第3章 数据的输入与输出

输入输出指信息流入、流出计算机主机。信息(程序或数据)从计算机的外部设备(如键盘、磁盘、光盘和扫描仪等)流入计算机主机称为输入,从计算机主机流向计算机的外部设备(如显示器、磁盘和打印机等)称为输出。

C++提供了两种数据的输入输出方式。一种是保留了C的输入输出系统,即采用在标准函数库中提供输入输出函数的方法为程序提供输入输出功能;另一种是基于对象机制的输入输出系统,对象即是流。

3.1 printf 与 scanf

printf 和 scanf 是标准输入输出函数,它们是C程序中输入输出必不可少的,在头文件 stdio.h 中声明了这两个函数。在C++中,保留了C的输入输出系统。为了使用这两个库函数,应使用命令 #include <stdio.h>将对应的头文件 stdio.h 包含到程序中。

3.1.1 格式输出函数 printf

printf 函数的一般格式为:

```
printf(格式控制字符串, 输出项1, 输出项2, …, 输出项n)
```

括号中的格式控制字符串和输出项都是函数参数。printf 函数的功能是将后面的参数按给定的格式输出。

格式控制字符串中有格式说明也有普通字符。格式说明由“%”、对齐和填补说明、宽度和精度说明以及格式字符组成,中间不能插入别的字符,如%d、%f 等。它的作用是将输出的数据转换成指定的格式输出。普通字符就是原样输出的字符。输出项n是需要输出的一些数据,可以是表达式。例如:

```
#include <stdio.h>
void main()
{
    int a = 10,b = 20;
    printf ("%d,%d",a,b);
}
```

上例中,双引号中的字符除了两个%d外,还有“,”字符,它是普通字符。a、b 的值分别为10,20,输出为:

```
10,20
```

1. 格式字符

表 3-1 说明几个主要的数值数据格式符与输出项形式之间的关系。

表 3-1 主要数值数据格式符与输出项形式之间的关系

格式符	输出项形式	说 明
d,i	十进制整数	无宽度说明时,按实际宽度输出
X,x	十六进制整数	以 0x 打头的数为十六进制数。用 x,符号 a～f 以小写形式输出;用 X,符号 A～F 以大写形式输出
o	八进制整数	以 0 打头的数为八进制数
f	以小数方式输出	无宽度说明时,小数部分按 float 型规定取 6 位
E,e	科学记数法	e+002 表示 10^2
c	字符方式输出	
s	以字符串格式输出	

2. 宽度和精度说明

宽度说明用于说明输出数据所占的总位数。宽度说明默认时,按实际宽度输出。

有宽度说明时,对整数部分及字符串和字符按“认宽不认紧”的原则处理。

有宽度说明时,可在宽度字段后加圆点和精度说明,说明实型数的精度。精度说明默认时,小数部分取 6 位。实数的小数部分是宽紧都认,但从原有有效位的最后一位开始,便是近似的。

3. 对齐和填补说明

一般情况下,在宽度说明前使用“+”号时,数据在指定的位置空间中按右对齐方式输出;当使用“-”号时,按左对齐方式输出。

当使用右对齐方式时,若在宽度说明的前方加一个 0,则将数据前多余空位用 0 填补。

【例 3-1】 格式输出举例。

```
#include <stdio.h>

void main()
{
   char    ch = 'h';
   int     count = -3;
   double fp = 251.7366;

   /* Display characters. */
   printf("Character:\n\t%c\t%d%3c\n",ch,ch,ch);

   /* Display integers. */
   printf( "Integer formats:\n"
           "\tDecimal:%d  Justified:%6d  Unsigned:%u\n",
           count, count, count);

   printf( "Decimal %d as:\n\tHex:%Xh  hex:0x%x  Octal:%o\n",
```

```
            count, count, count, count );

    /* Display in different radixes. */
    printf( "Digits 10 equal:\n\tHex: %i  Octal: %i  Decimal: %i\n",
            0x10, 010, 10 );

    /* Display real numbers. */
    printf( "Real numbers:\n\t%f %10.2f  %e  %E\n", fp, fp, fp, fp );

     /* Display strings. */
    printf("String:\n\t%3s, %-5.3s, %5.2s\n","Hello","Hello","Hello");
}
```

程序运行结果：

```
Character:
    h    104  h
Integer formats:
     Decimal: -3  Justified:     -3  Unsigned: 4294967293
Decimal -3 as:
     Hex: FFFFFFFDh  hex: 0xfffffffd  Octal: 37777777775
Digits 10 equal:
     Hex: 16  Octal: 8  Decimal: 10
Real numbers:
     251.736600      251.74 2.517366e+002 2.517366E+002
String:
     Hello,Hel  ,   He
```

上例中，%10.2f 表示总长度为 10 位，小数位数为 2 位。%－5.3s 中的负号表示左对齐，如果没有负号，则默认右对齐。5 表示格式宽度，3 表示截取字符串中 3 个字符。

如果要输出%本身，则双写%。例如：

```
printf ("%f%%",1.0/3);
```

输出结果为：

```
0.333333%
```

3.1.2 格式输入函数 scanf

scanf 函数的一般格式为：

```
scanf（格式控制字符串，地址 1，地址 2，…，地址 n）
```

格式控制字符串的含义同前，地址 n 是变量的地址。

- %d：用以输入整数，带 l 表示长整数，带 h 表示短整数。
- %c：用以输入字符。
- %o、%x：用以输入八进制数和十六进制数。%lo 和%lx 分别表示长八进制数和长十六进制数。
- %f：用以输入浮点数，%lf 和%Lf 分别表示输入 double 型数和 long double 型数。

- %e：与%f作用相同。
- %s：用以输入字符串，以非空字符开始，以空字符或回车结束。

【例 3-2】 格式输入举例。

```
#include <stdio.h>
void main()
{
    int a,b;
    char ch1,ch2;
    float f,g;
    scanf ("%d %d",&a,&b);
    scanf (" %c%c",&ch1,&ch2);
    scanf (" %f,%f",&f,&g);
    printf ("a = %d b = %d ch1 = %c ch2 = %c f = %f,g = %f",a,b,ch1,ch2,f,g);
}
```

程序运行结果：

```
23 456 ↵
ab ↵
23.6712,612.97 ↵

a = 23 b = 456 ch1 = a ch2 = b f = 23.671 200,g = 612.969 971
```

两个输入项之间一般用空格分开。如果规定了分隔字符，如“,”，则必须以逗号分开。如上例中，第1个scanf语句数值之间必须以空格或回车隔开，第3个scanf语句必须以逗号隔开。

在用%c格式输入时，空格字符和转义字符都作为有效字符输入。如上例中，执行第2个scanf语句时，若输入a b，则'a'赋给ch1，' '(空格)赋给ch2，而'b'被忽略。

scanf()中后面的地址参数可以是变量的地址，但不能是变量名，否则会将输入值存放在以变量值作为地址的内存空间中，导致意想不到的运行异常。

3.2 I/O流控制

I/O流是输入或输出的一系列字节，当程序需要在屏幕上显示输出时，可以使用插入操作符“<<”向cout输出流中插入字符。当程序需要执行键盘输入时，可以使用抽取操作符“>>”从cin输入流中抽取字符。

cin和cout是C++语言的标准输入流类istream和标准输出流类ostream的对象。要使用C++提供的输入输出流时，必须在程序的开头增加一行#include <iostream.h>，即包含输入输出流的头文件iostream.h。

3.2.1 cin

在程序执行期间，要给变量输入数据时，可以用cin来完成。

cin输入的一般格式为：

```
cin>>变量名1>>变量名2>>…>>变量名n;
```

其中,“>>”称为提取运算符,表示将程序暂停执行,等待用户从键盘上输入相应的数据。每个提取运算符后面只能跟一个变量名,但提取运算符可以多次使用,即用一个 cin 可以为一个变量提供输入值,也可以为多个变量提供输入值。

例如:

```
int x;
char y;
cin>>x>>y;  //输入时只需按下面形式输入即可:12 w↵
```

不管把什么基本数据类型的名字或值传给流,它都能懂。

在使用“>>”进行输入操作时,以空白作为一个数据的结束。即在一个语句中输入多个数据时,输入的几个数据间应以空格分隔。在输入的字符串中不能含有空格,一遇空格,便认为是本数据结束。

一条 cin>>语句的键盘操作以回车结束。当在一行中输入的数据比要提取的变量数多时,多余的数据将保留在流中,供下一次提取使用;流中的数据不足时,要等待用户的输入操作。例如对上述程序段,输入:

```
12 w↵
```

与输入:

```
12↵
w↵
```

的效果一样。

在检测空格的同时,系统还检查输入数据与变量的匹配。例如,对于程序段:

```
int x;
char y;
cin>>x>>y;
cout<<"x="<<x<<",y="<<y;
```

当输入:

```
12 34↵
```

时,输出为:

```
x=12,y=3
```

因为系统同时以类型分隔输入数据,将 12 赋给整型变量 x,将 3 赋给字符型变量 y。

在提取过程中,若遇到输入数据与变量类型不符,将返回零值,终止提取操作。

3.2.2 cout

在 C++中,与 cin 输入流对应的是 cout 输出流。cout 输出流的格式如下:

```
cout<<表达式1<<表达式2<<…<<表达式n;
```

其中,“<<”称为插入运算符,它把紧跟在它后面的表达式的值输出到显示器的当前光

标位置。和 cin 类似，用一个 cout 可以输出一个表达式的值，也可以输出多个表达式的值。

例如：

```
cout<<"20 + 30 = "<<20 + 30<<"\n" ;
```

也可以写成：

```
cout<<"20 + 30"    //行末无分号
   <<20 + 30
   <<"\n" ;
```

3.2.3 格式控制

流的默认格式输出有时不能满足特殊要求。用控制符可以对 I/O 流的格式进行控制。

1. 无参格式控制符

表 3-2 所示的几个无参格式控制符定义在头文件 iostream.h 中。

表 3-2 无参格式控制符

控制符	描　　述	输入输出
dec	数值数据采用十进制表示	I/O
hex	数值数据采用十六进制表示	I/O
oct	数值数据采用八进制表示	I/O
ws	提取空白字符	I
endl	产生一个换行('\n')	O
ends	产生一个空字符(NULL)，通常用来结束一个字符串	O
flush	强制将流从缓冲区写到相应设备，刷新流相关联的缓冲区	O

2. 带参格式控制符

表 3-3 所示的几个带参数的格式控制符定义在头文件 iomanip.h 中，使用时必须在程序开头添加一行 #include <iomanip.h>。

表 3-3 带参格式控制符

控制符	描　　述	输入输出
setfill(char ch)	设填充字符为 ch	O
setw(int w)	设域宽为 w 个字符	O
setprecision(int n)	设置浮点数输出的有效数字个数 n	O
setiosflags(ios::fixed)	固定的浮点显示	O
setiosflags(ios::scientific)	指数表示	I/O
setiosflags(ios::left)	左对齐	O
setiosflags(ios::right)	右对齐	O
setiosflags(ios::skipws)	忽略前导空白	O
setiosflags(ios::uppercase)	十六进制数大写输出	O
setiosflags(ios::lowercase)	十六进制数小写输出	O

3. 控制浮点数值显示

C++默认的流输出数值有效位是 6。如果 setprecision(n)与 setiosflags(ios::fixed)合

用,可以控制小数点右边的数字个数。setprecision(n)可控制输出流显示浮点数的数字个数,setiosflags(ios::fixed)是用定点方式表示实数。

【例 3-3】 分别用浮点、定点和指数方式表示一个实数。

```
#include <iostream.h>
#include <iomanip.h>

void main()
{
    double amount = 22.0/7;
    cout<<amount<<endl;
    cout<<setprecision(0)<<amount<<endl
        <<setprecision(1)<<amount<<endl
        <<setprecision(2)<<amount<<endl
        <<setprecision(3)<<amount<<endl
        <<setprecision(4)<<amount<<endl;

    cout<<setiosflags(ios::fixed);
    cout<<setprecision(8)<<amount<<endl;

    cout<<setiosflags(ios::scientific)<<amount<<endl;
    cout<<setprecision(6);    //重新设置成原默认设置
}
```

程序运行结果:

```
3.142 86
3
3
3.1
3.14
3.143
3.142 857 14
3.142 857 1
```

在用浮点表示的输出中,setprecision(n)表示有效位数。

第 1 行输出数值之前没有设置有效位数,所以用流的有效位数默认设置值 6。第 2 个输出设置了有效位数 0,C++最小的有效位数为 1,所以作为有效位数设置为 1 来看待。第 3～6 行输出按设置的有效位数输出。

在用定点表示的输出中,setprecision(n)表示小数位数。

第 7 行输出是与 setiosflags(ios::fixed)合用。所以 setprecision(8)设置的是小数点后面的位数,而非全部数字个数。

在用指数形式输出时,setprecision(n)表示有效位数。

第 8 行输出用 setiosflags(ios::scientific)来表示指数表示的输出形式,其有效位数沿用上次的设置值 8。

小数位数截短显示时,进行四舍五入处理。

4. 设置值的输出宽度

除了使用空格来强行控制输出间隔外,还可以用 setw(n)控制符。如果一个值需要比

setw(n)确定的字符数更多的字符,则该值将使用它所需要的所有字符。例如:

```
float amount = 3.141 59;
cout<<setw(4)<<amount<<endl;
```

其运行结果为:3.141 59。它并不按 4 位宽度,而是按实际宽度输出。

如果一个值的字符数比 setw(n)确定的字符个数更少,则在数字字符前显示空白。不同于其他控制符,setw(n)仅仅影响下一个数值输出。换句话说,使用 setw 设置的间隔方式并不保留其效力。例如:

```
cout<<setw(8)
   <<10
   <<20 << endl ;
```

程序运行结果:

```
_ _ _ _ _ _1020
```

运行结果中的下横线表示空格。整数 20 并没有按宽度 8 输出。setw()的默认值为宽度 0,即 setw(0),意思是按输出数值的表示宽度输出,所以 20 就紧挨 10 了。若要每个数值都有宽度 8,则每个值都要设置:

```
cout << setw(8) << 10
    << setw(8) << 20 << endl ;
```

5. 输出八进制和十六进制数

三个常用的控制符是 hex、oct 和 dec,它们分别对应十六进制、八进制和十进制数的显示。

【例 3-4】 分别用十进制、十六进制和八进制表示一个整数。

```
#include <iostream.h>

void main()
{
    int number = 1001;
    cout<<"Decimal:"<<dec<<number<<endl
        <<"Hexadecimal:"<<hex<<number<<endl
        <<"Octal:"<<oct<<number<<endl;
}
```

程序运行结果:

```
Decimal:1001
Hexadecimal:3e9
Octal:1751
```

1001 是一个十进制数,不能把它理解成十六进制或八进制数,因为它不是以 0x 或 0 开头。但在输出时,流根据控制符进行过滤,使其按一定的进制来显示。

6. 设置填充字符

setw 可以用来确定显示的宽度。默认时,流使用空格符来保证字符间的正确间隔。用 setfill 控制符可以确定一个非空格的字符来确定间隔。

例如,下面的程序:

```
#include <iostream.h>
#include <iomanip.h>

void main()
{
    cout<<setfill('*')
        <<setw(2)<<21<<endl
        <<setw(3)<<21<<endl
        <<setw(4)<<21<<endl;
    cout<<setfill(' '); //恢复默认设置
}
```

程序运行结果:

```
21
*21
**21
```

7. 左右对齐输出

默认时,I/O 流左对齐显示的内容。使用 setiosflags(ios::left)和 setiosflags(ios::right)可以控制输出对齐。例如:

```
#include <iostream.h>
#include <iomanip.h>

void main()
{
    cout<<setiosflags(ios::right)
        <<setw(5)<<1
        <<setw(5)<<2
        <<setw(5)<<3<<endl;
    cout<<setiosflags(ios::left)
        <<setw(5)<<1
        <<setw(5)<<2
        <<setw(5)<<3<<endl;
}
```

程序运行结果:

```
____1____2____3
1____2____3____
```

8. 强制显示小数点和符号

当程序输出下面的代码时:

```
cout << 10.0/5 << endl ;
```

默认的 I/O 流会简单地显示 2,而非 2.0,因为除法的结果是精确的。当需要显示小数点时,可以用 ios::showpoint 标志。例如:

```
#include <iostream.h>
#include <iomanip.h>

void main()
{
    cout << 10.0/5 << endl;

    cout << setiosflags(ios::showpoint)
         << 10.0/5 << endl;
}
```

程序运行结果：

```
2
2.000 00
```

默认时，I/O 流仅在负数之前显示值的符号。根据程序的用途，有时也需要在正数之前加上正号，可以用 ios::showpos 标志。例如：

```
#include <iostream.h>
#include <iomanip.h>

void main()
{
    cout << 10 <<" "<< -20 << endl;

    cout << setiosflags(ios::showpos)
         << 10 <<" "<< -20 << endl;
}
```

程序运行结果：

```
10 -20
+10 -20
```

习题 3

1. 读程序，写出运行结果

(1)
```
#include <stdio.h>
void main()
{
  float a = 123456789e5,b;
  b = a + 20;
  printf(" % f\n",b);
}
```

(2)
```
#include <stdio.h>
void main()
{
  int a,b;
  long c,d;
```