"格式控制字符串"是用双引号括起的字符串,用于指定输出数据的类型、格式、个数。

注意:putchar()只能输出一个字符,而 printf()函数可以输出多个数据,且为任意类型。

2. 格式控制

格式控制由格式控制字符串实现。格式控制字符串必须用英文的双引号括起来,它包括 3 种信息:

- 普通字符。需要在输出时原样输出,在显示中起提示作用。
- 转义字符。实现对应的操作,如'\n',实现换行操作。
- 格式说明。

格式控制字符串的一般形式为

```
%[修饰符]格式控制符
```

其中,"%"作为格式说明的引导符,放在开头位置,不可缺少;方括号[]表示该项为可选项。格式控制符表示输出数据的类型,用来进行格式转换,如表 3-1 所示。

表 3-1 printf()函数中的格式控制符

格式控制符	说　明	举　例	运行结果
d,i	以带符号的十进制形式输出整数(正数不输出正号)	int a = 567; printf("%d",a);	567

续表

格式控制符	说　明	举　例	运行结果
O	以八进制无符号形式输出整数(不输出前导符数字 0)	int a = 65; printf("%o",a);	101
x,X	以十六进制无符号形式输出整数(不输出前导符 0X)	int a = 255; printf("%x",a);	ff
u	以十进制无符号形式输出整数	int a = -1; printf("%u",a);	4294967295
c	输出一个字符	char a = 65; printf("%c",a);	A
s	输出字符串	printf("%s","ABC");	ABC
f	以小数形式输出单、双精度实数	float a = 567.789; printf("%f",a);	567.789000
e,E	以指数形式输出单、双精度实数	float a = 567.789; printf("%e",a);	5.677890e+002
g,G	由给定的值和精度,自动选用 e 和 f 中较短的一种	float a = 567.789; printf("%g",a);	567.789
%	输出百分号本身	printf("%%");	%

说明:

(1) 格式字符与输出项个数应相同,按先后顺序一一对应。

(2) 输出转换:如格式字符与输出项类型不一致,则自动按指定格式输出。

(3) 输出实数(单、双精度)时,系统将实数中的整数部分全部输出,小数部分默认输出 6 位。需要注意的是,单精度实数的有效位数一般为 7 或 8 位,双精度实数的有效位数一般为 16 或 17 位,而用 f 格式输出时,整数部分加小数部分的长度可能超过单精度实数本身的有效位,因此在输出的数字中并非全部数字都是有效数字。

(4) 用指数格式 e 输出时,标准输出宽度为 13 位,其中尾数的整数部分为非零数字占 1 位、小数点占 1 位、小数占 6 位、e 占 1 位、指数正负号占 1 位、指数占 3 位。

使用 printf()函数进行格式化输出时,还可以使用修饰符,指定宽度、精度、对齐方式等。修饰符的格式如表 3-2 所示。

表 3-2　printf()函数中的修饰符

修　饰　符	含　义
m	输出数据域宽,如果数据长度<m,则左端补空格;否则按实际输出
n	对实数,指定小数点后位数(对 n+1 位进行四舍五入)
	对字符串,指定实际输出位数
-	输出数据在域内左对齐(默认为右对齐)
+	指定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置自动填 0
#	在八进制或十六进制数前显示前导 0 或 0x
l	用于 d,o,x,u 前,指定输出精度为 long 型
	用于 e,f,g 前,指定输出精度为 double 型

【例 3-4】 多种修饰符、格式字符组合示例之一。

程序如下:

```
#include <stdio.h>
int main()
{
    int a = 1234;
    float f = 1.2345;
    char ch = 'a';
    static char str[] = "Hello,world! ";
    printf("a = %8d,a = %2d\n",a,a);
    printf("%f,%8f,%8.1f,%.2f,%.2e\n",f,f,f,f,f);
    printf("%3c\n",ch);
    printf("%s\n%15s\n%10.5s\n%2.5s\n%.3s\n", str, str, str, str, str);
    return 0;
}
```

运行结果为

```
a =   1234,a = 1234
1.234500,1.234500,     1.2,1.23,1.23e+000
  a
Hello,world!
   Hello,world!
     Hello
Hello
Hel
```

分析：程序中的“a=”是格式字符串中的普通字符，在其出现的位置上按原样输出。当指定输出宽度小于实际宽度时，按实际输出，所以第一个输出语句中的指定宽度为%2d，但实际输出为“1234”；第四个输出语句中指定字符串宽度为2.5，但还是按5位输出。第二个输出语句中指定格式为%8.1f，表示输出实数共占8个字符位置，其中1位小数(四舍五入)，不够的左端补空格。

【例3-5】 多种修饰符、格式字符组合示例之二。

程序如下：

```
#include <stdio.h>
int main()
{
    int a = 11,b = 22;
    float f = 1.2345;
    short m = -1;
    int n = 2234567890;
    printf("a = %5d, b = %5d\n",a,b);
    printf("a = %-5d, b = %-5d\n",a,b);
    printf("f = %+10.2f,f = %-10.1f,f = %10.3f\n",f,f,f);
    printf("m: %d, %o, %x, %u\n",m,m,m,m);
    printf("n = %ld\n",n);
    return 0;
}
```

运行结果为

```
a =   11, b =   22
a = 11   , b = 22
f =      +1.23,f = 1.2       ,f =     1.235
m: -1, 37777777777, ffffffff, 4294967295
n = -2060399406
```

观察【例 3-5】程序运行结果，分析原因。

注意：如果格式控制符少于输出项，则多余的输出项不输出。例如，

```
#include<stdio.h>
int main()
{
    int a,b,c,d;
    a=b=c=d=1;
    printf("%d %d %d\n",a,b,c,d);
    return 0;
}
```

运行结果为

```
1 1 1
```

如果格式控制符多于输出项，则最后输出随机数。例如，

```
int a,b,c,d;
a=b=c=1;
printf("%d %d %d %d\n",a,b,c);
```

运行结果为

```
1 1 1  4198896
```

3.2.3　格式输入函数 scanf()

1. scanf()函数调用的一般形式

scanf()函数称为格式输入函数，它使用的一般形式为

```
scanf ("格式控制字符串",地址列表);
```

函数功能：按规定格式从键盘输入若干任何类型的数据给地址列表中变量所指的单元，返回读入并赋给变量的数据个数；若遇文件结束则返回 EOF，出错返回 0。

说明：

(1) getchar()函数只能读入一个字符，而 scanf()函数可以一次输入多个数据，且为任意类型。

(2) 地址列表是由若干个地址组成的，列表可以是变量的地址、字符串的首地址、指针变量等，各地址之间以逗号“,”分隔。

例如，

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

“%c”是格式控制字符串，控制输入一个字符。& 是取地址运算符，&ch 表示取变量 ch 的地址。变量的值和变量的地址是两个不同的概念。变量的地址是 C 编译系统给变量分配的，有关变量地址的知识将在第 8 章指针中做详细介绍。

(3) 地址列表中各变量类型、变量地址的个数和顺序必须和格式控制字符串的参数一致。

例如，a、b 是 int 型，x 是 float 型，调用时可写成：

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

下列 scanf()函数的调用是错误的：

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

2. 格式控制

格式控制由格式控制字符串实现。格式控制字符串必须用英文的双引号括起来，它包括两种信息。

- 普通字符，即非格式控制字符。与 printf()函数不同，在提示输入时不显示，而在输入数据时，在对应位置需要输入与这些字符相同的字符，即原样输入。
- 格式字符。格式控制字符串的一般形式为

```
%[修饰符]格式控制符
```

其中，以“%”开头，方括号[]表示该项为可选项。

格式控制符和 printf()中的相同，有 d、i、o、x、u、c、s、f、e。例如，

```
scanf("%d",&a);              //输入 11 后,a=11;
scanf("%x",&a);              //输入 11 后,a=17,此时的 11 是被当作十六进制的 11 接收的,
                             //所以得到的值转换成十进制是 17
```

scanf()也可以使用修饰符，如表 3-3 所示。

表 3-3 scanf()函数中的修饰符

修 饰 符	含 义
h	用于 d,o,x 前,指定输入为 short 型整数
l	用于 d,o,x 前,指定输入为 long 型整数
	用于 e,f 前,指定输入为 double 型实数
m	指定输入数据宽度,如遇空格或不可转换字符则结束
*	抑制符,指定输入项读入后不赋给变量

说明：

输入指定的分隔符：

- 一般以空格、Tab 键或回车键作为分隔符；
- 用其他字符做分隔符，分隔格式串中两个格式符。

例如，

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

输入 3 个整型十进制数，以空白符(空格、Tab 键或回车键)分隔。

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

输入 2 个数，以“,”分隔。

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

输入的形式是：a=32，b=28(普通字符要照原样输入)。

(2) 输入数据时,遇以下情况认为该输入结束:

- 遇空格、Tab 键或回车键。
- 满宽度结束(如%3d,满 3 位数字)。
- 遇非法输入(与对应输出项的类型不符)。

例如,

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

若输入 1234a123p.26↙,则结果为"a=1234,b='a',c=123"。

(3) 用"%c"格式符时,空格和转义字符作为有效字符输入,输入的字符型数据不必分隔。例如,

```
scanf("%c%c%c", &ch1, &ch2, &ch3);
```

如果输入:abc↙,则结果为

```
ch1 = 'a',ch2 = 'b',ch3 = 'c';
```

如果输入:a␣b␣c↙,则结果为

```
ch1 = 'a',ch2 = '␣',ch3 = 'b'
```

特别要注意,当数值型数据与 char 型数据混合输入时,例如,

```
scanf("%d%d",&m,&n);
    scanf("%c",&ch);
```

错误输入:

```
32␣28↙
a↙
```

ch 得到的值为'↙','↙'被当作输入数据赋给了 ch。

正确输入应为:

```
32␣28a↙
```

(4) double 型数据输入时,必须用%lf 或%le 格式。

(5) 格式控制符中有普通字符时,必须照原样输入。

(6) 实型数输入时域宽不能用 m.n 形式的附加说明,即输入数据不能规定精度,例如,scanf("%7.2f",&a)是不合法的。

(7) 为了减少不必要的输入量,除了逗号、分号、空格符等简单字符以外,格式控制中尽量不要出现普通字符,也不要使用'\n'、'\t'等转义字符。

注意:"*"抑制符表示数据输入项要按指定格式进行转换,但不保存到变量中,即在地址列表中没有对应的地址项,一般用来吸收字符。

【例 3-6】 抑制符示例。

程序如下:

```
#include<stdio.h>
int main()
{
    int a, b;
    scanf("%2d%*3d%2d",&a,&b);
```

```
    printf("a = %d,b = %d\n",a,b);
    return 0;
}
```

运行结果为

```
1234567↙
a = 12,b = 67
```

分析：%*3抑制了中间3个数345，没有对应的输出项，故12传给了a，67传给了b。

3.3 顺序结构程序设计举例

前面已经介绍了一些基本语句，而程序是由语句来实现的，现在着手设计简单的顺序结构程序。顺序结构程序一般包括以下几部分。

(1) 编译预处理命令。

在程序的编写过程中，若要使用标准函数(库函数)，应该使用编译预处理命令，将相应的头文件包含进来，如include<stdio.h>、include<math.h>等。

(2) 函数。

函数的基本组成包括变量定义、提供原始数据、数据处理、输出结果。

相应地，在函数体中包含着顺序执行的各部分语句，主要有以下几部分：

① 变量类型的说明部分；

② 提供数据部分(可以是变量初始化、赋值语句或输入函数调用语句)；

③ 数据处理部分；

④ 输出部分。

【例3-7】 输入两个数a、b，实现两个数的交换。

分析：在两个变量交换过程中，不能直接赋值，如“a＝b；b＝a；”会把变量原有的值覆盖，应借助一个中间变量c，实现两个数的交换。

程序如下：

```
#include<stdio.h>
int main()
{
    int a,b,c;
    printf("请输入a,b两个数:\n");
    scanf("%d %d",&a,&b);
    c = a;a = b;b = c;
    printf("a = %d b = %d\n",a,b);
    return 0;
}
```

运行结果为

```
请输入a,b两个数:
12 34↙
a = 34 b = 12
```

【例 3-8】 输入 3 个小写字母，输出其 ASCII 码和对应的大写字母。

分析：大小写字母之间的 ASCII 码相差 32，即大写字母的 ASCII 码＝小写字母的 ASCII 码－32。用 printf()函数输出，可以控制输出格式，%d 为 ASCII 码，%c 为字母本身。

程序如下：

```
#include <stdio.h>
int main()
{
    char a,b,c,a1,b1,c1;
    printf("input three lowercase letters :");
    scanf("%c,%c,%c",&a,&b,&c);
    a1 = a - 32;b1 = b - 32;c1 = c - 32;
    printf("%d,%d,%d\n%c,%c,%c",a,b,c,a1,b1,c1);
    return 0;
}
```

运行结果为

```
input three lowercase letters :a,b,c↙
97,98,99
A,B,C
```

【例 3-9】 设计程序，使得用户可以以任意字符（回车、空格、制表符、逗号、其他）作为分隔符进行数据的输入。

分析：可以使用抑制符抑制两字符间的任意字符。

程序如下：

```
#include <stdio.h>
int main()
{
    int a,b;
    scanf("%d%*c%d",&a,&b);
    printf("a = %d,b = %d\n",a,b);
    return 0;
}
```

运行结果为

```
18&19↙
a = 18,b = 19
```

小结

本章主要介绍了 5 种基本语句、数据的输入与输出以及顺序结构程序的设计。顺序结构是程序中最简单的，也是最基本的结构，即语句按其书写顺序先后逐条执行。本章重点讲解了格式化输出函数 printf()和格式化输入函数 scanf()的功能及使用方法。

学习本章后，应掌握以下知识：

(1) C 语言的 5 种基本语句——表达式语句、函数调用语句、空语句、复合语句和流程控制语句；

(2) 输入函数 getchar()和输出函数 putchar()的使用;

(3) 利用输出函数 printf()和输入函数 scanf()实现数据的格式化输入输出,在使用时格式控制字符和数据列表的个数、类型要严格一致;

(4) 顺序结构程序的设计。

本章常见错误分析

常见错误实例	常见错误解析
#include<stdio.h>;	预处理命令后加分号。#开头的是预处理命令,不是语句,不能用分号结尾
print("Hello! "); PRINTF("hello! ");	将 printf()函数名写错,C 语言中区分大小写,函数名错误不能识别函数
printf(Hello!); scanf(%d,&x);	未给函数 printf()和 scanf()中的格式控制字符串加上双引号
scanf("%d",x);	未给 scanf()函数中的变量加取地址运算符 &
printf("x=%d");	用 printf()函数输出一个变量,未写输出变量

int x,y;
scanf("%d",&x,&y);
printf("%d,%f",x,y);
	在调用 scanf()函数和 printf()函数时,格式控制字符串与表达式变量类型和个数不一致,应改为:
	int x,y;
	scanf("%d%d",&x,&y);
	printf("%d,%d",x,y);

scanf("%7.2f",&f);	调用 scanf()函数输入浮点数时规定了精度。应改为: scanf("%7f",&f);
scanf("%d",&x); printf("x=%d");	程序中使用了中文的逗号、分号、单引号或双引号
scanf("%d\n",&x);	scanf()函数格式控制字符串中加了转义字符,应改为: scanf("%d",&a);
scanf("%d, "&x);	将分隔格式控制字符串和表达式间的逗号写到了格式控制字符串内
int x,y; scanf("%d%d",&(x+y));	对算术表达式取地址。表达式没有地址,不能用来取地址
printf("He can say "Hello! "");	输出单引号、双引号、反斜杠字符时,未在字符前用反斜杠构成转义字符。应改为: printf("He can say \"Hello!\ "");

习题 3

1. 基础篇

(1) 任何复杂的程序都可以由(　　)、(　　)和(　　)这 3 种基本结构组成。

(2) 用花括号组合在一起的多条语句称为(　　)。

(3) C 语言本身不提供输入输出语句,其输入输出操作是由(　　)来实现的。

(4) C 语言用(　　)函数能够实现精确的输出格式。

(5) 格式字符中,除了(　　)以外,其他均为小写字母。

(6) 格式字符(　　)表示显示一个 double 类型的数据值。

(7) 要输出长整型的数值,需用格式字符(　　)。

(8) (　　)标志使数据输出在域宽内左对齐。

(9) 执行语句"printf("%d,%d",a,b,c,d);"后,将在屏幕上输出(　　)个整数。

(10) 已有定义"int a; float b,x; char c1,c2;",为使 a=1,b=2.5,x=55.3,c1='Y',c2='y',正确的 scanf()函数调用语句是(　　)。

2. 进阶篇

(1) printf()中用到格式符%5s,其中数字 5 表示输出的字符串占用 5 列。如果字符串长度大于 5,则输出方式为(　　)。

A. 从左起输出该字符串,右补空格　　B. 按原字符长从左向右全部输出

C. 右对齐输出该字符串,左补空格　　D. 输出错误信息

(2) 若 w、x、y、z 均为 int 型变量,为了使以下语句的输出为 1234+123+12+1,正确的输入形式应为(　　)。

```
scanf("%4d+%3d+%2d+%1d",&x,&y,&z,&w);
printf("%4d+%3d+%2d+%1d\n",x,y,z,w);
```

A. 1234123121↙　　B. 1234+1234+1234+1234↙

C. 4↙　　D. 1234+123+12+1↙

(3) 已有如下定义和输入语句,若要求 a1、a2、c1、c2 的值分别为 10、20、A 和 B,当第一列开始输入数据时,正确数据输入方式是(　　)。("□"表示一个空格)

```
int a1,a2;
char c1,c2;
scanf("%d%c%d%c",&a1,&c1,&a2,&c2);
```

A. 10A20B↙　　B. 10□A20B↙　　C. 10A 20B↙　　D. 10 20↙ AB↙

(4) 有以下程序:

```
int main()
{
    int a;
    char c=10;
    float f=100.0;
    double x;
    a=f/=c*=(x=6.5);
    printf("%d□%d□%3.1f□%3.1f\n",a,c,f,x);
    return 0;
}
```

程序运行后的输出结果是(　　)。

A. 1□65□16.5　　B. 1□65□1.5□6.5

C. 1□65□1.0□6.5　　D. 2□65

(5) 执行以下程序时输入:123□456□789↙,输出结果是(　　)。

```
int main()
{
    char s;
```

```
    int c,i;
    scanf("%c",&c); scanf("%d",&i); scanf("%c",&s);
    printf("%c,%d,%c\n",c,i,s);
    return 0;
}
```

A. 123,456,789　　B. 1,456,789　　C. 1,23,456,789　　D. 1,23,

(6) 执行以下程序的输出结果是(　　)。

```
int main()
{
    char a;
    a='H'-'A'+'0';
    printf("%c,%d\n",a,a);
    return 0;
}
```

A. H,97　　B. 7,55　　C. 7,7　　D. 55,7

3. 提高篇

(1) 下面程序在运行时输入 90，结果为 x=0.500000；运行时输入 180,结果为 x=0.000000。程序中有 4 处错误,请找出错误并改正。

```
#include<math.h>
#define PI 3.1415926
int main()
{
    long d;
    double x;
    scanf("%d",d);
    x=1.0/2*sin(d*PI/180.0);
    printf("x=%f\n",x);
    return 0;
}
```

(2) 下面程序的功能是：输入一个华氏温度,如输入 98.6,则输出相应的摄氏温度为 37.0。程序中有 3 处错误,请找出错误并改正。

```
#include<stdio.h>
void main()
{
    double F,c;
    scanf("%f",&F);
    c=5/9(F-32);
    printf("F=%2.2lf\nc=%2.2lf\n",F,c);
 }
```

(3) 编写程序,从键盘输入一个 3 位数正整数,将它反序输出。例如,输入“123”,输出应为“321”。

(4) 编写程序,从键盘输入一个字符,请按顺序分别输出它的前驱字符、字符本身及后继字符的符号以及 ASCII 码值。

(5) 编写程序,从键盘输入两个整数,分别赋给变量 a 和 b,要求在不借助于其他变量的条件下,将变量 a 和 b 的值实现交换。