第3章 顺序结构

【学习要点】

- 字符的输入与输出。
- 数据的格式化输出。
- 数据的格式化输入。
- 顺序结构程序设计。

3.1 字符的输入与输出

在C语言中,字符有**字符常量**和**字符型变量**两种表示方法。字符常量是用**单引号**引起来的一个字符。例如,'A'是一个字符常量,而语句"char A;"中的A是一个字符型变量,可以用来存储任意一个字符。'9'是一个字符常量,而9则是一个整型常量。

C语言的标准库函数中,有专门用于字符的输入函数getchar()和输出函数putchar()。

3.1.1 字符常量的输出


字符常量的输出课堂练习

字符常量的值是固定不变的,其值不能修改,可用putchar()函数将一个字符常量输出到屏幕上。字符常量的输出主要有以下两种方法。

1. 单引号表示的字符常量的输出

putchar()的参数可以是**字符常量**,**必须加单引号**,例如:

```
putchar('a');          //输出字符 a
putchar('9');          //输出字符 9
putchar('\\');         //输出字符\
putchar('\101');       //输出字符 A。'A'的 ASCII 码为 65,其八进制表示为 101
putchar('\x42');       //输出字符 B。'B'的 ASCII 码为 66,其十六进制表示为 0x42
putchar('\n');         //输出换行符
```

2. 整型表达式表示的字符常量的输出

putchar()的参数也可以直接是**字符常量的ASCII码**(不能加单引号),或者**是计算结**

果为整数的表达式。例如：

```
putchar(97);          //输出字符 a,97 是'a'的 ASCII 码
putchar(57);          /*输出字符 9,57 是'9'的 ASCII 码。若写成 putchar(9);错误
                        但输出的是 ASCII 码为 9 的字符,它是一个不可见字符*/
putchar('a'-32);      /*输出字符 A, 'a'的 ASCII 码为 97,减 32 得到 65,
                        即'A'的 ASCII 码*/
```

字符型变量的输入/输出课堂练习

3.1.2 字符型变量的输入/输出

1. 字符型变量的输入

若要输入字符型变量的值，可用 getchar()函数实现。当程序调用 getchar()函数时，该函数会从输入缓冲区(也称为标准输入流)中读取字符，每调用一次就读取一个字符，即 getchar()函数一次读取一个字符，函数调用的返回值是其读取到的字符的 ASCII 码。如果用户一次输入了多个字符，则未被读取的字符会保留在输入缓冲区中，后续调用 getchar()函数时将被读取。

2. 字符型变量的输出

与字符常量相同，字符型变量的输出仍然可以使用 putchar()函数。例如：

```
char ch='A';          //定义字符型变量 ch 并进行初始化
putchar(ch);          //输出大写字母 A。变量名不能加单引号
putchar(ch+32+1);     /*输出小写字母 b。计算表达式 ch+32+1 时,会进行自动类型转换,
                        用 ch 中所存字符 A 的 ASCII 码 65 参与运算,即 65+32+1=98,
                        98 为小写字母 b 的 ASCII 码 */
```

【例 3.1】 从键盘输入由 3 个小写字母构成的人名，将其第一个字母转换成大写后输出完整的人名。

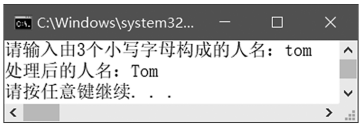
【问题分析】

大小写字母的 ASCII 码相差 32，即 $C_{大}=C_{小}-32$。英文名的 3 个英文字母可以用 3 个字符型变量来接收，然后将第一个变量转换成大写字母，再输出 3 个变量的值即可。

【编程实现】

```
1   #include <stdio.h>
2   int main()
3   {
4       char ch1,ch2,ch3;
5       printf("请输入由 3 个小写字母构成的人名: ");
6       ch1=getchar();          //接收第一个英文字母
7       ch2=getchar();          //接收第二个英文字母
8       ch3=getchar();          //接收第三个英文字母
9       ch1=ch1-32;             //将第一个小写字母转换为大写字母
10      printf("处理后的人名: ");
11      putchar(ch1);           //输出第一个英文字母
12      putchar(ch2);           //输出第二个英文字母
13      putchar(ch3);           //输出第三个英文字母
14      putchar('\n');          //输出一个换行符
15      return 0;
16  }
```

【运行结果】



【关键知识点】

(1) 代码第 6～8 行调用 getchar()函数分别接收 3 个英文字母，将接收字母的 ASCII 码分别赋值给对应的字符型变量 ch1、ch2、ch3。

(2) 调用 getchar()或 putchar()函数一次只能输入或输出一个字符，若要处理多个字符，必须多次调用函数才能实现。

【延展学习】

(1) getchar()函数能否接收不可显示的字符？

(2) 若每次输入的人名长度不相等，该如何处理？详见 7.4 节。


数据的格式化输出课堂练习

3.2 数据的格式化输出

在 C 语言的标准库函数中，专门用于格式化输出的函数为 printf()，在头文件 stdio.h 中声明，主要功能是向标准输出设备(一般指显示器)按规定格式输出信息。

1. printf()函数的一般格式

printf()函数的调用格式为：

```
printf(格式控制字符串,输出数据项参数表);
```

(1) **格式控制字符串**是双引号引起来的字符串，包含两部分：**普通字符**和**格式控制符**。在输出时，普通字符将按原样输出，格式控制符并不直接输出，而是用于控制 printf()函数中参数的转换和输出。每个格式控制符都由一个**百分号(%)**开始，以转换说明字符结束，说明输出数据项的类型、宽度、精度等输出格式，如表 3.1 所示。

表 3.1 printf()函数的格式控制符

格式控制符	数 据 类 型	描　　述
%d	int	输出有符号十进制整数，正数的符号省略
%u	unsigned	输出无符号十进制整数
%o	unsigned	输出无符号八进制整数(没有前导 0)
%x	unsigned	输出无符号十六进制整数(没有前导 0x)，十六进制的数码 abcdef 以小写形式输出
%X	unsigned	输出无符号十六进制整数(没有前导 0X)，十六进制的数码 ABCDEF 以大写形式输出
%f	float 或 double	输出十进制表示的浮点数，默认输出 6 位小数

续表

格式控制符	数 据 类 型	描　　述
%lf	double	输出十进制表示的浮点数，默认输出 6 位小数(lf 从 C99 开始加入 C 语言标准)
%e	float 或 double	输出科学记数法表示的浮点数，默认输出 6 位小数，e 以小写形式输出，如 3.500000e−3
%E	float 或 double	输出科学记数法表示的浮点数，默认输出 6 位小数，E 以大写形式输出，如 3.500000E−3
%g	float 或 double	自动选取%f 或%e 格式输出宽度较小的一种使用，且不输出无意义的 0
%c	char	输出一个字符
%s		输出一个字符串
%%		输出一个百分号%

(2) **输出数据项**可以是常量、变量或表达式，多个输出数据项之间用逗号隔开，每个数据项与格式控制字符串中的格式控制符一一对应。

例如，若有

```
int age=18;
char sex='M';
float score=98.5;
```

则相应的输出语句及其输出结果如图 3.1 所示，图中标明了格式控制说明符与输出参数表之间的一一对应关系，例如，第一个输出数据项"张三"，是字符串，因此双引号中的第一个格式控制符应为%s；第二个输出数据项 age+1 的计算结果为 int 型，因此双引号中的第二个格式控制符应为%d；以此类推，sex 对应格式控制符为%c，score 对应格式控制符为%f。

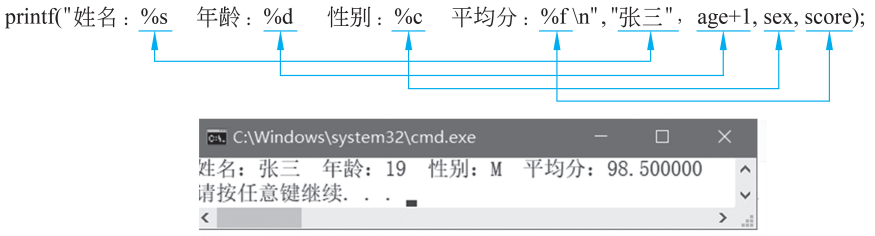


图 3.1　基本数据格式输出

【例 3.2】 从键盘输入一个 B～Z 的大写英文字母，输出该英文字母，及其前驱字母、后继字母的基本信息(包括字母及其 ASCII 码)。

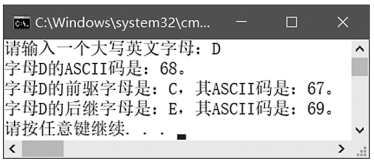
【问题分析】

从 ASCII 码表可以看出，大写英文字母的 ASCII 码是连续的，即'A'的 ASCII 码是 65，'B'的 ASCII 码是 66，以此类推。故除'A'和'Z'之外，所有字母的前驱字母的 ASCII 码就是该字母的 ASCII 码减 1，后继字母的 ASCII 码就是该字母的 ASCII 码加 1。

【编程实现】

```
#include<stdio.h>
int main()
{
    char ch1,ch2,ch3;
    printf("请输入一个大写英文字母: ");
    ch1=getchar();              //接收一个英文字母
    ch2=ch1-1;                  //得到 ch1 的前驱字母
    ch3=ch1+1;                  //得到 ch1 的后继字母
    printf("字母%c 的 ASCII 码是: %d。\n",ch1,ch1);
    printf("字母%c 的前驱字母是: %c,其 ASCII 码是: %d。\n",ch1,ch2,ch2);
    printf("字母%c 的后继字母是: %c,其 ASCII 码是: %d。\n",ch1,ch3,ch3);
    return 0;
}
```

【运行结果】



【关键知识点】

(1) 语句 printf("%c",ch)和语句 putchar(ch)等价,都是输出一个字符。

(2) 语句 printf("\n")和语句 putchar('\n')等价,都是在屏幕上输出一个换行符。请注意两条语句中分别用的是双引号、单引号。

(3) **字符型变量**,可以用格式控制符**%c** 进行输出,输出的是**字符**;也可以用格式控制符**%d** 进行输出,输出的是该字符的 **ASCII 码**。

2. printf()函数的格式修饰符

在 printf()函数的格式说明中,在%和格式符中间插入如表 3.2 所示的**格式修饰符**,可以对输出格式进行控制,如设置数据的域宽(输出宽度)、输出精度、左右对齐方式等。

表 3.2 printf()函数的格式修饰符

格式修饰符	用 法
l	修饰格式符 d、o、x、u 时,表示输出数据类型为 long
L	修饰格式符 f、e、g 时,表示输出数据类型为 long double
h	修饰格式符 d、o、x 时,表示输出数据类型为 short
输出宽度 m(m 为整数)	指定输出项输出时所占的列宽,即域宽 (1) 若 m>0 时,当输出数据宽度小于 m 时,在域内向右对齐,输出数据左边多余位补空格;若 m 前有前导符 0,则输出数据左边多余位补 0;当输出数据宽度大于或等于 m 时,按实际宽度全部输出数据; (2) 若 m<0,则输出数据在域内向左对齐,输出数据右边多余位补空格

续表

格式修饰符	用　法
显示精度.n(n为大于或等于0的整数)	由一个圆点及其后的整数构成,若与输出宽度同时使用,应写在输出宽度的后面,即形式为 m.n (1) 对于浮点数,用于指定输出的小数位数 (2) 对于字符串,用于指定从字符串左侧开始截取的子串的字符个数

【例 3.3】 输入球的半径,求其体积与表面积。要求将圆周率定义为常量,输出结果保留 2 位小数。

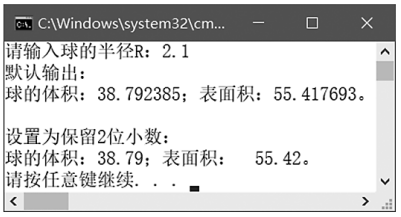
【问题分析】

根据球的体积公式 $V=\frac{4}{3}\pi R^3$ 与表面积公式 $S=4\pi R^2$ 即可求解。

【编程实现】

```
1    #include<stdio.h>
2    #define PI 3.1415926
3    int main()
4    {
5        double R,S,V;
6        printf("请输入球的半径 R: ");
7        scanf("%lf",&R);          //输入球的半径
8        V=4.0/3*PI*R*R*R;         //求球的体积
9        S=4*PI*R*R;               //求球的表面积
10       printf("默认输出: \n");
11       printf("球的体积: %f;表面积: %f。\n\n", V, S);
12       printf("设置为保留 2 位小数: \n");
13       printf("球的体积: %.2f;表面积: %7.2f。\n", V, S);
14       return 0;
15   }
```

【运行结果】



【关键知识点】

(1) 对于 double 型变量,输入时(程序第 7 行)必须使用格式符%lf,其详细解释见 3.3 节);而输出时可以用%f 或%lf,即第 11、13 行的%f 都可改为%lf。

(2) 第 8 行求体积的公式中,若写成 4/3,表示整除法,其计算结果为 1,无法求出正确的体积。所以应写成 4.0/3 或 4/3.0,表示普通除法。

(3) 从第 11 行代码对应的输出可以看到,浮点数默认输出 6 位小数。

(4) 第 13 行,输出球的体积时,将输出格式设置为%.2f,表示不设置输出宽度(按实际宽度输出),设置小数位数为 2 位,因此,输出值 38.79 紧挨着前面的中文冒号。输出表面积时,将输出格式设置为%7.2f,表示输出宽度为 7,小数位数为 2 位,而输出值 55.42 已经占了 5 个字符的宽度(包括小数点),故需要在该数前补充 2 个空格。在输出时,会进行四舍五入。

3.3 数据的格式化输入


数据的格式化输入课堂练习

在 C 语言的标准库函数中,专门用于格式化输入的函数为 scanf(),在头文件 stdio.h 中声明,主要功能是从标准输入设备(一般指键盘)按规定格式读取信息。

1. scanf()函数的一般格式

scanf()函数的调用格式为:

```
scanf(格式控制字符串,输入项参数地址表);
```

(1) **格式控制字符串**指定了输入的格式,它包含**格式控制符**和**分隔符**两部分。格式控制符用于指定各参数的输入格式,通常由%开始,并以一个格式字符结束,如表 3.3 所示。分隔符是指输入数据时两个数据之间的分隔符号,如空格、#号、逗号等。

表 3.3 scanf()函数的格式控制符

格式控制符	描　　述
%d	十进制整数的输入
%o	八进制整数的输入
%x	十六进制整数的输入
%c	一个字符的输入,空白字符也会作为一个有效字符输入
%f 或%e	float 型实数的输入
%%	输入一个百分号
%s	一个字符串的输入(字符串以空格、回车、制表符结束)

(2) **输入项参数地址表**是由**变量地址**组成的列表,参数之间用逗号隔开。通过**取地址运算符 &** 获取变量的地址,例如,&age 表示变量 age 的地址。scanf()函数要求必须指定接收数据的变量地址,否则数据不能读入到指定的内存单元。变量的地址列表必须与格式说明符依次对应。

2. scanf()函数的格式修饰符

与 printf()函数一样,在 scanf()函数的%和格式符中间也可以加入相关的**格式修饰符**,如表 3.4 所示。scanf()函数在输入数值型数据时,一般遇到**回车符**、**空格符**、**制表符(Tab)**、**非数值字符**时表示数据输入结束,而这些字符出现在数值前面时被忽略;在 scanf()函数中出现域宽修饰符时,当输入的数据达到输入域宽时,数据输入也会结束。

表 3.4　scanf()函数的格式修饰符

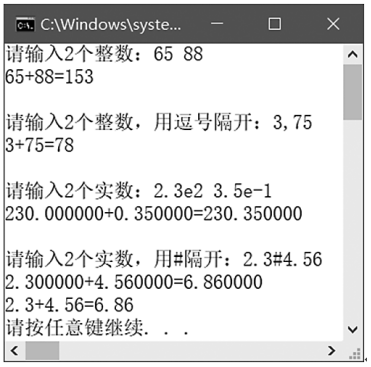
格式修饰符	描　　述
l	加在格式符 d、o、x、u 之前，用于输入 long 型数据 加在格式符 f、e 之前，用于输入 double 型数据
L	加在格式符 f、e 之前，用于输入 long double 型数据
h	加在格式符 d、o、x 之前，用于输入 short 型数据
域宽 m(正整数)	指定输入数据的宽度，系统自动按此宽度截取所需数据
*	表示对应的输入项在读入后不赋给任何变量

【例 3.4】　演示 scanf()函数的输入格式。

【编程实现】

```
1    #include<stdio.h>
2    int main()
3    {
4        int m, n;
5        float a, b;
6        double x, y;
7        printf("请输入 2 个整数: ");
8        scanf("%d%d",&m,&n);
9        printf("%d+%d=%d\n\n",m, n, m+n);
10       printf("请输入 2 个整数,用逗号隔开: ");
11       scanf("%d,%d",&m,&n);
12       printf("%d+%d=%d\n\n",m, n, m+n);
13       printf("请输入 2 个实数: ");
14       scanf("%f%f",&a,&b);
15       printf("%f+%f=%f\n\n",a, b, a+b);
16       printf("请输入 2 个实数,用#隔开: ");
17       scanf("%lf#%lf",&x,&y);
18       printf("%f+%f=%f\n",x, y, x+y);
19       printf("%g+%g=%g\n",x, y, x+y);
20       return 0;
21   }
```

【运行结果】



【关键知识点】

(1) 代码第 8、11、14、17 行的参数地址列表，必须在参数变量名前加上地址符 &，否

则数据不能输入。如语句 scanf("%d%d",m,n)会导致程序运行时异常终止。

(2) 第 8 行的 scanf()的两个格式说明符%d 之间没有任何分隔符,则输入的两个数据之间可以用**空格键**、**Tab 键**或**回车键**作为分隔符。通常采用空格作为分隔符输入数据,即 65 88↙ ,其中↙表示回车键。

(3) 若格式说明符之间用了指定的分隔符,输入数据时必须按照**指定的分隔符**(区分中英文状态)输入,否则数据输入失败。例如,第 11 行的两个%d 之间用英文逗号进行分隔,则数据的输入格式为:65,88↙,即两个整数之间必须用英文逗号隔开,若使用中文逗号、空格或其他符号隔开,则是错误的。

若将输入语句改为

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

则数据输入格式为:

```
m=65,n=88↙
```

(4) 对于 float 型变量,输入时可以使用格式说明符%f 或%e,如第 14 行所示;对于 double 型变量,输入时可以使用%lf 或%le(不能用%f 或%e),如第 17 行所示。程序运行时,从键盘输入的数据,既可以是普通小数(如 2.3),也可以是指数形式的(如 2.3e2)。

(5) 不管是 float 型还是 double 型变量,输出时既可以使用%f,也可以使用%lf,因此,第 15、18 行的%f 也可改为%lf。

(6) 对比第 18、19 行代码对应的输出,可以发现:用%f 输出时,默认输出 6 位小数,小数不够 6 位时补 0;用%g 输出时,不会输出无意义的 0。

【延展学习】

关于 scanf()函数的格式控制字符串,除了本例中的基本用法以外,还有以下用法。

(1) 通过格式修饰符**指定输入数据的宽度(即域宽)**,这样,在输入数据时,将自动按照域宽从输入的数据中截取所需数据。例如:

```
scanf("%2d%2d",&m,&n);
```

输入数据为:356289↙时,m 和 n 的宽度都为 2,故 m=35,n=62。

虽然在输入时可以指定域宽,但是不能指定小数的位数,例如:

```
float a; scanf("%5.2f",&a);
```

是错误的;将%5.2f 改为%5f 后,语句正确。

(2) 通过忽略输入修饰符 * **忽略输入数据中的部分内容**,例如,在输入两个数据时,若想以任意字符作为分隔符,则输入语句可写成:

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

其中,%*c 表示与其对应的输入项在读入后不赋给任何变量,也就意味着用户可以**用任意字符作为分隔符**来输入数据。当用户输入 65 88↙或 65, 88↙或 65&88↙时,m、n 均能正确获取数据 65、88。

如语句:

```
scanf("%2d%*3d%2d",&m,&n);
```

其中，%*3d 表示在从输入缓冲区读入数据时，有 3 个字符宽度的数据在读入后不赋给任何变量，即忽略这部分输入。若输入的数据为 123456789↙，m=12，n=67。

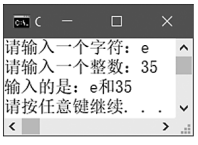
3. 用%c 输入字符时存在的问题及解决方法

请对比以下两段程序及其运行结果。

程序一：

```
#include<stdio.h>
int main()
{
    int a;
    char ch;
    printf("请输入一个字符: ");
    scanf("%c",&ch);
    printf("请输入一个整数: ");
    scanf("%d",&a);
    printf("输入的是: %c 和%d",ch,a);
    printf("\n");
    return 0;
}
```

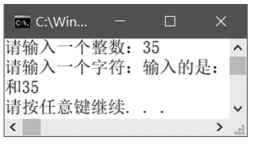
【运行结果】



程序二：

```
#include<stdio.h>
int main()
{
    int a;
    char ch;
    printf("请输入一个整数: ");
    scanf("%d",&a);
    printf("请输入一个字符: ");
    scanf("%c",&ch);
    printf("输入的是: %c 和%d",ch,
    a);
    printf("\n");
    return 0;
}
```

【运行结果】



程序一先输入字符，再输入整数，运行结果正常。程序二先输入整数，再输入字符，运行时出现了问题：程序直接跳过字符的输入。其原因是：输入 35↙后，35 被变量 a 接收，但回车符仍然留在输入缓冲区，接下来执行语句

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

时，直接从缓冲区提取回车符并存入变量 ch 中，最终导致了不正常的运行结果。

解决该问题的方法有如下 3 种。

(1) 在 scanf("%c",&ch)前增加一条语句 getchar()来接收上一次输入时存入缓冲区的回车符，这样 scanf("%c",&ch)便可正常接收用户输入的字符。

(2) 将输入语句改为 scanf("%*c%c",&ch)，此时%*c 将忽略上一次输入时存入缓冲区的回车符。

(3) **在格式说明符%c 前加一个空格**，即改为 scanf(" %c", &ch)，此处加入的空格作为分隔符。则程序运行时，用户在输入有效字符前可以按下任意多个空格键、Tab 键或回车键，这正好对应了上一次输入时存入缓冲区的回车符。

第 3 种方法更为常用。


顺序结构
程序设计
微视频

3.4 顺序结构程序设计

任何一个 C 语言源程序的整体结构都是顺序结构。顺序结构是最简单的程序结构，也是最常用的程序结构，只要按照解决问题的先后顺序写出相应的语句即可，其执行顺序是自上而下，依次执行。

顺序结构的程序主要包括数据的输入、处理、输出（Input、Process、Output），可用 IPO 图表示，如图 3.2 所示。作为初学者，应熟悉并掌握 IPO 图。

输入数据（从键盘输入初值或用“=”赋值）
处理数据
输出结果

图 3.2 顺序结构程序的 IPO 图

【例 3.5】 输入职工的职工号、性别、基本工资和奖励工资，计算其工资总和，输出其工资单（保留 2 位小数）。

【问题分析】

根据图 3.2 的 IPO 图，本例的 N-S 流程图如图 3.3 所示。

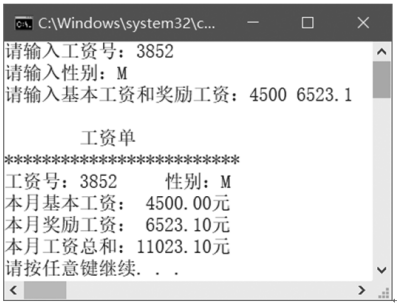
定义变量：职工号（int）、性别（char）、基本工资、奖励工资、工资总和（double）
从键盘输入职工号、性别、基本工资、奖励工资
计算工资总和
输出职工的基本信息和工资总和

图 3.3 例 3.5 算法的 N-S 流程图

【编程实现】

```
#include<stdio.h>
int main()
{
    int eNo;                          //职工号
    char sex;                         //性别,M 代表男,F 代表女
    double basic, bonus, total;       //依次为基本工资、奖励工资、工资总和
    printf("请输入职工号: ");
    scanf("%d",&eNo);
    printf("请输入性别: ");
    scanf("%c",&sex);                 //%c 前有一个空格
    printf("请输入基本工资和奖励工资: ");
    scanf("%lf%lf", &basic, &bonus);
    total=basic+bonus;
    printf("\n%14s\n","工资单");
    printf("**************************\n");
    printf("工资号: %-9d 性别: %c\n", eNo, sex);
    printf("本月基本工资: %8.2f 元\n", basic);
    printf("本月奖励工资: %8.2f 元\n", bonus);
    printf("本月工资总和: %8.2f 元\n", total);
    return 0;
}
```

【运行结果】



【关键知识点】

(1) 代码第 10 行的%c 前需要加一个空格，若不加，则运行该程序时，输入工资号后会跳过性别的输入。也可采用 3.3 节介绍的其他方法来解决这一问题。

(2) 第 14 行的%14s 表示输出一个字符串，输出宽度为 14 且右对齐。本例输出字符串“工资单”占 6 个字符的宽度，在其左侧补充了 8 个空格。

(3) 第 16 行的%－9d 表示输出一个整数，输出宽度为 9 且左对齐。因此，在输出 3852 时，在其右侧补充了 5 个空格。

(4) 第 17～19 行在输出数据时均进行了宽度设置，只要工资号不超过 9 位，且工资总和在 10 万元以内，那么工资单的四行数据就可以确保右侧对齐输出，以达到美观的效果。

3.5 常见错误小结

常见错误示例	错误描述及其解决方法	错误类型
printf("Welcome"); print("Welcome");	错误原因：误将 printf()函数写成 print()或 printf()，编译系统无法识别这些函数 解决方法：printf("Welcome");	链接错误
int a; scanf("a=%d, &m");	错误原因：误将参数地址表写入格式控制字符串的双引号中，运行时不能输入数据到变量中 解决方法：scanf("a=%d", &m);	运行错误
int a; scanf("a=%d", m);	错误原因：变量名前未加地址符 & 解决方法：scanf("a=%d", &m);	编译警告
printf(Welcome); scanf("a=%d", &m);	错误原因：缺少双引号，或者使用中文双引号 解决方法：printf("Welcome");scanf("a=%d", &m);	编译错误
int m, n; scanf("%2d%2d\n",&m,&n);	错误原因：在 scanf()的格式说明符中出现转义字符\n 解决方法：删掉\n	运行错误
double a; scanf("%f", &a);	错误原因：输入 double 数据时，格式说明符写成了%f 解决方法：将%f 改为%lf，即格式控制字符串中的格式说明符应与输入的数值类型一致	运行错误

续表

常见错误示例	错误描述及其解决方法	错误类型

int a=5;
printf("a=%d");
printf("a=", a);
	错误原因：第 1 次输出，printf()函数缺少了对应的输出项；第 2 次输出，printf()函数缺少了格式说明符
	解决方法：printf("a=%d", a);
		运行错误

int a, b; scanf("%d%d", &a, &b); 用户输入数据：23,45↙	错误原因：从键盘输入数据时使用的分隔符与 scanf()函数的格式控制字符串中指定的分隔符不一致 正确输入：23 45↙	运行错误
float m; scanf("%7.2f",&m);	错误原因：用 scanf()函数输入浮点数时不能设置精度 解决方法：scanf("%7f",&m);或者 scanf("%f",&m);	运行错误

3.6 练 习 题

一、单项选择题(注：□代表空格，↙代表回车)

1. 以下不能输出字符'B'的语句是(　　)。

A. putchar(66);　　B. putchar(B);

C. putchar('\x42');　　D. putchar('A'+1);

2. 以下不能输出字符'B'的语句是(　　)。

A. printf("%c\n",'a'-31);　　B. printf("%d\n",'A'+1);

C. printf("%c\n",66);　　D. printf("%c\n",'B');

3. 以下程序段的输出结果是(　　)。

```
int a=1, b=0;
printf("%d,",b=a+b);
printf("%d\n",a=2*b);
```

A. 1,2　　B. 1,0　　C. 3,2　　D. 0,0

4. 以下程序段的输出结果是(　　)。

```
char c1='A', c2='D';
printf("%d,%d", c1, c2-2);
```

A. 65,68　　B. A,68　　C. A,B　　D. 65,66

5. 以下程序段的输出结果是(　　)。

```
int x=32;
double y=3.141593;
printf("%d%8.6f",x,y);
```

A. 323.141593　　B. 32□3.141593

C. 32,3.141593　　D. 323.1415930

6. 以下程序段的输出结果是(　　)。

```
int x=32;
double y=3.141593;
printf("%d%9.5f",x,y);
```

A. 323.141593　　B. 323.14159

C. 32□3.14159　　D. 32□□3.14159

7. 有以下程序段：

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

若想通过键盘输入，使变量 a、b 的值分别为 3、5，则正确的输入是(　　)。

A. 3□5↙　　B. 3,5↙　　C. 3;5↙　　D. 35↙

8.有以下程序段：

```
int m,n,p;
scanf("m=%dn=%dp=%d",&m,&n,&p);
```

若想通过键盘输入，使变量 m、n、p 的值分别为 123、456、789，则正确的输入是(　　)。

A. m=123n=456p=789↙　　B. m=123 n=456 p=789↙

C. m=123,n=456,p=789↙　　D. 123□456□789↙

9. 有以下程序段：

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

若想通过键盘输入，使变量 a、b、c 分别为 3、4、5，则错误的输入是(　　)。

A. 3,4,5↙　　B. □□□3,4,5↙

C. 3,□□□4,□□□5↙　　D. 3□4□5↙

10. 有以下程序段：

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

若想通过键盘输入，使变量 a1、a2、c1、c2 的值分别为 22、44、'a'、'b'，输出结果是：

```
22,a,44,b
```

则正确的输入是(　　)。

A. 22□a44□b↙　　B. 22□a□44□b↙

C. 22,a,44,b↙　　D. 22a44b↙

11. 有以下程序段：

```
int a; float b;
scanf("%2d%f",&a,&b);
```

若从键盘输入：875□544.0↙，则变量 a 和 b 的值分别是(　　)。

A. 87 和 5.0　　B. 875 和 544.0　　C. 87 和 544.0　　D. 75 和 544.0

12.有以下程序段：

```
char a, b, c, d;
scanf("%c%c", &a, &b);
c=getchar();
d=getchar();
printf("%c%c%c%c\n", a, b, c, d);
```

若按下列方式输入数据(从第 1 列开始，↙代表回车，注意：回车也是一个字符)：

42↙

34↙

则输出结果是(　　)。

A. 42□34　　B. 42　　C. 4234　　D. 423

13. 有以下程序段：(说明：字符 0 的 ASCII 码值为 48)

```
char c1,c2;
scanf("%d",&c1);
c2=c1+9;
printf("%c,%c\n",c1,c2);
```

若从键盘输入：48↙

则输出结果是(　　)。

A. 48,57　　B. 4857　　C. 0,9　　D. 09

14. 以下输出语句中错误的是(　　)。

A. printf("%f\n",'s');　　B. printf("%d %c\n",'s','s');

C. printf("%c\n",'s'－32);　　D. printf("%c\n",65);

15. 若有定义

```
char ch; int a; double d;
```

若从键盘输入：12345□678910.36↙

以下能给各个变量正确赋值的输入语句是(　　)。

A. scanf("%d%c%lf",&a,&ch,&d);

B. scanf("%5d%2c%7.2lf",&a,&ch,&d);

C. scanf("%d%c%lf",a,ch,d);

D. scanf("5d%2c%7.2lf",&a,&ch,&d);

二、判断题

1. scanf()函数不能实现字符数据的输入。(　　)

2. 格式控制符%e 表示可以以指数形式输入一个浮点数。(　　)

3. 格式控制符%d%＊c%d 表示在输入两个整数时，这两个整数之间的间隔符可以是任意字符。(　　)

4. %7.2f 表示输出一个浮点数时，浮点数的宽度为 7(不包括小数点)，小数保留2 位。(　　)

5. 对于 double 型的实数，可以在 printf() 函数的格式化字符串中使用 n1.n2 的形式来指定输出宽度。n1 指定输出数据的宽度（包括小数点），n2 指定小数点后小数位数，也称为精度。（　　）

6. 格式控制符%－5f 表示输出的浮点数在域内靠右对齐。（　　）

7. 已定义了整型变量 d1 和 d2，则输入语句 scanf("%d,%d\n",&d1,&d2);可以实现数据的输入。（　　）

8. 输入浮点数时，可以在 scanf()函数中设置输入浮点数的精度。（　　）

9. 使用 getchar()函数时，只可以接收可显示字符。（　　）

10. 已定义了字符型变量 ch，语句 scanf(" %c",&ch);中，在%c 前面加了一个空格，可用于忽略上一次输入的回车键。（　　）

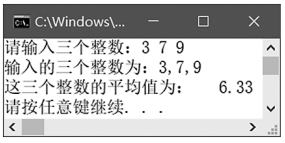
11. 在输入整数或实数等数值型数据时，输入数据之间必须用空格、回车符、制表符等间隔符隔开，间隔符个数不限。（　　）

12. 格式控制符%f 既可用于输出 float 型实数，也可用于输出 double 型实数。（　　）

三、编程题

1. **程序功能：**从键盘输入三个整数，计算其平均值并输出。要求：输出平均值时，数据宽度为 7，保留 2 位小数。输入/输出格式参见运行结果。

【运行结果】



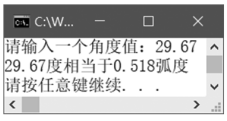
2. **编程实现：**有人从一山顶绝壁向下抛石头，经过 t 秒后听到石头落地的声音，请计算此山的高度 h(不考虑声音的传播用时)。

提示：自由落体公式 $h=\frac{1}{2}gt^2$。要求：将重力加速度 g 定义为符号常量，山的高度保留 1 位小数。输入/输出格式参见运行结果。

【运行结果】

3. **编程实现：**从键盘输入一角度值，计算并输出其对应的弧度值。要求：圆周率 π 使用符号常量，值取 3.14159；弧度值保留 3 位小数。输入/输出格式参见运行结果。

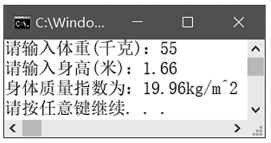
【运行结果】



4. **编程实现：**输入某人的身高、体重，求其 BMI 值。BMI 称为身体质量指数，是用体

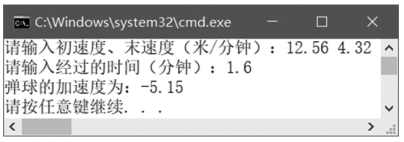
重(千克)除以身高(米)的平方得出的值,是目前国际上常用的衡量人体胖瘦程度以及是否健康的一个标准。输出结果保留 2 位小数。输入/输出格式参见运行结果。

【运行结果】



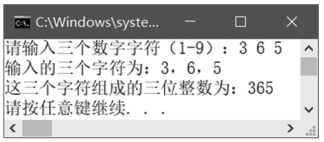
5. **编程实现**：一小孩在光滑的桌面弹弹球,请输入弹球的初速度、末速度和经过的时间,速度单位为“米/分钟”,时间单位为“分钟”,计算弹球的加速度(保留 2 位小数)并输出。输入/输出格式参见运行结果。提示：加速度=(末速度-初速度)÷经过的时间。

【运行结果】



6. **程序功能**：从键盘依次输入 3 个数字字符('0'除外),然后将这 3 个字符组成一个 3 位数输出。例如,输入的字符为'1'、'2'、'6',则组成的 3 位数为 126。输入/输出格式参见运行结果。

【运行结果】



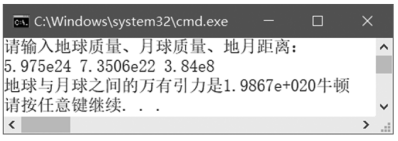
7. **程序功能**：计算地球与月球之间的万有引力。万有引力公式为：$F=\dfrac{Gm_1m_2}{r^2}$,其中：万有引力常量 $G=6.67\times10^{-11}\,N\cdot m^2/kg^2$,$m_1$ 和 m_2 的单位为千克(kg),r 的单位为米(m)。从键盘输入地球质量 m_1、月球质量 m_2、地月距离 r,输出地球与月球之间的万有引力。

提示：地球质量为 5.965×10^{24} kg,月球质量为 7.349×10^{22} kg,地月距离为 3.84×10^{8} m。

要求：在程序中用宏常量表示万有引力常量 G。输出结果保留 4 位小数。

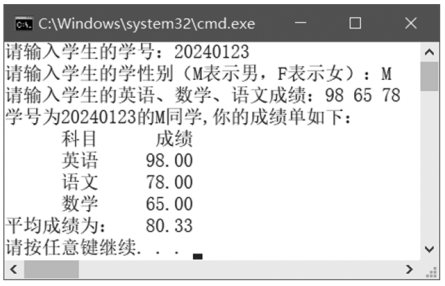
输入/输出格式参见运行结果。

【运行结果】



8. **程序功能**：从键盘输入某同学的学号、性别及三门课程的成绩,计算其平均成绩并输出。输出成绩保留 2 位小数,并注意输出格式美观,每项对齐,输入/输出格式参见运行结果。

【运行结果】



第 3 章练习题答案与解析

扫描二维码获取练习题答案与解析。

第 3 章　练习题答案与解析

第4章 选择结构

【学习要点】

在解决实际问题时，经常需要根据某个判断条件是否成立来选择执行不同的操作，此时需要用到选择结构。其中，简单的判断条件可以用关系表达式表示，复杂的判断条件通常用逻辑表达式表示。在C语言中，用于实现选择结构的语句有单分支语句if、双分支语句if-else和开关语句switch。本章的学习要点如下：

- 关系运算符与关系表达式。
- 逻辑运算符与逻辑表达式。
- 用if语句实现单分支选择结构。
- 用if-else语句实现双分支选择结构。
- 条件运算符与条件表达式。
- 用if-else嵌套语句实现多分支选择结构。
- 用switch语句实现多分支选择结构。

4.1 关系运算符与关系表达式

4.1.1 关系运算符

如果需要比较两个数据的大小，可以使用关系运算符。C语言提供的关系运算符如表4.1所示。

表4.1 关系运算符

运 算 符	含 义	优 先 级	结 合 方 向
<、<=	小于、小于或等于	7	自左至右
>、>=	大于、大于或等于		
==、!=	等于、不等于	8	

关系运算符的优先级高于赋值运算符、低于算术运算符。关系运算符的两个字符之间不能加空格，如>=不能写成 > =。


关系表达式
课堂练习

4.1.2 关系表达式

用关系运算符将两个表达式连接起来的式子称为**关系表达式**，语法格式如下：

```
<表达式>关系运算符 <表达式>
```

其中，表达式可以是常量、变量、算术表达式、关系表达式、赋值表达式等。

关系表达式通常用于表示一个判断条件，判断的结果只有两种可能性：**条件成立（真）**或**条件不成立（假）**。在 C 语言中，用整数 1 表示“真”，用整数 0 表示“假”，即关系表达式的值是整数 1 或 0。

例如，若判断条件为“n 不能被 3 整除”，可以用以下两个表达式来表示：

```
(1)n%3!=0          //若 n 除以 3 的余数不等于 0,则表达式的值为真
(2)n%3             //当 n%3 的结果等于 1 或 2 时,为真;当 n%3 的结果等于 0 时,为假
```

在 C 语言中，当数值型数据直接作为判断条件时，**非 0（含正数、负数）等价于“真”，0 等价于“假”**。

例如，已知

```
int A=5, B=9, C=2;
```

求下列各关系表达式的值。

```
(1)A<B             //比较结果为真,表达式的值为 1
(2)A==B<C          //等价于 A==(B<C),B<C 的值为假(0),A==0 为假(0),表达式的值为 0
(3)A<B<C           //等价于(A<B)<C,A<B 的值为真(1),1<C 为真(1),表达式的值为 1
```

在上例(3)中，B 的值并不在 A 和 C 之间，但表达式 A<B<C 的值为真，即数学中的不等式 A<B<C 在 C 程序中不能直接书写，那么这种情况应该如何表示呢？详见 4.2 节。

使用关系运算符时应注意：

(1)＝与＝＝的区别：＝为赋值运算符，其作用是将＝右侧表达式的值（即右值）赋给其左侧的变量（即左值）；＝＝为关系运算符，其作用是比较其左右两侧的值是否相等。＝＝的左右两侧可以是常量、变量或表达式。

(2) 编程时对浮点数一般不用＝＝进行判断，原因是十进制的浮点数转换成二进制存储时有精度损失，无法精确表示。一般通过判断两数之差的绝对值是否在可接受的范围内，从而确定两个浮点数是否相等。例如，比较两个 double 型变量 a 和 b 是否相等，可以通过判断关系表达式 fabs(a－b)<1e－8 是否成立来确定，若表达式成立，则表示 a 与 b 之差小于 10^{-8}，可以认为 a 和 b 相等。其中，fabs()函数的功能是求浮点数的绝对值，详见 6.3 节。

4.2 逻辑运算符与逻辑表达式

4.2.1 逻辑运算符

经过前面的分析可知，在 C 语言中，关系表达式 A<B<C 无法表示“B 大于 A 且 B 小